ЭЛЕМЕНТЫ ПРОГРАММИРОВАНИЯ

11. СОСТАВ ОПЕРАЦИЙ И СИСТЕМЫ КОМАНД В МАШИНАХ УНИВЕРСАЛЬНОГО НАЗНАЧЕНИЯ

1. Арифметические и логические операции

Любая задача решается на автоматической цифровой машине путем выполнения машиной заданной последовательности арифметических и логических операций, составляющих программу решения задачи.

Состав операций, которые должны выполняться машиной, определяется при проектировании машины на основе рассмотрения типовых вычислительных процессов решения.

В электронных счетных машинах обычно предусматривается выполнение основных арифметических операций:

— алгебраическое сложение двух чисел;
— алгебраическое вычитание;
— умножение двух чисел с учетом их знаков;
— деление чисел с учетом знаков.

Часто операция деления не является элементарной, а выполняется при помощи других операций.

Как было видно при рассмотрении арифметических основ построения машин (глава II), все арифметические операции в автоматических машинах сводятся, в конечном счете, к выполнению операции сложения.

Так, операция вычитания выполняется путем сложения с дополнительным или обратным кодом числа, операция умножения сводится к многократному сложению; операция прямого деления сводится к последовательному вычитанию делителя из делимого со сдвигом, т. е. также может быть выполнена при помощи сложения в обратном или дополнительном коде. Таким образом, любой вычислительный процесс, в принципе, можно было бы построить, используя только операцию сложения, однако для упрощения программирования задач и обеспечения большей гибкости в работе машины вводятся более сложные операции, выполняемые машиной по одной команде.

В арифметическом устройстве имеется обычно несколько отдельных устройств, каждое из которых выполняет либо одну операцию, либо несколько близких по своей структуре операций. Упрощение состава операций приводит к упрощению машины, но усложняет процесс программирования задач и увеличивает время решения. С другой стороны, увеличение состава операций позволяет упростить программирование задач и уменьшить время решения, но приводит обычно к существенному усложнению машины. Задача синтеза оптимальной системы операций является одной из важных задач, возникающих при разработке машины. Эта задача решается на основе совместного анализа различных вариантов программ для наиболее важных и типовых задач, составленных при разных системах операций и возможностей конструктивного осуществления необходимых устройств.

Существенное расширение математических возможностей машин и облегчение программирования достигается за счет применения стандартных подпрограмм, обеспечивающих выполнение машиной сложных действий без расчленения этих действий при программировании задачи на элементарные операции. Стандартные подпрограммы составляются для наиболее часто встречающихся типовых вычислений и могут рассматриваться как единые (сложные) операции.

Для выполнения операции по стандартной подпрограмме достаточно иметь в основной программе отсылку к соответствующей стандартной подпрограмме. После выполнения стандартной подпрограммы автоматически возобновляется последовательное выполнение команд основной программы с того места, в котором было приостановлено выполнение этой программы в связи с переходом к стандартной подпрограмме.

В некоторых машинах стандартные подпрограммы хранятся постоянно в машине в специальном запоминающем устройстве. В других машинах стандартные подпрограммы во время решения задач находятся в общем запоминающем устройстве, а в промежутках между решениями задач хранятся вне машины (на перфокартах, перфолентах или магнитных лентах) и вводятся в машину непосредственно перед решением задач вместе с основной программой.

Следует отметить, что нет резкой разницы между операциями, входящими в состав основных, и операциями, выполняемыми по стандартным подпрограммам. В одних машинах, например, операции извлечения квадратного корня, деления и даже вычисление $\sin x$ входят в состав основных операций, в других машинах эти операции выполняются по стандартным подпрограммам. Хотя и в том и в другом случае для выполнения перечисленных операций требуется одна команда в основной программе, однако на выполнение этих операций при помощи стандартных подпрограмм затрачивается, как правило, значительно больше машинного времени, чем в тех случаях, когда эти операции являются основными и выполняются при помощи специальных устройств.

Для обеспечения автоматической работы машины в состав выполняемых операций, помимо арифметических, вводятся логические операции. Можно указать четыре основных логических операции, которые являются необходимыми для универсальных автоматических машин и из которых могут быть сконструированы другие более сложные операции.

1. П о р а з р я д н о е л о г и ч е с к о е у м н о ж е н и е . Эта операция заключается в попарном перемножении одноименных разрядов заданных чисел по правилу

$$0 \cdot 0 = 0, \quad 0 \cdot 1 = 1 \cdot 0 = 0, \quad 1 \cdot 1 = 1.$$

Поразрядное логическое умножение применяется для выделения из данного числа какой-нибудь определенной части — определенной группы двоичных разрядов. Например, для выделения разрядов мантиссы или порядка, выделения знака числа или знака порядка и т. д. При операциях над командами может потребоваться выделить какой-нибудь адрес или код операции.

Для выделения определенной группы разрядов данного числа должно быть подобрано в качестве второго сомножителя специальное число, имеющее единицы в тех разрядах, которые соответствуют выделяемым разрядам заданного числа, и нули в остальных разрядах. Например, если требуется выделить первые три разряда в некотором шестиразрядном числе, то второй сомножитель должен иметь вид:

111000.

2. П о р а з р я д н о е л о г и ч е с к о е с л о ж е н и е . Эта операция выполняется по правилу:

$$0+0=0,\ 0+1=1,\ 1+0=1,\ 1+1=1.$$

Логическое поразрядное сложение используется при формировании чисел из частей заданных чисел. Например, может оказаться необходимым сформировать новое число таким образом, чтобы в нем мантисса была равна мантиссе одного числа, а порядок был бы равен порядку другого числа. Часто бывает необходимо при программировании образовывать новые команды из отдельных частей старых команд.

При формировании нового числа из частей заданных чисел нужно сначала путем поразрядного логического умножения выделить необходимые части чисел, а затем путем поразрядного логического сложения сформировать из этих частей новое число.

3. С д в и г ч и с л а н а о п р е д е л е н н о е к о л и ч е с т в о р а з р я д о в в п р а в о и л и в л е в о . Направление сдвига числа и количество разрядов, на которое производится сдвиг, определяются некоторым другим числом и могут быть известны заранее или вычислены в процессе решения задачи. Операция сдвига в машинах с плавающей запятой выполняется таким образом, что сдвигаются одновременно все разряды числа (мантисса, порядок, разряды знаков), т. е. после сдвига разряды мантиссы могут занимать разряды порядка и т. д., при этом цифры числа, выходящие при сдвиге за пределы разрядной сетки, теряются. Например, число

$$+0{,}1010011\cdot 10^{+1101} = |0|\ 1010011\ |0|\ 1101|$$

после сдвига на три разряда вправо будет иметь вид

$$|0|0001010|0|1101|$$

4. С р а в н е н и е . Эта операция осуществляется путем поразрядного выполнения операции, называемой сложением по модулю два, по правилу.

$$0+0=0,\quad 0+1=1,\quad 1+0=1,\quad 1+1=0.$$

В связи с этим в тех разрядах результата, которым соответствуют несовпадающие разряды сравниваемых чисел, будут получаться единицы, а совпадающим разрядам в исходных числах будут соответствовать нули. Таким образом, два числа будут считаться равными только в том случае, если совпадают все разряды в изображении этих чисел. Как нетрудно видеть, приведенное правило сложения по модулю два реализует логическую операцию отрицания равнозначности.

Кроме перечисленных арифметических и логических операций в автоматических машинах применяются и некоторые другие операции. Типовой состав операций и команд электронной цифровой машины универсального назначения приводится ниже.

2. Команды

На основе выбранного состава арифметических и логических операций разрабатывается система команд в машине.

Каждой операции присваивается определенный номер — код операции. Для записи операций в процессе составления программ применяются помимо номеров операций также некоторые другие символы: начальные буквы названий операций (например, сложение С, вычитание В и т.д.), знаки арифметических действий (сложение +, умножение × и т.д.).

Все команды могут быть разделены на два основных типа:

а) команды арифметических и логических операций, выполнение которых связано с какими-либо преобразованиями чисел (в том числе и чисел, изображающих команды);

б) команды управления, служащие для управления работой машины (изменение порядка выполнения программы, остановка машины, изменение режима работы и др.).

Команды арифметических и логических операций должны содержать в себе все необходимые сведения для выполнения требуемой операции: во-первых, указание, какую операцию необходимо выполнить — код операции, и, во-вторых, указания, откуда взять исходные числа и куда направить результат — адреса исходных данных и результата. В качестве адресов в команде указываются порядковые номера ячеек внутреннего запоминающего устройства, в которых находятся числа.

В командах, связанных с обращениями к внешним накопителям, в качестве адресов указываются номера зон накопителя, в которых находятся требуемые группы чисел. Напомним, что во внешних накопителях запись и считывание данных производится не отдельными числами, а группами чисел, которые располагаются в определенных зонах.

В различных машинах в зависимости от схемы машины и строения арифметических устройств применяются одно-, двух-, трех- и четырехадресные команды. В последнее время начали применяться и так называемые полутораадресные команды.

Как уже упоминалось раньше (§ 2), по конструкции машины могут быть либо машинами с естественным выполнением последовательности команд, либо машинами с принудительным выполнением последовательности команд.

Рассмотрим структуру команд при естественном порядке выполнения команд.

Трехадресные команды, помимо указания операции, которую требуется выполнить, содержат два адреса исходных чисел и адрес результата операции.

В случае трехадресных команд для выполнения арифметического действия, такого как $a\ \theta\ b = c$ (где θ означает какую-либо элементарную операцию), требуется, как правило, одна команда, т. е. в этом случае арифметическое действие выполняется за один такт работы машины.

Трехадресные команды наиболее полно отвечают логическому построению арифметических операций, в каждой из которых, как правило, участвуют два числа, являющиеся исходными данными, и третье число —

результат операции.*

Применение одноадресных команд предполагает наличие в арифметическом устройстве машины одного или нескольких регистров (быстродействующих запоминающих устройств на одно число), используемых при последовательных операциях для хранения промежуточных результатов. При одноадресных командах арифметические действия расчленяются на отдельные этапы таким образом, что каждая команда касается только одной входящей в вычисления величины, и для выполнения арифметической операции требуется несколько команд.

Например, команда для умножения двух чисел, находящихся в ячейках с номерами 0110 и 0120, с посылкой результата в ячейку с номером 0130, в случае трехадресных команд будет иметь вид

М 0110 0120 0130

и будет выполнена за один такт работы машины.

В случае одноадресных команд эта операция потребует наличия трех команд следующего содержания.

1. Взять число из ячейки О11О и направить в регистр «Р».
2. Взять число из ячейки 0120 и умножить его на число, находящееся в регистре «Р». При этом результат останется в регистре «Р».
3. Взять число из регистра «Р» и направить его в ячейку 0130.

