КИТОВ и Н. А. КРИНИЦКИЙ

# ЭЛЕКТРОННЫЕ ЦИФРОВЫЕ МАШИНЫ
## И ПРОГРАММИРОВАНИЕ

*Допущено*
*Министерством высшего образования СССР*
*в качестве учебного пособия*
*для высших учебных заведений*

ГОСУДАРСТВЕННОЕ ИЗДАТЕЛЬСТВО ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ
М О С К В А 1959

# ОГЛАВЛЕНИЕ

**ПРЕДИСЛОВИЕ** ... **8**

**ВВЕДЕНИЕ** ... **9**

**§ 1. Основные этапы развития вычислительной техники** ... **10**
1. Вычислительные устройства непрерывного действия. ... 10
2. Цифровые вычислительные машины. ... 11

**§ 2. Общая структурная схема электронной цифровой машины и принцип программного управления** ... **15**
1. Структурная схема. ... 15
2. Принцип программного управления. ... 16

**§ 3. Применение электронных цифровых машин** ... **17**
1. Применение для научных и технических исследований и разработок. ... 17
2. Применение для обработки информации. ... 18
3. Применение для автоматического управления производственными процессами. ... 19

ГЛАВА I
**АРИФМЕТИЧЕСКИЕ ОСНОВЫ ЭЛЕКТРОННЫХ ЦИФРОВЫХ МАШИН**

**§ 4. Позиционные системы счисления** ... **21**
1. Римская система счисления. ... 21
2. Понятие позиционной системы счисления. ... 21
3. Восьмеричная позиционная система счисления. ... 23
4. Двоичная позиционная система счисления. ... 23
5. Другие позиционные системы счисления. ... 24

**§ 5. Перевод чисел из одной позиционной системы счисления в другую** ... **25**
1. Перевод целых чисел. ... 25
2. Перевод дробей. ... 26
3. Перевод из восьмеричной системы счисления в двоичную и обратно. ... 27
4. Двоично-десятичная запись чисел. ... 28

**§ 6. Ввод чисел в машину и запись их в памяти** ... **28**
1. Запись чисел на перфокартах. ... 28
2. Запись чисел в ячейках памяти машины с фиксированной запятой. ... 29
3.Запись чисел в ячейках памяти машины с плавающей запятой. ... 31
4. Диапазон чисел, представимых в ячейках памяти машины. ... 34
5. Прямой код. ... 35

**§ 7. Двоичные сумматоры и операции над положительными числами, выполняемые с их помощью** ... **35**
1. Одноразрядный двоичный сумматор. ... 35
2. Многоразрядный двоичный сумматор. ... 36
3. Операции над положительными числами, выполняемые на сумматоре без переноса из старшего разряда. ... 37
4. Операции над положительными числами, выполняемые на сумматоре с циклическим переносом. ... 37

**§ 8. Алгебраическое сложение в дополнительном коде** ... **37**
1. Дополнительный код. ... 37
2. Модифицированный дополнительный код. ... 42

**§ 9. Алгебраическое сложение в обратном коде** ... **44**
1. Обратный код. ... 44
2. Модифицированный обратный код. ... 49

**§ 10. Сложение и вычитание нормализованных чисел. Умножение и деление в машинах с фиксированной и с плавающей запятой** ... **52**
1. Сложение и вычитание чисел в машинах с плавающей запятой. ... 52
2. Умножение и деление чисел в машинах. ... 54

ГЛАВА II
**МАТЕМАТИЧЕСКАЯ ЛОГИКА И ПОСТРОЕНИЕ СХЕМ ЭЛЕКТРОННЫХ ЦИФРОВЫХ МАШИН**

**§ 11. Начальные сведения из алгебры логики** ... **55**

1. Понятие высказывания и его значения истинности. 55
2. Сложные высказывания. Логические связи. Логические операции. 56
3. Геометрическое толкование алгебро-логических операций. 59
4. Связи между логическими операциями. 60

**§ 12. Преобразование логических выражений 61**
1. Нормальные формы логических выражений. 61
2. Постоянно-истинные и постоянно-ложные выражения. 61
3. Многообразие сложных логических выражений. 62
4. Преобразование логических выражений. 63

**§ 13. Переключательные (нелинейные) элементы электронных схем 64**
1. Электронные лампы. 65
2. Полупроводниковые диоды и триоды. 66
3. Магнитные сердечники. 69

**§ 14. Электронные схемы для основных логических операций 70**
1. Логические схемы на электронных лампах и полупроводниковых диодах. 70
2. Логические схемы на полупроводниковых триодах. 72
3. Логические схемы на магнитных сердечниках. 74

**§ 15. Комбинированные электронные логические схемы 75**
1. Двойной вентиль. 75
2. Одноразрядный преобразователь 75
3. Избирательная схема. 76
4. Схема сдвигателя. 76

**§ 16. Синтез избирательных логических схем 77**
1. Первый пример. 78
2. Второй пример. 79
3. Третий пример. 79

**§ 17. Синтез одноразрядных двоичных сумматоров 80**
1. Одноразрядный двоичный сумматор на два входа. 80
2. Одноразрядный двоичный сумматор на три входа. 81

ГЛАВА III

**ТЕХНИЧЕСКИЕ ПРИНЦИПЫ УСТРОЙСТВА ЭЛЕКТРОННЫХ ЦИФРОВЫХ МАШИН**

**§ 18. Принципы построения и типы машин 83**
1. Машины параллельного и последовательного действия. Фиксированная и плавающая запятая. 83
2. Адресность машин. 84

**§ 19. Устройство управления машины 85**
1. Назначение и состав устройства управления машины. 85
2. Порядок работы устройства управления. 85
3. Выполнение условных переходов. 87

**§ 20. Арифметические устройства 88**
1. Сумматоры параллельного действия. 88
2. Сумматоры последовательного действия. 90
3. Устройство для сложения и вычитания чисел с плавающей запятой. 91
4. Множительные устройства. 92

**§ 21. Запоминающие устройства 93**
1. Перфоленты и перфокарты. 93
2. Электромеханические и электронные реле. 94
3. Линии задержки. 94
4. Магнитные барабаны и ленты. 96
5. Электронно-лучевые запоминающие системы. 97
6. Запоминающее устройство на магнитных сердечниках. 99
7. Ферроэлектрическиезапоминающие устройства. 101

**§ 22. Основные образцы советских электронных цифровых вычислительных машин 101**
1. Машина *БЭСМ*. 101
2. Машина *Стрела*. 102

3. Машина *Урал*. 105
4. Машина *М-3*. 107

ГЛАВА IV

## НАПРАВЛЕНИЯ РАЗВИТИЯ ЭЛЕКТРОННЫХ ЦИФРОВЫХ МАШИН

**§ 23. Общие тенденции развития 107**
1. Машины для математических вычислений. 107
2. Машины для обработки информации. 110
3. Машины для автоматического управления. 111

**§ 24. Разработка новых принципов построения и усовершенствование конструкции электронных цифровых машин 113**
1. Повышение быстродействия машин. 113
2. Применение индексных регистров. 114
3. Применение принципа микропрограммного управления и повышение гибкости структуры машин. 114
4. Усовершенствование конструкции и технологии производства машин. 116
5. Разработка быстродействующих устройств ввода и вывода. 117

**§ 25. Разработка новых элементов 117**
1. Ферромагнитные элементы. 118
2. Полупроводниковые приборы. 119
3. Разработка новых переключательных и запоминающих элементов. 120

ГЛАВА V

## ИСХОДНЫЕ ДАННЫЕ ПРОГРАММИРОВАНИЯ ДЛЯ МАШИН *Стрела, М-3* и *Урал*

**§ 26. Порядок выполнения команд 121**
1. Машины с естественным порядком выполнения команд. 121
2. Машины с принудительным порядком выполнения команд 121
3. «Чтение» машиной содержимого ячейки. 122

**§ 27. Общая характеристика машины *Стрела* 123**
1. Запись чисел в ячейках памяти. 123
2. Структура команд и их запись. 124
3. Ввод программы в память машины. 125
4. Автоматическое управление работой машины. 126

**§ 28. Система операций и команд машины *Стрела* 127**
1. Таблица команд и операций. 127
2. Пояснение некоторых команд. 135
3. Константы, хранящиеся в УВК. 140

**§ 29. Сведения о машине *М-3*, необходимые для программирования 142**
1. Запись чисел в ячейках памяти. 142
2. Структура команд и их запись. 142
3. Ввод программы в память машины и вывод результатов. 143
4. Система операций и команд машины. 144

**§ 30. Сведения о машине *Урал*, необходимые для программирования 145**
1. Запись чисел в ячейках памяти. 145
2. Структура команд. 146
3. Основные особенности машины. 147
4. Система операций и команд машины *Урал*. 148

ГЛАВА VI

## ОСНОВЫ ПРОГРАММИРОВАНИЯ

**§ 31. Непосредственное программирование 151**
1. Общие указания. 151
2. Пример составления программы. 151
3. Разветвляющиеся программы. 154

**§ 32. Операторное программирование 158**
1. Логические схемы программ. 158
2. Свойства операторов 160
3. Правила начертания логических схем программ. 161
4. Примеры операторного программирования. 162

**§ 33. Циклические программы** ... **166**
1. Итерационный цикл. ... 166
2. Цикл с переадресацией. ... 167
3. Цикл с переадресацией и восстановлением. ... 170

**§ 34. Способы управления повторениями цикла** ... **173**
1. Оператор управления повторениями цикла. ... 173
2. Счетчик повторений цикла. ... 175
3. Цикл, повторяющийся пока монотонно изменяющаяся величина не перейдет через заданное значение. ... 177
4. Логические шкалы. ... 178

**§ 35. Некоторые общие приемы операторного программирования** ... **179**
1. Стандартные ячейки. ... 179
2. Операторы-подпрограммы. ... 181
3. Циркуляция величин в стандартных ячейках. ... 184
4. Операторы формирования. ... 185

**§ 36. Программирование для машин с фиксированной запятой** ... **186**
1. Программирование для одноадресной машины. ... 186
2. Программирование для двухадресной машины. ... 191

ГЛАВА VII

## МЕТОДЫ РУЧНОГО ПРОГРАММИРОВАНИЯ

**§ 37. Порядок работы при ручном программировании** ... **192**
1. Параметрически заданная схема счета. ... 192
2. Составление логической схемы. ... 194
3. Распределение памяти и составление программы. ... 196

**§ 38. Метод библиотечных подпрограмм** ... **197**
1. Библиотечные подпрограммы. ... 197
2. Пример открытой библиотечной подпрограммы. ... 197
3. Автоматизация метода библиотечных подпрограмм. ... 199
4. Объединяющая программа. ... 201

**§ 39. Автоматизация отдельных работ при ручном программировании** ... **201**
1. Автоматизация присвоения действительных адресов. ... 201
2. Автоматизация исправления некоторых ошибок, обнаруженных в программе. ... 203
3. Программа *ВУЗП*. ... 203

ГЛАВА VIII

## ОСОБЕННОСТИ РЕШЕНИЯ ЗАДАЧ НА ЭЛЕКТРОННЫХ ЦИФРОВЫХ МАШИНАХ

**§ 40. Методы контроля** ... **205**
1. Проверка программы и контроль правильности ее ввода. ... 205
2. Отладка программы. ... 207
3. Контроль правильности работы машины. ... 207
4. Контроль правильности вычислений. ... 209

**§ 41. Организация программы** ... **210**
1. Основные понятия и обозначения. ... 210
2. Примеры схем организации программ. ... 211

**§ 42. Выбор численного метода** ... **212**
1. Погрешность вычислений. ... 212
2. Связность алгорифма. ... 213
3. Учет стоимости и затрат времени. ... 213

**§ 43. Метод статистических испытаний (метод Монте-Карло)** ... **214**
1. Датчики случайных чисел. ... 214
2. Неравенство Чебышева. ... 214
3. Вычисление интеграла с помощью специально подобранного датчика. ... 215
4. Вычисление простых и кратных интегралов с помощью стандартного датчика. ... 215
5. Вычисление значений функции по ее обратной функции и решение уравнений. ... 217
6. Преобразование потока случайных чисел. ... 218
7. Теоретико-вероятностное моделирование. ... 219
8. Погрешности при решении задач. ... 219

9. Преимущества метода. ... 221

**§ 44. Способы задания и вычисления значений функций ... 221**
1. Выбор способа задания функции. ... 221
2. Аналитическое задание функции. ... 222
3. Задание функции с помощью дифференциального уравнения. ... 222
4. Способ аппроксимирующих многочленов. ... 222
5. Табличное задание функции. ... 223

**§ 45. Способы выбора значений функции из малых таблиц ... 223**
1. Регулярные таблицы. ... 223
2. Нерегулярные таблицы. Способ перебора. ... 224
3. Почти-регулярные таблицы. ... 224
4. Способ двухстепенного перебора. ... 226
5. Способ деления «пополам». ... 228
6. Способ скользящего начала (конца) таблицы. ... 228
7. Способы «плотного» размещения таблиц в памяти. ... 228
8. Примеры подпрограмм для выбора значений функции из таблицы. ... 230

ГЛАВА IX
**ФОРМАЛЬНЫЕ ПРЕОБРАЗОВАНИЯ ЛОГИЧЕСКИХ СХЕМ ПРОГРАММ**

**§ 46. Основные понятия ... 234**
1. Обозначения и простейшие формулы. ... 234
2. Основные логические переменные. ... 236
3. Распределение сдвигов схемы. ... 238
4. Выполнение схемы по заданной последовательности наборов значений основных логических переменных. ... 238
5. Значение схемы. ... 239
6. Равносильность схем. ... 240

**§ 47. Преобразование логических связей в схемах ... 241**
1. Подчиненность оператора логической функции. ... 241
2. Равносильность логических функций. ... 243
3. Исключение знаков логических связей ... 243

**§ 48. Преобразование логических схем ... 245**
1. Преобразование стрелок. ... 245
2. Перестановка операторов. ... 246
3. Исключение операторов. ... 246
4. Логические операторы, проверяющие значения тождественно постоянных логических функций. ... 246
5. Примеры преобразования схем. ... 248

ГЛАВА X
**ПРОГРАММИРУЮЩИЕ ПРОГРАММЫ**

**§ 49. Операторное автоматическое программирование и *ПП-С* ... 251**
1. Сущность операторного автоматического программирования. ... 251
2. Способ кодировки информации для *ПП-С* ... 252
3. Вид команд, составляемых *ПП-С* ... 254
4. Нестандартное программирование. ... 255
5. Организация программирующей программы *ПП-С* ... 257

**§ 50. Подготовка к кодировке информации для *ПП-С* ... 258**
1. Составление задания для кодировщиков. ... 258
2. Пример подготовки информации к кодировке. ... 262

**§ 51. Кодировка информации для *ПП-С* ... 266**
1. Первый массив информации ... 267
2. Второй массив информации. ... 269
3. Третий массив информации. ... 269
4. Подготовка данных для ввода информации в память машины. ... 270
5. Пример кодировки информации для *ПП-С*. ... 270

**§ 52. Эксплуатаия программирующей программы *ПП-С* ... 272**
1. Запись *ПП-С* на магнитную ленту ... 272
2. Ввод информации. ... 272
3. Пуск *ПП-С*. ... 272

4. Результаты, выдаваемые *ПП-С*. 272
5. Работа с *ПП-С* в случае нестандартного распределения памяти. 273

**§ 53. Краткое описание работы блоков *ПП-С*** 273
1. Блок *К*. 273
2. Блок *А′*. 274
3. Блок *Р′*. 275
4. Блок *А*. 276
5. Блок *Р*. 277
6. Блок *С*. 280
7. Блок *F*. 280
8. Блок *Э*. 282
9. Блок *О*. 282
10. Блок *П*. 283

ГЛАВА XI

## НЕАРИФМЕТИЧЕСКИЕ ВОЗМОЖНОСТИ ЭЛЕКТРОННЫХ ЦИФРОВЫХ МАШИН

**§ 54. Машинный перевод** 284
1. Опыт перевода с русского языка на английский на машине *ИБМ*-701 285
2. Опыт перевода с английского языка на русский на машине *БЭСМ*. 287
3. Опыт перевода с французского языка на русский с помощью машины *Стрела*. 288

**§ 55. Машинная игра** 291
1. Машины типа «словарь». 291
2. Машины, реализующие строго определенные правила игры. 291
3. Машины, использующие общие принципы оценки положений. 291
4. Машины, накапливающие «опыт» 291
5. Машины, играющие в шахматы. 292
6. Имитация условного рефлекса. 293

**ЛИТЕРАТУРА** 295

## ПРЕДИСЛОВИЕ

Настоящая книга была задумана как второе издание книги А. И. Китова, Н. А. Криницкого, П. Н. Комолова «Элементы программирования», выпущенной в 1956 г. издательством Артиллерийской инженерной академии им. Дзержинского. Однако значительный прогресс в области электронной вычислительной техники и теории программирования и стремление авторов создать книгу, по возможности соответствующую уровню развития этих областей, привели фактически к тому, что была написана новая книга. В нее вошли новые разделы, посвященные принципам построения электронных цифровых машин, вопросам преобразования логических схем программ, вопросам автоматизации программирования. Значительно расширены и другие разделы книги, в частности, кроме описаний одноадресной машины *Урал* и трехадресной машины *Стрела,* в книгу включено описание двухадресной машины *М-3* и краткие сведения о машине *БЭСМ.*

Основные идеи теории и методики программирования излагаются с таким расчетом, чтобы ими можно было воспользоваться при программировании для различных цифровых машин, хотя почти все примеры приведены применительно к конкретной машине *Стрела.*

Сведения об устройстве машин изложены в минимальном объеме, необходимом для уяснения принципов функционирования машин, без чего невозможен сознательный подход к вопросам программирования и, особенно, исследовательская работа в этой области.

Для правильного понимания возможностей электронных цифровых машин и областей их применения важно также иметь представление о современном уровне и перспективах развития этой техники, для чего введены соответствующие разделы.

В книге частично использованы материалы из книги А. И. Китова «Электронные цифровые машины», изданной в 1956 г. издательством

«Советское радио», а также современные отечественные и иностранные материалы. Введение и главы II—IV и XI написаны А. И. Китовым, главы I, V—X написаны Н. А. Криницким. Авторы пользуются случаем выразить глубокую признательность академику С. Л. Соболеву и всему коллективу кафедры вычислительной математики Московского государственного университета имени Ломоносова, обсуждавшему проспект книги и давшему ряд ценных рекомендаций, В. Д. Розенкнопу, сообщившему ряд данных о машине М-3 и просмотревшему соответствующую часть текста, Ю. И. Янову, давшему ряд ценных советов по разделу о преобразованиях логических схем программ; авторы глубоко признательны также В. Б. Орлову и М. М. Горячей, редактировавшим книгу и давшим ряд ценных советов не только по структуре книги, но и по ее содержанию.

*Авторы*

# ВВЕДЕНИЕ

Создание быстродействующих электронных вычислительных машин представляет собой одно из выдающихся достижений научной и технической мысли середины двадцатого столетия. В течение короткого времени эти машины совершили бурный скачок в своем развитии от первых громоздких мало надежных моделей до современных высокопроизводительных машин и получили широкое признание и применение.

Масштабы и темпы работ в области электронной вычислительной техники можно сравнить только с масштабами и темпами работ в области атомной энергии и реактивной техники.

Современные электронные вычислительные машины выполняют десятки тысяч арифметических действий в секунду, оперируют с огромным количеством данных (миллионы чисел) и позволяют в короткие сроки, измеряемые часами и минутами, решать сложнейшие задачи, на которые при ручных вычислениях потребовались бы годы работы.

Помимо вычислительной работы, электронные вычислительные машины имеют важнейшее непосредственное применение в технике в качестве управляющих органов в сложных автоматических системах. Эти системы могут представлять собой как отдельные агрегаты, станки, так и целые комплексы, транспортные системы, автоматические заводы, электростанции, системы военного назначения.

Электронные вычислительные машины используются везде, где необходимо выполнять в большом объеме однообразную умственную работу по определенным правилам или, как говорят, по определенным алгоритмам.

*Алгоритмом* называется система формальных правил, четко и однозначно определяющих процесс выполнения заданной работы.

Во всех областях умственной деятельности людей существует чрезвычайно большое количество разнообразных процессов, которые выполняются по строго определенным правилам. Так, например, в банковском деле или в статистике обработка поступающих материалов и их анализ производятся по строго определенным правилам, которые могут быть описаны в виде алгоритма. Аналогичное положение имеет место и при расчете заработной платы, определении стоимости изделий, планировании производства или снабжения и т. д.

Во многих случаях такие алгоритмы еще не созданы или не описаны только потому, что в этом не было пока практической необходимости, но при желании требуемые алгоритмы могут быть сформулированы с затратой большего или меньшего труда.

Коль скоро тот или иной вид умственной работы представлен в виде алгоритма, появляется возможность автоматизировать его выполнение с помощью электронных цифровых машин.

Если раньше задачи технического прогресса концентрировались в основном вокруг проблем замены физического труда человека работой машин (развитие средств производства, передвижения, связи, наблюдения и измерения и т. д.), то середина XX века ознаменовалась бурным развитием средств механизации умственного труда, и, несомненно, прогресс в этом направлении приведет к быстрому общему прогрессу во всех сферах человеческой деятельности.

Первая промышленная революция, которая началась более двухсот лет тому назад изобретением ткацких машин, привела к замене человеческих рук машинами почти во всех областях производства и вызвала колоссальный подъем производительности труда.

Сейчас мы переживаем вторую промышленную революцию, заключающуюся в применении машин для выполнения различных видов умственной работы людей, и последствия этой революции для дальнейшего развития человеческого общества в настоящее время даже трудно себе представить.

Следует заметить, что появление электронных цифровых вычислительных машин имеет большое значение и для развития комплекса биологических наук, и в первую очередь для изучения процессов высшей нервной деятельности, так как с помощью этих машин представляется возможным создать модели отдельных элементарных процессов работы нервной системы и процессов мышления и тем самым ближе подойти к раскрытию закономерностей в этой области.

Одной из важных особенностей техники электронных вычислительных машин является то, что в ней сочетается большой комплекс различных областей современной науки и техники, таких, как математический численный анализ, математическая логика, электроника, импульсная техника, физика полупроводников; она использует достижения этих областей и стимулирует их дальнейшее развитие.

Значение электронных вычислительных машин для коммунистического строительства в нашей стране трудно переоценить.

Широкое применение электронных вычислительных машин в нашей стране должно обеспечить резкий подъем советской науки и техники на новую более высокую ступень. Применение электронных машин для автоматического управления производственными процессами приведет к значительному повышению производительности труда, улучшению качества продукции и экономии материалов и энергии. В отличие от капиталистического общества, где внедрение электронных автоматических устройств влечет за собой увольнение трудящихся и ухудшение условий их жизни, в социалистическом обществе электронная автоматика и в том числе электронные вычислительные машины облегчают условия труда людей, освобождают их от наиболее трудоемкой, утомительной и однообразной умственной работы и способствуют, в конечном счете, повышению материального благосостояния трудящихся.

В нашей стране электронные машины находят применение для автоматического управления производственными

процессами, представляющими опасность для здоровья и жизни людей, как, например, в некоторых видах химической промышленности.

Важной областью будущего применения электронных цифровых машин является механизация и автоматизация процессов административно-хозяйственного управления, вплоть до государственного планирования, учета и контроля.

В тезисах доклада Н. С. Хрущева на 21 съезде КПСС отмечается большое значение электронной вычислительной техники и необходимость ее широкого развития. Указывается, что «переход к комплексной механизации и автоматически управляемому производству с применением средств электронной техники составляет наиболее характерную черту современного технического прогресса... Широкие перспективы в области автоматизации производственных процессов открывают достижения вычислительной техники. Применение современных вычислительных машин для управления производственными процессами позволяет автоматически выбирать и вести технологический процесс на наивыгоднейшем режиме... Успехи вычислительной математики имеют непосредственную связь с развитием автоматики».

Партия и правительство уделяют большое внимание развитию электронной вычислительной техники и ее применению в различных областях науки, техники и экономики.

## § 1. Основные этапы развития вычислительной техники

Усилия ученых и инженеров были с давних пор направлены на изыскание путей, облегчающих решение задач, требующих больших вычислений, на создание средств для механизации этой утомительной работы.

Вычислительная техника развивалась в двух направлениях — по пути создания 1) вычислительных машин и приборов непрерывного действия и 2) вычислительных машин и приборов дискретного счета или цифровых.

**1. Вычислительные устройства непрерывного действия.** В вычислительных машинах *непрерывного действия* математические величины изображаются в виде непрерывных значений каких-либо физических величин (длин, углов, напряжений и т. д.).

Примерами вычислительных устройств непрерывного действия являются обычная логарифмическая линейка и различные механические приборы: планиметры, интеграфы, интегриметры и др.

Для вычислительных устройств непрерывного действия точность вычислений ограничивается качеством изготовления отдельных узлов и принятыми допусками и доходит в лучших случаях до трех-четырех верных значащих цифр результата. Существенное повышение точности решения встречает непреодолимые технологические и эксплуатационные трудности. Это является принципиальным недостатком машин непрерывного действия и не имеет места в машинах дискретного счета.

Вычислительные машины непрерывного действия конструктивно состоят из целого ряда отдельных блоков, каждый из которых служит для выполнения одной какой-либо математической операции (сложения, вычитания, умножения, деления, интегрирования, образования заданной функции и т. д.). Эти блоки соединяются между собой в последовательности, отвечающей конкретному виду решаемого уравнения. Если вычислительная машина предназначается для решения только одного вида уравнений, то состав математических устройств машины и их соединение между собой постоянны. Такие машины являются узко специализированными. В качестве примера узко специализированной вычислительной машины непрерывного действия можно привести известные приборы управления артиллерийским зенитным огнем *(ПУАЗО)*.

При создании вычислительных машин непрерывного действия их стараются обычно сделать достаточно гибкими, позволяющими решать сравнительно широкий круг задач. С этой целью в машинах предусматривается возможность изменения как состава математических устройств, участвующих в решении задачи, так и порядка соединения этих устройств.

Любая задача на вычислительной машине непрерывного действия решается таким образом, что в необходимый момент времени на всех устройствах машины, участвующих в решении задачи, производятся одновременно все требуемые уравнением математические преобразования, соответствующие текущему значению независимого переменного. Отсюда ясно, что тип и сложность математических задач, которые могут быть решены на машине непрерывного действия, ограничены наличным составом оборудования машины.

Таким образом, все вычислительные машины непрерывного действия являются более или менее специализированными машинами.

Наиболее важным классом машин непрерывного действия являются электронные машины, предназначенные для интегрирования систем обыкновенных дифференциальных уравнений. Эти машины называются электронными интеграторами или электронными моделирующими установками. Последнее название объясняется тем, что с помощью этих установок воспроизводятся математические зависимости, описывающие (моделирующие) движение различных динамических систем.

В электроинтеграторах математические действия осуществляются с помощью электрических решающих схем, а участвующие в решении задачи величины изображаются в виде напряжений.

Электронные интеграторы относятся к числу современных быстродействующих математических машин.

Весьма большое значение для отработки различных систем автоматического управления и регулирования

движущихся объектов приобрел в последние годы метод моделирования путем сочетания реального образца аппаратуры управления объектом с интегрирующей машиной. Интегрирующая машина в данном случае заменяет собой движущийся управляемый объект. Она решает уравнения движения объекта и подает в аппаратуру управления электрические сигналы, характеризующие движение объекта: скорость, ускорение, характеристики колебательного движения и т. д. Таким образом, аппаратура управления получает от интегрирующей машины такие же сигналы, какие она получала бы от измерительных элементов, определяющих положение объекта во время реального движения. На основе сигналов о положении объекта аппаратура управления вырабатывает управляющие сигналы, которые подаются в качестве дополнительных входных данных в интегрирующую машину для решения уравнений движения. Таким образом, в лабораторной обстановке создаются условия для экспериментальной проверки и отработки любой аппаратуры управления движущимися объектами. Этот метод имеет большое значение, в частности, при отработке систем управления реактивных снарядов, самолетных автопилотов и других объектов.

Ясно, что создать реальные условия для работы аппаратуры управления можно только в том случае, если сигналы о движении объекта, вырабатываемые интегрирующей машиной, поступают в аппаратуру управления в естественном темпе времени. Для этого интегрирующая машина при решении уравнений движения использует в качестве независимого переменного текущее время. Это в свою очередь требует, чтобы машина решала достаточно быстро уравнения движения для кратковременных процессов. Принципиальной особенностью электроинтеграторов является то, что они имеют в качестве независимой переменной текущее время, в частности, и при решении дифференциальных уравнений, описывающих быстро протекающие процессы. Таким образом, моделирование уравнений движения путем сочетания математических машин с реальной аппаратурой управления объектами представляет собой такую область применения электроинтеграторов, в которой они пока еще не могут быть успешно заменены другими типами вычислительных машин.

Кроме того, к достоинствам электроинтеграторов относят сравнительную простоту конструкции, невысокую стоимость изготовления, компактность устройства.

Помимо моделирования, электроинтеграторы успешно применяются в различных научно-исследовательских учреждениях, в конструкторских бюро и на заводах для быстрых прикидочных инженерных расчетов, не требующих большой точности, и для качественного исследования различных дифференциальных уравнений.

**2. Цифровые вычислительные машины.** Рассмотрим второе направление в развитии вычислительной техники — развитие ц и ф р о вых вычислительных машин и приборов. Известным примером этого типа устройств является арифмометр. Сюда же относятся различные ручные счетно-клавишные машины.

На арифмометре и ручных счетно-клавишных машинах производятся только арифметические действия, но, используя широко разработанные численные методы математики, можно с помощью этих машин решать самые разнообразные математические задачи.

В цифровых машинах числа представляются в виде последовательности цифр, переменные величины — в виде последовательности их значений. Для изображения каждой цифры применяется какой-либо прибор (элемент), который может находиться в одном из нескольких резко разграниченных между собой состояний. Каждому состоянию элемента поставлена в соответствие определенная цифра. Для изображения числа служит набор таких элементов. Решение задачи сводится к выполнению отдельных арифметических действий над исходными числами. Поэтому цифровые машины часто называются также *машинами дискретного счета*.

Основными преимуществами цифровых вычислительных устройств перед математическими устройствами непрерывного действия являются значительно большая точность вычислений и универсальность.

Создать вычислительную машину непрерывного действия, дающую результаты решений с большей точностью, чем достижимая в настоящее время точность изготовления устройств, входящих в ее состав, — невозможно. Напротив, при создании цифровой вычислительной машины всегда можно обеспечить любую заданную т о ч н о с т ь ее работы, что достигается включением в ее состав достаточного количества элементов, изображающих разряды чисел. При этом требования к точности изготовления и стабильности работы самих элементов не повышаются. Достаточно, чтобы эти элементы — электронные лампы, реле, зубчатые колеса и др. — имели определенное количество резко выраженных фиксированных состояний, например, проводимость или непроводимость тока, замыкание или размыкание контакта, зацепление колеса на определенном зубце и т. д.

Таким образом, само цифровое устройство, производящее вычисления с высокой точностью, строится из элементов, имеющих относительно невысокую точность и стабильность работы и не требующих высокой точности изготовления. Важно только, чтобы разброс характеристик элементов и нестабильность их работы не превосходили пределов, при которых стирается разница между отдельными фиксированными состояниями элементов. В этом заключается п е р в о е важное преимущество принципа действия машин дискретного счета перед принципом работы машин непрерывного действия.

В т о р ы м важным преимуществом цифровых вычислительных машин перед машинами непрерывного действия является унив е р с а л ь н о с т ь их применения. В отличие от машин непрерывного действия цифровые вычислительные машины не моделируют решаемое уравнение, а служат для реализации численного алгорифма решения задачи. Известно, что широко разработанные математические численные методы сводят решение самых разнообразных математических задач к последовательному выполнению четырех арифметических действий:

сложения, вычитания, умножения и деления. При решении некоторых задач также пользуются и таблицами функций (тригонометрических, логарифмов и других), однако в принципе любая задача может быть решена и без использования таких заранее составленных таблиц функций, если допустить, что необходимые значения функций будут по мере надобности вычисляться по соответствующим приближенным формулам (с нужной степенью точности). Цифровые вычислительные машины выполняют четыре действия арифметики и тем самым решают разнообразные типы задач, т. е. являются в принципе машинами универсального назначения.

Длительное время существенным недостатком цифровых вычислительных машин была малая скорость вычислений. Для выполнения одной арифметической операции на арифмометре или на ручных счетно-клавишных машинах оператор производит сравнительно большое количество движений, связанных с вводом исходных чисел, выполнением самой операции и записью результата.

Затем появились электрифицированные настольные счетные машины, в которых был автоматизирован процесс выполнения арифметических операций. Однако эти машины, так же как и ручные арифмометры, служили только для выполнения индивидуальных операций: они требовали каждый раз задания от руки в машину исходных чисел, над которыми производилось действие, и указания нажатием кнопки, какое действие произвести.

Счетно-аналитические машины. В начале текущего столетия стали создаваться *счетно-аналитические* или *перфорационные машины,* необходимые для удовлетворения потребностей статистики, бухгалтерского учета и финансово-банковского дела.

В перфорационных машинах механизирован процесс задания чисел, над которыми выполняются операции, и процесс ввода чисел отделен от процесса выполнения арифметических операций. Для этого каждая цифра числа изображается в виде системы отверстий на соответствующем месте стандартного листа тонкого картона, называемого *перфокартой.* На перфокарты заранее заносятся в виде системы отверстий также специальные «команды» для управления работой машины — указания, какие операции и над какими числами производить. Колода перфокарт закладывается в машину.

Последовательно поступающие из колоды перфокарты «ощупываются» системой электрических контактов машины, которые в случае наличия отверстия на определенном месте перфокарты замыкаются и посылают сигнал в соответствующие блоки машины. В случае отсутствия отверстий сигнал не посылается.

В счетно-аналитических машинах настройка машины на решение той или иной задачи осуществляется посредством *коммутации.* Например, в многосчетчиковой суммирующей машине, так называемом табуляторе, сигналы, получаемые от перфокарты, направляются в необходимый разряд любого из счетчиков, что обеспечивается соответствующей коммутацией — соединением входных и выходных контактов отдельных устройств машины. Коммутация или настройка машины на решение определенной задачи осуществляется перед началом работы машины и сохраняется неизменной в течение всего процесса решения.

Постоянная коммутация не обеспечивает необходимой гибкости в работе, т. е. не позволяет перестраивать машину в ходе решения. Однако при решении математических задач часто возникает необходимость выбирать то или иное направление для продолжения вычислительного процесса в зависимости от получаемых промежуточных результатов или в зависимости от какого-либо другого признака. Поэтому уже в счетно-аналитических машинах были введены специальные схемы, позволяющие изменять установленные в машине направления передачи сигналов, получаемых от перфокарт, в зависимости от наличия пробивок на перфокартах в определенных позициях.

Например, в табуляторе производится либо последовательное суммирование чисел, если поступающие в машины перфокарты обладают определенным признаком, либо выдача накопленной суммы на печатающее устройство и переход к последовательному суммированию нового ряда чисел, поступающих с перфокарт, обладающих другим признаком. Выбор того или иного варианта для продолжения работы в табуляторе осуществляется путем сравнения двух последовательно поступающих перфокарт. Если совпадают пробивки в определенной позиции на обеих перфокартах, то производится суммирование. Если пробивки в этом месте не совпадают, то выдается ранее накопленная сумма и машина переходит к получению следующей суммы.

В процессе вычислений необходимо использовать полученные промежуточные результаты для дальнейших вычислений. С этой целью в счетно-аналитических машинах выдаются результаты не только в виде отпечатанных таблиц, но и в виде пробивок на итоговых перфокартах, которые вновь вставляются в машину человеком-оператором.

Процесс вычислений на счетно-аналитических машинах не полностью автоматизирован: для перехода от этапа вычислений, выполняемых одной машиной, к этапу вычислений, выполняемых другой машиной, а также при вводе в машину итоговых перфокарт требуется участие человека.

Автоматические быстродействующие цифровые вычислительные машины. Счетно-аналитические машины показали высокую эффективность в решении задач статистики, учета и т. п., но оказались неприспособленными для решения математических задач.

Для решения математических задач характерны:

1) необходимость выполнения больших последовательностей различных арифметических и логических операций;

2) наличие большого количества промежуточных результатов, которые сразу же используются для продолжения вычислений;

3) необходимость часто изменять направление вычислительного процесса в зависимости от результатов промежуточных вычислений.

Таким образом, для того чтобы одна машина полностью решала математические задачи, она должна производить все элементарные арифметические и логические операции, необходимые для решения задачи, обладать достаточным объемом устройства для хранения промежуточных и исходных величин и быть способной автоматически выбирать нужное продолжение дальнейших вычислений в зависимости от результатов промежуточных операций. Появление таких полностью автоматических вычислительных машин относится к сороковым годам настоящего столетия.

Интересно отметить, что главные идеи, положенные в основу построения современных программно-управляемых электронных машин, были высказаны свыше ста лет тому назад. Их высказал Чарльз Бэббидж, профессор математики Кэмбриджского университета в Англии. В 1833 г. он предложил проект «аналитической машины», предназначенной для составления таблиц различных функций, которые должны были вычисляться автоматически путем их аппроксимации многочленами.

Проект Бэббиджа не был осуществлен, так как в то время не было необходимых материально-технических условий для создания подобной машины.

Материалы проекта были опубликованы только в 1888 г., хотя в 1842 г. итальянским математиком Менабреа были опубликованы лекции, посвященные аналитической машине Бэббиджа.

В проекте Бэббиджа предусматривались основные части машины, характерные для современных машин: арифметическое устройство, названное «mill» — мельница; запоминающее устройство емкостью в 1000 пятидесятиразрядных десятичных чисел; устройство автоматического управления с хранением программы на перфокартах.

Основные принципы программного управления — условный переход, т. е. выбор того или иного направления вычислений в зависимости от предыдущих результатов, и модификация команд, т. е. преобразование адресов в командах, — также были предусмотрены Бэббиджем.

В современном виде основные идеи и принципы построения электронных программно-управляемых машин были сформулированы известным американским математиком Джоном фон Нейманом в 1946 г.

Создание первых образцов современных программно-управляемых машин является плодом коллективного труда многих людей.

Вначале автоматические вычислительные машины *(Марк*-I и *Марк*-II, США) строились как релейные машины из стандартных элементов (реле, счетчики, перфораторы и т. д.) счетно-аналитических машин с централизованной системой управления. Решение задачи сводилось к последовательному выполнению операций различными блоками машины, которые автоматически включались и выключались системой управления в соответствии с заранее составленной программой работы машины.

Программа содержала всю последовательность операций, необходимых для решения задачи. Эта последовательность операций кодировалась в виде чисел и записывалась в виде пробивок на длинной перфоленте. В процессе работы машины лента двигалась в постоянном направлении и последовательно вводила в устройство управления машины очередные команды, определявшие, какой блок включать в работу, какую операцию и над какими числами выполнить и куда передать результат операции. Скорость работы таких машин была очень мала: сложение двух чисел 0,3—0,5 *сек,* умножение — 5—6 *сек,* деление — 15 *сек.*

Составление программы для такой машины, а затем запись ее в виде пробивок на перфоленте для многих задач, требующих длительных вычислений, было делом очень трудоемким, что допускало применение этих машин только для многократного решения однообразных задач по одной и той же программе. Кроме того, машины такого типа, имея жесткую программу, не обеспечивали возможности автоматического изменения направления вычислительного процесса в зависимости от промежуточных результатов.

Для повторения часто встречающихся последовательностей операций (например, при вычислении квадратного корня, логарифма и т. Д.) соответствующие последовательности команд заносились на замкнутые перфоленты. Особая команда на основной перфоленте вводила в действие одну из дополнительных перфолент, а после ее выполнения продолжалась выдача команд с основной перфоленты.

Серьезным недостатком первых автоматических вычислительных машин являлась также малая емкость запоминающих устройств. Так, в машине *Марк*-II было внутреннее запоминающее устройство на сто десятизначных десятичных чисел. Эта машина имела в своем составе около 13 000 специальных шестиполюсных электромеханических реле.

Дальнейшее развитие автоматических вычислительных машин происходило путем увеличения скорости вычислений, увеличения объема запоминающих устройств и усовершенствования системы программного управления с целью расширения круга решаемых задач.

Резко увеличилась скорость работы автоматических вычислительных машин в результате перехода от механических элементов к электронным. Электронные счетчики строятся на основе схемы триггерной ячейки, которая была изобретена еще в 1918 г. советским ученым М. А. Бонч-Бруевичем. Если в механических счетчиках переход от одного числа к другому связан с перемещением некоторых инерционных деталей (вращение зубчатых колес, движение якоря реле и др.), то в электронных схемах переход от одного числа к другому связан с изменением состояния практически безынерционной системы электронных ламп и происходит в тысячи раз быстрее, чем в

механических счетчиках.

Первые быстродействующие электронные вычислительные машины, в которых были применены для построения счетных и запоминающих устройств электронные лампы, появились в 1945— 1950 гг. В быстродействующих вычислительных машинах арифметические операции выполняются в десятки тысяч раз быстрее, чем на обычных машинах. Новые запоминающие устройства в отличие от перфоленты, движущейся в постоянном направлении, позволяют изменять хранящиеся в них команды и переходить от каждой команды не только к последующей, но и к любой из предыдущих. Поэтому одновременно с введением в машину быстродействующих электронных элементов был сделан решающий шаг в области усовершенствования управления автоматической работой машины.

Вместо развернутых неизменных программ, содержащих полностью весь перечень операций, выполняемых машиной, стали применяться циклические программы, изменяемые самой машиной в процессе вычислений. При этом резко уменьшился как объем программ, так и работа по их составлению. Значительно возросла эффективность применения машин.

Однако составление программ для решения задач на быстродействующих цифровых вычислительных машинах является и в настоящее время достаточно сложным делом, требующим определенного навыка и знания особенностей конструкции и возможностей машины. Неудачно и нерационально составленная программа приводит к увеличению времени решения и непроизводительной работе машины, т. е. снижает эффективность применения машины.

Эта особенность универсальных цифровых быстродействующих вычислительных машин — сложность и трудоемкость процесса предварительной подготовки задач к решению на машине — является в настоящее время известным недостатком данного класса машин. Для устранения этого недостатка разрабатывается методика составления программы с помощью самих вычислительных машин (автоматическое программирование) и создаются специальные устройства для механизации процесса программирования. Высокая скорость выполнения операций, автоматическое управление и возможность применения самих машин для составления программ решения значительно расширяют круг решаемых математических задач.

Существенно изменилась и методика вычислительного процесса. Например, в быстродействующих машинах вместо общепринятой десятичной системы счисления стала применяться двоичная; эта система обладает целым рядом преимуществ для конструкции машин, о которых скажем ниже, но требует дополнительной работы по переводу исходных данных из десятичной системы в двоичную и результатов вычислений из двоичной системы в десятичную. Большая скорость выполнения операций позволила осуществлять эти преобразования очень быстро с помощью самой машины. Кроме того, так как быстродействующие машины предназначаются для решения математических задач, в которых результаты появляются после длинной цепочки вычислений, то добавление в начале и в конце вычислений небольших участков по переводу чисел из одной системы счисления в другую не приводит к существенному увеличению времени решения задач.

Если в первые годы развития быстродействующих цифровых вычислительных машин разрабатывались главным образом машины универсального назначения, то в настоящее время большое значение приобретают также специализированные цифровые быстродействующие вычислительные машины.

Специализированные цифровые машины строятся таким образом, что программа вычислений полностью или частично задается в машине схемно, путем соответствующей коммутации, и является постоянной. Известны специализированные цифровые машины следующих назначений:

1) машины для решения систем обыкновенных дифференциальных уравнений — цифровые дифференциальные анализаторы;

2) машины для вычисления коэффициентов и функций корреляции — корреляторы;

3) машины для решения систем линейных алгебраических уравнений;

4) специализированные машины для решения задач теории вероятностей и уравнений в частных производных методом статистических испытаний (методом Монте-Карло).

Кроме того, известны специальные цифровые вычислительные машины для математического предсказания погоды, информационной работы, переводов с одного языка на другой и др.

Достоинствами специализированных машин является простота конструкции и эксплуатации, меньшая стоимость, меньшие габариты и высокая эффективность в решении заданного круга задач.

Развитие быстродействующих электронных вычислительных машин не исключает необходимости в настольных счетных машинах и математических инструментах. Эти средства вычислительной техники являются незаменимыми при выполнении небольших вспомогательных расчетов самого различного характера. Настольные счетные машины и математические инструменты в настоящее время развиваются в направлении уменьшения их габаритов, повышения скорости и точности работы.

В настоящее время в нашей стране построен ряд электронных цифровых машин как универсальных, так и специального назначения, не уступающих по своим характеристикам большинству зарубежных машин. Первая быстродействующая электронная счетная машина *(БЭСМ),* созданная в нашей стране в 1953 г. в АН СССР под руководством академика С. А. Лебедева, долгое время была наиболее быстродействующей машиной в Европе.

Нашей промышленностью выпущено несколько образцов быстродействующей цифровой вычислительной машины *Стрела,* разработанной под руководством Героя Социалистического Труда Ю. Я. Базилевского, которые эффективно используются в ряде научно-исследовательских учреждений страны.

Серийно выпускается нашей промышленностью и находит широкое применение малая электронная цифровая вычислительная машина *Урал,* разработанная под руководством инженера Б. И. Рамеева. Под руководством члена-корреспондента АН СССР И.С. Брука разработаны электронные цифровые вычислительные машины *М-2* (средняя машина) и *М-3* (малая машина), которые также используются для практического решения различных задач.

Выдающимся достижением отечественной науки и техники является создание в 1958 г. новой быстродействующей машины, выполняющей 20 000 операций в секунду. На машинах решено большое количество важных и сложных задач. Масштабы производства и применения электронных цифровых вычислительных машин непрерывно расширяются.

Подводя итоги настоящего параграфа, рассмотрим общую классификацию современных быстродействующих вычислительных машин. Как мы уже знаем, эти машины по способу представления величин делятся на два основных типа:

1) машины непрерывного действия, представляющие собой математические модели, воспроизводящие физические процессы, описываемые решаемыми уравнениями;

2) машины дискретного действия или цифровые. Быстродействующие цифровые машины по принципу действия

в свою очередь делятся на

а) универсальные машины с программным управлением, реализующие любые алгорифмы решения задач;

б) специализированные машины с жестким программным управлением, реализующие один или несколько определенных алгорифмов решения задач.

По своему назначению современные вычислительные машины делятся на

1) машины вычислительные, предназначенные для решения задач, вводимых в машину человеком;

2) машины управляющие, предназначенные для обработки информации, поступающей в машину непосредственно от внешних объектов.

Приведенная классификация не учитывает деления машин по конструктивным признакам. Кроме того, возможны сочетания в одной машине одновременно нескольких принципов. Например, электронная вычислительная машина *Тайфун* (США) построена с использованием дискретного и непрерывного принципов действия.

Наибольшее развитие и применение в настоящее время получили два основных типа машин:

а) универсальные вычислительные машины дискретного действия с программным управлением;

б) моделирующие математические устройства непрерывного действия.

В дальнейшем мы будем рассматривать электронные машины (как вычислительные, так и управляющие) дискретного действия с программным управлением. Эти машины мы будем называть более кратко *электронными цифровыми машинами.*

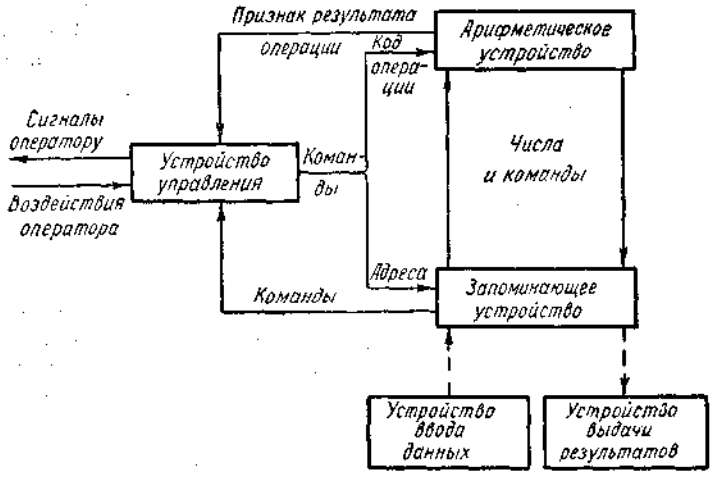
## § 2. Общая структурная схема электронной цифровой машины и принцип программного управления

**1. Структурная схема.** С точки зрения принципа работы любая электронная цифровая машина универсального назначения может рассматриваться как состоящая из трех основных частей (см. схему):

1) арифметического устройства, предназначенного для выполнения операций над числами;

2) запоминающего устройства, предназначенного для приема, хранения и выдачи чисел;

3) устройства управления, предназначенного для управления автоматической работой машины.




Упрощенная блок-схема цифровой вычислительной машины.

Арифметическое устройство служит для выполнения арифметических и логических операций над числами. Оно построено на основе счета электрических импульсов, представляющих числа в двоичной системе счисления. Так

как каждая цифра в двоичной системе имеет только одно из двух значений (0 или 1), то такие цифры могут быть представлены наличием или отсутствием электрического импульса.

К запоминающим устройствам в электронных цифровых машинах предъявляются два основных требования, которые до настоящего времени не удалось вполне реализовать в одном устройстве:

а) требование высокой скорости приема и выдачи чисел;

б) требование большой емкости запоминания.

В связи с этим запоминающее устройство в машине обычно состоит из двух отдельных устройств: внутреннего запоминающего устройства и внешнего запоминающего устройства. Удобно внутреннее запоминающее устройство называть памятью машины, а внешнее запоминающее устройство — накопителем.

Память имеет небольшую емкость; у большинства современных машин она рассчитана на одновременное хранение 1024 или 2048 чисел. Память непосредственно связана с арифметическим устройством и служит для выдачи чисел, участвующих в операции, и приема результатов. Она хранит обычно только те данные, которые необходимы для ближайшего ряда вычислений.

Внешние накопители обладают практически неограниченной емкостью, но имеют значительно меньшую скорость работы. Они не связаны непосредственно с арифметическим устройством.

Устройство автоматического управления служит для управления работой машины в процессе вычислений. Оно обеспечивает последовательное выполнение операций по программе, а также управление работой машины при выполнении отдельных операций.

Устройства ввода данных в машину и вывода результатов связаны непосредственно с запоминающими устройствами машины. Устройства ввода данных служат для восприятия вводимых данных, представленных в виде пробивок на перфокартах или перфолентах, преобразования их в электрические сигналы и передачи в запоминающие устройства машины.

Выводное устройство машины служит для преобразования выводимого из машины числового материала, представленного электрическими сигналами, в систему пробивок на перфокартах или перфолентах. Иногда ввод и вывод данных осуществляются с помощью записи на магнитные ленты. При вводе запись на ленту производится на специальном устройстве вне машины, а затем лента вставляется в машину. При выводе, наоборот, числа записываются на ленту внутри машины, и лента после окончания решения вынимается из машины.

Несмотря на то, что решение задачи, заключающееся в последовательном осуществлении предусмотренных программой операций, выполняется автоматически, в машинах имеется система ручного управления и контроля. Ручное управление служит для запуска и останова машины. Оно позволяет выполнять по отдельным операциям введенную программу в том случае, если возникает необходимость проверить работу машины.

Кроме рассмотренных устройств, образующих собственно вычислительную машину и связанных между собой системой магистралей, в состав любой электронной цифровой вычислительной машины входят еще внешние устройства, которые не имеют электрической связи с машиной.

Внешние устройства служат для подготовки исходных данных к вводу в машину и для оформления результатов решения. Обычно эти устройства работают медленнее, чем сама машина, и поэтому в некоторых машинах их ставят несколько комплектов для параллельной работы по подготовке или оформлению сразу целого ряда задач. Программа вычислений вводится в машину таким же образом, как и другие исходные данные.

Во внешних устройствах исходные данные и программа, записанные на бумаге, переносятся сначала на перфокарты, магнитные ленты или перфоленты, с которых в дальнейшем данные переписываются в запоминающее устройство машины.

К внешним устройствам относится также печатающее устройство, которое служит для печатания в виде числовых таблиц результатов решения задач, получаемых в машине в виде отверстий на перфокартах, перфолентах или в виде записи на магнитной ленте.

**2. Принцип программного управления.** Рассмотрим более подробно принцип программного управления.

Машина выполняет ограниченное число основных операций: сложение, вычитание, умножение, деление, перенос числа из одного места памяти в другое и некоторые другие. Поэтому решение задачи предварительно сводится к выполнению этих операций.

Каждая операция выполняется машиной под воздействием специального управляющего сигнала — команды.

*Команда* — это информация, представленная в форме, позволяющей ввести ее в машину, и определяющая действия машины в течение некоторого отрезка времени.

В общем случае команда как группа символов, воспринимаемых машиной, делится на несколько подгрупп. Одна подгруппа является *кодом операции* и определяет, что должна сделать машина — характер или вид операции; остальные подгруппы, называемые *адресами,* указывают, откуда взять числа для выполнения операции и куда направить результат операции. В качестве адресов могут указываться номера ячеек запоминающего устройства, где должны находиться эти числа. Команды программы для ввода в машину кодируются в виде чисел и хранятся в машине в ячейках памяти таким же образом, как и обычные числа.

Последовательность команд образует *программу* работы машины. Программа заранее составляется для каждой задачи и вводится в машину вместе с исходными данными перед решением задачи; после этого все решение выполняется машиной автоматически.

Команды программы выдаются из памяти машины для выполнения поочередно в заранее заданном порядке. По каждой команде выбранные из памяти в соответствии с адресами команды числа поступают в арифметическое устройство, в котором над ними производится операция, определяемая кодом операции; результат операции посылается обратно в память, в ячейку, номер которой указан в соответствующем адресе команды. Записью результата операции в память закапчивается выполнение данной команды, после чего устройство управления извлекает из памяти следующую команду.

Помимо команд арифметических операций, машиной выполняются и команды другого рода, необходимые для автоматической работы машины (команды передачи данных из одного места памяти в другое, выдачи данных из машины, остановки и др.).

## § 3. Применение электронных цифровых машин

**1. Применение для научных и технических исследований и разработок.** С развитием науки и техники непрерывно возрастает потребность в проведении сложных вычислений, так как развивается и усложняется применяемый математический аппарат, повышается удельный вес математических методов исследования, повышаются требования к точности анализа, к учету все большего количества различных факторов, влияющих на ход того или иного процесса, причем в большинстве случаев результаты решения задачи имеют практическую ценность лишь при условии достаточно быстрого выполнения расчетов.

Особенно большая необходимость в выполнении сложных вычислений возникла в последние годы в связи с развитием атомной физики, радиолокации, техники управляемых ракет.

Достигнутый в настоящее время уровень развития математической теории позволяет в принципе исследовать почти любую проблему, т. е. существующие математические методы дают возможность сформулировать в виде системы уравнений почти любую задачу, исходя из ее физического содержания. Но практическое решение составленных уравнений часто сопряжено с огромными трудностями. С другой стороны, существующие математические численные методы являются достаточно универсальными и позволяют решать самые разнообразные задачи с любой наперед заданной степенью точности. Однако для численного решения обычными ручными вычислительными средствами часто требуется настолько много времени, что задача практически оказывается неразрешимой.

Например, системы дифференциальных уравнений, описывающие пространственное движение телеуправляемых аэродинамических объектов, составленные с учетом колебательного движения объекта относительно его центра масс и с учетом работы автопилотов, являются настолько громоздкими, что совершенно недоступны для аналитического решения, а численное решение этих уравнений потребовало бы многих лет работы обычных вычислительных бюро.

Большое количество задач в еще недавнее время вообще не ставилось из-за невозможности практически решить их в приемлемые сроки. В этих случаях задачи решались упрощенно: рассматривались частные случаи, вводились упрощающие допущения, например, пренебрегали нелинейностями, не учитывали влияния некоторых факторов и т. п. При таких упрощениях получалось лишь качественное решение задачи, а часто даже решение, качественно отличное от действительного.

Электронные вычислительные машины обеспечивают резкий скачок в развитии физики и механики и, прежде всего, открывают возможность широкого практического применения общих математических методов в этих науках.

Применение электронных вычислительных машин позволяет по-новому организовать и процесс технического проектирования: путем быстрых расчетов различных вариантов конструкции и сравнения между собой этих вариантов выбирать оптимальные варианты и точно рассчитывать их характеристики. При этом процесс сравнения отдельных вариантов между собой и выбор оптимального варианта могут быть автоматизированы и выполняться самой вычислительной машиной по определенной программе.

Важна роль электронных вычислительных машин и для развития самой математики. Не говоря здесь о развитии численных методов, упомянем лишь о проведении на машинах различных математических экспериментов, имеющих целью исследование чисто математических задач. Такие задачи имеют место в теории чисел, в анализе, в математической логике.

Практика применения электронных вычислительных машин и анализ особенностей их конструкции показывают, что наиболее целесообразно применять эти машины для решения задач, требующих массовых расчетов различных вариантов по одной и той же программе.

Учитывая сравнительно малую скорость работы устройства ввода и вывода данных по сравнению со скоростью счета машины, целесообразно ставить для решения на электронных цифровых машинах такие задачи, которые имеют небольшой объем исходных данных и результатов решения и значительный объем счетной работы. К числу типовых задач, эффективно решаемых на электронных цифровых машинах, относятся:

1. Вычисление корней алгебраических уравнений. Необходимо при решении многих задач.

Для численного решения алгебраических уравнений высоких порядков эффективно применяются итерационные методы (метод Ньютона, метод хорд, метод Лобачевского).

2. Системы линейных алгебраических уравнений. Эти задачи имеют большое значение в науке и технике: при исследовании различных систем автоматического управления, при статистической обработке результатов наблюдений, при выполнении топографических расчетов.

Возможные методы решения систем алгебраических уравнений разделяются на две группы: итерационные и методы исключения. И те и другие состоят из серий стандартных вычислений и имеют отработанные программы.

3. Системы обыкновенных дифференциальных уравнений. Этот класс математических задач является важным и распространенным в науке и технике. Широко применяются обыкновенные дифференциальные уравнения во внешней и внутренней баллистике как управляемых, так и неуправляемых снарядов, в аэродинамике, в нелинейной теории колебаний, в теории автоматического регулирования и т. д.

Для численного решения систем обыкновенных дифференциальных уравнений используются широко известные методы Адамса — Штермера, Рунге—Кутта и др. Составлены программы, позволяющие интегрировать любое количество уравнений и даже автоматически выбирать шаг вычислений. Все эти методы благодаря присущей им цикличности вычислений очень удобны для применения на машине.

4. Уравнения в частных производных различных типов решаются в основном методом конечных разностей. Чрезвычайно большое значение имеет решение краевых задач для дифференциальных уравнений в частных производных в атомной физике, гидродинамике, газовой динамике и других областях науки и техники. Уже прочно вошли в практику массовое решение с помощью электронных цифровых машин задачи Дирихле, задачи Неймана и третьей краевой задачи для уравнений Лапласа и Пуассона, решение бигармонического уравнения, уравнения теплопроводности, волнового и телеграфного уравнений.

На электронных цифровых машинах может успешно производиться решение уравнений гиперболического типа по методу характеристик (задачи Коши, задачи Римана и смешанные задачи).

5. Вычисление значений кратных интегралов имеет большое применение в артиллерии для определения действительности стрельбы.

Для решения этих задач могут быть применены два метода: метод непосредственного суммирования и метод статистических испытаний (метод Монте-Карло). Метод непосредственного суммирования обеспечивает в принципе сколь угодно высокую точность, но требует даже для сравнительно простых задач огромной затраты машинного времени (дни и месяцы). Метод Монте-Карло, сводящийся к воспроизведению с помощью машины случайного процесса, описываемого данными интегралами, дает невысокую точность, но резко сокращает время решения задач.

6. Составление таблиц различных функций. С помощью электронной цифровой машины *БЭСМ* Академии наук СССР составлены, например, точные таблицы интегралов Френеля. На расчет 50 000 значений интегралов потребовалось всего около 2 часов машинного времени.

**2. Применение для обработки информации.** Электронные цифровые машины эффективно применяются для быстрой и точной обработки больших количеств самой разнообразной информации,

В экономической области эти машины используются для решения задач учета и планирования и, в частности, для расчета заработной платы, составления накладных, отчетов, составления графиков загрузки производства и использования рабочей силы, составления железнодорожных расписаний и т, д. Эти работы обычно выполняются в большом количестве различных вариантов (при разных исходных данных) по одним и тем же программам.

Внедрение машин зачастую связано с коренной перестройкой сложившегося при ручной работе порядка делопроизводства и прохождения документов. В отличие от прежних средств механизации конторского труда электронные цифровые машины обеспечивают не только выполнение расчетной работы, но и, самое главное, сортировку и анализ обрабатываемых данных, вплоть до выдачи законченных отчетных, справочных или исполнительных документов. В связи с этим программирование экономических и конторских задач является значительно более сложным делом, чем программирование математических вычислений. Но после того, как машинная методика обработки деловой информации разработана и составлены программы, эффект применения машин оказывается весьма высоким и эти машины быстро окупают расходы, связанные с их внедрением.

Наряду с созданием специальных электронных цифровых машин и разработкой алгоритмов решения экономических и конторских задач внедрение машин в указанных областях требует проведения широких научных исследований по применению математических методов в экономике, статистике и при анализе различных процессов управления. В связи с этим бурно развивается ряд разделов прикладной математики, таких, как теория игр, линейное программирование, динамическое программирование и др.

*Теория игр* представляет собой теоретическую основу методов построения алгоритмов, определяющих процессы борьбы против некоторых управляющих систем. Особенностями алгоритмов, определяющих процессы борьбы, являются необходимость учитывать наличие целеустремленного и организованного противодействия указанной системы и недостаток, как правило, некоторых сведений, характеризующих поведение этой системы. Последнее обстоятельство требует применения теоретико-вероятностных методов анализа. Подобные положения возникают не только в области вооруженной борьбы или экономической борьбы конкурентов; к задачам теории игр сводятся многие задачи экономического управления, организации производства, научных исследований и т. д.

*Линейное программирование* включает в себя методы решения широкого класса оптимизируемых задач, в которых участвует большое количество переменных, связанных между собой линейными зависимостями и подчиненных заданным ограничивающим условиям. При решении задач линейного программирования требуется найти такие

значения участвующих переменных, при которых некоторая величина, являющаяся их линейной формой и определяющая качество данного варианта решения (например, стоимость производства, транспортировки и т. п.), принимала бы наибольшее (наименьшее) значение. Методами линейного программирования решаются задачи транспортировки грузов, распределения производственных мощностей, материалов, заказов и т. д.

Задачи *динамического программирования* представляют собой развитие задач линейного программирования в направлении введения в рассмотрение элемента времени и представления процесса решения в виде последовательности выборов.

Кроме перечисленных областей математики, большое значение для применения машин в области автоматизации процессов обработки деловой информации имеют методы теории информации, определяющие способы оптимального кодирования, передачи и обработки информации и рациональные принципы построения систем обработки информации.

В настоящее время внедрение машин производится изолированно, т. е. путем их установки в отдельных вычислительных центрах или отдельных учреждениях. При этом ввод данных в машину и получение результатов производятся вручную, людьми. Следующим этапом применения машин для автоматизации конторского труда должно явиться создание некоторых разветвленных автоматизированных систем сбора и обработки информации, включающих в себя, помимо электронных цифровых машин, также автоматизированные линии связи и специальные автоматические устройства ввода данных в машины и выдачи результатов.

Для развития науки и техники будет иметь чрезвычайное значение применение *научно-информационных машин* с большой емкостью долговременной памяти.

В настоящее время имеет место колоссальное увеличение печатного материала по всем отраслям знаний и темпы поступления этих материалов непрерывно увеличиваются. Специалисты уже не в состоянии систематически просматривать литературу, относящуюся даже к достаточно узкому кругу вопросов. Положение значительно усложняется еще благодаря тому, что в связи с непрерывным разделением и взаимным проникновением различных наук сведения, относящиеся к одному и тому же вопросу, могут встречаться в самых различных областях науки. Библиографические справочники и информационные сборники не решают дела, так как их просмотр требует времени, а содержащиеся в них сведения являются далеко не полными и часто устаревшими. Зачастую легче вновь создать какую-либо схему или выяснить вновь какую-нибудь физическую закономерность, чем разыскать соответствующий литературный материал, если даже известно, что такой материал имеется. Многократное повторение научных открытий и изобретений становится в настоящее время типичной чертой развития науки и техники.

Научно-информационные машины должны обеспечить систематизацию научных знаний и отсеивание материалов повторяющихся, не имеющих ценности и затрудняющих поиск полезных сведений. Эти машины должны обеспечивать возможность быстрого просмотра и анализа содержания аннотаций и библиографии в соответствии с заданной тематикой и выдавать необходимые краткие сведения в виде микрофильмов или обычного печатного текста.

Аннотации печатных работ по мере их опубликования должны вводиться в машину в сокращенном, кодированном виде. Задаваемые вопросы также должны вводиться в машину в специальном кодированном виде, который будет определять порядок работы машины по поиску ответов. Поиск ответов машиной будет осуществляться при помощи программы, которая будет в некоторой степени воспроизводить процессы умственной работы человека, решающего аналогичную задачу.

В настоящее время разработаны технические средства, позволяющие создавать научно-информационные машины. Весьма трудной является разработка проблемы кодирования и поиска информации. Эта проблема решается путем создания на первом этапе узкоспециализированных научно-информационных машин с последующим постепенным расширением круга охватываемых вопросов.

**3. Применение для автоматического управления производственными процессами.** Автоматизация производственных процессов является основным направлением технического прогресса во всех отраслях промышленности. В настоящее время развитие техники достигло такого уровня, что во многих случаях дальнейшее повышение скоростей и точностей обработки, увеличение производительности или пропускной способности различного оборудования ограничивается физическими возможностями людей — операторов, следящих за ходом процессов и управляющих агрегатами. Сейчас назрела необходимость перехода к автоматическим системам управления производственными процессами, т. е. к таким системам, которые должны работать без непосредственного участия людей. Основой построения такого рода систем автоматического управления являются электронные вычислительные машины.

В общем случае процесс управления может включать в себя две стороны:

1) выработку команд управления и передачу их для исполнения управляемому объекту;

2) получение информации о действительном состоянии управляемого объекта и контроль за ходом выполнения выданных команд.

Первая сторона процесса осуществляется при помощи прямой (командной) связи между управляющим органом и управляемым объектом.

Вторая сторона процесса осуществляется при помощи обратной (информационной) связи, идущей от управляемого

объекта к управляющему органу.

Все автоматические системы делятся по принципу действия на три основных типа:

1. Системы жесткого управления, не имеющие замкнутой обратной связи. Эти системы работают, как правило, по заранее заданной жесткой неизменяемой программе, независимой от изменения условий работы и действительного движения управляемого объекта.

2. Системы автоматического регулирования с обратной связью, но с неизменной, жесткой структурной схемой. Такие устройства автоматически регулируют движение управляемого объекта, учитывая отклонения действительного движения объекта от требуемого, при каких-то постоянных, заранее определенных внешних условиях работы всей системы.

3. Самонастраивающиеся системы автоматического управления. Эти системы не только учитывают отклонения действительного движения управляемого объекта от требуемого, но и возможные изменения внешних условий работы всей автоматической системы в целом, автоматически приспосабливаясь к этим изменениям путем перестройки своей внутренней структуры.

Электронные цифровые машины находят применение в системах автоматического управления главным образом первого и третьего типов.

В первом типе автоматических систем цифровые машины выполняют роль устройств, задающих программу (командные вычислительные машины), или роль устройств, обрабатывающих информацию для выработки необходимых команд. Ясно, что при отсутствии обратной связи обрабатываемая информация не будет содержать сведений о характере движения самого объекта регулирования.

Примером применения цифровых машин для жесткого управления является программное управление металлорежущими станками, где цифровые машины используются для обработки данных о размерах деталей и вычисления нужного хода станка.

В третьем типе автоматических систем — самонастраивающихся систем — электронные цифровые машины органически встраиваются в замкнутую цепь автоматического управления и служат для решения вариационных задач по выбору оптимальных режимов работы системы. Подробнее об этом будет сказано несколько ниже.

Во втором типе автоматических систем до последнего времени в основном применялись электронные вычислительные машины непрерывного действия, хотя в принципе возможно применение и здесь устройств дискретного счета.

Помимо автоматического управления, электронные цифровые машины эффективно могут применяться и для полуавтоматического управления и контроля за сложными производственными, энергетическими или боевыми системами. При этом основное управление, принятие окончательных решений, остается за человеком, а цифровая машина служит для сбора, систематизации и обработки большого количества данных, поступающих из различных мест, с целью представления этих данных в удобном виде для обозрения и анализа (например, в виде кривой на экране осциллографа). На машину может быть также возложена задача расчета вариантов будущего поведения системы при тех или иных возможных решениях оператора (человека) с тем, чтобы облегчить оператору анализ обстановки и принятие окончательного решения.

Применение электронных цифровых машин для целей автоматического управления открывает новые возможности и перспективы в развитии автоматики.

Строившиеся до настоящего времени автоматические системы, зачастую весьма сложные, предназначались для работы при заранее известных условиях. Эти системы реагировали на изменения условий работы в соответствии с заранее установленными правилами, но не обладали способностью автоматически приспосабливаться к изменениям свойств регулируемого объекта и внешних условий.

Применение электронных цифровых машин позволяет осуществлять оптимальное регулирование, т. е. регулирование с предварительной оценкой возможностей. Сущность оптимального регулирования заключается в следующем.

Цифровая машина, получая данные, характеризующие действительное состояние управляемого объекта и внешнюю обстановку, рассчитывает возможные варианты будущего поведения системы при различных способах регулирования, учитывая изменения внешних условий, полученные экстраполяцией. Анализируя найденные решения на основе выбранного или заданного заранее критерия (например, по минимуму времени переходного режима), цифровая машина выбирает оптимальный вариант процесса регулирования.

Такие системы управления обладают свойством самоорганизуемости, т. е. способностью автоматически изменять свои параметры для достижения оптимальности процесса.

Важным достоинством электронных цифровых машин с точки зрения их использования в системах автоматического управления является большая гибкость применения.

Одна цифровая машина с программным управлением может заменить целый ряд вычислительных устройств непрерывного действия, особенно если эти устройства не обязательно должны работать одновременно. Для этого достаточно в цифровой машине иметь набор соответствующих программ и по мере необходимости переключать машину на работу по нужной программе. Так, например, электронная цифровая машина *Диджитак* (США), предназначенная для управления самолетом, может сначала включаться для решения навигационных задач, а затем для решения задач бомбометания.

Необходимым условием применения электронных цифровых машин для автоматического управления каким-либо процессом является наличие полного математического описания данного процесса (наличие алгоритма процесса), т. е. представление его в виде совокупности вполне определенных однозначных правил, определяющих поведение системы в каждом возможном положении. Для обеспечения оптимального регулирования в программу работы машины, помимо указанных основных правил, характеризующих поведение системы в каждом конкретном случае, должны быть введены некоторые общие правила изменения основных правил в зависимости от характера внешних условий и параметров объекта, т. е. должна быть задана некоторая «линия поведения» системы при изменении обстановки.

Следует заметить, что практическое применение цифровых машин для целей автоматического управления встречает иногда серьезные трудности, несмотря на принципиальную возможность такого использования. Эти трудности бывают в основном двух видов: во-первых, применение машин для полностью автоматического управления требует часто коренной переделки существующей схемы работы данной системы или технологического процесса производства и, во-вторых, иногда бывает так, что требуемая для осуществления каких-либо логических функций управления цифровая машина оказывается достаточно сложной в конструктивном отношении, в то время как человек-оператор выполняет эти функции сравнительно легко. Поэтому целесообразно при использовании цифровых машин в автоматических системах стремиться обеспечить рациональное сочетание функций человека и машины, передавая машине наиболее трудоемкую и однообразную часть работы, требующую высокой скорости выполнения и большого объема памяти.

В настоящее время электронные цифровые машины получают широкое практическое применение в машиностроении для автоматического управления металлорежущими станками, в химической, нефтяной и металлургической промышленности для автоматизации процессов выплавки стали, доменного производства, процессов переработки нефти, процессов получения различных химических продуктов и в других областях промышленности.

# ГЛАВА I
# АРИФМЕТИЧЕСКИЕ ОСНОВЫ ЭЛЕКТРОННЫХ ЦИФРОВЫХ МАШИН

## § 4. Позиционные системы счисления

Совокупность приемов наименования и записи чисел называется *счислением.* Читателю должны быть знакомы, по крайней мере, две системы счисления: римская и десятичная позиционная.

**1. Римская система счисления.** В римской системе счисления запись различных целых чисел производится с помощью цифр

I, V, X, L, C, D, Ми д., обозначающих числа один, пять, десять, пятьдесят, сто, пятьсот, тысяча и д. Несколько стоящих подряд одинаковых цифр изображают сумму чисел, обозначаемых этими цифрами. Например, II = I+I — два; XXX = X + X + X — тридцать. Пара цифр, в которой младшая цифра (обозначающая меньшее число) стоит слева от старшей (обозначающей большее число), изображает разность соответствующих чисел. Например, IV = V — I — четыре; XL = L — X — сорок; CM = M — C — девятьсот. Запись, состоящая из старшей цифры (или старшей группы цифр) и стоящей справа от нее младшей цифры (или группы цифр), изображает сумму чисел, отвечающих этим цифрам (или группам цифр). Например, XI = X + I —одиннадцать; XIV = X(IV) = X + IV — четырнадцать; XLIV = XL + IV — сорок четыре. Запись MCMLVII расшифровывается следующим образом:

$$MCMLVII = M + CM + L + V + II$$

и означает число 1957.

Римская система счисления неудобна в пользовании **и** в настоящее время почти не применяется.

**2. Понятие позиционной системы счисления.** Начнем знакомство с позиционными системами счисления с общепринятой десятичной системы, которая коренным образом отличается от римской. Она использует для записи чисел десять различных знаков-цифр: 0, 1,

2, 3, 4, 5, 6, 7, 8 и 9. Цифры обозначают десять последовательных целых чисел, начиная с нуля и кончая девятью. Число десять изображают уже двумя цифрами «10». Остальные числа записываются в виде последовательности цифр, разделенной запятой на целую и дробную части. Десятичную систему называют *позиционной* потому, что значение каждой цифры изменяется с изменением ее положения (позиции) в этой последовательности. Так, например, в записи 121,12 единица, стоящая слева на первом месте, означает количество сотен; единица, стоящая перед запятой, — количество единиц, а единица, стоящая после запятой, — количество десятых долей, содержащихся в числе. Последовательность цифр 121,12 представляет собой не что иное, как сокращенную запись выражения

$$1 \cdot 10^2 + 2 \cdot 10^1 + 1 \cdot 10^0 + 1 \cdot 10^{-1} + 2 \cdot 10^{-2}.$$

Точно так же

$$809{,}103 = 8 \cdot 10^2 + 0 \cdot 10^1 + 9 \cdot 10^0 + 1 \cdot 10^{-1} + 0 \cdot 10^{-2} + 3 \cdot 10^{-3}.$$

Количество различных цифр, применяемых в позиционной системе счисления, называют ее *основанием*

(основанием десятичной системы счисления служит число десять).

Построение совокупности обозначений для всевозможных чисел в десятичной системе счисления состоит из следующих этапов:

1) выбрав десять знаков (цифр) 0, 1,2, 3, 4, 5, 6, 7, 8, 9, мы дали обозначения десяти целым последовательным числам, начиная с нуля и кончая девятью;

2) введя обозначение «10» для числа десять (основания системы), мы мало прибавили к совокупности уже созданных обозначений, но совершили очень важный шаг в построении системы счисления;

3) теперь с помощью операции возведения в степень вводим обозначения для более обширной группы чисел, полагая:

$$a_{10}a_9 \ldots a_1a_0,\ a_{-1}\ldots a_{-9}a_{-10} = a_{10}10^{10} + a_9 10^9 + \ldots + a_1 10^1 + a_0 10^0 + a_{-1}10^{-1} + \ldots + a_{-9}10^{-9} + a_{-10}10^{-10} \qquad (I.1)$$

(здесь каждая из букв $a_{10}$, $a_9$,..., $a_0$, $a_{-1}$,..., $a_{-9}$, $a_{-10}$ может быть заменена любой из десяти основных цифр системы). Этот прием дает нам, в частности, обозначения для всех целых чисел, начиная с нуля (когда $a_{10} = a_9 = \ldots = a_1 = a_0 = a_{-1} = \ldots = a_{-9} = a_{-10} = 0$) и кончая числом 99999999999 (когда $a_{10} = a_9 = \ldots = a_1 = a_0 = 9$; $a_{-1} = \ldots = a_{-9} = a_{-10} = 0$);

4) применяя уже получившие обозначения целые числа в качестве показателей степеней, подобно тому как это было сделано в формуле (1.1) с числами 0, 1, ..., 10, мы еще более расширяем круг чисел, имеющих обозначения; этот прием может быть повторен сколько угодно раз.

Из приведенного описания приемов построения обозначений для чисел совершенно ясно, что среди целых чисел, получающих обозначения, есть сколь угодно большие. Нетрудно доказать, что *нет ни одного целого числа, остающегося вне построенной системы обозначений.*

Для этого докажем сначала, что если целое число $M$ имеет обозначение, то и число $M + 1$ тоже имеет обозначение.

Пусть обозначено $a_k\ a_{k-1}\ldots a_2\ a_1\ a_0$, то есть

$$M = a_k 10^k + a_{k-1}10^{k-1} + \ldots + a_1 10^1 + a_0 10^0 .$$

Рассмотрим два случая.

1. Последняя цифра изображения числа $M$ меньше, чем 9:
   $a_0 < 9$.
   Тогда
   $$a_0 + 1 = a'_0,$$
где $a'_0$ — число, обозначаемое одной цифрой. Следовательно,
$$M + 1 = a_k 10^k + a_{k-1}10^{k-1} + \ldots + a_1 10^1 + a_0 10^0 + 1 = a_k 10^k + a_{k-1}10^{k-1} + \ldots + a_1 10^1 + a'_0 10^0.$$

Это убеждает нас в том, что если $a_0 < 9$ для числа $M$, то число $M + 1$ имеет в десятичной системе счисления обозначение $a_k\ a_{k-1}\ldots a_2\ a_1\ a'_0$ .

2. Последняя цифра в изображении числа $M$ является девяткой:
   $a_0 = 9$.
   Для большей общности можно считать, что
   $$a_0 = a_1 = \ldots = a_{l-1} = 9, \quad a_l < 9, \text{ где } l - 1 \le k.$$
   Тогда

$$M+1 = a_k 10^k + \ldots + a_l 10^l + 9 \cdot 10^{l-1} + \ldots + 9 \cdot 10^1 + 9 \cdot 10^0 + 1 = a_k 10^k + \ldots + a_l 10^l + 9 \cdot 10^{l-1} + \ldots + 9 \cdot 10^1 + 10 + 0 \cdot 10^0 =$$
$$= a_k 10^k + \ldots + a_l 10^l + 9 \cdot 10^{l-1} + \ldots + 9 \cdot 10^2 + 10^2 + 0 \cdot 10^1 + 0 \cdot 10^0 = \ldots$$
$$\ldots = a_k 10^k + \ldots + a'_l 10^l + 0 \cdot 10^{l-1} + \ldots + 0 \cdot 10^1 + 0 \cdot 10^0,$$

где $a_l + 1 = a'_l$ — число, обозначаемое одной цифрой.

Следовательно, и во втором случае число $M + 1$ имеет обозначение: $a_k a_{k+1} \ldots a'_l \ldots 00$.

Пусть теперь $N$ — произвольное целое число. Прибавляя единицу сперва к числу 1, потом — к полученной первой сумме, затем к полученной второй сумме и т. д., мы получим в конце концов сумму, равную $N$. Так как число 1 имеет для себя обозначения, то и каждая из последовательно полученных сумм, а также и само число $N$ тоже имеют обозначения, что и требовалось доказать.

Десятичная система не является единственной возможной позиционной системой счисления.

В позиционной системе счисления с основанием $p$ используется $p$ различных между собой цифр, обозначающих последовательные целые числа, начиная с нуля и кончая числом $p$— 1. Остальные числа записываются в виде последовательностей цифр, в которых целая часть отделена от дробной части запятой и каждая цифра имеет значение в $p$ раз большее, чем та же цифра на предыдущем (ближайшем слева) месте.

Если буквы $a_n$, $a_{n-1}$, ..., $a_1$, $a_0$, $a_{-1}$, $a_{-2}$, ..., $a_{-m}$ обозначают цифры $p$-ичной системы счисления, то последовательность цифр

$$a_n, a_{n-1}, \ldots, a_1, a_0, a_{-1}\ a_{-2}, \ldots, a_{-m}$$

обозначает число

$$a_n p^n + a_{n-1}p^{n-1} + \ldots + a_1 p^1 + a_0 p^0 + a_{n-1}p^{n-1} + a_{n-2}p^{n-2} + \ldots + a_{-n}\ p^{-n}.$$

Обычно в качестве двух младших цифр (соответствующих числам нуль и один) во всех позиционных системах

счисления используют знаки 0 и 1. При этом основание системы счисления *p* записывается в виде последовательности цифр 10.

Все рассуждения, проведенные выше для десятичной системы счисления, могут быть проведены аналогичным способом для позиционной *p*-ичной системы счисления в общем виде.

**3. Восьмеричная позиционная система счисления.** Принимая за основание системы число восемь, мы получим восьмеричную систему. В этой системе счисления для записи всевозможных чисел применяют восемь различных цифр: 0, 1,2, 3, 4, 5, 6, 7, обозначающих целые последовательные числа, начиная с нуля и кончая семью. Число восемь записывают уже двумя цифрами в виде «10». Остальные числа представляют в виде последовательностей цифр. Число двести пятнадцать, которое в десятичной системе имеет начертание 215, в восьмеричной системе будет записано так:

$327 = 3 \cdot 10^2 + 2 \cdot 10^1 + 7 \cdot 10^0$ (здесь 10 означает восемь).

Для нас более привычна десятичная запись чисел. Поэтому, желая проверить правильность восьмеричного изображения числа двести пятнадцать, мы перепишем правую часть последнего равенства в десятичной системе. Получим:

$3 \cdot 8^2 + 2 \cdot 8^1 + 7 \cdot 8^0 = 3 \cdot 64 + 2 \cdot 8 + 7 = 192 + 16 + 7 = 215.$

При подготовке задач для решения на многих электронных цифровых машинах применяется запись чисел в восьмеричной системе.

Таблица сложения восьмеричных чисел (см. таблицу 1) по своему объему близка к десятичной таблице сложения. Правило пользования этой таблицей легко понять из следующего примера: чтобы найти сумму чисел 4 и 6, находим столбец таблицы, вверху которого стоит цифра 4, и строку ее, в левой клетке которой стоит 6. На пересечении столбца и строки читаем сумму 12 (напомним, что в восьмеричной записи 12 означает число десять).

Т а б л и ц а 1. Восьмеричная таблица сложения

| + | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 10 |
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 10 | 11 |
| 2 | 2 | 3 | 4 | 5 | 6 | 7 | 10 | 11 | 12 |
| 3 | 3 | 4 | 5 | 6 | 7 | 10 | 11 | 12 | 13 |
| 4 | 4 | 5 | 6 | 7 | 10 | 11 | 12 | 13 | 14 |
| 5 | 5 | 6 | 7 | 10 | 11 | 12 | 13 | 14 | 15 |
| 6 | 6 | 7 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 7 | 7 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 10 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 20 |

Этой же таблицей пользуются и как таблицей вычитания. Например, чтобы вычесть 5 из 11, находим вначале диагональ таблицы, в клетках которой стоят числа 11. Затем, идя вдоль строки таблицы, начинающейся с числа 5, находим клетку, лежащую на пересечении этой строки и найденной диагонали. Вверху столбца, проходящего через клетку пересечения, читаем разность 4.

С помощью таблицы сложение и вычитание восьмеричных чисел выполняются по тем же правилам, по которым мы производим эти действия в десятичной системе.

Пример 1.

| *Сложение:* | *Вычитание:* |
|---|---|
| 327,71102 | 11076,01 |
| + 35,67735 | − 705,62 |
| 365,61037 | 10170,17 |

Восьмеричная таблица умножения может быть представлена в таком виде, как на таблице 2. Правило получения произведения двух однозначных чисел по этой таблице аналогично правилу получения суммы по таблице сложения.

Т а б л и ц а 2. Восьмеричная таблица умножения

| X | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 10 |
| 2 | 0 | 2 | 4 | 6 | 10 | 12 | 14 | 16 | 20 |
| 3 | 0 | 3 | 6 | 11 | 14 | 17 | 22 | 25 | 30 |
| 4 | 0 | 4 | 10 | 14 | 20 | 24 | 30 | 34 | 40 |
| 5 | 0 | 5 | 12 | 17 | 24 | 31 | 36 | 43 | 50 |
| 6 | 0 | 6 | 14 | 22 | 30 | 36 | 44 | 52 | 60 |
| 7 | 0 | 7 | 16 | 25 | 34 | 43 | 52 | 61 | 70 |
| 10 | 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 100 |

С помощью таблицы умножения, пользуясь теми же правилами, которые применяются в десятичной системе счисления, производятся умножение и деление восьмеричных чисел.

Пример 2.

| *Умножение:* | *Деление:* |
|---|---|
| 173,261 | 336656,615 \| 1635 |
| x 16,35 | 1635 \| 173,261 |
| 1150565 | 15315 |
| 562023 | 14513 |
| 1344046 | 6026 |
| 173261 | 5327 |
| 3366,56615 | 4776 |
| | 3472 |
| | 13041 |
| | 12656 |
| | 1635 |
| | 1635 |
| | 0000 |

**4. Двоичная позиционная система счисления.** Меньше всего различных цифр требует двоичная система счисления — всего лишь две цифры: 0 и 1 (обозначающие целые числа нуль и единицу). Основание этой системы

— два — записывают уже двумя цифрами — «10». Целые числа, начиная с трех и кончая семью, представляются так: 11, 100, 101, 110, 111. Число двести пятнадцать в двоичной системе будет выглядеть так:

$11010111 = 1 \cdot 10^{111} + 1 \cdot 10^{110} + 0 \cdot 10^{101} + 1 \cdot 10^{100} + 0 \cdot 10^{11} + 1 \cdot 10^{10} + 1 \cdot 10^{1} + 1 \cdot 10^{\circ}$ (здесь 10 означает число два).

Показатели степени записаны тоже в двоичной системе счисления. Для проверки правильности двоичной записи перепишем правую часть последнего равенства в десятичной системе. Получим:

$$1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 128 + 64 + 16 + 4 + 2 + 1 = 215.$$

Таблицы сложения, вычитания и умножения двоичных чисел чрезвычайно просты. Каждая из них состоит всего из четырех строчек (см. таблицы 3, 4 и 5). С помощью этих таблиц сложение,

Таблица 3

Двоичная таблица сложения

| |
|---|
| 0 + 0 = 0 |
| 0 + 1= 1 |
| 1 + 0= 1 |
| 1 + 1 = 10 |

Таблица 4.

Двоичная таблица вычитания

| |
|---|
| 0 - 0 = 0 |
| 1 - 0 = 1 |
| 1 - 1 = 0 |
| 10 - 1 = 1 |

Таблица 5.

Двоичная таблица умножения

| |
|---|
| 0 • 0 = 0 |
| 0 • 1= 0 |
| 1 • 0 = 0 |
| 1 • 1 = 1 |

вычитание, умножение и деление двоичных чисел выполняются по тем же правилам, по которым мы привыкли складывать, вычитать, умножать и делить десятичные числа.

Пример 3.

*Сложение:*

```
  1100111,011
+   10011,111
  1111011,010
```

*Вычитание:*

```
  10110,1101
- 10001,1111
    100,1110
```

*Умножение:*

```
   1100111,1101
 x      11,011
   11001111101
+ 11001111101
  11001111101
 101011110,0101111
```

*Деление:*

```
11011101101 | 1001
1001        | 11000101
1001
1001
1011
1001
1001
1001
0000
```

**5. Другие позиционные системы счисления.** Если основание системы счисления больше десяти, то общепринятых (арабских) цифр уже не хватит для записи чисел. Придется ввести новые цифры.

Например, в случае шестнадцатеричной системы счисления это можно сделать так:

| | | | |
|---|---|---|---|
| 0 — нуль | 4 — четыре | 8 — восемь | $\bar{2}$ — двенадцать |
| 1 — один | 5 — пять | 9 — девять | $\bar{3}$ — тринадцать |
| 2 — два | 6 — шесть | $\bar{0}$ — десять | $\bar{4}$ — четырнадцать |
| 3 — три | 7 — семь | $\bar{1}$ — одиннадцать | $\bar{5}$ — пятнадцать |

Таблица 6. Запись чисел в различных позиционных системах счисления

| Десятичная система | Двоичная система | Троичная система | Пятеричная система | Восьмеричная система | Шестнадцатеричная система |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 10 | 2 | 2 | 2 | 2 |
| 3 | 11 | 10 | 3 | 3 | 3 |
| 4 | 100 | 11 | 4 | 4 | 4 |
| 5 | 101 | 12 | 10 | 5 | 5 |
| 6 | 110 | 20 | 11 | 6 | 6 |
| 7 | 111 | 21 | 12 | 7 | 7 |
| 8 | 1000 | 22 | 13 | 10 | 8 |
| 9 | 1001 | 100 | 14 | 11 | 9 |
| 10 | 1010 | 101 | 20 | 12 | $\bar{0}$ |
| 11 | 1011 | 102 | 21 | 13 | $\bar{1}$ |
| 12 | 1100 | 110 | 22 | 14 | $\bar{2}$ |

| Десятичная система | Двоичная система | Троичная система | Пятеричная система | Восьмеричная система | Шестнадцатеричная система |
|---|---|---|---|---|---|
| 13 | 1101 | 111 | 23 | 15 | $\bar{3}$ |
| 14 | 1110 | 112 | 24 | 16 | $\bar{4}$ |
| 15 | 1111 | 120 | 30 | 17 | $\bar{5}$ |
| 16 | 10000 | 121 | 31 | 20 | 10 |
| 17 | 10001 | 122 | 32 | 21 | 11 |
| 18 | 10010 | 200 | 33 | 22 | 12 |
| 19 | 10011 | 201 | 34 | 23 | 13 |
| 20 | 10100 | 202 | 40 | 24 | 14 |

Число двести пятнадцать, которое мы записывали уже в восьмеричной и двоичной системах, в шестнадцатеричной системе счисления будет выглядеть так:

$$\bar{3}\,7 = \bar{3} \cdot 10^1 + 7 \cdot 10^0.$$

Для проверки правильности записи перепишем правую часть последнего равенства в привычной для нас десятичной системе

$$13 \cdot 16^1 + 7 \cdot 16^0 = 13 \cdot 16 + 7 = 208 + 7 = 215.$$

В таблице 6 показана запись первых двадцати одного целых чисел в различных системах счисления.

## § 5. Перевод чисел из одной позиционной системы счисления в другую

**1. Перевод целых чисел.** Пусть в $p$-ичной системе счисления задано целое число $N$. Пусть $q$ — основание другой системы счисления, записанное в исходной $p$-ичной системе счисления.

Предположим, что изображение числа $N$ в $q$-ичной системе счисления найдено и имеет такой вид:

$$N = a_k 10^k + a_{k-1} 10^{k-1} + \ldots + a_1 10 + a_0, \qquad (I.2)$$

где $a_k$, $a_{k-1},\ldots$, $a_0$ — цифры $q$-ичной системы, а 10 — основание этой системы, т. е. $q$.

Заменим в правой части равенства (I. 2) числа $a_k$, $a_{k-1},\ldots$, $a_0$ и 10 их $p$-ичными изображениями $a^*_k$, $a^*_{k-1},\ldots$, $a^*_0$ и $q$. Получим:

$$N = a^*_k q^k + a^*_{k-1} q^{k-1} + \ldots + a^*_1 q + a^*_0. \qquad (I.3)$$

Деля обе части равенства (I. 3) на $q$, имеем:

$$\frac{N}{q} = a_k q^{k-1} * + a_{k-1} q^{k-1} * + \ldots + a_1 * + \frac{a^*_0}{q}, \qquad (I.4)$$

где $\frac{a^*_0}{q}$ представляет собой правильную дробь.

Из равенства (I. 4) видно, что при делении числа $N$ на $q$ остаток равен $a^*_0$, а частным будет

$$N_1 = a^*_k q^{k-1} + a^*_{k-1} q^{k-2} + \ldots + a^*_1.$$

Если теперь частное $N_1$ разделить на $q$, то получим **в** остатке $a^*_1$, а в новом частном

$$N_2 = a^*_k q^{k-2} + a^*_{k-1} q^{k-3} + \ldots + a^*_2.$$

Выполняя этот процесс $k$ раз, мы последовательно найдем все числа $a^*_0$, $a^*_1,\ldots$, $a_{k-1}$, причем последнее частное будет иметь вид

$$N_k = a^*_k.$$

Таким образом, путем последовательного деления числа $N$ и его частных на $q$ получим в виде остатков деления $p$-ичные записи $q$-ичных цифр (начиная с младшей), нужных для изображения числа $N$. Последовательное деление производят до тех пор, пока не получится частное, меньшее чем $q$. Это последнее частное дает нам старшую $q$-ичную цифру числа $N$. Деление выполняют **в** исходной системе счисления.

Проиллюстрируем на примерах найденный нами прием перевода целых чисел из одной системы счисления в другую.

П р и м е р 1. Пусть дано десятичное число 191. Переведем его в двоичную систему счисления

```
191 | 2
18   95 | 2
 11  8    47 | 2
 10  15   4    23 | 2
  1  14    7    2    11 | 2
        1   6    3    10   5  | 2
            1    2     1   4    2  | 2
                 1           1    2    1
                                  0
```

Двоичная запись десятичного числа 191 имеет следующий вид: 10111111.

П р и м е р 2. Переведем то же число 191 в восьмеричную систему счисления

```
191 | 8
16   23  | 8
 31  16    2
 24   7
  7
```

Восьмеричная запись десятичного числа 191 имеет такой вид: 277.

П р и м е р 3. Переведем то же число 191 в шестнадцатеричную систему

```
191 | 16
16   11
31
16
15
```

Для числа 11 в шестнадцатеричной системе мы ввели цифру $\bar{1}$, а для числа 15—цифру $\bar{5}$. Таким образом, шестнадцатеричная запись числа сто девяносто один: $\bar{1}\,\bar{5}$.

**2. Перевод дробей.** Пусть теперь $D$ — правильная $p$-ичная дробь. Предположим, что ее $q$-ичная запись нами найдена:

$$D = a_{-1}10^{-1} + a_{-2}10^{-2} + a_{-3}10^{-3} + \ldots \qquad (I.5)$$

где $a_{-1}$, $a_{-2}$, $a_{-3}$,...— $q$-ичные цифры, а 10 — основание системы счисления, т. е. $q$.

Заменяя входящие в правую часть равенства (I. 5) числа $a_{-1}$, $a_{-2}$, $a_{-3}$, ... и 10 их $p$-ичными изображениями $a^*_{-1}$, $a^*_{-2}$, $a^*_{-3}$,... и $q$, получим:

$$D = a^*_{-1}\,q^{-1} + a_{-2}\,q^{-2} + a^*_{-3}\,q^{-3} + \ldots \qquad (I.6)$$

Умножая обе части равенства (I.6) на $q$, имеем:

$$Dq = a^*_{-1} + a_{-2}\,q^{-1} + a^*_{-3}\,q^{-2} + \ldots. \qquad (I.7)$$

Целая часть числа (I.7) равна $a^*_{-1}$, а его дробной частью является

$$D_1 = a^*_{-2}\,q^{-1} + a^*_{-3}\,q^{-2} + \ldots$$

Умножая новую дробь $D_1$ на $q$, мы получим число, целая часть которого дает нам $a^*_{-2}$, дробная часть имеет вид

$$D_2 = a^*_{-3}\,q^{-1} + \ldots$$

Повторяя описанный процесс нужное нам количество раз, мы найдем одну за другой (в $p$-ичной записи) $q$-ичные цифры для изображения числа $D$.

Таким образом, путем последовательного умножения числа $D$ и дробных частей получающихся произведений на $q$ получим в виде целых частей этих произведений $p$-ичные записи $q$-ичных цифр, нужных для изображения правильной дроби $D$. Умножение выполняют в исходной системе счисления.

Проиллюстрируем на примерах найденный способ перевода правильных дробей из одной системы счисления в другую.

П р и м е р 4. Дана десятичная дробь 0,6875. Требуется найти ее двоичную запись

| | | |
|---|---|---|
| 0, | 6875 | |
| | ×2 | |
| 1, | 3750 | |
| | ×2 | |
| 0, | 7500 | |
| | ×2 | |
| 1, | 5000 | |
| | ×2 | |
| 1, | 0000 | |

П р и м е р 5. Ту же десятичную дробь 0,6875 требуется перевести в восьмеричную систему счисления

| | | |
|---|---|---|
| 0, | 6875 | |
| | ×8 | |
| 5, | 5000 | |
| | ×8 | |
| 4, | 0000 | |

П р и м е р 6. Переведем ту же дробь 0,6875 в шестнадцатеричную систему счисления

| | | |
|---|---|---|
| 0, | 6875 | |
| | ×16 | |
| 4, | 1250 | |
| 6, | 875 | |
| 11, | 0000 | |

Двоичная запись нашей дроби имеет следующий вид: 0,1011.

Восьмеричная запись нашей дроби имеет вид 0,54.

Шестнадцатеричная запись нашей дроби будет следующей; $0,\bar{1}$ ($\bar{1}$ — шестнадцатеричная цифра, обозначающая число одиннадцать).

Если число $N$ — неправильная дробь, то отдельно переводят целую часть $N$ и отдельно его дробную часть. Второй результат приписывают к первому. Например, десятичное число 191,6875 в двоичной системе будет иметь вид 10111111,1011. Его восьмеричная запись будет 277,54, а шестнадцатеричная 15,1 (для получения этих результатов использованы предыдущие примеры).

**3. Перевод из восьмеричной системы счисления в двоичную и обратно.** Рассмотрим такие две системы счисления, у которых основание одной является целой степенью основания другой. В этом случае перевод чисел из одной системы счисления в другую становится особенно простым.

Если $p$ и $q$ — соответственно основания систем счисления, причем $p = q^b$ ($p$, $q$, $k$ — целые числа), то для перевода числа из $p$-ичной системы в $q$-ичную каждую цифру $p$-ичной системы счисления заменим ее $k$-значным изображением в $q$-ичной системе счисления. Нам придется в дальнейшем встречаться с переводом чисел из восьмеричной системы счисления в двоичную и обратно ($8 = 2^3$). Поэтому ограничимся выводом правила перевода лишь для этого случая.

Пусть буквы $b_n$, $b_{n-1}$, … $b_0$, $b_{-1}$, ..., $b_{-m}$ означают собой цифры восьмеричной системы счисления (т. е. под каждой из этих букв понимают любую цифру от 0 до 7). Для удобства применим десятичную запись для чисел восемь и два, а также для показателей степеней.

Рассмотрим восьмеричное число

$$N=b_nb_{n-1}...b_0 = b_n \cdot 8^n + b_{n-1} \cdot 8^{n-1} + ... + b_1 \cdot 8^1 + b_0. \qquad (I.8)$$

Предположим, что нам удалось найти его двоичную запись

$$N' = a_ka_{k-1} ...a_0 = a_k2^k + a_{k-1}\,2^{k-1} + ... + a_32^3 + a_22^2 + a_12^1 + a_0. \qquad (I.9)$$

Записи (I.8) и (I.9) изображают одно и то же число, поэтому

$$N=N'. \qquad (I.10)$$

При делении обеих величин $N$ и $N'$ на 8 (с учетом того, что $8 = 2^3$) получим одинаковые остатки и одинаковые частные, т. е. будем иметь:

$$b_0 = a_22^2 + a_12^1 + a_0 = a_2a_1a_0; \qquad (I.11)$$

$$b_n \cdot 8^{n-1} + b_{n-1} \cdot 8^{n-2} + ... + b_2 \cdot 8^1 + b_1 = a_k \cdot 2^{k-3} + a_{k-1} \cdot 2^{k-4} + ... + a_5 \cdot 2^2 + a_4 \cdot 2^1 + a_3. \qquad (I.12)$$

Таким же путем из последнего равенства получим:

$$b_1 = a_5 \cdot 2^2 + a_4 \cdot 2^1 + a_3 = a_5a_4a_3$$

и т. д., т. е. для перевода восьмеричного числа в двоичную систему счисления нужно каждую восьмеричную цифру заменить равным ей трехзначным двоичным числом. Это правило сохраняет силу и для перевода восьмеричных дробей в двоичные. Действительно, пусть теперь

$$N=b_n\,b_{n-1}...b_0,\, b_{-1}b_{-2} \ldots b_{-m}.$$

Число $N \cdot 8^m$ будет целым. После перевода этого числа в двоичную систему полученное двоичное число остается разделить на $8^m = 2^{3m}$, т. е. поставить запятую так, чтобы справа от нее стояло ($3m$) цифр.

П р и м е р 7. Восьмеричное число 70,271 равно двоичному числу

111 000, 010 111 001 = 111 000, 010 111 001.

П р и м е р 8. Восьмеричное число 26,036 равно двоичному числу

010 110, 000 011 110=10110, 00001111.

Теперь легко сформулировать правило также для перевода двоичных чисел в восьмеричные. Нужно, начиная

от запятой, влево и вправо от нее разбить набор двоичных цифр, изображающий число, на тройки цифр; каждое полученное трехзначное число отдельно перевести в восьмеричную систему счисления (это нетрудно); если правая и левая группы цифр не будут полными тройками, их дополняют соответственно справа и слева нулями.

П р и м е р 9. Дано двоичное число 11,001111. Разбиваем набор цифр на группы по три, левую группу дополним нулем слева. Имеем (011), (001), (111). Теперь легко получаем восьмеричную запись нашего числа: 3,17.

П р и м е р 10. Двоичное число 1001101,0111 =(001) (001) (101), (011) (100)= = 115,34 (восьмеричное число).

Заметим, что перевод десятичных чисел в двоичные на практике никогда не выполняют непосредственно так, как было показано в вышеприведенных примерах. Обычно десятичные числа сперва преобразуют в восьмеричные, а затем от восьмеричной записи переходят уже к двоичной. Этот, казалось бы, более сложный путь в действительности является более коротким.

П р и м е р 11. Требуется десятичное число 725,9375 перевести в двоичную систему счисления.
Перевод числа в восьмеричную систему:

```
725 | 8          0, | 9375
72  | 90 | 8        |  ×8
 5  |  8 | 11 | 8  ---------
    | 10 |  8 |  1  7, | 5000
    |  8 |  3 |       |  ×8
    |  2 |            ---------
                      4, | 0000
```

Восьмеричная запись нашего числа: 1325,74.

Перевод восьмеричного числа в двоичное

001 011 010 101, 111 100= 1011010101,1111.

**4. Двоично-десятичная запись чисел.** При переводе на цифровой вычислительной машине десятичных чисел в двоичные промежуточной является двоично-десятичная запись чисел.
Двоично-десятичная запись чисел состоит в следующем. Каждая цифра десятичного числа записывается в виде четырехразрядного двоичного числа. Четверка двоичных цифр, изображающая десятичную цифру, называется тетрадой. Таким образом, двоично-десятичная запись числа отличается от его десятичной записи тем, что для изображения каждой цифры применяется не один знак, а четыре знака.

П р и м е р 12. Десятичное число 25 в двоично-десятичной системе счисления будет представлено так:

0010 0101.

П р и м е р 13. Десятичное число 893,2 изобразится в двоично-десятичной системе следующим образом:

1000 1001 0011, 0010.

П р и м е р 14. Перевод двоично-десятичного числа в десятичное столь же прост, например:

1001 0001 0101,1000 1001,0001=915,891.

Перевод двоично-десятичного числа в десятичное состоит в следующем: двоично-десятичное число разбивают на тетрады (от запятой вправо и влево) и каждую тетраду заменяют отвечающей ей десятичной цифрой.

Во внешних устройствах машины десятичные числа преобразуются в двоично-десятичные. Двоично-десятичные числа по специальной программе уже самой машиной переводятся в двоичные. Существуют машины, в которых все вычисления осуществляются в двоично-десятичной системе. При этом арифметические действия над двоично-десятичными числами выполняются по специальным правилам, которые мы здесь не приводим.

## § 6. Ввод чисел в машину и запись их в памяти

**1. Запись чисел на перфокартах.** Большинство современных электронных цифровых машин не имеет фоточитающих устройств и потому не «воспринимает» чисел, записанных на бланке. Для ввода чисел в память машины их сперва наносят на перфокарты в виде системы отверстий.

Колода перфокарт с нанесенными на них числами закладывается в приемник вводного устройства. Перфокарты захватываются машиной по одной и прощупываются системой контактов. В тех местах, где перфокарта имеет отверстия, контакты замыкаются и посылают в машину электрические сигналы, под воздействием которых производится запись чисел в памяти машины.

В некоторых машинах применяются не перфокарты, а тонкие бумажные или целлулоидные перфоленты, на которых числа тоже изображены системами отверстий. При вводе чисел в машину движущаяся лента прощупывается системой контактов аналогично тому, как это делается с перфокартами. В отечественной цифровой электронной машине *Урал* применяется зачерненная целлулоидная перфолента (изготовленная на основе обычной кинопленки). При вводе чисел в машину перфолента движется между источником света и системой фотоэлементов.

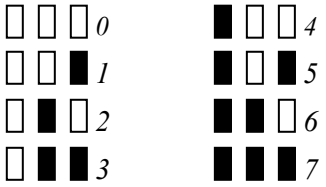
Световые лучи, попадающие на фотоэлементы при прохождении над ними отверстий перфоленты, вызывают возникновение электрических сигналов.

В некоторых машинах для ввода чисел в память применяется магнитная лента, на которой числа изображаются системами намагниченных участков. Считывание чисел с магнитной ленты производится специальной считывающей головкой, подобной считывающей головке обычного магнитофона.

В машине *Стрела* применяются перфокарты. На каждой перфокарте записывают двенадцать чисел, каждое в отдельной строке. Каждая строка разбита на некоторое число участков, равное количеству разрядов в ячейке памяти машины. Наличие отверстия на каком-либо участке означает, что в соответствующем разряде ячейки должна быть записана цифра 1. Отсутствие отверстия значит, что в соответствующем разряде ячейки должен стоять нуль. Таким образом, числа на перфокарте записываются в виде последовательности нулей и единиц.

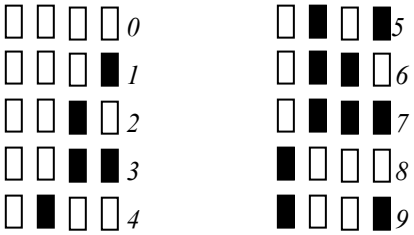
Как мы узнаем в дальнейшем, в машину приходится вводить числа, которые на бланке записаны либо в десятичной, либо в восьмеричной позиционной системе счисления.

Для записи каждой восьмеричной цифры отводятся три участка. Восьмеричная цифра изображается в виде трехзначного двоичного числа. Если изображать незачерненным прямоугольником участок, не имеющий отверстия, а черным — участок, на котором пробито отверстие, то восьмеричные цифры будут выглядеть на перфокарте следующим образом:




При таком изображении восьмеричных цифр число оказывается записанным в двоичной системе счисления.

Для записи каждой десятичной цифры в строке перфокарты отведено четыре участка. Каждая десятичная цифра записывается в виде четырехзначного двоичного числа. Десятичные цифры будут выглядеть на перфокарте следующим образом:




Как известно (см. § 5), такое изображение десятичных чисел называется двоично-десятичным.

Итак, восьмеричные числа на перфокарте представлены в виде двоичных чисел, а десятичные числа в виде двоично-десятичных. Перевод восьмеричных чисел в двоичные и десятичных в двоично-десятичные осуществляется весьма просто в процессе пробивки отверстий на перфокартах. Прибор, служащий для пробивки отверстий, называется перфоратором. Он снабжен клавишным устройством. Для перенесения числа с бланка на перфокарту нажимают клавиши, помеченные соответствующими цифрами. При этом в перфораторе набирается система пуансонов.

При нажиме на клавишу для переноса на перфокарту, например, восьмеричной цифры 5 в набор пуансонов включаются пуансоны для пробивки отверстия на двух крайних участках из трех отведенных для изображения восьмеричной цифры и т. п. При нажиме на клавишу для переноса, например, десятичной цифры 6 в набор пуансонов включаются пуансоны, служащие для пробивки отверстий на втором и третьем участках из четырех, отведенных для каждой десятичной цифры, и т. п.

Когда набор пуансонов для нанесения числа составлен, нажимом специальной кнопки осуществляется пробивка всей системы отверстий на перфокарте.

Перенос чисел на перфоленту осуществляется аналогичным образом.

**2. Запись чисел в ячейках памяти машины с фиксированной запятой.** В зависимости от способа представления в них чисел машины делятся на машины с *фиксированной* запятой и машины **с** *плавающей* запятой.

В машинах с фиксированной запятой применяется естественная форма записи чисел: число представляется в виде последовательности цифр, разделенной запятой на целую и дробную части.

Ячейка памяти такой машины состоит из знакового разряда (припишем ему номер нуль) и цифровых разрядов (занумерованных последовательными числами $1, 2, 3, \ldots$). Некоторое постоянное количество цифровых разрядов отведено для хранения целой части числа, остальные цифровые разряды предназначены для изображения его дробной части. Схематическое изображение ячейки с четырьмя разрядами для хранения целой части и восемью

разрядами для хранения дробной части числа *) представлено на рис. 1.

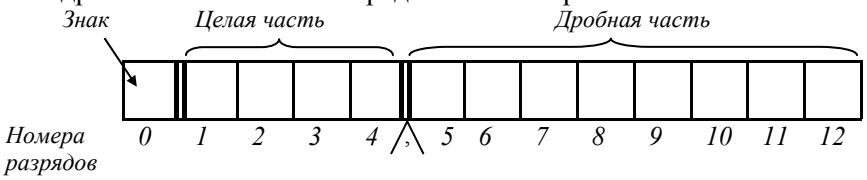



Рис. 1. Схематическое изображение ячейки памяти машины с фиксированной запятой.

В такой ячейке запятая фиксирована после четвертого разряда. Сама запятая в ячейке никак не изображена: просто все устройства машины так сконструированы, что запись чисел в ее ячейках производится в соответствии с присущим данной машине положением запятой. Так, в ячейке, приведенной на рис. 1, целая часть числа будет записана в разряды с номерами 1, .... 4, а дробная часть — в разряды с номерами 5, ..., 12.

Для изображения знака числа в машине приняты следующие обозначения:

$$«+» = 0;$$
$$«-» = 1.$$

Таким образом, в знаковом разряде ячейки стоит нуль, если записанное в ней число положительно, или единица, если ячейка хранит отрицательное число.

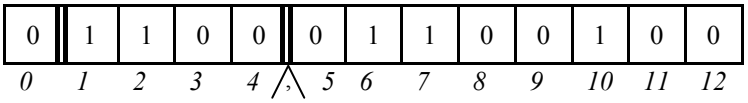



Рис. 2. Схема записи в ячейке памяти машины с фиксированной запятой двоичного числа + 1100,011001.

П р и м е р 1. Требуется записать в ячейке, приведенной на рис. 1, число, имеющее в десятичной системе счисления такой вид:

$$+ 12{,}390625 = +12\frac{25}{64}$$

Двоичное изображение этого числа будет следующим:

+ 1100, 011001.

При записи этого числа в ячейку ее разряды будут заполнены так, как показано на рис. 2.

П р и м е р 2. Десятичное число $- 12{,}390625 = -12\frac{25}{64}$ в двоичной записи будет иметь такой вид:

— 1100, 011001.

Поэтому разряды ячейки, хранящей это число, будут заполнены так, как показано на рис. 3.

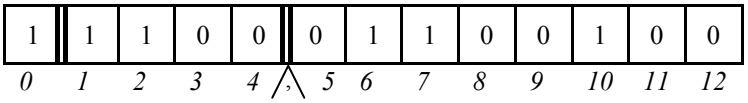



Рис. 3. Схема записи в ячейке памяти машины с фиксированной запятой двоичного числа - 1100,011001.

П р и м е р 3. Десятичное число +7,6 в двоичной системе счисления представится в виде периодической дроби +111, (1001). Из его дробной части в нашей ячейке могут быть помещены только восемь цифр. При этом, если первая из отбрасываемых цифр равна единице, то число округляется (прибавляется единица к младшему из записываемых разрядов). В нашем случае округление происходит так:

+ 111,100110011001 ... = +111,10011001 +0,00000001 = +111,10011010.

•Запись этого числа в ячейке приведена на рис. 4, *а,* а запись числа -111, (1001) — на рис. 4, *б.*

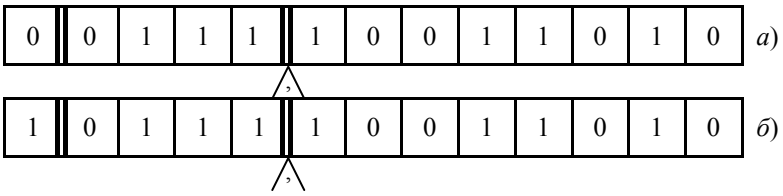



Рис. 4. Схема записи в ячейках памяти машины с фиксированной запятой двоичных чисел:
*а*) +111,(1001); *б*) -111,(1001).

При выполнении арифметических операций могут получаться результаты, целая часть которых содержит больше цифр, чем число разрядов ячейки, отведенных для хранения целой части. При этом переполняется разрядная сетка: одна из цифр целой части оказывается записанной в знаковый разряд ячейки; если количество цифр целой части больше чем на единицу превышает количество соответствующих разрядов ячейки, то, кроме того, часть ее цифр теряется. Например, если в результате какой-либо операции получится число +1010001, 1001, то в нашей ячейке

*) Обычно ячейка памяти машины содержит около сорока цифровых разрядов. Мы ограничиваемся двенадцатью цифровыми разрядами для упрощения и большей обозримости приводимых в дальнейшем рисунков и примеров.

оно запишется так, как показано на рис. 5, т. е. вместо правильного результата в ячейке будет изображено число -1, 1001.

Из приведенного примера видно, что при переполнении разрядной сетки искажается результат вычислений. Применяя машину с фиксированной запятой, нужно всегда помнить о возможности переполнения разрядной сетки. Все величины, входящие в решаемую задачу, нужно заранее умножать на *масштабные коэффициенты,* т. е. на множители, подобранные с таким расчетом, чтобы в процессе вычислений разрядная сетка не переполнялась.

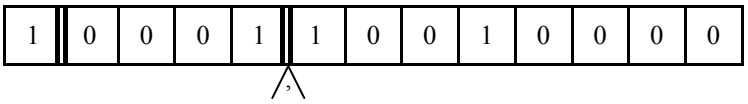



Рис. 5. Переполнение разрядной сетки ячейки при получении в результате арифметической операции числа, целая часть которого содержит больше цифр, чем могут вместить разряды ячейки, отведенные для хранения целой части (вместо числа +1010001,1001 в ячейке записано —1,1001).

Подбор масштабных коэффициентов обычно представляет собой нелегкую работу. В § 36 будет показано, как это делать.

Для облегчения подбора масштабных коэффициентов в большинстве машин запятая фиксирована перед первым цифровым разрядом ячейки (непосредственно после знакового разряда). При этом в ячейке могут быть записаны

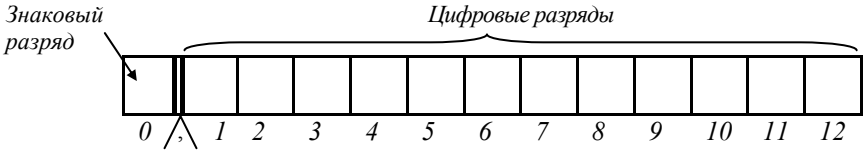



Рис. 6. Схема ячейки памяти машины с запятой, фиксированной после знакового разряда.

только правильные дроби. Превращение всех величин, фигурирующих в решаемой задаче, в правильные дроби достигается подходящим выбором масштабных коэффициентов. Перемножение двух правильных дробей всегда приводит опять к правильной дроби, так что переполнение разрядной сетки в машинах с запятой, фиксированной после знакового разряда, в результате умножения произойти не может.

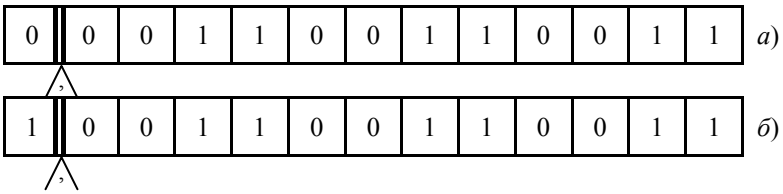



Рис. 7. Схема записи в ячейке памяти машины с запятой, фиксированной после знакового разряда, двоичных чисел: *а*) + 0,(0011); *б*) -0,(0011).

На рис. 6 изображена схема ячейки с запятой, фиксированной после знакового разряда. На рис. 7*, а* показана запись в такую ячейку десятичного числа +0,2, в двоичной записи имеющего вид +0, (0011). На рис. 7, *б* показана запись десятичного числа -0,2.

Диапазон положительных чисел [*)], которые могут быть записаны **в** ячейке памяти машины с фиксированной запятой, сравнительно невелик. Например, в ячейке, приведенной на рис. 1, могут быть изображены числа, начиная с 0,00000001 и кончая 1111,11111111 (или, в десятичной системе, начиная с $\frac{1}{256}$ и кончая $15\frac{255}{256}$). Всякое число по абсолютной величине меньшее, чем 0,000000001 (1/512), в такой ячейке будет представлено в виде нуля (само число 0,000000001= $\frac{1}{512}$ благодаря округлению при записи будет представлено в виде 0,00000001 = 1/256).

Отличные от нуля числа, воспринимаемые машиной как нуль, носят название *машинных нулей.*

**3.Запись чисел в ячейках памяти машины с плавающей запятой.** Применение машин с фиксированной запятой сопровождается серьезным неудобством — необходимостью использования масштабных коэффициентов.

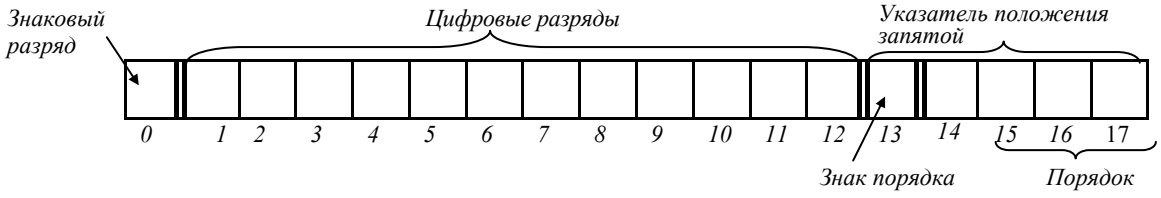



Рис. 8. Схема ячейки памяти машины с плавающей запятой.

[*)] Оценивая диапазон чисел, представимых в машинной ячейке, мы всегда будем говорить о положительных числах.

Эти трудности устранены в машинах с плавающей запятой следующим образом. Каждая ячейка памяти машины с плавающей запятой, кроме знакового и цифровых разрядов, имеет еще некоторое количество дополнительных разрядов, образующих указатель положения запятой (рис. 8).

Чтоб уяснить себе, как изображаются числа в такой ячейке, рассмотрим некоторое двоичное число *N*. Не изменяя его величины, умножим его на $10^0$ (10 — это два)

$$N = N \cdot 10^0. \qquad (I.13)$$

Правую часть равенства (I.13) можно преобразовать так, чтобы абсолютная величина первого сомножителя была меньше единицы. Если $|N|<1$, то никакого преобразования не требуется. Если же $|N| \geq 1$,то переносим запятую в двоичной записи первого сомножителя на соответствующее число разрядов влево, а показатель степени второго сомножителя увеличиваем на столько единиц, на сколько разрядов была перенесена запятая. Теперь равенство (I 13) примет такой вид:

$$N = m \cdot 10^p, \qquad (I.14)$$

где $|m|<1$, а $p$ — целое. Например, при двоичной записи $m$ и $p$

$$N_1 = -0{,}011 = -\,0{,}011 \cdot 10^0;$$
$$N_2 = +1011{,}01 = +\,0{,}101101 \cdot 10^{100},$$

или

$$N_2 = +1011{,}01 = +\,0{,}0101101 \cdot 10^{101},$$

или

$$N_2 = +\,1011{,}01 = +\,0{,}00101101 \cdot 10^{110} \text{ и т. д.}$$

Легко видеть, что при записи числа в виде (I.14) показатель степени второго сомножителя равен номеру разряда первого сомножителя (считая нулевым разряд целых единиц), после которого нужно поместить запятую для того, чтобы первый сомножитель совпал с числом *N*. Таким образом, показатель степени второго сомножителя показывает положение запятой, если число *N* изображать набором цифр дробной части числа *m*.

Если число *N* очень мало и в его двоичной записи между запятой и первой значащей цифрой содержится много нулей, то потребовалось бы большое количество разрядов ячейки памяти машины для представления дробной части множителя *m*. В этом случае целесообразно в формуле (I.13) первый сомножитель увеличить в $10^q$ ($q$ — целое положительное) раз путем переноса запятой на $q$ разрядов вправо, а второй сомножитель уменьшить в то же количество раз, т. е. умножить на $10^{-q}$.При этом в формуле (I.14) получится:

$$p = -\,q,$$

а множитель *m* при подходящем выборе $q$ не будет между запятой и первой значащей цифрой содержать большого количества нулей.

Например, если $N_3 = 0{,}00000000111011$, то можно положить

$$N_3 = 0{,}00000000111011 = 0{,}111011 \cdot 10^{-1000},$$

или

$$N_3 = 0{,}00000000111011 = 0{,}0111011 \cdot 10^{-111}.$$

Множитель *m* называют *мантиссой,* а показатель степени *p — порядком* числа *N*.

В ячейке машины с плавающей запятой запись числа *N* производится так: знак мантиссы записывается в знаковый разряд ячейки, а дробная часть мантиссы размещается в цифровых разрядах (целая часть всегда равна нулю, и поэтому ее запись излишня); разряды указателя положения запятой отведены для порядка. При этом первый разряд указателя служит для хранения знака порядка, а остальные его разряды — для хранения абсолютной величины порядка. Знаки (мантиссы и порядка) по-прежнему изображаются в соответствии с правилом

$$\text{« + »} = 0, \qquad \text{« — »} = 1.$$

Запись в ячейке памяти машины с плавающей запятой числа+ 101,00101 в форме $+0{,}10100101 \cdot 10^{+11}$ приведена

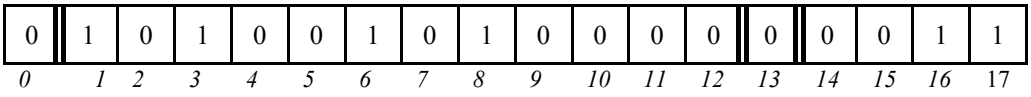



Рис. 9. Схема записи числа + 101,00101 = $+0{,}10100101 \cdot 10^{+11}$ в ячейке памяти машины с плавающей запятой.

на рис. 9, а в форме $+0{,}000010100101 \cdot 10^{+111}$ — на рис. 10. Запись числа *N* в виде (I.14)

$$N = m \cdot 10^p$$

называется *нормализованной* (коротко говорят: число *N нормализовано),* если

$$\frac{1}{10} \le |m| < 1 . \qquad \text{(I.15)}$$

При этом первый цифровой разряд ячейки содержит в себе 1. Если же

$$|m| < \frac{1}{10} , \qquad \text{(I.16)}$$

то говорят, что число $N$ не нормализовано. В случае ненормализованного числа первые несколько цифровых разрядов ячейки заполнены нулями. (Напоминаем, что 10 означает два. Впрочем, не только в двоичной, но и во всякой другой системе счисления число, приведенное к виду (I.14), при выполнении условия (I.15) называют *нормализованным*.)

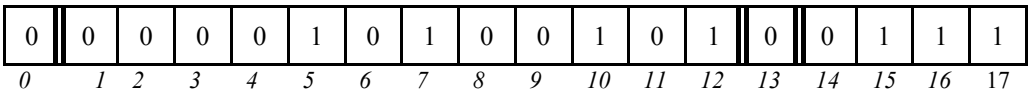



Рис. 10. Схема записи числа +101,00101= +0,000010100101 •$10^{+111}$ в ячейке памяти машины с плавающей запятой.

Например, число +0,1011 • $10^{100}$ — нормализованное, а +0,0001011 • $10^{101}$ — то же число, но не нормализованное.

Если число $N$ представляет собой малую двоичную дробь, то при нормализованной записи его в ячейке порядок окажется отрицательным. Например, число

$$N = +0{,}00000111001101$$

после нормализации имеет вид $N$ = +0,111001101 • $10^{-101}$. Соответствующая запись в ячейке показана на рис. 11.

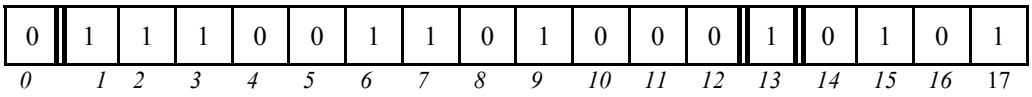



Рис. 11. Схема записи числа +0,00000111001101 = +0,111001101 • $10^{-101}$
в ячейке машины с плавающей запятой.

Считают, что порядок числа совпадает с номером цифрового разряда ячейки, хранящего ту цифру числа, после которой стоит запятая. Чтобы это соображение оставалось справедливым и в случае отрицательного порядка, мысленно отбрасывают знаковый разряд ячейки и дополняют ее условными, воображаемыми цифровыми разрядами, продолжающими ячейку влево и имеющими номера 0; -1; -2; -3; ... Считают, что в этих условных разрядах записаны нули (рис. 12).

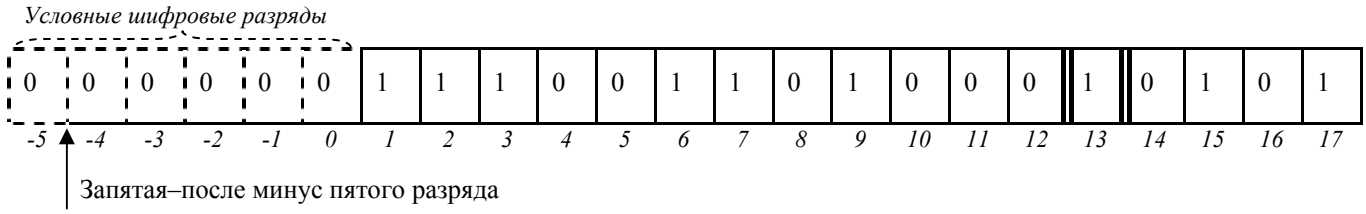



Рис. 12. Так следует представлять себе положение запятой, когда в ячейке машины с плавающей запятой изображено
число с отрицательным порядком.

Название *машина с плавающей запятой* происходит от того, что при записи чисел в ее ячейках запятая помещается (с помощью соответствующей записи в указателе положения запятой) после любого фактического или условного цифрового разряда ячейки.

При записи в ячейку числа в ненормализованном виде количество значащих цифр, изображающих число, может быть значительно меньше, чем при записи нормализованного числа. Например, число, десятичная запись которого -0,09, в нормализованном виде изображается в нашей ячейке двенадцатью двоичными значащими цифрами: -0,101110000101 • $10^{-11}$ (рис. 13).

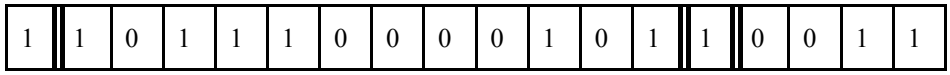



Рис. 13. В ячейке памяти машины с плавающей запятой представлено число -0,09 в нормализованном (двоичном) виде.

То же число в ненормализованном виде -0,00000000101110000101 • $10^{101}$ будет в той же ячейке изображено только четырьмя значащими цифрами (рис. 14).

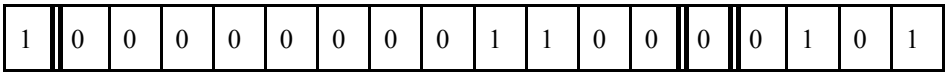



Рис. 14. В ячейке памяти машины с плавающей запятой представлено число -0,09 в ненормализованном (двоичном) виде. Восемь значащих цифр мантиссы утеряны, ее четвертая значащая цифра округлена.

Ясно, что при записи чисел в ячейке машины в ненормализованной форме погрешность в изображении их может быть больше, чем при записи в нормализованной форме. Во многих машинах при выполнении ряда арифметических действий предусмотрена автоматическая нормализация получаемого результата. Она состоит в том, что мантисса ненормализованного числа сдвигается влево на столько разрядов, что ее первая значащая цифра попадает в первый цифровой разряд. Одновременно от порядка числа отнимается число, равное количеству разрядов, на которое произведен сдвиг мантиссы. Содержимое знакового разряда ячейки не изменяется.

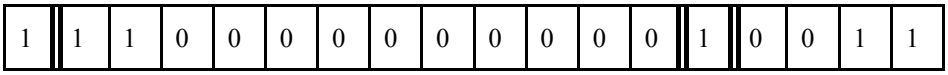



Рис. 15. Результат нормализации содержимого ячейки, изображенной на рис. 14.

На рис. 15 показан результат нормализации содержимого ячейки, изображенной на рис. 14. Как видно из примера, точность записи числа в ячейке при его нормализации не улучшается. Отсюда вытекает, что числа должны записываться в ячейки машины сразу в нормализованном виде.

**4. Диапазон чисел, представимых в ячейках памяти машины.** Нетрудно подсчитать в общем случае границы диапазона чисел, представимых в ячейке машины с фиксированной запятой. Пусть $n$ — количество цифровых разрядов, отведенных для хранения целой части числа, а $v$ — количество разрядов, предназначенных для дробной части. Очевидно, самое большое положительное число, которое можно записать в такой ячейке, равно

$$\underbrace{1\ldots.1}_{n\text{ цифр}},\ \underbrace{11\ldots.11}_{v\text{ цифр}} = \underbrace{10\ldots.0}_{(n+1)\text{цифр}},\underbrace{00\ldots.00}_{v\text{ цифр}} - 0,\underbrace{00\ldots.01}_{v\text{ цифр}} = 2^n - 2^{-v}.$$

Самое малое положительное число, которое может хранить ячейка, равно

$$0\ldots.0,\underbrace{00\ldots.01}_{v\text{ цифр}} = 2^{-v}.$$

Таким образом, диапазон чисел, представимых в ячейке машины с фиксированной запятой, определяется следующим неравенством:

$$2^{-v} \le |N| \le 2^n - 2^{-v}. \qquad (I.17)$$

Диапазон чисел, представимых в ячейках машины с плавающей запятой, значительно шире, чем в ячейках машины с фиксированной запятой (при одинаковом числе цифровых разрядов). Например, в рассматриваемой нами ячейке с двенадцатью цифровыми и четырьмя порядковыми разрядами можно изображать числа, начиная с $0{,}000000000001 \cdot 10^{1111}$ (в десятичной записи $2^{-27}$) и кончая $0{,}111111111111 \cdot 10^{+1111}$ (в десятичной записи 32 760).

Подсчитаем для общего случая границы диапазона чисел, представимых в ячейках машины, с плавающей запятой.

Пусть $\mu$ — число разрядов ячейки, отведенных для абсолютной величины мантиссы, а $\rho$ — число разрядов, предназначенных для абсолютной величины порядка. Очевидно, самым большим из положительных чисел, которое можно записать в такой ячейке, будет:

$$\underbrace{0{,}11\ldots.11}_{\mu\text{ цифр}} \cdot 10^{\overbrace{1\ldots1}^{\rho\text{ цифр}}} = (1 - 2^{-\mu}) \cdot 2^{2^{\rho}-1}$$

Самым малым нормализованным числом, представимым в рассматриваемой ячейке, будет:

$$0{,}10 \qquad 0 \cdot 10^{\overbrace{-1\ldots1}^{\rho\text{ цифр}}} = 2^{-1} \cdot 2^{-2^{\rho}+1} = 2^{-2^{\rho}}.$$

Наименьшее ненормализованное число, представимое в ячейке, будет в $2^{\mu-1}$ раз меньше (за счет того, что наименьшая мантисса ненормализованного числа может быть равна $0{,}00\ldots01 = 2^{-\mu}$).

Однако при решении задач ненормализованные числа обычно не применяются, потому что, как читатель увидит ниже, при выполнении арифметических действий над малыми ненормализованными числами они могут превращаться в машинные нули.

Поэтому диапазон чисел, представимых в машине с плавающей запятой, обычно характеризуют неравенством

$$2^{-2^{\rho}} \le |N| \le (1—2^{-\mu})\cdot 2^{2^{\rho}-1}. \qquad (I.18)$$

Чтобы наглядно себе представить, насколько для машины с плавающей запятой диапазон представимых чисел шире, чем для машины с фиксированной запятой, рассмотрим следующий пример.

Электронная цифровая машина *Стрела* имеет плавающую запятую; ячейки ее памяти состоят из сорока трех разрядов каждая. Для нее (два разряда являются знаковыми)

$$\mu = 35, \qquad \rho = 6$$

Формула (I.18) дает нам для машины *Стрела* следующий диапазон представимых чисел:

$$2^{-64} \le |N| \le (1 - 2^{-35}) \cdot 2^{63}.$$

Правая часть полученного неравенства представляет собой двенадцатизначное десятичное число.

Машина с запятой, фиксированной после знакового разряда, в случае сорокатрехразрядной ячейки имела бы (один разряд — знаковый):

$$n = 0, \qquad v = 42.$$

При этом в силу (I.13) диапазон представимых чисел был бы:

$$2^{-42} \le |N| \le 1 - 2^{-42}.$$

Широта диапазона представимых чисел делает ненужным умножение на масштабные коэффициенты величин, входящих в задачи, решаемые при помощи машин с плавающей запятой. Но это преимущество достигнуто путем присоединения к каждой ячейке дополнительных разрядов и усложнения правил, по которым выполняются арифметические операции, т. е. путем усложнения конструкции машины и увеличения ее стоимости.

**5. Прямой код.** Пусть $x$ — правильная двоичная дробь (положительная или отрицательная). В частном случае $x$ равен нулю. *Прямым кодом* числа $x$ называют число, которое обозначим символом $[x]_{пр}$, получаемое по следующей формуле:

$$[x]_{пр} = \begin{cases} x, & \text{если } x \ge 0 \\ 1 - x, & \text{если } x \le 0. \end{cases} \qquad (I.19)$$

Например,

$$[0{,}110111001]_{пр} = 0{,}110111001; \; [-0{,}110111001]_{пр} = 1 - (-0{,}110111001) = 1{,}110111001.$$

Из формулы (I.19) видно, что нуль имеет два отвечающих ему значения прямого кода:

$$\left.\begin{aligned} [0]_{пр} &= [+0{,}00\ldots 0]_{пр} = 0{,}00\ldots 0, \\ [0]_{пр} &= [-0{,}00\ldots 0]_{пр} = 1{,}00\ldots 0. \end{aligned}\right\} \qquad (I.20)$$

Мантисса числа, приведенного в двоичной системе счисления к виду $N = m10^{\rho}$, где $|m| < 1$, представляет собой правильную двоичную дробь, а ее изображение в ячейке машины с плавающей запятой совпадает с ее прямым кодом.

В машинах с запятой, фиксированной после знакового разряда (в ячейках таких машин записываются только правильные двоичные дроби), изображение числа в ячейке тоже совпадает с его прямым кодом.

## § 7. Двоичные сумматоры и операции над положительными числами, выполняемые с их помощью

Во многих машинах при передаче чисел из одного устройства в другое цифра 1 изображается сигналом высокого напряжения, а цифра 0 — низким напряжением. Одним из основных элементов арифметических устройств таких машин часто является *одноразрядный двоичный сумматор* (устройство см. § 20). Его условное изображение приведено на рис. 16.

**1. Одноразрядный двоичный сумматор.** Одноразрядный двоичный сумматор имеет три входа (на рисунке помечены буквами $A$, $B$ и $Z_1$) и два выхода (на рис. 16 обозначены буквами $C$ и $Z_2$).

Сумматор устроен так, что при подаче высокого напряжения на все три его входа на обоих его выходах возникает высокое напряжение. При подаче высокого напряжения на какие- либо два из его входов на выходе $C$ напряжение остается низким, а на выходе $Z_2$ становится высоким. При подаче высокого напряжения на какой-либо из входов на выходе $Z_2$ напряжение остается низким, а на выходе $C$ становится высоким. Если на всех трех входах напряжение низкое, то и на обоих выходах оно будет низким. Напоминаем, что высокое напряжение означает цифру

1, а низкое — цифру 0.

Таблица 7 Работа одноразрядного двоичного сумматора

| Напряжение на входах | | | Напряжение на выходах | |
|---|---|---|---|---|
| *A* | *B* | *Z1* | *C* | *Z2* |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | | |
| 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | | |
| 1 | 1 | 0 | | |
| 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | | |
| 1 | 1 | 1 | 1 | 1 |

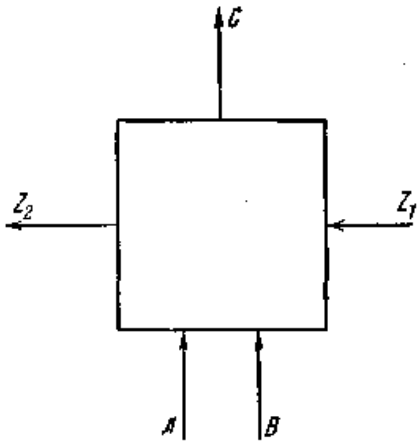



Рис. 16 Условное изображение одноразрядного двоичного сумматора с тремя входами и двумя выходами

Зависимость напряжений, возникающих на выходах сумматора, от напряжений, подаваемых на его входы, наглядно поясняет таблица 7.

Пусть α, β и ξ — однозначные двоичные числа. Если на входы *A* и *B* одноразрядного сумматора подать (в любом порядке) числа α и β, а на вход $Z_1$ — число ξ, то, как видно из таблицы 7, на выходе C получится число γ, которое можно вычислить по формуле

$$\left.\begin{array}{l} \gamma = \alpha + \beta + \xi - 10\xi' \\ \text{где} \\ \xi' = \begin{cases} 0, \text{ если} & \alpha + \beta + \xi < 10 \\ 1, \text{ если} & \alpha + \beta + \xi \geq 10 \end{cases} \end{array}\right\} \qquad \text{(I.21)}$$

**2. Многоразрядный двоичный сумматор.** Путем соединения между собой нескольких одноразрядных двоичных сумматоров получают *многоразрядные двоичные сумматоры.* Существуют две системы многоразрядных сумматоров:

1) многоразрядные сумматоры без переноса из старшего разряда (рис. 17);

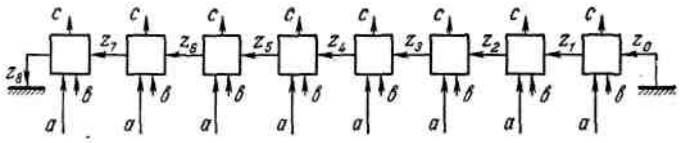


Рис. 17. Многоразрядный двоичный сумматор без переноса из старшего разряда.

2) многоразрядные сумматоры с циклическим переносом (из старшего разряда в младший); схема такого сумматора приведена на рис. 18.

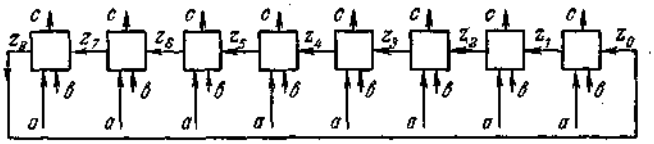


Рис. 18. Многоразрядный двоичный сумматор с циклическим переносом.

Одноразрядные сумматоры, из которых составлен многоразрядный сумматор, называют его *разрядами.* Разряды многоразрядного сумматора соединены между собой последовательно связями, которые помечены на рис. 17 и 18 буквами $z_1$, $z_2$, ..., $z_7$. Кроме того, каждый разряд имеет два свободных входа (обозначенных соответственно буквами *a* и *b*) и один выход (обозначенный буквой *c*).

Легко сообразить, что операции, выполняемые многоразрядными сумматорами указанных двух систем, между собой различны (в сумматоре первой системы перенос из старшего разряда, если он возникает, теряется, а в сумматоре второй системы этот перенос производится в младший разряд). Очевидно также, что операции, выполнимые на многоразрядных сумматорах, отличаются от обычного арифметического сложения. Тем не менее, специальная кодировка чисел позволяет с помощью таких сумматоров выполнять их сложение и вычитание.

**3. Операции над положительными числами, выполняемые на сумматоре без переноса из старшего разряда.** Операция, выполняемая каждым одноразрядным двоичным сумматором (т. е. каждым разрядом многоразрядного сумматора), записанная выше в виде формулы (I.21), в точности совпадает с операцией, выполняемой человеком над одноименными разрядами положительных двоичных чисел при их сложении. Именно, складываются одноименные разряды чисел и к полученному результату прибавляется единица переноса из ближайшего младшего разряда (если такой перенос возник). Если при этом получается величина меньшая, чем 10 (два), то эта величина принимается за соответствующий разряд суммы (перенос в старший разряд не производится). Если же получена величина, которая больше или равна 10, то производится перенос в старший разряд, а за соответствующий разряд суммы принимается полученная величина, уменьшенная на 10.

Из сказанного ясно, что операция над положительными двоичными числами, выполняемая многоразрядным двоичным сумматором, не имеющим переноса из старшего разряда, совпадает с обычным сложением, если количество разрядов сумматора не меньше, чем количество разрядов получаемой суммы. Если же количество разрядов сумматора недостаточно, то результат, выдаваемый сумматором, меньше суммы на величину теряемого переноса из старшего разряда.

Если считать, что запятая помещена после старшего разряда сумматора, а двоичные числа α и β положительны и каждое меньше, чем два, то операция, выполняемая сумматором без переноса из старшего разряда (обозначим ее символом $\oplus$), определяется формулой

$$\alpha \oplus \beta = \begin{cases} \alpha + \beta & \text{при } \alpha + \beta < 10 \\ \alpha + \beta - 10 & \text{при } \alpha + \beta \geq 10. \end{cases} \qquad \text{(I.22)}$$

Из формулы (I.22) видно, что $\alpha \oplus \beta = \beta \oplus \alpha$.

Предположим теперь, что запятая фиксирована после двух старших разрядов сумматора без переноса из старшего разряда. Пусть α и β — положительные двоичные числа, каждое из которых меньше, чем четыре. Операцию, выполняемую сумматором над числами *a* и β, обозначим символом $\oplus'$. Эта операция определяется формулой

$$\alpha \oplus' \beta = \begin{cases} \alpha + \beta & \text{при } \alpha + \beta < 100 \\ \alpha + \beta - 100 & \text{при } \alpha + \beta \geq 100. \end{cases} \qquad \text{(I.23)}$$

Меняя положение запятой относительно разрядов сумматора, можно было бы выполнять с его помощью еще ряд аналогичных операций. Нам в дальнейшем потребуются лишь операции $\oplus$ и $\oplus'$.

**4. Операции над положительными числами, выполняемые на сумматоре с циклическим переносом.** Легко сообразить, что операция над положительными двоичными числами, выполняемая сумматором **с** циклическим переносом, совпадает с обычным сложением, если количество разрядов сумматора не меньше, чем количество разрядов суммы чисел. Если же количество разрядов сумматора окажется недостаточным для того, чтобы вместить сумму, происходит (циклический) перенос из старшего разряда в младший. Последнее равноценно потере переноса из старшего разряда и одновременному прибавлению единицы в младший разряд.

Если считать, что запятая помещена после старшего разряда сумматора, а числа *a* и β положительны и каждое меньше, чем два, то, очевидно, операция, выполняемая сумматором с циклическим переносом (обозначим ее символом $\boxplus$), определяется формулой

$$\alpha \boxplus \beta = \begin{cases} \alpha + \beta, & \text{если } \alpha + \beta < 10 \\ \alpha + \beta - 10 + 10^{-n}, & \text{если } \alpha + \beta \geq 10 \end{cases} \qquad \text{(I.24)}$$

В последней строке формулы (I.24) *n* означает количество разрядов сумматора, отведенных для дробной части суммы. Считаем, что дробные части чисел *a* и β представлены каждая *n* цифрами (стоящими после запятой).Из формулы (I.24) видно, что $\alpha \boxplus \beta = \beta \boxplus \alpha$

Сравнивая формулу (I.24) с формулой (I.22), видим, что они отличаются наличием слагаемого $+10^{-n}$ во второй строке правой части формулы (I.24).

Считая, что запятая расположена после двух старших разрядов сумматора, а числа *a* и β являются положительными двоичными и каждое из них меньше четырех, получим следующую формулу, определяющую операцию $\boxplus'$, выполняемую над числами *a* и β сумматором с циклическим переносом:

$$\alpha \boxplus' \beta = \begin{cases} \alpha + \beta, & \text{при } \alpha + \beta < 100 \\ \alpha + \beta - 100 + 10^{-n}, & \text{при } \alpha + \beta \geq 100 \end{cases} \qquad \text{(I.25)}$$

В этой формуле по-прежнему буква *n* означает количество разрядов сумматора, отведенных под дробную часть суммы. Заметим, что для одного и того же сумматора (с неизменным общим числом разрядов) буква *n* в формуле (1.25) обозначает число, на единицу меньшее, чем в формуле (1.24).

## § 8. Алгебраическое сложение в дополнительном коде

**1. Дополнительный код.** Рассмотрим применение сумматора без переноса из старшего разряда для выполнения

арифметических операций над относительными числами.

Остановимся сперва на случае, когда запятая фиксирована после старшего разряда сумматора без переноса из старшего разряда и, следовательно, сумматор предназначен для выполнения операции $\oplus$.

Пусть $x$ и $y$ — двоичные числа, удовлетворяющие условиям

$$|x|<1, \quad |y|<1, \quad |x+y|<1. \tag{I.26}$$

Первые два неравенства всегда имеют место для чисел $x$ и $y$, поступающих на сумматор из памяти машины, ячейки которой имеют запятую, фиксированную перед первым цифровым разрядом (эти неравенства справедливы также для мантисс, хранящихся в ячейках машины с плавающей запятой). Третье неравенство необходимо для того, чтобы результат операции мог быть снова перенесен в ячейку памяти.

Чтобы произвести на сумматоре алгебраическое сложение чисел $x$ и $y$, их следует заменить некоторым кодом, так как сумматор производит не сложение, а операцию $\oplus$, и притом лишь над положительными числами, тогда как $x$ и $y$ могут быть как положительными, так и отрицательными. Обозначим временно этот код символом $f(\ )$ и потребуем, чтобы он удовлетворял условию

$$f(x) \oplus f(y) = f(x+y). \tag{I.27}$$

Кроме того, выбранный нами код должен удовлетворять также условию

$$0 \le f(x) < 10, \tag{I.28}$$

для того чтобы к нему всегда можно было применить операцию $\oplus$.

Покажем, что обоим выставленным нами требованиям удовлетворяет код, задаваемый формулой

$$f(x) = \begin{cases} x & \text{при } x \ge 0 \\ 10 + x & \text{при } x < 0 \end{cases} \tag{I.29}$$

В том, что $f(x)$ удовлетворяет при любом допустимом значении $x$ условию (I.28), легко убедиться простым подсчетом. Именно, при $0 \le x < 1$ формула (I.29) дает: $f(x) = x$; следовательно, $0 \le f(x) < 1$, а значит, и подавно $0 \le f(x) < 10$.

При $-1 < x < 0$ формула (I.29) дает:

$$f(x) = 10 + x. \tag{I.30}$$

Прибавляя число 10 ко всем трем частям неравенства $-1 < x < 0$, получим $1 < 10+x < 10$, а значит, и подавно $0 \le 10+x < 10$. Учитывая (I.30), можем написать:

$$0 \le f(x) < 10.$$

Остается доказать, что $f(x)$ удовлетворяет также требованию (I.27). Для этого сперва обратим формулу (I.29). Получим:

$$x = \begin{cases} f(x) & \text{при } x \ge 0 \\ f(x) - 10 & \text{при } x < 0 \end{cases} \tag{I.31}$$

Для доказательства придется рассмотреть четыре возможных случая, представляющихся при сложении чисел:

1) оба слагаемых неотрицательны;
2) одно из слагаемых неотрицательно, другое отрицательно, а сумма неотрицательна;
3) одно из слагаемых неотрицательно, другое отрицательно, а сумма отрицательна;
4) оба слагаемых отрицательны.

Переходим к рассмотрению каждого случая в отдельности.

1. Оба слагаемых неотрицательны. При этом

$$\left.\begin{array}{l} 0 \le x < 1 \\ 0 \le y < 1 \\ 0 \le x + y < 1 \end{array}\right\} \tag{I.32}$$

В силу (I.29) $f(x) = x, f(y) = y$, следовательно,

$$f(x) \oplus f(y) = x \oplus y \tag{I.33}$$

Третье неравенство из (I.32) показывает, что $0 \le x+y < 10$, поэтому в силу (I.22) получаем:

$$x \oplus y = x+y. \tag{I.34}$$

Наконец, на основании третьего неравенства из (I.32) и формулы (1.31) заключаем, что

$$x+y = f(x+y). \tag{I.35}$$

Соединив вместе (I.33), (I.34) и (I.35), имеем:

$$f(x) \oplus f(y) = f(x+y).$$

2. Одно из слагаемых неотрицательно, другое отрицательно, а сумма их неотрицательна. Для определенности будем считать, что $y<0$. Тогда

$$\left.\begin{array}{l} 0 \le x < 1 \\ -1 < y < 0 \\ 0 \le x + y < 1 \end{array}\right\} \qquad \text{(I.36)}$$

В силу (I.29) $f(x) = x$, $f(y) = 10 + y$; поэтому

$$f(x) \oplus f(y) = x \oplus (10+y). \qquad \text{(I.37)}$$

Так как $x+y \ge 0$, то $x + (10+y) \ge 10$. Поэтому с помощью формулы (I.22) получаем:

$$x \oplus (10+y) = x+(10+y)-10 = x+y. \qquad \text{(I.38)}$$

Третье неравенство из (I.36) вместе с (I.31) позволяют заключить, что

$$x+y = f(x+y). \qquad \text{(I.39)}$$

Соединяя вместе (I.37), (I.38) и (I.39), получаем:

$$f(x) \oplus f(y) = f(x+y).$$

3. Одно из слагаемых неотрицательно, другое отрицательно и сумма их отрицательна. Для определенности будем считать, что $y<0$. Тогда

$$\left.\begin{array}{l} 0 \le x < 1 \\ -1 < y < 0 \\ -1 < x + y < 0 \end{array}\right\} \qquad \text{(I.40)}$$

Как и при рассмотрении предыдущего случая, имеем:

$$f(x) \oplus f(y) = x \oplus (10+y). \qquad \text{(I.41)}$$

Третье неравенство из (I.40) показывает, что $x + (10 + y) < 10$, следовательно, в силу (I.22)

$$x \oplus (10+y) = x+(10+y) = 10 + (x+y). \qquad \text{(I.42)}$$

Третье неравенство из (I.40) вместе с формулой (I.31) дают:

$$10 + (x+y) = 10 + f(x+y) - 10 = f(x+y) \qquad \text{(I.43)}$$

Соединяя (I.41), (I.42) и (I.43), получаем:

$$f(x) \oplus f(y) = f(x+y).$$

4. Оба слагаемых отрицательны. Тогда

$$\left.\begin{array}{l} -1 < x < 0, \\ -1 < y < 0, \\ -1 < x + y < 0. \end{array}\right\} \qquad \text{(I.44)}$$

Из формулы (I.29) имеем, что $f(x) = 10 + x$, $f(y) = 10 + y$ и, следовательно,

$$f(x) \oplus f(y) = (10+x) \oplus (10+y). \qquad \text{(I.45)}$$

С помощью третьего неравенства из (I.44) видим, что

$$(10 + x)+(10+y) = 100 + (x+y) > 10.$$

Поэтому из формулы (I.22) получаем:

$$(10 + x) \oplus (10+y) = (10 + x) + (10 + y) - 10 = 10 + (x + y). \qquad \text{(I.46)}$$

С помощью третьего неравенства из (I.44) и формулы (I.29) можно написать:

$$10 + (x + y) = f(x + y). \qquad \text{(I.47)}$$

Соединяя (I45), (I.46) и (I.47), имеем:

$$f(x) \oplus f(y) = f(x + y).$$

Итак, мы доказали, что код (I.29) удовлетворяет условию (I.27). Этот код известен под названием

*дополнительного кода* и обозначается символом $[\ ]_{доп}$. Используя последнее обозначение, перепишем формулу (I.29) в виде

$$[x]_{доп} = \begin{cases} x, & \text{если } x \geq 0, \\ 10 + x, & \text{если } x < 0 \quad (10 \text{ означает два}). \end{cases} \qquad \text{(I.48)}$$

Например,

$[0{,}110111001]_{доп} = 0{,}110111001$; $[-\ 0{,}110111001]_{доп} = 10 + (-0{,}110111001) = 1{,}001000111$.

Таким образом, для положительного числа $x$ прямой и дополнительный коды совпадают, для отрицательного $x$ они различны. Внимательно рассматривая последний пример, замечаем, что дополнительный код отрицательного числа можно получить из этого числа следующим способом: знак числа отбросим. В разряде целых единиц поставим цифру 1. В дробной части числа каждую единицу заменим нулем, а нуль — единицей. К младшему разряду полученного числа прибавим единицу, т. е. если $x = -\ 0, x_1\ x_2\ x_3\ \ldots\ x_n$, где $x_1, x_2, x_3, \ldots, x_n$ — двоичные цифры, каждая из которых является либо нулем, либо единицей, то

$$[x]_{доп} = 1, \bar{x}_1, \bar{x}_2, \bar{x}_3 \ldots \bar{x}_n + \underbrace{0{,}000\ldots01}_{(n\text{-}1)\text{ нулей}} = 1, \bar{x}_1, \bar{x}_2\bar{x}_3 \ldots \bar{x}_n + 10^{-n}. \qquad \text{(I.49)}$$

Здесь $x_i$ является нулем, если $x_i = 1$, или единицей, если $x_i = 0$ (при $i = 1, 2, \ldots, n$).

Докажем, что формула (I.49), составленная в результате рассмотрения одного лишь примера, действительно верна, т. е. эквивалентна (для отрицательного $x$) формуле (1.48).

Для этого вычтем из $[x]_{доп}$, полученного по формуле (1.49), число $x$, учитывая, что $x_i + \bar{x}_i = 1$. Имеем:

$$[x]_{доп} - x = 1, 1, \bar{x}_1, \bar{x}_2 \ldots \bar{x}_n + 10^{-n} \ -\!\!-\!\!- (-\!\!-\ 0, x_1, x_2\ \ldots\ x_n) = = 1{,}11\ \ldots\ 11 + 0{,}00\ldots\ 01 = 10{,}00\ldots\ 00 = 10.$$

Из последнего результата получаем:

$$[x]_{доп} = 10 + x \quad \text{при} \quad x < 0,$$

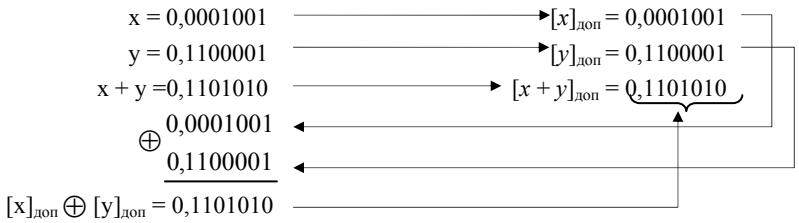
что и требовалось доказать.

Из формулы (1.48) вытекает, что число 0 имеет только одно значение дополнительного кода $[0]_{доп} = 0$.

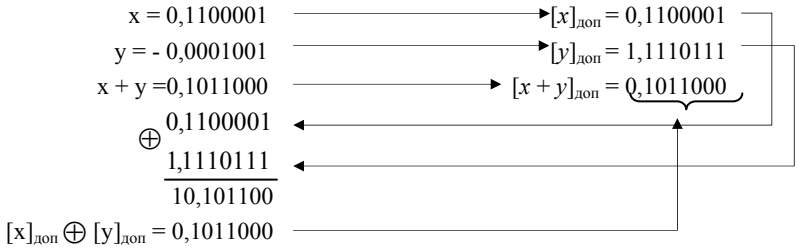
Старшая цифра дополнительного кода при изображении положительного числа является нулем, а при изображении отрицательного числа — единицей. Так как в машинах знак «+» изображают цифрой 0, а знак «—» (минус) — цифрой 1, то старший разряд дополнительного кода принято называть *знаковым разрядом.*

Проиллюстрируем примерами вышерассмотренные случаи сложения правильных двоичных дробей на сумматоре без переноса из старшего разряда.

Пример 1[*)].

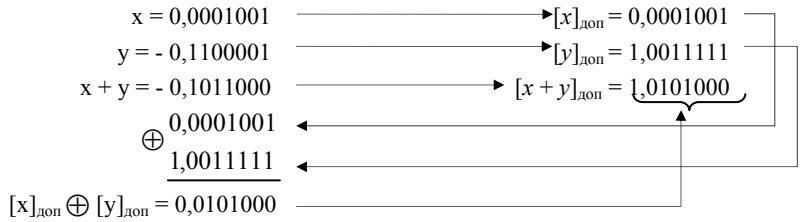



Пример 2.

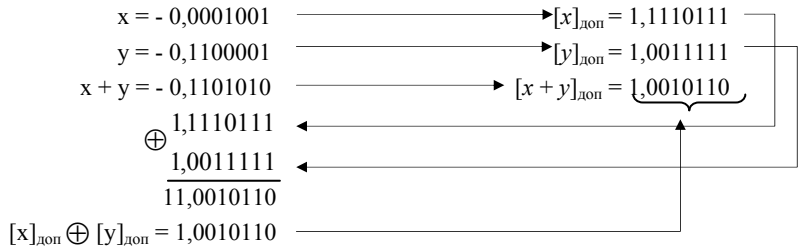



[*)] В этом и следующих примерах в левом столбце приведены в двоичной" системе счисления слагаемые и их сумма, а в правом столбце — дополнительные коды этих чисел. Под столбцами показана операция, выполняемая на сумматоре над кодами слагаемых и приводящая к получению кода суммы. В тех случаях, когда происходит потеря переноса из старшего разряда сумматора, внизу под чертой приведена сумма кодов слагаемых, а под ней код суммы, выдаваемый сумматором

Пример 3.



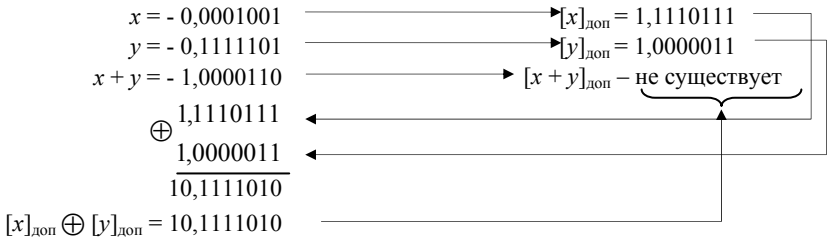

Пример 4.



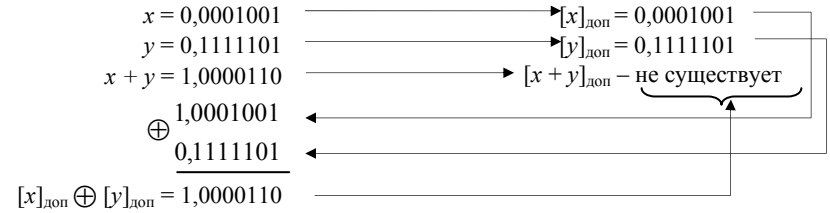

Покажем на примерах, что при $|x + y| \geq 1$ происходит переполнение разрядной сетки сумматора.

Пример 5

.




Пример 6.




Рекомендуем читателю самому придумать еще несколько примеров и по ним посмотреть, как переполняется разрядная сетка сумматора при сложении дополнительных кодов чисел, если сумма последних по абсолютному значению больше единицы.

Легко сообразить, что разрядная сетка переполняется лишь тогда, когда числа $x$ и $y$ имеют одинаковые знаки. Признаком переполнения может служить тот факт, что число, даваемое сумматором в качестве $x + y$, имеет знак, противоположный знакам слагаемых. Точнее, признак переполнения состоит в том, что на выходе сумматора в целой части получается нуль, тогда как целые числа величин $[x]_{доп}$ и $[y]_{доп}$ были единицами, или на выходе сумматора в целой части получается единица, тогда как целые части величины $[x]_{доп}$ и $[y]_{доп}$ были нулями.

Как уже указывалось, переполнение разрядной сетки при вычислениях недопустимо. Поэтому в машине, выполняющей сложение чисел в дополнительных кодах, предусматривают устройство, сравнивающее целые части слагаемых кодов и в случае их совпадения сравнивающее с одной из них целую часть величины, получающейся на выходе сумматора. При переполнении разрядной сетки это устройство должно посылать сигнал в управляющее устройство машины.

Заметим, что вычитание чисел легко сводится к сложению по формуле

$$x - y = x + (-y),$$

и, следовательно, при сложении величин $[x]_{доп}$ и $[-y]_{доп}$ на выходе сумматора без переноса из старшего разряда получится величина $[x - y]_{доп}$.

**2. Модифицированный дополнительный код.** Для алгебраического сложения на сумматоре без переноса из старшего разряда двоичных чисел $x$ и $y$, удовлетворяющих условию

$$|x|<1. \quad |y|<1,$$
$$|x+y|<1$$

можно вместо операции $\oplus$ воспользоваться операцией $\oplus'$, показанной в формуле (I.23) § 7. При этом операцию $\oplus'$ следует выполнять не над самими слагаемыми, а над их так называемым *модифицированным дополнительным кодом* (для которого принято обозначение $[\ \ ]_{доп}$).

*Модифицированный дополнительный код* правильной двоичной дроби определяется по формуле

$$[x]^{M}_{доп} = \begin{cases} x & при \quad x \geq 0, \\ 100 + x & при \quad x < 0 \end{cases} \qquad (I.50)$$

Если $x=0, x_1x_2... x_n$, то его модифицированный дополнительный код записывают в следующем виде:

$$[x]^{M}_{доп} = 00, x_1x_2x_3 \ldots x_n. \qquad (I.51)$$

При $x = 0, x_1x_2 \ldots x_n$

$$[x]^{M}_{доп} = 11, \bar{x}_1, \bar{x}_2 \ldots \bar{x}_n + 10^{-n}. \qquad (I.52)$$

Формула (1.51) не требует доказательства, а для получения формулы (1.52) применяются те же приемы, что и для получения соответствующей формулы для обычного дополнительного кода.

Пример 7.

1) $[0{,}110111001]^{M}_{доп} = 00{,}110111001;$

2) $[-0{,}110111001]^{M}_{доп} = 11{,}001000111.$

Старшие две цифры модифицированного дополнительного кода при изображении положительного числа являются нулями, а при изображении отрицательного числа — единицами. Выше было сказано, что знак «+» (плюс) в машинах принято изображать цифрой 0, а знак «-» (минус) — цифрой 1. Эти обстоятельства привели к тому, что старшие два разряда модифицированного дополнительного кода стали называть *знаковыми разрядами.*

Легко показать с помощью рассуждений, подобных тем, которые были проведены для дополнительного кода, что модифицированный дополнительный код удовлетворяет условию

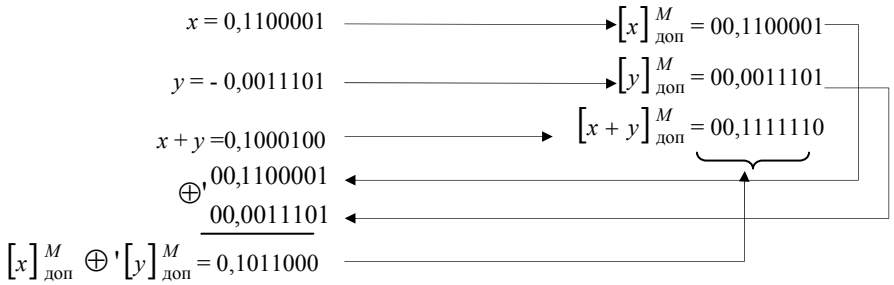
$$[x]^{M}_{доп} \oplus' [y]^{M}_{доп} = [x+y]^{M}_{доп}, \qquad (1.53)$$

причем для любого допустимого $x$

$$0 \leq x^{M}_{доп} < 100. \qquad (I.54)$$

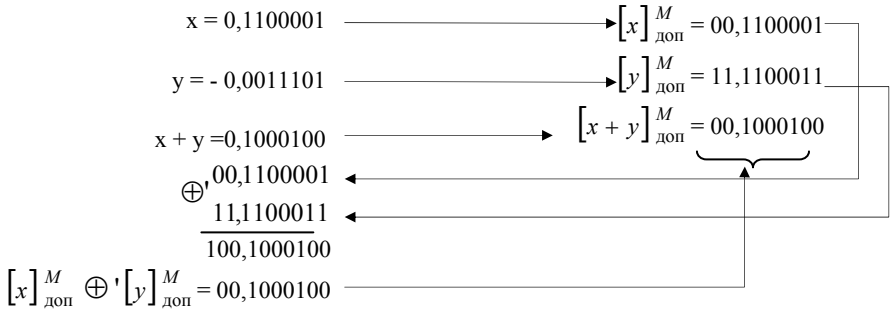
Не проводя этих рассуждений, проиллюстрируем наше утверждение примерами.

Пример 8*).

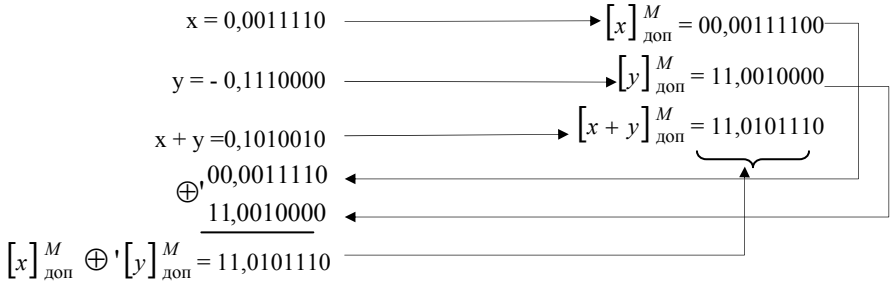



*) В этом и следующих примерах в левом столбце приведены в двоичной системе счисления слагаемые и их сумма, а в правом столбце — модифицированные дополнительные коды этих чисел. Внизу показана операция, выполняемая на сумматоре над кодами слагаемых и приводящая к получению кода суммы. В тех случаях, когда происходит потеря переноса из старшего разряда сумматора, внизу под чертой приведена сумма кодов слагаемых, а под ней код суммы, выдаваемый сумматором.

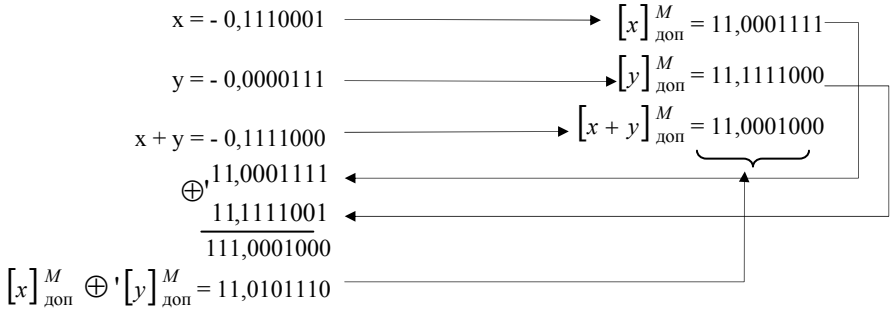
Пример 9.




Пример 10.

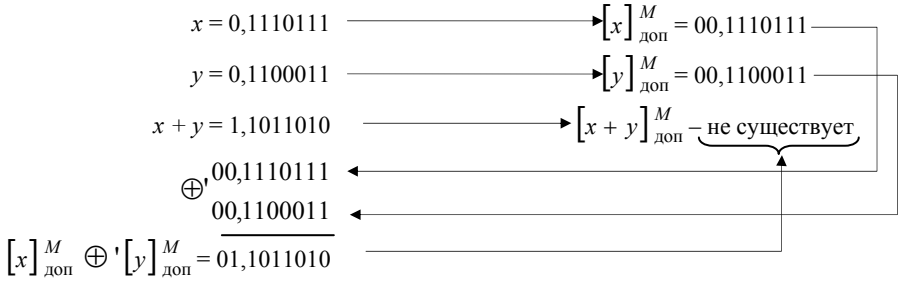



Пример 11.



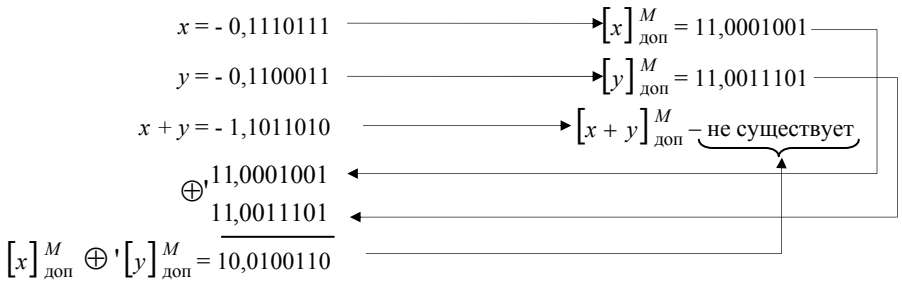

Как и при сложении обычных дополнительных кодов, в случае сложения модифицированных дополнительных кодов чисел $x$ и $y$ при $|x + y| \geq 1$ переполняется разрядная сетка сумматора. Ясно что это может происходить лишь в случае, если $x$ и $y$ имеют одинаковые знаки. Приведем два примера.

Пример 12.




В данном случае переполнение разрядной сетки сумматора выражается в том, что на выходе его получается результат, который нельзя назвать модифицированным дополнительным кодом какого-нибудь числа.

Пример 13.



Полученный результат так же, как в предыдущем примере, нельзя назвать модифицированным дополнительным кодом какого-либо числа.

Рассматривая приведенные примеры, заключаем, что признаком переполнения разрядной сетки при сложении чисел в модифицированном дополнительном коде с помощью сумматора без переноса из старшего разряда является *несовпадение цифр, получающихся на выходах двух старших разрядов сумматора.*

Для того чтобы уловить наступление переполнения разрядной сетки, машина, складывающая числа в модифицированном дополнительном коде, должна иметь устройство, сравнивающее цифры, полученные на выходах двух старших разрядов сумматора. При несовпадении этих цифр устройство должно передавать в управляющее устройство сигнал (импульс).

Контроль за переполнением при сложении чисел в обычном дополнительном коде сложнее хотя бы уже потому, что контролирующее устройство выполняет два сравнения, а не одно, как в случае применения модифицированного дополнительного кода.

Вычитание чисел в модифицированном дополнительном коде сводится к сложению в соответствии с формулой

$$x\text{-}y=x+(\text{-}y);$$

при этом

$$[x]^{M}_{\text{доп}} \oplus' [-y]^{M}_{\text{доп}} = [x-y]^{M}_{\text{доп}}$$

## § 9. Алгебраическое сложение в обратном коде

**1. Обратный код.** Рассмотрим применение сумматора с циклическим переносом для выполнения арифметических операций над относительными числами.

Для того чтобы произвести алгебраическое сложение двоичных чисел $x$ и $y$, удовлетворяющих условиям

$$|x|<1,$$
$$|y|<1,$$
$$|x+y|<1$$

на сумматоре с циклическим переносом, необходимо представить эти числа в некотором коде.

Остановимся сперва на случае, когда запятая фиксирована после одного старшего разряда сумматора, т. е. на случае, когда сумматор выполняет операцию. Обозначая искомый код символом $f$ (  ), потребуем, чтобы он удовлетворял условию

$$f(x) \boxplus f(y)=f(x+y). \qquad \text{(I.55)}$$

Другими словами, мы требуем, чтобы при выполнении операции $\boxplus$ над кодами двух чисел получался код суммы этих чисел. Кроме того, необходимо потребовать, чтобы код удовлетворял условию

$$0\le f(x)<10, \qquad \text{(I.56)}$$

без которого не всегда можно будет выполнять над кодами операцию $\boxplus$ .

Легко убедиться, что обоим условиям удовлетворяет код, определяемый формулой

$$f(x)=\begin{cases} x & \text{при} \quad x \ge 0, \\ 10+x-10^{-n} & \text{при} \quad x \le 0. \end{cases} \qquad \text{I.57)}$$

В том, что этот код удовлетворяет условию (I.56), можно убедиться с помощью рассуждений, подобных проведенным нами для дополнительного кода. Предоставляем читателю сделать это самостоятельно.

Заметим, что для нуля возможны два значения кода (I.57):

$$\left.\begin{aligned} f(0)&=f(+0)=0{,}00...0, \\ f(0)&=f(-0)=10-10^{-n}=1{,}11...1. \end{aligned}\right\} \qquad \text{(I.58)}$$

Первое из этих значений получается из верхней строки формулы (I.57), а второе — из ее нижней строки. Первое из них принято называть *кодом положительного нуля,* а второе — *кодом отрицательного нуля.* Повторяем, что каждое из них является кодом нуля.

Для того чтобы доказать, что выбранный нами код удовлетворяет условию (1.55), придется рассмотреть больше случаев, чем при изучении дополнительного кода. Отдельного рассмотрения требуют случаи, когда

1) оба слагаемых являются нулями;

2) слагаемые отличны от нуля, а сумма их равна нулю;
3) одно из слагаемых является нулем.

Кроме того, нужно рассмотреть четыре основных случая:

4) оба слагаемых положительны;
5) слагаемые имеют различные знаки и положительную сумму;
6) слагаемые имеют различные знаки и отрицательную сумму;
7) оба слагаемых отрицательны.

Приступаем к рассмотрению каждого из этих случаев в отдельности.

1. Оба слагаемых являются нулями: $x = 0, y = 0$. Непосредственно видно, что

$$f(+0) = \boxplus f(+0) = f(+0)$$

$$f(+0) = \boxplus f(-0) = f(-0) \qquad (I.59)$$

Покажем, что

$$f(-0) \boxplus f(-0) = f(-0). \qquad (I.60)$$

Действительно,

$$f(-0) \boxplus f(-0) = 1,11 \ldots 11 \boxplus 1,11 \ldots 11. \qquad (I.61)$$

Так как 1,11...11 + 1,11...11 = 11,11...10>10, то формула (I.24) дает:

$$1,11 \ldots 1101,11 \ldots 11 = 1,11 \ldots 11 \boxplus 1,11 \ldots 11 — 10+0,00 \ldots 01 = 1,11 \ldots 11. \qquad (I.62)$$

Формулы (I.61), (I.62) и вторая из (I.58) позволяют написать:

$$f(-0) \boxplus f(-0) = f(-0),$$

что и требовалось доказать.

Резюмируя полученные результаты, видим, что

$$f(0) \boxplus f(0) = f(0),$$

т. е. в рассматриваемом случае

$$f(x) \boxplus f(y) = f(x+y)$$

2. Оба слагаемых равны между собой по абсолютной величине и противоположны по знакам. Одно из них обозначим через $x$ ($x>0$), а другое —через — $x$. Покажем, что

$$f(x) \boxplus f(-x) = f(-0) \qquad (I.63)$$

Действительно, в силу того, что $x>0$, и, следовательно,—$x<0$, формула (1.57) дает:

$$f(x) \boxplus f(-x) = x \boxplus (10\text{-}x\text{-}10)^{-n} \qquad (I.64)$$

Так как $x + (10\text{-}x\text{-}10^{-n}) = 10\text{-}10^{-n}<10$, то с помощью формулы (I.24) получаем:

$$x + (10\text{-}x\text{-}10^{-n}) = x + (10\text{-}x\text{-}10^{-n}) = 10\text{-}10^{-n}. \qquad (I.65)$$

Сопоставляя (I.64), (I.65) и вторую формулу из (I.58), получаем:

$$f(x) + f(-x) = f(-0),$$

что и требовалось доказать.

3. Одно из слагаемых равно нулю, а другое отлично от нуля. Для определенности пусть $x \neq 0$, $y = 0$. При этом в силу (I.57)

$$0,00 \ldots 0 < f(x) < 1,11 \ldots 1. \qquad (I.66)$$

Покажем, что

$$f(x) \boxplus f(0) = f(x). \qquad (I.67)$$

Формула (I.67) очевидна, если $f(0) = 0,00...0$. Чтобы доказать ее в случае, когда $f(0) = 1,11...1$, достаточно, в силу (I.66), показать, что при

$$0 < a < 1,11 \ldots 1 \qquad (I.68)$$

справедлива формула

$$a \boxplus 1,11 \ldots 1 = a.$$

Заметим, что условие

$$a>0,$$

содержащееся в условии (I.68), означает, что $a \geq 0,00...01$, так как всякое положительное число $a = f(x)$, меньшее, чем 0,00...01, отвечает машинному нулю и будет подано на вход сумматора в виде нуля. Поэтому

$$a + 1,11 \ldots 11 \geq 0,00 \ldots 01 + 1,11 \ldots 11 = 10.$$

Следовательно, формула (I.24) даст нам:

$$a \boxplus 1{,}11\ldots11 = a + 1{,}11\ldots11 - 10 + 10^{-n} = a,$$

что и требовалось доказать.

Теперь остается рассмотреть еще четыре основных случая, приступая к которым, предварительно обратим формулу (I.57). Получим:

$$x = \begin{cases} f(x) & \text{при} \quad 0 \le x < 1, \\ f(x) - 10 + 10^{-n} & \text{при} \quad -1 < x \le 0. \end{cases} \tag{I.69}$$

4. Оба слагаемых положительны, так что

$$\left.\begin{aligned} 0 < x &< 1, \\ 0 < y &< 1, \\ 0 < x + y &< 1. \end{aligned}\right\} \tag{I.70}$$

В силу первых двух неравенств из (I.70) и формулы (I.57) имеем $f(x) = x; f(y) = y,$ следовательно,

$$f(x) \boxplus f(y) = x \boxplus y \tag{I.71}$$

Третье неравенство из (I.70) и формула (I.24) дают:

$$x \boxplus y = x + y. \tag{I.72}$$

Третье неравенство из (I.70) и формула (I.69) дают:

$$x + y = f(x + y). \tag{I.73}$$

Сопоставляя (I.71), (I.72) и (I.73), получаем:

$$f(x) \boxplus f(y) = f(x + y).$$

5. Одно из слагаемых положительно, другое отрицательно, а сумма положительна. Для определенности положим:

$$\left.\begin{aligned} 0 < x &< 1, \\ -1 < y &< 0, \\ 0 < x + y &< 1. \end{aligned}\right\} \tag{I.74}$$

Учитывая первые два неравенства из (I.74) и пользуясь формулой (I.57), имеем $f(x) = x; \quad f(y) = 10 + y - 10^{-n}$. Следовательно,

$$f(x) \boxplus f(y) = x \boxplus (10 + y - 10^{-n}). \tag{I.75}$$

В силу того, что $x + (10 + y - 10^{-n}) = 10 + (x + y) - 10^{-n} \ge 10$, формула (I.24) дает:

$$x \boxplus (10 + y - 10^{-n}) = x + (10 + y - 10^{-n}) - 10 + 10^{-n} = x + y. \tag{I.76}$$

На основании третьего неравенства из (I.74) и формулы (I.69) имеем:

$$x + y = f(x + y). \tag{I.77}$$

Сопоставляя (I.75), (I.76) и (I.77), получаем:

$$f(x) \boxplus f(y) = f(x + y).$$

6. Одно из слагаемых положительно, другое отрицательно, сумма отрицательна. Например, для определенности положим:

$$\left.\begin{aligned} 0 < x &< 1, \\ -1 < y &< 0, \\ -1 < x + y &< 0. \end{aligned}\right\} \tag{I.78}$$

Как и в предыдущем случае, имеем:

$$f(x) \boxplus f(y) = x \boxplus (10 + y - 10^{-n}). \tag{I.79}$$

Однако теперь

$$x + (10 + y - 10^{-n}) = 10 + (x+y) - 10^{-n} < 10$$

ввиду того, что $x + y < 0$. Следовательно, формула (I.24) дает:

$$x \boxplus (10 + y - 10^{-n}) = x + (10 + y - 10^{-n}) = 10 + (x+y) - 10^{-n}. \quad \text{(I.80)}$$

Третье неравенство из (I.78) вместе с формулой (I.69) позволяют написать, что

$$x+y = f(x+y) - 10 + 10^{-n}. \quad \text{(I.81)}$$

Сопоставляя (I.79), (I.80) и (I.81), имеем:

$$f(x) \boxplus f(y) = f(x+y).$$

7. Оба слагаемых отрицательны, т. е.

$$\left.\begin{aligned} -1 < x \quad &< 0, \\ -1 < y \quad &< 0, \\ -1 < x + y &< 0. \end{aligned}\right\} \quad \text{(I.82)}$$

На основании первых двух неравенств из (I.82) и формулы (I.57), напишем:

$$f(x) \boxplus f(y) = (10 + x - 10^{-n}) \boxplus (10 + y - 10^{-n}). \quad \text{(I.83)}$$

Ввиду того, что

$$(10 + x - 10^{-n}) + (10 + y - 10^{-n}) = 100 + (x + y) - 0{,}00 \ldots 010 > 10,$$

формула (I.24) дает:

$$(10 + x - 10^{-n}) \boxplus (10 + y - 10^{-n}) = (10 + x + 10^{-n}) + (10 + y - 10^{-n}) - 10 + 10^{-n} = 10 + (x+y) - 10^{-n}. \quad \text{(I.84)}$$

Третье неравенство из (I.82) вместе с формулой (I.69) позволяют написать, что

$$x+y = f(x+y) - 10 + 10^{-n}. \quad \text{(I.85)}$$

Соединяя (I.83), (I.84) и (I.85), получаем:

$$f(x) \boxplus f(y) = f(x+y).$$

Итак, мы доказали, что выбранный нами код (I.57) удовлетворяет условию (1.55).

Этот код получил название *обратного кода.* Его принято обозначать символом $[\ ]_{\text{обр}}$. Переписывая в новых обозначениях формулу (I.57), имеем:

$$[x]_{\text{обр}} = \begin{cases} x & \text{при} \quad x \geq 0 \\ 10 - 10^{-n} + x & \text{при} \quad x \leq 0 \end{cases} \quad \text{(I.86)}$$

(10 означает два; $n$ — количество знаков дробной части числа $x$). Например,

$$[0{,}110111001]_{\text{обр}} = 0{,}110111001;$$

$$[-0{,}110111001]_{\text{обр}} = 10 - 10^{1001} + (-0{,}110111001) = 10 - 0{,}000000001 - 0{,}110111001 =$$

$$= 1{,}111111111 - 0{,}110111001 = 1{,}001000110.$$

Таким образом, для положительного $x$ его прямой, дополнительный и обратный коды совпадают между собой. Для отрицательного $x$ все они различны. Внимательно рассматривая последний пример, видим, что для отрицательного двоичного числа обратный код получается следующим образом: знак числа отбрасывают; в качестве целой части кода берут 1; все нули, входящие в дробную часть числа, заменяют единицами, а все единицы заменяют нулями, т. е. если $x = -0, x_1 x_2 \ldots x_n$, где $x_1, x_2, \ldots, x_n$—двоичные цифры, каждая из которых является либо нулем, либо единицей, то

$$[x]_{\text{обр}} = 1, \bar{x}_1 \bar{x}_2 \ldots \bar{x}_n \quad \text{(I.87)}$$

Докажем, что формула (I.87), составленная нами после рассмотрения лишь одного примера, в самом деле верна, т. е. что она эквивалентна (при отрицательном $x$) формуле (I.86). Для этого от $[x]_{\text{обр}}$, полученного по формуле (I.87), отнимем $x$. Учитывая, что $xi + xi = = 1$, получаем:

$$[x]_{\text{обр}} - x = 1, \bar{x}_1 \bar{x}_2 \ldots \bar{x}_n - (-0, x_1 x_2 \ldots x_n) = 1{,}11 \ldots 11 = 10 - 0{,}00 \ldots 01 = 10 - 10^{-n}.$$

Из последнего результата получаем:

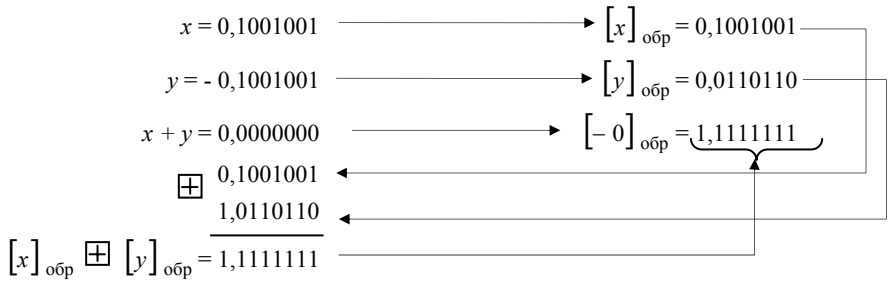
$$[x]_{\text{обр}} = 10 - 10^{-n} + x$$

при $x<0$, что и требовалось доказать (см. формулу (I.86)).

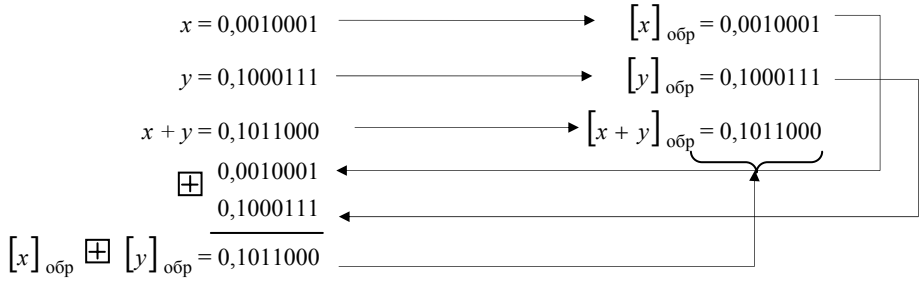
Старшая цифра обратного кода при изображении положительного числа является нулем, а при изображении отрицательного числа — единицей. Поэтому старший разряд обратного кода обычно называют *знаковым разрядом.*

Поясним примерами основные из рассмотренных выше случаев сложения правильных двоичных дробей (представленных обратным кодом) с помощью сумматора с циклическим переносом.

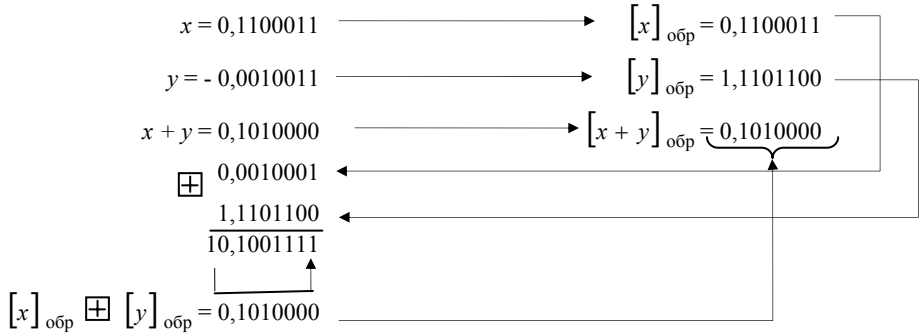
Пример*) 1.



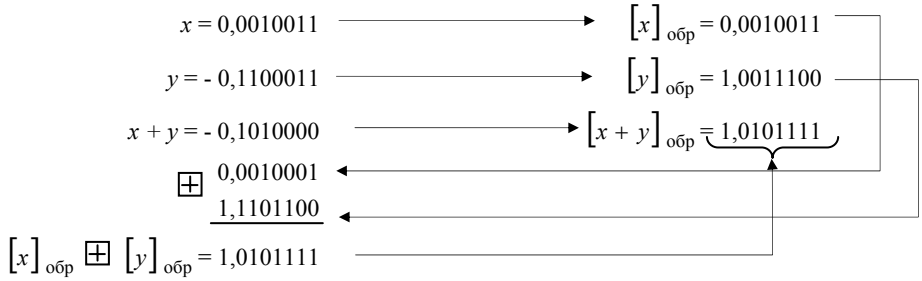

Пример 2.




Пример 3.



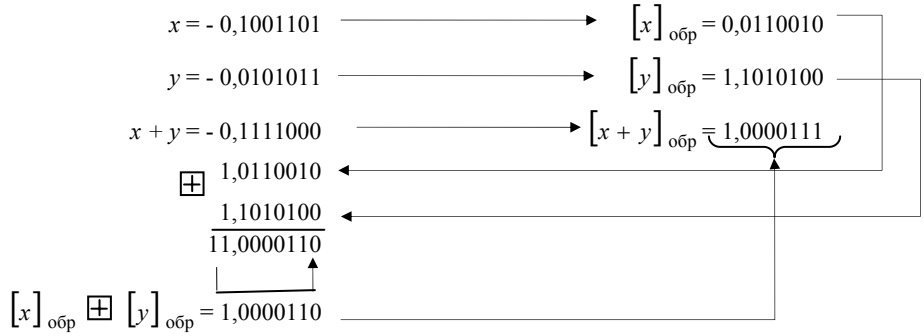

Пример 4.




*) В этом и следующих примерах в левом столбце приведены в двоичной системе счисления слагаемые и их сумма, а в правом столбце — обратные коды этих чисел. Внизу показана операция, выполняемая на сумматоре над кодами слагаемых и приводящая к получению кода суммы.

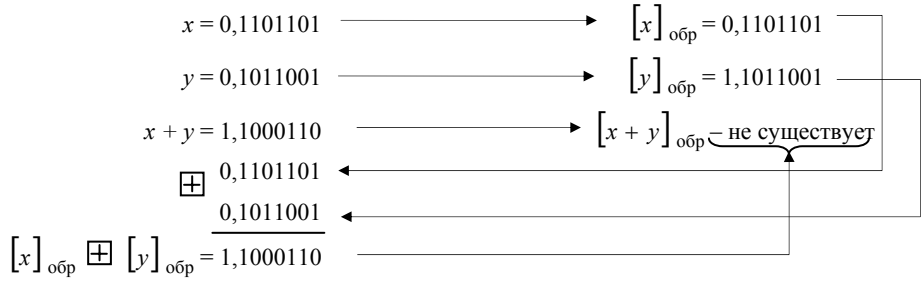
В тех случаях, когда происходит циклический перенос (из старшего разряда сумматора в младший), внизу под чертой приведена сумма кодов слагаемых и с помощью расположенной под ней стрелки изображен циклический перенос. Ниже этой стрелки показан код суммы, выдаваемый сумматором

Пример 5.



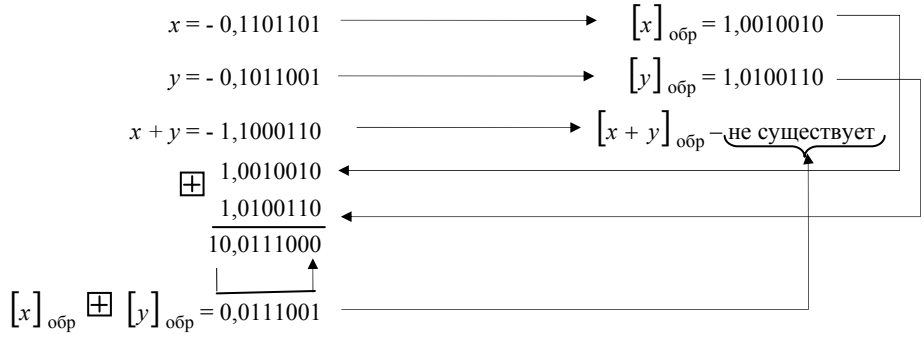

Теперь покажем на примерах, что если $|x + y| \geq 1$, то при сложении обратных кодов чисел $x$ и $y$ переполняется разрядная сетка сумматора, что искажает результат.

Пример 6.




Полученный результат 1,1000110 является обратным кодом отрицательного числа —0,0111001, которое не равно сумме $x + y$.

Пример 7.




В этом примере вместо отрицательной суммы $x + y = -1,1000110$ получен положительный результат 0,0111001.

Рекомендуем читателю самому придумать еще несколько примеров, по которым внимательно рассмотреть, как происходит переполнение разрядной сетки сумматора с циклическим переносом.

Легко видеть, что переполнение разрядной сетки получается лишь при сложении обратных кодов двух чисел, имеющих одинаковые знаки (целые части обратных кодов одинаковы). Признаком переполнения служит противоположность знака суммы знакам слагаемых. Точнее, при переполнении в целой части на выходе сумматора получается единица, если целые части складываемых обратных кодов были нулями. Если же целые части складываемых обратных кодов были единицами, в целой части на выходе сумматора получается нуль.

Для того чтобы уловить момент наступления переполнения разрядной сетки, в машине должно быть устройство, сравнивающее между собой целые части складываемых обратных кодов, и в случае их совпадения сравнивающее одну из них с целой частью результата, получаемого на выходе сумматора. В случае переполнения это устройство должно посылать сигнал в управляющее устройство машины.

Вычитание чисел можно заменить сложением согласно формуле

$$x - y = x + (-y).$$

Отсюда ясно, что, складывая на сумматоре с циклическим переносом $[x]_{обр}$ и $[-y]_{обр}$, мы получим в ответе $[x-y]_{обр}$.

**2. Модифицированный обратный код.** Для алгебраического сложения на сумматоре с циклическим переносом правильных двоичных дробей $x$ и $y$ можно вместо операции $\boxplus$ воспользоваться операцией $\boxplus'$, показанной в формуле (I.25) § 7.

При этом операцию $\boxplus'$ следует выполнять не над самими слагаемыми, а над их так называемым

*модифицированным обратным кодом* (для которого принято обозначение $[\ ]^{M}_{\text{обр}}$).

Модифицированный обратный код правильной двоичной дроби $x$ определяется по формуле

$$[x]^{M}_{\text{обр}} = \begin{cases} x & \text{при} \quad x \geq 0 \\ 100 - 10^{-n} + x & \text{при} \quad x \leq 0 \end{cases} \tag{I.88}$$

где $n$ — число знаков дробной части. Если $x = 0,x_1x_2...x_n$, то его модифицированный обратный код записывают в виде

$$[x]^{M}_{\text{обр}} = 00, x_1x_2 \ldots x_n. \tag{I.89}$$

При $x = -0, x_1x_2...x_n$ (т. е. $x \leq 0$)

$$[x]^{M}_{\text{обр}} = 11, x_1x_2 \ldots \bar{x}_n, \tag{I.90}$$

где $x_i + \bar{x}_i x_i = 1$.

Формула (I.89) не требует доказательства. Формулу (I.90) доказывают с помощью того же приема, которым была доказана соответствующая формула для обычного обратного кода.

Пример 8.

1) $[0,110111001]^{M}_{\text{обр}} = 00,110111001;$
2) $[-0,110111001]^{M}_{\text{обр}} = 11,001000110.$

Модифицированный обратный код нуля (подобно обычному обратному коду) имеет два значения:

$$[0]^{M}_{\text{обр}} = [-0]^{M}_{\text{обр}} = 00,00\ldots0,$$
$$[0]^{M}_{\text{обр}} = [+0]^{M}_{\text{обр}} = 11,11 \ldots 1.$$

Первое из этих значений обычно называют *кодом положительного нуля,* а второе — *кодом отрицательного нуля.*

Старшие две цифры модифицированного обратного кода при изображении положительного числа являются нулями, а при изображении отрицательного числа—единицами. Поэтому старшие два разряда модифицированного обратного кода обычно называют знаковыми разрядами.

Легко показать с помощью рассуждений, подобных проведенным для обратного кода, что модифицированный обратный код удовлетворяет условию

$$[x]^{M}_{\text{обр}} \boxplus' [y]^{M}_{\text{обр}} = [x + y]^{M}_{\text{обр}}. \tag{I.91}$$

При этом для любого допустимого $x$

$$0 \leq [x]^{M}_{\text{обр}} < 100. \tag{I.92}$$

Не проводя этих рассуждений, проиллюстрируем наше утверждение примерами.

Пример*) 9.

$x = 0,1100001 \longrightarrow [x]^{M}_{\text{обр}} = 00,1100001$

$y = -0,0011101 \longrightarrow [y]^{M}_{\text{обр}} = 00,0011101$

$x + y = 0,1111110 \longrightarrow [x + y]^{M}_{\text{обр}} = 00,1111110$

$\boxplus'\ 00,1100001$

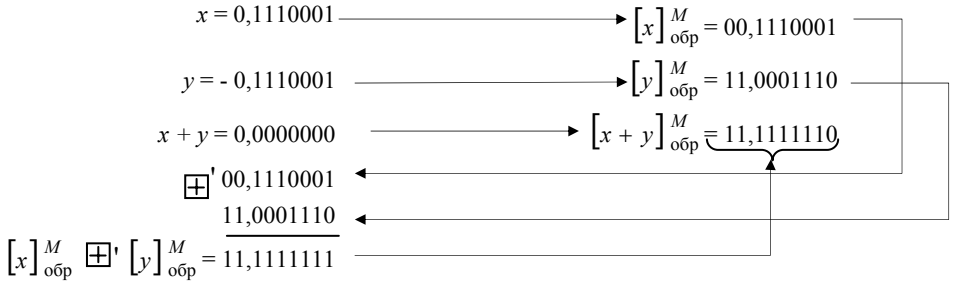
$\underline{\quad 00,0011101}$

$[x]^{M}_{\text{обр}} \boxplus' [y]^{M}_{\text{обр}} = 00,1111110$

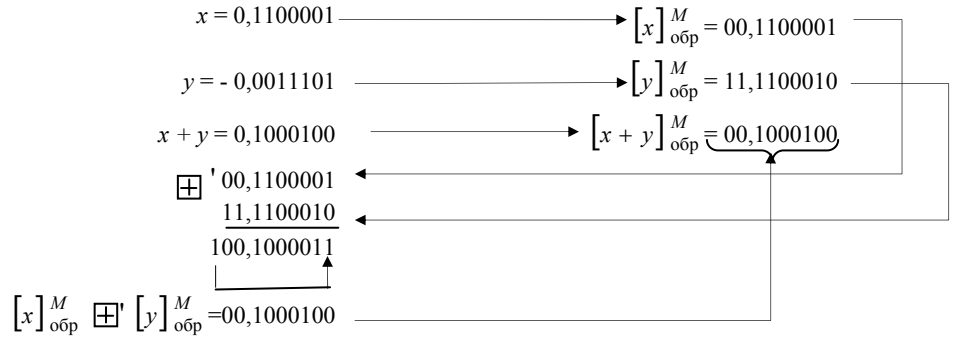
*) В этом и следующих примерах в левом столбце приведены в двоичной системе счисления слагаемые и их сумма, а в правом столбце — модифицированные обратные коды этих чисел.

Внизу показана операция, выполняемая сумматором над кодами слагаемых и приводящая к получению кода суммы. В тех случаях, когда происходит циклический перенос (из старшего разряда сумматора в его младший разряд) внизу под чертой приведена сумма кодов слагаемых. Расположенная под этой суммой стрелка изображает циклический перенос. Ниже стрелки записан код суммы, получаемый на выходе сумматора.

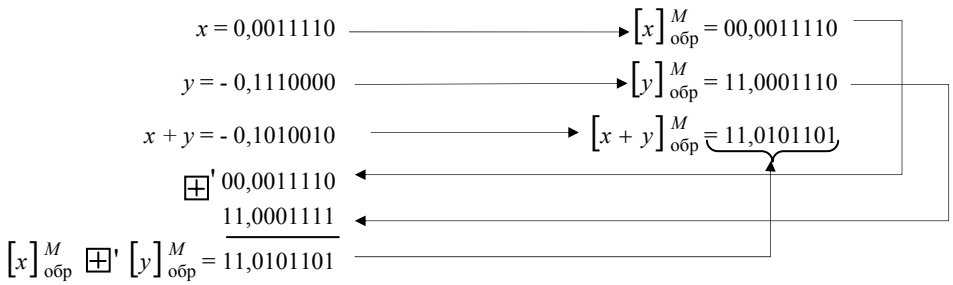
Пример 10.



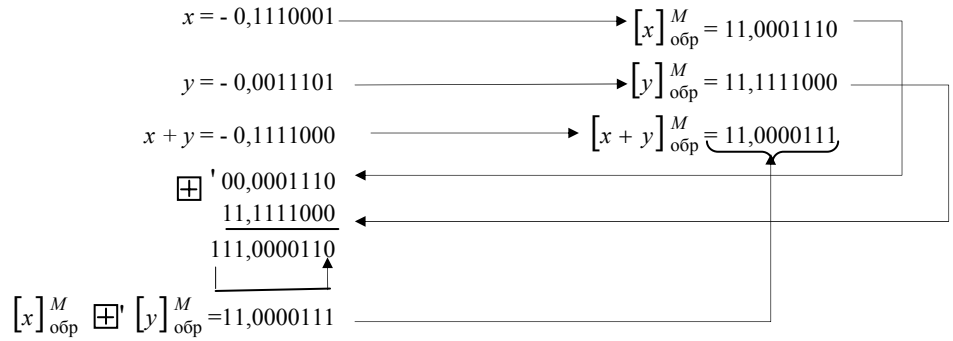

Пример 11.




Пример 12.

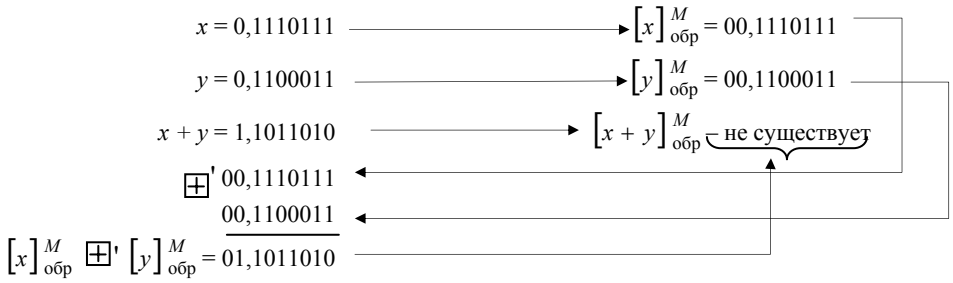



Пример 13.

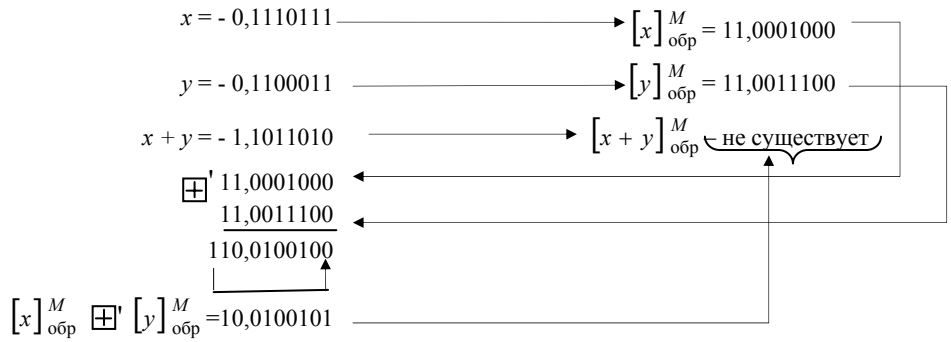



При сложении модифицированных обратных кодов чисел $x$ и $y$, у которых $|x + y| \geq 1$, происходит переполнение разрядной сетки сумматора с циклическим переносом. Переполнение может наступать лишь в случае, когда слагаемые $x$ и $y$ имеют одинаковые знаки. Для пояснения приведем два примера.

Пример 14. Оба слагаемые положительны.



П р и м е р 15. Если числа отрицательны, то переполнение разрядной сетки произойдет следующим образом:




Контроль за переполнением разрядной сетки при сложении чисел в модифицированном обратном коде (с помощью сумматора **с** циклическим переносом) ничем не отличается от контроля при сложении чисел в модифицированном дополнительном коде (с помощью сумматора без переноса из старшего разряда).

Вычитание чисел в модифицированном обратном коде сводится к сложению кода уменьшаемого с кодом вычитаемого, взятого с обратным знаком. В результате на выходе сумматора получается модифицированный обратный код разности.

## § 10. Сложение и вычитание нормализованных чисел. Умножение и деление в машинах с фиксированной и с плавающей запятой

**1. Сложение и вычитание чисел в машинах с плавающей запятой.** Сложение (и вычитание) чисел в машинах с запятой, фиксированной после нулевого разряда, производится в дополнительном, обратном, модифицированном дополнительном или модифицированном обратном коде (в зависимости от конструкции машины) так, как это было описано в предыдущих параграфах.

Остановимся особо на том, как выполняется сложение (и вычитание) чисел в машинах с плавающей запятой.

В этих машинах мантиссы чисел складываются в одном из модифицированных кодов. Для определенности остановимся на модифицированном обратном коде.

Операция сложения выполняется следующим образом: сначала уравниваются порядки слагаемых, т. е. то из слагаемых, порядок которого меньше, денормализуется так, чтобы его порядок стал равен порядку второго слагаемого. После этого мантиссы обоих слагаемых переводятся в модифицированный обратный код и складываются в сумматоре с циклическим переносом. При этом возможны три случая.

1. Сложение пройдет без переполнения разрядной сетки и нарушения нормализации. В этом случае результат переводится из модифицированного обратного кода в прямой код. Результат в качестве мантиссы переносится в ответную ячейку. Порядком результата является общий порядок обоих слагаемых.

2. Сложение пройдет без переполнения разрядной сетки сумматора, но результат после перевода в прямой код окажется ненормализованным. Это — так называемое нарушение нормализации вправо. Перед записью в ячейку производится нормализация результата.

3. При сложении происходит переполнение разрядной сетки сумматора. Если слагаемые положительны, то в нулевом и первом разрядах сумматора на выходе образуется комбинация цифр 01, если же слагаемые отрицательны, то комбинация цифр 10 (см. § 9). Такое переполнение разрядной сетки называют нарушением нормализации влево. Легко понять, что не произошло бы переполнения, если бы до перевода мантисс в модифицированный обратный код обе мантиссы были умножены на $10^{-1}$ (т. е. на $^1/_2$), а оба порядка увеличены на единицу. Однако нет надобности заставлять машину заново переделывать все выполненные ею действия, так как на выходе нулевого разряда сумматора получилась цифра, характеризующая знак суммы, а на выходе первого разряда сумматора получилась первая цифра суммы. Достаточно, не трогая нулевого разряда, передвинуть все полученные цифры на один разряд вправо, после чего цифру, полученную в нулевом разряде, занести в первый разряд. Теперь полученный результат переводится из модифицированного обратного кода в прямой и в качестве мантиссы переносится в ответную ячейку. Порядок суммы оказывается на единицу больше порядка каждого из слагаемых.

Поясним примерами. Все записи будем вести в двоичной системе счисления.

П р и м е р 1. При сложении не происходит нарушения нормализации.

Первое слагаемое записано в ячейке так:

(I) 0 11000011 0 0100.

Второе слагаемое

(II) 0 10001011 0 000,1.

П е р в ы й шаг. Денормализация второго слагаемого с тем, чтобы его порядок стал равен числу 100 (уравнивание порядков)

(II) 0 00010001 0 0100.

В т о р о й шаг. Перевод обеих мантисс в модифицированный обратный код

(I) 00 11000011,

(II) 00 00010001.

Т р е т и й шаг. Сложение мантисс

(I) ⊞ ' 00 11000011

(II) 00 00010001

00 11010100

Ч е т в е р т ы й шаг. Перевод результата в прямой код

0 11010100.

Запись ответа в ответную ячейку

0 11010100 0 0100.

П р и м е р 2. При сложении происходит нарушение нормализации вправо. Первое слагаемое представлено в ячейке так:

(I) 0 11000001 0 0011,

т. е. первое слагаемое равно числу +0, 11000001 • $10^{+11}$.

Второе слагаемое записано в ячейке следующим образом:

(II) 1 10000110 0 0110.

Оно равно числу —0,10000110• $10^{+110}$.

П е р в ы й шаг. Денормализация первого слагаемого с тем, чтобы его порядок стал равен числу ПО (уравнение порядков)

(I) 0 00011000 0 0110.

В т о р о й шаг. Перевод обеих мантисс в модифицированный обратный код

(I) 00 00011000;

(II) 11 01111001.

Т р е т и й шаг. Сложение мантисс

(I) ⊞ ' 00 00011000

(II) 11 01111001

11 10010001

Ч е т в е р т ы й шаг. Перевод результата в прямой код

1 01101110.

П я т ы й шаг. Нормализация результата

1 11011100,

при этом порядок вместо 110 становится 101.

Ш е с т о й шаг. Запись ответа в ответную ячейку

1 11011100 0 0101.

В ответе получим число

- 0,110 11100 • $10^{+101}$ = - 11011,1.

П р и м е р 3. При сложении происходит нарушение нормализации влево.

Первое слагаемое

(I) 1 11100111 0 1000.

Второе слагаемое

(II) 1 10100000 0 0111.

П е р в ы й шаг. Денормализация второго слагаемого с тем, чтобы его порядок стал равен числу 1000 (уравнивание порядков)

(II) 1 01010000 0 1000.

В т о р о й шаг. Перевод мантисс в модифицированный обратный код

(I) 11 00011000,

(II) 11 10101111.

Третий шаг. Сложение мантисс

(I) ⊞ ' 11 00011000

(II) 11 10101111

110 11000111

10 11001000

Ч е т в е р т ы й шаг. Исправление денормализации влево

11 01100100.

П я т ы й шаг. Переход к прямому коду

1 10011011.

Ш е с т о й шаг. Запись ответа в ответную ячейку

1 10011011 0 1001.

**2. Умножение и деление чисел в машинах.** Умножение чисел как в машинах с плавающей запятой, так и в машинах с фиксированной после нулевого разряда запятой производится в прямом коде.

Сначала содержимое нулевых разрядов (знаки чисел) складываются на одноразрядном сумматоре. При таком сложении перенос, если он возникает, теряется. В результате произведению приписывается знак в соответствии с известным правилом алгебры (см. таблицу 8).

Т а б л и ц а 8. Получение знака произведения

| Сложение нулевых разрядов | Правило знаков при умножении |
|---|---|
| 0 + 0 = 0 | + · + = + |
| 0 + 1 = 1 | + · - = - |
| 1 + 0 = 1 | - · + = - |
| 1 + 1 = 10 ~ 0 | - · - = + |

Затем производится перемножение самих чисел.

Умножение чисел в машинах с фиксированной запятой и умножение мантисс в машинах с плавающей запятой производится одинаково.

В двоичной системе счисления (и в прямом коде) числа изображаются с помощью только двух цифр: нулей и единиц. Поэтому умножение становится особенно простым. Первый сомножитель приходится умножать только на нули или единицы, являющиеся цифрами второго сомножителя. Получаемые частичные произведения следует подписывать друг под другом, сдвигая каждое новое произведение на один разряд. Частичные произведения, равные нулю, естественно, можно опускать. Благодаря этому умножение в машине можно осуществить путем последовательных сдвигов одного из сомножителей на количество разрядов, определяемое цифрами второго сомножителя, и сложения между собой получающихся результатов. В некоторых машинах начинается умножение первого сомножителя не на младший, а на старший разряд второго сомножителя с последовательным переходом от разряда к разряду второго сомножителя слева направо. Поскольку обычное умножение общеизвестно, ограничимся тем, что приведем пример на умножение, начинающееся умножением первого сомножителя на старший разряд второго сомножителя.

П р и м е р 4.

| | | | |
|---|---|---|---|
| | 0, 11001101 | | |
| | 0, 11101001 | × | |
| | 01100110 | 1 | – результат первого сдвига |
| | 00110011 | 01 | – результат второго сдвига |
| + | 00011001 | 101 | и т.д. |
| | 00000110 | 01101 | |
| | 00000000 | 11001101 | |
| | 0, 10111010 | 10010101 | |
| Округление: | 0, 10111011 | | |

Кроме того, в машине с плавающей запятой производится алгебраическое сложение порядков сомножителей, дающее порядок произведения. Если после умножения в ячейке машины с плавающей запятой результат получится ненормализованным, то он (до округления) подвергается нормализации.

При делении знак частного получается путем суммирования цифр, изображающих знаки делимого и делителя на одноразрядном сумматоре, так же как при умножении.

Деление чисел в машинах с фиксированной запятой и деление мантисс в машинах с плавающей запятой производятся одинаково. Оно осуществляется путем вычитания из делимого делителя, сдвига влево полученного остатка, нового вычитания (из результата сдвига) делителя, и т. д. Таким образом, деление в машине производится примерно так же, как оно выполняется в двоичной системе счисления карандашом на бумаге (см. пример на деление, § 4). Применяемый код — прямой.

Кроме того, в машине с плавающей запятой производится алгебраическое вычитание порядка делителя из порядка делимого, что дает порядок частного. Если в ячейке машины с плавающей запятой частное окажется ненормализованным, то производится его нормализация.

Во многих машинах операция деления $a : b$ заменена двумя операциями: $\left(\frac{1}{b}\right)a$. В таких машинах по величине $b$

вычисляется величина $\frac{1}{b}$ с помощью специальной стандартной подпрограммы одним из численных методов. Этим достигается упрощение конструкции арифметического устройства машины. Частное $a : b$ получается уже путем умножения числа $\frac{1}{b}$ на величину $a$.

# ГЛАВА II
# МАТЕМАТИЧЕСКАЯ ЛОГИКА И ПОСТРОЕНИЕ СХЕМ ЭЛЕКТРОННЫХ ЦИФРОВЫХ МАШИН

## § 11. Начальные сведения из алгебры логики

**1. Понятие высказывания и его значения истинности.** *Логика* — это наука о формах и законах мышления. *Математическая логика* — одна из ветвей общей логики, развивающаяся применительно к потребностям математики. Математическая логика оправдывает свое название не только потому, что она выросла из потребностей математики. Она сама строится как типичная математическая дисциплина и рассматривается не только как логика математики, но и как математика логики. В значительной мере она является результатом применения математических методов к проблемам формальной логики.

В математической логике используется тот же язык формул, который издавна применяется в математике; это освобождает математическую логику от неопределенности в толковании логических выражений, показывающих связи между суждениями, понятиями и т.д. Получение логических следствий из исходных посылок осуществляется путем формальных преобразований логических формул по правилам, аналогичным правилам алгебраических действий.

*Алгебра логики,* называемая также *исчислением высказываний,* представляет собой первую необходимую часть математической логики. Под *высказыванием* мы будем понимать всякое предложение, в отношении которого имеет смысл утверждение о его истинности или ложности, например: мир — материален; сегодня-—двенадцатое число; снег — красен; 9 — нечетное число. Из самого определения высказывания вытекает, что оно может быть либо *истинным,* либо *ложным.* Высказываний одновременно истинных и ложных или неистинных и неложных не существует. Отдельные высказывания будем обозначать заглавными буквами латинского алфавита $A$, $B$, $C$,...

В алгебре логики два высказывания считаются *различными,* если они имеют различное содержание. В этом случае они обозначаются различными буквами. Однако содержание высказываний учитывается только при введении их буквенного обозначения и в дальнейших рассмотрениях уже не принимается во внимание. Высказывания оцениваются только по их истинности или ложности (без учета их конкретного содержания). Считают, что значение истинности высказывания равно единице, если это высказывание истинно, и равно нулю, если оно ложно. Значение истинности каждого высказывания мы будем обозначать той же буквой, которой обозначаем само высказывание.

Два высказывания называют *эквивалентными,* если значения истинности их одинаковы. Эквивалентность двух высказываний обозначают знаком равенства. Таким образом, запись $A = B$ означает, что значения истинности высказываний $A$ и $B$ одинаковы, т. е. что они одновременно либо истинны, либо ложны. Запись $A = 1$ означает, что высказывание $A$ истинно. Запись $C = 0$ означает, что высказывание $C$ ложно.

Каждое конкретное высказывание имеет вполне определенное значение истинности. Но это значение истинности может быть и переменным. Например, значение истинности высказывания *«Сегодня — двенадцатое число»* равно единице лишь в течение двенадцати дней в году. В течение всех остальных дней года оно равно нулю.

Часто употребляют выражение: «рассмотрим некоторое высказывание Л». В этом случае говорят о произвольном высказывании, о «высказывании вообще». Значение истинности «высказывания вообще» тоже нужно считать переменным. Оно равно нулю или единице в зависимости от того, какое конкретное высказывание мы будем отождествлять с этим «высказыванием вообще».

Переменная величина, которая принимает лишь два значения (О или 1), называется *двоичной переменной.*

В дальнейшем мы встретимся с высказываниями, значения истинности которых определяются значениями истинности других высказываний, т. е. являются их функциями. Функции, которые принимают лишь два значения (0 или 1) и зависят от одной или нескольких двоичных переменных, называются *двоичными функциями.*

Аппарат математической логики, в частности алгебра логики, находит широкое применение в теории электронных цифровых машин.

В электронных цифровых машинах, работающих с числами, представленными в двоичной системе счисления или в различных двоично-кодированных системах счисления, используют в качестве простейших элементов различные двухпозиционные приборы, т. е. приборы или схемы, имеющие только два различных устойчивых состояния. Для электронных элементов, применяемых в машинах, одно из состояний характеризуется наличием на выходе элемента высокого уровня напряжения, а второе состояние — наличием низкого уровня напряжения (условились считать, что одно состояние обозначает двоичную единицу, а другое — двоичный нуль). Эти элементы могут служить для представления логических переменных — значений истинности высказываний. Одно состояние двухпозиционного элемента соответствует значению истинности, равному 1, а второе состояние — равному 0.

Такое соответствие между логическими высказываниями в математической логике, двоичными цифрами в

двоичной системе счисления и работой двух позиционных элементов и схем в электронных цифровых машинах позволяет с помощью алгебро-логической символики удобно описывать с функциональной точки зрения работу схем и блоков машин и осуществлять их анализ и синтез. Отдельные элементы и схемы, входящие в состав машины, представляют собой, по существу, устройства, реализующие различные логические высказывания. Естественно, что при описании работы реальных схем допускается их идеализация; например, не учитываются переходные процессы в схемах, т. е. допускается, что необходимые уровни напряжений устанавливаются и передаются по проводам мгновенно и могут изменяться только в заданные дискретные моменты времени ($\tau = 0, 1,2,3,...$).

Два состояния любой схемы, 0 и 1, являются взаимно дополнительными (т. е. обязательно имеет место одно из них). Это положение соответствует *закону исключенного третьего* в математической логике, согласно которому любое высказывание может быть либо ложным, либо истинным.

Аппарат математической логики в применении к теории электронных цифровых машин позволяет решать следующие основные задачи.

З а д а ч а  а н а л и з а  с х е м. Имея какую-то готовую схему, описать ее работу логическим выражением. Затем путем формальных преобразований полученного логического выражения, проанализировать вопрос об экономичности схемы, т. е. выяснить, нельзя ли получить более простую схему, содержащую меньшее количество элементов, которая бы выполняла заданные функции.

З а д а ч а  с и н т е з а  с х е м. Имея логическое выражение, описывающее некоторую логическую функцию, определить, из каких элементарных схем и каким образом должна быть построена сложная схема, реализующая заданную функцию. Для этой цели необходимо исходное логическое выражение рациональным образом преобразовать и расчленить на отдельные члены так, чтобы каждый из членов мог быть представлен элементарной схемой.

Помимо задач анализа и синтеза схем электронных цифровых машин, математическая логика находит применения в теории этих машин в виде так называемых логических схем и для точного и однозначного описания программ работы машин, формулировки логических условий. Сюда же непосредственно примыкают задачи, связанные с выработкой и оценкой оптимальных машинных алгоритмов решения задач.

И наконец, электронные цифровые программно-управляемые машины, представляющие собой устройства для логической переработки информации, позволяют решать формальные задачи математической логики и, в частности, задачи анализа и синтеза релейно-контактных схем и схем электронных цифровых машин.

Из сказанного видна тесная непосредственная связь между математической логикой и теорией электронных цифровых машин. По существу математическая логика представляет собой основу теории электронных цифровых машин, и знание элементов математической логики является необходимым как для работы в области проектирования машин, так и для работы в области программирования задач для электронных цифровых машин.

**2. Сложные высказывания. Логические связи. Логические операции.** Из одного или нескольких высказываний, принимаемых за простые, можно составлять *сложные высказывания.* Объединение простых высказываний в сложные в исчислении высказываний производится без учета внутреннего содержания (смысла) этих высказываний. Для объединения простых высказываний в сложные применяются знаки логических связей. Приведем несколько основных видов сложных высказываний и логических связей.

1. О т р и ц а н и е  в ы с к а з ы в а н и я  *A*. Обозначается символом $\overline{A}$. Читается: *не A*. Знак логической связи — читается как *не. Отрицанием* высказывания *A* называется сложное высказывание $\overline{A}$, которое истинно, когда *A* ложно, и ложно, когда *A* истинно. Значение истинности отрицания высказывания по значению истинности основного высказывания получается с помощью операции отрицания, определяемой таблицей 9.

Т а б л и ц а 9.
Отрицание

| |
|---|
| $\overline{0} = 1$ |
| $\overline{1} = 0$ |

Из этой таблицы видно, что

$$\overline{\overline{A}} = A, \qquad \text{(II.1)}$$

если под символом $\overline{\overline{A}}$ понимать ($\overline{\overline{A}}$). Действительно,

$$\overline{\overline{0}} = (\overline{\overline{0}}) = \overline{\overline{1}} =:0; \qquad \overline{\overline{1}} =(\overline{\overline{1}}) = \overline{\overline{0}} = 1.$$

2. К о н ъ ю н к ц и я  д в у х  в ы с к а з ы в а н и й. Обозначается символом *A*/*B*. Читается: *A и B*. Знак логической связи /\ имеет смысл союза *и. Конъюнкция* двух высказывании представляет собой сложное высказывание, которое истинно лишь в случае истинности обоих высказываний, его образующих. Во всех остальных случаях это сложное высказывание ложно.

Значение истинности конъюнкции *A*/*B* по значениям истинности составляющих ее высказываний *A* и *B* получим с помощью операции, называемой *логическим умножением* и определяемой таблицей 10.

Из определения конъюнкции, а также из таблицы 10 вытекает*), что

*) Справедливость формул (II.3) и (II.4) проверяется путем подстановки в них вместо *A* и *B* в различных комбинациях значений 0 и 1 и применения таблицы 10.

$$\left.\begin{aligned} A \wedge 0 &= 0, \\ A \wedge 1 &= A, \\ A \wedge A &= A, \\ A \wedge \overline{A} &= 0; \end{aligned}\right\} \qquad \text{(II.2)}$$

$$A \wedge B = B \wedge A; \qquad \text{(II.3)}$$

$$A \wedge (B \wedge C) = (A \wedge B) \wedge C \qquad \text{(II.4)}$$

Равенство (II.3) означает, что для логического умножения справедлив переместительный закон. Равенство (II.4) свидетельствует о справедливости для логического умножения сочетательного закона.

Таблица 10
Логическое умножение

| |
|---|
| $0 \wedge 0 = 0$ |
| $0 \wedge 1 = 0$ |
| $1 \wedge 0 = 0$ |
| $1 \wedge 1 = 1$ |

Таблица 11
Логическое сложение

| |
|---|
| $0 \vee 0 = 0$ |
| $0 \vee 1 = 1$ |
| $1 \vee 0 = 1$ |
| $1 \vee 1 = 1$ |

3. Дизъюнкция двух высказываний. Обозначается символом $A \vee B$. Читается: *A или B*. Знак логической связи $\vee$ имеет смысл союза *или.* В русском языке союз *или* употребляется в нескольких различных смыслах, поэтому здесь требуется уточнение. Знак $\vee$ имеет смысл *или,* употребляемого во фразе: «При звоне будильника Петр или Иван проснется» (здесь *или* не исключает возможности того, что проснутся оба). Существует еще исключающее *или,* например во фразе: «Выбирай: он или я». Исключающее *или* может быть также принято за один из видов логической связи, но его не следует смешивать с *или,* применяемым в дизъюнкции.

*Дизъюнкция* представляет собой сложное высказывание, которое ложно только в случае ложности обоих составляющих его высказываний и истинно в остальных случаях.

Значение истинности высказываний $A \vee B$ находят по значениям истинности составляющих его высказываний *A* и *B* с помощью операции, называемой *логическим сложением* и определяемой таблицей 11.

Из определения дизъюнкции и из таблицы 11 видна справедливость следующих формул*):

$$A \vee 0 = A,$$
$$A \vee 1 = 1,$$
$$A \vee A = A,$$
$$A \vee \overline{A} = 1;$$
$$A \vee B = B \vee A; \qquad \text{(II.5)}$$
$$A \vee (B \vee C) = (A \vee B) \vee C. \qquad \text{(II.6)}$$

Формула (II.5) означает, что для логического сложения справедлив переместительный закон. Формула (II.6) показывает справедливость для этой операции сочетательного закона.

4. Равнозначность двух высказываний. Записывается так: $A \sim B$. Читается: *A равнозначно B. Равнозначность* двух высказываний представляет собой сложное высказывание, истинное тогда, когда значения истинности составляющих высказываний одинаковы, и ложное в противном случае.

Значение истинности сложного высказывания $A \sim B$ получается по значениям истинности составляющих высказываний *A* и *B с* помощью логической операции, определяемой таблицей 12.

Таблица 12
Операция равнозначности

| |
|---|
| $0 \sim 0 = 0$ |
| $0 \sim 1 = 0$ |
| $1 \sim 0 = 0$ |
| $1 \sim 1 = 1$ |

Таблица 13
Отрицание равнозначности

| |
|---|
| $0 \not\sim 0 = 0$ |
| $0 \not\sim 1 = 1$ |
| $1 \not\sim 0 = 1$ |
| $1 \not\sim 1 = 1$ |

Путем непосредственной проверки потаблице 12 убеждаемся в справедливости формул

$$A \sim 1 = A; \qquad \text{(II.7)}$$

$$A \sim 0 = \overline{A}. \qquad \text{(II.8)}$$

5. Отрицание равнозначности двух высказываний. *Отрицание равнозначности* двух

*) Справедливость формул (II.5) и (II.6) проверяется путем подстановки в них вместо *A* и *B* в различных комбинациях значений 0 и 1 и применения таблицы 11

высказываний представляет собой более сложное высказывание, получаемое при помощи двух ранее описанных логических связей: $\overline{A \sim B}$. Нам удобно ввести специальный знак для изображения отрицания равнозначности. Введем для этого знак $\approx$ (читается: *неравнозначно)*. Итак, $\overline{A \sim B} = A \approx B$.

С помощью рассмотренных выше таблиц 9 и 12 получаем таблицу 13. Эта таблица определяет логическую операцию отрицания равнозначности, позволяющую по значениям истинности составляющих высказываний $A$ и $B$ найти значение истинности сложного высказывания $A \approx B$. Операция отрицания равнозначности используется в машинах как поразрядная операция, с помощью которой сравниваются числа.

Из таблицы 13 вытекает справедливость формулы

$$A \approx B = B \approx A. \qquad \text{(II.9)}$$

Легко видеть, что

$$A \approx 1 = \overline{A}. \qquad \text{(II.10)}$$

Путем непосредственной проверки по ранее приведенным таблицам логических операций убеждаемся также в справедливости следующей формулы:

$$A \sim B = (A \approx B) \approx 1. \qquad \text{(II.11)}$$

Путем непосредственной проверки можно убедиться в справедливости формулы

$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C), \qquad \text{(II.12)}$$

означающей, что для логического умножения справедлив (по отношению к логическому сложению) распределительный закон, подобно тому как в алгебре имеет место этот закон для обычного умножения (по отношению к обычному сложению):

$$a(b+c) = ab+ac. \qquad \text{(II.13)}$$

В алгебре логики имеет место еще второй распределительный закон для логического сложения (по отношению к логическому умножению):

$$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C) \qquad \text{(II.14)}$$

Справедливость этой формулы доказывается путем непосредственной подстановки вместо *A, B, C* всевозможных комбинаций из нулей и единиц: 0, 0, 0; 0, 0, 1; ...; 1, 1, 1.

Аналогом формулы (П.14) в обычной алгебре была бы формула

$$a + (bc) = (a+b)(a + c). \qquad \text{(II.15)}$$

Как известно, такая формула неверна.

В силу справедливости для логического умножения и логического сложения переместительного и сочетательного законов многочленные конъюнкции и дизъюнкции можно писать без скобок. Например, вместо $[(A \wedge B) \wedge C] \wedge D$ или вместо $A \wedge [B \wedge (C \wedge D)]$ можно просто писать $A \wedge B \wedge C \wedge D$. Точно так же вместо $[(A \vee B) \vee C] \vee D$ , или $A \vee [B \vee (C \vee D)]$. можно писать $A \vee B \vee C \vee D$.

Для дальнейшего уменьшения количества скобок в логических формулах принимают соглашение считать связь с помощью знака $\wedge$ более тесной, чем с помощью знака $\vee$, а последнюю — более тесной, чем связь с помощью знаков $\sim$ и $\approx$. Этим устанавливается правило записи выражений, аналогичное принятому в алгебре (в алгебре говорят, что умножение и деление являются «старшими» действиями по отношению к сложению и вычитанию; в процессе вычислений старшие действия выполняются раньше младших). Это соглашение позволяет вместо $(A \wedge B) \vee C$ писать просто $A \wedge B \vee C$ (таким образом, логическое умножение принимается старшим по отношению к логическому сложению). Однако для большей наглядности формул мы в дальнейшем в сложных выражениях часто будем употреблять скобки.

6. И м п л и к а ц и я д в у х в ы с к а з ы в а н и й . Обозначается символом $A \to B$. Читается: *если А, то В. Импликация* двух высказываний представляет собой сложное высказывание, которое ложно в том и только том случае, когда $A$ истинно, а $B$ ложно.

Следует заметить, что импликация не имеет смысла связи между причиной и следствием, т. е. из истинности $A$ еще не следует обязательно истинность $B$. Наоборот, для истинности сложного высказывания, образованного импликацией, достаточно уже ложности высказывания $A$.

Значения истинности импликации определяются таблицей 14.

Т а б л и ц а 14
Импликация

| |
|---|
| $0 \to 0 = 1$ |
| $0 \to 1 = 1$ |
| $1 \to 0 = 0$ |
| $1 \to 1 = 1$ |

Т а б л и ц а 15
Операция Шеффера

| |
|---|
| $0 / 0 = 1$ |
| $0 / 1 = 1$ |
| $1 / 0 = 1$ |
| $1 / 1 = 0$ |

7. О п е р а ц и я Ш е ф ф е р а ( н е с о в м е с т н о с т ь д в у х в ы с к а з ы в а н и й ) . *Связь Шеффера* двух высказываний $A$ и $B$ обозначается $A/B$ и представляет собой сложное высказывание, которое ложно в том и только том случае, когда оба составляющих высказывания истинны.

Значения истинности связи Шеффера определяются таблицей 15.

Логическая связь Шеффера играет важную роль в теории электронных цифровых машин и в теории логических схем. Все другие логические связи могут быть выражены через связь Шеффера, и таким образом электронная схема, реализующая связь Шеффера, является универсальным функциональным элементом, при помощи которого в принципе могут быть построены любые функциональные схемы как счетных, так и управляющих блоков машины.

**3. Геометрическое толкование алгебро-логических операций.** Не вдаваясь в логические тонкости, приведем простой и наглядный способ геометрического (вернее, теоретико-множественного) пояснения алгебро-логических операций.

Пусть дано некоторое высказывание $A$. Существует некоторое множество объектов, к которым это высказывание можно применять, причем, применяя это высказывание к каждому из упомянутых объектов, можно узнать, является ли оно для такого объекта истинным (равно 1) или ложным (равно 0). Объекты, для которых $A$ не равно ни 0 ни 1 или одновременно равно и 0, и 1, в наше множество не должны быть включены.

Например, для высказывания *«Позвоночное животное является хищником»* таким множеством объектов будет совокупность позвоночных животных. Применяя это высказывание к конкретному позвоночному животному, мы можем установить соответствующее значение истинности нашего высказывания. Корова является одним из объектов множества, для нее истинность высказывания равна 0. Волк также является одним из объектов, к которым приложимо высказывание; для этого объекта значение истинности высказывания равно 1.

Для высказывания *«Число x имеет синус меньший, чем* $^1/_2$*»* (т. е. $\sin x < {}^1/_2$) упомянутым множеством объектов является множество всех действительных чисел. Для $x$=1 значение истинности высказывания равно нулю (ибо $\sin 1 > {}^1/_2$ ), а для $x$ = 0,1 значение истинности высказывания равно 1 (ибо $\sin 0{,}1 < {}^1/_2$).

Для высказывания *«Некоторая точка плоскости удалена от точки С той же плоскости меньше чем на* 2» множеством объектов, к которым приложимо высказывание, является множество всех точек плоскости. Высказывание имеет значение истинности 1 для всех точек, лежащих внутри круга радиуса 2 с центром в точке $C$. Для точек, лежащих на окружности, ограничивающей упомянутый круг, и вне этого круга, значение истинности высказывания равно 0.

Представим себе, что каждый объект, к которому можно применять высказывание $A$, изображается некоторой точкой*) плоскости $x0y$ (т. е. поставлен во взаимно однозначное соответствие некоторой точке). Для большей наглядности ограничимся случаем, когда все указанные точки заполняют некоторый квадрат $Q$. Точки квадрата $Q$ разобьем на два класса. К одному классу отнесем те точки, для которых $A$= 1, а к другому — те, для которых $A$ = 0. Ясно, что каждая точка, принадлежащая квадрату $Q$, будет обязательно принадлежать одному из этих классов, и при этом только одному из них.

Множество точек квадрата $Q$, принадлежащих первому классу (для точек которого $A$ = 1), будем считать геометрическим изображением высказывания $A$ и для удобства будем называть множеством $A$. При этом может получиться, например, картина, приведенная на рис. 19. Высказывание $A$ изображено на этом рисунке в виде некоторой области, ограниченной замкнутым контуром. Очевидно, высказывание $\overline{A}$ (не $A$) будет тогда изображаться множеством всех остальных точек квадрата $Q$.

При таком изображении высказываний конъюнкция двух высказываний будет представляться пересечением двух множеств (рис. 20). Действительно, $A \wedge B = 1$ только тогда, когда $A = 1$ и $B = 1$, а это имеет место лишь для точек, одновременно принадлежащих множеству $A$ и множеству $B$ (их пересечению). В теории множеств такое множество обозначают символом $A \cdot B$.

Дизъюнкция двух высказываний $A \vee B$ будет изображаться множеством, которое получается путем объединения множеств $A$ и $B$ (рис. 21). В теории множеств такое множество обозначается символом $A + B$.

Высказывание $A \sim B$ ($A$ равнозначно $B$) изобразится так, как показано на рис. 22, ибо истинность $A \sim B$ равна 1 либо при $A = 1$, $B = 1$, либо при $A = 0$, B = 0. Из рис. 22, между прочим, можно усмотреть, что $A \sim B = \overline{A} \wedge \overline{B} \vee A \wedge B$; в справедливости этой формулы можно

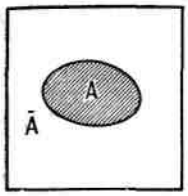



Рис. 19. $A$ изображено заштрихованной областью.

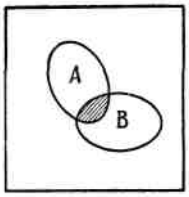



Рис. 20. $A \wedge B$ изображено заштрихованным множеством.

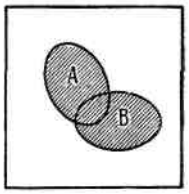



Рис. 21. $A \vee B$ изображено заштрихованным множеством.

легко убедиться путем проверки с помощью таблиц логических операций. Изображение высказывания $A / B$ приведено на рис. 23.

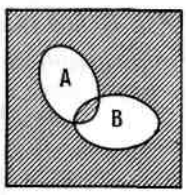



Рис. 22. $A \sim B$ изображено заштрихованным множеством.

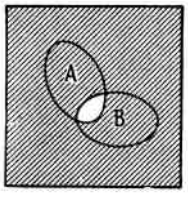



Рис. 23. $A/B$ изображено заштрихованным множеством.

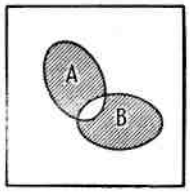



Рис. 24. $A \approx B$ изображено заштрихованным множеством.

*) Мы ограничимся случаем, когда множество объектов, к которым приложимо высказывание $A$, имеет мощность не большую, чем мощность континуума

Высказывание $A \approx B$ показано на рис. 24. Его изображение без труда получаем, учитывая, что $A \approx B = \overline{A \sim B}$.

**4. Связи между логическими операциями.** Приведенные выше логические операции не являются независимыми и могут выражаться друг через друга. Так, связь ~ (равнозначность), а значит, и связь $\approx$ (неравнозначность) можно представить с помощью связей $\bar{}$ *(не)*, $\wedge$ *(и)* и $\vee$ *(или)*. Предоставляем читателю путем непосредственной подстановки (с применением вышеприведенных таблиц логических операций) убедиться в справедливости следующих эквивалентностей:

$$A \sim B = (A \vee B) \wedge (A \vee B), \tag{II.16}$$

$$A \approx B = (A \wedge B) \vee (A \wedge B). \tag{II.17}$$

Отметим, кроме того, связь, существующую между конъюнкциями и дизъюнкциями:

$$\overline{A \wedge B} = \overline{A} \vee \overline{B}, \tag{II.18}$$

$$\overline{A \vee B} = \overline{A} \wedge \overline{B}. \tag{II.19}$$

Проверку справедливости этих связей легко осуществить с помощью таблицы 16.

Таблица16.Сопоставление значений логических функций

| $A$ | $B$ | $\overline{A}$ | $\overline{B}$ | $A \wedge B$ | $\overline{A \wedge B}$ | $\overline{A} \vee \overline{B}$ | $A \vee B$ | $\overline{A \vee B}$ | $\overline{A} \wedge \overline{B}$ | $A/B$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

Далее, импликация может быть выражена через конъюнкцию и дизъюнкцию следующим образом:

$$A \rightarrow B = \overline{A \wedge B}. \tag{II.20}$$

Справедливость этой формулы вытекает из того, что при истинности $A \rightarrow B$ одновременно не может быть $A$ истинным, а $B$ ложным. Но так как $\overline{\overline{B}} = B$ и согласно (II.18) $\overline{A \wedge \overline{B}} = \overline{A} \vee \overline{\overline{B}}$, то получим:

$$A \rightarrow B = \overline{A} \vee B. \tag{II.21}$$

Также справедливо выражение

$$A \vee B = \overline{A} \rightarrow B, \tag{II.22}$$

в чем можно убедиться простой подстановкой значений истинности согласно таблицам 9, 11 и 14.

Таким образом, видно, что все логические связи можно выразить при помощи трех основных связей: отрицания, дизъюнкции и конъюнкции. Эти последние сами в свою очередь могут быть выражены при помощи одной только связи Шеффера. Убедимся в этом.

Для связи Шеффера справедливо соотношение

$$A/A = \overline{A}. \tag{II.23}$$

Это видно из таблиц 9 и 15. Формула (II.23) дает выражение отрицания через связь Шеффера. Далее, связь Шеффера представляет собой не что иное, как отрицание конъюнкции:

$$A/B = \overline{A \wedge B}. \tag{II.24}$$

В этом легко убедиться с помощью таблицы 16.

Отрицая обе части равенства (II.24), получим:

$$A \wedge B = \overline{A / B}.$$

Применяя формулу (II.23), получим:

$$A \wedge B = (A/B) / (A/B). \tag{II.25}$$

Формула (II.23) дает выражение конъюнкции через функцию Шеффера.

Из формул (II.24) и (II.18) получим:

$$A/B = \overline{A} \vee \overline{B}. \tag{II.26}$$

Если в равенстве везде заменить основные высказывания их отрицаниями, то эквивалентность от этого не нарушится. Следовательно,

$$\overline{A} / \overline{B} = A \vee B. \tag{II.27}$$

И наконец, используя равенство (II.24), получим выражение дизъюнкции через связь Шеффера:

$$A \vee B = (A/A)/(B/B). \tag{II.28}$$

Аналогичным путем можно вывести выражения через связь Шеффера для всех остальных логических связей.

## § 12. Преобразование логических выражений

**1. Нормальные формы логических выражений.** Наиболее наглядно структура логического выражения видна тогда, когда оно приведено к одной из так называемых *нормальных форм.* Существуют две нормальные формы логических выражений. Первая — *конъюнктивная нормальная форма* — представляет собой некоторую конъюнкцию дизъюнкций, причем в каждой дизъюнкции отдельные члены представляют собой либо основные высказывания, либо их отрицания. Вторая — *дизъюнктивная нормальная форма* — представляет собой некоторую дизъюнкцию конъюнкций; в каждой конъюнкции отдельные члены являются либо основными высказываниями, либо их отрицаниями.

Преобразование логических выражений к той или иной нормальной форме осуществляется путем применения следующих основных правил (эти правила являются некоторым повторением того, что было изложено в предыдущих параграфах):

1) со знаками $\wedge$ и $\vee$ можно оперировать так же, как в алгебре оперируют со знаками $\times$ (умножения) и $+$ (сложения), пользуясь переместительным, сочетательным и распределительным законами;

2) выражения с двойным (и вообще четным) количеством отрицаний можно заменять на основные выражения:

$$\overline{\overline{A}} = A; \qquad \text{(II.29)}$$

3) отрицание конъюнкции можно заменить дизъюнкцией отрицаний, а отрицание дизъюнкции — конъюнкцией отрицаний:

$$\overline{A \wedge B} = \overline{A} \vee \overline{B},$$

$$\overline{A \vee B} = \overline{A} \wedge \overline{B}$$

(формулы (II.18) и (II.19));

4) выражение $A \to B$ можно заменить через $\overline{A} \vee B$ (формула (II.21)), а выражение $A \sim B$ можно заменить через $(\overline{A} \vee B) \wedge (A \vee \overline{B})$ (формула (II.16)).

Порядок пользования этими правилами следующий: сначала заменяем имеющиеся в выражении импликации и равнозначности, применяя правило 4; затем, применяя правило 3, приводим выражение к такому виду, когда знаки отрицания относятся к отдельным членам; наконец, применяя правила 1 и 2, раскрываем выражения и исключаем двухкратные знаки отрицания.

Пусть, например, надо преобразовать логическое выражение

$$(A \to B) \sim (\overline{B} \to \overline{A})$$

к нормальной конъюнктивной форме. Применяя правило 4, исключаем знаки импликации:

$$(\overline{A} \vee B) \sim (\overline{B} \vee \overline{A}).$$

Далее с помощью (II.16) получаем:

$$[(\overline{A} \vee B) \vee (\overline{\overline{B} \vee \overline{A}})] \wedge [(\overline{\overline{A} \vee B}) \vee (\overline{B} \vee \overline{A})].$$

С помощью правила 3 имеем:

$$[(\overline{A} \vee B) \vee (B \wedge A)] \wedge [(A \wedge \overline{B}) \vee (\overline{B} \vee \overline{A})].$$

Опускаем излишние скобки и, учитывая, что

$$B \vee B \wedge A = B / \vee (1 \vee A) = B \wedge 1 = B,$$
$$A \wedge \overline{B} \vee \overline{B} = \overline{B} \wedge (A \vee 1) = \overline{B} \wedge 1 = \overline{B},$$

окончательно имеем:

$$(\overline{A} \vee B) \wedge (\overline{A} \vee \overline{B}).$$

Заметим, что конъюнктивная нормальная форма не является наиболее простым видом логического выражения. Например, последнее выражение можно свести к $\overline{A}$.

**2. Постоянно-истинные и постоянно-ложные выражения.** Нормальная форма удобна для анализа логических выражений и, в частности, для выделения двух классов выражений. Выражения, принадлежащие одному из них, называются *постоянно-истинными,* принадлежащие другому — *постоянно-ложными.*

Истинность или ложность сложного высказывания зависит, вообще говоря, от значений истинности основных высказываний, входящих в состав сложного высказывания. Однако могут быть построены такие сложные высказывания, у которых значения истинности являются постоянными независимо от того, какие значения истинности принимают основные высказывания. При упрощении сложных логических выражений надо выявлять постоянно-истинные высказывания и опускать их или заменять одним символом.

Так как любое логическое выражение может быть приведено к нормальной форме, то для решения поставленного выше вопроса достаточно рассматривать только нормальные формы сложных выражений. Суждения о постоянной истинности или ложности сложного выражения могут быть получены на основе применения простых правил:

1) выражение $A \vee \overline{A}$ всегда истинно;
2) если $A$ истинно, а $B$ означает произвольное высказывание, то выражение $A \vee B$ истинно;
3) если $A$ и $B$ истинны, то и выражение $A \wedge B$ тоже истинно.

На основе перечисленных трех простых правил получается правило для определения истинности сложного выражения.

Постоянно-истинными являются те выражения, у которых в нормальной конъюнктивной форме в каждую дизъюнкцию по меньшей мере одно основное высказывание входит вместе со своим отрицанием. Ясно, что при этом в каждой дизъюнкции будет по меньшей мере один истинный член, а значит, будут истинны и все дизъюнкции, являющиеся членами конъюнкции, т. е. будет истинна вся конъюнкция, представляющая собой заданное логическое выражение. Аналогичным образом на основе дизъюнктивной нормальной формы определяются и постоянно-ложные выражения.

**3. Многообразие сложных логических выражений.** Рассмотрим следующий вопрос математической логики: сколько существует различных, т. е. логически не эквивалентных, сложных логических выражений, которые могут быть образованы путем комбинации конечного числа основных высказываний? Сложное выражение, образованное из $n$ простых высказываний $A_1, A_2, ..., A_n$ будет эквивалентно другому выражению в том случае, если для каждой комбинации значений истинности основных высказываний $A_1, A_2, ..., A_n$ оба сложных выражения будут иметь одинаковые значения истинности. Так как всего можно получить $2^n$ комбинаций значений истинности $n$ основных высказываний и для каждой из этих $2^n$ комбинаций основных высказываний сложное выражение . может быть либо истинным, либо ложным, то существует $2^{2n}$ различных сложных выражений, которые могут быть построены из $n$ простых основных высказываний.

Пусть мы имеем два основных высказывания $A$, $B$ или, иначе говоря, две независимые двоичные переменные. Из двух двоичных переменных можно образовать $2^2 = 4$ различные комбинации значений истинности и для них составить $2^{2^2} = 2^4 = 16$ различных видов сложных выражений (т. е. двоичных функций, зависящих от двух независимых двоичных переменных).

Сказанное наглядно иллюстрируется таблицей 17.

Рассмотрим, при помощи каких операций получаются эти $2^{2^2}$ высказываний, т. е. рассмотрим вопрос о видах связей между высказываниями в сложных высказываниях. В таблице 17 представлены 16 возможных сложных высказываний, получающихся при сочетании двух основных переменных $A$ и $B$. Можно легко убедиться, что таблица содержит все возможные случаи. Нижняя строчка в таблице 17 содержит сводку принятых сокращенных обозначений, где

1) $A \wedge B$ — *и*, конъюнкция;
2) $A \vee B$ — *или*, дизъюнкция;
3) $A \rightarrow B$ — *если, то,* импликация;
4) $A \leftarrow B$ — *то, если,* обратная импликация;
5) $A \sim B$ — равнозначность;
6) $\overline{A \wedge B} = A/B$ — несовместность или связь Шеффера, отрицание конъюнкции;
7) $\overline{A \vee B}$ — *ни... ни,* отрицание дизъюнкции;
8) $\overline{A \rightarrow B}$ — *если нет..., то нет...,* отрицание импликации;
9) $\overline{A \leftarrow B}$ — *всегда нет, если нет,* отрицание обратной импликации;
10) $\overline{A \sim B}$ — противоположность, отрицание равнозначности;
11) $A$ — сложное высказывание, не зависит от $B$;
12) $\overline{A}$ — отрицание $A$;
13) $B$ — сложное высказывание, не зависит от $A$;
14) $\overline{B}$ — отрицание $B$;
15) 1 — *всегда истинно;*
16) 0 — *всегда ложно.*

Т а б л и ц а 17. Двоичные функции от двух независимых двоичных переменных

| $A$ | $B$ | 1) | 2) | 3) | 4) | 5) | 6) | 7) | 8) | 9) | 10) | 11) | 12) | 13) | 14) | 15) | 16) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| | | $\wedge$ | $\vee$ | $\rightarrow$ | $\leftarrow$ | $\sim$ | $\overline{\wedge}$ | $\overline{\vee}$ | $\overline{\rightarrow}$ | $\overline{\leftarrow}$ | $\approx$ | $A$ | $\overline{A}$ | $B$ | $\overline{B}$ | 1 | 0 |

Рассмотрение случаев 6) ,..., 10) таблицы 17 показывает, что они просто являются отрицаниями случаев 1**)** ,..., 5).

**4. Преобразование логических выражений.** Применение перечисленных простых правил позволяет легко получать сложные высказывания, составлять логические равенства. В тех случаях, когда имеется два или три исходных высказывания, логические равенства легко доказываются простой подстановкой с проверкой.

Пусть требуется проверить равенство

$$(A \vee A) \longrightarrow A = 1, \qquad \text{(II.30)}$$

(«если $A$ или $A$, то $A$ является всегда истинным»). Таблица 18 доказывает справедливость этого равенства. Таким образом, выражение

$$(A \vee A) \longrightarrow A$$

является всегда истинным, независимо от того, истинно или ложно $A$. Проверим следующее выражение:

$$A \longrightarrow (B \vee A) = 1 \qquad \text{(II.31)}$$

или просто $A \longrightarrow (B \vee A)$. Проверку истинности проведем в форме таблицы 19. Полученные значения доказывают правильность выражения (II.31).

Таблица 18.

$(A \vee A) \rightarrow A = 1$

| $A$ | $A \vee A$ | $(A \vee A) \rightarrow A$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

Таблица19.

$A \rightarrow .(B \vee A)$

| $A$ | $B$ | $(B \vee A)$ | $A \rightarrow .(B \vee A)$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 |

Аналогичным путем может быть проверено выражение

$$(A \vee B) \longrightarrow (B \vee A) = 1$$

и выражение с тремя высказываниями $A, B, C$

$$(A \rightarrow B) \rightarrow [(C \vee A) \rightarrow (C \vee B)] = 1.$$

В таблице 20 приведена сводка некоторых правил преобразования логических выражений, наиболее часто употребляющихся при решении задач анализа и синтеза переключательных схем. Заметим, что выражения $A, B, C, D$ сами могут быть любыми сложными логическими выражениями.

Формулы 7) и 20) показывают, что можно присоединять к Левым частям равенств еще один или несколько членов из числа уже имеющихся в выражениях слева и от этого значение истинности выражения не изменится. В равенстве 18)

$$A \wedge B \wedge 1 = A \wedge B$$

можно поставить в качестве всегда-истинного выражения 1, например любое из выражений 3) — 6) таблицы 20, при этом сложное выражение не изменит значения истинности; так

$$A \wedge B \wedge (A \vee \overline{A}) = A \wedge B,$$

ибо $(A \vee \overline{A}) = 1$, т. е. является всегда истинным.

Т а б л и ц а 20. Некоторые правила преобразования логических выражений

| |
|---|
| 1) $(A \wedge \overline{A}) = 0$<br>2) $(\overline{A} \wedge \overline{B}) \wedge (A \vee B) = 0$<br>3) $A \vee \overline{A} = 1$<br>4) $(A \wedge B) \vee (A \wedge \overline{B}) \vee (\overline{A} \wedge B) \vee (\overline{A} \wedge \overline{B}) = 1$ |
| 5) $\overline{A} \vee (B \vee A) = 1$<br>6) $(A \vee B) \vee (\overline{A} \wedge \overline{B}) = 1$<br>7) $A \vee A = A \wedge A = A$ |
| 8) $A \wedge (A \vee B) = A$<br>9)) $A \vee (A \wedge B) = A$ |
| 10) $(A \wedge \overline{B}) \vee (A \wedge B) = A \wedge (\overline{B} \vee B) = A$<br>11) $A \vee (A \wedge B) = A \wedge (1 \vee B) = A$ |

| |
|---|
| 12) $A \vee (\overline{A} \wedge B) = A \vee B$ |
| 13) $\overline{(A \wedge B) \vee (B \wedge C) \vee (A \wedge C)} = (\overline{A} \vee \overline{B}) \wedge (\overline{B} \vee \overline{C}) \wedge (\overline{A} \vee \overline{C})$ |
| 14) $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$ |
| 15) $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$<br>16) $(A \vee B) \wedge (C \vee D) = (A \wedge C) \vee (A \wedge D) \vee (B \wedge C) \vee (B \wedge D)$<br>17) $(A \wedge B) \vee (C \wedge D) = (A \vee C) \wedge (A \vee D) \wedge (B \vee C) \wedge (B \vee D)$ |
| 18) $A \wedge B \wedge 1 = A \wedge B$<br>19) $A \vee B \vee 0 = A \vee B$<br>20) $A \vee B \vee C = A \vee B \vee C \vee C \vee B \ldots$ |

Заметим, что выражения, приведенные в таблице 20, содержат только связи *и, или* и отрицания основных высказываний *A, B, C*. Как было показано раньше, все высказывания можно выразить через конъюнкции, дизъюнкции и отрицания основных высказываний. Применение указанных в таблице 20 равенств позволяет во многих случаях существенно упростить сложные логические выражения и привести их к виду, наиболее удобному для схемной реализации.

В таблице 21 в качестве обобщения ранее приведенного материала дана сводка выражений некоторых связей через конъюнкции, дизъюнкции и отрицания основных высказываний.

Т а б л и ц а 21. Выражение некоторых связей в нормальной форме

| |
|---|
| 1) $A \rightarrow B = \overline{A} \vee B$ |
| 2) $A \leftarrow B = A \vee \overline{B}$ |
| 3) $A \sim B = (A \vee \overline{B}) \wedge (\overline{A} \vee B) = (\overline{A} \wedge \overline{B}) \vee (A \wedge B)$ |
| 4) $\overline{A \wedge B} = \overline{A} \wedge \overline{B}$ |
| 5) $\overline{A \vee B} = \overline{A} \vee \overline{B}$ |
| 6) $\overline{A \rightarrow B} = A \wedge \overline{B}$ |
| 7) $\overline{A \leftarrow B} = \overline{A} \wedge B$ |
| 8) $\overline{A \sim B} = (A \wedge \overline{B}) \vee (\overline{A} \wedge B) = (\overline{A} \vee \overline{B}) \wedge (A \wedge B)$ |

Из формул 4) и 5) таблицы 21 вытекает следующее весьма важное для практического использования правило: для выражения, которое образуется из основных высказываний и их отрицаний с помощью только конъюнктивной и дизъюнктивной связи, противоположное ему получается путем обмена местами знаков $\wedge$ и $\vee$ и замены основных высказываний их отрицаниями.

В применении к высказываниям, являющимся всегда-истинными, это правило приводит к так называемому *принципу двойственности:* из всегда-истинной формулы $P \sim Q$, обе части которой образованы из основных высказываний и их отрицаний путем использования конъюнктивной и дизъюнктивной связей, получаем снова всегда-истинное выражение, меняя местами знаки $\wedge$ и $\vee$. Так, например, выражение

$$[A \vee (B \vee C)] \sim [(A \wedge B) \vee (A \wedge C)]$$

является всегда-истинным (см. 14) табл. 20); из него по принципу двойственности получим также всегда-истинное выражение

$$[A \vee (B \wedge C)] \sim [(A \vee B) \wedge (A \vee C)]$$

(см. 15) табл. 20).

### § 13. Переключательные (нелинейные) элементы электронных схем

**К** переключательным (нелинейным) элементам, применяемым для построения схем в электронных цифровых машинах, относятся электронные лампы, полупроводниковые диоды и триоды, магнитные сердечники и некоторые другие детали. Эти элементы служат для переключения, фиксации или представления в машинах сигналов, передающих информацию и управляющих работой блоков и устройств машины. При этом они выполняют переключательные функции в общем процессе преобразования информации, реализуемом в машине.

Помимо переключательных функций, некоторые из указанных элементов (например, электронные лампы), а также ряд других элементов (конденсаторы, сопротивления и др.) применяются в схемах для вспомогательных

технических целей (формирование и усиление сигналов, изменение полярности сигналов, синхронизация и т. д.).

Мы кратко напомним принцип действия электронной лампы и более подробно остановимся на работе полупроводниковых и магнитных элементов, учитывая их новизну и особую важность для техники электронных цифровых машин.

Переключательные элементы применяются в первую очередь для электрического представления двоичных чисел в машинах.

Для этого необходимо обеспечить физическую реализацию двух различных сигналов, один из которых будет соответствовать нулю, а другой — единице.

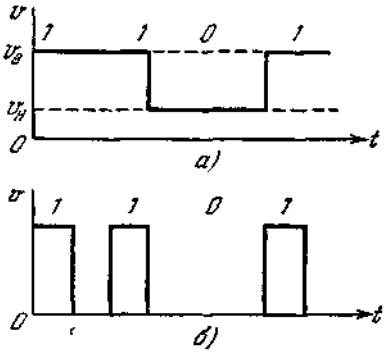



Рис.25. Представление цифр 0 и 1 электрическими сигналами.

Электрическими сигналами цифры 0 и 1 могут быть представлены двумя различными способами.

Статический способ. При этом способе применяются два различных уровня напряжения. Высокий уровень $v_B$ может, например, обозначать единицу, а низкий уровень $v_н$— нуль.

Соответствующий уровень напряжения сохраняется в течение всего времени представления данной цифры. Если подряд будут следовать несколько одинаковых цифр (1, 1, 1, ... или 0, 0, ...), то уровень напряжения сохраняется и в промежутках между моментами их задания (рис. 25, *а).*

Динамический способ. Цифры изображаются при помощи электрических импульсов определенной длительности. Наличие положительного импульса в определенный момент времени может соответствовать единице, а отсутствие положительного импульса или наличие отрицательного импульса — нулю.

При следовании подряд нескольких единиц напряжение в промежутках между моментами задания цифр падает до нулевого уровня (рис. 25, *б).*

**1. Электронные лампы.** Электронные лампы обладают способностью проводить ток только в одном направлении. Они являются нелинейными элементами, позволяющими создавать в машинах различные уровни напряжения, необходимые для представления двоичных цифр. Простейшей электронной лампой является диод (рис. 26, а). Диод имеет два электрода: анод и катод. Катод, нагретый до высокой температуры, испускает электроны, т. е. обладает термоэлектронной эмиссией.

Если к аноду приложен положительный потенциал, а к катоду — отрицательный, то электроны будут двигаться от катода к аноду, т. е. через диод пойдет электрический ток от анода к катоду (рис. 26, *б).* Если же приложить положительный потенциал к катоду, а отрицательный — к аноду, то ток через диод проходить не будет.

В первом случае на выходной клемме *С* будет высокое напряжение, почти равное потенциалу анода (так как внутреннее сопротивление диода, когда он проводит, значительно меньше нагрузочного сопротивления *R).* Во втором случае на клемме *С* будет низкий уровень напряжения, так как диод не проводит.

В ламповых диодах управление током, протекающим через диод, и полярностью выходного сигнала может осуществляться путем изменения полярности подводимого напряжения.

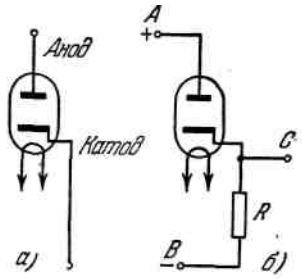



Рис.26. Диод.

Триод представляет собой электронную лампу, в которой между анодом и катодом расположен еще третий электрод — управляющая сетка (рис. 27, а). Управляющая сетка служит для управления током, протекающим через триод без изменения напряжения, приложенного между анодом и катодом.

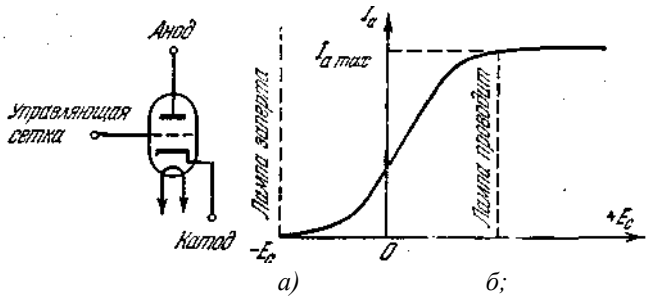



Рис. 27. Триод и его сеточная характеристика.

Приложенное к сетке триода отрицательное напряжение (смещение — $E_c$) уменьшает ток через триод и при определенной величине этого смещения ток через лампу полностью прекратится, лампа будет «заперта». Наоборот, при подаче на сетку положительного напряжения (смещения) ток через лампу увеличивается и при определенном значении этого смещения достигает максимума. Максимальное значение анодного тока через

лампу $I_{a\ max}$ соответствует тому, что почти все электроны, испускаемые катодом, попадают на анод (часть электронов попадает на сетку, образуя так называемый сеточный ток).

На рис. 27, *б* представлена типичная сеточная характеристика триода. Она имеет близкий к линейному наклонный участок и два нелинейных участка: верхний и нижний изгибы. Так как в цифровых машинах для представления двоичных чисел необходимы только два уровня напряжения, то на сеточной характеристике лампы используются две точки, расположенные по возможности дальше друг от друга за пределами линейного участка. Таким образом, электронные лампы в цифровых машинах находятся в двух резко отличающихся состояниях: или полностью заперты, или полностью открыты. Вследствие этого на работу схем изменение параметров ламп в процессе их эксплуатации не оказывает влияния.

Большое применение в электронных цифровых машинах имеют также лампы с пятью электродами — пентоды. Из трех сеток пентода обычно одна сетка обеспечивает нормальный режим работы самого пентода, а две другие могут быть использованы в качестве управляющих. В случае необходимости все три сетки пентода могут быть использованы в качестве управляющих. Пентод будет заперт, если хотя бы на одну из двух его управляющих сеток будет подано соответствующее отрицательное смещение.

**2. Полупроводниковые диоды и триоды.** В электронных цифровых машинах в настоящее время весьма широкое применение наряду с электронными лампами имеют полупроводниковые элементы: диоды и триоды. Во многих случаях эти элементы вполне заменяют электронные лампы; известны вычислительные машины, построенные полностью на полупроводниковых элементах.

Полупроводниковые *диоды,* подобно ламповым диодам, обладают выпрямительными свойствами, т. е. способностью пропускать электрический ток в одном направлении; при этом говорят, что «прямое» сопротивление полупроводникового диода во много раз меньше «обратного» сопротивления. Полупроводниковые диоды применяются для построения логических схем в электронных цифровых машинах.

Полупроводниковые *триоды,* называемые также *транзисторами,* представляют собой трехэлектродные приборы, обладающие, подобно электронным лампам, усилительными свойствами.

Основными преимуществами полупроводниковых приборов перед электронными лампами являются небольшие размеры, малое потребление энергии, высокая механическая прочность и долговечность.

Недостатками полупроводниковых приборов в настоящее время пока что являются значительный разброс параметров отдельных образцов приборов, нестабильность их характеристик во времени и зависимость характеристик от температуры.

Полупроводниковые приборы весьма эффективно могут применяться в импульсных переключательных схемах электронных цифровых машин, работающих по принципу *да — нет* (включено — выключено), так как здесь к ним не предъявляются высокие требования по стабильности работы. Наиболее широкое применение в настоящее время имеют германиевые полупроводниковые приборы. Ведется разработка и внедрение кремниевых приборов, которые по ряду свойств являются более перспективными.

Для того чтобы понять принцип действия полупроводниковых диодов и, особенно, триодов, необходимо вспомнить некоторые элементарные сведения о характере проводимости полупроводников.

Как известно, все вещества делятся на три группы: проводники, полупроводники и изоляторы.

В проводниках (к которым относятся металлы и их сплавы) проводимость электрического тока обеспечивается наличием между атомами вещества *электронного газа,* т. е. свободных электронов, которые легко отделяются от своих атомов и перемещаются в определенном направлении под действием приложенного электрического поля.

В изоляторах все атомы прочно удерживают свои электроны, свободные электроны отсутствуют и электрический ток проходить не может.

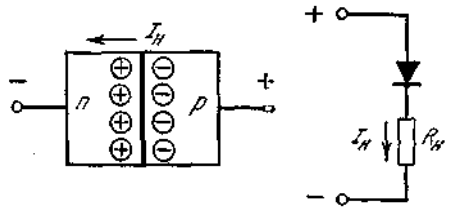
Полупроводники занимают промежуточное положение между проводниками и изоляторами. В полупроводниках имеет место проводимость двух видов: э*лектронная* и *дырочная.* Электронная проводимость в полупроводниках, так же как и в металлах, обусловлена движением электронов, но не свободных (как в металлах), а электронов, отрывающихся от своих атомов. При отрыве электрона от атома в атоме образуется пустое место — «дырка», не заполненная электроном и имеющая положительный заряд; на это место может перескочить электрон из соседнего атома, тогда дырка в данном атоме исчезнет, но появится в соседнем атоме. На новую дырку может перейти электрон из другого атома и т. д. Таким образом, дырка может перемещаться в направлении, соответствующем перемещению положительного заряда под действием электрического поля.

В чистых полупроводниках количество образующихся дырок равно количеству отрывающихся от атомов электронов, т. е. электронная и дырочная проводимости равны.

Для построения полупроводниковых диодов и триодов применяют не чистые полупроводники (германий или кремний), а полупроводники с примесями. Благодаря введению примеси полупроводник приобретает преимущественно одну какую-нибудь проводимость: электронную или дырочную. Принято называть электронную проводимость *проводимостью типа n,* а примеси, дающие эту проводимость, — *донорами.* Дырочную проводимость называют *проводимостью типа p,* а соответствующие примеси — *акцепторами.*

Германий и кремний являются четырехвалентными элементами, т. е. их атомы имеют в нормальном состоянии на внешней орбите по четыре электрона. При этом атомы являются электрически нейтральными. При отсутствии примесей атомы германия и кремния образуют кристаллическую решетку, в которой каждый атом связан с

четырьмя соседними атомами при помощи четырех валентных электронов, затрачивая по одному электрону на каждую связь. При введении в кристаллическую решетку атомов примеси, имеющей другое количество валентных электронов, нарушается соответствие между структурой кристаллической решетки и нейтральностью атомов. Например, если к германию добавить в качестве примеси мышьяк, имеющий пять валентных электронов, то один валентный электрон атома мышьяка не будет участвовать в построении кристаллической решетки и с этой точки зрения будет являться лишним. Для того чтобы сохранилась общая структура кристаллической решетки, атомы мышьяка должны отдать по одному валентному электрону, хотя это приведет к нарушению нейтральности атомов мышьяка в электрическом отношении. Эти избыточные электроны создают в полупроводнике преимущественно электронную проводимость.




*а)* *б)*

Рис. 28. Полупроводниковый диод.

Если же в германий в качестве примеси будут введены трехвалентные атомы, например атомы индия, то у атомов примеси будет не хватать по одному валентному электрону для построения кристаллической решетки данного типа, т. е. для обеспечения связей с четырьмя соседними атомами. В каждом атоме примеси будет образовываться дырка и полупроводник будет обладать преимущественно дырочной проводимостью.

Таким образом, путем введения соответствующих примесей могут быть получены полупроводники, обладающие преимущественно проводимостью типа *n* или *p*.

Полупроводниковые диоды и триоды строятся путем сочетания в одном элементе нескольких слоев полупроводника с различными типами проводимости. Соединение производится таким образом, чтобы между указанными слоями образовывался электронно-дырочный переход (или *(n — p)*-переход). На рис. 28, *а* показан электронно-дырочный переход, а на рис. 28, *б* показано изображение полупроводникового диода на схемах.

В полупроводниковых диодах необходим один электронно-дырочный переход, в полупроводниковых триодах необходимы два электронно-дырочных перехода. Разрабатываются полупроводниковые приборы с большим количеством электронно-дырочных переходов. Электронно-дырочный переход обладает свойством односторонней проводимости, которое в полупроводниковых диодах обусловливает их выпрямляющее действие.

Электронно-дырочный переход может быть образован, например, путем сплавления германия типа *n* (т. е. имеющего примесь мышьяка) с каплей индия. Вблизи индия образуется слой германия, обладающего проводимостью типа *p*, а в остальной массе германия сохранится проводимость типа *n*. Между этими областями образуется (*n — p*)-переход.

Таким путем получаются так называемые *плоскостные* диоды и триоды. Кроме того, существуют еще *точечные* диоды и триоды, в которых используются точечные контакты между тонкими металлическими проволоками и кристаллом германия. Кристалл германия имеет один тип проводимости, а на его поверхности в местах соприкосновения с контактными проволочками образуются участки с другим типом проводимости.

Односторонняя проводимость (*n — p*)-перехода обусловлена появлением на границе этого перехода двойного электрического слоя.

Так как в области *p* имеет место избыток дырок (т. е. недостаток электронов), а в области *n* имеет место избыток электронов, то вследствие диффузии через (*n — p*)-переход даже при отсутствии внешнего электрического поля происходит движение электронов (из *n*-области в *p*-область) и движение дырок (из *p*-области в *n*-область). Это движение приводит к образованию положительного заряда в *n*-области и отрицательного заряда в *p*-области.

Возникающее при этом электрическое поле образует потенциальный барьер, который препятствует дальнейшей диффузии электронов и дырок. Величина потенциального барьера определяется, с одной стороны, концентрациями примесей, обусловливающих диффузию, и, с другой стороны, количеством и энергией заряженных частиц в областях *n* и *p* , обусловливающих появление двойного электрического слоя. Ясно, что протекание электрического тока через диод будет различным в зависимости от полярности приложенного напряжения. Если положительный полюс источника тока будет подключен к области *p*, а отрицательный полюс — к области *n*, то произойдет пополнение обедненных электронами и дырками пограничных областей, уменьшится сопротивление перехода и пойдет значительный электрический ток.

При обратном подключении полюсов источника тока будет происходить еще большее обеднение пограничной *n*-области электронами и пограничной *p*-области дырками, величина потенциального барьера увеличится и протекание электрического тока будет затруднено.

Таким образом, полупроводниковый диод пропускает ток преимущественно в одном направлении, т. е. обладает свойством выпрямителя тока. Прямое направление тока (с малым сопротивлением) соответствует протеканию тока от *p*-области к *n*-области, а обратное направление (с большим сопротивлением) соответствует протеканию тока от *n*-области к *p*-области. По аналогии с электронной лампой в полупроводниковом диоде можно считать *p*-область — анодом, а *n*-область — катодом.

Рассмотрим принцип действия полупроводникового триода, или транзистора.

Транзистор имеет три электрода: *эмиттер (Э), коллектор (К)* и *базу (Б). На* рис. 29 показано обозначение транзистора, применяемое на принципиальных схемах. Часто это обозначение дается без кружка вокруг электродов. База является входным электродом; между эмиттером и базой подается входной сигнал. Коллектор является выходным электродом; выходной сигнал снимается между коллектором и базой.

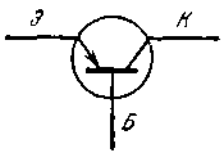


Рис. 29. Транзистор.

Основой транзистора является кристалл германия, в котором благодаря введению примесей кристаллическая решетка построена так, что имеется избыток валентных электронов, которые обеспечивают электронную проводимость полупроводника (проводимость типа *n*).

В местах соприкосновения эмиттера и коллектора с кристаллом германия благодаря введению примесей другого рода создается такая кристаллическая решетка, в которой имеет место недостаток валентных электронов, т. е. образуются положительно заряженные дырки. Эти участки обладают дырочной проводимостью или проводимостью типа р. При дырочной проводимости прохождение тока сводится к перескокам электронов из нейтральных атомов на свободные дырки, в результате чего появляются дырки в соседних атомах; затем другие электроны перескакивают на эти дырки, и т. д. Таким образом, происходит перемещение дырок в направлении движения тока.

На границах соприкосновения участков с *n*-проводимостью и *p*-проводимостью образуется двойной электрический слой, так как вследствие диффузии часть дырок из *p*-области переходит в *n*-область, а часть электронов из *n*-области переходит в *p*-область. Поэтому в *p*-области образуется недостаток положительных зарядов, а в *n*-области — их избыток. Этот двойной электрический слой препятствует дальнейшему переходу дырок от *p*-области в *n*-область и электронов от *n*-области в *p*-область, т. е. обладает односторонней проводимостью.

На рис. 30 показана одна из возможных схем включения транзистора, иллюстрирующая принцип его работы. Между эмиттером и базой подключена батарея $E_1$, имеющая небольшое напряжение (порядка 2 *в*). Полярность подключения батареи $E_1$ выбрана такой, чтобы способствовать переходу дырок от эмиттера (*p*-область) в основной кристалл германия (*n*-область) и переходу электронов из германия к эмиттеру, т. е. эта батарея подключена в направлении прямого тока, для которого переход от эмиттера к базе ((*p — n)*-переход) представляет небольшое сопротивление. Последовательно с батареей $E_1$ включается переменное управляющее напряжение сигнала, который требуется усилить ($U_{вх}$).

Между коллектором и базой включается батарея $E_2$ с высоким напряжением (десятки вольт) таким образом, что ее полярность противоположна направлению прямого тока от коллектора к базе, т. е. *(n — p)*-переход представляет для тока большое сопротивление. Последовательно с батареей $E_2$ включается сопротивление нагрузки $R_н$ с которого снимаются усиленные сигналы.

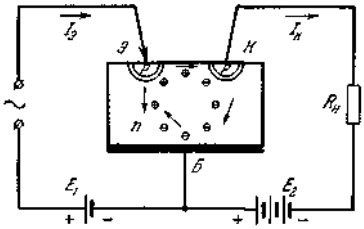


Рис. 30. Схема включения транзистора

При указанной полярности подключения батарей $E_1$ и $E_2$ дырки, эмитируемые в большом количестве эмиттером в основной кристалл, движутся не только к базе, но и в значительной части к коллектору. За счет поступления этих дырок на коллектор усиливается ток $I_к$, идущий по внешней цепи от коллектора к базе. Этот ток создает на сопротивлении $R_н$ падение напряжения, снимаемое в качестве выходного сигнала.

Изменяя входное напряжение, можно изменять ток эмиттера $I_э$ и тем самым изменять количество дырок, эмитируемых эмиттером, и таким образом управлять величиной тока коллектора, а следовательно, и величиной выходного сигнала.

По аналогии с электронной лампой отдельные электроды в транзисторе играют следующую роль: эмиттер играет роль катода, коллектор — роль анода и база — роль управляющей сетки. Следует заметить, что направление тока здесь обратное: вместо движения электронов от катода к аноду, которое имеет место в электронной лампе, в транзисторе имеет место движение дырок от эмиттера к коллектору.

Характерными для работы полупроводникового триода в переключающих схемах являются два режима: режим полного запирания, когда коллекторный ток равен нулю, и режим насыщения, когда коллекторный ток достигает максимального значения, определяемого количеством возникающих электронов и дырок. До режима насыщения увеличение тока эмиттера приводит к увеличению тока коллектора. При насыщении дальнейшее увеличение тока эмиттера не вызывает увеличения коллекторного тока.

В настоящее время в технике электронных цифровых машин широко применяются точечные и плоскостные полупроводниковые элементы. Точечные триоды имеют достаточно высокие частотные характеристики. Они работают при частотах свыше 1 мегагерца, но являются недостаточно стабильными и надежными. Плоскостные триоды обладают более стабильными характеристиками и более высокой надежностью, чем точечные, но имеют меньший частотный диапазон. Они могут работать в схемах, имеющих частоту следования импульсов лишь до нескольких сотен килогерц.

Так как требования стабильности и надежности работы элементов являются решающими для обеспечения надежности электронных цифровых машин, то плоскостные триоды в настоящее время имеют наибольшее практическое значение и применение. Однако ограниченные частотные характеристики этих триодов не позволяют использовать их при построении машин с высокими скоростями работы (порядка нескольких тысяч операций в секунду).

Высокими частотными характеристиками (до нескольких десятков мегагерц) обладают так называемые триоды с поверхностным барьером. Эти триоды изготавливаются из пластинки германия, обладающего проводимостью типа *n*. На этой пластинке электрохимическим способом вытравляются две лунки, расположенные на расстоянии порядка 5 микрон; в лунки вводятся капли индия. В непосредственной близости от индия образуются области с проводимостью типа *p*.

Таким образом, между германием и каплями индия создаются два (*p* — *n*)-перехода, образующих триод типа *p* — *n* — *p*. Помимо высоких частотных свойств и высокой стабильности работы, триоды с поверхностным барьером обладают высокой экономичностью: они потребляют сотые доли ватта.

В последнее время начали выпускаться еще так называемые *диффузионные* полупроводниковые триоды, обладающие весьма высокими частотными характеристиками и высокой стабильностью в работе.

**3. Магнитные сердечники.** Широкое применение в электронных цифровых машинах в настоящее время имеют магнитные ферритовые сердечники. Эти сердечники первоначально использовались для построения оперативных запоминающих устройств, но в последние годы начали широко применяться также для построения логических переключательных и счетных схем в сочетании с электронными лампами и полупроводниковыми диодами и триодами.

*Ферриты* представляют собой полупроводниковые материалы, обладающие магнитными свойствами. Ферриты приготовляются из смеси различных окислов металлов (магния, марганца, никеля и др.) с магнитным железняком. Благодаря высокому электрическому сопротивлению ферритов потери на образование токов Фуко при их перемагничивании незначительны, что обусловливает незначительное потребление энергии.

Магнитные ферритовые сердечники представляют собой небольшие кольца прямоугольного сечения (тороиды) размером от долей миллиметра до нескольких миллиметров.

Чем меньше сердечник и тоньше его стенка, тем выше допустимая частота работы, но тем меньше сигнал, который снимается с сердечника. На сердечниках имеется несколько обмоток из тонкой проволоки.

Ферритовые сердечники, применяемые в электронных цифровых машинах, обладают прямоугольной петлей гистерезиса. *Петля гистерезиса* — это кривая, показывающая зависимость между величиной тока, протекающего по обмотке (или напряженностью магнитного поля $H$), и величиной магнитной индукции сердечника $B$.

На рис. 31 показана идеализированная петля гистерезиса магнитного сердечника, показывающая свойство сердечника находиться в двух устойчивых состояниях, т. е. способность его запоминать двоичные цифры. Можно принять, например, что положительное состояние остаточного намагничивания ($+B_r$) соответствует запоминанию единицы, а отрицательное ($-B_r$) —запоминанию нуля.

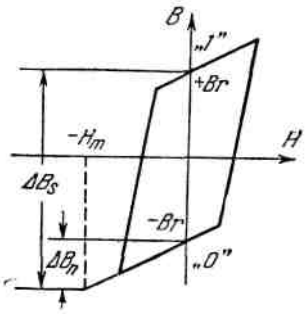


Рис 31. Идеализированная. петля гистерезиса магнитнего сердечника

Для приведения сердечника в то или иное состояние достаточно подать в его обмотку импульс тока, образующий магнитное поле необходимой напряженности. Проверка того, в каком состоянии находится сердечник, а также считывание записанной информации производятся путем подачи отрицательного импульса ($-H_m$) достаточной амплитуды и длительности. Если сердечник находился в состоянии 0, то перемагничивания не произойдет, а лишь несколько увеличится (на величину $\Delta B_n$) его отрицательная магнитная индукция и на выходной обмотке появится незначительный импульс, представляющий собой помеху. Если же сердечник находился в состоянии 1, то произойдет полное перемагничивание сердечника, его магнитная индукция изменится на величину $\Delta B_S$ и на выходной обмотке появится значительный импульс, представляющий сигнал. Чем лучше «прямоугольность» петли гистерезиса, тем выше отношение величины сигнала при считывании единицы к величине помехи при считывании нуля, характеризующее собой качество сердечника. Для имеющихся сердечников это отношение колеблется от 20 до 50 и выше. Время перемагничивания сердечников в зависимости от их размеров изменяется от нескольких долей микросекунды до нескольких микросекунд, что позволяет создавать на магнитных сердечниках достаточно быстродействующие переключательные схемы и запоминающие устройства.

Ценным качеством магнитных сердечников является их нечувствительность к изменению температуры, способность длительное время надежно сохранять записанную информацию.

Применение магнитных сердечников для построения логических переключательных схем основано на способности сердечников при некоторых соединениях передавать друг другу определенные сигналы и изменять свое состояние под действием этих сигналов.

На рис. 32 приведен простейший пример схемы соединения двух магнитных сердечников, обеспечивающей передачу сигнала от сердечника $M_1$ к сердечнику $M_2$.

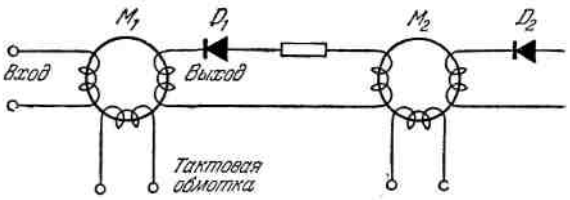



Рис. 32. Простейшая схема соединения двух магнитных сердечников.

Каждый сердечник имеет по три обмотки: одну входную, одну выходную и одну тактовую обмотку.

Тактовая обмотка служит для подачи сдвигающих импульсов, обусловливающих процесс передачи сигналов от одного сердечника к другому.

Пусть оба сердечника, $M_1$ и $M_2$, находятся в нулевом состоянии. При подаче на вход $M_1$ положительного импульса этот сердечник перейдет в состояние «1», а на вход сердечника $M_2$ импульс не поступит, так как диод $D_1$ включен таким образом, чтобы не пропускать возникающий при этом ток. Сердечник $M_2$ останется в нулевом состоянии. При поступлении тактового импульса, переводящего сердечник $M_1$ из состояния «1» в состояние «0», в выходной обмотке сердечника $M_1$ возникает ток противоположного направления, который поступит во входную обмотку сердечника $M_2$ и перебросит его из состояния «0» в состояние «1».

Таким образом, тактовый импульс переведет сердечник $M_1$ из состояния «1» в состояние «0», а сердечник $M_2$ — из состояния «0» в состояние «1», т. е. как бы осуществит передачу единицы от сердечника $M_1$ на сердечник $M_2$.

На сердечники, находящиеся в нулевом состоянии, тактовые импульсы влияния не оказывают, Помимо указанных выше обмоток, на сердечниках бывает еще так называемая обмотка запрета, которая наматывается в противоположном направлении по отношению к входной обмотке. Обмотка запрета нейтрализует действие входной обмотки: при поступлении сигналов одновременно на обе обмотки сердечник не изменяет своего состояния. Для «переброса» сердечника необходимо, чтобы сигнал был на входной обмотке и отсутствовал на обмотке запрета.

На схемах (рис. 33) магнитные сердечники изображаются обычно в виде кружков; обмотки изображаются стрелками, подходящими к кружку (входные) и выходящими от кружка (выходные). Внутри кружка у входных стрелок ставятся цифры 0 и 1, показывающие, в какое состояние переводится сердечник данным входным сигналом. Цифры около выходных стрелок показывают, в какое состояние должен быть переброшен сердечник, чтобы на данной выходной обмотке появился сигнал.

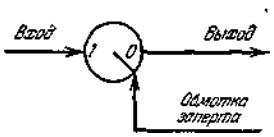



Рис. 33. Схематическое изображение магнитного сердечника.

Обмотка запрета показывается обычно наклонной линией, перечеркивающей (полно – стью или наполовину) сердечник.

Тактовые обмотки обычно на схемах не показываются, а их наличие подразумевается.

## § 14. Электронные схемы для основных логических операций

**1. Логические схемы на электронных лампах и полупроводниковых диодах.** Рассмотрим простейшие электронные схемы, с помощью которых в машинах реализуются приведенные выше основные логические операции *не, и, или.*

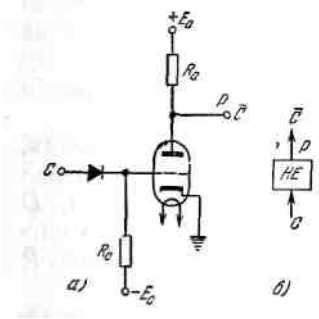



Рис. 34. Схема инвертора и его условное обозначение.

Схема инвертора, служащая для реализации логической операции *не* (нет), приведена на рис. 34, *а.*

Сигнал высокого уровня на выходе $P$ будет в том случае, когда на входе $C$ сигнал отсутствует. При этом на сетке триода будет низкое напряжение, запирающее триод, и потенциал точки $P$ будет равен потенциалу $+E_a$. При подаче на вход $C$ сигнала высокого уровня триод отпирается и благодаря падению напряжения на сопротивлении $R_a$ потенциал точки $P$ падает, т. е. сигнал на выходе исчезает. На рис. 34, б дано обозначение инвертора, применяемое на функциональных схемах.

Схема совпадения реализует логическую операцию *и.* Она представляет собой многополюсник, в общем случае, с $n$ входами и одним выходом. Сигнал на выходе Схемы появляется в том и только том случае, когда имеются сигналы на всех входах одновременно, т. е. эта схема реализует логическое произведение. Примеры схем совпадения приведены на рис. 35. Схему совпадения для двух сигналов иногда называют клапаном или вентилем.

На рис. 35, *а* показана схема совпадения на диодах. Сигнал на выходе $P$ будет иметь высокий уровень напряжения в том случае, если на катоды обоих диодов, т. е. в точки $A$ и $B$, подано высокое положительное напряжение.

При отсутствии хотя бы на одном входе $A$ или $B$ положительного сигнала высокого уровня соответствующий диод будет проводить и благодаря падению напряжения на сопротивлении $R$ потенциал точки $P$ понизится.

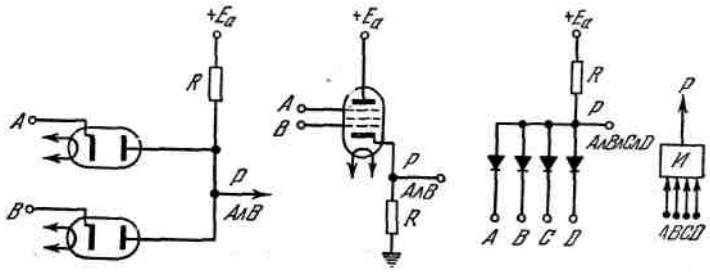



Рис. 35. Схема совпадения.

На рис. 35, *б* показана схема совпадения на пентоде. Ток через пентод проходит только при наличии положительных сигналов на обеих управляющих сетках пентода. Падение напряжения на сопротивлении *R* обеспечивает появление сигнала высокого уровня на выходе *P*.

На рис. 35, *в* приведена схема совпадения на выпрямителях, имеющая четыре входа, *A, B, C, D.* На выходе *P* появится сигнал высокого уровня в том случае, если на все четыре входа *A, B, C, D* поданы положительные сигналы. Отсутствие положительного сигнала хотя бы на одном входе вызовет ток через сопротивление *R* и падение потенциала точки *P*.

На рис. 35, *г* показано обозначение схемы совпадения, применяемое на функциональных схемах.

Приведенные схемы, как нетрудно видеть, могут служить и для реализации логической операции *или,* если пользоваться отрицательными сигналами. Собирательные схемы — схемы, реализующие операцию *или* для положительных сигналов, приведены на рис. 36. Каждая собирательная схема имеет несколько входов и один выход. Сигнал высокого уровня на выходе *P* появляется только в том случае, если имеется сигнал хотя бы на одном входе, т. е. реализуется формула логической суммы: $P = A \vee B \vee C \vee D$,

На рис. 36, *а* показана собирательная схема на диодах. Наличие положительного сигнала хотя бы на одном из входов *A* или *B* вызовет ток через сопротивление *R* и появление положительного сигнала на выходе *P*. На рис. 36, б приведена собирательная схема на триодах, а на рис. 36, *в* — собирательная схема на выпрямителях

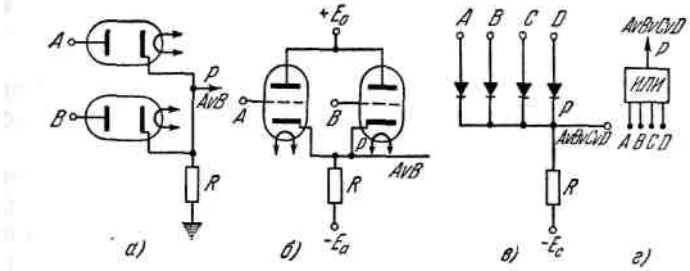



Рис. 36. Собирательные схемы.

. На рис. 36, *г* показано обозначение собирательной схемы, применяемое на функциональных схемах.

Очень важную роль играет в технике электронных цифровых машин электронная схема с двумя устойчивыми состояниями, так называемый триггер.

Триггерная схема (рис. 37) в простейшем случае содержит два триода, которые соединены перекрестными связями между анодами и сетками и имеют общее смешение. Это заставляет схему находиться в одном из двух устойчивых состояний до тех пор, пока внешний сигнал не изменит этого состояния.

В этой схеме анод первого триода $Л_1$ через сопротивление $R_4$, зашунтированное конденсатором $C_1$, соединяется с сеткой второго триода $Л_2$, а анод второго триода через $R_5$ и шунтирующий конденсатор $C_2$ соединяется с сеткой первого. Увеличение тока, проходящего через триод, вызывает понижение потенциала анода данного триода, а следовательно, и понижение напряжения на сетке другого триода.

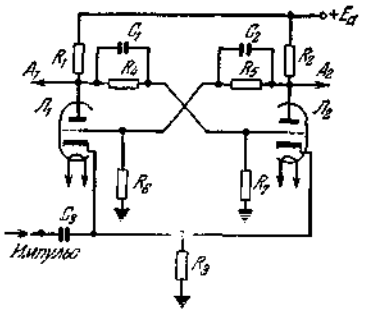



Рис. 37. Триггерная схема.

Например, увеличение тока через лампу $Л_1$ вызывает увеличение падения напряжения на сопротивлении $R_1$, что снижает потенциал на сетке $Л_2$. По мере понижения напряжения на сетке $Л_2$ анодный ток через эту лампу понижается, что приводит к уменьшению падения напряжения на сопротивлении $R_2$ и повышению

потенциала анода лампы .$Л_2$ В связи с этим повышается потенциал сетки лампы $Л_1$ и ток через эту лампу возрастает и т. д. Этот процесс продолжается до тех пор, пока лампа $Л_2$ не окажется полностью запертой, а через лампу $Л_1$ будет

идти максимальный ток. Схема примет устойчивое положение, характеризующееся наличием высокого потенциала на одном из анодов и низкого потенциала на другом. Аноды ламп могут быть использованы в качестве выходов схемы. Например, точка $A_1$ — основной выход, а точка $A_2$ — дополнительный выход.

Анодный ток открытой лампы дает падение напряжения как на ее собственном анодном сопротивлении, так и на общем катодном сопротивлении $R_3$.

Устойчивость работы триггера зависит от подбора сопротивлений; нужно чтобы потенциал на катодах ламп при одной полностью запертой лампе был выше потенциала сетки запертой лампы на определенную величину запирающего напряжения (напряжения отсечки). При этом на сетке запертой лампы создается отрицательное смещение, которое удерживает схему в устойчивом состоянии. Для того чтобы перевести схему в другое состояние, необходимо подать внешний импульс.

Существует несколько способов подачи внешних сигналов в триггерную схему — несколько способов управления триггерами. Например, можно подавать входные импульсы одной и той же полярности на один вход (на катоды обеих ламп), как показано на схеме. При подаче каждого нового сигнала схема будет поочередно переходить из одного состояния в другое и осуществлять счет поступающих импульсов по модулю 2. В этом случае схема будет представлять собой одноразрядный двоичный счетчик.

Применяется и другой способ управления триггером, при котором управляющие сигналы подаются в две точки схемы, например на сетки ламп. При этом в зависимости от полярности поданного сигнала схема может перейти или не перейти в другое состояние. Например, если на сетку проводящей лампы будет подан положительный сигнал, то он не изменит состояния схемы. Отрицательный же сигнал вызовет переход схемы в другое состояние.

Каждая триггерная схема может служить для запоминания одного разряда двоичного числа. Несколько ($n$) триггеров могут служить для запоминания $n$-разрядного двоичного числа, т. е. образуют регистр — запоминающее устройство на одно число.

Если несколько триггерных схем соединить между собой последовательно таким образом, чтобы сигнал с выхода одного триггера использовался как входной сигнал для следующего триггера, то образуется счетчик, т. е. схема для счета последовательно поступающих импульсов на вход триггера младшего разряда.

Каждый триггер соответствует определенному разряду двоичного числа, находящегося в данный момент в счетчике, а последовательная связь между триггерами обеспечивает переносы между разрядами.

Следует заметить, что триггер может использоваться как схема, осуществляющая логическую операцию отрицания. Если с основного выхода триггера снимается сигнал, обозначающий величину *A,* то с дополнительного выхода триггера снимается всегда сигнал $\overline{A}$ (не *A).*

**2. Логические схемы на полупроводниковых триодах.** Указанные выше свойства полупроводниковых триодов позволяют использовать их вместо электронных ламп в различных переключательных схемах электронных цифровых машин (вентили, триггеры, регистры, счетчики и др.).

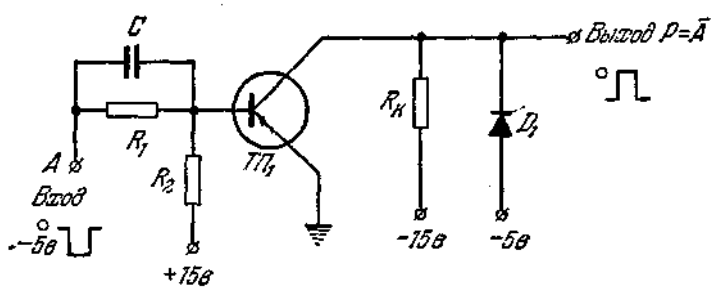



Рис. 38. Схема инвертора на полупроводниковом триоде типа $p$ — $n$ — $p$.

На рис. 38 показан пример схемы инвертора на полупроводниковом германиевом триоде типа $p$ — $n$ — $p$. Инвертор служит для реализации логической операции отрицания *(не).* На выходе инвертора появляется сигнал противоположного знака по отношению к сигналу, поданному на вход.

Триод $T\,П_1$ в нормальном состоянии при отсутствии сигнала на входе находится в полузакрытом состоянии, т. е. через него проходит небольшой коллекторный ток. При этом напряжение на базе равно нулю, а напряжение на коллекторе и, следовательно, на выходе устанавливается равным — 5 *в*. Это значение напряжения на выходе поддерживается при помощи диода $D_1$.

При подаче на вход (на базу) отрицательного импульса — 5 *в* триод полностью открывается и напряжение на коллекторе, а следовательно, и на выходе, делается равным нулю. Сопротивления $R_1$ и $R_2$ служат для установления необходимого режима работы схемы.

На рис. 39 приведена схема последовательного соединения трех полупроводниковых триодов с поверхностным барьером. Схема имеет три входа и реализует операцию *и* — логическую операцию конъюнкции для трех переменных. Пусть в исходном состоянии напряжение на базе всех трех триодов $ПТ_1$, $ПТ_2$, $ПТ_3$ равно нулю. Тогда все три триода будут закрыты и коллекторный ток $I_к$ трех последовательно соединенных триодов будет весьма мал. При этом напряжение на выходе будет близко к напряжению источника питания — *2 в*.

Для того чтобы на выходе появился положительный импульс, необходимо на базы всех трех триодов подать

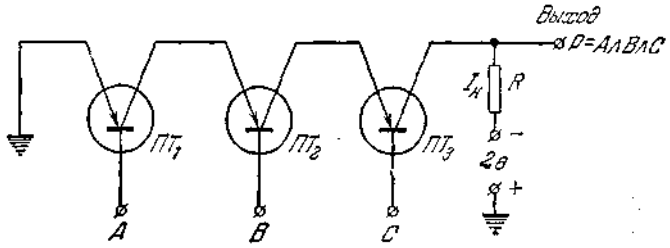



Рис. 39. Схема последовательного соединения трех полупроводниковых триодов с поверхностным барьером.

отрицательные напряжения. Тогда все три триода будут открыты и через сопротивление *R* пойдет коллекторный ток, который за счет падения напряжения на этом сопротивлении даст положительный импульс на выходе. Если хотя бы на базе одного из триодов будет отсутствовать отрицательное напряжение, то этот триод будет закрыт и коллекторный ток идти не будет, т. е. сигнал на выходе не появится.

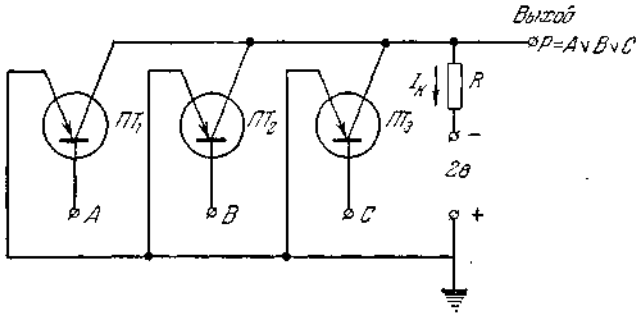



Рис 40. Собирательная схема, построенная на полупроводниковых приборах с поверхностным барьером.

На рис. 40 показана схема *или* на три входа, построенная на полупроводниковых приборах с поверхностным барьером. Схема представляет собой параллельное соединение трех триодов и реализует логическую операцию дизъюнкции.

Если на базах всех трех триодов отсутствуют отрицательные напряжения, то все три триода заперты и на выходе положительный импульс не появляется. Если будет подано отрицательное напряжение на базу хотя бы одного триода, то этот триод открывается, через него пойдет достаточно большой коллекторный ток, который вызовет падение напряжения на сопротивлении *R* и появление положительного сигнала на выходе.

На рис. 41 показана схема триггера на полупроводниковых триодах с поверхностным барьером. Схема по внешнему виду напоминает схему триггера на электронных лампах. В данном случае коллекторы и базы триодов $ПТ_1$ и $ПТ_2$ соединены крест-накрест, что обеспечивает получение схемы с двумя устойчивыми состояниями равновесия.

Для понимания работы схемы следует иметь в виду, во-первых, что полупроводниковые триоды запираются при напряжении на базе, равном нулю, и, во-вторых, что в режиме насыщения коллекторного тока напряжение на коллекторе приближается почти к нулю.

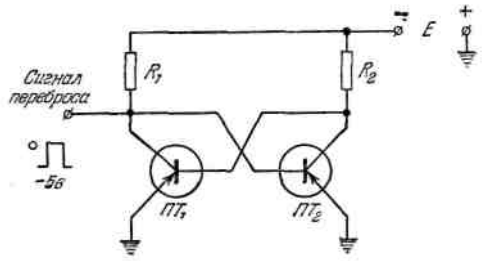



Рис. 41. Триггер на полупроводниковых триодах с поверхностным барьером.

Пусть в начальный момент триод $ПТ_1$ закрыт, его коллекторный ток близок к нулю и напряжение на его коллекторе будет приблизительно равно отрицательному напряжению источника питания — *E*. Это же отрицательное напряжение будет на базе триода $ПТ_2$, который будет благодаря этому полностью открыт. Через триод $ПТ_2$ будет идти значительный коллекторный ток (ток насыщения) и благодаря падению напряжения на

сопротивлении $R_2$ на коллекторе триода $ПТ_2$, а следовательно, и на базе триода $ПТ_1$ будет близкое к нулю напряжение, запирающее триод $ПТ_1$. В этом состоянии схема будет находиться до тех пор, пока внешний сигнал не перебросит ее в другое устойчивое состояние.

В качестве сигнала переброса может быть подан, например, положительный импульс на коллектор триода $ПТ_1$, поднимающий напряжение на коллекторе $ПТ_1$ и, следовательно, на базе триода $ПТ_2$ до нуля. При этом коллекторный ток триода $ПТ_2$ уменьшится, что приведет к понижению напряжения на базе триода $ПТ_1$ и открыванию этого триода. Увеличение коллекторного тока триода $ПТ_1$ благодаря падению напряжения на сопротивлении $R_1$ вызовет повышение до нуля напряжения на базе триода $ПТ_2$, что приведет к запиранию этого триода.

Таким образом, схема будет переброшена в другое устойчивое состояние (триод $ПТ_1$ открыт, а триод $ПТ_2$ заперт) и будет сколь угодно долго находиться в этом состоянии.

**3. Логические схемы на магнитных сердечниках.** На основе ферритовых сердечников могут быть построены схемы, выполняющие различные логические операции. На рис. 42, *а* показано символическое

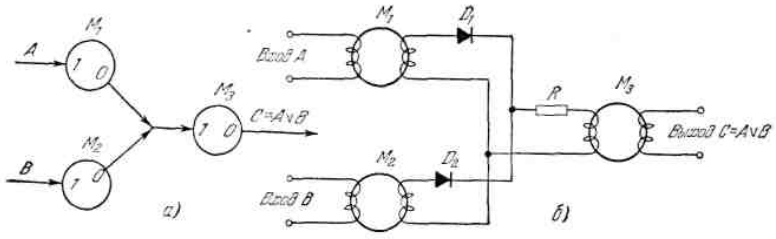



Рис. 42 . Собирательная схема на ферритовых сердечниках.

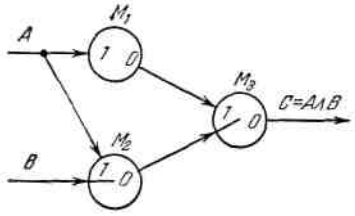



Рис. 43. Схема совпадения на ферритовых сердечниках

изображение схемы, реализующей логическую операцию дизъюнкции (*или*), а на рис. 42, *б* показана сама схема.

Выходные обмотки сердечников $M_1$ и $M_2$ через разделительные диоды $D_1$ и $D_2$ подключены параллельно к входной обмотке сердечника $M_3$. Сигнал на входе *А* перебрасывает сердечник $M_1$ в состояние «1», а сигнал на входе *Б* перебрасывает сердечник $M_2$ в состояние «1». Тактовый импульс перебрасывает сердечники $M_1$ и $M_2$ в состояние «0» и вызывает в выходных обмотках этих сердечников появление сигналов, которые (вместе или порознь) перебрасывают сердечник $M_3$ в состояние «1». Следующим тактовым импульсом $M_3$ перебрасывается в нуль и выдает сигнал на выходе *С*.

.Схема *или* может быть осуществлена с помощью одного сердечника, если на нем имеется несколько дополнительных обмоток.

На рис. 43 показана символическая схема, реализующая логическую операцию конъюнкции (*и*).

Сигнал на выходе сердечника $M_3$ появляется только при наличии сигналов на входах *А* и *В*. Если, например, поступит импульс только на вход *А,* то этот импульс перебросит в состояние «1» оба сердечника, $M_1$ и $M_2$ (сердечник $M_2$ перебросится в состояние «1», так как на обмотке запрета, на входе *Б,* сигнала нет). Тактовый импульс перебросит сердечники $M_1$ и $M_2$ в состояние «0, благодаря чему в выходных обмотках этих сердечников появятся сигналы. Так как сигнал от сердечника $M_2$поступит на обмотку запрета сердечника $M_3$ то сигнал, поступающий на входную обмотку этого сердечника, не вызовет его перемагничивания и при следующем тактовом импульсе на выходе сердечника $M_3$ сигнала не будет. Очевидно, что и при поступлении сигнала только на вход *Б* сигнал на выходе сердечника $M_3$ также не появится.

На рис. 44 приведена символическая схема динамического триггера на ферритовом сердечнике.

В отличие от описанных ранее схем триггеров на лампах и полупроводниковых триодах, называемых *статическими,* в *динамическом* триггере выходные сигналы представляются не постоянными (статическими) уровнями напряжения, а последовательностью импульсов. Если состояние триггера соответствует записи «1», то сего выхода выдается последовательность импульсов; если же состояние триггера соответствует нулю, то импульсы с выхода триггера не выдаются. Заметим, что динамические триггеры на электронных лампах требуют вдвое меньше ламп, чем статические триггеры. В нашем примере динамического триггера используется один сердечник с четырьмя обмотками, не считая тактовой обмотки. Одна обмотка представляет собой единичный вход, вторая обмотка (обмотка запрета) — нулевой вход, третья обмотка — обратную связь от выходной обмотки, являющейся четвертой обмоткой сердечника. Если требуется установить триггер в состояние «1», то подается импульс на единичный вход; если требуется установить триггер в состояние «О», то подается импульс на нулевой вход.

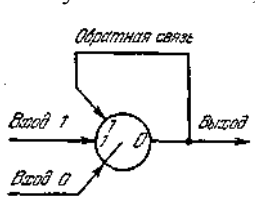



Рис. 44. Динамический триггер на ферритовом сердечнике.

В первом случае на выходе триггера получится последовательность импульсов, во втором случае импульсы на выходе триггера будут отсутствовать.

Импульс, поданный на единичный вход, перебросит сердечник в состояние «1». Тактовый импульс перебросит

сердечник в состояние «О» и вызовет появление сигнала в выходной обмотке, а следовательно, и в обмотке обратной связи. Импульс в обмотке обратной связи вновь перебросит сердечник в состояние «1» и подготовит его к следующему тактовому импульсу. Тактовый импульс перебросит сердечник в состояние «О» и вновь выдаст импульс на выходе и т. д. Таким образом, на выходе схемы будет выдаваться последовательность импульсов.

При подаче импульса на нулевой вход, т. е. на обмотку запрета, нейтрализующую действие обмотки обратной связи, сердечник будет постоянно оставаться в нулевом состоянии и на выходе импульсы будут отсутствовать.

Таким образом, триггер будет фиксировать два различных устойчивых состояния, соответствующих коду «1» и коду «О».

Нами рассмотрены примеры простейших схем на электронных лампах, полупроводниковых диодах и триодах и магнитных сердечниках. Из этих схем могут быть составлены более сложные логические и счетные схемы, используемые в электронных цифровых машинах.

## § 15. Комбинированные электронные логические схемы

Более сложные логические преобразования в вычислительных и управляющих схемах выполняются электронными схемами, составленными из рассмотренных выше элементарных схем (инверторов, схем совпадения, собирательных схем).

**1. Двойной вентиль.** Схема двойного вентиля, представляющая собой многополюсник с четырьмя входами и одним выходом, приведена на рис. 45, *а.*

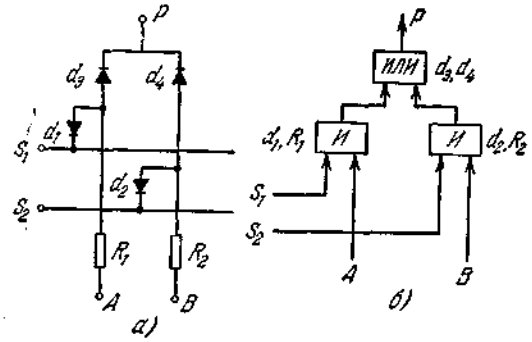



Рис. 45. Двойной вентиль.

На два входа *А* и *В* поступают сигналы, которые должны пройти на выход *Р.* В зависимости от того, на какой из управляющих входов, $S_1$или $S_2$ подан управляющий сигнал, на выход *Р* будет пропущен либо сигнал *А,* либо сигнал *В.* Если положительный управляющий сигнал высокого уровня подан на вход $S_1$ а на вход $S_2$ подан сигнал низкого уровня, то на выход *Р* пройдет сигнал Л, **а** сигнал *В* не появится на входе выпрямителя $d_4$ так как сопротивление $R_2$ значительно больше прямого сопротивления диода $d_2$ и почти все падение напряжения, соответствующее сигналу *В,* будет приходиться на сопротивление $R_2$ Если управляющий сигнал подан на входе $S_2$ , то, наоборот, на выход пройдет сигнал *В,* а сигнал *А* пройдет на шину $S_1$ через проводящий выпрямитель $d_1$. Выпрямители $d_3$ и $d_4$ служат для того, чтобы исключить обратное влияние на работу схемы сигнала, прошедшего на выход.

На рис. 45, *б* изображена функциональная схема двойного вентиля, составленная из схем совпадения и собирательных схем. Схема работает по следующей логической формуле:

$$P=(A \wedge S_1) \vee (B \wedge S_2) \qquad \text{(II.32)}$$

**2. Одноразрядный преобразователь** (рис. 46, *а)* можно получить, соединяя схему инверторов и двойных вентилей.

Сигнал, поданный на вход *А,* проходит на выход *Р* в прямом или инвертированном виде в зависимости от того, на какой из двух управляющих входов, $S_1$ или $S_2$, подан управляющий сигнал.

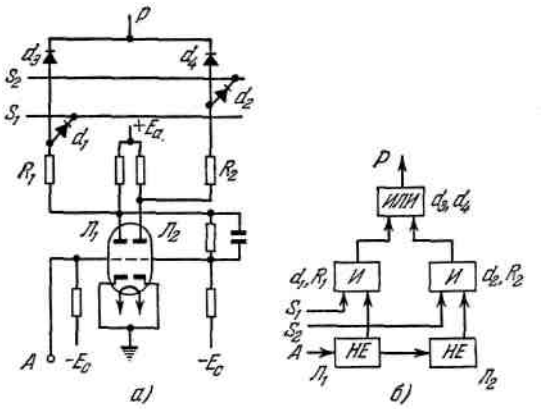



Рис. 46. Одноразрядный преобразователь.

Функциональная схема одноразрядного преобразователя, составленная из инверторов, схем совпадения и собирательной схемы, показана на рис. 46, *б.* Работа одноразрядного преобразователя описывается следующей логической формулой:

$$P=(\overline{A} \wedge S_1) \vee (B \wedge S_2). \qquad \text{(II.33)}$$

**3. Избирательная схема.** На рис. 47 приведена схема, которая работает таким образом, что каждой комбинации положительных сигналов, поданных на входы *A и B,* соответствует одна и **только** одна комбинация сигналов на выходах

$$P, \mathrm{Q}, R, S.$$

Могут быть получены четыре различные комбинации входных сигналов ($A$ = 0 и $B$ = 0; $A$ = 1 и $B$ = 0; $A$ = 0 и $B$ = 1; $A$ =1 и $B$= 1), и для каждой из этих комбинаций выходной сигнал будет появляться только на одном из выходов. Например, в случае комбинации входных сигналов $A$ — О, $B$ = 0 лампы $Л_1$ и $Л_3$ будут заперты (благодаря наличию на сетках этих ламп большого отрицательного смещения, обусловленного напряжением —$E_c$). Лампы $Л_2$ и $Л_4$ будут, наоборот, полностью отперты, и потенциал на анодах этих ламп будет низким.

Через проводящие выпрямители $d_3$, $d_6$, $d_7$, $d_8$ ток от шин $Ш_2$, $Ш_3$, $Ш_4$ будет протекать к анодам ламп $Л_2$ и $Л_4$, имеющим низкий потенциал, а шина $Ш_1$, соединенная через $d_1$ и $d_2$ с анодами ламп $Л_1$ и $Л_3$, имеющими высокий потенциал, будет иметь также высокий потенциал. Таким образом, комбинации входных сигналов $A$ = 0, $B$ = 0

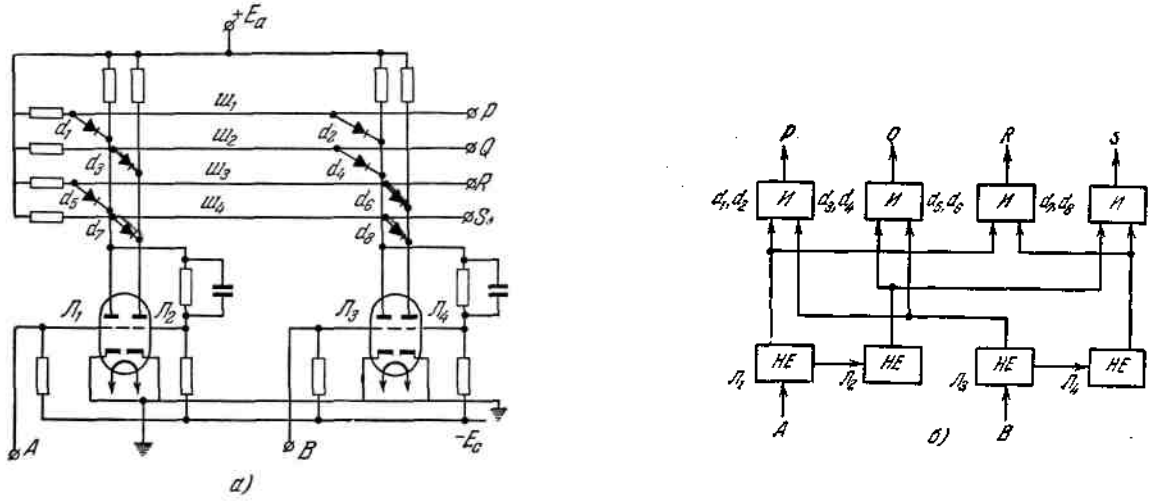



Рис. 47. Избирательная схема.

соответствует наличие сигнала только на выходе *P.* Функциональная схема рассмотренной избирательной схемы показана на рис. 47, *б.* Аналогичная картина имеет место и для других комбинаций входных сигналов.

Работа избирательной схемы описывается следующими логическими формулами:

$$\begin{aligned} P&=\overline{A} \wedge \overline{B}, \\ Q&=A \wedge \overline{B} \\ R&=\overline{A} \wedge B, \\ S&=A \wedge B. \end{aligned} \qquad \text{(II..34)}$$

Часто при выполнении операций над двоичными числами приходится производить сдвиг всех разрядов числа на определенное количество разрядов вправо или влево. Например, при выполнении операции нормализации двоичных чисел производится сдвиг мантиссы влево (см. стр. 57). Наоборот, при выравнивании порядков мантисса меньшего числа сдвигается вправо (см. стр. 90).

**4. Схема сдвигателя.** Схема сдвигателя, предназначенная для сдвига вправо четырехразрядных двоичных чисел, приведена на рис. 48. Сдвигаемое число подается на входы $A_1$, $A_2$, $A_3$, $A_4$.

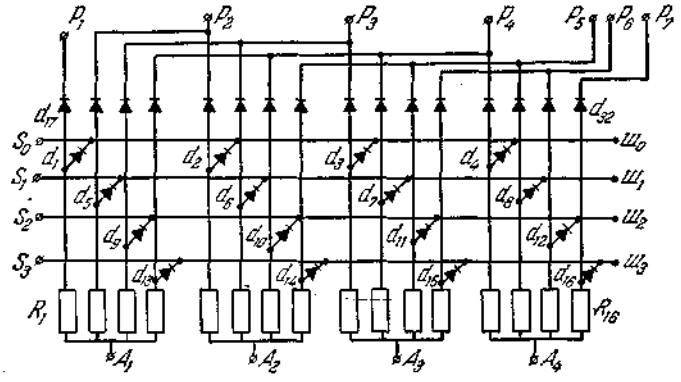



Рис. 48. Схема сдвигателя.

В зависимости от того, на какой из четырех управляющих входов, $S_0$, $S_1$, $S_2$ или $S_3$, будет подан управляющий сигнал высокого уровня, число пройдет без сдвига или будет сдвинуто соответственно на 1, 2 или 3 разряда вправо.

На приведенной схеме сдвигатель построен целиком на выпрямителях. При отсутствии сигналов на всех управляющих входах $S_0$, $S_1$, $S_2$, $S_3$ сигналы высокого уровня, поданные на входы $A_1$, $A_2$, $A_3$, $A_4$ через сопротивления $R_1$ — $R_{16}$, которые по своей величине больше прямого сопротивления выпрямителей $d_1$ — $d_{16}$, и затем через проводящие выпрямители $d_1$, $d_2$, $d_3$, $d_4$, ..., $d_{16}$ пройдут на шины $Ш_0$, $Ш_1$, $Ш_2$, $Ш_3$, и так как все падение напряжения будет приходиться на сопротивления $R_1$ — $R_{16}$, то ни на одном из выходов сдвигателя сигналов не будет. При подаче сигнала высокого уровня, например, на управляющий вход $S_2$ выпрямители $d_9$, $d_{10}$, $d_{11}$, $d_{12}$ проводить ток не будут и сигналы со входов $A_1$, $A_2$, $A_3$, $A_4$ пройдут на выходы $P_3$, $P_4$, $P_5$, $P_6$, т. е. произойдет сдвиг поданного на вход числа на два разряда вправо (в сторону младших разрядов). Если управляющий сигнал будет подан на вход So, то число пройдет без сдвига, т. е. появится на выходах $P_1$, $P_2$, $P_3$, $P_4$.

Работа сдвигателя описывается следующей группой логических формул:

$$\left.\begin{aligned}
P_1 &= A_1 \wedge S_0 \wedge \overline{S}_1 \wedge \overline{S}_2 \wedge \overline{S}_3;\\
P_2 &= (A_1 \wedge \overline{S}_0 \wedge \overline{S}_1 \wedge \overline{S}_2 \wedge \overline{S}_3) \vee (A_2 \wedge S_0 \wedge \overline{S}_1 \wedge \overline{S}_2 \wedge \overline{S}_3);\\
P_3 &= (A_1 \wedge \overline{S}_0 \wedge \overline{S}_1 \wedge S_2 \wedge \overline{S}_3) \vee (A_2 \wedge \overline{S}_0 \wedge S_1 \wedge \overline{S}_2 \wedge \overline{S}_3) \vee (A_3 \wedge S_0 \wedge \overline{S}_1 \wedge \overline{S}_2 \wedge \overline{S}_3);\\
P_4 &= (A_1 \wedge \overline{S}_0 \wedge \overline{S}_1 \wedge \overline{S}_2 \wedge S_3) \vee (A_2 \wedge \overline{S}_0 \wedge \overline{S}_1 \wedge S_2 \wedge \overline{S}_3) \vee (A_3 \wedge \overline{S}_0 \wedge S_1 \wedge \overline{S}_2 \wedge \overline{S}_3) \vee (A_4 \wedge S_0 \wedge \overline{S}_1 \wedge \overline{S}_2 \wedge \overline{S}_3);\\
P_5 &= (A_2 \wedge \overline{S}_0 \wedge \overline{S}_1 \wedge \overline{S}_2 \wedge S_3) \vee (A_3 \wedge \overline{S}_0 \wedge \overline{S}_1 \wedge S_2 \wedge \overline{S}_3) \vee (A_4 \wedge \overline{S}_0 \wedge S_1 \wedge \overline{S}_2 \wedge \overline{S}_3);\\
P_6 &= (A_3 \wedge \overline{S}_0 \wedge \overline{S}_1 \wedge \overline{S}_2 \wedge S_3) \vee (A_4 \wedge \overline{S}_0 \wedge \overline{S}_1 \wedge S_2 \wedge \overline{S}_3);\\
P_7 &= A_4 \wedge \overline{S}_0 \wedge \overline{S}_1 \wedge \overline{S}_2 \wedge S_3.
\end{aligned}\right\} \quad \text{(II.35)}$$

Эти формулы весьма просто получаются при рассмотрении порядка работы схемы. Каждое выражение для $P_1 ... P_7$ представляет собой дизъюнкцию конъюнкций, т. е. дизъюнктивную нормальную форму. (Для $P_1$ и $P_7$ эта форма имеет всего одну конъюнкцию.) Каждый конъюнктивный член соответствует одному случаю появления сигнала на данном выходе, а все конъюнкции, входящие в данную дизъюнктивную форму, представляют собой все случаи, когда возможно появление сигнала на данном выходе.

Выпрямители $d_{17}$ — $d_{32}$ служат для того, чтобы исключить обратное влияние выходных сигналов на работу схемы. Сопротивления $R_1$ — $R_{16}$ вместе с соответствующими выпрямителями $d_1$ — $d_{16}$ образуют вентили, работающие при совпадении сигналов высокого уровня. Эти вентили объединены в четыре группы, соответствующие наличию четырех входов $A_1$, $A_2$, $A_3$, $A_4$. Кроме того, сопротивления $R_1$ — $R_{16}$ необходимы для ограничения тока через выпрямители и для поддержания достаточно высокого уровня входных сигналов, так как в противном случае все входные сигналы независимо от наличия управляющих сигналов $S_0$, $S_1$, $S_2$, $S_3$ прошли бы через открытые в данный момент выпрямители.

## § 16. Синтез избирательных логических схем

В предыдущем параграфе мы рассмотрели некоторые логические электронные схемы, используемые в электронных цифровых машинах, и познакомились с применением алгебры логики для описания порядка работы этих схем.

Однако алгебра логики может применяться не только для функционального описания уже готовых схем с целью их анализа. Алгебра логики применяется также при решении задач *синтеза схем,* т. е. построения схем на основе заданных условий и выполняемых функций. Рассмотрим несколько простых примеров применения алгебры логики для синтеза логических электронных схем.

Общий порядок построения электронных логических схем при помощи алгебро-логических формул сводится к следующему.

Сначала на основании анализа функций, которые должна выполнять схема, заданных либо в словесной форме, либо в виде таблиц двоичных функций, составляются логические формулы, описывающие работу схемы. Затем производится анализ полученных логических выражений с целью выбора такого порядка построения схемы, который бы обеспечивал наименьшее количество элементов в схеме. Для этого просматриваются логические выражения с тем, чтобы выявить одинаковые повторяющиеся члены или части сложного логического выражения, которые можно реализовать при помощи одних и тех же частных схем.

Если, например, надо построить одну сложную электронную схему, которая должна обеспечить реализацию нескольких логических выражений (т. е. должна иметь несколько выходов), то необходимо выявить в этих логических выражениях по возможности более крупные одинаковые члены, которые могли бы быть реализованы одними и теми же электронными схемами.

Если, в частности, имеется возможность получать готовыми отрицания основных логических переменных, например с дополнительных выходов триггеров, то обычно целесообразно приводить логические выражения к такому виду, чтобы знаки отрицания относились к основным переменным. Если же для получения отрицаний

основных переменных необходимы специальные схемы (инверторы), то целесообразно иметь логические выражения в таком виде, чтобы отрицания относились к возможно более крупным членам и чтобы самих операций отрицания было по возможности меньше.

В настоящее время еще нет достаточно удобной общей методики выбора оптимальных схем. В конечном счете при построении практических схем решающее значение имеет наличие в распоряжении конкретных вариантов электронных схем для реализации логических операций. Часто бывает так, что сложная логическая операция может быть реализована сравнительно простой специальной схемой, в то время как при расчленении этой операции на простейшие потребуется значительно большее количество аппаратуры. Кроме того, при построении реальных электронных схем приходится учитывать ограничения, связанные с необходимостью усиления и формирования сигналов, которые при прохождении ряда элементов обычно искажают свою форму и ослабевают.

Однако изложенные выше общие соображения, с учетом сделанных оговорок, позволяют во многих случаях наметить правильный подход к выбору схем и получить схемы со сравнительно небольшим количеством элементов.

Рассмотрим несколько примеров построения простейших избирательных схем при помощи аппарата алгебры логики.

**1. Первый пример.** Пусть требуется построить избирательную схему, которая решала бы следующую задачу из комбинаторики: сигнал на выходе должен появляться только в том случае, если на любые два из трех входов (но только на два!) поступают сигналы. Практическое применение этой схемы возможно, например, для контроля за работой какого-либо устройства, состоящего из трех отдельных агрегатов, причем по условиям работы устройства требуется, чтобы всегда действовали два и только два из трех агрегатов.

Имеем три входа, т. е. три основных высказывания *A, B, C,* которые дают $2^3 = 8$ различных комбинаций; при этом в трех случаях из восьми выполняется требуемое условие: две из трех переменных имеют значение 1.

Рассмотрим сначала две переменные, например *A* и *B*, и составим из них логическое выражение, которое равно 1, т. е. истинно, в том случае, когда только одно из двух высказываний *A* и *B* истинно. Это обеспечивается применением отрицания равнозначности

$$A \approx B .$$

Если теперь присоединим третью переменную *C*, то желаемые случаи (наличие сигналов на двух из трех входов) будут иметь место при истинности выражения

$$( A \approx B ) \wedge C.$$

Если теперь повторить рассуждение, взяв сначала переменные *A* и *C* и присоединив затем переменную *B,* то получим еще одно выражение, также удовлетворяющее заданному условию:

$$( A \approx C ) \wedge B.$$

Оба рассмотренных варианта охватывают все возможные случаи работы искомой схемы. Они дают в трех из восьми возможных комбинаций переменных *A, B, C* истинные значения, причем в этих трех случаях обязательно две и только две переменные истинны (равны 1). Таким образом, получим следующую формулу, описывающую работу схемы:

$$P = [( A \approx B ) \wedge C] \vee [( A \approx C ) \wedge B]. \qquad \text{(II.36)}$$

Проверка этой формулы дана в таблице 22.

Т а б л и ц а 22. Проверка формулы, описывающей работу схемы

| A | B | C | Требуемые значения | Получаемые по (II.36) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |

Преобразуем выражение (II.36), используя формулу 8) таблицы 21,

$$P= \{[(A \wedge \overline{B} ) \vee ( \overline{A} \wedge B)] \wedge C\} \vee \{[(A \wedge \overline{C} ) \vee ( \overline{A} \wedge C)\} \wedge B\}.$$

Раскрываем фигурные и квадратные скобки (используем формулы 14) и 15) таблицы 20), получим следующее

выражение:

$$P = (A \wedge \overline{B} \wedge C) \vee (\overline{A} \wedge B \wedge C) \vee (A \wedge B \wedge \overline{C}) (\overline{A} \wedge B \wedge C)$$

Так как второй и последний члены одинаковы, то один из них можно отбросить:

$$P=(\overline{A} \wedge B \wedge C) \vee (A \wedge \overline{B} \wedge C) \vee (A \wedge B \wedge \overline{C}). \qquad \text{(II.37)}$$

На рис. 49, *а* показана функциональная схема и на рис. 49, *б* — принципиальная схема, соответствующая формуле (11.37). При построении этих схем предполагается, что значения отрицаний основных высказываний могут быть получены готовыми (например, с дополнительных выходов триггеров).

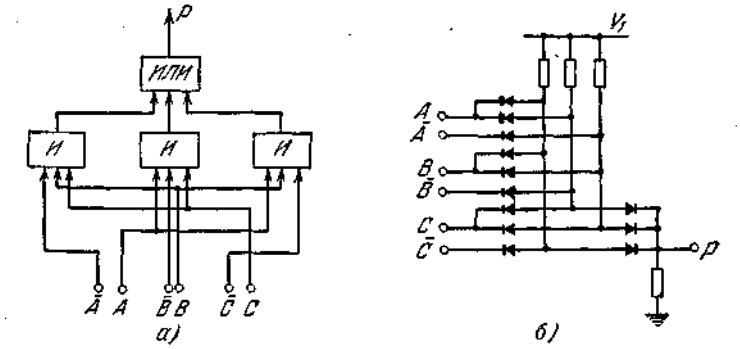



Рис. 49. Реализация логического выражения (II.37).

**2. Второй пример.** Рассмотрим теперь построение избирательной схемы, которая бы решала такую задачу: сигнал на выходе (значение 1) появляется, если хотя бы на два (любых) входа из трех поступили сигналы (значения 1).

Также имеем три основные переменные *A, B, C* и требуется составить логическую формулу, которая давала бы истинные значения в тех случаях, когда хотя бы две из трех переменных истинны.

Ясно, что этому требованию будет удовлетворять формула (II.37), если к ней присоединить справа еще один член

$$(A \wedge B \wedge C),$$

т. е. формула вида

$$P= (A \wedge B \wedge C) \vee (A \wedge B \wedge \overline{C}) \vee (A \wedge \overline{B} \wedge C) \vee (\overline{A} \wedge B \wedge C). \qquad \text{(II.38)}$$

Это выражение можно существенно упростить. Присоединяя в соответствии с формулой 20) таблицы 20 к правой части (II.38)

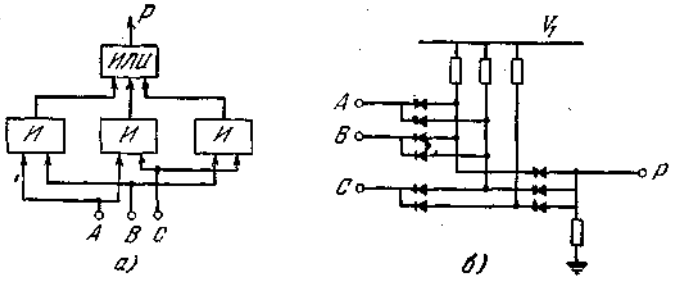



Рис. 50. Реализация логического выражения (II.39).

еще два раза первый член, группируем попарно члены и складываем их (логически), получим:

$$P=[(A \wedge B) \vee (C \vee \overline{C})] \vee [(A \wedge C) \wedge (B \vee \overline{B})] \vee [(B \wedge C) \wedge (A \vee \overline{A})].$$

Выражения $C \vee \overline{C}$, $B \vee \overline{B}$, $A \vee \overline{A}$ как члены конъюнкции **могут** быть отброшены. Окончательно получим:

$$P=(A \wedge B) \vee (A \wedge C) \vee (A \wedge C) \vee (B \wedge C). \qquad \text{(II.39)}$$

Эту формулу, вообще говоря, было бы легко получить и непосредственно из анализа постановки задачи, но приведенный вывод полезен с точки зрения иллюстрации порядка преобразования логических выражений.

На рис. 50, *а* показана функциональная схема, а на рис. 50, *б* — принципиальная схема, соответствующая формуле (II.39).

**3. Третий пример.** Рассмотрим еще один пример построения избирательной схемы. Пусть требуется осуществлять проверку последовательности некоторых двоичных сигналов, причем необходимо не допустить такого случая, чтобы за единицей следовал нуль. Таким образом, допустимыми будут последовательности вида

| $A$ | $B$ | $C$ | $D$ | $E$ | $F$ | $G$ | $H$.... |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 .... |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 .... |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 .... |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 .... |

Недопустимым случаем будет, например, такой:

$$00101111...$$

Таким образом, необходимо построить схему, которая бы обеспечивала попарное сравнение соседних членов ряда по правилу импликации «если, то»

$$A\to B,\ B\to C,\ C\to D,\ D\to E \quad \text{и т. д.}$$

Так как эти условия должны выполняться одновременно, то формула, описывающая работу схемы, будет иметь вид

$$P=(A\to B)\wedge(B\to C)\wedge(C\to D)\wedge... \qquad (II.40)$$

Преобразуем это выражение, используя формулу 1) таблицы 21.

$$P=(\overline{A}\vee B)/\backslash(\overline{B}\vee C)\wedge(\overline{C}\vee D)\wedge... \qquad (II.41)$$

Функциональная схема, соответствующая этой формуле, показана на рис. 51, *а*, а принципиальная схема — на рис. 51, *б*.

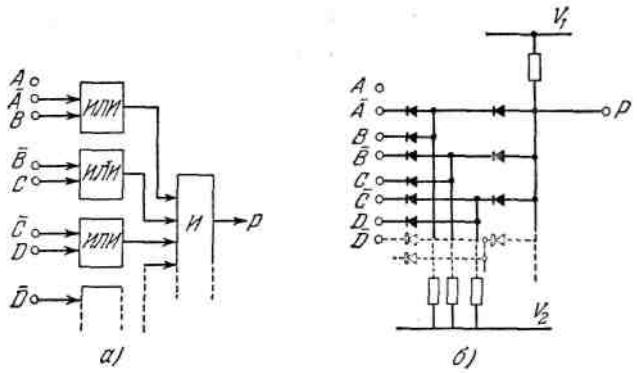



Рис. 51. Реализация логического выражения (11.41).

## § 17. Синтез одноразрядных двоичных сумматоров

**1. Одноразрядный двоичный сумматор на два входа.** Одноразрядный двоичный сумматор (полусумматор) на два входа служит для реализации операции сложения двух двоичных цифр в соответствии с таблицей двоичного сложения 23, где $A$ и $B$ — значения складываемых двоичных разрядов, $P$ — значение результата суммирования в данном разряде и $Q$ — значение переноса в соседний старший разряд.