В каждой из этих команд участвует только один адрес.

Эти три команды будут выполнены за три такта работы машины. Однако в тех случаях, когда необходимо выполнить такой ряд операций, в котором результат каждой предшествующей операции используется как исходное число для последующей операции, применение одноадресных команд не приводит к заметному увеличению количества команд и тактов работы машины по сравнению с трехадресными. Например, суммирование подряд всех чисел, находящихся в ячейках от 201 до 300, с записью результата в какую-нибудь *k*-тую ячейку потребует всего 102 одноадресных команды против 100 трехадресных команд. При программировании задач весьма часто можно так построить вычисления, чтобы результаты предыдущих операций непосредственно использовались в последующих операциях. Это обстоятельство вместе с существенным упрощением одноадресных машин в конструктивном отношении привело к тому, что во многих современных построенных и разрабатывающихся машинах, применяется одноадресная система команд (ИБМ-701, УНИВАК, ОРДВАК, ЭДСАК, «Урал»).

В двухадресных командах указываются адреса двух исходных чисел. Адрес результата совпадает с адресом одного из исходных чисел, т. е. результат операции остается в той ячейке памяти, в которой находилось одно из исходных чисел, которое таким образом, стирается. Такие команды применялись в тех машинах (ЭНИАК, МАРК-1), в которых было небольшое количество ячеек памяти, обладавших свойством суммирования. (Если в ячейке находится число *а* и туда посылается число *в,* то в этой ячейке будет записана сумма *а* + *в*).

В двухадресных командах могут указываться также адрес одного исходного числа и адрес результата, однако этот случай используется сравнительно редко, чаще указываются адреса двух исходных чисел.

В настоящее время почти во всех машинах память построена по принципу замещения, т. е. вновь посылаемое в какую-нибудь ячейку число стирает число, ранее находившееся в этой ячейке, и записывается на его место. Поэтому при использовании двухадресных команд, содержащих адреса исходных чисел, результаты операций получаются всегда в специально отведенных постоянных местах, а не на местах исходных чисел. Примером машины, использующей двухадресную систему команд, является ИРА-1103 (США).

Модификацией двухадресной системы команд является полутораадресная система команд. В этой системе в каждой команде также указываются два адреса, только один адрес указывается полным числом разрядов и может обозначать номер любой ячейки памяти, а второй адрес указывается половинным числом разрядов и может, следовательно, обозначать номера ячеек памяти только в некоторой определенной ее области. Последнее ограничение не является существенным, так как в большинстве случаев один из адресов относится к рабочим ячейкам, которые могут располагаться в определенной ограниченной области памяти. Вместе с тем полутораадресная система команд, сохраняя преимущества двухадресных команд, позволяет существенно уменьшить потребное для представления команды количество разрядов.

Четырехадресные команды используются только при принудительном порядке выполнения последовательности команд и в отличие от трехадресных команд содержат еще четвертый адрес, в котором указывается номер ячейки памяти, содержащей следующую команду. При четырехадресных командах нет необходимости записывать команды в той последовательности, в какой они будут выполняться машиной. Примером машин с четырехадресными командами являются машины ЭДВАК и Райтеон (США).

Принудительный порядок выполнения команд может быть осуществлен также с трехадресными и с двухадресными командами, но, конечно, не с одноадресными. При этом в двухадресных командах один адрес является адресом исходного числа, а второй — адресом следующей команды. По характеру выполняемых операций такая двухадресная команда аналогична рассмотренной выше одноадресной команде.

При трехадресных командах и принудительном порядке их выполнения один из адресов указывает адрес следующей команды, а два других используются как адреса двухадресных команд с естественным порядком выполнения.

Принудительный способ выполнения команд, вообще говоря, усложняет процесс программирования, так как при этом необходимо дополнительно определять порядок размещения команд в памяти машины. Однако этот способ является выгодным для машин последовательного действия, имеющих запоминающие устройства с последовательной выборкой чисел (магнитные барабаны, акустические линии задержки), так как позволяет существенно сократить время решения задач путем применения так называемого «оптимального программирования».

Сущность оптимального программирования заключается в том, что команды в памяти записываются таким образом, чтобы очередная команда находилась в ячейке памяти, следующей за той ячейкой, в которой должен быть записан результат предыдущей операции. При этом выборка очередной команды производится сразу же

* Трехадресная система команд используется в советских электронных цифровых машинах универсального назначения БЭСМ и „Стрела"

после записи результата предыдущей операции и не связана с потерей времени на ожидание поворота барабана или прохождение всех импульсов в акустических линиях задержки.

Мы будем рассматривать в дальнейшем, в основном, машину, в которой принят естественный порядок выполнения и трехадресная система команд.

Для определенности будем рассматривать разрядную сетку машины «Стрела», в которой для представления чисел и команд имеется 43 двоичных разряда. Эти разряды пронумерованы слева направо с нулевого по сорок второй.

При изображении двоичных чисел в нормальной форме нулевой разряд служит для изображения знака числа, 35 разрядов с 1-го по 35-й изображают мантиссу, 36-й разряд изображает знак порядка и 6 разрядов с 37-го по 42-й изображают порядок числа. 112

При таком распределении двоичных разрядов изображаемые числа будут соответствовать приблизительно 10-значным десятичным числам и иметь диапазон изменения от 10^{-19} до 10^{+19}. Для наглядности в табл. IV. 1 показана так называемая разрядная сетка машины, т. е. распределение разрядов между частями кода числа и кода команды.

Таблица IV. 1

Номера разрядов	0	1	2	.....	10	11	12	13	...	22	23	24	25	....	34	35	36	37	...	42
Части числа	±	мантисса															±	порядок		
Части команды	1 адрес								2 адрес					3 адрес			В.Р.	код операции		

Для изображения трехадресных команд используются те же 43 двоичных разряда, что и для изображения чисел. При этом для изображения каждого адреса отводится по 12 двоичных разрядов, что позволяет получить до 4096 различных адресов: от нулевого по 4095-й. ($2^{12}=4096$). Первые 12 разрядов, с нулевого по одиннадцатый, представляют адрес первого исходного числа, разряды с 12 по 23 — адрес второго исходного числа, разряды с 24 по 35 — адрес результата операции, а 36-й разряд является вспомогательным и служит для изображения специального контрольного знака. Шесть разрядов с 37 по 42 отводятся для изображения кода операции.

Для записи команд в процессе составления программы применяются обычно либо восьмеричная, либо шестнадцатиричная системы счисления, которые позволяют весьма просто производить перевод программы в двоичную систему счисления в процессе ввода программы на внешних устройствах машины.

Рассмотрим запись команд в восьмеричной системе счисления. Так как три двоичных разряда могут быть изображены одним восьмеричным разрядом, то для изображения 12-разрядного двоичного числа, обозначающего один адрес, достаточно иметь 4 восьмеричных разряда. Каждая команда в восьмеричной системе счисления может быть изображена в виде пятнадцатизначного числа.* При этом будет иметь место следующее распределение разрядов между отдельными частями команды: первые четыре разряда (слева) занимает адрес первого исходного числа; следующие четыре разряда занимает адрес второго исходного числа; разряды с 9 по 12 служат для изображения адреса результата операции; 13-й разряд служит для обозначения специального контрольного знака; 14-й и 15-й разряды служат для обозначения номера операции.

При буквенном обозначении кода операции буква, обычно, пишется слева от адресов, и в этом случае номер операции может не указываться. Например, запись команды

0075	0022	0025	0	01

обозначает: сложить числа (операция сложения имеет номер 01). находящиеся в ячейках с номерами 0075 и 0022, и направить результат в ячейку с номером 0025. Контрольный знак отсутствует. Эту же команду можно записать в виде:

С	0075	0022	0025

где буква С обозначает операцию сложения, а контрольный разряд и номер операции опущены. Можно записать эту команду и таким образом:

+	0075	0022	0025

Контрольные знаки в командах служат для различных вспомогательных целей, в частности, они применяются для того, чтобы отмечать в программе определенные места, в которых должна останавливаться машина при необходимости вести вычисления по отдельным участкам программы с целью проверки правильности программ и работы машины,

* В отличие от рассмотренного выше общего правила перевода восьмеричных чисел в двоичные, при котором каждый восьмеричный разряд изображается обязательно тремя двоичными разрядами, в данном случае при переводе команды в двоичную систему вспомогательный разряд изображается одним двоичным разрядом.

3. Переадресация

Решение большинства задач на машине требует, как правило, производства большого количества операций (10^6 - 10^8) и такого же количества команд для их выполнения. Резкое уменьшение количества команд, входящих в состав программы, по сравнению с количеством команд, действительно выполняемых машиной в процессе вычислений, достигается за счет автоматического преобразования и многократного повторения в процессе вычислений первоначально введенных команд программы, т. е. за счет использования циклических изменяемых программ.

Рассматривая всю команду как одно число, можно производить с ней арифметические и логические операции, как с обычными числами, что используется для преобразования команд. Изменениям подвергаются, как правило, адреса в командах, что называется переадресацией.

Например, если нужно произвести суммирование ста чисел, расположенных в ячейках $a+1$, $a+2$,....$a+100$, и получить результат в некоторой ячейке b, то для этого потребуется (при отсутствии преобразования команд) 100 команд следующего вида:

$$+b,\ a+1,\ b$$
$$+b,\ a+2,\ b$$
$$\ldots\ldots\ldots\ldots$$
$$+b,\ a+100,\ b$$

Видно, что используемые при этом команды отличаются только вторыми адресами, а именно: в каждой последующей команде второй адрес увеличивается на единицу.

Используя переадресацию команд, можно для этого расчета составить программу, содержащую всего четыре команды следующего назначения.

1. Команда арифметической операции ($+b$, $a+1$, b).
2. Команда, увеличивающая второй адрес предыдущей команды на единицу.
3. Команда, подсчитывающая, сколько раз уже выполнена первая команда.
4. Команда условного перехода, осуществляющая либо возвращение к выполнению первой команды (если еще не просуммированы все числа), либо переход к другим вычислениям или остановку машины.

Если подобные расчеты требуется выполнить не один раз, а несколько, то для обеспечения возможности повторения расчетов по этой же программе необходимо добавить еще две команды, восстанавливающие программу в первоначальный вид после окончания каждого варианта расчета.

5. Команда, восстанавливающая команду № 1.
6. Команда, восстанавливающая команду № 3.

Восстановление команд может производиться либо посылкой в ячейку, содержащую измененную команду, первоначального кода этой команды, хранящегося где-либо в другой ячейке памяти, либо обратным преобразованием измененной команды.