Сравнивая таблицу 23 с таблицей 17, представляющей все возможные логические функции двух переменных, видим, что величина $P$ получается путем применения к величинам $A$ и $B$ логической операции, называемой отрицанием равнозначности ($\approx$), а величина $Q$ получается как логическое произведение (конъюнкция) $A$ и $B$:

$$P = A\approx B,$$
$$Q = A\wedge B.$$

Таблица 23.
Сложение двух двоичных цифр

| $A$ | $B$ | $P$ | $Q$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 |

Из таблицы 21 получаем для операции отрицания равнозначности следующие выражения через конъюнкции, дизъюнкции и отрицания:

$$A\sim B=(A\wedge\overline{B})\vee(\overline{A}\wedge B)=(\overline{A}\vee\overline{B})\wedge(A\vee B). \qquad (II.42)$$

Ясно, что можно построить несколько вариантов сумматора.

На рис. 52, *а* показана функциональная схема одноразрядного двоичного сумматора, реализующего следующие логические выражения:

$$\left.\begin{aligned} P &= (A\wedge\overline{B})\vee(\overline{A}\wedge B),\\ Q &= A\wedge B. \end{aligned}\right\} \qquad (II.43)$$

В данном случае необходимо реализовать три операции логического умножения, одну операцию логического сложения и две операции отрицания. На рис. 52, *б* и 52, *в* показаны два варианта принципиальных схем, соответствующих приведенной функциональной схеме.

Вариант 52, *б* можно использовать, когда нет возможности одновременно со значениями основных переменных $A$ и $B$ получать и их отрицания $\overline{A}$ и $\overline{B}$, и поэтому в схеме 52, *б* предусмотрены два ламповых инвертора для образования отрицаний $\overline{A}$ и $\overline{B}$.

Вариант 52, *в* целесообразно применять, когда имеется возможность одновременно со значениями основных высказываний $A$ и $B$ получать и их отрицания $\overline{A}$ и $\overline{B}$, например с дополнительных выходов триггеров, как это показано на схеме. В этом случае отпадает необходимость в двух ламповых инверторах и схема собственно сумматора (без учета триггеров) получается значительно проще.

Для случая, когда нет готовых отрицаний основных переменных, можно путем простого преобразования логических выражений получить более простую схему, чем 52, *б.*

Имеем согласно формуле (II.42)

$$P = (\overline{A} \vee \overline{B}) \wedge (A \vee B)$$

Заменяя согласно формуле 4) таблицы 21 $\overline{A} \vee \overline{B}$ на $\overline{A \wedge B}$

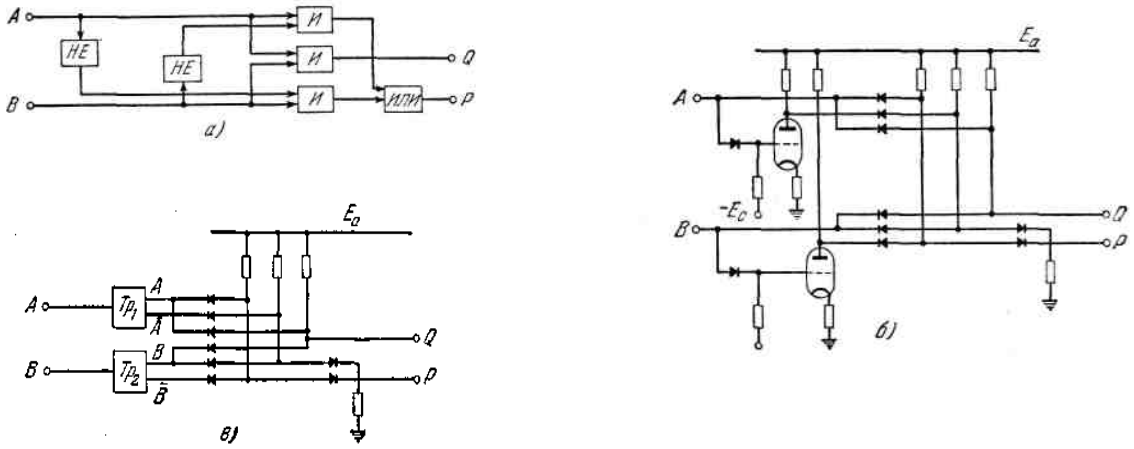



Рис. 52. Одноразрядный двоичный сумматор на два входа.

получим для построения сумматора следующие логические формулы:

$$\left.\begin{aligned} P &= (\overline{A \wedge B}) \wedge (A \vee B), \\ Q &= A \wedge B. \end{aligned}\right\} \qquad \text{(II.44)}$$

Видно, что в схеме должны быть реализованы три операции логического умножения, одна операция логического сложения и одна операция отрицания.

На рис. 53, *а* показана функциональная схема, а на рис. 53, *б* принципиальная схема одноразрядного двоичного сумматора, который имеет меньше элементов (на один инвертор), чем вариант 52, б, но сложнее, чем вариант 52, *в* (без учета триггеров $Тр_1$ и $Тр_2$).

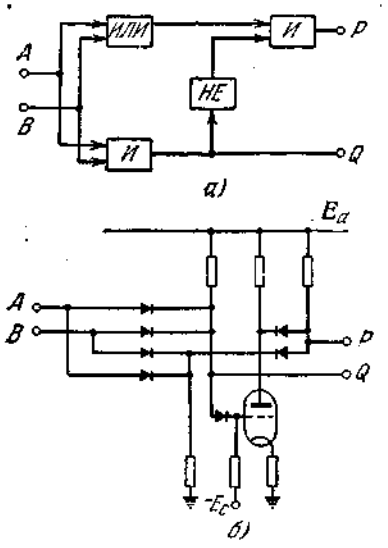



Рис. 53. Одноразрядный двоичный сумматор на два входа.

**2. Одноразрядный двоичный сумматор на три входа.** Одноразрядный двоичный сумматор на три входа служит для сложения трех двоичных цифр в соответствии с таблицей 24 двоичного сложения, где $A$ и $B$ — значения двоичных цифр, представляющих слагаемые данного разряда, $C$ — значение переноса в данный разряд из соседнего младшего разряда, $P$ — значение результата суммирования в данном разряде и $Q$ — значение переноса в соседний старший разряд. Заметим, что одноразрядный двоичный сумматор на три входа, является основным элементом для построения сумматоров для много разрядных двоичных чисел,

которые представляют собой в основном цепочки из последовательно соединенных одноразрядных двоичных сумматоров на три входа.

На основе таблицы 24 двоичного сложения трех цифр составим логические выражения для величин $P$ и $Q$. Величина $P$ принимает значение 1 **в** четырех из восьми случаев — 2), 3), 4), 8), — а **в** остальных случаях — 1), 5), 6), 7) - $P$ равно нулю. Для того чтобы получить формулу для $P$, нужно значения истинности величин $A$, $B$, $C$ для каждого из случаев 2), 3), 4), 8) соединить между собой знаками конъюнкции $\wedge$, так как только в том случае, когда имеют место эти значения $A$, $B$ $C$, величина $P$ принимает значение 1.Полученные для каждого из случаев 2), 3), 4), 8) конъюнкции значений $A$, $B$, $C$ нужно соединить между собой знаками дизъюнкции, так как $P$ принимает значение 1 в любом

Таблица 24.

Двоичное сложение трех цифр

| | $A$ | $B$ | C | $P$ | $Q$ |
|---|---|---|---|---|---|
| 1) | 0 | 0 | 0 | 0 | 0 |
| 2) | 0 | 0 | 1 | 1 | 0 |
| 3) | 0 | 1 | 0 | 1 | 0 |
| 4) | 1 | 0 | 0 | 1 | 0 |
| 5) | 0 | 1 | 1 | 0 | 1 |
| 6) | 1 | 0 | 1 | 0 | 1 |
| 7) | 1 | 1 | 0 | 0 | 1 |
| 8) | 1 | 1 | 1 | 1 | 1 |

из случаев 2), 3), 4), 8).

$$P = (\overline{A} \wedge \overline{B} \wedge C) \vee (\overline{A} \wedge B \wedge \overline{C}) \vee (A \wedge \overline{B} \wedge \overline{C}) \vee (A \wedge B \wedge C) \qquad \text{(II.45)}$$

Аналогичным способом выводим логическое выражение для величины Q, которая истинна (равна единице) в случаях 5), 6), 7), 8).

$$Q = (\overline{A} \wedge B \wedge C) \vee (A \wedge \overline{B} \wedge C) \vee (A \wedge B \wedge \overline{C}) \vee (A \wedge B \wedge C) \qquad \text{(II.46)}$$

На рис. 54 приведена функциональная схема одноразрядного двоичного сумматора на три входа, реализующая составленные логические выражения (II.45) и (II. 46). Эта схема содержит восемь схем совпадения *(и)* на три входа, три инвертора *(не)* для получения отрицаний и две собирательные схемы *(или)* на четыре входа.

Путем преобразования логических выражений (II.45) и (II.46) может быть получена более простая схема одноразрядного двоичного сумматора на три входа.

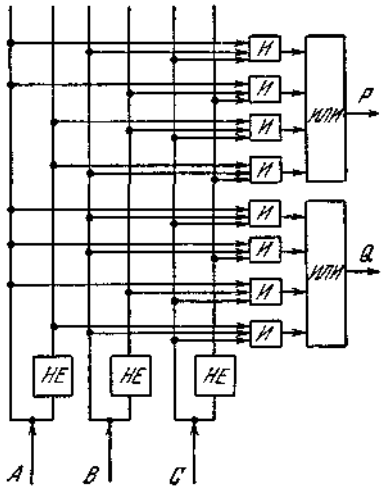


Рис. 54. Однородный двоичный сумматор на

Преобразование будем вести таким образом, чтобы сократить число операций, в частности число отрицаний, обеспечить выполнение отдельных операций над возможно более крупными членами и выделить одинаковые члены, которые могли бы быть реализованы одними и теми же схемами.

Итак, имеем исходные выражения (II.45) и (II.46):

$$P = (\overline{A} \wedge \overline{B} \wedge C) \vee (\overline{A} \wedge B \wedge \overline{C}) \vee (A \wedge \overline{B} \wedge \overline{C}) \vee (A \wedge B \wedge C)$$

$$Q = (\overline{A} \wedge B \wedge C) \vee (A \wedge \overline{B} \wedge C) \vee (A \wedge B \wedge \overline{C}) \vee (A \wedge B \wedge C)$$

Если сравнить формулы (II.46) и (II.38), то нетрудно заметить, что эти формулы совпадают, и таким образом для переменной $Q$ может быть использовано выведенное ранее выражение (II.39):

$$Q = (A \wedge B) \vee (A \wedge C) \vee (B \wedge C)...$$

Проведем необходимые преобразования для переменной $P$. Для этого воспользуемся сначала правилом получения противоположного выражения (стр. 113). Ставим знак отрицания над первыми тремя членами в выражении (II.45), одновременно меняем местами знаки $\wedge$ и $\vee$ и основные высказывания заменяем их отрицаниями, получим:

$$P = (A \wedge B \wedge C) \vee \overline{(A \vee B \vee \overline{C}) \wedge (A \vee \overline{B} \vee C) \wedge (\overline{A} \vee B \vee C)}$$

или

$$P = (A \wedge B \wedge C) \vee [\overline{\overline{A} \vee (\overline{B} \vee C)] \wedge [\overline{B} \vee (A \vee C)] \wedge [\overline{C} \vee (A \vee B)}].$$

Преобразуем отдельно выражение, стоящее под знаком отрицания:

$$[\overline{A} \vee (B \vee C)] \wedge [\overline{B} \vee (A \vee C)] \wedge [\overline{C} \vee (A \vee B)].$$

Перемножив квадратные скобки, получим:

$$(\overline{A} \wedge \overline{B} \wedge \overline{C}) \vee [\overline{A} \wedge \overline{B} \wedge (A \vee B)] \vee [\overline{A} \wedge (A \vee C) \wedge \overline{C}] \vee [\overline{A} \wedge (A \vee C) \wedge (A \vee B)] \vee [(B \vee C) \wedge$$
$$\wedge \overline{B} \wedge \overline{C}] \vee [(B \vee C) \wedge \overline{B} \wedge (A \vee B)] \vee [(B \vee C) \wedge (A \vee C) \wedge \overline{C}] \vee [(B \vee C) \wedge (A \vee C) \wedge (A \vee B)].$$

Следующие постоянно-ложные члены, входящие в дизъюнкцию, можно сократить:

$$\overline{A} \wedge \overline{B} \wedge (A \vee B) = 0$$
$$\overline{A} \wedge \overline{C} \wedge (A \vee C) = 0$$
$$\overline{B} \wedge \overline{C} \wedge (B \vee C) = 0$$

Получим выражение в следующем виде:

$$(\overline{A} \wedge \overline{B} \wedge \overline{C}) \vee [(A \vee B) \wedge (A \vee C) \wedge (B \vee C)] \vee \overline{A} \wedge (A \vee B) \wedge (A \vee C)] \vee$$
$$\vee [\overline{B} \wedge (A \vee B) \wedge (B \vee C)] \vee [\overline{C} \wedge (A \vee C) \wedge (B \vee C)].$$

Легко показать справедливость следующих равенств:

$$(A \vee B) \wedge (A \vee C) \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C) \vee (B \wedge C), \qquad \text{(II.47)}$$
$$\overline{A} \wedge (A \vee B) \wedge (A \vee C) = \overline{A} \wedge B \wedge C \qquad \text{(II.48)}$$
$$\overline{B} \wedge (B \vee A) \wedge (B \vee C) = A \wedge \overline{B} \wedge C \qquad \text{(II.49)}$$
$$\overline{C} \wedge (C \vee B) \wedge (C \vee A) = A \wedge B \wedge \overline{C} \qquad \text{(II.50)}$$

Используя эти равенства, получим наше выражение в виде следующей дизъюнктивной нормальной формы:

$$(\overline{A} \wedge \overline{B} \wedge \overline{C}) \vee (A \wedge B) \vee (A \wedge C) \vee (B \wedge C) \vee (\overline{A} \wedge B \wedge C) \vee (A \wedge \overline{B} \wedge C) \vee (A \wedge B \wedge \overline{C}) \qquad \text{(II.51)}$$

Ясно, что три последние трехчленные конъюнкции могут быть отброшены, так как в эту же дизъюнктивную нормальную форму (II.51) входят три соответствующие двучленные конъюнкции, которые делают излишними трехчленные конъюнкции. Выражение под знаком отрицания примет следующий вид:

$$\overline{(\overline{A} \wedge \overline{B} \wedge \overline{C}) \vee (A \wedge B) \vee (A \wedge C) \vee (B \wedge C)}..$$

Для того чтобы уменьшить число операций отрицания, разорвем знак отрицания на две части согласно формуле 5) таблицы 21:

$$\overline{\overline{A} \wedge \overline{B} \wedge \overline{C}} \wedge \overline{(A \wedge B) \vee (A \wedge C) \vee (B \wedge C)}$$

Применяя к первому члену правило образования противоположного выражения (стр. 113), получим:

$$(A \vee B \vee C) \wedge \overline{(A \wedge B) \vee (A \wedge C) \vee (B \wedge C)}$$

Окончательно выражение для переменной $P$ получим в следующем виде:

$$P = (A \wedge B \wedge C) \wedge [(A \vee B \vee C) \wedge \overline{(A \wedge B) \vee (A \wedge C) \vee (B \wedge C)}] \qquad \text{(II.52)}$$

Для переменной $Q$ мы ранее получили выражение (II.39):

$$Q = (A \wedge B) \vee (A \wedge C) \vee (B \wedge C).$$

Функциональная схема одноразрядного двоичного сумматора на три входа, построенная на основе полученных формул (II.52) и (II.39), показана на рис. 55.

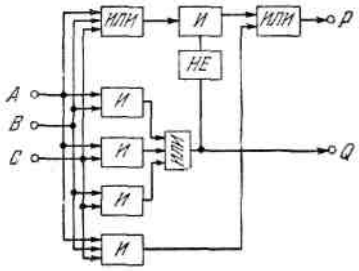


Рис. 55. Одноразрядный двоичный сумматор на три входа

Эта схема содержит пять схем совпадения *(и)* на два входа, две собирательные схемы *(или)* на три входа, одну собирательную схему на два входа и один инвертор *(не)*.

Таким образом, видно значительное упрощение функциональной схемы как по количеству элементарных схем, входящих в эту схему, так и по числу входов в элементарных схемах.

Заметим, что уменьшение количества входов обычно приводит к упрощению соответствующих схем.

Рассмотренные примеры преобразования логических выражений с целью упрощения соответствующих электрических схем иллюстрируют по существу «способ попыток», требующий большой изобретательности работника. Этот способ полезен при инженерном проектировании переключательных схем. Кроме него, известны различные теоретические способы получения схем с минимальным количеством элементов (аналитические, табличные, геометрические), которые применимы при определенных ограничениях (использование только элементов *и* и *или,* исключение операцией отрицания и т. д.). Мы эти методы рассматривать не будем.

# ГЛАВА III
# ТЕХНИЧЕСКИЕ ПРИНЦИПЫ УСТРОЙСТВА ЭЛЕКТРОННЫХ ЦИФРОВЫХ МАШИН

Во введении (§ 2) была приведена упрощенная блок-схема электронной цифровой программно-управляемой машины и рассмотрен принцип программного управления. Мы уже знаем, что любая электронная программно-управляемая машина состоит из трех основных частей: арифметического устройства, запоминающего устройства и устройства управления. Кроме того, в состав машины входят специальные устройства для ввода информации (чисел и команд) в машину и для вывода информации из машины. В настоящей главе мы познакомимся с техническими принципами построения и работы устройств машины.

## § 18. Принципы построения и типы машин

Электронные цифровые машины по своей конструкции делятся на несколько типов.

**1. Машины параллельного и последовательного действия. Фиксированная и плавающая запятая.** Во-первых, бывают машины *параллельного действия* и машины *последовательного действия.* Выбор того или иного принципа построения машины определяется типами применяемых запоминающих и арифметических устройств. В машинах параллельного действия все числа передаются внутри машины одновременно всеми своими разрядами по соответствующему числу параллельных каналов; выполнение операций над числами в арифметическом устройстве ведется одновременно над всеми разрядами; выборка и запись чисел в запоминающие устройства также осуществляется одновременно всеми разрядами. Этот принцип обеспечивает высокую скорость работы, но приводит к усложнению аппаратуры по сравнению с последовательным принципом, при котором передача чисел и выполнение операций над ними осуществляются последовательно разряд за разрядом.

Далее, машины делятся на машины с *фиксированной запятой* и машины с *плавающей запятой* (см. § 6). Выбор того или иного принципа построения машины оказывает в основном влияние на конструкцию арифметического устройства, которое оказывается более сложным в машинах с плавающей запятой за счет введения дополнительных

блоков для действий с порядками чисел. При фиксированном положении запятой существенно упрощается конструкция машины, но зато значительно сильнее ограничивается диапазон изменения представимых чисел (см. § 6) по сравнению с машинами с плавающей запятой и, следовательно, усложняется программирование задач.

Для того чтобы обеспечить в машине с фиксированной запятой такие же удобства при программировании и решении задач, как и в машине с плавающей запятой, необходимо иметь количество разрядов для представления числа примерно в полтора раза большее по сравнению с количеством разрядов, отведенных для представления мантиссы в машине с плавающей запятой.

Применение различных принципов построения машин (параллельные или последовательные, с фиксированной или плавающей запятой) оказывает влияние на количество аппаратуры, скорость выполнения операций, удобство технической эксплуатации.

В машинах с плавающей запятой при выполнении операций затрачивается дополнительное время на выравнивание порядков и нормализацию, а в машинах с фиксированной запятой увеличивается время выполнения операций (особенно умножения и деления) за счет увеличения количества разрядов в числах.

Для машин параллельного действия в зависимости от классов задач, для решения которых предназначается машина, может оказаться в одних случаях более выгодным использование плавающей запятой, в других — фиксированной.

Для машин последовательного действия увеличение количества разрядов для представления чисел почти не приводит к увеличению оборудования, но понижает быстродействие машины. Применение плавающей запятой в машинах последовательного действия приводит к увеличению оборудования за счет усложнения устройства управления примерно на 10%. Таким образом, в машинах последовательного действия выгоднее применение фиксированной запятой, так как обычно одним из основных требований является для этих машин простота конструкции и эксплуатации, а не высокое быстродействие.

**2. Адресность машин.** Наконец, машины разделяются в зависимости от принятой *адресности* команд. Наиболее распространенными являются машины с *трехадресной* системой команд. Трехадресная команда имеет в своем составе три адреса, которые распределяются следующим образом:

1) адрес первого числа;
2) адрес второго числа;
3) адрес результата операции.

Для выполнения одной трехадресной команды требуется четырехкратное обращение к запоминающему устройству: один раз для выборки команды, второй раз для выборки первого числа (по первому адресу), третий раз для выборки второго числа (по второму адресу) и четвертый раз для записи в запоминающее устройство результата операции (по третьему адресу).

Большое распространение имеет также *одноадресная* система команд. Она принята, например, в советской серийной машине *Урал,* относящейся к классу малых машин и предназначенной для сравнительно небольших расчетов. Уменьшение количества адресов в команде упрощает конструкцию машины, так как при этом команда может быть представлена меньшим количеством разрядов. Однако для программирования трехадресные команды удобнее, чем одноадресные.

Следует заметить, что скорость выполнения машиной вычислений почти не зависит от выбранной адресности команд, так как общее время всех обращений к запоминающему устройству, необходимых для выполнения каждой полной операции, не зависит от адресности команд. Сравним, например, два случая: трехадресные команды и одноадресные команды. При трехадресных командах для выполнения одной операции необходимо четырехкратное обращение к запоминающему устройству. При одноадресной системе команд для выполнения каждой команды необходимо дважды обращаться к запоминающему устройству: один раз для выборки очередной команды и второй раз для выборки числа по заданному в команде адресу. Практика показывает, что при одноадресной системе команд для выполнения определенного объема вычислений требуется в среднем в два раза больше команд, чем при трехадресной системе. Поэтому общее количество обращений к запоминающему устройству, необходимое для выполнения полной операции, включая запись результата, оказывается также равным четырем.

Быстродействие машины зависит от двух основных факторов: от времени, затрачиваемого на выполнение операций арифметическим устройством, и от времени, необходимого для обращения к запоминающему устройству. Важнейшей задачей, возникающей при разработке машины, является уменьшение этих времен и их рациональное согласование.

Не имеет смысла повышать быстродействие арифметического устройства, если оно связано в работе со сравнительно медленно действующим запоминающим устройством и основное время такта работы машины тратится на выборку чисел из запоминающего устройства. Справедливо также и обратное положение.

Структура машины в значительной степени зависит от выбранной адресности команд, от состава элементарных операций, от выбранных способов осуществления сложных операций (умножение, деление, извлечение корня и др.). Выполнение этих операций по стандартным подпрограммам упрощает устройство машины, но увеличивает время, затрачиваемое на решение задач, и требует либо создания специальных запоминающих устройств для хранения подпрограмм, либо расширения емкости основных запоминающих устройств.

## § 19. Устройство управления машины

**1. Назначение и состав устройства управления машины.** Следует заметить, что в отличие от арифметических и запоминающих устройств, имеющих определенную автономность в машинах, устройство управления машины органически связано со всей машиной и воплощает в себе особенности ее структуры. Отсюда следует, во-первых, что структурная схема всей машины и устройства управления должны рассматриваться совместно и, во-вторых, что практически невозможно рассматривать абстрактно устройство управления машин вообще, безотносительно к какому-нибудь конкретному типу машин. Мы рассмотрим характерный пример построения структурной схемы машины и устройства управления применительно к трех-адресной машине параллельного действия. Изучение этой схемы, являющейся в значительной мере типовой, позволит уяснить общие черты структуры и особенности построения устройств управления программно-управляемых машин.

В общем случае устройство управления машины включает в себя центральное устройство управления и ряд устройств управления отдельными устройствами машины: устройство управления арифметическим устройством, устройство управления запоминающим устройством (заметим, что центральное устройство управления машины и устройство управления запоминающим устройством тесно связаны друг с другом; мы в данном случае под последним подразумеваем ту часть устройства управления, которая связана с работой только запоминающего устройства, например управление восстановлением информации, непосредственно процессами записи и выдачи данных и др.).

В этом параграфе мы будем рассматривать в основном работу центрального устройства управления.

Устройство управления в общем случае предназначается для управления автоматической работой машины и, в частности, для обеспечения следующих функций:

1) управления автоматическим вводом программы и исходных данных в машину; управления выводом информации из машины;
2) управления выбором команд программы, выдаваемых для исполнения;
3) управления исполнением команд, в том числе выборкой чисел, выполнением требуемых операций и записью результатов **в** запоминающее устройство;
4) обеспечения возможности контроля и управления работой машины со стороны оператора.

После сделанных общих замечаний перейдем к рассмотрению структурной схемы и устройства управления электронной цифровой машины. Упрощенная структурная схема машины и устройства управления для трехадресной машины параллельного действия представлена на рис. 56. На этом же рисунке показаны основные связи устройства управления с запоминающим и арифметическим устройством.

Устройство сигнализации и ручного управления также относится к системе управления машиной, хотя и не входит в центральное устройство управления. Это устройство предназначено для систематического контроля за работой машины, оперативного изменения хода решения задачи, а также для наладки и проверки машины.

Функционально центральное устройство управления можно разделить на четыре блока.

1. Блок формирования такта работы машины, который предназначается для выработки управляющих сигналов, определяющих работу машины в процессе выполнения каждой команды.

2. Блок выработки адреса команды. Этот блок предназначен для обеспечения последовательной выборки команд программы из запоминающего устройства. Кроме того, блок выработки адреса команды должен обеспечивать возможность изменения последовательности выполнения команд программы при выполнении команд так называемых условных и безусловных переходов. Этот блок позволяет перейти в любое место программы, предусмотренное соответствующей командой перехода. Основной частью этого блока является счетчик адресов команд, фиксирующий в каждом такте адрес очередной команды.

3. Блок фиксации команды. Этот блок предназначен для фиксации выданной из запоминающего устройства очередной команды и выдачи отдельных адресов этой команды (первого, второго и третьего — в случае трехадресной команды) для обращений к запоминающему устройству по этим адресам. Основной частью блока фиксации команд является регистр команд, состоящий из регистров адресов команды и регистра кода операции. Регистр кода операции этого блока фиксирует код операции и выдает его для расшифровки в дешифратор кодов операций.

4. Блок выбора ячейки по заданному адресу предназначен для выборки чисел по заданному адресу. Например, в случае запоминающих устройств на электронно-лучевых трубках этот блок преобразует условный двоичный код адреса ячейки запоминающего устройства в управляющие напряжения, обеспечивающие нужное отклонение электронного луча и выборку из заданной ячейки числа или команды. Заметим, что этот блок иногда относят к местному управлению запоминающего устройства и не включают в состав центрального устройства управления машины.

5. Дешифратор кодов операций представляет собой избирательную схему для расшифровки кодов операции. Этот блок по коду операции, поступающему из блока фиксации команды, вырабатывает сигналы на соответствующих выходных проводах, которые обеспечивают выполнение требуемой операции определенными устройствами машины (в основном арифметическим устройством).

**2. Порядок работы устройства управления.** На рис. 56 каждый блок устройства разделен на несколько характерных каскадов, имеющих принципиальное значение независимо от величины и количества аппаратуры в

каскадах.

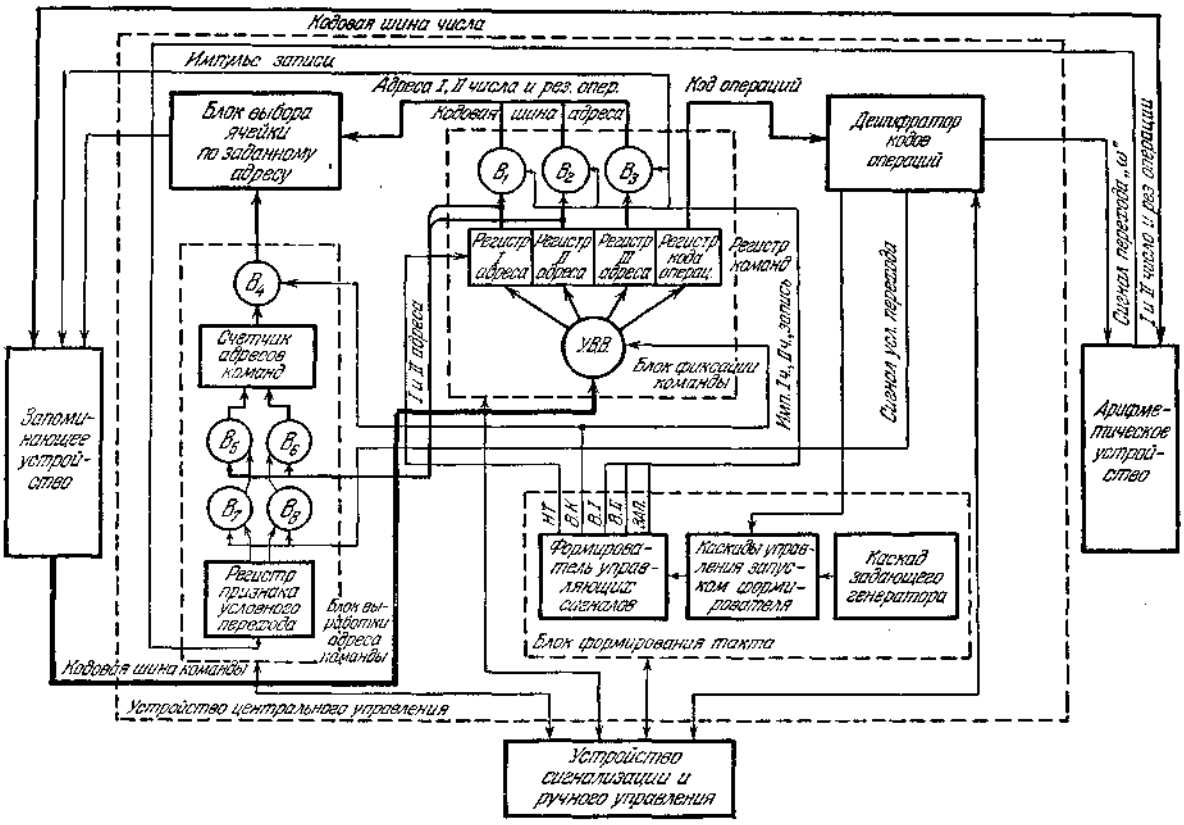



Рис.56 Упрощенная структурная схема трехадресной машины параллельного действия.

Основным синхронизирующим элементом центрального устройства управления машины является каскад задающего генератора, который вырабатывает непрерывную последовательность импульсов заданной частоты. Эти импульсы, пройдя через каскады управления запуском формирователя, поступают в формирователь, который формирует основной такт машины.

*Такт* машины состоит из строго определенной серии управляющих импульсов, обеспечивающих необходимую последовательность работы отдельных каскадов центрального устройства управления во время выполнения той или иной операции.

В трехадресных машинах параллельного действия в каждом такте выделяются пять основных управляющих импульсов, которые в соответствии с их назначением можно назвать следующим образом: импульс начала такта, импульс выдачи команды, импульс выдачи I-го числа, импульс выдачи II-го числа, импульс записи результата операции. Помимо этих основных управляющих импульсов, участвующих при выполнении большинства операций, в формирователе получаются и вспомогательные управляющие импульсы, которые используются в специальных режимах, характерных для каждой конкретной машины. Эти вспомогательные импульсы мы рассматривать не будем.

Каскад управления запуском формирователя служит для запуска и остановки формирователя; он прерывает в необходимых случаях (например, при команде «остановка») поступление от задающего генератора синхронизирующих импульсов на формирователь, прекращает дальнейшую выработку тактов машины и таким образом останавливает машину. Поступление синхронизирующих импульсов на формирователь прекращается также и при таких специальных операциях, как операции обмена информацией между внутренним и внешним запоминающими устройствами, при вводе исходного материала и выводе результатов решения из машины. В таких случаях синхронизация работы машины осуществляется медленнодействующими внешними устройствами, а каскады управления запуском формирователя обеспечивают выработку такта машины только после прихода внешнего синхронизирующего сигнала. Следует отметить, что, если сигнал остановки машины придет в середине такта (например, сигнал остановки будет подан с пульта ручного управления), то каскады управления запуском формирователя обеспечат выполнение текущего такта до конца и лишь затем прекратят прохождение запускающих импульсов на формирователь. Таким образом, каскады управления запуском в любом случае, если формирование такта уже началось, продолжают формирование полного такта машины.

Начало каждого такта определяется импульсом «начало такта» (*НТ*). Этот импульс, во-первых, увеличивает на единицу адрес предыдущей команды, зафиксированный на счетчике адресов команд, с которого в дальнейшем будет считываться адрес ячейки, содержащей очередную команду, и, во-вторых, приводит в нулевое положение все элементы блока фиксации команды, подготавливая этот блок к приему выданной из запоминающего устройства следующей команды.

Следует отметить, что число, зафиксированное в счетчике адресов команд, с каждым тактом увеличивается на единицу, что обеспечивает последовательную выборку всех команд программы.

Импульс «выдача команды» (*ВК*), следующий за импульсом «начало такта», открывает вентиль $B_4$ *) и пропускает двоичный код адреса команды на блок выбора ячейки. При этом соответствующая данному адресу команда извлекается из запоминающего устройства и поступает на узел входных вентилей (*УВВ*), которые в этот момент также открыты импульсом «выдача команды». Пройдя через узел входных вентилей, код команды записывается на регистр команд блока фиксации команды. Как нетрудно заметить, в блоке фиксации команды каждому адресу и коду операции соответствует свой отдельный регистр, фиксирующий соответствующую группу двоичных разрядов.

После прохождения импульса «выдача команды» из формирователя выдается импульс «выдача I-го числа» *(В. I)*. Этот импульс открывает вентили $B_1$, которые пропускают на блок выбора ячейки адрес первого числа. По этому адресу выбирается из запоминающего устройства первое число и через соответствующие кодовые шины число передается в арифметическое устройство. Таким образом, первое число, необходимое для выполнения операции, поступает в арифметическое устройство, где запоминается на соответствующем регистре (заметим кстати, что одним из вспомогательных сигналов, вырабатываемых формирователем такта, о которых мы упомянули выше, является сигнал, разрешающий прием числа на регистр арифметического устройства).

После поступления числа на регистр арифметического устройства формирователь выдает следующий основной сигнал такта: «выдача II-го числа» *(В. II)*. По этому сигналу открываются вентили $B_2$ и второй адрес команды, находящийся в блоке фиксации команды, поступает на блок выбора ячейки, который обеспечивает выборку кода числа из ячейки, заданной вторым адресом команды (заметим, что сигналы, разрешающие выдачу как первого, так и второго числа из запоминающего устройства, также относятся к числу вспомогательных сигналов, вырабатываемых формирователем такта). Второе число по кодовой шине числа поступает из запоминающего устройства в арифметическое устройство. Одновременно с блока фиксации команды (с разрядов кода операции) выдается код операции на блок дешифратора кода операции. Последний в зависимости от кода операции выдает на одном из своих выходных проводов сигнал, определяющий операцию и устройство машины, которое должно ее выполнять. Если эта операция относится к арифметическому устройству, то арифметическое устройство производит соответствующую сигналу операцию над выданными из запоминающего устройства числами. Результат произведенной операции выдается на кодовую шину числа, по которой поступает в запоминающее устройство.

Запись результата в запоминающее устройство происходит по импульсу записи, который является пятым основным сигналом такта и выдается из формирователя в конце такта. Этот импульс открывает вентили $B_3$, и код адреса ячейки, в которую необходимо записать результат операции, проходит на блок выбора ячейки. Одновременно импульс записи поступает в запоминающее устройство и обеспечивает запись результатов в соответствующую третьему адресу ячейку.

В конце каждого такта работы машины из арифметического устройства может выдаваться специальный сигнал, названный на рис. 56 сигналом перехода $\omega$. Этот сигнал, качественно характеризующий результат операции, определяет направление условного перехода при выполнении соответствующей команды программы. Он может выдаваться, например, при получении отрицательного результата в операции сложения, при несовпадении двух чисел в операции сравнения и т. д. Выданный из арифметического устройства признак направления условного перехода фиксируется на регистре признака условного перехода.

На этом такт работы машины заканчивается и начинается следующий такт. Если очередная команда не является командой условного или безусловного перехода, то работа центрального устройства управления в следующем такте аналогична работе в предыдущем такте. Заметим, что регистр признака условного перехода приводится в середине такта в нулевое положение одним из вспомогательных импульсов, получаемых в формирователе. Цепь прохождения этого импульса на рисунке не указана.

**3. Выполнение условных переходов.** Если очередная команда является командой условного или безусловного перехода, то блок дешифратора кода операции расшифровывает код операции условного перехода и вырабатывает сигнал команды условного перехода, который поступает на вентили $B_7$ и $B_8$. Вентили $B_7$ и $B_8$ управляют вентилями $B_5$ и $B_6$, на которые поступают соответственно коды первого и второго адреса команды, установленной на регистре команд. Если в предыдущем такте не было сигнала, соответствующего признаку условного перехода $\omega = 1$, то регистр признака условного перехода сохраняет нулевое положение. В этом случае открытым окажется вентиль $B_7$ и сигнал команды условного перехода проходит через $B_7$, и открывает вентиль $B_5$, обеспечивая этим запись на счетчик адресов команд кода первого адреса. При наличии в предыдущем такте сигнала признака направления условного перехода $\omega = 1$, фиксируемого на регистре признака условного перехода, открывается вентиль $B_8$. В этом случае сигнал команды условного перехода, пройдя через $B_8$, открывает вентиль $B_6$ и обеспечивает запись на счетчик адресов команд кода второго адреса.

В следующем такте очередная команда выдается уже не из следующей по порядку ячейки запоминающего устройства, а из той ячейки, которая соответствует адресу, записанному на счетчике адресов команд. Таким

*) На рис. 56 для удобства изображения показан один вентиль. В действительности под $B_4$ следует понимать группу вентилей с общим управляющим входом. Каждый из этих вентилей соответствует одному разряду передаваемого кода. Это замечание относится и к другим вентилям, через которые проходят коды адресов.

образом, в зависимости от результата предыдущей операции осуществляется переход к выполнению либо команды, адрес которой указан в первом адресе команды условного перехода, либо команды, адрес которой указан во втором адресе команды условного перехода.

В дальнейшем все следующие команды будут выполняться последовательно одна за другой, начиная с той команды, которая записана на счетчике адресов команд. Такой порядок выполнения команд будет сохраняться до тех пор, пока не встретится вновь команда условного или безусловного перехода или машина не будет остановлена.

## § 20. Арифметические устройства

*Арифметические устройства* предназначаются для выполнения арифметических и логических операций над числами и командами **в** машинах. В состав арифметических устройств входит обычно несколько отдельных устройств различного функционального назначения. Основными из них являются устройства для сложения и вычитания и множительные устройства. В настоящем параграфе мы кратко рассмотрим некоторые упрощенные функциональные схемы этих устройств. Основными элементами, из которых строятся эти устройства, являются одноразрядные сумматоры, триггеры, регистры, вентили, сдвигатели. Примеры принципиальных схем этих элементов были рассмотрены в предыдущей главе.

Рассмотрим сначала устройство сложения и вычитания. Основной частью этих устройств является *сумматор.* Сумматоры могут быть двух типов: *комбинационные* и *накапливающие.*

Комбинационный сумматор имеет два выхода, по которым одновременно подаются оба складываемых числа, представленные в виде высоких и низких уровней напряжения. Сумма появляется одновременно с вводом слагаемых (если пренебречь переходными процессами) и значение ее определяется тем состоянием электронной цифровой схемы, которое она принимает при одновременном приложении электрических сигналов, характеризующих оба числа. При снятии хотя бы одного слагаемого значение суммы исчезает.

Накапливающие сумматоры могут иметь либо два входа (по одному на каждое из слагаемых), либо один вход для обоих слагаемых. В том и другом случае складываемые числа подаются в сумматор не одновременно и значение суммы будет определяться результатом сложения всех введенных в разное время в сумматор слагаемых. Накапливающий сумматор сохраняет значение суммы и после исчезновения сигналов, характеризующих слагаемые. Стирание суммы производится специальным сигналом. Накапливающий сумматор может рассматриваться как соединение комбинационного сумматора и регистра — запоминающего устройства на одно число. Сумматор, построенный из триггеров, будет являться накапливающим и без специального запоминающего регистра, так как триггерные схемы обладают свойством сохранять свое положение. Вводимое в накапливающий сумматор число прибавляется к числу, которое до этого находилось в сумматоре, и полученная сумма замещает прежнее число.

По характеру выполнения операций над разрядами чисел сумматоры могут быть *последовательного* или *параллельного* действия. В сумматор последовательного действия разряды вводимого числа подаются последовательно один за другим, начиная с младшего разряда; при этом ввод $n$-разрядного числа требует в $n$ раз больше времени, чем ввод одного разряда. В параллельный сумматор все разряды какого-нибудь числа вводятся одновременно и время ввода при этом равно времени ввода одного разряда. Параллельный сумматор, обеспечивая более быстрый ввод чисел, в то же время требует больше входных каналов и аппаратуры, чем сумматор последовательного действия.

**1. Сумматоры параллельного действия.** На рис. 57 приведена блок-схема комбинационного сумматора параллельного действия с последовательным переносом единицы. Этот сумматор обеспечивает сложение двух $n$-разрядных двоичных чисел. Цифры соответствующих разрядов первого и второго слагаемых и суммы обозначены через $x_i$, $y_i$, $z_i$.

Для управления вводом слагаемых предусмотрены две группы вентилей. Каждая из групп вентилей имеет общий управляющий вход. Каждый одноразрядный сумматор имеет три входа и два выхода.

Для $i$-го одноразрядного сумматора $\upsilon_i$ обозначает перенос в старший разряд и $\upsilon_{i-1}$ — перенос в данный разряд из младшего разряда. Данная схема предусматривает возможность вычитания чисел, заданных в обратном коде (см. § 9), для чего в схеме имеется

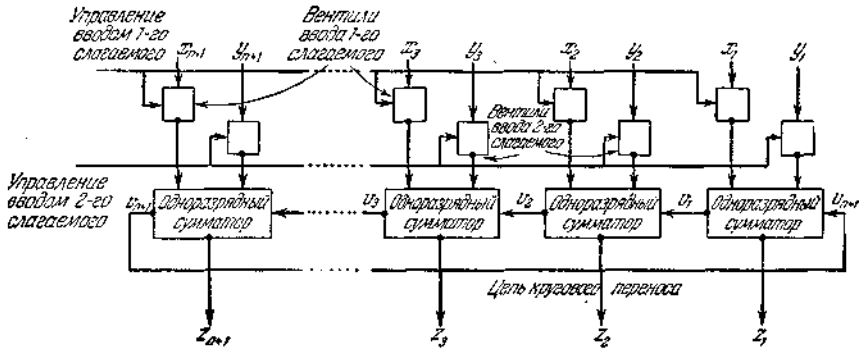



Рис. 57. Блок-схема комбинационного сумматора параллельного действия с последовательным переносом единицы.

линия циклического переноса из старшего разряда в младший и дополнительно должен быть введен ($n$ + 1)-й старший разряд. При сложении чисел с запятой, фиксированной перед старшим разрядом числа, ($n$ + 1)-й разряд будет представлять собой разряд целых единиц — знаковый разряд.

Перевод отрицательных чисел в обратный код должен производиться отдельным блоком арифметического устройства.

Для хранения отрицательных чисел в машине имеются два способа, зависящих от конструкции машины.

Отрицательные числа можно хранить в запоминающем устройстве в обратном коде, и тогда их можно непосредственно подавать в сумматор. При этом необходимость в специальном преобразовании чисел в обратный код возникает только при вычитании положительных чисел. Практически этот способ хранения отрицательных чисел редко применяется, так как при этом необходимо запоминать, в каком коде хранится число, и кроме того, при умножении и делении приходится переводить числа в прямой код.

Отрицательные числа можно также хранить в запоминающем устройстве в прямом коде с отрицательным знаком. При этом перед подачей в сумматор все разряды отрицательного числа должны пройти через преобразователь, который заменит 1 на 0 и 0 на 1.

Время сложения двух чисел в параллельном сумматоре комбинационного типа при последовательном способе переноса единицы определяется в основном временем переноса между разрядами. Наиболее длинный перенос будет в случае, когда

$$x_{n+1} + y_{n+1} = 1 + 1 = 10,$$

а во всех остальных разрядах

$$x_i + y_i = 1.$$

Тогда единица переноса, попадающая через линию циклического переноса в младший разряд, должна пройти $n$ + 2 разряда.

На рис. 58 показана блок-схема параллельного накапливающего сумматора с одновременным переносом. Схема предназначена для сложения $n$-разрядных чисел с фиксированным положением запятой, заданных в системе счисления с основанием $B$, не обязательно равным двум.

Сложение отрицательных чисел предусматривается в обратном коде: при помощи преобразователя, подобного рассмотренному в § 15 предыдущей главы, каждый разряд отрицательного числа перед подачей в сумматор должен быть преобразован в дополнение до $B$—1. В знаковом (старшем) разряде для отрицательных чисел при этом указывается цифра $B$—1 (для положительных чисел в знаковом разряде ставится нуль).

Для выполнения действий над числами в обратном коде в схеме сумматора предусмотрен дополнительный ($n$ + 1)-й разряд, являющийся знаковым разрядом, для этой же цели предусмотрена и линия циклического переноса из старшего разряда (знакового) в младший. Каждый одноразрядный сумматор в данном случае является счетчиком, т. е. он обеспечивает не только суммирование цифр соответствующих разрядов, но и сохранение полученной цифры данного разряда суммы.

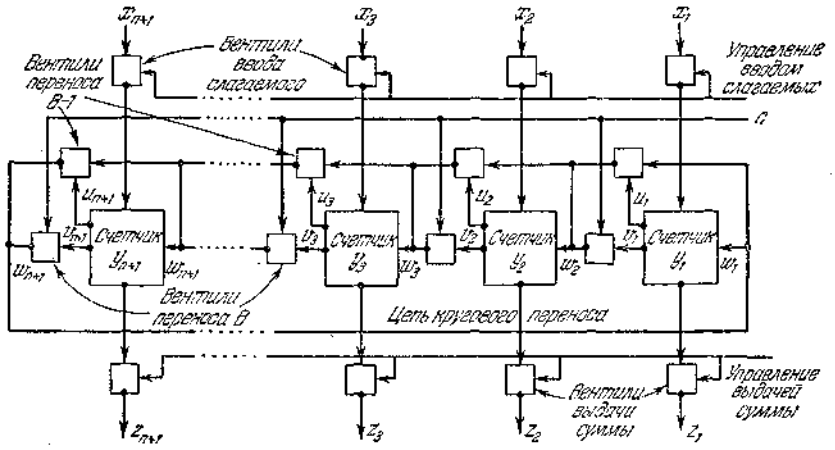



Рис.58. Блок-схема параллельного накапливающего сумматора с одновременным переносом.

Одноразрядные счетчики для системы счисления с основанием $B$ могут быть построены в виде цепочек из $B$ триггеров. Такие счетчики должны обладать способностью принимать $B$ устойчивых состояний. Для ввода слагаемых в рассматриваемой схеме предусматривается один вход на каждый из разрядов $x_1, x_2, \ldots, x_{n+1}$, по которому слагаемые вводятся в сумматор поочередно. (Может вводиться и больше двух слагаемых.)

Управление вводом слагаемых осуществляется при помощи группы вентилей с общим управляющим входом, а выводом суммы — общей цепью управления для вентилей выдачи суммы $z_1, z_2, \ldots, z_{n+1}$. Каждый одноразрядный счетчик имеет два входа $x_i$ $\omega_i$ и три выхода $u_i$, $v_i$, $z_i$.

Если некоторый $i$-й счетчик находится в состоянии, соответствующем цифре $x_i$, и в него поступает цифра $y_i$ то счетчик перейдет в состояние $p_i = x_i + y_i - \upsilon_i B$, где $\upsilon_i$ — сигнал, соответствующий единице переноса в старший разряд и появляющийся тогда, когда $x_i + y_i \geq B$. В формуле $\upsilon_i$ равно единице при наличии единицы переноса и равно нулю при ее отсутствии.

В отличие от рассмотренного выше комбинационного параллельного сумматора, в котором имел место последовательный перенос между разрядами, в данной схеме предусматривается одновременный перенос между разрядами. При этом процесс поразрядного сложения и процесс переносов разделены между собой во времени. Сначала выполняется поразрядное сложение, а затем подается специальный сигнал с, поступающий на вентили переноса $B$, по которому осуществляются все переносы. Одновременное выполнение всех переносов между разрядами обеспечивается введением в схему, кроме вентилей переноса $B$, открываемых сигналами $\upsilon_i = 1$, еще вентилей прямого циклического переноса, открываемых сигналами $u_i$ появляющимися, когда $x_i + y_i = B\text{-}1$. Сигнал $u_i = 1$, обеспечивает переход импульса переноса $\omega_i$ возникающего в младшем $(i\text{-}1)$-м разряде, сразу в старший $(i + 1)$-й разряд. Если в. $i$-м разряде появилась величина $\upsilon_i = 1$, то вентиль $i$-го переноса будет открыт для сигнала $c$. Сигнал $c$ подается после окончания процессов поразрядного сложения в счетчиках и установления всех сигналов $\upsilon_i$ и $u_i$ открывающих вентили переносов $B$ и $B$-1.

Ясно, что в каком-либо разряде не могут одновременно возникнуть оба импульса переноса $\upsilon_i = 1$ и $u_i = 1$, т.е. всегда $u_i \cdot \upsilon_i = 0$.

По сигналу $c$ происходят переносы и схема принимает окончательное состояние, при котором значение цифры $i$-го разряда суммы определяется выражением

$$z_i = x_i + y_i + \omega_i - (\upsilon_i + u_i\omega_i)\, B.$$

Так как либо обе величины $\upsilon_i$ и $u_i$ равны нулю, либо одна из них равна нулю, а вторая — единице, то значение цифры данного разряда суммы будет:

$$z_i = x_i + y_i + \omega_i \quad \text{при} \quad x_i + y_i + \omega_i < B;$$
$$z_i = x_i + y_i + \omega_i - B \quad \text{при} \quad x_i + y_i + \omega_i \geq B.$$

В такой схеме время, расходуемое для переносов, зависит только от времени прохождения импульсов переноса через вентили, управляемые сигналами $u_i$ и не зависит от времени перехода счетчика из состояния $B$-1 в нулевое состояние.

Следует заметить, что возникшие импульсы переносов $u_i$, и $\upsilon_i$ должны задерживаться линиями задержки или фиксироваться при помощи триггеров, так как наличие переходных процессов во время осуществления переносов может нарушить правильность работы схемы. После окончания каждой операции суммирования эти триггеры должны устанавливаться в нулевое положение, для чего необходимы дополнительные цепи управления.

В данном накапливающем сумматоре отрицательные числа всегда получаются и выдаются в обратном коде (т. е. в виде поразрядных дополнений до $B$-1).

Выдача суммы $z_1, z_2, \ldots, z_{n+1}$ осуществляется при помощи специальных вентилей и может производиться в любое время, кроме времени выполнения операций, так как в последнем случае выдаваемая сумма не будет учитывать поступающего слагаемого. Перед выполнением каждой новой группы операций суммирования чисел все счетчики при помощи специальных схем возвращаются в исходное состояние.

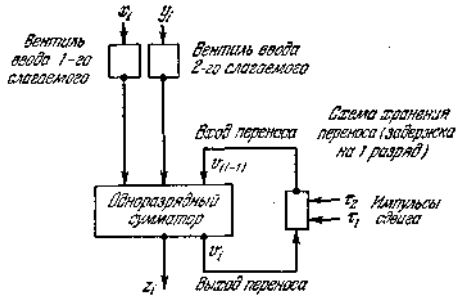



Рис. 59. Блок-схема комбинационного сумматора последовательного действия.

**2. Сумматоры последовательного действия.** Рассмотрим примеры сумматоров последовательного действия. На рис. 59 приведена блок-схема комбинационного сумматора последовательного действия. В этом сумматоре используется один одноразрядный сумматор и схема для задержки единицы переноса на время, равное интервалу между двумя последовательно поступающими разрядами чисел. В случае двоичной системы счисления схема работает по правилу

$$z_i = x_i + y_i + \upsilon_{i\text{-}1} - 2\upsilon_i$$

где $z_i$ — значение цифры $i$ - го разряда суммы; $x_i$ и $y_i$ — значение цифр $i$-х разрядов слагаемых, $\upsilon_{i\text{-}1}$ — единица переноса из младшего разряда, $\upsilon_i$ — единица переноса в старший разряд.

Через $\tau_1$ и $\tau_2$ на рис. 59 обозначены тактовые импульсы, осуществляющие необходимый сдвиг в триггерах, образующих схему задержки, для обеспечения одновременности поступления импульсов переноса $\upsilon_{i\text{-}1}$ и входных сигналов $x_i$ и $y_i$.

Так как в таком сумматоре последовательного действия невозможно осуществить циклический перенос, то он используется либо для сложения положительных чисел, либо для сложения отрицательных чисел, заданных в дополнительном коде; при этом циклическая передача не нужна и импульс $\upsilon_{\text{-}1}$ будет равняться нулю, а импульс $\upsilon_n$ будет отбрасываться.

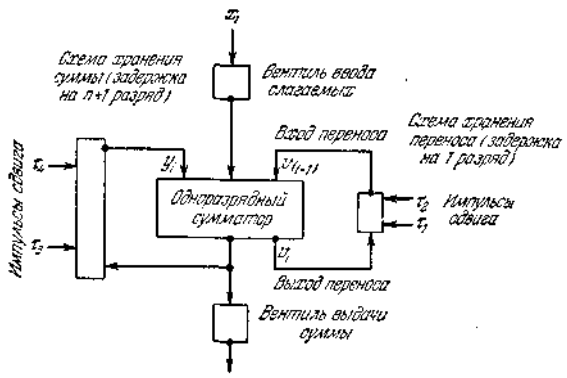



Рис. 60. Блок-схема комбинационного сумматора последовательного действия с циклическим

Сложение отрицательных чисел в обратном коде требует наличия циклической передачи единицы переноса из старшего разряда в младший, что может быть сделано только путем добавления к данному сумматору специального запоминающего устройства на одно число. Это устройство должно сохранять полученное при первом суммировании значение суммы и вновь подавать его в сумматор для повторного сложения с единицей циклического переноса. Схема такого сумматора приведена на рис. 60. Наличие схемы для хранения суммы (схемы задержки па ($n$ + 1)-й разряд) превращает, по существу, сумматор комбинационного типа в накапливающий сумматор. Этот сумматор может быть использован для сложения отрицательных чисел как в обратном, так и в дополнительном коде в зависимости от того, каким образом учитываются импульсы переносов $\upsilon_{-1}$ и $\upsilon_n$.

Если сложение ведется в обратном коде, то циклическая передача обеспечивается путем сохранения при помощи двойной триггерной схемы импульса переноса $\upsilon_n$ и подачи его на вход сумматора уже в качестве импульса $\upsilon_{-1}$ для повторного суммирования. При этом выдача окончательного значения суммы возможна только на втором этапе суммирования, т е. после прибавления единицы циклического переноса. Выдача суммы возможна также и в любой последующий момент времени, так как эта сумма будет храниться в накапливающей схеме сумматора до стирания ее специальным сигналом.

Выдача суммы осуществляется путем сложения этой суммы с нулем. В случае, если используется сложение в дополнительном коде, то импульс $\upsilon_n$ отбрасывается и импульс $\upsilon_{-1}$ приравнивается нулю. Линия задержки на ($n$ + 1)-й разряд используется в этом случае просто для хранения последовательно накапливаемых сумм. Сумма может выдаваться в любое время, включая и время ввода очередного слагаемого.

Для преобразования отрицательных чисел в обратный или дополнительный код перед подачей их в сумматор должны применяться специальные устройства.

**3. Устройство для сложения и вычитания чисел с плавающей запятой.** Мы рассмотрели примеры блок-схем сумматоров, предназначенных для сложения и вычитания чисел с фиксированным положением запятой. Рассмотрим теперь порядок работы устройства для сложения и вычитания чисел, заданных в нормальной форме с плавающей запятой.

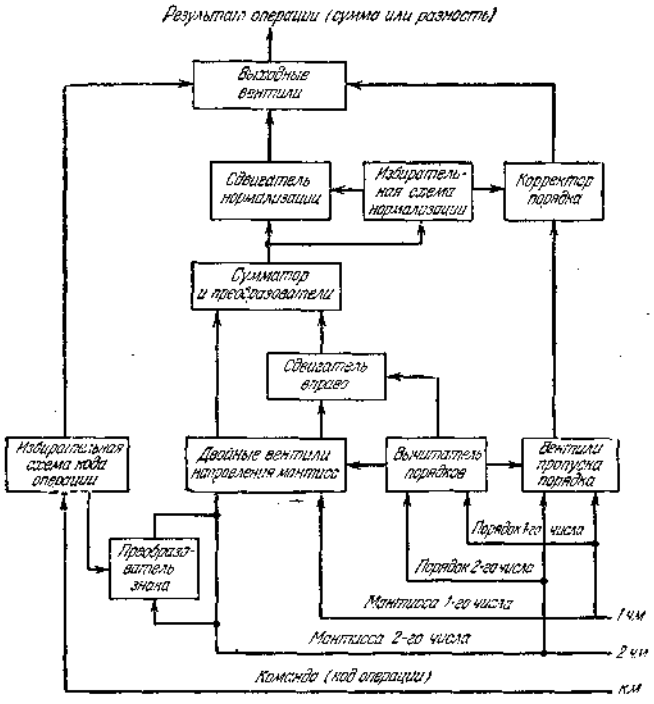



Рис. 61. Блок-схема устройства параллельного действия для сложения и вычитания чисел в машине с плавающей запятой.

На рис. 61 показана упрощенная блок-схема такого устройства параллельного действия. В качестве составных частей устройства используются сумматоры с фиксированной запятой для сложения мантисс и вычитания порядков, а также вентили, преобразователи, сдвигатели и избирательные схемы. Работа устройства происходит в соответствии с правилами действий над числами в нормальной форме (см. § 6). В устройство из двух числовых магистралей 1ЧМ и 2ЧМ поступают два исходных числа, а из командной магистрали КМ поступает код операции (сложить или вычесть).

Рассмотрим выполнение операций сложения. Порядки чисел поступают в схему вычитателя порядков, в которой из порядка первого *) числа вычитается порядок второго числа. Если результат получится положительный, то это значит, что порядок первого числа больше порядка второго числа. В этом случае вычитателем порядков выдается сигнал, переключающий схему двойных вентилей таким образом, что она направляет мантиссу первого числа в сумматор мантисс, а мантиссу второго числа — в сдвигатель вправо. В сдвигатель вправо из вычислителя порядков поступает разность порядков, показывающая, на сколько разрядов вправо должна быть сдвинута мантисса второго числа. Мантисса второго числа после

*) Первым числом мы будем называть число, -поступившее по первой магистрали, и вторым числом — число, поступившее по второй магистрали.

сдвига поступает в сумматор мантисс. Вычитатель порядков вы дает также сигнал в схему вентилей пропуска порядка, обеспечивающий пропуск в корректор порядка большего порядка, в данном случае порядка первого числа. Таким образом выполняется первый этап сложения — выравнивание порядков складываемых чисел.

В сумматоре мантисс производится сложение мантисс по правилам сложения чисел с фиксированной запятой. В сумматор мантисс включены также преобразователи, управляемые знаковыми разрядами, которые производят преобразование мантисс отрицательных слагаемых в обратный или дополнительный код и обратное преобразование мантиссы суммы, полученной в обратном или дополнительном коде, в отрицательное число, представленное в прямом коде.

Следует заметить, что вычитатель порядков содержит также схемы для преобразования отрицательных порядков в обратный или дополнительный код.

Из сумматора выходит мантисса суммы, которая поступает в сдвигатель нормализации и на избирательную схему нормализации. В случае, если мантисса суммы является ненормализованной, то избирательная схема выдает в сдвигатель нормализации сигналы, обеспечивающие необходимый для нормализации сдвиг мантиссы, и в корректор порядка — сигналы, обеспечивающие соответствующее изменение порядка результата. Нормализованная мантисса и скорректированный порядок поступают на выходные вентили для выдачи их из устройства в качестве результата операции.

Выполнение операции вычитания отличается от выполнения операции сложения тем, что избирательная схема кода операции при вычитании изменяет знак второго числа на обратный. В остальном устройство работает точно так же, как и при выполнении операции сложения чисел. Заметим, что при вычитании чисел вычитаемое должно подаваться в устройство по второй магистрали.

Избирательная схема кода операции управляет также выходными вентилями. Исходные числа поступают в устройство из числовых магистралей постоянно, в том числе и во время выполнения машиной других операций, не относящихся к данному устройству. Но при этом выходные вентили остаются закрытыми и из данного устройства в другие устройства машины ничего не выдается.

Избирательная схема кода операции открывает выходные вентили только во время выполнения операций, относящихся к данному устройству, и результаты только таких операций (сложение и вычитание) выдаются из данного устройства в машину. Обычно с помощью одного подобного устройства в машине выполняются и некоторые другие операции, например сложение и вычитание модулей чисел, операции над командами и др. Естественно, что расширение функций устройства требует введения дополнительных схем.

Из приведенного описания блок-схем различных устройств сложения и вычитания видно, что устройства, предназначенные для операций с числами в нормальной форме, являются значительно более сложными, чем устройства для операций над числами с фиксированным положением запятой.

**4. Множительные устройства.** Перейдем к рассмотрению множительных устройств. На рис. 62 дана упрощенная блок-схема устройства для умножения двух *n*-разрядных двоичных чисел с фиксированным положением запятой. Множительное устройство имеет два *n*-разрядных регистра (регистр множителя и регистр множимого), схему для сдвига множимого, вентиль и накапливающий 2n-разрядный сумматор произведения. С регистром множителя соединена схема для управления сдвигателем и вентилем.

Сдвигатель последовательно производит сдвиг множимого на один разряд влево при каждом сдвиге. Вначале сдвигатель установлен таким образом, что все разряды множимого при открытом вентиле будут проходить в крайние правые *n* разрядов накапливающего сумматора. Устройство управления в зависимости от значения цифры младшего разряда множителя (0 или 1) выдает сигнал, управляющий вентилем. Если цифра равна 1, то вентиль открывается и пропускает множимое в сумматор; если цифра равна 0, то вентиль остается закрытым и множимое в сумматор не поступает. Затем устройство управления устанавливает сдвигатель для сдвига множимого на один разряд влево, а для управления вентилем вы дается следующая цифра множителя.

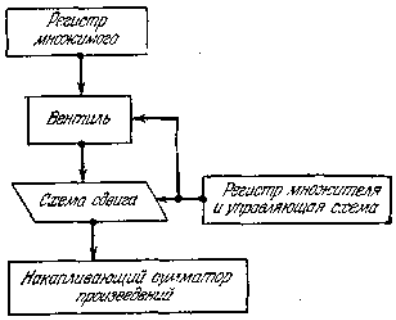



Рис. 62. Блок-схема устройства для умножения двоичных чисел в машине с фиксированной запятой

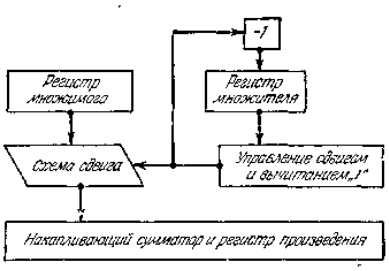



Рис. 63. Блок-схема устройства для умножения десятичных чисел в. машине. с фиксированной запятой

Если эта цифра равна 1, то множимое пропускается в сумматор уже со сдвигом на один разряд влево и суммируется с содержимым сумматора. Если очередная цифра равна нулю, то множимое в сумматор не

проходит. После этого снова производится сдвиг влево и для управления вентилем выдается следующая цифра множителя, и т. д.

Процесс продолжается до тех пор, пока не будут исчерпаны все цифры множителя. В результате в накапливающем сумматоре будет получено значение произведения, имеющее $2n$ разрядов. Это — наиболее упрощенный вариант двоичного множительного устройства. Можно предусмотреть округление частных произведений, что позволит сократить число разрядов в сумматоре, или, с другой стороны, использовать $2n$-разрядный регистр сумматора также и для хранения множителя.

Из рассмотрения процесса умножения видно, что в регистре сумматора сначала заняты только $n$ правых разрядов, а левые разряды заполняются постепенно. При этом, расположив множитель в $n$ левых разрядах регистра сумматора, можно последовательно сдвигать его влево таким образом, чтобы очередная управляющая цифра

множителя находилась все время в крайнем левом разряде регистра сумматора. В этом случае не требуется иметь специальный регистр для хранения множителя.

На рис. 63 показана блок-схема устройства для умножения двух $n$-разрядных чисел в десятичной системе счисления. Эта схема иллюстрирует метод построения десятичных множительных устройств путем многократного сложения. Устройство состоит из двух регистров (множимого и множителя), схемы сдвига, схемы, управляющей сдвигом и вычитанием единицы, и накапливающего сумматора с регистром произведения.

Схема сдвига, связывающая регистр множимого с накапливающим сумматором, служит для умножения множимого на целые положительные или целые отрицательные степени десяти ($10^m$, где $m = 0, 1,2,..., n$-1 или $m = 0,-1, -2 ,..., -(n-1)$. Схема управления управляет схемой сдвига пропускающей множимое в сумматор, и осуществляет при каждом прохождении множимого в сумматор вычитание единицы из той цифры множителя, на которую в данный момент производится умножение.

Работа схемы управления зависит от последовательных цифр множителя, начиная с цифры младшего разряда. Когда очередная цифра множителя в результате повторных вычитаний единицы сделается равной нулю, схема управления переходит к следующей цифре множителя и одновременно выдает в схему сдвига сигнал, устанавливающий эту схему в положение, обеспечивающее сдвиг множимого еще на один разряд влево. Как и в случае двоичного множительного устройства, работа начинается при установке схемы сдвига в положение, соответствующее нулевому сдвигу.

Управляющая схема регистра множителя вначале установлена на управление от младшей цифры множителя. Если эта цифра отлична от нуля, то управляющая схема пропускает множимое в крайние правые разряды накапливающего сумматора и одновременно вычитает единицу из данной цифры множителя. Множимое столько раз пропускается в сумматор и суммируется там с содержимым сумматора, сколько единиц обозначает данная цифра множителя. При получении нуля в данном разряде множителя управляющая схема переходит к следующей цифре множителя и меняет установку сдвигателя на один разряд влево. Этот процесс продолжается до тех пор, пока не будут пройдены все цифры множителя и число, находящееся в регистре множителя, не станет равным нулю. После этого в регистре произведения будет получено значение произведения.

Устройство для умножения чисел в нормальной форме должно иметь, помимо рассмотренных частей, сумматор порядков и дополнительные схемы для нормализации результата.

Рассмотренные примеры блок-схем устройств сложения и вычитания и устройств умножения являются весьма упрощенными.

Действительные варианты схем, применяемые в машинах, значительно сложнее. Однако общие принципы построения и порядок работы этих устройств в своей основе остаются теми же самыми, что и в рассмотренных нами примерах.

## § 21. Запоминающие устройства

Существует чрезвычайно большое количество различных типов запоминающих устройств, отличающихся друг от друга по своим принципам действия, емкости запоминания, быстродействию, длительности сохранения информации и другим характеристикам.

В настоящем параграфе будут кратко рассмотрены основные типы устройств, получивших практическое применение в вычислительных машинах, а также некоторые типы перспективных запоминающих устройств.

**1. Перфоленты и перфокарты.** Запоминающие устройства на перфолентах и перфокартах (см. § 6) служат для ввода исходной информации в машину и для вывода результатов решения задач из машины.

Перфорация данных, вводимых в машину, выполняется вручную оператором при помощи специального электромеханического прибора — перфоратора; при выводе данных из машины перфорация осуществляется автоматически. Считывание данных с перфокарт или перфолент может производиться двумя способами: механическим и фотоэлектрическим. При механическом считывании карта или лента ощупывается специальными щеточками и в зависимости от наличия или отсутствия отверстия на данном месте карты или ленты замыкается или не замыкается электрический контакт, посылающий соответствующий импульс тока.

При фотоэлектрическом считывании непрозрачная лента продвигается над щелью в диафрагме. Размеры щели таковы, что над ней может находиться только одно отверстие из каждой колонки. Лента снаружи освещается

постоянным источником света. В зависимости от наличия отверстия в том месте ленты, которое находится над щелью диафрагмы, на фотоэлементы попадает или не попадает свет. Каждый раз при прохождении над щелью отверстия в данной колонке соответствующий фотоэлемент дает электрический импульс.

Запись на перфолентах не стирается, и поэтому ленты и карты могут использоваться только как средства постоянной памяти. Перфоленты употребляются в виде рулонов или в виде бесконечных петель (замкнутых лент), обеспечивающих повторный ввод данных. Карты используются в виде массивов или колод. Достоинством карт по сравнению с лентами является возможность удобной замены карт при порче или при изменении вариантов расчетов, а также возможность легкой перестановки карт, т. е. изменение порядка вводимого материала.

Недостатком перфолент и перфокарт является малая скорость ввода и вывода данных, ограничиваемая скоростью механического перемещения. Особенно это относится к устройствам с механическим ощупыванием, в которых наибольшая скорость считывания может достигнуть не более 100 цифр в секунду по каждой колонке. При фотоэлектрическом считывании скорость считывания может быть значительно повышена (до нескольких тысяч чисел в секунду).

На рис. 64 показана фотография стандартной перфокарты.

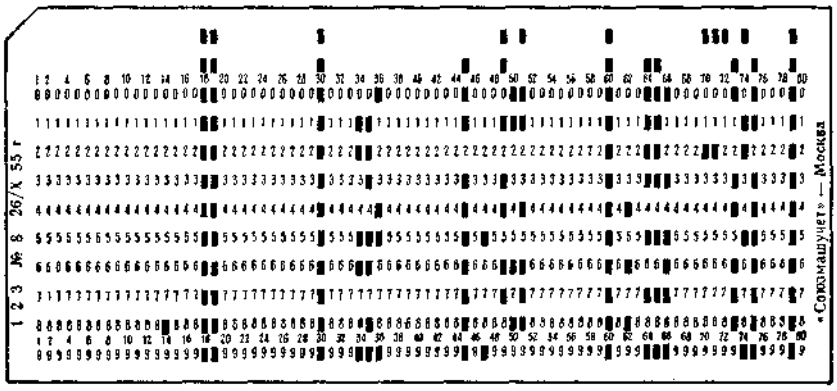



Рис. 64. Стандартная перфокарта.

**2. Электромеханические и электронные реле.** В ранних вычислительных машинах (*Марк* I, *Марк* II и др.) для построения запоминающих устройств широко использовались электромеханические реле. Недостатком таких устройств является очень ограниченная емкость и малая скорость работы. Так, в машине *Марк* II емкость запоминающего устройства составляла всего 100 чисел. Надежное срабатывание электромеханического реле требует не меньше 6 *мсек*. Для запоминания одного разряда двоичного числа (0 или 1) требуется одно реле, и кроме того, дополнительные элементы для построения управляющих схем запоминающего устройства. Помимо малой скорости работы, к недостаткам электромеханических реле относится также невысокая надежность этих элементов. Частицы пыли, грязи, окисление контактов могут часто выводить реле из строя.

Применение электронных ламп позволило построить быстродействующие электронные реле — триггеры, скорость работы которых в тысячи раз выше, чем у электромеханических реле. Принцип действия триггерной ячейки был рассмотрен в § 14. Каждая триггерная ячейка может «запомнить» только один двоичный разряд (0 или 1) и для запоминания большого количества данных требуется большое количество ламп. Помимо ламп, образующих триггерные ячейки, требуются также лампы для построения управляющих схем запоминающего устройства, обеспечивающих выборку и запись нужных данных. В общем случае количество ламп, затрачиваемых на управление, оказывается не меньшим, чем количество ламп, входящих непосредственно в запоминающие триггерные ячейки. Поэтому, несмотря на большое быстродействие, запоминающие устройства большой емкости на электронных триггерных ячейках не строятся. В машине ЭНИАК, например, имелось внутреннее запоминающее устройство на триггерах емкостью всего в 20 чисел.

Применение электронных триггеров удобно для построения регистров (запоминающих устройств на одно число), используемых в арифметических и управляющих устройствах.

**3. Линии задержки.** Принцип действия запоминающих устройств, применяющих линии задержки, заключается в том, что информация, подведенная к одному концу линии, распространяется в виде волн в среде, образующей линию, с определенной конечной скоростью и через некоторое время появляется на другом конце линии.

С конца линии та же информация может быть снова подведена к началу, и таким образом создается замкнутая цепь циркуляции информации. Так как при распространении волн вдоль линии неизбежны искажения и затухания, то перед каждой повторной подачей информации в линию сигналы проходят через усилители и формирователи; таким образом обеспечивается практически сколь угодно долгое сохранение данных.

Для того чтобы не слишком увеличивать геометрические размеры линий задержки в качестве проводящей среды применяются такие вещества, в которых колебания распространяются достаточно медленно, например ртуть. Возможно применение акустических и электромагнитных линий задержки.

Наибольшее распространение в электронных цифровых машинах получили акустические ртутные линии. На рис. 65 показана блок-схема запоминающего устройства на ртутной линии.

Ртутная линия задержки представляет собой металлическую трубку, наполненную ртутью и закрытую с концов кристаллами кварца. Как известно, кварц обладает пьезоэлектрическими свойствами. Под действием приложенной разности потенциалов в кристалле кварца возбуждаются механические колебания и, наоборот, под воздействием механических колебаний (сжатие) образуются электрические сигналы. Обычно трубки имеют диаметр 1—2 *см* и длину 50—100 *см*. Количество импульсов, которое может сохраняться в трубке, равно произведению времени прохождения ультразвуковых волн вдоль трубки на частоту повторения импульсов. Практически это число приблизительно равно 1000.

С целью обеспечения хорошего сохранения формы импульсов используется несущая частота, которая модулируется импульсами звуковой частоты. Несущая частота должна выбираться такой, чтобы она соответствовала резонансной частоте пьезокристаллов, кварца, которая лежит в пределах от 5 до 30 *мгц*. Демодуляция производится на выходном конце трубки. Длительность импульса составляет приблизительно 1 *мксек*.

Ртуть в качестве среды, передающей колебания, выбрана также исходя из соображений наилучшей передачи колебаний от пьезокристаллов к передающей среде и обратно.

При переходе ультразвуковых колебаний из одной среды в другую на поверхности раздела сред часть энергии отражается обратно. При этом, так же как и в оптике, отношение прошедшей энергии и отраженной энергии зависит от коэффициента поглощения ультразвуковых колебаний в обеих средах, который в свою очередь

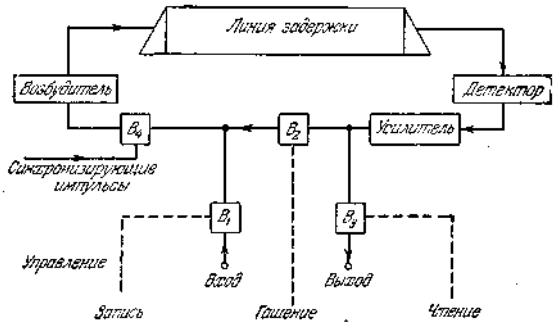



Рис. 65. Запоминающее устройство на ртутной линии задержки.

зависит от упругих свойств среды. Упругие свойства ртути близко подходят к свойствам кварца, и поэтому условия согласования ртутной линии с кристаллами кварца выполняются достаточно хорошо. Это значит, что энергия упругих колебаний кварца на входном конце почти полностью передается в ртуть и, наоборот, энергия звуковых колебаний ртути на выходном конце линии почти полностью поглощается кварцем и превращается им в электрическую энергию.

Конструкция крепления кристаллов предотвращает отражение звуковых воли в ртути. Электронные переключатели (вентили) служат для прекращения циркуляции импульсов (стирания информации) — $B_2$, для записи информации — $B_1$ и для считывания информации — $B_3$.

Ртутная линия задержки для обеспечения синхронной работы с остальными устройствами машины требует весьма точного регулирования температуры, так как скорость распространения звуковых волн в ртути зависит от температуры. Иногда для обеспечения синхронизма вместо поддержания постоянной температуры регулируют частоту основного генератора импульсов в соответствии с изменением температуры.

Например, в машине *УНИВАК* используется запоминающее устройство из 18 ртутных трубок, каждая из которых может сохранять 32 х 36 двоичных знаков. Регулирование температуры ртути осуществляется двумя способами: грубое регулирование производится в зависимости от сжатия или расширения ртути при изменении температуры, а точное регулирование осуществляется при помощи одной из трубок, по которой циркулируют стандартные импульсы, сравниваемые по частоте и по фазе с импульсами основного генератора. Чем длиннее трубки, тем более точное регулирование температуры требуется для этих трубок, поэтому в машинах употребляются и короткие трубки малой емкости.

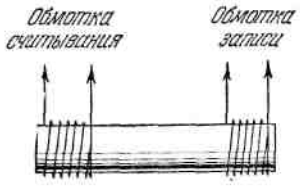



Рис. 66. Магнитострикционная линия задержки.

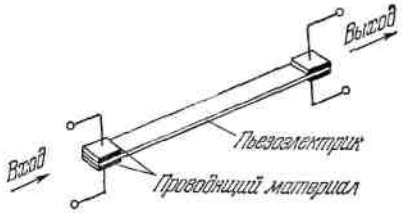



Рис. 67. Линия задержки из пьезоэлектрика.

Основными достоинствами ртутных линий задержки являются однородность, легкость соединения с преобразователями, отсутствие поперечных волн.

Помимо жидкостных ртутных линий задержки, применяются твердые линии задержки, к числу которых относятся магнитострикционные линии задержки (рис. 66) и линии задержки из пьезоэлектриков (рис. 67). Материалом для магнитострикционных линий задержки может служить никелевая проволока или феррит. Твердые линии задержки обеспечивают малые габариты устройств и высокую надежность работы.

Электромагнитные линии задержки могут быть трех основных типов: волноводные линии, электрические длинные линии с распределенными параметрами и искусственные длинные линии с сосредоточенными параметрами. Электромагнитные линии задержки могут быть эффективно использованы для запоминания данных в электронных счетных машинах, если частота повторения импульсов значительно больше, чем $10^6$ в 1 сек.

При частотах повторения импульсов меньше, чем 1 *мгц,* эти линии могут быть применены для сохранения всего нескольких импульсов. Опыт применения электромагнитных линий задержки в цифровых машинах еще мал, по использование этих линий для других целей показывает их высокую надежность.

Электромагнитные линии задержки обладают весьма большим затуханием, но у них практически отсутствует потеря энергии на входе и выходе линии.

Акустические линии, наоборот, имеют весьма малое затухание, однако в этих линиях большая часть энергии тратится при преобразованиях на концах линии.

Основным недостатком запоминающих устройств на линиях задержки является периодическая система выборки информации из устройства. Сохраняемые в устройстве данные могут быть получены лишь в моменты прохождения соответствующих сигналов через усилитель. Таким образом, при считывании необходимо ждать, пока информация дойдет до конца трубки. Время поиска данных в такой системе может быть равно полному времени прохождения волн вдоль линии. Например, для ультразвуковых волн в ртутной линии длиной около 1 *м* это время будет порядка 1 *мсек.*

Запоминающие устройства на линиях задержки особенно удобно применять в машинах последовательного действия, так как последовательная (поразрядная) выдача данных из трубки хорошо согласуется с последовательным способом передачи данных по магистралям машины и с последовательным способом работы арифметических устройств. Однако запоминающие устройства с линиями задержки широко применяются и в машинах параллельного действия. Основным достоинством этих устройств является их высокая надежность.

**4. Магнитные барабаны и ленты.** Использование магнитной записи позволяет удобно создавать запоминающие устройства очень большой емкости. Техника магнитной записи хорошо разработана для целей звукозаписи, и накопленный в этой области опыт был использован при создании запоминающих устройств вычислительных машин.

Процесс магнитной записи кратко сводится к следующему. На поверхности магнитного материала, движущегося с определенной скоростью, под действием специальных электромагнитов — записывающих головок — создаются отдельные намагниченные участки — магнитные диполи. Магнитный материал обладает свойством остаточного магнетизма, поэтому магнитные диполи сохраняются и после прекращения действия записывающих головок.

В записывающие головки подаются импульсы электрического тока, соответствующие записываемой информации и определяющие вид записанных магнитных диполей. Диполи располагаются на поверхности барабанов или лент один за другим в направлении движения, образуя магнитные дорожки. Число дорожек равно числу записывающих головок.

При чтении магнитный материал перемещается относительно считывающих головок, которые по своей конструкции аналогичны записывающим головкам (часто для записи и чтения используют одни и те же головки). Часть магнитного потока с того участка магнитного материала, который находится в непосредственной близости от зазора, замыкается через сердечник головки. Изменение во времени этого потока при движении материала наводит э. д. с. в обмотке считывающей головки. Индуцированная э. д. с. представляет собой сигнал, характеризующий прочитанную цифру.

На рис. 68 показана схема магнитного барабана.

Барабан с поверхностью, покрытой слоем из магнитного материала, вращается с большой скоростью (порядка 6000—7500 *об/мин).* Записывающие и считывающие головки расположены по образующим барабана. На поверхности барабана образуются магнитные диполи. Запись данных производится импульсами электрического тока длительностью в несколько микросекунд. Значению 1 соответствует положительный импульс, значению

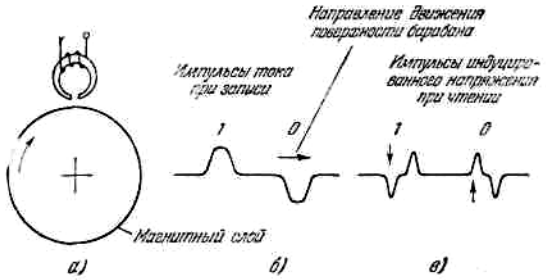



Рис. 68. Схема магнитного барабана и принцип его действия.

0 соответствует отрицательный импульс (рис. 68, *б).*

При считывании данных величина э. д. с. зависит от скорости изменения магнитного потока в сердечнике считывающей головки. Получающееся изменение напряжения на выходе считывающей головки (рис. 68, *в*) после соответствующего усиления преобразуется в последовательность импульсов стандартной формы. Эти импульсы затем используются в арифметическом или других устройствах машины.

Формирование таких стандартных импульсов на основе сигналов, получаемых от считывающих головок, может производиться несколькими методами. Так называемый *метод высечения* состоит в том, что при помощи специальных схем совпадения в заданные моменты времени определяется полярность напряжения, индуцированного проходящим магнитным диполем в считывающей головке. Это напряжение используется для управления соответствующими формирующими схемами, которые выдают стандартный импульс, соответствующий полярности считанного сигнала.

Синхронизация моментов подачи напряжений, отпирающих схемы для определения полярности диполей и моментов прохождения диполей под считывающей головкой, осуществляется при помощи синхронизирующих импульсов, получаемых от *диполей времени,* постоянно нанесенных на барабан вдоль одной из дорожек.

Кроме метода высечения, для считывания данных могут применяться методы, основанные на дифференцировании или интегрировании считываемых импульсов.

Существуют две системы записи данных: запись с размагничиванием и запись с перемагничиванием. В первом случае можно иметь три различных состояния магнитного материала (нулевое, положительное и отрицательное), но достижимая скорость записи и считывания данных оказывается небольшой. Во втором случае осуществляется запись путем полного насыщения магнитного материала, который в этом случае может иметь только два состояния (положительное или отрицательное). Этот способ является надежным и позволяет получить более высокие скорости работы.

При этом сигнал на барабане может быть записан только в том случае, если он имеет знак, противоположный знаку намагниченности данного места барабана. Важно, чтобы запись каждый раз производилась на одних и тех же местах, так как в противном случае считывание может оказаться невозможным.

Магнитные ленты отличаются от барабанов тем, что магнитный материал наносится не на поверхность барабана, а на поверхность гибкой и плотной ленты, которая наматывается на катушки. При работе лента, перематываясь с одной катушки на другую, движется под записывающими и считывающими головками. Запись и чтение производятся точно таким же образом, как и в случае барабана.

Достижимая поверхностная плотность диполей на ленте и барабане зависит от конструкции головок и колеблется от 4 до 30 диполей на сантиметр длины. Ширина дорожек, т. е. ширина диполей, может быть от 3—4 *мм* до 1 *мм.* Применяемые магнитные головки сходны с головками, используемыми в технике звукозаписи.

Частота повторения считываемых импульсов зависит от скорости вращения барабана или движения ленты и плотности записи. Емкость запоминающих устройств на магнитных барабанах и лентах зависит от их размеров и количества и доходит до сотен тысяч чисел на каждый барабан или ленту.

Достоинствами магнитных запоминающих устройств являются, помимо практически неограниченной емкости, надежность работы, простота принципа действия, отсутствие повреждения записи информации при считывании, а также возможность длительного сохранения информации без реставрации, даже при отклонении источников питания. Последнее достоинство особенно ценно при использовании магнитных лент, которые могут храниться длительное время отдельно от машины с записанной на них информацией.

Основные недостатки магнитных запоминающих устройств (барабанов и лент) связаны с наличием в этих устройствах существенно важных механических частей, движущихся с большими скоростями. Так, стабильность частоты выдачи и записи данных зависит от постоянства скорости движения барабана или ленты. Надежность записи и считывания зависит от точности соблюдения зазора между магнитным материалом и головками. Весьма большое значение имеет точность изготовления барабана и направляющих устройств ленты, а также механизмов движения. Например, небольшие изменения диаметра барабана для разных углов поворота могут привести к появлению ложных сигналов или, наоборот, к пропаданию записи. При большой скорости вращения барабанов возникают значительные радиальные ускорения, которые требуют особых конструктивных мер для обеспечения прочности.

Барабаны изготовляются полностью из алюминия. Магнитная поверхность толщиной 0,01—0,03 *мм* наносится либо гальваническим осаждением, либо распылением суспензии окиси железа, при этом обращается особое внимание на обеспечение однородности магнитного слоя.

К числу недостатков магнитных запоминающих устройств относится невозможность произвольной выборки и записи информации и необходимость затрачивать время на ожидание поворота барабана или перемещение ленты. Это время ожидания для магнитных барабанов может доходить до 10 *мсек.* Особенно сильно этот недостаток проявляется в магнитных лентах, в которых для поиска нужной информации требуется перематывание ленты, на что может затрачиваться время, измеряемое секундами.

**5. Электронно-лучевые запоминающие системы.** Все электроннолучевые или электростатические запоминающие системы отличаются большой скоростью действия (время для записи, считывания и восстановления считываемой информации может быть меньше 5 *мксек*). Емкость устройств этого типа в настоящее время составляет 1024— 2048 чисел. Важной особенностью этих устройств является возможность произвольной выборки и записи

данных в отличие от линий задержки и магнитных барабанов и лент, которые допускают выборку данных только в определенном порядке. В настоящее время электронно-лучевые запоминающие устройства представляют собой основной вид оперативной памяти во многих быстродействующих универсальных цифровых машинах.

Электронно-лучевые трубки, используемые в электронных цифровых машинах, по своему устройству и принципу действия несколько напоминают трубки, используемые в телевидении и радиолокации. Имеется три основных типа запоминающих устройств, использующих электронно-лучевые трубки со статическим хранением зарядов:

а) системы с *задерживающей сеткой;*
б) системы с *поверхностным перераспределением зарядов;*
в) устройства типа *селектрон* с *поддерживающим лучом*

Обычно каждая электронно-лучевая трубка служит для сохранения одного определенного разряда числа, а всего в запоминающем устройстве используется столько трубок, сколько имеется разрядов в двоичном числе. При этом обеспечивается возможность одновременного управления всеми трубками при помощи одной управляющей схемы.

Электронно-лучевая трубка с задерживающей сеткой показана на рис. 69. Трубка состоит из электронного прожектора, формирующего узкий пучок электронов, двигающихся с большой скоростью к аноду, представляющему собой металлическую сигнальную пластину, покрытую тонким слоем диэлектрика, системы отклоняющих пластин, задерживающей сетки и коллектора с выходным контактом. К сигнальной пластине при помощи контакта подводятся электрические импульсы при записи.

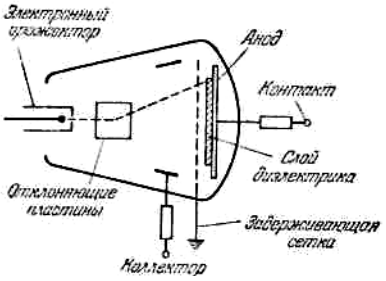


Рис. 69. Электронно-лучевая трубка с задерживающей сеткой.

Принцип действия трубки состоит в том, что под влиянием падающих электронов в том месте экрана, куда попадает поток, выбиваются вторичные электроны. В зависимости от разности потенциалов между экраном и сеткой большее или меньшее количество вторичных электронов возвращается обратно на экран или же проходит через сетку и осаждается на коллекторе (имеющем вид обруча). Если на анод (сигнальную пластину) подан положительный заряд по отношению к сетке, то большинство вторичных электронов вернется на экран и данное место экрана получит отрицательный заряд. Если на анод в момент записи будет подан отрицательный по отношению к сетке потенциал, то большинство вторичных электронов уйдет на коллектор и в этом месте экрана образуется положительный заряд.

Так как экран сделан из диэлектрика, то образовавшиеся в разных местах экрана заряды растекаются медленно и могут сохраняться длительное время. Направляя электронный луч при помощи системы отклоняющих пластин в разные точки экрана, можно таким образом на экране записать значительное количество двоичных цифр. Положительные заряды на экране соответствуют записи единиц, а отрицательные — записи нулей.

Таким образом, экран из диэлектрика и сигнальная пластина представляют собой две обкладки конденсатора, состоящего из большого числа отдельных элементарных конденсаторов, каждый из которых служит для запоминания одной двоичной цифры.

Считывание данных осуществляется путем направления электронного луча в заданное место экрана, где до этого была записана требуемая цифра. В зависимости от того, какой заряд был накоплен в данном месте экрана, между сигнальной пластиной и коллектором возникает выходной сигнал той или другой полярности. Этот сигнал после соответствующего усиления и формирования используется в других устройствах машины. При считывании цифра, записанная в данном месте экрана, стирается, поэтому в устройстве

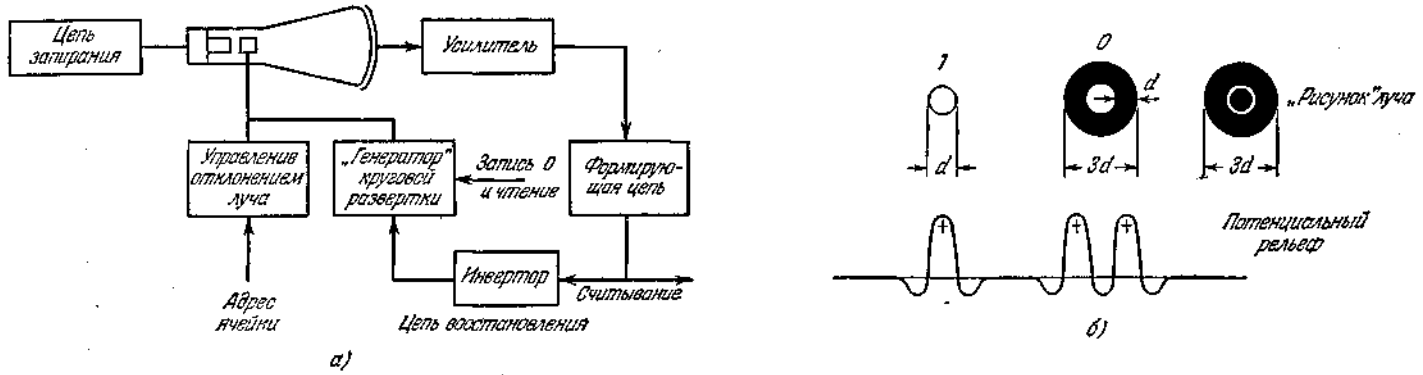


Рис. 70. Запоминающее устройство на электронно-лучевых трубках с поверхностным перераспределением зарядов.

предусматривается автоматическая запись того же числа в данном месте экрана сразу же после считывания.

Так как накопленные заряды постепенно стекают, то для длительного сохранения эти заряды должны периодически восстанавливаться. С этой целью в устройствах предусматриваются специальные схемы восстановления записи, обеспечивающие последовательное считывание и одновременную запись всех данных, хранящихся в запоминающем устройстве.

На рис. 70 изображена блок-схема запоминающего устройства на электронно-лучевых трубках с поверхностным перераспределением зарядов. В этой системе также используется явление вторичной эмиссии электронов с поверхности диэлектрического экрана. В зависимости от скорости падающих на экран первичных электронов с поверхности экрана вылетает большее или меньшее количество вторичных электронов. С увеличением (до определенного предела) скорости падающих первичных электронов увеличивается количество выбиваемых ими вторичных электронов и при определенном значении скорости падающего потока экрана в данном месте будет приобретать положительный заряд.

Для записи двоичных чисел на поверхности экрана электронный луч должен образовывать два *рисунка* или два различных распределения зарядов. Эти рисунки должны существенно отличаться друг от друга, чтобы обеспечить достаточную величину считываемого сигнала.

Из различных типов рисунков наиболее удобным и эффективным оказался рисунок *точка — кольцо.* Образование положительного заряда в данном месте экрана в виде точки соответствует записи единицы, а образование положительного заряда в виде кольца соответствует записи нуля. На рис. 70, б показан потенциальный рельеф (распределение зарядов), получающийся на экране трубки при записи нуля или единицы. Из рисунка видно, что величина заряда в случае записи единицы (точки) имеет меньшую величину, чем заряд нуля (кольцо).

Рассмотрим блок-схему, показанную на рис. 70, *а.*

Запирающая цепь запирает электронный луч на то время, когда отклоняющая система срабатывает для направления луча из одной точки экрана в другую. Генератор круговой развертки при записи нуля или при считывании нуля или единицы осуществляет развертку луча по кругу малого радиуса вокруг той точки, куда направлен центр луча.

Если при считывании луч направлен в такое место экрана, где до этого было записано кольцо, то образующийся выходной сигнал будет мал. Если же в данном месте экрана раньше была записана точка, то на выходном электроде, расположенном сзади экрана, возникнет импульс тока, который после соответствующего усиления и формирования будет использоваться как сигнал единицы. В этой системе также необходимо периодическое восстановление записи.

Принцип действия запоминающего устройства типа селектрон основан на том, что бомбардируемый потоком электронов экран из диэлектрика может принимать два фиксированных значения потенциала: или потенциал катода, испускающего электронный поток, или потенциал анода (ускоряющего электрода). Долговременное сохранение (поддержание) потенциала на экране осуществляется с помощью специального рассеивающего электронного прожектора, испускающего широкий поток медленных электронов. Между катодом и экраном в селектроне помещается ускоряющая сетка-коллектор.

Диэлектрический экран разбит на отдельные прямоугольные участки при помощи двух взаимно-перпендикулярных групп параллельных стерженьков, помещенных между катодом и коллектором. Эти стерженьки служат для выбора соответствующего участка на экране. Расстояния между стерженьками подобраны так, что электронный поток от катода может пройти к экрану только через ту клетку, которая образована четырьмя стерженьками, имеющими положительный потенциал.

Выводы от стерженьков комбинируются между собой таким образом, чтобы при минимальном числе выводов из селектрона обеспечивалась возможность выбора любой клетки. Так, для селектрона, имеющего 256 клеток, достаточно иметь 18 выводов.

Нормально все клетки в селектроне открыты и рассеянный электронный поток облучает равномерно весь экран, поддерживая сколь угодно долго записанные на экране данные.

Для записи 0 (или 1) в определенной ячейке все ячейки селектрона, кроме выбранной, запираются подачей отрицательного напряжения на соответствующие выводы стерженьков, а на сигнальную пластину, расположенную позади экрана, подается положительный (или отрицательный) импульс. После снятия этого импульса потенциалы всех ячеек возвращаются к первоначальному уровню, кроме выбранной ячейки, в которой произошло увеличение или уменьшение заряда. Это изменение заряда уравновешивается током смещения.

Считывание данных осуществляется также путем выделения определенной клетки и измерения тока смещения, возникающего между сигнальной пластиной и коллектором при подаче определенного потенциала на сигнальную пластину. В некоторых электростатических системах с поддерживающим лучом используются отдельные прожекторы для записи данных и для считывания. Конструкция таких трубок весьма сложна. Основным их достоинством является отсутствие необходимости восстанавливать информацию как для ее сохранения, так и при считывании.

Недостатком электронно-лучевых запоминающих систем является ограничение числа непрерывных повторных обращений в одну и ту же точку экрана, так как при длительном воздействии электронного луча на одно и то же место экрана могут возникнуть искажения в распределении зарядов в соседних точках. Однако этот недостаток в настоящее время не имеет большого значения. Существующие трубки обладают высокой фокусировкой электронного потока и обеспечивают практически достаточное число повторных обращений в одну и ту же точку.

**6. Запоминающее устройство на магнитных сердечниках.** Одним из наиболее перспективных типов запоминающих устройств являются системы с использованием магнитных материалов с прямоугольной петлей гистерезиса, так называемых ферритов.

Мы уже рассматривали общие свойства магнитных сердечников и применение их для построения логических переключательных схем (§§ 13, 14). Рассмотрим применение этих сердечников для построения запоминающих устройств электронных цифровых машин.

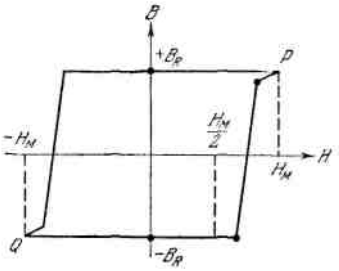


Рис. 71. Идеализированная петля гистерезиса

Кольцевые сердечники из ферромагнитного материала обеспечивают сохранение двоичных цифр (0 и 1) в виде остаточного магнетизма сердечника той или иной полярности.

Принцип действия запоминающего устройства основан на том, что для перемагничивания сердечника из материала с прямоугольной петлей гистерезиса необходимо создать поле определенной напряженности, причем если напряженность поля будет меньше этой величины, например, в два раза, то сердечник не перемагнитится даже при многократном приложении импульсов поля с такой напряженностью. На рис. 71 показана идеализированная петля гистерезиса. Если, например, магнитное состояние сердечника характеризуется точкой *P,* то для перемагничивания сердечника необходимо поле напряженностью не меньше, чем $-H_м$, и наоборот, если состояние сердечника соответствует точке , то для его перемагничивания необходимо поле напряженностью $+H_м$.

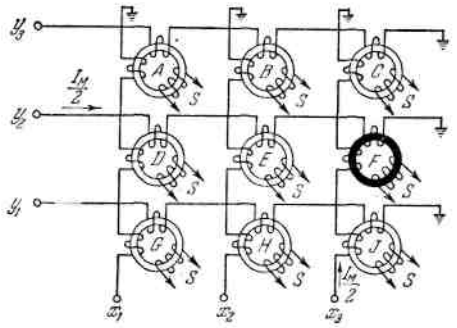


Рис. 72. Запоминающее устройство на девяти сердечниках.

Каждый сердечник служит для запоминания одной двоичной цифры (0 или 1). Сердечники располагаются правильными рядами, образуя плоскую или пространственную прямоугольную систему. Каждый сердечник имеет две (при плоском расположении) или три (при пространственном расположении) первичные обмотки, служащие для записи данных и выбора нужного сердечника, и одну вторичную обмотку, служащую для считывания. На рис. 72 показана двумерная схема соединения сердечников, состоящая из девяти сердечников.

Горизонтальные ряды сердечников образуют отдельные регистры с количеством двоичных разрядов, равным числу сердечников в ряду. Первичные обмотки сердечников каждого ряда (по одной от каждого сердечника) соединены последовательно, образуя цепь, открывающую соответствующий регистр ($y_1, y_2, y_3$).

Оставшиеся вторые первичные обмотки сердечников соединены последовательно по вертикальным колонкам, образуя линии подачи соответствующих разрядов записываемых чисел ($x_1, x_2, x_3$). Вторичные обмотки сердечников *S* служат для выдачи считываемых двоичных цифр. Эти обмотки также соединяются последовательно для всех сердечников, образующих одну вертикальную колонку, и таким образом, одинаковые разряды всех регистров соединены последовательно и имеют общий выход.

Если, например, пустить два тока величиной ½ по цепям $x_3$ и $y_2$, то напряженность поля достигает величины $H_м$ только в сердечнике *F,* в котором одновременно будут действовать оба тока. В сердечниках *D, E,* С, *J* напряженность будет $\frac{H_м}{2}$ и в силу прямоугольного характера петли гистерезиса не сможет перемагнитить эти сердечники. Сердечники *A, B, G, H,* вообще, не будут испытывать действия какого-либо поля. Сердечник же *F* намагнитится полностью в соответствии с полярностью действующею суммарного поля.

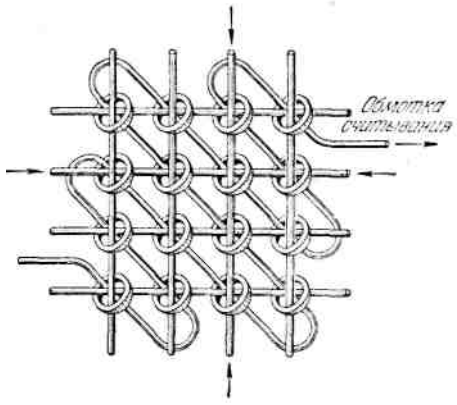


Рис. 73. Запоминающее устройство на кольцевых сердечниках.

Если сердечник *F* предварительно был намагничен в противоположном направлении, то процесс перемагничивания его вызовет появление в обмотке *S* выходного сигнала большой амплитуды. На выходных обмотках других сердечников никакого сигнала не появится, независимо от того, были они раньше намагничены или нет, так как они не подвергаются перемагничиванию.

Если же направление остаточного магнетизма сердечника *F* совпадает с направлением перемагничивающего поля, то на выходной обмотке *S* сигнала не будет, так как перемагничивания не произойдет.

Построены запоминающие устройства на 100 000 и более магнитных кольцевых сердечниках, допускающие произвольную выборку чисел. В запоминающих устройствах на магнитных сердечниках время записи и считывания составляет менее 10 *мксек.*

Кольцевые сердечники обычно не имеют специальных обмоток; вместо обмоток сердечники пронизаны тремя проводниками. Один проводник относится к вертикальной системе проводников, второй проводник относится к горизонтальной системе проводников, а третий проводник, представляющий собой обмотку считывания, последовательно пронизывает все сердечники (рис. 73) данного разряда. В случае двумерной схемы, как на рис. 72, это будут сердечники одной вертикальной колонки; в случае трехмерной схемы это будут сердечники, относящиеся

к одной плоскости (рис. 73). Основными достоинствами запоминающих устройств на магнитных сердечниках являются простота конструкции, высокая надежность действия, высокая скорость записи и считывания, возможность получения достаточно большой емкости сравнительно простыми средствами, возможность сохранять записанную информацию неограниченно долгое время без всякой затраты энергии.

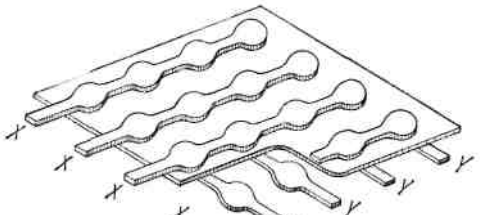


Рис. 74. Запоминающее устройство на ферроэлектриках

**7. Ферроэлектрическиезапоминающие устройства.** Существуют так называемые *ферроэлектрические* материалы, имеющие зависимость диэлектрической проницаемости от напряженности электрического поля такого же вида, как и кривая намагничивания для ферромагнитных материалов. К таким веществам, например, относится титанат бария..

На рис. 74 показано запоминающее устройство с использованием ферроэлектриков. Принцип действия здесь такой же, как и в случае магнитных сердечников с прямоугольной петлей гистерезиса, только вместо токов здесь используются напряжения. К одной из проводящих полосок *X* подводится положительный потенциал, а к одной из полосок *Y* подводится равный по величине отрицательный потенциал.

Элементарная площадка пластины из ферроэлектрика, находящаяся между обеими полосками, электризуется в направлении, соответствующем приложенному полю. Элементы пластины из ферроэлектрика, расположенные вдоль полосок *X* и *Y,* испытывают действие поля с вдвое меньшей напряженностью и в силу прямоугольного характера зависимости диэлектрической проницаемости от напряженности поля останутся практически в том же состоянии, что и до приложения поля. Остальные элементы пластины из ферроэлектрика вообще не будут подвергаться действию поля. При считывании поле образуется в противоположном направлении. Импульс тока получается большой величины только в том случае, если соответствующий элемент был раньше наэлектризован в противоположном направлении.

Управляющие схемы для записи и считывания данных для фероэлектрических запоминающих устройств могут быть весьма различными.

На рис. 75 показана схема запоминающего устройства на ферроэлектрике с использованием конденсаторов. Запись единицы осуществляется приложением напряжения одной полярности, а запись нуля — приложением напряжения противоположной полярности. При считывании данных подается напряжение такой же полярности, как и при записи нуля. При этом, если была записана единица,

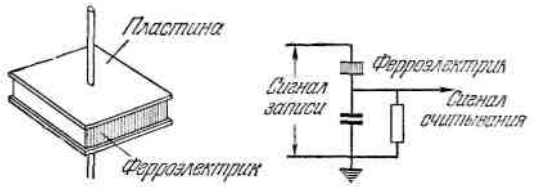


Рис. 75. Запоминающее устройство на ферроэлектрике с использованием конденсатора.

то получится выходной импульс большой величины, если же был записан нуль, то выходной сигнал будет незначительным. Выходной ток при считывании заряжает конденсатор, и по величине напряжения на конденсаторе определяется, какая цифра была записана.

Основные преимущества запоминающих устройств, построенных на ферроэлектриках, состоят в следующем: для записи и хранения данных почти не требуется затраты мощности, хранение большого количества данных производится на очень малых площадях и данные могут храниться в течение длительного времени без восстановления. Так, исследования показали, что кристаллы титаната бария способны к запоминанию на площади 3,2 $см^2$ до 250 двоичных знаков.

Кроме того, ферроэлектрики пригодны для работы при низких напряжениях (менее 10 *в*), что важно для создания вычислительных машин на полупроводниковых триодах.

## § 22. Основные образцы советских электронных цифровых вычислительных машин

В заключение настоящей главы, посвященной техническим принципам устройства электронных цифровых программно-управляемых машин, мы кратко опишем некоторые машины, созданные в Советском Союзе.

**1. Машина *БЭСМ*.** Весьма совершенной машиной по своим логическим и техническим принципам и математическим возможностям в нашей стране является машина *БЭСМ* (быстродействующая электронная счетная машина), созданная в 1953 г. в Академии наук СССР под руководством Героя Социалистического труда академика С. А. Лебедева. Общий вид машины показан на рис. 76.

*БЭСМ* работает со скоростью 10 000 трехадресных команд в секунду. Машина оперирует с двоичными числами, которые соответствуют приблизительно девятиразрядным десятичным числам. Оперативное запоминающее устройство построено на магнитных сердечниках и имеет емкость 2048 чисел.

*БЭСМ* имеет внешний накопитель, построенный на четырех магнитофонах. Запись и хранение информации

производятся на узких магнитных лентах; на каждую ленту может быть записано до 30 тысяч чисел, и общая емкость внешнего накопителя составляет 120 000 чисел. Скорость работы внешнего накопителя характеризуется записью или считыванием 400 чисел в секунду.

Кроме того, в машине *БЭСМ* имеется промежуточный накопитель на магнитном барабане емкостью 5120 чисел со скоростью выборки чисел до 800 в секунду.

Данные для ввода в машину вручную перфорируются на бумажную ленту. Ввод в машину осуществляется путем пропускания бумажной ленты с пробитыми данными под источником света. Свет, проходя через пробитые в ленте отверстия на фотоэлемент, вызывает появление электрических сигналов, поступающих в машину. Скорость ввода данных в машину с перфоленты составляет 20 чисел в секунду.

Результаты решения задач фиксируются на магнитной ленте, которая после этого вынимается из машины и вставляется в специальное фотопечатающее устройство. Фотопечатающее устройство переводит информацию, записанную на магнитной ленте двоичным кодом, в десятичные цифры, которые фиксируются в виде таблиц на ленте. Скорость работы фотопечатающего устройства 200 чисел в секунду. Помимо фотопечатающего устройства, *БЭСМ* имеет еще электромеханическое печатающее устройство, связанное непосредственно с машиной и работающее значительно медленнее (1,5 числа в секунду). Это устройство используется только при выводе из машины небольших количеств чисел.

*БЭСМ* имеет около 5 тысяч электронных ламп и собрана на одной основной стойке (см. рис. 76). Слева от стойки виден пульт управления, служащий для пуска и остановки машины и для контроля за ее работой. Машина *БЭСМ* построена в основном из стандартных двухламповых и четырехламповых ячеек, на которых монтируются различные элементарные схемы (триггеры, вентили, усилители и т. д.). Длительная эксплуатация *БЭСМ* показала ее высокую надежность и устойчивость в работе. Полезное время работы составляет в среднем 72%, время, потерянное за счет неисправностей, — 8% и время, затраченное на профилактику, — 20%. Время работы машины без сбоев в течение 10 часов и выше составляет примерно 70% от полезного времени работы машины.

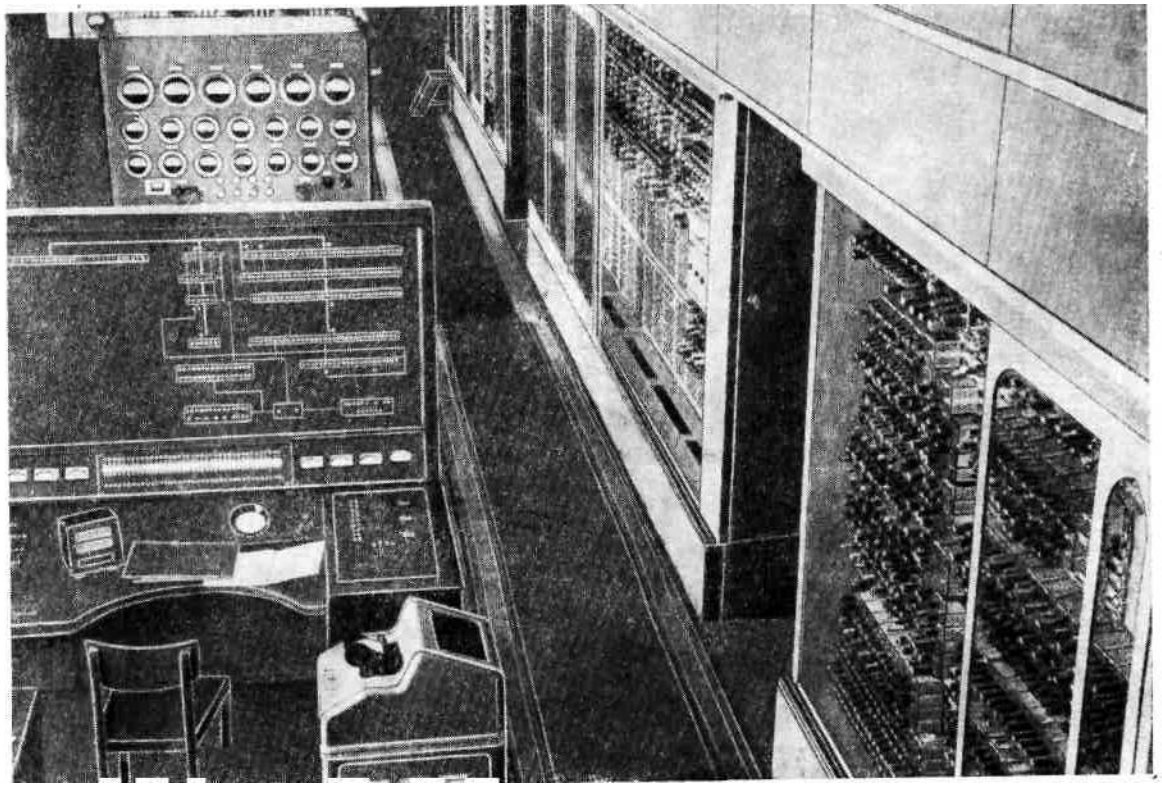


Рис. 76 Быстродействующая электронная счетная машина БЭСМ Академии наук СССР.

Для машины разработана система контрольных задач — тестов, позволяющих быстро находить неисправности в машине, а также система профилактических испытаний, позволяющая заранее обнаруживать места возможных неисправностей и их предотвращать.

**2. Машина *Стрела*.** На рис. 77 показан внешний вид электронной цифровой вычислительной машины *Стрела,* созданной в 1953 г. под руководством Героя Социалистического Труда Ю. Я. Базилевского. Ряд машин этого типа успешно эксплуатируется в различных научных учреждениях нашей страны.

Машина *Стрела* принадлежит к классу больших машин и обладает высокоразвитой и логически законченной структурой, что обеспечивает ее большую производительность при решении сложных и громоздких по объему вычислений задач.

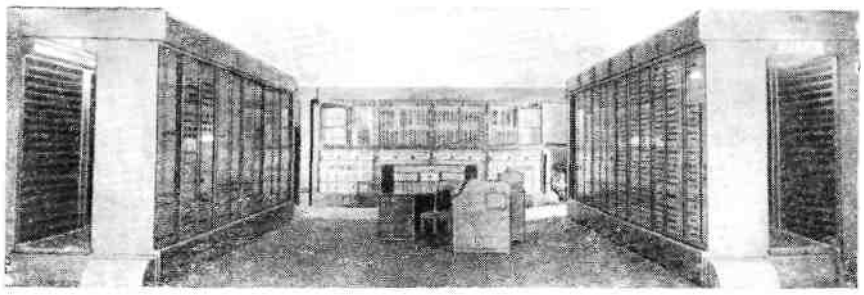


Рис. 77. Электронная цифровая вычислительная машина *Стрела.*

Машина *Стрела* собрана на трех основных стойках, расположенных в виде буквы П (рис. 77). Справа находится стойка арифметического устройства, слева — стойка внешнего накопителя и некоторых вспомогательных устройств, посредине находится стойка оперативного запоминающего устройства и устройства управления. В центре расположен пульт ручного управления и устройства ввода данных (слева) и вывода результатов (справа).

Машина *Стрела* обеспечивает скорость вычисления в 2000—3000 трехадресных операций в секунду. На рис. 78 показано арифметическое устройство машины. Оно выполняет арифметические операции (сложение, вычитание, умножение) и ряд дополнительных операций (вычитание модулей чисел, сдвиг числа, выделение части числа и другие).

На рис. 79 показано устройство для подготовки перфокарт к вводу в машину. Это устройство состоит из двух частей: клавишного устройства (справа) и входного перфоратора. На клавишном устройстве работает человек-оператор, который читает с бланка, закрепленного вверху на валике, числа или команды программы и нажимает соответствующие клавиши. После того как набраны все цифры числа, нажимается специальная клавиша и число пробивается в виде ряда отверстий в одной строчке перфокарты.

Подготовленная колода перфокарт вынимается из входного перфоратора и вводится в читающее устройство машины, которое автоматически вводит данные в оперативное запоминающее устройство машины. Оперативное запоминающее устройство машины построено на электронно-лучевых трубках и имеет емкость 2048 чисел (или команд). Внешний вид оперативного запоминающего устройства показан на рис. 80. На рис. 81 показан внешний вид одной электронно-лучевой трубки.

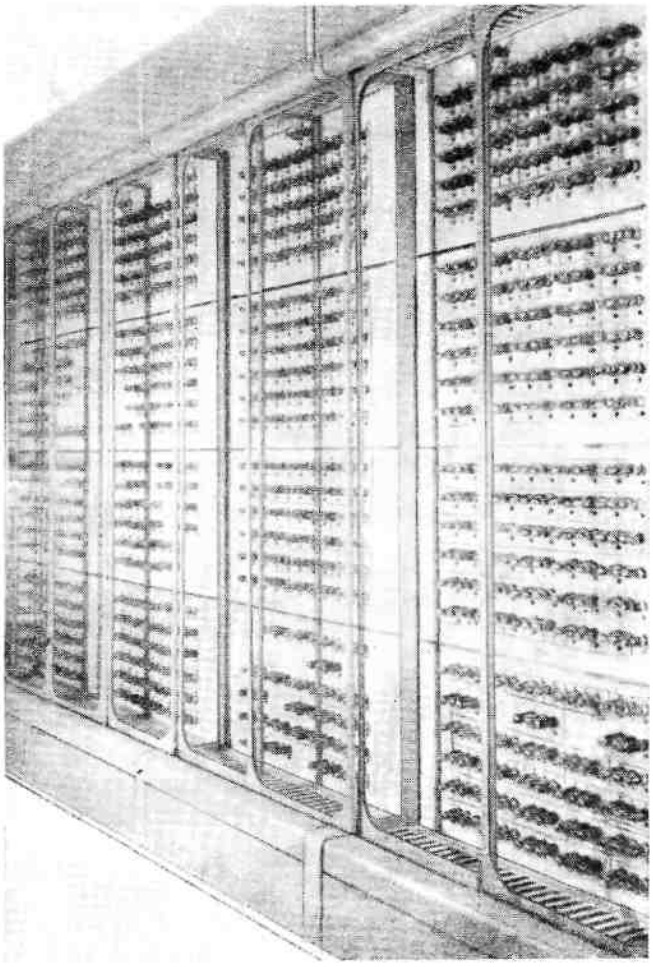


Рис. 78. Арифметическое устройство машины *Стрела.*

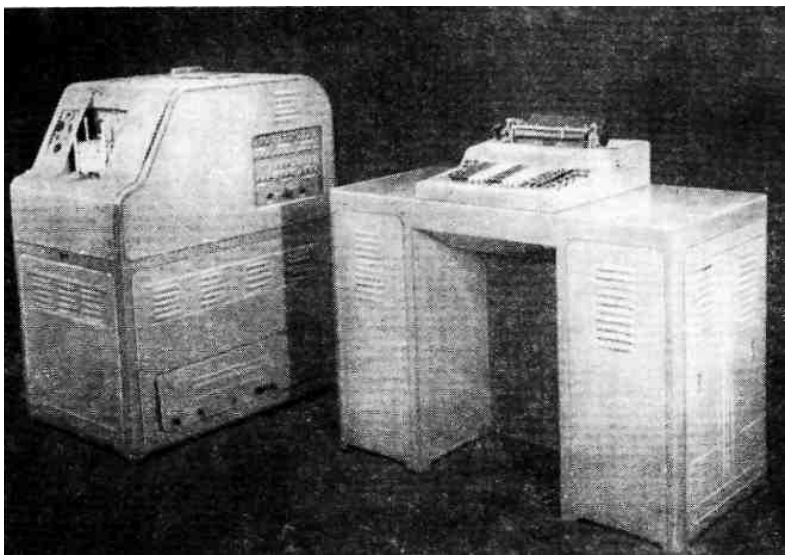


Рис. 79. Устройство для подготовки перфокарт к вводу в машину

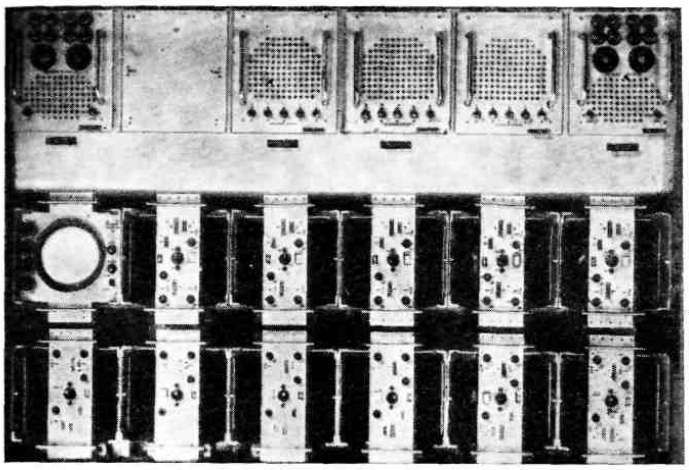


Рис. 80. Внешний вид оперативного запоминающего устройства

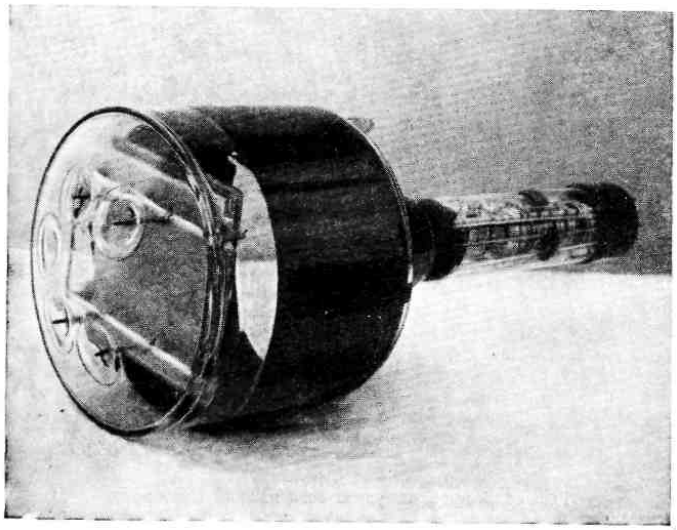
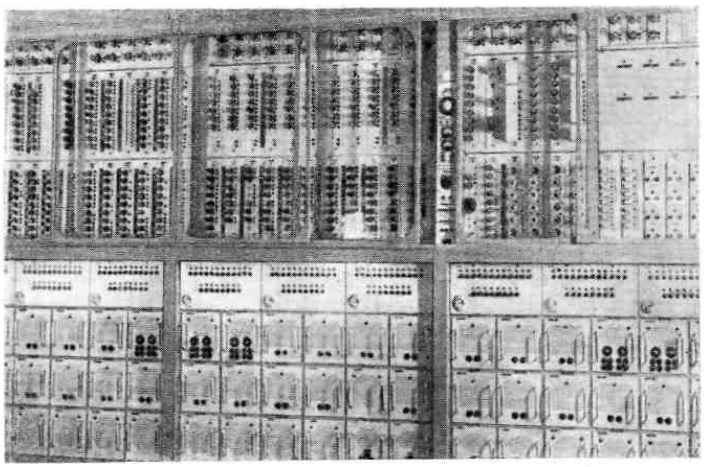


Рис. 81. Электронно-лучевая трубка машины *Стрела.*

В машине 43-разрядные двоичные коды представляются параллельным способом, т. е. прием, запись и выдача кодов производятся одновременно по всем 43 разрядам. В соответствии с этим в оперативном запоминающем устройстве имеется 43 электронно-лучевые трубки — по одной трубке на каждый разряд. Машина оперирует с числами с плавающей запятой, которые соответствуют практически 10-11-разрядным десятичным числам, и допускает возможность их изменений в пределах от $10^{-19}$ до $10^{+19}$.

Внешний накопитель включает в себя два блока с магнитной лентой шириной 125 *мм* и длиной до 100 *м.* На магнитной ленте числа располагаются группами по зонам; на каждой ленте могут быть записаны 253 зоны различного размера таким образом, чтобы общая емкость каждой ленты не превышала 100 000 чисел. Всего внешний накопитель машины *Стрела* может вместить в себя до 200 000 чисел. Внешний накопитель показан на рис. 82, на котором видны катушки с магнитной лентой и лентопротяжный механизм. С одной катушки лента сматывается, а на другую — наматывается.

На рис. 83 показан внешний вид устройства управления машины *Стрела,* а на рис. 84 — пульт ручного управления машиной. Пульт ручного управления позволяет оператору запускать и останавливать машину, следить за ходом выполнения команд программы в процессе автоматической работы, а также вводить и выводить из оперативного запоминающего устройства отдельные числа и команды во время остановки машины.



.Рис. 83. Устройство управления машины *Стрела*

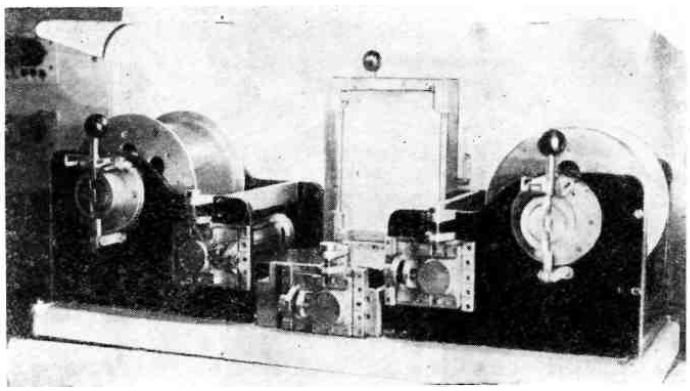


Рис. 82. Внешний накопитель машины *Стрела*

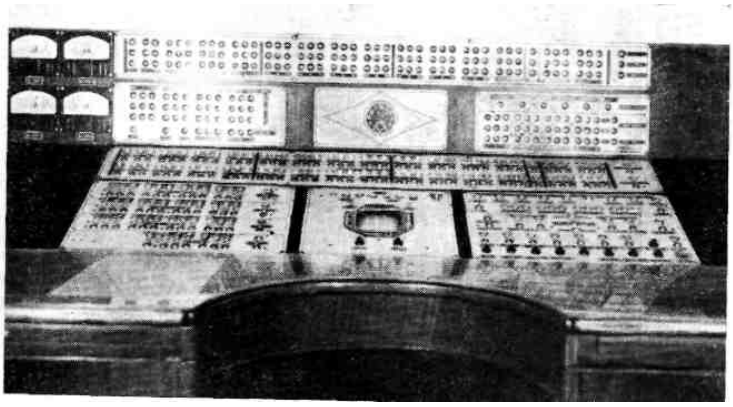


Рис. 84. Пульт ручного управления машиной *Стрела.*

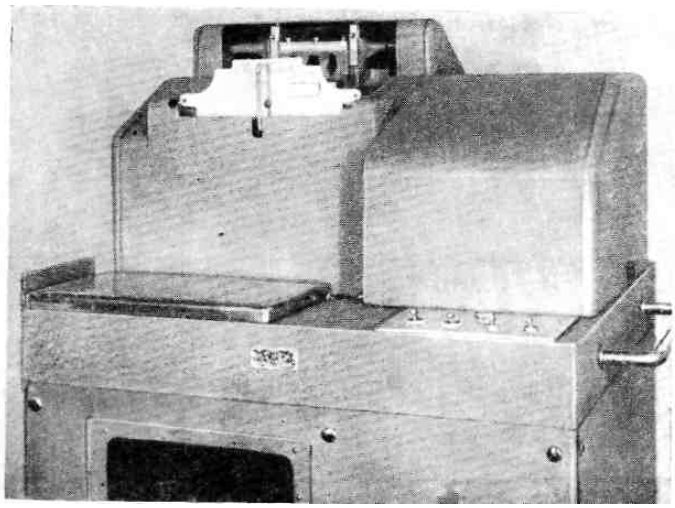


Рис. 85. Устройство ввода данных в машину *Стрела.*

На рис. 85 показано устройство ввода данных в машину. В это устройство, называемое также *читающим* устройством, вкладывается колода перфокарт с пробитыми на них исходными данными и программой решения задачи, и читающее устройство передает эту информацию в виде электрических сигналов в оперативное запоминающее устройство.

На рис. 86 показано устройство вывода результатов, называемое выходным перфоратором. Результаты решения задачи, полученные в оперативном запоминающем устройстве машины, передаются в виде электрических сигналов в выходной перфоратор и здесь представляются в виде системы отверстий на перфокартах.

На рис. 87 показано печатающее устройство машины. Это устройство имеет специальный приемник (слева), в который вставляется колода перфокарт, с пробитыми на них результатами решения, и устройство переводит перфорированные данные на перфокартах в десятичную систему счисления и печатает их в виде числовых таблиц на бумаге в трех экземплярах.

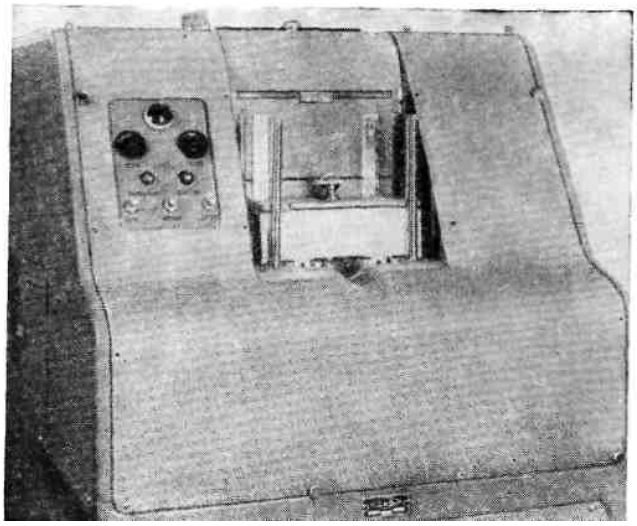


Рис. 86. Выходной перфоратор машины *Стрела.*

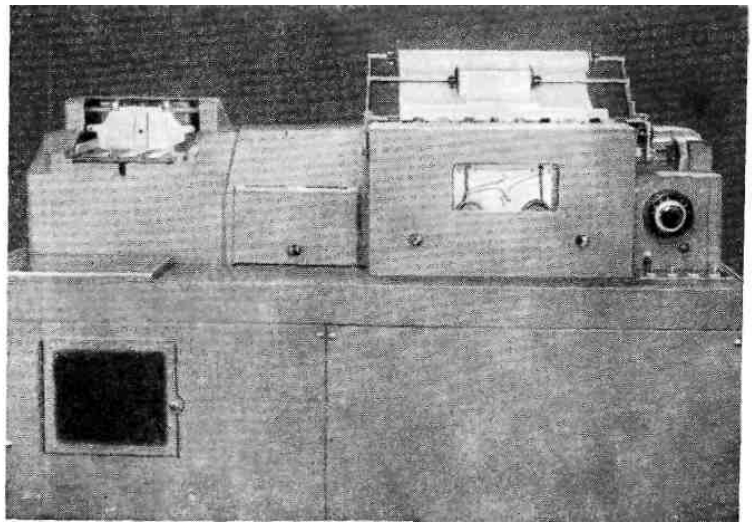


Рис. 87. Печатающее устройство машины *Стрела.*

Машина *Стрела* имеет около шести тысяч электронных ламп и несколько десятков тысяч полупроводниковых выпрямителей (диодов). Общая потребляемая машиной мощность 150 *квт,* в том числе сама машина потребляет 75 *квт,* 25 *квт* идет на вентиляционную установку и 50 *квт* на холодильную установку.

Машина *Стрела* в течение длительной эксплуатации показала высокую надежность и устойчивость работы. Среднее время полезной работы машины составляет 15—18 часов в сутки. Для машин типа *Стрела* разработаны достаточно полные системы контрольных задач, позволяющих проверять машину и находить неисправности, а также система профилактических мероприятий, повышающих надежность работы машины.

**3. Машина *Урал*.** На рис. 88 показан внешний вид электронной цифровой вычислительной машины *Урал,* созданной в 1954 г. под руководством инженера Б. И. Рамеева. Эта машина относится к классу малых машин универсального назначения и предназначена для использования в научно-исследовательских институтах, конструкторских бюро, на заводах, в высших учебных заведениях и в других учреждениях, связанных с выполнением значительных вычислительных работ.

Машина *Урал* серийно выпускается нашей промышленностью. Эта машина значительно меньше по количеству аппаратуры, чем машина *Стрела* и *БЭСМ,* проще в производстве и эксплуатации. Машина оперирует с двоичными числами, которые соответствуют приблизительно 10-разрядным десятичным числам. Она работает по одноадресной системе команд и выполняет 100 команд в секунду. Оперативное запоминающее устройство емкостью 1024 числа построено на магнитном барабане; скорость вращения барабана 6000 *об/мин,* и таким образом, время выборки одного числа составляет 10 *мсек.*

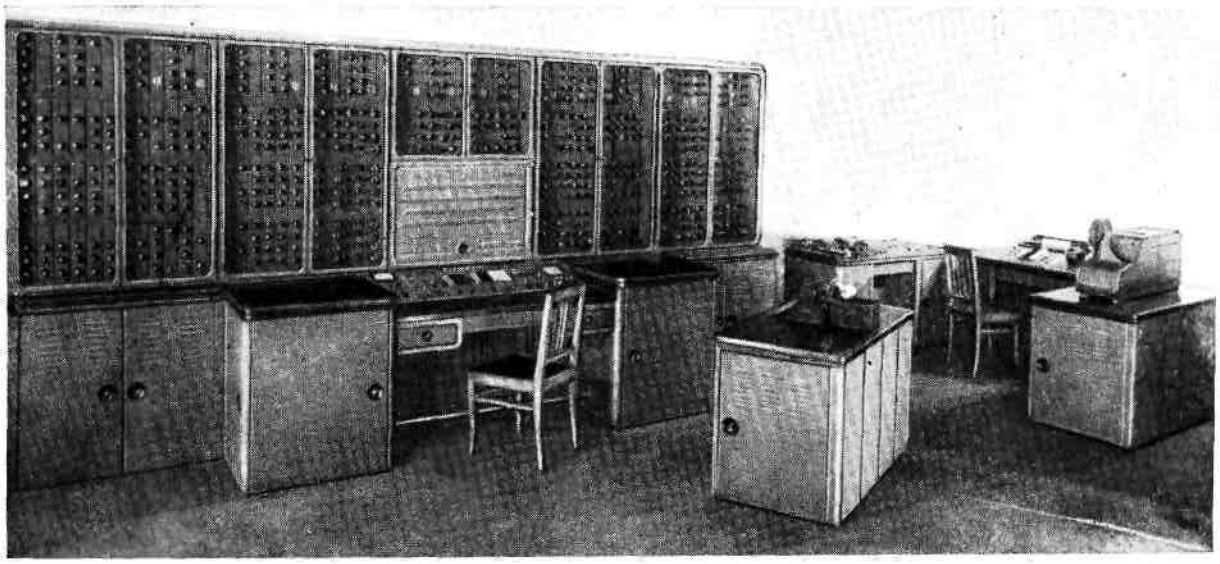


Рис. 88 Малая цифровая электронная машина *Урал*.

Внешний накопитель машины *Урал* имеет емкость 40 000 чисел; запись и хранение информации осуществляются на узкой магнитной ленте (35 *мм)* по зонам. При работе внешнего накопителя лента перемещается со скоростью 2 *м/сек,* что обеспечивает скорость записи или чтения 4500 чисел в минуту.

Ввод данных в машину производится при помощи перфорированной ленты. Для этой цели используется стандартная кинолента шириной 35 *мм.* Перфорирование осуществляется вручную на специальных входных перфораторах. Затем лента склеивается в кольцо и вставляется в устройство ввода. Чтение данных с ленты производится при помощи фоточитающих приспособлений, которые при прохождении через них ленты с отверстиями посылают в машину электрические сигналы.

Вывод результатов производится из оперативного запоминающего устройства в печатающее устройство, которое выдает результаты в виде цифровых таблиц в десятичной системе счисления со скоростью 100 чисел в минуту.

Машина *Урал* построена на одноламповых типовых ячейках и имеет 800 электронных ламп и 3000 полупроводниковых выпрямителей (диодов). Потребляемая мощность составляет 8 *квт.* Машина собрана на одной стойке, посредине стойки расположен пульт управления (рис. 88). Рядом видны внешние устройства машины.

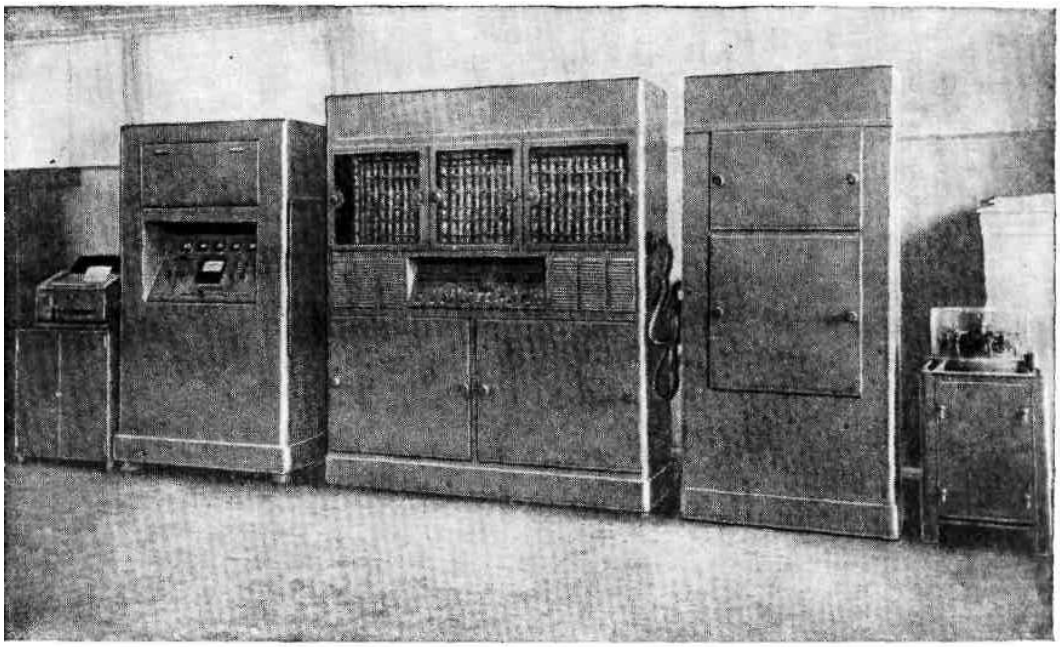


Рис.89. Малая цифровая электронная вычислительная машина М-3

Вся машина может быть размещена в комнате площадью 60 $м^2$.

Входные устройства машины (клавишное устройство, контрольно-считывающее устройство и входной перфоратор) работают независимо от электронной части машины и могут располагаться в отдельной комнате.

**4. Машина *М-3*.** На рис. 89 показана малогабаритная универсальная цифровая вычислительная машина *М-3,* созданная под руководством члена-корреспондента АН СССР И. С. Брука.

*М-3* предназначена для выполнения широкого круга математических вычислений сравнительно небольшого объема. Достоинствами машины являются небольшие габариты, простота эксплуатации, невысокая стоимость.

Машина параллельного действия с фиксированной запятой, оперирует с 30-разрядными двоичными числами (31 разряд является разрядом знака числа), что соответствует девятиразрядным десятичным числам.

Машина имеет одно запоминающее устройство на магнитном барабане емкостью 2048 чисел или команд. 30 арифметических операций в секунду. В случае подключения к машине запоминающего устройства на магнитных сердечниках возможно повышение быстродействия машины до 1500 операций в секунду. Ввод данных в машину и вывод результатов осуществляются при помощи стандартной перфоленты и стандартной телеграфной аппаратуры (трансмиттер и телетайп) со скоростью 7 чисел в секунду. Более быстрый ввод данных производится при помощи фотоэлектрического вводного устройства, которое обеспечивает скорость ввода 30 десятичных чисел в секунду. Потребляемая мощность составляет 10 *квт.* Для размещения машины достаточна площадь 30—40 $м^2$. Машина имеет 770 электронных ламп и 3000 купроксных диодов. В машине *М-3* используется двухадресная система команд, при которой каждая команда состоит из кода операции и двух адресов чисел.

Машина выполняет четыре арифметических действия и ряд логических и вспомогательных операций: ввод чисел с перфоленты, условный и безусловный переходы, логическое умножение и др. Особенностью построения машины является применение асинхронного принципа работы устройства управления. В отличие от рассмотренного нами (§ 19) порядка работы устройства управления, построенного по синхронному принципу, в машине *М-3* последовательность работы отдельных блоков и устройств машины определяется взаимодействием этих блоков между собой по принципу *приказ—ответ.* Переход к выполнению следующего элементарного действия в машине происходит только после того, как будет получен сигнал об окончании выполнения предыдущего действия. При таком способе обеспечивается в значительной степени независимость работы отдельных устройств машины, что облегчает наладку машины. Весьма важным является также то, что в случае нарушений в ходе работы какого-либо устройства происходит остановка машины и имеется возможность сравнительно просто обнаружить неисправное устройство. Недостатком машины *М-3* является отсутствие внешнего накопителя на магнитных лентах. Машина построена из типовых одноламповых, двухламповых и четырехламповых ячеек и собрана в трех шкафах. Главный шкаф содержит в себе арифметическое устройство, устройство управления, пульт управления и электронный блок устройства ввода и вывода.

Шкаф запоминающего устройства содержит магнитный барабан и схемы управления магнитного барабана.

Шкаф питания содержит стабилизаторы, выпрямители и пульт управления питанием.

## ГЛАВА IV
## НАПРАВЛЕНИЯ РАЗВИТИЯ ЭЛЕКТРОННЫХ ЦИФРОВЫХ МАШИН

### § 23. Общие тенденции развития

Основной тенденцией в развитии техники электронных цифровых машин является стремление к максимальному повышению их эффективности, что наряду со значительным расширением областей применения этих машин обусловило необходимость некоторого приспособления машин к условиям применения и особенностям решаемых задач,

В настоящее время определились три основных направления в развитии электронных цифровых машин:

1) машины для выполнения сложных математических вычислений, возникающих при разработке различных научных и технических проблем и вопросов;
2) машины для обработки информации;
3) машины для автоматического управления производственными агрегатами и другими реальными объектами.

Рассмотрим основные особенности и типовые примеры машин для каждого из указанных направлений.

**1. Машины для математических вычислений.** Для электронных цифровых машин, предназначенных для научных и технических расчетов, требование эффективности сводится в основном к получению возможно большего полезного машинного времени. Это значит, что машина должна каждые сутки большую часть времени затрачивать на полезную вычислительную работу и по возможности меньше простаивать из-за неисправностей и различного рода регламентных работ.

К машинам этого класса не предъявляется требование безотказности в работе. Отдельные сбои, вообще говоря, допустимы. Эти сбои приводят к необходимости повторного решения задач и, в конечном счете, к некоторому понижению показателя эффективности.

Поэтому в машинах этого класса обычно не применяются такие сложные и достаточно громоздкие способы обеспечения надежности, как дублирование машин или дублирования их отдельных устройств.

Для машин этого класса характерными являются следующие основные черты:

1. Разветвленная и гибкая система команд, обеспечивающая удобство программирования. Большое количество различных арифметических и логических операций, в том числе сложных операций (умножение, деление, извлечение корня, получение тригонометрических функций и др.).

2. Большой объем оперативных запоминающих устройств, предназначенных для хранения как числовой информации, так и программ.

3. Наличие промежуточных запоминающих устройств на магнитных барабанах сравнительно большой емкости. Соотношение между емкостью оперативных запоминающих устройств и промежуточных запоминающих устройств составляет обычно 2—5.

4. Внешнее запоминающее устройство на магнитных лентах имеет не особенно большую емкость; его емкость превышает емкости оперативного запоминающего устройства не более чем в 50—100 раз.

5. Устройства ввода и вывода обычного типа, использующие перфокарты или перфоленты. В электронных цифровых машинах, предназначенных для математических вычислений, устройства ввода и вывода, как правило, включают в себя всего один работающий комплект (естественно, что, кроме того, имеются еще запасные комплекты).

6. Основным методом контроля правильности работы является логический контроль или *макроконтроль.* При макроконтроле проверяется правильность выполнения не отдельных операций, а правильность работы машины в целом, на основе проверки результатов решения по отдельным достаточно крупным участкам вычислительного процесса. Зачастую такая проверка осуществляется путем повторного решения и сравнения результатов решения целых задач. В некоторых машинах частично применяются и методы схемного контроля правильности операций и даже отдельные схемы для автоматического исправления ошибок.

7. Для машин этого класса характерным является большое разнообразие решаемых задач и программ для них; оценка эффективности применения машины обычно производится с учетом затрат труда на программирование. Производственный процесс работы на машине состоит из чередующихся законченных циклов работы, соответствующих отдельным задачам.

Класс машин, предназначенных для математических вычислений, характеризуется весьма широким диапазоном требований к быстродействию и производительности машин. Эти машины, в зависимости от их производительности можно разделить на три группы, к каждой из которых предъявляются требования различного характера:

а) уникальные большие машины, предназначенные для решения особо сложных математических вычислительных задач;

б) серийные машины средних размеров, предназначенные для выполнения крупных вычислительных работ;

в) вычислительные машины малых размеров, предназначенные для выполнения массовых научных и инженерных расчетов сравнительно небольшой сложности.

Необходимость создания сверхбыстродействующих уникальных машин вызвана тем, что современная наука выдвигает весьма важные для практики математические задачи, требующие выполнения огромного объема вычислений.

В связи с этим в настоящее время разрабатываются машины с весьма большими скоростями работы. Фирмой ИБМ (США) в 1955 г. была построена большая уникальная машина *НОРК* со скоростью 60 000 сложений и 30 000 умножений в секунду. Машина *НОРК* имеет 8000 электронных ламп и 25 000 диодов. Имеются сообщения о создании в США электронной цифровой машины *ЛАРК,* имеющей скорость приблизительно 120 000 сложений и 60 000 умножений в секунду. Машина *ЛАРК* построена на полупроводниковых триодах и диодах, имеет оперативное запоминающее устройство на магнитных сердечниках емкостью 25 000 чисел со временем обращения 4 *мксек* и запоминающее устройство на магнитном барабане емкостью 3 *млн.* кодов. В машине *ЛАРК* используется представление чисел как с фиксированной запятой, так и с плавающей запятой в двоично-десятичной системе счисления; машина оперирует с числами, имеющими 11 десятичных разрядов и один знаковый.

Наибольший интерес с точки зрения перспектив развития электронной цифровой техники представляет разработка в США сверхбыстродействующей машины *СТРЕТЧ,* которая должна выполнять 2 млн. сложений в секунду или 500 тысяч умножений в секунду. Эта машина проектируется как комплекс из трех крупных агрегатов, каждый из которых будет представлять собой, по существу, целую машину. Первый агрегат будет включать в себя систему ввода и вывода данных, объединенную с внешним запоминающим устройством на магнитных лентах. Второй агрегат представляет собой электронную цифровую машину последовательного действия, предназначенную для выполнения предварительных вычислений. Эта машина является промежуточной между системой внешних устройств и третьим агрегатом — центральной машиной параллельного действия, выполняющей основной объем вычислений.

Машина *СТРЕТЧ* будет приспособлена к работе с числами, соответствующими 12—15 десятичным разрядам как с фиксированной, так и с плавающей запятой. Скорость промежуточной машины будет равна 300—500 тысяч сложений в секунду или 100 000 умножений в секунду. Основным назначением промежуточной машины будет являться подготовка команд программы для центральной машины путем расшифровки и обработки условной математической и логической формулировок задачи и схемы программы. Проектная скорость работы центральной машины характеризуется следующими данными: сложение и вычитание с фиксированной запятой—0,2 *мксек*;

сложение и вычитание с плавающей запятой — 0,6 *мксек*; умножение — 1,2 *мксек*; передача числа — 0,2 *мксек.* Представляет интерес применение в машине *СТРЕТЧ* различных видов запоминающих устройств. В центральной машине предусматривается применение сверхбыстродействующей оперативной памяти на специальных магнитных сердечниках емкостью 512 чисел со временем обращения 0,5 *мксек.* Основное оперативное запоминающее устройство на магнитных сердечниках емкостью 8192 числа и со временем обращения 2 *мксек* будет построено по блочному принципу (т. е. будет допускать возможность наращивания).

Новым видом памяти является так называемая промежуточная память на магнитных дисках большой емкости (до 1 миллиона чисел) со временем обращения 4 *мксек.* Конструкция такого устройства разработана фирмой ИБМ еще в 1955 г. под маркой *ИБМ*-305. Это устройство имеет 50 магнитных дисков диаметром около 50 *см,* закрепленных на общей вертикальной оси. Запись чисел производится по концентрическим дорожкам с обеих сторон дисков, причем на каждой стороне имеется 10—15 дорожек. Магнитные головки для записи и считывания данных размещаются сбоку дисков и могут перемещаться в зависимости от выбранной дорожки и стороны диска.

В машинах *СТРЕТЧ* предполагается широкое применение специальных высокочастотных полупроводниковых диодов и триодов, которые будут работать на основной частоте 10 *мгц.* Для повышения надежности предусматривается система автоматического контроля правильности работы, а в некоторых местах даже схемы для автоматического обнаружения и исправления ошибок.

Однако постройка машин, обладающих столь высокими скоростями работы, связана со значительным увеличением сложности конструкции и огромным составом оборудования. В связи с этим наряду с созданием больших уникальных машин проводятся эксперименты по применению двух и больше машин для одновременной работы по решению на них одной сложной задачи. Путь комплексирования нескольких обычных машин для их совместной работы по решению задач представляется достаточно эффективным, если требуется обеспечивать выполнение особо громоздких вычислительных работ.

В настоящее время также явно определилась необходимость иметь в большом количестве машины средних размеров и достаточно высокой производительности, предназначенные для обслуживания различных научно-исследовательских учреждений, конструкторских бюро и т. п. Для этих машин основными требованиями являются простота конструкции, уменьшение оборудования и удобство эксплуатации. К машинам среднего класса относятся рассмотренные нами в главе III советские машины *БЭСМ* и *Стрела.*

Характерным примером машин среднего класса является также машина *Пегас* английской фирмы Ферранти. Это универсальная машина, использующая двухадресный код, обозначающий либо одно число, либо две команды. Ее арифметическое устройство имеет несколько регистров, построенных на твердых (никелевых) линиях задержки, в каждой из которых может циркулировать один код, имеющий 39 двоичных разрядов. Семь таких регистров служат для выполнения любых арифметических и логических операций, 32 регистра предназначены только для сложения и вычитания, отдельные регистры предназначены для ввода и вывода информации. Основным запоминающим устройством машины *Пегас* является магнитный барабан, вращающийся со скоростью 4000 *об/мин,* имеющий емкость 5120 ячеек по 42 разряда; 1024 ячейки запоминающего устройства отводятся для хранения программы ввода данных, стандартных подпрограмм и подпрограмм контроля работы машины.

Для упрощения дешифрирования адресов весь объем памяти делится на 40 дорожек по 128 ячеек. Каждая дорожка содержит 16 блоков по 8 ячеек. Магнитный барабан изготовлен из алюминиевого сплава, имеет длину 250 *мм* и диаметр 156 *мм.* Скорость работы машины характеризуется выполнением приблизительно 2000 операций в секунду.

Машина *Пегас* имеет весьма развитую систему команд и разнообразный состав операций, что значительно облегчает работу программистов. Передача данных из регистров арифметического устройства в память и обратная передача могут совершаться одновременно с выполнением арифметических операций. В машине предусмотрены шесть операций условного перехода по заданному адресу, а также две операции сравнения. Особенностью машины является возможность преобразования первого адреса команды перед ее выполнением. Для этого в коде команды, кроме двух адресов и кода операций, имеется несколько разрядов, указывающих адрес числа, которое будет прибавляться к первому адресу команды перед ее выполнением.

Машина *Пегас,* обладая значительной емкостью памяти и сравнительно высокой скоростью работы, невелика по размерам и имеет значительно меньше оборудования, чем другие машины, обладающие аналогичными параметрами (например, *УНИВАК*). Машина имеет блочную конструкцию и состоит (кроме пульта управления) из двух шкафов с размерами 2,3x0,6x2,1 *м* и 1,6x0,6x2,1 *м.*

Характерным примером современной цифровой вычислительной машины универсального назначения с очень малыми размерами является машина фирмы Лайбраскоп. Эта машина оперирует с 30-разрядными двоичными числами. Память емкостью 4096 чисел построена на магнитном барабане, вращающемся со скоростью 3600 *об/мин.* Скорость работы машины составляет около 100 операций в секунду, а скорость ввода и вывода данных — около 600 знаков в минуту. Помимо четырех арифметических действий, машина может выполнять команды условного и безусловного переходов и преобразование адресов. Все устройства машины размещены внутри стального кожуха, в котором находятся также блок питания и установка для кондиционирования воздуха. Машина имеет всего 100 электронных ламп и 1300 диодов и является, по-видимому, наиболее компактной из всех электронных цифровых вычислительных машин универсального назначения, существующих в настоящее время.

К числу малых машин относится также советская машина *Урал,* которая была рассмотрена нами раньше.

**2. Машины для обработки информации.** В последние годы электронные цифровые машины получили широкое применение для обработки информации в экономике, административно-управленческой области, а также в науке и технике, когда возникает необходимость, в обработке огромного количества экспериментальных данных.

Основными особенностями электронных цифровых машин, предназначенных для обработки информации, являются:

1. Огромная емкость внешних запоминающих устройств, построенных на основе использования магнитной ленты. Соотношение между емкостью внешних запоминающих устройств и емкостью оперативной памяти в машинах этого типа составляет величину порядка 1000 и более. По существу, внешнее запоминающее устройство с необходимыми схемами для записи и выборки информации является основным устройством электронной цифровой машины, предназначенной для обработки информации.

2. Сравнительно простая структура арифметического устройства, которое должно выполнять только простые арифметические операции (даже без деления) и ряд простых логических операций, необходимых для осуществления проверок и выборок данных.

3. Наличие большого количества (10—50 комплектов) параллельно и независимо работающих устройств ручного ввода данных. Эти устройства работают длительное время и одновременно с работой машины по выполнению вычислений. Устройства вывода данных в машинах этого типа являются весьма производительными и специфичными; они обеспечивают печатание результатов сразу в форме отчетов, сводок, счетов и т. п., либо в виде графиков.

4. Необходимость контроля отдельных элементарных действий, т. е. применение так называемого *микроконтроля.* С этой целью в состав слов, представляющих информацию, вводятся специальные дополнительные разряды, позволяющие автоматически проверять правильность записи и считывания чисел, их передачу между устройствами, а также выполнение отдельных операций. Методы микроконтроля в машинах этого типа часто сочетаются с методами логического контроля, осуществляемого путем проверки результатов обработки достаточно крупных порций информации.

5. Непрерывный процесс обработки информации и систематическое поступление новых данных, сочетающееся с периодической выдачей результатов вычислений.

В отличие от *последовательно-циклических* задач, какими являются математические вычислительные задачи, задачи по обработке информации являются *параллельно-циклическими.* Это значит, что весь процесс обработки информации состоит из повторения огромного числа сравнительно простых однотипных последовательностей операций над различными исходными данными.

Характерным примером системы для обработки информации является система *БИЗМАК.*

*БИЗМАК* представляет собой сложный комплекс различных машин и устройств, насчитывающий в своем составе до 200 отдельных агрегатов. Эти машины и устройства связаны между собой линиями передачи информации и управляющих сигналов и имеют централизованную систему управления. *БИЗМАК* используется (с 1956 г.) в системе материально-технического снабжения мотомеханизированных войск США для автоматической обработки информации. Назначение системы *БИЗМАК* состоит в ведении учета и организации снабжения запасными частями и деталями, насчитывающими до 250 000 различных наименований. Отдельные поступающие сообщения содержат сведения о наличии деталей на различных складах, о перевозках, о получении продукции от фирм, о необходимых уровнях запасов и т. д., а также специальные коды, характеризующие каждую деталь в отдельности (ее шифр, новая деталь или использованная, габариты, вес, ограничения на перевозку и т. д.). Среднее количество знаков в одном сообщении, относящемся к одной детали, составляет около 300; во многих случаях оно достигает 1500. Ежедневно обрабатывается до 65 000 документов, содержащих около 1 300 000 сообщений. Таким образом, система выполняет огромный объем канцелярской работы, связанной с записью и хранением данных, их обработкой, выполнением вычислений и выработкой логических решений. В отличие от ранее известных средств механизации учета (счетно-клавишные, бухгалтерские, фактурные и т. п. машины), при которых механизировался только сам процесс счета, в данной системе осуществлена комплексная механизация всей работы, включая подготовительную работу и работу по анализу и сортировке данных. Такой подход позволяет повысить эффективность обработки информации и высвободить большое количество служащих.

В систему *БИЗМАК* входят следующие основные устройства:

1. Внешние устройства для подготовки данных к вводу в систему; сюда относятся клавишные устройства (перфораторы) для записи данных на перфоленты и перфокарты. Запись (перфорирование) осуществляется вручную операторами.

2. Устройства для переписи данных с перфолент и перфокарт на магнитные ленты основного запоминающего устройства. Перепись с перфолент осуществляется без какой-либо предварительной обработки; при переписи с перфокарт одновременно производится предварительная сортировка данных и их перестановка в нужном порядке.

3. Основное запоминающее устройство на магнитных лентах; это устройство представляет собой картотеку из большого количества блоков магнитных лент с лентопротяжными механизмами, схемами записи и считывания и схемами управления. Схемы управления обеспечивают выбор и подключение нужного блока магнитной ленты и поиск на ленте требуемых зон. В системе *БИЗМАК* используется магнитная лента шириной 16 *мм;* на ленту

записываются данные по 14 дорожкам с плотностью 50 знаков на 1 *см;* скорость перемещения ленты составляет 2 *м/сек,* что обеспечивает возможность считывания и передачи до 10 000 знаков в секунду. Для обеспечения надежности данных и осуществления контроля правильности применяется способ двойной записи. Четырнадцать дорожек разделены на две группы по семь дорожек в каждой группе; запись данных осуществляется одновременно в две группы дорожек со смещением по длине ленты на 3 *мм.* Смещение введено для того, чтобы исключить возможность одновременных искажений информации в обеих группах, обусловленных местными дефектами материала ленты. Отдельные ленты объединяются в группы, образующие так называемые ленточные установки, которые могут подключаться при помощи магистралей к другим устройствам системы.

4. Электронная цифровая машина, входящая в состав системы *БИЗМАК,* представляет собой программно-управляемую машину с трехадресной системой команд, работающую в двоичной системе счисления. Скорость вычислений около 25 000 сложений в секунду; машина имеет оперативное запоминающее устройство на магнитных сердечниках и отдельное запоминающее устройство для хранения программы. Особенностью машины являются ее способность обрабатывать не только цифровую, но и буквенную информацию и способность подключать к себе в качестве входа до 5 магнитных лент и в качестве выхода до 10 магнитных лент.

5. В состав *БИЗМАК* входят три электронных устройства для сортировки информации, записанной на магнитных лентах. Каждое такое устройство представляет собой электронную цифровую машину с жесткой коммутируемой программой; система команд состоит всего из 13 различных операций, включающих в себя сравнение, замещение, выделение, объединение и стирание данных. К каждому сортирующему устройству может подключаться от двух до шести магнитных лент. Сортировка данных составляет основной объем работы системы и превышает в три раза объем собственно вычислительной работы, выполняемой системой.

6. Запрашивающее устройство, предназначенное для выборочного поиска данных из обработанной информации. Основной процесс работы системы представляет собой, по существу, непрерывный процесс обработки новых объемов информации, поступающей периодически. Однако наряду с этим часто возникает необходимость (например, для справок) обратиться к старому материалу. Для этой цели в составе системы и предусмотрено запрашивающее устройство. При помощи этого устройства оператор может проверить содержимое любой ленты и выдать на печать данные, записанные по определенному адресу зоны, либо данные, обладающие определенными признаками.

7. *БИЗМАК* имеет выходные устройства двух типов:

1) электромеханические печатающие устройства, выдающие данные в виде таблиц в трех экземплярах со скоростью 600 строк в минуту (по 120 знаков в строке);

2) устройства для перевода данных с магнитных лент на перфоленты, с которых данные на специальных внешних устройствах печатаются в виде документов.

8. Центральная управляющая станция осуществляет управление согласованной работой всей аппаратуры системы. Она подключает блоки магнитных лент к различным устройствам системы (входным, сортирующим, к вычислительной машине, выходным и другим устройствам), осуществляет переключение связей между устройствами и управляет пуском и остановом отдельных машин и устройств системы. Эти действия центральная станция может осуществлять как автоматически (в соответствии с заранее установленной в ней программой), так и не автоматически (с помощьюоператора системы).

Кроме описанных устройств, в составе системы имеется еще ряд специальных пультов операторов, позволяющих контролировать перфорацию на перфоленты, перфокарты, управлять работой отдельных устройств и машин системы.

Одной из особенностей системы *БИЗМАК* является применение переменной длины разрядной сетки (длины кодированного двоичного слова) для кодирования данных, что позволяет более рационально использовать емкость запоминающих устройств на магнитных лентах. Система *БИЗМАК* отражает в себе основные характерные черты электронных цифровых систем, предназначенных для обработки информации. В настоящее время в подобных системах значительную роль играют люди-операторы, участвующие во вводе и выводе данных, осуществляющие управление переключением устройств, поиском и запросом данных и контролирующие правильность функционирования системы. Дальнейшее развитие электронных цифровых систем, предназначенных для обработки информации, помимо технических усовершенствований, идет в направлениях: а) усложнения и расширения логических функций, выполняемых системой, б) уменьшения степени участия людей в управлении работой системы, в) введения автоматизированных связей с непосредственными источниками и получателями информации и автоматических устройств ввода и вывода данных, г) разработки более эффективных средств и методов для непосредственной оперативной связи людей-операторов с машиной в процессе анализа и обобщения информации.

**3. Машины для автоматического управления.** Если в первые годы для целей автоматического управления реальными объектами применялись электронные цифровые машины универсального назначения (например, *ИРА* 1103, США), то в последние годы созданы машины, специально приспособленные для этой цели. Эти машины выделяются в особый, очень важный класс машин, обладающих рядом характерных свойств. К числу таких свойств относятся:

1. Наличие гибкой и разветвленной системы внешних устройств, обеспечивающих непосредственную оперативную связь машины с источниками информации и управляемыми объектами.

2. Приспособленность системы управления машины к работе в реальном масштабе времени; это значит, что машина должна постоянно согласовывать ход вычислительного процесса с характером поступления внешних данных, приходящих в основном независимо от работы машины, и с ходом процесса регулирования управляемого объекта. В частности, машины этого класса, как правило, обладают способностью прерывать ход вычислений на время приема новых порций информации.

3. Высокое быстродействие, обусловленное необходимостью производить обработку информации и вычисления в реальном масштабе времени, не допускающем длительных задержек в выдаче результатов вычислений после поступления исходных данных.

4. Большая емкость оперативных запоминающих устройств и почти полное отсутствие внешних запоминающих устройств на магнитных лентах. В некоторых системах магнитные ленты используются, но не в процессе непосредственной работы машины по обработке информации, а для документирования хода процессов управления. В машинах этого класса широко используются промежуточные запоминающие устройства на магнитных барабанах и часто специальные запоминающие устройства для хранения программ.

5. Большая гибкость и сложность системы команд, включающей в себя, помимо обычных арифметических и логических операций, ряд команд для связи с внешними объектами и ряд команд для специальных проверок поступающей информации.

6. Наличие специальных устройств для оперативной связи человека-оператора с машиной в процессе работы. В частности, имеются системы наглядного отображения результатов решений и специальные пульты операторов для ввода команд в машину в процессе работы.

К машинам этого класса предъявляются чрезвычайно высокие требования по надежности и безотказности в работе, которые обеспечиваются всеми известными способами повышения надежности, включая технические, схемные способы, логический контроль и исправление ошибок и даже дублирование и страивание машин и устройств.

Характерным примером машины, предназначенной для автоматического управления реальными объектами, является электронная цифровая машина *АНФСК-7,* (*ANFSQ-7*), созданная в 1956 г. в США для автоматической обработки информации о воздушном противнике и управления средствами противовоздушной обороны. Эта машина представляет собой основную часть сложной системы, включающей в себя входное устройство, выходное устройство, устройство отображения данных и ряд других вспомогательных устройств. Для обеспечения надежности работы в системе используются две одинаковые, параллельно работающие электронные цифровые машины. Входные и выходные устройства системы сами по себе являются многоканальными и не требуют специального дублирования.

Входное устройство представляет собой промежуточное или буферное запоминающее устройство на магнитных барабанах, связывающих источники информации с электронной цифровой машиной. В связи с тем, что поступление информации является нерегулярным и не зависит от работы машины, в буферных магнитных барабанах предусмотрены две независимые системы записи и считывания данных; первая связана с источниками информации, а вторая — с оперативным запоминающим устройством машины.

В качестве выводных устройств также используются магнитные барабаны, которые служат для согласования темпа выдачи данных из машины с пропускной способностью линий связи.

Система отображения данных построена на основе использования специальных электронно-лучевых трубок со знаковой индикацией типа *Характрон.* На экранах этих трубок изображаются в наглядной форме данные о воздушной обстановке, вырабатываемые машиной. В качестве устройств для документирования процессов управления и обработки информации используются запоминающие блоки на магнитных лентах.

Собственно электронная цифровая машина представляет собой машину параллельного действия с одноадресной системой команд, работающую в двоичной системе счисления. В машине предусмотрено выполнение сложной системы арифметических и логических операций.

В качестве оперативных запоминающих устройств используются два блока памяти на магнитных сердечниках емкостью 4096 чисел по 32 разряда со временем обращения 6 *мксек.* Быстродействие машины характеризуется выполнением сложения за 12 *мксек,* умножения за 15,5 *мксек* и деления за 53 *мксек.*

Особенностью системы управления машины является наличие режима прерывания вычислений по основной программе на время переписи данных из буферной памяти в оперативную. В машине имеется, помимо оперативного запоминающего устройства, также промежуточное запоминающее устройство на магнитных барабанах, включающее восемь барабанов общей емкостью 3 244 032 двоичных числа. Ряд специальных пультов управления, предусмотренных в машине, позволяет операторам вмешиваться в работу машины, вводить новые команды в программу, запрашивать требуемые данные и визуально контролировать работу отдельных устройств машины.

Конструирование машины производилось с учетом повышенных требований к надежности работы, что приводило в ряде случаев к увеличению состава аппаратуры. Во всей системе, включающей два комплекта электронной цифровой машины и все вспомогательные устройства, насчитывается около 25 000 электронных ламп.

В настоящее время построение управляющих электронных цифровых машин идет в основном по пути применения новых полупроводниковых и ферритовых элементов, обладающих длительным сроком службы и высокой надежностью.

## § 24. Разработка новых принципов построения и усовершенствование конструкции электронных цифровых машин

**1. Повышение быстродействия машин.** Существуют два пути повышения быстродействия электронных цифровых машин: *технический* и *структурный.*

Технический путь заключается в отработке и применении элементов, схем и устройств машин, допускающих работу при более высоких частотах и коротких импульсах. Существенное повышение техническим путем быстродействия основных устройств электронной цифровой машины требует разработки новых элементов (ламповых, полупроводниковых, магнитных) и связано с переходом на миллимикросекундную технику.

Структурный путь повышения быстродействия электронной цифровой машины заключается в разработке принципов организации работы машины, построения ее структуры, системы программирования, в выборе рационального алгорифма задачи, в расчленении всей задачи на отдельные части, которые могут решаться параллельно и т. д. Рассмотрим некоторые структурные пути повышения быстродействия электронных цифровых машин.

Совмещение операций. Как показывают многочисленные сообщения в печати, существенное повышение быстродействия электронных цифровых машин достигается во многих случаях за счет совмещения во времени различных операций, выполняемых отдельными частями машины. Например, часто применяется совмещение во времени операций вызова команд и вызова чисел из запоминающих устройств, а также совмещение во времени процессов вызова команд и выполнения арифметических действий. Широкое применение получает принцип совмещения во времени операций, выполняемых вводными и выводными устройствами, с непосредственной работой машины по выполнению вычислений. Возможно также совмещение во времени процессов обмена информацией между внешним накопителем и оперативным запоминающим устройством с процессом вычислений.

За счет использования индексных регистров, которые применяются, в частности, в упомянутой уже машине *АНФСК-7,* осуществляется совмещение во времени операций преобразования адресов команд и операций над числами.

Запоминающие устройства с независимыми системами обращения. Существенное повышение быстродействия электронных цифровых машин может быть достигнуто путем перехода к многоадресным системам команд за счет совмещения во времени операций вызова команд и вызова и записи нескольких чисел в оперативные запоминающие устройства. Для этого необходимы оперативные запоминающие устройства, допускающие одновременно несколько независимых обращений к различным или одним и тем же ячейкам.

При обычном типе оперативного запоминающего устройства *(ОЗУ),* допускающем обращение только в одну ячейку, наиболее целесообразной системой адресности является одноадресная. При этой системе адресности получается не меньшая скорость вычислений, чем при трехадресной (даже может быть большая), так как среднее число обращений к памяти на каждую полную операцию равно четырем в обоих случаях и выполняются они поочередно. В то же время одноадресная система проще, чем трехадресная, с точки зрения ее реализации в электронной цифровой машине.

Однако при наличии, например, нескольких отдельных *ОЗУ* и *АУ,* работающих одновременно и независимо, существенное повышение быстродействия может быть достигнуто при применении многоадресной системы команд, так как при этом каждая сложная команда может выполняться за время одного такта.

Построение *ОЗУ* с тремя независимыми системами обращения возможно путем изменения структуры обычного *ОЗУ* и его системы управления. Например, *ОЗУ* разделяется на три блока, соответствующих трем адресам в командах, и предусматриваются три независимые системы обращения к *ОЗУ,* работающие одновременно (две системы на считывание, одна — на запись). По существу это сводится к применению в машине нескольких независимых оперативных запоминающих устройств, которые могут работать одновременно. *ОЗУ* нескольких различных типов, допускающие одновременные и независимые обращения, предусматриваются, например, в машине *СТРЕТЧ.*

Принцип ступенчатого и параллельного решения. В уникальных машинах универсального назначения, предназначенных для решения весьма сложных задач, может* применяться так называемая *ступенчатая* организация вычислительного процесса.

При этом вычислительная машина, по существу, представляет собой комплекс машин, выполняющих различные стадии вычислений. Первая машина является подготовительной и служит для подготовки задачи для ее решения последующими машинами. В функции подготовительной машины может входить, например, автоматическое составление программы решения задачи на основе достаточно кратких исходных данных, задаваемых в виде алгебраических формул или логических зависимостей. Вторая машина может использоваться для предварительной подготовки исходных данных и выполнения начальных этапов вычислительного процесса. Следующие машины должны выполнять основной объем вычислений по программам, составленным подготовительной машиной и с использованием данных первой машины. Известны опыты по решению одной задачи на двух машинах *СЕАК* и *ДИСЕАК,* которые работали по своим программам и осуществляли обмен информацией в процессе вычислений.

Более совершенный способ организации работы двух машин, из которых первая является подготовительной, а

вторая — основной, применен в разрабатываемой в США быстродействующей машине *СТРЕТЧ,* которая была рассмотрена нами выше.

Дальнейшее повышение скорости вычислений может быть достигнуто путем расчленения задач (алгорифмов их решений) на отдельные независимые задачи, которые могут решаться одновременно *(параллельно).* Анализ возможности такого расчленения процесса решения задач необходимо проводить, исходя из имеющегося времени на решение всей задачи, с учетом достижимого быстродействия машины и ее отдельных частей. Естественно, что расчленение процесса решения задач ведет к увеличению количества машин, и поэтому целесообразно искать решение в первую очередь на пути максимального повышения быстродействия одной машины.

**2. Применение индексных регистров.** В последние годы во многих машинах в систему центрального управления вводят так называемые *индексные регистры,* которые служат для автоматического изменения адресов команд без использования арифметического устройства.

В коде команд предусмотрены специальные разряды для индексов, показывающие необходимость модификации адресной части данной команды и указывающие тот индексный регистр, который необходим для этой модификации. Перед выполнением данной команды к ее адресной части автоматически прибавляется содержимое указанного индексного регистра и в таком виде команда выдается для выполнения. После этого число, находящееся в индексном регистре, автоматически изменяется на заданную величину (обычно на единицу). Конструкция индексных регистров и предусматривает возможность посылки в эти регистры различных кодов, которые будут определять собой процессы переадресации команд. Для осуществления указанных посылок в системе команд предусматриваются специальные команды.

Способ индексных регистров обладает следующими достоинствами:

а) обеспечивает повышение быстродействия, так как не затрачивается специально время на модификацию команд;

б) обеспечивается возможность автоматического контроля числа повторений циклов в программах по изменению содержимого индексных регистров;

в) исключается необходимость восстановления измененных команд, так как команды всегда хранятся в запоминающем устройстве в своем первоначальном виде.

**3. Применение принципа микропрограммного управления и повышение гибкости структуры машин.** Как известно из предыдущего, обычная схема построения и организации работы электронных цифровых машин с программным управлением предусматривает последовательное выполнение команд программы, т. е. работу машины по тактам. *Такт* — это время, необходимое для выполнения одной команды. Кроме того, под *тактом* следует понимать также совокупность действий, совершаемых в машине в процессе выполнения одной команды. Таким образом, понятие такта относится к работе машины в целом.

Например, типовой такт одноадресной машины состоит из следующих типовых этапов, выполняемых отдельными устройствами машины [*)]:

а) выборка команды;

б) выборка числа;

в) выполнение арифметической операции.

Каждый этап такта состоит в свою очередь из ряда действий, выполняемых одновременно или последовательно во времени отдельными функциональными блоками, входящими в состав устройства. *Функциональный блок* — это часть аппаратуры, выполняющая постоянно одну и ту же функцию — действие под влиянием одного управляющего сигнала. Хотя понятие блока является функциональным, оно обычно соответствует физическому представлению автономного блока. Однако не исключаются случаи, когда отдельные детали будут являться общими для нескольких различных функциональных блоков. *Действие* функционального блока представляет собой физический процесс в блоке, начало которого задается одним сигналом, являющимся внешним по отношению к этому блоку, и ход которого определяется уже только свойствами самой схемы блока, выполняющего это действие (наличием и характером переходных процессов, длиной цепочек отдельных элементов—разрядов и т. д.).

Так, этап выборки команды состоит из следующих действий:

1) увеличения на единицу кода, находящегося в счетчике адресов команд;
2) гашения «в нуль» регистра адреса команды;
3) приема кода адреса команды на регистр адреса;
4) работы дешифратора адреса команды;
5) непосредственной выборки команды и передачи ее на регистр команд.

Этап выборки числа из оперативного запоминающего устройства (*ОЗУ*) состоит из элементарных действий:

1) гашения «в нуль» регистра кода адреса *ОЗУ;*
2) передачи кода адреса с регистра команд на регистр кода адреса *ОЗУ;*
3) работы дешифратора адреса *ОЗУ;*

[*)] Заметим, что в современных машинах нет резкого разграничения функций между устройствами при выполнении этапов; хотя каждый этап выполняется преимущественно одним устройством, другие устройства тоже принимают в этом участие. Например, если в машине нет специального накопителя программ, то в выполнении этапа вызова команды участвует, помимо устройства управления, и оперативное запоминающее устройство.

4) непосредственной выборки числа и выдачи числа в кодовую шину числа (*КШЧ*);

5) регенерации (при необходимости).

Этап выполнения операций арифметическим устройством (*АУ*) состоит из следующих действий:

1) приема кода числа из *КШЧ* на регистр *АУ;*

2) непосредственного выполнения операции сумматором;

3) обращения кода результата (при необходимости).

В зависимости от выполняемых функций блоки могут быть разделены на три группы:

1) участвующие в вычислениях, т. е. в операциях над информацией;

2) участвующие в управлении ходом вычислительного процесса, т. е. оперирующие с кодами команд и управляющими сигналами;

3) участвующие как в вычислительном процессе, так и в управлении этим процессом, т. е. оперирующие как с кодами чисел, так и с кодами команд.

Любая машина полностью характеризуется составом своих функциональных блоков и системой выполняемых ими действий. Путем изменения соединений функциональных блоков и последовательностей выполняемых ими действий можно в достаточно широких пределах варьировать систему математических команд в данной машине. При проектировании электронных цифровых машин необходимо производить анализ не только системы математических команд, но и системы элементарных действий и состава функциональных блоков машины, что позволит более рационально строить машины.

Назначением устройства центрального управления в любой электронной цифровой машине является, по существу, выработка необходимых последовательностей управляющих сигналов, которые включают в работу различные функциональные блоки машины, в том числе и блоки самого устройства управления. Отличия в построении устройств управления различных машин заключаются в разных способах выработки этих последовательностей сигналов и в назначении самих сигналов. В некоторых случаях отдельные устройства машины (арифметические, запоминающие и др.) имеют свои местные устройства управления, и тогда управляющие сигналы от центрального устройства управления поступают более редко и воздействуют не на отдельные функциональные блоки, а на целые устройства машины. В других случаях центральное устройство управления вырабатывает более полную последовательность управляющих сигналов и управляет работой отдельных блоков и даже элементарных схем.

При обычном способе построения устройства центрального управления машины выработка последовательностей управляющих сигналов осуществляется схемным путем, т. е. имеется генератор основных импульсов и ряд логических схем, образованных из элементарных схем «и», «или» и «нет», которые в зависимости от кода операции выполняемой команды выделяют из серии стандартных импульсов необходимые сочетания последовательных сигналов и направляют их в соответствующие блоки и устройства.

В последние годы начал разрабатываться и применяться другой способ выработки последовательностей управляющих сигналов, получивший название *микропрограммирования.* При этом способе заранее определяются все последовательности управляющих сигналов, необходимые для выполнения всех машинных операций. Каждая последовательность сигналов, относящаяся к определенной машинной операции, образует своего рода программу выполнения машиной отдельной команды, так называемую *микропрограмму,* а каждый сигнал, входящий в эту микропрограмму, является *микрокомандой.* Эти последовательности сигналов — микропрограммы — записываются в специальное запоминающее устройство.

Работа машины с микропрограммным управлением, так же как и обычной машины, заключается в последовательном выполнении команд программы. После вызова для исполнения очередной команды устройство управления по коду операции данной команды находит в специальной памяти соответствующую микропрограмму (точнее, ее первую микрокоманду) и выдает последовательно все микрокоманды, входящие в состав этой микропрограммы. Сигналы от каждой микрокоманды управляют соответствующими функциональными блоками машины и обеспечивают выполнение требуемых действий, а последовательность микрокоманд обеспечивает выполнение заданной команды. После выполнения всей микропрограммы происходит вызов следующей команды основной программы и затем вызов и выполнение соответствующей микропрограммы и т. д. При выполнении микропрограмм предусматривается возможность условных переходов, что позволяет изменять ход выполнения отдельных операций и команд программы даже в процессе их выполнения. В частности, это может обеспечить повышение быстродействия за счет прерывания тактов по мере выполнения операций. Система микропрограммного управления обеспечивает значительную гибкость структуры машины и возможность изменения системы команд и состава машинных операций в зависимости от особенностей решаемых задач и условий применения машины.

Для машин с микропрограммным управлением необходимо, помимо оперативного запоминающего устройства, еще специальное запоминающее устройство для микропрограмм сравнительно небольшой емкости, но значительно более быстродействующее, чем основное оперативное запоминающее устройство. Это обусловлено тем, что в процессе работы на каждое обращение к основному оперативному запоминающему устройству приходится несколько обращений к памяти, содержащей микропрограммы. В каждой ячейке (строке) этой памяти записывается одна микрокоманда и все микрокоманды имеют одинаковое число разрядов. Каждый разряд в коде микрокоманды соответствует определенному функциональному блоку; наличие единицы в этом разряде означает,

что данный блок должен быть включен в работу, наличие нуля в этом разряде означает, что данный блок действовать не должен. Таким образом, общее число разрядов в коде микрокоманды равно общему количеству различных функциональных блоков *(управляемых точек)* в машине.

Могут применяться два способа задания порядка следования микрокоманд в микропрограммах. При первом *(жестком)* способе микрокоманды каждой микропрограммы записываются подряд за первой микрокомандой и последняя микрокоманда выдает сигнал об окончании данной микропрограммы. При втором *(плавающем)* способе в каждой микрокоманде содержится также код, указывающий следующую микрокоманду. Достоинством последнего способа является возможность использования одних и тех же микрокоманд в различных микропрограммах.

С помощью микропрограммного управления обеспечивается возможность сравнительно простой реализации различных сложных операций, таких, как извлечение корня, вычисление тригонометрических функций и т. п., которые обычно получаются с помощью достаточно сложных подпрограмм с затратой большого машинного времени. Кроме того, микропрограммирование позволяет более просто и стройно с логической точки зрения построить систему управления электронной цифровой машины.

Недостатками принципа микропрограммного управления являются трудность получения высокого быстродействия машины, так как это требует резкого повышения скорости работы микропрограммной памяти, и наличие большого количества связей устройства центрального управления машины с остальными устройствами.

**4. Усовершенствование конструкции и технологии производства машин.** Наряду с исследованиями принципов построения электронных цифровых машин широкие работы ведутся в области усовершенствования конструкции этих машин и разработки технологии их массового изготовления. Одним из важных мероприятий в этой области является применение метода *печатного монтажа*.

В отличие от обычных схем, в которых радиодетали соединяются между собой монтажными проводами, в печатных схемах соединение отдельных деталей, укрепленных на изолирующей плате, осуществляется посредством покрытия дорожек, заменяющих провода, тонким слоем проводящего вещества (распылением через трафарет, штамповкой и др.). В последнее время способом печати изготовляются не только монтажные провода, но и значительная часть радиодеталей, в том числе сопротивления, конденсаторы небольшой емкости, катушки индуктивности, потенциометры, переключатели. Производство аппаратуры методом печати позволяет получить высокое качество всех электрических соединений и контактов, благодаря чему такая аппаратура обладает большой надежностью и долговечностью при малых габаритах и весе.

Основным достоинством печатных схем по сравнению **с** обычными является возможность автоматизации процесса производства, что облегчает и ускоряет выпуск продукции и намного уменьшает ее стоимость. Возможность массового выпуска продукции посредством автоматизации производства имеет особенно важное оборонное значение.

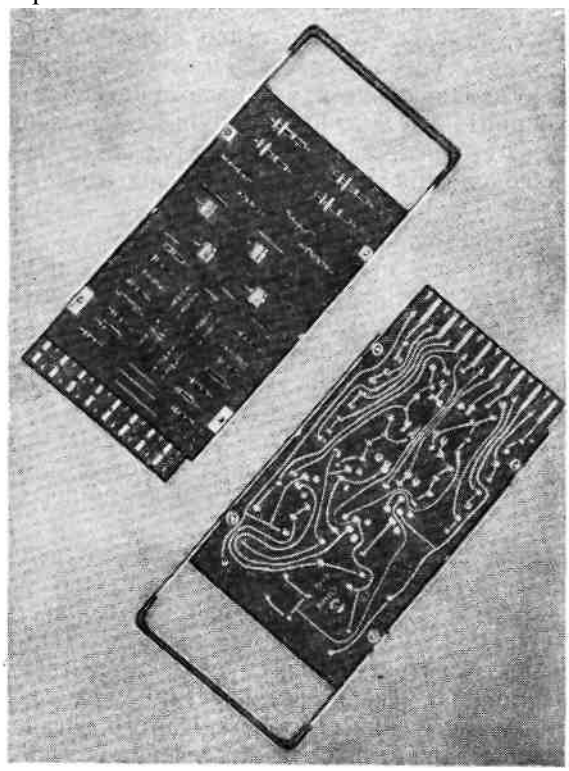


Рис. 90. Стандартные блоки, изготовленные способом печати.

Помимо исследований, направленных на разработку технологии изготовления печатных схем, ведутся также работы по проектированию и строительству автоматических линий для сборки электронного оборудования, по расширению сфер применения печатных схем и т. д. Изготовление способом печати стандартных блоков позволяет быстро собирать из них сложные вычислительные машины (рис. 90).

Большое развитие в настоящее время получила тенденция агрегатного построения машин. При этом отдельные устройства машины (запоминающие, арифметическое и т. д.) конструктивно оформляются в виде отдельных шкафов, включающих в себя также все необходимые управляющие и вспомогательные схемы. Из таких шкафов может собираться машина, причем в зависимости от потребностей в состав машины включается большее или меньшее количество типовых агрегатов.

Широкое распространение получило также производство стандартных блоков электронных цифровых машин.

Каждая машина состоит из ограниченного числа различных по своему функциональному назначению узлов и элементарных устройств, соединенных в общую схему. Это позволяет по различным схемам осуществлять сборку машин из стандартных блоков, включающих в себя один или несколько элементарных устройств, и стандартных узлов. Блочный способ значительно упрощает производство машин, вследствие чего резко снижается их стоимость, уменьшаются сроки ввода в действие и открываются широкие возможности для механизации и автоматизации производства. Создаются благоприятные условия для

массового производства машин в разных местах страны. Стандартизация блоков и узлов исключает возможность появления ошибок в схеме при монтаже и в значительной мере облегчает эксплуатацию машин, позволяя осуществлять ее менее квалифицированным персоналом.

Среди выпускаемых стандартных блоков имеются: усилители, триггеры, мультивибраторы, формирователи прямоугольных импульсов, линии задержки, генераторы, схемы совпадения, собирательные схемы, магнитные барабаны, устройства для ввода и вывода и др., т. е. все элементы и узлы, необходимые для строительства машин.

Применение стандартных блоков для сборки машин и внедрение техники печатных схем создают условия для полной автоматизации процесса изготовления электронных цифровых машин.

**5. Разработка быстродействующих устройств ввода и вывода.** Узким местом в конструкции современных быстродействующих машин являются устройства для ввода и вывода, так как ввод исходного материала в машину и вывод из нее результатов решения осуществляются электромеханическими устройствами, работающими со значительно меньшими скоростями, чем остальные электронные устройства машин. Вследствие этого задачи с большим числом исходных данных и большим числом результатов решать на быстродействующих машинах не всегда экономически выгодно, так как время, необходимое для вывода материала, значительно превысит полезное машинное время решения задачи. Проблема увеличения скорости работы устройств ввода и вывода не решена удовлетворительно до настоящего времени.

Для ввода исходного материала в машину и вывода результатов из нее используются перфокарты, перфоленты с механическим и фотоэлектрическим считыванием, магнитные ленты и магнитная проволока. Вследствие малой скорости работы устройств ввода и вывода в некоторых случаях одновременно используется несколько видов этих устройств, что сокращает непроизводительное время на эти операции, но удорожает стоимость машины. Использование нескольких типов устройств ввода и вывода позволяет эффективно применять машины даже для решения небольших задач массового характера, имеющих значительное количество исходных числовых данных и результатов решения. Такие задачи являются характерными для экономико-статистических и коммерческих расчетов.

Улучшение технических параметров устройств ввода и вывода идет по пути увеличения скорости их работы. В настоящее время разрабатываются сложные устройства для фотографического и электронного ввода и вывода данных, которые бы обеспечили возможность непосредственного чтения буквенного и цифрового печатного текста и ввод его в машину, а также непосредственно выдачу в виде печатных текстов и цифровых таблиц результатов решения задач на машинах.

Например, фотооптическое устройство для ввода данных в вычислительную машину считывает данные с микроснимка документа, на котором эти данные были записаны от руки специальными знаками. Считываемые знаки преобразуются в электрические сигналы оптическим путем и записываются на магнитную ленту. Микропленка передвигается перед объективом. Пленка освещается при помощи электронно-лучевой трубки, на пластины которой подаются развертки по строке и по кадру. За объективом помещен фотоэлемент, который образует электрические импульсы, соответствующие знакам на пленке.

Средняя скорость считывания и ввод информации в машину составляет 2000 двоичных и 250 десятичных знаков в 1 сек. В устройстве предусмотрен автоматический контроль правильности поступающей в машину информации.

Большое развитие в последние годы получили устройства для визуальной выдачи результатов решения задач. Примером такого рода устройств является специальная электронно-лучевая трубка *Характрон*. Эта трубка позволяет получать на экране диаметром 30—50 *см* результаты решения задач и необходимую информацию в наглядной форме в виде светящихся букв, цифр и различных специальных знаков. *Характрон* имеет довольно сложное устройство. В отличие от обычных электронно-лучевых трубок он имеет две отклоняющие и фокусирующие системы, работающие последовательно по длине луча. На пути луча перед этими системами установлена матрица — пластина с отверстиями в виде букв, цифр, стрелок и других знаков. Первая отклоняющая система направляет луч в то место матрицы, где помещен требуемый знак. Пройдя через отверстие соответствующей формы, луч приобретает такую же форму сечения.

Вторая отклоняющая система направляет луч в требуемую точку экрана, где под действием луча высвечивается соответствующий знак.

Большое количество работ ведется по созданию устройств для автоматического чтения печатного буквенного и цифрового текста и непосредственного ввода его в электронную цифровую машину. Построен ряд действующих приборов, которые показали положительные результаты. Принцип действия подобных приборов заключается в следующем. Тонкий луч света пробегает несколько раз справа налево и сверху вниз по увеличенному изображению буквы, осуществляя тем самым ее разложение на элементы. Отражаясь от буквы, луч света попадает на фотоэлементы. В зависимости от формы буквы получается определенное чередование темных и светлых мест и определенная последовательность импульсов тока в фотоэлементах, которая воспринимается машиной как сигнал данной буквы. Подобным образом могут автоматически опознаваться различные знаки и фигуры.

## § 25. Разработка новых элементов

Одним из наиболее важных направлений развития техники электронных цифровых машин является разработка

новых элементов, обладающих повышенными техническими характеристиками (быстродействие, надежность, экономичность, малые габариты, вес и т. д.).

Область изысканий новых элементов, построенных на новых физических и технических принципах, является весьма широкой; она связывает технику электронных цифровых машин с различными областями науки: физикой, химией, радиоэлектроникой и др. Достижения в разработке новых элементов оказывают значительное влияние на принципы построения электронных цифровых машин.

Так, применение полупроводниковых диодов и триодов, магнитных сердечников сильно двинуло вперед технику электронных цифровых машин, отразилось на принципах логического построения схем, расширило области применения машин.

Возможности электронных цифровых машин в значительной степени определяются тем, на каких элементах строятся эти машины.

В печати непрерывно появляются сообщения о новых разработках элементов для цифровых машин. В связи с этим не представляется возможным дать сколько-нибудь полный обзор работ в этой области.

Мы ограничимся кратким описанием наиболее перспективных направлений, которые, на наш взгляд, имеют принципиальное значение для развития техники электронных цифровых машин. К таким направлениям можно отнести в настоящее время разработку новых ферромагнитных элементов, новых полупроводниковых приборов, новых запоминающих элементов.

**1. Ферромагнитные элементы.** Наиболее характерным примером нового типа ферромагнитных приборов являются *транс-флюксоры* — магнитные сердечники **с** управляемым магнитным потоком.

Трансфлюксоры представляют собой магнитные ферритовые сердечники с несколькими отверстиями (от двух до шести и более). Трансфлюксоры имеют несколько обмоток, которые позволяют менять конфигурацию магнитного потока в сердечнике, благодаря чему трансфлюксор имеет не только свойства запоминающего элемента, но и вентильные свойства.

Ценным свойством трансфлюксора является возможность считывания информации без ее нарушения. В обычных ферритовых сердечниках при считывании информации происходит ее стирание, и для того чтобы сохранить в дальнейшем записанную информацию, необходимо ее вновь записать на данный сердечник. В трансфлюксоре при считывании информации автоматически происходит ее перезапись и не требуется специального восстановления.

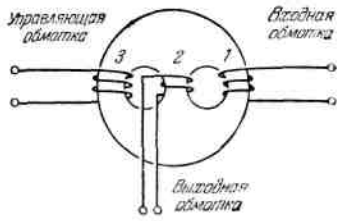


Рис. 91. Схема простейшего трансфлюксора.

С помощью одного сердечника с несколькими отверстиями можно выполнять сложные логические операции; например, на сердечнике с шестью отверстиями можно построить схему одноразрядного двоичного сумматора на два входа (полусумматора), при этом один такой сердечник заменяет в логической схеме до 12 транзисторов.

Принцип действия простейшего трансфлюксора (рис. 91) сводится к следующему. Сердечник имеет два отверстия и три обмотки: управляющую, входную и выходную. Для магнитного потока имеются три пути — три магнитопровода (*/, 2, 3)*. Поперечное сечение магнитопровода *3* должно быть не меньше суммы поперечных сечений магнитопроводов *1* и *2.* Поперечные сечения магнитопроводов *1* и *2* должны быть равны. Трансфлюксор с двумя отверстиями может выполнять функции запоминающего элемента и клапана.

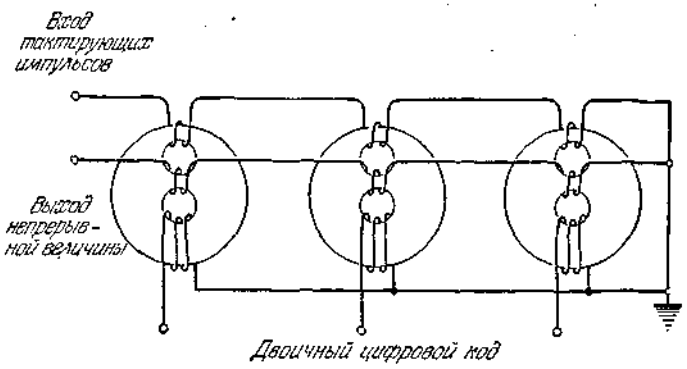


Рис 92 Схема на трансфлюксорах для преобразования информации в непрерывную.

Для того чтобы заблокировать сердечник, т. е. запретить прохождение входного сигнала на выход, необходимо подать в управляющую обмотку импульс тока, достаточный для того, чтобы намагнитить сердечник до насыщения. При этом возбуждение входной обмотки не будет оказывать влияния на магнитный поток, замыкающийся вокруг контура 1—2, и на выходной обмотке сигнала не будет.

Для разблокирования трансфлюксора необходимо в управляющую обмотку подать импульс тока, который бы перемагнитил магнитопровод 2, но не перемагнитил магнитопровод 1. В этом случае при подаче сигнала во входную обмотку происходит изменение магнитного потока вокруг контура *1—2* и в выходной обмотке появляется сигнал формы в непрерывную. Двоичный цифровой код подается поразрядно в виде импульсов тока на управляющие обмотки трансфлюксоров. Под действием сигналов соответствующие сердечники разблокируются. Выходные обмотки трансфлюксоров соединены последовательно, так что выходное напряжение, представляющее собой сумму выходных- напряжений отдельных трансфлюксоров, пропорционально поданному двоичному числу. Перед приемом нового числа на

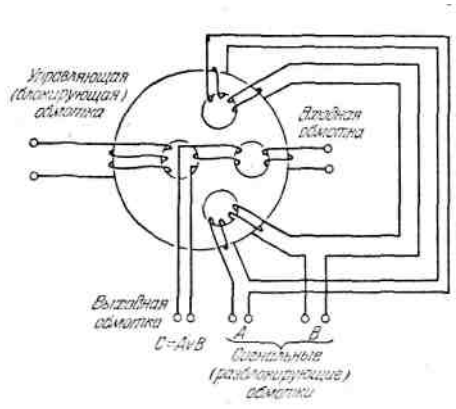



Рис. 93. Трансфлюксор, реализующий дизъюнкцию (*или*).

управляющие обмотки трансфлюксоров предварительно должен быть подан блокирующий сигнал. После подачи двоичного кода сигнал на выходе схемы сохраняется до тех пор, пока не будет подан следующий блокирующий (стирающий) сигнал.

На рис. 93 показан трансфлюксор с четырьмя отверстиями, реализующий логическую операцию *или.*

При подаче соответствующего импульса тока на управляющую обмотку трансфлюксор блокируется, т. е. весь сердечник намагничивается до насыщения в таком направлении, что сигналы, подаваемые на входную обмотку, не оказывают влияния на выход.

Трансфлюксор может быть разблокирован подачей сигналов на вход *A* или на вход *B* или на оба входа *A* и *B* вместе. При этом сигнал, подаваемый на вход, будет проходить на выход трансфлюксора.

Трансфлюксоры с большим числом отверстий и обмоток позволяют осуществлять более сложные логические и счетные операции и могут быть использованы вместо ламп и полупроводниковых триодов при построении переключательных и счетных схем в электронных цифровых машинах.

Достоинствами трансфлюксоров являются простота конструкции, малые габариты, надежность действия, долговечность.

**2. Полупроводниковые приборы.** Создание новых полупроводниковых приборов идет тремя путями. Первый путь заключается в разработке новых способов управления механизмом получения (*p* — *n*)-переходов и их характером в полупроводниковом приборе. Этот путь привел к разработке новых технологических процессов получения (*p* — *n*)-переходов: сплавление, выращивание, диффузия и др., на основе которых в настоящее время получаются наиболее качественные и высокочастотные полупроводниковые диоды и триоды. Сюда же относится изыскание новых полупроводниковых материалов для изготовления приборов.

Второй путь состоит в разработке принципиально новых способов управления (*p* — *n*)-переходом при помощи дополнительных внешних приспособлений. Примером такого рода полупроводниковых приборов является так называемый *спазистор.*

Спазистор представляет собой полупроводниковый прибор, имеющий один электронно-дырочный переход (подобно полупроводниковому диоду). В отличие от полупроводникового диода, в котором электронно-дырочный переход не является управляемым, в спазисторе за счет введения дополнительных электродов обеспечивается управление электронно-дырочным переходом и резко повышаются частотные пределы работы по сравнению с обычными транзисторами.

В транзисторах частотные характеристики ограничиваются конечным временем пролета носителей заряда (электронов и дырок) через область базы и коллекторную область. Основное ограничение налагает базовая область, в которой действующее поле обычно весьма невелико и скорость перемещения носителей мала.

В спазисторе вообще отсутствует такой электрод, как база, и исключается участие базовой области в рабочем процессе прибора.

Конструктивно спазистор оформляется в виде полупроводникового диода, у которого в область электронно-дырочного перехода (рис. 94) введены два дополнительных электрода: *инжектор (И)* и *модулятор* (*М*).

Инжектор имеет потенциал ниже потенциала области перехода и является для этой области источником дополнительных электронов.

Модулятор включается таким образом, чтобы в его цепи не проходило заметного тока. Он служит для управления потоком электронов из инжектора путем изменения напряжения, приложенного между модулятором и инжектором.

Таким образом, переменное напряжение, приложенное ко входу прибора (между модулятором и инжектором), влияет на электронный поток, испускаемый инжектором, и на основной ток, проходящий через прибор и сопротивление нагрузки.

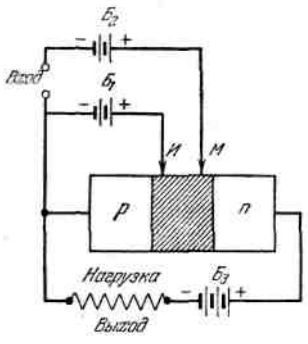



Рис. 94. Спазистор.

В результате на выходе — сопротивлении нагрузки — появляются усиленные колебания входного сигнала, т. е. спазистор выполняет роль усилителя.

Результаты экспериментов, опубликованные в печати, показывают, что новые приборы типа спазисторов будут способны работать при частотах до 10 000 *мгц* при температурах до 500° С. Третьим направлением в развитии полупроводниковых приборов является создание элементов со многими электронно-дырочными переходами.

Как мы уже упоминали, в полупроводниковом диоде имеется один переход, в транзисторах — два перехода; строятся также полупроводниковые приборы с тремя и большим количеством переходов. При этом используются установившиеся методы получения (*p* — *n*)-переходов, но путем сочетания в одном приборе различных областей кристалла *p*- или *n*-проводимости, разделенных

многими выпрямляющими переходами, обеспечивается получение комплексных приборов, выполняющих сложные функции. При этом достигаются упрощение схем аппаратуры и технологии ее производства, сокращение числа элементов и уменьшение стоимости аппаратуры. Работы в этом направлении аналогичны работам в ламповой технике, направленным на совмещение в одном баллоне нескольких ламп и увеличение числа электродов в лампах.

**3. Разработка новых переключательных и запоминающих элементов.** Примером использования новых физических принципов для создания переключательных и запоминающих элементов могут служить разработки криотронов и фотохромной памяти. *Криотрон* представляет собой электронный переключательный элемент, работа которого основана на явлении сверхпроводимости. Конструктивно криотрон выполнен в виде небольшой проволочки — вентиля длиной 2—3 *мм,* на которой намотана в один слой управляющая обмотка из проволоки толщиной в человеческий волос. Криотроны помещаются в ванночках, наполненных жидким гелием, поддерживающим температуру, близкую к абсолютному нулю.

Как известно, при такой температуре металлы приобретают свойство сверхпроводимости, т. е. их электрическое сопротивление становится ничтожно малым. Если же на металл, находящийся в сверхпроводящем состоянии, воздействует магнитное поле, то свойство сверхпроводимости теряется и металл восстанавливает свое обычное сопротивление электрическому току. Таким образом, вентильная проволочка криотрона может находиться в двух устойчивых состояниях: состоянии сверхпроводимости и обычном состоянии. Магнитное поле, управляющее криотроном, создается при пропускании тока через управляющую обмотку.

При отсутствии тока в управляющей обмотке вентиль криотрона является сверхпроводящим, т. е. находится в открытом состоянии; при подаче тока в управляющую обмотку вентиль восстанавливает свое сопротивление — запирается. Эти простые элементы можно соединить в цепи для выполнения всех функций электронных ламп, полупроводниковых триодов, магнитных сердечников и других запоминающих и переключательных элементов. Подсчитано, что все схемы большой электронной вычислительной машины, будучи построены на криотронах, занимали бы объем не больше 0,1 $м^3$. Криотроны могут быть применены для построения малогабаритных накопителей чрезвычайно большой емкости, обладающих произвольной выборкой чисел. При этом, например, по-видимому, будет более выгодно постоянно хранить в таких накопителях таблицы умножения многозначных чисел и операцию умножения заменить простым обращением к такому накопителю. При помощи таких накопителей можно будет широко пользоваться различными специальными таблицами, в частности, баллистическими таблицами для выполнения артиллерийских расчетов.

В Массачузетском технологическом институте (США) ведутся работы по созданию электронной справочной машины-каталога на криотронах, в которой будет примерно 215 000 криотронов. Для построения подобной машины на электронных лампах потребовалось бы 50 000 ламп. Криотроны могут быть использованы для построения машин, переводящих с одного языка на другой, для которых потребовались бы многие тысячи электронных ламп, если бы их строить на обычных элементах.

Достоинствами криотронов являются простота конструкции, позволяющая автоматизировать процессы изготовления криотронов и сборки схем, а также незначительное потребление энергии, обусловленное ничтожным сопротивлением сверхпроводящих элементов.

Недостатком криотронов в настоящее время является большое время переключения; например, триггер на криотронах переходит из одного состояния в другое за 500 *мксек.* Ведутся работы по повышению быстродействия криотронов.

Другим интересным примером использования новых физических явлений для построения запоминающих элементов электронных цифровых машин является изобретение *фотохромного* запоминающего устройства. Принцип работы этого устройства основан на изменении окраски специального красителя (фотохромного вещества) под действием света определенной длины волны. Запоминающими элементами являются частицы кристалла, растворенные в масле, смешанном с раствором желатина и других коллоидов.

Такой раствор образует «отвердевшую» жидкость, которая тонким слоем может быть нанесена на бумагу или другой материал. Частицы красителя, заключенные в желатине, образуют капсулы диаметром 0,003 *мм,* каждая из которых может запоминать нули или единицы. Под действием синего света капсулы окрашиваются в синий цвет, что соответствует записи единицы; под действием желтого света капсулы обесцвечиваются, что соответствует стиранию записанной информации.

Свет другого цвета является нейтральным, т. е. не окрашивает и не обесцвечивает капсулы и может быть использован для считывания записанной информации.

Фотохромная память представляет интерес с точки зрения создания дешевых быстродействующих оперативных запоминающих устройств весьма большой емкости. Работы в этой области находятся еще в самой начальной стадии исследования. Одной из трудностей является точное управление световым лучом и его фокусировка, обеспечивающие обращение к заданным запоминающим элементам без нарушения записей в соседних элементах.

# ГЛАВА V
# ИСХОДНЫЕ ДАННЫЕ ПРОГРАММИРОВАНИЯ ДЛЯ МАШИН *Стрела, М-3 И Урал*

## § 26. Порядок выполнения команд

В зависимости от порядка выполнения команд машины подразделяются на два типа:

а) машины с *естественным* порядком выполнения команд;

б) машины с *принудительным* порядком выполнения команд. От типа машины зависит характер использования адресов команд этой машины.

**1. Машины с естественным порядком выполнения команд.** В машинах с *естественным* порядком выполнения команд вся совокупность команд записывается в некоторой группе ячеек памяти, имеющих последовательные номера. После выполнения команды, хранящейся в некоторой ячейке, машина переходит к выполнению другой команды, хранящейся в ячейке с номером на единицу большим. И так до тех пор, пока не встретится специальная *команда перехода,* которая передает управление не следующей, а некоторой другой команде (см. ниже), или специальная *команда останова* (под воздействием которой машина остановится).

Для машин с естественным порядком выполнения команд приведем несколько примеров команд в том виде, в каком их записывают на бланках для программ.

П р и м е р 1. Для некоторых трехадресных машин, например, возможна такая команда:

0101 0237 0421 01, (V.1)

имеющая следующее содержание: «Взять число из ячейки № 0101. Другое число взять из ячейки № 0237. Эти числа сложить (считаем, что 01 — код операции сложения). Результат операции (сумму) записать в ячейку № 0421». Эта команда состоит из четырех групп цифр: трех адресов и кода операции. Количество адресов, упоминаемых в команде, обусловлено конструкцией машины.

Взаимное расположение адресов и кода операции может быть и не таким, как в приведенной выше записи (V.1). Например, возможен и такой вид той же трехадресной команды:

01 0101 0237 0421. (V.2)

Взаимное расположение кода операции и адресов в команде тоже связано с конструкцией машины. Для каждой машины структура команд является вполне определенной, так что нельзя по своему усмотрению иногда писать команду в виде (V.1), а иногда в виде (V.2).

П р и м е р 2. Команды для двухадресных машин с естественным порядком выполнения команд имеют, например, следующий вид:

0101 0237 01,

что значит: «Число, хранящееся в ячейке № 0101, сложить с числом, хранящимся в ячейке № 0237 (01 — код операции сложения). Результат записать в ячейке № 0237»*.

Для того чтобы сумма была записана в ячейке № 0421, нужна еще одна команда:

0237 0421 24,

которая означает: «Число из ячейки № 0237 перенести в ячейку № 0421» (24 — код операции переноса числа из одной ячейки памяти в другую).

П р и м е р 3. Особой разновидностью двухадресных машин являются так называемые *полутораадресные* машины. Команды для таких машин являются двухадресными, но второй адрес в них записан не четырьмя, а лишь двумя восьмеричными цифрами. Такое упрощение команд становится возможным благодаря тому, что конструкцией машин предусмотрено лишь небольшое количество ячеек памяти, номера которых могут фигурировать во втором адресе команды. Например, это могут быть ячейки с номерами, начиная от 3700 и кончая 3777 (всего шестьдесят четыре ячейки). Чтобы сложить числа, записанные в ячейках № 0101 и № 0237, а сумму записать в ячейку № 0421, пришлось бы воспользоваться командами:

0101 54 21,

«Число из ячейки № 0101 перенести в ячейку № 3754» (21 — код операции переноса числа из ячейки памяти, упоминаемой в первом адресе, в ячейку, упоминаемую во втором адресе);

0237 54 01,

«Число из ячейки № 0237 сложить с числом, хранящимся в ячейке № 3754. Сумму записать в ту же ячейку № 3754»;

0421 54 26,

«Перенести число из ячейки № 3754 в ячейку № 0421» (26 — код операции переноса числа из ячейки, упомянутой во втором адресе, в ячейку, упомянутую в первом адресе команды).

П р и м е р 4. Сумматор одноадресной машины обычно устроен так, что он может хранить перенесенное в него число или результат операции до момента выполнения следующей операции над числами.

Одноадресная команда может иметь, например, такой вид:

0101 02.

Такая команда означает: «Взять число из ячейки № 0101 и записать его в сумматор» (02 — код операции переноса числа из ячейки памяти в сумматор). Команда

0237 01

будет иметь такое содержание: «Число, которое находится в ячейке № 0237, прибавить к содержимому сумматора (01 — код операции сложения). Сумму оставить в сумматоре вместо хранившегося там ранее слагаемого». Команда

0421 20

будет гласить: «Содержимое сумматора записать в ячейку № 0421» (20 — код операции переноса числа из сумматора в ячейку памяти).

Приведенные нами три одноадресные команды равноценны одной (приведенной в первом примере) трехадресной команде 0101 0237 0421 01.

Заметим, что одноадресные машины обязательно, а трехадресные обычно имеют естественный порядок выполнения команд.

**2. Машины с принудительным порядком выполнения команд.**

В каждой из команд, применяемых для управления машинами с *принудительным* порядком выполнения команд,

* Всякое число (и команда), записанное в какой-нибудь ячейке, сохраняется в ней до тех пор, пока в эту ячейку не будет произведена запись нового числа. При этом прежнее число автоматически сотрется,

указывается номер ячейки, хранящей ту команду, которая должна выполняться вслед за данной. Отсюда ясно, что одноадресные машины не могут иметь принудительного порядка выполнения команд (в команде нет места для указания номера ячейки, содержащей следующую команду). Для полутораадресных машин применение принудительного порядка выполнения команд нецелесообразно: такая машина либо допускала бы только очень короткие программы, либо позволяла бы хранить лишь небольшое количество чисел (в зависимости от того, под команды или под числа была бы отведена та небольшая группа ячеек, номера которых могут быть представлены в укороченном адресе команды).

Принудительный порядок выполнения команд применяется главным образом в машинах, имеющих в качестве оперативной памяти вращающийся магнитный барабан. При этом ячейками памяти являются участки барабана, расположенные вдоль его образующих. Считывание и запись чисел производится с помощью расположенных вдоль барабана магнитных головок. Время оборота барабана довольно велико, во всяком случае значительно больше того времени, которое расходуют устройство управления и арифметическое устройство на выполнение команд.

Отводя ячейки памяти для команд, исходных данных и результатов операций так, чтобы к тому моменту, когда команда считана, под головками находилась ячейка с необходимым исходным числом, к моменту, когда операция выполнена, — ячейка, отведенная для ее результата, а к моменту окончания записи результата — ячейка с очередной командой, можно добиться уменьшения времени, расходуемого машиной на решение задачи. При таком расположении команд, исходных данных и результатов последовательные команды программы оказываются размещенными в ячейках, номера которых не являются последовательными. Оптимальное расположение программы, исходных данных и результатов в оперативной памяти машин с принудительным порядком выполнения команд (так называемое *оптимальное распределение памяти)* представляет собой своеобразную и весьма нелегкую задачу.

П р и м е р 5. Четырехадресные машины, как правило, имеют принудительный порядок выполнения команд. Для управления ими применяются команды, три адреса которых используются так же, как в трехадресных командах, а четвертый служит для указания номера ячейки, хранящей очередную команду.

Например, команда может иметь такой вид:

0101 0237 0421 0057 01,

что значит: «Взять число из ячейки № 0101. Второе число взять из ячейки № 0237. Сложить их (01 — код операции сложения). Сумму записать в ячейку № 0421. Перейти к выполнению команды, хранящейся в ячейке № 0057».

П р и м е р 6. Трехадресная машина с принудительным порядком выполнения команд работает, как двухадресная. Вот примеры команд для такой машины:

0101 0237 0055 01,

«Число, хранящееся в ячейке № 0101, сложить с числом, записанным в ячейке № 0237 (01 — код операции сложения). Сумму записать в ячейке № 0237. Перейти к выполнению команды, находящейся в ячейке № 0055». В ячейке № 0055 хранится очередная команда. Например,

0237 0421 0105 45,

что значит: «Из ячейки № 0237 перенести число в ячейку № 0421 (45 — код операции переноса числа). Затем перейти к выполнению команды, хранящейся в ячейке № 0105».

П р и м е р 7. Двухадресные машины с принудительным порядком выполнения команд работают как одноадресные и имеют обычно сумматор, запоминающий полученный результат до момента выполнения следующей операции над числами. Вот примеры команд для такой машины:

0101 0055 17,

«Число из ячейки № 0101 перенести в сумматор (17 — код операции переноса числа из памяти в сумматор). Затем перейти к выполнению команды, хранящейся в ячейке № 0055». В ячейке № 0055 хранится очередная команда. Например,

0237 0105 01,

что значит: «Взять число из ячейки № 0237 и сложить (01 — код операции сложения) с содержимым сумматора. Сумму оставить в сумматоре. Затем перейти к выполнению команды, хранящейся в ячейке № 0105». Для того чтобы сумма оказалась записанной в ячейке № 0421, в ячейке № 0105 должна храниться, например, такая команда:

0421 0110 21,

что значит: «Содержимое сумматора перенести (21 — код операции переноса числа из сумматора в ячейку памяти) в ячейку № 0421. Перейти к выполнению команды, записанной в ячейке № 0110».

Трехадресные машины с естественным порядком выполнения команд весьма удобны; они более других соответствуют характеру арифметических действий. Как правило, арифметические действия производятся над двумя исходными числами и приводят к третьему числу — результату операции. Одноадресные машины наиболее просты по конструкции и поэтому являются наиболее дешевыми. Большинство существующих программно-управляемых цифровых машин являются либо одноадресными, либо трехадресными.

Однако исследования, проведенные недавно в Отделении прикладной математики Математического института им. Стеклова АН СССР, показали, что серьезными достоинствами обладают также двухадресные машины (с естественным порядком выполнения команд). Программы для таких машин при решении математических задач оказываются примерно лишь на 25—30% длиннее, чем для трехадресных, и почти в два раза короче, чем для одноадресных машин. Конструкция двухадресной машины хотя и сложнее одноадресной, но значительно проще, чем конструкция трехадресной машины. Большинство существующих электронных цифровых машин имеет естественный порядок выполнения команд, поэтому в дальнейшем имеются в виду только машины этого типа.

**3. «Чтение» машиной содержимого ячейки.** В большинстве цифровых программно-управляемых машин как число (десятичное или двоичное), так и команда могут быть помещены в любую ячейку памяти. Кроме того, как увидит читатель дальше, при решении задач на машинах часто применяются так называемые *вспомогательные числа,* не имеющие количественного значения. Вспомогательными числами являются специально составленные наборы нулей и единиц. Они могут быть нужны для преобразования команд программы и т. п. Вспомогательное число тоже может быть помещено в любую ячейку памяти.

Что бы ни хранилось в ячейке памяти — число, команда, или вспомогательное число, — содержимое ячейки является последовательностью нулей и единиц. Выписав на бумаге эту последовательность, мы получим так называемый *двоичный код* содержимого ячейки. Этим двоичным кодом могут быть изображены как число (десятичное или двоичное), так и команда или вспомогательное число (двоичный код вспомогательного числа графически

тождествен с этим вспомогательным числом).

Во многих машинах каждая команда занимает в памяти целую ячейку. Предполагая, что содержимое ячейки является командой, и, выписывая на бумаге эту команду в том виде, который установлен для записи команд данной машины на бланках для программ, мы получим изображение содержимого ячейки в так называемом *коде команд.* Число (двоичное или десятичное), команду или вспомогательное число для многих машин можно представить в коде команд.

Если в ячейке хранится десятичное число, еще не переведенное в двоичную систему счисления, то, выписывая его в форме, принятой при записи на бланках для программ, получим так называемый *числовой код* содержимого ячейки.

Не всякое содержимое ячейки можно представить в числовом коде, так как в нем могут встречаться такие сочетания нулей и единиц, которые не соответствуют ни одной десятичной цифре. Например, сочетание 1110 не является тетрадой двоично-десятичной системы. В случае машины, в которой принята двоично-десятичная запись десятичных чисел (т. е. для большинства машин), такое сочетание нельзя передать десятичной цифрой числового кода.

В дальнейшем, после описания правил записи команд и десятичных чисел на бланках для программ с целью ввода их в конкретные машины, мы поясним на примерах все три перечисленных выше кода содержимого ячейки.

У читателя, вероятно, возник вопрос: «Если число, команда и вспомогательное число могут иметь одинаковое физическое изображение в ячейке памяти, то как «узнает» машина, что записано в ячейке?» Поясним это.

При пуске машины происходит считывание содержимого определенной ячейки (например, некоторой начальной ячейки). Это содержимое воспринимается машиной как команда. Программист при размещении программы в памяти машины должен позаботиться о том, чтобы в этой ячейке действительно стояла н у ж н а я к о м а н д а, а не что-нибудь другое.

После выполнения первой команды машина переходит к считыванию содержимого очередной ячейки (следующей по номеру при естественном порядке выполнения команд; указанной в только что выполненной команде при принудительном порядке выполнения команд). В этой ячейке также должна стоять команда и так далее.

Числа и вспомогательные числа должны стоять в ячейках, указанных в адресах соответствующих команд. Программа должна быть построена так, чтобы ни одна ячейка, хранящая не команду, не могла получить управления (т. е. чтобы ее содержимое не было считано машиной в качестве команды).

## § 27. Общая характеристика машины *Стрела*

**1. Запись чисел в ячейках памяти.** Советская цифровая программно-управляемая машина *Стрела* является машиной с плавающей запятой.

Каждая ячейка памяти машины «Стрела» состоит из сорока трех разрядов. Разряды считаются занумерованными слева направо

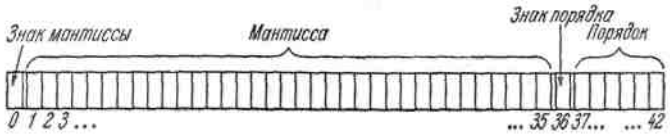



Рис. 95. Распределение разрядов ячейки памяти машины *Стрела* при хранении двоичного числа.

(т. е. от старших к младшим) **с** помощью десятичных чисел 0, 1, 2, ..., 42. Схема распределения разрядов ячейки при хранении двоичного числа приведена на рис. 95. Нулевой разряд служит для хранения знака мантиссы. Разряды, начиная с первого и кончая тридцать пятым, отведены для ее цифровой части. В тридцать шестом разряде записывается знак порядка, а в последних шести разрядах, с тридцать седьмого по сорок второй, хранится абсолютная величина порядка.

В ячейке памяти машины *Стрела* может храниться также и десятичное число.

Пусть $N$ — некоторое десятичное число. Его можно представить в нормализованном виде

$$N = M \cdot 10^P \text{ (10 —десять),}$$

где $\frac{1}{10} \leq |M| < 1$, $P$ — целое. Как и в случае двоичного нормализованного числа, $M$ называют *мантиссой,* а $P$ — *порядком.* Для хранения мантиссы $M$ отведены разряды, начиная с нулевого и кончая тридцать шестым. При

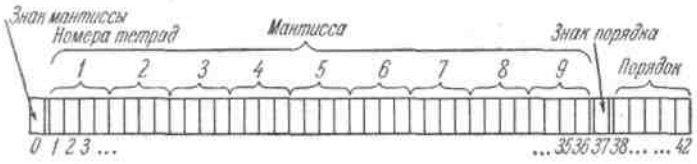



Рис. 96. Распределение разрядов ячейки памяти машины *Стрела* при хранении десятичного числа.

этом нулевой разряд является знаковым, а остальные тридцать шесть разрядов — цифровыми. Каждая десятичная цифра мантиссы записывается в ячейке с помощью тетрады (четверки двоичных цифр). Таким образом, мантисса представлена в ячейке памяти машины *Стрела* в двоично-десятичной системе счисления (см. § 5). В указанных тридцати шести цифровых разрядах могут быть размещены девять тетрад, т. е. девять десятичных значащих цифр мантиссы.

Для хранения порядка числа предназначены разряды, начиная с тридцать седьмого и кончая сорок вторым. Тридцать седьмой разряд отведен для знака порядка. Порядок изображается в ячейке машины *Стрела* в виде целого пятизначного двоичного числа.

Схема распределения разрядов ячейки при хранении десятичного числа показана на рис. 96.

Обращаем внимание читателя на то, что при записи двоичного числа для мантиссы (с ее знаком) отведено тридцать шесть разрядов, а для порядка (с его знаком) — семь разрядов. При записи десятичного числа для мантиссы (с ее знаком) отведено тридцать семь разрядов, а для порядка (с его знаком) — шесть разрядов.

Оперативная память машины *Стрела* состоит из 2048 ячеек. Ячейки эти занумерованы в двоичной системе счисления с помощью следующих чисел:

000 000 000 000;
000 000 000 001;
000 000 000 010;
... ... ... ...
011 111 111 111

Неудобство применения двоичной нумерации ячеек при составлении команд совершенно очевидно (большая «длина» двоичного изображения чисел). Поэтому, составляя команды, номера ячеек записывают в восьмеричной системе счисления. При переносе команд на перфокарты (§ 10) восьмеричные числа чисто механическим путем преобразуются в двоичные. К моменту ввода команд в память машины входящие в команды номера ячеек оказываются записанными уже в двоичной системе счисления. Каждый номер ячейки принято записывать в виде четырехзначного восьмеричного числа: 0000, 0001, 0002, ..., 3777.

**2. Структура команд и их запись.** Машина *Стрела* является трехадресной машиной с естественным порядком выполнения команд. Каждая команда состоит из трех адресов, контрольного знака и кода операции.

При записи на бланках для программ адреса представляются в виде четырехзначных восьмеричных чисел, контрольный знак может быть либо нулем, либо единицей (и, следовательно, является однозначным двоичным числом). Код операции записывается в виде двузначного восьмеричного числа.

Например, команда может иметь такой вид:

0065 0231 1101 0 01,

или

0065 0231 1101 1 01.

Обе вышеприведенные команды, несмотря на различие в контрольных знаках, имеют одинаковое содержание: «Число, хранящееся в ячейке № 0065, сложить (01 — код операции сложения) с числом, которое находится в ячейке № 0231; сумму записать в ячейку № 1101».

Двоичный код содержимого ячейки, хранящей команду, получается из вышеприведенной записи, если каждую восьмеричную цифру заменить отвечающей ей тройкой двоичных цифр, а контрольный знак сохранить в виде одной двоичной цифры.

Для двух приведенных выше команд двоичный код имеет следующий вид:

000 000 110 101 000 010 011 001 001 001 000 001 0 000 001

000 000 110 101 000 010 011 001 001 001 000 001 1 000 001

В соответствии со сказанным для хранения каждого адреса в ячейке отведено по двенадцати разрядов, для контрольного знака — один разряд и для кода операции — шесть разрядов.

Схема распределения разрядов ячейки при хранении в ней команды приведена на рис. 97. Для первого адреса отведены разряды с нулевого по одиннадцатый, для второго адреса — с двенадцатого по двадцать третий, для третьего адреса — с двадцать четвертого по тридцать пятый, для контрольного знака — тридцать шестой разряд, для кода операции — разряды с тридцать седьмого по сорок второй.

Назначение адресов и кода операции читателю уже известно. Контрольный знак на содержание команды не влияет. Обычно его

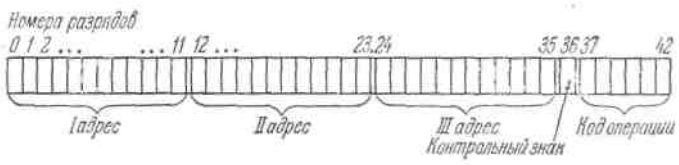



Рис. 97. Распределение разрядов ячейки памяти машины *Стрела* при хранении команды.

используют при проверке программы. Для этого путем переключения специального тумблера, находящегося на пульте управления, машину *Стрела* переводят на работу в «режиме контрольных остановов». При этом режиме, выполнив команду, имеющую контрольный знак, равный единице, машина каждый раз останавливается (подробнее см. § 40).

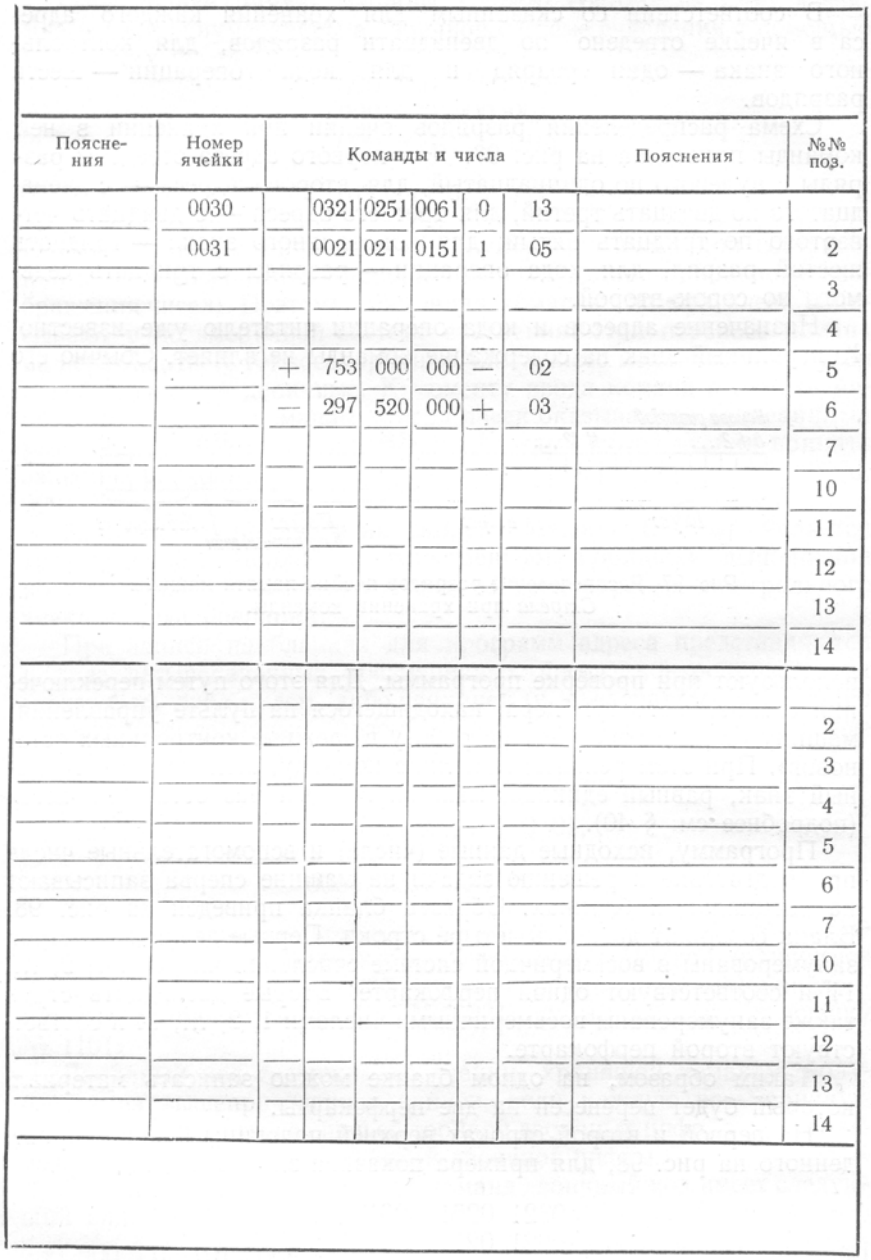
Программу, исходные данные (числа) и вспомогательные числа при подготовке к решению задачи на машине сперва записывают на специальных бланках. Образец бланка приведен на рис. 98.Бланк содержит двадцать четыре строки. Первые двенадцать строк занумерованы в восьмеричной системе счисления числами 1, 2, ..., 14 и соответствуют одной перфокарте. Вторые двенадцать строк также занумерованы восьмеричными числами 1, 2, 14 и соответствуют второй перфокарте.

Таким образом, на одном бланке можно записать материал, который будет перенесен на две перфокарты.

На первой и второй строках верхней половины бланка, приведенного на рис. 98, для примера показана запись команд

0321 0251 0061 0 13
0021 0211 0151 1 05,

подлежащих размещению в ячейках памяти 0030 и 0031.



| Поясне-ния | Номер ячейки | Команды и числа | | | | | | Пояснения | №№ поз. |
|---|---|---|---|---|---|---|---|---|---|
| | 0030 | | 0321 | 0251 | 0061 | 0 | 13 | | |
| | 0031 | | 0021 | 0211 | 0151 | 1 | 05 | | 2 |
| | | | | | | | | | 3 |
| | | | | | | | | | 4 |
| | | + | 753 | 000 | 000 | — | 02 | | 5 |
| | | — | 297 | 520 | 000 | + | 03 | | 6 |
| | | | | | | | | | 7 |
| | | | | | | | | | 10 |
| | | | | | | | | | 11 |
| | | | | | | | | | 12 |
| | | | | | | | | | 13 |
| | | | | | | | | | 14 |
| | | | | | | | | | |
| | | | | | | | | | 2 |
| | | | | | | | | | 3 |
| | | | | | | | | | 4 |
| | | | | | | | | | 5 |
| | | | | | | | | | 6 |
| | | | | | | | | | 7 |
| | | | | | | | | | 10 |
| | | | | | | | | | 11 |
| | | | | | | | | | 12 |
| | | | | | | | | | 13 |
| | | | | | | | | | 14 |

Рис. 98. Бланк программы для машины *Стрела.*

На пятой и шестой строках той же половины бланка показана запись десятичных чисел

$$+\,0{,}00753 \quad \text{и} \quad -\,297{,}52.$$

Приводя их к нормализованному виду, имеем:

$$+\,0{,}753 \cdot 10^{-2} \quad \text{и} \quad -\,0{,}29752 \cdot 10^{3}.$$

Мантисса первого числа +753 000 000, а его порядок – 02. Мантисса второго числа –297 520 000, а порядок +03.

Таким образом, на бланке и мантисса и порядок записываются в десятичной системе счисления. При этом абсолютная величина порядка не должна быть больше, чем 19 (десятичное). Это ограничение вызвано следующими причинами. Во-первых, даже самое большое по абсолютной величине нормализованное десятичное число, имеющее порядок меньший, чем –19, т.е. $0{,}999999999 \cdot 10^{-20}$, является машинным нулем*. Во-вторых, даже самое малое по абсолютной величине десятичное нормализованное число, имеющее порядок больший, чем +19, а именно $0{,}1 \cdot 10^{+20}$, переполняет разрядную сетку *Стрелы.* Вооружившись таблицами логарифмов, читатель легко убедится в этом с помощью несложных вычислений.

**3. Ввод программы в память машины.** Для ввода в память машины числа и команды, записанные на бланках, должны быть перенесены на перфокарты (говорят: «Должны быть пробиты на перфокартах»).

На каждой перфокарте может быть пробито двенадцать чисел (или команд). Каждое число пробивается на отдельной строке перфокарты. Таким образом, можно считать, что перфокарта разделена на двенадцать строк. В свою очередь каждая строка перфокарты разделена на сорок три участка (соответственно количеству разрядов ячейки памяти машины *Стрела).* Каждый участок предназначен для изображения содержимого одного разряда

* После перевода в двоичную систему счисления.

ячейки (0 — если на участке нет отверстия, 1 — если на участке пробито отверстие).

При пробивке команд каждая восьмеричная цифра пробивается в виде тройки двоичных цифр. Двоичная цифра контрольного знака пробивается в виде одной двоичной цифры. При пробивке Десятичного числа знак «+» или «–» пробивается в виде одной двоичной цифры (соответственно в виде 0 или 1), каждая десятичная Цифра мантиссы пробивается в виде тетрады (четырех двоичных Цифр). Порядок пробивается сразу в виде пятиразрядного двоичного числа (конструкция входного перфоратора не позволяет пробивать порядки большие по абсолютной величине, чем 19). Преобразование чисел осуществляется при этом автоматически, без каких-либо вычислений, производимых человеком; оператор только нажимает соответствующие клавиши клавишного устройства, на которых написаны восьмеричные или десятичные цифры. Кроме того, на левом конце каждой перфокарты автоматически пробивается номер задачи и номер перфокарты в двоично-десятичной системе.

На рис. 64 изображена перфокарта с пробитыми на ней числами (стр. 94).

Внешний накопитель машины *Стрела* состоит из двух бобин с ферромагнитной лентой. Запись чисел на магнитные ленты осуществляется путем переноса этих чисел на ленту из оперативной памяти. Считывание чисел с магнитной ленты состоит в их переносе в оперативную память.

Каждую из магнитных лент разбивают на зоны (на одной ленте может быть до пятисот одиннадцати зон). На каждой зоне может быть записано от одного до двух тысяч сорока восьми чисел (большего количества чисел не может вместить оперативная память). Зоны первой магнитной ленты имеют восьмеричные номера от 4001 до 4777, зоны второй магнитной ленты — от 5001 до 5777.

Запись чисел из памяти в зону магнитной ленты и считывание их оттуда в память могут производиться только группами. С каждой зоны можно произвести считывание любого количества чисел (не большего, чем их там хранится), начиная обязательно с первого числа зоны. Разбивка магнитной ленты на зоны производится с помощью специально составленной маленькой программы. Запись чисел в зоны можно производить либо с помощью специальной программы, либо предусмотрев соответствующие команды в программе решения задачи.

В составе машины *Стрела* имеется специальное запоминающее устройство, в котором постоянно хранятся некоторые часто встречающиеся константы. Это устройство получило название *устройства выдачи констант* (сокращенно *УВК*). Записывать числа в *УВК* в процессе решения задачи нельзя. В случае необходимости замены какого-нибудь числа, хранящегося в *УВК,* другим, это производится заранее, до решения задачи. Таким образом, во время решения задачи на машине из *УВК* можно брать числа и использовать их для вычислений, но получаемые результаты записывать в *УВК* нельзя. Ячейкам *УВК* присвоены восьмеричные номера, начиная с 7400 и кончая 7777.

Наконец, в машине *Стрела* имеется специальное запоминающее устройство, в котором постоянно хранятся стандартные подпрограммы — небольшие программы, с помощью которых вычисляются значения часто встречающихся функций ($\frac{1}{x}$, In $x$, $e^x$, sin $x$ и т. п.). Это запоминающее устройство называют *накопителем стандартных подпрограмм* (сокращенно *НСП).*

Ячейка № 0000 оперативной памяти машины *Стрела* постоянно хранит в себе число нуль. Всякое другое число, записываемое в эту ячейку, в ней «гасится». Всякое число, записанное в любую ячейку, кроме ячейки №0000, сохраняется в ней до тех пор, пока на его место не будет произведена запись нового числа. При переносе числа из некоторой ячейки в другую происходит как бы «фотографирование» числа, так что после переноса одинаковые числа оказываются в двух ячейках. Если в процессе выполнения команды из ячейки берется число для выполнения какой-нибудь операции, то содержимое ячейки не меняется (кроме случая, когда в команде предусмотрена запись результата на место исходного числа).

Из вышесказанного видно, что если целое восьмеричное число $A$ является адресом команды, означающим место хранения числа (или группы чисел), то

$A$ является
- номером ячейки оперативной памяти при $0000 \le A \le 3777$;
- номером зоны первой магнитной ленты . при $4001 \le A \le 4777$;
- номером зоны второй магнитной ленты при $5001 \le A \le 5777$;
- номером ячейки устройства *УВК* при $7400 \le A \le 7777$.

**4. Автоматическое управление работой машины.** При выполнении ряда операций из арифметического устройства машины в управляющее устройство, в зависимости от определенного признака у результата операции, передается специальный сигнал, называемый сигналом ω. Признаком, обусловливающим появление сигнала ω при сложении, является отрицательный знак суммы, при сравнении чисел — их несовпадение, и т. п. Если вместе с результатом операции вырабатывается сигнал ω, то говорят, что ω = 1. Если отсутствует сигнал ω, то говорят, что ω = 0.

Некоторые операции ни при каком результате не сопровождаются появлением сигнала ω. Для этих операций всегда ω = 0. Передача сигнала ω из арифметического устройства в управляющее устройство предусмотрена для автоматического управления работой машины в зависимости от получаемых при вычислениях результатов. Автоматическое управление осуществляется с помощью операции условного перехода, состоящей в передаче управления машиной, не следующей по порядку записи в ячейках команде, а одной из двух заданных команд в зависимости от значения ω.

В результате некоторых операций происходит переполнение разрядной сетки, состоящее в получении результата с порядком большим, чем 77 (восьмеричное). При этом происходит автоматический останов машины.

При получении результатов операций, имеющих порядок меньший, чем –77, в качестве ответа получается машинный нуль.

## § 28. Система операций и команд машины *Стрела*

Для выполнения каждой операции предусмотрена определенная команда, представляющая собой число, образованное пятью группами цифр: тремя адресами, контрольным знаком (нулем либо единицей) и кодом операции (код операции является одновременно и ее номером).

При буквенном обозначении команды мы будем опускать контрольный знак в тех случаях, когда он не влияет на содержание программы. Буквенная запись команды имеет такой вид:

$$a, b, c; \theta.$$

Здесь $a$, $b$ и $c$ — адреса команды, $\theta$ — код операции.

Команда, состоящая из нулей, не вызывает выполнения какой-либо операции и лишь передает управление следующей команде, записанной в очередной ячейке.

В дальнейшем будем применять следующие обозначения.

Если ячейка (ее номер) обозначена буквой $a$, то хранящееся в ней число обозначается той же буквой, заключенной в круглые скобки: $(a)$. Если некоторое число обозначено буквой $N$, то ячейка (ее номер), хранящая это число, обозначается той же буквой, заключенной в угловые скобки: $\langle N \rangle$.

Номером команды мы будем называть номер хранящей ее ячейки.

Говоря об операции, выполняемой машиной над числами $(a)$ и $(b)$, и желая подчеркнуть, что результат направляется в ячейку $c$, мы будем иногда употреблять запись вида

$$(a)*(b) = (c),$$

где символ * представляет знак операции. Например, в случае сложения будем писать:

$$(a) + (b) = (c).$$

Такая запись придает знаку равенства двойной смысл: при сравнении величин, стоящих в левой и правой частях формулы, он обозначает их равенство, а при описании операции он указывает на то, что символ $(c)$, стоящий в правой части формулы, принимает значение результата, указанного в ее левой части. В частности, формулу

$$(a) = (c)$$

можно понимать и как утверждение равенства величин $(a)$ и $(c)$ и как обозначение операции переноса числа $(a)$ в ячейку $c$.

**1. Таблица команд и операций.** В таблице 25 приведена система команд машины *Стрела*. Коды операций, показанные в таблице, являются восьмеричными числами. Команды расположены не в порядке номеров операций, а сгруппированы по характеру этих операций. Знаком $P(c)$ обозначен порядок числа, хранящегося в ячейке $c$. Для записи порядка принята восьмеричная система счисления.

Таблица 25. Система команд машины «Стрела»

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Условие, при котором происходит переполнение разрядной сетки | Условие, при котором результат является машинным нулем | Действия, выполняемые машиной по команде |
|---|---|---|---|---|---|
| 01 | Сложение: $a, b, c$; 01 | $(c) < 0$ | $P(c) > 77$ | $P(c) < -77$ | Числа $(a)$ и $(b)$ алгебраически складываются. Сумма нормализуется и записывается в ячейку $c$. |
| 03 | Вычитание: $a, b, c$; 03 | $(c) < 0$ | $P(c) > 77$ | $P(c) < -77$ | От числа $(a)$ отнимается число $(b)$. Разность нормализуется и записывается в ячейку $c$. |
| 05 | Умножение: $a, b, c$; 05 | $\lvert(c)\rvert \geqslant 1$ | $P(c) > 77$ | $P(c) < -77$ | Числа $(a)$ и $(b)$ перемножаются. Произведение записывается в ячейку $c$. Если хотя бы один из сомножителей не нормализован, то получается $(c) = 0$. |
| 04 | Вычитание модулей: $a, b, c$; 04 | $(c) < 0$ | — | $P(c) < -77$ | От абсолютной величины числа $(a)$ отнимается абсолютная величина числа $(b)$. Результат записывается в ячейку $c$. |
| 06 | Сложение порядков: $a, b, c$; 06 | $P(c) \geqslant 1$ | $P(c) > 77$ | $P(c) < -77$ | В ячейку $c$ записывается число, имеющее мантиссу числа $(a)$ и порядок, равный $P(a) + P(b)$. Нормализация результата не производится. Если число $(a)$ является ненормализованным, то и результат не нормализован. |

Продолжение

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Условие, при котором происходит переполнение разрядной сетки | Условие, при котором результат является машинным нулем | Действия, выполняемые машиной по команде |
|---|---|---|---|---|---|
| 07 | Вычитание порядков: *a*, *b*, *c*; 07 | $P(c) \geqslant 1$ | $P(c) > 77$ | $P(c) < -77$ | В ячейку *c* записывается число, имеющее мантиссу числа (*a*) и порядок, равный $P(a) - P(b)$. Нормализация результата не производится. Если число (*a*) является ненормализованным, то и результат не нормализован. |
| 10 | Перенос числа с присвоением знака другого числа: *a*, *b*, *c*; 10 | $P(c) \geqslant 1$ | — | — | В ячейку *c* записывается число, имеющее абсолютную величину числа (*a*) и знак числа (*b*). Результат не нормализуется. Если число (*a*) не нормализовано, то и результат не нормализован. |
| 12 | Сложение чисел без округления: *a*, *b*, *c*; 12 | $(c) = 0$ | $P(c) > 77$ | $P(c) < -77$ | Числа (*a*) и (*b*) алгебраически складываются. Округление тридцать пятого разряда мантиссы не производится. Результат нормализуется и записывается в ячейку *c*. |

Продолжение

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Условие, при котором происходит переполнение разрядной сетки | Условие, при котором результат является машинным нулем | Действия, выполняемые машиной по команде |
|---|---|---|---|---|---|
| 12 | Выделение целой части: *a*, 7575, *c*; 12 | $(c) = 0$ | — | — | В ячейку *c* записывается целая часть числа (*a*). Результат нормализован. |
| 11 | Выделение части: *a*, *b*, *c*; 11 | Разряды ячейки *c* содержат только нули | — | — | Содержимое каждого разряда ячейки *a* по таблице логического умножения умножается на содержимое одноименного разряда ячейки *b*. Результат заносится в одноименный разряд ячейки *c*. |
| 13 | Формирование: *a*, *b*, *c*; 13 | Разряды ячейки *c* содержат только нули | — | — | Содержимое каждого разряда ячейки *a* по таблице логического сложения складывается с содержимым одноименного разряда ячейки *b*. Результат записывается в одноименный разряд ячейки *c*. |
| 16 | Сравнение: *a*, *b*, *c*; 16 | Содержимое хотя бы одного разряда ячейки *c* не равно нулю | — | — | Производится отрицание равнозначности содержимого каждого разряда ячейки *a* с содержимым одноименного разряда ячейки *b*. Результат записывается в одноименный разряд ячейки *c*. |

Продолжение

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Условие, при котором происходит переполнение разрядной сетки | Условие, при котором результат является машинным нулем | Действия, выполняемые машиной по команде |
|---|---|---|---|---|---|
| 14 | Сдвиг по порядку: $a, b, c$; 14 | Разряды ячейки $c$ содержат только нули | — | — | Содержимое всех разрядов ячейки $a$ сдвигается на $\lvert P(b) \rvert$ разрядов. При $P(b) > 0$ сдвиг производится влево. При $P(b) < 0$ сдвиг производится вправо. Если $\lvert P(b) \rvert > 37$ (десятичное число), то $(c) = 0$. |
| 14 | Сдвиг по адресу: 1) $a, 4000 + k, c$; 14 или $a, 5000 + k, c$; 14, 2) $a, 4100 + k, c$; 14 или $a, 5100 + k, c$; 14 | Разряды ячейки $c$ содержат только нули | —<br>—<br>— | —<br>—<br>— | Содержимое всех разрядов ячейки $a$ сдвигается влево на $k$ разрядов. Результат записывается в ячейку $c$. При $k < 37$ (десятичное) $(c) = 0$.<br>Содержимое всех разрядов ячейки $a$ сдвигается вправо на $k$ разрядов. Результат записывается в ячейку $c$. При $k < 37$ (десятичное) $(c) = 0$. |
| 17 | Контрольное суммирование: $a, b, c$; 17 | — | — | — | Числа $(a)$ и $(b)$ складываются по всем разрядам с циклическим переносом. Результат записывается в ячейку $c$. |

Продолжение

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Условие, при котором происходит переполнение разрядной сетки | Условие, при котором результат является машинным нулем | Действия, выполняемые машиной по команде |
|---|---|---|---|---|---|
| 02 *) | Специальное сложение: $a, b, c$; 02 | — | — | — | К адресам команды, хранящейся в ячейке $a$, прибавляются «адреса» вспомогательного числа из ячейки $b$. При этом перенос из старшего разряда третьего адреса в младший разряд второго адреса и из старшего разряда второго адреса в младший разряд первого, а также из старшего разряда первого адреса не производится. Эти переносы теряются. Код операции сохраняется. Видоизмененная команда записывается в ячейку $c$. |

*) Операции (и команды) 02 и 15 служат для преобразования команд. Они выполняются над двумя числами, из которых одно является командой, а другое — специально составленным набором цифр (нулей и единиц) для видоизменения команды. Такие наборы цифр, не являющиеся командами и не имеющие количественного значения, называются *вспомогательными числами*. Для сокращения речи часть набора цифр вспомогательного числа, занимающую разряды ячейки, отведенные для одного из адресов команды, называют *адресом* (первым, вторым и третьим) *вспомогательного числа*. В качестве вспомогательного числа иногда используют команды или числа, имеющие количественное значение.

Продолжение

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Условие, при котором происходит переполнение разрядной сетки | Условие, при котором результат является машинным нулем | Действия, выполняемые машиной по команде |
|---|---|---|---|---|---|
| 15 | Специальное вычитание: *a, b, c*; 15 | — | — | — | От адресов команды ($a$) вычитаются «адреса» вспомогательного числа ($b$). Код операции сохраняется. Видоизмененная, таким образом, команда записывается в ячейку $c$.<br>«Заем» в старший разряд уменьшаемого адреса происходит без изменения младшего разряда соседнего (слева) адреса. |
| 62 *) | Вычисление обратной величины: *a, n, c*; 62 | — | $(a + i) = 0$ **), $i = 0, 1, 2, \ldots, n$ | — | Вычисляется $n + 1$ величин, обратных числам ($a$), $(a + 1), \ldots, (a + n)$, и записывается в ячейки с номерами $c, c + 1, \ldots, c + n$. |

*) Приводимые ниже команды с операциями 62, 63, 64, 66, 67, 70, 72, 73 и 74, а также 60 являются командами, по которым передается управление стандартным подпрограммам, постоянно хранящимся в специальном запоминающем устройстве. После конца работы стандартной подпрограммы управление получает очередная команда основной программы (которая не должна быть командой условного перехода).

Тактом работы машины называется совокупность процессов, протекающих в машине при выполнении какой-либо элементарной операции (сложение чисел, сравнение их и т. п.). Стандартные подпрограммы для своего выполнения требуют многих тактов: от десяти до ста пятидесяти (некоторые). Второй адрес $n$ показывает, что данная операция должна быть произведена над группой из $n + 1$ числа. При $n = 0$ вычисления по этим командам производятся над одним числом.

**) Если это условие выполнено хотя бы при одном значении $i$, счет прерывается и происходит останов.

Продолжение

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Условие, при котором происходит переполнение разрядной сетки | Условие, при котором результат является машинным нулем | Действия, выполняемые машиной по команде |
|---|---|---|---|---|---|
| 63 | Извлечение квадратного корня: *a, n, c*; 63 | — | $(a + i) < 0$ *), $i = 0, 1, 2, \ldots, n$ | — | Извлекаются квадратные корни из $n + 1$ чисел ($a$), $(a + 1), \ldots, (a + n)$ и записываются в ячейки с номерами $c$, $c + 1, \ldots, c + n$. Если хотя бы одно из подкоренных чисел отрицательно, то счет прерывается и происходит останов. Кроме того, при $(a + n) > 0$ в ячейку № 0002 производится запись числа $1 : \sqrt{(a + n)}$. |
| 64 | Вычисление показательной функции: *a, n, c*; 64 | — | $P(c + i) > 77$ *), $i = 0, 1, 2, \ldots, n$ | $P(c + i) < -77$ | Вычисляются $n + 1$ чисел $e^{(a)}$, $e^{(a+1)}, \ldots, e^{(a+n)}$ и записываются в ячейки с номерами $c, c + 1, \ldots, c + n$. |
| 66 | Вычисление логарифма: *a, n, c*; 66 | — | $(a + i) \leqslant 0$ *), $i = 0, 1, 2, \ldots, n$ | $P(c + i) < -77$ | Вычисляются $n + 1$ чисел $\ln(a)$, $\ln(a + 1), \ldots, \ln(a + n)$ и записываются в ячейки с номерами $c$, $c + 1, \ldots, c + n$. Если хотя бы одно из чисел $(a), (a + 1), \ldots, (a + n)$ отрицательно или равно нулю, счет прерывается и происходит останов. |

*) Если это условие выполнено хотя бы при одном значении $i$, счет прерывается и происходит останов.

Продолжение

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Условие, при котором происходит переполнение разрядной сетки | Условие, при котором результат является машинным нулем | Действия, выполняемые машиной по команде |
|---|---|---|---|---|---|
| 67 | Вычисление синуса: $a, n, c$; 67 | — | — | $P(c+i) < -77$ | Вычисляются числа $\sin(a)$, $\sin(a+1), \ldots, \sin(a+n)$ и записываются в ячейки $c, c+1, \ldots, c+n$. |
| 73 | Вычисление арктангенса: $a, n, c$; 73 | — | — | $P(c+i) < -77$ | Вычисляются числа $\operatorname{arctg}(a)$, $\operatorname{arctg}(a+1), \ldots, \operatorname{arctg}(a+n)$ и записываются в ячейки $c, c+1, \ldots, c+n$. |
| 74 | Вычисление арксинуса: $a, n, c$; 74 | — | Хотя бы при одном значении $i$ $\lvert a+i \rvert > 1$ | — | Вычисляются $n+1$ чисел $\arcsin(a)$, $\arcsin(a+1), \ldots, \arcsin(a+n)$ и записываются в ячейки с номерами $c, c+1, \ldots, c+n$. Если хотя бы одно из чисел $(a+i)$ по абсолютной величине больше единицы, счет прерывается и происходит останов. Кроме того, в ячейку № 0002 производится запись величины $\arccos(a+n)$. |

Продолжение

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Условие, при котором происходит переполнение разрядной сетки | Условие, при котором результат является машинным нулем | Действия, выполняемые машиной по команде |
|---|---|---|---|---|---|
| 72 | Перевод чисел в двоичную систему: $a, n, c$; 72 | — | — | — | Перевод $n+1$ чисел $(a), (a+1), \ldots, (a+n)$ из двоично-десятичной системы счисления в двоичную и запись их в ячейки $c, c+1, \ldots, c+n$. |
| 70 | Перевод чисел в десятичную систему: $a, n, c$; 70 | — | — | — | Перевод чисел $(a), (a+1), \ldots, (a+n)$ из двоичной системы счисления в двоично-десятичную и запись их в ячейки $c, c+1, \ldots, c+n$. |
| 43 *) | Перенос чисел с ленты в память (Л → П): $a, n, c$; 43 | — | — | — | Перенос $n+1$ чисел из зоны $a$ магнитной ленты соответственно в ячейки $c, c+1, \ldots, c+n$ памяти машины. Перенос начинается с первого числа, записанного в зоне $a$. |

*) Команды 43, 46, 41 и 44 являются командами о групповых переносах чисел. При $n = 0$ переносу подвергается только одно число.

Продолжение

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Условие, при котором происходит переполнение разрядной сетки | Условие, при котором результат является машинным нулем | Действия, выполняемые машиной по команде |
|---|---|---|---|---|---|
| 46 | Перенос чисел из памяти на ленту (П → Л): $a$, $n$, $c$; 46 | — | — | — | Перенос $n+1$ чисел $(a)$, $(a+1)$, ..., $(a+n)$ из памяти в зону номер $c$ магнитной ленты. |
| 41 | Перенос чисел с перфокарт в память (Перф. → П): 0, $n$, $c$; 41 | — | — | — | Перенос $n+1$ чисел с перфокарт в память соответственно в ячейки $c$, $c+1$, ..., $c+n$. |
| 44 | Перенос чисел из памяти на перфокарты (П → Перф.): $a$, $n$, 0; 44 | — | — | — | Перенос $n+1$ чисел $(a)$, $(a+1)$, ..., $(a+n)$ из памяти на перфокарты. |
| 45 | Перенос чисел из памяти в память (П → П): $a$, $n$, $c$; 45 | — | — | — | Перенос $n+1$ чисел $(a)$, $(a+1)$, ..., $(a+n)$ соответственно в ячейки $c$, $c+1$, ..., $c+n$ |

Продолжение

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Условие, при котором происходит переполнение разрядной сетки | Условие, при котором результат является машинным нулем | Действия, выполняемые машиной по команде |
|---|---|---|---|---|---|
| 60 | Групповая передача с контролем: $a$, $n$, $c$; 60 | — | — | — | Производится перенос $n$ чисел, контролирующийся сравнением их сумм до переноса и после переноса. Суммы получаются с помощью операции 17. При несовпадении сумм машина останавливается, выдавая их на пульт управления. При переносе с перфокарт в память начальную сумму пробивают впереди чисел, подлежащих вводу. В прочих случаях обе суммы вычисляются автоматически. Для переноса чисел с перфокарт в память $a = 0000$, $0001 \leqslant c \leqslant 3777$; из памяти на перфокарты $0001 \leqslant a \leqslant 3777$, $c = 0000$; с магнитной ленты в память $m+1 \leqslant a \leqslant m+777$, $0001 \leqslant c \leqslant 3777$, где $m = 4000$ или 5000; из памяти на магнитную ленту $0001 \leqslant a \leqslant 3777$, $m+1 \leqslant c \leqslant m+777$, где $m = 4000$ или 5000; из одних ячеек памяти в другие $0001 \leqslant a \leqslant 3777$, $0001 \leqslant c \leqslant 3777$. При переносе чисел в память ячейка $c$ служит для получения суммы, а числа записываются в ячейки $c+1$, $c+2$, ..., $c+n$. Для некоторых экземпляров машины контрольная сумма располагается не впереди, а позади массива чисел. |

Продолжение

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Условие, при котором происходит переполнение разрядной сетки | Условие, при котором результат является машинным нулем | Действия, выполняемые машиной по команде |
|---|---|---|---|---|---|
| 20 *) | Условный переход первого типа: *a*, *b*, *c*; 20 | — | — | — | Если после выполнения предыдущей команды $\omega = 0$, то управление передается команде (*a*). Если же $\omega = 1$, то управление передается команде (*b*). Одновременно в ячейку *c* производится запись нуля. Если $c = n$, где $n$ — номер ячейки, хранящей описываемую команду условного перехода, то последняя сперва выполняется, а затем стирается и заменяется нулем. |
| 27 | Условный переход второго типа: *a*, *b*, *c*; 27 | — | — | — | Если после выполнения предыдущей команды $\omega = 0$, то управление передается команде (*a*). Если же $\omega = 1$, — то команде (*b*). Одновременно в ячейку *c* автоматически записывается команда возврата $n + 1$, $n + 1$, *c*; 20, где $n$ — номер ячейки, хранящей описываемую команду условного перехода. |

*) Под воздействием команд, приведенных выше, машина выполняет те или иные действия с числами. Команды 20, 27, 25, 40 и 26 являются командами управления, по которым передается управление, подводится нужная зона магнитной ленты или останавливается машина.

Продолжение

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Условие, при котором происходит переполнение разрядной сетки | Условие, при котором результат является машинным нулем | Действия, выполняемые машиной по команде |
|---|---|---|---|---|---|
| 25 | Подвод ленты: *a*, 0, 0; 25 | — | — | — | Подводится под считывающую головку зона *a* магнитной ленты. Эта команда выполняется одновременно со следующими за ней командами, не относящимися к магнитной ленте. Выполнение команды прерывается, если среди выполняемых команд встретится новая команда, относящаяся к магнитной ленте. |
| 40 | Останов: *a*, *b*, 0; 40 | — | — | — | Машина останавливается и выдает на пульт управления числа (*a*) и (*b*). |
| 26 | Сравнение и останов при несовпадении: *a*, *b*, *c*; 26 | Содержимое хотя бы одного разряда ячейки *c* не равно нулю | — | — | Команда отличается от команды с кодом операции 16 тем, что при $\omega = 1$ происходит останов с выдачей на пульт управления чисел (*a*) и (*b*). |

Продолжение

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Условие, при котором происходит переполнение разрядной сетки | Условие, при котором результат является машинным нулем | Действия, выполняемые машиной по команде |
|---|---|---|---|---|---|
| 30 *) | Предварительная команда: 0, $n$, 0; 30 **) | — | — | — | Групповая операция вида $a, b, c$; Θ. |
| 31 | Предварительная команда: 0, $n$, 0; 31 | — | — | — | Групповая операция вида $a$, $b$, [$c$]; Θ ***). |

*) Операции, выполняемые машиной над числами (сложение, сравнение, сдвиг и т. п.), могут быть сделаны групповыми. Для этого перед командой о выполнении одной из таких операций ставится *предварительная команда групповой операции*. Под воздействием предварительной команды групповой операции машина настраивается для работы в режиме групповой операции. Исполнительная команда (команда о выполнении операции над числами) переносится в специальный регистр и оттуда выполняется. Затем адреса исполнительной команды (один, два или все три в зависимости от кода предварительной команды) увеличиваются на единицу (в регистре), после чего команда снова выполняется, и т. д. Число, стоящее во втором адресе предварительной команды, указывает, сколько раз должна быть выполнена исполнительная команда: если во втором адресе предварительной команды стоит число $n$, то исполнительная команда будет выполнена $n + 1$ раз.

Запись как предварительной команды, так и исполнительной в ячейках памяти машины при выполнении групповой операции не изменяется.

**) Если в третьем адресе предварительной команды групповой операции вместо 0000 написать какой-либо другой номер ячейки памяти, то, кроме настройки машины на работу в режиме групповой операции, будет произведена запись нуля в упомянутую ячейку (говорят: будет произведена «очистка» ячейки).

***) Чтобы отметить те адреса исполнительной команды, которые преобразуются (в регистре) в процессе выполнения групповой операции, мы заключаем их в квадратные скобки. Если адрес исполнительной команды не заключен в квадратные скобки, то это значит, что при выполнении групповой операции он остается неизменным.

Продолжение

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Условие, при котором происходит переполнение разрядной сетки | Условие, при котором результат является машинным нулем | Действия, выполняемые машиной по команде |
|---|---|---|---|---|---|
| 32 | Предварительная команда: 0, $n$, 0; 32 | — | — | — | Групповая операция вида $a$, [$b$], $c$; Θ. |
| 33 | Предварительная команда: 0, $n$, 0; 33 | — | — | — | Групповая операция вида $a$, [$b$], [$c$]; Θ. |
| 34 | Предварительная команда: 0, $n$, 0; 34 | — | — | — | Групповая операция вида [$a$], $b$, $c$; Θ. |
| 35 | Предварительная команда: 0, $n$, 0; 35 | — | — | — | Групповая операция вида [$a$], $b$, [$c$]; Θ. |
| 36 | Предварительная команда: 0, $n$, 0; 36 | — | — | — | Групповая операция вида [$a$], [$b$], $c$; Θ. |
| 37 | Предварительная команда: 0, $n$, 0; 37 | — | — | — | Групповая операция вида [$a$], [$b$], [$c$]; Θ. |

**2. Пояснение некоторых команд.** Содержимое всякой ячейки памяти машины *Стрела* можно представить как в двоичном коде, так и в коде команд. Покажем на примерах способы перехода от одного из этих кодов к другому.

Рассмотрим команду

0231 0342 1215 0 12.

Двоичный код этой команды получим, заменяя каждую ее восьмеричную цифру соответствующей тройкой двоичных цифр:

000 010 011 001 000 011 100 010 001 010 001 101 0 001 010.

Обратно, если дан некоторый двоичный код

10101101100011101001011101100101101110 10010,

то для перевода его в код команд группируем двоичные цифры следующим образом:

101 011 011 000 111 010 010 111 011 001 011 011 1 010 010.

Заменяя тройки двоичных цифр соответствующими восьмеричными цифрами, получаем: 5330 7227 3133 1 22.

Предположим теперь, что дано число + 0,0072351. Получим его числовой код. Сперва нормализуем число: $+0{,}72351 \cdot 10^{-2}$. Затем с помощью девяти цифр запишем мантиссу и с помощью двух цифр — порядок: +723 510 000 –02. Это и есть числовой код.

Двоичный код получим, заменяя «+» цифрой 0, «–» — цифрой 1, каждую десятичную цифру мантиссы — двоично-десятичной тетрадой, а порядок — пятизначным двоичным числом. Получим:

0 0111 0010 0011 0101 0001 0000 0000 0000 0000 1 00010.

Группируем вышеописанным способом двоичные цифры:

001 110 010 001 101 010 001 000 000 000 000 000 0 100 010.

Код команд получим, заменяя тройки двоичных цифр соответствующими восьмеричными цифрами: 1621 5210 0000 0 42.

Сделав эти замечания, переходим к пояснениям некоторых команд, входящих в систему команд машины *Стрела.*

1. Поразрядное логическое умножение называют *выделением части,* так как с помощью этой операции можно выделить любую нужную нам часть набора двоичных цифр, изображающих число (говорят: любую часть числа). Для этого производят поразрядное логическое умножение между числом, из которого выделяется часть, и вспомогательным набором цифр, зависящим от того, какая часть должна быть выделена.

Например, если в ячейке № 0020 стоит команда

(0020) = 0125 0675 0115 0 01,

то ее третий адрес можно выделить с помощью поразрядного логического умножения числа (0020) на вспомогательное число

0000 0000 7777 0 00

(представляющее собой набор двоичных цифр

000 000 000 000 000 000 000 000 111 111 111 111 0 000 000).

Если это вспомогательное число хранится в ячейке № 0376 и результат должен быть записан в ячейку №0500, то команда для выделения третьего адреса из (0020)будет иметь следующий вид:

0020 0376 0500 0 11.

При этом в ячейку № 0500 записывается набор цифр

0000 0000 0115 0 00.

Если в ячейке № 0721 хранится некоторое число *N,* то путем поразрядного логического умножения числа *N* = (0721) на «число» 4000 0000 0000 0 00 (представляющее собой набор двоичных цифр

100 000 000 000 000 000 000 000 000 000 000 000 0 000 000)

выделяется знак числа *N.* Если этот набор записан в ячейку № 0100, то выделение знака получим по команде, например, такого вида:

0721 0100 0200 0 11.

При этом в ячейке № 0200 запишется нуль, если *N* положительно (будет выработан сигнал $\omega = 1$), или код 4000 0000 0000 0 00 (в двоичной системе имеет вид 10000...0), если *N* отрицательно (и будет выработан сигнал $\omega = 0$).

2. Поразрядное логическое сложение называют *формированием,* так как с помощью этой операции из частей двух чисел можно сформировать новое число. Например, если в ячейке № 100 хранится «число» 0625 0000 0000 0 00 (набор двоичных цифр

000 110 010 101 000 000 000 000 000 000 000 000 0 000 000),

а в ячейке № 101—«число» 0000 0250 0125 0 01 (набор двоичных цифр

000 000 000 000 000 010 101 000 000 001 010 101 0 000 001),

то с помощью команды 0100 0101 0102 0 13 получают в ячейке №0102 «число» 0625 0250 0125 0 01 (набор цифр

000 110 010.101 000 010 101 000 000 001 010 101 0 000 001),

представляющее собой в данном случае команду: «Сложить число (0625) с числом (0250), и результат записать в ячейку № 0125».

Формирование какого-либо числа с нулем дает в качестве результата операции само исходное число. Поэтому команду формирования часто применяют для переноса числа.

Например, по команде

0040 0000 0041 0 13

число (0040) запишется в ячейку № 0041.

3. *Сравнение.* Если в ячейке № 0047 записано двоичное число

011 101 110 000 010 000 000 111 000 101 011 111 0 010 100 = 3560 2007 0537 0 24 (в коде команд),

а в ячейке № 0201 — двоичное число

010 001 010 100 001 010 11O 011 001 011 110 101 1011 101 = 2124 1263 1365 1 35,

то по команде

0047 0201 0050 0 16

в ячейке № 0050 получится набор двоичных цифр

001 100 100 100 011 010 110 100 001 110 101 010 1 001 001 = 1444 3264 1652 1 11,

т. е. число, представляющее результат операции, будет иметь единицы в тех разрядах, которым соответствуют несовпавшие разряды у сравниваемых чисел. При этом будет выработан сигнал ω= 1, свидетельствующий о том, что сравниваемые числа не совпадают между собой.

Операция сравнения чаще всего употребляется не для получения численного результата (этот результат нужен при решении не математических, а логических задач), а ради получения того или иного значения сигнала *ω* для управления работой машины. Поэтому в третьем адресе команды с кодом 16 часто ставят нуль. Например,

0047 0201 0000 0 16.

При этом результат операции никуда не записывается, так как нулевая ячейка всегда хранит нуль и никакой записи не воспринимает.

4. *Сдвиг.* Сдвиг числа в разрядной сетке ячейки машины *Стрела* может выполняться по команде *Сдвиг по порядку,* или по команде *Сдвиг по адресу.*

Первая из этих команд имеет вид *a*, *b, c,* 14, а вторая может иметь один из четырех видов: *a,* 4000 + *k, c,* 14; *a,* 5000 + *k, c,* 14; *a,* 4100 + *k, c*, 14; *a,* 5100 + *k, c,* 14.

В команде сдвига по порядку второй адрес является номером ячейки памяти и, следовательно, всегда меньше восьмеричного числа 4000. В любой из четырех команд сдвига по адресу второй адрес больше числа 4000. Именно по этому признаку управляющее устройство машины «отличает» команду сдвига по адресу от команды сдвига по порядку.

Рассмотрим сперва сдвиг по порядку. В этом случае для сдвига числа, хранящегося в некоторой ячейке, используют в качестве вспомогательного числа либо специальный набор цифр, либо число, имеющее количественное значение, либо команду. Важно, чтобы изображение этого вспомогательного числа в ячейке машины имело в 36-м разряде 0 при необходимости сдвига влево, или 1 при необходимости сдвига вправо, а в разрядах с 37-го по 42-й — число, равное количеству разрядов, на которое производится сдвиг.

Например, пусть в ячейке № 100 находится число

111 111 000 000 000 000 000 000 000 000 001 111 1 111 111.

Требуется сдвинуть его вправо на десять разрядов и результат поместить в ячейку № 200. При этом в ячейке № 200 получится следующий результат:

000 000 000 011 111 100 000 000 000 000 000 000 0 000 001.

Поместим в ячейку, например, № 50 вспомогательное число

000 000 000 000 000 000 000 000 000 000 000 000 1 001 010.

Нужная нам операция производится машиной по команде

0100 0050 0200 0 14.

Тот же результат получился бы, если бы ячейка № 50 содержала число

011 111 000 000 000 000 000 000 000 000 000 000 1 001 010

или команду с контрольным знаком, равным единице, и кодом операции 12. Например,

0625 0000 0500 1 12.

Тот же результат можно было бы получить с помощью одной из следующих команд сдвига по адресу:

0100 4112 0200 0 14,

0100 5112 0200 0 14.

5. *Контрольное суммирование.* Операция номер 17 применяется при контроле правильности результатов в

процессе машинного счета. Подробнее об этом будет сказано в § 40. Покажем на примерах, как выполняется операция 17 над двоичными кодами содержимого ячеек машины *Стрела.* Операция 17 является аналогом рассмотренной нами в § 9 операции ⊞. Знак операции контрольного суммирования обозначим символом ⊞″.

Пример 1.

```
   010 010 111 001 011 101 011 011 000 111 101 111 1 001 111
⊞″
   100 110 110 110 110 111 100 001 110 101 011 001 0 000 000
   111 001 110 000 010 100 111 100 111 101 011 001 0 000 000
```

Пример 2.

```
   010 110 101 000 010 011 110 101 111 000 101 011 0 110 011
⊞″
   110 101 111 101 101 010 101 110 001 011 110 101 1 001 100
 1 001 100 100 101 111 110 100 100 000 100 100 000 1 111 111
 └-----------------------------------------------------▲
   001 100 100 101 111 110 100 100 000 100 100 001  0 000 000
```

В случае необходимости произвести контрольное суммирование ручным способом, удобнее пользоваться не двоичным кодом, а кодом команд. При этом нужно не забывать, что в коде команд контрольный знак является двоичной цифрой, а все остальные цифры — восьмеричные. Если в предыдущих двух примерах заменить двоичные коды кодами команд, то контрольное суммирование будет выполняться так, как показано в примерах 3 и 4.

Пример 3.

```
   2271 3533 0757 1 17
⊞″
   4666 6741 6551 0 61
   7160 2474 7531 0 00
```

Пример 4.

```
   2650 2365 7053 0 63
⊞″
   6575 5256 1365 1 14
1  1445 7644 0440 1 77
└-----------------▲
   1445 7644 0441 0 00
```

В дальнейшем, поясняя свойства контрольного суммирования, будем выполнять эту операцию над кодами команд. Числа

```
0000 0000 0000 0 00,
7777 7777 7777 1 77
```

по отношению к контрольному суммированию оба являются н у л я м и, т. е. выполняя операцию 17 над каким-либо числом *N* и одним из этих нулей, мы получим в результате *N.* В отношении первого из нулей это совершенно очевидно. Поэтому приведем пример только второго нуля.

Пример 5.

```
   4157 02I3 0621 1 23
⊞″
   7777 7777 7777 1 77
1 4157 0213 0621 1 22
└---------------------▲
   4157 0213 0621 1 23
```

Заметим, что если дан некоторый код команд *N,* то, выписывая дополнения его восьмеричных цифр до семи, а контрольного знака до единицы, получим новый код *N′* такой, что

$$N \boxplus'' N' = 7777\ 7777\ 7777\ 1\ 77.$$

Код *N′* будем называть *дополнением кода N.*

Пример 6. Пусть

$$N = 3251\ 6235\ 7012\ 0\ 61,$$
$$N' = 4526\ 1542\ 0765\ 1\ 16.$$

Легко видеть, что

$$N \boxplus'' N' = 7777\ 7777\ 7777\ 1\ 77.$$

6. *Условный переход первого типа.* Предположим, что в состав некоторой программы входят команды [*]

```
 ....  ....  ....  .... . ..
0025)  0100 0101  0200 0 01
0026)  0027 0040 0500 0 20
 ....  ....  ....  .... . ..
```

По команде № 25 машина выполнит операцию

$$(0100) + (0101) = (0200).$$

[*] Первое число, стоящее перед скобкой, означает номер ячейки, хранящей второе число, стоящее после скобки

Если число (0200) окажется нулем или положительным, то при сложении выработается сигнал $\omega = 0$ и в результате команды № 26 машина перейдет к выполнению команды № 27. Если число (0200) окажется отрицательным, то выработается сигнал $\omega = 1$ и в результате команды № 26 машина перейдет к выполнению команды № 40. В обоих случаях, кроме того, в ячейку № 500 будет записан нуль.

Если бы команда № 26 имела такой вид:

0026) 0027 0040 0000 0 20

(в третьем адресе — нуль), то запись нуля не производилась бы никуда. Если бы в третьем адресе команды условного перехода стоял номер ячейки, в которой хранится она сама:

0026) 0027 0040 0026 0 20,

то вместе с переходом был бы записан нуль на место этой команды, т. е. команда условного перехода сама бы себя, как говорят, погасила. Если бы первый и второй адреса команды № 26 были одинаковы, например

0026) = 0040 0040 0500 0 20,

то эта команда была бы командой безусловного перехода. Независимо от результата предыдущей операции машина переходила бы к выполнению команды № 40.

7. *Условный переход второго типа.* Передача управления при условном переходе второго типа производится по тому же правилу, что и при условном переходе первого типа. Но в ячейку, указанную в третьем адресе команды перехода, записывается не нуль, а команда возврата.

Например, если в ячейке № 52 находится команда

0052) 0100 0100 0120 0 27,

то при выполнении этой команды машина перейдет к выполнению команды, записанной в ячейке № 100, а в ячейку № 120 запишет команду

0120) 0053 0053 0120 0 20.

Дойдя до этой команды, машина вернется к тому месту программы, откуда был сделан переход второго типа, и одновременно сотрет в ячейке № 120 команду возврата.

8. *Специальное сложение* служит для преобразования команд — увеличения их адресов.

Пусть в ячейке № 100 хранится команда

0075 0133 0056 0 01,

а в ячейке № 200 — вспомогательное число

0001 0002 0003 0 00.

В результате выполнения команды

0100 0200 0101 0 02

в ячейке № 0101 будет записана команда

0076 0135 0061 0 01.

Если бы в ячейке № 0200 хранилось, например, число

0001 0002 0003 1 12,

то результат операции остался бы прежним («код» и «контрольный знак» вспомогательного числа в операции специального сложения не участвуют).

З а м е ч а н и е . Пусть $N = a, b, c;\ \theta$ — некоторая команда. Предположим, что с помощью однократного или многократного применения операции специального сложения адреса этой команды были увеличены соответственно на $\Delta a$, $\Delta b$ и $\Delta c$ единиц. Предположим, кроме того, что числа $\Delta a$, $\Delta b$ и $\Delta c$ — неотрицательны и что

$$a + \Delta a \le 7777, \qquad b + \Delta b \le 7777, \qquad c + \Delta c \le 7777.$$

Тогда в результате вышеописанной модификации команда $N$ примет такой вид:

$$M = a + \Delta a, \qquad b + \Delta b, \qquad c + \Delta c; \qquad \theta.$$

Если $N'$ — дополнение команды $N$, то, выполняя операцию 17 (контрольное суммирование) над $M$ и $N'$, получим следующее вспомогательное число: $K = \Delta a, \Delta b, \Delta c; 0$. Объясняется это тем, что при сформулированных выше условиях операции 02 и 17 дают одинаковые результаты и, значит,

$$M = N \boxplus'' K.$$

Отсюда (см. п. 5) получаем:

$$M \boxplus'' N' = N \boxplus'' K \boxplus'' N' = N \boxplus'' N' \boxplus'' K = 7777\ 7777\ 7777\ 1\ 77 \boxplus'' K = K.$$

П р и м е р 7.

$N =$ 0241 0032 5121 0 05,
$N' =$ 7536 7745 2656 1 72,
$\Delta a = 5, \quad \Delta b = 0, \quad \Delta c = 21,$
$M =$ 0246 0032 5142 0 05.

Выполняя операцию контрольного суммирования над числами $M$ и $N'$, получаем:

```
  0246 0032 5142 0 05
⊞''
  7536 7745 2656 1 72
1 0005 0000 0020 1 77
  └---------------▲
  0005 0000 0021 0 00
```

9. *Специальное вычитание,* как и специальное сложение, служит для преобразования команд.

Если в ячейке *a* содержится команда

1125 0223 0224 0 04,

а в ячейке *b* — вспомогательное число

0001 0001 0002 0 10,

то по команде *a, b, c,* 15 в ячейке *c* будет получен результат

1124 0222 0222 0 04.

Таким образом, специальное вычитание позволяет уменьшать адреса команд.

10. *Увеличение и уменьшение адресов команд.* Вследствие того, что при специальных сложении и вычитании нет переносов (или заемов) из адреса в адрес, можно с помощью каждой из этих операций как увеличивать, так и уменьшать адреса команд.

Если нужно с помощью специального сложения (вычитания) уменьшить (увеличить) адрес команды на положительное целое число *k,* то в качестве соответствующего адреса вспомогательного числа нужно взять восьмеричное число 10 000 – *k.*

Например, требуется первый адрес команды

0221 0344 0175 0 05

увеличить на одну единицу, а третий — уменьшить на две единицы. Это можно произвести с помощью специального сложения. Вспомогательное число, которое потребуется, будет:

0001 0000 7776 0 00.

Тот же результат получается от специального вычитания из нашей команды вспомогательного числа

7777 0000 0002 0 00.

В дальнейшем для вспомогательных чисел, применяемых при переадресации, используем следующие обозначения и наименования:

0001 0000 0000 0 00 = 1 (I) — единица первого адреса;
0000 0001 0000 0 00 = 1 (II) — единица второго адреса;
0000 0000 0001 0 00 = 1 (III) — единица третьего адреса;
0001 0001 0000 0 00 = 1 (I, II) — единица первого и второго адресов;
7777 0000 0000 0 00 = –1 (I) — отрицательная единица первого адреса и т. п.;

вообще

$$m\ n\ p\ 000 = m\ (\mathrm{I})\ n\ (\mathrm{II})\ \ p\ (\mathrm{III}),$$

то-есть *m* первого, *n* второго, *p* третьего адресов;

$$10\ 000 - m\ \ 10\ 000 - n\ \ 10\ 000 - p\ \ 0\ 00 = -m(\mathrm{I}) - n\ (\mathrm{II}) - p(\mathrm{III}),$$

то-есть минус *m* первого, минус *n* второго, минус *p* третьего адресов.

Например,

0003 0006 0001 0 00 = 3 (I) 6 (II) 1 (III);
0000 7774 0002 0 00 = —4 (II) 2 (III).

11. *Групповые операции.* Приведем примеры на применение групповых операций:

а) групповая операция с кодом предварительной команды 30 применяется только в. случае, когда один из первых адресов исполнительной команды совпадает с третьим.

Пример 8. Число *x* возвести в *n-ю* степень. Число *x* записываем в ячейку № 100, а единицу — в ячейку № 101. В ячейке № 101 получим $x^n$ в результате выполнения команд

0000 *n*–I 0000 0 30;
0100 0101 0101 0 05;

б) групповая операция с кодом предварительной команды 34 успешно применяется для вычисления суммы или произведения группы чисел, расположенных в ячейках с последовательными номерами.

Пример 9. Вычислить сумму чисел, записанных в ячейках с номерами 0101, 0102, ..., 0120. В ячейку № 0121 запишем нуль. Нужную нам сумму получим в ячейке № 0121 после выполнения команд

0000 0017 0000 0 34;
[0101] 0121 0121 0 01;

в) групповая операция с кодом предварительной команды 35 применяется для умножения каждого числа, принадлежащего группе чисел, записанных в ячейках с последовательными номерами, на одно и то же число, или для прибавления к каждому из них одного и того же числа.

Пример 10. К каждому из чисел, хранящихся в ячейках 0050, 0051,... ..., 0077, прибавить число (0100). Результаты записать на места исходных чисел. Это производится при помощи команд

0000 0027 0000 0 35;
[0050] 0100 [0050] 0 01;

г) с помощью групповой операции с кодом предварительной команды 37 можно вычислять попарные суммы или попарные произведения чисел, принадлежащих двум группам чисел, каждая из которых записана в ячейках с

последовательными номерами.

П р и м е р 11. Составить попарные произведения чисел, одна группа которых хранится в ячейках 0100, 0101, ..., 0177, а другая группа — в ячейках 0200, 0201, ..., 0277. С помощью команд

0000 0077 0000 0 37;
[0100] [0200] [0300] 0 05

получим требуемые попарные произведения в ячейках 0300, 0301,..., 0377.

З а м е ч а н и е . Для облегчения запоминания кодов групповых операций полезно следующее мнемоническое правило. Младшую восьмеричную цифру кода групповой операции запишем в виде тройки двоичных цифр. Старшую из них поставим в соответствие первому адресу исполнительной команды, следующую — второму адресу, а младшую — третьему адресу. Те адреса исполнительной команды, которым отвечают при этом единицы, в процессе выполнения групповой операции будут изменяться. Адреса, которым отвечают нули, будут оставаться неизменными.

Например, если код групповой операции равен 35, то переводя цифру 5 в двоичную систему, получим 101. Отсюда заключаем, что

0000 *n* 0000 0 35

есть предварительная команда групповой операции вида

[*a*], *b,* [*c*]; θ.

12. *Выделение целой части числа.* Выделение целой части числа выполняется с помощью операции сложения без округления младшего разряда.

Если число *N* хранится в ячейке *a,* а его целую часть нужно записать в ячейку *c*, то для этого применяется команда вида

*a* 7575 *c* 12.

Число 7575 является номером ячейки *УВК,* в которой хранится вспомогательное число, которое можно назвать *минус нулем* 43-го (восьмеричного) *порядка.* В коде команд

(7575) = 4000 0000 0000 0 43.

В двоичном коде

(7575) = 1 00000000000000000000000000000000 0 100 011,

то-есть в знаковом разряде содержится единица (минус), в цифровых разрядах — мантисса равна нулю, в разрядах порядка — число

+ 100 011 $_{(двоичное)}$ = +43 $_{(восьмеричное)}$ = +35 $_{(десятичное)}$.

Сложение без округления выполняется машиной так же, как обычное сложение (код операции 01), с той разницей, что при уравнении порядков не происходит округления мантиссы, т. е. те цифры мантиссы, которые при этом выходят за пределы разрядной сетки, теряются без какого-либо округления последней цифры из остающихся в разрядной сетке.

Покажем на примере, как с помощью описанной операции происходит выделение целой части. Предположим, что в ячейке *a* хранится двоичное число

(*a*) = –1011001, 1110001.

В двоичном коде имеем:

(*a*) = 1 10110011110001000000000000000000000 0 000 111..

Порядок числа (*a*) равен +111 (семь). Порядок числа (7575), как уже говорилось, равен +100011 (тридцать пять).

Начинается операция 12 с выравнивания порядков чисел (*a*) и (7575). При этом порядок (*a*) увеличивается до тридцати пяти, т. е. на двадцать восемь единиц, а мантисса одновременно сдвигается вправо на такое же количество разрядов. Получается (*a*) = = 1 00000000000000000000000001011001 0 100 011. При этом младшие цифры мантиссы (в нашем примере двадцать восемь цифр 111 000 10....0) выходят за пределы разрядной сетки и теряются,

Порядок числа (*a*) равен количеству двоичных цифр в его целой части. Из тридцати пяти цифр, изображающих мантиссу, теряется тридцать пять минус порядок, т. е. тридцать пять минус количество цифр целой части числа. Следовательно, остаются только цифры целой части.

Далее происходит сложение чисел (*a*) и (7575). Сумма будет равна числу (*a*), так как содержимое ячейки (7575) равно нулю (ибо его мантисса равна нулю). Затем происходит нормализация результата, т. е. его сдвиг влево, на столько разрядов, чтобы первая значащая цифра мантиссы оказалась в первом разряде ячейки. Другими словами, мантисса сдвигается влево ровно на столько разрядов, на сколько она перед этим сдвинулась вправо (в нашем примере на двадцать восемь разрядов). Одновременно порядок уменьшается на такое же количество единиц. Полученный результат равен целой части числа (*a*). Этот результат записывается в ячейку *c.*

В нашем примере

(*c*)=1 1011 001 00000000000000000000000000 0 000 11),

т. е. $c = -0{,}1011001 \cdot 10^{111} = -1011001$.

Заметим, что с помощью команды

*a,* 7575, *c;* 01

машина произведет округление числа (*a*), хранящегося в ячейке *a,* до ближайшего к нему целого числа. Читатель может убедиться в этом самостоятельно путем рассуждений, подобных только что приведенным. Нужно только помнить, что при уравнивании порядков будет происходить округление мантиссы (т. е. прибавление единицы к младшему из сохраняющихся разрядов, если старший из теряемых разрядов равен единице).

**3. Константы, хранящиеся в УВК.** В таблице 26 приводим часть констант, постоянно хранящихся в *УВК.*

Т а б л и ц а 26. Некоторые константы, хранящиеся в *УВК*

| Номер ячейки | Константа, представленная в коде команд | Пояснение |
|---|---|---|
| 7400 | 2000 0000 0000 0 01 | 1(единица) |
| 7423 | 0001 0000 0000 0 00 | 1(I) |
| 7424 | 0000 0001 0000 0 00 | 1(II) |
| 7425 | 0000 0000 0001 0 00 | 1(III) |
| 7426 | 0001 0001 0000 0 00 | 1 (I), 1 (II) |
| 7427 | 0000 0001 0001 0 00 | 1 (11), 1 (III) |
| 7430 | 0001 0000 0001 0 00 | 1 (I), 1 (III) |
| 7431 | 0001 0001 0001 0 00 | 1 (I), 1 (II), 1 (III) |
| 7434 | 7777 7777 7777 0 00 | Вспомогательное число для выделения адресной части команды. |
| 7435 | 3777 7777 7777 1 77 | Вспомогательное число для выделения абсолютной величины нормализованного числа. |
| 7436 | 0000 0000 0000 1 77 | Вспомогательное число для выделения контрольного знака и кода операции команды. |
| 7437 | 0000 0000 0000 0 77 | Вспомогательное число для выделения кода операции. |

П р о д о л ж е н и е

| Номер ячейки | Константа, представленная в коде команд | Пояснение |
|---|---|---|
| 7440 | 0000 0000 0000 1 00 | Вспомогательное число для выделения контрольного знака. |
| 7446 | 7777 0000 0000 0 00 | Вспомогательное число для выделения первого адреса. |
| 7447 | 0000 7777 0000 0 00 | Вспомогательное число для выделения второго адреса. |
| 7450 | 0000 0000 7777 0 00 | Вспомогательное число для выделения третьего адреса. |
| 7451 | 7777 7777 0000 0 00 | Вспомогательное число для выделения первого и второго адресов. |
| 7452 | 0000 7777 7777 0 00 | Вспомогательное число для выделения второго и третьего адресов. |
| 7453 | 7777 0000 7777 0 0 | Вспомогательное число для выделения первого и третьего адресов. |
| 7455 | 2000 0000 0000 0 02 | 2 |
| 7461 | 2000 0000 0000 0 00 | 1/2 |
| 7523 | 2650 1171 4640 0 01 | $\sqrt{2}$ |
| 7560 | 3110 3755 2420 0 01 | $\pi/2$ |
| 7575 | 4000 0000 0000 0 43 | Вспомогательное число для выделения целой части. |

## § 29. Сведения о машине *М-3*, необходимые для программирования

**1. Запись чисел в ячейках памяти.** Советская цифровая программно-управляемая машина *М-3* имеет запятую, фиксированную после знакового разряда. Каждая ячейка памяти машины *М-3* состоит из тридцати одного разряда. Будем считать эти разряды занумерованными слева направо (т. е. от старших разрядов к младшим) с помощью десятичных чисел 0, 1,2, ..., 30. Схема распределения разрядов ячейки при хранении в ней двоичного числа показана на рис. 99. Нулевой разряд является знаковым, разряды, начиная с первого и кончая тридцатым, — цифровыми.

В ячейку памяти машины *М-3* может быть введена правильная десятичная дробь. При этом каждая десятичная цифра изображается тетрадой (четверкой двоичных цифр), так что в ячейке десятичное число оказывается записанным в двоично-десятичной системе счисления. В ячейку может быть записана десятичная

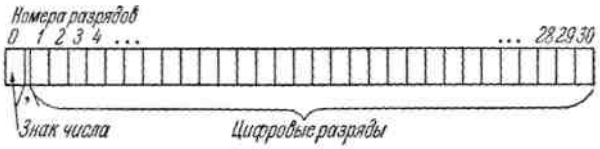



Рис. 99. Распределение разрядов ячейки памяти машины *М-3* при хранении двоичного числа.

дробь, имеющая семь знаков после запятой. Каждая тетрада требует четырех разрядов; таким образом, для записи всех семи тетрад требуется двадцать восемь разрядов. При этом два младших цифровых разряда ячейки остаются не использованными.

Схема распределения разрядов ячейки, при хранении в ней десятичного числа, приведена на рис. 100. Нулевой разряд является знаковым, разряды с первого по двадцать восьмой — цифровыми, двадцать девятый и тридцатый разряды не используются.

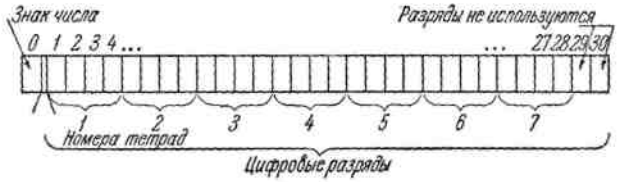



Рис. 100. Распределение разрядов ячейки памяти машины *М-3* при хранении десятичного числа.

Оперативная память машины *М-3* представляет собой магнитный барабан, вращающийся со скоростью 3000 оборотов в минуту, что позволяет машине выполнять приблизительно тридцать операций в секунду. Предусмотрена возможность замены магнитного барабана запоминающим устройством из ферритовых сердечников. Так как арифметическое устройство и устройство управления машины являются быстродействующими, то при указанной замене ее быстродействие возрастает приблизительно до полутора тысяч полных операций в секунду.

Память машины *М-3* состоит из 2048 (десятичное число) ячеек с номерами (в восьмеричной системе счисления) 0000, 0001, 0002,... ..., 3777.

В отличие от машины *Стрела* и *Урал* машина *М-3* не имеет ячейки, постоянно хранящей нуль. В любую ее ячейку может быть записано и из нее считано какое угодно число (из допустимого диапазона).

**2. Структура команд и их запись.** Машина *М-3* является двухадресной машиной и имеет естественный порядок выполнения команд.

При записи на бланке для программ команда может иметь, например, следующий вид:

01 0321 0652.

Такая команда означает: «От числа, стоящего в ячейке № 0321

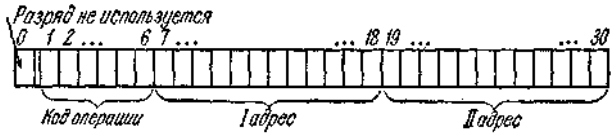



Рис. 101. Распределение разрядов ячейки памяти машины *М-3* при хранении команды.

отнять число, которое хранится в ячейке № 0652 (01 — код операции вычитания). Разность записать в ячейку №0652».

Число, записанное в ячейку памяти, хранится в этой ячейке до тех пор, пока в нее не будет произведена запись нового числа. При этом прежнее число автоматически стирается.

Как видно из приведенного выше примера команды, при записи на бланке на первом месте ставят код операции (в виде двузначного восьмеричного числа), а вслед за ним первый и второй адреса (каждый в виде четырехзначного восьмеричного числа). Таков же порядок расположения команды в ячейке памяти.

На рис. 101 приведена схема распределения разрядов ячейки при хранении в ней команды. Нулевой разряд не используется (его содержимое безразлично), разряды с первого по шестой отведены для кода операции, разряды с седьмого по восемнадцатый — для первого адреса команды, а с девятнадцатого по тридцатый — для второго адреса.

Форма бланка программы приведена на рис. 102. На этом рисунке для примера показана (на первой строке)

запись команды

21 0321 0621,

подлежащей размещению в ячейке памяти 0121, и (соответственно на второй и третьей строках) запись двух десятичных чисел:

+0,8913251 и —0,8025000. Из этого примера видно, что, записывая на бланке десятичные числа, разряд целых единиц и запятую опускают. Изображение вышеприведенных двух чисел при этом получается следующим: +8913251 и —8025000.

**3. Ввод программы в память машины и вывод результатов.** Для ввода в машину команды программы должны быть перенесены на бумажную ленту шириной 18 *мм* с помощью перфорирующего устройства, аналогичного применяемому на телеграфном аппарате телетайп. При этом пробивают номер ячейки, знак конца номера, затем команду, подлежащую вводу, и специальный знак, являющийся сигналом для записи команды в ячейку памяти. При отсутствии такого сигнала запись расположенной перед ним команды не производится. В конце всей программы пробивается особый знак, означающий конец ввода.

| № п/п | Код операции | I адрес | II адрес | Пояснения |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 0121 | 21 | 0321 | 0621 | |
| | + 8 | 913 | 251 | |
| | — 8 | 025 | 000 | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Рис. 102. Бланк программы для машины *М-3.*

Для того чтобы произвести ввод программы в машину, нажимают находящуюся на пульте управления кнопку с надписью «пуск». Бумажная лента с нанесенной на ней программой протягивается между системой контактов, замыкающихся при прохождении отверстий и посылающих сигналы, на основании которых происходит запись в ячейку. Предусмотрено также фоточитающее устройство, состоящее из источника света и системы фотоэлементов. Бумажная лента в таком устройстве движется между источником света и фотоэлементами. При прохождении между ними отверстий свет попадает на фотоэлементы, посылающие при этом электрические сигналы для записи чисел в ячейку. Наличие отверстия на соответствующем участке ленты означает единицу, отсутствие отверстия — нуль.

На ленте могут быть также записаны в виде некоторого массива десятичные числа без указания номеров ячеек, в которые каждое должно быть занесено. Перенос таких чисел в память (одного за другим, начиная с самого первого) может быть произведен путем включения в программу соответствующих команд. Результаты решения задач выводятся из памяти машины *М-3* путем печатания их на рулоне бумаги печатающим устройством той же системы, которая применяется в телеграфном аппарате телетайп.

Специального накопителя стандартных программ (*НСП*) или устройства выдачи констант (*УВК*) машина *М-3* не имеет. Стандартные программы, равно как и все константы, встречающиеся при решении задачи (в том числе и нуль), должны быть введены в ее память.

В составе арифметического устройства машина *М-3* имеет регистр (запоминающее устройство для хранения одного числа). При выполнении каждой операции ее результат запоминается в регистре. При выполнении следующей операции старое содержимое регистра стирается и на его место заносится новый результат. Кроме записи в регистр, результат операции может записываться в ячейку памяти, номер которой указан во втором адресе команды, или выдаваться на печать. Числа, имеющие количественное значение, хранятся на барабане в прямом коде, а в регистре — в обратном коде.

Алгебраическое сложение чисел производится путем сложения (операция $\boxplus$) на сумматоре обратных кодов этих чисел.

При выполнении операций сложения, вычитания, умножения, деления и поразрядного логического умножения в случае получения отрицательного результата операции («отрицательный нуль» тоже считается отрицательным результатом) вырабатывается сигнал $\omega = 1$. В остальных случаях считается, что $\omega = 0$. Точнее говоря, сигнал $\omega=1$ вырабатывается в тех случаях, когда после выполнения операции в знаковом разряде регистра оказывается знак минус.

При переполнении разрядной сетки сумматора происходит останов машины. Групповых операций машина

не производит.

**4. Система операций и команд машины.** Каждая команда машины *М-3* состоит из кода операции и двух адресов. Код операции в свою очередь состоит из двух восьмеричных цифр.

Обозначим через λ первую цифру кода операции, а через μ— вторую. Буквой *a* будем обозначать первый адрес команды, а буквой *b* — второй адрес команды. Тогда в общем виде можно изобразить команду так:

$$\lambda\mu;\ a,\ b.$$

Например, если λ = 1, μ = 3, *a* = 0215, *b* = 0321, то конкретный вид команды будет следующим:

$$13\ 0215\ 0321.$$

За некоторыми исключениями, которые будут описаны ниже, вид операции определяется второй цифрой кода операции. Значения цифры $\mu$ в зависимости от вида операции приведены в таблице 27.

Т а б л и ц а 27. Значение μ. в зависимости от вида операции

| Вид операции | μ |
|---|---|
| Сложение..... | 0 |
| Вычитание..... | 1 |
| Умножение..... | 3 |
| Деление……..... | 2 |
| Поразрядное логическое умножение……..... | 6 |

Поразрядное логическое умножение производится над содержимым одноименных разрядов двух ячеек (или ячейки и регистра), за исключением знаковых разрядов. В знаковом разряде результата получается тот же знак, что и в знаковом разряде ячейки, указанной во втором адресе команды. Первая цифра λ кода операции указывает на особенности этой операции.

Введем следующие обозначения: $(b)_0$ — содержимое ячейки *b* до выполнения операции; $(b)$ — содержимое ячейки *b* после выполнения операции; $(r)_0$—содержимое регистра до выполнения операции; $(r)$ — содержимое регистра после выполнения операции; $(a)$ — содержимое ячейки *a;* ∗ — знак, обозначающий операцию (например, ∗ означает +, если имеется в виду операция сложения, и т. п.); = результативный знак (знак равенства); этот знак пишется после формулы, обозначающей операцию, перед символом, обозначающим результат операции.

В таблице 28 приведены особенности операций и отвечающие им значения λ.

Например, желая перемножить числа $(a)$ и $(b)_0$ и результат записать в ячейку *b,* а также выдать на печать, мы с помощью таблиц 27 и 28 находим:

$$\mu = 3;\ \lambda = 4.$$

Таблица 28. Особенности выполнения операции и соответствующие значения λ

| Особенности выполнения операции | λ |
|---|---|
| $(a)*(b)_0=(r)=(b)$ | 0 |
| $(a)*(b)_0=(r)$, причем $(b)=(b)_0$ | 1 |
| $(r)_0*(a)=(r)=(b)$ | 2 |
| $(r)_0*(a)=(r)$, причем $(b)=(b)_0$ | 3 |
| $(a)*(b)_0=(r)=(b)$ Результат печатается на рулоне бумаги | 4 |
| $\|(a)\|*\|(b)\|=(r)$, причем $(b)=(b)_0$ | 5 |
| $(r)_0*(a)=(r)=(b)$ Результат печатается на рулоне бумаги | 6 |
| $\|(r)_0\|*\|(a)\|=(r)$, причем $(b)=(b)_0$ | 7 |

Необходимая команда имеет такой вид:

43 *a b.*

Если, кроме того, $a = 0325$, $b = 0112$, то получаем команду

43 0325 0112.

Кроме команд, получаемых с помощью таблиц 27 и 28, система команд машины *М-3* содержит еще команды, приведенные в таблице 29.

Т а б л и ц а 29. Команды управления машины *М-3*

| Код операции | Название операции и вид команды | Действия, выполняемые машиной по команде |
|---|---|---|
| 07<br>27 | Ввод числа с перфоленты:<br>07; 0 *b*<br>27; 0 *b* | Одно число с бумажной перфолнты переносится в ячейку памяти *b*. На регистре это число не фиксируется. |
| 05<br>15 | Перенос числа:<br>05 *ab*<br>15 *ab* | Число (*a*) переносится в ячейку *b.* |
| 45<br>55 | Перенос числа и его печатание:<br>45 *ab*<br>55 *ab* | Число (*a*) переносится в ячейку *b* и одновременно печатается. |
| 24 | Безусловный переход по первому адресу:<br>24 *ab* | Передача управления команде (*a*) и одновременный перенос $(r)_0$ в ячейку *b.* При этом $(r)=(r_0)$ |
| 64 | Безусловный переход по первому адресу и печать содержимого регистра:<br>64 *ab* | То же, что для кода 24 и выдача на печать содержимого регистра. |
| 74 | Безусловный переход по второму адресу:<br>74 0 *b* | Передача управления команде (*b*) и выполнение операции<br>$\|(r)_0\|=(r)$ |
| 34 | Условный переход:<br>34 *ab* | Передача управления при ω=1 команде (*a*), при ω=0 команде (*b*). Содержимое регистра не меняется |
| 37 | Останов:<br>37 *a* 0 | Машина останавливается. На пульт управления выдается (*a*). Содержимое регистра не изменяется. |

## § 30. Сведения о машине *Урал*, необходимые для программирования

**1. Запись чисел в ячейках памяти.** Отечественная электронная программно-управляемая машина *Урал* является машиной с запятой, фиксированной после знакового разряда.

Оперативная память этой машины состоит из вращающегося магнитного барабана и содержит две тысячи сорок восемь ячеек с восьмеричными номерами, начиная с 0000 и кончая 3777. Каждая ячейка имеет восемнадцать разрядов, которые считают занумерованными справа налево (т. е. от младших разрядов к старшим) с помощью десятичных чисел 1, 2, ..., 18. Такие ячейки называют *короткими* ячейками.

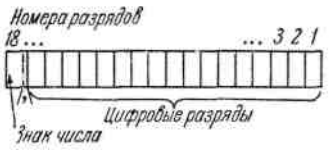



Рис. 103. Распределение разрядов короткой ячейки памяти машины *Урал* при хранении в ней двоичного числа.

Схема распределения разрядов короткой ячейки при хранении в ней двоичного числа приведена на рис. 103. Восемнадцатый разряд является знаковым, а разряды с первого по семнадцатый предназначены для хранения абсолютной величины числа. В короткой ячейке может быть записана правильная двоичная дробь с семнадцатью знаками после запятой.

В ряде случаев такое количество двоичных знаков после запятой является недостаточным. Поэтому конструкцией машины предусмотрено использование пар коротких ячеек в качестве так называемых *длинных* или *полных* ячеек. В полную ячейку могут быть объединены две последовательные короткие ячейки, первая из которых имеет четный номер. Например, из ячеек с номерами 0002 и 0003 можно образовать полную ячейку. Из ячеек с номерами 0000 и 0001 полная ячейка не может быть образована, что обусловлено конструктивными особенностями машины *Урал.* Как и *М-3, Урал* не имеет ячейки, постоянно хранящей нуль.

Для обозначения полной ячейки используют меньший из номеров, входящих в ее состав коротких ячеек, увеличенный на восьмеричное число 4000. Например, полные ячейки, образованные из пар коротких ячеек 0002, 0003 и 3752, 3753, обозначаются соответственно следующими числами: 4002 и 7752. Эти числа принято называть номерами полных ячеек. Таким образом, номер полной ячейки всегда больше восьмеричного числа 4000 и является четным числом.

Разряды полной ячейки считаются занумерованными справа налево с помощью десятичных чисел 1, 2,..., 36. При этом разряды от 1-го по 18-й совпадают с одноименными разрядами короткой ячейки (входящей в состав полной ячейки), имеющей нечетный номер. Разряды, начиная с 19-го и кончая 36-м, совпадают с разрядами, начиная с 1-го и кончая 18-м, короткой ячейки (входящей в состав полной ячейки), имеющей четный номер. Схема распределения разрядов полной ячейки при хранении в ней двоичного числа приведена на рис. 104. Разряд 36-й является знаковым, разряды с 1-го по 35-й предназначены для хранения абсолютной величины числа.

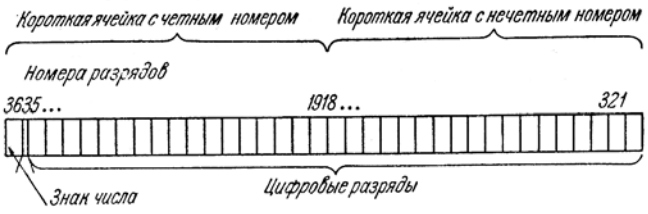



Рис. 104. Распределение разрядов полной ячейки памяти машины *Урал* при хранении двоичного числа.

Десятичное число можно ввести только в полную ячейку памяти машины *Урал.* Схема записи десятичного числа в полной ячейке приведена на рис. 105. В ячейку вводится правильная десятичная дробь, дробная часть которой представлена девятью (десятичными) цифрами. При этом тридцать шестой разряд ячейки служит для хранения знака числа, а остальные тридцать пять разрядов ячейки — для хранения цифр.

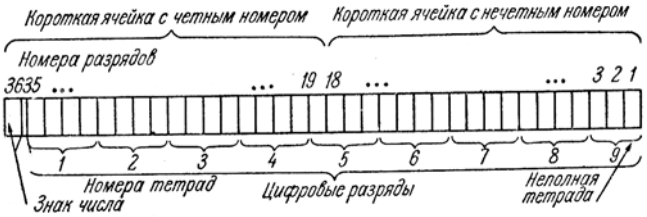



Рис. 105. Распределение разрядов полной ячейки памяти машины *Урал* при хранении десятичного числа.

Как известно, в двоично-десятичной записи каждая десятичная цифра представляется в виде четырех двоичных цифр (образующих тетрады). Однако для записи младшей, девятой, тетрады в ячейке остается всего только три разряда. Последний двоичный знак девятой тетрады введенного в ячейку десятичного числа оказывается утерянным. Тетрада 0001 как бы заменяется тетрадой 0000, тетрада 0011 —тетрадой 0010 и т. д., тетрада 1001 — тетрадой 1000. Вызванная этим погрешность незначительна.

Сумматор машины *Урал* является одновременно запоминающим устройством, способным хранить полученный результат операции. Кроме того, в состав арифметического устройства машины *Урал* входит регистр — запоминающее устройство для хранения одного числа. Как сумматор, так и регистр по количеству цифровых разрядов соответствуют полной ячейке памяти (т. е. имеют по тридцать пять цифровых разрядов). При выполнении команд, предусматривающих операции над числами (кроме посылки на барабан), не использующих прежнее содержимое регистра в качестве исходных данных, в начале рабочего такта содержимое регистра автоматически сбрасывается. Лучше всего заносить число в регистр с таким расчетом, чтобы в очередном такте работы машины это число было использовано для вычислений.

**2. Структура команд.** Машина *Урал* является одноадресной машиной и (поэтому) имеет естественный порядок выполнения команд.

На бланках для программ команды для машины *Урал* записываются в виде двух групп восьмеричных цифр. Первая группа состоит из двух цифр и является кодом операции. Вторая группа представляет собой четырехзначное число и является адресом.

Например, команда может иметь такой вид:

01 1155,

что означает: «Содержимое короткой ячейки № 1155 прибавить к

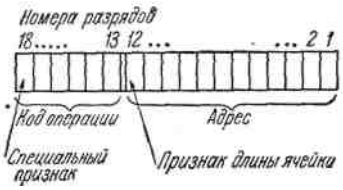


Рис. 106. Распределение разрядов короткой ячейки памяти машины *Урал* при хранении команды.

содержимому сумматора (01 — код операции сложения). Сумму оставить в сумматоре вместо второго слагаемого».

Если в команде предусмотрено действие над числом, хранящимся в полной ячейке, то ее адрес больше числа 4000 и является четным. Например, команда

01 5154

означает: «Содержимое полной ячейки, состоящей из коротких ячеек № 1154 и 1155, прибавить к содержимому сумматора. Сумму оставить в сумматоре».

Команды в памяти машины *Урал* хранятся в коротких ячейках. При этом разряды с первого по двенадцатый хранят адрес, а с тринадцатого по восемнадцатый (всего шесть разрядов) — отведены для кода операции (рис. 106).

Код операции не бывает больше восьмеричного числа 37 (см. систему команд в следующем параграфе), имеющего в двоичной системе счисления такой вид: 011 111. Таким образом, старший разряд ячейки (восемнадцатый) не нужен для записи кода операции. Содержимое этого разряда не влияет на содержание команды и используется для хранения специального признака. Если в восемнадцатом разряде ячейки стоит нуль, то код операции (при записи команды на бланке) записывают просто в виде двузначного восьмеричного числа. Если в восемнадцатом разряде ячейки стоит цифра 1, то перед кодом операции (при записи на бланке) ставят знак «—» (минус).

При этом команда может иметь такой вид:

—01 1155,

или

—01 5154.

Назначение признака, изображаемого знаком «—», стоящим перед кодом операции, будет объяснено далее.

Номер короткой ячейки машины *Урал* не превышает восьмеричного числа 3777, имеющего в двоичной записи такой вид:

011 111 111 111.

Таким образом, в старшем разряде (двенадцатом), хранящем адрес команды, оказывается цифра 0. Увеличение адреса на 4000, которое применяется для обозначения полной ячейки, сводится к тому, что в двенадцатый разряд записывается цифра 1.

Для управляющего устройства *Урала* признаком того, что операция должна быть выполнена над содержимым короткой ячейки, является наличие нуля в старшем двоичном разряде адреса. Признаком необходимости выполнения операции над числом, записанным в полной ячейке, является наличие единицы в этом разряде.

Бланки программы имеют форму, приведенную на рис. 107. На первых четырех строках бланка, изображенного на этом рисунке, для примера записаны команды

02 1155,
02 5154,
—02 1155,
—02 5154,

подлежащие вводу в ячейки памяти 0026, 0027, 0030 и 0031. На пятой и седьмой строках записаны соответственно десятичные числа

+ 0,23007

и

— 0,75 0000006,

подлежащие размещению в полных ячейках 4032 и 4034. Эти числа представлены там в таком виде:

+2300700000;

— 750000006.

При пробивке (на перфоленту) программ на входном перфораторе **в** графе бланка «Печать на клавишном устройстве» для контроля печатаются пробиваемые числа.

**3. Основные особенности машины.** Запоминающего устройства для постоянного хранения стандартных подпрограмм (*НСП*), а также устройства выдачи констант (*УВК*) машина *Урал* не имеет. Подпрограммы для вычисления значений часто встречающихся функций, а также все константы, необходимые при решении задачи, должны быть введены непосредственно в оперативную память машины (в том числе и 0).

| Пояснения | Номер ячейки | Число | | Печать на клавишном устройстве | Пояснения |
|---|---|---|---|---|---|
| | | Код операции | Адрес | | |
| | 0026 | 02 | 1155 | | |
| | 0027 | 02 | 5154 | | |
| | 0030 | -02 | 1155 | | |
| | 0031 | -02 | 515/, | | |
| | 4032 | + 230 | 070000 | | |
| | – 4034 – | -75 0 | 0000C6 | | |

Рис. 107. Бланк программы для машины *Урал.*

В качестве накопителя в машине *Урал* используется магнитная лента. Для ввода чисел в память применяется перфолента (а не перфокарты, как у *Стрелы).* Выдача результатов наружу тоже производится на перфоленту или непосредственно на печать. Хранятся числа во всех запоминающих устройствах *Урала* в прямом коде, кроме сумматора, в котором число изображается всегда в обратном модифицированном коде. Однако при операциях над числами, хранящимися в ячейках памяти, и содержимым сумматора различие кодов автоматически учитывается и при программировании приниматься в расчет не должно.

Машина снабжена на пульте управления восемью тумблерами, которые называются ключами. Каждый из ключей находится в одном из двух положений: включен или выключен. Ключам приписаны номера 0, 1, 2,..., 7. В системе команд для машины *Урал* содержится специальная команда обращения к ключу, адресом которой служит номер одного из ключей (см. ниже). Если во время выполнения этой команды ключ, упомянутый в ее адресе, выключен, следующая за этой командой очередная команда выполняется. В противном случае она пропускается. Включение ключей производится перед началом выполнения программы. Применяется оно с целью проверки правильности программы. Например, после команды обращения к ключу может стоять команда останова машины (аналогия с контрольным остановом *Стрелы)* или команда выдачи содержимого сумматора (промежуточного результата вычислений) на перфоленту и т. п. Как и в *Стреле,* при выполнении ряда операций арифметическое устройство *Урала* вырабатывает сигнал ω, равный нулю или единице в зависимости от результата операции. Значение сигнала ω используется для управления ходом вычислительного процесса с помощью команд условного перехода.

В большинстве случаев сигнал ω = 1 вырабатывается при отрицательном результате операции. Под отрицательным результатом мы понимаем также и «отрицательный нуль», получение которого возможно, так как в сумматоре результаты получаются в модифицированном обратном коде (напоминаем, что модифицированным обратным кодом «отрицательного нуля» является 11,1...1).

Если в результате какой-нибудь операции происходит переполнение разрядной сетки, то при включенном на пульте управления тумблере «блокировка переполнения» выполнение программы продолжается. Если же указанный тумблер не включен, то при включенном тумблере «останов по переполнению» машина останавливается, а при выключенном — пропускает одну команду (машины ранних выпусков передают управление команде (0001)).

Конструкция машины *Урал* значительно проще конструкции машины *Стрела.* Эта машина гораздо меньше по своим габаритам и дешевле. Скорость работы машины *Урал* значительно меньше, чем у *Стрелы.* За одну секунду *Урал* выполняет сто одноадресных команд против 2000 трехадресных команд *Стрелы* (каждая трехадресная команда соответствует приблизительно двум-трем одноадресным).

**4. Система операций и команд машины *Урал*.** В таблице 30 приведена система команд машины *Урал.*

В этой таблице приняты следующие обозначения: *a* или *b* — номер ячейки памяти; *r* — регистр; *s* — сумматор; (*a*), (*r*) — соответственно содержимое ячейки *a* или регистра *r*, $(s)_0$ — содержимое сумматора до выполнения операции; (*s*) — его содержимое после выполнения операции.

Т а б л и ц а 30. Система команд машины *Урал*

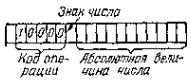
| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого ω = 1 | Действия, выполняемые машиной по команде |
|---|---|---|---|
| 01 | Сложение 1:<br>01*a* | $(s) \leqslant -0$ | К содержимому сумматора прибавляется (алгебраически) число (*a*) и результат записывается в сумматор:<br>$(s)_0 + (a) = (s)$. |
| 02 | Сложение 2:<br>02*a* | $(s) \leqslant -0$ | В сумматор засылается нуль, так что становится $(s)_0 = 0$, после чего выполняется операция $(s)_0 + (a) = (s)$. Переполнение разрядной сетки произойти не может. |
| 03 | Вычитание:<br>03*a* | $(s) \leqslant -0$ | Производится операция<br>$(s)_0 - (a) = (s)$. |
| 04 | Вычитание модулей:<br>04*a* | $(s) \leqslant -0$ | От абсолютной величины числа, записанного в сумматоре, отнимается абсолютная величина числа (*a*). Результат записывается в сумматор:<br>$\|(s)_0\| - \|(a)\| = (s)$.<br>Переполнение разрядной сетки произойти не может.<br>В первых образцах машины выполняется операция<br>$\|(r)\| - \|(a)\| = (s)$. |
| 05 | Умножение 1:<br>05*a* | $(s) \leqslant -0$ | Содержимое регистра умножается на число (*a*) и результат прибавляется к содержимому сумматора:<br>$(r) \cdot (a) + (s)_0 = (s)$. |
| 06 | Умножение 2:<br>06*a* | $(s) \leqslant -0$ | Производится операция<br>$(s)_0 \cdot (a) = (s)$. |
| 07 | Деление:<br>07*a* | $(s) \leqslant -0$ | Производится операция<br>$(s)_0 : (a) = (s)$.<br>Деление требует четырех рабочих тактов машины. |
| 10 | Формирование знака:<br>10*a* | $(s) \leqslant -0$ | Выполняется операция<br>$\text{sign}\,(a) \cdot \|(s)_0\| = (s)$,<br>где $\text{sign}\,(a)$ — знак числа (*a*). |

Продолжение

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Действия, выполняемые машиной по команде |
|---|---|---|---|
| 11 | Сдвиг: 11*a* | $(s) = 0$ | Содержимое регистра $(r)$ сдвигается на число разрядов, равное числу, записанному в разрядах сумматора, имеющих номера, начиная с девятнадцатого и кончая тридцать шестым. Содержимое остальных разрядов сумматора на результат операции не влияет. Сдвиг происходит влево, если $(s)_0 > 0$, и вправо, если $(s)_0 < 0$.<br>Результат записывается в сумматор. |
| 12 | Выделение части *): 12*a* | $(s) = 0$ | Число $(s)_0$ переводится в прямой код. Последний поразрядно логически умножается на $(a)$. Результат переводится в модифицированный обратный код и записывается в $s$. |
| 13 | Формирование *): 13*a* | $(s) = 0$ | Число $(s)_0$ переводится в прямой код. Последний поразрядно логически складывается с $(a)$. Результат переводится в модифицированный обратный код и записывается в $s$. |
| 14 | Сравнение *): 14*a* | $(s) \neq 0$ | Число $(s)_0$ переводится в прямой код. Между полученным результатом и числом $(a)$ производится поразрядно операция отрицания равнозначности. Результат переводится в модифицированный обратный код и записывается в $s$. |
| 15 | Нормализация: 15*a* | — | Число $(s)_0$ нормализуется, после чего мантисса записывается в ячейку $a$, а порядок — в старшую половину сумматора (т. е. в его разряды с девятнадцатого по тридцать шестой: знак попадает в тридцать шестой разряд, а младшая цифра — в девятнадцатый). |

*) Если $a$ — короткая ячейка, то ее номер должен быть четным. Это требование обусловлено конструкцией машины.

Продолжение

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Действия, выполняемые машиной по команде |
|---|---|---|---|
| 16 | Посылка на барабан (в память): 16*a* | Сохраняется предыдущее значение сигнала $\omega$ | Число $(s)_0$ пересылается в ячейку $a$. Содержимое сумматора сохраняется. |
| 17 | Посылка в регистр: 17*a* | $(r) = +0$ | Число $(a)$ переводится в прямой код и переносится в регистр. При этом $(s)_0 = (s)$. |
| 20 | Посылка адреса в сумматор: 20*k* | $(s) \leqslant -0$ | Число $k$, являющееся адресом команды, переносится в сумматор.<br>Число $k$ является одиннадцатиразрядным двоичным числом (его абсолютная величина не может превышать восьмеричного 3777). Распределение разрядов ячейки, хранящей команду 20*k*:<br>  <br><br>При переносе в сумматор знак числа $k$ записывается в тридцать шестой разряд сумматора, а абсолютная величина числа $k$ — в разряды с девятнадцатого по двадцать девятый. В остальные разряды сумматора записываются нули. |
| 21 | Условный переход: 21*a* | — | Если перед выполнением этой команды было $\omega = 0$, то управление передается очередной команде. Если же было $\omega = 1$, то управление получает команда $(a)$. Содержимое $r$ и $s$ не меняется. |
| 22 | Безусловный переход: 22*a* | — | Управление передается команде $(a)$. Содержимое $r$ и $s$ не меняется. |

Продолжение

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Действия, выполняемые машиной по команде |
|---|---|---|---|
| 23 | Обращение к ключу: 23*m* | Сохраняется предыдущее значение $\omega$ | *m* — номер ключа. Если ключ № *m* включен, то пропускается следующая команда программы. Если он выключен, то следующая команда программы выполняется. |
| 25 | Посылка в индексный регистр *): 25 *m*, где $m = 4000\,n' + n$ $(0 \leqslant m \leqslant 7777)$. | — | Число *n* записывается в индексный регистр *R*, а число *n'* — в специальный одноразрядный регистр $\rho$. |
| 24 | Условный переход по индексу *): 24*b* | — | При $(R) = 0$ происходит переход к следующей команде; при $(R) > 0$ содержимое индексного регистра уменьшается на $(\rho) + 1$ и управление передается команде (*b*). |
| 26 | Одиночное суммирование: 26*a* | — | Выполняется операция $(s)_0 \boxplus'' \boxplus'' (a) = (s)$ с циклическим переносом из 37-го (старшего знакового) разряда сумматора в первый. Служит для получения контрольных сумм. |
| 30 | Изменение команд: 30*a* | — | Ячейка *a* должна быть короткой. Обратный код ее содержимого складывается в регистре команд (с циклическим переносом) с содержимым ячейки, следующей за той, которая хранит эту команду. Получившаяся при этом команда выполняется в следующем такте вместо очередной команды программы. |
| 31 | Чтение с перфоленты: 31$a_{нач}$ 01*c* 00$a_{кон}$ | — | Из зоны № *c* перфоленты переписываются последовательно числа в ячейки памяти с номерами, начиная с $a_{нач}$ и кончая $a_{кон}$. |

*) См. пояснение, приведенное после таблицы.

Продолжение

| Номер и код операции | Название операции и вид команды | Условие, при выполнении которого $\omega = 1$ | Действия, выполняемые машиной по команде |
|---|---|---|---|
| 31 | Чтение с магнитной ленты: 31$a_{нач}$ 02*c* 00$a_{кон}$ | — | Из зоны № *c* магнитной ленты переписываются последовательно числа в ячейки памяти с номерами, начиная с $a_{нач}$ и кончая $a_{кон}$. Ячейки памяти, в которые производится запись, должны быть полными. |
| 31 | Запись на магнитную ленту: 31$a_{нач}$ 03*c* 00$a_{кон}$ | — | Из ячеек памяти с номерами от $a_{нач}$ до $a_{кон}$ числа переносятся в зону № *c* магнитной ленты. Ячейки памяти, из которых переносятся числа на магнитную ленту, должны быть полными. |
| 32 | Печать результата: 32 0000 | — | Печатается или перфорируется (в зависимости от положения на пульте управления переключателя «печать — перфорирование») число, модифицированный обратный код которого хранится в сумматоре. Печатание производится в десятичной системе счисления (в числовом коде) или в восьмеричной системе (в коде команд) в зависимости от положения тумблера, находящегося на пульте управления и помеченного надписью «Печать». |
| 34 | Пропуск интервала: 34 0000 | — | Бумажная лента печатающего устройства перемещается без печати на ней на один интервал. |
| 37 | Останов: 37*a* | — | Останов машины с выдачей на пульт управления числа (*a*) и одновременным переносом этого числа в сумматор. |

П о я с н е н и е к о м а н д 25*m* и 24*b*. Команда 25*m* ставится перед группой команд (но не обязательно непосредственно перед этой группой), которые должны быть выполнены многократно. Команда 24*b* должна быть поставлена непосредственно после упомянутой группы. Перед кодами операций команд, адреса которых при повторных выполнениях должны изменяться, ставят знак «—» (минус). Это соответствует специальному признаку — единице в восемнадцатом разряде короткой ячейки, содержащей команду. Все подлежащие такой модификации команды должны в качестве адресов иметь либо только номера коротких ячеек, либо только номера полных ячеек. Многократно выполняются все команды, стоящие перед командой 24*b*, начиная с команды, размещенной в ячейке *b*.

Если адресами модифицируемых команд являются номера коротких ячеек, то команда 25*m* имеет вид 25*n*, где *n* — целое число, не большее, чем 3777 (восьмеричное). В первый раз модифицируемые команды выполняются так, будто их адреса уменьшены на *n,* во второй раз —- так, будто их адреса уменьшены на *n* — 1, и т. д. В последний раз эти команды выполняются в том виде, в каком они записаны в программе. Таким образом, вся вышеупомянутая группа команд выполняется *n* + 1 раз.

Если адресами модифицируемых команд являются номера полных ячеек, то команда 25*m* имеет вид 25 4000 + *n,* где *n* — целое четное число, не превосходящее 3776 (восьмеричное). В первый раз модифицируемые команды выполняются так, будто их адреса уменьшены на *n,* во второй раз — так, будто их адреса уменьшены на *n* — 2, и т. д. В последний раз они выполняются в том виде, в котором записаны в программе. Таким образом, вся вышеупомянутая группа команд выполняется *n*/2+1 раз.

Опытный программист находит много других применений команд 25*m* и 24*b*. Для выполнения этих команд в машине *Урал* имеется так называемый индексный регистр, который мы обозначили буквой *R,* и специальный одноразрядный регистр, обозначенный у нас буквой *ρ*. Если содержимое индексного регистра отлично, от нуля, то каждая команда, имеющая знак «—» (минус) перед кодом операции, выполняется так, будто ее адрес уменьшен на содержимое индексного регистра.

З а м е ч а н и е . По команде 00 0000 машина совершает холостой такт и переходит к выполнению очередной команды.

# ГЛАВА VI
# ОСНОВЫ ПРОГРАММИРОВАНИЯ

## § 31. Непосредственное программирование

**1. Общие указания.** Как уже говорилось, электронная цифровая машина выполняет ту или иную операцию под воздействием соответствующей команды. Последовательность команд, при выполнении которых цифровая электронная машина осуществляет решение задачи, называется *программой.*

Процесс подготовки математической задачи для ее решения на цифровой электронной машине состоит из двух этапов. Первый этап заключается в выборе численного метода решения, метода оценки погрешности, которая будет получаться при решении, в составлении расчетных формул, т. е. в *арифметизации* задачи. Вторым этапом является *программирование*[*] .

Программирование в свою очередь состоит из трех частей:

1) определения порядка, в котором при решении задачи на машине выполняется вычислительный процесс;

2) размещения в запоминающих устройствах машины материала, относящегося к решению задачи (исходных данных задачи, команд, вспомогательных чисел, а также промежуточных и окончательных результатов, которые будут получаться);

3) составления команд и необходимых вспомогательных чисел.

Вторая и третья части программирования тесно связаны одна с другой. Не зная номеров ячеек, хранящих исходные данные и вспомогательные числа, не решив вопроса, в какие ячейки нужно помещать результаты, нельзя составлять команды программы. С другой стороны, не зная заранее количества команд программы, особенно при составлении длинных программ, трудно решить вопрос о размещении материала в запоминающих устройствах машины.

Эти трудности значительно уменьшаются, если не размещать сразу материал в ячейках с конкретными номерами, а использовать комбинированные буквенно-числовые обозначения. Например, можно условиться, что программа размещается в ячейках с номерами $a + 1$, $a + 2$, $a + 3$, ..., исходные данные задачи — в ячейках с номерами $b + 1$, $b + 2$, ...; вспомогательные числа по мере их появления при программировании размещаются в ячейках с номерами $e + 1$, $e + 2$, ... Рабочие ячейки, т. е. ячейки, применяемые для кратковременного хранения промежуточных результатов, обозначаются через $c + 1$, $c + 2$, ...; результаты решения заносятся в ячейки с номерами $d + 1$, $d + 2$, ... После составления программы буквам *a*, *b, c, d, e* можно придать конкретные значения и тем самым разместить материал в запоминающих устройствах машины.

**2. Пример составления программы.** Приведем несложный пример на составление программы для машины *Стрела.*

Составить программу для решения системы двух линейных уравнений с двумя неизвестными

$$\left.\begin{aligned} Ax + By &= C \\ Dx + Ey &= F \end{aligned}\right\}$$

при условии, что определитель этой системы отличен от нуля:

$$\begin{vmatrix} A & B \\ C & D \end{vmatrix} \neq 0$$

Нетрудно получить следующие формулы для решения нашей системы:

$$x = \frac{CE - BF}{AE - BD}, \qquad y = \frac{AF - CD}{AE - BD}.$$

Для вычисления по этим формулам и составим программу. Порядок, в котором в данном случае должен осуществляться вычислительный процесс, очевиден. Исходные данные задачи (числа *A, B, C, D, E, F)* разместим

[*] В дальнейшем методика программирования излагается применительно к машинам с естественным порядком выполнения команд. Все примеры программ, кроме приведенных в § 36, составлены для машины *Стрела,*

следующим образом:

$$b+1)\ A, \quad b+2)B, \quad b+3)C, \quad b+4)D, \quad b+5)E, \quad b+6)F.$$

Программу будем размещать в ячейках с номерами $a + 1$, $a + 2$,... (номера ячеек — восьмеричные числа). Рабочим ячейкам назначим номера $c + 1$, $c + 2$, ... Для результатов отведем ячейки $d + 1$ и $d + 2$.

Перейдем к составлению команд (содержимое рабочих ячеек после выполнения каждой команды показано в таблице 31).

Первая команда. «Перемножить числа $A$ и $E$; произведение записать в рабочую ячейку $c + 1$»:

$$a+1)\ \ b+1 \quad b+5 \quad c+1 \quad 05.$$

Вторая команда. «Перемножить числа $B$ и $D$; произведение записать в ячейку $c + 2$»:

$$a+2)\ \ b+2 \quad b+4 \quad c+2 \quad 05.$$

Третья команда. «От числа $AE = (c + 1)$ отнять число $BD = (c + 2)$; разность записать в ячейку $c + 1$»:

$$a+3)\ \ c+1 \quad c+2 \quad c+1 \quad 03.$$

Четвертая команда: «Вычислить величину, обратную числу $AE—BD = (c + 1)$; результат записать опять в ячейку $c + 1$»:

$$a+4)\ \ c+1 \quad 0000 \quad c+1 \quad 62.$$

Пятая команда. «Перемножить числа $C$ и $E$; произведение записать в ячейку $c + 2$»:

$$a+5)\ \ b+3 \quad b+5 \quad c+2 \quad 05.$$

Шестая команда. «Перемножить числа $B$ и $F$; произведение записать в ячейку $c + 3$».

$$a+6)\ \ b+2 \quad b+6 \quad c+3 \quad 05.$$

Седьмая команда. «От числа $CE=(c + 2)$ отнять число $BF = (c + 3)$; разность записать в ячейку с+2»:

$$a+7)\ \ c+2 \quad c+3 \quad c+2 \quad 03.$$

Восьмая команда. «Перемножить числа $(c+1)$ и $(c + 2)$; произведение записать в ячейку $d + 1$»:

$$a+10)\ \ c+1 \quad c+2 \quad d+1 \quad 05.$$

При выполнении этой команды будет получена искомая величина $x$.

Девятая команда. «Перемножить числа $A$ и $F$; произведение записать в ячейку $c + 2$»:

$$a+11)\ \ b+1 \quad b+6 \quad c+2 \quad 05.$$

Десятая команда. «Перемножить числа $C$ и $D$; произведение записать в ячейку $c + 3$»:

$$a+12)\ \ b+3 \quad b+4 \quad c+3 \quad 05.$$

Одиннадцатая команда. «От числа $AF = (c + 2)$ отнять число $CD = (c + 3)$; разность записать в ячейку $c + 2$».

$$a+13)\ \ c+2 \quad c+3 \quad c+2 \quad 03.$$

Двенадцатая команда. «Перемножить числа $(c + 1)$ и $(c + 2)$; произведение записать в ячейку $d+2$».

$$a+14)\ \ c+1 \quad c+2 \quad d+2 \quad 05.$$

При выполнении этой команды будет получена вторая искомая величина $y$.

Выполняя составленные нами команды, машина решит заданную систему уравнений. Но, как читатель помнит, все вычисления она производит в двоичной системе счисления. Следовательно, и величины $x$ и $y$ будут получены в виде двоичных чисел. Необходимо преобразовать их в десятичные числа.

Тринадцатая команда. «Два числа $(d + 1)$ и $(d + 2)$ перевести из двоичной системы счисления в десятичную; результаты записать соответственно в ячейки $d + 1$ и $d + 2$».

$$a+15)\ \ d+1 \quad 0001 \quad d+1 \quad 70.$$

Теперь предусмотрим вывод из машины полученных результатов.

Четырнадцатая команда: «Два числа $(d+1)$ и $(d+2)$ перенести на перфокарты».

$$a+16)\ \ d+1 \quad 0001 \quad 0000 \quad 44.$$

Пятнадцатая команда. «Останов»:

$$a+17)\ \ 0000 \quad 0000 \quad 0000 \quad 40.$$

Т а б л и ц а 31. Содержимое ячеек памяти после выполнения каждой команды

| После выполнения команды № | Содержимое | | | | |
|---|---|---|---|---|---|
| | рабочих ячеек | | | ответных ячеек | |
| | $c$+1 | $c$+2 | $c$+3 | $d$+1 | $d$+2 |
| $a$+1 | $AE$ | – | – | – | – |
| $a$+2 | $AE$ | $BD$ | – | – | – |
| $a$+3 | $AE-BD$ | $BD$ | – | – | – |
| $a$+4 | $\frac{1}{AE-BD}$ | $BD$ | – | – | – |
| $a$+5 | $\frac{1}{AE-BD}$ | $CE$ | – | – | – |
| $a$+6 | $\frac{1}{AE-BD}$ | $CE$ | $BF$ | – | – |
| $a$+7 | $\frac{1}{AE-BD}$ | $CE-BF$ | $BF$ | – | – |
| $a$+10 | $\frac{1}{AE-BD}$ | $CE-BF$ | $BF$ | $x=\frac{CE-BF}{AE-BD}$ | – |
| $a$+11 | $\frac{1}{AE-BD}$ | $AF$ | $BF$ | $x=\frac{CE-BF}{AE-BD}$ | – |
| $a$+12 | $\frac{1}{AE-BD}$ | $AF$ | $CD$ | $x=\frac{CE-BF}{AE-BD}$ | – |
| $a$+13 | $\frac{1}{AE-BD}$ | $AF-CD$ | $CD$ | $x=\frac{CE-BF}{AE-BD}$ | – |
| $a$+14 | $\frac{1}{AE-BD}$ | $AF-CD$ | $CD$ | $x=\frac{CE-BF}{AE-BD}$ | $y=\frac{AF-CD}{AE-BD}$ |

Очевидно, если мы не желаем до ввода в машину исходных данных задачи вручную переводить их в двоичную систему счисления, нашей программе необходимо еще предпослать команду: «Шесть десятичных чисел, первое из которых хранится в ячейке $b$+1, преобразовать в двоичные, и результаты записать в последовательные ячейки, начиная $b$ + 1».

$a$ + 0)  $b$ + 1  0005  $b$ + 1  72.