Для переадресации команд должны быть подобраны и введены в машину некоторые дополнительные числа. Так, например, для изменения первого адреса на единицу нужно к общему 15-разрядному числу, обозначающему команду, прибавить или вычесть число 000100000000000.*

Аналогично для изменения второго адреса на единицу нужно прибавить или вычесть 000000010000000 и для третьего адреса — 000000000001000. Для изменения одновременно первого и второго адреса на единицу нужно использовать число 000100010000000, для изменения всех трех адресов — 000100010001000 и т. д. Все приведенные выше числа даны в восьмеричной системе счисления. При вводе в машину производится преобразование восьмеричных чисел в двоичную систему счисления по правилу, указанному в примечании на стр. 113.

Для удобства программирования помимо обычных операций сложения и вычитания, используемых при действиях над числами в нормальной форме, в машинах предусматриваются специальные операции сложения и вычитания, используемые при действиях над командами.

При выполнении операций специального сложения и вычитания участвующие в операции числа не нормализуются, и результат операции также выдается без нормализации. Ясно, что нормализация чисел, изображающих команды, недопустима, так как привела бы к полному искажению смысла команд. Кроме того, в операциях специального сложения и вычитания исключается возможность переносов между группами разрядов, соответствующими различным адресам в преобразуемой команде, хотя последнее и не является обязательным. В некоторых машинах нет специальных операций для действий над командами, а используются обычные операции сложения и вычитания с соответствующей блокировкой нормализации, как исходных чисел, так и результата операций.

Возможность производства операций над командами, т. е. переадресация команд, имеет важное значение для автоматизации вычислений. Поясним сущность переадресации еще одним примером.

Пусть имеется некоторая команда (в восьмеричной системе счисления).

0044	0033	0023	0	01

Эта команда означает: взять одно число из ячейки с адресом 44, взять другое число из ячейки с адресом 33, сложить оба числа и направить результат в ячейку с адресом 23.

Предположим, что нам нужно изменить эту команду таким образом, чтобы в следующий цикл вычислений сложение было проведено над числами, находящимися в соседних с первоначальными ячейках, т. е. в ячейках с номерами 45 и 34, и результат был бы направлен в ячейку 24. Для этого нужно к восьмеричному числу, обозначающему код команды, прибавить следующее число:

* В машинах операции с командами производятся не в восьмеричной системе счисления как в приводимых примерах, а в двоичной.

C=000100010001000.

После чего команда примет нужный вид:

0045	0034	0024	0	01

Подобные переадресации команд при решении задач производятся многократно, причем возможны изменения адресов на различные величины, как в сторону увеличения, так и в сторону уменьшения.

Иногда необходимо заменить в команде a_1, a_2, a_3, θ адрес a_i(i = 1, 2, 3) некоторым другим числом b, хранящимся в определенной ячейке памяти машины, причем заранее неизвестна величина разности b — a_i, которую нужно было бы прибавить к a_i, чтобы получить b. Ясно, что в этом случае невозможно осуществить указанное преобразование команды непосредственным прибавлением b — a_i к a_i. Предварительно нужно очистить i-ый адрес команды a_1 a_2, a_3, θ, т. е. заменить все разряды i-го адреса команды нулями, что может быть осуществлено с помощью специальной команды выделения части числа, реализуемой с помощью рассмотренной выше операции поразрядного логического умножения. После того как соответствующий адрес команды очищен, может быть произведено сложение кода команды с найденным значением числа b, Сложение должно быть произведено при помощи специальной операции сложения, причем число b должно быть предварительно сдвинуто таким образом, чтобы нужные разряды этого числа попали на место соответствующего адреса.

4. Условные переходы

При численном решении математических задач весьма часто требуется изменять порядок вычислений в зависимости от получаемых промежуточных результатов.

Помимо изменений порядка вычислений, связанных с оценкой получающихся промежуточных результатов (что имеет место и при ручных расчетах), в автоматических машинах необходимо изменять порядок выполнения команд, также и для обеспечения возможности построения циклических вычислительных процессов.

Для этого необходимо обеспечить автоматическое возвращение назад к определенному месту программы и повторение несколько раз заданного участка программы с соответствующими изменениями.

Для изменения порядка выполнения команд программы в машинах предусмотрены специальные команды условного перехода, позволяющие выбрать то или иное продолжение вычислительного процесса в зависимости от получаемых результатов.

Команда условного перехода имеет такое же строение, как и остальные команды (т. е. состоит из кода операции, адресов, контрольного разряда), только в адресах этой команды указываются не номера ячеек памяти, содержащих числа, а номера ячеек памяти, содержащих команды, к которым необходимо перейти после выполнения команды условного перехода.

Как правило, команды условного перехода во всех машинах построены таким образом, что реализуют выбор одного из двух возможных направлений продолжения вычислений в зависимости от наличия или отсутствия определенного признака в результатах предшествующих операций. Таким образом, команды условного перехода реализуют в машинах логические двоичные функции, т. е. функции, принимающие два значения в зависимости от аргументов, принимающих тоже только два значения.

Если необходимо осуществить выбор одного направления из большего числа возможных вариантов, то это может быть сделано последовательным применением нескольких команд условного перехода.

Итак, условные переходы производятся на основе некоторых признаков, установленных заранее и характеризующих результат операции, предшествующей команде условного перехода.

Такими признаками могут являться:

—наличие положительного или отрицательного знака результата;

— совпадение или несовпадение всех разрядов двух чисел при их сравнении;

— равенство нулю выделенной части числа;

— получение результата, меньшего по абсолютному значению, чем единица, и другие.

Поясним смысл команды условного перехода для случая, когда в качестве признака для выбора продолжения вычислений используется знак результата предшествующей операции. При этом команда условного перехода реализует следующее правило.*

Если результат положителен, то следующей после команды условного перехода должна выполняться команда, находящаяся в ячейке памяти, номер которой указан по первому адресу команды условного перехода.

Если же результат предыдущей операции отрицателен, то следующей после команды условного перехода будет выполняться команда, находящаяся в ячейке памяти с номером, указанным по второму адресу в команде условного перехода.

В случае трехадресных команд третий адрес в команде условного перехода либо совсем не используется, либо используется для вспомогательных целей.

Пусть, например, в программе будут записаны следующие две команды:

(K+1)	B	0014	0015	0020
(K+2)	E	K+n	K+m	0000,

где K+1 и K+2 — номера двух соседних ячеек памяти, в которых хранятся эти команды.

B — код операции вычитания, означающий, что из числа, находящегося в ячейке № 14, нужно вычесть число,

* Рассматриваемая команда условного перехода используется в машине „Стрела". В машине БЭСМ команда условного перехода указывает одновременно адреса сравниваемых чисел и адрес команды, к которой необходимо перейти при наличии определенного признака. При отсутствии признака должна выполняться следующая команда.

находящееся в ячейке №15, и результат направить в ячейку № 20;

E — означает код команды условного перехода;

K+n и *K+m* — номера ячеек памяти, в которых находятся две какие-то другие команды программы.

Пусть в ячейке с номером 0014 находится некоторое число *a* и в ячейке с номером 0015 — число *b.*

Если a≥b, то при вычитании получится положительный результат и команда условного перехода обеспечит выполнение в качестве следующей команды, не очередной команды, взятой из ячейки *K+3,* а команды, взятой; из ячейки *K+n.*

Если *a<b,* то результат вычитания будет отрицательный и команда условного перехода обеспечит переход к выполнению команды, взятой из ячейки *K+m.*

После перехода по команде условного перехода к выполнению какой-либо *N*-ой команды дальнейшее выполнение программы (при естественном порядке) идет последовательно, начиная с этой команды (*N, N*+1, *N*+2,....) до тех пор, пока не встретится новая команда условного перехода или же команда остановки.

Обычно в командах условного перехода (двух, трех, четырехадресных) указываются два адреса команд, к которым должен быть совершен переход после команды условного перехода. При одноадресной системе команд команда условного перехода содержит, как и любая одноадресная команда, всего один адрес, в котором указывается номер команды, к которой должен быть совершен переход при наличии некоторого признака в результатах предшествующих операций; при отсутствии же этого признака выполняется следующая за командой условного перехода очередная команда программы. Это правило используется и в некоторых машинах, имеющих многоадресные системы команд.

Команды условного перехода имеют большое значение для построения программ полностью автоматических вычислительных процессов. Сравнивая различные величины, участвующие в вычислениях, можно обеспечить выработку нужных признаков в определенные моменты вычислений для выполнения условных переходов и изменения хода вычислительного процесса, для преобразования и повторения отдельных участков программы.

Иногда необходимо в процессе вычислений произвести так называемый безусловный переход от одной команды к другой, т. е. после выполнения данной команды перейти к выполнению заранее определенной команды программы независимо от текущих результатов.

Ясно, что в машинах с принудительным порядком выполнения команд почти вся программа состоит из таких безусловных переходов, так как в каждой предыдущей команде указывается адрес следующей команды.

В машинах же с естественным порядком выполнения команд такие безусловные переходы в зависимости от конструкции машины осуществляются либо с помощью специальных команд безусловного перехода, либо при помощи рассмотренных выше команд условного перехода. В последнем случае на местах первого и второго адресов в команде условного перехода необходимо поставить одинаковые адреса, указывающие номер той команды, к которой должен быть совершен безусловный переход.

5. Пример системы команд

Ниже приводится перечень типовых команд электронной цифровой машины универсального назначения. Система команд трехадресная с естественным порядком выполнения программы. Для каждой команды указан номер в восьмеричной системе счисления, представляющий собой код операции.

Каждая команда изображается строчкой из четырех клеток: впервой клетке указывается символ операции (С— сложение, В—вычитание и т. д.), во второй, третьей и четвертой клетках малыми буквами обозначены адреса исходных чисел и результата. При пояснении содержания команд для краткости принято следующее правило.

Если какая-нибудь величина *a* находится в ячейке с номером *n,* то это записывается:

$$a = (n).$$

Наоборот, если номер ячейки, в которой находится величина *a,* равен *n,* то это записывается:

$$n = \langle a \rangle.$$

Команды арифметических и логических операций

Команда № 01

С	*а*	*б*	*в*

Алгебраическое сложение чисел *(а)* и *(б)* и направление результата по адресу *в.*

Команда № 02

$С_1$	*а*	*б*	*в*

Специальное сложение, используемое для изменения адресов команд в сторону увеличения. Команда *(а)* или *(б)* складывается с числом *(б)* или *(а)* и результат направляется в ячейку с адресом *в.* Как правило, адрес изменяемой команды совпадает с адресом *в.*

Команда № 03

В	*а*	*б*	*в*

Вычитание с учетом знаков обоих чисел. Из числа *(а)* вычитается число (*б*) и результат направляется в ячейку *в*.