Теперь можно считать, что составление программы окончено. Собирая вместе все полученные команды, имеем программу в буквенном виде:

| | | | | |
|---|---|---|---|---|
| $a$+ 0) | $b$+ 1 | 0005 | $b$+ 1 | 72 |
| $a$+ 1) | $b$+ 1 | $b$+ 5 | $c$ + 1 | 05 |
| $a$+ 2) | $b$+ 2 | $b$+ 4 | $c$ + 2 | 05 |
| $a$+ 3) | $c$+ 1 | $c$+ 2 | $c$ + 1 | 03 |
| $a$+ 4) | $c$+ 1 | 0000 | $c$ + 1 | 62 |
| $a$+ 5) | $b$+ 3 | $b$+ 5 | $c$ + 2 | 05 |
| $a$+ 6) | $b$+ 2 | $b$+ 6 | $c$ + 3 | 05 |
| $a$+ 7) | $c$+ 2 | $c$+ 3 | $c$ + 2 | 03 |
| $a$+10) | $c$+ 1 | $c$+ 2 | $d$ + 1 | 05 |
| $a$+11) | $b$+ 1 | $b$+ 6 | $c$ + 2 | 05 |
| $a$+12) | $b$+ 3 | $b$+ 4 | $c$ + 3 | 05 |
| $a$+13) | $c$+ 2 | $c$+ 3 | $c$ + 2 | 03 |
| $a$+14) | $c$+ 1 | $c$+ 2 | $d$ + 2 | 05 |
| $a$+15) | $d$+ 1 | 0001 | $d$ + 1 | 70 |
| $a$+16) | $d$+ 1 | 0001 | 0000 | 44 |
| $a$+17) | 0000 | 0000 | 0000 | 40 |

Теперь остается заменить буквы конкретными числами и получить программу в ее окончательном виде.

Основную программу обычно располагают в ячейках, начиная с № 20, так как ячейки № 1, 2, 3, 4, 5 используются в качестве рабочих для стандартных подпрограмм (вычисляющих значения функций $\sqrt{x}$, $\ln x$, $\sin x$ и др.). Если в этих ячейках располагать команды основной программы, то они могут быть испорчены в случае наличия в программе команд о выполнении сложных операций. Это часто нежелательно даже при условии, что команды, хранившиеся в первых ячейках, уже выполнены (например, введя один раз программу, мы хотим с ее помощью решить несколько однотипных задач). Кроме того, ячейки, начиная с № 4 и кончая № 17, используют обычно для записи программы ввода. Поэтому положим $a$ = 20. Основная программа будет занимать ячейки, начиная с № 0020 и кончая № 0037. Далее положим $b$ = 37. Исходные данные разместятся в ячейках с № 40, 41, 42, 43, 44 и 45.

Таким образом, основная программа вместе с исходными данными займет ячейки, начиная с № 0020 и кончая № 0045, всего 26 (двадцать две ячейки)*.

Полагаем теперь $c$ = 45. Тогда рабочие ячейки будут иметь номера 46, 47 и 50.

Наша программа невелика. Нет никакого основания экономить емкость памяти. Поэтому не будем стараться разместить результаты счета в освобождающихся рабочих ячейках или ячейках, хранивших вначале исходные

* Нумерация ячеек и их подсчет ведутся в восьмеричной системе счисления.

данные. Положим $d = 50$. Тем самым для размещения результатов счета мы назначим ячейки № 51 и 52.

Итак, мы положили:

$$a = 20; \quad b = 37; \quad c = 45; \quad d = 50.$$

Теперь нетрудно переписать нашу программу в ее окончательном виде:

```
0020)  0040  0005  0040   0  72
0021)  0040  0044  0046   0  05
0022)  0041  0043  0047   0  05      0040)   A
0023)  0046  0047  0046   0  03      0041)   B
0024)  0046  0000  0046   0  62      0042)   C      Исходные данные
0025)  0042  0044  0047   0  05      0043)   D
0026)  0041  0045  0050   0  05      0044)   E
0027)  0047  0050  0047   0  03      0045)   F
0030)  0046  0047  0051   0  05      0046)
0031)  0040  0045  0047   0  05      0047)   Рабочие ячейки
0032)  0042  0043  0050   0  05      0050)
0033)  0047  0050  0047   0  03      0051)   Ответные ячейки
0034)  0046  0047  0052   0  05      0052)
0035)  0051  0001  0051   0  70
0036)  0051  0001  0000   0  44
0037)  0000  0000  0000   0  40
```

Для того чтобы выполнить эту программу, ее нужно ввести в память машины. Это можно сделать вручную, набирая на пульте управления каждую команду и вводя в соответствующую ячейку памяти. Но такой способ нецелесообразен. Он связан с внесением большого количества ошибок, что совершенно недопустимо, и требует большой затраты времени (особенно для ввода длинных программ). Ввод рабочей программы и исходных данных в машину осуществляется с помощью специальной программы ввода. В простейшем случае (которым мы пока что ограничимся) программа ввода может состоять из одной команды. В нашем случае это будет: «Ввести двадцать два числа с перфокарт в ячейки памяти, начиная с ячейки № 0020» (при этом во втором адресе команды должно стоять 0025, т. е. двадцать один);

0000 0025 0020 0 41.

Программа ввода наносится на отдельную перфокарту, которую кладут сверху колоды перфокарт, содержащих основную программу. При нажиме на пульте управления кнопки начального пуска машина захватывает первую перфокарту из числа лежащих в приемнике читающего устройства. Это должна быть перфокарта с программой ввода. Содержимое перфокарты вводится в ячейки памяти с номерами от 0004 до 0017. Затем машина последовательно выполняет команды, начиная с записанной в ячейке № 0004.

Команду о вводе основной программы мы поместим в первой строке перфокарты. Остальные строки перфокарты в нашем простейшем случае будут заполнены нулями. Наша программа ввода будет иметь следующий вид:

```
0004)  0000 0025 0020 0 41
0005)  0000 0000 0000 0 00
…..   ….   ….   ….   .  ..
0017) 0000 0000 0000 0 00
```

По первой ее команде машина ведет в память основную программу. Ячейки от № 0005 до 0017 будут машиной просмотрены без выполнения каких-либо операций, так как они заполнены нулями. Затем машина выполнит последовательно все команды рабочей программы (первая из которых записана в ячейке № 0020), выдаст на перфокарту ответ задачи и остановится.

Составленная нами программа вполне пригодна для счета по ней, но она имеет существенный недостаток. В ней не предусмотрен контроль правильности ввода программы, не предусмотрен также контроль правильности счета машины. Если в работе машины произойдет сбой и результаты счета будут ошибочными, мы об этом не сможем узнать, не прибегая к повторному решению задачи и сравнению между собой двух результатов. О методах автоматического контроля будет сообщено ниже.

**3. Разветвляющиеся программы.** При решении многих задач ход вычислительного процесса зависит от значений исходных данных или промежуточных результатов, что обусловлено зависимостью расчетных формул от этих величин. Получив исходные данные или соответствующие промежуточные результаты, сперва определяют направление хода вычислительного процесса (выбирают расчетные формулы) и лишь тогда производят вычисления. Такие вычислительные процессы называют *разветвляющимися.* Нередко встречается вычислительные процессы, разветвляющиеся в трех направлениях и более.

Разветвляющимся вычислительным процессам отвечают *разветвляющиеся программы.* Разветвление программы осуществляется при помощи двух команд. Первая из них должна быть такой, чтобы при ее выполнении вырабатывался сигнал ω, равный единице, если должна быть осуществлена одна из ветвей вычислительного процесса, и равный нулю при необходимости осуществления другой ветви, Вторая команда представляет собой команду условного перехода.

П р и м е р 1. Составим программу для вычисления определенного интеграла

$$\int_1^x t^n dt \qquad (x>0). \tag{VI.1}$$

Здесь исходными данными являются два числа: $x$ и $n$. При $n \neq -1$ значение интеграла вычисляется по формуле

$$\int_1^x t^n dt = \frac{x^{n+1}-1}{n+1} \tag{VI.2}$$

при $n = -1$ применяется другая формула

$$\int_1^x t^n dt = \ln x \tag{VI.3}$$

Исходные данные разместим следующим образом;

$b$+1) $x$, $b$ + 2) $n$, $b$ + 3) — 1.

Программу по-прежнему размещаем в ячейках с номерами $a$ + 1, $a$ + 2, ... В качестве рабочих ячеек берем ячейки с номерами $c$ + 1, $c$ + 2, ...

С о с т а в л е н и е п р о г р а м м ы .

1. Перевод исходных данных из десятичной системы в двоичную:

$a$ + 1) $b$ + 1 0002 $b$ + 1 72.

2. Определение направления вычислительного процесса:

$a$ + 2) $b$ + 2 $b$ + 3 0000 16;

$a$ + 3) $g$ + 1 $a$ + 4 0000 20.

При $n \neq -1$ управление передается команде ($a$ + 4), при $n = -1$ — команде ($g$+ 1).

3. Вычисление интеграла при $n \neq -1$ по формуле (VI.2):

| | | | | |
|---|---|---|---|---|
| $a$+4) | $b$+2 | $b$+3 | $c$+1 | 03 |
| $a$+5) | $b$+1 | 0000 | $c$+2 | 66 |
| $a$+6) | $c$+1 | $c$+2 | $c$+2 | 05 |
| $a$+7) | $c$+2 | 0000 | $c$+2 | 64 |
| $a$+10) | $c$+2 | $b$+3 | $c$+2 | 01 |
| $a$+11) | $c$+1 | 0000 | $c$+1 | 62 |
| $a$+12) | $c$+1 | $c$+2 | $d$+1 | 05 |
| $a$+13) | $h$+1 | $h$+1 | 0000 | 20 |

Содержимое ячеек памяти после выполнения каждой команды этого участка программы представлено таблицей 32.

Т а б л и ц а 32. Содержимое ячеек памяти при вычислении (VI.2)

| После выполнения команды № | Содержимое | | |
|---|---|---|---|
| | рабочих ячеек | | ответной ячейки |
| | $c$+1 | $c$+2 | $d$+1 |
| $a$+ 4 | $n$+1 | – | – |
| $a$+ 5 | $n$+1 | $\ln x$ | – |
| $a$+ 6 | $n$+1 | $(n+1)\ln x$ | – |
| $a$+ 7 | $n$+1 | $x^{n+1}$ | – |
| $a$+10 | $n$+1 | $x^{n+1}$–1 | – |
| $a$+11 | $\frac{1}{n+1}$ | $x^{n+1}$–1 | – |
| $a$+12 | $\frac{1}{n+1}$ | $x^{n+1}$–1 | $\frac{x^{n+1}-1}{n+1}$ |
| $a$+13 | $\frac{1}{n+1}$ | $x^{n+1}$–1 | $\frac{x^{n+1}-1}{n+1}$ |

Ответ мы помещаем в ячейку $d$ + 1. Команда ($a$ + 13) представляет собой безусловный переход к выполнению команды ($h$ + 1) о преобразовании двоичных чисел в десятичные.

4. Вычисление интеграла при $n = -1$ по формуле (VI.3):

$g$ + 1) $b$+1 0000 $d$+1 66.

5. Преобразование ответа в десятичное число (полагаем $g$ + 2 = $h$ + 1,т. е. $h$= $g$ + 1):

$h$ + 1) $d$+1 0000 $d$+1 70

6. Вывод ответа на перфокарту:

$h$ + 2) $d$ + 1 0000 0000 44.

7. Останов машины:

$h$ + 3) 0000 0000 0000 40.

Производим распределение памяти машины, для чего полагаем:

$a = 0017$, $g = a + 13 = 0017 + 13 = 0032$, $h = g + 1 = 0033$, $b = h + 3 = 0036$; $c = b + 3 = 0041$; $d = c + 2 = 0043$.

После замены букв числами, получаем следующую программу:

```
0020)   0037   0002   0037   0   72
0021)   0040   0041   0000   0   16   } Разветвление программы
0022)   0033   0023   0000   0   20   }

0023)   0040   0041   0042   0   03
0024)   0037   0000   0043   0   66
0025)   0042   0043   0043   0   05
0026)   0043   0000   0043   0   64   } 1-я ветвь программы
0027)   0043   0041   0043   0   01
0030)   0042   0000   0042   0   62
0031)   0042   0043   0044   0   05
0032)   0034   0034   0000   0   20

0033)   0037   0000   0044   0   66   } 2-я ветвь программы

0034)   0044   0000   0044   0   70
0035)   0044   0000   0000   0   44   } Общий конец обеих ветвей
0036)   0000   0000   0000   0   40

0037)   .....x.....
0040)   .....n.....                    } Исходные данные.
0041)   .....-1.....

0042)
0043)   } – Рабочие ячейки
0044)     – Ответ
```

Простейшая программа для ввода полученной рабочей программы:

```
0004)   0000   0021   0020   0   41
0005)   0000   0000   0000   0   00
.....
0017)   0000   0000   0000   0   00
```

П р и м е р 2. Составить программу для решения квадратного уравнения

$$px^2 + qx + r = 0. \tag{VI.4}$$

Здесь исходными данными являются числа $p$, $q$ и $r$. Решение уравнения осуществляется по общеизвестной формуле

$$x = \frac{-q \pm \sqrt{q^2 - 4pr}}{2p}. \tag{VI.5}$$

При решении этой задачи начало вычислительного процесса одинаково при всяких значениях $p$, $q$ и $r$. Оно состоит в вычислении подкоренного выражения $q^2 - 4pr$. Но дальнейший ход вычислительного процесса зависит от знака полученной величины (т. е. в конечном счете зависит от значений величин $p$, $q$, $r$). Про такие вычислительные процессы говорят, что их ход зависит от значений промежуточных результатов вычислений.

Исходные данные размещаем следующим образом:

$$b+1)\ p, \qquad b+2)\ q, \qquad b+3)\ r, \qquad b+4)\ 2.$$

Программу будем размещать в ячейках $a + 1$, $a + 2$, ... Рабочими ячейками пусть будут ячейки $c + 1$, $c + 2$,... Для ответов отведем ячейки $d + 1$, $d + 2$,... Содержимое рабочих ячеек после выполнения каждой команды показано в таблице 33.

С о с т а в л е н и е п р о г р а м м ы .

1. Вычисление величин $\frac{1}{2p}$, $q^2 - 4pr$ и $-q$:

```
a+ 0)   b + 1   0003    b + 1   72
a+ 1)   b + 1   b + 4   c + 1   05
a+ 2)   c + 1   b + 4   c + 2   05
a+ 3)   c + 2   b + 3   c + 2   05
a+ 4)   c + 1   0000    c + 1   62
a+ 5)   0000    b + 2   c + 3   03
a+ 6)   b + 2   b + 2   c + 4   05
a+ 7)   c + 4   c + 2   c + 2   03
```

2. Определение направления вычислительного процесса (разветвлениепрограммы):

```
a+10)   0000    c + 2   0000   01*;
a+11)   a+12    h + 1   0000   20.
```

При сложении числа 0 с числом $q^2 - 4pr$ вырабатывается сигнал $\omega = 0$, если результат (выражение $q^2 - 4pr$) положителен или равен нулю. При этом управление передается команде $(a + 12)$. Если же результат отрицателен, то вырабатывается сигнал $\omega = 1$ и управление получает команда $(h + 1)$.

---

* По существу, команда $(a + 10)$ является лишней. Точно такие же значения сигнала $\omega$ при том же знаке величины $q^2 - 4pr$ вырабатывает и команда $(a + 7)$. Следовательно, команда условного перехода может быть поставлена непосредственно после нее. Мы включили в программу команду $(a + 10)$ для того, чтобы отделить выбор дальнейшего хода вычислений от предшествовавшего ему вычисления величины $q^2 - 4pr$.

Если $q^2 - 4pr \geq 0$, то в качестве ответа мы получим два действительных числа $x_1$ и $x_2$. При $q^2 - 4pr < 0$ квадратное уравнение имеет комплексные сопряженные корни $\alpha + i\beta$ и $\alpha - i\beta$. Эти комплексные корни вполне определяются двумя действительными числами $\alpha$ и $\beta$. Таким образом, ответом нашей задачи всегда будут два действительных числа. Для того чтобы, получив ответ, знать, являются ли выданные машиной числа корнями квадратного уравнения $x_1$ и $x_2$, или действительной и мнимой частями $\alpha$ и $\beta$ корней, необходимо предусмотреть выдачу на перфокарты вместе с ответом задачи какого-либо признака. Например, можно условиться, что вместе с ответом задачи будет выдано число 2, если корни действительные, и число 4, если они комплексные. Примем это условие.

3. Первая ветвь программы (случай действительных корней):

а) занесение в ячейку $d + 1$ числа 2, являющегося признаком действительных корней:

$a$+12) $b$ + 4 0000 $d$+1 13;

б) вычисление корней:

| | | | | |
|---|---|---|---|---|
| $a$+13) | $c$+ 2 | 0000 | $c$+ 2 | 63; |
| $a$+14) | $c$+ 3 | $c$+ 2 | $c$+ 4 | 03; |
| $a$+15) | $c$+ 4 | $c$+ 1 | $d$+ 2 | 05; |
| $a$+16) | $c$+ 3 | $c$+ 2 | $c$+ 4 | 01; |
| $a$+17) | $c$+ 4 | $c$+ 1 | $d$+ 3 | 05; |
| $a$+20) | $g$+ 1 | $g$+ 1 | 0000 | 20. |

Последняя команда представляет собой команду безусловного перехода к команде о преобразовании ответа из двоичной системы счисления в десятичную систему.

4. Вторая ветвь программы (случай комплексных корней):

а) занесение в ячейку $d + 1$ числа 4, являющегося признаком комплексных корней:

| | | | | |
|---|---|---|---|---|
| $h$+ 1) | $b$+ 4 | $b$+ 4 | $d$+ 1 | 05; |

б) перемена знака у величины $q^2 - 4pr$ и вычисление действительной и мнимой частей корней уравнения:

| | | | | |
|---|---|---|---|---|
| $h$+ 2) | 0000 | $c$+ 2 | $c$ + 2 | 03; |
| $h$+ 3) | $c$+ 2 | 0000 | $c$ + 2 | 63; |
| $h$+ 4) | $c$+ 3 | $c$ + 1 | $d$ + 2 | 05; |
| $h$+ 5) | $c$+ 2 | $c$ + 1 | $d$ + 3 | 05. |

5. Перевод ответа в десятичную систему счисления, выдача его на перфокарту и останов машины (полагаем $h + 6 = g + 1$, т. е. $g = h + 5$):

| | | | | |
|---|---|---|---|---|
| $g$+ 1) | $d$+ 1 | 0002 | $d$ + 1 | 70; |
| $g$+ 2) | $d$+ 1 | 0002 | 0000 | 44; |
| $g$+ 3) | 0000 | 0000 | 0000 | 40. |

Составление программы окончено. Собираем вместе все полученные команды:

| | | | | | |
|---|---|---|---|---|---|
| $a$+0) | $b$+1 | 0003 | $b$+1 | 72 | |
| $a$+1) | $b$+1 | $b$+4 | $c$+1 | 05 | |
| $a$+2) | $c$+1 | $b$+4 | $c$+2 | 05 | |
| $a$+3) | $c$+2 | $b$+3 | $c$+2 | 05 | |
| $a$+4) | $c$+1 | 0000 | $c$+1 | 62 | |
| $a$+5) | 0000 | $b$+2 | $c$+3 | 03 | |
| $a$+6) | $b$+2 | $b$+2 | $c$+4 | 05 | |
| $a$+7) | $c$+4 | $c$+2 | $c$+2 | 03 | |
| $a$+10) | 0000 | $c$+2 | 0000 | 01 | Разветвление программы |
| $a$+11) | $a$+12 | $h$+1 | 0000 | 20 | |
| $a$+12) | $b$+4 | 0000 | $d$+1 | 13 | Первая ветвь программы |
| $a$+13) | $c$+2 | 0000 | $c$+2 | 63 | |
| $a$+14) | $c$+3 | $c$+2 | $c$+4 | 03 | |
| $a$+15) | $c$+4 | $c$+1 | $d$+2 | 05 | |
| $a$+16) | $c$+3 | $c$+2 | $c$+4 | 01 | |
| $a$+17) | $c$+4 | $c$+1 | $d$+3 | 05 | |
| $a$+20) | $g$+1 | $g$+1 | 0000 | 20 | |
| $h$+1) | $b$+4 | $b$+4 | $d$+1 | 13 | Вторая ветвь программы |
| $h$+2) | 0000 | $c$+2 | $c$+2 | 03 | |
| $h$+3) | $c$+2 | 0000 | $c$+2 | 63 | |
| $h$+4) | $c$+3 | $c$+1 | $d$+2 | 05 | |
| $h$+5) | $c$+2 | $c$+1 | $d$+3 | 05 | |
| $g$+1) | $d$+1 | 0002 | $d$+1 | 70 | Общий конец обеих ветвей |
| $g$+2) | $d$+1 | 0002 | 0000 | 44 | |
| $g$+3) | 0000 | 0000 | 0000 | 40 | |

Память машины распределим, полагая

| | | |
|---|---|---|
| $a = 0020$; | $h = a + 20 = 0040$; | $g = h + 5 = 0045$; |
| $b = g + 3 = 0050$; | $c = b + 4 = 0054$; | $d = c + 4 = 0060$. |

Заменить в программе буквы их численными значениями и составить простейшую программу ввода предоставляем самому читателю.

Таблица 33. Содержимое ячеек памяти после выполнения каждой команды

| № команды | Рабочих ячеек | | | | Ответных ячеек | | |
|---|---|---|---|---|---|---|---|
| | $c+1$ | $c+2$ | $c+3$ | $c+4$ | $d+1$ | $d+2$ | $d+3$ |
| $a+1$ | $2p$ | — | — | — | — | — | — |
| $a+2$ | $2p$ | $4p$ | — | — | — | — | — |
| $a+3$ | $2p$ | $4pr$ | — | — | — | — | — |
| $a+4$ | $\frac{1}{2p}$ | $4pr$ | — | — | — | — | — |
| $a+5$ | $\frac{1}{2p}$ | $4pr$ | $-q$ | — | — | — | — |
| $a+6$ | $\frac{1}{2p}$ | $4pr$ | $-q$ | $q^2$ | — | — | — |
| $a+7$ | $\frac{1}{2p}$ | $q^2-4pr$ | $-q$ | $q^2$ | — | — | — |
| $a+10$ | | | | | | | |
| $a+11$ *) | | | | | | | |
| $a+12$ | $\frac{1}{2p}$ | $q^2-4pr$ | $-q$ | $q^2$ | 2 | — | — |
| $a+13$ | $\frac{1}{2p}$ | $\sqrt{q^2-4pr}$ | $-q$ | $q^2$ | 2 | — | — |
| $a+14$ | $\frac{1}{2p}$ | $\sqrt{q^2-4pr}$ | $-q$ | $-q-\sqrt{q^2-4pr}$ | 2 | — | — |

*) После команды $(a+11)$ выполняется либо $(a+12)$, либо $(h+1)$.

Продолжение

| № команды | Рабочих ячеек | | | | Ответных ячеек | | |
|---|---|---|---|---|---|---|---|
| | $c+1$ | $c+2$ | $c+3$ | $c+4$ | $d+1$ | $d+2$ | $d+3$ |
| $a+15$ | $\frac{1}{2p}$ | $\sqrt{q^2-4pr}$ | $-q$ | $-q-\sqrt{q^2-4pr}$ | 2 | $x_1=\frac{-q-\sqrt{q^2-4pr}}{2p}$ | — |
| $a+16$ | $\frac{1}{2p}$ | $\sqrt{q^2-4pr}$ | $-q$ | $-q+\sqrt{q^2-4pr}$ | 2 | $x_1=\frac{-q-\sqrt{q^2-4pr}}{2p}$ | — |
| $a+17$ *) | $\frac{1}{2p}$ | $\sqrt{q^2-4pr}$ | $-q$ | $-q+\sqrt{q^2-4pr}$ | 2 | $x_1=\frac{-q-\sqrt{q^2-4pr}}{2p}$ | $x_2=\frac{-q+\sqrt{q^2-4pr}}{2p}$ |
| $h+1$ | $\frac{1}{2p}$ | $q^2-4pr$ | $-q$ | $q^2$ | 4 | — | — |
| $h+2$ | $\frac{1}{2p}$ | $4pr-q^2$ | $-q$ | $q^2$ | 4 | — | — |
| $h+3$ | $\frac{1}{2p}$ | $\sqrt{4pr-q^2}$ | $-q$ | $q^2$ | 4 | — | — |
| $h+4$ | $\frac{1}{2p}$ | $\sqrt{4pr-q^2}$ | $-q$ | $q^2$ | 4 | $\alpha=\frac{-q}{2p}$ | — |
| $h+5$ *) | $\frac{1}{2p}$ | $\sqrt{4pr-q^2}$ | $-q$ | $q^2$ | 4 | $\alpha=\frac{-q}{2p}$ | $\beta=\frac{\sqrt{4pr-q^2}}{2p}$ |

*) После этой команды выполняется команда $(g+1)$.

## § 32. Операторное программирование

**1. Логические схемы программ.** Уже в последних двух примерах предыдущего параграфа составление программы было связано с некоторыми трудностями. По мере усложнения задач, увеличения количества операций,

необходимых для их решения, усложнения вычислительного процесса трудности программирования резко возрастают. Естественно, что перед программистами возник вопрос, нельзя ли вычислительный процесс расчленить на простые части с тем, чтобы для каждой такой части составить свою программу, а программу для решения задачи получить соединением частных программ.

Для получения ответа на такой вопрос проанализируем процесс решения задач.

После того как задача арифметизирована, сведена к ряду расчетных формул, ее решение состоит из некоторой последовательности более или менее самостоятельных этапов. Так, в примере 1 § 31 первый этап заключается в проверке выполнения некоторого логического условия для выяснения вопроса о том, по каким формулам считать; второй этап состоит из арифметических вычислений. В примере 2 § 31 первый этап вычислительного процесса состоит из арифметических вычислений (величины $q^2 - 4pr$ и некоторых других величин); вторым этапом — логическим — является выяснение вопроса, как дальше продолжать вычисления; третий этап — снова арифметические вычисления.

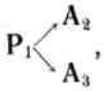
Таким образом, мы видим, что всякий вычислительный процесс может быть разбит на некоторое количество логических и арифметических этапов, число которых для более или менее сложных задач может быть довольно большим.

Этапы, на которые разделяется процесс решения задачи, мы будем называть *операторами.* Соответственно характеру этапов операторы могут быть *арифметическими* и *логическими.*

Разделив вычислительный процесс на операторы, составим для каждого оператора свою частную программу. Но для того чтобы частные программы были между собой согласованными, чтобы при их объединении получалась программа решения задачи, нужно каким-либо способом описать взаимосвязь операторов. Это можно сделать с помощью операторной схемы вычислительного процесса (известной под названием *схемы счета).*

Обозначим арифметические операторы буквой **А**, снабженной снизу номером ($\mathbf{A_6}$, $\mathbf{A_{10}}$ и т. п.). Логические операторы обозначим буквой **Р**, тоже снизу снабженной номером ($\mathbf{P_3}$, $\mathbf{P_{21}}$ и т. п.). Переход от одного оператора к другому обозначим стрелкой.

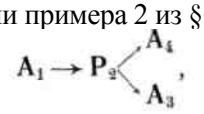
Теперь вычислительный процесс при решении задачи, приведенной в примере 1 § 31, может быть изображен следующим образом:



(VI.6)

где $\mathbf{P_1}$ — логический оператор, проверяющий выполнение условия $n \neq -1$. При выполнении этого условия происходит переход к оператору $\mathbf{A_2}$, ведущему счет по формуле (VI.2). При невыполнении условия работает оператор $\mathbf{A_3}$, считающий по формуле (VI.3).

Вычислительный процесс при решении задачи примера 2 из § 31 представляется схемой



(VI.7)

где $\mathbf{A_1}$— арифметический оператор, вычисляющий $q^2 - 4pr$, $-q$ и $\frac{1}{2p}$; $\mathbf{P_2}$— логический оператор, проверяющий выполнение условия $q^2 - 4pr < 0$). При выполнении этого условия происходит переход к арифметическому оператору $\mathbf{A_4}$, вычисляющему комплексные корни квадратного уравнения. При невыполнении условия работает оператор $\mathbf{A_3}$, вычисляющий действительные корни.

Эти схемы хорошо описывают характер вычислительных процессов, наглядно показывают, что они являются разветвляющимися.

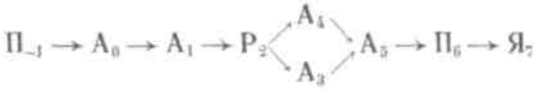
Полученные нами схемы решения остаются в силе независимо от того, какими вычислительными средствами осуществляется решение задачи. При ручном счете действия, подразумевающиеся под арифметическими операторами, могут быть выполнены на счетах или на арифмометре, или даже карандашом на бумаге. Действия, скрывающиеся за логическими операторами, производит в своей голове человек, решающий задачу.

При решении задачи на электронной цифровой универсальной машине все эти действия выполняет машина под воздействием команд введенной в нее программы.

Однако действий, представленных арифметическими и логическими операторами, недостаточно для того, чтобы решить задачу на электронной автоматической вычислительной машине. Вычислительному процессу, осуществляемому на такой машине, присущи еще некоторые этапы. К их числу относится, прежде всего, этап (соответственно оператор) переноса с перфокарт в память машины всего материала, относящегося к решению задачи. Затем этап (оператор) перевода десятичных чисел в двоичную систему счисления (его можно считать арифметическим оператором). Наконец, после того как задача решена, полученные результаты должны быть преобразованы в десятичную систему счисления (еще один арифметический оператор) и выданы на перфокарты (еще один оператор переноса), после чего машина должна быть остановлена (оператор останова).

Дополняя полученные выше схемы счета (VI.6) и (VI.7) операторами, наличие которых обусловлено спецификой вычислительного процесса, осуществляемого на электронной автоматической вычислительной машине, получим:

(VI.8)



(VI.9)

В этих схемах **П** обозначает операторы переноса, а **Я** — останов машины. В дальнейшем для более сложных вычислительных процессов схема программы еще более отличается от схемы счета.

Описанные здесь операторные схемы программ известны под названием *логических схем программ.*

**2. Свойства операторов**. Итак, с одной стороны, оператор — это некоторый этап вычислительного процесса; с другой стороны, оператор — это группа команд программы.

Являясь группой команд программы, оператор в то время, когда машина выполняет составляющие его команды, у п р а в л я е т работой машины.

Естественно, что частные программы операторов, т. е. группы команд, составляющих операторы, подчинены определенным условиям. Например, получать управление от других операторов может только одна команда каждого оператора. В противном случае стрелка в логической схеме не давала бы нам достаточных сведений для независимого составления частной программы каждого оператора.

Перечислим свойства оператора как группы команд.

В дальнейшем операторы будут подразделяться на элементарные и обобщенные.

*Элементарным оператором* называется группа команд программы, обладающая следующими свойствами:

1. Свойством *эффективности.* Элементарный оператор выполняет некоторые действия с числами, нужные для решения задачи на машине.

Под действиями с числами мы понимаем получение какого-либо числа или значения сигнала ω с помощью одного или нескольких чисел. В частности, перенос чисел из одного запоминающего устройства в другое или из одной части памяти в другую является действием с числами.

2. Свойством *упорядоченности.* Управление извне (от другого оператора) может получать лишь одна команда элементарного оператора — первая. Передача управления от команды к команде элементарного оператора происходит только в одном определенном порядке, как правило, в порядке номеров команд (номеров ячеек, хранящих команды). Управление вовне (к другим операторам) может передавать лишь одна команда элементарного оператора — последняя.

3. Свойством *связности.* Если первая команда элементарного оператора получила управление, то его получит в свое время и каждая другая команда элементарного оператора.

4. Свойством *автономности.* Условная передача управления элементарным оператором может происходить в зависимости от значения сигнала со, выработанного только самим этим элементарным оператором (а не другим).

5. Свойством *простоты.* Элементарный оператор должен выполнять наименьшее возможное количество видов работы. Мы всегда старались, чтобы оператор выполнял либо только арифметический счет, либо только проверку выполнения логического условия, или переадресацию и т.п. Но иногда, для того чтобы элементарный оператор выполнял только один определенный вид работы, приходится вводить в программу дополнительные команды, не нужные для получения решения задачи. Поэтому вместо требования к элементарному оператору выполнять только один определенный вид работы мы ставим требование выполнять наименьшее возможное количество видов работы.

Отдельные команды, поскольку они удовлетворяют перечисленным выше условиям, могут быть рассматриваемы как самостоятельные операторы. Однако команды [*]

*a b* 0000 20;

*a b* 0000 27;

*a a* 0000 20;

*a a* 0000 27

никогда не бывают самостоятельными операторами. Ни одна из них не обладает свойством эффективности, а первые две, кроме того, не автономны. Из-за неавтономности не являются операторами также команды

*a b c* 27

и

*a b c 20.*

Наоборот, иногда команду

*a a c* 20

и всегда команду

*a a c 27*

следует считать самостоятельными операторами. Первая из них является *оператором переноса* (нуля в ячейку *c*), а вторая — *оператором формирования* (она формирует в ячейке *c* команду возврата). В дальнейшем мы назовем ее из соображений удобства *обращением.*

Из числа команд, не удовлетворяющих вышеприведенным условиям, принято соглашение считать элементарными операторами команду *останова* (оператор **Я**) и команду *подвода зоны магнитной ленты* (обозначается символом **У**).

Разбиение программы на операторы не является однозначным. Например, арифметический оператор часто можно разделить на несколько арифметических операторов, а иногда даже каждую из его команд можно считать самостоятельным оператором. Но такое «измельчение» операторов нецелесообразно, так как оно приводит к удлинению логической схемы программы и лишает ее обозримости. Нужно стараться объединить в каждом операторе возможно большее количество команд.

Для наиболее часто встречающихся элементарных операторов приняты стандартные обозначения, приведенные в таблице 34.

[*] Здесь и всюду далее в книге, кроме специально оговоренных случаев, примеры приводятся для машины *Стрела.*

Т а б л и ц а 34. Некоторые операторы и их стандартные обозначения

| №п/п | Наименование оператора | Выполняемая работа | Обозначение |
|---|---|---|---|
| 1 | Арифметический оператор | Арифметический счет | **А** |
| 2 | Логический оператор | Проверка выполнения логического условия | **Р** |
| 3 | Оператор переноса | Перенос чисел из одного запоминающего устройства в другое или из одной части памяти в другую | **П** |
| 4 | Оператор переадресации (по параметру ***i***) | Переадресация команд | **F(*i*)** |
| 5 | Оператор восстановления (по параметру ***i***) | Восстановление команд путем приведения их к виду, соответствующему начальному значению параметра ***i*** | **О(*i*)** |
| 6 | Засылка | Вынесение величин в стандартные ячейки | **З** |
| 7 | Обратная засылка | Перенос последовательности значений величины из стандартной ячейки в последовательность ячеек | **З*** |
| 8 | Засылка команд | Занесение перед повторной работой оператора новых команд на места некоторых его команд | **К** |
| 9 | Обращение | Обращение к группе операторов с номерами *m*, *m*+1, …, *n* | **Е(*m*; *n*)** |
| 10 | Оператор циркуляции | Циркуляция чисел в стандартных ячейках | **Ц** |
| 11 | Оператор формирования | Формирование новых команд | Ф |
| 12 | Подвод ленты | Подвод зоны магнитной ленты | **У** |
| 13 | Останов | Останов машины | **Я** |
| 14 | Нестандартный оператор | Любой оператор, отличный от выше перечисленных | **Н** |

В этой таблице приведены некоторые операторы, с которыми читатель еще не встречался, но познакомится в дальнейшем.

**3. Правила начертания логических схем программ.** Для удобства записи логических схем операторы, образующие схему, располагают в одну строку. При этом приняты следующие правила:

1. Порядковый номер оператора в данной схеме изображается нижним индексом оператора. Нумерация операторов — сквозная.

2. Если оператор зависит от параметра, то этот параметр изображается верхним индексом оператора (например, $\mathbf{A}_2^{\mathbf{i}}$, $\boldsymbol{P}_{10}^{\mathbf{j}}$ и т. п.).

3. Если знаки двух операторов стоят в схеме рядом, то оператор, записанный слева, передает управление оператору, записанному справа,

4. Если между двумя записанными рядом знаками операторов стоит точка с запятой, то от оператора, записанного слева, нет передачи управления оператору, записанному справа.

5. Передача управления оператору, записанному не рядом справа, обозначается стрелкой. Например, полученные ранее схемы (VI.8) и (VI.9) будут выглядеть так:

$$\Pi_{-1}A_0P_1\overset{\frown}{A_2};\ A_3A_4\Pi_5Я_6$$

$$\Pi_{-1}A_0A_1P_2A_3;\ \overset{\frown}{A_4}A_5\Pi_6Я_7.$$

Для удобства записи горизонтальную часть стрелки можно опускать. При этом начало стрелки отмечается маленькой вертикальной стрелкой, направленной от оператора и снабженной номером оператора, которому передается управление, а конец стрелки—маленькой вертикальной стрелкой, направленной к оператору и помеченной номером оператора, от которого передается управление. Тогда схемы (VI.8) и (VI.9) примут такой вид:

$$\Pi_{-1}A_0P_1\overset{\downarrow 4}{A_2};\ A_3A_4\Pi_5Я_6,\quad \uparrow 2\ \ \downarrow 3\ \ \uparrow 1$$

$$\Pi_{-1}A_0A_1P_2A_3;\ \overset{\downarrow 4}{A_4}A_5\Pi_6Я_7.\quad \uparrow 2\ \ \downarrow 5\ \ \uparrow 3$$

6. Так как в машине при разветвлении программы передача управления не следующей по порядку команде осуществляется с помощью одной из операций условного перехода на основании значения сигнала ω, то удобно логические условия, проверку которых выполняют логические операторы, формулировать так, чтобы значение сигнала со было значением истинности высказывания *Условие выполнено.* Например, если для выбора направления вычислительного процесса производится сравнение чисел, то следует считать, что логический оператор проверяет истинность высказывания *Сравниваемые числа между собой не совпадают.* При начертании схем программ принято стрелки, отвечающие сигналу ω = 1, размещать над строкой операторов, а отвечающие сигналу ω = 0 — под строкой операторов; размещение стрелок, не связанных со значением сигнала ω, произвольно. В приведенных выше примерах это правило соблюдено.

Иногда вместо стрелок, указывающих передачу управления между операторами, применяют знаки, которые были введены сотрудником отдела прикладной математики АН СССР Ю. И. Яновым в разработанной им своеобразной «алгебре» операторов.

Именно, вместо стрелки, проходящей над строкой операторов, после оператора, передающего управление, ставится знак «$\overset{k}{\Gamma}$» (где *k*— номер оператора, которому передается управление). При этом перед оператором, получающим управление, должен быть поставлен знак «$\overset{l}{\urcorner}$» (где *l* — номер оператора,

передающего управление). Вместо $\underset{k}{\lfloor}$ стрелки, проходящей под строкой операторов, после оператора, передающего управление, ставится знак $\underset{k}{\lfloor}$ (где $k$ — номер оператора, получающего управление). При этом перед оператором, получающим управление, ставится знак $\underset{l}{\rfloor}$ (где $l$ — номер оператора, передающего управление). Например, логическая схема

$$A_1P_2A_3;\ A_4A_5\Pi_6$$

с помощью этих знаков записывается следующим образом:

$$A_1P_2\underset{4}{\lfloor}A_3\overset{5}{\lceil};\ \underset{2}{\rfloor}A_4\overset{3}{\rceil}A_5\Pi_6.$$

От логического оператора могут вести две стрелки (верхняя и нижняя), поэтому в логической схеме программы может встретиться сочетание символов $\underset{i}{\overset{k}{\lceil\lfloor}}$. Это сочетание обычно заменяют символом. $\underset{i}{\overset{k}{\lceil}}$ Например, возможна логическая схема

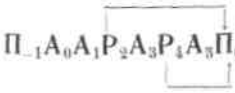
$$\Pi_{-1}A_0A_1P_2;\ A_3A_4A_5P_6A_7\Pi_{10}Я_{11},$$

которую можно записать следующим образом:

$$\Pi_{-1}A_0A_1P_2\underset{5}{\overset{4}{\lfloor}};\overset{6}{\rceil}A_3\overset{2}{\rceil}A_4\underset{2}{\rfloor}A_5P_6\overset{3}{\lceil}A_7\Pi_{10}Я_{11}.$$

Точно так же к оператору может вести несколько стрелок. Если к нему ведут одна верхняя и одна нижняя стрелки, перед ним может возникнуть сочетание символов $\overset{k}{\rceil\rfloor}$ или $\overset{k}{\rfloor\rceil}$. Вместо этого сочетания часто пишут один символ $\underset{i}{\overset{k}{\rfloor}}$. Например, логическая схема



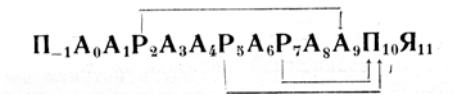


$$\Pi_{-1}A_0A_1P_2A_3P_4A_5\Pi_6Я_7$$

может быть записана в таком виде:

$$\Pi_{-1}A_0A_1P_2\overset{6}{\lceil}A_3P_4\underset{6}{\lfloor}A_5\underset{4}{\overset{2}{\rfloor}}\Pi_6Я_7.$$

Если к оператору ведет несколько верхних стрелок или несколько нижних стрелок, то перед таким оператором может возникнуть сочетание символов $\overset{i\ k\ l}{\rceil\rceil\rceil}$ или $\underset{i\ k\ l\ m}{\rfloor\rfloor\rfloor\rfloor}$ и т. п., которые обычно заменяют символами $\overset{i,k,l}{\rceil}$ или соответственно $\underset{i.k.l.m}{\rfloor}$ и т. п. Например, логическая схема



$$\Pi_{-1}A_0A_1P_2A_3A_4P_5A_6P_7A_8A_9\Pi_{10}Я_{11}$$

может быть записана так:

$$\Pi_{-1}A_0A_1P_2\overset{9}{\lceil}A_3A_4P_5\underset{10}{\lfloor}A_6P_7\underset{10}{\lfloor}A_8\overset{2}{\rceil}A_9\underset{5,7}{\rfloor}\Pi_{10}Я_{11}$$

Символы $\overset{i,k,l}{\rceil}$ и $\underset{m,n}{\rfloor}$ сливаются в символ $\overset{i,k,l}{\underset{m,n}{\rfloor}}$ и т.д.

Во многих случаях для упрощения логической схемы программы удобно объединить элементарные операторы в группу и обозначить ее одной буквой. Группа элементарных операторов может быть обозначена в логической схеме программы одной буквой только при условии, что получать управление извне (от операторов, не принадлежащих группе) может лишь один элементарный оператор группы. Такую группу элементарных операторов называют *обобщенным оператором.*

В состав обобщенного оператора могут входить элементарные операторы различных видов (различного функционального назначения). Мы в качестве примера приведем обобщенный логический оператор, представляющий собой группу элементарных операторов одного вида — логических (см. стр. 163, формула VI.16).

**4. Примеры операторного программирования.** Операторное программирование состоит в следующем. Процесс решения задачи расчленяется на этапы — арифметические и логические операторы. С помощью операторов

составляется схема решения задачи (схема счета). Эта схема дополняется операторами, свойственными машинному выполнению вычислительного процесса. Получается логическая схема программы. Для каждого оператора в порядке его записи в логической схеме составляется своя частная программа. Совокупность этих частных программ дает программу решения задачи.

Операторное программирование имеет большие преимущества перед непосредственным программированием:

1) облегчается работа по составлению программы;

2) значительно уменьшается количество ошибок при программировании;

3) наличие логической схемы программы и описания входящих в нее операторов облегчает проверку готовых программ;

4) появляется возможность возобновлять работу по составлению программы после длительного перерыва или поручать продолжение начатой работы другому программисту без необходимости для последнего читать ранее составленную часть программы.

Проиллюстрируем операторное программирование на примерах.

Пример 1. Составить логическую схему программы для вычисления действительного значения функции

$$y = \sqrt{x^2 + x - 10} + \ln(15 - x^2 - x) \qquad \text{(VI.10)}$$

по заданному значению независимой переменной $x$.

Вводим обозначение

$$z = x^2 + x, \qquad \text{(VI.11)}$$

после чего формула (VI. 10) принимает следующий вид:

$$y = \sqrt{z - 10} + \ln(15 - z) \qquad \text{(VI.12)}$$

Пусть, как обычно, оператор $\mathbf{П}_0$ вводит программу и исходные данные в память машины; $\mathbf{A}_1$ переводит исходные данные в двоичную систему счисления. Пусть оператор $\mathbf{A}_2$ вычисляет величину $z$ по формуле (VII. 11). Оператор $\mathbf{P}_3$ проверяет выполнение логического условия $\alpha_2$:

$$z - 10 < 0; \qquad \text{(VI.13)}$$

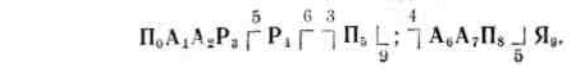
если это условие не выполнено, то $z - 10 \geq 0$ и корень $\sqrt{z-10}$ имеет действительное значение. Оператор $\mathbf{P}_4$ проверяет выполнение условия $\alpha_2$:

$$z - 15 < 0; \qquad \text{(VI.14)}$$

если это условие выполнено, то ln (15 — $z$) имеет действительное значение. Если корень и логарифм действительны, то оператор $\mathbf{A}_6$ вычисляет значение функции $y$. В противном случае оператор $\mathbf{П}_5$ выдает на перфокарту некоторый набор цифр, являющийся признаком того, что функция $y$ действительного значения не имеет. Оператор $\mathbf{A}_7$ переводит результат (если $y$ — действительное число) в десятичную систему счисления; $\mathbf{П}_8$ выдает его на перфокарту; оператор $\mathbf{Я}_9$ останавливает машину.

Логическая схема программы имеет такой вид:

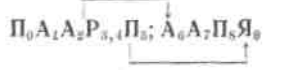
или



Группу элементарных логических операторов $\mathbf{P}_3$ и $\mathbf{P}_4$ можно обозначить одной буквой $\mathbf{P}_{3,4}$. Это будет логический обобщенный оператор, проверяющий выполнение сложного логического условия а:

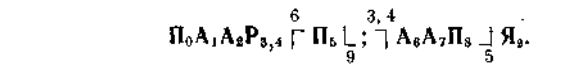
$$\alpha = \bar{\alpha}_1 \wedge \alpha_2 \qquad \text{(VI.15)}$$

При $\alpha = 1$ оператор $\mathbf{P}_{3,4}$ передает управление оператору $A_6$, при $\alpha = 0$ — оператору $\mathbf{П}_5$. Логическая схема программы примет такой вид:



(VI.16)

или



Пример 2. Составить программу для вычисления по заданному значению независимого переменного $x$ значения функции

$$\phi(x) = f(x^2 + x - 1), \qquad \text{(VI.17)}$$

где

$$f(y) = \begin{cases} y^5 - 1 & \text{при} \quad |y| \leqslant 1, \\ 2y - 1 & \text{при} \quad 1 < |y| \leqslant 2, \\ \frac{1}{8} y^5 - 1 & \text{при} \quad |y| > 2. \end{cases} \qquad \text{(VI.18)}$$

Легко видеть, что вычислительный процесс, который нам предстоит запрограммировать, разветвляется в трех направлениях. Программа тоже должна разветвляться в трех направлениях. Но так как машина *Стрела* позволяет производить разветвления только в двух направлениях (сигнал ω имеет только два значения), то разветвление в трех направлениях мы осуществим как два последовательных разветвления в двух направлениях каждое.

Разместим следующим образом исходные данные задачи:

$b + 1)$ $x$, $b + 3)$ 2,
$b + 2)$ 1, $b + 4)$ 1/8.

Составление логической схемы программы

Пусть $\mathbf{П}_0$ — оператор ввода программы; $\mathbf{A}_1$ — оператор, преобразующий исходные данные в двоичные числа; $\mathbf{A}_2$ — оператор, вычисляющий выражение $y = x^2 + x - 1$; $\mathbf{P}_3$ — оператор, проверяющий выполнение логического условия

$$1 - |y| < 0.$$

Если это условие не выполнено (т. е. если $1 - |y| \geq 0$, т е . $|y| \leq 1$), то после оператора $\mathbf{P}_3$ работает оператор $\mathbf{A}_4$, ведущий вычисления по верхней формуле из (VI.18). Если условие выполнено ( $|y| > 1$), то после $\mathbf{P}_3$ должен работать новый логический оператор $\mathbf{P}_5$,

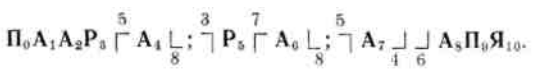
проверяющий выполнение логического условия

$$2 - |y| < 0.$$

Если последнее условие не выполнено (т. е. если, с одной стороны, как установил оператор $\mathbf{P}_3$, $|y| > 1$, а, с другой стороны, $|y| \leq 2$, что дает вместе $1 < |y| \leq 2$), то после $\mathbf{P}_5$ работает оператор $\mathbf{A}_6$, считающий по второй формуле из выражения (VI.18). Если же условие выполнено (т. е. $|y| > 2$), то после $\mathbf{P}_5$ работает $\mathbf{A}_7$, ведущий счет по третьей формуле из выражения (VI. 18). Наконец, после любого из операторов, $\mathbf{A}_4$, $\mathbf{A}_6$, $\mathbf{A}_7$, должен работать оператор $\mathbf{A}_8$, переводящий результаты вычисления из двоичной системы в десятичную, затем оператор $\mathbf{П}_9$, выдающий результат на перфокарту, и оператор $\mathbf{Я}_{10}$, останавливающий машину.

Логическая схема программы будет следующей:

или



Оператор $\mathbf{П}_0$ запрограммируем последним, после того как будет составлена остальная программа и сделано распределение памяти. Номера ячеек, хранящих команды каждого оператора, получающего управление не только от предыдущего по схеме (к такому оператору идут стрелки), будем обозначать с помощью новой буквы. Так, при программировании операторов $\mathbf{A}_1$ $\mathbf{A}_2$, $\mathbf{P}_3$, $\mathbf{A}_4$ используем ячейкиа $a + 1$, $a + 2$, ... Для операторов $\mathbf{P}_5$ и $\mathbf{A}_6$ отведем ячейки $h + 1$, $h + 2$,...; для $\mathbf{A}_7$ — ячейки $g + 1$, $g + 2$, ...; для $\mathbf{A}_8$, $\mathbf{П}_9$, $\mathbf{Я}_{10}$ — ячейки $k + 1$, $k + 2$, ... Рабочими ячейками у нас, как обычно, будут $c + 1$, $c + 2$, ...; ответной ячейкой — $d + 1$.

Составление программы

Оператор $\mathbf{A}_1$ (перевод исходных данных в двоичную систему):

$a + 1)$ $b + 1$ 0003 $b + 1$ 72.

Оператор $\mathbf{A}_2$ (вычисление величины $y = x^2 + x - 1$):

$a + 2)$ $b + 1$ $b + 1$ $c + 1$ 05;
$a + 3)$ $b + 1$ $c + 1$ $c + 1$ 01;
$a + 4)$ $c + 1$ $b + 2$ $c + 1$ 03.

Содержимое рабочей ячейки в процессе выполнения $\mathbf{A}_2$ представлено таблицей 35.

Оператор $\mathbf{P}_3$ (проверка выполнения условия $1 - |y| < 0$; это условие можно переписать следующим образом, не изменяя его смысла: $|1| - |y| < 0$, что позволит нам для его проверки применить операцию 04 — вычитание модулей; условие мы сформулировали так, что при его выполнении сигнал ω равен единице, а при невыполнении — нулю; разность модулей $|1| - |y|$ нам не нужна, поэтому в третьем адресе команды о вычитании модулей будет стоять нуль — в ячейку № 0 машина не производит никакой записи):

$a + 5)$ $b + 2$ $c + 1$ 0000 04;
$a + 6)$ $a + 7$ $h + 1$ 0000 20.

Оператор $\mathbf{A}_4$ (счет по верхней формуле из выражения (VI. 18):

$a + 7)$ $c + 1$ $c + 1$ $c + 2$ 05;
$a + 10)$ $c + 1$ $c + 2$ $c + 2$ 05;
$a + 11)$ $c + 2$ $b + 2$ $d + 1$ 03;
$a + 12)$ $k + 1$ $k + 1$ 0000 20.

По команде $(a + 12)$ машина производит безусловный переход к первой команде оператора $\mathbf{A}_8$, как того требует логическая схема программы.

Содержимое ячеек памяти в процессе выполнения $\mathbf{A}_4$ представлено таблицей 35.

Оператор $\mathbf{P}_5$ (проверка выполнения условия $2 - |y| < 0$ или $|2| - |y| < 0$):

$h + 1)$ $b + 3$ $c + 1$ 0000 04;
$h + 2)$ $h + 3$ $g + 1$ 0000 20.

Оператор $\mathbf{A}_6$ (счет по второй формуле из выражения (VI. 18):

$h + 3)$ $b + 3$ $c + 1$ $c + 1$ 05,
$h + 4)$ $c + 1$ $b + 2$ $d + 1$ 03,
$h + 5)$ $k + 1$ $k + 1$ 0000 20.

Таблица 35. Содержимое ячеек памяти в процессе выполнения операторов *)

| Оператор | № команды | Содержимое рабочих ячеек | | Содержимое ответной ячейки $d+1$ |
|---|---|---|---|---|
| | | $c+1$ | $c+2$ | |
| $\mathbf{A}_2$ | $a+2$ | $x^2$ | – | – |
| | $a+3$ | $x^2 + x$ | – | – |
| | $a+4$ | $y = x^2 + x - 1$ | – | – |
| $\mathbf{A}_4$ | $a+7$ | $y$ | $y^2$ | – |
| | $a+10$ | $y$ | $y^3$ | – |
| | $a+11$ | $y$ | $y^3$ | $\Phi(x) = y^3 - 1$ |
| | $a+12$ | $y$ | $y^3$ | $\Phi(x) = y^3 - 1$ |
| $\mathbf{A}_6$ | $h+3$ | $2y$ | $y^3$ | – |
| | $h+4$ | $2y$ | $y^3$ | $\Phi(x) = 2y - 1$ |
| | $h+5$ | $2y$ | $y^3$ | $\Phi(x) = 2y - 1$ |
| $\mathbf{A}_7$ | $g+1$ | $y$ | $y^2$ | – |
| | $g+2$ | $y$ | $y^4$ | – |
| | $g+3$ | $y$ | $y^5$ | – |
| | $g+4$ | $y$ | $y^5 : 8$ | – |
| | $g+5$ | $y$ | $y^5 : 8$ | $\Phi(x) = y^5 : 8 - 1$ |
| *)При чтении таблицы надо помнить, что операторы $\mathbf{A}_2$, $\mathbf{A}_4$, $\mathbf{A}_6$ и $\mathbf{A}_7$ могут выполняться либо в последовательности $\mathbf{A}_2$ $\mathbf{A}_4$, либо в последовательности $\mathbf{A}_2$ $\mathbf{A}_6$, либо в последовательности $\mathbf{A}_2$ $\mathbf{A}_7$ (см. логическую схему на стр. 316) | | | | |

Оператор $\mathbf{A}_7$ (счет по нижней формуле из выражения (VI. 18)):

$g+1)$ $c+1$ $c+1$ $c+2$ 05,
$g+2)$ $c+2$ $c+2$ $c+2$ 05,
$g+3)$ $c+1$ $c+2$ $c+2$ 05,
$g+4)$ $b+4$ $c+2$ $c+2$ 05,
$g+5)$ $c+2$ $b+2$ $d+1$ 03.

Содержимое ячеек памяти в процессе выполнения операторов $\mathbf{A}_6$ и $\mathbf{A}_7$ представлено таблицей 35.
Оператор $\mathbf{A}_8$ (перевод результата ($d+1$) в десятичную систему):

$k+1)$ $d+1$ 0000 $d+1$ 70.

Оператор $\mathbf{П}_9$ (выдача результата на перфокарту):

$k+2)$ $d+1$ 0000 0000 44.

Оператор $Я_{10}$ (останов машины):

$k+3)$ 0000 0000 0000 40.

В команде останова мы не используем возможности выдачи на пульт управления содержимого ячеек, номера которых указаны в первых двух адресах этой команды, так как в нашей задаче выдавать на пульт управления нечего.

Р а с п р е д е л е н и е п а м я т и
Полагаем: $a = 0017$; $h = a + 12 = 0031$; $g = h + 5 = 0036$; $k = g + 5 = 0043$; $b = k + 3 = 0046$; $c = b + 4 = 0052$; $d = c + 2 = 0054$.

Вводить в память машины нужно весь материал, начиная с команды (0020) и кончая числом (0052), т. е. всего 33 (двадцать семь) числа. После замены букв их числовыми значениями и составления простейшего оператора ввода программы получается программа, приведенная в таблице 36.

Т а б л и ц а 36. Программа вычисления значения функции, заданной равенствами (VI. 17) и (VI.18)

| № ячейки | Содержимое ячейки | Примечание |
|---|---|---|
| 0004<br>0005<br>.....<br>0017 | 0000 0032 0020 0 41<br>0000 0000 0000 0 00<br>.....<br>0000 0000 0000 0 00 | Оператор ввода $\mathbf{П}_0$ |
| 0020 | 0047 0003 0047 0 72 | Оператор $\mathbf{A}_1$ |
| 0021<br>0022<br>0023 | 0047 0047 0053 0 05<br>0047 0053 0053 0 01<br>0053 0050 0053 0 03 | Оператор $\mathbf{A}_2$ |
| 0024<br>0025 | 0053 0050 0000 0 04<br>0026 0032 0000 0 20 | Оператор $\mathbf{P}_3$ |
| 0026<br>0027<br>0030<br>0031 | 0053 0053 0054 0 05<br>0053 0054 0054 0 05<br>0054 0050 0055 0 03<br>0044 0044 0000 0 20 | Оператор $\mathbf{A}_4$ |
| 0032<br>0033 | 0051 0053 0000 0 04<br>0034 0037 0000 0 20 | Оператор $\mathbf{P}_3$ |
| 0034<br>0035<br>0036 | 0051 0053 0053 0 05<br>0053 0050 0055 0 03<br>0044 0044 0000 0 20 | Оператор $\mathbf{A}_6$ |
| 0037<br>0040<br>0041<br>0042<br>0043 | 0053 0053 0054 0 05<br>0054 0054 0054 0 05<br>0053 0054 0054 0 05<br>0052 0054 0054 0 05<br>0054 0050 0055 0 03 | Оператор $\mathbf{A}_7$ |
| 0044 | 0055 0000 0055 0 70 | Оператор $\mathbf{A}_8$ |
| 0045 | 0055 0000 0000 0 44 | Оператор $\mathbf{П}_9$ |
| 0046 | 0000 0000 0000 0 40 | Оператор $\mathbf{Я}_{10}$ |
| 0047<br>0050<br>0051<br>0052 | …… $x$ .....<br>…… 1 .....<br>…… 2 .....<br>……1/8 ..... | Исходные данные |
| 0053<br>0054 | | Рабочие ячейки |
| 0055 | | Ответная ячейка |

## § 33. Циклические программы

При решении многих задач вычислительный процесс имеет *циклический* характер. Это значит, что решение задачи состоит в многократном повторении вычислений по одним и тем же формулам с подстановкой в них каждый раз новых числовых значений вместо одной или нескольких букв. Многократно повторяющийся участок такого вычислительного процесса называется *циклом.*

Вот пример циклического вычислительного процесса. Требуется составить таблицу значений функции

$$f(x)=\frac{ax^2+bx+c}{\sqrt{x^2+1}}$$

для следующих значений *x:*

$$x_1;\ x_2;\ldots;\ x_n$$

Решение задачи состоит в *n*-кратном вычислении по формуле

$$f(x)=\frac{ax_i^2+bx_i+c}{\sqrt{x_i^2+1}};\quad i=1, 2, \ldots, n$$

Вычислительный процесс, с помощью которого будет составлена требуемая таблица, состоит из одного цикла, число выполнений которого заранее известно (оно равно *n*).

Но во многих вычислительных процессах количество выполнений цикла зависит от получаемых при вычислениях результатов и заранее неизвестно. Например, это имеет место при решении такой задачи.

Требуется решить методом итераций уравнение

$$x=f(x).$$

Приближенное значение искомого корня задано: $x = x_0$. Вычисления проводить до тех пор, пока абсолютное значение разности двух последовательных приближенных значений корня не станет меньше малого положительного наперед заданного числа ε.

Решение уравнения методом итераций состоит в последовательных вычислениях по формуле $x_{i+1} = f(x_i)$, где $i = 0, 1, 2,...$, и сравнений величины $|x_{i+1} — x_i|$ с числом ε.

Само собой разумеется, что уравнение решается методом итераций лишь при условии, что итерационный процесс будет сходящимся. Это не всегда имеет место. Одним из достаточных условий для сходимости итерационного процесса является выполнение в окрестности искомого корня (содержащей также и его начальное приближенное значение $x_0$) неравенства

$$|f'(x)|<1.$$

Для циклического вычислительного процесса, если количество выполнений каждого входящего в него цикла известно заранее, может быть составлена развернутая программа — серия команд о выполнении всех операций, входящих в состав вычислительного процесса. При этом каждому выполнению цикла будет отвечать свой участок программы. Развернутые программы циклических вычислительных процессов требуют большой затраты времени для своего составления и чрезвычайно громоздки. Зачастую они настолько велики, что не могут быть целиком введены в память машины. Их неудобства очевидны.

Для циклических вычислительных процессов с неизвестным заранее количеством выполнений циклов развернутую программу составить невозможно.

Для выполнения на машине циклических вычислительных процессов применяются *циклические программы. Циклом программы* называется ее участок, многократно используемый в процессе решения задачи, причем так, что после выполнения его последней команды управление (если к этому моменту еще не выполнено заданное условие) передается его первой команде. Каждому циклу вычислительного процесса отвечает цикл программы.

**1. Итерационный цикл.** В простейшем случае цикл программы содержит оператор переноса чисел, арифметический оператор и логический оператор, обеспечивающий повторное выполнение цикла. Ниже приводится операторная схема такого цикла:

или

(здесь стрелка, помеченная цифрой 1, означает получение циклом управления извне; стрелка, помеченная цифрой 2, означает передачу управления вовне).

Подобные циклы встречаются в программах решения задач об отыскании корня уравнения методом итераций и потому получили название *итерационных циклов.*

П р и м е р 1. Составить программу для решения методом итераций уравнения

$$1/65\ x^3 + x — 5{,}0001 = 0, \qquad \text{(VI.19)}$$

если приближенное значение искомого корня известно:

$$x_0 = 4$$

Вычисления продолжать-до тех пор, пока абсолютная величина разности двух последовательных приближенных значений корня не станет меньше наперед заданного числа $\omega = 0{,}0001$.

Преобразуя формулу (VI. 19) к виду

$$x = 5{,}001—1/65\ x^3, \qquad \text{(VI.20)}$$

легко убеждаемся, что вблизи от $x_0 = 4$ действительно существует корень уравнения и что производная от функции $5{,}001\text{-}1/65x^3$ в

окрестности корня по абсолютной величине меньше единицы. Итерационный процесс будет сходящимся. Решать уравнение (VI.20) будем по формуле

$$x_{i+1} = 5{,}001 - 1/65x_i^3 \quad \text{где} \quad i = 0,1,2,... \tag{VI.21}$$

до тех пор, пока не получим неравенства

$$|x_{i+1} - x_i| - |\varepsilon| < 0. \tag{VI.22}$$

Исходные данные задачи располагаем в ячейках памяти следующим образом:

$$b+1)\ x_0 = 4, \qquad b+2)\ 5{,}001, \qquad b+3)\ 1/65, \qquad b+4)\ \varepsilon = 0{,}0001.$$

Составление логической схемы программы

Пусть $\mathbf{П}_0$ — оператор ввода программы; $\mathbf{A}_1$ — оператор, переводящий исходные данные в двоичную систему счисления; $\mathbf{П}_2$ — оператор, переносящий содержимое ячейки $b + 1$ в рабочую ячейку $c + 1$; $\mathbf{A}_3$ ведет счет по формуле (VI.21) и результат записывает в ячейку $b + 1$; $\mathbf{P}_4$ проверяет выполнение условия (VI.22). При невыполнении этого условия управление передается оператору $\mathbf{П}_2$, при выполнении — задача решена и управление получает оператор $\mathbf{A}_5$, переводящий ответ (он будет получен в ячейке $b + 1$) в десятичную систему счисления. Затем $\mathbf{П}_6$ выдает результат на перфокарту, а $\mathbf{Я}_7$ останавливает машину. Логическая схема программы будет следующей:

или

(Читатель в ней без труда найдет итерационный цикл.)

Составление программы

Оператор $\mathbf{A}_1$:

| | | | | |
|---|---|---|---|---|
| $a+1)$ | $b+1$ | 0003 | $b+1$ | 72. |

Оператор $\mathbf{П}_2$:

| | | | | |
|---|---|---|---|---|
| $a+2)$ | $b+1$ | 0000 | $c+1$ | 13. |

Оператор $\mathbf{A}_3$:

| | | | | |
|---|---|---|---|---|
| $a+3)$ | $c+1$ | $c+1$ | $c+2$ | 05; |
| $a+4)$ | $c+1$ | $c+2$ | $c+2$ | 05; |
| $a+5)$ | $b+3$ | $c+2$ | $c+2$ | 05; |
| $a+6)$ | $b+2$ | $c+2$ | $b+1$ | 03. |

Оператор $\mathbf{P}_4$:

| | | | | |
|---|---|---|---|---|
| $a+7)$ | $b+1$ | $c+1$ | $c+1$ | 03; |
| $a+10)$ | $c+1$ | $b+4$ | 0000 | 04; |
| $a+11)$ | $a+2$ | $a+12$ | 0000 | 20. |

Оператор $\mathbf{A}_5$:

| | | | | |
|---|---|---|---|---|
| $a+12)$ | $b+1$ | 0000 | $b+1$ | 70. |

Оператор $П_6$:

| | | | | |
|---|---|---|---|---|
| $a+13)$ | $b+1$ | 0000 | 0000 | 44. |

Оператор $Я_7$:

| | | | | |
|---|---|---|---|---|
| $b+14)$ | 0000 | 0000 | 0000 | 40. |

Распределение памяти получим, полагая $a = 0017$, $b = a + 14 = 0033$ и $c = b + 4 = 0037$.

Составление простейшего оператора ввода и замену в программе букв их численными значениями предоставляем читателю.

**2. Цикл с переадресацией. В** примере 1 исходным было лишь одно значение величины $x$, а именно $x_0$. Цикл выполнялся столько раз, сколько требовалось для получения ответа с определенной степенью точности. Рассмотрим теперь задачу другого вида, решение которой тоже осуществляется при помощи циклического вычислительного процесса.

Пример 2. Составить программу для вычисления таблицы значений функции

$$y = x^5 + 2x + 3 \tag{VI.23}$$

для заданных возрастающих значений независимой переменной величины

Исходные данные разместим следующим образом:

$$b-3)\ x_n, \qquad b-2)\ 2, \qquad b-1)\ 3, \qquad b)\ 0, \qquad b+1)\ x_1 \quad b+2)\ x_2, ..., \qquad b+n-1)\ x_{n-1}, \qquad b+n)\ x_n.$$

Начальное содержимое ячейки $b$ в решении задачи не участвует. Для определенности мы записали в эту ячейку нуль.

Составление логической схемы

Пусть $\mathbf{П}_0$ — оператор ввода программы; $\boldsymbol{A}_1$ — оператор, переводящий исходные данные из десятичной системы в двоичную.

Далее, пусть оператор $\mathbf{П}_2$ производит групповой перенос чисел: из ячейки $b + 1$ в ячейку $b$, из ячейки $b + 2$ в ячейку $b + 1$ и так далее, наконец, из ячейки $b + n$ в ячейку $b + n$ — 1. Пусть оператор $\mathbf{A}_3$ берет число из ячейки $b$ вычисляет отвечающее ему значение функции по формуле (VI.23) и полученный результат записывает в ячейку $b + n$.

После первого выполнения операторов $\mathbf{П}_2$ и $\mathbf{A}_3$ содержимое ячеек $b$, $b + 1$,.., ..., $b + n$ примет следующий вид:

$$b)\ x_1 \quad b+1)\ x_2, \quad b+2)\ x_3, ...., \quad b+n-1)\ x_n, \quad b+n)\ y_1.$$

После n-кратного выполнения операторов $\mathbf{П}_2$ и $\mathbf{A}_3$ содержимое ячеек примет такой вид:

$$b)\ x_n, \quad b+1)\ y_1, \quad b+2)\ y_2, \ \ldots, \quad b+n-1)\ y_{n-1}, \quad b+n)\ y_n.$$

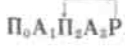
Таким образом, на местах заданных значений независимой переменной будут стоять искомые значения функции.

Оператор $\mathbf{P}_4$ будет сравнивать числа $(b)$ и $(b$ — 3), т. е. проверять выполнение условия

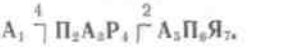
$$(b) \neq (b-3). \tag{VI.24}$$

Если условие (VI.24) будет выполнено, то будет происходить переход к оператору $\mathbf{П}_2$; если оно не будет выполнено (что произойдет после выполнения цикла, во время которого будет вычислена величина $y_n$), то управление будет передано оператору $\mathbf{A}_5$, переводящему результаты в десятичную систему счисления, после которого оператор $\mathbf{П}_6$ выдаст результаты на перфокарты, а оператор $\mathbf{Я}_7$ остановит машину.

Ниже приводится логическая схема программы:



или



Составление программы

Оператор $\mathbf{A}_1$:

$a+1)$ $b-3$ $n+3$ $b-3$ 72.

Оператор $\mathbf{П}_2$:

$a+2)$ $b+1$ $n-1$ $b$ 45.

Оператор $\mathbf{A}_3$:

$a+3)$ $b$ $b$ $c+1$ 05:
$a+4)$ $c+1$ $c+1$ $c+1$ 05;
$a+5)$ $b$ $c+1$ $c+1$ 05;
$a+6)$ $b-2$ $b$ $c+2$ 05;
$a+7)$ $c+1$ $c+2$ $c+1$ 01;
$a+10)$ $c+1$ $b-1$ $b+n$ 01.

Оператор $\mathbf{P}_4$:

$a+11)$ $b$ $b-3$ 0000 16;

$a+12)$ $a+13$ $a+2$ 0000 20.

Оператор $\mathbf{A}_5$:

$a+13)$ $b+1$ $n-1$ $b+1$ 70

Оператор $\mathbf{П}_6$:

$a+14)$ $b+1$ $n-1$ 0000 44.

Оператор $\mathbf{Я}_7$:

$a+15)$ 0000 0000 0000 40.

Распределение памяти получим, полагая

$a = 0017$; $b - 4 = a+15$, т. е. $b = a + 21 = 0040$; $c = b+ n = 0040 + n$

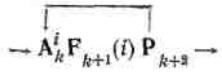
Недостаток составленной нами в примере 2 программы состоит в том, что при каждом выполнении цикла будет производиться ненужный для решения задачи перенос $n$ чисел, т. е. будут непроизводительно расходоваться $n$ тактов работы машины. Цикл будет выполняться $n$ раз, так что всего будет непроизводительно израсходовано $n^2$ рабочих тактов. На выполнение непроизводительных переносов будет расходоваться машинное время. Кроме того, вследствие увеличения числа тактов, потребных для решения задачи, возрастает вероятность ошибок за счет сбоев в работе машины. Наконец, более сложные задачи создают путаницу в переносе чисел, затрудняющую программирование.

Можно избавиться от необходимости переносов чисел, если оператор, ведущий счет по формуле (VI.23), будет брать исходные данные не из одной постоянной ячейки, а при каждом новом выполнении цикла из ячейки, в которой хранится новое значение независимой переменной. Для этого нужно изменять адреса ряда команд упомянутого оператора. Изменение адресов команд называется *переадресацией.* Таким образом, вместо оператора переноса в состав цикла включается новый оператор — *оператор переадресации.*

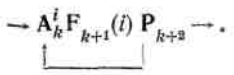
Обозначим его символом $\mathbf{F}(i)$, где буква $i$ обозначает номер выполнения данного оператора и называется *параметром,* отвечающим оператору переадресации.

Операторы, в состав которых входят команды, изменяющиеся при каждом выполнении $\mathbf{F}(i)$, называются *зависящими от параметра i.* Зависимость оператора от параметра, как уже говорилось в § 32, показывают с помощью значка, обозначающего этот параметр, поставленного справа наверху возле буквы, изображающей оператор (например, $\mathbf{A}_5^i$). В простейшем случае оператор $\mathbf{F}(i)$ выполняется при каждом выполнении цикла и, следовательно, номер выполнения оператора переадресации совпадает с номером выполнения цикла. При этом параметр $i$ называют *параметром цикла.*

Мы пришли к новому виду цикла, известному под названием *цикла с переадресацией.* Его логическая схема в простейшем случае такова:



или



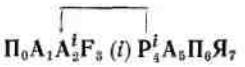
Пример 3. Составить программу, содержащую цикл с переадресацией, для задачи из примера 2. Размещение исходных данных:

$b-1)\ 2,\quad b)\ 3,\quad b+1)\ x_1,\ \ldots,\quad b+n)\ x_n,\quad b+n+1)\ \xi$

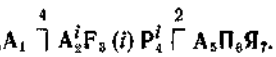
Здесь $\xi$ — любое число, удовлетворяющее условию

$$\xi \neq x_i, \qquad i = 1, 2 \ldots\ldots\ldots n.$$

Логическая схема программы



или



Оператор $\mathbf{П}_0$ вводит программу и исходные данные в память машины; $\mathbf{A}_1$ переводит исходные данные в двоичную систему счисления; $\mathbf{A}_2^i$ ведет счет по формуле (VI.23); $F_3$ (г) переадресует команды оператора $\mathbf{A}_2^i$; $\mathbf{P}_4^i$ управляет выполнениями цикла, а $\mathbf{A}_5$, $\mathbf{П}_6$, $\mathbf{Я}_7$ ничем не отличаются от одноименных операторов предыдущего примера.

Составление программы

Оператор $\mathbf{A}_1$:

$a+1)$ $b$—\ $n+2$ $b$—\ 72.

Оператор $\mathbf{A}_2^i$ (в этом и следующих операторах одной звездочкой отмечены адреса команд, зависящие от параметра $i$):

$a+2)$ $b+1^*$ $b+1^*$ $c+1$ 05;
$a+3)$ $c+1$ $c+1$ $c+1$ 05;
$a+4)$ $b+1^*$ $c+1$ $c+1$ 05;
$a+5)$ $b+1^*$ $b-1$ $c+2$ 05;

*a*+ 6) *c* + I *c* + 2 *c* + 1 01;
*a*+ 7) *c* + 1 *b* *b* + 1* 01.

Для переадресации команд (*a*+ 2), (*a* + 4), (*a* + 5) и (*a* + 7) потребуются вспомогательные числа 1 (I, II) 1 (I) и 1 (III). Такие числа имеются в датчике констант машины *Стрела.* Не приводя конкретные номера ячеек датчика констант, будем считать, что эти числа хранятся в ячейках следующим образом:

*e*+1) 1(I), *e* + 2) 1(I, II), *e* + 3) 1(III).

Оператор $\mathbf{F}_3$ (*i*):

*a*+10) *a* + 2 *e* + 2 *a* + 2 02;
*a*+11) *a* + 4 *e* + 1 *a* + 4 02;
*a*+12) *a* + 5 *e* + 1 *a* + 5 02;
*a*+13) *a* + 7 *e* + 3 *a* + 7 02.

Кроме того, в этот оператор войдет команда о переадресации одной из команд оператора $\mathbf{P}_4^i$:

*a*+14) *a*+15 *e* + 1 *a*+15 02.

Оператор $\mathbf{P}_4^i$ сравнивает содержимое переменной ячейки, из которой $\mathbf{A}_2^i$ будет брать значение независимой переменной при следующем выполнении цикла, с содержимым ячейки *b* + *n* + 1. Ввиду того, что команда о сравнении будет переадресована до своего первого действия, в ее первом адресе указана не ячейка *b* + 2, а ячейка *b*+1 (оператор $\mathbf{F}_3$ (*i*) стоит в логической схеме перед оператором $\mathbf{P}_4^i$).

Оператор $\mathbf{P}_4^i$:

*a*+15) *b*+1* *b* + *n* + 1 0000 16;
*a*+16) *a*+17 *a* + 2 0000 20.

Оператор $\mathbf{A}_5$:

*a*+17) *b*+1 *n*—1 *b*+1 70,

Оператор $\mathbf{\Pi}_6$:

*a*+20) *b*+1 *n*— 1 0000 44.

Оператор $\mathbf{Я}_7$:

*a*+21) 0000 0000 0000 40.

Для того чтобы произвести распределение памяти, положим

$$a = 0017, \quad b = a + 21 = 0040, \quad c = b + n + 1 == 0041 + n.$$

Составление простейшего оператора ввода и замену в программе букв их численными значениями предоставляем сделать самому читателю.

Программы, в состав которых входят циклы с переадресацией, известны под названием *циклических изменяемых программ.*

Рассмотрим подробнее операцию переадресации, выполняемую в программе оператором **F**(*i*). Как было сказано, переадресация заключается в том, что при каждом выполнении оператора переадресации к адресам изменяемой команды производится прибавление некоторых чисел.

Пусть адрес команды, изменяемый оператором **F**(*i*) до первого выполнения **F**(*i*), имеет вид $a_0$. Обозначим через *h*(*k*) число, прибавляемое к этому адресу при *k*-м выполнении оператора **F**(*i*), Если адрес команды не подвергался изменениям другими операторами. то после *i*-го выполнения оператора **F**(*i*) он принимает такой вид:

$$a(i) = a_0 + h(1) + h(2) + \ldots + h(i),$$

т. е.

$$a(i) = a_0 + \sum_{k=1}^{i} h(k).$$

Частным случаем переадресации является наиболее распространенный случай, когда $h(k) = h = \text{const}$. При этом

$$\sum_{k=1}^{i} h(k) = ih$$

и

$$a(i) = a_0 + ih.$$

В этом случае говорят, что изменяемый адрес подвергается переадресации по параметру *i* с *постоянным шагом h.* В дальнейшем будем различать переадресацию с *постоянным шагом* и переадресацию с *переменным шагом.*

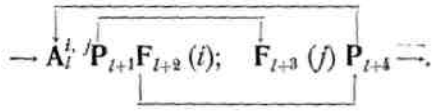
Предположим теперь, что изменяемый адрес до выполнения операторов переадресации есть $a_0$ и что он подвергается изменениям операторами переадресации $\mathrm{F}(i_1)$, $\mathrm{F}(i_2)$,...,$\mathrm{F}(i_n)$. Если этот адрес не изменялся никакими другими операторами, то после того как $\mathrm{F}(i_1)$ проработал $i_1$, раз, $\mathrm{F}(i_2)$ проработал $i_2$ раз,..., $F(i_n)$ проработал $i_n$ раз, он будет иметь вид:

$$a(i_1, i_2, \ldots, i_n) = a_0 + \sum_{k=1}^{i_1} h_1(k) + \sum_{k=1}^{i_2} h_2(K) + \cdots + \sum_{K=1}^{i_n} h_n(k),$$

где $h_m(k)$ есть число, прибавляемое к рассматриваемому адресу оператором $\mathrm{F}(i_\mathrm{m})$ при *k*-м выполнении этого оператора ($m = 1, 2,..., n$). Такое изменение адресов может иметь место, например, для команд оператора $A_l^{ij}$ в циклах, имеющих следующие логические схемы:

1. Цикл в цикле:

2. Цикл с двумя параметрами:



Возможны случаи, когда каждый адрес некоторой команды подвергается переадресации только по одному параметру, но эти параметры у разных адресов различны. В дальнейшем будем различать случаи, когда изменяемая команда зависит от одного параметра и когда изменяемая команда зависит от нескольких параметров.

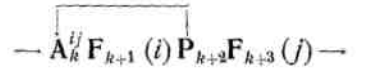
**3. Цикл с переадресацией и восстановлением.** В некоторых случаях оказывается необходимым команды цикла, измененные в результате переадресации, приводить в их первоначальный вид. Это необходимо, например, в случае, когда один цикл входит в состав другого цикла.

Приведение команд цикла, зависящих от некоторого параметра, **к** виду, соответствующему начальному значению этого параметра, называется *восстановлением по данному параметру* и осуществляется одним из двух способов:

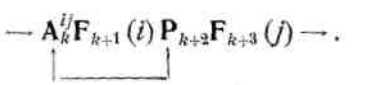
1) с помощью оператора переадресации, уничтожающего результаты предыдущих переадресаций;

2) с помощью оператора восстановления (обозначается символом $\mathbf{O}(i)$, где $i$ — параметр восстанавливаемого цикла), который заносит на места измененных команд цикла эти же команды в их первоначальном виде (для чего подлежащие восстановлению команды должны быть в своем первоначальном виде в качестве вспомогательных чисел запасены в специально отведенных ячейках).

Логическая схема цикла с переадресацией и восстановлением при помощи оператора переадресации может иметь следующий вид:

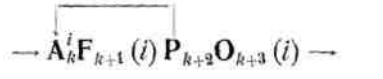


или



Переадресация по параметру $j$ уничтожает результаты предыдущих переадресаций, произведенных оператором $\mathbf{F}_{k+1}(i)$.

Логическая схема цикла с переадресацией и восстановлением путем засылки начального вида команд может выглядеть следующим образом:



или

Как видно из приведенных схем, оператор, восстанавливающий цикл, сам в состав этого цикла не входит.

Пример 4. Составить программу для вычисления таблицы значений функции

$$y = \alpha_0 \sin^8 x + \alpha_1 \sin^7 x + \ldots + \alpha_7 \sin x + \alpha_8 \qquad \text{(VI.25)}$$

для заданных значений независимой переменной $x = x_1, x_2, \ldots, x_n$. С помощью подстановки

$$z = \sin x \qquad \text{(VI.26)}$$

преобразуем формулу (VI.25) к виду

$$y = \alpha_0 z^8 + \alpha_1 z^7 + \ldots + \alpha_7 z + \alpha_8 . \qquad \text{(VI.27)}$$

Для того чтобы придать вычислениям по формуле (VI.27) однообразный циклический характер, приведем ее к следующему виду:

$$y = (((((((\alpha_0 z + \alpha_1) z + \alpha_2) z + \alpha_3) z + \alpha_4) z + \alpha_5) z + \alpha_6) z + \alpha_7) z + \alpha_8) . \qquad \text{(VI.28)}$$

Исходные данные расположим следующим образом:

$$b)\ \alpha_0,\quad b{+}1)\ \alpha_1,\quad b{+}2)\ \alpha_2,\quad b{+}10)\ \alpha_2;$$
$$d{+}1)\ x_1,\quad d{+}2)\ x_2,\quad d{+}3)\ x_3,\quad \ldots,\quad d{+}n)\ x_n,\quad d{+}n{+}1)\ \xi$$

Здесь $\xi$ — любое число, удовлетворяющее условию

$$\xi \neq x_i, \qquad i = 1, 2, \ldots, n,$$

и

$$d = b{+}10.$$

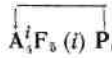
Составление логической схемы

Пусть $\mathbf{\Pi}_0$ — оператор ввода программы; $\mathbf{A}_1$— оператор, переводящий исходные данные в двоичную систему счисления.

Пусть $\mathbf{A}_2^j$ вычисляет величину $z_j = \sin x_j$ и записывает ее в ячейку $c + 1$; $\mathbf{\Pi}_3$ переносит число $a_0$ из ячейки $b$ в ячейку $c + 2$; $\mathbf{A}_4^i$ вычисляет выражение

$$(c{+}1)(c{+}2){+}(b{+}i)$$

и записывает результат в ячейку $c$+ 2. При первом выполнении оператора $\mathbf{A}_4^i$ (параметр $i$ имеет значение 1) будет вычислена величина, заключенная в первые внутренние скобки формулы (VI.28): $a_0 z_i + a_1$. Оператор $\mathbf{F}_5(i)$ производит переадресацию оператора $\mathbf{A}_4^i$ по параметру $i$. Оператор $\mathbf{P}_6$ заставляет цикл



выполняться восемь раз. При этом последовательно будут вычислены все восемь скобок формулы (VI.28), и в ячейке $c$+2 будет получена величина $y_i$. Оператор $\mathbf{O}_7(i)$ восстанавливает оператор $\mathbf{A}_4^i$ в его первоначальном виде. Оператор $\mathbf{\Pi}_8^j$ переносит $y_i$ из ячейки $c + 2$ в ячейку $d+j$ (в которой раньше хранилось значение независимой переменной $x_j$). Оператор $\mathbf{F}_9(j)$ производит переадресацию операторов $\mathbf{A}_2^j$, $\mathbf{\Pi}_8^j$ и

$\mathbf{P}_{10}^{j}$. Оператор $\mathbf{P}_{10}^{j}$ обеспечивает повторение всех вычислений для каждого из заданных значений величины $x$ путем сравнения $x_{i+1}$ с $\xi$. Оператор $\mathbf{A}_{11}$ переводит полученные результаты в десятичную систему счисления, после чего $\mathbf{П}_{12}$ выдает их на перфокарты, а $\mathbf{Я}_{13}$ останавливает машину. Логическая схема программы будет следующей:

$$\overline{\mathbf{П}_0\mathbf{A}_1\dot{\mathbf{A}}_2^{j}\mathbf{П}_3\overline{\dot{\mathbf{A}}_4^{i}\mathbf{F}_5\,(i)\,\mathbf{P}_6}\mathbf{O}_7\,(i)\,\mathbf{П}_8^{j}\mathbf{F}_9\,(j)\,\mathbf{P}_{10}^{j}}\mathbf{A}_{11}\mathbf{П}_{12}\mathbf{Я}_{13}$$

или

$$\mathbf{П}_0\mathbf{A}_1\overset{10}{\urcorner}\mathbf{A}_2^{j}\mathbf{П}_3\overset{6}{\urcorner}\mathbf{A}_4^{i}\mathbf{F}_5\,(i)\,\mathbf{P}_6\overset{4}{\ulcorner}\mathbf{O}_7\,(i)\mathbf{П}_8^{j}\mathbf{F}_9\,(j)\,\mathbf{P}_{10}^{j}\overset{2}{\ulcorner}\mathbf{A}_{11}\mathbf{П}_{12}\mathbf{Я}_{13}.$$

Составление программы *

Оператор $\mathbf{A}_2^{i}$ (двумя звездочками в этом и других операторах отмечены адреса команд, зависящие от параметра $j$):

$a$+1) $d$+1** 0000 $c$ + 1 67.

Оператор $\mathbf{П}_3$:

$a$+ 2) $b$ 0000 $c$ + 2 13.

Оператор $\mathbf{A}_4^{i}$ (одной звездочкой отмечены адреса команд, зависящие от параметра $i$):

$a$+ 3) $c$ + 2 $c$ + 1 $c$ + 2 05

$a$+ 4) $b$+1* $c$+ 2 $c$ + 2 01.

Оператор $\mathbf{F}_5(i)$ (в ячейку $e$ + 1 помещено вспомогательное число 1 (I)):

$a$ + 5) $a$ + 4 $e$ + \ $a$ + 4 02.

Перейдем к оператору $\mathbf{P}_6$. Для того чтобы заставить цикл, связанный с параметром $i$, выполняться подряд восемь раз, используем следующий прием. В ячейку $e$ + 2 заранее запишем нуль, а в $e$ + 3 — вспомогательное число 1000 (I). При каждом выполнении цикла будем складывать содержимое ячеек $e$ + 2 и $e$ + 3 с помощью операции специального сложения и результат записывать в ячейку $e$ + 2. Содержимое последней ячейки будем сравнивать с нулем и при несовпадении снова повторять весь цикл. После восьмикратного выполнения специального сложения должно получиться число 1 0000 (I), Однако вследствие того, что перенос из старшего разряда каждого адреса при специальном сложении теряется, в ячейке $e$+2 окажется записанным нуль. Это будет признаком того, что цикл, отвечающий параметру $i$, проработал восемь раз.

Оператор $\mathbf{P}_6$:

$a$+ 6) $e$ + 2 $e$ + 3 $e$ + 2 02;

$a$+ 7) $e$ + 2 0000 0000 16;

$a$+10) $h$ + 1 $a$ + 3 0000 20.

Оператор $\mathbf{O}_7$ ($i$):

$h$+1) $e$ + 4 0000 $a$ + 4 13

(в ячейке $e$ + 4 хранится вспомогательное число $b$+ 1 $c$ + 2 $c$ + 2 01, нужное для восстановления команды ($a$ + 4)).

Оператор $\mathbf{П}_8^{j}$:

$g$+1) $c$ + 2 0000 $d$+1** 13.

Оператор $\mathbf{F}_9$ ($j$) (в ячейке $e$ + 5 хранится вспомогательное число 1 (III));

$g$+2) $a$ + 1 $e$+1 $a$+1 02;

$g$+3) $g$ + 1 $e$ +5 $g$+1 02;

$g$+4) $g$ + 5 $e$+1 $g$+5 02.

Оператор $\mathbf{P}_{10}$ (при ω= 0 передает управление оператору $\mathbf{A}_{11}$ в ячейку $g$ + 7):

$g$+5) $d$+1** $d$ + $n$ +1 0000 16;

$g$+6) $g$+7 $a$ + 1 0000 20.

Для решения этой же задачи составим программу с восстановлением при помощи оператора переадресации. Логическая схема такой программы при сохранении обозначений, принятых для операторов в предыдущей логической схеме программы, будет следующей:

$$\overline{\mathbf{П}_0\mathbf{A}_1\dot{\mathbf{A}}_2^{j}\mathbf{П}_3\overline{\dot{\mathbf{A}}_4^{ij}\mathbf{F}_5\,(i)\,\mathbf{P}_6\mathbf{П}_8^{j}\mathbf{F}'_9\,(j)\,\mathbf{P}_{10}^{j}}}\mathbf{A}_{11}\mathbf{П}_{12}\mathbf{Я}_{13}$$

или

$$\mathbf{П}_0\mathbf{A}_1\overset{10}{\urcorner}\mathbf{A}_2^{j}\mathbf{П}_3\overset{6}{\urcorner}\mathbf{A}_4^{ij}\mathbf{F}_5\,(i)\,\mathbf{P}_6\overset{4}{\ulcorner}\mathbf{П}_8^{j}\mathbf{F}'_9\,(j)\,\mathbf{P}_{10}^{j}\overset{2}{\ulcorner}\mathbf{A}_{11}\mathbf{П}_{12}\mathbf{Я}_{13}.$$

Вместо операторов $\mathbf{O}_7$ ($i$) и $\mathbf{F}_9$ ($j$) будет теперь один оператор $\mathbf{F'}_9$ ($j$). Для его работы потребуется вспомогательное число 10(1), которое мы поместим в ячейку $e$ + 4 вместо стоявшего там вспомогательного числа, служившего для восстановления команды ($a$ + 4).

Оператор $\mathbf{F'}_9(j)$:

$g$ +2) $a$ +1 $e$ +1 $a$+ 1 02;

$g$ +3) $g$+ 1 $e$+5 $g$+ 1 02;

$g$ +4) $g$+5 $e$+1 $g$+5 02;

$g'$+4) $a$+ 4 $e$+4 $a$+ 4 15.

Остальные операторы новой программы ничем не будут отличаться от одноименных операторов старой программы. Чтобы не изменять обозначений номеров ячеек, хранящих команды, при составлении первой программы мы обозначили ячейку, хранящую команду оператора $\mathbf{O}_7(i)$, через $h$ + 1, а последнюю ячейку, содержащую команду оператора $\mathbf{F'}_9$ ($j$) новой программы, через $g'$ + 4.

Оператор восстановления $\mathbf{O}(i)$ по параметру $i$ уничтожает все изменения, произведенные в изменяемых командах оператором $\mathbf{F}(i)$.

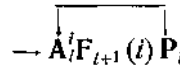
В общем случае оператор $\mathbf{O}(i_k)$ преобразует каждый адрес вида $a(i_1,i_2,\ldots,i_{k-1},i_k,i_{k+1},\ldots,i_n)$ к виду $a(i_1,i_2,\ldots,i_{k-1},i_k^{(0)},i_{k+1},\ldots,i_n)$. Оператор восстановления располагается в схеме вне цикла, содержащего $\mathbf{F}(i_k)$. В

* В этом и во всех последующих примерах программ операторы переноса программы с перфокарт в память, преобразования десятичных чисел в двоичные, преобразования двоичных чисел в десятичные, переноса результатов из памяти на перфокарты и останова машины мы будем опускать, ограничиваясь приведением их в логических схемах программ.

рассмотренных выше примерах мы видели, что восстановление может производиться путем занесения на места переадресованных команд этих команд в их первоначальном виде. Такое восстановление известно под названием *восстановления в первоначальном виде* и возможно в том случае, когда производится восстановление по всем параметрам. В частности, такое восстановление возможно в случае, когда адреса изменяемых команд зависят только от одного параметра (переадресуются только одним оператором переадресации).

Кроме восстановления в начальном виде, мы встречали также восстановление путем переадресации, когда измененным командам придается их «начальный» вид (вид, который они имели бы, если бы не подвергались переадресации по параметру $i$) путем вычитания от них сумм чисел, прибавленных в процессе переадресации. Однако возможные случаи восстановления не исчерпываются двумя приведенными случаями. Приведем краткий перечень наиболее распространенных приемов, применяемых для осуществления восстановления (в том числе и уже знакомых нам).

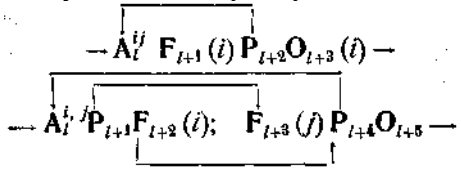
**I**. Команды оператора $\mathbf{A}_l^i$ зависят только от одного параметра; логическая схема цикла имеет, например, такой вид:



1. Как уже говорилось, в этом случае возможно по параметру $i$ восстановление в начальном виде.
2. Если число выполнений оператора $\mathbf{F}(i)$ (т. е. всего цикла) заранее известно, то, как в случае переадресации с постоянным шагом, так и в случае переадресации с переменным шагом, иногда бывает целесообразным восстановление путем переадресации. Например, такое восстановление удобно, если число подлежащих восстановлению команд велико, а число констант, путем вычитания которых (с помощью операции 15 или 02) можно произвести восстановление, значительно меньше. При этом количество рабочих тактов машины, необходимых для восстановления, не возрастает сравнительно со случаем восстановления в первоначальном виде, а количество ячеек, занятых материалом, необходимым для восстановления, уменьшается.
3. При точно неизвестном заранее, но большом количестве выполнений оператора $\mathbf{F}(i)$ обычно используют восстановление в начальном виде.

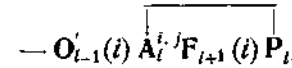
**II**. Если команды оператора $\mathbf{A}_l^{i,j}$ (для простоты ограничиваемся случаем только двух параметров $i$ и $j$) зависят каждая только от одного параметра, все рекомендации, приведенные в пункте **I**, остаются в силе.

Логические схемы циклов при этом могут иметь, например, такой вид:

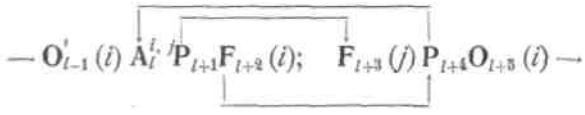


**III**. Команды оператора $\mathbf{A}_l^{i,j}$ зависят сразу от двух параметров (по крайней мере некоторые команды).

1. В этом случае восстановление в начальном виде невозможно, так как восстановление по одному параметру не должно уничтожать последствий переадресаций по другому параметру. Однако возможно восстановление, аналогичное восстановлению в начальном виде, осуществляемое с помощью следующего приема. Впереди цикла располагают подготовительный оператор (обозначим его символом $\mathbf{O}'(i)$, который до начала работы цикла «фотографирует» все его команды, изменяемые оператором $\mathbf{F}(i)$, и помещает полученные «снимки команд» в рабочие ячейки. После цикла располагают оператор восстановления $\mathbf{O}(i)$, который после конца работы цикла запасенные «снимки команд» занесет на места этих команд. При таком восстановлении логическая схема цикла может иметь такой вид:

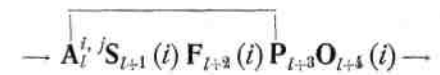


или

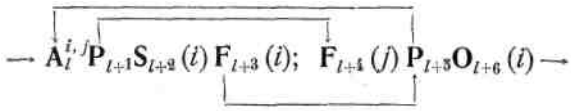


2. Если программисту заранее известно количество выполнений оператора $\mathbf{F}(i)$, то возможно применение как при переадресации с переменным шагом, так и при переадресации с постоянным шагом восстановления путем переадресации. Иногда при этом бывает удобно оператор $\mathbf{O}(i)$ объединить с оператором $\mathbf{F}(i)$, например, в случае цикла в цикле.
3. Если количество выполнений оператора $\mathbf{F}(i)$ заранее точно не известно, но велико, для осуществления восстановления путем переадресации в состав цикла приходится включать счетчик подсчитывающий число выполнений оператора $\mathbf{F}(i)$ (если команды $\mathbf{A}_l^{i,j}$ переадресуются только с постоянным шагом) или суммы чисел, прибавляемых оператором $\mathbf{F}(i)$ к адресам команд (если происходит переадресация с переменным шагом). Такие счетчики требуют дополнительного расхода рабочих тактов машины и применяются обычно в тех случаях, когда программист при программировании испытывает острый недостаток в ячейках памяти и может этим способом добиться экономии ячеек.

Если обозначить оператор, ведущий счет выполнений оператора $\mathbf{F}(i)$ или сумм чисел, прибавляемых этим оператором к командам оператора $\mathbf{A}_l^{i,j}$, символом $\mathbf{S}(i)$, то логическая схема цикла в описанном случае может иметь, например, такой вид:



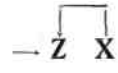
или



Обычно оператор $\mathbf{S}(i)$ в схемах в явном виде не показывают, а объединяют его с оператором $\mathbf{F}(i)$ и потому стандартного обозначения этот оператор не имеет.

## § 34. Способы управления повторениями цикла

**1. Оператор управления повторениями цикла.** Как уже указывалось, в состав каждого цикла программы должен входить оператор (в ряде случаев обобщенный), управляющий повторениями цикла. Если совокупность всех операторов, входящих в цикл, кроме оператора, управляющего его повторениями, рассматривать как некоторый обобщенный оператор и обозначить буквой **Z**, а последний оператор — буквой **X**, то простейший цикл можно схематически изобразить следующим образом:



Рассмотрим простейшие виды операторов, применяемых для управления повторениями цикла. Будем считать, что первая команда обобщенного оператора **Z** хранится в ячейке $\alpha$, а его последняя команда — в ячейке $\alpha+\mu$. Обобщенному оператору **Z** отвечает часть программы

$\alpha)$ $a,$ $b,$ $c;$ $\theta;$

.....

$\alpha+\mu)$ $a',$ $b',$ $c';$ $\theta'.$

Первая из команд оператора **X** содержится **в** ячейке $\alpha + \mu + 1$.

Цикл, работающий дважды. Управление выполнением цикла, работающего дважды, можно осуществлять одной командой условного перехода, перед которой, однако, должна оставаться «пустая» (заполненная нулями) ячейка:

$\alpha + \mu.+ 1)$ 0000 0000 0000 0 00;

$\alpha + \mu + 2)$ $a$ $a$ $\alpha+\mu+ 1$ 0 27.

После того как цикл проработает, получает управление команда $(\alpha + \mu + 1)$. Это — «нулевая» команда, не содержащая никакого приказа. От нее управление получит команда $(\alpha + \mu + 1)$. Последняя передаст управление начальной команде цикла $(a)$ и одновременно запишет в ячейку $\alpha + \mu + 1$ команду возврата $\alpha + \mu + 3$ $\alpha + \mu + 3$ $\alpha + \mu + 1$ 20. После второго выполнения цикла управление снова получит команда $(\alpha + \mu + 1)$. Она передаст управление команде $(\alpha + \mu + 3)$ и одновременно сама себя погасит, заменит нулями.

Достоинство приведенного оператора управления повторением цикла состоит в том, что его не нужно восстанавливать: после конца своей работы он приходит в свой первоначальный вид.

Замечание. В тех случах, когда не требуется восстановления команд программы, можно вместо двух команд, образующих **X**, поставить одну команду:

$\alpha + \mu + 1)$ $a$ $a$ $\alpha + \mu + 1$ 0 20.

Это — команда самогасящаяся. После первой работы цикла она передаст управление начальной его команде и одновременно заменит себя нулями. После второй работы цикла управление машиной будет передано дальше.

Пример 1. Составить программу для вычисления функции

$$y = x^2 + \sqrt{1 + \sqrt{1 + x}}$$

Начальные данные размещаем следующим образом:

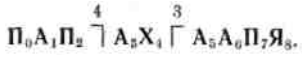
$b + 1)$ $x,$ $b + 2)$ 1.

В качестве рабочих используем ячейки $c + 1$ и $c + 2$. Результаты вычислений поместим в ячейку $c + 1$.

Пусть оператор $\mathbf{П}_0$ вводит программу в память машины; оператор $\mathbf{A}_1$ переводит исходные данные в двоичную систему счисления; оператор $\mathbf{П}_2$ переносит $x = (b + 1)$ в ячейку $c + 1$; оператор $\mathbf{A_3}$ вычисляет $\sqrt{1 + (c + 1)}$ и заносит результат опять в ячейку $c$+1; оператор $\mathbf{X}_4$ управляет с помощью только что описанного способа повторениями цикла (заставляет работать $\mathbf{A}_3$ дважды); оператор $\mathbf{A}_5$ оканчивает вычисление величины $y$; $\mathbf{A}_6$ осуществляет перевод результата из двоичной системы в десятичную; $\mathbf{П}_7$ выводит результат счета на перфокарту; $\mathbf{Я}_8$ останавливает машину.

Логическая схема программы имеет такой вид:

или



Составление программы

Оператор $\mathbf{П}_2$:

$a + 1)$ $b+1$ 0000 $c+ 1$ 13.

Оператор $\mathbf{A}_3$:

$a + 2)$ $b + 2$ $c+1$ $c+ 1$ 01,

$a + 3)$ $c + 1$ 0000 $c+ 1$ 63.

Оператор $\mathbf{X}_4$:

$a + 4)$ 0000 0000 0000 000,

$a + 5)$ $a + 2$ $a+ 2$ $a+ 4$ 27.

Оператор $\mathbf{A}_5$:

$a+6)\quad b+1\qquad b+1\quad c+2\qquad 05,$
$a+7)\quad c+1\qquad c+2\quad c+1\qquad 01.$

В таблице 37 приведено содержимое рабочих ячеек после выполнения машиной каждой операции. Сравнивая первую и вторую графы таблиц, мы видим, что цикл $\overbrace{\mathbf{A}_3,\ \mathbf{X}_4}$— работает дважды.

Таблица 37. Содержимое рабочих ячеек в процессе выполнения программы

| № операции | № команды | Содержимое рабочих ячеек | |
|---|---|---|---|
| | | $c$+1 | $c$+2 |
| 1 | $a$+1 | $x$ | – |
| 2 | $a$+2 | $1+x$ | – |
| 3 | $a$+3 | $\sqrt{1+x}$ | – |
| 4 | $a$+4 | $\sqrt{1+x}$ | – |
| 5 | $a$+5 | $\sqrt{1+x}$ | – |
| 6 | $a$+2 | $1+\sqrt{1+x}$ | – |
| 7 | $a$+3 | $\sqrt{1+\sqrt{1+x}}$ | – |
| 10 | $a$+4 | $\sqrt{1+\sqrt{1+x}}$ | – |
| 11 | $a$+5 | $\sqrt{1+\sqrt{1+x}}$ | – |
| 12 | $a$+6 | $\sqrt{1+\sqrt{1+x}}$ | $x^2$ |
| 13 | $a$+7 | $x^2+\sqrt{1+\sqrt{1+x}}$ | $x^2$ |

Для управления циклом, работающим дважды, может быть также использован оператор **X** с *мерцающей единицей* или с кувыркающейся единицей.

Остановимся вначале на операторе с *мерцающей единицей.* Предположим, что в двух ячейках записано число +1:

$\beta+1)+1,\quad \beta+2)+1$

Оператор **X** состоит из следующих команд:

$\alpha+\mu+1)\quad \beta+1\quad \beta+2\quad \beta+1\quad 16$
$\alpha+\mu+2)\quad \alpha\qquad h\qquad 0000\quad 20.$

По первой из этих двух команд происходит сравнение чисел $(\beta+1)=+1$ и $(\beta+2)=+1$. Эти числа совпадают, так что вырабатывается сигнал $\omega=0$. Результат сравнения равен нулю. Он записывается в ячейку $\beta+1$ вместо числа +1. Вторая команда оператора **X** передает управление команде ($a$). При вторичном выполнении **X** по первой из его команд происходит сравнение чисел $(\beta+1)=0$ и $(\beta+2)=1$. Эти числа не совпадают, так что вырабатывается сигнал $\omega=1$. Результат сравнения равен +1. Он заносится в ячейку $\beta+1$ на место числа 0. Вторая команда оператора **X** передает теперь управление команде ($h$). При дальнейших выполнениях оператора **X** описанный процесс повторяется.

Название *мерцающая единица* происходит оттого, что при многократном выполнении оператора **X** в ячейке $\beta+1$ то появляется, то исчезает (мерцает) число +1. Вместо единицы в ячейки $\beta+1$ и $\beta+2$ можно поместить любое число, отличное от нуля. При этом работа оператора **X** не изменится.

Рассмотрим теперь оператор **X** с *кувыркающейся единицей.* Пусть в двух ячейках записаны числа +1 и —1:

$\beta+1)+1,\quad \beta+2)-1.$

Рассмотрим оператор **X**, состоящий из команд:

$\alpha+\mu+1)\quad \beta+1\quad \beta+2\quad \beta+1\quad 05$
$\alpha+\mu+2)\quad \beta+1\quad \beta+2\quad 0000\quad 16$
$\alpha+\mu+3)\quad a\qquad h\qquad 0000\quad 20.$

При первом выполнении оператора **X** по команде $(\alpha+\mu+1)$ выполняется операция

$(+1)(-1)=-1.$

Содержимое ячейки $\beta+1$ превращается в –1. По команде $(\alpha+\mu+2)$ сравниваются числа $(\beta+1)$ и $(\beta+2)$. Они совпадают, так что вырабатывается сигнал $\omega=0$. Команда $(\alpha+\mu+3)$ передает управление в ячейку $\alpha$..

При втором выполнении оператора **X** по команде $(\alpha+\mu+1))$ выполняется операция

$(-1)(-1)=+1.$

Результат записывается в ячейку $\beta+1$. По команде $(\alpha+\mu+2)$ сравниваются числа $(\beta+1)$ и $(\beta+2)$. Теперь они не совпадают. Вырабатывается сигнал $\omega=1$. Команда $(\alpha+\mu+3)$ передает управление в ячейку $h$. При дальнейших выполнениях оператора **X** описанный процесс повторяется. При этом в ячейке $\beta+1$ чередуются числа +1,–1,+1,–1,.. Отсюда произошло название *кувыркающаяся единица.*

Логический оператор с «мерцающей единицей» или с «кувыркающейся единицей» может быть использован во всех тех случаях, когда некоторая группа команд получает управление многократно и после ее выполнения управление передается поочередно в одном из двух направлений. Использование таких операторов для управления циклом, выполняющимся дважды, является лишь одним из частных случаев использования мерцающей и кувыркающейся единицы.

Цикл, работающий трижды. Управление таким циклом производит оператор **X**, состоящий из команд:

$\alpha+\mu+1)$ 0000 0000 0000 0 00;
$\alpha+\mu+2)$ $a$ $a$ $\alpha+\mu+1$ 27;
$\alpha+\mu+3)$ $a$ $a$ $\alpha+\mu+1$ 27.

Увеличивая количество подряд записанных команд

$$a \quad a \quad \alpha+\mu+1 \quad 27,$$

мы можем построить оператор **X**, обеспечивающий многократное повторение работы цикла. Оператор **X**, как и в случае цикла, работающего дважды, не требует восстановления, однако занимает при этом много ячеек памяти. Целесообразнее при большом количестве повторений цикла применять другие виды оператора **X**.

**2. Счетчик повторений цикла.** Счетчик повторений цикла состоит из трех ячеек, в которых перед началом работы помещают число $n$, определяющее, сколько раз должен проработать цикл, и числа 0 и 1:

$$\beta+1)\ n,\ \beta+2)\ 0,\ \beta+3)\ 1.$$

Оператор **X** состоит из следующих команд:

$\alpha+\mu+1)$ $\beta+3$ $\beta+2$ $\beta+2$ 01;
$\alpha+\mu+2)$ $\beta+2$ $\beta+1$ 0000 16;
$\alpha+\mu+3)$ $h$ $a$ 0000 20.

По первой из этих команд к числу $(\beta+1)=0$ прибавляется $(\beta+3)=1$; результат заносится в ячейку $\beta+2$; теперь $(\beta+2)=1$ (это означает, что цикл уже один раз проработал). По второй команде $(\beta+2)$ сравнивается с $(\beta+1)=n$; при несовпадении этих чисел последняя из команд передает управление начальной команде цикла; при совпадении, что произойдет после $n$-кратной работы цикла, управление передается вовне, команде $h$.

Иногда с тем, чтобы использовать ячейку $\beta+3$ одновременно для хранения константы переадресации, счетчик видоизменяют следующим образом:

$$\beta+1)\ n(1),\quad \beta+2)\ 0,\quad \beta+3)\ 1(1)$$

(вместо единиц первого адреса могут быть использованы единицы второго или третьего адресов). При этом оператор **X** имеет такой вид:

$\alpha+\mu+1)$ $\beta+2$ $\beta+3$ $\beta+2$ 02;
$\alpha+\mu+2)$ $\beta+2$ $\beta+1$ 0000 16;
$\alpha+\mu+3)$ $h$ $a$ 0000 20.

Пример 2. Составить программу для вычисления величины $\sin x$ при $|x|<\frac{\pi}{2}$.

Известно, что

$$\sin x=\frac{x}{1!}-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\ldots \qquad \text{(VI.29)}$$

Считаем, что число членов ряда (VI.29), дающее $\sin x$ с нужной точностью, нам известно. Пусть оно равно $k$.

Если $u_i$ — член ряда (VI.29) с номером $i$, а $u_{i+1}$— следующий за ним член, то, очевидно,

$$u_{i+1}=u_i\left[-\frac{x^2}{(2i+2)(2i+3)}\right]. \qquad \text{(VI.30)}$$

Последней зависимостью воспользуемся для упрощения вычислений. Исходные данные разместим в памяти машины следующим образом:

$$b+1)\ x,\ b+2)\ 2,\ b+3)\ 1,\ b+4)\ 1,\ b+5)\ k.$$

В качестве рабочих ячеек используем ячейки с номерами $c+1$, $c+2$, ... Для результата вычислений отведем ячейку $c$. Ячейки $b+3$, $b+4$ и $b+5$ являются счетчиком повторений цикла. В ячейке $b+4$ в начале счета стоит не нуль, а единица, потому что $u_1$ — первый член ряда — равен $x$ и его не нужно вычислять.

Пусть оператор $\mathbf{\Pi}_0$ вводит программу в память, а $\mathbf{A}_1$ переводит исходные данные в двоичную систему счисления, оператор $\mathbf{\Pi}_2$ переносит $(b+1)=x$ в ячейки $c$ и $c+1$, нуль — в ячейку $c+2$ и единицу — в $c+3$; оператор $\mathbf{A}_3$ вычисляет величину $-x^2$; оператор $\mathbf{A}_4$ по известной величине $u_i$ с помощью формулы (VI.30) вычисляет $u_{i+1}$ и, складывая величины $u_1+u_2+\ldots$, последовательно получает частные суммы ряда $s_1, s_2, \ldots$; оператор $\mathbf{P}_5$ с помощью счетчика повторений управляет работой цикла; оператор $\mathbf{A}_6$ переводит результат в десятичную систему, а оператор $\mathbf{\Pi}_7$ выводит его на перфокарту; $\mathbf{Я}_8$ останавливает машину. Логическая схема программы будет иметь такой вид:

или

Составление программы

Оператор $\mathbf{\Pi}_2$:

$a+1)$ $b+1$ 0000 $c$ 13
$a+2)$ $b+1$ 0000 $c+1$ 13
$a+3)$ 0000 0000 $c+2$ 13
$a+4)$ $b+3$ 0000 $c+3$ 13

Оператор $\mathbf{A}_3$:

$a+5)$ $b+1$ $b+1$ $c+4$ 05
$a+6)$ 0000 $c+4$ $c+4$ 03

Оператор $\mathbf{A}_4$:

| | | | | |
|---|---|---|---|---|
| *a* + 7) | *c* + 2 | *b* + 2 | *c* + 2 | 01 |
| *a*+10) | *c* + 3 | *b* + 2 | *c* + 3 | 01 |
| *a*+11) | *c* + 2 | *c* + 3 | *c* + 5 | 05 |
| *a*+12) | *c* + 5 | 0000 | *c* + 5 | 62 |
| *a*+13) | *c* + 4 | *c* + 5 | *c* + 5 | 05 |
| *a*+14) | *c* + 1 | *c* + 5 | *c* + 1 | 05 |
| *a*+15) | *c* | *c* + 1 | *c* | 01 |

Оператор $\mathbf{P}_5$ (при ω = 0 передает управление оператору $\boldsymbol{A}_6$ в ячейку *a* + 21):

| | | | | |
|---|---|---|---|---|
| *a*+16) | *b* + 3 | *b* + 4 | *b* + 4 | 01 |
| *a*+17) | *b* + 4 | *b* + 5 | 0000 | 16 |
| *a*+20) | *a* + 21 | *a* + 7 | 0000 | 20 |

В таблице 38 показано содержимое рабочих ячеек при работе программы.

Т а б л и ц а 38. Содержимое рабочих ячеек в процессе вычисления синуса

| Оператор | № пп. | № команды | Содержание рабочих ячеек | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | *c* | *c*+1 | *c*+2 | *c*+3 | *c*+4 | *c*+5 |
| $\mathbf{П}_2$ | 1 | *a*+1 | $s_1 = x$ | – | – | – | – | – |
| | 2 | *a*+2 | $x$ | $x$ | – | – | – | – |
| | 3 | *a*+3 | $x$ | $x$ | 0 | – | – | – |
| | 4 | *a*+4 | $x$ | $x$ | 0 | 1 | – | – |
| $\mathbf{A}_3$ | 5 | *a*+5 | $x$ | $x$ | 0 | 1 | $x^2$ | – |
| | 6 | *a*+6 | $x$ | $x$ | 0 | 1 | $-x^2$ | – |
| $\mathbf{A}_4$ | 7 | *a*+7 | $x$ | $x$ | 2 | 1 | $-x^2$ | – |
| | 10 | *a*+10 | $x$ | $x$ | 2 | 3 | $-x^2$ | – |
| | 11 | *a*+11 | $x$ | $x$ | 2 | 3 | $-x^2$ | 2·3 |
| | 12 | *a*+12 | $x$ | $x$ | 2 | 3 | $-x^2$ | 1/2·3 |
| | 13 | *a*+13 | $x$ | $x$ | 2 | 3 | $-x^2$ | $-\frac{x^2}{2 \bullet 3}$ |
| | 14 | *a*+14 | $x$ | $-\frac{x^3}{2 \bullet 3}$ | 2 | 3 | $-x^2$ | $-\frac{x^2}{2 \bullet 3}$ |
| | 15 | *a*+15 | $s_2 = x - \frac{x^3}{2 \bullet 3}$ | $-\frac{x^3}{3!}$ | 2 | 3 | $-x^2$ | $-\frac{x^2}{2 \bullet 3}$ |
| $\mathbf{P}_5$ | 16 | *a*+16 | | | | | | |
| | 17 | *a*+17 | | | | | | |
| | 20 | *a*+20 | | | | | | |
| $\mathbf{A}_4$ | 21 | *a*+7 | $s_2$ | $-\frac{x^3}{3!}$ | 4 | 3 | $-x^2$ | $-\frac{x^2}{3!}$ |
| | 22 | *a*+10 | $s_2$ | $-\frac{x^3}{3!}$ | 4 | 5 | $-x^2$ | $-\frac{x^2}{3!}$ |
| | 23 | *a*+11 | $s_2$ | $-\frac{x^3}{3!}$ | 4 | 5 | $-x^2$ | 4·5 |
| | 24 | *a*+12 | $s_2$ | $-\frac{x^3}{3!}$ | 4 | 5 | $-x^2$ | $\frac{1}{4 \bullet 5}$ |
| | 25 | *a*+13 | $s_2$ | $-\frac{x^3}{3!}$ | 4 | 5 | $-x^2$ | $-\frac{x^2}{4 \bullet 5}$ |
| | 26 | *a*+14 | $s_2$ | $+\frac{x^5}{5!}$ | 4 | 5 | $-x^2$ | $-\frac{x^2}{4 \bullet 5}$ |
| | 27 | *a*+15 | $s_3 = s_2 + \frac{x^5}{5!}$ | $+\frac{x^5}{5!}$ | 4 | 5 | $-x^2$ | $-\frac{x^2}{4 \bullet 5}$ |

С ч е т ч и к, и с п о л ь з у ю щ и й и з м е н я е м у ю к о м а н д у ц и к л а. Описанный счетчик повторений цикла, кроме ячеек, занятых программой, требует еще трех ячеек. Можно создать счетчик циклов,

требующий только одной дополнительной ячейки, для чего в специальную ячейку записывают окончательный вид одной из команд цикла, которая после каждого повторения цикла видоизменяется (благодаря, например, переадресации). В конце цикла ставят оператор, сравнивающий эту изменяемую команду с ее окончательным видом и передающий управление начальной команде цикла в случае их несовпадения и дальнейшим командам при совпадении.

П р и м е р 3. Составить программу для вычисления по схеме Горнера значения многочлена

$$P_n(x) = B_0x'' + B_1x^{n-1} + \ldots + B_{n-1}x + B_n. \qquad \text{(VI.31)}$$

при любом заданном значении $x$.

Для счета по схеме Горнера многочлен (VI.31) следует преобразовать следующим образом:

$$\begin{aligned} P_n(x) &= (B_0x + B_1)\,x^{n-1} + B_2x^{n--2} + \ldots + B_{n+1}x + B_n; \\ P_n(x) &= ((B_0x + B_1)x + B_2)\,x^{n-2} + B_3x^{n-3} + \ldots + B_{n-1}x + B_n; \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ P_n(x) &= (\underbrace{\ldots((B_0x + B_1)x + B_2)x + B_3)x + \ldots + B_{n-1})x + B_n}_{n \text{ скобок}} \end{aligned} \qquad \text{(VI.32)}$$

Формула (VI.32) применяется для вычислений. Исходные данные разместим так:

$b$-1) $x$, $b$) $B_0$, $b+1$) $B_1$, … , $b+n$)$B_n$,
$d$+l) l(II), $d$+ 2) $c$+l $b$ +$n$+1 $c$+1 01, $d$+3) $c$+l $b$+1 $c$+1 01.

В качестве рабочих ячеек используем ячейки $c$+ 1, $c$+2, ... Ответ поместим в ячейку $c$ + 1. Пусть операторы $\mathbf{П}_0$ и $\mathbf{A}_1$ имеют обычный смысл; оператор $\mathbf{П}_2$ пересылает число $B_0$ из ячейки $b$ в ячейку $c$+ 1; оператор $A_3^i$ умножает ($c$ + 1) на $x$, прибавляет к произведению $B_i$ ($i$ = 1, 2, ..., $n$) и результат опять записывает в $c$+ 1; $\mathbf{F}_4(i)$ осуществляет переадресацию оператора $\mathbf{A}_3^i$; оператор $\mathbf{P}_5$ управляет повторениями цикла с помощью только что описанного вида счетчика; $\mathbf{O}_6(i)$ восстанавливает команды оператора $A_3^i$ для того, чтобы обеспечить возможность повторного применения программы без повторного ее ввода; $\mathbf{A}_7$ переводит результат в десятичную систему; $\mathbf{П}_8$ выдает результат на перфокарту. $\mathbf{Я}_9$ останавливает машину. Логическая схема программы имеет вид

$$\mathbf{П}_0\mathbf{A}_1\mathbf{П}_2\overbrace{\mathbf{A}_3^i\mathbf{F}_4(i)\,\mathbf{P}_5}\mathbf{O}_6(i)\,\mathbf{A}_7\mathbf{П}_8\mathbf{Я}_9$$

или

$$\mathbf{П}_0\mathbf{A}_1\mathbf{П}_2\,\overset{5}{\urcorner}\,\mathbf{A}_3^i\mathbf{F}_4(i)\,\mathbf{P}_5\,\overset{3}{\llcorner}\,\mathbf{O}_6(i)\,\mathbf{A}_7\mathbf{П}_8\mathbf{Я}_9.$$

Программа будет следующей:

Оператор $\mathbf{П}_2$:

| | | | | |
|---|---|---|---|---|
| $a$ + 1) | $b$ | 0000 | $c$+1 | 13 |

Оператор $\mathbf{A}_3^i$:

| | | | | |
|---|---|---|---|---|
| $a$ + 2) | $c$ + 1 | $b$ – 1 | $c$ + 1 | 05 |
| $a$ + 3) | $c$ + 1 | $b$ + 1* | $c$ + 1 | 01 |

Оператор $\mathbf{F}_4(i)$:

| | | | | |
|---|---|---|---|---|
| $a$ + 4) | $a$ +3 | $d$ + 1 | $a$ + 3 | 02 |

Оператор $\mathbf{P}_5$:

| | | | | |
|---|---|---|---|---|
| $a$ + 5) | $a$ +3 | $d$ + 2 | 0000 | 16 |
| $a$ + 6) | $a$ +7 | $a$ + 2 | 0000 | 20 |

Оператор $\mathbf{O}_6(i)$:

| | | | | |
|---|---|---|---|---|
| $a$ + 7) | $d$ +3 | 0000 | $a$ + 3 | 13. |

В таблице 39 приведено содержимое рабочей ячейки $c$ + 1 и ячейки $a$ + 3 счетчика после выполнения каждой команды программы.

Ввиду того, что сравнение команды $a$ + 3 производится после переадресации, окончательный вид этой команды $c$ + 1 $b$ + $n$ + 1 $c$ + 1 01 (в ячейке $d$+ 2), а не $c$+ 1 $b$+ $n$ $c$+ 1 01.

Таблица 39. Работа счетчика

| № пп. | № команды | $a$ + 3 | $d$ + 2 | $c$ + 1 |
|---|---|---|---|---|
| 0 | – | $c$+1 $b$+1 $c$+1 01 | $c$+1 $b$+$n$+1 $c$+1 01 | – |
| 1 | $a$+1 | $c$+1 $b$+1 $c$+1 01 | $c$+1 $b$+$n$+1 $c$+1 01 | $B_0$ |
| 2 | $a$+2 | $c$+1 $b$+1 $c$+1 01 | $c$+1 $b$+$n$+1 $c$+1 01 | $B_0x$ |
| 3 | $a$+3 | $c$+1 $b$+1 $c$+1 01 | $c$+1 $b$+$n$+1 $c$+1 01 | $B_0x$+ $B_1$ |
| 4 | $a$+4 | $c$+1 $b$+2 $c$+1 01 | $c$+1 $b$+$n$+1 $c$+1 01 | $B_0x$+ $B_1$ |
| 5 | $a$+5 | $c$+1 $b$+2 $c$+1 01 | $c$+1 $b$+$n$+1 $c$+1 01 | $B_0x$+ $B_1$ |
| 6 | $a$+6 | $c$+1 $b$+2 $c$+1 01 | $c$+1 $b$+$n$+1 $c$+1 01 | $B_0x$+ $B_1$ |
| 7 | $a$+2 | $c$+1 $b$+2 $c$+1 01 | $c$+1 $b$+$n$+1 $c$+1 01 | ($B_0x$+ $B_1$)$x$ |
| 10 | $a$+3 | $c$+1 $b$+2 $c$+1 01 | $c$+1 $b$+$n$+1 $c$+1 01 | ($B_0x$+ $B_1$)$x$+ $B_2$ |
| 11 и т.д. | $a$+4 | $c$+1 $b$+2 $c$+1 01 | $c$+1 $b$+$n$+1 $c$+1 01 | ($B_0x$+ $B_1$)$x$+ $B_2$ |

**3. Цикл, повторяющийся пока монотонно изменяющаяся величина не перейдет через заданное значение.** Предположим, что в ячейке β +1 при каждом выполнении цикла получается некоторая величина, монотонно возрастающая в результате каждого выполнения цикла. Цикл должен выполняться до тех пор, пока эта

величина не превзойдет некоторого числа, хранящегося в ячейке β+2. Оператор **X**, управляющий выполнениями цикла, может иметь такой вид:

$$\begin{array}{llll} \alpha+\mu+1) & \beta+2 & \beta+.1 & 0000\quad 03, \\ \alpha+\mu+2) & \alpha & h & 0000\quad 20. \end{array}$$

Этот оператор передает управление начальной команде (*a*) цикла каждый раз, пока

$$(b+2)-(b+1)\geq 0,$$

т. е. пока

$$(b+1)\leq(b+2)$$

Но как только окажется, что

$$(b+1)>(b+2),$$

управление будет передано в ячейку *h*.

Если бы в ячейке β+1 получалась монотонно убывающая величина и циклу надлежало бы выполняться до тех пор, пока эта величина не станет меньше числа, запасенного в ячейке β +2, то оператор **X** мог бы иметь такой вид:

$$\begin{array}{llll} \alpha+\mu+1) & \beta+1 & \beta+2 & 0000\quad 03 \\ \alpha+\mu+2) & \alpha & h & 0000\quad 20. \end{array}$$

Если бы в ячейке β +1 получалась величина, которая при каждом выполнении цикла убывает по абсолютному значению (т. е. бесконечно малая), и требовалось бы, чтобы цикл повторялся до тех пор, пока не станет справедливым неравенство

$$(b+1)<(b+2),$$

где (*b*+2) — положительное малое число, то, переписав последнее неравенство следующим образом:

$$|b+1)|-|b+2)|<0,$$

мы могли бы составить оператор **X** в таком виде:

$$\begin{array}{llll} \alpha+\mu+1) & \beta+1 & \beta+2 & 0000\quad 04 \\ \alpha+\mu+2) & h & \alpha & 0000\quad 20. \end{array}$$

В заключение заметим, что строение цикла может быть значительно сложнее, чем только что описанное. Например, в состав цикла может входить несколько логических операторов, каждый из которых может прекратить выполнения цикла. Сами логические операторы, входящие в состав цикла, могут быть весьма разнообразны по содержанию проверяемых ими условий. Однако цикл повторяется всегда до тех пор, пока в результате его выполнений некоторое условие (или одно из нескольких условий) не изменит своего значения истинности.

**4. Логические шкалы.** Для управления повторениями цикла или для разветвления программы могут быть использованы *логические шкалы,* предложенные М. Р. Шура-Бура.

В простейшем случае логическая шкала занимает две ячейки. Одна ячейка содержит собственно логическую шкалу: специально подобранный набор нулей и единиц. Во второй ячейке помещен набор цифр, состоящий из единицы, записанной в знаковом разряде ячейки, и нулей в остальных разрядах. Это — *указатель шкалы*.

С помощью такой простейшей логической шкалы осуществляют передачу управления в одном из двух заданных направлений, если заранее известно, как должны чередоваться переходы в одном и в другом направлениях. Если в знаковом разряде ячейки, вмещающей шкалу, стоит 1, то передача управления производится в первом направлении, если же 0, то во втором направлении. После каждой передачи управления шкала сдвигается влево на один разряд.

П р и м е р 4. Написать программу для составления таблицы значений функции

$$y=\{[(x^4+1)^4+1]^4+1\}^4+1$$

при следующих значениях независимой переменной:

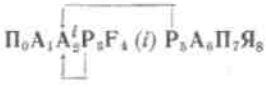
$$x=x_1,\ x_2,\ x_3,\ x_4,\ x_5,\ x_6,\ x_7,\ x_{10},\ x_{11},\ x_{12}.$$

Исходные данные расположим в памяти следующим образом:

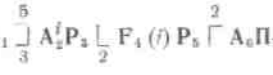
*b*) 1, *b*+1) $x_1$, …, *b*+12) $x_{12}$;

*l*+1) 1 (I, II, III), *l* + 2) 1 (I, III), *l* + 3) 0000 0000 0000 0 01.

Специальные рабочие ячейки в данном случае не потребуются. В качестве рабочих, а также ответных ячеек мы используем ячейки, хранящие исходные данные. Логическая схема программы будет иметь такой вид:



или



$\Pi_0$ и $A_1$ имеют обычный смысл; оператор $A_2$ ведет счет по формуле

$$z_i=x_i^4+1$$

и полученный результат записывает на место $x_i$; $P_3$ с помощью логической шкалы трижды возвращает управление оператору $A_2^i$, а на четвертый раз передает его оператору $F_4(i)$. После нового получения управления оператором $A_2^i$ оператор $P_3$ повторяет этот «цикл передачи управления». $F_4(i)$ переадресует изменяемые команды оператора $A_2^i$; $P_5$ контролирует конец вычислений. Вычисления прекращаются после того, как последняя единица логической шкалы уйдет влево за пределы разрядной сетки. $A_6$ переводит результаты в десятичную систему счисления; $\Pi_7$ выдает их на перфокарты; $Я_8$ останавливает машину.

Логическую шкалу и ее указатель разместим соответственно в ячейках $c + 1$ и $c + 2$. При составлении шкалы будем считать, что возврат управления оператору $\mathbf{A}_2^i$ обозначен цифрой 1, а передача управления оператору $\mathbf{F}_4(i)$ — цифрой 0. Сдвиг шкалы будем производить раньше проверки содержимого ее нулевого разряда, поэтому первую цифру шкалы поставим в первый разряд, а нулевой разряд, начальное содержимое которого не используется, для определенности займем нулем. Получим:

$(c + 1)$ = 011 101 110 111 011 101 11О 111 011 101 11О 111 0 111 000 = = 3567 3567 3567 0 70 (в коде команд);
$(c + 2)$ =100 000 000 000 000 000 000 000 000 000 000 000 0 000 000 = = 4000 0000 0000 0 00 (в коде команд).

П р о г р а м м а (адреса, зависящие от $i$, отмечены звездочкой):

Оператор $\mathbf{A}_2^i$:

| | | | | |
|---|---|---|---|---|
| $a$ +1) | $b + 1$* | $b$+ 1* | $b$+1* | 05 |
| $a + 2$) | $b + 1$* | $b + 1$* | $b$+1* | 05 |
| $a + 3$) | $b$ +1* | $b$ | $b$+1* | 01 |

Оператор $\mathbf{P}_3$:

| | | | | |
|---|---|---|---|---|
| $a + 4$) | $c + 1$ | $l + 3$ | $c$+ 1 | 14 |
| $a + 5$) | $c + 1$ | $c + 2$ | 0000 | 11 |
| $a + 6$) | $a + 1$ | $a + 7$ | 0000 | 20 |

Оператор $\mathbf{F}_4(i)$:

| | | | | |
|---|---|---|---|---|
| $a + 7$) | $a + 1$ | $l$+ 1 | $a + 1$ | 02 |
| $a$+10) | $a$ +2 | $l + 1$ | $a + 2$ | 02 |
| $a$+11) | $a$ +3 | $l + 2$ | $a + 3$ | 02 |

Оператор $\mathbf{P}_6$: (при $\omega = 0$ передает управление оператору $\mathbf{A}_6$ в ячейку $a + 14$):

| | | | | |
|---|---|---|---|---|
| $a$+ 12) | $c + 1$ | 0000 | 0000 | 16 |
| $a$+ 13) | $a$+14 | $a + 1$ | 0000 | 20 |

Логические шкалы могут осуществлять передачу управления и больше, чем в двух направлениях. При этом в указателе шкалы должна стоять не одна единица, а набор из нескольких единиц. Например, для передачи управления в одном из трех или в одном из четырех возможных направлений в указателе стоят две единицы в нулевом и первом разрядах, а в прочих разрядах — нули.

Работа шкалы состоит в этом случае в том, что производится сдвиг шкалы на два разряда влево, затем логическое умножение содержимого указателя на содержимое шкалы и сравнение выделенных при этом разрядов шкалы с запасенными в соответствующих ячейках наборами цифр. В зависимости от того, с каким набором будет происходить совпадение, управление передается в том или ином направлении.

Если число разрядов одной ячейки недостаточно для работы шкалы, то последняя может быть расположена в нескольких ячейках; при этом оператор, работающий с использованием логической шкалы, усложняется.

Наконец, в случае необходимости логическую шкалу можно восстанавливать, и тогда логическая шкала может быть использована многократно.

П р и м е р 5. Дана логическая схема программы

$$\Pi_0 A_1 \overbrace{A_2^i F_3(i)} P_4 A_5 \Pi_6 Я_7$$

или

$$\Pi_0 A_1 \overset{4}{\urcorner} A_2^i \overset{4}{\urcorner} F_3(i) P_4 \overset{2,3}{\ulcorner} A_5 \Pi_0 Я_7.$$

Оператор $\mathbf{P}_4$ должен с помощью логической шкалы передать управление вначале шесть раз оператору $\mathbf{A}_1^i$, потом один раз оператору $\mathbf{F}_3(i)$, снова два раза оператору $A_2^i$ и, наконец, один раз оператору $\mathbf{A}_5$ (первые команды операторов $\mathbf{A}_2^i$, $\mathbf{F}_3$ $(i)$. $\mathbf{P}_4$ и $\mathbf{A}_5$ хранятся соответственно в ячейках $a + 1$, $b$+ 1, $d + 1$, $e + 1$); построить логическую шкалу и написать команды, составляющие оператор $\mathbf{P}_4$.

Пусть 00 означает передачу управления оператору $\mathbf{A}_2^i$, 01 — оператору $\mathbf{F}_3(i)$, 10 — оператору $\mathbf{A}_5$. Тогда логическая шкала будет иметь такой вид:

$(c$+1)= 11 00 00 00 00 00 00 01 00 00 10 00000000000000000000.

Как и в предыдущем примере, начинать работу шкалы будем с ее сдвига. При этом нулевой и первый разряды шкалы не будут использованы. Для определенности мы заполнили их единицами (их содержимое безразлично). Указатель шкалы будет следующим:

$(c + 2)$= 11 00000000000000000000000000000000000000000000.

В качестве рабочей ячейки возьмем $c + 3$. В ячейке $c + 4$ поместим набор цифр

$(c + 4)$ = 01 00000000000000000000000000000000000000000000.

В ячейку $c$ + 5 запишем вспомогательное число для сдвига шкалы

$(c$+ 5) = 000 000 000 000 000 000 000 000 000 000 000 000 0 000 010.

Оператор $\mathbf{P}_4$:

| | | | | |
|---|---|---|---|---|
| $d + 1$) | $c$+ 1 | $c + 5$ | $c + 1$ | 14; |
| $d + 2$) | $c + 1$ | $c + 2$ | $c + 3$ | 11; |
| $d + 3$) | $d + 4$ | $a + 1$ | 0000 | 20; |
| $d + 4$) | $c + 3$ | $c + 4$ | 0000 | 16; |
| $d + 5$) | $b + 1$ | $e + 1$ | 0000 | 20. |

## § 35. Некоторые общие приемы операторного программирования

**1. Стандартные ячейки.** В некоторых случаях удобно, чтобы арифметический оператор имел стандартные входные ячейки, т. е. одну или несколько ячеек, в которые исходные данные заранее переносятся с тем, чтобы оператор брал их оттуда для своей работы.

Иногда стандартные ячейки применяются для уменьшения числа команд переадресации, чем достигается экономия ячеек, занятых программой, и рабочих тактов, а значит, и машинного времени (расходуемого на решение задачи).

Включение в программу оператора, переносящего числа из последовательности ячеек в стандартные ячейки (такой оператор называется засылкой и обозначается символом 3), и соответствующее изменение команд операторов, использующих эти числа, представляют собой один из приемов программирования, известный под названием *вынесения величин в стандартные ячейки.*

П р и м е р 1. Составить программу для вычисления таблицы значений функции

$$y = x\left\{\ln\left[\sqrt{x\sin(x+1)}+x\right]+x\right\}+e^{x}+\sqrt{x}=f(x) \qquad \text{(VI.33)}$$

для значений $x_1, x_2, ..., x_n$ независимой переменной $x$.

1. Составим сперва программу без применения стандартных ячеек. Пусть операторы $\mathbf{П}_0$ и $A_1$ имеют обычный смысл, оператор $\mathbf{A}_2^i$, вычисляет $y_i=f(x_i)$, $i = 1, 2, ..., n$; $\mathbf{F}_3$ $(i)$ переадресует $\mathbf{A}_2^i$; $\mathbf{P}_4$ контролирует, не окончено ли вычисление таблицы; $\mathbf{A}_5$ переводит результаты из двоичной в десятичную систему счисления; $\mathbf{П}_6$ переносит результаты на перфокарты; $\mathbf{Я}_7$ останавливает машину.

Логическая схема программы будет следующей:

$$\mathbf{П}_0\mathbf{A}_1\overleftarrow{\mathbf{A}_2^i\mathbf{F}_3\,(i)\,\mathbf{P}_4}\mathbf{A}_5\mathbf{П}_6\mathbf{Я}_7$$

или

$$\mathbf{П}_0\mathbf{A}_1\,{}^{4}\urcorner\,\mathbf{A}_2^i\mathbf{F}_3\,(i)\,\mathbf{P}_4\,{}^{2}\ulcorner\,\mathbf{A}_5\mathbf{П}_6\mathbf{Я}_7.$$

Распределение памяти:

$b$) 1, $b$+1) $x_1$ $b$ + 2) $x_2$, ..., $b + n$) $x_n$;

$d$+l) 1(1), $d$ + 2)1 (III).

П р о г р а м м а (адреса команд, зависящие от $i$, помечены звездочкой):

Оператор $\mathbf{A}_2^i$:

| | | | | |
|---|---|---|---|---|
| $a$+ 1) | $b$+1* | $b$ | $c$+1 | 01; |
| $a$+ 2) | $c$+1 | 0000 | $c$+1 | 67; |
| $a$+ 3) | $b$+1* | $c$ + 1 | $c$+1 | 05; |
| $a$+ 4) | $c$+1 | 0000 | $c$+1 | 63; |
| $a$+ 5) | $b$+1* | $c$ + 1 | $c$+1 | 01; |
| $a$+ 6) | $c$ + 1 | 0000 | $c$+1 | 66; |
| $a$+ 7) | $b$+1 * | $c$ + 1 | $c$+1 | 01; |
| $a$+10) | $b$+1 * | $c$ + 1 | $c$+1 | 05; |
| $a$+11) | $b$+1 * | 0000 | $c$+2 | 64; |
| $a$+12) | $c$ + 1 | $c$ + 2 | $c$+1 | 01; |
| $a$+13) | $b$+1 * | 0000 | $c$+2 | 63; |
| $a$+14) | $c$ + 1 | $c$ + 2 | $b$+1* | 01. |

Оператор $\mathbf{F}_3(i)$:

| | | | | |
|---|---|---|---|---|
| $a$+15) | $a$+ 1 | $d$ + 1 | $a$+ 1 | 02; |
| $a$+16) | $a$+ 3 | $d$ + 1 | $a$+ 3 | 02; |
| $a$+17) | $a$+ 5 | $d$ + 1 | $a$+ 5 | 02; |
| $a$+20) | $a$+ 7 | $d$ + 1 | $a$+ 7 | 02; |
| $a$+21) | $a$+10 | $d$ + 1 | $a$+10 | 02; |
| $a$+22) | $a$+11 | $d$ + 1 | $a$+11 | 02; |
| $a$+23) | $a$+13 | $d$ + 1 | $a$+13 | 02; |
| $a$+24) | $a$+14 | $d$ + 2 | $a$+14 | 02. |

Программирование операторов $\mathbf{P}_4$, $\mathbf{A}_5$, $\mathbf{П}_6$ и $\mathbf{Я}_7$ приводить не будем. В таблице 40 показано содержимое рабочих ячеек после первого выполнения каждой из команд оператора $A_2^i$.

Т а б л и ц а 40. Содержимое рабочих ячеек в процессе выполнения команд оператора $A_2^i$

| № команды | Содержимое ячеек | |
|---|---|---|
| | $c$+1 | $c$+2 |
| $a$+1 | $x$+1 | – |
| $a$+2 | $\sin(x+1)$ | – |
| $a$+3 | $x\sin(x+1)$ | – |
| $a$+4 | $\sqrt{x\sin(x+1)}$ | – |
| $a$+5 | $x+\sqrt{x\sin(x+1)}$ | – |
| $a$+6 | $\ln\,[x+\sqrt{x\sin(x+1)}\,]$ | – |
| $a$+7 | $x+\ln\,[x+\sqrt{x\sin(x+1)}\,]$ | – |
| $a$+10 | $x\{\,x+\ln\,[x+\sqrt{x\sin(x+1)}\,]\}$ | – |
| $a$+11 | $x\{\,x+\ln\,[x+\sqrt{x\sin(x+1)}\,]\}$ | $e^x$ |
| $a$+12 | $e^x+x\{\,x+\ln\,[x+\sqrt{x\sin(x+1)}\,]\}$ | $e^x$ |
| $a$+13 | $e^x+x\{\,x+\ln\,[x+\sqrt{x\sin(x+1)}\,]\}$ | $\sqrt{x}$ |
| $a$+14 | $e^x+x\{\,x+\ln\,[x+\sqrt{x\sin(x+1)}\,]\}$ | $\sqrt{x}$ |

2. Приведем теперь программу для случая применения стандартных входных ячеек. Логическая схема будет несколько иной:

$$\Pi_0 A_1 \overbrace{З_2^i A_3^i F_4\ (i)\ P_5} A_6 \Pi_7 Я_8$$

или

$$\Pi_0 A_1 \overset{5}{\urcorner} З_2^i A_3^i F_4\ (i)\ P_5 \overset{2}{\ulcorner} A_6 \Pi_7 Я_8.$$

Здесь операторы $\Pi_0$ и $A_1$ имеют обычное значение, $З_2^i$ переносит $x_i$ из ячейки $b + 1$ в стандартную входную ячейку оператора $A_3^i$, $A^i$ по $x_i$ вычисляет $f(x_i)$ и выдает результат в ячейку $b + i$; $F_4(i)$ переадресует $З_2^i$ и последнюю команду оператора $A_3^i$; $P_5$ контролирует конец вычисления таблиц; $A_6$ переводит результаты в десятичную систему; $\Pi_7$ переносит их на перфокарты; $Я_8$ — останов.

Размещение начальных данных и вспомогательных чисел остается прежним. Результаты по-прежнему будем записывать на месте исходных данных.

Стандартная входная ячейка оператора $A_3^i$ имеет номер $c + 3$, ячейки $c + 1$ и $c + 2$ являются рабочими ячейками.

П р о г р а м м а (звездочкой отмечены адреса, зависящие от параметра $i$):

Оператор $З_2^i$:

| | | | | |
|---|---|---|---|---|
| $a$+ 1) | $b$ +1* | 0000 | $c$+3 | 13 |

Оператор $A_3^i$:

| | | | | |
|---|---|---|---|---|
| $a$ + 2) | $c$+3 | $b$ | $c$+1 | 01 |
| $a$+ 3) | $c$+1 | 0000 | $c$+1 | 67 |
| $a$+ 4) | $c$+3 | $c$ + 1 | $c$+1 | 05 |
| $a$+ 5) | $c$+1 | 0000 | $c$+I | 63 |
| $a$+ 6) | $c$+3 | $c$ + 1 | $c$+1 | 01 |
| $a$+ 7) | $c$+1 | 0000 | $c$+1 | 66 |
| $a$+10) | $c$+3 | $c$ + 1 | $c$+1 | 01 |
| $a$+11) | $c$+3 | $c$ + 1 | $c$+1 | 05 |
| $a$+12) | $c$+3 | 0000 | $c$+2 | 64 |
| $a$+13) | $c$+1 | $c$ + 2 | $c$+1 | 01 |
| $a$+14) | $c$+3 | 0000 | $c$+2 | 63 |
| $a$+15) | $c$+1 | $c$ + 2 | $b$+1* | 01 |

Оператор $F_4(i)$:

| | | | | |
|---|---|---|---|---|
| $a$+16) | $a$+1 | $d$ + 1 | $a$+1 | 02 |
| $a$+17) | $a$+15 | $d$ + 2 | $a$+15 | 02. |

Программирование остальных операторов не приводим. Содержимое рабочих ячеек при выполнении этой программы приведено в таблице 41.

Т а б л и ц а 41. Содержимое рабочих ячеек в процессе выполнения операторов $З_2^i$ и $A_3^i$

| № команды | Содержимое ячеек | | |
|---|---|---|---|
| | $c$+3 | $c$+1 | $c$+2 |
| $a$+1 | $x_1$ | – | – |
| $a$+2 | $x_1$ | $x_1$+1 | – |
| $a$+3 | $x_1$ | $\sin(x_1+1)$ | – |
| $a$+4 | $x_1$ | $x_1 \sin(x_1+1)$ | – |
| $a$+5 | $x_1$ | $\sqrt{x_1 \sin(x_1+1)}$ | – |
| $a$+6 | $x_1$ | $x_1+\sqrt{x_1 \sin(x_1+1)}$ | – |
| $a$+7 | $x_1$ | $\ln [x_1+\sqrt{x_1 \sin(x_1+1)}\,]$ | – |
| $a$+10 | $x_1$ | $x_1+\ln [x_1+\sqrt{x_1 \sin(x_1+1)}\,]$ | – |
| $a$+11 | $x_1$ | $x_1\{ x_1+\ln [x_1+\sqrt{x_1 \sin(x_1+1)}\,]\}$ | – |
| $a$+12 | $x_1$ | $x_1\{ x_1+\ln [x_1+\sqrt{x_1 \sin(x_1+1)}\,]\}$ | $e^{x_1}$ |
| $a$+13 | $x_1$ | $e^{x_1}+x_1\{ x_1+\ln [x_1+\sqrt{x_1 \sin(x_1+1)}\,]\}$ | $e^{x_1}$ |
| $a$+14 | $x_1$ | $e^{x_1}+x_1\{ x_1+\ln [x_1+\sqrt{x_1 \sin(x_1+1)}\,]\}$ | $\sqrt{x_1}$ |
| $a$+15 | $x_1$ | $e^{x_1}+x_1\{ x_1+\ln [x_1+\sqrt{x_1 \sin(x_1+1)}\,]\}$ | $\sqrt{x_1}$ |

Сравнивая две приведенные программы, мы замечаем некоторую экономию команд переадресации за счет использования стандартных ячеек. Эта экономия была бы еще заметнее, если бы в операторе $A_2^i$ первой программы было больше команд, подлежащих переадресации.

**2. Операторы-подпрограммы.** Стандартные входные ячейки также часто необходимы еще в случае, когда оператор является, по существу, подпрограммой, к которой приходится обращаться из разных мест программы. В *этом* случае он, кроме входных, имеет также стандартные выходные ячейки.

Обращение к элементарному или обобщенному оператору, обозначим его буквой **R**, осуществляется с помощью команды № 27 (см. таблицу 25 и § 32, стр. 160).

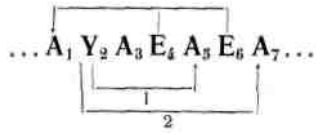
*а а с 27,*

которая сама становится оператором.

Оператор **R**, к которому мы обращаемся из разных мест программы, должен оканчиваться «нулевой командой»,

т. е. ячейкой, заполненной нулями. В эту ячейку при обращении к **R** происходит запись команды возврата к основной программе*. При программировании важно не забыть о необходимости «нулевой» команды.

Посмотрим для примера логическую схему




(VI.34)

Здесь $\mathbf{A}_1$ — оператор, к которому производится обращение; $\mathbf{Y}_2$ — «нулевая» команда; $\mathbf{A}_3$, $\mathbf{A}_5$ и $\mathbf{A}_7$ — арифметические операторы; $\mathbf{E}_4$, и $\mathbf{E}_6$ — команды обращения к $A_{1t}$ записывающие в ячейку** $\langle\mathbf{Y}_2\rangle$ команду возврата.

После первой работы оператора $\mathbf{A}_1$ $\mathbf{Y}_2$ представляет собой нулевую команду и передает управление оператору $\mathbf{A}_3$.

При обращении $\mathbf{E}_4$ к $\mathbf{A}_1$ одновременно в ячейку $\langle\mathbf{Y}_2\rangle$ производится запись команды возврата к тому месту программы, которое следует за $\mathbf{E}_4$, т. е. к оператору $\mathbf{A}_5$.

После второй работы оператора $\mathbf{A}_1$ $\mathbf{Y}_2$ передает управление оператору $\mathbf{A}_5$, что на схеме показано стрелкой с номером 1. При обращении $\mathbf{E}_6$ к оператору $\mathbf{A}_1$ в ячейку $\langle\mathbf{Y}_2\rangle$ будет записана команда возврата к оператору $\mathbf{A}_7$. Значит, $\mathbf{Y}_2$ после третьей работы $\mathbf{A}_1$ передаст управление к оператору $\mathbf{A}_7$, что на логической схеме программы показано стрелкой с номером 2.

Таким образом, команда $\mathbf{Y}_2$ передает управление в различных направлениях, в зависимости от того, откуда получал управление оператор $\mathbf{A}_1$.

Для упрощения логической схемы программы нулевую команду $\mathbf{Y}_2$ можно присоединить к оператору $\mathbf{A}_1$ Получающийся при этом новый оператор обозначим через $\overline{A_1}$ (черта над буквой означает наличие в конце оператора нулевой команды).

Обращение **E** к оператору $\overline{A_1}$ обозначим символом **E(1)**. Это избавит нас от необходимости проводить большое количество стрелок, показывающих передачу управления в логической схеме. Вместо логической схемы (VI. 34) теперь составим логическую схему следующего вида:

$$\ldots \overline{\mathrm{A}}_1 \mathrm{A}_2 \mathrm{E}_3 (1)\ \mathrm{A}_4 \mathrm{E}_5 (1)\ \mathrm{A}_6 \ldots \qquad \text{(VI .35)}$$

(нумерация операторов изменилась за счет того, что команда $\mathbf{Y}_2$ в новой схеме явно не записана).

В этом примере управление передается оператором обращения $\mathbf{E}_3$ тому оператору схемы, после которого расположена «свободная» ячейка, предназначенная для занесения в нее команды возврата.

Если оператор, к которому производится обращение, имеет номер *m,* то обращение к нему будем обозначать символом $[\mathbf{E}_k(m)]$. В некоторых случаях управление передают одному оператору (обозначим его номер буквой *m*), а команду возврата записывают в свободную ячейку, расположенную вслед за последней ячейкой другого оператора (номер которого обозначим через *n*). В этом случае оператор обращения удобно обозначать символом

$$\mathbf{E}_k(m; n).$$

Например, если в схеме имеется группа операторов

$$\rightarrow \mathrm{A}_5^j \mathrm{F}_6 (i)\ \overline{\mathrm{A}}_7 \rightarrow,$$

то обращение к этой группе обозначают символом $\mathbf{E}_k$ (5; 7).

В некоторых случаях логический оператор оканчивается командой условного перехода не первого, а второго типа. При этом в зависимости от выполнения или невыполнения условия, проверяемого логическим оператором, производится обращение к той или иной группе операторов (обычно одна из таких групп является частью другой, причем обе группы заканчиваются одним и тем же оператором). Удобно для уменьшения количества знаков передачи управления в логической схеме программы изображать такие условные обращения следующим образом:

$$\mathbf{P}_k(m_1, m_2; n),$$

где $m_1$ — номер оператора, которому передается управление при ω=0, $m_2$ — номер оператора, которому передается управление ω =1, *n* — номер оператора, после последней команды которого расположена «нулевая» команда, вместо которой при условном обращении происходит запись команды возврата.

При этом логические схемы программ могут выглядеть следующим образом:

$$\ldots\mathbf{A}_{10}\mathbf{P}_{11}(22, 21, 23)\mathbf{A}_{12}\ldots\mathbf{A}_{21}\mathbf{A}_{22}\mathbf{A}_{23}\ldots;$$
$$\ldots\mathbf{P}_{12}(17, 19, 19)\mathbf{A}_{13}\ldots\mathbf{A}_{17}\mathbf{A}_{18}\mathbf{A}_{19}\ldots$$

В качестве программы, содержащей обращения к операторам, использующим стандартные ячейки, приведем программу для решения дифференциального уравнения методом Эйлера с пересчетом.

Пример 2. Дано дифференциальное уравнение

$$y' = \frac{10}{x^2 + y^2} + x = f(x, y) \qquad \text{(VI.36)}$$

Требуется найти методом Эйлера с пересчетом его частное решение,удовлетворяющее начальному условию

---

* Например, если первая команда оператора **R** имеет номер *a*+ 1, его последняя команда *a*+ *k*— 1, а в ячейке *a* + *k* записаны нули, и если команда обращения стоит в ячейке № *b,* то команда обращения имеет такой вид: *b*) *a* + 1 *a* + 1 *a* + *k* 27. По этой команде получает управление команда (*a* + 1), а в ячейку *a* + *k* производится запись команды возврата *a* + *k*) *b* + 1 *b*+1 *a* + *k* 20. Команда возврата — самогасящаяся, после своего исполнения она себя «зачеркивает» — заменяет нулями.

** Напоминаем, что угловые скобки применяются для обозначения той ячейки, в которой хранится величина или команда, заключенная в эти скобки.

$$y_{x=1} = 1 \qquad \text{(VI.37)}$$

и определенное на промежутке

$$1 \le x \le 10 \qquad \text{(VI.38)}$$

Шаг интегрирования взят следующий:

$$k = 0{,}01. \qquad \text{(VI.39)}$$

Как известно, интегрирование дифференциального уравнения методом Эйлера с пересчетом состоит в последовательных вычислениях по формулам:

$$\text{а)}\quad y'_i = f(x_i, y_i); \qquad \text{г)}\quad \bar{y}'_{i+1} = f(x_{i+1}, \bar{y}_{i+1});$$

$$\text{б)}\quad \bar{y}'_{i+1} = y_i + hy_i; \qquad \text{д)}\quad \theta_i = \frac{1}{2}(y'_i + \bar{y}'_{i+1}); \qquad \text{(VI.40)}$$

$$\text{в)}\quad x_{i+1} = x_i + h; \qquad \text{е)}\quad y_{i+1} = y_i + h\theta_i,$$

где

$$i = 0, 1, 2, \ldots, n-1.$$

Логическая схема программы

Пусть оператор $\mathbf{A_1}$ вычисляет по формуле

$$y' = f(x, y) = \frac{10}{x^2 + y^2} + x$$

величины (VI 40а) и (VI.40г), а оператор $\bar{\mathbf{A}}_2$ вычисляет по формуле $y_{i+1} = y_i + hy'_i$ величины (VI.406) и (VI. 40е). Оба эти оператора будут иметь стандартные входные и выходные ячейки. Ряд операторов переноса передает исходные данные в стандартные входные ячейки и пересылает промежуточные результаты из входных ячеек на хранение. К операторам $\bar{\mathbf{A}}_1$, $\bar{\mathbf{A}}_2$ будут производиться обращения.

Приведем описание работы всех операторов цикла программы при первом его действии в виде таблицы 42.

Таблица 42. Работа операторов цикла программы при первом его выполнении

| Оператор | Его назначение | Рабочие ячейки | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | *b*–1 | *b* | *b*+1 | *c*+1 | *c*+2 | *c*+3 | *c*+4 | *c*+5 | *c*+6 |
| | | $x_0$ | $y_0$ | | | | | | | |
| $\mathbf{П}_0$ | Перенос исходных данных в стандартные ячейки | $x_0$ | $y_0$ | | $X_0$ | $Y_0$ | | | | |
| $\overline{\mathbf{A}}_1$ | Счет по формуле $(c+3) = \frac{10}{(c+1)^2 + (c+2)^2} + (c+1)$ | | | | $x_0$ | $y_0$ | $Y'_0$ | | | |
| $\overline{\mathbf{A}}_2$ | Счет по формуле $(c+4) = (c+2) + h(c+3)$ | | | | | $y_0$ | $y'_0$ | $\bar{Y}_1$ | | |
| $\mathbf{П}_3$ | Перенос $y_0$ и $y'_0$ на хранение. Перенос $y_1$ в стандартную ячейку $c$+2. Вычисление $x_1 = x_0 + h$ | | | | $x_0$ $X_1$ | $y_0$ $\bar{Y}_1$ | $y'_0$ | $\bar{y}_1$ | $Y_0$ | $Y'_0$ |
| $\mathbf{E}_4(1)$ | Обращение к $\overline{\mathbf{A}}_1$ | | | | $x_1$ | $\bar{y}_1$ | $\bar{Y}'_1$ | | | |
| $\mathbf{A}_5$ | Счет по формуле $(c+3) = \frac{1}{2}[(c+3) + (c+6)]$ | | | | | | $\bar{y}'_1$ $\Theta_0$ | | | $y'_0$ |
| $\mathbf{П}_6$ | Перенос $y_0$ во входную ячейку оператора $\overline{\mathbf{A}}_2$ | | | | | $Y_0$ | | | $y_0$ | |
| $\mathbf{E}_7(2)$ | Обращение к $\overline{\mathbf{A}}_2$ | | | | | $y_0$ | $\Theta_0$ | $Y_1$ | | |
| $\mathbf{З}^{*i}_8$ | Перенос $y_1$ во входную ячейку $\overline{\mathbf{A}}_1$ и выдача результата на хранение (обратная засылка) | | | $Y_1$ | | $Y_1$ | | $y_1$ $y_1$ | | |

В этой таблице малыми буквами набраны исходные данные для работы операторов, а заглавными — результаты их работы. Ячейки $c$ + 1 и $c$ + 2 являются стандартными входными ячейками оператора $\bar{\mathbf{A}}_1$; ячейка $c$ + 3 — его стандартная выходная ячейка; ячейки $c$ + 2 и $c$ + 3 — стандартные входные ячейки оператора $\bar{\mathbf{A}}_2$, ячейка $c$ + 4 — его стандартная выходная ячейка. Операторы ввода и перевода исходных данных опускаем. Логическая схема имеет такой вид:

$$\mathbf{П}_0\overline{\mathbf{A}_1\mathbf{A}_2}\mathbf{П}_3\mathbf{E}_4\,(1)\,\mathbf{A}_5\mathbf{П}_6\mathbf{E}_7\,(2)\,\mathbf{З}^{*i}_8\mathbf{F}_9\,(i)\,\mathbf{P}_{10}\mathbf{A}_{11}\mathbf{П}_{12}\mathbf{Я}_{13}$$

или

$$\mathbf{П}_0\,{}^{10}\!\urcorner\,\overline{\mathbf{A}}_1\overline{\mathbf{A}}_2\mathbf{П}_3\mathbf{E}_4\,(1)\,\mathbf{A}_5\mathbf{П}_6\mathbf{E}_7\,(2)\mathbf{З}^{*i}_8\mathbf{F}_9\,(i)\,\mathbf{P}_{10}\,{}^{1}\!\ulcorner\,\mathbf{A}_{11}\mathbf{П}_{12}\mathbf{Я}_{13}.$$

Оператор $\mathbf{F}_9(i)$ производит переадресацию оператора $\mathbf{З}^{*i}_8$; $\mathbf{P}_{10}$ управляет выполнениями цикла; $\mathbf{A}_{11}$ переводит результаты в десятичную систему; $\mathbf{П}_{12}$ переносит их на перфокарты; $\mathbf{Я}_{13}$ останавливает машину.

Распределение памяти:

$b-6)$ $c$+4 0000 $b+n+1$ 13, $b-5)$1 (Ш), $b-4)$ 1/2, $b-3)10$, $b-2)h$, $b-1)x_0$, $b)y_0$,

для результатов отведем ячейки

$$b+1, b+2, \ldots, b+n$$

Здесь $n = 1604$ (восьмеричное).

Программирование:

| | | | | | |
|---|---|---|---|---|---|
| Оператор $\mathbf{П}_0$: | $a+1$) | $b-1$ | 0000 | $c+1$ | 13; |
| | $a+2$) | $b$ | 0000 | $c+2$ | 13. |
| Оператор $\bar{\mathbf{A}}_1$: | $a+3$) | $c+1$ | $c+1$ | $c+3$ | 05; |
| | $a+4$) | $c+2$ | $c+2$ | $c+4$ | 05; |
| | $a+5$) | $c+3$ | $c+4$ | $c+3$ | 01; |
| | $a+6$) | $c+3$ | 0000 | $c+3$ | 62; |
| | $a+7$) | $c+3$ | $b-3$ | $c+3$ | 05; |
| | $a+10$) | $c+1$ | $c+3$ | $c+3$ | 01; |
| | $a+11$) | 0000 | 0000 | 0000 0 | 00. |
| Оператор $\bar{\mathbf{A}}_2$: | $a+12$) | $b-2$ | $c+3$ | $c+4$ | 05; |
| | $a+13$) | $c+2$ | $c+4$ | $c+4$ | 01; |
| | $a+14$ | 0000 | 0000 | 0000 | 00. |
| Оператор $\mathbf{П}_3$: | $a+15$) | $c+2$ | 0000 | $c+5$ | 13; |
| | $a+16$) | $c+3$ | 0000 | $c+6$ | 13; |
| | $a+17$) | $c+4$ | 0000 | $c+2$ | 13; |
| | $a+20$) | $c+1$ | $b-2$ | $c+1$ | 01. |
| Оператор $\mathbf{E}_4$: | $a+21$) | $a+3$ | $a+3$ | $c+11$ | 27. |
| Оператор $\mathbf{A}_5$: | $a+22$) | $c+3$ | $c+6$ | $c+4$ | 01; |
| | $a+23$) | $c+4$ | $b-4$ | $c+3$ | 05. |
| Оператор $\mathbf{П}_6$: | $a+24$) | $c+5$ | 0000 | $c+2$ | 13. |
| Оператор $\mathbf{E}_7$: | $a+25$) | $a+12$ | $a+12$ | $a+14$ | 27. |
| Оператор $\mathbf{З}^{*'}{}_8^i$: | $a+26$) | $c+4$ | 0000 | $c+2$ | 13; |
| | $a+27$) | $c+4$ | 0000 | $b+1$* | 13. |
| Оператор $\mathbf{F}_9(i)$: | $a+30$) | $a+27$ | $b-5$ | $a+27$ | 02. |
| Оператор $\mathbf{P}_{10}$: | $a+31$) | $a+27$ | $b-6$ | 0000 | 16; |
| | $a+32$) | $a+33$ | $a+3$ | 0000 | 20. |

**3. Циркуляция величин в стандартных ячейках.** В некоторых случаях достигается экономия команд переадресации и рабочих тактов машины с помощью приема, называемого *циркуляцией* (величин) в стандартных ячейках.

Сущность циркуляции состоит в том, что с каждым повторением цикла производятся переносы чисел из одной группы ячеек в другую. Так, в примере 1 § 33 оператор $\mathbf{П_2}$ фактически производит циркуляцию величин в ячейках $b+1$, $b+2$,..., $b+n$. Циркуляция бывает выгодной при вычислении выражений вида

$$\Phi(x_i, y_i, \ldots, w_i) = F\,[\varphi(x_i, y_i, \ldots, w_i); \varphi(x_{i+1i}, y_{i+1}, \ldots, w_{i+1}); \ldots; \varphi(x_k, y_k, \ldots, w_k)]$$

Экономия рабочих тактов машины и команд переадресации достигается так.

После того, как при вычислении $\Phi\,(x_{i-1}, y_{i-1}, \ldots, w_{i-1})$ числа

$$\varphi(x_{i-1}, y_{i-1}, \ldots, w_{i-1}),\quad \varphi(x_i, y_i, \ldots, w_i),\quad \ldots,\quad \varphi(x_{k-1}, y_{k-1}, \ldots, w_{k-1})$$

уже получены, для вычисления $\Phi\,(x_i, y_i, \ldots, w_i)$ не хватает числа $\varphi(x_k, y_k, \ldots, w_k)$.

Числа $\varphi(x_i, y_i, \ldots, w_i)$; $\varphi(x_{i+1i}, y_{i+1}, \ldots, w_{i+1})$; …; $\varphi(x_{k-1}, y_{k-1}, \ldots, w_{k-1})$ перемещают в стандартных ячейках; недостающее число $\varphi(x_k, y_k, \ldots, w_k)$ вычисляют и записывают в ту стандартную ячейку, которая до переноса хранила $\varphi(x_{k-1}, y_{k-1}, \ldots, w_{k-1})$; имея теперь все необходимое, вычисляют очередное значение $\Phi(x_i, y_i, \ldots, w_i)$.

Описанный цикл повторяют до тех пор, пока не получат всех нужных значений величины **Ф**.

Без применения циркуляции пришлось бы либо при получении каждого значения **Ф** вычислять все нужные для этого величины φ, либо расставлять числа φ в последовательность ячеек и производить переадресации как той команды, которая расставляла бы эти числа в последовательность ячеек, так и всех тех команд, которые брали бы числа φ из последовательности ячеек для вычислений.

Пример 3. Вычислить с помощью параболической формулы (формулы Симпсона) определенный интеграл

$$\int_\alpha^\beta y dx = \int_\alpha^\beta \ln(x + \sqrt{1 + x^2})\,dx\,.$$

Пусть интервал интегрирования разделен на $2n$ частей равной длины (обозначим длину такой части через $h$).

Как известно, параболическая формула численного интегрирования дает (см. стр. 411):

$$\int_\alpha^\beta y dx = S_n\,,$$

где $S_n$ получено путем вычислений по формулам

$$S_0 = 0;\quad S_{i+1} = S_i + \Delta S_i;\quad \Delta S_i = h/3(y_{2i} + 4y_{2i+1} + y_{2i+2});\quad i = 0, 1. 2, \ldots, n\text{-}1$$

Исходные данные разместим следующим образом ($\alpha = x_0$):

$$b+1)\ \alpha = x_0,\quad b+2)\ h,\quad b+3)\ 1,\quad b+4)\ h/3,\quad b+5)\ 4;$$
$$d+1)\ 0,\quad d+2)\ 1\ (\text{I}),\quad d+3)\ 0,\quad d+4)\ n\ (\text{I}).$$

Пусть оператор $\mathbf{A}_1$ производит вычисления по формуле

$$y = \ln(x + \sqrt{x^2 + 1})$$

и записывает результат в рабочую ячейку $c+4$. Оператор $\mathbf{Ц}_2$ производит циркуляцию, перенося число $(c+2)$ в ячейку $c+1$, число $(c+3)$ в ячейку $c+2$ и число $(c+4)$ в ячейку $c+3$. Оператор $\mathbf{P}_3$ в начале работы программы заставляет операторы $\mathbf{A}_1$ и $\mathbf{Ц}_2$ проработать трижды, а

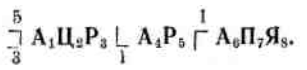
затем при каждом новом получении управления оператором $\mathbf{A}_1$ заставляет их работать дважды. Оператор $\mathbf{A}_4$ по числам

$$y_{2i} = (c + 1), \quad y_{2i+1} = (c + 2) \quad \text{и} \quad y_{2i+2} = (c + 2)$$

вычисляет $\Delta S_i$, где $i = 0, 1, 2, ..., n - 1$, и прибавляет полученную величину к содержимому ячейки $d + 1$. В ячейку $d + 1$ предварительно записан нуль. В конце работы программы в ячейке будет получен искомый интеграл. $\mathbf{P}_5$ проверяет с помощью счетчика, расположенного в ячейках $d + 2$, $d + 3$ и $d + 4$, не окончен ли процесс численного интегрирования. $\mathbf{A}_6$ переводит ответ в десятичную систему счисления; $\mathbf{П}_7$ выдает его на перфокарту; $\mathbf{Я}_8$ останавливает машину. Операторы ввода программы и перевода исходных данных в двоичную систему счисления нами опущены.