К о м а н д а № 04

B_1	*а*	*б*	*в*

Получение разности модулей двух чисел $|(a)| - |(б)| = (в)$

К о м а н д а № 05

М	*а*	*б*	*в*

Умножение двух чисел (*а*) и (*б*) и направление результата в ячейку *в*.

К о м а н д а № 06

Н	*а*	*б*	*в*

Поразрядное логическое умножение двух чисел, находящихся в ячейках *а* и *б*. Команда используется для выделения из какого-либо числа или кода команды части, определенной специальным числом

К о м а н д а № 07

Ф	*а*	*б*	*в*

Поразрядное логическое сложение двух чисел (*а*) и (*б*) и направление результата в ячейку *в*. Команда используется при формировании чисел и команд из частей.

К о м а н д а № 10

Сд	*а*	*б*	*в*

Сдвиг содержимого ячейки *а* на количество разрядов, равное порядку числа (*б*). Если порядок числа (*б*) положителен, то сдвиг производится влево, в сторону старших разрядов, если отрицателен, то — вправо. При этом знаки числа, выходящие за разрядную сетку ячейки, теряются.

К о м а н д а № 11.

B_2	*а*	*б*	*в*

Специальное вычитание, используемое для уменьшения адресов команд. В ячейке *а* находится преобразуемая команда, а в ячейке *б* — специально подобранное число. Обычно адреса *а* и *в* совпадают.

К о м а н д а № 12

Ср	*а*	*б*	*в*

Сравнение двух чисел (*а*) и (*б*) путем поразрядного сложения сравниваемых чисел по модулю два. В ячейке *в* получается число, имеющее единицы в тех разрядах, в которых произошло несовпадение сравниваемых чисел (*а*) и (*б*).

К о м а н д ы у п р а в л е н и я

К о м а н д а № 13

Е	*а*	*б*	0000

Условная передача управления либо команде (*а*), либо команде (*б*) в зависимости от результата предшествующей операции. Для оценки результатов различных операций используются следующие признаки.

Для операций сложения, вычитания и вычитания модулей оценивается знак результата: при положительном или равном нулю результате осуществляется переход к команде (*а*), при отрицательном результате — к команде (*б*).

Результат операции умножения оценивается путем сравнения с единицей: переход к команде (*а*) происходит, если результат больше или равен единице, и к команде (*б*) — если меньше единицы.

Если условный переход происходит по результату операции сравнения, то переход к команде *(а)* происходит в случае совпадения сравниваемых чисел, и к команде (*б*) — при несовпадении хотя бы одного разряда сравниваемых чисел.

После операции Н (логическое поразрядное умножение) переход по команде условного перехода Е осуществляется к команде (*а*) в том случае, когда результат поразрядного логического умножения, т. е. выделяемая часть числа отлична от пуля, и к команде (*б*), когда она равна нулю.

Если нужно произвести в программе безусловный переход к какой-либо команде, находящейся, например, в ячейке *а,* то по первому и второму адресу команды условного перехода ставятся одинаковые адреса и эта команда примет вид:

Е	*а*	*а*	0000

Третий адрес в данной команде не используется, и поэтому на его месте указываются нули.

К о м а н д а № 14

У	*а*	0000	0000

Команда выполняется параллельно с ходом других операций и обеспечивает заблаговременную постановку зоны внешнего накопителя с адресом *а* в рабочее положение.

К о м а н д а № 15

Я	0000	0000	0000

Команда выполняет безусловную остановку машины.

К о м а н д ы г р у п п о в ы х п е р е с ы л о к

Для осуществления переписи чисел между накопителями служат специальные команды групповых пересылок. По второму адресу в этих командах ставится число n, обозначающее количество чисел в группе, которое должно быть переписано.

Групповые пересылки всегда производятся в порядке возрастания номеров ячеек в памяти.

К о м а н д а № 16

$П_1$	0000	*п*	*в*

Команда $П_1$ обеспечивает передачу данных из входных устройств (с перфокарт, перфолент и т. п.) в память. По третьему адресу команды *в* указывается начальный адрес группы ячеек памяти, куда должны записываться числа. С перфолент или перфокарт данные переписываются подряд, начиная с первой строки.

К о м а н д а № 17

$П_2$	0000	*п*	*в*

Команда $П_2$ обеспечивает передачу группы *п* чисел из входных устройств во внешний накопитель в зону *в*

К о м а н д а № 20

$П_3$	*а*	*п*	*в*

Команда обеспечивает последовательную (построчную) передачу *п* чисел из зоны *а* внешнего накопителя в ячейки памяти, начиная с ячейки с номером *в.*

К о м а н д а № 21

$П_4$	*а*	*п*	0000

Команда обеспечивает передачу в выходное устройство (на перфоленты или перфокарты) группы *п* чисел из памяти, начиная с номера *а.* Запись на перфоленту или перфокарты начинается, как правило, с первой строки и поэтому дополнительных указаний для адресов записи не требуется.

К о м а н д а № 22

$П_5$	*а*	*п*	*в*

Команда $П_5$ обеспечивает перепись группы *п* чисел из одного места памяти с начальным номером *а* в другое место памяти с начальным номером *в.*

К о м а н д а № 23

$П_6$	*а*	*п*	*в*

Команда $П_6$ обеспечивает перенос группы *п* чисел из памяти с начальным номером *а* во внешний накопитель в зону с адресом *в.*

К о м а н д а № 24

$П_7$	*а*	*п*	0000

Команда $П_7$ служит для передачи *п* чисел из зоны с адресом *а* внешнего накопителя в выходное устройство.

Следует заметить, что в современных машинах, например в машине ИБМ-701, предусматривается возможность выполнения команд $П_2$ и $П_7$ одновременно с ходом других операций, что приводит к существенной экономии машинного времени при решении задач с большим количеством исходных данных и результатов вычислений.

К о м а н д ы с т а н д а р т н ы х п о д п р о г р а м м

Как уже отмечалось ранее, в электронных цифровых машинах универсального назначения широко применяется метод стандартных подпрограмм. Состав стандартных подпрограмм в различных машинах бывает весьма разнообразным. Даже для одной и той же машины этот состав может изменяться в зависимости от требований конкретных задач, решаемых на машине.

Ниже приводится перечень команд некоторых наиболее употребительных стандартных подпрограмм и указываются применяемые в подпрограммах расчетные формулы. Эти формулы не являются единственно возможными и приводятся здесь, главным образом, для иллюстрации принципов действия подпрограмм.

Приводимые характеристики стандартных подпрограмм в отношении количества команд и тактов работы также преследуют цель — дать общее представление о методе стандартных подпрограмм, и могут являться только ориентировочными при оценке подпрограмм подобных типов. В общем случае количество тактов работы машины при выполнении стандартных подпрограмм, реализующих итерационные процессы, зависит от требуемой точности вычисления функций. Приводимые ниже данные о количестве тактов относятся к случаю, когда машина оперирует с десятизначными десятичными числами и вычисления функций производятся с точностью приблизительно до единицы десятого разряда.

К о м а н д а № 25

Д	*а*	*б*	*в*

Стандартная подпрограмма служит для выполнения операции деления: число *(а)* делится на число (б) и частное направляется в ячейку *в.*[*]

Операция деления выполняется в два этапа: сначала получается величина, обратная делителю, а затем делимое множится на эту обратную величину. Вычисление величины, обратной делителю, может производиться по итерационной формуле

$$y_{n+1} = y_n(2 - y_n x).$$

При $x = d \cdot 10^p$, где $1/10 < d < 1$, в качестве первого приближения можно принять $y_0 = 10^{-p}$. (Здесь "10" — это два в двоичной системе счисления). В этом случае стандартная подпрограмма будет иметь ориентировочно 8 — 10 команд и будет выполняться за 18 — 20 тактов работы машины. (Тактом работы машины называется время выполнения одной типовой команды).

[*] Из конструктивных соображений в вычислительной машине „Стрела" осуществляется стандартная подпрограмма получения обратной величины, а операция деления выполняется при помощи двух команд: команды, отсылающей к этой подпрограмме, и команды умножения. Мы условно объединили обе эти команды в одну команду стандартной подпрограммы деления, придав тем самым команде деления такой же вид, какой она имеет в машинах со специальной операцией деления.

К о м а н д а № 26

✓	*а*	0000	*в*

Команда обеспечивает получение величины $\sqrt{x}$ из величины $x = (a)$ и направление результата в ячейку *в*.

Сначала вычисляется $\frac{1}{\sqrt{x}}$ по итерационной формуле

$$y_{n-1} = \frac{1}{10} y_n (11 - x y_n^{10}),$$

где в качестве первого приближения принято

$$y_0 = 10^{E(\frac{p}{10})*}$$

Затем умножением на x получается величина $\sqrt{x}$. Стандартная подпрограмма содержит 14 команд и выполняется за 40 тактов.

К о м а н д а № 27

e^x	*а*	0000	*в*

Команда обеспечивает образование e^x для величины $x = (a)$ и направление результата в ячейку *в*. Вычисление проводится путем разложения e^x в степенной ряд. Стандартная подпрограмма содержит 20 команд и выполняется за 40 тактов.

К о м а н д а № 30

$\ln x$	*а*	0000	*в*

Команда обеспечивает получение функции $\ln x$ для величины $x = (a)$ и направление результата в ячейку *в*. Вычисления проводятся путем разложения $\ln x$ в ряд. Подпрограмма содержит 15 команд и выполняется за 60 тактов.

К о м а н д а № 31

$\sin x$	*а*	0000	*в*

Команда обеспечивает получение функции $\sin x$ и направление результата в ячейку *в*. Вычисления проводятся в два этапа: сначала значение аргумента приводится к первой четверти, а затем вычисляется значение функции при помощи разложения в ряд. Подпрограмма содержит 18 команд и выполняется за 25 тактов.

К о м а н д а № 32

Пр_1	*а*	*п*	*в*

Преобразование группы *п* чисел, расположенных в ячейках *а*, *а*+1..., из двоично-десятичной системы в двоичную и направление результатов в ячейки *в*, *в*+1,... Преобразование может производиться по методу, изложенному в § 6. Подпрограмма содержит 14 команд и выполняется за 50 тактов (на каждое число).

К о м а н д а № 33

Пр_2	*а*	*п*	*в*

Преобразование группы *п* чисел, расположенных в ячейках *а*, *а* + 1..., из двоичной системы в двоично-десятичную и направление результатов в ячейки *в*, *в* + 1... Преобразование может проводиться по методу, изложенному в § 6. Подпрограмма содержит около 30 команд и выполняется за 100 тактов (на каждое число).

К о м а н д а № 34

---

* Выражение *E(x)* обозначает целую часть числа *x*. Все числа представлены в двоичной системе.

К	*а*	*п*	*в*

Команда контрольного суммирования. По этой команде производится формальное сложение группы *п* чисел, расположенных в ячейках памяти, начиная с номера *а,* и результат направляется в ячейку *в.* Формальное сложение заключается в том, что суммирование чисел и кодов команд производится одинаково по правилам сложения чисел с фиксированной запятой (без нормализации и учета порядков) с той лишь разницей, что единица переноса, получающаяся в старшем разряде, прибавляется путем циклического переноса в младший разряд. Таким образом, может быть получена формальная сумма любых кодов, находящихся в памяти в любой момент времени. Эта команда используется для контроля правильности вычислений методом контрольных сумм (см. ниже §13).

6. Групповые операции

Групповой операцией называется такой режим работы машины, когда какая-либо операция производится последовательно над каждым числом из заданной группы чисел, приводя к образованию новой группы чисел. Получаемая группа чисел в общем случае может занимать произвольное место в памяти и, в частности, записываться на место прежней группы.

Групповые операции могут выполняться при помощи циклических подпрограмм, включенных в основную программу. В этом случае, как нетрудно видеть, для выполнения групповой операции, например вида:

$$a_1 + b = c_1$$
$$a_2 + b = c_2$$
$$\ldots\ldots\ldots\ldots$$
$$\ldots\ldots\ldots\ldots$$
$$a_n + b = c_n$$

необходимо иметь в подпрограмме, по крайней мере, четыре команды: команду для выполнения самой операции, команду для изменения адресов предыдущей команды, команду для контроля окончания цикла и команду условного перехода. Кроме того, обычно нужны команды еще для восстановления измененных в процессе выполнения программы команд. Помимо увеличения объема программы, такой способ выполнения групповых операций приводит к значительному увеличению количества тактов работы машины, а, следовательно, и времени решения задач.

Поэтому в большинстве электронных цифровых машин универсального назначения предусматриваются для выполнения групповых операций специальные устройства, обеспечивающие автоматическое стандартное изменение соответствующих адресов в командах.

Выполнение какой-либо групповой операции осуществляется обычно при помощи двух команд: подготовительной команды и исполнительной команды. Подготовительная команда настраивает машину на режим выполнения групповых операций, а исполнительная команда определяет, какая операция и над какими числами должна быть выполнена.

Может быть, например, принята следующая форма задания групповых операций.

В подготовительной команде код операции показывает, что это подготовительная команда групповой операции, первый и второй адреса, показывают шаги изменения первого и второго адресов исполнительной команды, а третий адрес — число повторений данной групповой операции.

В исполнительных командах коды операций выполняют обычные функции, а адреса показывают начальные номера ячеек двух групп исходных чисел и группы результатов операций. Изменение адресов в исполнительной команде осуществляется, как правило, только в сторону увеличения, причем адреса для результатов операций увеличиваются на единицу с каждым повторением. Например, групповая операция для попарного суммирования двух групп по *п* чисел, расположенных в ячейках

$$a, \quad a + 3, a + 6 \ldots a + 3(n - 1)$$
$$b, \quad b + 2, b + 4 \ldots b + 2(n - 1),$$

с записью результатов в ячейки

$$c, \quad c + 1, c + 2, \ldots c + (n - 1)$$

будет иметь вид

Ц, 3, 2, *п,*
С, *а, b, с,*

где Ц — код подготовительной команды групповой операции.

При помощи групповых операций можно производить любые арифметические или логические действия над группами чисел, предусмотренные системой команд, принятой в машине.

Удобно при помощи групповых операций производить попарное перемножение двух групп чисел, суммирование ряда заданных чисел, перенос группы чисел из одного участка памяти в другой и т. д.

Ценность введения групповых операций заключается в том, что при их применении обеспечивается сокращение числа тактов работы машины и объема программы по сравнению с теми случаями, когда эти же действия выполняются при помощи циклов в основной программе.

Сокращение достигается не только за счет уменьшения количества команд, осуществляющих непосредственно

арифметические действия над заданными числами, но, главным образом, за счет сокращения количества вспомогательных команд (для подсчета циклов, для преобразования и восстановления команд и т. п.).

Поэтому применение групповых операций выгодно даже в тех случаях, когда исходные группы чисел содержат всего три — четыре числа.

7. Пример одноадресной системы команд

В последнее время в электронных вычислительных машинах, как в малых, так и в больших, получают все большее применение системы одноадресных команд. Учитывая сказанное выше, а также то, что нам в дальнейшем придется пользоваться одноадресными командами при изложении вопросов автоматизации программирования, приведем краткие сведения о типовой системе команд одноадресного кода, предварительно сделав следующие замечания.

1. В одноадресных машинах могут использоваться один или несколько различных постоянных регистров для задания и запоминания участвующих в операциях величин, например, отдельный регистр для операций сложения и вычитания, отдельный регистр для умножения и т. д. Соответственно этому имеется обычно несколько команд, обеспечивающих пересылку данных в регистры и обратно.

Будем считать, что в машине имеется только один регистр Р, который используется для всех операций; при этом результат операции остается всегда в регистре.

С точки зрения техники программирования такое допущение не вносит каких-либо ограничений, так как количество и характер операций в любой программе будет одинаковым в обоих случаях.

2. Контрольные знаки в командах служат для разделения программы на участки и обеспечивают возможность задания режима работы машины по «контрольным точкам», т. е. режима работы с остановками на каждом контрольном знаке.

3. Адреса в командах в различных машинах имеют различное количество разрядов и записываются при программировании либо в восьмеричной системе, либо в десятичной.

Будем считать, что адреса записываются трехзначными десятичными числами от 000 до 999.

Код операции записывается двухзначными десятичными числами и для контрольного знака отведен один десятичный разряд.

Таким образом, полный код одноадресной команды при наших предположениях будет представлять собой шестизначное десятичное число.

4. Состав команд в основном совпадает с приведенным выше типовым составом трехадресных команд. Передача управления по команде условного перехода после команд С, $В_1$, $В_2$, М, Ср, Н производится по тем же правилам и критериям, что и в трехадресной системе команд.

5. Предполагаем, что приводимая система одноадресных команд относится к машине с фиксированной запятой. В этом случае для преобразования команд не требуются операции специального сложения и вычитания, так как действия над командами могут производиться при помощи обычных операций сложения и вычитания. (При выполнении этих операций в машинах с фиксированной запятой не происходит нормализации исходных чисел и результатов операций).

Система одноадресных команд приведена в табл. IV.2.

Таблица IV.2

Таблица одноадресных команд

Команды			Символ операции	Содержание команд	Пояснения
Код операций	Адрес	Вспомогательный разряд			
1	2	3	4	5	6
01	n	0	$П_1$	$(n) \to Р$	Число (n) переносится в регистр
02	n	0	С	$(Р) + (n) \to Р$	Число (n) складывается с числом в регистре и результат остается в регистре
03	n	0	$В_1$	$(Р) - (n) \to Р$	Из числа, находящегося в регистре, вычитается число (n) и разность записывается в регистре
04	n	0	$В_2$	$\|(Р)\| - \|(n)\| \to Р$	Вычитание модулей
05	n	0	М	$(Р) \cdot (n) \to Р$	Число в регистре множится на число, находящееся в ячейке n, и результат остается в регистре
06	n	0	Д	$(Р) : (n) \to Р$	Число в регистре делится на число в ячейке n и частное остается в регистре
07	n	0	$П_2$	$(Р) \to n$	Перенос числа из регистра в ячейку n

Продолжение

Команды			Символ операции	Содержание команды	Пояснения
Код операций	Адрес	Вспомогательный разряд			
1	2	3	4	5	6
08	n	0	$Е_1$	Условный переход	Если число в регистре не отрицательное, то выполнять очередную команду. В противном случае перейти к выполнению команды в ячейке n
09	n	0	$Е_2$	Безусловный переход	Перейти к выполнению команды в ячейке n
10	n	0	Сд	Сдвиг	Сдвинуть все разряды числа в регистре на количество разрядов, равное числу (n). Если число (n) положительное, то сдвиг производится влево; в противном случае—вправо
11	n	0	Н	Выделение части числа	Поразрядное логическое умножение числа в регистре на число (n)
12	n	0	Ф	Формирование числа	Поразрядное логическое сложение числа в регистре с числом (n)
13	n	0	Ср	Сравнение	Сравниваются числа (P) и (n), результат сравнения записывается в регистре таким образом, что в несовпадающих разрядах получаются единицы, а в совпадающих—нули
14	n	0	Я	Остановка	Машина останавливается и на пульт управления выдается содержимое ячейки n
15	n_1 m n_2	0 0 0	К О О	—	Перепись данных из зоны m внешнего накопителя в ячейки памяти от n_1 до n_2
16	n_1 m n_2	0 0 0	Л О О	—	Перепись данных из памяти (от ячейки n_1 до n_2) во внешний накопитель (в зону m)
17	n_1 n_2	0 0	Р О	—	Выдача результатов из машины. Выдается содержимое памяти от ячейки n_1 до ячейки n_2

12. НЕПОСРЕДСТВЕННОЕ ПРОГРАММИРОВАНИЕ

1. Последовательность команд и размещение данных в памяти

Непосредственное программирование заключается в составлении в соответствии с выбранным численным методом последовательности команд программы и плана размещения данных в памяти. В последовательность команд должны входить не только команды, непосредственно необходимые для производства вычислений, но и вспомогательные команды, обеспечивающие автоматическую работу машины (команды для преобразования команд, для переноса чисел и др.).

Программа записывается на стандартном бланке. Для трехадресной машины можно рекомендовать бланк, имеющий форму, представленную табл. IV.3.

Таблица IV.3

№№ п.п.	Команды						Пояснения
1	2	3	4	5	6	7	8

В первую колонку записываются в восьмеричной системе счисления порядковые номера команд, соответствующие номерам ячеек памяти, в которых будут находиться эти команды. Во вторую колонку записывается буквенное обозначение операции — символ операции. В третью, четвертую и пятую колонки

записываются адреса исходных чисел и результатов операций. Шестая колонка служит для обозначения контрольных знаков, разделяющих всю программу на отдельные этапы. В седьмую колонку проставляются коды операций. В восьмой колонке даются необходимые пояснения и примечания к программе.

План размещения исходных и промежуточных данных в памяти машины, определяющий порядок заполнения и освобождения ячеек в процессе решения задачи, является неотъемлемой составной частью программы.

Обычно все поле памяти машины делится на две части: первая часть используется для записи программы, а вторая часть — для записи исходных данных и промежуточных и окончательных результатов решения задачи.

Ячейки памяти, используемые для размещения данных, могут быть, в свою очередь, разделены на две категории: ячейки, постоянно занятые одними и теми же величинами, и так называемые «рабочие» ячейки, временно занимаемые для хранения различных величин, в основном для хранения промежуточных результатов вычислений.