Логическая схема программы


или




Программа

| | | | | | |
|---|---|---|---|---|---|
| Оператор $\mathbf{A}_1$ | $a$+ 1) | $b + 1$ | $b + 1$ | $c + 1$ | 05; |
| | $a$+ 2) | $b + 3$ | $c + 1$ | $c + 1$ | 01; |
| | $a$+ 3) | $c + 1$ | 0000 | $c + 1$ | 63; |
| | $a$+ 4) | $b + 1$ | $c + 1$ | $c + 1$ | 01; |
| | $a$+ 5) | $c + 1$ | 0000 | $c + 4$ | 66; |
| | $a$+ 6) | $b + 1$ | $b + 2$ | $b + 1$ | 01. |
| Оператор $\mathbf{Ц}_2$ | $a$+ 7) | $c + 2$ | 0002 | $c + 1$ | 45. |
| Оператор $\mathbf{P}_3$ | $a$+10) | $b + 1$ | $b + 1$ | $b$ +10 | 20; |
| | $a$+11) | $b + 1$ | $b + 1$ | $b$ +10 | 27. |
| Оператор $\mathbf{A}_4$ | $a$+12) | $c + 2$ | $b + 5$ | $c + 5$ | 05; |
| | $a$+13) | $c + 1$ | $c + 5$ | $c + 5$ | 01; |
| | $a$+14) | $c + 3$ | $c + 5$ | $c + 5$ | 01; |
| | $a$+15) | $b + 4$ | $c + 5$ | $c + 5$ | 05; |
| | $a$+16) | $c + 5$ | $d + 1$ | $d + 1$ | 01. |
| Оператор $\mathbf{P}_5$ | $a$+17) | $d + 3$ | $d + 2$ | $d + 3$ | 02; |
| | $a$+20) | $d + 3$ | $d + 4$ | 0000 | 16; |
| | $a$+21) | $b$ +22 | $a + 1$ | 0000 | 20. |

**4. Операторы формирования.** Бывают случаи, когда при составлении программы отдельные команды не могут быть составлены, потому что адреса (или, в более редких случаях, коды операций) этих команд заранее неизвестны и станут известными только после того, как часть программы будет выполнена машиной.

Можно было бы в программе поставить команду выдачи на перфокарты результатов, необходимых для составления таких заранее неизвестных команд и команду останова. В ходе решения задачи на машине пришлось бы (после останова) составить нужные команды и ввести их в соответствующие ячейки памяти, а затем снова пустить машину. Однако такое решение вопроса нельзя признать удовлетворительным хотя бы потому, что остановы машины в процессе решения задачи для составления и ввода в память недостающих команд могут значительно увеличить расход машинного времени.

Составление упомянутых выше заранее неизвестных команд следует возложить на саму машину, для чего в программу должен быть включен специальный оператор (или несколько операторов). Операторы, составляющие команды программы, получили название *операторов формирования.* Не следует смешивать операторы формирования с командами или операциями формирования (код операции формирования 13). Для составления команд, конечно, может быть использована операция формирования, но могут быть использованы и любые другие операции, входящие в систему операций машины (например, операции 02, 15, 06, 07 и др.).

Выясним работу оператора формирования на следующем примере. Предположим, что обобщенный оператор $\mathbf{Q}_1$ вычисляет некоторую монотонно возрастающую последовательность чисел

$$x_1, \ x_2, \ \ldots, x_n$$

и записывает эти числа в последовательные ячейки

$$b + 1, \quad b + 2, \quad \ldots, \quad b + n.$$

Требуется составить часть программы, следующую за обобщенным оператором $\mathbf{Q}_1$, которая выбирала бы из последовательности

$$x_1, \ x_2, \ \ldots, x_n$$

подпоследовательность чисел, удовлетворяющих неравенству

$$x_i \geq k$$

и переносила бы их в ячейки, начиная с $c$+1.

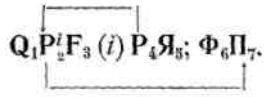
Пусть оператор $\mathbf{P}_2^{\,i}$ проверяет условие

$$x_i < k.$$

Если это условие выполнено, он передает управление оператору переадресации $\mathbf{F}_3(i)$, если не выполнено, то управление передается оператору $\mathbf{Ф}_6$. Оператор $\mathbf{P}_4$ проверяет, все ли числа $x_1, \ x_2, \ \ldots, \ x_n$ уже просмотрены. Если все эти числа просмотрены и искомая подпоследовательность не обнаружена, то оператор $\mathbf{Я}_5$ производит останов машины. Оператор $\mathbf{Ф}_6$ формирует команды оператора $\mathbf{П}_7$, который переносит числа выбранной подпоследовательности в ячейки $c$+1, $c$+2, ...

Команду переноса чисел выбранной подпоследовательности в ячейки $c$+1, $c$+2, ... нельзя составить заранее, так как неизвестно, в какой ячейке стоит первое из чисел $x_i$ удовлетворяющее неравенству $x_i \geq k$, и сколько чисел удовлетворяют такому неравенству.

Логическая схема составляемой части программы имеет такой вид:




Предположим, что необходимые исходные данные расположены в памяти машины следующим образом:

$$b)\ k,\quad b+1)\ x_1,\quad b+2)\ x_2,\quad \ldots,\quad b+n)x_n;$$

$$e+1)\ 1(1),\quad e+2)\ (n-1)\ (\mathrm{I}),\quad e+3)\ 7777(1),\quad e+4)\ (b+n)\ (\mathrm{I}),\quad e+5)\ 0000\ 0000\ c+1\ 45.$$

Программа:

$\mathbf{P}_2^i$:

| | | | | |
|---|---|---|---|---|
| $a+1)$ | $a+1^*$ | $b$ | 0000 | 03; |
| $a+2)$ | $a+7$ | $a+3$ | 0000 | 20. |

$\mathbf{F}_3(i)$:

| | | | | |
|---|---|---|---|---|
| $a+3)$ | $a+1$ | $e+1$ | $a+1$ | 02. |

$\mathbf{P}_4$:

| | | | | |
|---|---|---|---|---|
| $a+4)$ | $e+2$ | $e+1$ | $e+2$ | 03. |
| $a+5)$ | $a+1$ | $a+6$ | 0000 | 20. |

$\mathbf{Я}_5$:

| | | | | |
|---|---|---|---|---|
| $a+6)$ | 0000 | 0000 | 0000 | 40. |

$\mathbf{Ф}_6$:

| | | | | |
|---|---|---|---|---|
| $a+7)$ | $a+1$ | $e+3$ | $c+1$ | 11; |
| $a+10)$ | $e+4$ | $c+1$ | $c+2$ | 15; |
| $a+11)$ | $c+2$ | 4114 | $c+2$ | 14; |
| $a+12)$ | $c+1$ | $c+2$ | $c+1$ | 13; |
| $a+13)$ | $c+1$ | $e+5$ | $a+14$ | 13. |

$\mathbf{П}_7$:

| | | | | |
|---|---|---|---|---|
| $a+14)$ | 0000 | 0000 | 0000 | 00. |

Оператор $\mathbf{П}_7$ состоит из нулевой команды, так как нужная команда заранее неизвестна. Она будет своевременно составлена оператором $\mathbf{Ф}_6$. Оператор $\mathbf{Ф}_6$ выделяет первый адрес команды $(a+1)$. К моменту работы оператора $\mathbf{Ф}_6$ этот адрес равен $b + i'$ — номеру ячейки, содержащей первое из чисел, удовлетворяющих неравенству $x_{i'} \geq k$.. Вычитая этот адрес из числа $(b+n)$ (I), находящегося в ячейке $e+4$, оператор $\mathbf{Ф}_6$ получает:

$$[b+n-(b+i')](I) = (n-i')(I)$$

т. е. представленное в виде единиц первого адреса число, на единицу меньшее, чем количество элементов найденной подпоследовательности. Оператор $\mathbf{Ф}_6$ передвигает это число во второй адрес и формирует из $(b+i')$ (I), $(n-i')$ (II) и содержимого ячейки $e+5$ команду

$$b+i'\quad n-i'\quad c+1\quad 45,$$

которую и записывает в ячейку $a+14$.

По этой команде, представляющей оператор $\mathbf{П}_7$, машина переносит подпоследовательность $x'_i, x'_{i+1}, \ldots, x_n$ в ячейки $c+1, c+2, \ldots, c+n-i'$.

В операторе $\mathbf{Ф}_6$ вместо операции формирования (13) при формировании команд можно использовать операцию специального сложения (02). При этом $\mathbf{Ф}_6$ будет иметь следующий вид:

| | | | | |
|---|---|---|---|---|
| $a+7)$ | $a+1$ | $e+3$ | $c+1$ | 11; |
| $a+10)$ | $e+4$ | $c+1$ | $c+2$ | 15; |
| $a+11)$ | $c+2$ | 4114 | $c+2$ | 14; |
| $a+12)$ | $c+1$ | $c+2$ | $c+1$ | 02; |
| $a+13)$ | $e+5$ | $c+1$ | $a+14$ | 02. |

С операторами формирования читатель встретится в дальнейшем в примерах подпрограмм выбора значений функции из таблицы (см. § 45).

Обращаем внимание читателя на оригинальный оператор $\mathbf{P}_4$, проверяющий, все ли числа $x_1, x_2, \ldots, x_n$ просмотрены оператором $\mathbf{P}_2^i$.

Оператор $\mathbf{P}_4$ вычитает из числа $(n-1)(1) = n-1\ 0000\ 0000\ 0\ 00$ с помощью операции 03 число $1(1) = 0001\ 0000\ 0000\ 0\ 00$. Результат операции 03 всегда нормализуется, так что после первой работы оператора $\mathbf{P}_4$ в ячейке $e+2$ будет стоять не $(n-2)(1)$, а некоторое число, полученное в результате выполнения операции 03.

Этот результат будет неотрицателен, так что после выполнения команды $(a+1)$ вырабатывается сигнал $\omega=0$.

При $(n-1)$-м выполнении оператора $\mathbf{P}_4$ в ячейке $e+2$ будет получен нуль и сигнал $\omega$ все еще будет равен единице. И лишь при $n$-м выполнении оператора $\mathbf{P}_4$ в ячейке $e+2$ получится отрицательное число и будет выработан сигнал $\omega=1$.

## § 36. Программирование для машин с фиксированной запятой

**1. Программирование для одноадресной машины.** Основные принципы программирования одинаковы для трехадресных и одноадресных (и других) электронных цифровых вычислительных машин. Не зависят они и от того,

относится машина к типу машин с фиксированной или с плавающей запятой.

Однако принадлежность машины к типу машин с фиксированной запятой обусловливает некоторые специфические особенности процесса подготовки задачи для решения. Эти особенности состоят в необходимости подбора масштабных коэффициентов для всех участвующих в задаче величин (см. § 6).

Использование в машине системы одноадресных команд также налагает некоторый отпечаток на программирование. Программы ряда задач для одноадресной машины значительно длиннее, чем для трехадресной, так как для выполнения многих операций требуется три одноадресные команды вместо одной трехадресной. Но существуют задачи, программы которых для одноадресной и трехадресной машин содержат почти одинаковое количество команд. Это — задачи, сводящиеся к таким вычислениям, при которых одной из исходных величин каждой арифметической операции является результат предыдущей операции. При этом количество одноадресных команд сокращается за счет того, что отсутствует необходимость переносов исходных величин операций из памяти машин в сумматор и переносов результатов операций из сумматора в память машины (для хранения).

Для уяснения особенностей подготовки задач при решении на одноадресной машине с фиксированной запятой приведем пример на программирование для машины *Урал.*

Пусть надо найти методом Эйлера с пересчетом частное решение дифференциального уравнения

$$y' = \frac{10}{x^2 + y^2} + x = f(x, y), \tag{VI.41}$$

определенное на промежутке

$$1 \le x \le 10 \tag{VI.42}$$

и удовлетворяющее начальному условию

$$y|_{x=4} = 1. \tag{VI.43}$$

Шаг интегрирования взять

$$h = 0{,}01. \tag{VI.44}$$

Напомним, что численное интегрирование дифференциального уравнения методом Эйлера с пересчетом состоит в последовательных вычислениях по серии формул

$$\left.\begin{aligned} y'_i &= f(x_i, y_i), \\ \bar{y}_{i+1} &= y_i + h y'_i, \\ x_{i+1} &= x_i + h, \\ \bar{y}'_{i+1} &= f(x_{i+1}, \bar{y}_{i+1}), \\ \theta_i &= \frac{y'_i + \bar{y}'_{i+1}}{2}, \\ y_{i+1} &= y_i + h\theta_i, \end{aligned}\right\} \tag{VI.45}$$

где

$i = 0,\ 1, 2, .., n;$ $\qquad n = 9/h = 900$ (десятичное) $= 1604$ (восьмеричное).

П о д б о р м а с ш т а б н ы х к о э ф ф и ц и е н т о в .

Масштабные коэффициенты выбираются так, чтобы все входные величины, а также результаты вычислений были меньше единицы.

Для такого выбора масштабных коэффициентов определим границы изменения переменных величин, входящих в дифференциальное уравнение (VI. 41).

Очевидно, что справедливо неравенство

$$y' = \frac{10}{x^2 + y^2} + x \le \frac{10}{x^2} + x\,.$$

Кроме того, при $1 \le x \le 10$

$$y' = \frac{10}{x^2 + y^2} + x > x \ge 1\,,$$

т. е. $y' > 0$ и, следовательно, $y$ — возрастающая функция.

Величина

$$\frac{10}{x^2} + x$$

в промежутке $1 \le x \le 10$ имеет только один экстремум—минимум, так как ее производная $-\frac{20}{x^3} + 1$ обращается на этом промежутке один раз в нуль и при переходе через этот нуль меняет знак с отрицательного на положительный.

Отсюда заключаем, что $\frac{10}{x^2}+x$ достигает наибольшего значения на одном из концов нашего промежутка. Подсчет показывает, что это наибольшее значение равно 11. Итак,

$$1 \le y' < \frac{10}{x^2}+x \le 11 \tag{VI.46}$$

т. е.

$$|\, y' \,| \le 11$$

Интегрируя неравенство $y' < \frac{10}{x^2}+x$ в пределах от 1 до $x$, получаем после несложных преобразований

$$0 < y \le -\frac{10}{x}+\frac{x^2}{2}+10{,}5 \le 59{,}5$$

Число 59,5 в правой части последнего неравенства получено путем замены $x$ числом 10 в выражении $-\frac{10}{x}+\frac{x^2}{2}+10{,}5$. Это выражение, имея положительную производную, является возрастающим и, следовательно, наибольшее значение на отрезке $1 \le x \le 10$ принимает в точке $x$=10.

Итак, мы теперь видим, что

$$\left.\begin{aligned} 1 \le x \le 10, \\ 1 \le y \le 59{,}5, \\ 1 \le y' \le 11. \end{aligned}\right\} \tag{VI.47}$$

Эти границы установлены довольно грубо, но во всяком случае величины $x$, $y$, $y'$ не выходят за их пределы.

Для того чтобы не осложнять наш пример излишними трудностями, выберем для величин $x$ (а значит, и для $h$) и $y$ общий масштабный коэффициент $m$.

Положим

$$\left.\begin{aligned} x = m\xi, \\ y = m\eta, \\ h = m\bar{h}, \end{aligned}\right\} \tag{VI.48}$$

где $m$ выбрано так, чтобы были справедливы неравенства

$$|\xi| < 1, \quad |\eta| < 1, \quad |\,\bar{h}\,| \le 1.$$

Очевидно, последние неравенства будут обеспечены при

$$m \ge 60. \tag{VI.49}$$

Подставляя выражения (VI. 48) в выражение (VI. 41) и заменяя $y'$ по формуле

$$y' = kz,$$

где $k$—масштабный коэффициент, выбранный так, что $|\, z \,| < 1$, получаем из уравнения (VI. 41)

$$kz = \frac{10}{m^2(\xi^2+\eta^2)}+m\xi \tag{VI.50}$$

или

$$z = \frac{\dfrac{10}{m^2 k}}{\xi^2+\eta^2}+\frac{m}{k}\xi \tag{VI.51}$$

Чтобы при вычислении величины $z$ по формуле (VI. 51) не переполнялась разрядная сетка, масштабные коэффициенты должны удовлетворять следующим условиям.

1. В силу (VI.47) имеем $|y'| = |kz| < 11$. Необходимо, чтобы выполнялось неравенство $|z| < 1$. Это будет обеспечено при условии что

$$k > 11; \tag{VI.52}$$

2. Должно быть $\frac{m}{k} \le 1$. Знак равенства здесь допустим, так как в (VI. 51)$|\xi|<1$. Таким образом, необходимо, чтобы имело место неравенство

$$m \le k. \tag{VI.53}$$

3. Нужно, чтобы выполнялось неравенство

$$\frac{10}{m^2 k} < 1$$

откуда следует, что необходима справедливость неравенства

$$m^2 k > 10. \qquad \text{(VI. 54)}$$

4. Наконец, при всех $\xi$ и $\eta$, а значит, и при наименьших должно быть справедливо неравенство

$$\left| \frac{\frac{10}{m^2 k}}{\xi^2 + \eta^2} \right| < 1$$

т. е. в силу (VI. 48), (VI. 42) и (VI. 43)

$$\frac{10}{k(x^2 + y^3)_{\min}} = \frac{10}{2k} < 1,$$

откуда

$$k > 5. \qquad \text{(VI.55)}$$

Сопоставляя неравенства (VI. 49), (VI. 52), (VI. 53), (VI. 54), (VI. 55), видим, что они все будут выполнены, если

$$m \geq 60; \qquad k \geq m. \qquad \text{(VI.56)}$$

При выполнении условий (VI. 56) вычисления, представленные первой, третьей и четвертой формулами из выражения (VI.45), будут происходить без переполнения разрядной сетки машины. Подставим выражение (VI.48) в остальные формулы из выражения (VI.45). Получим:

$$\left.\begin{aligned} \bar{\eta}_{i+1} &= \eta_i + (k\bar{h}) z_i, \\ \bar{\theta}_i &= \frac{1}{2} z_i + \frac{1}{2} \bar{z}_{i+1}, \\ \eta_{i+1} &= \eta_i + (k\bar{h}) \bar{\theta}_i. \end{aligned}\right\} \qquad \text{(VI.57)}$$

Здесь $\bar{\theta}_i$ - получено по формуле

$$\theta_i = k\bar{\theta}_i,$$

Рассмотрим величину $k\bar{h}$. Являясь произведением числа *k,* которое больше единицы, и $\bar{h}$, эта величина не может быть вычислена на машине. Но

$$k\bar{h} = \frac{km\bar{h}}{m} = \frac{k}{m} h.$$

Видим, что при $k \leq m$ величина $k\bar{h}$ меньше единицы. Если $k\bar{h}$ подсчитать заранее, то вычисления по формулам (VI. 57) будут проходить без переполнения разрядной сетки машины.

Было бы удобно выбрать для масштабных коэффициентов значения, удовлетворяющие (VI. 56) и являющиеся отрицательными степенями числа десять. Это позволило бы по результатам счета машины получать истинные результаты задачи простым переносом запятой. Однако, чрезмерно уменьшая масштабные коэффициенты, мы вводим в расчетные формулы крайне малые величины, которые будут записываться в ячейку машины с большой относительной ошибкой или даже могут оказаться машинными нулями. В нашем случае такой малой величиной будет величина $\frac{10}{m^2 k}$, входящая в формулу (VI.51). Участвуя в вычислениях, эта величина будет переносить свою погрешность в результат вычислений. Поэтому приходится жертвовать удобствами ради повышения точности получаемых результатов и идти на выполнение ручного пересчета результатов, даваемых машиной. Мы положим

$$m = k = 64 = 2^6. \qquad \text{(VI.58)}$$

Выбранные числа *m* и *k* удовлетворяют всем наложенным на них условиям.

Теперь получаем:

$$\begin{aligned} x &= 2^6 \xi; \quad k\bar{h} = h = 0{,}01 \\ y &= 2^6 \eta \qquad \theta_i = 2^6 \bar{\theta}_i. \\ y' &= 2^6 z \end{aligned}$$

Промежуток изменения величины $\xi$ будет:

$$\frac{1}{2^6} \le \xi \le \frac{10}{2^6} = \frac{5}{2^5}. \tag{VI.59}$$

Начальные условия принимают такой вид:

$$\eta_{\xi=2^{-6}} = 2^{-6} \tag{VI.60}$$

Формулы счета, получаемые из (VI. 41) и (VI. 45), будут следующими:

$$\left.\begin{aligned} z_i &= \frac{\left(\frac{5}{2^{17}}\right)}{\xi_i^2 + \eta_i^2} + \xi_i, \qquad \overline{\eta}_{i+1} = \eta_i + 0{,}01 z_i, \\ \xi_{i+1} &= \xi_i + \overline{\eta} = \xi_i + \frac{0{,}01}{2^6}, \quad \overline{z}_{i+1} = \frac{\left(\frac{5}{2^{17}}\right)}{\xi_{i+1}^2 + \overline{\eta}_{i+1}^2} + \xi_{i+1}, \\ \overline{\theta}_i &= 0{,}5 z_i + 0{,}5 \overline{z}_{i+1}, \qquad \eta_{i+1} = \eta_i + 0{,}01 \overline{\theta}_i. \end{aligned}\right\} \tag{VI.61}$$

После получения результатов машинного счета истинные результаты будем получать по формуле

$$y_i = 64_i. \tag{VI.62}$$

Составление программы

Логическая схема для интегрирования дифференциального уравнения (VI. 41) методом Эйлера с пересчетом получена нами в § 35 (при составлении программы для трехадресной машины с плавающей запятой). Эта логическая схема с незначительными изменениями остается в силе и теперь. Воспроизведем ее здесь:

$$\Pi_0 \bar{A}_1 \bar{A}_2 \Pi_3 E_4 (1)\, A_5 \Pi_6 E_7 (2)\, З^{*i}_{s} F_9 (i)\, P_{10} A_{11} \Pi_{12} Я_{13}$$

или

$$\Pi_0 \overset{10}{\overline{\phantom{x}}} \bar{A}_1 \bar{A}_2 \Pi_3 E_4 (1)\, A_5 \Pi_6 E_7 (2)\, З^{*i}_{s} F_9 (i) P_{10} \overset{1}{\overline{\phantom{x}}} A_{11} \Pi_{12} Я_{13}.$$

Особенности машины *Урал,* отличающие эту машину от *Стрелы,* требуют введения некоторых изменений в эту логическую схему.

Во-первых, *Урал* не имеет устройства для хранения стандартных подпрограмм, и соответствующие подпрограммы для перевода чисел из десятичной системы счисления в двоичную и из двоичной системы в десятичную должны быть включены в рабочую программу. Поэтому оператор $\mathbf{A}_{11}$ логической схемы программы для *Стрелы* будет теперь заменен некоторым обобщенным оператором. Впрочем, мы не будем приводить здесь этот обобщенный оператор и сохраним для него обозначение $\mathbf{A}_{11}$.

Во-вторых, система команд *Урала* позволяет управлять выполнениями цикла с помощью двух «окаймляющих» команд, первая из которых имеет код операции 25, а вторая — 24. Поэтому оператор $\mathbf{P}_{10}$ мы заменим двумя операторами (каждый состоит из одной команды): оператором $\mathbf{P}_0$, стоящим перед $\mathbf{A}_1$ и оператором $\overline{P}_{10}$, стоящим вместо $\mathbf{P}_{10}$. Оператор переадресации становится ненужным. Теперь логическая схема программы станет следующей:

$$\Pi_{-1} P_0 \bar{A}_1 \bar{A}_2 \Pi_3 E_4 (1)\, A_5 \Pi_6 E_7 (2)\, З^{*i}_{s} P_{10} A_{11} \Pi_{12} Я_{13}$$

или

$$\Pi_{-1} \overset{10}{\overline{\phantom{x}}} P_0 \bar{A}_1 \bar{A}_2 \Pi_3 E_4 (1)\, A_5 \Pi_6 E_7 (2)\, З^{*i}_{s} \bar{P}_{10} \overset{r0}{\overline{\phantom{x}}} A_{11} \Pi_{12} Я_{13}.$$

Смысл остальных операторов остается прежним.

Исходные данные разместим следующим образом*:

$$b-12)\bar{h} = \frac{0{,}01}{2^6}, \quad b-10)\frac{5}{2^{17}}, \quad b-6)0{,}01, \quad b-4)0{,}5, \quad b-2)\xi_0, \quad b)\eta_0, \quad b'+1)22a+41, \quad b'+2)22a+53;$$

* Напоминаем, что длинные ячейки машины *Урал* имеют четные номера большие, чем 4000 (восьмеричное)

где $b$ — четное, а $b'+1$ и $b'+2$ — номера коротких ячеек. Для результатов отведем ячейки $b+2$, $b+4$, ... (с четными номерами).

Операторы $\overline{A}_1$ и $\overline{A}_2$ имеют стандартные входные и выходные ячейки. Операторы переноса перемещают исходные данные для работы каждого из этих операторов в их входные ячейки, а результаты счета операторов переносят из них для хранения.

Так как система команд машины *Урал* не имеет команды условного перехода второго типа, то команды возврата приходится заносить на соответствующие места, что и выполняют операторы обращения. Потом эти команды приходится стирать, что у нас производит оператор $\mathbf{З}^*_8{}^i$.

Приводим программу:

| | | | |
|---|---|---|---|
| $\mathbf{П}_{-1}$ | $a$+1) | 02 | $b$–2; |
| | $a$+2) | 16 | $c$+2; |
| | $a$+3) | 02 | $b$ ; |
| | $a$+4) | 16 | $c$+4; |
| $\overline{\mathbf{P}}_0$ | $a$+5) | 25 | 2$n$+3776. |
| $\overline{\mathbf{A}}_1$ | $a$+6) | 02 | $c$+2 |
| | $a$+7) | 06 | $c$+2; |
| | $a$+10) | 17 | $c$+4; |
| | $a$+11) | 05 | $c$+4; |
| | $a$+12) | 16 | $c$+6; |
| | $a$+13) | 02 | $b$–10; |
| | $a$+14) | 07 | $c$+6; |
| | $a$+15) | 01 | $c$+2 |
| | $a$+16) | 16 | $c$+6; |
| | $a$+17) | 00 | 0000; |
| $\overline{\mathbf{A}}_2$ | $a$+20) | 02 | $c$+4; |
| | $a$+21) | 17 | $b$–6; |
| | $a$+22) | 05 | $c$+6; |
| | $a$+23) | 16 | $c$+10; |
| | $a$+24) | 00 | 0000; |
| $\mathbf{П}_3$ | $a$+25) | 02 | $c$+2; |
| | $a$+26) | 01 | $b$–12; |
| | $a$+27) | 16 | $c$+2; |
| | $a$+30) | 02 | $c$+4; |
| | $a$+31) | 16 | $c$+12; |
| | $a$+32) | 02 | $c$+6; |
| | $a$+33) | 16 | $c$+14; |
| | $a$+34) | 02 | $c$+10; |
| | $a$+35) | 16 | $c$+4; |
| $\mathbf{E}_4(1)$ | $a$+36) | 02 | $b'$+1; |
| | $a$+37) | 16 | $a$ +17; |
| | $a$+40) | 22 | $a$ +6; |
| $\mathbf{A}_5$ | $a$+41) | 02 | $b$ – 4; |
| | $a$+42) | 06 | $c$ +6; |
| | $a$+43) | 17 | $b$ – 4; |
| | $a$+44) | 05 | $c$ +14; |
| | $a$+45) | 16 | $c$ +6; |
| $\mathbf{П}_6$ | $a$+46) | 02 | $c$ +12; |
| | $a$+47) | 16 | $c$ +4; |
| $\mathbf{E}_7(2)$ | $a$+50) | 02 | $b'$+2; |
| | $a$+51) | 16 | $a$ +24; |
| | $a$+52) | 22 | $a$ +20; |
| $\mathbf{З}^*_8{}^i$ | $a$+53) | 02 | 0000 |
| | $a$+54) | 16 | $a$ +17; |
| | $a$+55) | 16 | $a$ +24; |
| | $a$+56) | 02 | $c$ +10; |
| | $a$+57) | 16 | $c$ +4; |
| | $a$+60) | –16 | $b$ + 2$n$; |
| $\overline{\mathbf{P}}_{10}$ | $a$+61) | 24 | $a$ +6; |

(Операторы $\mathbf{А}_{11}$, $\mathbf{П}_{12}$, $\mathbf{Я}_{13}$ не приводим). Здесь рабочие ячейки у нас обозначены через $c$+2, $c$+4, ... Они являются длинными, поэтому $c$ — четное число большее, чем 3776, для того чтобы $c+2$ было больше, чем 4000. Адрес команды ($a$ + 5) равен 4000 + ($n$ — 2) (стр. 151).

**2. Программирование для двухадресной машины.** Программы для двухадресных машин по своей длине мало превосходят программы, составленные для трехадресных машин. Программирование для двухадресной машины мы поясним на примере составления программы для машины *М-3*. Эта машина, как читателю известно, имеет фиксированную запятую, и, следовательно, при подготовке задач для решения на *М-3* необходимо производить подбор масштабных коэффициентов. Составим программу для решения задачи, сформулированной в предыдущем пункте этого параграфа. При этом воспользуемся расчетными формулами (VI. 61). Схема программы ничем не будет отличаться от логической схемы, составленной нами ранее (см. § 35) для машины *Стрела:*

$$\overbrace{\mathbf{П}_0\overline{\mathbf{A}}_1\overline{\mathbf{A}}_2\mathbf{П}_3\mathbf{E}_4(1)\,\mathbf{A}_5\mathbf{П}_6\mathbf{E}_7(2)\,\mathbf{З}^*{}^i_6\mathbf{F}_9(i)}\mathbf{P}_{10}\mathbf{A}_{11}\mathbf{П}_{12}\mathbf{Я}_{13}.$$

Смысл операторов остается прежним. Исходные данные разместим следующим образом:

$$b-5)\,\overline{h}=\frac{0{,}01}{2^6},\quad b-4)\,\frac{5}{2^{17}},\quad b-3)0{,}01,\quad b-2)0{,}6,\quad b\text{ — }1)\xi_0,\quad b)\eta_0;$$

$e$ + 1) 74 0000 $a$ + 23, $e$ +2) 74 0000 $a$+ 31, $e$ + 3) 05 0000 0000, $e$ + 4) 00 00000001, $e$ + 5) 05 $c$ + 4 $b$+ $n$.

Для результатов отведем ячейки $b + 1$, $b + 2$, ...

Операторы $\overline{A}_1$ и $\overline{A}_2$ имеют стандартные входные и выходные ячейки. Операторы переноса перемещают исходные данные для работы этих операторов в их входные ячейки, а результаты счета переносят из выходных ячеек для хранения. Так как система команд машины *М-3* не имеет команды условного перехода второго типа, то команды возврата приходится записывать в соответствующие ячейки, что и выполняют операторы обращения. Потом эти команды приходится стирать, что выполняет оператор $\mathbf{З}^*_8{}^i$.

Программа:

| | | | | |
|---|---|---|---|---|
| $\mathbf{П}_0$ | $a$+1) | 05 | $b$–1 | $c$+1 |
| | $a$+2) | 05 | $b$ | $c$+1 |
| $\overline{\mathbf{A}}_1$ | $a$+3) | 05 | $c$+1 | $c$+3 |
| | $a$+4) | 03 | $c$+1 | $c$+3 |
| | $a$+5) | 13 | $c$+2 | $c$+2 |
| | $a$+6) | 20 | $c$+3 | $c$+3 |
| | $a$+7) | 02 | $b$–4 | $c$+3 |
| | $a$+10) | 00 | $c$+1 | $c$+3 |
| | $a$+11) | 05 | 0000 | 0000 |
| $\overline{\mathbf{A}}_2$ | $a$+12) | 13 | $b$–1 | $c$+3 |
| | $a$+13) | 20 | $c$+2 | $c$+4 |
| | $a$+14) | 05 | 0000 | 0000 |
| $\mathbf{П}_3$ | $a$+15) | 05 | $c$+2 | $c$+5 |
| | $a$+16) | 05 | $c$+3 | $c$+6 |
| | $a$+17) | 05 | $c$+4 | $c$+2 |
| | $a$+20) | 00 | $b$–5 | $c$+1 |
| $\mathbf{E}_4(1)$ | $a$+21) | 05 | $e$+1 | $a$+11 |
| | $a$+22) | 74 | 0000 | $a$+11 |
| $\mathbf{A}_5$ | $a$+23) | 00 | $b$–2 | $c$+3 |
| | $a$+24) | 00 | $b$–2 | $c$+6 |
| | $a$+25) | 00 | $c$+3 | $c$+3 |
| $\mathbf{П}_6$ | $a$+26) | 00 | $c$+3 | $c$+3 |
| $\mathbf{E}_7(2)$ | $a$+27) | 05 | $e$+2 | $a$+14 |
| | $a$+30) | 74 | 0000 | $a$+12 |
| $\mathbf{З}^*_8{}^i$ | $a$+31) | 05 | $e$+3 | $a$+11 |
| | $a$+32) | 05 | $e$+3 | $a$+14 |
| | $a$+33) | 05 | $c$+4 | $c$+2 |
| | $a$+34) | 05 | $c$+4 | $b$+1* |
| $\mathbf{F}_9(i)$ | $a$+35) | 00 | $e$+4 | $a$+34 |
| $\mathbf{Р}_{10}$ | $a$+36) | 05 | $a$+34 | $c$+3 |
| | $a$+37) | 01 | $e$+5 | $c$+3 |
| | $a$+40) | 34 | $a$+41 | $a$+3 |

В ячейке $a + 41$ стоит первая команда оператора $\mathbf{A}_{11}$. Операторы $\mathbf{A}_{11}$, $\mathbf{П}_{12}$ и $\mathbf{Я}_{13}$ не приводим.

Одна и та же задача запрограммирована нами трижды: для трехадресной машины *Стрела* (§ 35), для двухадресной машины *М-3* и для одноадресной машины *Урал.* При этом программа для машины *Стрела* состоит из двадцати шести команд, программа для *М-3* — из тридцати двух команд (всего на шесть команд, т. е. на 23%, больше) и программа для *Урала* состоит из сорока девяти команд (на 23 команды, т. е. на 90%, больше, чем для *Стрелы,* и на 17 команд, т. е. на 53%, больше, чем для *М-3*).

Объясним некоторые особенности программы для машины *М-3.*

1. Последние команды операторов $\overline{A}_1$ и $\overline{A}_2$ взяты не «нулевыми» (не 00 0000 0000), так как такие команды означали бы: «Содержимое нулевой ячейки сложить с содержимым нулевой ячейки (т. е. удвоить) и сумму записать в нулевую ячейку». В машине *М-3* нулевая ячейка ничем не отличается от остальных, следовательно, при наличии какого-либо числа в нулевой ячейке в результате многократного выполнения «нулевых» команд произошло бы переполнение разрядной сетки. Последние команды операторов $\overline{A}_1$ и $\overline{A}_2$ в нашей программе имеют вид 05 0000 0000 и не могут привести к переполнению разрядной сетки.

2. Оператор $\mathbf{Р}_{10}$ определяет конец счета путем вычитания от запасенной команды $(e + 5) = 05\ c + 4\ b + n$ команды $(a + 34)$,стоящей в программе. После вычисления $\eta_n$ получается:

$$(a + 34) = 05 \quad c+4 \quad b+n+1,$$

так что

$$(e + 5) - (a + 34) < 0.$$

При этом вырабатывается сигнал $\omega = 1$ и команда $(a + 40)$ оператора $\mathbf{Р}_{10}$ передает управление уже не оператору $\overline{A}_1$, а оператору $\mathbf{A}_{11}$. Операцией сравнения в операторе $\mathbf{Р}_{10}$ мы воспользоваться не могли, так как такой операции нет в системе команд машины *М-3.*

## ГЛАВА VII
## МЕТОДЫ РУЧНОГО ПРОГРАММИРОВАНИЯ

### § 37. Порядок работы при ручном программировании

**1. Параметрически заданная схема счета.** Работа программиста начинается с того момента, когда подлежащая машинному решению задача уже арифметизирована.

Эта работа весьма облегчается, если для выбранного алгорифма решения задачи составлено описание, позволяющее быстро и точно уяснить все его особенности и удобное для дальнейшей работы по подготовке задачи к

машинному решению. Наиболее очевидной необходимость такого описания становится, если выбор численного метода для решения задачи производится одним специалистом, а программирование — другим.

Как показывает практика, весьма удобна и применима к широкому классу численных методов форма записи, получившая название *параметрически заданной схемы счета.* При составлении параметрически заданной схемы счета каждая переменная величина задачи рассматривается как последовательность ее значений. Члены такой последовательности обозначаются той же буквой, что и сама переменная величина, и снабжаются индексом — номером данного значения переменной величины в последовательности ее значений.

Для того чтобы указать связь между номерами значений различных переменных величин, необходимых для вычислений по формуле, вводят целочисленные переменные, называемые *параметрами схемы счета,* и в формулах в качестве индексов значений переменных величин записывают целочисленные функции этих параметров,

Например, вместо формул

$$r_1 = x_1 + y_1^2,$$
$$r_2 = x_2 + y_2^2,$$
$$\dots$$
$$r_{200} = x_{200} + y_{200}^2,$$
$$\rho_1 = x_1 + y_2 + z_1,$$
$$\rho_2 = x_2 + y_4 + z_3$$
$$\dots$$
$$\rho_{100} = x_{100} + y_{200} + z_{199}$$

пишут формулы

$$r_i = x_i + y_i^2$$
$$\rho_j = x_j + y_{2j} + z_{2j-1}$$

При этом указывают, для каких значений параметров должны быть произведены вычисления по формулам. В приведенном примере вычисления по первой формуле должны быть произведены для всех значений $1 \le i \le 200$, а во второй формуле — для всех значений $1 \le j \le 100$.

При решении задачи численным методом нередко приходится, перед тем как производить вычисления, решать вопрос о том, по какой из нескольких формул (групп формул) должно быть определено значение некоторой величины (группы величин), т. е. решение задачи численным методом состоит из ряда логических и вычислительных актов. Эти акты вычислительного процесса должны быть отражены в параметрически заданной схеме счета.

Схема счета составляется следующим образом:

1) перечисляют исходные данные для решения задачи;

2) выписывают группы формул, каждая из которых представляет отдельный вычислительный акт алгорифма;

3) для каждой из таких групп указывают условия, при выполнении которых должны быть произведены вычисления по формулам данной группы;

4) для каждой группы формул указывают, при каких значениях параметров должны быть произведены вычисления;

5) перечисляют искомые результаты решения.

Параметрически заданная схема счета является универсальным описанием численного метода, не зависящим от свойств исполнителя алгорифма. Ею может руководствоваться, в частности, и человек-вычислитель. Для этого он выбирает группу формул такую, что а) значения величин, фигурирующие в условиях, разрешающих вычисления, известны (либо содержатся в исходных данных, либо уже вычислены);

6) условия, разрешающие вычисления, выполнены;

в) все значения величин, исходных для вычисления по формулам, известны (имеются в исходных данных задачи или уже вычислены) или могут быть вычислены с помощью формул этой группы.

Затем вычислитель производит вычисления по формулам этой группы. Формулу за формулой он выбирает в такой последовательности, чтобы для каждой выбранной формулы все необходимые значения величин были уже известны.

Вычисления по формулам производятся для всех значений параметров, указанных в схеме счета для данной группы формул.

После того как все вычисления, предусмотренные в параметрически заданной схеме счета, произведены, задача решена. Остается выписать значения тех величин, которые перечислены в схеме счета в качестве искомых результатов.

Параметрически заданная схема счета, вообще говоря, не дает однозначного описания алгорифма для осуществления численного решения задачи. Она описывает некоторый класс таких алгорифмов. Действительно, во многих случаях в группе формул, допускающей вычисления, может оказаться несколько формул, одновременно допускающих вычисления. Выбирая одну из таких формул, мы выбираем один алгорифм (или один подкласс алгорифмов), описанный параметрически заданной схемой счета. Точно также в ряде случаев последовательность значений параметров, для которых должны быть произведены вычисления, допускает произвол в выборе. Наконец, возможно, что сразу несколько групп формул являются допустимыми для вычислений и возможен произвол в выборе одной из этих групп.

Параметрически заданная схема счета не содержит никаких указаний о способах и месте хранения исходных

данных задачи, а также промежуточных и окончательных результатов.

Необходимо отметить, что большая часть ошибок, допускаемых математиком при составлении параметрически заданной схемы счета, носит локальный характер и допускает локальное исправление при их обнаружении.

Итак, параметрически заданная схема счета обладает тремя особенностями:

а) неоднозначностью описания алгорифма численного метода;

б) произвольностью размещения в запоминающих устройствах исходных данных и результатов решения задачи;

в) возможностью локального исправления подавляющего большинства ошибок.

Эти особенности делают параметрически заданную схему счета весьма удобной для целей программирования способом описания численного метода.

Параметрически заданная схема счета как форма записи численного метода решения задачи была сформулирована в окончательном виде Э. З. Любимским и С. С. Камыниным.

**2. Составление логической схемы.** Программирование состоит из двух основных этапов: составления логической схемы программы (коротко — схемы) и составления программы.

Логическая схема программы составляется путем переработки параметрически заданной схемы счета. Группы математических формул, представляющие собой отдельные вычислительные акты алгорифма, подвергаются анализу, и формулы в них располагаются в определенном порядке, в котором будут производиться вычисления. В дальнейшем эти группы математических формул соответствуют арифметическим операторам. Условия, разрешающие счет по формулам арифметических операторов, соответствуют логическим операторам. Попутно приходится частично решать вопрос о размещении исходных данных и результатов в запоминающих устройствах машины. Намечаются группы операций, при выполнении которых машиной будет производиться изменение значений параметров, от которых зависят формулы действующих операторов. Эти группы операций соответствуют операторам переадресации и восстановления. Для обеспечения необходимой последовательности изменений параметров в алгорифм включаются новые группы условий, т. е. новые логические операторы. Наконец, в ряде случаев в перерабатываемый алгорифм приходится включать группы операций, необходимых для обеспечения возможности выполнения операторов переадресации, восстановления или логических операторов, большей частью предусматривающих переносы чисел или команд будущей программы из одних ячеек памяти в другие, и т. п.

Таким образом, переход от параметрически заданной схемы счета к логической схеме программы состоит, во-первых, в расширении решающего алгорифма и, во-вторых, в его конкретизации (т. е. в выборе одного-единственного решающего алгорифма из класса алгорифмов, представленного схемой счета).

Приведем пример, поясняющий параметрически заданную схему счета и переход от нее к логической схеме программы.

Предположим, что дано дифференциальное уравнение

$$\frac{dy}{dx} = f(x, y, z) \tag{VII.1}$$

с начальными условиями

$$x_0 = 0; \qquad y\,|_{x=x_0} = y_0 \tag{VII.2}$$

причем $z$ представляет собой функцию от независимого переменного $x$, которая при $0 \le x < 1$ определена как сумма ряда

$$z = \sum_{j=1}^{\infty} \frac{x^j}{y^2}, \tag{VII.3}$$

а для $1 \le x < X$— как корень уравнения

$$z = \psi(x,z). \tag{VII.4}$$

Требуется решить приведенное дифференциальное уравнение и составить таблицу значений функции $y$, соответствующих значениям $x_0$, $x_1 = x_0 + h$, …, $x_i = x_0 + ih$, …, $x_n = x_0 + nh = X$ независимого переменного.

Предположим, что, анализируя задачу, математик пришел к следующим выводам.

1. Дифференциальное уравнение (VII.1) можно решать методом Эйлера с шагом интегрирования, равным $h$.
2. При определении с необходимой точностью суммы ряда (VI1.3) достаточно удерживать первые 100 членов этого ряда.
3. Вычисление корней уравнения (VII.4) можно осуществлять методом итераций, принимая за начальное значение $\xi_{i,0}$ очередного искомого корня $z_i = z(x_i)$ ранее полученный корень $z_{i-1}$. При этом получаемые приближенные значения корней значительно меньше числа 100. За искомый корень $z_i$ следует принимать приближенное значение этого корня $\xi_{i+k}$, если выполнено условие $|\xi_{i,k} - \xi_{i,k-1}| < \varepsilon$. Число $\varepsilon$ подсчитано.
4. Ошибка в определении $x_n = x_0 + nh = X$ не может превзойти величины $\frac{h}{1000}$. Поэтому можно считать, что очередное значение аргумента $x_i$ равно $X$, если $|X - x_i| < \frac{h}{1000}$. При выполнении указанного условия вычисление произвести последний раз.

Параметрически заданную схему счета, описывающую выбранный метод решения задачи, можно представить в виде таблицы 43.

Т а б л и ц а 43. Параметрически заданная схема счета

| Исходные данные | $x_0, y_0, \zeta_{i,-1}$ ($\zeta_{i,-1}$ при любом $i$ равно 100) | | |
|---|---|---|---|
| | Группа расчетных формул | Условия, разрешающие вычисления | Значения параметров, при которых следует вести вычисления |
| *а* | $u_{i,j} = \frac{x_i^j}{j^2}$ | $X + \frac{h}{1000} \geq x_i, \quad x_i < 1$ | $i$ – любое<br>$j$ – 1, 2, ..., 100 |
| *б* | $z_i = \sum_{j=1}^{100} u_{i,j}$ | $X + \frac{h}{1000} \geq x_i, \quad x_i < 1$ | $i$ – любое |
| *в* | $\zeta_{i,0} = z_{i-1}, \quad \zeta_{j+1} = \psi(X_i, \zeta_{i,j})$ | $X + \frac{h}{1000} \geq x_i, \quad x_i \geq 1; \quad \lvert \zeta_{i,j} - \zeta_{i,j-1} \rvert \geq \varepsilon;$ | $i, j$ – любое |
| *г* | Полагать $z_i = \zeta_{i,j}$ | $X + \frac{h}{1000} \geq x_i, \quad x_i \geq 1; \quad \lvert \zeta_{i,j} - \zeta_{i,j-1} \rvert < \varepsilon;$ | $i, j$ – любое |
| *д* | $y_{i+1} = y_i + hf(x_i, y_i, z_i), \quad x_{i+1} = x_i + h$ | $X + \frac{h}{1000} \geq x_i$ | $i$ – любое |
| *е* | Полагать $Y = y_{i+1}$ | $X + \frac{h}{1000} < x_i$ | $i$ – любое |
| Результаты вычислений | $y_0, y_1, y_3, \ldots, y_i, \ldots, y_n = Y$ | | |

Имея приведенную параметрически заданную схему счета, легко можно составить логическую схему программы.

Действительно, анализируя условия, разрешающие вычисления по расчетным формулам, видим, что все вычисления можно проводить только при выполнении условия

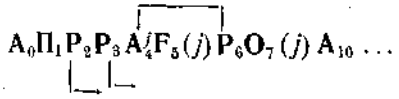
$$X + \frac{h}{1000} \geq x_i. \tag{VII.5}$$

Поэтому в начале логической схемы расположим оператор $\mathbf{P}_2$, проверяющий это условие (до этого оператора будут стоять $\mathbf{П}_0\mathbf{A}_1$ — операторы ввода и перевода исходных данных в двоичную систему счисления). Затем, если условие (VII.5) выполнено, то управление получит оператор $\mathbf{P}_3$, проверяющий условие

$$x_1 < 1, \tag{VII.6}$$

разрешающее вычисление величины $z_i$ в виде суммы ряда. При невыполнении условия (VII.5) перейдем к переводу результатов в десятичную систему счисления и выдаче их на перфокарты.

При выполнении условия (VII.6) переходим к оператору $A_4^j$, вычисляющему $u_{i,j}$, для переадресации которого служит $\mathbf{F}_5(j)$, а для осуществления повторений счета оператором $A_4^j$ при $j$=1, 2, ...., 100 служит оператор $\mathbf{P}_6$. Операторы $\longrightarrow A_4^j \mathbf{F}_5(j) \mathbf{P}_6 \longrightarrow$ образуют цикл, который после вычисления всех членов ряда должен быть восстановлен к первоначальному виду. Для этого после цикла помещаем оператор восстановления $\mathbf{O}_7(j)$. Затем располагаем оператор $\mathbf{A}_{10}$, вычисляющий $z_i = \sum_{j=1}^{100} u_{i,j}$. Таким образом, нами уже составлена часть логической схемы:




(VII.7)

После оператора $\mathbf{A}_{10}$ мы должны переходить к оператору, выполняющему группу формул *д,* ибо никаких других вычислений до получения нового значения независимого переменного производить больше нельзя.

Если условие (VII. 6) не выполнено, то с помощью оператора $\mathbf{P}_{11}$ предусматриваем проверку условия

$$|\xi_{i,j} - \xi_{i,j-1}| < \varepsilon \tag{VII.8}$$

При выполнении условия (VI1.8) переходим к оператору, ведущему счет по расчетным формулам *д,* а при невыполнении — к вычислению значения $z_t$ как корня уравнения (VI1.4). Это делает оператор $\mathbf{A}_{12}$ (формула *в*). Получаемый результат в соответствующие ячейки переносит оператор $\mathbf{П}_{13}$, передающий управление опять оператору $\mathbf{P}_{11}$.

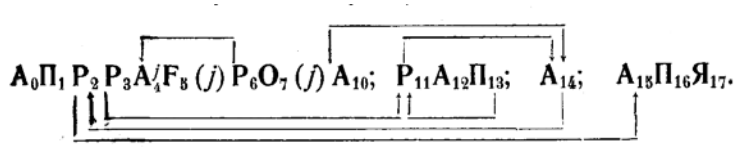
Нами составлена новая часть схемы:


(VII.9)

После того как $z_i$ определено, оператор $\mathbf{P}_{11}$ (а в ранее приведенной части схемы оператор $\mathbf{A}_{10}$) передает управление

оператору $\mathbf{A}_{14}$, который производит вычисления по группе формул *д* и передает управление оператору $\mathbf{P}_2$.

Окончательно получаем следующую логическую схему:




В этой схеме операторы $\mathbf{A}_{15}$ $\mathbf{A}_{16}$ $\mathbf{Я}_{17}$ производят перевод результатов в десятичную систему, выдачу их на перфокарты и останов машины.

Таким образом, составленная после выбора численного метода решения параметрически заданная схема счета облегчает составление логической схемы программы.

Однако в некоторых случаях описание вычислительного процесса в виде параметрически заданной схемы счета или в виде операторной схемы счета, о которой говорилось в § 32, не нужно и можно непосредственно составлять логическую схему программы.

**3. Распределение памяти и составление программы.** Объединяя операции, которые должна реализовать машина, в операторы, мы должны помнить о том, что оператор — это группа команд машины. При этом необходимо соблюдать условия, приведенные в § 32, которым должен удовлетворять оператор.

При составлении логической схемы программы нужно стараться, чтобы количество безусловных передач управления, изображаемых стрелками, идущими от нелогических операторов, было минимальным.

Попутно с составлением логической схемы программы производят ориентировочное распределение памяти машины. Окончательное распределение памяти, как читатель уже знает (см. § 31), возможно только после составления программы (в буквенно-числовых обозначениях). В ряде случаев даже ориентировочное распределение памяти — размещение исходных данных, отведение мест для результатов счета, грубая прикидка числа команд программы — позволяет заметить обстоятельства, влияющие на вид логической схемы программы. Например, может обнаружиться, что в памяти машины не хватает места для всех результатов счета. При этом, составляя логическую схему, необходимо предусмотреть получение результатов «по частям», в виде групп чисел и вывод каждой такой группы на перфокарты или на магнитную ленту для того, чтобы освободить место в памяти машины для новой группы результатов.

После составления логическая схема программы должна быть проверена, так как может оказаться, что в ней пропущены какие-нибудь засылки, восстановления или другие операторы. После исправления ошибок логической схемы программы приступают к программированию.

Операторы, зависящие от параметров, должны быть помечены соответствующими значками сверху. При программировании удобно адреса команд, подлежащие переадресации, помечать звездочками. Справа от программы можно провести несколько вертикальных граф (по числу параметров) и показывать в них зависимость адресов команд от параметров. Все это облегчает построение операторов переадресации.

Одновременно с составлением программы удобно заполнять таблицу, в которой показывается содержимое рабочих ячеек после выполнения (хотя бы при первой работе оператора) каждой команды арифметической операции. Это уменьшает опасность допущения ошибок.

Распределение памяти и составление программы производятся сперва в буквенно-числовых обозначениях.

Если в процессе дальнейшего программирования обнаруживается, что были пропущены нужные команды, то нет надобности переделывать все команды программы. Недостающие команды можно добавить. Например, если между командой $(a + 20)$ и командой $(a + 21)$ нужно вставить пять команд, то придаем им номера $a' + 20$, $a' + 21$, $a' + 22$, $a' + 23$, $a' + 24$ и записываем прямо тут же на свободной части бланка.

После того как буквенная программа составлена, буквам присваивают конкретные числовые значения. При этом нетрудно учесть и все вставки.

При составлении программы нужно всегда помнить о возможности допущения ошибок. Поэтому нужно заранее предусматривать возможности их устранения. В частности, размещая в памяти машины программу, исходные данные, рабочие ячейки, отводя ячейки для результатов счета, не следует разбрасываться и разбивать на мелкие группы ячейки, оставшиеся незанятыми. Если в готовой программе будут обнаружены ошибки — пропуск группы команд или ошибка, для устранения которой нужно группу команд заменить большей группой, то наличие достаточно большого массива свободных ячеек позволяет внести исправления, не прибегая к переделке всей программы. Например, если пропущена группа команд, то программу можно исправить следующим образом.

Та команда, после которой должны быть расположены пропущенные команды, переписывается в свободном месте памяти. Вслед за ней 'записываются все пропущенные команды, а после последней из них ставится нулевая команда. На месте команды, изъятой из программы, пишется команда обращения (с кодом 27) к вынесенной на свободное место памяти группе команд. После такой переделки программы может потребоваться изменение некоторых команд в других частях программы, например команд переадресации.

Вместо команды с кодом 27 может быть использована команда условного перехода первого типа (с кодом 20). В этом случае в конце группы добавленных команд ставят не «нулевую» команду, а команду безусловного перехода, передающую управление соответствующей команде основной программы.

Само собой разумеется, что изменения программы, вызванные ее исправлениями, должны быть отражены в логической схеме программы.

К готовой для использования программе должна быть приложена ее подробная логическая схема с описанием каждого оператора и указанием номеров ячеек, занятых командами каждого оператора, а также подробное распределение памяти.

Методика программирования значительно усложняется, если исходные данные или результаты, а тем более, если и программа не могут быть размещены в памяти машины. В этом случае приходится прибегать к использованию магнитной ленты. Вопрос размещения материала, относящегося к решению задачи, в запоминающих устройствах машины и организации обмена числами между различными запоминающими устройствами может представлять значительные трудности.

## § 38. Метод библиотечных подпрограмм

**1. Библиотечные подпрограммы.** Почти одновременно с возникновением программирования зародилась и идея накапливать его результаты с целью использования их при составлении новых программ. Вычислительный центр, обслуживающий определенный круг заказчиков, встречается чаще всего с определенным типом задач. В программах решения таких задач участвуют одинаковые по своему содержанию части программ. Эти части программ могут быть заготовлены раз и навсегда в качестве библиотечных подпрограмм.

При составлении новых программ библиотечные подпрограммы включаются в их состав как нечто целое.

Итак, *библиотечными подпрограммами* называют собранные в коллекцию (библиотеку) программы отдельных участков вычислительного процесса, используемые как целые части при составлении программ для решения конкретных задач.

В зависимости от предусмотренного при их составлении способа включения в основную программу библиотечные подпрограммы делятся на два типа:

а) *замкнутые* библиотечные подпрограммы,

б) *открытые* библиотечные подпрограммы.

*Замкнутые* библиотечные подпрограммы составлены в окончательном конкретном виде и должны быть введены в определенное место памяти. Включение их в основную программу состоит в том, что в основной программе ставится несколько команд для переноса исходных для подпрограмм данных во входные ячейки подпрограммы и, кроме того, ставится команда, передающая управление первой из команд подпрограммы. В конце подпрограммы ставится команда возврата к основной программе. Результаты работы подпрограммы получаются в выходных ячейках подпрограммы. В машинах, имеющих команды условного или безусловного перехода с возвратом (вроде команды 27 *Стрелы),* библиотечная подпрограмма оканчивается «холостой» командой, занимающей ячейку, предназначенную для автоматической записи в нее команды возврата к основной программе. В этом случае подпрограмма не содержит ни одной заранее неизвестной команды.

Итак, характерно, что замкнутая библиотечная подпрограмма:

1) вводится в строго определенные ячейки памяти;

2) составляется в действительных номерах ячеек, и команды ее содержат действительные адреса (кроме последней команды, возвращающей управление основной программе; в этой команде может быть неизвестным заранее адрес той команды основной программы, которой должно быть передано управление);

3) имеет входные ячейки, в которые должны быть перед началом ее работы записаны исходные данные;

4) выдает результаты в определенные выходные ячейки.

Таким образом, замкнутая подпрограмма является обособленной. Достоинством замкнутых подпрограмм является то, что они включаются в основные программы без всяких изменений. Их можно хранить в виде заранее заготовленной колоды перфокарт.

Их недостаток состоит в том, что не всякие две подпрограммы библиотеки могут быть совместно включены в одну и ту же рабочую программу, так как ограниченность памяти машины не дает возможности для различных подпрограмм, имеющихся в библиотеке, отвести непересекающиеся участки памяти.

*Открытые* подпрограммы включаются в основную программу в виде ее целых кусков.

Такие подпрограммы не могут быть составлены с действительными номерами ячеек и адресами команд, так как заранее не известно, в какие ячейки памяти они будут введены, из каких ячеек они получат исходные данные для своей работы и какие ячейки используют в качестве рабочих и выходных.

Открытая подпрограмма хранится обычно в виде буквенной записи на бланках. Для включения в основную программу она переписывается программистом на бланки основной программы с заменой букв действительными номерами ячеек и адресами. Тем не менее, выгода от применения такой подпрограммы очевидна. Ее переписку может производить малоквалифицированный программист.

Буквенная запись открытых библиотечных подпрограмм позволяет составлять их в общем виде, например, такая подпрограмма может быть составлена для решения системы $n$ дифференциальных уравнений (а не для решения системы определенного числа уравнений).

Достоинство открытых библиотечных подпрограмм состоит в возможности составления их в общем виде, а также в том, что любые две или несколько открытых подпрограмм могут быть совместно включены в общую основную программу. Недостаток их состоит в необходимости затраты труда на переписку на бланк основной программы и на присвоение буквенным номерам ячеек и адресам команд конкретных числовых значений.

Библиотека стандартных подпрограмм должна иметь каталог, в котором указаны номера всех подпрограмм, тип (замкнутая или открытая) каждой из них, приведены формулы, для которых составлена каждая подпрограмма. Для замкнутой подпрограммы указываются входные и выходные ячейки и номера ячеек, в которые она должна быть введена. Для открытой подпрограммы указывается, какие буквы обозначают номера ячеек, хранящих команды, какие означают номера входных, рабочих и выходных ячеек и т. п.

**2. Пример открытой библиотечной подпрограммы.** В качестве примера открытой библиотечной подпрограммы приведем библиотечную подпрограмму, составленную опытным программистом А. М. Бухтияровым для решения системы $n$ дифференциальных уравнений первого порядка методом Рунге—Кутта. Подпрограмма составлена в общем виде.

Пусть дана система $n$ обыкновенных дифференциальных уравнений первого порядка

$$y_i' = f_i(x, y_1, y_2, \ldots, y_n) \qquad (i = 1, 2, \ldots, n)$$

с начальными условиями

$$x = x_0 \quad y_i = y_{i,0} \qquad (i = 1, 2, \ldots, n).$$

Пусть

$$x_{k+1} = x_k + h \qquad (k = 0, 1, 2, \ldots),$$

причем $h = \text{const}$ — шаг интегрирования. Значение функции $y_i$ в точке $x = x_k$ будем обозначать символом $y_{iik}$.

Расчетные формулы для вычисления методом Рунге—Кутта значений $y_i$ при $x = x_{k+1}$ можно представить в

следующем виде:

$$y_{i,k+1} = y_{i,k} + \sum_{j=1}^{4} K_{i,j}\alpha_j ,$$

где

$$K_{i,1} = f_i(x_k, y_{1k}, \ldots, y_{nk})\Delta x_1,$$
$$K_{i,2} = f_i(x_k+\Delta x_1,\ y_{1,k}+K_{1,1},\ y_{2,k}+K_{2,1},\ \ldots,\ y_{n,k}+K_{n,1})\ \Delta x_2,$$
$$K_{i,3} = f_i(x_k+\Delta x_2,\ y_{1,k}+K_{1,2},\ y_{2,k}+K_{2,2},\ \ldots,\ y_{n,k}+K_{n,2})\ \Delta x_3,$$
$$K_{i,4} = f_i(x_k+\Delta x_3,\ y_{1,k}+K_{1,3},\ y_{2,k}+K_{2,3},\ \ldots,\ y_{n,k}+K_{n,3})\ \Delta x_4,$$

Причем

$$\alpha_1 = \frac{1}{3}, \quad \alpha_2 = \frac{2}{3}, \quad \alpha_3 = \frac{1}{3}, \quad \alpha_4 = \frac{1}{6}, \qquad \Delta x_1 = \frac{h}{2}, \quad \Delta x_2 = \frac{h}{2}, \quad \Delta x_3 = h, \quad \Delta x_4 = h.$$

Числа $\alpha_j$ и $\Delta x_j$ можно представить в виде

$$\alpha_j = \frac{1}{3}2^{-l}j; \qquad \Delta x_j = h2^{-m}j$$

($l_j$ и $m_j$ — целые числа) и получать с помощью операции 07 (вычитания порядков). Для этого в библиотечной подпрограмме используется специальная константа, которую будем обозначать буквой μ.. Величины $l_j$ и $m_j$ должны иметь значения

$$l_1 = 0, \quad m_1=1, \quad l_2 = 1, \quad m_2 = 1, \quad l_3 = 0, \quad m_3 = 0, \quad l_4=1, \quad m_4 = 0.$$

Каждую из них представим в виде шестиразрядного двоичного числа и запишем в прямом коде. Будем иметь:

$$[l_1]_{\text{пр}} = 0\ 000\ 000, \quad [m_1]_{\text{пр}} = 0\ 000\ 001, \quad [l_2]_{\text{пр}} = 1000\ 001,$$
$$[m_2]_{\text{пр}} = 0000\ 001, \quad [l_3]_{\text{пр}} = 0\ 000\ 000, \quad [m_3]_{\text{пр}} = 0000000,$$
$$[l_4]_{\text{пр}} = 0\ 000\ 001, \quad [m_4]_{\text{пр}} = 0\ 000\ 000.$$

Располагая эти коды в порядке $m_4$, $l_4$, $m_3$, $l_3$,. $m_2$, $l_2$ составим набор из пятидесяти шести двоичных цифр. Удерживая последние сорок три цифры этого набора, будем иметь упомянутую выше константу:

$$\mu = 1\ 000000000000000000000110000010000000010\ 000\ 000$$

или в коде команд

$$\mu = 4000\ 0006\ 0201\ 0\ 00.$$

Первоначально порядок константы μ равен $l_1$. Величина $\alpha_1$ вычисляется путем уменьшения порядка числа $\frac{1}{3}$ на $l_1$; т. е. на порядок константы μ. (операцией 07— вычитания порядков). Затем константа μ сдвигается вправо на семь разрядов. Порядок μ становится равным $m_1$. Величина $\Delta x_1$ вычисляется путем уменьшения порядка числа $h$ на $m_1$ (с помощью операции 07). Затем $\mu$ снова сдвигается на семь разрядов вправо. Продолжая этот процесс, можно вычислить все значения $\alpha_j$ и $\Delta x_j$.

Константа μ используется в программе также для контроля количества выполнений цикла, зависящего от параметра $j$ и вычисляющего $\sum_{j=1}^{4} K_{i,j}\alpha_j$ .

Для упрощения программы к системе уравнений

$$y'_i = f(x, y_1, y_2, \ldots, y_n) \qquad i=1,2,\ldots, n,$$

добавлено еще одно уравнение:

$$x'=1.$$

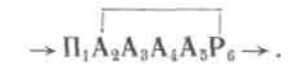
Теперь значения $x_{k+1}$ и $x_k + \Delta x_j$ можно вычислять по той же схеме, что и $y_{i,\,k+1}$ и $y_{i\,,k} + K_{i\,,\,j}$ Распределение адресов библиотечной подпрограммы показано в таблице 44, где

$$a_i = a_0+i, \quad b_i = b_0+ i, \quad c_i = c_0+i, \quad d_i = d_0+i, \quad (i = 1, 2, \ldots, n)$$

причем

$$b_0 = a_n+1, \quad c_0 = b_n+1.$$

Логическая схема библиотечной подпрограммы имеет такой вид:




В этой схеме $\mathbf{П}_1$ — оператор, переносящий числа из ячеек $a_0$—$a_n$ соответственно в ячейки $b_0$—$b_n$ и $c_0$—$c_n$, а также число μ в ячейку $r$. $\mathbf{A}_2$ — оператор, вычисляющий значения правых частей уравнений системы. Этот оператор для каждой конкретной системы дифференциальных уравнений имеет свой вид и потому в подпрограмме не приведен. $\mathbf{A}_3$ — оператор, вычисляющий $\alpha_j$ , сдвигающий ($r$) на семь разрядов вправо и вычисляющий $\Delta x_j$. $\mathbf{A}_4$ вычисляет

$y_{i+k} + K_{i,j}$. $\mathbf{A_5}$ вычисляет $y_{i+k} + \sum_{j=1}^{4} K_{ij}\alpha_j$ . $\mathbf{P}_6$ — оператор, сдвигающий ($r$) на семь разрядов вправо и контролирующий количество выполнений цикла.

Таблица 44. Распределение адресов библиотечной подпрограммы

| Номера ячеек | Содержимое ячеек | Пояснения |
|---|---|---|
| $a_0$<br>$a_1$<br>$a_2$<br>...<br>$a_n$ | $x$<br>$y_1$<br>$y_2$<br>...<br>$y_n$ | Перед началом ($k$+1)-го шага интегрирования в этих ячейках находятся числа $x_k$, $y_{i,\ k}$. После совершения ($k$+1)-го шага интегрирования здесь будут находиться числа $x_{k+1}$, $y_{i,\ k+1}$. |
| $b_0$<br>$b_1$<br>$b_2$<br>...<br>$b_n$ | $x$<br>$y_1$<br>$y_2$<br>...<br>$y_n$ | В этих ячейках сохраняются на время выполнения одного шага интегрирования числа $x_k$, $y_{i,\ k}$. |
| $c_0$<br>$c_1$<br>$c_2$<br>...<br>$c_n$ | $x$<br>$y_1$<br>$y_2$<br>...<br>$y_n$ | В эти ячейки засылаются $x_k$, $y_{i,\ k}$ для вычисления оператором $\mathbf{A}_2$ значений правых частей уравнений системы. |
| $d_0$<br>$d_1$<br>$d_2$<br>...<br>$d_n$ | $x'$<br>$y'_1$<br>$y'_2$<br>...<br>$y'_n$ | В эти ячейки записываются результаты вычисления правых частей уравнений системы, получаемые оператором $\mathbf{A}_2$. |

Библиотечная п о д п р о г р а м м а*

| | | | | | | |
|---|---|---|---|---|---|---|
| $\mathbf{П}_1$ | $e$+1) | $a_0$ | 2$n$+1 | $b_0$ | 0 | 45 |
| | $e$+2) | <μ> | 0000 | $r$ | 0 | 13 |
| $\mathbf{A}_2$ | $e$+3) | …….. | | | | |
| | …… | | | | | |
| | $e$+$g$) | …….. | | | | |
| $\mathbf{A}_3$ | $e$+$g$+1) | <1/3> | $r$ | <$\alpha_j$> | 0 | 07 |
| | $e$+$g$+2) | $r$ | 4107 | $r$ | 0 | 14 |
| | $e$+$g$+3) | <$h$> | $r$ | <$\Delta x_j$> | 0 | 07 |
| $\mathbf{A}_4$ | $e$+$g$+4) | 0000 | $n$ | 0000 | 0 | 35 |
| | $e$+$g$+5) | $d_0$ | <$\Delta x_j$> | $d_0$ | 0 | 05 |
| | $e$+$g$+6) | 0000 | $n$ | 0000 | 0 | 37 |
| | $e$+$g$+7) | $b_0$ | $d_0$ | $c$ | 0 | 01 |
| $\mathbf{A}_5$ | $e$+$g$+10) | 0000 | $n$ | 0000 | 0 | 35 |
| | $e$+$g$+11) | $d_0$ | <$\alpha_j$> | $d_0$ | 0 | 05 |
| | $e$+$g$+12) | 0000 | $n$ | 0000 | 0 | 37 |
| | $e$+$g$+13) | $a_0$ | $d_0$ | $a_0$ | 0 | 01 |
| $\mathbf{P}_6$ | $e$+$g$+14) | $r$ | 4107 | $r$ | 0 | 14 |
| | $e$+$g$+15) | $e$+3 | $e$+$g$+16 | 0000 | 0 | 20 |

**3. Автоматизация метода библиотечных подпрограмм.** Очевидно, для включения открытой библиотечной подпрограммы в состав основной программы необходимо выполнить трудоемкую работу, имеющую ярко выраженный формальный характер. Для этой работы существуют вполне определенные правила, которые могут быть представлены в виде некоторой программы. Следовательно, включение библиотечной подпрограммы в основную программу может быть произведено с помощью специальной программы, которую называют *объединяющей* программой.

Ввиду того, что буквенно-числовые обозначения в большинство машин не могут быть введены, библиотечную подпрограмму, предназначенную для автоматического включения в основные программы, составляют в действительных или условных адресах. Условными адресами называются числа, заменяющие собой буквенно-числовые обозначения.

Для того чтобы выяснить формальные правила для включения библиотечной подпрограммы в состав основной программы и обрисовать содержание информации, необходимой для этого, проанализируем библиотечную подпрограмму.

* Напоминаем, что симаол <$N$> означает номер ячейки, хранящей число $N$..

Если библиотечная подпрограмма составлена в общем виде (например, для решения системы любого числа дифференциальных или алгебраических уравнений и т. п.), то адреса ее команд и используемые ею константы делятся на два класса:

1) общие адреса и константы;

2) конкретные адреса и константы.

Общие адреса и константы подпрограммы зависят от одного или нескольких чисел, называемых параметрами подпрограммы. Прежде чем производить включение подпрограммы в основную программу, необходимо задать значения ее параметров и в соответствии с этими значениями вычислить конкретные значения общих адресов и констант.

Для конкретизации общих адресов и констант библиотечной подпрограммы применяется специальная конкретизирующая программа.

Вид конкретизирующей программы зависит от вида библиотечной подпрограммы, поэтому для каждой библиотечной подпрограммы должна быть составлена своя конкретизирующая программа. Информацией для работы конкретизирующей программы, кроме библиотечной подпрограммы, являются числовые значения параметров подпрограммы. В дальнейшем предполагается, что библиотечная подпрограмма имеет уже конкретный вид.

Библиотечная подпрограмма состоит из ряда команд (собственно подпрограммы) и ряда относящихся к ней чисел. Числа, относящиеся к подпрограмме, называют ее *константами* и делят на три группы: числовые константы, константы переадресации и константы восстановления.

К числовым константам относятся числа, имеющие количественное значение. Обычно это — коэффициенты формул, по которым библиотечная подпрограмма ведет вычисления.

К группе констант переадресации относят все вспомогательные числа, необходимые для работы подпрограммы и не зависящие ни от вида программы, в которую производится включение библиотечной подпрограммы, ни от места оперативной памяти, в котором размещена подпрограмма и ее элементы. В эту группу входят константы переадресации, константы для выделения частей чисел или команд, константы для сравнения с ними содержимого счетчиков и т. п.

К группе констант восстановления принадлежат все вспомогательные числа, имеющие вид команд и подлежащие при включении библиотечной подпрограммы в основную программу такой же переработке, как и команды подпрограммы. Сюда относятся в первую очередь константы для восстановления изменяемых команд подпрограммы, далее константы для формирования команд и т. п.

Кроме того, библиотечная подпрограмма использует при своей работе ряд рабочих ячеек.

При составлении библиотечной подпрограммы удобно ее команды, числовые константы, константы переадресации, константы восстановления и рабочие ячейки считать расположенными в виде сплошных массивов. Из дальнейшего читатель увидит, что и для автоматического включения подпрограммы в состав основной программы такое распределение адресов подпрограммы представляет большие удобства. Будем поэтому предполагать, что в библиотечной подпрограмме команды, константы каждого вида (числовые, переадресации, восстановления) и рабочие ячейки представлены в виде массивов.

При включении библиотечной подпрограммы в состав основной программы константы переадресации и числовые константы остаются неизменными, а константы восстановления изменяются по тем же правилам, что и команды. Следовательно, при составлении библиотечной подпрограммы нельзя какую-либо константу использовать (для экономии места) и как константу переадресации или числовую, и как константу восстановления. Это требование не накладывает значительных ограничений на вид библиотечных подпрограмм.

Адреса команд и констант восстановления библиотечной подпрограммы можно разбить на следующие классы:

1. *Внутренние адреса.* Внутренними адресами называют адреса, не зависящие от того, в какую программу включается подпрограмма, но зависящие от расположения в оперативной памяти основных массивов подпрограммы. К этому классу относятся адреса, являющиеся номерами ячеек, отведенных для команд и констант подпрограмм, и номерами ее рабочих ячеек.

2. *Внешние адреса.* Внешними адресами называются адреса, не зависящие от расположения подпрограммы и всех ее массивов в оперативной памяти, но зависящие от той основной программы (и принятого для нее распределения памяти), в которую производится включение подпрограммы. Внешними адресами являются номера ячеек, отведенных для команд основной программы (ее части, внешней относительно подпрограммы); для исходных данных задачи и промежуточных результатов, служащих исходными данными для работы подпрограммы; для результатов, выдаваемых подпрограммой и используемых основной программой.

3. *Постоянные адреса.* Постоянными адресами являются адреса, не зависящие ни от вида программы, в которую производится включение подпрограммы, ни от места оперативной памяти, отведенного для самой подпрограммы и ее массивов. Например, второй адрес предварительной команды групповой операции будет постоянным адресом. Номера ячеек *УВК* являются постоянными адресами и т. п.

При включении библиотечной подпрограммы в состав основной программы необходимо все ее внутренние адреса видоизменить в соответствии с ее расположением в оперативной памяти машины. При этом бывает целесообразным:

а) рабочие ячейки подпрограммы разместить в той же части памяти, где размещены остальные рабочие ячейки основной программы;

б) константы каждого вида, используемые подпрограммой, расположить в тех же местах памяти, где расположены соответствующие виды констант основной программы, и произвести экономию констант (т. е. исключение тех констант подпрограммы, которые совпадают с константами, используемыми в основной программе).

При этом нужно иметь в виду, что не всегда рабочие ячейки и константы подпрограммы являются «одиночными» и позволяют произвольное размещение в памяти. Некоторые рабочие ячейки должны быть расположены группами и иметь последовательные номера (например, рабочие ячейки, в которые производится групповой перенос каких-либо чисел). Точно так же некоторые группы констант должны быть расположены в ячейках с последовательными номерами (например, константы, участвующие в групповых операциях).

Включение библиотечной подпрограммы в состав основной программы заключается в том, что после изменения внутренних адресов (в соответствии с размещением подпрограммы в оперативной памяти машины) внешние адреса подпрограммы заменяются номерами ячеек, хранящих отвечающие им команды и числа основной программы.

Таким образом, для включения библиотечной подпрограммы в основную программу необходимо знать распределение ее адресов. Поэтому к каждой подпрограмме должна быть приложена таблица распределения адресов (сокращенно *ТРА*). В этой таблице должны быть приведены следующие данные:

1) номера ячеек, занятых собственно подпрограммой (обычно указывается номер начальной ячейки и количество команд);

2) номера ячеек, содержащих числовые константы; перечень неделимых групп констант;

3) номера ячеек, содержащих константы переадресации; перечень неделимых групп констант;

4) номера ячеек, содержащих константы восстановления; перечень неделимых групп констант;

5) номера рабочих ячеек, перечень неделимых групп рабочих ячеек;

6) перечень внешних адресов подпрограммы с указанием, какой величине отвечает каждый из адресов; например, если подпрограмма ведет счет по формуле

$$z = f(x, y),$$

то должно быть указано, какие адреса отвечают каждой из букв *x, y, z*;

7) особо должен быть указан выход из подпрограммы (*выходом* называется адрес, по которому передается управление после конца работы подпрограммы); если подпрограмма имеет несколько выходов, то должны быть даны пояснения, при каких условиях подпрограмма передает управление каждому выходу;

8) обычно входом подпрограммы является ее первая команда; если входом является не первая команда подпрограммы или если входов несколько, то должны быть указаны и снабжены пояснениями все ее входы.

**4. Объединяющая программа.** Объединяющая программа, предназначенная для включения библиотечных подпрограмм в основные программы или для составления программ из библиотечных подпрограмм, должна выполнять следующие операции:

1) расположение библиотечных подпрограмм и частей основной программы в определенном порядке;

2) переработку внутренних адресов библиотечных подпрограмм в соответствии с их размещением относительно основной программы;

3) одновременно с переработкой внутренних адресов (номеров рабочих ячеек и ячеек, хранящих константы) производить допустимую экономию констант;

4) связывать в одно целое библиотечные подпрограммы и части основной программы путем переработки внешних адресов библиотечных подпрограмм;

5) оставлять неизменными постоянные адреса библиотечных подпрограмм;

6) по окончании переработки библиотечных подпрограмм объединяющая программа должна готовую программу выдавать на перфокарты или записывать на магнитную ленту; вместе с выдачей готовой программы должна производиться выдача данных о распределении адресов готовой программы. Эти данные должны содержать следующие сведения:

а) о начальных номерах ячеек каждой части основной программы и каждой «вмонтированной» в основную программу библиотечной подпрограммы;

б) об общем распределении памяти для готовой программы, т. е. о том, какие ячейки отведены для исходных данных, для программы, для констант каждого вида, для результатов счета, какие ячейки являются рабочими.

Информация для работы объединяющей программы состоит из частей основной программы, библиотечных подпрограмм и таблиц соответствия адресов библиотечных подпрограмм адресам составляемой программы (указывающих, как должны быть изменены внутренние и внешние адреса подпрограмм).

Такие таблицы соответствия адресов (сокращенно *ТСА*) составляются программистом на основании приложенных к библиотечным подпрограммам таблиц распределения адресов (*ТРА*).

Отметим, что экономию констант при включении библиотечных подпрограмм в основную программу объединяющая программа может производить только в том случае, если ей задается информация о расположении констант основной программы. Такая информация становится наиболее краткой, если программа, в которую должны быть включены подпрограммы, составляется так, чтобы ее команды, числовые константы, константы переадресации и константы восстановления располагались сплошными массивами. При этом и объединяющая программа получается наиболее простой.

Иногда применяются объединяющие программы, рассчитанные на включение библиотечных подпрограмм в основные программы, составленные по особым правилам в условных адресах (или в условных числах) (см. §53, п.6).При этом после включения библиотечных подпрограмм в основную программу получается общая программа, требующая для приведения в окончательный вид дополнительной переработки. Такие объединяющие программы самостоятельного применения не имеют. Они могут являться частями более сложных программирующих программ.

## § 39. Автоматизация отдельных работ при ручном программировании

Стремление облегчить труд программистов и уменьшить количество ошибок, допускаемых при составлении и переписке программ, а также при устранении обнаруженных в программах ошибок, привело к мысли: возложить часть технической работы, связанной с программированием, на саму электронную цифровую вычислительную машину.

Составление программ, под управлением которых цифровая машина выполняла бы ту или иную подсобную работу, необходимую при программировании, требовало тщательного изучения правил, по которым эту работу выполняет человек, и точной формулировки таких правил. Полная система правил, по которым выполняется какая-либо работа, называется ее *алгорифмом.*

В настоящем параграфе рассмотрим две программы, предназначенные для выполнения отдельных работ, необходимых при программировании.

**1. Автоматизация присвоения действительных адресов.** После составления программы в буквенно-числовых обозначениях, подсчета количества ее команд и числа ячеек, необходимых для всего числового материала, относящегося к программе, программист производит распределение памяти. Следующий этап работы состоит **в** замене буквенно-числовых обозначений действительными номерами ячеек, т. е. в присвоении программе действительных адресов. Правила, по которым производится присвоение действительных адресов, весьма просты, и

потому автоматизация этой работы (передача ее цифровой электронной машине) особых трудностей не встречает.

Буквенно-числовые обозначения нельзя ввести в память большинства современных цифровых машин. Но вместо буквенно-числовых обозначений можно применять числовые обозначения, подобные буквенно-числовым. Например, вместо *a*+1, *a*+2, *a*+3, ... можно писать 1001, 1002,1003,..., вместо *b*+1, *b*+2, *b*+3,...— писать 2001, 2002, 2003, ... и т. д. Такие числовые обозначения получили название *символических адресов.*

Программист вручную составляет программу в символических адресах и производит распределение памяти. Работа же по замене символических адресов действительными адресами возлагается на электронную цифровую машину. Машина выполняет эту работу по специальной, раз и навсегда составленной программе. Исходными данными для такой специальной программы являются, во-первых, программа, составленная в символических адресах, и, во-вторых, таблица соответствия между группами символических адресов и группами действительных адресов. В такой таблице, например, может быть указано, что символическому адресу вида 1000 отвечает действительный адрес 0553. Это значит, что последовательным символическим адресам 1001, 1002, 1003, ... должны отвечать действительные адреса 0554, 0555, 0556, ...

После переработки символических адресов производится выдача готовой программы на перфокарты.

Для более сложной перерабатывающей программы вместо таблиц соответствия символических адресов действительным адресам может служить информацией таблица последовательности символических адресов. Эта таблица может иметь, например, такой вид:

1001 — 1020
1100 —1125
1021 — 1077
1126 —1755
……..

Содержание приведенной таблицы следующее. Символические адреса, начиная с 1001 и кончая 1020, должны быть заменены последовательными действительными адресами, начиная с некоторого определенного номера (например, начиная с 0020). Затем символические адреса, начиная с 1100 и кончая 1125, должны быть заменены последовательными действительными адресами, начиная с действительного адреса, который на единицу больше последнего, уже использованного действительного адреса, и т. д.

Перерабатывающая программа с помощью таблицы последовательности символических адресов составляет таблицу их соответствия действительным адресам, после чего производит в перерабатываемой программе замену символических адресов действительными адресами.

Метод символических адресов удобен при ручном программировании и освобождает программиста от формальной работы по замене буквенно-числовых обозначений действительными адресами. Метод символических адресов позволяет легко исправлять некоторые ошибки, обнаруженные в процессе программирования. К числу таких ошибок относится, например, пропуск команд. Например, требовалось составить программу для счета по формуле

$$y = ax^6 + \sin(bx + \sqrt{x^2 - 1} + \sqrt{x^2 + (\ln x)^2 + 1}) .$$

В результате ошибки была составлена программа для счета по формуле

$$y = ax^2 + \sin(bx + \sqrt{x^2 - 1}) .$$

При этом символические адреса были использованы следующим образом. Номера команд обозначались числами 1001, 1002, …, рабочие ячейки — числами 3001, 3002, ... Исходные данные обозначались так:

$x$ = (2001); $a$ = (2002); $b$ = (2003); 1 = (2004).

Для обозначения результатов счета были применены символические адреса 4001, 4002, ...

Составленная программа имела следующий вид:

| | | | | | |
|---|---|---|---|---|---|
| 1001) | 2001 | 2001 | 3001 | 0 | 05 |
| 1002) | 2002 | 3001 | 3002 | 0 | 05 |
| 1003) | 2003 | 2001 | 3003 | 0 | 05 |
| 1004) | 3001 | 2004 | 3004 | 0 | 03 |
| 1005) | 3004 | 0000 | 3004 | 0 | 63 |
| 1006) | 3003 | 3004 | 3003 | 0 | 01 |
| 1007) | 3003 | 0000 | 3003 | 0 | 67 |
| 1010) | 3002 | 3003 | 4001 | 0 | 01 |

Произведя проверку, программист установил, что после команды (1006) необходимо добавить ряд команд для устранения допущенной ошибки. Сделать это можно следующим образом:

| | | | | | |
|---|---|---|---|---|---|
| 1001) | 2001 | 2001 | 3001 | 0 | 05 |
| 1002) | 2002 | 3001 | 3002 | 0 | 05 |
| 1003) | 2003 | 2001 | 3003 | 0 | 05 |
| 1004) | 3001 | 2004 | 3004 | 0 | 03 |
| 1005) | 3004 | 0000 | 3004 | 0 | 63 |
| 1006) | 3003 | 3004 | 3003 | 0 | 01 |
| 1011) | 2001 | 0000 | 3004 | 0 | 66 |
| 1012) | 3004 | 3004 | 3004 | 0 | 05 |
| 1013) | 3001 | 3004 | 3004 | 0 | 01 |
| 1014) | 3004 | 2004 | 3004 | 0 | 01 |
| 1015) | 3004 | 0000 | 3004 | 0 | 63 |
| 1016) | 3003 | 3004 | 3003 | 0 | 01 |
| 1007) | 3003 | 0000 | 3003 | 0 | 67 |
| 1010) | 3002 | 3003 | 4001 | 0 | 01 |

Для того чтобы присвоение действительных адресов было выполнено машиной правильно, таблица последовательности символических адресов должна иметь такой вид:

1001 — 1006
1011 — 1016
1007 — 1010
.....

**2. Автоматизация исправления некоторых ошибок, обнаруженных в программе.** При ручном программировании нередко допускают ошибки, состоящие в пропуске одной или нескольких команд программы. Исправление такого рода ошибок производят одним из двух способов.

Первый способ состоит в том, что в программе делают так называемую «выноску», т. е. размещают ту команду, после которой имеется пропуск, и вслед за нею пропущенную группу команд вне программы, в свободной части оперативной памяти, а на месте команды, изъятой из программы, ставят команду условного перехода к этой выноске. В конце выноски ставят команду возврата, т. е. команду безусловного перехода к команде программы, которая должна выполняться вслед за пропущенными командами. Такое исправление программы усложняет ее логическую схему и затрудняет ее проверку и отладку на машине. Трудности делаются значительными при наличии большого количества выносок.

Второй способ исправления программы состоит в том, что программу раздвигают, вписывают на освободившееся место пропущенные команды, а остальные команды программы просматривают и, если нужно, изменяют их адреса. Этот способ при выполнении его вручную по существу сводится к пересоставлению программы. Он может быть целесообразным только при условии, что исправление адресов программы будет выполнено самой цифровой машиной (автоматически).

Иногда переделка программы, аналогичная только что описанной, производится не для устранения ошибок, а с целью приспособления программы для решения новой задачи. При этом может потребоваться, кроме вставления новых команд, также удаление некоторой группы команд из программы или замена отдельной команды или группы команд новыми командами. Иногда бывает желательным перемещение какой-нибудь группы чисел из одних ячеек памяти в другие (т. е. «перенумерация» группы чисел).

Для вставления команд, удаления команд, замены команд и перенумерации чисел, рабочих ячеек или команд, научным сотрудником Г. Д. Фроловым разработана специальная программа, получившая название программы *ВУЗП* (программы вставления, удаления, замены, перенумерации).

**3. Программа *ВУЗП*.** Для работы программы *ВУЗП* в машину, кроме самой программы *ВУЗП,* вводятся управляющая информация и подлежащая исправлению программа. Управляющая информация состоит из приказов (команд для программы *ВУЗП*) и относящихся к ним групп команд (предназначенных для вставления или используемых при замене). По каждому из приказов программа *ВУЗП* выполняет одну из своих операций: перенумерацию, замену, удаление или вставление.

Операции, выполняемые программой *ВУЗП,* состоят в следующем:

1. Перенумерация (адресов) состоит в том, что все адреса перерабатываемой программы и управляющей информации, удовлетворяющие некоторому неравенству

$$d \le A \le d+l$$

($d$, $l$ — два восьмеричных числа, $A$ — адрес команды), увеличиваются или уменьшаются на одно и то же число.

Операция перенумерации производится программой *ВУЗП* по следующему приказу, имеющему вид команды:

$$d \quad l \quad c \quad 0 \quad 03,$$

где $d$ — адрес, начиная с которого $l$ адресов подлежат изменению. При этом адрес $d$ должен быть заменен на $c$; $d + 1$ — на $c + 1$...; $d + l$ — 1 — на $c + l$— 1. Код операции перенумерации 03.

2. Замена команды или группы команд состоит в том, что

а) на место заменяемой команды (группы команд) записывается новая команда (группа команд);

б) просматриваются адреса перерабатываемой программы и управляющей информации и, если нужно, исправляются.

Эта операция выполняется программой *ВУЗП* по следующему приказу, имеющему вид команды:

$$a \quad l \quad 0000 \quad 0 \quad 02,$$

где $a$ — восьмеричное число, на единицу большее номера команды (т. е. отведенной для команды ячейки), после которой начинается группа заменяемых команд; $l$ — количество команд, подлежащих замене. Код операции замены 02.

Непосредственно вслед за приказом замены должны быть записаны те команды, которыми производится замена.

3. Удаление команды или группы команд заключается в том, что

а) перерабатываемая программа сдвигается так, что удаляемые команды оказываются стертыми;

б) производится просмотр адресов перерабатываемой программы и управляющей информации и необходимое исправление их.

Программа *ВУЗП* выполняет эту операцию на основании следующего приказа, имеющего вид команды:

$$a \quad l \quad 0000 \quad 0 \quad 00,$$

где $a$ — восьмеричное число на единицу большее, чем номер команды (в программе, подлежащей переработке), после которой расположена группа команд, подлежащих удалению; $l$ — число удаляемых команд. Код операции удаления 00.

4. Вставление команды или группы команд состоит в том, что

а) обрабатываемая программа раздвигается;

б) на освободившееся место записывается вставляемая команда (группа команд);

в) производится просмотр адресов перерабатываемой программы и управляющей информации и, если это нужно, исправление адресов.

Операция вставления выполняется программой *ВУЗП* по приказу

$$a \quad l \quad 0000 \quad 0 \quad 01,$$

где $a$ — число на единицу большее, чем номер команды перерабатываемой программы, после которой должна быть помещена вставляемая группа команд; $l$ — число вставляемых команд. Код операции вставления 01.

Непосредственно вслед за приказом должна стоять вставляемая группа команд.

Информация для работы программы *ВУЗП* составляется на стандартных бланках для программирования следующим образом.

Считается, что строки бланков имеют номера, начиная с 6020. На первой строке первого бланка записывается начальная строка информации, т. е. число, имеющее вид команды

| | | | | |
|---|---|---|---|---|
| $m_1m_2$ | $a_{\text{нач}}$ | $N$ | 0 | 00 |

Здесь $m_1$ и $m_2$ — двузначные восьмеричные числа; $m_1$ — количество первых строк первой перфокарты из комплекта, содержащего перерабатываемую программу, на которых пробит материал, не подлежащий переработке; $m_1$ удовлетворяет неравенству $0 \le m_1 \le 13$ (восьмеричное), ибо, если первая перфокарта содержит только материал, не требующий переработки, она может быть отброшена; $m_2$ — количество последних строк последней перфокарты комплекта, содержащего перерабатываемую программу, на которых пробит материал, не подлежащий переработке; очевидно, тоже $0 \le m_2 \le 13$. Число $a_{\text{нач}}$ означает начальный номер команд перерабатываемой программы до исправления. $N$ — восьмеричное число, являющееся адресом, начиная с которого адреса программы не подлежат видоизменению. Это число необходимо для того, чтобы программа *ВУЗП* могла перерабатывать программы, у которых числовой материал или рабочие ячейки расположены в конце памяти машины. Без ограничения адресов, подлежащих переработке, адреса, относящиеся к таким числам и рабочим ячейкам, могли бы стать больше, чем 3777, т. е. «выйти» за пределы оперативной памяти.

Вслед за указанной строкой на бланках последовательно выписывают все приказы о перенумерации, затем все приказы о заменах, затем все приказы об удалении и, наконец, все приказы о вставлении.

Как уже говорилось, непосредственно вслед за каждым приказом о замене или вставлении записывается группа относящихся к нему команд (команд, которыми производится замена, или команд, которые должны быть вставлены). При составлении этих команд все адреса, относящиеся к программе, подлежащей переработке, пишут в действительном виде (отвечающем неисправленной программе). Все адреса, относящиеся к группам вставляемых команд или команд, которыми производится замена (например, адреса команд условного перехода, переадресации и т. п.), должны совпадать с номерами соответствующих строк бланка (т. е. превосходить 6020).

Для программы *ВУЗП* составлена и пробита на перфокартах специальная программа ввода, для использования которой необходимо приложить дополнительную перфокарту с двумя строками:

$$\begin{array}{llllll} 0000 & n_1 - 1 & 0000 & 0 & 00 \\ 0000 & n_2 - 1 & 0000 & 0 & 00, \end{array}$$

где $n_1$ — количество строк, занимаемых управляющей информацией (т. е. начальной строкой информации, приказами и относящимися к приказам группами команд); $n_2$ — количество строк, занимаемых исправляемой программой (включая не подлежащие обработке первые строки первой перфокарты и последние строки последней перфокарты). По этой программе ввода в память машины вводится сперва программа *ВУЗП,* затем управляющая информация и непосредственно в следующие за информацией ячейки — подлежащая переработке программа.

Правила, по которым программа *ВУЗП* перерабатывает исправляемую программу, чрезвычайно просты.

1. Производится просмотр управляющей информации и последовательное выполнение приказов. После выполнения каждого приказа этот приказ и относящаяся к нему группа команд (если такая имеется) стираются. Выполнение приказов сводится к следующему.

2. Если этого требует операция, производится включение группы команд в состав перерабатываемой программы.

3. Производится просмотр управляющей информации и перерабатываемой программы. Одновременно выполняется необходимое исправление адресов, описанное ниже.

4. Адреса, начинающиеся цифрой 6 и являющиеся внутренними для включенной в обрабатываемую программу группы команд, уменьшаются на номер начальной строки этой группы и увеличиваются на число $a$ (стоящее в первом адресе приказа). Остальные адреса, начинающиеся цифрой 6, остаются неизменными.

5. Вторые адреса приказов и вторые адреса команд, имеющих код операции, не меньший чем 30, никаким изменениям не подвергаются.

6. Вторые адреса команд, имеющих код операции 14, не подвергаются исправлению, если превосходят 3777.

7. При выполнении операции удаления все адреса $A$, удовлетворяющие неравенству

$$a + l \le A < N$$

уменьшаются на число $l$.

8. При выполнении операции вставления все адреса $A$, удовлетворяющие неравенству

$$a \le A < N$$

увеличиваются на число $l$.

9. При выполнении операции перенумерации все адреса $A$, удовлетворяющие неравенству

$$d \le A < d + l,$$

увеличиваются (или уменьшаются) на одно и то же число.

10. После того как все приказы программой *ВУЗП* выполнены, переработка исправляемой программы окончена. Производится ее выдача на перфокарты.

# ГЛАВА VIII
# ОСОБЕННОСТИ РЕШЕНИЯ ЗАДАЧ НА ЭЛЕКТРОННЫХ ЦИФРОВЫХ МАШИНАХ

## § 40. Методы контроля

Неправильности в работе электронной цифровой машины называют *сбоями.* Сбои подразделяют на *систематические* и *случайные.* Систематические сбои происходят в результате неисправности устройств машины и носят устойчивый характер (повторяются при пересчете). Они устраняются техническим составом, обслуживающим машину, путем своевременной замены вышедших из строя деталей (электронных ламп, сопротивлений и т. п.) и последующей наладки машины. Машина, подготовленная для решения задачи, как правило, в начале работы систематических сбоев не делает; они могут появиться во время решения, и тогда решение необходимо прервать для их устранения.

Случайные сбои могут происходить и при работе совершенно исправной машины. Наличие их связано отчасти с недостаточной надежностью работы отдельных устройств машины (с недостатками ее конструкции), отчасти с внешними, не поддающимися учету причинами. Например, сбой может произойти под влиянием какого-нибудь мощного случайного электромагнитного импульса (помехи), воздействовавшего на машину. В отличие от систематических, случайные сбои не носят устойчивого характера. Появление случайного сбоя приводит к получению неправильного результата, однако останавливать и проверять машину при случайном сбое не следует. Достаточно пересчитать тот результат, который оказался неправильным.

Наконец, неправильные результаты можно получать даже при отсутствии как систематических, так и случайных сбоев в работе машины. Это вызывается ошибками в программе, которые могут возникать на любом этапе подготовки к решению задачи.

Рассмотрим теперь особенности контроля на различных стадиях подготовки и решения задач с помощью электронных цифровых машин.

**1. Проверка программы и контроль правильности ее ввода.** Уже при составлении программы решения задачи могут быть допущены ошибки. Тщательной проверкой программы на бланках стараются эти ошибки устранить. Одна неправильно написанная цифра в адресе или коде операции команды, как правило, приводит в негодность всю программу.

В процессе переноса программы с бланка на перфокарты опять возможны ее искажения. Эти искажения выявляются и устраняются следующим способом. Программа переносится на перфокарты, как говорят, в две руки, т. е. ее пробивают на двух комплектах перфокарт два человека независимо друг от друга. Затем каждая перфокарта первого комплекта сравнивается с соответствующей перфокартой второго комплекта при помощи специального электрического устройства, называемого *контрольником.* Одноименные перфокарты, оказывающиеся при сравнении не тождественными, извлекаются из комплектов. Они подлежат замене новыми перфокартами, пробиваемыми также в две руки и сличаемыми затем на контрольнике.

При наличии восьмеричного печатающего устройства можно также применять следующий метод контроля. Программа наносится лишь на один комплект перфокарт. Последний вкладывается в печатающее устройство, и записанные на нем команды отпечатываются на бумажной ленте. Затем полученный отпечаток сличается с программой, записанной на бланках.

Программа, записанная на одном или на двух идентичных комплектах перфокарт, подлежит вводу в машину.

Процесс ввода программы в машину опять может сопровождаться ее искажением. Это искажение может быть вызвано сбоем в работе читающего устройства, памяти или управляющего устройства машины.

Во избежание этого в программе ввода необходимо предусмотреть контроль правильности ввода. При наличии двух идентичных комплектов перфокарт это можно осуществить следующим образом: предусмотреть ввод программы с первого комплекта перфокарт, контрольное суммирование всех введенных при этом чисел и запоминание полученной первой контрольной суммы, затем ввод программы со второго комплекта перфокарт в те же ячейки, в которые была введена программа в первый раз (при этом прежние числа стираются и заменяются новыми), контрольное суммирование всех введенных чисел и сравнение второй контрольной суммы с первой контрольной суммой.

При совпадении обеих контрольных сумм считают, что программа, введенная со второго комплекта перфокарт, не содержит искажений. Такое заключение основывается на следующих соображениях.

Перед началом ввода программы работа машины была проверена обслуживающим персоналом, и машина начинает свою работу без систематических сбоев. Если в процессе работы машины произойдет случайный сбой, то два введенных образца программы должны оказаться нетождественными. Вероятность же совпадения двух контрольных сумм при нетождественности программы, введенной в первый и во второй раз, чрезвычайно мала. Программа ввода может иметь такой вид:

```
0004)  0000    n—1   0020   0  41   — ввод первого комплекта перфокарт;
0005)  0000    n—1   0000   0  34 }
0006)  [0020]  0015  0015   0  17 } — первое контрольное суммирование
0007)  0000    n—1   0020   0  41   — ввод второго комплекта перфокарт;
0010)  0000    n—1   0000   0  34 }
0011)  [0020]  0016  0016   0  17 } — второе контрольное суммирование
0012)  0015    0016  0000   0  16   — сравнение контрольных сумм;
0013)  0020    0014  0000   0  20   — команда условного перехода;
0014)  0015    0016  0000   0  40   — останов при неправильном вводе;
0015)  0000    0000  0000   0  00 }
0016)  0000    0000  0000   0  00 } — ячейки для контрольных сумм;
```

```
0017)  0000    0000    0000    0  00    — неиспользуемая ячейка.
```

Здесь *n* — количество команд вводимой программы. Вводится программа в ячейки, начиная с 0020.

Практика показала, что контроль правильности ввода, основанный на использовании двух идентичных комплектов перфокарт, может приводить к большой непроизводительной затрате машинного времени. Например, если программу придется многократно вводить, то ввод ее с двух комплектов перфокарт потребует вдвое больше времени, чем ввод с одного комплекта. Можно обойтись вводом рабочей программы с одного комплекта перфокарт, если воспользоваться следующим способом контроля.

После нанесения программы на перфокарты производится ее ввод, например, по следующей программе ввода:

```
0004)  0000    n—1     0020    0  41    — ввод программы с перфокарт;
0005)  0000    n—1     0000    0  34    — первое контрольное суммирование
0006)  [0020]  0011    0011    0  17
0007)  0011    0000    0000    0  44    — выдача контрольной суммы на перфокарту;
0010)  0000    0000    0000    0  40    — останов;
0011)  0000    0000    0000    0  00    — ячейка для контрольной суммы;
0012)  0000    0000    0000    0  00
0013)  0000    0000    0000    0  00
0014)  0000    0000    0000    0  00
0015)  0000    0000    0000    0  00    — неиспользуемые ячейки.
0016)  0000    0000    0000    0  00
0017)  0000    0000    0000    0  00
```

По этой программе ввода машина введет рабочую программу, просуммирует ее команды (с помощью операции 17), выдаст контрольную сумму на перфокарту и остановится.

Сумму, выданную на перфокарту, с помощью входного перфоратора переносят на последнюю строку (отвечающую ячейке 0017) перфокарты, содержащей новую программу ввода, например такую:

```
0004)  0000    n—1     0020    0  41    — ввод программы с перфокарт;
0005)  0000    n—1     0000    0  34
0006)  [0020]  0012    0012    0  17    — контрольное суммирование;
0007)  0012    0017    0000    0  16    — сравнение новой и прежней контрольных сумм;
0010)  0020    0011    0000    0  20    — команда условного перехода;
0011)  0012    0017    0000    0  40    — останов при неправильном вводе;
0012)  0000    0000    0000    0  00    — ячейка для новой контрольной суммы;
0013)  0000    0000    0000    0  00
0014)  0000    0000    0000    0  00
0015)  0000    0000    0000    0  00    — неиспользуемые ячейки;
0016)  0000    0000    0000    0  00
0017)                                   — старая контрольная сумма.
```

Последняя программа ввода применяется в дальнейшем для ввода рабочей программы с одного комплекта перфокарт. Она предусматривает ввод рабочей программы с одного комплекта перфокарт, ее контрольное суммирование и сличение новой контрольной суммы со старой контрольной суммой, записанной в ячейку 0017. При совпадении двух контрольных сумм считают, что ввод рабочей программы произошел без искажений. При их несовпадении машина останавливается и выдает на пульт управления несовпадающие между собой контрольные суммы.

Можно первый ввод программы в оперативную память машины (когда контрольная сумма еще неизвестна) произвести по следующей программе ввода:

```
0004)  0000    n—1     0020    0  41    — ввод программы с перфокарт;
0005)  0000    n—1     0000    0  34
0006)  0020    0013    0013    0  17    — суммирование программы;
0007)  0015    0015    0000    0  20    — безусловный переход к командам, формирующим новую программу ввода;
0010)  0013    0012    0000    0  16
0011)  0020    0011    0000    0  20
0012   0012    0013    0000    0  40    — вспомогательные числа для формирования новой программы ввода;
0013)  0000    0000    0000    0  00
0014)  0000    0000    0000    0  00
```

```
0015)  0010   0004   0007   0  45 }
0016)  0004   0013   0000   0  44 } — формирование новой программы ввода и выдача ее на перфокарту;
0017)  0000   0000   0000   0  40   — останов.
```

По этой программе ввода будет составлена и выдана на перфокарту новая программа ввода, предназначенная для последующих вводов рабочей программы. Новая программа ввода будет иметь следующий вид:

```
0004)  0000   n—1    0020   0  41   — ввод программы с перфокарт;
0005)  0000   n—1    0000   0  34 }
0006)  0020   0013   0013   0  17 } — контрольное суммирование;
0007)  0013   0012   0000   0  16   — сравнение новой и прежней контрольных сумм;
0010)  0020   0011   0000   0  20   — условный переход;
0011)  0012   0013   0000   0  40   — останов при неправильном вводе;
0012)                               — старая контрольная сумма.
0013)  0000   0000   0000   0  00   — ячейка для новой контрольной суммы.
0014)  0000   0000   0000   0  00 }
0015)  0010   0004   0007   0  45 }
0016)  0004   0013   0000   0  44 } — неиспользуемые ячейки, содержащие остатки прежней программы ввода
0017)  0000   0000   0000   0  40 }
```

В приведенных примерах программ ввода размещение вводимой программы предусмотрено в ячейках, начиная с 0020; это, конечно, не является обязательным.

В заключение заметим, что контрольное суммирование производится с помощью операции 17, а не 01 потому, что в последнем случае почти наверное произойдет переполнение разрядной сетки и останов машины раньше, чем будет окончено вычисление контрольной суммы. Ведь при сложении команд с помощью операции 01 коды операций будут восприняты машиной как порядки чисел, а среди кодов операций встречаются такие, как 72, 70 и т.п., что соответствует очень большим порядкам.

**2. Отладка программы.** После того как программа введена, можно, казалось бы, приступить к ее выполнению, т. е. к машинному решению запрограммированной задачи. Но в действительности это не так. Дело в том, что проверка на бланках, осуществляемая «вручную», зачастую не выявляет всех ошибок, которые могли быть допущены в программе. Дополнительная проверка в машине правильности программы, введенной уже в память, называется *отладкой* ее на машине.

Один из способов отладки программы состоит в ее проверке при работе машины в режиме контрольных остановов. В программе еще при записи ее на бланках в некоторых командах должны быть проставлены контрольные знаки, равные единице. Расстановка контрольных знаков, равных единице, должна быть выполнена так, чтобы по остановам машины можно было проверить правильность логики программы.

При наличии контрольного знака 1 в некоторой команде машина выполняет эту команду и останавливается, одновременно выдавая на пульт управления выполненную команду, номер ячейки, хранящей команду, которая должна выполняться следующей, и содержимое трех ячеек, номера которых стоят в адресах выполненной команды. По выдаваемым на пульт управления числам можно судить (при целесообразном расположении контрольных знаков), все ли участки программы работают и правильно ли производится передача управления командами условного и безусловного переходов.

Лишь после такой проверки программы она считается отлаженной и используется для решения задачи. Комплект перфокарт, хранящий программу, считается тоже отлаженным на машине, и при использовании для повторного решения задач программа, вводимая с него, отладке больше не подвергается.

Вместо отладки программы с помощью контрольных остановов может быть произведена ее отладка путем сравнения результатов, даваемых машиной, с результатами, полученными заранее при помощи ручных вычислений. Обычно для этого вместо действительных исходных данных задачи берутся упрощенные исходные данные, равные нулям и единицам. Это делается для облегчения ручного счета, нужного для получения контрольных результатов.

После того как программа введена, на место исходных данных задачи заносятся упрощенные исходные данные и производится контрольное решение задачи. Программа считается отлаженной, если результаты, даваемые машиной, достаточно близки с контрольными результатами, полученными ручным счетом. Полного совпадения результатов может и не быть, так как машина ведет счет, вообще говоря, приближенно, хотя и с очень большой точностью.

В заключение заметим, что существуют специальные контролирующие программы, с помощью которых автоматически производится отладка программы на машине. Недостаток места не позволяет нам здесь остановиться на описании таких программ.

Контроль на последующих этапах должен быть предусмотрен в самой программе. Контролироваться может, во-первых, правильность работы машины, во-вторых, правильность вычислений.

**3. Контроль правильности работы машины.** Непрерывная проверка правильности работы машины необходима для исключения влияния случайных сбоев. Она осуществляется, например, *двойным счетом* каждого этапа задачи и сравнением получаемых первого и второго результатов. Если каждый счет приводит к большому количеству результатов, то применяется контрольное суммирование и сравнение не самих результатов, а их контрольных сумм. Покажем, как двойной счет может быть реализован в программе:

```
a+1)  …….  }
……    ……   } Участок программы, осуществляющий счет
a+k)  …….  }  и выдающий его результат (или контрольную
                сумму результатов) в ячейку b+1
```

| | | | | | | |
|---|---|---|---|---|---|---|
| $a+k+1)$ | 0000 | 0000 | 0000 | 0 | 00 | |
| $a+k+2)$ | $b+1$ | 0000 | $b+2$ | 0 | 13 | |
| $a+k+3)$ | $a+1$ | $a+1$ | $a+k+1$ | 0 | 27 | |
| $a+k+4)$ | $b+1$ | $b+2$ | 0000 | 0 | 16 | |
| $a+k+5)$ | $a+k+7$ | | $a+k+6$ | 0000 | 0 | 20 |
| $a+k+6)$ | $b+1$ | $b+2$ | 0000 | 0 | 40 | |
| $a+k+7)$ | ……. | | | | | |
| …… | | | | | | Продолжение вычислений. |
| ……. | | | | | | |

Выполнив команды $(a+1)$, ... , $(a+k)$, машина записывает результат счета (или контрольную сумму результатов счета) в ячейку $b+1$. Ячейка $a+k+1$ заполнена нулями (она предназначена для записи команды возврата при обращении к участку программы $a+1$, ..., $a+k$ для вторичного счета). Команда $(a+k+2)$ обеспечивает перенос полученного результата из ячейки $b+1$ в ячейку $b+2$ для хранения. Команда $(a+k+3)$ представляет собой обращение к контролируемому участку программы для вторичного счета. При этом новый результат оказывается записанным в ячейку $b+1$. По команде $(a+k+4)$ сравниваются старый и новый результаты счета. При совпадении их команда $(a+k+5)$ передает управление следующему участку программы для продолжения вычислений. При несовпадении результатов управление получает команда $(a+k+6)$, останавливающая машину и выдающая на пульт управления несовпавшие между собой результаты первого и второго счетов,

Иногда вместо двойного счета для контроля правильности работы машины употребляют так называемый *двойной-тройной* счет. Покажем, как он осуществляется в программе:

| | | | | | |
|---|---|---|---|---|---|
| $a+1)$ | $a+2$ | $a+2$ | $a+k+1$ | 0 | 27 |
| $a+2)$ | $b+2$ | 0000 | $b+3$ | 0 | 13 |
| $a+3)$ | $b+1$ | 0000 | $b+2$ | 0 | 13 |
| $a+4)$ | …… | | | | Контролируемый участок программы, |
| …… | | | | | выдающий результат (или контрольную сумму |
| $a+k)$ | …… | | | | результатов) в ячейку $b+1$ |
| $a+k+1)$ | 0000 | 0000 | 0000 | 0 | 00 |
| $a+k+2)$ | $b+1$ | $b+2$ | 0000 | 0 | 16 |
| $a+k+3)$ | $a+k+11$ | $a+2$ | $a+k+1$ | 0 | 27 |
| $a+k+4)$ | $b+1$ | $b+2$ | 0000 | 0 | 16 |
| $a+k+5)$ | $a+k+11$ | $a+k+6$ | 0000 | 0 | 20 |
| $a+k+6)$ | $b+1$ | $b+3$ | 0000 | 0 | 16 |
| $a+k+7)$ | $a+k+11$ | $a+k+10$ | 0000 | 0 | 20 |
| $a+k+10)$ | 0000 | 0000 | 0000 | 0 | 40 |
| $a+k+11)$ | ……. | | | | |
| ……. | | | | | Продолжение вычислений |
| ……. | | | | | |

В ячейке $a+1$ стоит команда обращения к участку программы $a+2$, $a+3$, ..., $a+k$, передающая управление команде $a+2$ и записывающая команду возврата в ячейку $a+k+1$. Ячейки $b+1$, $b+2$, $b+3$ называются контрольными ячейками. Пусть до работы программы $(b+1)=x$; $(b+2)=y$; $(b+3)=z$, где $x$, $y$ и $z$ — любые числа. Команда $(a+2)$ преобразует содержимое контрольных ячеек так:

$$(b+1)=x; \qquad (b+2)=y; \qquad (b+3)=y,$$

а команда $(a+3)$ так:

$$(b+1)=x; \qquad (b+2)=x; \qquad (b+3)=y.$$

Затем проработает контролируемый участок программы $a+4$, ... ..., $a+k$, выдающий результат своего счета $r_1$ в ячейку $b+1$. Содержимое контрольных ячеек примет такой вид:

$$(b+1)=r_1; \qquad (b+2)=x; \qquad (b+3)=y.$$

В ячейке $a+k+1$ стоит упомянутая выше самогасящаяся команда возврата, которая передает управление команде $(a+2)$ и сама себя стирает. Команда $(a+2)$ преобразует содержимое контрольных ячеек так:

$$(b+1)=r_1; \qquad (b+2)=x; \qquad (b+3)=x,$$

а команда $(a+3)$ так:

$$(b+1)=r_1; \qquad (b+2)=r_1; \qquad (b+3)=x.$$

Снова проработает контролируемый участок программы и запишет результат второго счета в ячейку $b+1$. Содержимое контрольных ячеек станет следующим:

$$(b+1)=r_2; \qquad (b+2)=r_1; \qquad (b+3)=x.$$

В ячейке $a+k+1$ теперь записаны нули, поэтому управление перейдет сперва к команде $(a+k+2)$, а затем к команде $(a+k+3)$. Эти команды произведут сравнение чисел $(b+1)=r_2$ и $(b+2)=r_1$. Если окажется, что $r_2=r_1$,

то управление будет передано команде ($a + k + 11$), начинающей новый участок программы. Если же окажется, что $r_2 \neq r_1$, то управление снова будет передано команде ($a + 2$) (при этом в ячейку $a + k + 1$ будет записана команда возврата), а ($a + 2$) преобразует содержимое контрольных ячеек следующим образом:

$$(b+1) = r_2; \qquad (b+2) = r_1; \qquad (b+3) = r_1.$$

Команда ($a + 3$) произведет дальнейшее преобразование содержимого контрольных ячеек:

$$(b+1) = r_2; \qquad (b+2) = r_2; \qquad (b+3) = r_1.$$

После этого в третий раз проработает контролируемый участок программы и запишет результат своего третьего счета $r_3$ в ячейку $b + 1$, Теперь содержимое контрольных ячеек будет следующим:

$$(b+1) = r_3; \qquad (b+2) = r_2; \qquad (b+3) = r_1.$$

В ячейке $a + k + 1$ теперь стоит команда возврата, которая передает управление команде ($a + k + 4$). Команды ($a + k + 4$), ($a + k + 5$), ($a + k + 6$), ($a + k + 7$) и ($a + k + 10$) осуществляют сравнение $r_3$ с $r_2$ и с $r_1$. Если $r_3$ совпадет хотя бы с одним из чисел $r_2$, $r_1$, то результат третьего счета считается правильным и управление передается команде $a + k + 11$, начинающей новый участок счета. При несовпадении $r_3$ ни с $r_2$, ни с $r_1$ происходит останов машины.

Нужно иметь в виду, что команды программы являются числами и хранятся в памяти так же, как числа. Поэтому при сбое в работе машины могут оказаться искаженными не только числа, но и команды. Недостаток контроля с помощью двойного-тройного счета состоит в том, что в случае порчи программы в течение первого счета результаты второго и третьего счетов могут быть неправильными, но совпадающими между собой. В этом случае они будут расценены как правильные. Двойной-тройной счет нужно сопровождать проверкой сохранности программы.

Контроль правильности работы машины считается в настоящее время совершенно необходимым. В большинстве современных программ применяется либо двойной, либо двойной-тройной счет. Это усложняет логические схемы программы, увеличивает, по крайней мере, вдвое машинное время, необходимое для решения задачи, значительно увеличивает средства, расходуемые на решение задачи. Но с этими недостатками приходится пока что мириться ввиду того, что в настоящее время еще не разработаны более дешевые и удобные универсальные методы проверки правильности результатов машинного счета.

Проверку сохранности программы предусматривают независимо от того, применяется ли двойной-тройной контрольный счет или нет. Эту проверку осуществляют путем контрольного суммирования программы и сличения контрольной суммы с контрольной суммой, запасенной заранее. Эта проверка должна производиться на тех этапах работы программы, на которых программа принимает свой первоначальный вид.

Очень удобен при решении длинных задач следующий прием, обеспечивающий, кроме проверки правильности вычислений, также проверку сохранности программы и периодическое ее обновление в памяти машины. Программу строят так, чтобы после выполнения каждой определенной части вычислений машина выводила все содержимое памяти на магнитную ленту и повторяла этап вычислений вторично. Если содержимое памяти после вторичного счета совпадает с записанным на ленте, то записанное на ленте вводится в память и машина переходит к следующей части вычислений.

**4. Контроль правильности вычислений.** Для некоторых задач удается вместо двойного счета применить какой-либо другой способ контроля, например контроль правильности вычислений (а не работы машины). Правильность вычислений в качестве своего составного элемента' содержит правильность работы машины. Но, помимо этого, контроль правильности вычислений является дополнительной проверкой правильности подготовки задачи к решению. При этом могут выявиться, например, принципиальные ошибки, необнаруженные при проверке программы на бланках и при отладке. Таким образом, этот вид контроля является более действенным, но не всегда может быть применен.

Правильность вычислений проверяется, например, одним из следующих способов.

1. Если получаемые в процессе решения задачи результаты должны удовлетворять какому-нибудь соотношению, известному заранее и не используемому для получения этих результатов, то через определенные промежутки вычислительного процесса производится (предусмотренная в программе) проверка, удовлетворяют ли получаемые результаты с достаточной степенью точности такому контрольному соотношению. Например, при вычислении таблицы значений синусов и косинусов можно производить проверку, удовлетворяют ли получаемые машиной числа соотношению

$$\sin^2 \alpha + \cos^2 \alpha = 1.$$

Проверка такого соотношения контролирует как правильность работы машины, так и правильность выполняемой ею программы.

Применяя проверку контрольного соотношения, нужно не забывать, что машина производит вычисления с большой точностью, но не абсолютно точно, так что контрольное соотношение, имеющее вид равенства, даже при правильных результатах практически никогда не будет выполнено. Контрольным соотношениям нужно придавать вид неравенств. Например, последнее соотношение следует представить так:

$$|\sin^2 \alpha + \cos^2 \alpha - 1| < \varepsilon,$$

где ε— достаточно малое положительное число.

2. В некоторых случаях удается ввести вспомогательную контрольную величину, связь которой с результатами, получаемыми при решении задачи, заранее известна. Контрольная величина вычисляется одновременно с другими величинами, являющимися в решаемой задаче искомыми. Через определенные отрезки вычислительного процесса производится (предусмотренная в программе) проверка выполнения связи между искомыми величинами задачи и контрольной величиной. Например, интегрируя систему дифференциальных уравнений

$$\left.\begin{aligned} x'_t &= ax + by + ct + d, \\ y'_t &= mx + ny + pt + q \end{aligned}\right\} \qquad \text{(VIII.1)}$$

при начальных данных

$$\left.\begin{aligned} y|_{t=0} &= y_0 \\ x|_{t=0} &= x_0 \end{aligned}\right\} \qquad \text{(VIII.2)}$$

мы можем ввести дополнительное дифференциальное уравнение, складывая между собой заданные уравнения и вводя новую функцию

$$x+y = u,$$

$$(x+y)'_t = (a+m)x + (b+n)y + (c+p)t + (d+q) =$$

$$= (a+m)(x+y) + (b+n-a-m)y + (c+p)t + (d+q)$$

т. е.

$$u'_t = (a+m)u + (b+n-a-m)y + (c+p)t + (d+q)\,. \qquad \text{(VIII.3)}$$

В программе мы предусмотрим интегрирование уравнения (VII 1.3) при начальных условиях

$$u|_{t=0} = x_0 + y_0$$

совместно с системой (VIII.1) при начальных условиях (VIII.2) и периодическую проверку выполнения соотношения

$$u = x + y.$$

3. В некоторых случаях бывает целесообразным контроль правильности вычислений способом подстановки. Например, при решении системы алгебраических уравнений методом главных элементов или при решении системы дифференциальных уравнений методом Эйлера, Рунге—Кутта или Адамса можно предусмотреть в программе подстановку в эти уравнения получаемых численных значений искомых величин (и их производных, если уравнения дифференциальные) и оценку результатов подстановки. При этом требуется, чтобы абсолютные величины разностей между значениями левых частей уравнений и соответствующих им правых частей не превосходили достаточно малого положительного числа e, выбранного математиком заранее. Например, получая решение дифференциального уравнения.

$$y' = f(x, y)$$

в виде последовательности значений $y_1$, $y_2$, ..., $y_i$ ..., можно периодически каким-либо методом численного дифференцирования вычислять $y'_i$ и проверять выполнение неравенства

$$|\, y'_i - f(x_i, y_i)\,| < \varepsilon\,.$$

Мы привели здесь лишь небольшое число наиболее известных способов программного контроля правильности работы машины и вычислений. Проблема разработки программных методов контроля, не связанных с осуществлением повторного счета (требующего большого расхода машинного времени), сейчас весьма актуальна.

## § 41. Организация программы

**1. Основные понятия и обозначения.** Говоря об *организации программы,* мы имеем в виду совокупность крупных функционально различных частей этой программы и взаимодействие между ними.

Большинство программ, приведенных выше в качестве примеров, имеют простейшую организацию. Они состоят из трех функционально различных частей, действующих последовательно: из ввода программы и исходных данных, рабочего счета и выдачи результатов. Обозначая символом **V** ввод, символом **R** рабочий счет и символом **W"** выдачу результатов, можно описать организацию таких программ следующей схемой:

**VRW".**

Как правило, организация программ бывает значительно сложнее. Например, если в программе предусмотрен контроль с помощью двойного счета, схема организации программы может быть следующей:

$$\mathbf{VR\Sigma_1 E(R)\Sigma_2 P(\sigma_1, \sigma_2)W''Я.}$$

Здесь **V, R** и **W"** имеют прежнее значение. Остальные символы означают: $\Sigma_1$— первое контрольное суммирование, $\Sigma_2$— второе контрольное суммирование, $\mathbf{P(\alpha_1, \alpha_2)}$—сравнение контрольных сумм, **Я** — останов при их несовпадении, **E(R)** — обращение к **R**, т. е. повторение **R**.

Перечислим основные элементы организации и приведем стандартные символы для их обозначения:

**V, V′, V″** — ввод соответственно программы и исходных данных, только программы, только исходных данных;

$\mathbf{L}_i$, $\mathbf{L}'_i$, $\mathbf{L}_i$"— запись на магнитную ленту (в зону номер *i*) соответственно программы и числового материала, только программы, только числового материала;

$\mathbf{Л}_i$- — считывание из зоны номер *i* магнитной ленты.

Σ, Σ', Σ" — контрольное суммирование соответственно программы и числового материала, только программы, только результатов; контрольные суммы обозначаются символами σ, σ ', σ ";.

**R** — рабочий счет;

**P**($\sigma_1$, $\sigma_2$) — логический оператор, производящий сравнение контрольных сумм $\sigma_1$ и $\sigma_2$; при $\sigma_1 \neq \sigma_2$ управление передается по верхней стрелке;

**P**(*i*<*n*) — логический оператор, проверяющий условие *i*<*n,* где *i* — переменный параметр, *n* — число; он же увеличивает на единицу значение параметра *i*; при $i < n$ управление передается по верхней стрелке;

**П**, **П′**, **П"** — перенос из одной части оперативной памяти в другую соответственно программы и чисел, только

программы, только чисел;

**Ф** (*i*—*j*) — замена индексов *i* на *j* и *j* на *i* при символах **L** и **Л**;

**W, W′, W"** — выдача на перфокарты соответственно программы и результатов, только программы, только результатов;

**Е** — обращение к некоторой части программы;

**Я** — останов машины.

Приведенные символы могут быть снабжены переменными или постоянными индексами. Например, знаки $\mathbf{R}_1$ и $\mathbf{R}_2$ могут обозначать две части вычислительного процесса, знаки $\mathbf{L}_1$ и $\mathbf{L}_2$ могут обозначать запись в разные зоны магнитной ленты. Символы $\Sigma_1$ и $\Sigma_2$, означающие первое и второе контрольное суммирование, мы уже встречали в схеме организации программы, предусматривающей контроль правильности вычислений способом двойного счета.

Введенная система символов, обозначающих элементы организации, позволяет легко читать и без пояснений понимать схемы организации программ.

**2. Примеры схем организации программ.** Организация применяемых на практике программ бывает обычно довольно сложной. Составление логической схемы программы обычно начинают с продумывания и составления схемы организации программы. Составление логической схемы программы без предварительного составления схемы организации допустимо лишь при простейших организациях.

Приведем несколько примеров схем организации программы.

1. Программа, предусматривающая контроль правильности вычислений с помощью двойного-тройного счета, может иметь организацию, изображаемую следующей схемой:

$$\mathbf{VR}\Sigma_1''\mathbf{E(R)}\Sigma_2''\mathbf{P}\,(\sigma_1'',\ \sigma_2'')\,\mathbf{E(R)}\Sigma_3''\mathbf{P}\,(\sigma_3'',\ \sigma_1'')\,\mathbf{P}\,(\sigma_3'',\ \sigma_2'')\,\mathbf{W''Я}.$$

Просматривая эту схему, без труда читаем: ввод программы и исходных данных (**V**); рабочий счет (**R**); первое контрольное суммирование результатов ($\Sigma_1$"); повторение рабочего счета (**E(R)**); второе контрольное суммирование результатов ($\Sigma_2$"); сравнение контрольных сумм $\sigma_1$" и $\sigma_2$" (**P**($\sigma_1$", $\sigma_2$")); при совпадении $\sigma_1$" и $\sigma_2$"— выдача результатов на перфокарты (**W**"); при несовпадении — новое повторение рабочего счета (**E(R)**) с последующим контрольным суммированием его результатов ($\Sigma_3$"); затем сравнение $\sigma_3$" и $\sigma_1$" (P($\sigma_3$", $\sigma_1$")); при совпадении $\sigma_3$" и $\sigma_1$" — выдача результатов (**W"**), при несовпадении — сравнение контрольных сумм $\sigma_3$" и $\sigma_2$" (P($\sigma_3$", $\sigma_2$")); при совпадении $\sigma_3$" и $\sigma_2$" —выдача результатов (**W"**), при несовпадении — останов (**Я**).

2. Составляя программу для решения большой серии однотипных задач (как говорят, вариантов задачи), иногда предусматривают возможность эпизодических перерывов в решении, во время которых программу «снимают с машины». Для того чтобы решение можно было возобновлять, начиная не с самого первого, а с первого еще нерешенного варианта, иногда применяют прием, называемый *способом передаточной перфокарты.* Сущность этого приема заключается в том, что вместе с результатами каждого варианта выдается дополнительная, так называемая *передаточная перфокарта.* Возобновляя решение после перерыва, передаточную перфокарту вводят вместе с программой и исходными данными. Передаточная перфокарта несет на себе числа, с помощью которых программа перестраивает себя для возобновления счета не с самого первого, а с очередного варианта. При вводе программы для решения вариантов, начиная с первого, в качестве передаточной обычно берут перфокарту, содержащую нули.

Организация описанной программы может быть следующей;

$$\mathbf{VV}_0'\mathbf{R}_i\Sigma_1''\mathbf{E}(\mathbf{R}_i)\Sigma_2''\mathbf{P}\,(\sigma_1'',\ \sigma_2'')\,\mathbf{W}_i''\mathbf{W}_i'\mathbf{P}\,(i<n)\,\mathbf{Я}.$$

Здесь $\mathbf{V'_0}$ — ввод начальной передаточной перфокарты. Для того чтобы начать счет с ($k$ + 1)-го варианта, нужно $\mathbf{V}_0'$ заменить на $\mathbf{V}_k'$. $\mathbf{W}_i'$ — выдача передаточной перфокарты.

3. Программа, в которой предусмотрен контроль правильности вычислений способом двойного счета с обновлением содержимого оперативной памяти, может иметь следующую организацию:

$$\mathbf{VL}_1\mathbf{R}_i\Sigma_1\mathbf{L}_2\mathbf{Л}_1\mathbf{E}(\mathbf{R}_i)\Sigma_2\mathbf{P}\,(\sigma_1,\ \sigma_2)\,\mathbf{Я};\ \mathbf{Л}_2\Phi\,(1-2)\,\mathbf{P}\,(i<n)\,\mathbf{W}''.$$

Прочитаем эту схему организации. Работа программы начинается с ввода себя и исходных данных (**V**). Затем программа и исходные данные записываются на первую зону магнитной ленты ($L_1$). Далее идет участок рабочего счета ($\mathbf{R}_{i=1}$). Результаты счета и программа суммируются ($\Sigma_1$) и записываются на вторую зону магнитной ленты ($\mathbf{L_2}$). Программа и исходные данные вызываются из первой зоны магнитной ленты ($\mathbf{Л_1}$), снова выполняется первый участок рабочего счета ($\mathbf{R}_{i:=i}$), осуществляется второе суммирование программы и результатов ($\Sigma_2$). Сравниваются суммы $\sigma_1$ и $\sigma_2$. Если они не совпали, происходит останов (**Я**); если совпали, то содержимое памяти обновляется путем считывания материала из второй зоны магнитной ленты ($\mathbf{Л}_2$). Затем индексы символов $\mathbf{L_2}$ $\mathbf{Л_1}$, $\mathbf{Л_2}$ заменяются: 1 на 2 и 2 на 1 (**Ф** (1 — 2)), так что схема принимает вид

$$\mathbf{VL}_1\mathbf{R}_i\Sigma_1\mathbf{L}_1\mathbf{Л}_2\mathbf{E}(\mathbf{R}_i)\Sigma_2\mathbf{P}\ \ (\sigma_1,\ \sigma_2)\,\mathbf{Я};\ \ \mathbf{Л}_1\Phi\,(1-2)\mathbf{P}\,(i<n)\,\mathbf{W}''.$$

После этого осуществляется изменение параметра (теперь $i$ =2) и переход (**P** ($i < n$)) к выполнению второго участка рабочего счета ($\mathbf{R}_{i=2}$). Работа программы продолжается до тех пор, пока не станет $i = n$, после чего результаты выдаются на перфокарты **(W").**

## § 42. Выбор численного метода

Применяя электронную цифровую машину для решения математических задач, необходимо учитывать ее особенности как вычислительного инструмента. Вот основные из них.

1. Большое количество цифр в изображении чисел, над которыми производятся арифметические действия. При этом в машине с плавающей запятой числа представлены постоянным количеством значащих цифр, или, как говорят, с одинаковой максимальной о т н о с и т е л ь н о й погрешностью. В машине с фиксированной запятой числа имеют одинаковое количество цифр после запятой, т. е. представлены, как говорят, с одинаковой максимальной а б с о л ю т н о й погрешностью.

2. Большая скорость выполнения операций над числами, хранящимися в оперативной памяти. При ручном счете нередко приходится отказываться от какого-нибудь численного метода, потому что, несмотря на свою принципиальную простоту, он требует вычислений с большим количеством значащих цифр и выполнения большого количества «шагов решения». Для машинного решения эти качества численного метода могут оказаться несущественными. Машина всегда ведет вычисления с большим количеством значащих цифр. Быстрота выполнения ею арифметических операций делает нередко допустимым увеличение количества «шагов решения».

3. Сравнительно малая скорость ввода исходных данных и программы, а также вывода результатов.

4. Сравнительно малая скорость обмена числами между оперативной памятью и накопителем, например магнитной лентой.

5. Ограниченная емкость оперативной памяти при практически безграничной емкости накопителя (например, накопителя на магнитных лентах).

6. Возможность случайных сбоев в работе машины и вытекающая отсюда необходимость контроля правильности получаемых результатов.

Выбирая численный метод, нельзя забывать о необходимости проверки правильности вычислений при решении задачи на машине. Широко распространенный способ контроля путем повторных расчетов сопровождается огромным расходом машинного времени. Более экономным может оказаться контроль путем проверки каких-либо заранее известных соотношений между вычисляемыми величинами.

Некоторые численные методы не требуют вообще контроля правильности вычислений. Таковы, например, итерационные методы. Достоинством этих методов является то обстоятельство, что получение ошибочного результата при одной из итераций не приводит к ухудшению окончательного результата вычислений, а лишь увеличивает количество итераций, которые должна произвести машина. В случае быстрой сходимости итерационного процесса затраты машинного времени при использовании итерационного метода могут быть сравнительно небольшими.

Выбирая численный метод, следует также учитывать особенности подготовки задачи для решения ее на машине:

1) необходимость «арифметизации» задачи, т. е. ее сведения к выполнению последовательности арифметических действий (практически можно использовать и другие элементарные действия, для которых уже составлены заранее стандартные подпрограммы);

2) трудоемкость процесса программирования;

3) необходимость отладки программы на машине.

**1. Погрешность вычислений.** Условия задачи, предназначенной для решений на электронной цифровой машине, всегда содержат указания о требуемой точности решения (обычно задается значение максимальной допустимой погрешности).

Только тот численный метод пригоден для решения задачи, который позволяет получить решение с погрешностью, не превышающей допустимую. Это обстоятельство требует соответствующего выбора численного метода из набора известных, а иногда и разработки нового метода. Сказанное относится и к тем численным методам, которые теоретически позволяют получать результат со сколь угодно высокой точностью путем увеличения числа шагов. Не следует забывать, что одновременно с увеличением числа шагов нужно увеличивать точность счета в каждом шаге, т. е. увеличивать количество цифр в представлении чисел, участвующих в вычислениях. Электронные цифровые машины считают с большим количеством цифр. Однако и при таком количестве цифр в результате вычислений получается погрешность (будем называть ее *арифметической*), которая, складываясь с погрешностью метода, сводит иногда на нет уменьшение последней.

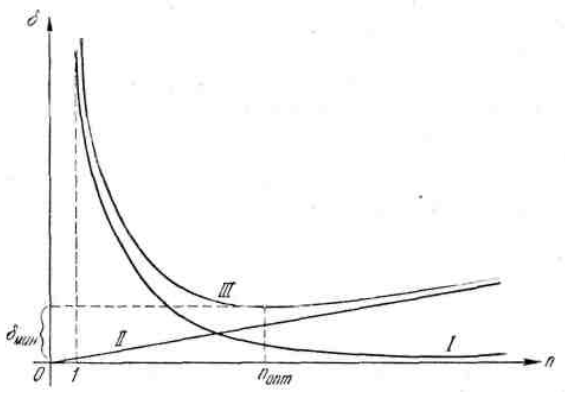



Рис. 108. Общая погрешность метода решения.

Например, в первом приближении можно считать, что арифметическая абсолютная погрешность при решении дифференциального уравнения методом Эйлера пропорциональна числу шагов, а абсолютная погрешность метода асимптотически стремится к нулю при увеличении числа шагов. При этом общая погрешность метода решения имеет некоторый минимум, отвечающий оптимальному числу шагов (рис. 108). Минимальная общая абсолютная

погрешность $\delta_{min}$ и оптимальное число шагов $n_{опт}$ характеризуют данный численный метод в применении к решению данной задачи на данной цифровой вычислительной машине. Ясно, что если допустимая по условию задачи абсолютная погрешность меньше, чем $\delta_{min}$, то рассматриваемый численный метод непригоден для решения данной задачи.

**2. Связность алгорифма.** Важной характеристикой метода является связность алгорифма, применяемого в этом методе. *Связностью алгорифма* называется количество чисел, которые нужно накапливать в запоминающих устройствах машины для перехода от одного этапа вычислений к другому. Если связность алгорифма велика, возникает необходимость переноса промежуточных результатов из памяти в накопитель и последующего ввода их из накопителя в память. Машинное время, расходуемое на решение задачи, при этом значительно возрастает.

Некоторые из известных численных методов позволяют путем незначительного изменения уменьшать связность своего алгорифма. Поясним это на простом примере.

Метод Симпсона для вычисления определенного интеграла состоит в применении приближенной формулы

$$\int_a^b y\,dx = \frac{h}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + \ldots + 2y_{2n-2} + 4y_{2n-1} + y_{2n}).$$

Для перехода от этапа вычислений величин $y_i$ к этапу вычисления $\int_a^b y\,dx$ требуется запомнить $2n + 1$ чисел $y_0$, $y_1$, …, $y_{2n}$. Связность алгорифма в этом случае равна $2n + 1$. Но вычисление по вышеприведенной полной формуле Симпсона можно заменить вычислениями по следующим формулам:

$$S_0 = 0;$$

$$\Delta S_i = \frac{h}{3}(y_{2i} + 4y_{2i+1} + y_{2i+2});$$

$$S_{i+1} = S_i + \Delta S_i;$$

$$i = 0, 1, 2, \ldots, n — 1.$$

Искомый интеграл получится при этом по формуле

$$\int_a^b y\,dx = S_n$$

При таком процессе для перехода от этапа вычислений величин $y_{2i}$, $y_{2i+1}$, $y_{2i+2}$ к этапу вычисления $S_{i+1}$ требуется запоминание лишь четырех чисел: $y_{2i}$, $y_{2i+1}$, $y_{2i+2}$, $S_i$. Связность логарифма оказывается равной четырем.

Конечно, не всякий численный метод допускает видоизменения, позволяющие уменьшить связность его алгорифма.

Для получения решения задачи с заданной степенью точности можно применять, различные численные методы. Так, для решения обыкновенных дифференциальных уравнений с начальными условиями разработано много численных методов, из которых наиболее известны методы Эйлера, Рунге—Кутта, Адамса.

**3. Учет стоимости и затрат времени.** Выбирая тот или иной метод из числа допустимых или разрабатывая новый, нужно стараться получить решение задачи с возможно меньшим расходом материальных средств и в кратчайшее время. Стоимость решения задачи и затраты времени определяются характером и объемом работы по ее подготовке и расходом машинного времени при ее решении. Приведем некоторые ориентировочные данные.

При решении задачи время расходуется на:

1) выбор численного метода или его разработку,
2) программирование,
3) отладку программы на машине,
4) решение задачи на машине по отлаженной программе.

Расход материальных средств можно примерно оценить, учитывая следующее:

1. Анализ задачи и выбор (в наиболее трудных случаях) или разработка нового численного метода осуществляются обычно высококвалифицированным математиком — старшим научным сотрудником, стоимость рабочего дня которого составляет около 100 руб.

2. Выбор численного метода в более легких случаях, составление логической схемы программы, большую часть программирования и отладку программы на машине производит младший научный сотрудник, рабочий день которого стоит около 50 руб.

3. Машинное время, расходуемое на решение задачи, стоит около 300 руб. за час.

4. Применение программирующей программы (см. главу X) значительно уменьшает расход времени на программирование и отладку программы на машине, всего на 8—10 минут увеличивая расход машинного времени (что зачастую полностью покрывается экономией машинного времени при отладке программы на машине). Таким образом, применение программирующей программы позволяет уменьшить как расход материальных средств на решение задачи, так и расход времени.

В тех случаях, когда решение задачи необходимо получить в кратчайший срок, а соображения экономии материальных средств отодвигаются на второй план, обычно приходится ограничиваться выбором численного метода из набора уже известных (разработка нового метода может потребовать большого, заранее неизвестного расхода времени). Нередко даже отдают предпочтение более простым и хорошо известным методам, требующим мало времени для программирования и отладки программы на машине, хотя бы при этом несколько увеличивался расход машинного времени. Если машинное время, расходуемое на решение задачи, во много раз меньше, чем время, необходимое для программирования, то может оказаться выгодным увеличение в несколько раз машинного

времени при параллельном сокращении в несколько раз времени программирования за счет применения принципиально более простых численных методов.

В том случае, когда быстрота получения результатов не является основным фактором, на первое место выдвигается требование минимального расхода средств. Пусть $T_1$ и $m_1$ — количество расходуемых часов и стоимость одного часа работы старшего научного сотрудника; $T_2$ и $m_2$ — то же для младшего научного сотрудника; $T_3$ и $m_3$ — количество расходуемых часов и стоимость одного часа машинного времени. Пусть, кроме того, $t_1$, $t_2$, $t_3$ и $t_4$ — соответственно время, расходуемое на выбор численного метода, программирование, отладку программы на машине и решение задачи на машине. Если, в случае срочных задач, при выборе численного метода приходилось стремиться к тому, чтобы минимальной была величина

$$T = t_1 + t_2 + t_3 + t_4,$$

то в последнем случае нужно добиваться получения минимума величины

$$M = T_1 m_1 + T_2 m_2 + T_3 m_3$$

Эти формулы приведены не в качестве расчетных, а лишь для пояснения основных принципов, которыми нужно руководствоваться при выборе численных методов.

Например, если требуется провести решение большого количества (многих сотен или тысяч) однотипных задач-вариантов, то становится весьма целесообразным особенно тщательно разработать численный метод и составить особенно тщательно программу для максимального возможного уменьшения расхода машинного времени на один вариант. Это было бы нецелесообразно при составлении программы для решения только одного варианта.

## § 43. Метод статистических испытаний (метод Монте-Карло)

К числу методов, не требующих специального контроля правильности вычислений, относится численный метод, применение которого стало возможным только после появления быстродействующих цифровых, вычислительных машин. Это *метод статистических испытаний,* или *метод Монте-Карло* (последнее название происходит от названия города Монте-Карло, знаменитого своими игорными домами, в которых, в частности, ведется игра на рулетке). Под этим названием объединяются методы решения математических или физических задач с помощью многократных случайных испытаний. Связь между некоторыми случайными процессами и решениями дифференциальных уравнений параболического типа в частных производных была установлена еще в 1899 г. Аналогичная связь между другой группой случайных процессов и решением задачи Дирихле для эллиптических разностных и дифференциальных уравнений в частных производных была исследована в 1928 г.

**1. Датчики случайных чисел.** Для применения метода Монте-Карло необходимо располагать каким-либо достаточно богатым и быстродействующим источником случайных чисел.

Применение таблиц случайных чисел не оправдывает себя, так как эти таблицы в случае достаточной их обширности приходится хранить не в оперативной памяти, а в накопителе машины. Выборка их из накопителя связана с большой потерей машинного времени, не говоря уже о трудности подготовки таких таблиц для ввода в запоминающие устройства машины.

Удовлетворительных результатов иногда достигают, применяя так называемые *псевдослучайные* числа, т. е. числа, получаемые с помощью какого-нибудь специального алгорифма, однако обладающие тем свойством, что их последовательность по своему характеру близка к некоторой последовательности случайных чисел.

Однако самые лучшие результаты, как в смысле точности получаемых ответов, так и в смысле экономии машинного времени достигаются с помощью специальных датчиков случайных чисел, подключаемых к машине, решающей задачу методом Монте-Карло. Вырабатываемые датчиком случайные числа вводятся в отведенное для этой цели место оперативной памяти машины.

Действие таких датчиков основано на использовании каких-либо физических случайных процессов, например, радиоактивного распада, шумов в электронных лампах и т. п.

Поток случайных чисел, производимых датчиком, характеризуется, во-первых, диапазоном этих чисел и, во-вторых, законом распределения, которому этот поток подчинен.

Пусть $\xi$ означает случайное число. *Диапазон* потока случайных чисел задается в виде неравенства

$$a \le \xi < b \qquad \text{(VIII.4)}$$

*Законом распределения,* которому подчинен поток случайных чисел, называют такую функцию

$$\Phi = \Phi(\xi), \qquad \text{(VIII.5)}$$

которая определена на промежутке (VIII.4) и обладает следующим свойством: при любых, $\alpha$, $\beta$, удовлетворяющих неравенству $a \le \alpha \le \beta \le b$, имеет место

$$p(\alpha \le \xi \le \beta) = \int_{\alpha}^{\beta} \Phi(\xi) d\xi, \qquad \text{(VIII.6)}$$

где $p(A)$ означает вероятность выполнения условия $A$, которое написано внутри скобок этого символа, т. е. в нашем случае вероятность того, что очередное $\xi$ число будет удовлетворять неравенству $\alpha \le \xi \le \beta$.

Применяя датчик случайных чисел, мы всегда знаем как диапазон, так и закон распределения, характеризующий даваемый датчиком поток случайных чисел.

**2. Неравенство Чебышева.** Для метода Монте-Карло фундаментальное значение имеет известное неравенство Чебышева:

$$P\{|\nu - p| < \varepsilon\} \ge 1 - \frac{p(1-p)}{n\varepsilon^2}, \qquad \text{(VIII.7)}$$

где $\nu$ — частость получения некоторого события, $p$ — его вероятность, $n$ — количество проведенных испытаний, $\varepsilon$—

малое положительное число. Частость события равна

$$\nu = \frac{m}{n} \qquad \text{(VIII.8)}$$

где $m$ — количество тех испытаний, при которых имело место интересующее нас событие (или, как говорят, $m$ — количество благоприятных случаев).

Из неравенства Чебышева следует, что вероятность того, что

$$|\nu - p| < \varepsilon$$

тем ближе к единице, чем большее число испытаний было проведено. Иными словами, частость можно принимать за приближенное (с точностью до $\varepsilon$) значение вероятности с тем большей уверенностью, чем больше было проведено испытаний для определения частости. Вероятность $P\{|\nu - p| < \varepsilon\}$ обычно называют *надежностью.*

Способ определения числа $n$, при котором надежность больше заданного значения $\delta$, мы получаем опять-таки из неравенства Чебышева*. Легко подсчитать, что величина $p$ $(1 - p)$ для $0 \le p \le 1$ имеет максимум при $p = 1/2$. Значит, правая часть (VIII.7) удовлетворяет неравенству

$$1 - \frac{p(1-p)}{n\varepsilon^2} \ge 1 - \frac{\frac{1}{2}\left(1 - \frac{1}{2}\right)}{n\varepsilon^2} = 1 - \frac{1}{4n\varepsilon^2} \qquad \text{(VIII.9)}$$

Для определения $n$, дающего надежность, превосходящую заданное значение $\delta$, полагаем:

$$1 - \frac{1}{4n\varepsilon^2} \ge \delta \quad , \qquad \text{(VIII. 10)}$$

откуда получаем:

$$n \ge \frac{1}{4(1-\delta)\varepsilon^2}. \qquad \text{(VIII.10)}$$

Поясним сущность метода Монте-Карло на примере нескольких несложных задач.

**3. Вычисление интеграла с помощью специально подобранного датчика.** Требуется вычислить определенный интеграл

$$\int_{\alpha}^{\beta} f(x)dx \qquad (\alpha<\beta),$$

причем имеется датчик случайных чисел, дающий поток случайных чисел $\xi$ с диапазоном

$$a \le \xi < b$$

и законом распределения $f(\xi)$, совпадающим с подынтегральной функцией. Кроме того, предположим, что $\beta < b$. В силу формулы (VIII.6) имеем:

$$\int_{\alpha}^{\beta} f(x)dx = \int_{\alpha}^{\beta} f(\xi)d\xi = p(\varepsilon \le \beta).$$

В силу неравенства Чебышева (VIII.7) можно считать, что

$$\int_{\alpha}^{\beta} f(x)dx \approx \nu,$$

где $\nu$ — частость получения события, состоящего в том, что $\xi \le \beta$.

Для вычисления рассматриваемого интеграла с нужной степенью точности достаточно заставить машину проверять, выполнено ли неравенство $\xi \le \beta$ для случайных чисел, задаваемых датчиком, и подсчитывать общее количество проверенных чисел ($n$) и количество случаев, в которых неравенство было выполнено ($m$). При достаточно большом $n$ мы с большой уверенностью можем полученное $\nu = \frac{m}{n}$ считать значением искомого интеграла.

Подчеркнем, что, решив задачу методом Монте-Карло, мы не можем гарантировать, что нами получен правильный ответ, мы лишь имеем достаточно большую степень уверенности в этом. На первых порах это обстоятельство может смутить. Однако, поразмыслив, мы заметим, что в большинстве случаев, если не всегда, гарантией мы считаем именно высокую степень уверенности.

**4. Вычисление простых и кратных интегралов с помощью стандартного датчика.** Совершенно ясно, что нельзя для каждой решаемой методом Монте-Карло задачи строить свой датчик случайных чисел. Обычно для решения самых разнообразных задач применяют один **и** тот же датчик. Чаще всего применяются датчики с диапазоном

$$0 \le \xi \le 1$$

* Формула Чебышева дает для $n$ сильно завышенное значение. Практически удобнее для определения $n$ пользоваться формулой Муавра-Лапласа.

и так называемым равномерным законом распределения случайных чисел, который для указанного диапазона имеет вид

$$\Phi=1.$$

Покажем, как можно с помощью такого датчика найти

$$I=\int_{\alpha}^{\beta} f(x)dx$$

Пусть $M$ и $N$ — соответственно наибольшее и наименьшее значения функции $f(x)$ на интервале $a$, β. Делая в интеграле замену переменного по формуле

$$x=(\beta—a)z+a,$$

с помощью элементарных преобразований получаем:

$$I=(\beta-a)\left[(M-N)\int_0^1 \frac{f[(\beta-a)z+a]-N}{M-N}dz+N\right]$$

Полагая

$$\frac{f[(\beta-a)z+a]-N}{M-N}=\varphi(z),$$

мы видим, что задача вычисления интеграла $I$ свелась к вычислению интеграла

$$I_1=\int_0^1 \varphi(z)dz,$$

причем на интервале интегрирования имеют место неравенства

$$0\le\varphi(z)\le 1$$

Достаточно показать применение метода Монте-Карло для вычисления интеграла $I_1$.

Из определения закона распределения вытекает, что если для потока чисел, даваемого датчиком, Ф(ξ) является законом распределения, то и для потока, который мы получим, беря числа, даваемые датчиком не подряд, а через одно, или через два, или через $k$ — 1 чисел, закон распределения будет тем же самым. Поэтому можно в случае необходимости считать, что датчик производит не один, а $k$ потоков чисел.

Будем считать, что датчик с равномерным законом распределения производит два потока случайных чисел:

$$\xi_1, \xi_2, \ldots, \xi_i, \ldots$$
$$\eta_1, \eta_2, \ldots, \eta_i, \ldots$$

В силу (VIII.6)

$$\left.\begin{aligned} p(\alpha\le\xi\le\beta)&=\int_{\alpha}^{\beta} d\xi,\\ p(\gamma\le\eta\le\delta)&=\int_{\gamma}^{\delta} d\eta. \end{aligned}\right\} \qquad \text{(VIII.11)}$$

Вероятность того, что точка $A(\xi, \eta)$ попадает в квадрат $q$

$$\alpha\le\xi\le\beta,$$
$$\gamma\le\eta\le\delta,$$

площадью $S_q$, будет

$$p(A\subset q)=p(\alpha\le\xi\le\beta)p(\gamma\le\eta\le\delta) \qquad \text{(VIII.12)}$$

(здесь знак $\subset$ означает слово «принадлежит»), так как вероятность наступления одновременно двух независимых между собой событий равна произведению их вероятностей. В силу (VIII.11) и (VIII.12) имеем:

$$p(A\subset q)=\int_{\alpha}^{\beta} d\xi\int_{\gamma}^{\delta} d\eta=\int_{\alpha}^{\beta}\int_{\gamma}^{\delta} d\xi d\eta=S_q. \qquad \text{(VIII. 13)}$$

Как известно, интеграл

$$\int_0^1 \varphi(z)dz=\int_0^1 \varphi(\xi)d\xi$$

численно равен площади области $Q$, ограниченной линиями η = 0, ξ = 1, η = φ(ξ), ξ= 0. Разбивая эту область на малые квадраты $q_i$ с площадями $S_{q_i}$, имеем:

$$\int_0^1 \varphi(z)dz \approx \sum_i S_{q_i} = \sum_i p(A \subset q_i).$$

Заставляя стороны квадратов $q_i$ стремиться к нулю, а их количество неограниченно возрастать, в пределе получим:

$$\int_0^1 \varphi(z)dz = p(A \subset Q) \qquad \text{(VIII. 14)}$$

Но точка $A$ (ξ, η) принадлежит области $Q$ в том случае, если

$$\eta \le \varphi(\xi).$$

Поэтому формуле (VIII. 14) можно придать такой вид:

$$\int \varphi(z)dz = p[\eta \le \varphi(\xi)] \qquad \text{(VIII.15)}$$

Последняя формула позволяет для вычисления рассматриваемого интеграла применить метод Монте-Карло. Беря по два последовательных числа, даваемых датчиком случайных чисел, будем первое из них считать за ξ, а второе — за η. Пусть $n$ — общее количество рассмотренных пар случайных чисел, а $m$ — количество пар, удовлетворяющих условию η < φ (ξ). Считая $\nu = \frac{m}{n}$ приближенным значением вероятности

$$p[\eta \le \varphi(\xi)]$$

получаем:

$$\int_0^1 \varphi(z)dz \approx \nu$$

Легко видеть, что в случае $k$-кратного интеграла справедлива формула

$$\int_0^1 \dots \int_0^1 \varphi(z_1, z_2, \dots, z_k)dz_1 dz_2 \dots dz_k = p[\eta \le \varphi(\xi_1, \xi_2, \dots, \xi_k)]$$

В этом случае каждое испытание требует получения $k + 1$ последовательных случайных чисел, из которых $k$ первых мы считаем соответственно за $\xi_1$, $\xi_2$, ..., $\xi_k$, а последнее — за η.

Так как неравенство Чебышева (VIII.7) справедливо для частости и вероятности любого случайного события, то для получения ответа с одной и той же точностью ε и надежностью δ при вычислении $k$-кратного интеграла требуется столько же испытаний, сколько и при вычислении однократного интеграла. При этом каждое испытание требует $k + 1$ случайных чисел вместо двух.

Говорят, что количество потребных случайных чисел с увеличением кратности интеграла возрастает пропорционально его кратности (увеличенной на единицу).

При вычислении кратных интегралов методом прямоугольников, методом трапеций или по формуле Симпсона количество узловых точек равно $N^k$, где $N$ — количество точек разбиения каждого промежутка

$$0 \le z_j \le 1 \qquad (j=1, 2,..., k).$$

Следовательно, количество узловых точек растет вместе с кратностью интеграла по экспоненциальному закону

$$n = N^k = e^{k\ln N}.$$

Приведенное сопоставление позволяет понять, что метод Монте-Карло может быть эффективно использован для вычисления интегралов высокой кратности в тех случаях, когда другие численные методы становятся неосуществимыми из-за огромного количества необходимых вычислений и вытекающей из этого необходимости огромного расхода машинного времени.

**5. Вычисление значений функции по ее обратной функции и решение уравнений.** В силу (VIII.6) для потока случайных чисел с равномерным законом распределения и диапазоном случайных чисел

$$a \le \xi \le a + 1$$

имеет место равенство

$$p(\xi \le \beta) = \int_\alpha^\beta d\xi$$

где $a \le \beta \le a+1$.

Пусть $\xi = f(\zeta)$ — монотонно возрастающая дифференцируемая функция, а $\zeta = \theta(\xi)$ —обратная ей функция. Тогда с помощью замены переменных получим:

$$p(\varepsilon \le \beta) = \int_{\theta(\alpha)}^{\theta(\beta)} f'(\zeta)d\zeta$$

Очевидно, $p(\xi \le \beta) = p[\theta(\xi) \le \theta(\beta)]$. Поэтому можно написать:

$$p[\theta(\xi) \le \theta(\beta)] = \int_{\theta(\alpha)}^{\theta(\beta)} f'(\zeta)d\zeta \qquad \text{(VIII. 16)}$$

Заменяя в последней формуле θ(β) на *x* и выполняя интегрирование, получим:

$$p[\theta(\xi) \le x] = \int_{\theta(\alpha)}^{x} f'(\zeta)d\zeta = f(x) - f[\theta(a)]$$

Очевидно, $f[\theta(a)] = a$, поэтому имеем:

$$p[\theta(\xi) \le x] = f(x) - a$$

или

$$f(x) = a + p[\theta(\xi) \le x] \qquad \text{(VIII. 17)}$$

Пользуясь формулой (VIII.17), нужно не забывать, что она справедлива не для любого *x,* а лишь для такого, которое удовлетворяет условиям

$$\theta(a) \le x \le \theta(a+1),$$

ибо *x* есть не что иное, как θ(β), которое в силу монотонного возрастания $f(\xi)$ и в силу того, что $a \le \beta \le a + 1$, подчиняется неравенствам

$$\theta(a) \le \theta(\beta) \le \theta(a+1).$$

Полученная формула может быть использована для вычисления методом Монте-Карло значений функции.

П р и м е р 1. Если $f(x) = \sqrt[k]{x}$, то $\theta(\xi) = \xi^k$, поэтому.

$$\sqrt[k]{x} = a + p[\xi^2 \le x]$$

где $a^k \le, x \le (a+1)^k$,т. е. в качестве *a* следует взять целую часть корня.

П р и м е р 2. Если $f(x) = \arcsin x$, то $\theta(\xi) = \sin\xi$ и, следовательно,

$$\arcsin x = p\,[\sin\xi \le x]$$

в промежутке от 0 до sin 1.

Если бы $f(\xi)$ была монотонно убывающей функцией, мы вместо (VIII. 17) имели бы:

$$f(x) = a + p[\theta(\xi) \ge x] \qquad \text{(VIII. 18)}$$

при $\theta(a) \ge x \ge \theta(a+1)$.

Формулы (VIII.17) и **(VIII.**18) можно применять для приближенного решения уравнений. Пусть, например, дано уравнение

$$F(y) = 0 \qquad \text{(VIII. 19)}$$

и известно, что корень этого уравнения заключен в интервале (α, β) таком, что

$$a \le \alpha < \beta \le a + 1$$

причем $F(y)$ монотонно возрастает в интервале $(a, a+1)$. Обозначим

$$F(y) = x,$$

и пусть обратная функция для $F(y)$ будет:

$$y = f(x).$$

В силу (VIII. 17) можем написать:

$$f(x) = p[F(\xi) \le x] + a.$$

Полагая в последней формуле $x = 0$, в качестве $f(0)$ мы получим $y^*$ — искомый корень уравнения (VIII.19), поэтому

$$y^* = a + P[F(\xi) \le 0] \qquad \text{(VIII.20)}$$

Если бы $F(y)$ вблизи корня была монотонно убывающей функцией, то вместо (VI11.20) мы получили бы:

$$y^* = a + P[F(\xi) \ge 0] \qquad \text{(VIII.21)}$$

**6. Преобразование потока случайных чисел.** Рассмотрим формулу (VIII. 16)

$$p[\theta(\xi) \le \theta(\beta)] = \int_{\theta(\alpha)}^{\theta(\beta)} f'(\zeta)d\zeta,$$

где $f(\zeta)$ — монотонно возрастающая функция (т. е. $f'(\zeta) \ge 0$); очевидно, в ней $\theta(\xi) = \zeta$.

Полагая $\theta(a) = a'$, $\theta(\beta) = \beta'$ получим:

$$p[\zeta \le \beta'] = \int_{\alpha_1}^{\beta'} f'(\zeta)d\zeta, \qquad \text{(VIII.22)}$$

Последняя формула справедлива при любом β', удовлетворяющем неравенству $a' \le \beta' \le b'$, где $b' = \theta(a+1)$. Следовательно, будет справедливо также равенство

$$p[\zeta \le \alpha'] = \int_{a'}^{\alpha'} f'(\zeta) d\zeta \qquad \text{(VIII.23)}$$

Вычитая (VIII.23) из (VIII.22) и учитывая, что

$$p[\zeta \le \beta'] - p[\zeta \le \alpha'] = p[\alpha' \le \zeta \le \beta'],$$

получим:

$$p(\alpha' \le \zeta \le \beta') = \int_{\alpha'}^{\beta'} f'(\zeta) d\zeta, \qquad \text{(VIII.24)}$$

коль скоро

$$a' \le \alpha' \le \beta' \le b'$$

Полученный результат убеждает нас в справедливости следующей теоремы:

Т е о р е м а . *Если поток чисел* $\xi$ *подчинен равномерному закону распределения и имеет диапазон* $a \le \xi \le 1$, *то поток чисел*

$$\zeta = f(\xi),$$

*где* $f$ *— возрастающая функция, подчинен закону распределения* $f'(\zeta)$.

Эта теорема позволяет, имея датчик с равномерным законом распределения, получать поток чисел, подчиненный нужному нам закону распределения.

Если датчик поставляет нам поток случайных чисел с диапазоном

$$0 \le \bar{\xi} \le 1$$

то для решения уравнения

$$F(y) = 0$$

формуле (VI 11.20) с помощью последней теоремы придаем вид

$$y^* = a + p[F(a + \bar{\xi}) \le 0 \qquad \text{(V1II.25)}$$

Если какой-либо закон распределения $f'(\zeta)$ особенно часто приходится применять, то можно изготовить специальный переходной блок, включаемый между датчиком случайных чисел и электронной цифровой машиной. Этот блок будет, получая числа $\xi$, преобразовывать их в числа $\zeta = f(\xi)$ и вводить последние в машину. В частности, очень удобен переходной блок, представляющий собой запоминающее устройство, постоянно хранящее в ячейках с номерами $\xi_i$ числа $\zeta_i$. Если датчик производит число $\xi_i$, то это число воспринимается переходным блоком как сигнал для выдачи в машину из ячейки номер $\xi_i$ числа $\zeta_i$.

**7. Теоретико-вероятностное моделирование.** Интересной особенностью метода Монте-Карло является возможность применения его для решения задач, не сформулированных в виде уравнений или формул, если эти задачи допускают теоретико-вероятностную «модель», осуществляемую с помощью одного или нескольких потоков случайных чисел.

Например, представим себе такую задачу. Положение цели точно известно. Цель считается пораженной, если снаряд падает от нее на расстоянии, не превосходящем ρ. Известно, что случайные отклонения $\delta_x$ и $\delta_y$ координат точки падения снаряда от цели подчинены соответственно законам распределения $f'(\delta_x)$ и $f'(\delta_y)$ и имеют диапазоны $f(0) \le \delta(x) \le f(1)$ и $\varphi(0) \le \delta(y) \le \varphi(1)$. Требуется определить вероятность поражения цели одиночным выстрелом. Будем считать, что цель помещена в начале координат. Выстрелом будем считать появление двух последовательных случайных чисел, производимых датчиком с равномерным законом распределения и диапазоном (0,1). Обозначим первое из чисел через $\xi$, а второе — через η. Вычисляя $f(\xi)$ и $\varphi(\eta)$, будем считать эти числа значениями случайных отклонений точки падения снаряда от цели. Если будет получаться, что $f^2(\xi) + \varphi^2(\eta) \le \rho^2$, то, очевидно, снаряд упал на расстоянии от цели, которое не превосходит ρ, и поразил цель. Следовательно, искомая вероятность поражения цели одиночным выстрелом равна вероятности

$$p[f^2(\xi) + \varphi^2(\eta) \le \rho^2]$$

Для решения задачи нужно произвести $n$ испытаний (где $n$ определено с таким расчетом, чтобы обеспечить нужную точность ε с достаточной надежностью), каждое из которых состоит в получении двух случайных чисел и в проверке выполнения неравенства

$$f^2(\xi) + \varphi^2(\eta) \le \rho^2.$$

Частость получения такого неравенства дает нам приближенное значение вероятности поражения цели одиночным выстрелом. Такой же результат можно получить с помощью осуществления $n$ испытаний, каждое из которых представляет собой выстрел из орудия по цели. Ясно, что решение описанной задачи методом Монте-Карло на цифровой электронной машине в материальном отношении значительно выгоднее испытаний, проводимых в натуре.

**8. Погрешности при решении задач.** Мы знаем, что при решении задачи любым численным методом результат получается с погрешностью, которая слагается из погрешности метода и арифметической погрешности. Это справедливо и для метода Монте-Карло. Для него погрешность можно оценить с помощью приведенной выше формулы Чебышева. Арифметическая погрешность при этом методе возникает за счет использования не любых случайных чисел, принадлежащих диапазону $(a, a+1)$, а лишь чисел, имеющих разрядность, обусловленную

разрядностью ячеек памяти применяемой электронной цифровой машины. Чтобы уяснить природу погрешности, связанной с разрядностью случайных чисел, рассмотрим решение методом Монте-Карло задачи о вычислении интеграла

$$I = \int_0^1 f(x)dx ,$$

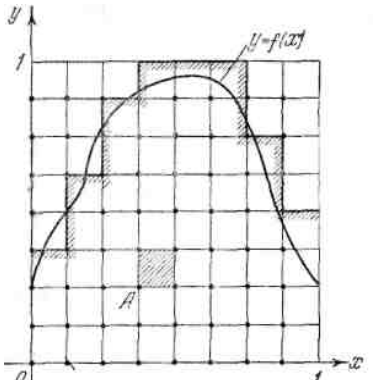


Рис. 109. Фигура, площадь которой при методе Монте-Карло вычисляется вместо $\int_0^1 f(x)dx$

где $0 \le f(x) \le 1$, при диапазоне случайных чисел $0 \le \xi < 1$ и равномерном законе их распределения.

Предположим, что используемые случайные числа являются правильными двоичными дробями, содержащими каждая $k$ знаков после запятой. Такие случайные числа будем называть *возможными.*

Если $\xi_B$, $\gamma_B$ — пара последовательно полученных возможных случайных чисел, то в силу формулы (VIII. 15) для вычисления интеграла $I$ имеем:

$$\int_0^1 f(x)dx = p(\eta_B < f(\xi_B)) \qquad \text{(VIII.26)}$$

Каждой паре возможных случайных чисел $\xi_B$, $\gamma_B$ отвечает случайная точка в квадрате $0 \le x < 1$, $0 \le y < 1$. Проведем в этом квадрате прямые

$$x = a_i , \quad y = a_i ,$$

где $a_0 = 0$; $a_{i+1} = a_i + 2^{-k}$, $i = 1, 2, 3, \ldots$ При этом упомянутый квадрат окажется разбитым на элементарные квадраты со сторонами, равными $2^{-k}$ Случайные точки $(\xi_B, \gamma_B)$ могут совпадать лишь с вершинами элементарных квадратов. Вершины элементарных квадратов, не лежащие на прямых $x = 1$ или $y = 1$, будем называть *допустимыми точками* (рис. 109).

Обозначим через $N$ количество всех допустимых точек (а значит, и всех элементарных квадратов), а через $N^*$ количество допустимых точек, расположенных ниже кривой $y = f(x)$. Легко сообразить, что

$$p(\eta_B < f(\xi_B)) = \frac{N^*}{N} .$$

Таким образом, желая вычислить интеграл $I$ методом Монте-Карло, мы будем в действительности вычислять приближенное значение отношения $\frac{N^*}{N}$. Арифметическая погрешность $\delta$ будет при этом, очевидно, равна

$$\delta = \left| I - \frac{N^*}{N} \right| .$$

Поставим каждой допустимой точке в соответствие тот элементарный квадрат, для которого она является левой нижней вершиной (на рис. 109 допустимой точке $A$ отвечает заштрихованный элементарный квадрат). Площадь каждого элементарного квадрата $\sigma$ равна $2^{-2k}$. Очевидно,

$$\frac{N^*}{N} = \frac{N^*\sigma}{N\sigma} = \frac{N^{*\sigma}}{1} = N^*\sigma$$

Это равенство показывает, что при вычислении по методу Монте-Карло интеграла $I$ мы вычисляем вместо площади, ограниченной линиями $x = 0$, $y = 0$, $x = 1$, $y = f(x)$, численно равной интегралу, площадь некоторой ступенчатой фигуры, состоящей из $N^*$ элементарных квадратов.

Разобьем промежуток (0, 1) на участки монотонности функции $f(x)$ и занумеруем эти участки. Очевидно, на $i$-м участке монотонности площадь ступенчатой фигуры отличается от площади, ограниченной сверху кривой $y = f(x)$, не больше, чем на величину

$$2^{-k}|(y_{max})_i - (y_{\min})_i|$$

(напоминаем, что сторона элементарного квадрата равна $2^{-k}$). Тогда интеграл $I$ будет отличаться от площади ступенчатой фигуры не больше, чем на

$$\sum_i |(y_{\max})_i - (y_{\min})_i| \, 2^{-k} = 2^{-k} \sum_i |(y_{\max})_i - (y_{\min})_i| = 2^{-k} V(f)$$

где $V(f)$ — полное изменение * $f(x)$ на промежутке (0, 1). Для арифметической погрешности $\delta$ получилась оценка

$$\delta < 2^{-k} V(f).$$

Для некоторых функций $f(x)$ погрешность $\delta$ может быть весьма значительной. В качестве примера приведем «пилообразную» функцию, график которой показан на рис. 110. Площадь $s$, ограниченная на этом рисунке прямыми $x = 0$, $y = 0$, $x = 1$ и кривой $y = f(x)$, близка к нулю, тогда как значение интеграла, получаемое методом Монте-Карло, будет тем ближе к единице, чем большее количество испытаний для его получения мы

* См. И. П. Натансон, *Теория функций вещественной переменной,* изд. 2, Гостехиздат, 1957 (стр. 234)

предусмотрим.

Мы видим, что арифметическая погрешность при вычислении интеграла $I$ методом Монте-Карло зависит, во-первых, от разрядности применяемых случайных чисел и, во-вторых, от характера подынтегральной функции (от величины ее полного изменения на промежутке интегрирования). Следовательно, применяя метод Монте-Карло для вычисления интегралов, необходимо либо ограничиваться классом функций, имеющих достаточно малое полное изменение $V(f)$, либо применять искусственные приемы для увеличения количества разрядов случайных чисел. Решение любой задачи методом Монте-Карло можно истолковать как вычисление некоторого однократного или многократного интеграла и с помощью рассуждений, аналогичных приведенным, получить оценку арифметической погрешности, возникающей при этом решении.

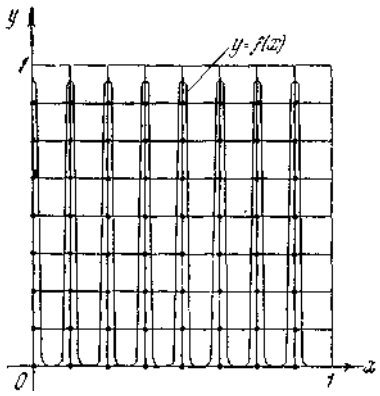


Рис. 110. «Пилообразная» функция.

**9. Преимущества метода.** Для решения многих задач метод Монте-Карло значительно более выгоден, чем другие известные методы. В ряде случаев он оказывается единственным небезнадежным методом из числа известных. Например, вычисление интегралов высокой кратности обычными методами может требовать сотен лет машинного времени, а по методу Монте-Карло такие интегралы вычисляются машиной в течение часов или даже минут. Практика показывает, что метод Монте-Карло становится выгоднее других методов при кратности интегралов, превосходящей 4.

Связность алгорифма при решении задач методом Монте-Карло весьма невелика — нужно от испытания к испытанию хранить лишь пару чисел: $n$ и $m$; в более сложных случаях — несколько пар чисел $n_i$ и $m_i$.

Метод Монте-Карло может быть использован для решения весьма разнообразных задач.

Наконец, метод Монте-Карло не требует специального контроля правильности вычислений, так как случайные сбои в работе машины происходят обычно не чаще, чем один раз в 20—30 минут. Ошибка же в оценке отдельного испытания или нескольких отдельных испытаний из десятков или даже сотен тысяч испытаний практически не повлияет на точность результата, даваемого методом.

## § 44. Способы задания и вычисления значений функций

**1. Выбор способа задания функции.** Решение на машине многих задач бывает связано с вычислением значений некоторой функции $y = f(x)$ (или нескольких функций). Функция $y = f(x)$ может быть задана многими способами, например:

а) в явном виде с помощью аналитической формулы, содержащей конечное число основных операций (арифметических, а также операций вида $|x|$, $x^n$, $e^x$, $\ln x$, $\sin x$, $\arcsin x$ и т.д.);

б) в неявном виде с помощью аналитической формулы вида

$$F(x,y) = 0,$$

содержащей конечное число основных операций;

в) с помощью дифференциального уравнения, одним из частных решений которого является $y=f(x)$\

г) с помощью системы целых многочленов, каждый из которых близок к функции на определенном промежутке изменения независимого переменного; этот способ задания называется способом аппроксимирующих многочленов;

д) с помощью таблицы значений функции; этот способ задания часто применяется, если значения функции получены экспериментальным путем; иногда в условиях задач фигурируют функции, заданные графически (график может быть получен, например, с помощью самопишущего прибора в процессе эксперимента), обычно от графического задания функции путем обмера графика переходят к табличному заданию.

Кроме перечисленных наиболее часто встречающихся способов задания функции возможны многие другие. Каждый из описанных способов задания функции является эффективным в том смысле, что допускает определенный способ вычисления значений этой функции.

При этом нужно иметь в виду, что при численном решении задач получение значения функции осуществляется всегда приближенно, с определенной точностью. Поэтому многие способы задания функции и вычисления ее значений, теоретически требующие выполнения бесконечного числа операций, на практике причисляются к способам, связанным с конечным числом операций. Например, задание функции в виде сходящегося степенного ряда мы относим к случаю а), удерживая конечное число первых членов ряда и отбрасывая остальные члены.

Составляя программу для решения задачи, можно остановиться на том способе задания функции, который применен в условии задачи. Вычисление значений функции будет при этом осуществляться с помощью приемов, отвечающих этому способу. Однако в ряде случаев существует возможность перейти от одного способа задания функции к другому и соответственно от одного способа получения ее значений к другому. Так, любой способ задания функций может быть заменен табличным или способом аппроксимирующих многочленов.

Выбирая способ задания и соответственно вычисления значений функции, следует учитывать два обстоятельства:

1. Количество ячеек памяти, необходимое для осуществления выбираемого способа. Если количество ячеек становится очень большим, возникает необходимость применения магнитной ленты.

2. Машинное время, расходуемое на вычисление всех значений функции, необходимых при решении задачи. Это время можно характеризовать количеством рабочих тактов машины, если не применяется магнитная лента. В общем случае удобно исчислять его в обычных единицах времени.

Сравнивая между собой способы, не требующие применения магнитной ленты, можно характеризовать каждый из них количеством рабочих тактов машины при вычислении одного значения функции.

Отметим, что сравнение способов по расходу машинного времени, как правило, является лишь ориентировочным, так как для вычисления значений функции, отвечающих различным значениям независимого переменного (при одном

**и** том же способе), может расходоваться различное время. Обычно, подсчитывая машинное время, исходят из его расхода на вычисления самых невыгодных значений функции.

Количество ячеек памяти, необходимое при том или ином способе задания функции, существенно именно при решении вопроса о том, можно ли обойтись быстродействующей оперативной памятью машины или нужно применять медленный внешний накопитель.

Основным фактором, определяющим выбор способа задания функций (из числа возможных) является общее машинное время, расходуемое на вычисление всех нужных значений функции. Как правило, следует выбирать тот способ, который требует меньшего машинного времени. И лишь в случае ничтожного машинного времени можно выбирать способ, сообразуясь с легкостью программирования.

Рассмотрим в отдельности каждый из вышеприведенных способов задания функции.

**2. Аналитическое задание функции.** Пусть функция задана явно с помощью аналитической формулы, содержащей конечное число основных операций.

Машина *Стрела,* как читатель знает, имеет специальное запоминающее устройство *(НСП),* постоянно хранящее стандартные программы для вычисления значений ряда элементарных функций. Для машин, не имеющих *НСП,* обычно составляют библиотеку стандартных подпрограмм для вычисления элементарных функций. Подпрограммы, нужные при решении конкретной задачи, либо включают в состав основной программы, либо вводят на определенное место в память и предусматривают в основной программе обращения к ним.

Таким образом, если функция $y = f(x)$ задана с помощью конечного числа операций (арифметических, $|x|$, $\sqrt{x}$, $\ln x$ и т. п.), можно составить подпрограмму для вычисления значений *f(x),* которая может быть, в частности, включена в состав основной программы. Если оказывается, что на вычисление значений $f(x)$ при этом расходуется много времени, возникает вопрос о переходе от указанного способа задания функции к какому-нибудь другому, например к табличному. При этом приходится, конечно, вычислять значения функции, входящие в таблицу. Последнее может быть целесообразным, если количество значений функции, необходимое для составления таблицы, значительно меньше количества значений, которые должны быть вычислены в процессе решения задачи.

Пусть функция задана неявно с помощью аналитической формулы

$$F(x, y) = 0, \qquad \text{(VIII.27)}$$

содержащей конечное число основных операций.

В этом случае следует уравнение (VIII.27), определяющее $y$ как неявную функцию $x$, представить в виде

$$y = \theta(x, y). \qquad \text{(VIII.28)}$$

Для получения значения $y_k = f(x_k)$ решают уравнение (VIII.28) методом итераций, положив в нем $x = x_k$. Для этого производят последовательность вычислений

$$(y_k)_{i+1} = \theta[x_k, (y_k)_i], \quad i = 0, 1. 2. \ldots,$$

до тех пор, пока значение $y_k$ не будет получено с нужной степенью точности. Метод итераций читателю уже знаком.

Заметим, что, преобразуя уравнение (VIII.27) к виду (VII1.28), нужно следить за тем, чтобы получающееся уравнение обеспечивало сходимость итераций. Кроме того, нужно решить вопрос о том, как выбирать начальное приближенное значение $(y_k)_0$.

Применение метода итераций может потребовать большого расхода машинного времени. Поэтому необходимо решить вопрос, нельзя ли вместо того, чтобы вычислять все значения величины $y$, нужные по ходу решения задачи, заранее рассчитать таблицу значений функции $y = f(x)$ или систему аппроксимирующих ее полиномов. Это бывает целесообразным, если для построения таблицы или системы аппроксимирующих полиномов требуется гораздо меньше значений функции $y$, чем их приходится вычислять в процессе решения задачи.

**3. Задание функции с помощью дифференциального уравнения.** Пусть функция задана дифференциальным уравнением

$$y^{(n)} = F(x, y, y', \ldots, y^{(n-1)}) \qquad \text{(VIII.29)}$$

и начальными условиями:

$$y|_{x=x_1} = y_1, \quad y'|_{x=x_1} = y'_1, \ldots, y^{(n-1)}|_{x=x_1} = y_1^{(n-1)} \qquad \text{(VIII.30)}$$

Кроме того, пусть задан промежуток изменения независимого переменного $(x_1, X)$.

Значения функции $y = f(x)$ можно получать путем численного интегрирования дифференциального уравнения (VIII.29). Если при этом известно, что значения $x$, для которых в ходе решения задачи будут вычисляться значения $y$, монотонно возрастают, и если правая часть уравнения (VIII.29) для своего вычисления требует небольшого количества тактов, то этот способ может быть довольно выгодным. Если же значения $x$, для которых нужно будет получать значения $y$, не представляют монотонной последовательности, возникает вопрос, не будет ли более экономичным заранее составить таблицу значений функции $y = f(x)$ или подобрать аппроксимирующие многочлены. Для этого, конечно, тоже потребуется численное интегрирование дифференциального уравнения.

**4. Способ аппроксимирующих многочленов.** Интервал изменения независимой переменной $x$ разбивают на несколько частичных интервалов и на каждом из этих частичных интервалов заменяют $f(x)$ многочленом (целым относительно $x$). Степень каждого многочлена и его коэффициенты должны быть подобраны так, чтобы выполнялись следующие условия:

1) степень каждого многочлена должна быть возможно более низкой;

2) значения многочленов должны отличаться от соответствующих значений функций на величину меньшую, чем некоторое малое число e, характеризующее допустимую погрешность получаемых значений функции. Если $P_n(x)$ — многочлен, то на отвечающем ему частичном интервале изменения величины $x$ должно выполняться неравенство

$$|f(x) - P_n(x)| < \varepsilon$$

Если такой подбор многочленов (их называют *аппроксимирующими многочленами)* удалось осуществить, то можно считать, что все многочлены имеют одинаковую степень (у некоторых многочленов для этого считают коэффициенты при старших степенях $x$ равными нулю). Вместо ввода в память машины большой по объему таблицы значений функции или большой по объему программы для вычисления ее значений теперь вводят небольшую таблицу коэффициентов аппроксимирующих многочленов и включают в основную программу решения задачи подпрограмму, которая для получения значения $f(x)$ выбирает из этой таблицы (в зависимости от значения $x$) соответствующую группу коэффициентов и с их помощью вычисляет значение аппроксимирующего многочлена, являющееся приближенным значением нашей функции.

Таким образом, одним из методов получения значений функции является кусочная аппроксимация ее многочленами.

**5. Табличное задание функции.** Таблица значений функции $f(x)$ представляет собой пару последовательностей чисел

$$\left.\begin{aligned} &x_1, x_2, x_3, \ldots, x_n, \\ &y_1, y_2, y_3, \ldots, y_n, \end{aligned}\right\} \qquad \text{(VIII.31)}$$

которые связаны зависимостью $y_i = f(x_i)$. Последовательность $x_1\ x_2,\ \ldots,\ x_n$ будем считать монотонно возрастающей.

По таблице (VIII.31) можно непосредственно отыскивать лишь значения функции, соответствующие значениям независимого переменного $x_1\ x_2,\ \ldots,\ x_n$, входящим в таблицу. В общем случае (точнее, при любом $x$, лежащем в промежутке между $x_1$ и $x_n$) отыскание значения функции распадается на два этапа.

1. В ы б о р  из т а б л и ц ы . Под этим подразумевается нахождение частичного промежутка $(x_i, x_{i+1})$, в котором лежит заданное значение $x$. Обычно для этого ищут значение индекса $i$, при котором выполняются неравенства

$$x_i \le x < x_{i+1}. \qquad \text{(VIII.32)}$$

Иногда (см. ниже о «способе скользящего конца таблицы») бывает удобнее использовать неравенства

$$x_i < x \le x_{i+1} \qquad \text{(VIII.33)}$$

(это зависит от того, к какому из двух смежных промежутков относить разделяющую их точку $x_t$).

2. И н т е р п о л я ц и я . Определив значение индекса $i$, соответствующее заданному $x$, применяют один из способов интерполяции, получают искомое значение $y$ с помощью табличных значений $y_i$ и $y_{i+1}$ (а также с помощью других смежных табличных значений функции, если того требует применяемый способ интерполяции) Выбор способа интерполяции обусловлен характером функциональной зависимости $y = f(x)$ и допустимой погрешностью получаемого значения $y$.

Заметим, что таблицы удобно подразделять на большие и малые. Если удалось разместить таблицу в оперативной памяти машины, будем называть ее *малой* таблицей. Если же для размещения таблицы пришлось прибегнуть к медленному накопителю (например, магнитной ленте), таблицу будем считать *большой.*

Разделение таблиц на большие и малые является относительным. Одна и та же таблица для одной задачи будет считаться малой, а для другой может оказаться большой (в зависимости от количества ячеек, занятых другим материалом, относящимся к задаче, — программой, исходными данными, промежуточными и окончательными результатами и т. п.).

Окажется ли таблица большой или малой — зависит также от изобретательности программиста (это ярко иллюстрирует приведенный в следующем параграфе способ размещения таблиц в памяти, разработанный П. Н. Комоловым).

## § 45. Способы выбора значений функции из малых таблиц

Способы выбора из больших таблиц можно оценить только в процессе рассмотрения общей организации программы, о которой коротко сказано в § 41. Остановимся подробнее на наиболее известных способах выбора из малых таблиц. Примеры подпрограмм выбора приведены в конце настоящего параграфа.

**1. Регулярные таблицы.** Малая таблица, которой поставлена в соответствие монотонно возрастающая (вместе с $x$) функция $\beta(x)$, удовлетворяющая условию

$$\beta(x) = i, \qquad \text{(VIII.34)}$$

называется *регулярной.* (Ясно, что, рассматривая регулярную таблицу вне связи с функцией $\beta(x)$, мы можем ее считать нерегулярной.) Функцию $\beta(x)$ называют *характеристикой регулярной* таблицы. В силу монотонности возрастания функции $\beta(x)$ для нее существует обратная функция $\delta(y)$, удовлетворяющая условию $\delta(y) = x_i$.

Если некоторая таблица является регулярной, то, вместо того чтобы хранить в памяти машины обе последовательности (VIII.31), составляющие эту таблицу, достаточно разместить в памяти только последовательность значений функции

$$y_1, y_2, y_3, \ldots, y_n \qquad \text{(VIII.35)}$$

Допустим, что последовательность (VIII.35) размещена в памяти следующим образом:

$$\langle y_1 \rangle = b + 1, \quad \langle y_2 \rangle = b + 2, \ldots, \quad \langle y_n \rangle = b + n$$

Зная $\beta(x)$, можно найти $b+i$ — номер ячейки, хранящей число $y_i$, а нужную для интерполяции величину $x_{i+1} - x_i$ можно получить по формуле

$$x_{i+1} - x_i = \delta(i+1) - \delta(i) \qquad \text{(VIII.36)}$$

Предположим, что дано значение независимого переменного $x$, для которого нужно найти $y = f(x)$. Вместо того

чтобы искать $x_i$ и $x_{i+1}$, удовлетворяющие неравенствам (VIII. 32), можно сразу найти номер ячейки $b + i$, хранящей величину $y_i$, отвечающую значению независимого переменного $x_i$. Для этого достаточно вычислить величину

$$b + E[\beta(x)] = b + i, \tag{VIII.37}$$

где символ $E[N]$ (читается «антье от $N$») означает, как обычно, целую часть числа $N$. Зная номер $b + i$ ячейки, хранящей $y_i$, можем с помощью интерполяции табличных значений функции вычислить необходимое нам значение $y$ (очевидно, $\langle y_{i+1}\rangle = b + i + 1$).

Рассмотрим, например, таблицы с постоянным шагом независимого переменного. Так называют таблицы, у которых последовательность значений независимого переменного $x_1, x_2, x_3, \ldots, x_n$ удовлетворяет условию

$$x_i = x_0 + ih, \tag{VIII.38}$$

где $h$ = const.

Таблицы с постоянным шагом независимого переменного относятся к числу регулярных таблиц. Характеристику такой таблицы можно получить следующим образом. Из формулы (VIII. 38) вытекает, что

$$\frac{x_i - x_0}{h} = i \tag{VIII.39}$$

Рассматривая выражение

$$\beta(x) = \frac{x - x_0}{h} \tag{VIII.40}$$

убеждаемся в том, что $\beta(x)$ является характеристикой, ибо, во-первых, $\beta(x)$ — монотонно возрастающая функция, а во-вторых, в силу (VIII.39)

$$\beta(x_i) = i. \tag{VIII.41}$$

Теперь нетрудно воспользоваться приемом выбора из таблицы, описанным выше (см. (VIII.37)). Вычисляя величину

$$b + i = b + E\left[\frac{x - x_0}{h}\right], \tag{VIII.42}$$

мы находим номер ячейки, хранящей число $y_i$, отвечающее значению $x_t$ независимого переменного. Остается путем интерполяции получить искомую величину $y$.

**2. Нерегулярные таблицы. Способ перебора.** Такие таблицы должны быть представлены в памяти двумя составляющими их последовательностями чисел (VIII.31). Отыскание промежутка $(x_i\ x_{i+1})$, удовлетворяющего условию (VIII.32), приходится осуществлять путем анализа последовательности $x_1, x_2, x_3, \ldots, x_n$ (способом перебора). Предположим, что таблица размещена в памяти машины следующим образом:

$$\langle x_1\rangle = c+1, \quad \langle x_2\rangle = c + 2, \ldots, \quad \langle x_i\rangle = c + i, \ldots, \quad \langle x_n\rangle = c + n;$$
$$\langle y_1\rangle = b+1, \quad \langle y_2\rangle = b + 2, \ldots, \quad \langle y_i\rangle = b + i, \ldots, \quad \langle y_n\rangle = b + n;$$

Введем обозначение:

$$b - c = \Delta. \tag{VIII. 43}$$

Выбор способом перебора состоит в последовательной проверке неравенства

$$x < x_1$$
$$\ldots\ldots$$
$$x < x_i,$$
$$x < x_{i+1}$$
$$\ldots\ldots$$
$$x < x_n.$$

Проверка этих неравенств продолжается до тех пор, пока не будет найдено первое значение индекса $i + 1$, при котором неравенство оказывается выполненным. К адресу ячейки, хранящей $x_{i+1}$, который при этих проверках будет найден, прибавляем поправку $\Delta$ (см. формулу (VIII.43)), после чего получаем номер ячейки $b + i + 1$, хранящей число $y_{i+1}$, нужное для интерполяции.

**3. Почти-регулярные таблицы.** Таблицу, которой поставлена в соответствие функция $\beta(x)$, монотонно возрастающая и удовлетворяющая условию

$$|\beta(x_i) - i| \le k,$$

где $k \ge 0$ — целое число, называют *почти-регулярной* таблицей, если $k$ значительно меньше, чем $n$ (например, в несколько десятков раз).

Функцию $\beta(x)$ принято называть *характеристикой почти-регулярной таблицы,* а число $k$ — *модулем нерегулярности.*

Очевидно, любая таблица была бы почти-регулярной, если бы не было предъявлено требование, в силу которого $k$ должно быть мало по сравнению с $n$.

Действительно, откажемся временно от требования малости величины $k$ и рассмотрим функцию

$$\beta(x) = \frac{n-1}{x_n - x_1}(x - x_1) + 1.$$

Это — функция монотонно возрастающая. Кроме того,

$$\beta(x_1) = 1, \quad \beta(x_n) = n.$$

В силу монотонности $\beta(x)$ при $x_1 \le x \le x_n$ справедливо неравенство

$$1 = \beta(x_1) \le \beta(x) \le \beta(x_n) = n.$$

Следовательно,

$$1 \le \beta(x) \le n$$

Отнимая от каждой части этого неравенства $i$, получаем:

$$1 - i \le \beta(x_i) - i \le n - i. \qquad \text{(VIII.44)}$$

Так как $\max(i - 1) = \max(n - i) = n - 1$, то из неравенства (VIII.44) следует, что

$$|\beta(x_i) - i| \le n - 1.$$

Мы доказали, что $\beta(x)$ является характеристикой таблицы, причем модуль нерегулярности равен $k = n - 1$. Отсюда следует, что произвольную таблицу можно считать почти-регулярной, если не требовать, чтобы $k$ было значительно меньше, чем $n$. Таким образом, становится очевидным, что требование, гласящее, что $k$ должно быть малым по сравнению с $n$, является существенным.

Из таблиц, имеющих малый модуль нерегулярности $k$, удобно производить выбор с помощью следующего приема. Пусть $x$ — заданное значение независимого переменного. Вычисляем величину

$$c + E[\beta(x)] = c + i^*.$$

Можно утверждать, что искомое число $c + i$ — номер ячейки, хранящей нужную нам величину $x_i$,—содержится в интервале $(c + i^* - k,\ c + i^* + k)$. В этом интервале производят путем перебора поиск ячейки $c + i$, двигаясь влево или вправо от $c + i^*$ в зависимости от того, выполнено или нет неравенство $x < (c + i^*)$.

В частном случае может оказаться, что $\beta(x)$ удовлетворяет условию

$$0 \le i - \beta(x_i) \le k, \qquad \text{(VIII.45)}$$

т.е.

$$\beta(x_i) \le i \le \beta(x_i) + k \qquad \text{(VIII.46)}$$

В этом случае выбор из таблицы удобно осуществлять так: сперва вычислить величину

$$c + E[\beta(x)] = c + i^*.$$

Затем способом перебора искать $x_i$ в ячейках, начиная с $c + i^*$ и кончая $c + i^* + k$. Таблица, характеристическая функция которой удовлетворяет условию (VIII.46), как говорят, имеет *нарушение регулярности вправо*.

*Нарушение регулярности влево* характеризуется тем, что

$$0 \le \beta(x_i) - i \le k \qquad \text{(VIII.47)}$$

или, что то же,

$$\beta(x_i) - k \le \beta(x_i) \qquad \text{(VIII.48)}$$

В этом случае выбор из таблицы удобно производить так: сперва вычислить величину

$$c + E[\beta(x)] = c + i^*,$$

а затем обычным перебором искать $x_i$ в ячейках с номерами от $c + i^* - k$ до $c + i^*$.

В качестве примера почти-регулярных таблиц рассмотрим таблицы с почти-постоянным шагом независимого переменного. Обозначим

$$\frac{x_n - x_1}{n - 1} = h.$$

Число $h$ называют *средним шагом* таблицы. Если для чисел последовательности

$$x_1, x_2 x_3, \ \ldots, \ x_n$$

справедливо неравенство

$$|x_i - (x_0 + ih)| \le k_1 \qquad \text{(VIII.49)}$$

где $x_0 = x_1 - h$, а $k_1 \ge 0$ — достаточно малое число, то таблицу называют таблицей с *почти-постоянным шагом* независимого переменного.

Очевидно, если допускать сколь угодно большое $k_1$, то любая таблица имеет почти-постоянный шаг независимого переменного.

Интерес представляет случай, когда $\frac{k_1}{h}$ во много раз меньше числа $n$.

Таблица с почти-постоянным шагом независимого переменного является, конечно, почти-регулярной. Характеристику ее легко получить следующим образом. Если разделить обе части неравенства (VIII.49) на $h$, получим:

$$\left| \frac{x_i - x_0}{h} - i \right| \le \frac{k_1}{h} < E\left[ \frac{k_1}{h} + 1 \right],$$

т. е.

$$\left|\frac{x_i - x_o}{h} - i\right| < k \qquad \text{(VIII.50)}$$

где введено обозначение $k = E\left[\frac{k_1}{h} + 1\right]$. Составляя функцию

$$\beta(x) = \frac{x - x_0}{h}, \qquad \text{(VIII.51)}$$

видим, что, во-первых, она является монотонно возрастающей, а во-вторых, в силу (VIII.50) удовлетворяет неравенству

$$|\beta(x_i) - i| < k \qquad \text{(VIII.52)}$$

Отсюда заключаем, что эта функция является характеристикой таблицы, имеющей модуль нерегулярности

$$k = E\left[\frac{k_1}{h} + 1\right].$$

Для нахождения номера ячейки, хранящей число $x_t$ такое, что $x_i < x < x_{i+1}$, вычисляем величину

$$c + E\,[\beta(x) - k] = c + i^*.$$

Одна из ячеек, начиная с $c + i^* - k$ и кончая $c + i^* + k$, содержит искомое число $x_i$. Отыскание этого числа осуществляют способом, описанным на стр. 435 для почти-регулярных таблиц.

В частных случаях может быть отклонение от постоянного шага влево или вправо.

*Отклонение* от постоянного шага *вправо* характеризуется тем, что условие (VIII.52) принимает вид

$$0 < i - \beta(x_i) \le k$$

или, учитывая (VI 11.51),

$$\frac{x_i - x_0}{h} \le i \le \frac{x_i - x_0}{h} + k.$$

В этом случае для выбора из таблицы следует сперва вычислить величину

$$c + i^{**} = E\left[c + \frac{x - x_0}{h}\right],$$

а затем искать $x_i$ в ячейках, начиная с $c + i^{**}$ и кончая $c + i^{**} + k$. При *отклонении* от постоянного шага влево неравенство (VIII. 52) принимает такой вид:

$$0 \le \beta(x_i) - i \le k$$

или, учитывая (VIII. 51),

$$\frac{x_i - x_0}{h} - k \le i \le \frac{x_i - x_0}{h}.$$

В этом случае для выбора из таблицы следует вычислить

$$c + i^{**} = E\left[c + \frac{x_i - x_0}{h}\right],$$

а затем искать $x_i$ в ячейках, начиная с $c + i^{**}$ и кончая $c + i^{**} - k$.

**4. Способ двухстепенного перебора.** Выбор из таблицы способом двухстепенного перебора состоит в следующем.

Последовательность ячеек, хранящих числа $x_1\, x_2, \ldots, x_n$ (соответственно и сама эта последовательность чисел), разбивается на $m$ частей, каждая из которых содержит $\nu = \frac{n}{m}$ ячеек в случае, если $\frac{n}{m}$ — целое число. Если же $\frac{n}{m}$ оказывается дробью, то $\nu$ вычисляют по формуле

$$\nu = E\left[\frac{n}{m} + 1\right]. \qquad \text{(VIII.53)}$$

Затем проверяем, выполнено ли неравенство

$$(m - 1)\,\nu < n < m\nu. \qquad \text{(VIII.54)}$$

Если это неравенство выполнено, то считаем последовательность ячеек разбитой на $m$ частей, имеющих по $\nu$ ячеек в каждой.

Если неравенство (VIII.54) не выполнено, то проверяем, будет ли иметь место неравенство

$$(m - 2)\,v < n < (m - 1)v. \qquad \text{(VIII.55)}$$

В случае выполнения этого неравенства считаем последовательность ячеек разделенной на $m^* = m - 1$ частей, каждая из которых содержит по $v$ ячеек.

Если неравенство (VIII. 55) не выполнено, то проверяется неравенство

$$(m - 3)v < n < (m - 2)v \qquad \text{(VIII.56)}$$

и т. д., пока не получится неравенство вида

$$(m - j - 1)v < n < (m - j)v, \qquad \text{(VIII.57)}$$

которое будет выполнено. При этом считаем таблицу разбитой на $m^* = m - j$ частей по $v$ ячеек в каждой части. Последняя из частей не будет содержать полного числа ячеек. Эту часть дополним, присоединяя к ней недостающие ячейки и записывая в них число $x_n$.

После того как последовательность ячеек дополнена, она содержит $vm^*$ ячеек (в случае целого $\frac{n}{m}$ имеем $m^*=m$ и дополнительных ячеек нет). Прибавляем к ней еще одну ячейку, в которую записываем $x_n$.

Теперь таблица подготовлена к выбору из нее значений функции способом двухстепенного перебора.

Прежде чем перейти к описанию двухстепенного перебора, покажем на примерах описанную подготовку таблицы (обратите внимание на последние ячейки).

П р и м е р 1. Пусть $n = 14$ (десятичное) и мы хотим разбить таблицу на семь частей. Имеем $v = 14/7 = 2$ — целое.

Последовательности значений независимого переменного, записанной в памяти машины, придаем такой вид:

| | | |
|---|---|---|
| $(c+1) = x_1$ | $(c+6) = x_6$ | $(c+11) = x_{11}$ |
| $(c+2) = x_2$ | $(c+7) = x_7$ | $(c+12) = x_{12}$ |
| $(c+3) = x_3$ | $(c+8) = x_8$ | $(c+13) = x_{13}$ |
| $(c+4) = x_4$ | $(c+9) = x_9$ | $(c+14) = x_{14}$ |
| $(c+5) = x_5$ | $(c+10) = x_{10}$ | $(c+15) = x_{14}$ |

П р и м е р 2. Пусть $n = 9$ и мы хотим разбить последовательность ячеек, хранящих $x_1, x_2, x_3, \ldots, x_n$ на $m = 7$ частей. Имеем 9/7 –дробь, $v = E\left(\frac{9}{7} + 1\right) = 2$.

Вычисляем

$$mv = 7 \cdot 2 = 14,$$
$$(m - 1)\,v = 6 \cdot 2 = 12,$$
$$(m - 2)\,v = 5 \cdot 2 = 10,$$
$$(m - 3)\,v = 4 \cdot 2 = 8.$$

Видим, что $n = 9$ удовлетворяет неравенству

$$(m - 3)\,v < n < (m - 2)\,v.$$

Отсюда заключаем, что таблица фактически разбита нами на

$$m^* = m - 2 = 7 - 2 = 5$$

частей по две ячейки в каждой части. Последняя (пятая) часть содержит только одну ячейку. Дополняем эту часть до полного количества (до двух) ячеек. Кроме того, добавляем еще одну ячейку. В дополнительных ячейках записываем число $x_0$.

После подготовки последовательность ячеек, хранящих значения независимого переменного, принимает такой вид:

| | | |
|---|---|---|
| $(c+1) = x_1$ | $(c+5) = x_5$ | $(c+9) = x_9$ |
| $(c+2) = x_2$ | $(c+6) = x_6$ | $(c+10) = x_9$ |
| $(c+3) = x_3$ | $(c+7) = x_7$ | $(c+11) = x_9$. |
| $(c+4) = x_4$ | $(c+8) = x_8$ | |

В приведенных примерах применение способа двухстепенного перебора практически бесполезно. Таблицы, содержащие так мало чисел, нами выбраны исключительно для того, чтобы легче было пояснить подготовку таблиц.

Перейдем к описанию способа двухстепенного перебора.

Сначала производится перебор первой степени, состоящий в последовательной проверке неравенств:

$$x < x_1,$$
$$x < x_{v+1}$$
$$x < x_{2v+1}$$
$$\ldots\ldots$$
$$x < x_{mv+1}$$

Проверка неравенств продолжается до тех пор, пока не будет найдено первое из чисел $k + 1$ такое, что неравенство

$$x < x_{(k+1)v+1}$$

оказывается выполненным.

Затем производится перебор второй степени, т. е. обычный перебор, в процессе которого в ячейках с номерами $kv + 1$, $kv + 2$, $kv + 3$, ..., $(k + 1)v$ отыскивается необходимое для нас число $x_i$ такое, что $x_i \leq; x < x_{i+1}$.

При выборе из таблицы способом двухстепенного перебора максимальное число проверок неравенства, очевидно, равно $w = m + v$. Нетрудно приближенно подсчитать значение величины $m$, при котором количество проверок неравенств будет наименьшим. Для этого считаем, что $v = \frac{n}{m}$ (при $\frac{n}{m}$ не целом последнее равенство является лишь приближенным). Получаем:

$$w = m + \frac{n}{m}$$

Ищем минимум последней функции, приравнивая нулю ее производную по $m$:

$$\frac{dw}{dm} = 1 - \frac{n}{m^2} = 0$$

Отсюда находим

$$m = \sqrt{n}\,.$$

Таким образом, величину $m$ следует брать близкой к числу $\sqrt{n}$. Например, при $n$=600 берем $m = 25 \approx \sqrt{600}$.

Кроме способа двухстепенного перебора, возможны способы трехстепенного перебора, четырехстепенного перебора и т. д. На их описании останавливаться не будем.

**5. Способ деления «пополам».** Выбор из таблицы способом деления «пополам» состоит в следующем. Последовательность чисел $x_1, x_2, \ldots, x_n$ делится на две приблизительно равные по количеству чисел части, для чего вычисляется величина $E\,[n/2] = m$. Затем проверяется неравенство

$$x < x_m. \qquad \text{(VIII.58)}$$

Если неравенство (VIII.58) выполнено, то делению на две примерно равные части подвергается последовательность чисел. $x_1, x_2, \ldots, x_m$ Если же неравенство (VIII.58) не выполнено, то делению примерно пополам подвергается последовательность чисел $x_m, x_{m+1}, \ldots, x_n$

Описанный процесс повторяется до тех пор, пока не будет получена часть последовательности, состоящая из одного числа. Это число является одним из искомых чисел $x_t$ или $x_{i+1}$, участвующих в неравенствах

$$x_i \le x \le x_{i+1}$$

При делении примерно пополам некоторой части последовательности, начинающейся с $x_\lambda$ и кончающейся $x_\mu$, находим число $E\left[\frac{\mu - \lambda}{2}\right] = \delta$. Затем производим проверку выполнения неравенства

$$x < x_{\lambda+\delta} \qquad \text{(VIII.59)}$$

Если неравенство (VIII.59) выполнено, то справедливо условие

$$x_\lambda \le x < x_{\lambda+\delta}$$

Если неравенство (VIII.59) не выполнено, то справедливо условие

$$x_{\lambda+\delta} \le x < x_\mu$$

В первом случае переходим к части последовательности, начинающейся с числа $x_\lambda$ и кончающейся числом $x_{\lambda+\delta}$. Во втором случае — к части последовательности, начинающейся числом $x_{\lambda+\delta}$ и кончающейся числом $x_\mu$.

**6. Способ скользящего начала (конца) таблицы.** Предположим, что из условия задачи видно, что значения независимого переменного, для которых нужно будет находить значения функции, монотонно возрастают (убывают), т. е. видно, что, вычислив функцию для некоторого $x = \bar{x}$, в следующий раз мы будем вычислять значение функции для $x > \bar{x}$ ($x < \bar{x}$). В этом случае удобен выбор из таблицы способом скользящего начала (конца) таблицы.

Для определенности остановимся на способе скользящего начала таблицы. Сущность его весьма проста. Сперва считают, что таблица начинается с чисел $x_1$ $y_1$. После того как был произведен первый выбор из таблицы значения функции, отвечающего некоторому значению $x$, удовлетворяющему условию

$$x_i \le x < x_{i+1},$$

считают, что таблица начинается числами $x_i$, $y_i$ и т. д. Выбор из таблицы производится одним из вышеописанных способов.

Итак, по мере выполнения выборов из таблицы при способе скользящего начала таблицы «длина» этой таблицы (количество чисел в каждой из двух составляющих ее последовательностей) все время уменьшается. При способе скользящего начала вполне целесообразно производить каждый выбор обычным перебором.

Если значения $x$, для которых мы вычисляем по таблице функцию $y$, образуют монотонно убывающую последовательность (при каждом новом выборе $x$ меньше, чем при предыдущем), то скользящим должен быть конец таблицы. В этом случае каждый выбор можно производить простым перебором, начиная, однако, перебор не с начала таблицы, а с ее конца. Вместо неравенства (VIII. 32) при этом удобно применять неравенство (VIII.33).

**7. Способы «плотного» размещения таблиц в памяти.** Ниже приводим два способа компактного размещения в памяти машины таблиц с постоянным шагом независимого переменного. Эти способы в применении к машине *Стрела* были разработаны П. Н. Комоловым.

Способ разностей. Опишем этот способ вначале применительно к машине с фиксированной запятой, ячейка которой содержит $N$ разрядов (из них $N$ — 1 цифровых разрядов). Предположим, что все числа $y_1, y_2, \ldots, y_n$ являются правильными дробями. Вычислим величины

$$\eta_1 = \max y_i, \quad \eta_2 = \min y_i, \quad \text{при } i = 1, 2, \ldots, n.$$

Введем обозначения

$$y_0 = \frac{\eta_1 + \eta_2}{2}, \qquad \delta = \frac{\eta_1 - \eta_2}{2}$$

Очевидно,

$$\delta = \max|y_i - y_0|, \quad i=1, 2, ..., n.$$

Предположим, что при записи в ячейке значащие цифры числа δ занимают последние $k$ — 1 ее цифровых разрядов. Очевидно, числа $\delta_i = y_i - y_0$ также являются правильными дробями и значащие цифры каждого из них при записи в ячейке будут занимать не более чем $k$ — 1 младших цифровых разрядов. Учитывая еще знаковый разряд, видим, что для изображения каждого из чисел $\delta_i$ фактически требуется всего $k$ разрядов ячейки. Если

$$E\left[\frac{N}{k}\right] = \nu \geq 2,$$

го указанное выше обстоятельство позволяет в одной ячейке искусственно разместить несколько (именно $\nu$) разностей.

Таблица теперь может быть размещена так: в ячейке $b$ запишем число $y_0$, а в ячейках, начиная с $b + 1$, будем размещать по $\nu$ последовательных разностей $\delta_г$.

Выбор из такой компактной таблицы нужно производить следующим образом. Пусть дан $x$. Ищем нужное для интерполяции табличное значение $y_i$. Вычисляя

$$b + i' = b + 1 + E\left[\frac{x - x_1}{\nu h}\right],$$

получим номер ячейки, хранящей разность $\delta_i$. По формуле

$$\mu = 1 + E\left\{\left[\frac{x - x_1}{\nu h} - E\left(\frac{x - x_1}{\nu h}\right)\right]\nu\right\}.$$

найдем номер участка ячейки, хранящей разность $\delta_i$. Перенося содержимое первого разряда выделенного участка в знаковый разряд некоторой рабочей ячейки α, а содержимое остальных разрядов этого участка в младшие цифровые разряды ячейки α, получим:

$$(\alpha) = \delta_{i;}.$$

Остается выполнить операцию $(b) + (\alpha) = y_0 + \delta_i = y_i$. Зная расположение $\delta_i$ , легко выбираем из таблицы так же $\delta_{i+1}$, а если нужно, то и другие разности, и производим интерполяцию, в результате которой получаем искомое значение функции $y$.

Для машины с плавающей запятой этот способ становится несколько более сложным. Пусть ячейка машины с плавающей запятой состоит из $N$ разрядов, из них $N_1$ разрядов отведены для абсолютной величины мантиссы, $N_2$ разрядов — для абсолютной величины порядка и два разряда — знаковых:

$$N = N_1 + N_2 + 2.$$

Пусть $\eta_1, \eta_2, y_0, \delta, \delta_i$. имеют прежний смысл, а $p_0, p$ и $p_t$ — соответственно порядки чисел $y_0$, δ и $\delta_i$. Очевидно,

$$p_i \leq p$$

Рассмотрим таблицу, для которой $p < p_0$.

Преобразуем числа $\delta_i$ так, чтобы их порядки стали равны $p$. При этом мантиссы чисел $\delta_i$ будут сдвинуты в разрядной сетке ячеек вправо соответственно на $p - p_i$ разрядов. Младшие разряды мантисс (в количестве, равном $p - p_i$) выйдут за пределы разрядной сетки и будут утеряны.

Далее, отбросим от каждой мантиссы по $p_0 - p$ младших разрядов и округлим у каждой из них последний из сохраняющихся разрядов. Теперь каждая мантисса будет занимать (не считая знакового разряда) по $N_i - (p_0 - p)$ цифровых разрядов. В обще сложности каждая мантисса потеряла соответственно

$$(p - p_i) + (p_0 - p) = p_0 - p_i$$

разрядов.

Убедимся в том, что для вычисления величины $y_i$ по формуле

$$y_i = y_0 + \delta_i$$

отброшенные нами разряды мантисс не нужны. Для этого вспомним, что операция сложения начинается в машине с плавающей запятой с выравнивания порядков слагаемых. Так как $p_i < p_0$, то порядок числа $\delta_i$ будет увеличен на $p_0 - p_t$ единиц и сделан равным $p_0$, а мантисса числа $\delta_i$ будет сдвинута вправо на $p_0 - p_i$ разрядов. При этом $p_0 - p_i$ младших ее разрядов будет утеряно (именно столько мы их в общей сложности и отбросили при вышеописанном «укорачивании» мантисс), а последний из сохраняющихся разрядов будет округлен.

Вернемся к рассмотрению чисел $\delta_i$ с «укороченными» мантиссами. Порядки этих чисел сделаны одинаковыми, равными $p$. Их нет надобности хранить совместно с каждой мантиссой. А для хранения каждой мантиссы с учетом ее знака требуется по

$$k = N_1 - (p_0 - p) + 1$$

разрядов. Если

$$E\left[\frac{N}{k}\right] = \nu \geq 2,$$

то в каждой ячейке машины можно разместить по нескольку (именно по $\nu$) мантисс чисел $\delta_i$.

Это позволяет представить нашу таблицу следующим способом. В некоторой ячейке с номером $b$ — 1 можно поместить набор цифр, содержащий нули в знаковом и цифровых разрядах мантиссы и $p$ с его знаком в разрядах порядка. В ячейке $b$ запишем число $y_0$. Каждую из ячеек, начиная с $b + 1$, разобьем на $\nu$ участков по $k$ разрядов и

в этих участках расположим «укороченные» мантиссы чисел (с их знаками). Если $N/k$ не является целым числом, то при этом $N$ — $kv$ разрядов каждой ячейки, хранящей «укороченные» мантиссы, окажутся неиспользованными.

Выбор из такой таблицы производится следующим образом. Вычисляя

$$b + i' = b + 1 + E\left[\frac{x - x_0}{vh}\right]$$

получаем номер ячейки, хранящей мантиссу числа $\delta_i$. По формуле

$$\mu = 1 + E\left\{\left[\frac{x - x_1}{vh} - E\left(\frac{x - x_1}{vh}\right)\right]v\right\}$$

находим номер участка ячейки, в котором записана эта мантисса.

Содержимое найденного участка переносим в старшие разряды некоторой рабочей ячейки $\alpha$ (так, чтобы содержимое первого разряда участка попало в знаковый разряд ячейки). В порядковые разряды ячейки $\alpha$ помещаем $p$, беря его из ячейки $b$ — 1. Теперь

$$(\alpha) = \delta_i.$$

По формуле

$$(b) + (\alpha) . = y_0 + \delta_i = y_i.$$

Получаем необходимое нам для интерполяции табличное значение $y_i$. Зная номер ячейки и номер ее участка, где хранится $\delta_i$, легко получаем остальные табличные величины, нужные для интерполяции.

Заметим, что описанное выше «уплотнение» таблицы следует производить также на машине с помощью соответствующей несложной программы.

Способ конечных разностей. Способ конечных разностей отличается от вышеописанного способа разностей тем, что вместо разностей $\delta_i = y_i - y_0$ используются конечные разности первого порядка

$$\Delta y_2 = y_2 - y_1$$
$$\Delta y_3 = y_3 - y_2$$
$$\cdots\cdots\cdots$$
$$\Delta y_n = y_n - y_{n-1}$$

В каждой ячейке памяти располагают по нескольку таких конечных разностей подобно вышеописанному.

При использовании конечных разностей время, расходуемое на выбор из таблицы, значительно возрастает, так как вместо формулы

$$y_i = y_0 + \delta_i$$

приходится вести вычисления по формуле

$$y_i = y_1 + \sum_{j=2}^{i} \Delta y_j$$

Подробное описание этого способа не приводим.

**8. Примеры подпрограмм для выбора значений функции из таблицы.** Если задано значение $x$ и нужно при помощи таблицы получить $y = f(x)$, то, как было сказано выше, эта задача состоит из следующих основных частей:

1) нахождение необходимых для интерполяции табличных значений величины $y$ (достаточно найти $y_i$);
2) получение искомого значения $y$ путем интерполяции.

При решении этой задачи на машине процесс получения величины $y = f(x)$ изображается схемой

$$\mathbf{P}_1\mathbf{\Phi}_2\mathbf{Q}_3$$

где $\mathbf{P}_1$ — обобщенный оператор, находящий $<y_i>$;

$\mathbf{\Phi}_2$— оператор, формирующий $Q_3$;

$\mathbf{Q}_3$ — оператор, производящий интерполяцию.

Мы в нижеследующих примерах ограничимся программами оператора $\mathbf{P}_1$, который должен в качестве результата давать номер ячейки, хранящей $y_i$, представленный в виде соответствующего количества единиц первого или второго адреса.

Выбор из таблицы с постоянным шагом независимого переменного. Пусть исходные данные хранятся в памяти следующим образом:

$$(b + j) = y_i, \qquad j = 1, 2, \ldots, n$$

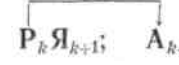
$(d+1) = x_1$, $(d + 2) = 1/h$, $(d + 3) = b + 1$ (I), $(d + 4) = 0000\ 0000\ 0000\ 0\ 13$ (в коде команд), $(d + 5) = x_n$;

$$(c + 1) = x;$$

$c + 2$, $c + 3$ — рабочие ячейки.

Номер $b + 1$ ячейки, хранящей $y_i$, должен быть получен в первом адресе ячейки $c + 2$.

Подпрограмма выбора имеет следующую логическую схему:




где $\mathbf{P}_к$ — логический оператор, проверяющий выполнение условия

$$x_1 \le x < x_n;$$

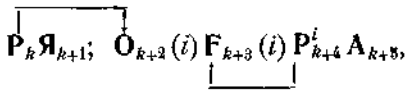
$\mathbf{Я}_{k+1}$ — останов, в случае невыполнения указанного условия;
$\mathbf{A}_{k+2}$ — оператор, вычисляющий $b + i$ (I).
Составим подпрограмму выбора:

| | | | | | |
|---|---|---|---|---|---|
| $\mathbf{P}_k$: | $a$+ 1) | $c$-+ 1 | $d$ + 1 | $c$ + 2 | 0 03 |
| | $a$+ 2) | $a$ + 3 | $a$ + 5 | 0000 | 0 20 |
| | $a$+ 3) | $c$ + 1 | $d$ + 5 | 0000 | 0 03 |
| | $a$+ 4) | $a$ + 5 | $a$ + 6 | 0000 | 0 20 |
| $\mathbf{Я}_{k+1}$: | $a$+ 5) | $c$ + 1 | 0000 | 0000 | 0 40 |
| $\mathbf{A}_{k+2}$: | $a$+ 6) | $d$ + 2 | $c$ + 2 | $c$ + 2 | 0 05 |
| | $a$+ 7) | $c$ + 2 | 7575 | $c$ + 2 | 0 12 |
| | $a$+10) | $c$ + 2 | $d$ + 4 | $c$ + 3 | 0 07 |
| | $a$+11) | $c$ + 2 | $c$ + 3 | $c$ + 2 | 0 14 |
| | $a$+12) | $d$ + 3 | $c$ + 2 | $c$ + 2 | 0 02 |

В ячейке $c + 2$ получено $b + i$ (I).
Если известно, что $x_1 \le x < x_n$, то команды $(a + 2)$ — $(a + 5)$ из подпрограммы выбора можно исключить.
В ы б о р с п о с о б о м п е р е б о р а . Пусть исходные данные расположены в памяти машины так:

$$(c+j) = x_j, \quad (b+y) = y_j \quad j=1, 2, \ldots. n;$$
$$(e + 1) = 1 \text{ (II)}, \qquad (e + 3) = 7777 \text{ (II)},$$
$$(e + 2) = d+1 \;\; c+1 \;\; 0000\ 0\ 03, \qquad (e + 4) = \Delta - 1 \text{ (II)};$$
$$(d+1) = x, \qquad d + 2,\ d + 3 \text{ — рабочие ячейки.}$$

Номер $b + i$ ячейки, хранящей $y_i$, должен быть получен во втором адресе ячейки $d + 3$. Знаком $\Delta$ обозначено $b$— $c$. В настоящем случае логическая схема подпрограммы выбора имеет вид




где $\mathbf{P}_k$ — проверяет условие $x_1 \le x < x_n$;
$\mathbf{Я}_{k+1}$ — останов, если это условие не выполнено;
$\mathbf{O}_{k+2}(i)$ — восстановление $\mathbf{P}^i_{k+4}$ к начальному виду;
$\mathbf{F}_{k+3}(i)$ — переадресация оператора $\mathbf{P}_{k+4}$;
$\mathbf{P}^i_{k+4}$ — проверка неравенства $x<x_{i+1}$;
$\mathbf{A}_{k+5}$ — оператор, вычисляющий $b + I$ (II).
Составим подпрограмму выбора:

| | | | | | |
|---|---|---|---|---|---|
| $\mathbf{P}_k$: | $a$+ 1) | $d$+ 1 | $c$+ 1 | 0000 | 0 03 |
| | $a$+ 2) | $a$+ 3 | $a$+ 5 | 0000 | 0 20 |
| | $a$+ 3) | $d$+ 1 | $c$+ $n$ | 0000 | 0 03 |
| | $a$+ 4) | $a$+ 5 | $a$+ 6 | 0000 | 0 20 |
| $\mathbf{Я}_{k+1}$: | $a$+ 5) | $d$+ 1 | 0000 | 0000 | 0 40 |
| $\mathbf{O}_{k+2}(i)$: | $a$+ 6) | $e$+ 2 | 0000 | $a$+10 | 0 13 |
| $\mathbf{F}_{k+3}(i)$: | $a$+ 7) | $a$+10 | $e$+ 1 | $a$+10 | 0 02 |
| $\mathbf{P}^i_{k+4}$: | $a$+10) | $d$+ 1 | $c$+ 1* | 0000 | 0 03 |
| | $a$+11) | $a$+ 7 | $a$+12 | 0000 | 0 20 |
| $\mathbf{A}_{k+5}$: | $a$+12) | $a$+10 | $e$+ 3 | $d$+ 2 | 0 11 |
| | $a$+13) | $d$+ 2 | $e$+ 4 | $d$+ 3 | 0 02 |

В ячейке $d + 3$ получено $b + i$ (II).
Если известно, что всегда имеет место неравенство

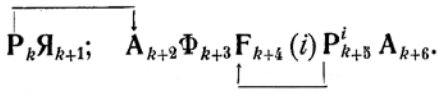
$$x_1 \le x < x_n,$$

то можно исключить операторы $\mathbf{P}_k$, $\mathbf{Я}_{k+1}$ из логической схемы, а представляющие их команды — из подпрограммы выбора.
В ы б о р из т а б л и ц ы с п о ч т и - п о с т о я н н ы м ш а г о м н е з а в и с и м о г о п е р е м е н н о г о . Пусть средний шаг равен $h$, модуль нерегулярности $k$. Таблица расположена в памяти машины так же, как в предыдущем примере. Остальные исходные данные размещены следующим образом:

$$(e + 1) = x_0, \qquad (e + 2) = 1/h,$$
$$(e + 3) = 0000\ 0000\ 0000\ 0\ 27,$$
$$(e+4) = c - k \text{ (II)},$$
$$(e + 5) = d + 1 \;\; 7777\ 0000\ 0\ 03,$$
$$(e + 6) = 1(11), \quad (e + 7) = 7777(\text{II}), \quad (e + 10) = \Delta - 1 \text{ (II)};$$
$$(d + 1) = x,$$
$$d + 2, \quad d + 3 \text{ — рабочие ячейки,}$$

Номер $b + i$ ячейки, хранящей $y_i$, должен быть получен во втором адресе ячейки $d + 2$.

Логическая схема подпрограммы выбора будет иметь вид



Здесь $\mathbf{P}_k$ — оператор, проверяющий условие $x_1 \le x < x_n$,

$\mathbf{Я}_{k+1}$ — останов в случае невыполнения этого условия,

$\mathbf{A}_{k+2}$ — вычисление величины $E[\beta(x)]$ — $k + c$;

$\mathbf{Ф}_{k+3}$ — формирование оператора $\mathbf{P}^i_{k+5}$;

$\mathbf{F}_{k+4}(i)$ — переадресация оператора $\mathbf{P}^i_{k+5}$;

$\mathbf{P}^i_{k+5}$ — проверка неравенств вида $x<x_{i+1}$;

$\mathbf{A}_{k+6}$ — вычисление $b + i$ (II).

Составим подпрограмму выбора:

| | | | | | | |
|---|---|---|---|---|---|---|
| $\mathbf{P}_k$: | $a$+ 1) | $d$+ 1 | $c$+ 1 | 0000 | 0 | 03 |
| | $a$+ 2) | $a$+ 3 | $a$+ 5 | 0000 | 0 | 20 |
| | $a$+ 3) | $d$+ 1 | $c$+ $n$ | 0000 | 0 | 03 |
| | $a$+ 4) | $a$+ 5 | $a$+ 6 | 0000 | 0 | 20 |
| $\mathbf{Я}_{k+1}$: | $a$+ 5) | $d$+ 1 | 0000 | 0000 | 0 | 40 |
| $\mathbf{A}_{k+2}$: | $a$+ 6) | $d$+ 1 | $e$+ 1 | $d$+ 2 | 0 | 03 |
| | $a$+ 7) | $d$+ 2 | $e$+ 2 | $d$+ 2 | 0 | 05 |
| | $a$+10) | $d$+ 2 | 7575 | $d$+ 2 | 0 | 12 |
| | $a$+11) | $d$+ 2 | $e$+ 3 | $d$+ 3 | 0 | 07 |
| | $a$+12) | $d$+ 2 | $d$+ 3 | $d$+ 2 | 0 | 14 |
| | $a$+13) | $d$+ 2 | $e$+ 4 | $d$+ 2 | 0 | 02 |
| $\mathbf{Ф}_{k+3}$: | $a$+14) | $e$+ 5 | $d$+ 2 | $a$+16 | 0 | 02 |
| $\mathbf{F}_{k+4}(i)$: | $a$+15) | $a$+16 | $e$+ 6 | $a$+16 | 0 | 02 |
| $\mathbf{P}^i_{k+5}$: | $a$+16) | 0000 | 0000* | 0000 | 0 | 00 |
| | $a$+17) | $a$+15 | $a$+20 | 0000 | 0 | 20 |
| $\mathbf{A}_{k+6}$: | $a$+20) | $a$+16 | $e$+ 7 | $d$+ 2 | 0 | 11 |
| | $a$+21) | $d$+ 2 | $e$+10 | $d$+ 3 | 0 | 02 |

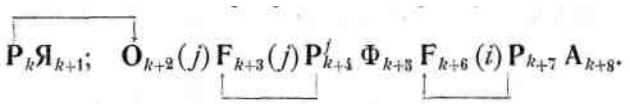
В ячейке $d + 3$ получено $b + i$ (II).

Выбор способом двухстепенного перебора. Считаем таблицу расположенной по-прежнему. Остальные исходные данные размещаем в памяти так:

$(e + 1) = v$(II), $(e + 2) = 1$(11), $(e + 3) = d$+1 $c$+1 0000 0 03,
$(e + 4) = \Delta - 1$(11), $(e + 5) = 7777$ (II), $(e + 6) = v - 1$ (II).

Номер $b + i$ ячейки, хранящей $y_i$, должен быть получен во втором адресе ячейки $d + 2$.

Логическая схема подпрограммы выбора будет иметь такой вид:



Здесь $\mathbf{P}_k$, $\mathbf{Я}_{k+1}$ имеют прежний смысл;

$\mathbf{O}_{k+2}(j)$ — восстановление оператора $\mathbf{P}^j_{k+4}$;

$\mathbf{F}k+3(j)$ — переадресует $\mathbf{P}^j_{k+4}$;

$\mathbf{P}^j_{k+4}$ — при переборе первой степени проверяет неравенства вида $x<x_3$;

$\mathbf{Ф}_{k+5}$ — оператор, формирующий $\mathbf{P}^i_{k+7}$;

$\mathbf{F}_{k+6}(i)$ — переадресует $\mathbf{P}^i_{k+7}$;

$\mathbf{P}^i_{k+7}$ — при переборе второй степени проверяет неравенства вида $x<x_i$;

$\mathbf{A}_{k+8}$ — вычисляет $b + i$ (II).

Составим подпрограмму выбора:

| | | | | | | |
|---|---|---|---|---|---|---|
| $\mathbf{P}_k$: | $a$+ 1) | $d$+ 1 | $c$+ 1 | 0000 | 0 | 03 |
| | $a$+ 2) | $a$+ 3 | $a$+ 5 | 0000 | 0 | 20 |
| | $a$+ 3) | $d$+ 1 | $c$+ $n$ | 0000 | 0 | 03 |
| | $a$+ 4) | $a$+ 5 | $a$+ 6 | 0000 | 0 | 20 |
| $\mathbf{Я}_{k+1}$: | $a$+ 5) | $d$+ 1 | 0000 | 0000 | 0 | 40 |
| $\mathbf{O}_{k+2}(j)$: | $a$+ 6) | $e$+ 3 | 0000 | $a$+10 | 0 | 13 |
| $\mathbf{F}_{k+3}(j)$: | $a$+ 7) | $a$+10 | $e$+ 1 | $a$+10 | 0 | 02 |

| | | | | | | |
|---|---|---|---|---|---|---|
| $\mathbf{P}^{j}_{k+4}$: | $a$+10) | $d$+ 1 | $c$+1+$v$ | 0000 | 0 | 03 |
| | $a$+11) | $a$+ 7 | $a$+12 | 0000 | 0 | 20 |
| $\mathbf{\Phi}_{k+5}$: | $a$+12) | $a$+10 | $e$+ 6 | $a$+14 | 0 | 15 |
| $\mathbf{F}_{k+6}(i)$: | $a$+13) | $a$+14 | $e$+ 2 | $a$+14 | 0 | 02 |
| $\mathbf{P}^{i}_{k+7}$: | $a$+14) | 0000 | 0000* | 0000 | 0 | 00 |
| | $a$+15) | $a$+13 | $a$+16 | 0000 | 0 | 20 |
| $\mathbf{A}_{k+8}$: | $a$+16) | $a$+14 | $e$+ 5 | $d$+ 2 | 0 | 11 |
| | $a$+17) | $d$+ 2 | $e$+ 4 | $d$+ 2 | 0 | 02 |

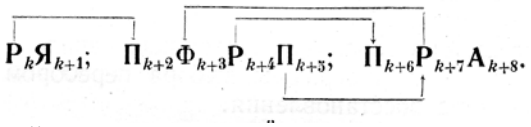
В ячейке $d$+ 2 получено $b + i$ (II).

В ы б о р  с п о с о б о м  д е л е н и я  « п о п о л а м » . Таблицу считаем размещенной в памяти машины по-прежнему. Остальные исходные данные разместим следующим образом:

$(e+1) = 1/2$, $(e + 2) = 1$, $(e + 3) = c + 1$, $(e + 4) = c + n$, $(e + 5) = 0000\ 0000\ 0000\ 0\ 27$,
$(e + 6) = d + 1\ 0000\ 0000\ 0\ 03$; $(e + 7) = \Delta$ (II),
$(d + 1) = x$,
$d + 2$, $d + 3$, $d + 4$, $d + 5$ – рабочие ячейки.

Номер $b + i$ «ячейки, хранящей $y_i$, должен быть получен во втором адресе ячейки $d$ + 2.

Логическая схема подпрограммы выбора будет иметь такой вид:




Здесь $\mathbf{P}_k$ , $\mathbf{Я}_{k+1}$ — имеют прежний смысл;

$\mathbf{П}_{k+1}$ – оператор переноса чисел;

$\mathbf{\Phi}_{k+3}$ — вычисляет номер ячейки $<x_{i*}>$, являющейся «серединой» последовательности ячеек $<x_i>$, и формирует $\mathbf{P}_{k+4}$;

$\mathbf{P}_{k+4}$ — проверяет неравенство $x<x_{i*}$;

$\mathbf{П}_{k-5}$ — записывает номер «средней ячейки на место номера первой ячейки$<x_1:>$;

$\mathbf{П}_{k+6}$ — записывает номер средней ячейки на место номера последней ячейки $<x_n>$;

$\mathbf{P}_{k+7}$ — условие конца поиска $<x_i>$;

$\mathbf{A}_{k+8}$ — вычисляет $b + i$ (II).

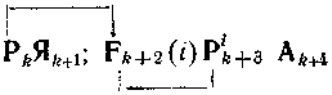
Составим подпрограмму выбора:

| | | | | | | |
|---|---|---|---|---|---|---|
| $\mathbf{P}_k$: | $a$+ 1) | $d$+ 1 | $c$+ 1 | 0000 | 0 | 03 |
| | $a$+ 2) | $a$+ 3 | $a$+ 5 | 0000 | 0 | 20 |
| | $a$+3) | $d$+ 1 | $c$+ $n$ | 0000 | 0 | 03 |
| | $a$+ 4) | $a$+ 5 | $a$+ 6 | 0000 | 0 | 20 |
| $\mathbf{Я}_{k+1}$: | $a$+ 5) | $d$+ 1 | 0000 | 0000 | 0 | 40 |
| $\mathbf{П}_{k+2}$: | $a$+ 6) | $e$+ 3 | 0001 | $d$+ 3 | 0 | 45 |
| $\mathbf{\Phi}_{k+3}$: | $a$+ 7) | $d$+ 3 | $d$+ 4 | $d$+ 2 | 0 | 01 |
| | $a$+10) | $d$+ 2 | $e$+ 1 | $d$+ 2 | 0 | 05 |
| | $a$+11) | $d$+ 2 | 7575 | $d$+ 2 | 0 | 12 |
| | $a$+12) | $d$+ 2 | $e$+ 5 | $d$+ 5 | 0 | 07 |
| | $a$+13) | $d$+ 2 | $d$+ 5 | $d$+ 5 | 0 | 14 |
| | $a$+14) | $e$+ 6 | $d$+ 5 | $a$+15 | 0 | 13 |
| $\mathbf{P}_{k+4}$: | $a$+15) | 0000 | 0000* | 0000 | 0 | 00 |
| | $a$+16) | $a$+17 | $a$+21 | 0000 | 0 | 20 |
| $\mathbf{П}_{k+5}$: | $a$+17) | $d$+ 2 | 0000 | $d$+ 3 | 0 | 13 |
| | $a$+20) | $a$+22 | $a$+22 | 0000 | 0 | 20 |
| $\mathbf{П}_{k+6}$: | $a$+21) | $d$+ 2 | 0000 | $d$+ 4 | 0 | 13 |
| $\mathbf{P}_{k+7}$: | $a$+22) | $d$+ 4 | $d$+ 3 | $d$+ 2 | 0 | 03 |
| | $a$+23) | $d$+ 2 | $e$+ 2 | 0000 | 0 | 16 |
| | $a$+24) | $a$+25 | $a$+ 7 | 0000 | 0 | 20 |
| $\mathbf{A}_{k+10}$: | $a$+25) | $d$+ 3 | $e$+ 5 | $d$+ 5 | 0 | 07 |
| | $a$+26) | $d$+ 3 | $d$+ 5 | $d$+ 3 | 0 | 14 |
| | $a$+27) | $d$+3 | $e$+ 7 | $d$+ 2 | 0 | 02 |

В ячейке $d$ + 2 получено $b + i$ (II).

В ы б о р  с п о с о б о м  с к о л ь з я щ е г о  н а ч а л а . Подпрограмма выбора способом скользящего начала весьма проста. Она отличается от подпрограммы выбора перебором тем, что не содержит оператора восстановления.

Вот ее логическая схема (т. е. логическая схема обобщенного оператора $\mathbf{P}_1$):



Исходные данные считаем размещенными точно так же, как в примере на выбор перебором. Номер $b + i$ ячейки, хранящей $y_i$, получается во втором адресе ячейки $d + 2$.

Составим подпрограмму выбора:

| | | | | | | |
|---|---|---|---|---|---|---|
| $\mathbf{P}_k$: | $a+1$) | $d+1$ | $c+1$ | 0000 | 0 | 03 |
| | $a+2$) | $a+3$ | $a+5$ | 0000 | 0 | 20 |
| | $a+3$) | $d+1$ | $c+n$ | 0000 | 0 | 03 |
| | $a+4$) | $a+5$ | $a+6$ | 0000 | 0 | 20 |
| $\mathbf{Я}_{k+1}$: | $a+5$) | $d+1$ | 0000 | 0000 | 0 | 40 |
| $\mathbf{F}_{k+2}(i)$: | $a+6$) | $a+7$ | $e+1$ | $a+7$ | 0 | 02 |
| $\mathbf{P}^i_{k+3}$: | $a+7$) | $d+1$ | $c+1$* | 0000 | 0 | 03 |
| | $a+10$) | $a+6$ | $a+11$ | 0000 | 0 | 20 |
| $\mathbf{A}_{k+4}$: | $a+11$) | $a+7$ | $e+3$ | $d+2$ | 0 | 11 |
| | $a+12$) | $d+2$ | $e+4$ | $d+3$ | 0 | 02 |

В ячейке $d + 3$ получено $b + i$ (II).

## ГЛАВА IX
## ФОРМАЛЬНЫЕ ПРЕОБРАЗОВАНИЯ ЛОГИЧЕСКИХ СХЕМ ПРОГРАММ

### § 46. Основные понятия

**1. Обозначения и простейшие формулы.** При изучении вопроса о преобразовании логических схем будем все операторы разделять на *логические* и *нелогические.*

Логический оператор будем обозначать символом $\mathbf{P}_i$, нелогический — символом $\mathbf{D}_i$; если принадлежность оператора к тому или другому классу не представляет интереса, будем обозначать оператор символом $\mathbf{Q}_i$ ($i$ — некоторый номер, т. е. положительное целое число), например $\mathbf{P}_1$, $\mathbf{P}_{20}$, $\mathbf{D}_5$, $\mathbf{Q}_{70}$ и т. д. В случаях, когда значение $i$ безразлично, будем индекс опускать.

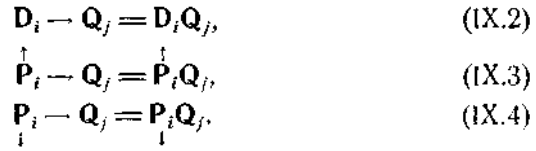
Прежде всего, условимся, что знак равенства означает, что символы, между которыми он поставлен, в любой логической схеме взаимозаменяемы.

Нами было принято условие, что если некоторый оператор $\mathbf{Q}_i$ передает управление оператору $\mathbf{Q}_j$ и эти операторы в логической схеме записаны рядом, то можно писать либо $\mathbf{Q}_i \rightarrow \mathbf{Q}_j$, либо $\mathbf{Q}_i\,\mathbf{Q}_j$.

Таким образом,

$$\mathbf{Q}_i \rightarrow \mathbf{Q}_j = \mathbf{Q}_i\mathbf{Q}_j. \qquad \text{(IX.1)}$$

К оператору может вести любое количество стрелок, но от каждого оператора идет строго установленное их количество: от нелогического оператора идет только одна стрелка, от логического оператора идут две стрелки. Конкретизируя формулу (IX.1), можно написать:



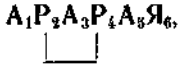
$$\mathbf{D}_i \rightarrow \mathbf{Q}_j = \mathbf{D}_i\mathbf{Q}_j, \qquad \text{(IX.2)}$$

$$\overset{\uparrow}{\mathbf{P}}_i \rightarrow \mathbf{Q}_j = \overset{\uparrow}{\mathbf{P}}_i\mathbf{Q}_j, \qquad \text{(IX.3)}$$

$$\underset{\downarrow}{\mathbf{P}_i} \rightarrow \mathbf{Q}_j = \underset{\downarrow}{\mathbf{P}_i}\mathbf{Q}_j. \qquad \text{(IX.4)}$$

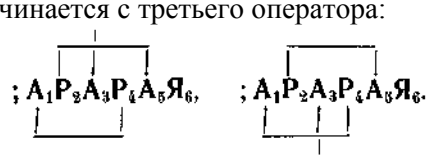
При записи формул (IX.2)—(IX.4) мы опустили стрелки, идущие от оператора $\mathbf{Q}_j$, поскольку эти формулы касаются только стрелок, идущих от оператора $\mathbf{D}_i$ или $\mathbf{P}_i$. Кроме того, мы не показали, идет ли в (IX.3) верхняя стрелка, а в (IX.4) нижняя стрелка, начинающаяся у оператора $\mathbf{P}_i$, вправо или влево, ибо это тоже не существенно в наших формулах. В дальнейшем при записи формул мы также всегда будем опускать подробности, не имеющие к этим формулам прямого отношения.

Если выполнение схемы должно начинаться не с первого оператора, перед схемой будем ставить знак «;», а оператор, с которого начинается выполнение схемы, будем отмечать ведущей к нему стрелкой, расположенной над строкой или под строкой операторов. Например, выполнение следующей схемы начинается с первого оператора;



$$\mathbf{A}_1\mathbf{P}_2\mathbf{A}_3\mathbf{P}_4\mathbf{A}_5\mathbf{Я}_6,$$

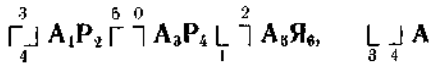
а выполнение следующих двух схем начинается с третьего оператора:



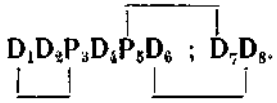
$$;\ \mathbf{A}_1\mathbf{P}_2\mathbf{A}_3\mathbf{P}_4\mathbf{A}_5\mathbf{Я}_6, \qquad ;\ \mathbf{A}_1\mathbf{P}_2\mathbf{A}_3\mathbf{P}_4\mathbf{A}_5\mathbf{Я}_6.$$

При записи логических схем с помощью «уголков» (см. § 32) в случае, когда выполнение схемы начинается не

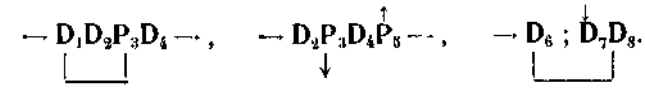
с первого оператора, будем в начале схемы ставить знак $\overset{i}{\Gamma}$ или $\underset{i}{\llcorner}$, где $i$ — номер оператора, с которого начинается выполнение схемы, а перед этим оператором — знак $\overset{0}{\urcorner}$ или $\underset{0}{\lrcorner}$. При такой записи последние две схемы будут иметь следующий вид:



Группу операторов,стоящих в логической схеме подряд, вместе с примыкающими к ним началами и концами стрелок, мы будем называть операторным выражением. Рассмотрим, например, логическую схему



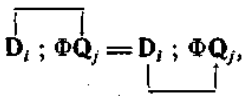
Для этой схемы $\rightarrow \mathbf{D}_1\,\mathbf{D}_2$ является выражением. Точно так же выражениями будут:



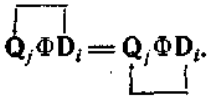
В тех случаях, когда конкретный вид выражения для нас безразличен, будем его обозначать одной из заглавных греческих букв, например $\Phi$ или $\Psi$ .

Ранее было принято условие, что запись $\mathbf{Q}_i$; $\mathbf{Q}_j$ обозначает, что операторы $\mathbf{Q}_i$ и $\mathbf{Q}_j$ стоят в логической схеме рядом, но передачи управления от оператора $\mathbf{Q}_i$ оператору $\mathbf{Q}_j$ нет. Символ «;» (точка с запятой) следует относить к оператору $\mathbf{Q}_i$.

В дальнейшем, при преобразованиях логических схем знаки операторов могут по определенным правилам меняться между собой местами. Чтобы при этом не возникало недоразумений, условимся, что стрелку, идущую от нелогического оператора к другому оператору, расположенному не непосредственно справа от первого, можно располагать как над строкой операторов, так и под нею. В принятых нами обозначениях для операторных выражений это соглашение можно представить в виде следующих формул

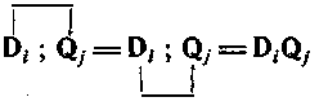


(IX.5)



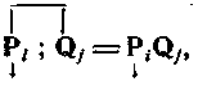
(IX.6)

Условимся также, что

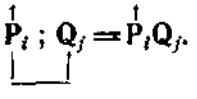


(IX.7)

и что

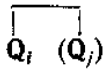


(IX.8)



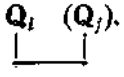
(IX.9)

Для удобства записи формул мы будем в дальнейшем в тех случаях, когда направление стрелки (вперед или назад) безразлично, а конец ее почему-либо указать необходимо, обозначать символом



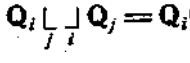
(IX.10)

как $\overline{\mathbf{Q}_i\,\Phi\,\mathbf{Q}_j}$, так и $\overline{\mathbf{Q}_j\,\Psi\,\mathbf{Q}_i}$. В аналогичном смысле будем употреблять символ



(IX.11)

При использовании полускобок или «уголков» формула (IX.1) запишется так:



(IX.12)

$$\mathbf{Q}_i \overset{j}{\lceil} \overset{i}{\urcorner} \mathbf{Q}_j = \mathbf{Q}_i \mathbf{Q}_j. \qquad \text{(IX.13)}$$

Формулы (IX.5) и (IX.6) примут следующий вид:

$$\mathbf{D}_i \overset{j}{\lceil} \Phi \overset{i}{\urcorner} \mathbf{Q}_j = \mathbf{D}_i \underset{j}{\llcorner} \Phi \underset{i}{\lrcorner} \mathbf{Q}_j,$$

$$\overset{i}{\urcorner} \mathbf{Q}_j \Psi \mathbf{D}_i \overset{j}{\lceil} = \underset{i}{\lrcorner} \mathbf{Q}_j \Psi \mathbf{D}_i \underset{j}{\llcorner}.$$

Формулы (IX.7), (IX.8) и (IX.9) можно объединить в две формулы:

$$\mathbf{Q}_i \underset{j}{\llcorner} ; \underset{i}{\lrcorner} \mathbf{Q}_j = \mathbf{Q}_i \mathbf{Q}_j,$$

$$\mathbf{Q}_i \overset{j}{\lceil} ; \overset{i}{\urcorner} \mathbf{Q}_j = \mathbf{Q}_i \mathbf{Q}_j.$$

Заменяя в формулах (IX.12) и (IX.13) $\mathbf{Q}_i$ знаком $\mathbf{P}_i$, можно написать:

$$\mathbf{P}_i \overset{k}{\underset{j}{\sqsubset}} \underset{i}{\lrcorner} \mathbf{Q}_j = \mathbf{P}_i \overset{k}{\lceil} \mathbf{Q}_j,$$

$$\mathbf{P}_i \overset{j}{\underset{k}{\sqsubset}} \overset{i}{\urcorner} \mathbf{Q}_j = \mathbf{P}_i \underset{k}{\llcorner} \mathbf{Q}_j.$$

Принятые нами выше условные записи (IX.10) и (IX.11) при использовании полускобок будем изображать так:

$$\overset{\sqcap}{\mathbf{Q}_i} \quad (\mathbf{Q}_j) = \mathbf{Q}_i \overset{j}{\lceil} (\overset{i}{\urcorner} \mathbf{Q}_j),$$
$$\underset{\sqcup}{\mathbf{Q}_i} \quad (\mathbf{Q}_j) = \mathbf{Q}_i \underset{j}{\llcorner} (\underset{i}{\lrcorner} \mathbf{Q}_j).$$

В частности, например,

$$\overset{\sqcap}{\mathbf{P}_i} \quad (\mathbf{Q}_j)(\mathbf{Q}_k) = \mathbf{P}_i \overset{j}{\underset{k}{\sqsubset}} (\overset{i}{\urcorner} \mathbf{Q}_j)(\underset{i}{\lrcorner} \mathbf{Q}_k).$$

**2. Основные логические переменные.** Логические операторы схем производят проверку ряда условий. Систему условий (принимаемых нами за элементарные), через которые можно выразить (используя для этого логические связи) все условия, проверяемые логическими операторами, обозначим, вместе с их значениями истинности, символами

$$p_1, p_2 \ldots, p_m.$$

Двоичные переменные $p_1, p_2, \ldots, p_m$ будем называть *основными логическими переменными* логической схемы.

Рассмотрим в качестве примера логическую схему программы составления таблицы значений функции $y(x)$, заданной неявно соотношением

$$F(x,y) = 0. \qquad \text{(IX.14)}$$

Пусть $x_1, x_2, \ldots, x_n$ — значения независимого переменного, для которых требуется найти значения функции $y_1 y_2, \ldots, y_n$. Предположим, что для решения задачи уравнение (IX.14) было приведено к виду

$$y = f(x,y),$$

допускающему решение методом итераций; кроме того, было найдено число $\alpha$, являющееся удовлетворительным начальным приближением первого искомого значения $y_1$ функции $y(x)$. В качестве начального приближения для каждого следующего искомого значения функции $y$ решено брать предыдущее вычисленное ее значение.

Таким образом, решение задачи сведено к вычислениям по формуле

$$(y_k)_{i+1} = f[x_k, (y_k)_i] \qquad (i = 0, 1, 2, 3, \ldots), \qquad \text{(IX.15)}$$

производимым до тех пор, пока не начнет выполняться неравенство

$$|(y_k)_{i+1} - (y_k)_i| < \varepsilon \qquad \text{(IX.16)}$$

где $\varepsilon$— некоторое (тоже заранее выбранное) положительное число, характеризующее точность, необходимую при вычислении величин $y_1, y_2, \ldots, y_n$. Величину $(y_k)_{i+1}$, удовлетворяющую неравенству (IX.16), будем считать искомым значением функции $y_k$. Как было уже сказано,

$$\left.\begin{aligned} (y_1)_0 &= a, \\ (y_{k+1})_0 &= y_k \end{aligned}\right\} \qquad \text{(IX.17)}$$

Предположим, что числа $x_1, x_2, \ldots, x_n$ расположены в памяти машины в последовательных ячейках $b+1, b+2, \ldots, b+n$. В эти же ячейки будут записываться результаты вычислений: $y_k$ будет записано в ячейку, содержавшую ранее $x_k$.. Логическая схема программы, осуществляющей описанный выше вычислительный процесс, может быть следующей:

$$\overset{\sqcap}{\mathbf{\Pi}_0 \mathbf{A}_1 \mathbf{\Pi}_2 \mathbf{A}_3^k \mathbf{P}_4 \mathbf{\Pi}_5^k \mathbf{F}_6 (k) \mathbf{P}_7 \mathbf{A}_8 \mathbf{\Pi}_9 \mathbf{Я}_{10}}. \qquad \text{(IX.18)}$$

Здесь $\mathbf{\Pi}_0$ — ввод программы и исходных данных в память; $\mathbf{A}_1$ — перевод исходных данных в двоичную систему

счисления; $\mathbf{П}_2$ — перенос числа из ячейки *b,* хранившей вначале число $(y_1)_0$, в стандартную ячейку r; $\mathbf{A}_3^k$ — оператор, ведущий вычисления по формуле

$$f[x_k, (r)] = (b); \qquad \text{(IX.19)}$$

$\mathbf{P}_4$— логический оператор, проверяющий выполнение неравенства

$$|(b) - (r)| < \varepsilon, \qquad \text{(IX.20)}$$

которое после занесения в ячейки *b* и *r* соответствующих чисел (после первого же выполнения оператора $\mathbf{A}_3^k$) делается и в течение всей работы схемы остается равносильным условию (IX.16)

$$|(y_k)_{i+1} - (y_k)_i| < \varepsilon;$$

$\mathbf{П}^k{}_5$ — оператор переноса числа (*b*) в ячейку *b* + *k* (т. е. числа $y_k$ в ячейку, хранившую ранее $x_k$); $\mathbf{F}_6$ (*k*) — оператор переадресации; $\mathbf{P}_7$ — логический оператор, проверяющий выполнение условия

$$k \neq n; \qquad \text{(IX.21)}$$

$\mathbf{A}_8$ — перевод результатов из двоичной в десятичную систему счисления; $\mathbf{П}_9$ — выдача результатов на перфокарты; $\mathbf{Я}_{10}$ — останов.

Обозначим условие (IX.16), выполнение которого проверяет в логической схеме (IX.18) оператор $\mathbf{P}_4$, а также значение истинности этого условия символом $p_1$. Условие (IX.21), проверяемое оператором $\mathbf{P}_7$, а также его значение истинности обозначим символом $p_2$.

Очевидно, что $p_1$ и $p_2$ — двоичные переменные, являющиеся *основными логическими переменными* нашей схемы.

Ясно, что до начала вычислений говорить о значении истинности условия $p_1$ бессмысленно, ибо еще не вычислена величина $(y_1)_1$, позволяющая в первый раз проверить справедливость неравенства (IX.16). Значение истинности условия $p_2$ можно указать. До начала вычислений $k = 1$, и будем считать, что $n>1$.

Таким образом, до начала выполнения программы и соответственно логической схемы

$p_1$ — не имеет значения,
$p_2 = 1$.

Оператор $\mathbf{П}_0$ не может изменить значений $p_1$ и $p_2$, он всего лишь производит ввод исходных данных и программы в память машины. Следовательно, после выполнения $\mathbf{П}_0$ и перед выполнением $A_1$

$p_1$ — не имеет значения,
$p_2 = 1$

Точно так же после выполнения $\mathbf{A}_1$ и перед выполнением $\mathbf{П}_2$

$p_1$ — не имеет значения,
$p_2 = 1$

После выполнения $\mathbf{П}_2$ условие $p_1$ по-прежнему не имеет определенного значения, хотя условие (IX.20) будет иметь значение истинности 1. Однако нужно иметь в виду, что (IX.20) является лишь случайной формой условия (IX.16). При несколько ином распределении памяти, но при той же самой логической схеме программы условие (IX.16) могло бы иметь другую конкретную форму. Поэтому считаем, что и после выполнения $\mathbf{П}_2$

$p_1$ — не имеет определенного значения,
$p_2 = 1$

Затем выполняется оператор $\mathbf{A}_3^k$. Теперь условие $p_1$ приобретает смысл. $\mathbf{A}_3^k$ не может изменить условия $p_2$. Таким образом, может оказаться, например, что в результате работы $\mathbf{A}_3^k$

$p_1$=0,
$p_2$=1.

Оператор $\mathbf{P}_4$ заведомо не меняет значений $p_1$ и $p_2$, а лишь передает управление в направлении, зависящем от этих значений. В данном случае работа оператора $\mathbf{P}_4$ определяется тем, что $p_1 = 0$. Происходит передача управления оператору $\mathbf{П}_2$.

Перед работой $\mathbf{П}_2$ переменные $p_1$ и $p_2$ сохраняют прежние значения. $\mathbf{П}_2$ не может изменить этих значений. Следовательно, перед вторичной работой оператора $\mathbf{A}_3^k$ имеем:

$p_1 = 0$,
$p_2 = 1$.

После выполнения оператора $\mathbf{A}_3^k$ может оказаться, что

$p_1 = 1$
$p_2 = 1$

в результате чего $\mathbf{P}_4$, не изменяя значений основных логических переменных, передает управление оператору $\mathbf{П}_3^{\kappa}$

Можно было бы и далее описать процесс выполнения программы и отвечающей ей логической схемы (IX.18) до его окончания в результате получения равенства $k = n$, при котором $p_2 = 0$.

Рассмотрев приведенное описание выполнения логической схемы, можно сделать следующие замечания:

1. Перед выполнением первого оператора (или даже каждого из нескольких первых операторов) схемы некоторые из основных логических переменных могут не иметь определенного значения. Отвечающая схеме программа должна быть построена так, чтобы это обстоятельство не помешало ее нормальному выполнению.

2. Нелогические операторы, вообще говоря, м о г у т и з м е н я т ь значения основных логических переменных, тогда как логические операторы з а в е д о м о их не изменяют.

3. Некоторые из нелогических операторов не влияют на значения основных логических переменных. Логическая схема не содержит в себе указаний о том, какие именно нелогические операторы могут изменять значения и каких логических переменных, — эти сведения, если неизвестна программа, д о л ж н ы быть з а д а н ы д о п о л н и т е л ь н о . Не имея таких сведений, нельзя проследить за выполнением логической схемы.

4. Если выписать последовательность наборов значений, принимаемых основными логическими переменными перед выполнением каждого нелогического оператора программы, то с помощью этой последовательности легко воспроизвести отвечающее ей выполнение логической схемы.

Сделав эти выводы, перейдем к некоторым общим формулировкам. Мы будем рассматривать логические схемы, которые состоят из логических и нелогических операторов.

**3. Распределение сдвигов схемы.** Наши дальнейшие рассуждения будут относиться только к таким логическим схемам, логические операторы которых не изменяют значений переменных $p_1$, $p_2$, $p_m$, тогда как в процессе выполнения нелогических операторов эти значения могут изменяться. Будем предполагать также, что каждый нелогический оператор входит в схему только один раз. Это не значит, что в схеме не может быть одинаковых нелогических операторов. Просто одинаковые нелогические операторы, стоящие в разных местах схемы, должны быть обозначены различными символами.

Кроме того, ограничимся рассмотрением схем, которые в процессе своего выполнения остаются неизменными, т.е. в которых операторы не возникают и не исчезают и стрелки, идущие от операторов, не подвергаются изменениям (например, к схемам программ, в которых используются условные переходы второго рода, т.е. команды с кодом операции 27, дальнейшие результаты не применимы непосредственно).

В рассмотренном выше примере логической схемы были использованы стандартные обозначения операторов. При дальнейшем изучении логических схем мы будем их записывать в символах $\mathbf{D}_i$ и $\mathbf{P}_i$, так как для всех дальнейших рассуждений безразличен конкретный смысл операторов и имеет значение лишь их принадлежность к классу нелогических или логических операторов. Кроме того, выражения, состоящие из нелогических операторов и представляющие собой обобщенные операторы (см. § 32), удобно принимать за один оператор. Это делает логическую схему более короткой.

П р и м е р 1. При переходе к обозначениям $\mathbf{D}_i$ и $\mathbf{P}_i$ приведенная выше схема (IX.18) примет такой вид:

(IX.22)

Здесь

$$\mathbf{D}_1 = \mathbf{П}_0\mathbf{A}_1, \qquad \mathbf{D}_2 = \mathbf{П}_2\mathbf{A}^k{}_3, \qquad \mathbf{P}_3 = \mathbf{P}_4 \qquad \mathbf{D}_4 = \mathbf{П}^k{}_5\mathbf{F}_6\,(k), \qquad \mathbf{P}_5 = \mathbf{P}_7, \qquad \mathbf{D}_6 = \mathbf{A}_8\mathbf{П}_9\mathbf{Я}_{10}.$$

Каждому нелогическому оператору $\mathbf{D}_1$ поставим в соответствие совокупность тех из основных логических переменных, которые могут быть изменены в процессе выполнения этого оператора. Эту совокупность будем обозначать символом $P_i$ и называть *сдвигом* основных логических переменных, соответствующим оператору $\mathbf{D}_i$.

Множество всех совокупностей $P_i$, относящихся к логической схеме, будем называть *распределением сдвигов.*

П р и м е р 2. Например, для схемы (IX.22) распределение сдвигов имеет такой вид:

$$P_1 = \{\ \},$$
$$P_2 = \{p_1\},$$
$$P_4 = \{p_2\},$$
$$P_6 = \{\ \},$$

где { } — пустая совокупность, т. е. операторы $\mathbf{D}_1$ и $\mathbf{D}_6$ в этой схеме не могут изменять основных логических переменных, а операторы $\mathbf{D}_2$ и $\mathbf{D}_4$ могут изменять (но не обязательно при каждом выполнении) только соответствующие переменные $p_1$ и $p_2$.

**4. Выполнение схемы по заданной последовательности наборов значений основных логических переменных.** Символами $\Delta_0$, $\Delta_1$, $\Delta_2$, ...будем обозначать наборы значений основных логических переменных. Так как каждая из переменных принимает только два значения (0 или 1),то число различных между собой наборов равно $2^m$. Для краткости и удобства мы записываем наборы значений переменных $p_1$, $p_2$, ..., $p_m$ в виде $m$-значных двоичных чисел, опуская запятые.

П р и м е р 3. Если система основных логических переменных состоит из $p_1$, $p_2$, $p_3$,то $m = 3$ и число различных наборов будет равно $2^3 = 8$. Приводим эти наборы:

1. $p_1$=0, $p_2$=0, $p_3$=0; $\Delta_0$=000;

2. $p_1=0,\quad p_2=0,\quad p_3=1;\quad \Delta_0=001;$
3. $p_1=0,\quad p_2=1,\quad p_3=0;\quad \Delta_0=010.$

Остальные наборы будут следующими:

В п. 2 настоящего параграфа мы рассмотрели выполнение логической схемы вместе с выполнением п

$\Delta_3 = 011;\quad \Delta_4 = 100;\quad \Delta_5 = 101;\quad \Delta_6 = 110;\quad \Delta_7 = 111.$программы, которой она соответствовала, и наблюдали замену одного набора значений основных логических переменных другим при выполнении нелогических операторов (иногда последующий набор оказывался тождественным предыдущему). При этом перед нами как бы проходила последовательность наборов значений основных логических переменных, соответствующая данному выполнению схемы.

Рассмотрим теперь логическую схему, не связывая ее с определенной программой, но считая, что заданы ее логические операторы в виде функций основных логических переменных и некоторое распределение сдвигов. Такой схеме могут соответствовать различные программы, так что ее следует рассматривать без учета содержания нелогических операторов. По виду схемы уже нельзя судить, изменяет ли тот или другой нелогический оператор значения основных логических переменных, известно только, какие из них он может изменять. Нельзя также, вообще говоря, определить, в каких направлениях будет происходить передача управления логическими операторами, то есть нельзя, отправляясь от начального набора значений основных логических переменных, проследить выполнение схемы. Однако, имея последовательность наборов значений основных логических переменных, можно по ней воспроизвести выполнение схемы.

Если в начале выполнения логической схемы некоторые из основных логических переменных не имеют определенных значений, то им будем произвольно приписывать значения, сохраняющиеся до тех пор, пока не выполнится нелогический оператор, который придаст этим логическим переменным определенные значения.

Пусть $E$ — некоторая конечная или бесконечная последовательность наборов:

$$E = \Delta_{i_1}, \Delta_{i_2}, \Delta_{i_3}, \ldots \qquad \text{(IX.23)}$$

Чтобы в дальнейшем не различать случаи, когда последовательность $E$ конечна и когда бесконечна, будем считать эту последовательность наборов всегда бесконечной. Для этого последний из наборов, входящих в $E$, будем считать повторяющимся бесконечное число раз.

Чтобы воспроизвести по $E$ процесс выполнения логической схемы, надо поступать в соответствии со следующим правилом:

Правило I. В начале процесса отмеченными считаются первый набор из $E$ и тот оператор схемы, с которого начинается ее выполнение (т. е. либо первый, либо указанный стрелкой). Далее поступаем так:

1. Рассматриваем отмеченный оператор и проверяем, является ли он логическим. Если да, то переходим к пункту 2. Если нет — к пункту 3.

2. С помощью отмеченного набора вычисляем значение истинности условия, проверяемого отмеченным логическим оператором. В соответствии с этим значением определяем, куда передает управление в схеме отмеченный оператор.

После этого считаем отмеченным тот оператор, которому передается управление, и не меняем отмеченный набор. Переходим снова к пункту 1.

3. Выписываем отмеченный нелогический оператор. Переходим к пункту 4.

4. Проверяем, передает ли выписанный оператор управление какому-либо оператору схемы. Если да, то переходим к пункту 5. Если нет — процесс окончен.

5. Считаем отмеченными те операторы, которым передается управление от выписанного, и следующий по порядку набор в $E$. Переходим к пункту 1.

Предположим, что при выполнении схемы для некоторой последовательности наборов

$$E = \Delta_{i_1}, \Delta_{i_2}, \Delta_{i_3}, \ldots$$

справедливо следующее условие: каждый набор, отмеченный после выписывания некоторого нелогического оператора, и набор, который ему непосредственно предшествует в последовательности $E$, содержат *одинаковые* значения тех основных логических переменных, которые не принадлежат сдвигу упомянутого нелогического оператора. В этом случае $E$ называют *допустимой последовательностью наборов.*

**5. Значение схемы.** При выполнении схемы для произвольной последовательности наборов

$$E = \Delta_{i_1}, \Delta_{i_2}, \Delta_{i_3}, \ldots$$

в соответствии с правилом I получается строка, в которой выписаны нелогические операторы схемы в том порядке, в котором они выполнялись. Эта строка нелогических операторов называется *значением схемы,* отвечающим последовательности $E$.

Выполняя логическую схему для заданной допустимой последовательности $E$ в соответствии с правилом I, мы получим так называемое *допустимое значение* логической схемы.

При выполнении схемы возможны три случая:

а) количество символов в значении схемы конечно и процесс ее выполнения заканчивается выписыванием оператора, не передающего никуда управления;

б) количество символов в значении схемы конечно, но процесс выполнения бесконечен; в этом случае происходит,

говоря «машинным» языком, зацикливание логической схемы на логических операторах; тогда говорят, что *значение схемы заканчивается пустым периодом,* и отмечают это, приписывая в конце значения схемы символ ( );

в) количество символов в значении схемы бесконечно.

Пример 4. Вернемся к рассмотрению схемы (IX.22)

имеющей распределение сдвигов

$$P_1 = \{\quad\},\qquad P_2 = \{p_1\},\qquad P_4 = \{p_2\},\qquad P_6 = \{\quad\}.$$

Напомним, что оператор $\mathbf{P}_3$ проверяет значение истинности логической переменной $p_1$, а $\mathbf{P}_5$ — логической переменной $p_2$. Различных между собой наборов значений основных логических переменных для этой схемы существует всего четыре, именно:

$$\Delta_0 = 00,\qquad \Delta_1 = 01,\qquad \Delta_2 = 10,\qquad \Delta_3 = 11.$$

Пусть дана допустимая последовательность наборов

$$E = \Delta_1\,\Delta_1\,\Delta_1\,\Delta_3\,\Delta_2\,\Delta_2\,\Delta_2\,\Delta_2\ldots$$

Требуется построить отвечающее ей значение схемы (IX.22).

Отмечаем первый набор из *E,* рассматриваем первый из операторов схемы. Он является нелогическим. Выписываем его:

$$\mathbf{D_1}.$$

Отмечаем второй набор из *E* и рассматриваем очередной (т. е. $\mathbf{D}_2$) оператор схемы. Вторым набором из *E* является опять Δ,. Иначе и не могло быть, ибо $P_1 = \{\ \}$, т. е. $\mathbf{D}_1$ не изменяет значений основных логических переменных. Оператор $\mathbf{D}_2$ — нелогический. Выписывая его, имеем:

$$\mathbf{D}_1\mathbf{D}_2$$

Отмечаем следующий набор из *E.* Это опять $\Delta_1$. Оператор $\mathbf{D}_2$ мог изменить значение переменной $p_1$ но в данном случае не изменил. Рассматриваем очередной оператор $\mathbf{P}_3$. Это — логический оператор, проверяющий условие $p_1$. Из отмеченного набора получаем $p_1 = 0$. Следовательно, $\mathbf{P}_3$ передает управление по нижней стрелке опять оператору $\mathbf{D}_2$. Выписывая $\mathbf{D}_2$, имеем:

$$\mathbf{D}_1\mathbf{D}_2\mathbf{D}_2.$$

Отмечаем следующий набор из *E* — это $\Delta_3$, и рассматриваем очередной оператор — $\mathbf{P}_3$. Это — логический оператор. Он передает управление оператору $\mathbf{D}_4$ (так как в наборе $\Delta_3$ $p_1 = 1$), не изменяя значений переменных $p_1$ и $p_2$. Выписывая $D_4$, имеем:

$$\mathbf{D}_1\mathbf{D}_2\mathbf{D}_2\mathbf{D}_4.$$

Отмечаем следующий набор из *E* — это $\Delta_2$. Рассматривая очередной оператор схемы — $\mathbf{P}_5$, видим, что он передает управление оператору $\mathbf{D_6}$ (так как в $\Delta_2$ $p_2 = 0$). Выписывая $\mathbf{D}_6$, имеем:

$$\mathbf{D}_1\mathbf{D}_2\mathbf{D}_2\mathbf{D}_4\mathbf{D}_6.$$

Отмечая следующий набор из *E,* видим, что это опять $\Delta_2$. Этот набор дает нам значения основных логических переменных после выполнения всей схемы. Выполнение схемы окончилось потому, что оператор $\mathbf{D}_6$ никакому оператору схемы не передает управления (т. е. является последним). Остальные наборы из *E* все одинаковы.

Таким образом, допустимое значение схемы (IX.22) — назовем ее буквой *S*, — отвечающее допустимой последовательности *E,* имеет такой вид:

$$\boldsymbol{Z}_S(E) = \mathbf{D}_1\mathbf{D}_2\mathbf{D}_2\mathbf{D}_4\mathbf{D}_6.$$

Символ $Z_S(E)$ следует читать так: *значение схемы S, отвечающее последовательности наборов E.*

Пример 5. Рассмотрим теперь допустимую для схемы (IX.22) последовательность наборов $E_1$:

$$E_1 = \Delta_1\Delta_1\Delta_1\Delta_3\Delta_3\Delta_3\ \ldots$$

($\Delta_3$ повторяется бесконечное число раз).

Выполняя логическую схему в соответствии с $E_1$, имеем:

$$Z_S(E_1) = \mathbf{D}_1\mathbf{D}_2\mathbf{D}_2\mathbf{D}_4\mathbf{D}_2\mathbf{D}_4\ \ldots$$

Полученное значение логической схемы бесконечно.

Пример 6. Для получения примера значения схемы с пустым периодом рассмотрим логическую схему $S_1$:

(IX.24)

Пусть распределение сдвигов этой схемы имеет такой вид:

$$P_1 = \{p_1,\ p_2\},\qquad P_3 = \{p_1\},\qquad P_5 = \{\ \}.$$

Оператор $P_2$ пусть проверяет сложное условие $p_1 \wedge p_2$, а оператор $\mathbf{P}$ — условие $p_1 \vee p_2$. Дана допустимая последовательность наборов

$$E = \Delta_0\Delta_3\Delta_3\Delta_1\Delta_1\Delta_1\ \ldots$$

Отвечающее ей значение схемы, в чем легко убедиться проверкой, имеет такой вид

$$Z_{S_1}(E) = \mathbf{D}_1\mathbf{D}_3\mathbf{D}_3\ (\ ).$$

**6. Равносильность схем.** Рассмотрим две логические схемы $S_1$ и $S_2$, записанные так, что символы $\mathbf{D}_i$, присутствующие сразу в обеих этих схемах, имеют одинаковый смысл, т. е. обозначают одни и те же операторы.

Говорят, что схемы $S_1$ и $S_2$ *равносильны,* если для всякой последовательности наборов, допустимой хотя бы для одной из них, значения этих схем совпадают.

Равносильность схем записывается с помощью знака равенства. Таким образом,

$$S_1 = S_2,$$

если для всякой последовательности наборов *E*, допустимой для $S_1$ или для $S_2$, значения $Z_{S_1}(E)$ и $Z_{S_2}(E)$ графически тождественны. Определение равносильности логических схем не является эффективным в том смысле, что для установления равносильности путем непосредственного сравнения значений этих схем часто требуется

произвести бесконечное количество действий. Тем не менее это определение достаточно для вывода ряда формул, с помощью которых логические схемы могут быть преобразованы в равносильные им новые схемы.

## § 47. Преобразование логических связей в схемах

Выражение Φ называется *равносильным* выражению Ψ, если при замене в любой логической схеме выражения Φ выражением Ψ получается логическая схема, равносильная первоначальной.

То есть

$$\Phi = \Psi,$$

если

$$S(\Phi) = S(\Psi).$$

**1. Подчиненность оператора логической функции.** Говорят, что оператор $\mathbf{Q}_i$ некоторой логической схемы при данном распределении сдвигов *подчинен логической функции* $\alpha(p_1, p_2, \ldots, p_m)$, если всякий набор $\Delta_s$, при котором управление передается оператору $\mathbf{Q}_i$, обращает функцию α в единицу.

Пусть $\nabla = \{\Delta_{S_1}, \Delta_{S_2}, \ldots, \Delta_{S_k}\}$ есть совокупность всех различных между собой наборов, при которых может выполняться оператор $\mathbf{Q}_i$ (при этом $k \leq 2^m$). Тогда функция α, которой подчинен оператор $\mathbf{Q}_i$, должна удовлетворять условию

$$a(\Delta_{S_i}) = 1 \qquad (i = 1, 2, \ldots, k). \qquad \text{(IX.25)}$$

Здесь $\alpha(\Delta_{S_i})$ есть значение функции α при тех значениях основных логических переменных, которые образуют $\Delta_{S_i}$.

Если $k < 2^m$, то существуют наборы, не входящие в $\nabla$. При этом существует несколько функций, удовлетворяющих на наборах, принадлежащих $\nabla$, условию (IX.25) и отличающихся между собой хотя бы на одном из наборов, не принадлежащих $\nabla$. Оператор $\mathbf{Q}_i$ подчинен любой из этих функций.

Заметим, что в частности, если оператор $\mathbf{Q}_i$ подчинен логической функции α, то, какова бы ни была логическая функция γ, он подчинен также логическим функциям $\alpha \vee \gamma$ и $\gamma \to \alpha$ (так как эти функции при $\alpha = 1$ принимают значения $1 \vee \gamma = 1$, $\gamma \to 1 = 1$).

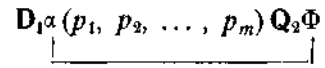
Если $\mathbf{Q}_i$ подчинен каждой из логических функций α и β, то легко сообразить, что он подчинен также и функциям $\alpha \wedge \beta$ и $\alpha \sim \beta$.

Отметим еще одно простое, но важное свойство понятия подчиненности. Предположим, что оператор $\mathbf{Q}_i$ подчинен логической функции α. Если существует логическая функция β такая, что $\alpha \to \beta = 1$ (достаточно даже, чтобы высказывание $\alpha \to \beta$ являлось истинным т о л ь к о перед выполнением оператора $\mathbf{Q}_i$), то оператор $\mathbf{Q}_i$ подчинен функции β.

Действительно, высказывание $\to\beta$ может быть ложным только в случае, когда $\alpha = 1$, $\beta = 0$. Так как это высказывание является истинным, то равенство $\beta = 0$ не может иметь места тогда, когда $\alpha = 1$, т. е. перед выполнением оператора $\mathbf{Q}_i$ обязательно $\beta = 1$ (ибо при этом $\alpha = 1$), откуда и вытекает подчиненность оператора $\mathbf{Q}_i$ логической функции β.

Поясним определение подчиненности оператора логической функции на примерах. Для удобства мы будем в дальнейшем вместо логических операторов вписывать в схему логические функции, проверяемые ими. В случае использования уголков мы будем заключать эти логические функции в скобки и снабжать индексом, обозначающим номер оператора.

I. В логической схеме

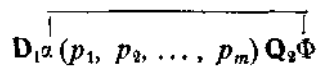



перед выполнением оператора $\mathbf{Q}_2$ всегда должно быть

$$\alpha(p_1, p_2, \ldots, p_m) = 1.$$

Следовательно, при любом распределении сдвигов в этой схеме оператор $\mathbf{Q}_2$ подчинен логической функции $\alpha(p_1, p_2, \ldots, p_m)$.

II. В логической схеме



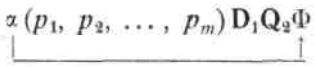

перед выполнением оператора $\mathbf{Q}_2$ всегда

$$\alpha(p_1, p_2, \ldots, p_m) = 0.$$

Следовательно, оператор $\mathbf{Q}_2$ подчинен логической функции

$$\alpha(p_1, p_2, \ldots, p_m).$$

III. В логической схеме




нельзя считать, что $\mathbf{Q}_2$ подчинен логической функции $\alpha(p_1,p_2, .., p_m)$, так как оператор $\mathbf{D}_1$ при своем выполнении может изменить некоторые из основных логических переменных (именно те, которые указаны в $P_1$) так, что перед выполнением $\mathbf{Q}_2$ не всегда будет $\alpha = 1$. Однако легко сообразить, что перед выполнением $\mathbf{Q}_2$ всегда будет равна единице логическая функция $\beta\ (p_1, p_2, ..., p_m)$, получаемая из $\alpha\ (p_1, p_2, ..., p_m)$ по формуле

$$\beta\ (p_1, p_2,..., p_m) = \max_{P_1} \alpha\ (p_1, p_2, \dots , p_m).$$

Функция β обладает следующим свойством. Она принимает значение 1 каждый раз, когда $\alpha = 1$, и сохраняет это значение даже после того, как будут изменены любым образом логические переменные, указанные в $P_1$.

Назовем логические переменные, не указанные в $P_1$, *активными* по отношению к β. Приведенная формула показывает, что функция β получается из функции α следующим образом. Для каждой комбинации значений активных логических переменных перебирают все возможные комбинации значений остальных логических переменных. В качестве значения функции β берут наибольшее получающееся при этом значение функции α.

П р и м е р 7. Если $\alpha\ (p_1\ p_2, p_3) = p_1 \wedge p_2\ \ p_3$ причем $P=\{p_1\}$, то активными переменными будут $p_2$, $p_3$. Чтобы найти $\beta\ (p_1\ p_2, p_3)$, фиксируют значения активных логических переменных и перебирают значения неактивных (в данном случае $p_1$).

1. Пусть $p_2 = 0, p_3 = 0$. Тогда

$$\alpha\ (p_1, 0, 0) = p_1 \wedge 0 \vee 0 = 0,$$

следовательно,

$$\beta\ (p_1, 0, 0) = 0.$$

2. Пусть $p_2 = 0, p_3 = 1$. Тогда

$$\alpha\ (p_1, 0, 1) = p_1 \wedge 0 \vee 1 = 1,$$

следовательно,

$$\beta\ (p_1, 0, 1)=1.$$

3. Пусть $p_2 = 1, p_3 = 0$. Тогда

$$\alpha\ (p_1, 1, 0) = p_1 \wedge 1 \vee 0 = \begin{cases} 1 \ при \quad p_1 = 1, \\ 0 \ при \quad p_1 = 0, \end{cases}$$

следовательно,

$$\beta(p_1, 1, 0) = 1.$$

4. Пусть $p_2 = 1, p_3 = 1$. Тогда

$$\alpha\ (p_1, 1, 1) = p_1 \wedge 1 \vee 1 = 1,$$

следовательно,

$$\beta(p_1, 1, 1) = 1$$

Из сопоставления полученных значений функции $\beta\ (p_1, p_2, p_3)$

$$\beta\ (p_1, 0, 0) = 0,$$
$$\beta\ (p_1, 0, 1) = 1,$$
$$\beta\ (p_1, 1, 0) = 1,$$
$$\beta\ (p_1, 1, 1) = 1$$

видим, что эта функция зависит только от $p_2$ и $p_3$, причем

$$\beta(p_1, p_2, p_3) = \max_{p_1} p_1 \wedge p_2 \vee p_3 = p_2 \vee p_3.$$

В общем виде замеченное нами свойство можно сформулировать так:

Если нелогический оператор $\mathbf{D}_j$ подчинен логической функции $\alpha\ (p_1, p_2, ..., p_m)$ и передает управление оператору $\mathbf{Q}_i$, то последний подчинен логической функции

$$\beta\ (p_1, p_2, ..., p_m) = \max_{Pj} \alpha\ (p_1, p_2, .... p_m),$$

где $P_j$ — совокупность основных логических переменных, значения которых могут измениться при выполнении $\mathbf{D}_j$. Рассмотрим выражение

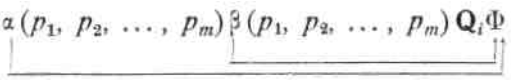
$$\mathbf{D}_j\ \mathbf{D}_{j+1}\ \mathbf{D}_{j+2} \dots \mathbf{D}_{j+k}\ \mathbf{Q}_{j+k+1},$$

где оператор $\mathbf{D}_j$ подчинен логической функции $\alpha\ (p_1, p_2, ..., p_m)$. Пусть $P_j$, $P_{J+1}$, $P_{j+2}$, …, $P_{j+k}$ — сдвиги основных логических переменных, отвечающие нелогическим операторам $\mathbf{D}_j$, $\mathbf{D}_{j+1}$, $\mathbf{D}_{j+2}$,..., $\mathbf{D}_{j+1}$. Очевидно, оператор $\mathbf{Q}_{j+k+1}$ будет подчинен логической функции

$$\beta\ (p_1, p_2, \dots , p_m) = \max_{P_{j+k}} (\max_{P_{j+k-1}} (\dots (\max_{P_j} \alpha\ (p_1, p_2,..., p_m)) \dots))\bullet$$

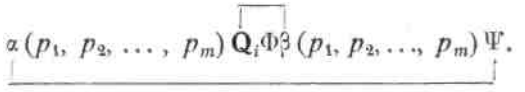
IV. Если логический оператор подчинен функции $\alpha\ (p_1, p_2, .., p_m)$ и передает управление непосредственно оператору $\mathbf{Q}_i$, то, очевидно, оператор $\mathbf{Q}_i$ тоже подчинен этой функции, ибо при своем выполнении логический оператор не может менять значения основных логических переменных.

Однако, как уже говорилось, $\mathbf{Q}_i$ может быть подчинен одновременно нескольким логическим функциям, например, в логической схеме



оператор $Q_i$ подчинен функциям α ($p_1, p_2, .., p_m$ ) и β ($p_1, p_2, .., p_m$), так как перед его выполнением всегда α=1 и β=1.

V. Рассмотрим схему



Очевидно, в такой схеме оператор $\mathbf{Q}_i$ получает управление либо при α = 1, либо при β = 1. Легко видеть, что $\mathbf{Q}_i$ подчинен логической функции

$$\alpha \vee \beta,$$

хотя не подчинен ни α , ни β в отдельности.

**2. Равносильность логических функций.** Логическая схема, приведенная к виду, удобному для преобразований, состоит из символов, обозначающих нелогические операторы и основные логические переменные. Между этими символами могут быть логические связи (показанные с помощью знаков логических связей $\wedge, \vee, \neg, \rightarrow, \sim, \overline{\vee}$) и стрелки или полускобки, которые мы будем называть знаками операторной связи.

Логические функции α ($p_1, p_2, ..., p_m$) и β ($p_1, p_2, ..., p_m$) мы будем называть *равносильными,* если при замене в любой логической схеме одной из них на другую получается логическая схема, *равносильная* первоначальной.

В виде формул это можно записать так:

$$\alpha (p_1, p_2, ..., p_m) = \beta (p_1, p_2, ..., p_m),$$

если

$$S(\alpha)=S(\beta).$$

Подчеркнем, что логическая функция $\alpha (p_1, p_2, ..., p_m)$ может стоять в логической схеме в различных местах. При этом подразумевается возможность замены α ($p_1, p_2, ..., p_m$) на β ($p_1, p_2, ..., p_m$) как во всех местах сразу, так и лишь в части мест.

Здесь и в дальнейшем запись вида α ($p_1, p_2, ..., p_m$) означает, что функция α зависит от всех или от части переменных $p_1, p_2, ..., p_m$.

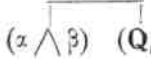
Совершенно очевидно, что две логические функции, которые логически эквивалентны (см. § 11), являются равносильными. В самом деле, логически эквивалентные функции при одинаковых комбинациях значений основных логических переменных $p_1, p_2, ..., p_m$ имеют одинаковые значения. Поэтому как при проверке значений одной из них, так и при проверке значений другой логический оператор будет передавать управление совершенно одинаково. То есть при замене логической функции на эквивалентную получится логическая схема, все значения которой совпадают с соответствующими значениями первоначальной (то есть полученная логическая схема будет равносильна первоначальной). В связи с этим как логическую эквивалентность, так и равносильность логических функций мы записываем с помощью одного и того же знака — знака равенства.

Из сказанного вытекает следующее правило:

> П р а в и л о II. В логической схеме можно логическую функцию заменить любой другой логической функцией, которая ей эквивалентна.

Рассмотрим, например, выражение

где оператор, проверяющий условие α ($p_1, ...., p_m$), подчинен функции β ($p_1, ..., p_m$). Легко видеть, что так как оператор, проверяющий условие α выполняется только при β = 1, то при выполнении этого оператора α/\β = α. Отсюда согласно правилу II в приведенном выражении можно функцию α заменить на α/\β:



Полученный результат можно сформулировать в виде следующего правила:

> Правило III. Если логический оператор, проверяющий значение логической функции α ($p_1, ..., p_m$), подчинен логической функции β ($p_1, ..., p_m$), то его можно заменить логическим оператором, проверяющим значение функции α/\β.

**3. Исключение знаков логических связей**. Покажем, каким образом можно от знаков логических связей перейти в схеме к знакам операторных связей.

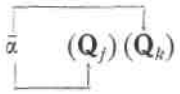
Так как в дальнейшем часто будет возникать необходимость замены одного логического оператора двумя,

условимся, что прежний номер оператора мы будем сохранять лишь за одним из новых операторов, второму новому оператору мы будем присваивать произвольный номер, еще не примененный в логической схеме.

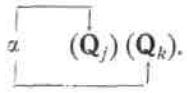
Пусть даны две логические функции.

$$\alpha = \alpha\,(p_1, p_2, \ldots, p_m),$$
$$\beta = \beta\,(p_1, p_2, \ldots, p_m).$$

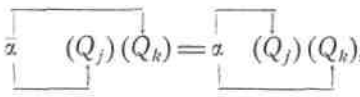
Сопоставим выражения



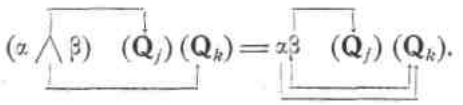
и



Легко видеть, что при $\alpha = 1$ оба выражения передают управление одному и тому же оператору $\mathbf{Q}_i$, а при $\alpha = 0$ — одному и тому же оператору $\mathbf{Q}_k$. Очевидно, что рассматриваемые логические операторы равнозначны. Отсюда вытекают правило и формула:

> Правило IV. Знак логического отрицания, стоящий над формулой, проверяемой логическим оператором, можно отбросить, одновременно перенося верхнюю стрелку, идущую от логического оператора, вниз, нижнюю — вверх, т. е.
>
> 
>
> или, что то же,
>
> $$(\alpha)_i \overset{k}{\sqcap}_j (\underset{i}{\sqcup}\, \mathbf{Q}_j)(\overset{i}{\neg}\, \mathbf{Q}_k) = (\alpha)_i \overset{j}{\sqcap}_k (\overset{i}{\neg}\, \mathbf{Q}_j)(\underset{i}{\sqcup}\, \mathbf{Q}_k).$$

Рассмотрим логический оператор, проверяющий значение функции $\alpha \wedge \beta$. Путем проверки легко убедиться в том, что справедлива формула



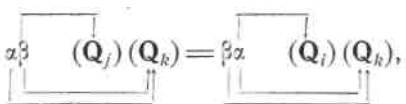
(IX.26)

Действительно, рассматривая левую часть этой формулы, видим, что если хотя бы одна из логических функций $\alpha$ или $\beta$ равна нулю, то $\alpha \wedge \beta = 0$ и управление передается оператору $\mathbf{Q}_k$. И только в случае $\alpha = 1$, $\beta = 1$ управление будет передано оператору $\mathbf{Q}_j$. Теперь рассмотрим правую часть формулы. Если $\alpha = 0$, то при любом значении $\beta$ управление получит $\mathbf{Q}_k$. Если $\alpha = 1$, то управление получает $\beta$, причем если $\beta = 0$, то управление передается опять $\mathbf{Q}_k$ и лишь при $\alpha = 1$ и $\beta = 1$ управление получит $\mathbf{Q}_j$.

Совершенно очевидно, что при замене в схеме логического оператора, стоящего в левой части формулы, парой логических операторов, приведенных в правой части формулы, ни одно из допустимых значений логической схемы не изменится — это убеждает нас в справедливости формулы (IX.26). Доказанная нами формула при записи с помощью полускобок принимает такой вид:

$$(\alpha \wedge \beta)_i \overset{j}{\sqcap}_k (\overset{i}{\neg}\, \mathbf{Q}_j)(\underset{i}{\sqcup}\, \mathbf{Q}_k) = (\alpha)_i \underset{k}{\sqcup} (\beta)_{i'} \overset{j}{\sqcap}_k (\overset{i}{\neg}\, \mathbf{Q}_j)(\underset{i,i'}{\sqcup}\, \mathbf{Q}_k).$$

Ввиду того, что конъюнкция подчиняется переместительному закону, из формулы (IX.26) без труда получаем:

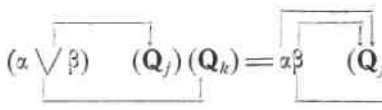


или, что то же,

$$(\alpha)_i \underset{i}{\sqcup} (\beta)_k \overset{m}{\sqcap}_i = (\beta)_i \underset{i}{\sqcup} (\alpha)_k \overset{m}{\sqcap}_i.$$

Теперь рассмотрим логический оператор, проверяющий дизъюнкцию двух

логических функций $\alpha$ и $\beta$. Рассуждая примерно так же, как при изучении оператора, проверяющего конъюнкцию, убеждаемся в справедливости формулы

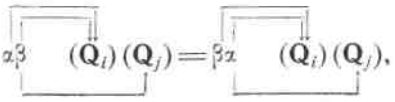


или, что то же,

$$(\alpha \vee \beta)_i \overset{j}{\underset{k}{\sqcap}} (\overset{i}{\neg}\, \mathbf{Q}_j)\,(\underset{i}{\sqcup}\, \mathbf{Q}_k) = (\alpha)_{i'} \overset{j}{\sqcap} (\beta)_i \overset{i}{\underset{k}{\sqcap}} (\overset{i,\,i'}{\neg}\, \mathbf{Q}_j)\,(\underset{i}{\sqcup}\, \mathbf{Q}_k).$$

$$(\alpha \vee \beta)_i \overset{j}{\underset{k}{\sqcap}} (\overset{i}{\neg}\, \mathbf{Q}_j)\,(\underset{i}{\sqcup}\, \mathbf{Q}_k) = (\alpha)_{i'} \overset{j}{\sqcap} (\beta)_i \overset{i}{\underset{k}{\sqcap}} (\overset{i,\,i'}{\neg}\, \mathbf{Q}_j)\,(\underset{i}{\sqcup}\, \mathbf{Q}_k).$$

Ввиду того, что дизъюнкция подчиняется переместительному закону, легко убеждаемся, что



(IX.27)

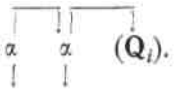
или, что то же,

$$(\alpha)_i \overset{m}{\sqcap} (\beta)_k \overset{m}{\underset{i}{\sqcap}} = (\beta)_i \overset{m}{\sqcap} (\alpha)_k \overset{m}{\underset{i}{\sqcap}}.$$

Как известно, все логические связи могут быть представлены с помощью связей $\wedge$ (u) $\vee$ (или) и $\bar{}$ (не). Следовательно, с помощью полученных выше формул любые знаки логических связей, присутствующие в логической схеме, могут быть заменены знаками операторных связей.

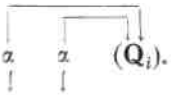
## § 48. Преобразование логических схем

**1. Преобразование стрелок.** Предположим, что в разных местах логической схемы стоят два логических оператора, проверяющих значение одной и той же функции $\alpha\,(p_1, p_2, \ldots, p_m)$, причем верхняя стрелка от одного из этих операторов (назовем его первым) ведет к другому (назовем его вторым), а верхняя стрелка от второго ведет к некоторому оператору $\mathbf{Q}_i$. Независимо от взаимного расположения в схеме обоих логических операторов и оператора $\mathbf{Q}_i$, будем изображать этот случай следующим образом:

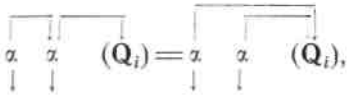


(IX.28)

Очевидно, если при выполнении первого логического оператора $\alpha = 1$, то управление будет передано второму логическому оператору. При этом по-прежнему $\alpha = 1$ (логические операторы не изменяют значений основных логических переменных) и, следовательно, второй логический оператор передаст управление оператору $\mathbf{Q}_i$. Ясно, что вместо стрелок, показанных в выражения (IX.28), можно, не изменяя ни одного из допустимых значений схемы, применить следующее расположение стрелок:



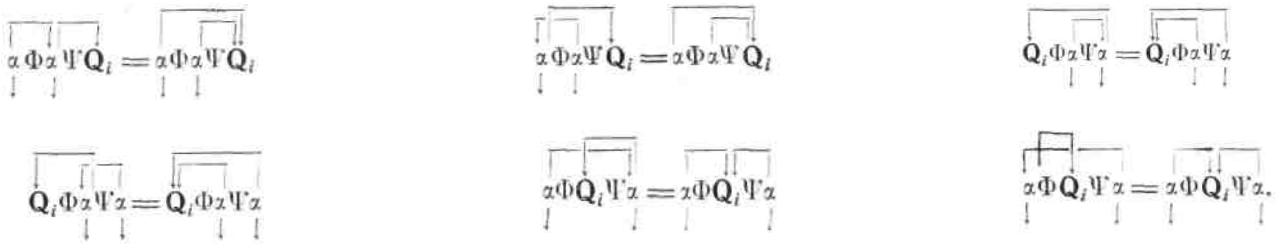
Отсюда получается формула



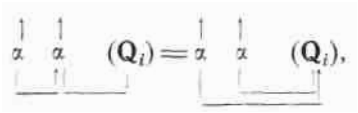
или, что то же,

$$(\alpha)_j \overset{k}{\underset{}{\sqcup}} \overset{f}{\neg} (\alpha)_k \overset{i}{\underset{}{\sqcup}} (\overset{k}{\neg}\, \mathbf{Q}_i) = (\alpha)_j \overset{i}{\underset{}{\sqcup}} (\alpha)_k \overset{i}{\underset{}{\sqcup}} (\overset{f,\,k}{\neg}\, \mathbf{Q}_i).$$

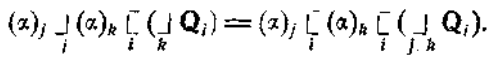
Полученная нами в условной записи формула охватывает следующие случаи:



Легко убедиться в том, что такие же шесть формул справедливы и для нижних стрелок. В общем виде эти формулы можно записать так:



или, что то же,



Полученные нами формулы, позволяющие производить преобразования стрелок, удобно объединить в следующее правило.

> Правило V. Если в логической схеме присутствуют два логических оператора, проверяющих значение одной и той же функции, причем от первого из них идет стрелка ко второму, то конец этой стрелки можно передвинуть вдоль одноименной стрелки, идущей от второго логического оператора.

Кроме того, справедливо еще одно правило, описывающее преобразование стрелки, обратное только что сформулированному (получается, если в вышеприведенных формулах поменять местами их левые и правые части).

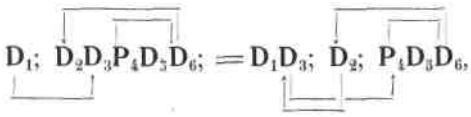
> Правило VI. Если к некоторому оператору ведут одноименные стрелки, начинающиеся возле логических операторов, проверяющих значение одной и той же функции, то конец одной из стрелок можно перенести к началу другой.

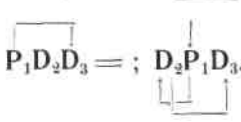
Одноименными мы называем стрелки, если обе они являются верхними или если обе они являются нижними (напомним, что различать верхние и нижние стрелки можно только, если они начинаютсяу логического оператора).

**2. Перестановка операторов.** Совершенно очевидно следующее правило:

> Правило VII. В логической схеме всякий оператор можно перенести в промежуток между двумя другими операторами, в начало схемы или в конец схемы, лишь бы при этом все начала и концы стрелок, принадлежащие операторам до преобразования, принадлежали тем же операторам после преобразования.

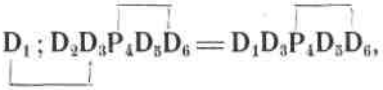
Например:

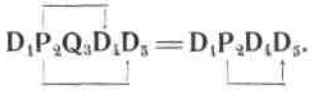
а) 

**б)** 

**3. Исключение операторов.** Очевидна справедливость следующего правила.

> Правило исключения VIII. Если к некоторому оператору Q логической схемы не ведет ни одной стрелки, то этот оператор вместе с идущими от него стрелками можно вычеркнуть из логической схемы (напомним, что если перед первым оператором не стоит «;», то считается, что к нему ведет стрелка).

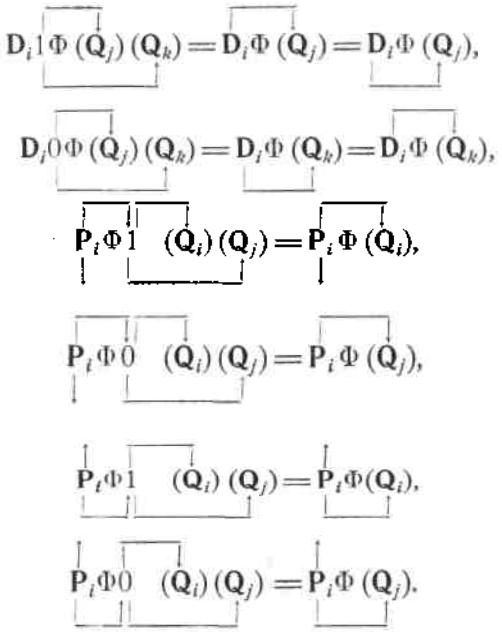
а) 

б) 

**Замечание.** Напомним, что мы условились рассматривать только такие схемы, в процессе выполнения которых стрелки не возникают и не исчезают. Если бы допускалось преобразование стрелок, то стрелка, отсутствующая в начале выполнения схемы, могла бы появиться в процессе ее выполнения и исключение оператора, к которому вначале не ведет ни одной стрелки, а потом будут вести стрелки, было бы ошибочным.

**4. Логические операторы, проверяющие значения тождественно постоянных логических функций.** Условимся всегда истинное условие обозначать символом 1, а всегда ложное — символом 0. Тогда, очевидно,

справедливы формулы



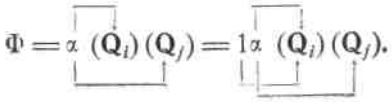
Последние шесть формул позволяют высказать еще одно правило.

Правило соединения стрелок IX. Если от некоторого оператора идет стрелка к логическому оператору, проверяющему тождественно постоянную функцию (т. е. равную тождественно либо нулю, либо единице), то последний оператор можно вычеркнуть вместе с той из идущих от него стрелок, которая не отвечает значению тождественно постоянной функции, а вторую стрелку соединить со стрелкой, идущей от первого из операторов. Характер новой «удлиненной» стрелки определяется ее началом.

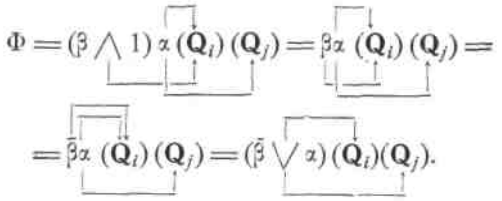
Опираясь на последнее правило и правило III, получим в виде следствия

Правило X. Если логический оператор, проверяющий значение функции $\alpha$, подчинен функции $\beta$, то этот логический оператор можно заменить новым логическим оператором, проверяющим функцию $\overline{\beta} \vee \alpha$.

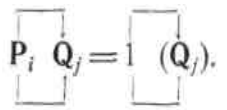
Доказательство правила X.



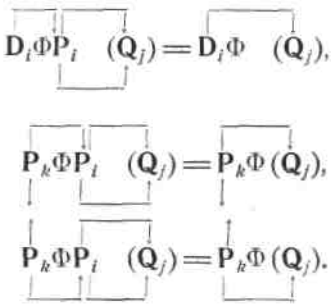
Очевидно, оператор $\underset{\downarrow}{1}\rightarrow$ тоже подчинен функции $\beta$. Следовательно,



Отметим, что логический оператор $\mathbf{P}_i$, обе стрелки от которого ведут к одному и тому же оператору $\mathbf{Q}_j$, можно считать равносильным логическому оператору, проверяющему значение тождественно постоянной логической функции. Например, можно написать:



Отсюда с помощью правила IX легко получаем такие следствия:



Из приведенных выше формул и правил преобразования логических схем можно вывести ряд следствий, удобных для применения в тех или других конкретных случаях.

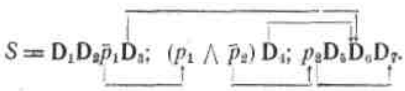
Полученные нами формулы и правила преобразований образуют систему логических преобразований логических схем, т. е. преобразований, при которых учитывается лишь структура логических операторов, тогда как каждый нелогический оператор принимается за нечто единое, целое и неизменное.

Приведенная система преобразований логических схем разработана с применением другой символики Ю.И.Яновым, который кроме того доказал полноту системы этих преобразований [*]).

При этом еще раз подчеркнем, что описанные преобразования допустимы не для всех логических схем, а лишь для схем, удовлетворяющих определенным условиям (уже приведенным выше). Полноту системы логических преобразований следует понимать так: если две схемы равносильны (в вышеуказанном смысле), то одну из них (любую) можно с помощью конечного числа этих преобразований свести к другой. Не следует думать, что нельзя ввести другое понятие равносильности схем, при котором рассмотренная система логических преобразований перестанет быть полной и, следовательно, будет требовать дополнительных формул и правил.

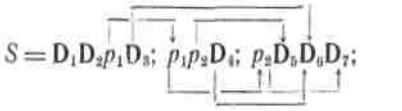
**5. Примеры преобразования схем.** В заключение приведем несколько несложных примеров на преобразование схем.

П р и м е р 1. Рассмотрим схему



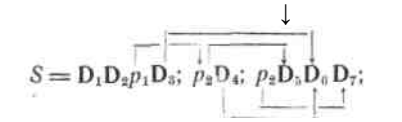
При любом распределении сдвигов эта схема может быть упрощена следующим образом:

а) перейдем от логических связей к операторным:

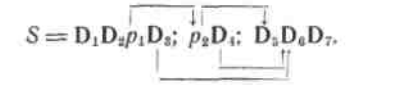


б) произведем перенос стрелки вдоль одноименной стрелки у операторов, проверяющих одну и ту же логическую функцию:

в) исключим оператор, к которому не ведет ни одной стрелки (оператор $p_1 \rightarrow$, стоящий после $D_3$):

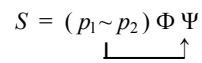


г) исключим оператор, к которому не ведет ни одной стрелки оператор $p_2$—.после $D_4$):



Полученная логическая схема является окончательной.

П р и м е р 2. Рассмотрим схему

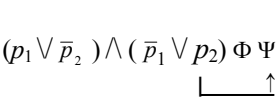


при произвольном распределении сдвигов. Преобразуем ее так:

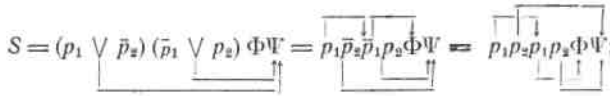
а) заменим логическую функцию $p_1 \sim p_2$ логической функцией, которая ей эквивалентна:

$$(p_1 \vee \bar{p}_2) \wedge (\bar{p}_1 \vee p_2) = p_1 \sim p_2,$$

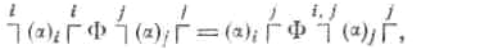
получим:



б) перейдем от логических связей к операторным:



[*]) При доказательстве полноты системы преобразований предполагалось, что каждый иелогический оператор входит в схему только один раз. Это не значит, что в схеме не может быть одинаковых операторов. Просто одинаковые нелогические операторы, стоящие в разных местах схемы, должны быть обозначены различными символами. Для полноты системы преобразований нужна еще пара формул, не имеющих практического значения и потому не приведенных выше, а именно:



и аналогичная формула для нижних стрелок.

в) произведем перенос верхней стрелки вдоль одноименной стрелки другого логического оператора (проверяющего ту же логическую функцию, что и первый):

$$S = p_1\, p_2\, p_1\, p_2\, \Phi\Psi;$$

г) оператор, проверяющий значение $p_1$, который стоит в схеме после $p_2$, подчинен логической функции $\bar{p}_1$, следовательно, в силу правила III его можно заменить оператором, проверяющим функцию $\bar{p}_1 \wedge p_1 = 0$,

$$S = p_1\, p_2\, 0\, p_2\, \Phi\Psi;$$

д) применяем правило соединения стрелок:

$$S = p_1\, p_2\, p_2\, \Phi\Psi;$$

здесь, удаляя оператор 0, мы с его нижней стрелкой соединили нижнюю стрелку, идущую от первого из операторов $\overset{\uparrow}{p_2}\rightarrow$.

Пример 3. Рассмотрим схему

$$S = (\bar{p}_1 \wedge p_2)\, \mathbf{D}_1\, (p_1 \vee p_2)\, \mathbf{D}_2 \Phi$$

при следующем распределении сдвигов:

$$P_1 = \{p_1\};\quad P_2 = \{p_1, p_2\}.$$

Преобразуем ее следующим образом:

а) переходим от логических связей к операторным:

$$S = \bar{p}_1\, p_2\, \mathbf{D}_1\, p_1\, p_2\, \mathbf{D}_2 \Phi = p_1\, p_2\, \mathbf{D}_1 p_1\, p_2 \mathbf{D}_2 \Phi;$$

б) по правилу скольжения конца стрелки имеем:

$$S = p_1 p_2 \mathbf{D}_1 p_1 p_2 \mathbf{D}_2 \Phi;$$

в) пользуясь формулой (IX.27), в силу которой

$$p_1 p_2 \mathbf{D}_2 \Phi = p_2 p_1 \mathbf{D}_2 \Phi,$$

имеем:

$$S = p_1 p_2 \mathbf{D}_1 p_2 p_1 \mathbf{D}_2 \Phi;$$

г) по правилу скольжения конца стрелки имеем:

$$S = p_1 p_2 \mathbf{D}_1 p_2 p_1 \mathbf{D}_2 \Phi;$$

:

д) теперь мы видим, что оператор $\overset{\uparrow}{p_2}\rightarrow$, стоящий после $\mathbf{D}_1$ подчинен функции $p_2$ (так как $\mathbf{D}_1$ не может изменять значения $p_2$), следовательно, его можно заменить оператором

$$\bar{p}_2 \vee \overset{\uparrow}{p_2}\rightarrow = \overset{\uparrow}{1}\rightarrow;$$

по правилу соединения стрелок получим

$$S = p_1 p_2 \mathbf{D}_1 \bar{p}_2 \vee p_2 p_1 \mathbf{D}_2 \Phi = p_1 p_2 \mathbf{D}_1 1\, p_1 \mathbf{D}_2 \Phi = p_1 p_2 \mathbf{D}_1 ; p_1 \mathbf{D}_2 \Phi;$$

е) в последней схеме оператор $p_1\rightarrow$, стоящий после $\mathbf{D}_1$, подчинен функции $\bar{p}_1$ и, следовательно, может быть заменен оператором,

проверяющим значение логической функции $\bar{p}_1 \wedge p_1 = 0$; имеем:

$$S = p_1 p_2 \mathbf{D}_1 ; 0\, \mathbf{D}_2 \Phi;$$

ж) по правилу соединения стрелок имеем:

$$S = p_1 p_2 \mathbf{D}_1 ; \mathbf{D}_2 \Phi = p_1 p_2 \mathbf{D}_1 \mathbf{D}_2 \Phi.$$

П р и м е р 4. Требуется упростить схему

$$S = \mathbf{D}_1 p_1 \mathbf{D}_2 p_2 p_1 p_4 ; \mathbf{D}_3 p_2 \mathbf{D}_4 ; \mathbf{D}_5 \Phi,$$

если распределение сдвигов таково:

$$P_1 = \{ p_1, p_2, p_3, p_4\},$$
$$P_2 = \{ p_2, p_3, p_4\},$$
$$P_3 = \{ p_1, p_3, p_4\},$$
$$P_4, = P_5 = \{p_1, p_2, p_3, p_4\}.$$

а) переставим оператор $p_4^{\uparrow}_{\downarrow}$ в промежуток между операторами $\mathbf{D}_4$ и $\mathrm{D}_5$;

имеем:

$$S = \mathbf{D}_1 p_1 \mathbf{D}_2 p_2 p_1 \mathbf{D}_3 p_2 \mathbf{D}_4 ; p_4 \mathbf{D}_5 \Phi;$$

б) воспользуемся правилом скольжения конца стрелки:

$$S = \mathbf{D}_1 p_1 \mathbf{D}_2 p_2 p_1 \mathbf{D}_3 p_2 \mathbf{D}_4 ; p_4 \mathbf{D}_5 \Phi;$$

в) оператор $\uparrow p_1 \rightarrow$, стоящий перед $\mathbf{D}_3$, подчинен теперь логической функции $\bar{p}_1$ (оператор $\mathbf{D}_2$ не может менять значение $p_1$); заменим его оператором, проверяющим значение логической функции $\bar{p}_1 \wedge p_1, = 0$, и воспользуемся правилом соединения стрелок:

$$S = \mathbf{D}_1 p_1 \mathbf{D}_2 p_2 \; 0 \; \mathbf{D}_3 p_2 \mathbf{D}_4 ; p_4 \mathbf{D}_5 \Phi = \mathbf{D}_1 p_1 \mathbf{D}_2 p_2 \mathbf{D}_3 p_2 \mathbf{D}_4 ; p_4 \mathbf{D}_5 \Phi;$$

г) оператор $p_2^{}_{\downarrow} \rightarrow$, стоящий после $\mathbf{D}_3$, подчинен логической функции $p_2$ (ибо — $\downarrow$ см. распределение сдвигов — $\mathbf{D}_3$ не может изменять значение переменной $p_2$); заменим его оператором, проверяющим значение логической функции $p_2 \vee p_2 = 1$ и воспользуемся правилом соединения стрелок:

$$S = \mathbf{D}_1 p_1 \mathbf{D}_2 p_2 \mathbf{D}_3 \; 1 \; \mathbf{D}_4 ; p_4 \mathbf{D}_5 \Phi = \mathbf{D}_1 p_1 \mathbf{D}_2 p_2 \mathbf{D}_3 \mathbf{D}_4 ; p_4 \mathbf{D}_5 \Phi,$$

## ГЛАВА X
## ПРОГРАММИРУЮЩИЕ ПРОГРАММЫ

В настоящее время высшим достижением в области автоматизации программирования являются программирующие программы, основанные на операторном методе программирования.

В СССР существует ряд таких программ, составленных научными коллективами различных учреждений, работающих в области применения электронных цифровых машин. Первая из программирующих программ такого типа была создана коллективом научных работников в Математическом институте им. В. А. Стеклова АН СССР. В этой главе приведено описание одной из наиболее развитых операторных программирующих программ, известной под названием *ПП-С* (программирующая программа для машины *Стрела). ПП-С* составлена под руководством Н. А. К р и н и ц к о г о коллективом научных сотрудников в составе: А. М. Б у х т и я р о в а , Г. Д. Ф р о л о в а , И. В. П о т т о с и н а , Л. В. В о й т и ш е к , А. А. Л е в и н о й и Л . М. З и к е в с к о й .

При составлении программ с помощью *ПП-С* математик должен выбрать численный метод для решения задачи, выбрать расчетные формулы и составить логическую схему программы. Кроме того, ему иногда приходится составить вручную отдельные небольшие части программы. Остальная (большая) часть программы составляется самой электронной цифровой машиной, управляемой *ПП-С.* Даже распределение памяти в ряде случаев *ПП-С* производит автоматически, без непосредственного участия человека.

Опыт эксплуатации *ПП-С* показал, что составление логической схемы программы и задания кодировщикам должен выполнять высококвалифицированный программист.

Кодировку и составление программы на машине с помощью *ПП-С* должны производить техники-программисты. Готовая программа передается опять программисту для отладки на машине и решения задачи.

Если логическая схема и задание кодировщикам были составлены без ошибок, то готовая программа почти не требует отладки.

Применение программирующей программы значительно облегчает труд программистов и сокращает время программирования.

Расход машинного времени на автоматическое программирование вполне окупается экономией машинного времени при отладке программы на машине.

Программы, составленные с помощью *ПП-С,* не уступают по своим качествам программам, составленным вручную. Работа над дальнейшим развитием *ПП-С* и расширением ее возможностей продолжается.

## § 49. Операторное автоматическое программирование и *ПП-С*

### 1. Сущность операторного автоматического программирования.

Операторный метод автоматического программирования основывается на следующем. Просмотр большого количества программ, составленных опытными программистами для решения математических задач и считающихся хорошими, показал, что эти программы состоят главным образом из операторов четырех типов:

1) арифметических операторов **A**;
2) логических обобщенных операторов **P**;
3) операторов переадресации **F**;
4) операторов восстановления к первоначальному виду **O**.

Правила, по которым программисты составляют команды операторов этих типов, легко могут быть сформулированы, т. е. процесс программирования этих операторов легко поддается алгорифмизации.

Отсюда следует, что программирование операторов перечисленных типов может быть переложено на электронную цифровую машину.

Тот же анализ показывает, что в составе программ решения математических задач встречаются также операторы других типов, например, операторы формирования, циркуляции и другие. Однако последние операторы составляют по количеству команд меньшую часть программы, чем операторы вышеперечисленных четырех типов.

Из сказанного следует, что при создании операторной программирующей программы естественно вначале принять за стандартные операторы типов **A**, **P**, **F**, **O**. Остальные операторы можно принять за нестандартные и обозначать их буквой **H** (**H** — обобщенный оператор, состоящий из элементарных операторов любых типов). Составление нестандартных операторов остается за человеком.

Искусный программист может в качестве нестандартного оператора взять целую группу обобщенных операторов, обладающую тем свойством, что от стандартных операторов, стоящих в логической схеме, управление получает лишь первый из обобщенных операторов группы, а другие обобщенные операторы могут получать управление только друг от друга и от команд других нестандартных операторов. Однако при таком строении нестандартных операторов нестандартное программирование значительно усложняется, а логическая схема становится менее подробной и, значит, менее удобной при проверке программы и отладке ее на машине.

Программирующая программа будет автоматически составлять стандартные операторы, а нестандартные операторы — без изменения включать в программируемую программу. Программирующую программу следует составлять таким образом, чтобы без значительной ее переделки было возможно расширение класса стандартных операторов.

По мере совершенствования программирующей программы и расширения класса ее стандартных операторов, все большая и большая часть работы по программированию будет передаваться машине. Тем не менее возможность включения нестандартных операторов должна сохраняться. Эта возможность делает программирующую программу гибкой, не ограничивающей изобретательности программиста, применимой к составлению широкого класса рабочих программ.

Естественно потребовать, чтобы программирующая программа могла включать в составляемые программы также библиотечные подпрограммы (тем более, что алгорифм включения библиотечных подпрограмм нам уже известен). При этом библиотечные подпрограммы следует считать обобщенными операторами (их обозначают буквой **C**), принадлежащими к классу стандартных операторов.

Для описываемой в этой главе программирующей программы *ПП-С* стандартными операторами являются операторы типов **A, P, C, F, O**.

Нередки случаи, когда программируемая программа содержит не все типы стандартных операторов. Для экономии машинного времени нужно обеспечить возможность исключения из программирующей программы тех ее частей, которые при этом оказываются излишними.

Возможность расширения класса стандартных операторов без значительной переделки программирующей программы, возможность исключения из работы частей программирующей программы и сохранение наибольшей части оперативной памяти машины для размещения в ней программируемой программы лучше всего обеспечиваются разбиением программирующей программы на части, так называемые блоки, каждая из которых составляет лишь операторы определенного типа или выполняет определенную часть вспомогательной работы.

Для того чтобы программирующая программа составила рабочую программу, должна быть задана информация о подлежащей составлению программе. Читатель уже знает, что вся необходимая информация содержится в логической схеме программы и приложенном к этой схеме описании каждого оператора.

Логическая схема программы и ее описание содержат следующие сведения:

1) тип каждого оператора, входящего в схему программы, и его порядковый номер;

2) взаимное расположение операторов различного типа;

3) направления передачи управления операторами друг другу;

4) данные о том, какие операторы и по каким параметрам переадресуются каждым оператором переадресации;

5) данные о том, какие операторы восстанавливаются к своему начальному виду каждым оператором восстановления.

Очевидно, эти сведения необходимы, но их еще не достаточно для составления программы. Нужен ряд дополнительных сведений, которые содержатся в описании операторов схемы:

6) математические формулы, по которым ведет счет каждый из арифметических операторов;

7) логические формулы, по которым каждый из логических операторов производит выбор одного из возможных направлений передачи управления;

8) библиотечные подпрограммы типа **С,** обозначенных в схеме символами $\mathbf{C}_i$;

9) сведения, устанавливающие связи между библиотечными подпрограммами и составляемой программой;

10) полностью составленные нестандартные операторы в своем окончательном виде.

Для построения операторов переадресации нужно знать:

11) шаг переадресации по каждому параметру;

12) положение внутри переадресуемых операторов команд, которые необходимо модифицировать; адреса команд, подлежащие модификации.

Для построения операторов восстановления необходимо знать:

13) положение изменяемых команд в восстанавливаемых операторах;

14) начальный вид восстанавливаемых команд.

Данные, указанные в п. 12, могут быть получены только после составления операторов типов **А**, **Р**, **С** при условии, что известна

15) зависимость адресов от параметров.

Сведения, перечисленные в пп. 1—11 и 15, должны быть заданы программирующей программе, а сведения, указанные в пп. 12—14, могут быть ею получены в процессе составления программируемой программы.

**2. Способ кодировки информации для *ПП-С.*** Для того чтобы исходная информация могла быть введена в оперативную память машины, она должна быть представлена в виде чисел, т. е. закодирована. Восьмеричные числа, являющиеся кодами информации в ее начальном виде, а также в виде, в котором она передается от одного блока программирующей программы другому, называются *условными числами.* Система условных чисел, используемых в программирующей программе *ПП-С,* приведена в таблице 45. В этой таблице *величинами* называются буквы и числа, входящие в состав исходных математических и логических формул. *Числовыми константами* называются числа, входящие в исходные формулы (обычно — числовые коэффициенты) или являющиеся исходными данными, в том случае, если их не относят к классу величин (программист по своему усмотрению может отнести число к классу величин или к классу числовых констант).

*Рабочими ячейками* называются ячейки, применяемые для хранения промежуточных результатов, используемых только тем оператором, который их вычислил. Те промежуточные результаты, которые являются исходными данными для других операторов, необходимо относить к классу величин. Их необходимо обозначать буквами. При этом требуется вносить соответствующие изменения в исходные формулы. Остальные наименования, приведенные в таблице 45, не требуют пояснений.

Заметим, что многие условные числа имеют двоякий смысл. Так, условное число величины означает как саму величину, так и ячейку (ряд последовательных ячеек), отведенную для хранения этой величины (ряда последовательных значений величины). Условное число рабочей ячейки обозначает как рабочую ячейку, так и промежуточный результат, вносимый в эту ячейку. Условное число константы (восстановления, переадресации или числовой) обозначает как саму константу, так и отведенную для ее хранения ячейку. Условное число номера оператора обозначает как номер оператора, так и первую команду этого оператора и ячейку, хранящую эту команду.

Рассматривая таблицу 45, видим, что лишь число 0, знаки математических действий (кроме знака возведения в степень), знак равенства, знаки логических связей и константы, закоммутированные в *УВК,* имеют индивидуальные условные числа. Прочие условные числа, фигурирующие в таблице 45, имеют следующее строение:

$$u = U + n$$

где $u$ — условное число, $U$ — некоторое число, являющееся признаком класса, к которому принадлежит условное число, и $n$ — число, характеризующее индивидуальное значение условного числа. Например, условное число

$$u = 5012$$

имеет признак класса величин

$$U = 5000$$

и индивидуальный признак

$n = 0012$,

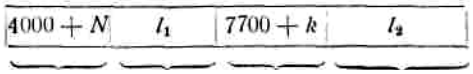
являющийся в данном случае номером величины.

Таблица 45. Условные числа, применяемые в *ПП-С*

| №. п/п | Наименование | Условное число | Примечание |
|---|---|---|---|
| 1 | Величина | 5000 + *n* | *n* — номер величины, $1 \le n \le 777$ |
| 2 | 0 (нуль) | 5000 | |
| 3 | 1 (единица) | 7400 | |
| 4 | Рабочая ячейка | 6000 + n | *n* — номер рабочей ячейки, $1 \le n \le 377$ |
| 5 | Константа восстановления | 4400 + *n* | *n* — номер константы, $1 \le n \le 177$ |
| 6 | Константа переадресации | 4600 + *n* | *n* — номер константы, $1 \le n \le 177$ |
| 7 | Числовая константа | 6400 + *n* | *n* — номер константы, $1 \le n \le 377$ |
| 8 | Номер оператора или таблица констант | 4000 +N | *N*— номер оператора, $1 \le N \le 377$ |
| 9 | Команда нестандартного оператора | 7000 + *n* | *n* — номер команды нестандартного оператора, отсчитываемый от его начала; такое условное число может стоять только внутри нестандартного оператора; $1 \le n \le 377$ |
| 10 | Параметр переадресации | 7700 + *n* | *n* — номер параметра, $1 \le n \le 77$ |
| 11 | (. . .( (открытые скобки) | 0400 + *n.* | *n* — количество открытых скобок, стоящих подряд, $1 \le n \le 77$ |
| 12 | )...) (закрытые скобки) | 0200 + *n* | *n* — количество закрытых скобок, стоящих подряд, $1 \le n \le 77$ |
| 13 | Левые «прямые скобки» знака абсолютной величины | 0500 + *n* | *n* — количество стоящих подряд левых прямых скобок, $1 \le n \le 77$ |
| 14 | Правые «прямые скобки» знака абсолютной величины | 0300 + *n* | *n* — количество стоящих подряд правых прямых скобок, $1 \le n \le 77$ |
| 15 | = (знак равенства) | 0077 | Имеет *смысл* результативного знака, кроме формул вида *a = b,* в которых означает перенос (*a*) в ячейку *(b)* |
| 16 | Знаки действия (кроме возведения в степень, сдвига по адресу и тех действий, которые выполняются по стандартным программам *НСП).*<br><br>Например, + (плюс) | Условные числа совпадают с соответ-ствующими кодами операций<br><br>0001 | Любая операция, предусмотренная в машине, считается арифметическим действием. Например, даже условный переход можно записать в виде арифметической формулы $A_{12}$«0020 » $H_{17}$= 0.<br><br>Такая формула означает команду условного перехода к оператору $A_{12}$ при ω = 0 или к оператору $H_{17}$ при ω = 1. Формула *a = b* равноценна формуле *a* «0013» 0= *b* и означает команду <a>0000<b>13 |
| 17 | Знак возведения в положительную целую степень «*n*». Например, знак возведения в шестую степень | 1000 + *n*<br>1006 | $1 \le n \le 77$ |
| 18 | Знак действия «Сдвиг по адресу» | 0114 | |
| 19 | Знак действия выполняемого машиной по стандартной программе *НСП*<br><br>Например,<br>arcsin<br>arccos | 10 *K* + *n*<br><br>0740<br>0742 | К—код операции, отвечающей данному действию,<br>*n* — номер фиксированной ячейки памяти, в которую стандартная программа производит запись результата. Если запись производится в любую ячейку (указываемую в третьем адресе команды), следует брать n= 0<br><br>Код операции вычисления arcsin и arccos равен 74 |
| 20 | Логические открытые скобки [. .[ | 1200 + *n* | *n* — количество стоящих подряд открытых логических скобок, $1 \le n \le 77$ |
| 21 | Логические закрытые скобки ]...] | 1100 +*n* | *n* — количество стоящих подряд логических закрытых скобок, $1 \le n \le 77$ |
| 22 | Номер логической формулы (внутри логического оператора) | 7000 + *n* | *n* — номер логической формулы, $1 \le n \le 377$ |

| №. п/п | Наименование | Условное число | Примечание |
|---|---|---|---|
| 23 | Знаки логических условий<br>< (меньше)<br>≤ (меньше или равно)<br>> (больше)<br>≥ (больше или равно)<br>≡ (поразрядное совпадение)<br>≠ (не равно, т. е. нет поразрядного совпадения) | <br>0060<br>0061<br>0062<br>0063<br>0064<br>0065 | |
| 24 | Знаки логических связей<br>∧ (*и*)<br>∨ (*или*) | <br>0070<br>0071 | |
| 25 | Константа, хранящаяся в УВК.<br>Например, 1 (I) | Действительный адрес константы.<br>7423 | |

Кроме условных чисел, в *ПП-С* используются еще специальные числа, называемые *характеристиками операторов.* Характеристики имеют на бланке для программирования и в ячейке памяти машины такой вид:




I адрес II адрес III адрес контрольный знак и код операции

где 4000 + *N* — условное число номера оператора, $l_1$— количество строк бланка (ячеек памяти), следующих за строкой (ячейкой), на которой записана характеристика, и содержащих информацию об операторе; *k — признак типа оператора* (см. таблицу 46); $l_2$ для различных типов операторов имеет различное значение и будет пояснено ниже. Таблицы констант (восстановления, переадресации и числовых)также снабжены характеристиками вышеприведенного вида.

Т а б л и ц а 46. Типовые признаки операторов и таблиц констант

| Наименование типа оператора или таблицы констант | Типовой признак |
|---|---|
| Арифметический оператор..... | 01 |
| Логический оператор..... | 04 |
| Стандартная подпрограмма..... | 05 |
| Оператор переадресации..... | 02 |
| Оператор восстановления..... | 06 |
| Нестандартный оператор..... | 11 |
| Таблица констант восстановления..... | 21 |
| Таблица констант переадресации..... | 22 |
| Таблица числовых констант..... | 23 |

Еще некоторые специальные виды чисел, используемые в *ПП-С,* будут пояснены в дальнейшем.

**3. Вид команд, составляемых *ПП-С*.** Информация, задаваемая для *ПП-С,* последовательно обрабатывается ее блоками, причем каждый блок работает только один раз. Поэтому команды программируемой программы не могут возникать в порядке их расположения. Например, сперва составляются все арифметические команды программируемой программы и лишь потом команды переадресации и т. п.

Таким образом, в процессе составления программируемой программы неизвестно, какой номер в ней будет иметь каждая команда и сколько команд будет содержать тот или другой стандартный оператор. Невозможно, как правило, также заранее определить ячейки, в которых будут размещены константы (восстановления, переадресации и числовые).

Это обстоятельство требует, чтобы *ПП-С* составляла команды в специальном виде и лишь после того, как вся программа будет готова, приводила их в обычный вид. *ПП-С* составляет команды в условных числах. Вместо номеров ячеек, хранящих величины или константы, а также вместо номеров рабочих ячеек, в качестве адресов команд используются условные числа величин, констант и рабочих ячеек. Коды операций этих команд совпадают с обычными. Например, если требуется сложить величину *x* с числовой константой 1,7 и результат записать в рабочую ячейку и если 5001, 6405 и 6003 — соответствующие условные числа, то команда будет иметь такой вид:

5001 6405 6003 0 01.

В командах условного перехода в качестве первого и второго адреса *ПП-С* ставит условные числа номеров операторов, к которым должен происходить переход. Например, команда

4015 4052 0000 0 20

означает переход при ω = 0 к оператору № 15, а при ω= 1 к оператору № 52.

В качестве первого адреса команды переадресации берется номер оператора, команда которого должна быть модифицирована, во втором адресе ставится условное число константы переадресации, а в третьем адресе — номер модифицируемой команды, отсчитанной от начала содержащего ее оператора. Например, команда переадресации может иметь такой вид:

4011 4602 0005 0 02

или, если константа переадресации содержится в *УВК* (например, единица первого адреса), команда переадресации может быть следующей:

4011 7423 0005 0 02.

В первом адресе команды восстановления ставится условное число номера восстанавливаемого оператора, во втором адресе — условное число константы восстановления, а третьем адресе — номер восстанавливаемой команды, отсчитанный от начала содержащего ее оператора. Если восстанавливается одиночная команда, то код операции берется равным 00, если же производится групповое восстановление *n* команд, то в коде операции ставится это число *n.* Признаком того, что команда является командой восстановления, служит принадлежность ее оператору восстановления. Например, команда одиночного восстановления может иметь вид

4011 4405 0005 0 00.

Команда группового восстановления может быть следующей:

4011 4405 0005 0 04.

Последняя команда означает, что четыре команды оператора № 11, начиная с пятой команды, должны быть восстановлены путем группового переноса на их место четырех констант восстановления, начиная с константы, обозначенной условным числом 4405 (остальные три константы обозначены условными числами 4406, 4407, 4410).

Таким образом, программа, полученная с помощью *ПП-С,* не имеет обычного вида и требует дополнительной переработки. Эту переработку называют *присвоением действительных адресов.* Присвоение действительных адресов требует распределения памяти для программируемой программы. Распределение памяти может быть либо сделано программистом после просмотра программы, составленной в условных числах, либо автоматически выполнено самой *ПП-С.* Программы, составляемые *ПП-С,* подчинены одному ограничению: в них команды, константы восстановления, константы переадресации, числовые константы и рабочие ячейки располагаются неделимыми массивами.

**4. Нестандартное программирование.** Выше было сказано, что нестандартные операторы, входящие в логическую схему программируемой программы, должны быть составлены программистом вручную. Процесс составления нестандартных операторов получил название *нестандартного программирования.*

Нестандартное программирование выполняется по следующим правилам:

1. Нестандартный оператор состоит из команд и специальных чисел, называемых в дальнейшем *индикатами.* Индиката при записина бланке имеет вид команды. Адресами индикаты являются либо 0000, либо номера операторов. Контрольный знак равен нулю. Код операции равен 55. Например, индиката может иметь вид

4012 0000 4011 0 55.

Индиката относится к команде следующей непосредственно за ней. При подсчете числа команд, или при определении номера команды в операторе, индиката и следующая за ней команда считаются одной командой.

2. При составлении команд их контрольные знаки (кроме случая, указанного в и. 5) берутся равными нулю. Коды операций употребляются обычные.

3. В адресе команд вместо номеров ячеек, хранящих величины или константы, а также вместо номеров рабочих ячеек записываются соответствующие условные числа. Например, команда «Умножить величины, обозначенные условными числами 5001 и 5013, и произведение записать в рабочую ячейку 6017» имеет такой вид:

5001 5013 6017 0 05.

4. В команды нестандартных операторов можно включать действительные адреса, даже все три (под действительными адресами понимаются числа, начиная с 0000 и кончая 3777, а также начинаяс 7400 и кончая 7777). Например,

а) иногда бывает важно включение «нулевых» команд:

0000 0000 0000 0 00;

б) команда о переносе содержимого ячейки, отвечающей величине 5003, и содержимого одиннадцати

следующих за ней ячеек врабочие ячейки, первая из которых имеет номер 6015, пишется так:

5003 0013 6015 0 45;

в) иногда программисту при составлении нестандартного оператора может быть неясно значение какого-нибудь адреса, например номера зоны магнитной ленты; такой адрес можно оставить нулевым, с тем чтобы уже в готовой программе дополнительно пробить его на соответствующей перфокарте.

5. Если второй адрес команды, входящей в нестандартный опеатор, не подлежит обработке программирующей программой, контрольный знак этой команды должен быть равен единице (если код равен 30—37, контрольный знак можно взять равным нулю; то же в командах групповых операций). Второй адрес команды не должен подвергаться обработке в том случае, когда, являясь действительным, он совпадает по своему начертанию с одним из условных чисел.

Например, команда сдвига величины 5003 на пять разрядов вправо с записью результата в рабочую ячейку 6021 имеет такой вид:

5003 4105 6021 1 14.

Здесь второй адрес совпадает с условным числом оператора.

В команде сдвига по адресу второй адрес должен иметь вид 4000 + *n* или 4100 + *n,* где *n* — количество разрядов, на которое производится сдвиг. Машина *«Стрела»* допускает для второго адреса этой команды также вид 5000 + *n* или 5100 + *n.* Однако последний вид при нестандартном программировании применять не следует, так как второй адрес команды сдвига может совпасть с условным числом величины, зависящей от какого-нибудь параметра. В некоторых случаях это приведет к составлению команды переадресации, не предусмотренной в программируемой программе.

6. Команда о подводе зоны ленты, имеющая код 25 и первый адрес, совпадающий с одним из условных чисел, пишется в действительном виде (*ПП-С* ее не обрабатывает).

7. В командах условного и безусловного перехода, передающих управление операторам, в соответствующих адресах записываются условные числа операторов. Например, команда о передаче управления при ω = 0 оператору № 12, а при ω = 1 — оператору № 30 имеет такой вид:

4012 4030 0000 0 20.

При необходимости, кроме того, очистить ячейку, например рабочую ячейку 6012, команда имеет такой вид:

4012 4030 6012 0 20.

8. Вместо номеров ячеек, хранящих команды составляемого нестандартного оператора, пишутся условные числа номеров команд 7000 + *n*, где *n* — номер команды. Например, команда переадресации второй команды составляемого оператора будет иметь такой вид:

7002 7424 7002 0 02.

Команда условного перехода при ω = 0 ко второй, а при ω = 1 к 20-й команде составляемого оператора имеет такой вид:

7002 7020 0000 0 20.

9. Индиката вместе с непосредственно следующей за ней командой считаются одной командой (см. п. 1). Если в некотором адресе индикаты стоит условное число 4000 + *N,* а в одноименном адресе следующей за индикатой команды стоит некоторое число *n,* то это значит, что указанный адрес команды означает номер ячейки, хранящей команду, принадлежащую оператору, обозначенному условным числом 4000 + *N*, и имеющую, считая от начала оператора, номер *n.* Число 0000, стоящее в адресе индикаты, не меняет смысла содержимого одноименного адреса следующей за ней команды. Например,

а) 4015 0000 4015 0 55 —индиката,
0006 7425 0006 0 02 —команда

(прибавить единицу третьего адреса к шестой команде оператора № 15);

б) 4015 0000 0000 0 55 —индиката,
0005 7446 6041 0 11—команда

(выделить первый адрес из пятой команды оператора № 15).