Особенностью работы машинной памяти является то, что запись чисел в ячейки не зависит от прежнего содержания этих ячеек. Запись нового числа автоматически уничтожает старую запись и поэтому не следует посылать числа в те ячейки, содержимое которых может понадобиться в дальнейшем.

Будем считать, что в рассматриваемой нами машине имеется память емкостью в 2048 ячеек.

Все ячейки памяти пронумерованы подряд, начиная с 0 до 2047 в десятичной системе счисления или от 0 до 3777 в восьмеричной системе счисления.

Будем считать также, что в ячейке, имеющей нулевой номер, постоянно записан нуль. При этом если нуль потребуется в процессе расчета, то его можно взять из нулевой ячейки. Если же в эту ячейку направить в процессе счета результат какой-нибудь операции, то этот результат там не будет записан, т. е. в нулевой ячейке сохранится число нуль. Поэтому, если после выполнения какой-либо команды не требуется сохранить результат, а нужно получить лишь признак результата, используемый для передачи управления той или иной команде программы, то в данной команде в качестве третьего адреса может быть указан нулевой адрес.

При составлении плана размещения данных в машинной памяти необходимо учитывать в первую очередь возможность построения наиболее простым образом необходимых циклов вычислений. Для этой цели соответствующие исходные данные должны размещаться в определенном порядке, отвечающем порядку их использования при выполнении циклов вычислений и допускающем последовательную выборку этих данных путем однообразной модификации адресов команд.

Кроме того, при размещении данных в памяти должна быть предусмотрена простота переписи данных как между памятью и накопителями, так и между запоминающими устройствами и входными и выходными устройствами. Для этого данные, которые необходимо переписывать, должны объединяться в возможно более крупные группы для переписи в один прием и размещаться в памяти в той последовательности, в какой они должны переписываться.

Одновременно с составлением плана размещения данных во внутренней памяти подготавливается схема размещения данных на магнитных лентах, барабанах или перфолентах внешних накопителей. Размещение данных во внешнем накопителе осуществляется большими группами по зонам. Каждая зона имеет свой номер — адрес зоны. Внешний накопитель в вычислениях непосредственно не участвует, а является резервом для внутренней памяти.

В связи с этим важным вопросом, возникающим при составлении программ для сложных задач, является планирование обмена информацией между внутренней памятью и внешним накопителем. Условимся также для определенности считать, что внешний накопитель в рассматриваемой машине имеет 512 зон, в каждую из которых может быть записано до 512 чисел. Зонам внешнего накопителя присвоены номера от 4000 до 4777 в восьмеричной системе счисления или от 2048 до 2559 в десятичной системе.

В качестве простейшего примера покажем, как составляется программа для решения квадратного уравнения

$$ax^2 + bx + c = 0$$

по общеизвестной формуле

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

для случая, когда $b^2 - 4ac \geq 0$.

Исходные данные расположим, как показано в табл. IV.4.

Таблица IV.4

Номера ячеек	0100	0101	0102	0103
Хранимое в ней число	*a*	*b*	*c*	2

В качестве рабочих ячеек используем ячейки с номерами 0075, 0076, 0077 и в конце счета ячейку 0100.

В табл. IV.5 приведена последовательность команд [*], а в табл. IV.6 показано содержимое рабочих ячеек после

[*] В данном случае и в дальнейшем мы используем типовую систему команд, приведенную в разделе 5 (стр. 120).

выполнения каждой команды. Нумерацию команд программы начнем с номера 0020.

Таблица IV.5

№№ п.п.	команды						Пояснения
1	2	3	4	5	6	7	8
0020	М	0100	0103	0075	0	05	—
0021	М	0075	0103	0076	0	05	—
0022	М	0076	0102	0076	0	05	—
0023	М	0101	0101	0077	0	05	—
0024	В	0077	0076	0076	0	03	—
0025	√	0076	0000	0076	0	26	—
0026	В	0000	0101	0077	0	03	—
0027	В	0077	0076	0100	0	03	—
0030	С	0077	0076	0077	0	01	—
0031	Д	0100	0075	0100	0	25	—
0032	Д	0077	0075	0077	0	25	—

Таблица IV.6

Номера выполненных команд	Номера рабочих ячеек			
	0075	0076	0077	0100
После 0020 . . .	$2a$	—	—	a
„ 0021 . . .	$2a$	$4a$	—	a
„ 0022 . . .	$2a$	$4ac$	—	a
„ 0023 . . .	$2a$	$4ac$	b^2	a
„ 0024 . . .	$2a$	b^2-4ac	b^2	a
„ 0025 . . .	$2a$	$\sqrt{b^2-4ac}$	b^2	a
„ 0026 . . .	$2a$	$\sqrt{b^2-4ac}$	$-b$	a
„ 0027 . . .	$2a$	$\sqrt{b^2-4ac}$	$-b$	$-b-\sqrt{b^2-4ac}$
„ 0030 . . .	$2a$	$\sqrt{b^2-4ac}$	$-b+\sqrt{b^2-4ac}$	$-b-\sqrt{b^2-4ac}$
„ 0031 . . .	$2a$	$\sqrt{b^2-4ac}$	$-b+\sqrt{b^2-4ac}$	$x_1=\frac{-b-\sqrt{b^2-4ac}}{2a}$
„ 0032 . . .	$2a$	$\sqrt{b^2-4ac}$	$x_2=\frac{-b+\sqrt{b^2-4ac}}{2a}$	$x_1=\frac{-b-\sqrt{b^2-4ac}}{2a}$

Первая команда (0020).
«Умножить число *a* на 2 и результат записать в рабочую ячейку 0075». Эта команда имеет вид

0020. М 0100 0103 0075 0 05.

Вторая команда (0021).
«Полученное число *2a* умножить еще раз на два и записать в рабочую ячейку 0076».

0021. М 0075 0103 0076 0 05.

Третья команда (0022).
«Полученное число 4*a* умножить на *c* и результат записать в рабочую ячейку 0076».

0022. М 0076 0102 0076 0 05.

Четвертая команда (0023).
«Возвести число *b* в квадрат и результат записать в рабочую ячейку 0077».

0023. М 0101 0101 0077 0 05.

Пятая команда (0024).
«Из числа b^2 отнять число 4*ac* и результат записать в ячейку 0076».

0024. В 0077 0076 0076 0 03.

Шестая команда (0025).

«Извлечь квадратный корень из числа b^2 —4ac и результат записать в ячейку 0076».

0025 ✓ 0076 0000 0076 0 26.

Седьмая команда (0026).

«В ячейку 0077 записать число —b».

0026 В 0000 0101 0077 0 03.

Восьмая команда (0027).

„От числа — b отнять число $\sqrt{b^2-4ac}$ и результат записать в ячейку 0100".

0027. В 0077 0076 0100 0 03.

Девятая команда (0030).

„К числу — b прибавить число $\sqrt{b^2-4ac}$ и результат записать в ячейку 0077".

0030. С 0077 0076 0077 0 01.

Десятая команда (0031).

„Вычислить первый корень уравнения $x_1=\dfrac{-b-\sqrt{b^2-4ac}}{2a}$; для этого число $-b-\sqrt{b^2-4ac}$, находящееся в ячейке 0100, разделить на число 2a, находящееся в ячейке 0075, и результат направить в ячейку 0100".

0031 Д 0100 0075 0100 0 25.

Одиннадцатая команда (последняя) (0032).

«Вычислить второй корень уравнения $x_2=\dfrac{-b+\sqrt{b^2-4ac}}{2a}$; для этого число $-b+\sqrt{b^2-4ac}$, находящееся в ячейке 0077, разделить на число $2a$ и результат направить в ячейку 0077".

0032 Д 077 0075 0077 0 25.

В табл. IV.5 записаны подряд (в восьмеричной системе) все приведенные выше команды. Они образуют программу для решения квадратного уравнения.

2. Блок-схемное программирование

С усложнением задач трудности программирования быстро возрастают. Это заставляет применять различные приемы, облегчающие технику программирования.

Основным приемом, используемым при непосредственном (не автоматическом) программировании сложных задач, является метод блок-схем, заключающийся в расчленении всей задачи на более или менее самостоятельные простые части и программировании задачи по частям.

При составлении программ целесообразно придерживаться следующего порядка.

1. Составляется блок-схема программы. Процесс численного решения задачи расчленяется на отдельные этапы вычислений, и каждый этап изображается графически в виде прямоугольника, на котором пишется название этапа. Отдельные прямоугольники соединяются между собой стрелками в той последовательности, в какой должны выполняться этапы вычислений.

2. Составляются разделы программы для отдельных этапов вычислений. При этом если возможно, то используются участки программ ранее решенных задач. Это в отдельных случаях сокращает работу по программированию и уменьшает возможность появления ошибок в программах.

При первоначальном составлении программ для отдельных этапов адреса в командах удобно указывать сначала в условном виде, обозначив начальные адреса отдельных участков поля машинной памяти буквами, от которых вести дальнейший отсчет адресов, например, K + 1, K + 2, K + 3 и т. д. Это вызывается следующими соображениями.

Во-первых, заранее не известно, сколько команд будет содержать программа, и, следовательно, не известно, в каких ячейках будут располагаться команды и исходные данные.

Во-вторых, в программу, составленную в условном буквенном виде, легче вносить всякие изменения и исправления, что является неизбежным при первоначальном составлении программы.

Можно принять также следующий прием для первоначального обозначения адресов в командах: адреса

обозначать теми же буквами или цифрами, что и величины, хранящиеся в этих ячейках, записывая эти обозначения в угловых скобках. Например, <*a*> обозначает номер ячейки, в которой находится величина *a*. Этот прием облегчает чтение и проверку программы.

3. Выбираются или составляются контрольные и вспомогательные участки программы: программы ввода данных, перевода данных из одной системы счисления в другую и др.

4. Составляется план размещения данных в памяти машины. При первоначальном составлении плана размещения данных можно также для отдельных групп материала пользоваться буквенными адресами, например, $A + 1$, $A + 2$... или $C + 1$, $C + 2$... и т. д. Сначала размещается программа, подсчитывается общее количество ячеек, необходимых для записи программы, затем отдельные участки программ объединяются вместе и номера ячеек памяти, в которых размещается программа (порядковые номера команд программы), записываются своими действительными значениями.

Программу обычно размещают, начиная с некоторой *k*-ой ячейки ($k \geq 20$), так как первые ячейки памяти используются для вспомогательных целей (размещения программы ввода данных, для выполнения стандартных подпрограмм и др.).