10.. Константы восстановления тех команд нестандартного оператора, которые не содержат в себе чисел вида 7000 + *n,* являются «фотографиями» этих команд. Если команде нестандартного оператора предшествует индиката, то константой ее восстановления является «фотография» индикаты и команды. Если команда нестандартного оператора содержит адрес вида 7000 + *n,* то константа ее восстановления состоит из индикаты, имеющей в одноименном адресе условное число составляемого нестандартного оператора, и самой подлежащей восстановлению команды, в которой число 7000+*n* заменено числом *n.* Например,

а) для команды

7011 7446 6012 0 11,

принадлежащей нестандартному оператору № 12, константа восстановления будет снабжена индикатой

4012 0000 0000 0 55 —индиката,
0011 7446 6012 0 11—собственно константа восстановления;

б) для команды (имеющей индикату), принадлежащей нестандартному оператору № 12,

0000 4013 0000 0 55 —индиката,
7015 0007 0000 0 16 —команда

константа восстановления будет снабжена индикатой

4012 4013 0000 0 55 —индиката,
0015 0007 0000 0 16 — собственно константа восстановления.

11. Каждый нестандартный оператор снабжается так называемой «характеристикой» (заглавием), записанной в виде команды следующим образом:

| 4000+$N$ | $l_1$ | 7711 | $l_2$ |
|---|---|---|---|
| I адрес | II адрес | III адрес | контрольный знак и код операции |

Здесь $N$ — порядковый номер оператора, $l_1$ — общее количество строк бланка, занятых командами и индикатами, $l_2$ — количество индикат оператора. Число 11, входящее в число 7711 (третий адрес) является типовым признаком нестандартного оператора. Например, характеристика

4012 0025 7711 0 03

будет означать, что вслед за ней записан нестандартный оператор № 12, относящийся к нему материал расположен на 25 строках, число индикат равно 3.

12. При нестандартном программировании можно в адресах команд вместо условных чисел величин писать буквы, которыми эти величины обозначены в расчетных формулах. Например, если $x$ обозначено условным числом 5012, а $y$ — условным числом 5020, то вместо команды

5012 5020 6001 0 05

можно написать команду

$x$ $y$ 6001 0 05.

**5. Организация программирующей программы *ПП-С*.** Как уже говорилось, операторную программирующую программу целесообразно строить из отдельных блоков. Программирующая программа *ПП-С* состоит из следующих блоков.

Блок *К* предназначен для проверки правильности информации, по которой должна быть составлена подлежащая программированию программа. Точнее, этот блок выявляет ряд часто встречающихся ошибок. При наличии таких ошибок он указывает их место в информации, при отсутствии — вызывает очередной блок *ПП-С*.

Блок *А′* производит предварительную обработку информации и подготовляет ее для обработки блоком *А*.

Блоки *Р′* и *Р* вместе образуют логический блок. Блок *Р′* приводит информацию о логических операторах к виду, позволяющему блоку *А* составить команды логических операторов. Блок *Р* работает после блока *А* и придает логическим операторам их окончательный вид.

Блок *А* — арифметический. Составляет арифметические операторы программируемой программы.

Блок *С* — блок стандартных подпрограмм. Включает стандартные подпрограммы в составляемую программу.

Блок *F* — блок переадресации. Составляет операторы переадресации.

Блок *Э* осуществляет экономию рабочих ячеек, используемых арифметическими операторами и операторами переадресации составляемой программы.

Блок *О* — блок восстановления — строит операторы восстановления. (Восстановлением здесь называется занесение вместо измененной команды ее начального вида.)

Блок *П* — блок присвоения действительных адресов — заменяет условные числа, в которых составлялась программируемая программа, действительными адресами и приводит программу к ее окончательному виду.

Каждый блок программирующей программы записан на отдельной зоне магнитной ленты в порядке, представленном в таблице 47. Исходные данные (информация), на основании которых машина должна составить рабочую программу, также записываются на магнитную ленту.

Начинается работа программирующей программы путем вызова с ленты в оперативную память машины исходных данных (информации) и блока *К*. Дальнейший вызов блоков *ПП-С* осуществляется автоматически после работы каждого блока на основании специальной «ведущей таблицы» (*ВТ*), содержащей перечень зон магнитной ленты, хранящих соответствующие блоки.

Т а б л и ц а 47. Размещение блоков *ПП-С* на магнитной ленте

| № п/п | Наименование | Номер зоны магнитной ленты |
|---|---|---|
| 1 | Исходная информация…….. | 4001 |
| 2 | Блок *К*…….… | 4002 |
| 3 | Блок *А*'…… | 4003 |
| 4 | Блок *Р*'……..... | 4004 |
| 5 | Блок *А*…… | 4005 |
| 6 | Зона для частично переработанной информации…….. | 4006 |
| 7 | Блок *Р*…… | 4007 |
| 8 | Блок *С*…… | 4010 |
| 9 | Блок *F*…… | 4011 |
| 10 | Блок *Э*……..… | 4012 |
| 11 | Блок *О*…….…… | 4013 |
| 12 | Зона для частично обработанной информации…….. | 4014 |
| 13 | Блок *П*……. | 4015 |

Правильность получаемых результатов контролируется методом двойного счета, который осуществляется тремя циклами. Вначале информация проверяется блоком *К* и обрабатывается блоками *А′, Р′* и *А* и вычисляется контрольная сумма полученной переработанной информации. После этого снова вызывается исходная информация и блок *К.* Осуществляется повторная обработка информации теми же блоками. Контрольная сумма второго результата сличается с ранее полученной контрольной суммой. Совпадение контрольных сумм считается признаком правильности полученных результатов. Переработанная информация записывается на магнитную ленту (в зону № 4006).

Затем начинается второй цикл контроля, охватывающий блоки *Р, С, F, Э, О.* Третьим циклом контроля охватывается блок *П.*

После работы блока *О* информация превращена уже в программу, но последняя составлена не в действительных адресах, а в специальных кодах — условных числах. В этом виде программа записывается в зону магнитной ленты № 4014 (а в случае надобности выдается блоком *О* также и на перфокарты), после чего автоматически вызывается блок *П.* Последний переводит программу на «язык» действительных адресов и выдает ее на перфокарты уже в окончательном виде.

Блоки программирующей программы должны обрабатывать информацию в следующей последовательности:

*К, А′, Р′, А, Р, С, F, Э, О, П.*

Как уже говорилось, блоки, которые при составлении конкретной рабочей программы не нужны, из этой последовательности исключаются. Осуществляется это следующим образом. Блок *К,* просматривающий информацию, задаваемую для *ПП-С,* устанавливает, какого типа операторы входят в состав программы, подлежащей составлению, и формирует специальную *ведущую таблицу* (кратко — *ВТ*). В ведущей таблице указано, какие блоки *ПП-С* должны участвовать в работе. Окончив свою работу, блок *К* на основании *ВТ* вызывает с магнитной ленты следующий блок. Каждый очередной блок по окончании своей работы также на основании *ВТ* вызывает следующий блок.

## § 50. Подготовка к кодировке информации для *ПП-С*

Подготовка информации для *ПП-С* производится после того, как задача, для которой нужно составить программу, полностью изучена и арифметизирована, т. е. после того, как для нее выбран численный метод решения.

Состоит эта подготовка из следующих этапов:

1) Составление логической схемы из операторов типов

**А, Р, С, F, О, Н**;

2) Составление задания для кодировщиков.

**1. Составление задания для кодировщиков.** Задание для кодировщиков составляется следующим образом.

На специальном бланке выписываются все буквы, присутствующие в расчетных формулах, и поставленные им в соответствие условные числа величин, а также все константы (восстановления, переадресации и числовые), составленные при нестандартном программировании, и поставленные им в соответствие условные числа констант*. После заполнения этот бланк называется *словарем* задачи.

Присутствующие в расчетных формулах числа (коэффициенты) можно, так же как и присутствующие в формулах буквы, обозначить условными числами величин. Но можно им условных чисел не присваивать. Тогда они считаются числовыми константами. Присвоение условных чисел числовым константам производится автоматически самой *ПП-С.*

* Константа восстановления, состоящая из индикаты собственно константы, обозначается одним условным числом (а не двумя)

Затем последовательно выписываются символы, обозначающие операторы логической схемы, и вслед за каждым из них — информация о содержании этих операторов.

Вслед за информацией о всех операторах, входящих в логическую схему, выписываются таблицы запасенных констант (восстановления, переадресации и числовых).

Составляется *таблица зависимости величин от параметров* (*ТЗП*), необходимая для работы блока *F*.

В простейших случаях, когда возможно так называемое стандартное распределение памяти (подробно ниже, на стр. 262), задается информация о распределении памяти.

В более сложных случаях эта информация может быть составлена только после того, как *ПП-С* составит программу в условных числах.

И н ф о р м а ц и я об о п е р а т о р е **А**. Информация об арифметических операторах записывается в виде математических формул, по которым должен вестись счет. При записи формул приняты следующие правила.

1. Буква, обозначающая результат, ставится справа от знака равенства. Правая часть формулы должна состоять из одной буквы.

2. В формуле $a = b$ знак равенства имеет смысл знака переноса числа $a$ в ячейку ($b$). Такая формула эквивалентна команде

$$< a > \quad 0000 \quad < b > \quad 0 \qquad 13.$$

3. Кроме букв и чисел, в математических формулах в качестве величин, над которыми производятся операции, могут фигурировать также номера операторов (в формуле пишут символы, обозначающие операторы в логической схеме, например $\mathbf{A}_{10}$, $\mathbf{H}_{17}$ и т. д.).

4. В математических формулах могут содержаться все арифметические и логические операции, предусмотренные в машине *Стрела,* а также одноместные операции (операции, производимые над одной величиной), выполняемые машиной по стандартным подпрограммам, закоммутированным в *НСП,* такие как $\sin x$, $\ln x$ и т. д. Некоторые из операций, предусмотренных в машине, не имеют общепринятых обозначений для знаков действия. В качестве знака такой операции применяется взятый в кавычки код операции. Например, допустима формула

$$(x+y) \text{ «16» } z = r.$$

Здесь «16» означает, что необходимо над числами ($x + y$) и $z$ произвести поразрядную операцию отрицания равнозначности. Исключение составляет операция *сдвиг по адресу;* ее знак действия имеет вид «114». Например, возможна формула

$$a \text{ «114» } 4105 = r,$$

которая эквивалентна команде

$$<a> \quad 4105 \quad < r > \quad 1 \qquad 14.$$

5. Кроме операций, предусмотренных в машине *Стрела,* математические формулы могут содержать также следующие операции:

а) деление $a : b$;

б) вычисление модуля числа (точнее, абсолютной величины) $|a|$;

в) возведение в целую положительную степень $a^n$;

г) изменение знака на противоположный; символ «—» (минус), если он стоит в начале формулы, после открытой скобки, после знака одноместной операции, после левого знака абсолютной величины, означает операцию изменения знака.

6. Знак умножения в формулах не пишется (например, вместо $a \cdot b$ необходимо писать $ab$).

7. Математические формулы должны быть записаны в одну строку (например, черта дроби должна быть заменена знаком «:» и т. п.).

8. Выражения, являющиеся аргументами одноместных операций (таких, как степень, $\sin x$, $\ln x$), заключаются в скобки, если они состоят более чем из одной буквы или числа. Знак возведения в степень ставится после выражения, возводимого в степень. Знаки остальных одноместных операций ставятся впереди своих аргументов.

9. В тех случаях, когда порядок действия является общепринятым или безразличен, скобки не ставятся (например, $a + b + c + d$ или $a—b+c : d$ или $a \sin b$). Во всех остальных случаях порядок действий определяется скобками. Все скобки, применяемые в формулах, являются круглыми. При этом необходимо, чтобы число открытых скобок было равно числу закрытых.

10. Наличие в математической формуле тривиальных скобок не допускается. Тривиальными скобками мы называем пару скобок, содержащих внутри себя только одну величину, например (а).

И н ф о р м а ц и я об о п е р а т о р е **Р**. Знаки $<$ (меньше), $\leq$ (не больше), $>$ (больше), $\geq$ (не меньше), $\equiv$ (поразрядно совпадают), $\neq$ (не равно, т. е. нет поразрядного совпадения) называются *знаками условий.*

*Комплексом* будем называть либо конечную последовательность математических формул, составленных по правилам, приведенным в описании информации для арифметических операторов, либо конечную последовательность математических формул, в конце которой поставлено математическое выражение (левая часть математической формулы). Если последним элементом комплекса является математическая формула, то это не должна быть формула вида $a = b$. В частном случае комплекс может состоять из одной формулы, одного выражения, одного числа или одной буквы (буква и число являются частными видами математических выражений).

Например, комплексами являются:

— число 25;
— буква *x;*
— математическое выражение $x^2+y^2$;
— математическая формула $x^2 + y^2 = r$;
— последовательность математических формул

$$x^2 + 3yz = r_1, \qquad x^2 + y^2 = r_2;$$

— последовательность математических формул, в конце которой поставлено математическое выражение:

$$x^2+3yz = r_1, \qquad x^2 + y^2 = r_2, \qquad 3x+\sin y.$$

*Главной частью комплекса* называется математическое выражение, которым оканчивается комплекс, или левая часть последней формулы, которой оканчивается комплекс.

Два комплекса, соединенные знаком условия, называются *условием.* Вот примеры условий:

$$x>2;$$
$$a \neq b;$$
$$x^2+y^2 \leq 5;$$
$$x + \sin x = r, \qquad x^2 + y^2 < x + 2z;$$
$$x^2 + y^2 = r \equiv a;$$
$$x + y = r_1, \qquad x^2 + y^2 = r_2 < x + 2z = r_3.$$

Логическое содержание условия получим, вычеркивая из комплексов, образующих это условие, все, кроме их главных частей. Например, логическое содержание условий

1) $x^2+ y^2 = r_1, \qquad x + 2<y + z = r_2,$
2) $x^2+y^2 = r_1, \qquad x^3 + x = r_2 \geq y+ z = r_3, \qquad y^2 + 3$

будет следующим:

1) $x + 2<y + z,$
2) $x^3 + x \geq y^2+3.$

Математические формулы, входящие в комплексы, означают, что, кроме проверки условия для определения направления передачи управления, необходимо произвести вычисления, указанные в этих формулах и результаты сохранить (например, для других вычислений).

Например, если в программируемой программе нужно предусмотреть только проверку условия, гласящего, что $x^2 + y^2$ меньше, чем 3, то условие записывают в виде

$$x^2+y^2<3$$

Если, кроме проверки этого условия, необходимо получить значение выражения $x^2 + y^2$ (например, для того, чтобы подставить в какую-либо дальнейшую математическую формулу), то условие записывают так:

$$x^2 + y^2 = r <3.$$

Если нужно вычислить значения величин $x^2 + y^2$, $x + 3z$ и проверить выполнение неравенства

$$x + y < x + 3z,$$

то пишут:

$$x^2 + y^2 = r_1, \qquad x + y < x + 3z = r_2.$$

Допускается также применение сложных условий, представляющих собой три комплекса, соединенных согласованными знаками неравенств. Например:

$$f(x) < \varphi(x) \leq \theta(x),$$
$$f(x) = r_1<\varphi(x) = r_2 \leq \theta(x)$$

и т. п.

Строка условий, соединенных знаками логических связей, называется *логической формулой* (в частности, логическая формула может состоять из одного условия). Отметим, что упомянутые выше сложные условия фактически являются уже простейшими логическими формулами. Например:

$$x^2+y< x^3+1\leq x +y^2$$

означает:

$$x^2 + y < x^3 + 1 \wedge x^3 + 1 \leq x + y^2$$

При необходимости указания порядка логических связей используются так называемые логические скобки. В качестве логических скобок применяются квадратные скобки (напомним, что для указания порядка действий в математических формулах применяются только круглые скобки).

Количество открытых квадратных скобок в логической формуле должно быть равно количеству закрытых квадратных скобок. При составлении логических формул применяются только два вида логических связей «/\» (*и*) и «\/» (*или*). В знаке «—» (*не*) необходимости нет. Он исключается путем соответствующей формулировки условия. Например, вместо условия $\overline{x > 2}$ (*не* «$x$ >2») обычно пишут $x\leq 2$.

Вот примеры логических формул:

$$x>0 \wedge x^2+y^2=r <3,$$

$$[x>0 \vee y<5] \wedge x^2+y^2\neq 6$$ и т. п.

Информация для составления логического оператора представляет собой последовательность занумерованных

логических формул, после каждой из которых стоит так называемый *знак операторной связи.*

Знак операторной связи представляет собой пару условных чисел, каждое из которых может обозначать, либо номер логической формулы, входящей в информацию о данном логическом операторе, либо номер оператора. Первое из чисел знака операторной связи указывает, куда должно быть передано управление, если значение истинности логической формулы равно 0, а второе — куда должно быть передано управление, если значение истинности логической формулы равно 1.

Например, информация для составления логического оператора может иметь такой вид:

I. $\underbrace{x > 0}_{\text{1-ая логическая формула}}$ $\underbrace{7002,\quad 4005}_{\text{Знак операторной связи}}$

II. $\underbrace{x^2 + y^2 = r < 5 \wedge x - z > 3}_{\text{2-ая логическая формула}}$ $\underbrace{4012,\quad 4015}_{\text{Знак операторной связи}}$

Эта информация имеет следующее содержание:

«Логический оператор должен проверить условие, составляющее формулу I:

$$x > 0.$$

Если это условие не выполнено, то необходимо проверить сложное условие, представленное формулой II. Если условие I выполнено, то управление должно быть передано оператору № 5.

Формула II истинна, если одновременно выполняются два условия:

$$x^2 + y^2 = r < 5,$$
$$x - z > 3.$$

Если формула II имеет значение истинности 0, то управление должно быть передано оператору № 12; в противном случае управление нужно передать оператору № 15. Величина $x^2+y^2$ вычисленная для проверки условия, нужна в дальнейшем для вычислений».

И н ф о р м а ц и я об о п е р а т о р е C. Информацией о стандартной подпрограмме является таблица соответствия между адресами подпрограммы и условными числами, принятыми в данной задаче. Эта таблица имеет, например, такой вид:

*СПП* № 7

1) 0020 — 0100 — 5001,
2) 0101— 0105 — 6003,
3) 0106 — 4605,
4) 0501 — 4012,
5) 0502 — 4017.

Приведенная таблица имеет следующее содержание:

«В программируемую программу должна быть включена стандартная подпрограмма (*СПП*), имеющая в каталоге № 7. Ячейки подпрограммы, начиная с 0020 и кончая 0100, соответствуют условным числам величин, начиная с 5001 (и кончая 5101); ячейки, начиная с 0101 и кончая 0105, соответствуют рабочим ячейкам, начиная с 6003 (и кончая 6007); ячейка 0106 отвечает условному числу константы переадресации 4605. Ячейке 0501 отвечает условное число оператора 4012, а ячейке 0502 — условное число оператора 4017».

В правом столбце таблицы соответствия могут стоять условные числа операторов, например при наличии команд условной передачи управления в конце или внутри стандартной программы.

И н ф о р м а ц и я об о п е р а т о р е **F**. Информация об операторе переадресации содержит указания о том, какие операторы и по каким параметрам должны быть переадресованы оператором переадресации. Например:

1) $\mathbf{A}_{12} — \mathbf{H}_{17};\ i;$
2) $\mathbf{A}_{20};\ j;$
3) $\mathbf{A}_{31} — \mathbf{A}_{35};\ k.$

Приведенная таблица имеет следующее содержание: «Переадресовать по параметру *i* все операторы, начиная с 12-го и кончая 17-м. Оператор $\mathbf{A}_{20}$ переадресовать по параметру *j*. Все операторы, начиная с 31-го и кончая 35-м, переадресовать по параметру *k* ».

И н ф о р м а ц и я об о п е р а т о р е **O**. Информация об операторе восстановления представляет собой перечень номеров операторов, которые должны быть восстановлены. Например:

1) $\mathbf{H}_{20},$
2) $\mathbf{A}_{22} — \mathbf{A}_{23}.$

Эта таблица означает: «Восстановить оператор $\mathbf{H}_{20}$ и все операторы, начиная с 22-го и кончая 23-м».

Стандартные операторы восстановления могут быть составлены программирующей программой только для восстановления операторов, для которых составлены блоком **F** стандартные операторы переадресации.

И н ф о р м а ц и я об о п е р а т о р а х **H**. Информацией о нестандартных операторах являются сами эти операторы, составленные вручную программистом.

Отметим, что при нестандартном программировании можно вместо условных чисел в адресах команд писать

буквы, фигурирующие в расчетных формулах. Замена букв условными числами является формальной работой, которую без труда выполняют кодировщики.

Т а б л и ц а *ТЗП*. Таблица зависимости от параметров *ТЗП* содержит указания того, какие величины от каких параметров зависят, является шаг переадресации постоянной или переменной величиной, чему он равен (если он постоянный) или каким условным числом обозначен (если он переменный), нужно ли его прибавлять или вычитать. Например:

От *i* зависят:

| | |
|---|---|
| 5001—5010; | *h* (шаг переадресации)= 1 (плюс); |
| 5020 — 5025 | *h* = 3 (минус); |
| 5027; | *h* = 2 (плюс); |
| 5010 — 5014; | переменный, равен величине 5077. |

От *j* зависят: ...

и т. д.

Р а с п р е д е л е н и е п а м я т и . *ПП-С* рассчитана на два вида распределения памяти: стандартное и нестандартное.

При стандартном распределении памяти рабочие ячейки получают действительные адреса начиная с 0010. Вслед за рабочими ячейками располагается программа, за нею константы восстановления, константы переадресации, числовые константы. Вслед за последними располагаются величины.

Нестандартное распределение памяти — это произвольное распределение, сделанное самим программистом. Нестандартное распределение памяти предполагает лишь одно ограничение на размещение в памяти элементов программируемой программы: ограничение, состоящее в неделимости массивов а) программы, б) констант восстановления, в) констант переадресации, г) числовых констант, д) рабочих ячеек.

В задании для кодировщиков в случае, если программист принимает решение о целесообразности стандартного распределения памяти, делается запись: «Распределение памяти стандартное».

Затем в словаре задачи против условных чисел проставляются условные адреса величин (т. е. составляется таблица распределения величин — *ТРВ*).

Условные адреса величин составляются так, будто величины должны быть расположены в ячейках, начиная с ячейки 0001. Действительные адреса величин будут определены самой *ПП-С* в соответствии со стандартным распределением памяти.

Если программист избрал нестандартное распределение памяти, то в задании для кодировщиков он делает надпись: «Распределение памяти нестандартное».

В этом случае в словаре задачи против условных чисел должны быть проставлены действительные адреса величин (составлена *ТРВ*) и, кроме того, должна быть составлена *ТРП* (таблица распределения памяти), в которой приведены истинные начальные адреса неделимых массивов программируемой программы Например:

| | |
|---|---|
| программа | — 0030, |
| константы восстановления | — 0557, |
| константы переадресации | —0301, |
| числовые константы | —3132, |
| рабочие ячейки | —3170. |

Внесение истинных адресов в словарь задачи и составление *ТРП* (при нестандартном распределении памяти) не всегда могут быть произведены заблаговременно, раньше чем составлена программа в условных числах. Это может быть обусловлено тем, что заранее неизвестны количество команд программы и количество констант каждого вида. В этом случае *ТРП* и *ТРВ* в задание для кодировщиков не включаются. Программист составит их после изучения программы, которую он получит в условных числах, после работы блока *О*. При этом блок *П* должен работать уже после ввода в машину *ТРП* и *ТРВ*.

**2. Пример подготовки информации к кодировке.** Для пояснения правил, по которым производится составление задания кодировщиком, рассмотрим следующий пример.

Требуется получить программу составления таблицы с двумя входами значений функции $F(x, \gamma)$, где $x$ — независимое переменное, а $\gamma$ — постоянный параметр (не смешивать с параметрами циклов).

Дано

$$F(x,\gamma) = \frac{x\ln x + 2{,}1\sqrt{x} + 1{,}5\gamma^2 + \gamma\sqrt{3{,}25x^2\sin x + x + 1}}{\gamma + 1{,}7 + e^x + \operatorname{arctg} x}$$

Таблицу надо составить для заданной последовательности значений $x$ ($x > 0$):

$$x_1, x_2, \ldots, x_i, \ldots x_{200}.$$

и заданной последовательности значений $\gamma$:

$$\gamma_1, \gamma_2, \ldots, \gamma_j, \ldots, \gamma_{100}$$

(индексы записаны в восьмеричной системе). Функция $F(x, \gamma)$ может принимать как действительные, так и мнимые значения.

Составление логической схемы. Программист решает:

1) рассматривать вычисление всех значений функции $F(x,\gamma)$, отвечающих фиксированному значению параметра $\gamma$, как вариант решения задачи (при $\gamma = \gamma_i$ будет считаться, что вариант имеет номер $j$);

2) правильность решения каждого варианта проверять методом двойного счета (см. § 40), после чего выдавать результаты на перфокарты;

3) для того чтобы после перерыва в решении задачи (например, из-за сбоя в работе машины или по другой причине) можно было продолжать решение, начиная с того варианта, который следует за последним, уже решенным вариантом, воспользоваться приемом, известным под названием способа передаточных перфокарт (см. § 41);

4) так как значения функции могут оказываться комплексными, отводить для каждого значения функции по две ячейки: одну для действительной части, а другую для мнимой части.

Исходя из этих соображений, составляется логическая схема программы:

$$\mathbf{F}_1(k)\,\mathbf{P}^i_2\mathbf{A}^{j,k}_3;\ \mathbf{A}^{j,k}_4\mathbf{A}^i_5\mathbf{F}_6(i)\mathbf{P}_7\mathbf{H}_{10}\mathbf{O}_{11}(i)\,\mathbf{H}_{12}\mathbf{F}_{13}(j)\,\mathbf{P}_{14}\mathbf{H}_{15}. \qquad (X.1)$$

Нумерация операторов в этой логической схеме произведена в восьмеричной системе счисления. Здесь

$\mathbf{F}_1(k)$ увеличивает все адреса команд, зависящие от параметра $j$, на передаточную константу $\chi$ и тем самым настраивает программу на решение задачи с $(\chi + 1)$-го варианта. Таким образом, считается, что от параметра $k$ зависят те же величины, что и от параметра $j$;

$\mathbf{P}^i_2$ ведет счет по формулам

$$u = x_i \ln x_i + 2{,}1\sqrt{x_i},$$

$$w = 1{,}7 + e^{xi} + \operatorname{arctg} x_i,$$

$$\upsilon = 3{,}25x_i^2 \sin x_i + x_i + 1$$

и проверяет выполнение условиям $\upsilon < 0$; если $\upsilon < 0$, он передает управление $A_4^{j,k}$, в противном случае — оператору $\mathrm{A}_3^{\mathrm{j,k}}$;

$\mathrm{A}_3^{\mathrm{j,k}}$ засылает нуль в ответную ячейку $<\overline{\psi}>$ (признак того, что ответ — действительное число) и ведет счет по формуле

$$F(x_i, \gamma_i) = \overline{\varphi} = \frac{u + 1{,}5\gamma_j^2 + \gamma_j\sqrt{\upsilon}}{\gamma_j + w}$$

передает управление оператору $\mathbf{A}^i_5$;

$A_4^{j,k}$ ведет счет по формулам

$$\operatorname{Re} F(x_{i,}\gamma_j) = \overline{\varphi} = \frac{u + 1{,}5\gamma_j^2}{\gamma_i + w},$$

$$\operatorname{Im} F(x_i, \gamma_j) = \overline{\psi} = \frac{\gamma_j\sqrt{-\upsilon}}{\gamma_i + w};$$

$\mathbf{A}^i_5$ переносит содержимое ячеек $<\overline{\varphi}>$ и $<\overline{\psi}>$ соответственно в ячейки $<\varphi_i>$ и $<\psi_i>$;

$\mathbf{F}_6(i)$ переадресует по параметру $i$ (величину $x_i$ с шагом, равным 1, величины $\varphi_i$ и $\psi_i$ — с шагом 2);

$\mathbf{P}_7$ — логический оператор — проверяет с помощью счетчика параметра $i$, для всех ли значений величины $x$ получены значения функции $F(x_i,\gamma_j)$;

$\mathbf{H}_{10}$ восстанавливает счетчик параметра $i$;

$\mathbf{O}_{11}(i)$ восстанавливает операторы $\mathbf{P}^i_2$ и $\mathbf{A}^i_5$;

$\mathbf{H}_{12}$ переводит результаты в десятичную систему, осуществляет контрольный счет и выдает результаты варианта на перфокарты; вычисляет передаточную константу $\chi$ и выдает ее на отдельную перфокарту;

$\mathbf{F}_{13}(j)$ переадресует по параметру $j$ (величину $\gamma_j$ с шагом, равным 1);

$\mathbf{P}_{14}$ логический оператор, проверяющий, для всех ли значений параметра $\gamma$ выполнены вычисления; если не для всех, то он передает управление оператору $\mathbf{P}^i_2$; если для всех, то передает управление нестандартному оператору $\mathrm{H}_{15}$;

$\mathbf{H}_{15}$ — останов машины.

Составление словаря. Словарь рассматриваемой задачи представлен таблицей 48.

Т а б л и ц а 48. Составление словаря

| Условное число | Буква или числа | Адрес | Условное число | Буква или числа | Адрес |
|---|---|---|---|---|---|
| 5001 | $x_i$ | | 5012 | $c$ | |
| 5002 | $\gamma_i$ | | 5013 | $c_1$ | |
| 5003 | $u$ | | 5014 | $c_2$ | |
| 5004 | $v$ | | 5015 | $d$ | |
| 5005 | $w$ | | 5016 | $d_1$ | |
| 5006 | $\overline{\varphi}$ | | 5017 | $d_2$ | |
| 5007 | $\overline{\psi}$ | | 5020 | $\mathbf{\Sigma}_1$ | |
| 5010 | $\varphi_i$ | | 5021 | $\mathbf{\Sigma}_2$ | |
| 5011 | $\Psi_i$ | | 5022 | $\chi$ | |

В этой таблице $c$ и $c_1$, обозначают одну и ту же ячейку счетчика значений параметра $i$. Одна и та же ячейка обозначена двумя буквами потому, что в дальнейшем содержимое этой ячейки будет исходной величиной для вычисления по формуле

$$c \text{ «02» } 7423 = c_1,$$

а результат вычисления надо будет записать в эту же ячейку. При распределении памяти обоим условным числам,5010 и 5011, ставится в соответствие один и тот же действительный адрес. $c_2$ — вторая ячейка счетчика значений параметра $i$. Аналогично $d$, $d_1$ и $d_2$ означают две ячейки счетчика значений параметра $j$. $\mathbf{\Sigma}_1$ и $\mathbf{\Sigma}_2$ — первая и вторая контрольные суммы.

Составление основного массива информации

$$\underline{\mathbf{F}_1(k)}$$

Переадресовать по $k$:

1) $\mathbf{A}_3^{j,k} - \mathbf{A}_4^{j,k}$.

$$\underline{\mathbf{P}_2^i}$$

$$x_i \ln x_i + 2{,}1\sqrt{x_i} = u,$$

$$1{,}7+e^{xi}+\operatorname{arctg} x_i = w$$

$$3{,}25\, x_i^2 \sin x_i + x_i + 1 = v < 0; \quad \mathrm{A}_3^{j,k}\mathrm{A}_4^{j,k}.$$

$$\underline{\mathbf{A}_3^{j,k}}$$

$$5000 = \overline{\psi}$$

$$(u+1{,}5\,\gamma_j^2 + \gamma_j\sqrt{v}) : (\gamma_j + w) = \overline{\varphi},$$

4005 «20» 4005=5000

$$\underline{\mathbf{A}_4^{j,k}}$$

$$(u+1{,}5\,\gamma_j^2) : (\gamma_j + w) = \overline{\varphi},$$

$$\gamma_j\sqrt{(-v)} : (\gamma_j + w) = \psi.$$

$$\underline{\mathbf{A}_5^i}$$

$$\overline{\varphi} = \varphi_i, \overline{\psi} = \psi_i.$$

$$\underline{\mathbf{F}_6(i)}$$

Переадресовать по $i$

1) $\mathrm{P}_2^i$,

2) $\mathrm{A}_5^i$.

$$\underline{\mathbf{P}_7}$$

$c$ «20» $7423 = c_1 \neq c_2$ 4010, 4002.

**Н**$_{10}$

1) 0000 0000 5012 0 13

**О**$_{11}$

Восстановить:

1) $P_2^i$,

2) $A_5^i$.

**Н**$_{12}$

| | | | | | |
|---|---|---|---|---|---|
| | 4012 | 0015 | 7711 | 0 | 00 — характеристика. |
| 1) | 5010 | 0377 | 5010 | 0 | 70 |
| 2) | 0000 | 0377 | 5020 | 0 | 34 |
| 3) | 5010 | 5020 | 5020 | 0 | 17 |
| 4) | 0000 | 0000 | 0000 | 0 | 00 |
| 5) | 5020 | 0000 | 5021 | 0 | 13 |
| 6) | 4002 | 4002 | 7004 | 0 | 27 |
| 7) | 5020 | 5021 | 0000 | 0 | 16 |
| 10) | 7012 | 7011 | 0000 | 0 | 20 |
| 11) | 5020 | 5021 | 0000 | 0 | 40 |
| 12) | 5010 | 0377 | 0000 | 0 | 44 |
| 13) | 5015 | 7431 | 5022 | 0 | 02 |
| 14) | 0000 | 0200 | 0000 | 0 | 45 |
| 15) | 5022 | 0000 | 0000 | 0 | 44 |

Предпоследняя команда этого оператора служит для того, чтобы разделить во времени выдачу на перфокарты результатов счета и передаточной константы. Без этой команды передаточная константа будет пробита не на отдельной перфокарте, а на последней перфокарте результатов.

**F**$_{13}$($j$)

Переадресовать по $j$:

1) $A_3^{j,k} - A_4^{j,k}$

**Р**$_{14}$

$d$ «02» 7431 $= d_1 \neq d_2$ 4015, 4002.

**Н**$_{15}$

1) 0000 0000 0000 0 40.

**Т**$_{16}$

Запасенных констант восстановления нет.

**Т**$_{17}$

Запасенных констант переадресации нет.

**Т**$_{20}$

Запасенных числовых констант нет.

*ТЗП*

От параметра $i$ зависят:

$x_{i;}$ $h = 1$ (плюс);

$\varphi_i$, $\psi_i$; $h = 2$ (плюс),

От параметра $j$ зависит:

$\gamma_j$; $h = 1$ (плюс).

От параметра $k$ зависит:

$\gamma_j$ — шаг переменный, записан в ячейке 5022 (плюс).

Распределение памяти. Избираем стандартное распределение памяти. Об этом делается соответствующая запись в конце задания кодировщикам. Кроме того, производится условное присвоение адресов величинам, вносимое в словарь. Словарь при этом принимает вид, представленный таблицей 49.

Таблица 49. Таблица распределения величин {ТРВ)

| Условное число | Буква или число | Адрес | Условное число | Буква или число | Адрес |
|---|---|---|---|---|---|
| 5001 | $x_i$ | 0003 | 5013 | $c_1$ | 0311 |
| 5002 | $\gamma_j$ | 0203 | 5014 | $c_2$ | 0001 |
| 5003 | $u$ | 0304 | 5015 | $d$ | 0303 |
| 5004 | $v$ | 0305 | 5016 | $d_1$ | 0303 |
| 5005 | $w$ | 0306 | 5017 | $d_2$ | 0002 |
| 5006 | $\overline{\varphi}$ | 0307 | 5020 | $\mathbf{\Sigma}_2$ | 0313 |
| 5007 | $\psi$ | 0310 | 5021 | $\mathbf{\Sigma}_1$ | 0314 |
| 5010 | $\varphi_i$ | 0315 | 5022 | $\chi$ | 0312 |
| 5011 | $\Psi_i$ | 0316 | | | |
| 5012 | $c$ | 0311 | $a_{\max} = 0316$ | | |
| | | | $m = 377$ | | |

В этой таблице $a_{\max}$ — наибольший из условных адресов, приведенных в третьем столбце словаря задачи, $m$ — число ячеек, следующих за ячейкой, отвечающей условному адресу $a_{\max}$, и отведенных для хранения значений величин, предусмотренных в программе. Фактически у нас $m$— количество ячеек, отведенных под ответы и следующих за первой из ячеек величины $\psi_i$ (числа $\varphi_i$ и $\psi_i$ стоят через одно).

## § 51. Кодировка информации для *ПП-С*

Кодировка информации для *ПП-С* состоит главным образом в замене чисел и букв в информации, составленной программистами, условными числами.

Информация, составленная программистом, состоит из трех массивов. Основной массив, далее будем называть его первым, содержит описание логической схемы и входящих в нее операторов. Второй массив содержит *ТЗП,* третий массив — *ТРВ* и *ТРП.*

Для обеспечения правильности кодировки ее проводят одновременно в две руки. По окончании кодировки каждого массива производится сравнение двух вариантов, которые должны быть тождественными и в этом случае признаются правильными.

При кодировке информации она переписывается на стандартные бланки для программирования. С этих бланков в дальнейшем производится перенос информации на перфокарты и ввод в ячейки памяти машины. При этом содержимое каждой строки бланка вводится в отдельную ячейку. Такое взаимнооднозначное соответствие между ячейками части памяти машины, отведенной для информации, и строками бланка, на которых кодировщик записывает информацию в условных числах, позволяет говорить «номер строки» вместо «номер ячейки».

Каждая строка бланка разбита на участки, отвечающие трем адресам команд, контрольному знаку и коду операции. Эти участки мы будем для краткости называть соответственно адресами, контрольным знаком и кодом операции. Каждое условное число кодировщик записывает в одном адресе.

Первый массив информации имеет следующую структуру. В первой строке массива записывается характеристика первого из операторов логической схемы. Вслед за ней производится запись закодированной информации об этом операторе. Во втором адресе характеристики ставится число $l_1$ — количество строк, следующих за характеристикой и занятых информацией об операторе.

Затем выписывается характеристика второго оператора и за ней информация о втором операторе и т. д. После того как выписана характеристика последнего оператора логической схемы и информация о нем, выписывают характеристику таблицы констант восстановления и все константы восстановления, характеристику таблицы констант переадресации и все константы переадресации и, наконец, характеристику таблицы числовых констант и все числовые константы. Таблицы констант могут не содержать запасенных констант. В этом случае характеристики таблиц все равно должны быть включены в первый массив информации. Константы могут присутствовать в таблицах, если они были запасены при нестандартном программировании. Числовые константы при этом должны быть записаны в десятичной системе счисления, в обычном виде, применяемом для записи на бланках десятичных чисел. Перевод числовых констант в двоичную систему счисления будет произведен самой *ПП-С.*

Таким образом, первый массив информации состоит из частей, каждая из которых «озаглавлена» характеристикой и содержит информацию об отдельном операторе или является таблицей констант. Такую часть мы будем в дальнейшем называть *оператором первого массива* (информации). В результате работы *ПЛ-С* операторы массива перерабатываются в операторы составляемой программы.

Вторым массивом информации является *ТЗП* (таблица зависимости величин). Этот массив используется *ПП-С* при обработке ею первого массива. Третий массив информации состоит из *ТРВ* (таблицы распределения величин), а в случае нестандартного распределения памяти, кроме того, содержит *ТРП* (таблицу распределения памяти).

**1. Первый массив информации**

К о д и р о в к а и н ф о р м а ц и и об о п е р а т о р е **А**. Составляется характеристика арифметического оператора с незаполненным вторым адресом:

| | | | | |
|---|---|---|---|---|
| 4000 + *N* | | 7701 | 0 | 00 |

Формулы, которыми представлен арифметический оператор, кодируются, т. е. заменяются последовательностями условных чисел с помощью словаря задачи. Условные числа последовательно записываются в адреса строк бланка вслед за характеристикой.

Следующие подряд несколько однотипных скобок обозначаются одним условным числом вида 0400 + *n* (открытые скобки) или 0200 + *n* (закрытые скобки). Например:

((((— обозначаются условным числом 0404
))) — обозначаются условным числом 0203

Встречающиеся в формулах числовые коэффициенты, Не включенные в словарь задачи, записываются на бланке в отдельной строке в том виде, который принят для записи десятичных чисел, подлежащих вводу в память машины *Стрела.* При этом один или два адреса предыдущей строки бланка могут оказаться пустыми. Эти пустые адреса заполняются восьмеричными числами 0000. Если порядок десятичного нормализованного числа равен нулю, то считают его равным отрицательному нулю, т. е. на бланке в графе, отведенной пол контрольный знак, пишут 1. Например, формула

$$a + b + 1{,}6 = R$$

при условии, что буквам $a$, $b$ и $R$ соответствуют условные числа 5010, 5011 и 5020, будет кодироваться так:

| | | | | |
|---|---|---|---|---|
| 5010 | 0001 | 5011 | | |
| 0001 | 0000 | 0000 | | |
| +160 | 000 | 000 | + | 01 |
| 0077 | 5020 | | | |

При тех же обозначениях формула

$$a + b + 0{,}16 = R$$

будет кодироваться так:

| | | | | |
|---|---|---|---|---|
| 5010 | 0001 | 5011 | | |
| 0001 | 0000 | 0000 | | |
| +160 | 000 | 000 | 1 | 00 |
| 0077 | 5020 | | | |

Если среди коэффициентов математических формул встречаются числа, закоммутированные в *УВК,* то целесообразно в качестве отвечающих им условных чисел брать адреса *УВК.*

После того как все формулы, входящие в арифметический оператор, закодированы, производится подсчет в восьмеричной системе строк, занятых условными числами (не считая характеристики). Полученное число записывают во второй адрес характеристики.

К о д и р о в к а и н ф о р м а ц и и о **Р**. Перед информацией о логическом операторе ставят характеристику с незаполненным вторым адресом:

| | | | |
|---|---|---|---|
| 4000 + *N* \| | 7704 | 0 | 00 |

Все символы, входящие в формулы, приведенные в задании для кодировщиков, заменяют условными числами в соответствии со словарем задачи и обозначениями, принятыми для знаков действий, знаков условий и знаков логических связей. Информация записывается в «адресах» строк бланков непосредственно вслед за характеристикой. Правила записи математических формул и выражений, входящих в логическую информацию, те же, что при кодировке арифметических операторов.

После окончания кодировки может оказаться, что в последней строке не все адреса заняты условными числами. Их следует заполнить числами 0000. Затем подсчитывают в восьмеричной системе счисления количество строк, занятых информацией (без характеристики). Полученное число вписывают во второй адрес характеристики.

К о д и р о в к а и н ф о р м а ц и и о **С.** Составляется характеристика

| | | | | |
|---|---|---|---|---|
| 4000 + *N* | | 7005 | | |

с незаполненными разрядами второго адреса, контрольного знака и кода операции.

Вслед за характеристикой записывается так называемая переменная информация о стандартной программе (таблица соответствия адресов стандартной подпрограммы условным числам программируемой программы). Эта информация получается путем кодирования информации, приведенной в задании кодировщикам, и имеет такой вид:

| | | | | |
|---|---|---|---|---|
| $\alpha$ | $\omega$ | $\gamma$ | 0 | 00 |

где $\alpha$ и $\omega$ — меньший и больший из группы адресов, принадлежащих стандартной подпрограмме, а $\gamma$ — условное число, отвечающее $\alpha$. Эта запись означает, что адреса $\alpha$, $\alpha + 1$, $\alpha + 2$, ..., $\alpha + n = \omega$, относящиеся к стандартной программе, соответствуют последовательным условным числам, начиная с $\gamma$, т. е. условным числам $\gamma$, $\gamma + 1$, $\gamma + 2$, ..., $\gamma + n$. Если группа адресов, принадлежащих стандартной подпрограмме, состоит из одного адреса, то в приведенной записи должно быть $\omega = \alpha$. В этом случае строка информации принимает такой вид:

| | | | | |
|---|---|---|---|---|
| $\alpha$ | $\alpha$ | $\gamma$ | 0 | 00 |

После того как переменная информация вся закодирована, производится подсчет в восьмеричной системе количества занятых ею строк (характеристика не считается) и полученное число записывается в разрядах контрольного знака и кода операции характеристики.

Если после записи переменной информации остаются незаполненные строки в части бланка, отвечающей отдельной перфокарте, эти строки обводятся фигурной скобкой, возле которой делается надпись: «без маркеров».

Эта надпись делается для того, чтобы оператор внешних устройств, который будет переносить информацию на перфокарты, не пробил на оставшихся свободных строках перфокарты нулей. Непосредственно после перфокарты, хранящей переменную информацию о стандартной подпрограмме, будет сложена колода перфокарт со стандартной подпрограммой, взятая из библиотеки стандартных подпрограмм.

В каталоге стандартных подпрограмм отыскивают паспорт необходимой стандартной подпрограммы и указанную в нем (в восьмеричной системе) «длину» (количество строк) стандартной подпрограммы складывают с числом, ранее записанным в разрядах контрольного знака и кода операции характеристики. Полученное число записывают во второй адрес характеристики. Это число представляет собой общую «длину» (количество ячеек памяти) переменной информации и стандартной подпрограммы.

К о д и р о в к а и н ф о р м а ц и и об **F.** Составляется характеристика

| | | | | |
|---|---|---|---|---|
| 4000+$N$ | $l_1$ | 7702 | 0 | 00 |

где $l_1$ — количество строк информации об операторе переадресации, приведенной в задании кодировщикам.
Приведенные в задании кодировщикам символы и буквы, обозначающие параметры, заменяются условными числами в соответствии со словарем и переписываются вслед за характеристикой. При этом каждая строка имеет такой вид:

| | | | |
|---|---|---|---|
| $4000 + N_1 \mid 4000 + N_2$ | $7700 + n$ | 0 | 00 |

Если в задании для кодировщиков в строке стоит лишь один символ оператора, отвечающее ему условное число пишется дважды: в первом и втором адресах строки.

К о д и р о в к а и н ф о р м а ц и и об **O**. Кодировка информации об операторе восстановления производится по тем же правилам, по которым кодируется оператор переадресации, с той разницей, что вместо признака оператора переадресации 02 в характеристику включается признак оператора восстановления 06 и в третьих адресах строк бланка вместо условных чисел параметров записываются числа 0000.

З а п и с ь о п е р а т о р а **H**. Если нестандартный оператор составлен программистом в условных числах, этот оператор вместе со своей характеристикой (которая составлена самим программистом) попросту переписывается на бланки, отведенные для основного массива информации. Если программист вместо условных чисел писал в адресах команд буквы, то эти буквы с помощью словаря заменяются условными числами.

З а п и с ь т а б л и ц з а п а с е н н ы х к о н с т а н т. После того как информация об операторах закодирована и переписана на бланки, к ней приписывают таблицы запасенных констант.

Перед таблицей констант восстановления ставится характеристика

| | | | | |
|---|---|---|---|---|
| 4000+ $N$ | $l_1$ | 7721 | | $l_2$ |

где $l_1$— количество всех строк таблицы, а $l_2$ — количество индикат, входящих в таблицу ($l_2$ может занимать разряды контрольного знака и кода операции).

Перед таблицей констант переадресации ставится характеристика

| | | | | |
|---|---|---|---|---|
| $4000 + N$ | $l_1$ | 7722 | 0 | 00 |

а перед таблицей запасенных числовых констант — характеристика

| | | | | |
|---|---|---|---|---|
| $4000 + N$ | $l_1$ | 7723 | 0 | 00 |

где $l_1$— число строк в таблице.

При переписке констант на бланк нельзя изменять порядок расположения в этих таблицах запасенных констант, так как условное число, отвечающее запасенной константе, определяется местом константы в таблице.

Если нет запасенных констант какого-либо вида, то характеристика таблицы констант все равно в массив

информации включается. При зтом $l_1$, а в случае констант восстановления и $l_2$ должны быть нулями.

**2. Второй массив информации.** Составление второго массива информации начинают на новом бланке. В него входит закодированная *ТЗП* (таблица зависимости величин от параметров).

В задании для кодировщиков *ТЗП* разбита на участки, каждый из которых отвечает определенному параметру.

Эти участки последовательно кодируются и снабжаются характеристиками *по* следующим правилам.

Каждый участок *ТЗП* снабжается характеристикой вида

| | | | | |
|---|---|---|---|---|
| 0000 | *l* | 7700 + *n* | 0 | 00 |

где *l*— число строк участка (в восьмеричной системе), полученное путем подсчета числа строк соответствующего участка *ТЗП* в задании для кодировщиков; 7700 + *n* — условное число соответствующего параметра.

Вслед за характеристикой переносятся из задания для кодировщиков сами строки участка *ТЗП.* В первом и втором адресе записываются условные числа величин, приведенные в задании для кодировщиков, в третьем адресе — шаг переадресации или (в случае переменного шага) условное число отвечающей ему величины. В клетке строки бланка, отведенной для контрольного знака, пишут нуль, если шаг переадресации постоянный, и единицу, если шаг переадресации переменный. В клетке строки бланка, отведенной для кода операции, пишут 02, если переадресация состоит в прибавлении шага переадресации (в задании для кодировщиков написано «.плюс»), или 15, если шаг переадресации должен вычитаться (в задании для кодировщиков написано «минус»). Таким образом, при пёреадресации с постоянным шагом $h$ = 3 (плюс) величин, начиная с 5010 и кончая 5015, строка *ТЗП* будет выглядеть так:

5010 5015 0003 0 02.

При переадресации тех же величин с шагом $h$ = 4 (минус) строка *ТЗП* будет иметь такой вид:

5010 5015 0004 0 15.

При переадресации величин от 5020 до 5031 (с переменным шагом) на содержимое ячейки 5077 (плюс) строка будет следующей:

5020 5031 5077 1 02.

При переадресации тех же величин (с переменным шагом) на содержимое ячейки 5170 (минус) строка примет такой вид:

5020 5031 5170 1 15.

Если в задании для кодировщиков в строке *ТЗП* фигурирует только одно условное число величины, это условное число записывается на бланке дважды: в первом и во втором адресах. При этом строка может иметь такой вид:

5010 5010 0003 0 02.

После того как все участки *ТЗП* перенесены на бланки, в очередной строке второго массива информации ставится набор цифр

7777 0000 0000 0 00,

являющийся признаком конца *ТЗП.* На этом второй массив информации закончен.

**3. Третий массив информации.** Третий массив информации составляется кодировщиками только в случае стандартного распределения памяти и состоит из *ТРВ* (таблицы распределения величин). В случае нестандартного распределения памяти третий массив отсутствует.

С т а н д а р т н о е р а с п р е д е л е н и е п а м я т и . Таблица распределения величин начинается с нового бланка и составляется кодировщиком по следующим правилам.

Начинается *ТРВ* характеристикой, имеющей такой вид:

| | | | | |
|---|---|---|---|---|
| 0000 | λ | *m* | 0 | 00 |

где λ — восьмеричное число, определяемое по формуле

$$\lambda = u_{\max} - 5000,$$

причем $u_{\max}$ — наибольшее условное число величины из используемых в информации о программируемой программе; число $\lambda$ принято называть *количеством используемых величин; m* — число ячеек (в восьмеричной системе), отведенных для значений той величины, которой присвоен условный адрес $a_{\max}$. При этом $a_{max}$— наибольший условный адрес из третьего столбца словаря задачи.

Следующие сто восемьдесят три строки бланков отведены непосредственно под *ТРВ.* Клетки (адреса) упомянутых строк составляют три столбца. Клетки, входящие в первый столбец (считая сверху вниз), отвечают последовательным условным числам, начиная с 5001 и кончая 5267. Клетки второго столбца соответствуют условным числам, начиная с 5270 и кончая 5556. Клетки, составляющие третий столбец отвечают остальным условным числам величин, начиная с 5557. Последние тридцать восемь клеток третьего столбца свободны (никаким условным числам не отвечают).

В клетках, отвечающих величинам, использованным в программируемой задаче (перечисленным в словаре задачи), при расписывании *ТРВ* записывают условные адреса этих величин. Если клетки столбца при этом заполняются не все подряд, то в промежуточных клетках пишут число 0000.

При задании третьего массива информации в таком виде машиной будет произведено стандартное распределение памяти, описанное на стр. 262. *ПП-С* допускает еще один вид стандартного распределения памяти (так сказать, «не вполне стандартного»), при котором рабочие ячейки располагаются в ячейках памяти, начиная с 0010; вслед за ними размещается некоторое количество величин; затем расположены программа и таблицы констант (восстановления, переадресации, числовых) и, наконец, опять некоторый массив величин.

Для получения такого стандартного распределения памяти в первом адресе характеристики *ТРВ* необходимо поместить наименьший условный адрес величин, подлежащих размещению после таблиц констант. Все величины, которым присвоены меньшие условные адреса, чем указанный в первом адресе характеристики, будут размещены после рабочих ячеек и перед программой.

Нестандартное распределение памяти. Нестандартное распределение памяти, как правило, производит программист после изучения программы, составленной с помощью *ПП-С* в условных числах (без обработки информации блоком *П*). Составляется оно по следующим правилам.

В адресах первых семи строк бланка, отведенного для третьего массива информации, с помощью восьмеричных чисел записываются:

а) начало группы рабочих ячеек (действительный номер первойячейки этой группы);

б) число 0000;

в) начало программы (действительный номер ячейки, в которойдолжна быть записана контрольная сумма программы, командыпрограммы размещаются вслед за этой ячейкой);

г) число 0000;

д) начало группы констант восстановления;

е) начало группы констант переадресации;

ж) начало группы числовых констант.

Эти семь строк представляют собой *ТРП* (таблицу распределения памяти).

В восьмой по счету строке бланка записывается характеристика таблицы распределения величин, имеющая такой вид:

| | | | | |
|---|---|---|---|---|
| 0000 | *l* | 0000 | 0 | 00 |

где *l* — количество строк, занятых на бланках таблицей распределения величин.

После характеристики в следующих за ней (не более чем ста восьмидесяти трех) строках бланка расписывается таблица распределения величин по таким же правилам, как и при стандартном распределении памяти, с тем отличием, что вместо условных адресов величин указываются их действительные адреса.

**4. Подготовка данных для ввода информации в память машины.** После того как все три массива закодированной информации составлены, кодировщик подсчитывает (в восьмеричнойсистеме):

$L_1$ — количество строк, занятых первым массивом информации;

$L_2$— количество строк, занятых вторым массивом;

$L_3$ — количество строк, занятых третьим массивом (если третий массив отсутствует, то полагают $L_3$= 0001).

После этого на отдельном бланке составляется таблица *L,* состоящая из четырех строк.

Первые три из них имеют такой вид:

| | | | | |
|---|---|---|---|---|
| 0000 | $L_1$— 1 | 0000 | 0 | 00 |
| 0000 | $L_2$— 1 | 0000 | 0 | 00 |
| 0000 | $L_3$— 1 | 0000 | 0 | 00 |

Четвертая строка содержит признак вида распределения памяти. При стандартном распределении памяти она содержит число

0000 0000 0000 0 00,

а при нестандартном — число

7777 7777 7777 1 77.

На этом работа кодировщика закончена. Перфокарта, на которую будет перенесена таблица *L*, в дальнейшем называется перфокартой *L*.

5. **Пример кодировки информации для *ПП-С*.** Для пояснения правил кодировки информации для *ПП-С* продолжим пример, использованный нами ранее в § 50 Информация, составленная программистом (см. стр. 264-265), после кодировки будет иметь следующий вид:.

Первый массив инфорации

| Команды и числа | | | | |
|---|---|---|---|---|
| 4001 | 0001 | 7702 | 0 | 00 |
| 4003 | 4004 | 7701 | | |
| 4002 | 0017 | 7704 | 0 | 00 |
| 5001 | 0660 | 5001 | | |
| 0001 | 0000 | 0000 | | |
| 210 | 000 | 000 | 0 | 01 |
| 0630 | 5001 | 0077 | | |
| 5003 | 0000 | 0000 | | |
| 170 | 000 | 000 | 0 | 01 |
| 0001 | 0640 | 5001 | | |
| 0001 | 0730 | 5001 | | |
| 0077 | 5005 | 0000 | | |
| 325 | 000 | 000 | 0 | 01 |
| 5001 | 5001 | 0670 | | |
| 5001 | 0001 | 5001 | | |
| 0001 | 7400 | 0077 | | |
| 5004 | 0060 | 5000 | | |
| 4003 | 4004 | 0000 | | |
| 4003 | 0012 | 7701 | 0 | 00 |
| 5000 | 0077 | 5007 | | |
| 0401 | 5003 | 0001 | | |
| 150 | 000 | 000 | 0 | 01 |
| 5002 | 5002 | 0001 | | |
| 5002 | 0630 | 5004 | | |
| 0201 | 0620 | 0401 | | |
| 5002 | 0001 | 5005 | | |
| 0201 | 0077 | 5006 | | |
| 4005 | 0020 | 4005 | | |
| 0077 | 5000 | 0000 | | |
| 4004 | 0013 | 7701 | 0 | 00 |
| 0401 | 5003 | 0001 | | |
| 150 | 000 | 000 | 0 | 01 |
| 5002 | 5002 | 0201 | | |
| 0620 | 0401 | 5002 | | |
| 0001 | 5005 | 0201 | | |
| 0077 | 5006 | 5002 | | |

Второй массив инфорации

| Команды и числа | | | | |
|---|---|---|---|---|
| 0000 | 0001 | 7701 | | |
| 5002 | 5002 | 5022 | 1 | 02 |
| 0000 | 0002 | 7702 | | |
| 5001 | 5001 | 0001 | 0 | 02 |
| 5010 | 5011 | 0002 | 0 | 02 |
| 0000 | 0001 | 7703 | | |
| 5002 | 5002 | 0001 | 0 | 02 |
| 7777 | 0000 | 0000 | 0 | 00 |

| Команды и числа | | | | |
|---|---|---|---|---|
| 630 | 0401 | 0003 | | |
| 5004 | 0201 | 0620 | | |
| 0401 | 5002 | 0001 | | |
| 5005 | 0201 | 0077 | | |
| 5007 | 0000 | 0000 | | |
| 4005 | 0002 | 7701 | 0 | 00 |
| 5006 | 0077 | 5010 | | |
| 5007 | 0077 | 5011 | | |
| 4006 | 0002 | 7702 | 0 | 00 |
| 4002 | 4002 | 7702 | | |
| 4005 | 4005 | 7702 | | |
| 4007 | 0003 | 7704 | 0 | 00 |
| 5012 | 0002 | 7423 | | |
| 0077 | 5013 | 0065 | | |
| 5014 | 4010 | 4002 | | |
| 4010 | 0001 | 7711 | 0 | 00 |
| 0000 | 0000 | 5012 | 0 | 13 |
| 4011 | 0002 | 7706 | 0 | 00 |
| 4002 | 4002 | 0000 | | |
| 4005 | 4005 | 0000 | | |
| 4012 | 0015 | 7711 | 0 | 00 |
| 5010 | 0377 | 5010 | 0 | 70 |
| 0000 | 0377 | 5020 | 0 | 34 |
| 5010 | 5020 | 5020 | 0 | 17 |
| 0000 | 0000 | 0000 | 0 | 00 |
| 5020 | 0000 | 5021 | 0 | 13 |
| 4002 | 4002 | 7004 | 0 | 27 |
| 5020 | 5021 | 0000 | 0 | 16 |
| 7012 | 7011 | 0000 | 0 | 20 |
| 5020 | 5021 | 0000 | 0 | 40 |
| 5010 | 0377 | 0000 | 0 | 44 |
| 5015 | 7431 | 5022 | 0 | 02 |
| 0000 | 0200 | 0000 | 0 | 45 |
| 5022 | 0000 | 0000 | 0 | 44 |
| 4013 | 0001 | 7702 | 0 | 00 |
| 4003 | 4004 | 7703 | | |
| 4014 | 0003 | 7704 | | |
| 5015 | 0002 | 7431 | | |
| 0077 | 5016 | 0065 | | |
| 5017 | 4015 | 4002 | | |
| 4015 | 0001 | 7711 | 0 | 00 |
| 0000 | 0000 | 0000 | 0 | 40 |
| 4016 | 0000 | 7721 | 0 | 00 |
| 4017 | 0000 | 7722 | 0 | 00 |
| 4020 | 0000 | 7723 | 0 | 00 |

Третий массив информации

| Команды и числа | | | | |
|---|---|---|---|---|
| 0000 | 0022 | 0377 | 0 | 00 |
| 0003 | | | | |
| 0203 | | | | |
| 0304 | | | | |
| 0305 | | | | |
| 0306 | | | | |
| 0307 | | | | |
| 0310 | | | | |
| 0315 | | | | |
| 0316 | | | | |

| Команды и числа | | | | |
|---|---|---|---|---|
| 0311 | | | | |
| 0311 | | | | |
| 0001 | | | | |
| 0303 | | | | |
| 0303 | | | | |
| 0002 | | | | |
| 0313 | | | | |
| 0314 | | | | |
| 0312 | | | | |

Таблица *L*

| Команды и числа | | | | |
|---|---|---|---|---|
| 0000 | 0120 | 0000 | 0 | 00 |
| 0000 | 0007 | 0000 | 0 | 00 |
| 0000 | 0022 | 0000 | 0 | 00 |
| 0000 | 0000 | 0000 | 0 | 00 |

## § 52. Эксплуатаия программирующей программы *ПП-С*

Информация, перенесенная на бланки кодировщиками, пробивается на трех массивах перфокарт* (соответственно трем массивам информации), после чего можно приступить к автоматическому программированию.

Эксплуатация программирующей программы осуществляется следующим образом.

**1. Запись *ПП-С* на магнитную ленту.** Каждый блок *ПП-С*имеет свою программу ввода. Перфокарта с этой программой ввода включена в комплект перфокарт, хранящих блок, и отделена от перфокарт с программой блока одной непробитой перфокартой.

Комплекты перфокарт блоков *ПП-С,* подлежащих вводу, соединяются в общую колоду и вкладываются в читающее устройство машины.

Нажимом кнопки начального пуска вводят первый из блоков в память машины. После записи блока на ленту автоматически происходит ввод и запись следующего блока и т. д. После ввода и записи на магнитную ленту блока *П* происходит останов.

**2. Ввод информации.** Ввод информации и запись ее на соответствующие зоны магнитной ленты производят с помощью приложенной к *ПП-С* специальной программы № 1. Эта программа нанесена на трех перфокартах.

Программу № 1 вместе с комплектами перфокарт, содержащими первый, второй и третий массивы информации, складывают в общую колоду следующим образом. Вслед за тремя перфокартами программы № 1 помещают перфокарту *L,* затем располагают перфокарты со всеми тремя массивами информации (в порядке номеров массивов), отделяя каждый массив от другого непробитой перфокартой.

Ввод и запись всей информации производится так: после нажима на пульте машины кнопки начального пуска машина вводит в память информацию, суммирует ее и останавливается.

Теперь необходимо, не «стирая» записанного в памяти машины, повторить ввод (прежним способом). Машина снова введет материал в память, просуммирует его, сравнит новую контрольную сумму с ранее полученной и в случае совпадения контрольных сумм автоматически запишет информацию на магнитную ленту.

**3. Пуск *ПП-С*.** К *ПП-С* приложены небольшие программы, каждая из которых занимает всего одну перфокарту, предназначенные для вызова с магнитной ленты каждого блока *ПП-С*. Эти программы называются *картами вызова* блоков. Имеется также три *карты вызова информации* (кроме третьего массива, который вызывает сам блок *П*) соответственно трем зонам магнитной ленты, на которых может находиться информация.

Для того чтобы начала работать *ПП-С,* вводят нажимом кнопки *«Начальный пуск»* карту вызова блока *К*. Происходит вызов с магнитной ленты этого блока и останов машины в ячейке № 0016. Для того чтобы *ПП-С* приступила к автоматическому программированию, остается нажать на пульте управления кнопку *«Пуск»*.

Если по какой-либо причине перед пуском *ПП-С* в памяти машины информация отсутствует, с помощью карты № 1 вызова информации ее вызывают с магнитной ленты. После вызова информации происходит останов в ячейке № 0016. Пуск *ПП-С* по-прежнему осуществляют нажимом кнопки *«Пуск»*.

**4. Результаты, выдаваемые *ПП-С*.** После работы блока *О ПП-С* выдает на перфокарты программу,

* При нестандартном распределении памяти — на двух массивах перфокарт

составленную в условных числах, и другую информацию в следующем виде.

1. Первый массив перфокарт:

а) количество команд программы (без констант);
б) количество констант восстановления;
в) количество констант переадресации;
г) количество числовых констант;
д) количество рабочих ячеек;
е) таблица характеристик (*ТХ*).

Таблица характеристик (*ТХ*) представляет собой последовательность характеристик операторов составленной программы. Каждая характеристика имеет вид, несколько отличный от того, в котором характеристики записывались при кодировке первого массива информации. Характеристики представлены следующим образом:

$$\underbrace{4000+N}_{\text{I адрес}}\ \underbrace{l_1}_{\text{II адрес}}\ \underbrace{l_2}_{\text{III адрес}}\ 0\ \underbrace{k}_{\text{Код операции}}$$

Здесь $N$ — номер оператора; $l_1$— количество команд, входящих в этот оператор (в случае нестандартного оператора или таблицы констант восстановления — общее количество команд или констант и индикат); $l_2$ отлично от нуля только в характеристиках нестандартных операторов, стандартных (библиотечных) подпрограмм и таблицы констант восстановления и равно числу собственно команд или собственно констант.

2. Второй массив перфокарт содержит программу, составленную в условных числах, и таблицы используемых ею констант. Перед каждым оператором и каждой таблицей констант стоит характеристика того же вида, что и в *ТХ*.

3. После работы блока *П ППС* выдает шесть массивов перфокарт:

первый массив — числовые константы;
второй массив — константы переадресации;
третий массив — константы восстановления;
четвертый массив — готовая программа,
пятый массив — *ТХ* (таблица характеристик),
шестой массив — *ТРВ* (таблица распределения памяти для величин).

Строками *ТХ* являются переработанные характеристики операторов, имеющие такой вид:

| | | | | |
|---|---|---|---|---|
| $4000+N$ | $l$ | $a$ | 0 | $k$ |

где $N$ — номер оператора; / — количество команд, входящих в оператор; $a$ — действительный адрес первой команды оператора; $k$ — типовой признак оператора. В *ТХ* входят также характеристики таблиц запасенных констант.

Таблица *ТРВ* представляет собой столбец (два или даже три столбца, в зависимости от количества материала) действительных номеров ячеек, расположенных в порядке возрастания условных чисел величин. *ТРВ,* выдаваемая блоком *П,* имеет тот же вид, который имеет начальная *ТРВ,* входящая в состав закодированной информации для *ППС*. Начинается *ТРВ* с характеристики.

**5. Работа с *ППС* в случае нестандартного распределения памяти.** Как правило при нестандартном распределении памяти третий массив информации составляется лишь после анализа программы, выданной блоком *О* в условных числах. Этот массив записывают на магнитную ленту следующим образом.

Специальную программу ввода № 2, приложенную к *ППС,* объединяют в один комплект перфокарт с третьим массивом. Нажимом кнопки *«Начальный пуск»* этот массив вводят в память. При этом происходит останов (вследствие несовпадения контрольных сумм, так как полученная сумма вводимого материала сравнивается с нулем, полученным благодаря предварительной очистке памяти). Вторично вкладывают комплект перфокарт в читающее устройство. Нажимом кнопки *«Начальный пуск»* производят вторичный ввод третьего массива. Машина сравнивает контрольную сумму, полученную при вторичном вводе, с контрольной суммой, получившейся при первом вводе. Если сумма совпадает (ввод правилен), происходит автоматическая запись третьего массива на ленту. Затем с помощью соответствующих карт вызова вызывают информацию (из зоны № 4014 магнитной ленты) и блок *П.* Нажимом кнопки *«Пуск»* приводят в действие *ППС*.

После работы блока *П* машина выдает на перфокарты материал в том порядке, который описан выше.

### § 53. Краткое описание работы блоков *ППС*

**1. Блок *К*.** Первоначально *ППС* не имела блока контроля. В процессе эксплуатации *ППС* выяснилось, что при кодировке информации часто допускаются некоторые однотипные ошибки элементарного характера. Эти ошибки приводили к тому, что *ППС* отказывала в работе. Поиск ошибок в информации был связан с большой затратой времени. Эти обстоятельства навели на мысль составить контролирующий блок *К,* выявляющий часто встречающиеся ошибки, и заодно возложить на него работу по составлению *ВТ* (ведущей таблицы), которую прежде составляли вручную при кодировке информации.

Блок *К* выявляет следующие ошибки:

а) в характеристике оператора неправильно указано число строкследующей за ней информации;

б) в характеристике участка *ТЗП* неверно указана длина участка;

в) в конце *ТЗП* отсутствует признак ее конца (то-есть отсутствует число 7777 0000 0000 0 00);

г) в операторе переадресации первого массива имеется условное число параметра, для которого нет участка в *ТЗП;*

д) в арифметическом или логическом операторе первого массива число открытых скобок не равно числу закрытых скобок того же типа (напоминаем, что используются скобки двух видов: математические круглые и логические квадратные).

При обнаружении перечисленных ошибок, блок *К* выдает на перфокарты данные о том, в каких местах информации и *какого* типа ошибки содержатся. Такими данными являются:

а) характеристики операторов, в которых неправильно указано число строк;

б) характеристики участков *ТЗП,* в которых неправильно указаны «длины» участков;

в) число 7777 0000 0000 0 00 в случае отсутствия его в конце *ТЗП;*

г) номер параметра, для которого отсутствует участок *ТЗП;*

д) код вида

| | | | | |
|---|---|---|---|---|
| 4000 + N | 7000 + n | α | 0 | 00 |

при неравенстве количества открытых и закрытых скобок одинакового вида; в этом коде $N$ — номер оператора, $n$ — номер логической формулы (если оператор нелогический, то $n = 0$), α — признак, характеризующий вид скобок; для арифметических скобок α = 0400, для логических скобок α = 1200.

После выдачи на перфокарты сведений об обнаруженных ошибках работа *ПП-С* прекращается.

Если в информации ошибок не оказалось, блок *К* составляет *ВТ* (ведущую таблицу) и заполняет ряд стандартных ячеек данными, нужными для работы других блоков *ПП-С.* Такими данными являются номер первой свободной ячейки (после первого массива информации), а также последние условные числа, использованные в информации для обозначения величин, констант (восстановления, переадресации и числовых) и рабочих ячеек.

Затем на основании *ВТ* блок *К.* вызывает очередной блок программирующей программы.

**2. Блок *А′*.** Блок *А′* подготавливает исходные данные об арифметических и логических операторах, входящие в состав первого массива информации, к виду, удобному для работы блока Л. Пробегая по характеристикам, блок *А′* отыскивает арифметические и логические операторы и выполняет следующие действия:

1) числовые константы, стоящие в закодированных математических формулах, переносит в таблицу запасенных числовых констант;

2) присваивает перенесенным числовым константам соответствующие условные числа и ставит эти условные числа на места, ранее занятые константами; освобождающиеся при этом адреса ячеек заполняет числами 0000 (числовая константа занимает целую ячейку, а условное число — только один адрес ячейки);

3) изменяет кодировку некоторых математических действий. В соответствии с принятыми правилами записи формул пишут:

$$1 : b\,,$$
$$1 : \sqrt{b}\,,$$
$$a : \sqrt{b}\,.$$

В таком же виде эти формулы кодируются при составлении информации для *ПП-С.* Система операций и команд машины *Стрела* требует представления вышеприведенных операций в виде

$$\left(\frac{1}{b}\right)$$

(вычисление обратной величины),

$$\frac{1}{\sqrt{b}}$$

(вычисление величины, обратной корню),

$$a \cdot \frac{1}{\sqrt{b}}$$

Блок *А′* производит изменение кодировки операций в соответствии с таблицей 50. Появляющиеся при этом числа 0000 не являются для блока *А* условными числами и на его работу не влияют;

4) после того как обработаны все арифметические и логические операторы, блок *А′* производит перевод из десятичной в двоичную систему счисления всех числовых констант, сведенных в таблицу числовых констант.

Таблица 50 Изменение кодировки блоком А’

| №пп | Первоначальная кодировка | | | Кодировка после работы блока | | |
|---|---|---|---|---|---|---|
| 1 | 7400 | 0620 | | 0620 | 0000 | |
| 2 | 7400 | 0620 | 0630 | 0632 | 0000 | 0000 |
| 3 | а | 0620 | 0630 | а | 0632 | 0000 |

**3. Блок *P′*.** Блок $P'$ подготавливает информацию о логических операторах, содержащуюся в первом массиве информации, к обработке ее блоками $A$ и $P$.

Информация о логическом операторе состоит из логических формул, после каждой из которых стоит знак операторной связи. В свою очередь логические формулы состоят из условий, соединенных знаками логических связей. Каждое простое условие представляет собой пару комплексов, соединенных знаком условия; сложное условие состоит из трех комплексов, между которыми стоят знаки условий (являющиеся согласованными знаками неравенств).

Блок $P'$ преобразует каждое условие. При этом используются две закрепленные за ним и блоком $P$ рабочие ячейки с условными числами $\rho_1$= 6001 и $\rho_2$ = 6002, не используемые другими блоками.

Условимся комплексы, последними элементами которых являются математические выражения, обозначать буквами

$$f,\ \varphi,\ \psi,\ F,$$

а комплексы, последними элементами которых являются математические формулы, — символами

$$f = R_1,\quad \varphi = R_2,\quad \psi = R_3,\quad F=R.$$

Буквы $R_1, R_2, R_3, R$ обозначают правые части формул, являющихся последними в комплексах.

Комплекс вида $f$, сводящийся к одному числу или одной букве, будем обозначать буквой $a$. В тех случаях, когда речь идет о комплексах без учета характера их последних элементов, т. е. о «комплексах вообще», будем обозначать их символами $K_1, K_2, K_3$, К.

Пусть, кроме того, σ означает любой из знаков условий « < », « ≤ », « > », « ≥ », точно так же $s$ — любой из знаков условий « ≡», «≠». Наконец, знаком «*» будем обозначать любой из знаков условий σ, $s$.

Будем говорить, что условие имеет *нормальную форму* или является *нормальным,* если оно имеет вид

$$F=R*a,$$

где $a$ в частном случае может быть и нулем.

Блок $P'$ приводит условия, входящие в логические формулы, к нормальному виду. Поясним, как это делается. Вначале рассмотрим простое условие $K_1 * K_2$.

Если $K_1$ имеет вид $f$, то блок $P'$ приписывает к нему символы $=\rho_1$ и превращает его в $f = R_1$ (где $R_1=\rho_1$). Если $K_1$ с самого начала имеет вид $f = R_1$, то $K_1$ пока никаким изменениям не подвергается.

Затем блок $P'$ анализирует комплекс $K_2$. Если $K_2$ сводится к одному числу, т. е. имеет вид $a$, то условие приведено уже к нормальному виду. Если $K_2$ имеет вид φ, но не сводится к одному числу, то блок $P'$ приписывает к этому комплексу символы $=\rho_2$, приводя его тем самым к виду $\varphi = R_2$ (где $R_2 = \rho_2$). Если оказывается, что $K_2$ сразу имеет вид $\varphi = R_2$, то этот комплекс пока никаким преобразованиям не подвергается. Таким образом, условие

$$K_1*K_2$$

оказывается приведенным к виду

$$f=R_1 * \varphi=R_2.$$

Дальнейшие преобразования условия зависят от того, чем является знак *: знаком σ или знаком $s$.

Пусть знак * означает σ .Условие

$$f = R_1\ \sigma\ \varphi = R_2$$

переписывается в виде

$$f=R_1,\quad \varphi = R_2,\quad R_1 — R_2 = \rho_1\sigma\ 0.$$

В последнем условии слева стоит комплекс вида $F = R$ (где $R = = \rho_1$), следовательно, оно имеет нормальную форму

$$F=R\ \sigma\ 0.$$

Теперь пусть знак * означает $s$. Условие

$$f=R_1\ s\ \varphi = R_2$$

переписывается блоком в следующем виде:

$$\varphi =R_2,\quad f = R_1 s R_2.$$

Левую часть последнего условия составляет комплекс вида $F = R_1$, так что оно имеет нормальную форму:

$$F = R_1\ s\ R_2.$$

Рассмотрим теперь сложное условие

$$K_1\sigma_1K_2\sigma_2K_3$$

где $\sigma_1$ и $\sigma_2$ — согласованные знаки неравенств. Блок *P′,* прежде всего, анализирует комплекс $K_2$. Если этот комплекс сводится к одному числу, то условие имеет такой вид:

$$K_1\,\sigma_1\,a\,\sigma_2\,K.$$

Если $K_2$ не сводится к одному числу и имеет вид $\varphi$, то, приписывая к нему символы $= \rho_2$, блок преобразует $K_2$ в $\varphi = R_2$ (где $R_2 = \rho_2$). Если $K_2$ сразу имеет вид $\varphi = R_2$, то этот комплекс пока никаким изменениям не подвергается. Аналогично $K_1$ приводится к виду $f = R_1$, и $K_3$ приводится к виду $\psi = R_3$. Условие принимает форму

$$f=R_1\,\sigma_1\,\varphi=R_2\,\sigma_2\,\psi=R_3,$$

затем ему придается форма

$$\varphi = R_2,\quad f=R_1\,\sigma_1\,R_2\,\sigma_2\,\psi=R_3.$$

Условие и в этом случае приведено к виду, в котором второй его комплекс состоит из одного числа. Теперь сложное условие заменяется конъюнкцией двух простых условий:

$$\varphi = R_2,\quad f = R_1\sigma R_2/\backslash\psi=R_3\,\tilde{\sigma}_2R_2$$

Связь знака $\tilde{\sigma}_2$ со знаком $\sigma_2$ показана в таблице 51.

Т а б л и ц а 51. Связь знака $\tilde{\sigma}_2$ со знаком $\sigma_2$

| $\sigma_2$ | < | ≤ | > | ≥ |
|---|---|---|---|---|
| $\tilde{\sigma}_2$ | > | ≥ | < | ≤ |

Каждое из простых условий, входящих в полученную конъюнкцию, имеет нормальную форму.

Приведение условий к нормальному виду блок *P′* осуществляет последовательно, просматривая логическую формулу слева направо. После приведения каждого условия к нормальному виду его левому комплексу предпосылается характеристика с типовым признаком 01 и номером оператора $N_{max}+ i$, где $N_{max}$ — наибольший номер оператора, фигурирующий в первом массиве до начала обработки блоком *P′* данного логического оператора, а *i*— номер просмотренного условия, отсчитанный от начала логического оператора.

Таким образом, левый комплекс условия превращается в арифметический оператор первого массива информации. В конце этого оператора блок *P′* ставит две математические формулы вида

5000 0077 5000,

соответствующие нулевым командам :

0000 0000 0000 0 13.

Этим обеспечивается оставление блоком *A* в конце составленного им арифметического оператора двух свободных ячеек, нужных для работы блока *P*. Попутно с преобразованием левых комплексов условий в арифметические операторы блок *P′* переносит просматриваемую им логическую формулу в специально отведенное место оперативной памяти (как говорят, на *операционный стол),* заменяя в ней левые комплексы условий условными числами номеров операторов, в которые превращены эти левые комплексы. Например, если логическая формула имела вид

$$x^2+y^2 = R_1>2\ /\backslash\ x + 2z =R_2<3\ \backslash/\ x^2 + 2y^2 = R_3 \neq 1$$

и в конце нее стоял знак операторной связи 4012, 4015 и если $N_{max} = 4050$, то логическая формула переносится на операционный стол в виде

4051 >2 /\ 4052 <3 \/ 4053 ≠ 1 4012 4015

Таким образом обрабатывается последовательно вся информация о логическом операторе — формула за формулой. При этом в первом массиве постепенно появляются новые арифметические операторы, а на операционном столе постепенно воспроизводятся все логические формулы и знаки операторной связи, входящие в состав логического оператора.

По окончании обработки информации о логическом операторе перед материалом, скопившимся на операционном столе, ставится характеристика с типовым признаком 04; этот материал переносится в первый массив информации и размещается там непосредственно вслед за последним из арифметических операторов, полученным из логического оператора массива. Этот материал представляет собой информацию для блока *P*.

После того как все логические операторы массива переработаны вышеописанным образом, блок *P*' свою работу заканчивает и вызывает блок *A*.

**4. Блок *A*.** Блок *A* (арифметический) участвует в составлении логических операторов и составляет арифметические операторы. Он перерабатывает закодированные математические формулы в группы команд. Одновременно с составлением команд блок *A* производит их *экономию,* т. е. исключает команды повторного выполнения операций, уже запрограммированных в составляемом операторе. Отыскав в первом массиве характеристику с типовым признаком 01, блок *A* просматривает следующую за ней информацию. Найдя первую

операцию, которая может быть выполнена, блок формирует отвечающую ей команду и, если эта команда не является первой в составляемом операторе, сравнивает ее с ранее составленными командами. При наличии такой же команды новая команда уничтожается (*экономится*), при отсутствии — новая команда присоединяется к ранее составленным.

В качестве первого и второго адресов команды блок *А* берет условные числа оперируемых величин, стоящие в закодированной арифметической формуле, которую он обрабатывает, в качестве третьего адреса — записывает очередное условное число рабочей ячейки. В качестве кода операции используется условное число знака действия. При умножении такое условное число в формуле отсутствует; в этом случае блок *А* ставит код операций 05. Условными числами оперируемых величин для блока *А* являются условные числа величин, констант, рабочих ячеек и номеров операторов.

При составлении в каждой команде ставится новое условное число рабочей ячейки. Впоследствии экономию рабочих ячеек осуществляет блок *Э*.

После включения новой команды в составляемый оператор блок *А* стирает в обрабатываемой информации использованные условные числа оперируемых величин и знака действия и заменяет их условным числом рабочей ячейки, взятым из третьего адреса новой команды. При уничтожении новой команды (если она сэкономилась) вместо использованных для ее составления условных чисел оперируемых величин и знака действия записывается условное число рабочей ячейки, взятое из третьего адреса той команды, с которой совпала новая команда.

В третий адрес последней команды, получающейся при обработке арифметической формулы, вместо условного числа рабочей ячейки ставится условное число результата, стоящее в правой части формулы. Такая команда экономии не подлежит.

Для более полной экономии в некоторых случаях составляется не одна, а сразу несколько новых команд и каждая из них сравнивается со всеми ранее составленными командами. Приведем эти случаи.

1. Если в математической формуле фигурирует сумма или произведение двух чисел, которым отвечают условные числа $u_1$ и $u_2$, то составляются сразу две команды:

$$u_1 \quad u_2 \quad r \quad \theta,$$
$$u_2 \quad u_1 \quad r \quad \theta,$$

где $r$ — условное число рабочей ячейки, $\theta$ при сложении равно 01, а при умножении 05.

2. Если в математической формуле встречается сумма или произведение трех чисел, которым отвечают условные числа $u_1$, $u_2$, $u_3$, то составляется шесть новых команд:

$$u_1 \quad u_2 \quad r \quad \theta,$$
$$u_2 \quad u_1 \quad r \quad \theta,$$
$$u_1 \quad u_3 \quad r \quad \theta,$$
$$u_3 \quad u_1 \quad r \quad \theta,$$
$$u_2 \quad u_3 \quad r \quad \theta,$$
$$u_3 \quad u_2 \quad r \quad \theta$$

(*r* и $\theta$ имеют прежний смысл).

3. Если в математической формуле имеется выражение

$$a + b - c,$$

причем числам *a, b, c* соответствуют условные числа $u_1$, $u_2$, $u_3$, то составляются четыре команды:

$$u_1 \quad u_2 \quad r \quad 01,$$
$$u_2 \quad u_1 \quad r \quad 01,$$
$$u_1 \quad u_3 \quad r \quad 03,$$
$$u_2 \quad u_3 \quad r \quad 03.$$

Если ни одна из нескольких команд не может быть сэкономлена, то в составляемый оператор включается первая из них. Составление каждой новой команды начинается с просмотра всей информации с начала. При этом последовательно составляются все команды, отвечающие первой арифметической формуле, затем — второй арифметической формуле, и т. д. После составления одного арифметического оператора блок *А* находит следующую характеристику с типовым признаком 01, и т. д. Окончив составление всех арифметических операторов, блок *А* вызывает на основании *ВТ* следующий блок *ПП-С*.

**5. Блок *Р*.** Блок *Р* приводит логические операторы, составление которых начато блоком *А*, к окончательному виду. Для этого используются информация, подготовленная блоком *Р′*, и части логических операторов, составленные блоком *А* в виде арифметических операторов.

Как уже было сказано, информация, составленная блоком *Р′*, представляет собой группу логических формул, состоящих из условий, приведенных к нормальному виду, причем вместо левых комплексов условий в этих формулах записаны условные числа номеров операторов. В конце каждой формулы стоит знак операторной связи. Для краткости будем называть номера операторов, представленные в левых частях условий, номерами условий. О знаке операторной связи, следующем за условием, будем говорить, что он принадлежит этому условию. Числа, из которых состоит знак операторной связи (условные числа номеров логических формулили номеров операторов), будем называть *отсылками*. Первая отсылка отвечает значению 0 истинности условия, а вторая — значению 1.

Напомним, что операторы (части логического оператора), составленные блоком *А*, оканчиваются двумя

нулевыми командами вида

5000 5000 5000 0 13.

Эти нулевые команды предусмотрены для того, чтобы на их место блок *P* мог поставить команду проверки условия (если она требуется) и команду условного перехода.

Работа блока *P* состоит в том, что, анализируя логические формулы, он определяет знаки операторной связи, принадлежащие каждому условию, затем просматривает последнюю ненулевую команду оператора, отвечающего этому условию, и в зависимости от кода операции этой команды и знака операторной связи добавляет к оператору команду проверки условия (если она нужна) и команду условного перехода. После того как вся информация о логическом операторе проанализирована и все команды условного перехода проставлены, блок *P* объединяет все части логического оператора в один целый оператор, уничтожая характеристики, стоящие перед этими частями. При этом команды условного перехода, которые были составлены в виде

$4000 + N_1$ $4000 + N_2$ 0000 0 20

(где $N_1$ и $N_2$ — номера объединяемых операторов), преобразуются к тому виду, который они имеют в нестандартных операторах, а именно, к виду

$7000 + n_1$ $7000 + n$ 0000 0 20

(где $n_1$ и $n_2$ — номера соответствующих команд логического оператора, отсчитанные от его начала).

После того как все логические операторы приведены блоком *P* к окончательному виду, производится вызов (на основании *ВТ)* очередного блока.

Приведем описание алгорифма, с помощью которого блок *P* определяет знаки операторной связи, принадлежащие условиям. Этот алгорифм основывается на следующих формулах (см. § 47, *Исключение знаков логических связей):*

$$(\alpha \wedge \beta)(Q_i)(Q_j) = \alpha\,\beta\,(Q_i)(Q_j),$$

$$(\alpha \vee \beta)(Q_i)(Q_j) = \alpha\,\beta\,(Q_i)(Q_j).$$

В этих формулах α и β следует рассматривать как простые или сложные условия, а стрелки — как отсылки. Нижняя стрелка изображает собой первую, а верхняя стрелка — вторую отсылку.

В логических формулах, которые блок *P* анализирует, сложные условия заключены обычно в квадратные (логические) скобки. Исключения составляют многозначные конъюнкции, представляющие собой сложные условия, но не заключенные в скобки, так как знак $\wedge$ «старше» знака $\vee$.

Из приведенных формул вытекают следующие правила:

1) последнее (простое или сложное) условие, входящее в состав сложного условия, имеет тот же знак операторной связи, что и все сложное условие;

2) при определении знаков операторной связи простых условий, входящих в сложное, порядок этих условий не меняется;

3) одна из отсылок условия α ведет к условию β, эта отсылка называется *стандартной;* другая отсылка, принадлежащая α, является нестандартной и зависит от отсылок условия β;

4) если α и β связаны знаком $\wedge$, то нестандартной отсылкой α является первая отсылка; она совпадает с первой отсылкой, принадлежащей β, т. е. при переходе от β к α первая отсылка сохраняется;

5) если α и β соединены знаком $\vee$, то нестандартной отсылкой α является вторая отсылка; она совпадает со второй отсылкой β, т. е. при переходе от β к α вторая отсылка сохраняется;

6) если β является сложным условием, т. е. имеет вид

$$\beta_1 \wedge \beta_2$$

или

$$\beta_1 \vee \beta_2$$

то стандартная отсылка α ведет к $\beta_1$, нестандартная отсылка α определяется по отсылкам $\beta_2$ или, что то же самое, по отсылкам сложного условия β.

Эти правила позволяют определение знаков операторной связи условий, входящих в логическую формулу, анализируемую блоком *P,* производить с помощью следующего алгорифма:

а) последнему условию логической формулы присваивается знак операторной связи, стоящий в конце формулы;

б) логическая формула просматривается блоком *P* справа налево (т. е. с конца);

в) если некоторому условию, имеющему номер $N_i$ присвоен знак операторной связи (*M, N*) и если между условием $N_i$ и ближайшим к нему слева условием $N_{i-1}$ стоит знак $\wedge$, то условию $N_{i-1}$ присваивается знак операторной связи $(N_i, N)$;

г) если между условиями $N_i$ и $N_{i-1}$ стоит знак $\vee$, то условию $N_{i-1}$ присваивается знак операторной связи $(M, N_i)$;

д) если при просмотре логической формулы (справа налево) встречается закрытая скобка, то отвечающий ей знак

операторнойсвязи (т. е. знак операторной связи ближайшего от нее слева условия) запоминается до тех пор, пока не встретится отвечающая ей открытая скобка; при определении нестандартной отсылки условия, стоящего левее открытой скобки, может потребоваться знак операторной связи, принадлежащий закрытой скобке;

е) если при просмотре логической формулы встречается конъюнкция, то знак операторной связи последнего условия этой конъюнкции запоминается до тех пор, пока не «окончится» эта конъюнкция; этот знак операторной связи может понадобиться для определения нестандартной отсылки условия, стоящего в формуле левее упомянутой конъюнкции.

Описанный алгорифм определения знаков операторной связи отличен от алгорифмов, применяемых в других операторных программирующих программах, и имеет то достоинство, что не требует многократных просмотров логической формулы в различных направлениях. Однако это достоинство связано с недостатком (правда, несущественным), состоящим в необходимости запоминать знаки операторных связей, принадлежащие скобкам или конъюнкциям.

После окончания анализа последней логической формулы, входящей в информацию о логическом операторе, блок *P* переходит к анализу предпоследней и т. д., т. е. логические формулы анализируются, начиная с последней.

По мере присвоения условиям знаков операторной связи в первом массиве отыскиваются соответствующие арифметические операторы и в их конце (на месте нулевых команд) записываются команда проверки условия и команда условного перехода. В некоторых случаях, в зависимости от знака условия и последней ненулевой команды арифметического оператора, команда проверки условия может экономиться.

В таблице 52 показаны команды, проставляемые блоком *P* в конце операторов. Символы *m* и *n* означают соответственно первую и вторую отсылки. Символом *a* обозначен правый комплекс нормального условия, отвечающего оператору; в командах буква *a* обозначает также условное число, отвечающее этому комплексу. Особый случай, не вошедший в эту таблицу, представляет собой условие, логическое содержание которого сводится к неравенству

$$|xy| \sigma 1.$$

Последние ненулевые команды оператора, проверяющего это условие, составляются блоком *A* в виде

*x* *y* *r* 05,
*r* 7435 $\rho_1$ 11,

Т а б л и ц а 52. Команды, формируемые блоком *P*

| Значение *a* | Код операции последней команды оператора | Команды, формируемые в случае знака условия | |
|---|---|---|---|
| | | ≡ | ≠ |
| *a* ≠0 | Любой код | $\rho_l$ *a* 0 16<br>*n m* 0 20 | $\rho_l$ *a* 0 16<br>*m n* 0 20 |
| *a*=0 | Любой код, кроме 11, 12, 13, 14, 16 | $\rho_l$ *a* 0 16<br>*n m* 0 20 | $\rho_l$ *a* 0 16<br>*m n* 0 20 |
| *a*=0 | 11, 12, 13,14 | *m n* 0 20 | *n m* 0 20 |
| *a*=0 | 16 | *n m* 0 20 | *m n* 0 20 |
| | Знак условия относится к группе $\sigma$ | | |
| Значение *a* | Код операции последней команды оператора | Команды, формируемые в случае знака условия | |
| | | < | ≤ |
| *a* ≠0 | Любой код | $\rho_l$ *a* 0 03<br>*m n* 0 20 | *a* $\rho_l$ 0 03<br>*n m* 0 20 |
| *a*=0 | Любой код, кроме 01, 03, 04 | $\rho_l$ *a* 0 03<br>*m n* 0 20 | *a* $\rho_l$ 0 03<br>*n m* 0 20 |
| *a*=0 | 01, 03, 04 | *m n* 0 20 | 0 $\rho_l$ 0 03<br>*n m* 0 20 |

| Значение *a* | Код операции последней команды оператора | Команды, формируемые в случае знака условия | |
|---|---|---|---|
| | | > | ≥ |
| *a* ≠0 | Любой код | *a* $\rho_l$ 0 03<br>*m n* 0 20 | $\rho_l$ *a* 0 03<br>*n m* 0 20 |
| *a*=0 | Любой код, кроме 01, 03, 04 | *a* $\rho_l$ 0 03<br>*m n* 0 20 | $\rho_l$ *a* 0 03<br>*n m* 0 20 |
| *a*=0 | 01, 03, 04 | 0 $\rho_l$ 0 03<br>*m n* 0 20 | *n m* 0 20 |

где *x, y, r* — соответствующие условные числа. При σ , означающем знак «<», блок *P* на место команды

*r* 7435 $\rho_1$ 11

поставит команду

*n* *m* 0 20.

При, σ означающем знак «>», на то же место будет поставлена команда

*m* *n* 0 20.

При σ, означающем один из знаков « ≤», «>», блок *P* действует так же, как в случае $a \neq 0$, приведенном в таблице 52.

Как мы видим, блок *P* не требует, чтобы условия, входящие в информацию о логическом операторе, были сформулированы с соблюдением правила, гласящего, что значение сигнала ω, вырабатываемого при проверке условия, должно быть значением истинности условия. Блок *P* сам обеспечивает выполнение этого правила.

После того как расставлены все команды условного перехода, производится объединение всех операторов, являющихся частями логического оператора, в один логический оператор. Для этого уничтожаются характеристики, составленные блоком *P*, а команды условного перехода приводятся к виду, употребляемому в нестандартных операторах.

После составления всех логических операторов, в соответствии с *BT* (ведущей таблицей) производится вызов очередного блока *ПП-C*.

**6. Блок *C*.** Блок *C* является, по существу, специализированной объединяющей программой, включающей библиотечные подпрограммы (операторы типа **C**) в -составляемую программу.

Информация для блока *C* (оператор первого массива) состоит из двух частей: переменной и постоянной. Переменная часть информации была описана выше; по отношению к включаемой библиотечной подпрограмме она является таблицей соответствия адресов (*TCA*) библиотечной подпрограммы условным числам составляемой программы. Постоянная часть информации состоит в свою очередь из двух частей:

а) таблицы распределения адресов (*TPA*) библиотечной подпрограммы;

б) библиотечной подпрограммы (выше мы называли ее стандартной подпрограммой).

Стандартные библиотечные подпрограммы должны быть составлены в действительных адресах с соблюдением следующих условий:

а) команды, константы (восстановления, переадресации и числовые) и рабочие ячейки должны представлять собой единые массивы;

б) эти массивы должны быть размещены непосредственно друг за другом в том порядке, в котором они перечислены в предыдущем пункте.

Постоянная часть информации начинается строкой, имеющей такой вид:

| | | | | |
|---|---|---|---|---|
| $l_1$ | $l_2$ | | 0 | 00 |

где $l_1$ — «длина» *TPA,* а $l_2$ — «длина» подпрограммы вместе с константами.

Затем следует *TPA,* состоящая из строк, имеющих такой вид:

| | | | | |
|---|---|---|---|---|
| *a* | ω | 0000 | 0 | *k* |

где α — адрес начала массива, ω — адрес конца массива, *k* — признак вида массива (см. таблицу 53).

Т а б л и ц а 53. Признаки вида массива

| №пп | Наименование массива | Признак вида массива |
|---|---|---|
| 1 | Внутренние адреса стандартной подпрограммы (номера команд)… | 40 |
| 2 | Константы восстановления……. | 21 |
| 3 | Константы переадресации…… | 22 |
| 4 | Числовые константы…… | 23 |
| 5 | Рабочие ячейки……. | 24 |

Если какой-либо из перечисленных выше массивов адресов в стандартной подпрограмме отсутствует, то соответствующая строка в *TPA* не включается.

Вслед за *TPA* расположена стандартная подпрограмма со своими константами.

Работа блока *C* аналогична работе описанной выше объединяющей программы (см. § 38) и заключается в том, что каждый оператор типа **C** превращается им в нестандартный оператор.

После того как все операторы $\boldsymbol{C}_i$, первого массива переработаны, блок *C* с помощью *BT* вызывает очередной блок *ПП-C*.

**7. Блок *F*.** К моменту работы блока *F* (переадресации) команды операторов, подлежащих переадресации, должны уже быть составлены в условных числах. Переадресации могут подвергаться логические операторы, арифметические операторы, стандартные подпрограммы и нестандартные операторы. Отсюда вытекает, что блок *F* должен работать после блоков *P', A, P, C*.

Информацией для работы блока *F* являются:

1) уже составленные в условных числах команды операторов, подлежащих переадресации (в первом массиве информации);

2) информация о каждом операторе переадресации (в первом массиве информации);

3) таблица зависимости величин от параметров *ТЗП* (во втором массиве информации).

Работа блока *F* состоит в следующем. Отыскивается информация о первом (по порядку логической схемы) операторе переадресации. Каждая строка этой информации имеет такой вид:

| | | | | |
|---|---|---|---|---|
| $4000 + N_1$ | $4000 + N_2$ | $i$ | 0 | 00 |

где $N_1$ и $N_2$ — номера операторов, $i$ — наименование параметра. Эта строка означает, что все операторы, номера $N$ которых удовлетворяют неравенству

$$N_1 \le N \le N_2,$$

подлежат переадресации по параметру $i$.

Далее в *ТЗП* отыскивается участок зависимостей величин от параметра $i$. Каждая строка этого участка имеет вид

| | | | | |
|---|---|---|---|---|
| $a_\alpha$ | $a_\omega$ | $h$ | 0 | $d$ |

или

| | | | | |
|---|---|---|---|---|
| $a_\alpha$ | $a_\omega$ | $b$ | 1 | $d$ |

Эти строки означают, что любая величина $a$, условное число которой (обозначенное той же буквой) имеет вид $5000 + n$ и удовлетворяет неравенству

$$a_\alpha \le a \le a_\omega,$$

зависит от параметра $i$. Характер зависимости указан в разряде контрольного знака. Если контрольный знак равен нулю, то шаг переадресации параметра является постоянным и его абсолютная величина равна $h$. Если контрольный знак равен единице, то шаг переадресации является переменным и его абсолютная величина будет находиться в ячейке, обозначенной условным числом $b$. Если абсолютная величина шага переадресации должна прибавляться к адресу величины, то $d = 02$; если же она должна вычитаться, то $d = 15$.

Далее, блок *F* поочередно находит в первом массиве информации операторы, подлежащие переадресации, и составляет необходимые константы переадресации и команды переадресации. Осуществляется это так. Каждая команда оператора, подлежащего переадресации, анализируется: делается проверка, удовлетворяют ли ее адреса неравенству вышеуказанного вида. Выполнение такого неравенства хотя бы для одного адреса анализируемой команды говорит о необходимости составления команды переадресации. При этом различаются два случая: переадресации с постоянным шагом и переадресации с переменным шагом.

В случае необходимости переадресации с постоянным шагом блок *F* составляет константу переадресации, используя для этогочисла $h$, стоящие в третьих адресах .строк *ТЗП*. Константа переадресации может иметь такой вид:

| | | | | |
|---|---|---|---|---|
| $h_1$ | $h_2$ | $h_3$ | 0 | 00 |

После составления константы производится проверка: не совпадает ли она с основными константами переадресации, имеющимися в *УВК* (их адреса 7423—7431), или с константами переадресации, составленными ранее и запасенными в таблице констант переадресации. Если вновь составленная константа при этом не может быть «сэкономлена», то блок заносит ее в таблицу констант переадресации и присваивает ей условное число. Затем составляется команда переадресации, имеющая такой вид:

| | | | | |
|---|---|---|---|---|
| $4000 + N$ | $4600 + n$ | $\nu$ | 0 | 02 или 15 |

Здесь $N$ — номер оператора, подлежащего переадресации, $n$ — номер константы переадресации, $\nu$ — номер анализируемой команды, считая от начала оператора, которому она принадлежит.

Если для переадресации используется константа *УВК,* то во втором адресе команды переадресации будет записан действитель ный адрес *УВК,* например;

| | | | | |
|---|---|---|---|---|
| $4000 + N$ | 7423 | $\nu$ | 0 | 02 или 15 |

Заметим, что если некоторые адреса анализируемой команды должны быть увеличены, а другие — уменьшены, то предпочтение отдается операции 02. Только если два или все три адреса подлежат уменьшению, то применяется операция 15. Константа переадресации при этом составляется одна. Например, если второй адрес анализируемой команды подлежит уменьшению, а первый и третий — увеличению, то константа переадресации может иметь такой вид:

| | | | | |
|---|---|---|---|---|
| $h_1$ | $10000 - h_2$ | $h_3$ | 0 | 00 |

В случае переадресации с переменным шагом предполагается, что этот шаг вычислен и записан в некоторой ячейке $b$ в виде трех одинаковых чисел, стоящих в трех адресах. Блок *F* сперва составляет команду выделения шага переадресации из адресов ячейки $b$, одноименных с адресами анализируемой команды, к которым должен быть прибавлен (или отнят) этот шаг переадресации, а затемсоставляется команда переадресации. При этом команды выделения шага и переадресации имеют такой вид:

| | | | | |
|---|---|---|---|---|
| *b* | λ | *r* | 0 | 11 |
| 4000+*N* | *r* | *v* | 0 | 02 или 15 |

Здесь *b* — номер ячейки, в которой получено путем вычислений значение переменного шага переадресации, λ — константа для выделения нужного адреса числа (*b*) (имеется в *УВК*), *r*— рабочая ячейка, остальные буквы имеют прежние значения.

При составлении команд выделения переменного шага переадресации блок *F* проверяет, не была ли уже ранее в данном операторе переадресации составлена такая команда, и если была, то вторично ее не включает в оператор переадресации, сохраняя только команду

$$4000+N \quad r \quad v \quad 0 \quad (15 \text{ или } 02),$$

в которой *r* заменяется соответствующим условным числом рабочей ячейки.

Таким образом, блок *F* составляет все операторы переадресации, предусмотренные в логической схеме программируемой программы, после чего в соответствии *с BT* вызывает с магнитной ленты очередной блок.

**8. Блок *Э*.** Блок *Э* (экономии рабочих ячеек) производит экономию рабочих ячеек, использованных блоками *A, P* и *F* при составлении операторов вида **A**, **P**, **F**. Эти блоки при составлении команд использовали каждый раз новое условное число рабочей ячейки.

Блок *Э* просматривает последовательно каждую команду операторов типа **A**, **P**, **F**, начиная с последней. В свою очередь каждая команда анализируется, начиная с третьего адреса. При обнаружении условного числа $r_n$ рабочей ячейки ей присваивается новое условное число рабочей ячейки $\rho_k$. При дальнейшем просмотре команд все условные числа $r_n$ заменяются на $\rho_k$ включительно до тех пор, пока $r_n$ не встретится в третьем адресе команды. После этого условное число $\rho_k$ считается «освободившимся» и употребляется для замены нового условного числа рабочей ячейки $r_m$. Исключением является случай, когда $r_n$ присутствует сразу в третьем и втором адресах анализируемой команды. В этом случае $\rho_k$ не считается «освободившимся».

Поясним описанный алгорифм экономии рабочих ячеек на примере.

П р и м е р 1. Дана формула

$$((a + b + c)\,(a + b) — d + c^7)\,k = R.$$

Предположим, что буквы a, *b, c, d, k, R* обозначены последовательными условными числами, начиная с 5001.

Блок ***A*** *для* счета по приведенной формуле составит команды (перенумерованы с конца для удобства пояснений):

| | | | | | |
|---|---|---|---|---|---|
| 9) | 5001 | 5002 | 6003 | 0 | 01 |
| 8) | 6003 | 5003 | 6004 | 0 | 01 |
| 7) | 6003 | 6004 | 6005 | 0 | 05 |
| 6) | 6005 | 5004 | 6006 | 0 | 03 |
| 5) | 5003 | 5003 | 6007 | 0 | 05 |
| 4) | 0000 | 0004 | 0000 | 0 | 30 |
| 3) | 5003 | 6007 | 6007 | 0 | 05 |
| 2) | 6007 | 6006 | 6010 | 0 | 01 |
| 1) | 6010 | 5005 | 5006 | 0 | 05 |

Здесь условные числа рабочих ячеек начинаются с 6003, ибо условные числа 6001 и 6002 используются только блоками *P′* и *P* и экономии не подлежат. При обработке этого оператора (начиная с последней команды) блок *Э,* прежде всего, заменит в 1-й команде $r_{10} = 6010$ на $\rho_1 = 6001$. Во 2-й команде $\rho_1$ освобождается и применяется для замены им условного числа $r_6 = 6006$. В этой же команде $r_7 = 6007$ заменяется на $\rho_2 = 6002$. В 3-й команде $r_7$ присутствует в третьем и втором адресах. Поэтому $p_2$ не освобождается. В 5-й команде освобождается $\rho_2$, в 6-й команде освобождается и $\rho_1$. В этой же команде $\rho_1$ используется для замены им числа $r_5 = 6005$. В 7-й команде $\rho_1$ освобождается и тут же применяется для замены им $r_4 = 6004$; $\rho_2$ применяется для замены числа $r_3 = 6003$. В 7-й команде освобождается $\rho_1$, а в 9-й команде — $\rho_2$. Экономия ячеек окончена и после нее обработанная группа команд принимает такой вид:

| | | | | | |
|---|---|---|---|---|---|
| 9) | 5001 | 5002 | 6002 | 0 | 01 |
| 8) | 6002 | 5003 | 6001 | 0 | 01 |
| 7) | 6002 | 6001 | 6001 | 0 | 05 |
| 6) | 6001 | 5004 | 6001 | 0 | 03 |
| 5) | 5003 | 5003 | 6002 | 0 | 05 |
| 4) | 0000 | 0004 | 0000 | 0 | 30 |
| 3) | 5003 | 6002 | 6002 | 0 | 05 |
| 2) | 6002 | 6001 | 6001 | 0 | 01 |
| 1) | 6001 | 5005 | 5006 | 0 | 05 |

При этом оказывается, что в командах используются всего две рабочие ячейки вместо шести.

Кроме экономии рабочих ячеек, блок *Э* определяет наибольшее условное число рабочих ячеек, использованное в программе после экономии, и записывает его в стандартную ячейку. Это условное число необходимо для работы блока *П.* Определяя его, блок *Э* просматривает также операторы вида $\mathbf{C}_i$ и $\mathbf{H}_i$, учитывая и рабочие ячейки, используемые в групповых операциях.

После окончания описанной работы производится в соответствии с *BT* вызов очередного блока.

**9. Блок *О*.** Блок *О* (восстановления) составляет операторы восстановления и должен работать после блоков *F* и *Э*.

Информация для блока *О* задана в первом массиве в виде строк

| | | | | |
|---|---|---|---|---|
| $N_1$ | $N_2$ | 0000 | 0 | 00 |

где $N_1$ и $N_2$ — условные числа операторов. Все операторы, условные числа N которых удовлетворяют неравенству

$$N_1 \le N \le N_2$$

подлежат восстановлению в первоначальном виде.

Блок *О* просматривает все операторы переадресации и находит в их командах номера операторов и номера команд, подлежащих восстановлению; по номеру оператора и номеру команды находит саму восстанавливаемую команду и, если она не содержит адресов вида 7000 + *n,* заносит ее в таблицу констант восстановления (вместе с индикатой, если такая имеется). Если команда содержит адреса вида 7000 + *n*, то к ней составляется индиката (если ее нет). В соответствующие адреса индикаты ставится условное число номера оператора, содержащего восстанавливаемую команду, а в самой команде вместо адресов вида 7000 + *n* ставятся числа *n.* Если восстанавливается одиночная команда, то составляется команда восстановления вида

| | | | | |
|---|---|---|---|---|
| 4000 + *N* | 4400 + *n* | *v* | 0 | 00 |

где *N* — номер восстанавливаемого оператора, 4400 + *n* — условное число константы восстановления, *v* — номер восстанавливаемой команды, отсчитанный от начала оператора *).

В том случае, когда восстанавливаемые команды стоят единым массивом, блок *О* формирует групповую команду восстановления, имеющую вид

| | | | | |
|---|---|---|---|---|
| 4000 + *N* | 4400 + *n* | *v* | 0 | *m* |

Здесь *N* — номер восстанавливаемого оператора, 4400 + *n* — условное число первой из констант восстановления (остальные следуют непосредственно за ней), *v* — номер первой из восстанавливаемых команд, отсчитанный от начала оператора*), *m* — количество восстанавливаемых команд, образующих сплошной массив.

Как уже говорилось, блок *О* после своей работы может выдавать на перфокарты программу, составленную в условных числах.

Недостатком блока *О* является то, что он производит восстановление команд в их первоначальном (до переадресации) виде, а не восстановление по параметру.

**10. Блок *П*.** Блок *П* (присвоения действительных адресов) перерабатывает программу, составленную в условных числах, в программу с действительными адресами. К моменту его работы первый массив информации уже представляет собой программу в условных числах. Операторы первого массива еще снабжены характеристиками. Для присвоения командам программы действительных адресов блок *П* использует третий массив информации, состоящий из *ТРП* и *ТРВ.* Стандартное распределение памяти (если оно выбрано для составляемой программы) производится блоком *П* без участия человека. Блок *П* начинает свою работу с просмотра ячейки, хранящей признак вида распределения памяти. При наличии в этой ячейке признака стандартного распределения памяти блок *П* составляет *ТРП,* подсчитывая, а также используя для этого запасенные другими блоками в стандартных ячейках наибольшие применяемые в программе условные числа, общее количество команд, содержащихся в программе, величин, констант восстановления, констант переадресации, числовых констант и рабочих ячеек. После составления *ТРП* блок *П* с ее помощью перерабатывает *ТРВ,* заменяя в ней условные адреса величин истинными адресами. Затем составляется *ТХ* (таблица характеристик).

При обнаружении блоком *П* признака нестандартного распределения памяти работа блока *П* начинается с составления *ТХ.*

Характеристики изымаются из состава первого массива информации, и в их третьих адресах проставляются истинные номера ячеек, предназначенных для первых команд, отвечающих характеристикам операторов. Затем эти характеристики в том взаимном порядке, в каком они находились в первом массиве информации, размещаются на специально отведенном месте оперативной памяти. В дальнейшем *ТХ* используется при обработке команд, оставшихся в первом массиве.

Следующий этап работы блока состоит в замене условных чисел действительными адресами.

Выполняется это в следующей последовательности:

1. Просматривается каждый адрес, и, если в нем обнаруживается условное число величины, оно заменяется ее действительным адресом, взятым из *ТРВ.* Если в адресе встречается условное число константы (восстановления, переадресации или числовой) или рабочей ячейки, блок *П* стирает первые (двоичные) цифры этого условного числа (определяющие его вид), уменьшает результат на единицу и прибавляет к нему номер начальной ячейки соответствующей таблицы констант или массива рабочих ячеек, взятый из *ТРП.*

2. В индикатах производится замена номеров операторов уменьшенными на единицу номерами действительных начальных ячеек этих операторов (в соответствии с *ТХ*). Затем индиката с помощью операции 02 (специальное сложение) складывается со следующей за ней командой (или константой восстановления).

*) При определении номера команды характеристика оператора не учитывается.

3. В командах переадресации условные числа номеров операторов и номеров команд, отсчитанных от начала этих операторов, заменяются уменьшенной на единицу суммой номера начальной ячейки оператора и номера команды, отсчитанного от начала оператора. При этом в соответствующих адресах команд переадресации оказываются действительные номера ячеек, отведенных для подлежащих переадресации команд.

4. Просматриваются команды восстановления. Третий адрес команды восстановления составляется так же, как первый и третий адреса команд переадресации. Номер ячейки, отведенной для константы восстановления, перемещается из второго адреса команды в первый адрес. Затем содержимое кодовых разрядов ячейки переносится во второй адрес ячейки. Если это содержимое равно нулю в кодовые разряды ставится код операции 13 (команда одиночного восстановления), а если отлично от нуля, то второй адрес уменьшается на единицу и в кодовые разряды ячейки ставится «45» (команда группового восстановления).

5. В командах условного перехода (с кодами 20 и 27) условные числа номеров операторов заменяются действительными номерами начальных ячеек операторов.

6. Адреса вида 7000 + *n* (условные числа номеров команд внутри данного оператора) преобразуются путем стирания цифры 7, стоящей в начале такого условного числа, и прибавления к оставшемуся числу уменьшенного на единицу начального адреса того оператора, в котором расположена обрабатываемая команда.

7. В ходе присвоения действительных адресов команда, содержащая код операции 25 (подвод магнитной ленты), не видоизменяется. В командах, содержащих контрольный знак, равный 1, обрабатываются только первый и третий адреса, второй адрес остается неизменным. После обработки такой команды контрольный знак заменяется нулевым.

Адреса команд, не превосходящие восьмеричного числа 4000, а также адреса не меньшие, чем восьмеричное число 7400 (это — действительные адреса), не подвергаются изменениям.

После окончания присвоения действительных адресов блок *П* выдает на перфокарты готовую программу, таблицы констант и *ТХ*.

Примечание. За время подготовки этой книги к печати *ППС* подвергалась дальнейшим улучшениям. Блоки *А′* и *Р′* объединены в один блок; блок *О* переделан в блок восстановления по параметру; составлен новый блок — блок циклов. Такое изменение *ППС* привело к упрощению вида задаваемой ей информации и к улучшению качества составляемых ею программ.

# ГЛАВА XI
# НЕАРИФМЕТИЧЕСКИЕ ВОЗМОЖНОСТИ ЭЛЕКТРОННЫХ ЦИФРОВЫХ МАШИН

## § 54. Машинный перевод

Перевод с одного языка на другой относится к одному из весьма трудоемких и однообразных видов умственного труда.

Работа переводчика может быть описана в виде сложной системы большого количества правил (т. е. может быть алгорифмизирована), руководствуясь которыми можно переводить тексты, не понимая их смысла.

В настоящее время алгорифмизация работы переводчика произведена лишь приближенно. Тем не менее разработанная система правил с успехом может быть применена для перевода научных и технических текстов.

Алгорифм перевода можно представить в виде программы для электронной цифровой машины.

При составлении такой программы необходимо обеспечить получение с помощью машины правильного в смысловом отношении перевода, который был бы понятен соответствующему специалисту, не знающему иностранного языка, и пригоден для окончательного редактирования.

При составлении программ для машинного перевода возникают две основные трудности:

а) если переводимому слову отвечает несколько эквивалентов на другом языке, нужно добиться, чтобы машина выбирала тот из них, который требуется по смыслу текста;

б) после того как слова одного языка заменены словами другого языка, последние должны быть грамматически достаточно правильно согласованы между собой и объединены в предложения.

Эти трудности, вообще говоря, играют основную роль и при обычном переводе в связи с чрезвычайно большим разнообразием грамматических правил, наличием большого количества исключений и зависимостью смысла отдельных слов от контекста.

На основе анализа значений и способов применения отдельных слов в предложениях нужно установить однозначные правила, по которым чаще всего и в наиболее важных случаях используются слова на втором языке. Работы по алгорифмизации перевода, проведенные в различных учреждениях, показали, что системы таких правил могут значительно отличаться одна от другой.

Проблема машинного перевода, помимо практического значения, имеет и большой теоретический интерес, так как для программирования необходима полная формализация процесса перевода, что требует доведения до предельной отчетливости способов описания выражений одной и той же мысли на различных языках. Связанные с этим исследования вызвали к жизни новую область языкознания — *математическую лингвистику*.

Эксперименты по автоматическому переводу, проведенные на разных цифровых электронных машинах, показали необходимость создания специальных «переводческих» машин.

Для развития проблематики машинного перевода и математической лингвистики необходима совместная работа лингвистов, математиков и инженеров по электронным цифровым машинам, а также тесное взаимодействие между группами научных работников, разрабатывающих различные переводческие алгорифмы. Существенное значение имеет разработка символики для записи переводческих правил и схем переводческих программ, а также изучение упрощенных моделей языков.

Для машинного перевода, прежде всего, должен быть составлен специальный *машинный словарь*. Учитывая трудности проблемы на начальном этапе, составляются отдельные словари для определенных областей науки и техники, так называемые *отраслевые словари.*

Особенности машинного словаря обусловлены характером переводческого алгорифма и особенностями обоих связываемых им языков. Описанный ниже словарь для перевода с русского языка на английский содержит в качестве переводимых слов иногда лишь корни слов, иногда слова во множественном числе (например, слово «мысли»), некоторые глаголы повторяются в нем по нескольку раз в различных временах и с разными личными окончаниями. Отдельно включены в него падежные окончания существительных. В словаре указывается для каждого переводимого слова один или несколько эквивалентов и, кроме того, для каждого слова даются так называемые дополнительные коды (условные числа), определяющие характер использования этого слова в предложениях (в типовых случаях). Например, дополнительными кодами могут определяться: связь данного слова с предыдущими или последующими словами, рекомендации по целесообразному выбору одного из нескольких эквивалентов данного слова, связь с предлогами и др. Эти дополнительные условные числа необходимы для управления автоматической работой машины в процессе перевода.

Программа работы машины по автоматическому переводу чрезвычайно сложна (она содержит несколько тысяч команд). По своему характеру она до некоторой степени напоминает универсальную программирующую программу, предназначенную для автоматического составления машиной вычислительных программ. Между обоими программами существует принципиальное сходство: обе программы описывают процесс преобразования информации из одной формы в другую без изменения ее содержания.

**1. Опыт перевода с русского языка на английский на машине *ИБМ*-701.** В настоящее время работы по переводу с одного языка на другой с помощью электронных цифровых машин находятся в начальной стадии, однако полученные уже в этом направлении результаты показывают его практическую осуществимость в ближайшем будущем.

В январе 1954 г. в Нью-Йорке была проведена экспериментальная публичная демонстрация перевода с русского языка на английский с помощью электронной цифровой машины *ИБМ*-701. Для испытания был специально подготовлен словарь из 250 русских слов, применяемых в области политики, юстиции, математики, химии, металлургии, связи и военного дела, записанных латинскими буквами (машина *ИБМ-701* может воспринимать непосредственно печатный буквенный текст в латинском алфавите). Слова были подобраны так, чтобы каждое русское слово имело один или два английских эквивалента. Каждому слову были приписаны в словаре три дополнительных кода. Выдержки из словаря приведены в таблице 54.

Предварительная работа по машинному переводу велась в течение полутора лет институтом языкознания Джорджтаунского университета и специалистами по электронным цифровым машинам лабораторий фирмы *ИБМ.* Разработаны были основные синтаксические правила, которые при помощи упомянутых трех дополнительных кодов, указанных в словаре, обеспечивают возможность формального перевода правильно построенных стандартных фраз научного или политического характера. Как указывается в литературе, для перевода произвольного русского текста на английский язык требуется около 100 правил такого рода. Приводим некоторые из основных правил:

1. Определение перестановки по предыдущему тексту. Если первый код переводимого слова есть 110, то производится проверка третьего кода предыдущего полного слова. Если он равен 21, то надо изменить порядок этих слов (т. е. слово с третьим кодом 21 должно следовать за словом с первым кодом 110), если он не равен 21, то порядок сохраняется. В обоих случаях для слова с первым кодом 110 берется первый английский эквивалент.

2. Выбор первым способом эквивалентен по последующему тексту. Если первый код данного слова есть 121**,** то проверяется, чему равен второй код следующего полного или части слова (корня или окончания). Если этот второй код равен 221, то для слова с кодом 121 берется первый английский эквивалент; если второй код следующего слова равен 222, то берется второй английский эквивалент. В обоих случаях порядок слов сохраняется.

Таблица 54. Выдержки из словаря для *ИБМ*-701

| Русские слова | Английский эквивалент | | Коды | | |
|---|---|---|---|---|---|
| | I | II | I | II | III |
| к | to | for | 121 | 000 | 23 |
| кислородн — | oxygen | — | 000 | 000 | 00 |
| лишени — | deprival | — | 000 | 222 | 00 |
| материал — | material | — | 000 | 000 | 00 |
| мы | we | — | 000 | 000 | 23 |
| мысли | thoughts | — | 000 | 000 | 00 |

| Русские слова | Английский эквивалент | | Коды | | |
|---|---|---|---|---|---|
| | I | II | I | II | III |
| мног - | many | — | 000 | 000 | 00 |
| медь | copper | — | 000 | 000 | 21 |
| мест — | place | site | 151 | 000 | 23 |
| механическ — | mechanical | — | 000 | 242 | 00 |
| международн — | international | — | 000 | 000 | 00 |
| на | on | for | 121 | 000 | 23 |
| нападени — | attack | attacks | 121 | 000 | 00 |
| наука | a science | — | 000 | 242 | 00 |
| обработка | processing | — | 000 | 000 | 00 |
| объект | objective | objectives | 121 | 000 | 00 |
| офицер | an officer | | 000 | 000 | 00 |
| — ого | of | — | 131 | 000 | 23 |
| — ом | by. | — | 131 | 000 | 00 |
| определяет | determines | — | 000 | 000 | 00 |
| определяется | is determined | — | 000 | 000 | 00 |
| оптическ — | optical | — | 000 | 000 | 00 |
| орудие | gun | — | 000 | 241 | 00 |
| отдел — | section | — | 000 | 000 | 00 |
| отделение | devision | squad | 121 | 242 | 00 |
| отношение | relation | the relation | 151 | 000 | 00 |

3. Выбор эквивалента и определение перестановки по предыдущему тексту. Если первый код данного слова есть 131, а третий код предыдущего полного слова или части слова (корня или окончания) равен 23, то для слова с кодом 131 надо взять второй английский эквивалент и сохранить порядок следования слов. В остальных случаях для слова с кодом 131 нужно взять первый английский эквивалент и изменить порядок следования слов на обратный.

4. Выбор эквивалента по предыдущему тексту. Если первый код данного слова есть 141, а второй код предыдущего полного слова или предыдущей части слова (корня или окончания) есть 241, то для каждого слова берется первый английский эквивалент, если же второй код равен 242, то берется второй эквивалент. Порядок слов сохраняется. При других возможных значениях второго кода предыдущего слова выбор эквивалента производится по другим правилам.

5. Выбор вторым способом эквивалента по последующему тексту. Если первый код данного слова есть 151, а третий код следующего полного или части слова (корня или окончания) равен 25, то для слова с первым кодом 151 берется второй английский эквивалент, в остальных случаях берется первый эквивалент. Порядок слов сохраняется.

6. Независимый перевод. Если первый код данного слова есть 000, то для этого слова берется первый английский эквивалент и сохраняется порядок по отношению к предыдущему слову.

В качестве примера, поясняющего применение этих правил, приводится перевод предложения: «Величина угла определяется отношением длины дуги к радиусу». В таблице 55 записаны выбранные из словаря слова и части слов, соответствующих заданному предложению, с указанием английских эквивалентов и дополнительных управляющих кодов. В последней колонке для пояснения указаны номера применявшихся правил. Формальное применение указанных правил обеспечивает получение правильного перевода: «Magnitude of angle is determined by the relation of length of arc to radius».

При проведении опытного машинного перевода составленный словарь из 250 слов был записан на перфокартах вместе с управляющими кодами, а затем введен в магнитное запоминающее устройство машин. Программа машинного перевода, разработанная для машины *ИБМ*-701, содержала около 2400 одноадресных команд.

Эта программа также была полностью введена в машину. После этого в машину вводились русские предложения, подлежащие переводу. Предложения пробивались на перфокартах стандартным кодом *ИБМ*-701. Через 5—8 сек. после ввода данных машина печатала на выходном устройстве английский перевод.

Таблица 55, Пример перевода

| Русские слова | Английский эквивалент | | Коды | | | Какое правило применено |
|---|---|---|---|---|---|---|
| | I | II | I | II | III | |
| величина | magnitude | | 000 | 000 | 00 | 6 |
| угол — | coal | angle | 121 | 000 | 25 | 2 |
| — а | of | — | 131 | 222 | 25 | 3 |
| определяется | is determined | — | 000 | 000 | 00 | 6 |
| отношени — | relation | the relation | 151 | 000 | 00 | 5 |
| — ем | by | — | 131 | 000 | 00 | 3 |
| длин — | length | — | 000 | 000 | 00 | 6 |
| — и *) | of | — | 131 | 000 | 25 | 3 |

| Русские слова | Английский эквивалент | | Коды | | | Какое правило применено |
|---|---|---|---|---|---|---|
| | I | II | I | II | III | |
| дуг — | arc | — | 000 | 000 | 00 | 6 |
| — и | of | — | 131 | 000 | 25 | 3 |
| к | to | for | 121 | 000 | 23 | 2 |
| радиус — | radius | — | 000 | 221 | 23 | 6 |
| — у | to | | 131 | 000 | | 3 |
| *) Русские окончания Ы и И при переводе не различаются. | | | | | | |

В отчете о пробном машинном переводе на *ИБМ*-701 приводятся следующие примеры переводов:

Качество угля определяется калорийностью.

Крахмал вырабатывается механическим путем из картофеля.

Обработка повышает качество нефти.

Динамит приготовляется химическим процессом из нитроглицерина с применением инертных соединений.

Международное понимание является важным фактором в решении политических вопросов.

The quality of coal is determined by calory content.

Starch is produced by mechanical methods from potatoes.

Processing impoves the quality of crude oil.

Dynamite is prepared by chemical process from nitroglycerine with admixture of inert compaunds.

International understanding constitutes an important factor in decision of political questions.

Проведенные первые опыты сразу же показали полную возможность практического использования электронных цифровых машин для выполнения переводов с одного языка на другой, в первую очередь для выполнения переводов научных и технических текстов. Последние характеризуются большим однообразием и правильностью в построении фраз и большей определенностью в использовании слов, чем художественная литература.

**2. Опыт перевода с английского языка на русский на машине *БЭСМ*.** Работа по применению машин для перевода с одного языка на другой успешно ведется в нашей стране в Институте точной механики и вычислительной техники АН СССР и в Математическом институте имени В. А. Стеклова АН СССР. Под руководством профессора Д. Ю. Панова была составлена программа для перевода с английского языка на русский с помощью быстродействующей электронной счетной машины (*БЭСМ*). При разработке «переводческого алгорифма» был избран путь, состоящий в воспроизведении работы, выполняемой переводчиком.

Эта работа слагается из следующих этапов:

1. Чтение английской фразы, подлежащей переводу.

2. Выявление тех слов переводимой фразы, которые знакомы переводчику. Выяснение некоторых грамматических признаков этих слов как по их окончаниям, так и путем сопоставления их друг с другом и с остальными словами фразы.

3. Отыскание остальных слов фразы в словаре, получение отвечающих им русских слов и выяснение относящихся к ним грамматических признаков. Сюда относятся:

а) непосредственный перевод слов, имеющих один русский эквивалент;

б) отыскание слов с отсылкой (т. е. слов, являющихся производными от других; например, для перевода these в словаре делается отсылка к слову this);

в) выбор значения слов, имеющих несколько русских эквивалентов, путем сопоставления этих слов с другими словами фразы.

4. Отбрасывание слов, которые не войдут в состав русской фразы (например, вспомогательных глаголов).

5. Составление русской фразы из найденных слов в соответствии с полученными грамматическими признаками.

6. Выявление английских слов, не входящих в состав применяемого словаря, для перевода их с помощью других средств.

7. Окончательное редактирование русской фразы.

Оставляя в стороне этап 7, являющийся литературной обработкой перевода, мы видим, что все операции, производимые переводчиком, можно более или менее точно описать с помощью системы правил, позволяющей выполнять их автоматически.

Однако и степень сложности правил и необходимый объем словаря зависят от вида переводимого текста.

Так, для перевода художественной литературы нужны запас в несколько сотен тысяч слов и специальный словарь идиом (выражений, не допускающих дословного перевода на другой язык, вроде «съел на этом деле собаку» и т. д.). Кроме того, при выполнении такого перевода возникают еще трудности, обусловленные тем, что выражения, применяемые в художественной литературе, бывают тесно связанными с природой языка, а также с бытом и жизнью народа.

Напротив, для перевода научных и технических текстов нужен словарь приблизительно из двух тысяч слов,

среди которых количество слов, имеющих по нескольку эквивалентов на другом языке, относительно невелико. Идиомы в таких текстах совершенно отсутствуют. Построение фраз сравнительно простое.

Общая схема автоматического перевода научно-технических текстов состоит из четырех этапов.

Первый этап представляет собой ввод английского текста в машину. Второй этап состоит в анализе английского текста и переводе английских слов. Третий этап представляет собой синтез из полученных русских слов русского текста и, наконец, четвертый, последний, этап — вывод из машины русского текста.

Для ввода переводимого текста в машину каждая буква латинского алфавита заменяется определенным набором цифр. При этом каждое английское слово оказывается замененным некоторым числом.

По той же системе в запоминающем устройстве машины записаны английские слова, входящие в словарь. Отыскание слов в словаре легко производится с помощью операции сравнения.

Часто приходится встречаться со словами, которые не фигурируют в словаре непосредственно. Например, мы не найдем там слова equations потому, что оно имеет окончание — s, отвечающее множественному числу, тогда как в словаре приведены существительные в единственном числе. Кроме окончания — s, могут встречаться и другие окончания, отличающие переводимое слово от слов, имеющихся в словаре, например: — ing, —ed, —er, —est, —e, —y и т. д.

Если слово переводимого текста не совпадает в точности ни с одним из слов, приведенных в словаре, то производится проверка: не имеет ли слово одного из упомянутых выше окончаний. Найденное окончание отбрасывается, после чего снова производится поиск слова в словаре.

Словарь для автоматического перевода состоит из двух частей: английской и русской.

В английской части словаря каждое английское слово приведено в своем цифровом обозначении и снабжено порядковым (словарным) номером. Кроме номера соответствующего русского слова (или нескольких номеров, если имеется несколько русских эквивалентов), сюда включен также ряд признаков (тоже закодированных в виде чисел), относящихся к грамматике русского слова. Например, для существительного указаны его род, склонение, означает ли оно одушевленный предмет и т. п.

Выбор нужного русского слова, когда английскому слову соответствует несколько русских, производится с помощью отдельной части программы — дополнения к словарю. Эта часть программы представляет собрание правил, по которым проверяется связь переводимого слова с предыдущими и последующими словами. От нее и зависит, какое из русских слов должно быть выбрано. Если слово из переводимой фразы не будет найдено ни в словаре, ни с помощью дополнения к словарю, то это слово останется непереведенным.

Русская часть словаря состоит из русских слов, тоже записанных в виде чисел, получающихся заменой букв русского алфавита определенными наборами цифр.

После того как найден номер русского слова, составляется цифровой эквивалент слов (для записи которого в машине *БЭСМ* требуются две ячейки). В него входят: номер английского слова, номер русского слова, взятые из словаря грамматические сведения о русском слове. Эти грамматические сведения представлены в виде наборов цифр, каждый из которых записывается всегда в одних и тех же разрядах ячейки, что облегчает их нахождение.

На этом заканчивается работа первой части программы для перевода, осуществляющей анализ английского текста.

Вторая часть программы производит синтез русских предложений, видоизменяя русские слова, извлекаемые из словаря, в соответствии с правилами грамматики и расставляя их по местам.

В программу перевода входят еще подпрограммы, носящие названия «синтаксис» и «изменение порядка слов». Первая из них расставляет знаки препинания, а вторая изменяет в русской фразе расположение слов по правилам русской грамматики.

Для опытов автоматического перевода на электронной счетной машине *БЭСМ* был составлен словарь из 952 английских и 1073 русских слов, предназначенный для перевода математического текста.

Ниже приведены английский текст из книги Милна «Численное решение дифференциальных уравнений» и его русский перевод, произведенный машиной.

«Introduction

When a practical problem in science or technology permits mathematical formulation, the chances are rather good that it leads to one or more differential equations. This is true certainly of the vast category of problems associated with force and motion, so that wherever we want to know the future path of Jupiter in the heavens or the path of an electron electron microscope we resort to differential equations. The same is true for the study of phenomena in continuous media, propagation waves, flow of heat, diffusion, static or dinamic electricity, etc., except we here deal with partial differential equations».

«Введение

Если практическая задача в на- уке или технике допускает математическую формулировку, шансы довольно велики, что это приводит к одному или более дифференциальным уравнениям. Это верно безусловно для обширной категории задач, свя занных с силой и движением, так чтохотим ли мы знать будущий путь in an Юпитера в небесах или путь электрона в электронном микроскопе, мы прибегаем к дифференциальным уравнениям. То же верно для, изучения явлений в непрерывной среде, распространения волн, that потока тепла, диффузии, статического или динамического электричества, и т. д., за исключением того, что мы здесь будем рассматривать дифференциальные уравнения в частных производных».

.

### 3. Опыт перевода с французского языка на русский с помощью машины *Стрела*.

Весьма интересная работа в области автоматического перевода текстов с одного языка на другой проделана под общим руководством профессора А. А. Ляпунова двумя группами научных сотрудников, из которых одна возглавляется О. С. Кулагиной, а другая — Т. Н. Молошной.

Группа О. С. Кулагиной составила программу для перевода с помощью машины *Стрела* текстов с французского языка на русский, а группа Т. Н. Молошной — программу для перевода на той же машине текстов с английского языка на русский.

При разработке программы для переводов французских текстов было установлено, что обычная грамматика является недостаточно формальной и требует в ряде случаев для отнесения слов к тому или иному грамматическому классу учета их смысла. Так как при автоматическом переводе смысловое содержание слов не учитывается, то была разработана «машинная» грамматика французского и русского языков, в которой слова классифицируются по чисто формальным принципам. Наименования для классов слов (частей речи) были сохранены общепринятые, хотя состав этих классов отличается от принятых в обычной грамматике.

Приступая к составлению переводящей программы, ставили перед собой задачу добиться получения переводов, правильно передающих смысл оригинала и построенных правильно в грамматическом отношении. Программа предназначалась для переводов математических и технических, а не литературных или газетных текстов. Это ограничение было принято для того, чтобы уменьшить объем словарей за счет:

а) отсутствия идиом,

б) уменьшения числа встречающихся в тексте различных слов,

в) уменьшения многозначности слов (например, в математическом тексте глагол poser имеет смысл «полагать», а в литературном тексте он может, кроме того, означать «класть», «ставить» и т. п.).

Для составления отраслевого словаря было обследовано большое количество научных текстов (в общей сложности свыше 20 000 слов), в составе которых было обнаружено 3200 различных слов. Из этих слов были отобраны те, которые встречались в обследованных текстах более четырех раз. Таких слов оказалось 1156. Они и были включены в машинный словарь. Кроме того, при анализе текстов было выявлено около 250 «оборотов». *Оборотами* называются словосочетания, не допускающие пословного перевода (не смешивать с идиомами), такие, как, например, «par exemple» (при пословном переводе получится «на примере», тогда как следует перевести «например»).

Машинный словарь, как и в программе, разработанной в Институте точной механики и вычислительной техники, был составлен из двух частей: французской и русской. В словарь включены не сами слова, а их «основы». *Основой* называется часть слова, которая при ряде грамматических изменений этого слова остается графически неизменной. Например, для слова «poser» и «книжка» основами являются «pos» и «книжк». Если основа слова недостаточна (состоит из очень малого количества букв) или в разных случаях имеет различный вид, в словарь включается несколько основ данного слова. Например, для глагола «быть» необходимо указать четыре основы «быть», «есть», «был», «буд». Первые две из них совпадают с соответствующими формами глагола «быть», третья присутствует в словах «был», «была», «были», четвертая — в словах «буду», «будешь», «будет» и т. д. Каждая основа во французской части словаря снабжена дополнительной информацией, содержащей ряд признаков, из которых главными являются:

а) номер перевода, т. е. соответствующего слова в русской части словаря (если переводов — русских слов — несколько, то они приведены в русской части словаря под общим номером);

б) признак, указывающий на то, какой частью речи является слово;

в) указание на возможность участия слова в оборотах (в виде номера «особенности» — пояснено ниже);

г) указание на наличие омонимии (пояснена ниже);

д) грамматическая характеристика (например, род существительного, вид глагола и т. п.).

В русской части словаря каждая основа также снабжена дополнительной информацией (тип выбора основ, вид изменения окончаний и др.). Словарь оборотов записан в виде последовательности номеров особенностей слов, составляющих оборот, и признаков, позволяющих при синтезе русской фразы учесть эти особенности.

Алгорифм перевода получен эмпирически, путем анализа работы переводчика. Переводящая программа осуществляет перевод «по-фразно». Иногда пофразный перевод оказывается недостаточным, например невозможно решить вопрос о точном переводе местоимения, заменяющего слово, употреблявшееся в другой фразе. Однако редактирование полученного перевода человеком в этих случаях не представляет труда.

Первым этапом перевода является поиск (в алфавитном порядке) слов, входящих в состав переводимой фразы, во французской части словаря и извлечение оттуда информации об этих словах.

Затем производится анализ французской фразы. Этот анализ производится в последовательности, определяемой принадлежностью слов к частям речи:

1) глаголы,
2) предлоги,
3) существительные,
4) местоимения,
5) причастия,
6) прилагательные.

Этот порядок не является случайным и обусловлен грамматическими особенностями частей речи: глаголы допускают независимый анализ; предлоги управляются глаголами; падежи существительных определяются предлогами, и т. д.

Под *омонимией* понимается многозначность переводимых слов. Омонимия бывает двух родов. Омонимия первого рода состоит в том, что данное слово может быть использовано в качестве различных частей речи (например, в русском языке слово «столовая» может быть употреблено как существительное и как прилагательное). Такая омонимия французских слов устраняется путем анализа их окружения в фразе. Омонимия второго рода представляет собой многозначность слова, которое может быть применено только как одна вполне определенная часть речи (например, в русском языке существительное «коса» может означать: 1) прическа, 2) режущий инструмент, 3) отмель в реке). Такая омонимия французского слова не может быть устранена путем формального анализа фразы. Для ее устранения необходимо понимать смысл фразы или иметь список слов (по-видимому, чрезвычайно обширный), наличие которых в окружении омонима определяет его перевод. Поясним это на примере русских омонимов. Если говорится расчесала, заплела (или тому подобное) косу или густая, пышная и т. п. коса, то коса — это прическа. Наоборот, если сказано точить, затупить, поломать (или тому подобное) косу, то коса — это режущий инструмент. Ограниченность запоминающих устройств машины в настоящее время не позволяет применять такие перечни раскрывающих омонимию слов. Переводящая программа в случае омонимии второго рода ставит в переводе сразу все русские значения французского омонима. Выбор необходимого значения должен осуществить человек при редактировании перевода.

После анализа французской фразы производится синтез русской фразы и выдача ее на перфокарты.

Переводящая программа состоит из 17 блоков. Практика показала, что при переводе фраз, состоящих не более чем из 43 слов, требуется выполнить 45—50 тысяч тактов, на что расходуется около 20—25 сек. машинного времени. Однако ввиду того, что словарь (который не может быть размещен в оперативной памяти машины *Стрела*) записан на магнитной ленте, обращение к словарю отнимает около 2—2,5 минуты. При осуществлении перевода машина выполняет от общего числа операций около 45% логических операций (логическое умножение, логическое сложение, сдвиги, сравнение), около 25% переадресаций и около 25% передач управления (условных переходов). Лишь около 5% приходится на долю других операций. Французская часть словаря требует около 18 000 ячеек памяти, а русская — около 12 000.

Таким образом, проделанная работа показывает полную возможность и целесообразность создания электронных цифровых специализированных машин, предназначенных для автоматического перевода текстов. Переводящая машина должна иметь большое по емкости постоянное запоминающее устройство, содержащее обе части словаря и позволяющее производить быстрое считывание. Программа для перевода может также храниться в этом запоминающем устройстве. Для перехода от перевода с одного языка к переводу с другого языка необходимо предусмотреть возможность смены блоков постоянной памяти, хранящих первую часть словаря и подлежащие замене части переводящей программы. Такая машина могла бы производить удовлетворительный перевод текстов со скоростью, превышающей скорость, возможную для человека.

Первый перевод с французского языка на русский был произведен с помощью описанной программы в июне 1956 г.

Была переведена французская фраза: «Nous allons conciderer d'abord deux sistems d'equations». При этом была получена русская фраза «Мы рассмотрим сначала две системы уравнений» (приведенный перевод редактированию не подвергался).

При составлении вышеописанной программы было замечено, что она состоит из нескольких десятков типов обобщенных операторов, причем операторы каждого типа различаются значениями некоторого количества определенных параметров. Это позволило произвести описание переводящей программы в виде последовательности символов, каждый из которых обозначает тип оператора, указывая в скобках возле каждого символа значения параметров, придающих ему конкретный смысл. Такая возможность навела на мысль составить «заготовки» операторов каждого типа и построить компилирующую программу, которая по перечню операторов и указанию значений определяющих их параметров составляла бы из этих заготовок программы для перевода с того или иного языка некоторой группы родственных языков. Такая компилирующая программа составлена О. С. Кулагиной. Информация для компилирующей программы в виде строки стандартных операторов с указанием значений их параметров должна быть задана лингвистом или математиком совместно с лингвистом.

Большой интерес представляет также программа перевода с английского языка на русский, составленная группой Т. Н. Молошной. В основу этой программы положен другой принцип перевода, основанный на структурном анализе английского и русского языков. Описание этой программы не приводим ввиду трудностей, связанных с изложением сущности структурного анализа языков.

Из сказанного ясно, что в ближайшее время электронные цифровые машины найдут широкое применение в области переводов научных и технических текстов с одного языка на другой.

Переводы, полученные сейчас на начальном этапе, еще далеки от совершенства. Задача дальнейшей работы заключается в совершенствовании методики перевода, в разработке полной и точной системы правил перевода.

Естественно, что усложнение системы правил перевода приведет к дальнейшему увеличению и усложнению программы работы цифровой машины.

В связи со спецификой работы по автоматическому переводу и недостаточной приспособленностью для этой цели обычных электронных цифровых машин возникает задача создания специализированных цифровых машин, предназначенных для автоматического перевода. Эти машины должны иметь большой объем внешних накопителей для хранения словаря, большой объем памяти для хранения программы, обладать большим быстродействием и выполнять основные логические операции. В то же время арифметическое устройство в машинах может быть значительно проще, чем в цифровых вычислительных машинах универсального назначения.

### § 55. Машинная игра

Для исследования логических возможностей машин и для разработки совершенных автоматических устройств весьма интересной и важной является проблема создания машин, обладающих способностью выбирать (вычислять) оптимальные ходы в различных играх (шахматы, шашки, кости, карты и др.), а также проблема разработки соответствующих программ для универсальных цифровых машин.

Обычные правила игры представляют собой вполне определенный комплекс условий, которые позволяют задать в машину с большей или меньшей степенью определенности необходимые исходные данные для целеустремленной работы по вычислению оптимальных вариантов.

Дискретная природа известных игр (расчленение на отдельные чередующиеся ходы) хорошо согласуется с цифровым принципом работы электронных цифровых машин, не требуя каких либо преобразований непрерывных величин в дискретные и обратно.

Машины для выбора оптимальных ходов могут быть основаны на нескольких различных принципах, определяющих степень совершенства, сложность и гибкость «игры» машины.

В соответствии с этим «играющие» машины могут быть разделены на четыре основные группы в порядке возрастания степени совершенства и сложности «игры» этих машин.

**1. Машины типа «словарь».** В машинах этого типа заранее должны быть записаны в памяти все ответы на все положения, которые могут возникнуть в результате ходов противника.

В процессе игры машина просто сравнивает каждое очередное положение с тем запасом, который хранится в памяти, отыскивает такое же положение и соответствующий ему ответ.

Машины, основанные на этом принципе, должны иметь чрезвычайно большие запоминающие устройства, которые могли бы вместить огромное число возможных вариантов.

Ясно, что объем работы по подготовке и введению в машину этих вариантов чрезвычайно велик и для большинства известных игр практически невыполним. Однако этот принцип может использоваться для некоторых игр в комбинации с другими принципами на начальной стадии игры, т. е. когда число возможных вариантов сравнительно невелико.

**2. Машины, реализующие строго определенные правила игры.** В основе некоторых игр лежит строгая математическая теория, которая позволяет при помощи определенных формул в любом положении рассчитать необходимый оптимальный ход. Таким образом, вычислительная машина, применяемая для подобных игр, должна обычным способом производить вычисления по программе, составленной для соответствующих расчетных формул.

**3. Машины, использующие общие принципы оценки положений.** В большинстве игр невозможно точно указать в каждом положении единственный оптимальный ход, который бы обеспечивал более успешное продвижение к выигрышу, чем любой другой ход. Однако существуют общие принципы для оценки положений, возникающих в ходе игры, позволяющие сравнивать различные возможные ходы. Эти принципы в своей основе являются общими для всех игроков, правильно играющих в данную игру (шашки, шахматы, карты и др.), но в зависимости от квалификации и индивидуальных особенностей игроков могут быть различные подходы к истолкованию и применению этих общих принципов.

В машину при помощи соответствующей программы могут быть заложены необходимые критерии для применения этих общих принципов в определенных положениях. Ясно, что совершенство игры по такому методу в основном зависит от совершенства исходных общих принципов игры. Это в одинаковой мере относится как **к** «играющей» машине, так и к людям.

**4. Машины, накапливающие «опыт»** . В машину задаются правила игры, элементарная тактика игры и методы улучшения этой тактики в результате опыта, т. е. методы «обучения» машины.

Могут быть использованы различные методы обучения машины; к числу таких методов относятся:

а) пробные ходы с запоминанием благоприятных и устранением неблагоприятных возможностей;

б) подражание более успешно играющему противнику;

в) обучение машины в результате одобрения или неодобрения действий машины с указанием характера допущенных ошибок со стороны внешнего наблюдателя — «учителя»;

г) анализ самой машиной после окончания партии или в процессе игры допущенных ошибок и общего хода игры с целью выработки общих принципов тактики игры.

Машины, играющие по принципу «словаря», и машины, вычисляющие оптимальные ходы по определенным постоянным расчетным формулам, не представляют особого интереса с точки зрения их логических возможностей.

Программы, обеспечивающие поиск и выборку нужных ответов из словаря, аналогичны программам выборки

необходимых значений функций из таблиц при математических вычислениях.

Программы для вычисления оптимальных ходов по заданным расчетным формулам также не отличаются от обычных вычислительных программ.

Значительный интерес представляют машины, играющие по принципу общей оценки положения, и обучаемые машины, которые мы рассмотрим более подробно.

Характерным примером машин, выбирающих оптимальные ходы на основе общих принципов оценки обстановки, являются машины, играющие в шахматы.

**5. Машины, играющие в шахматы.** Шахматная игра представляет собой игру, подчиненную строго определенным правилам, каждое из которых сводится в конечном счете к последовательности альтернативных выборов (выборов одного из двух взаимоисключающих ответов «да» или «нет») и вполне может быть реализовано на электронной цифровой машине.

Составление программы, обеспечивающей задание в машину полной информации о шахматной позиции (расположение фигур, наличие свободных клеток, очередность хода, взятие пешек на проходе и др.) и дающей возможность выбирать ходы, допускаемые правилами игры, не содержит принципиальных трудностей. Проблема заключается в том, чтобы построить такую программу, которая дает возможность машине выбирать относительно хорошие ходы. Следует заметить, что не ставится вопрос о возможности осуществления машиной оптимальной игры, так как даже человеческий разум не достигает этого.

Необходимо, чтобы машина могла оказать серьезное сопротивление игроку определенной шахматной категории. Сложность шахматной игры заключается в чрезвычайно большом количестве возможных вариантов продолжения партий и необходимости учитывать при анализе обстановки большое количество факторов. Сущность шахматной стратегии сводится к тому, чтобы, продумывая на несколько ходов вперед возможные продолжения игры, оценить возникающие позиции и выбрать такие ходы, которые в будущем, по мнению игрока, должны привести к выигрышу партии. Своеобразие игры отдельных шахматистов заключается в различном подходе к оценке позиций и в различной способности обдумывать продолжение игры на большее или меньшее количество ходов вперед.

Программа работы машины также предусматривает возможность оценки позиций по определенным, заранее заданным в машину критериям и последовательную проверку всех возможных своих ходов и ответов противника на несколько ходов вперед. Чем большее количество ходов вперед будет проверять машина, тем совершеннее будет ее игра. Однако ясно, что проверка на каждый лишний ход вперед приводит к резкому увеличению количества проверяемых вариантов и, помимо усложнения программы, приводит к значительному увеличению времени нахождения очередного хода.

Критерии для оценки позиций машиной должны быть заданы в виде определенного математического алгорифма, соответствующего некоторой стратегии игры.

Обычно оценки шахматной позиции производятся по элементам (соотношение по численности и качеству фигур, изолированное или защищенное положение королей, занятость центра, наличие проходных пешек, занятость открытых линий и т. п.), причем ясно, что отдельные элементы далеко не равноценны между собой. Кроме того, относительная ценность того или иного элемента (или, как говорят, преимущества) даже в одной и той же позиции с точки зрения различных игроков может быть различной. Однако представляется возможным оценить в среднем (т. е. по мнению большинства игроков) отдельные элементы позиции определенным количеством очков. Наиболее просто это сделать, очевидно, для оценки качества фигур, и в шахматной игре, вообще говоря, установился определенный эквивалент ценности отдельных фигур. Для других элементов позиции, в том числе элементов, характеризующих степень развития фигур, также представляется возможным установить определенное количество очков, оценивающих этот элемент. Так как один и тот же элемент в различных положениях (например, в дебюте и в эндшпиле) может иметь различную ценность, то может быть введена дополнительная зависимость количества очков, соответствующих элементу, от типа позиции.

Например, может быть установлено такое распределение очков для отдельных элементов: Король (КР) — 200; Ферзь (Ф) — 9; Ладья (Л) —5; Слон (С) — 3; Конь (К) — 3; Пешка (П) — 1; отсталая пешка ($П_1$) — 0,5; изолированная пешка ($П_2$) — 0,4; сдвоенная пешка ($П_3$) — 0,3; подвижность (М) — 0,1.

Подвижность М представляет собой чисто позиционное преимущество и может определяться количеством свободных полей, имеющихся в распоряжении той или другой стороны, для передвижения наиболее сильных фигур.

Таким же образом могут быть введены определенные оценки в очках и для других элементов. Вопрос рационального расчленения шахматных позиций на элементы и определения соответствующих оценок для этих элементов, по-видимому, еще как следует не разработан и представляет собой задачу шахматной теории. Для оценки машинной позиции в целом вводится некоторая функция ценности позиции $\varphi(P)$, которая позволяет на основе указанных оценок отдельных элементов оценивать качество всей позиции. Эта функция характеризует собой подход «игрока» (машины) к оценке позиций, т. е. определяет по существу своеобразие игры, стратегию игры.

В наиболее простом случае эта функция может представлять собой сумму очков, соответствующих отдельным элементам позиции, и иметь следующий вид:

$$\varphi(P) = 200(Кр — Кр') + 9(Ф — Ф') + 5(Л — Л') + 3(К — К' + С — С') + (П — П') + 0{,}5(П_1 — П'_i) + \\ + 0{,}4\,(П_2 — П'_2 + 0{,}3\,(П_3 — П'_3 + 0{,}1\,(М — М') + \ldots,$$

где буквы Кр, Ф, Л, К, С, П, $П_1$, $П_2$, $П_3$ обозначают количества соответствующих фигур у белых, а те же буквы со штрихами — количества таких же фигур у черных; М и М' — определенные выше характеристики подвижности. Эта функция $\varphi(P)$ позволяет приближенно оценивать качество позиций. Если считать, что машина играет белыми, то увеличение этой функции будет свидетельствовать об улучшении позиции и успешности игры.

В позиции $P$ машина подсчитывает на $p$ ходов вперед все возможные варианты своих ходов и ответов противника $\alpha_1$, $\beta_1$, $\alpha_2$, $\beta_2$, ..., $\alpha_p$, $\beta_p$, (где $\alpha_i$ — ходы машины, $\beta_i$ — ходы противника) и из всех вариантов выбирает наиболее выгодный вариант $\bar{\alpha}_1, \bar{\beta}_1, \bar{\alpha}_2, \bar{\beta}_2, ... \bar{\alpha}_p, \bar{\beta}_p$, т. е. такой вариант, для которого функция ценности $\varphi$ ($P\bar{\alpha}_1, \bar{\beta}_1, \bar{\alpha}_2, \bar{\beta}_2, ... \bar{\alpha}_p, \bar{\beta}_p$), будет иметь максимальное значение

$$\varphi(P_{\bar{\alpha}_1,\bar{\beta}_1,...\bar{\alpha}_p,\bar{\beta}_p}) = \max(\min(\max(\min(...(\max(\min \varphi(P_{\alpha_1,\beta_1,...\alpha_p,\beta_p}))))))),$$

$$\alpha_1 \quad \beta_1 \quad \alpha_2 \quad \beta_2 \qquad \alpha_p \quad \beta_p$$

где $P_{\alpha_1,\beta_1,\alpha_2,\beta_2,...,\alpha_p,\beta_p}$ — позиция, получающаяся из $P$ после ходов $\alpha_1, \beta_1, ..., \alpha_p, \beta_p$. Когда найдена такая последовательность ходов $\bar{\alpha}_1, \bar{\beta}_1, ..., \bar{\alpha}_p, \bar{\beta}_p$, , то ход $\alpha_1$ является искомым, наилучшим при данной «стратегии» игры ходом.

Процесс выбора «оптимального хода» осуществляется путем последовательного определения значения функции ценности получающейся позиции для каждого хода. Может получиться так, что, просчитывая на несколько ходов вперед ($p$ порядка 3), машина выберет такой вариант продолжения, что первый или второй ход приведет к уменьшению функции ценности позиции (необходимость пожертвовать фигуру и др.), но в результате всех $p$ ходов функция ценности позиции должна быть максимальной для машины.

Если в течение этих $p$ ходов будет иметься возможность поставить мат, то машина его обязательно сделает; также если будет возможность самой избежать мата, то машина его избежит. Машина не «прозевает» свою фигуру и не пропустит возможность взять фигуру противника. В общем такая машина могла бы играть в шахматы так же, как играет в эту игру большинство людей, но, безусловно, она проиграла бы игроку высокого класса, способному продумывать продолжение игры на большее, чем р, количество ходов.

Описанная выше «стратегия» машинной игры в шахматы может быть названа элементарной стратегией, так как она предполагает непосредственный просчет и проверку всех возможных вариантов ходов (в том числе и явно бессмысленных с точки зрения играющего человека) и неизменность этой стратегии в ходе игры. Эта стратегия может быть усовершенствована путем введения дополнительной проверки, получающейся после $p$-го хода позиции на устойчивость. Под устойчивой позицией подразумевается позиция, в которой противник не может взять менее ценной фигурой более ценную, король не находится под шахом, и т. д. Если позиция $P$, выбранная по максимуму функции ценности $\varphi(P)$, является в то же время неустойчивой, то машина просчитывает новые варианты до получения устойчивой позиции.

Далее, усовершенствованная стратегия машинной игры предусматривает исключение из рассмотрения явно бессмысленных ходов (которые человек, играя в шахматы, отбрасывает из общих соображений). Для того чтобы отбросить большое количество возможных вариантов, связанных с такими ходами, в программе предусматриваются оценка возможных ответных ходов противника по определенной шкале «силы» ходов и рассмотрение только наиболее сильных ответов противника. Шкала силы устанавливается приблизительно такой: шах, взятие фигуры, нападение на фигуру, развитие, защита, рокировка и т. д.

Таким образом, видно, что изложенные принципы построения программы для машинной игры в шахматы основываются на четком задании совокупности однозначных правил, определяющих алгорифм выбора «оптимального» хода на основе общей оценки обстановки. Этот алгорифм, будучи реализован в виде программы работы электронной цифровой машины, обеспечит возможность машине играть достаточно хорошо в шахматы по неизменной стратегии, определяемой установленным видом функции ценности позиции.

Для того чтобы машина могла усовершенствовать в процессе игры свою «стратегию», менять «стиль» игры, необходимо обеспечить в программе возможность изменения по определенным критериям вида функции, оценивающей позицию, с накоплением «опыта» игры машины, т. е. необходимо придать машине некоторую способность к самоорганизации или к «обучению».

Следует отметить, что разработка вопросов применения электронных цифровых машин для игры в шахматы, шашки и другие игры, исследование вопросов машинной тактики и стратегии игры, в том числе создание программ самосовершенствующейся стратегии игры, имеют большое экономическое и военное значение. Строя такие автоматы и исследуя их работу, можно изучить законы построения целого класса автоматических устройств, которые могут быть применены в промышленности и в военном деле.

**6. Имитация условного рефлекса.** Имитация процессов выработки условных рефлексов у животных может быть осуществлена на обычной электронной цифровой программно-управляемой машине при помощи сравнительно простой программы.

Сущность этой программы сводится к следующему.

В машину оператором может быть задан стимул — раздражитель различной интенсивности в виде одного

числа из заданной группы целых чисел. В ответ на этот стимул машина может реагировать рядом различных способов, обозначенных целыми числами, выдаваемыми машиной. Получив ответ машины, оператор может выразить свое одобрение или неодобрение в зависимости от того, совпадает или нет напечатанная цифра с требуемой. Одобрение или неодобрение выражается путем ввода в определенную ячейку памяти третьего целого числа, величина которого определяет степень одобрения или неодобрения.

В начале работы машина выдает на стимулы случайные ответы. Указание одобрения увеличивает шансы на выдачу в будущем той цифры, которая была выдана непосредственно перед одобрением, указание неодобрения уменьшает эти шансы.

После ряда повторений машина начинает уверенно давать нужную цифру даже при незначительном стимуле.

Если же при ряде правильных ответов машина не будет получать одобрения, а, наоборот, будет получать неодобрение, то она может сбиться, т. е. забыть выработанный «условный рефлекс», и опять начнет печатать беспорядочные ответы.

Идея построения одной из таких программ коротко сводится **к** следующему.

В машине имеется восемь чисел $a_0, a_x, a_2, ..., a_7,$ расположенных в определенных ячейках памяти $k_0, k_1, k_2, ..., k_7.$ Каждому номеру ячейки $k_0, ..., k_7$ поставлена в соответствие одна из восьми цифр 0, 1, 2,…,7, характеризующих собой вид реакции машины на стимул. Любая из этих цифр (0, 1, 2, ..., 7) может быть напечатана машиной в ответ на стимул. Числа $a_0, a_1, a_2, ..., a_7$ в процессе работы машины изменяются.

Когда машина получает стимул, она выбирает из этих чисел наибольшее число $a_i$ прибавляет к нему число, соответствующее силе стимула, и если сумма превосходит определенную минимальную величину, то печатает в качестве ответа цифру $i$, соответствующую номеру ячейки $k_i,$ в которой находится выбранное наибольшее значение $a_i.$ Если сумма меньше установленного предела, то машина печатает некоторое число $x,$ означающее, что машина не знает, что нужно печатать. После того как оператор выразит одобрение или неодобрение по поводу напечатанной цифры, машина увеличивает или уменьшает число $a_i$ на величину, зависящую от силы одобрения или неодобрения. Затем к $a_i$ прибавляется небольшое случайное положительное или отрицательное число, которое получается отдельным вычислением (или генератором случайных чисел), не имеющим отношения к основной программе.

Случайное число прибавляется для создания нерегулярности в поведении машины.

Для того чтобы машина могла «забывать» рефлекс, если нет одобрения, предусматривается периодическое умножение всех восьми чисел от $a_0$ до $a_7$ на постоянное число, меньшее единицы, что обеспечивает стремление всех чисел $a_0, ..., a_7$ к нулю при отсутствии одобрения. Сущность действия одобрения заключается в увеличении вероятности получения той цифры в будущем, для которой было выражено одобрение, и при многократном повторении это создает впечатление выработки условного рефлекса.

Описанная программа является простой и весьма ограниченной. По этой программе можно «обучить» машину только тому, что заранее предусмотрено для нее составителем программы. В этой программе точно расписаны все действия машин; можно усложнять вид «рефлексов», но подобные программы, точно описывающие поведение машины для каждого случая, не могут имитировать обучение машины любым рефлексам, которые будут предложены уже после составления и введения в машину программы. Для получения такой возможности необходима обобщенная программа обучения, которая позволяла бы обучать машину всему, что будет задано, независимо от планов составителя программы.

Такая программа должна быть построена на базе существенно новых идей и должна, по-видимому, обеспечивать преобразование и расширение самой программы в процессе обучения.

## ЛИТЕРАТУРА


Автоматы, Сборник статей, под ред. Шеннона, ИЛ, 1956.

Быстродействующая вычислительная машина М-2, под ред. И. С. Брука, Гостехиздат, 1957.

Быстродействующие вычислительные машины, пер. с англ., под ред. Д. Ю. Панова, ИЛ, 1952.

Вопросы теории математических машин, Сборник, вып. 1, под ред. Ю. Я. Базилевского, Физматгиз, 1958.

Вычислительная техника и ее применения, Сборник под ред. акад. С. А. Лебедева, Госэнергоиздат, 1959.

Вычислительные машины СЕАК и ДИСЕАК, под ред. В. М. Тарасевича, ИЛ, 1958.

Машинный перевод, Сборник статей под ред. П. С. Кузнецова, ИЛ, 1957.

Проблемы кибернетики, Сборник, вып. 1, под ред. А. А. Ляпунова, Физматгиз, 1958.

Синтез электронных вычислительных и управляющих схем, пер. с англ., под ред. В. И. Шестакова, ИЛ, 1954.

Система стандартных подпрограмм, Сборник под ред. М. Р. Шура-Бура, Физматгиз, 1958.

Блекуэлл Д. и Гиршик М. А., Теория игр и статистических решений, ИЛ, 1958.

Бондаренко В. Н., Плотников И. Т., Полозов П. П., Программирование задач для машины «Урал», Изд. Артиллерийской инж. академии им. Дзержинского, 1957.

Брук И. С, Об управляющих машинах, Природа, № 5, 1955.

Быховский М. Л., Основы электронных математических машин дискретного счета, Успехи матем. наук, т. IV, вып. 3 (31), 1949.

Винер Н., Кибернетика или управление и связь в животном и машине, Сов. Радио, 1958.

Гаврилов М. А., Теория релейно-контактных схем, Изд-во АН СССР, 1950.

Гильберт Д., Аккерман В., Основы теоретической логики, ИЛ, 1947.

Ершов А. П., Программирующая программа для быстродействующей электронной счетной машины, Изд-во АН СССР, 1958.

Каган Б. М., Тер-Микаэлян Т. М., Решение инженерных задач на автоматических цифровых вычислительных машинах, Госэнергоиздат, 1958.

Карцев М. А., Арифметические устройства электронных цифровых машин, Физматгиз, 1958.

Китов А. И., Электронные цифровые машины, Сов. Радио, 1956.

Китов А. И., Криницкий Н. А., Электронные вычислительные машины, Изд-во АН СССР, 1958.

Китов А. И., Криницкий Ні А., Комолов П. Н., Элементы программирования (для электронных цифровых машин), Изд. Артиллерийской инж. академии им. Дзержинского, 1956.

Кольман Э., Что такое кибернетика? Вопросы философии, № 4, 1955.

Кузнецов П. С, Ляпунов А. А., Реформатский А. А., Основные проблемы машинного перевода, Вопросы языкознания, № 5, 1958.

Кулагина О. С, Мельчук И. А., Машинный перевод с французского на русский, Вопросы языкознания, № 5, 1956.

Лебедев С. А., Электронные вычислительные машины, Изд-во АН СССР, 1956.

Ляпунов А. А., Кулагина О. С, Использование вычислительных машин для перевода с одного языка на другой, Природа, № 8, 1955.

Ляпунов А. А. и Шестопал Г. А., Об алгоритмическом описании процессов управления, Сб. «Математическое просвещение», вып. 2, 1957.

Майоров Ф. В., Электронные цифровые вычислительные машины, Природа, № 11, 1954.

Моисеев В., Автоматические вычислительные машины, Трансжелдор-издат, 1957.

Морз Ф., Кимбелл Д., Методы исследования операций, пер. с англ., Сов. Радио, 1956.

Панов Д. Ю., Автоматический перевод, Изд-во АН СССР, 1956.

Полетаев И. Л., Сигнал, Сов. Радио, 1957.

Ричардс Р. К., Арифметические операции на цифровых вычислительных машинах, ИЛ, 1957.

Соболев С. Л., Китов А. И., Ляпунов А. А., Основные черты кибернетики, Вопросы философии, № 4, 1955.

Соболев С. Л., Ляпунов А. А., Кибернетика и естествознание, Вопросы философии, № 5, 1958.

Трахтенброт Б. А., Алгорифмы и машинное решение задач, Гостех-издат, 1957.

Уилкс М., Уилер Д., Гилл С, Составление программ для электронных счетных машин, пер. с англ., под ред. Д. Ю. Панова, ИЛ, 1950.

Electronic Device Simulates Processes of Human Brain, Aviation Week, July 7, 1958.

Electronic translation, J. Franklin Inst., 1954, v. 257, № 3, стр. 257—260.

Farley B. and Clark W., Simulation of self-organising systems by digital computer, Transactions IRE, 1954, PGTT-4, IX, стр. 76.

Klass P., D gital computers take to the Air, Aviation week, 1954, v. 60, № 19, стр. 62—68.

Schliebs G., Cber die Grundzuge eines Programms fur eine Schachspie-lende Rechenmaschine, Funk und Ton, 1953, B. 7, № 5, стр. 257—265.

Shannon C, Programming a computer for playing chess, Phil. Mag., 1950, 7 ser., v. 41, № 1, стр. 256.

Shannon C, Moore E., Machine aid for switching eircuit design, Proc. IRE, 1953, v. 41, № 10, стр. 1348—1351.