После того, как определено место программы во внутренней памяти, размещаются все остальные исходные данные, вспомогательные величины и специальные числа. Определяются истинные значения адресов всех величин и производится расписывание адресов в командах программы. Заполняются графы 7 и 8 в бланках программы, т. е. указывается содержание операций, выполняемых по отдельным командам, и даются необходимые пояснения и примечания, облегчающие чтение и проверку программы.

5. Производится детальная проверка всей программы как самим программирующим, так и другими лицами и составляется инструкция оператору по применению данной программы.

При составлении программ нужно особенно внимательно следить за тем, чтобы в ячейки памяти, содержимое которых может понадобиться в процессе вычислений, не посылались еще другие результаты.

3. Программирование циклических процессов вычислений

Как уже говорилось, вычислительный процесс называется циклическим, если он сводится к многократным вычислениям по одним и тем же формулам, в которые при каждом новом их применении подставляются новые исходные данные.

Повторяющиеся этапы такого вычислительного процесса называются его циклами. Как известно, для большинства численных методов характерна цикличность вычислительного процесса.

Основным принципом программирования циклического вычислительного процесса является построение для него циклической программы.

Циклический вычислительный процесс в тех случаях, когда количество циклов заранее известно, можно представить в виде развернутой программы, последовательно записав в виде серии команд все этапы вычислений. При большом количестве циклов это приводит к чрезвычайно длинным программам, зачастую практически непригодным. (Эти программы могут просто не поместиться в память машины).

При очень большом количестве циклов или в случае, когда количество циклов зависит от получаемых результатов (например, от точности результатов при решении уравнения способом последовательных приближений), составление развернутой программы становится вообще невозможным.

Составление циклической программы состоит в следующем. Составляется участок программы, реализующий первый цикл вычислительного процесса. Затем ставится ряд команд, служащих для видоизменения адресов команд этого участка — команды переадресации, с тем, чтобы при повторном его применении велись вычисления с уже новыми исходными данными. Далее ставится некоторое количество команд, нужных для контроля числа повторений этого участка программы и обеспечивающих (в зависимости от результатов вычислений или числа повторений) либо повторения всей серии команд, либо переход к дальнейшим вычислениям. Вся полученная теперь серия команд называется циклом программы.

В качестве простейшего примера циклической программы рассмотрим программу вычисления функции e^x при помощи разложения в ряд

$$e^x = 1 + x + \frac{x^2}{1 \cdot 2} + \frac{x^3}{1 \cdot 2 \cdot 3} + \ldots + \frac{x^n}{1 \cdot 2 \cdot 3 \ldots n} + \ldots$$

Цикличность процесса вычислений здесь очевидна: значение каждого нового члена ряда a_{n+1} получается из значения вычисленного члена ряда a_n одним и тем же приемом, а именно путем умножения на x и деления на число $n + 1$.

$$a_{n+1} = \frac{a_n x}{n+1}.$$

Вычисление каждого нового значения суммы ряда производится путем прибавления к предыдущему значению суммы S_{n-1} очередного члена a_n

$$S_n = S_{n-1} + a_n,$$

при этом $S_0=1$, $a_0=1$.

Блок-схема программы имеет вид, показанный на рис. IV.1.

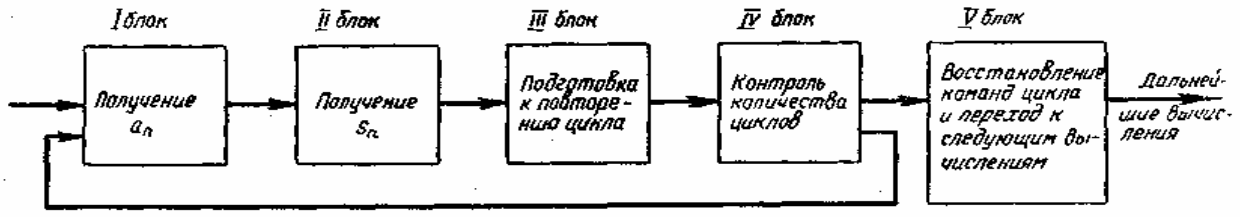

Рис. IV.1.

Выберем в качестве критерия для окончания вычислений получение заданного числа N членов в разложении в ряд. При этом нужно помнить, что нумерация членов ряда начинается с нуля

$$a_0 = 1,\ a_1 = \frac{x}{1!},\ \ldots,\ a_n = \frac{x^n}{n!},\ \ldots$$

и что номер последующего члена ряда, входящего в S_n, на единицу меньше числа членов. Команды программы будем размещать в ячейках памяти, начиная с номера 20. Исходные данные для вычислений размещаем в ячейках памяти, начиная с номера 100 (см. табл. IV.7).

Таблица IV.7

Номера ячеек	100	101	102	103	104	105	106
Данные	1, (S_n)	1, (a_n)	N	1, (n)	1	x	Рабочая ячейка

В ячейке 100 будут помещаться в дальнейшем значения частных сумм S_n, в ячейке 101 - значения a_n, в ячейке 103 - значения n. Программа приведена в табл. IV.8 (в восьмеричной системе).

Таблица IV.8

№№ п/п.	Команды						Пояснения
1	2	3	4	5	6	7	8
0020	М	0101	0105	0101	0	05	I блок $a_{n-1} \cdot x$
0021	Д	0101	0103	0101	0	25	I блок $a_{n-1} \cdot x \cdot \frac{1}{n} = a_n$
0022	С	0100	0101	0100	0	01	II блок $S_{n-1} + a_n = S_n$
0023	С	0103	0104	0103	0	01	III блок $n + 1$
0024	В	0103	0102	0106	0	03	IV блок $(n+1) - N$
0025	Е	0026	0020	0000	0	13	
0026	R	0100	a	b	0	—	Использование e^x для дальнейших вычислений.
0027	Ф	0104	0000	0100	0	07	V блок $1 \to (100)$
0030	Ф	0104	0000	0101	0	07	V блок $1 \to (101)$
0031	Ф	0104	0000	0103	0	07	V блок $1 \to (103)$

В восьмой колонке указаны вычисления, которые производит машина при каждом выполнении цикла программы.

Команды 20, 21 служат для вычисления члена a_n.

Команда 20 производит умножение величины a_{n-1}, находящейся в ячейке 101, на значение аргумента x, находящееся в ячейке 105. Результат посылается в ячейку 101, в которой будет получаться величина a_n.

Команда 21 осуществляет деление величины $a_{n-1}x$ на n.

Команда 22 служит для получения величины S_n; к величине S_{n-1}, находящейся в ячейке 100, прибавляется величина a_n и полученная величина S_n направляется в ту же ячейку 100.

Команда 23 служит для увеличения текущего значения n на единицу, при этом в ячейке 103 получается число n + 1, где под n подразумевается номер следующего, подлежащего к вычислению члена ряда.

Команда 24 служит для контроля количества вычисленных членов ряда: из числа $n + 1$ вычитается заданное число членов N.

При получении положительного или нулевого результата, т. е. при $n + 1 \geq N$, команда 25 передает управление

команде 26; при получении отрицательного результата, т. е. при $n + 1 < N$, команда 25 передает управление команде 20, т. е. производится вычисление следующего члена ряда. Команда 26 предусматривает использование вычисленного значения e^x: либо перенос его в другую ячейку с последующим выводом на печать, либо использование для последующих вычислений. В команде 26 буква *R* означает какую-либо операцию, а *a* и *b* — какие-либо адреса.

Команды 27, 30, 31 служат для восстановления первоначального вида данного участка программы. Команда 27 восстанавливает в ячейке 100 начальное значение $S_0 = 1$. Команда 30 восстанавливает первоначальное значение $a_0 = 1$ в ячейке 101. Команда 31 восстанавливает начальное значение $n = 1$ в ячейке 103. После этого программа оказывается подготовленной для вычисления e^x при новом значении аргумента *x*.

В приведенном примере отсутствовало преобразование команд. Для иллюстрации этой стороны дела рассмотрим пример программы для вычисления значения полинома

$$y = a_0x^n + a_1x^{n-1} + a_2x^{n-2} + .. + a_{n-1}x + a_n.$$

Проводя рассуждения в общем виде, для наглядности проиллюстрируем их с помощью многочлена четвертой степени

$$y = a_0x^4 + a_1x^3 + a_2x^2 + a_3x + a_4.$$

Расчет полиномов удобно вести по так называемой схеме Горнера. Для этого многочлен переписываем в следующем виде

$$y = (... (((a_0x + a_1)x + a_2)x + a_3)x + ... + a_{n-1})x + a_n.$$

Для нашего многочлена 4-й степени это дает

$$y = (((a_0x + a_1)x + a_2)x + a_3)x + a_4.$$

Далее вводим обозначения

$$b_0 = 0x + a_0$$
$$b_1 = b_0x + a_1$$
$$\cdots\cdots\cdots\cdots$$
$$b_n = b_{n-1}x + a_n$$

Очевидно, что $y = b_n$.

Итак, вычисление значения многочлена свелось к последовательным вычислениям по формуле

$$b_0 = 0 \cdot x + a_0$$
$$b_i = b_{i-1}x + a_i \qquad i = 1,2,...n.$$

Критерием окончания процесса будет являться получение заданного числа $n + 1 = 5$ циклов вычислительного процесса. Блок-схема программы показана на рис. IV.2.

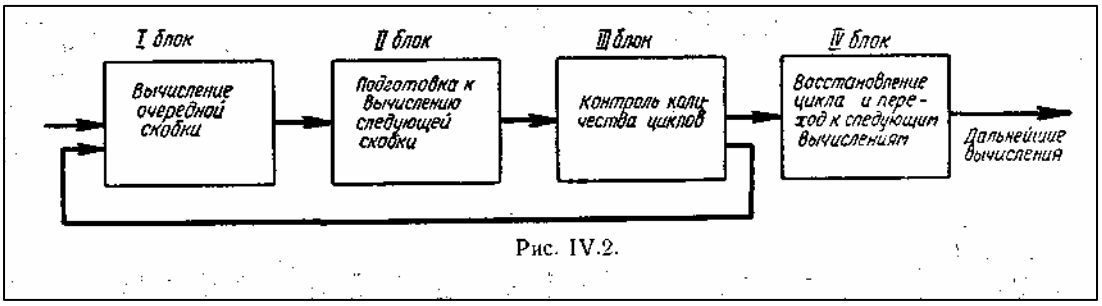

Рис. IV.2.

Размещение данных в памяти показано в табл. IV.9, а программа — в табл. IV.10.

Таблица IV.9

Номера ячеек	0100	0101	0102	0103	0104	0105	0106	0107	0110	0111	.....	0111+n
Данные	$n = 4$	x	0,(*y*)	0,(*i*)	1	*C*	0	*G*	—	a_0	.....	a_n

где *C* — 000000010000000 - единица второго адреса, *G* — 010201110102001.

Таблица IV.10

№№ п/п.	Команды						Пояснения	
1	2	3	4	5	6	7	8	
0020	M	0102	0101	0102	0	05	I блок	$b_{i-1} \cdot x$
0021	C	0102	0111	0102	0	01		$b_{i-1} \cdot x + a_i = b_i$
0022	C_1	0021	0105	0021	0	02	II блок	Преобразование команды $i+1$
0023	C	0103	0104	0103	0	01		
0024	B	0100	0103	0110	0	03	III блок	$n-(i+1)$
0025	E	0020	0026	0000	0	13		
0026	R	0102	a	b	0	—	Использование y для дальнейших вычислений	
0027	Ф	0106	0000	0102	0	07	IV блок	$0 \rightarrow (102)$
0030	Ф	0106	0000	0103	0	07		$0 \rightarrow (103)$
0031	Ф	0107	0000	0021	0	07		Восстановление команды в ячейке 21

Особенностью данной программы является наличие преобразования команд; с каждым циклом увеличивается на единицу второй адрес команды 21. Для этой цели используется команда 22, где C_1 — операция специального сложения.

В качестве специального числа, прибавление которого к коду команды обеспечивает увеличение второго адреса на единицу, принято число C = 000000010000000, т. е. число, характеризующееся наличием единицы на 8-разрядном месте — на месте единиц второго адреса.

Команда 23 служит для увеличения на единицу текущего номера вычисляемого члена полинома. Команды 24 и 25 обеспечивают контроль числа выполненных циклов вычислений, т. е. контроль количества вычисленных членов ряда и передачу управления.

Команда 26 предусматривает использование вычисленного значения *y*.

Команды 27, 30, 31 служат для восстановления команд программы и ячеек памяти в их первоначальном виде, причем изменявшаяся команда 21 восстанавливается путем засылки в ячейку 21 первоначального кода этой команды, хранящегося в ячейке 107.

В приведенных примерах для контроля количества выполненных циклов была отведена специальная ячейка, называемая счетчиком циклов, ее содержимое увеличивалось на единицу с каждым новым выполнением цикла и сравнивалось с некоторым постоянным числом, заданным заранее и определяющим необходимое число повторений циклов.

Часто при программировании циклических процессов можно не вводить специального счетчика циклов, а использовать для этой цели команду, которая подвергается изменению с каждым выполнением цикла. Для этого нужно определить, какой вид должна принять эта команда после повторения необходимого числа циклов, запасти эту команду в ее конечном коде в какой-либо ячейке и сравнивать после каждого повторения цикла текущий код команды с конечным кодом.

Так, в последнем примере можно запасти в ячейке 100 вместо числа $n = 4$ команду 21 в том виде, который она будет иметь после выполнения 5-го цикла (010201160102001), и сравнивать с этим значением текущий код команды 21.

Программа при этом примет вид, показанный в табл. IV.11.

Таблица IV.11

№№ п.п.	Команды						Пояснения
1	2	3	4	5	6	7	8
0020	M	0102	0101	0102	0	05	—
0021	C	0102	0111	0102	0	01	—
0022	C_1	0021	0105	0021	0	02	—
0023	C_p	0100	0021	0110	0	12	—
0024	E	0025	0020	0000	0	13	—
0025	R	0102	a	b	0	—	—
0026	Ф	0106	0000	0102	0	07	—
0027	Ф	0107	0000	0021	0	07	—

Данный вариант программы значительно выгоднее предыдущего, так как он содержит на две команды меньше, и кроме того, при этом освобождается ячейка 103.

Часто помимо основного цикла вычислений, характеризующего собой принятый численный метод решения, могут быть построены в программе еще внутренние циклы для выполнения отдельных участков вычислений внутри основного цикла, а внутри внутренних циклов могут быть еще циклы и т. д.

Развернутая программа (циклического вычислительного процесса) имеет значительный объем и требует большего труда для своего составления и большего времени для ввода в машину (внешние устройства работают у

большинства машин медленно). Зато при работе по такой программе машина выполняет лишь самые необходимые для решения задачи операции.

При циклической программе, кроме вычислительных операций, машина производит операции для модификации и восстановления команд и контроля числа циклов. Машинное время, потребное для решения задачи по такой программе, больше чем для решения по развернутой программе. Но разница эта делается тем меньше, чем больше вычислений входит в один цикл. Количество труда, необходимого для составления циклической программы, объем этой программы и время, затрачиваемое на ввод ее в машину, значительно меньше, чем для развернутой программы, поэтому при программировании задач, за очень редким исключением, составляют циклические программы.

4. Некоторые приемы построения разветвлений в программах

При рассмотрении циклических программ мы уже встретились с двумя способами построения разветвлений в программах:

а) построение разветвлений при помощи специального счетчика циклов, т. е. отдельной ячейки, содержимое которой увеличивалось на единицу с каждым выполнением цикла;

б) построение разветвлений с использованием в качестве счетчика циклов какой-либо команды или величины, изменяемой стандартным образом при каждом выполнении цикла.

Эти приемы, вообще говоря, являются основными и широко используются при программировании задач.

В настоящем разделе мы рассмотрим еще некоторые употребительные приемы построения разветвлений.

П р и м е р 1.

Пусть требуется построить такую программу, в которой после выполнения некоторой последовательности команд A нужно *a* раз выполнить последовательность команд B и *m* раз последовательность команд C, затем перейти к выполнению некоторой команды L.

Естественно, что при многократном повторении блоков *B* и *C* внутри них будет производиться переадресация и восстановление команд. Характер этих действий зависит от конкретного содержания указанных блоков команд и нами в данном случае не рассматривается.

Порядок выполнения необходимых переходов между блоками представлен на рис. IV.3 в виде блок-схемы.

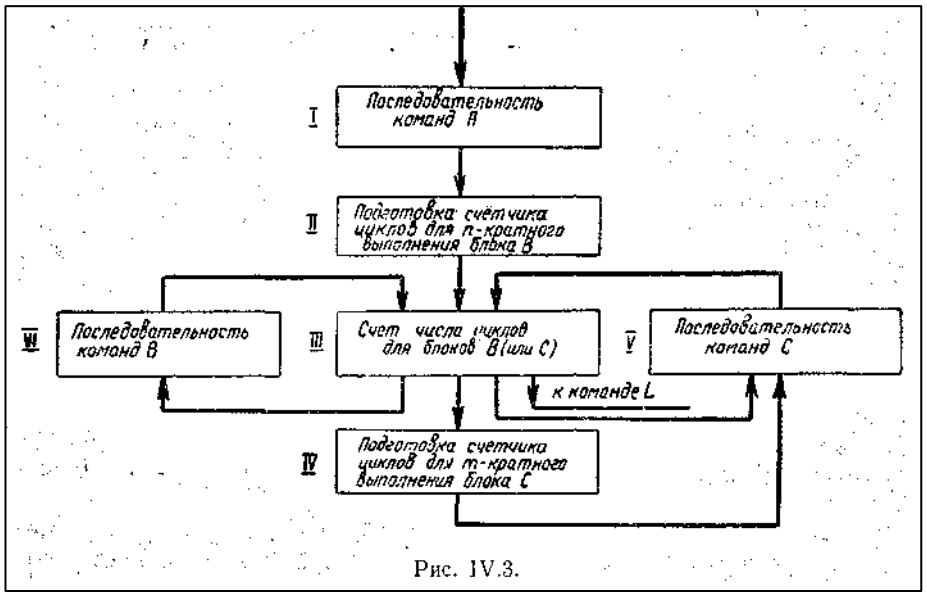

Рис. IV.3.

В табл. IV.12 приведена в условных буквенных адресах программа, реализующая рассматриваемый случай разветвления.

Таблица IV. 12

№№ п/п.	Команды						Пояснения
1	2	3	4	5	6	7	8
K_1 ⋮ K_2	Последовательность команд A						Блок I
K_2+1	Ф	$<n>$	0000	d	0	07	Блок II
K_2+2	Ф	$a+1$	0000	K_2+4	0	07	
K_2+3	B	d	$<1>$	d	0	03	Блок III
K_2+4							
K_2+5	Ф	$<m-1>$	0000	d	0	07	Блок IV
K_2+6	Ф	$a+2$	0000	K_2+4	0	07	
K_2+7 ⋮ K_3	Последовательность команд C						Блок V
K_3+1	E	K_2+3	K_2+3	0000	0	13	
K_3+2 ⋮ K_4	Последовательность команд B						Блок VI
K_4+1	E	K_2+3	K_2+3	0000	0	13	

В ячейках $a+1$ и $a+2$ находятся коды команд. $(a+1) = E, K_3 + 2, K_2+5, 0000$. $(a+2) = E, K_2 + 7, L, 0000$.

Пример 2.

Пусть требуется построить такой участок программы, в котором после выполнения некоторой последовательности команд A нужно переходить поочередно то к выполнению команды L, то к выполнению команды M. Такой переход может быть построен в виде табл. IV.13. В ячейку a первоначально засылается любое положительное число. При каждом выполнении команды (B, 0000, a, a) знак результата этой операции меняется, и последующая команда E передает управление то команде L, то команде M.

Таблица IV. 13

№№ п. п.	Команды						Пояснения
1	2	3	4	5	6	7	8
m_1 . m_2	Последовательность команд A						—
m_2+1	B	0000*)	a	a	0	03	—
m_2+2	E	M	L	0000	0	13	—

* Напомним, что в нулевой ячейке в рассматриваемой нами машине постоянно хранится число нуль.

Пример 3.

При программировании сложных переходов и разветвлений часто пользуются так называемым приемом «шкалы и щупа»*. Пусть, например, выполнение некоторой последовательности команд A заканчивается переходом к команде L каждый раз в течение первых пяти повторений этой последовательности A, затем в течение восьми раз выполнение последовательности команд A заканчивается переходом к команде M, затем в течение шести раз — к команде L, а затем в течение четырех раз к команде M. После этого весь цикл переходов повторяется сначала и так n раз.

Для программирования такого рода разветвлений поступают следующим образом.

В некоторую фиксированную ячейку памяти a помещается специальное двоичное число, в котором распределение единиц и нулей в разрядах соответствует порядку переходов в данном разветвлении. Для нашего случая это число, называемое «шкалой», будет иметь вид:

* Идея этого приема принадлежит М. Р. Шура-Бура. 148

(a) =

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	…….	42
1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	Любые значения	

В другую ячейку памяти b помещается специальное двоичное число, называемое «щупом» и имеющее единицу в нулевом разряде и нули во всех остальных разрядах. Это число будет иметь вид

(b) =

0	1	2	3	4	5	……	42
1	0	0	0	0	0	0 0 0 0 ……	0

После первого выполнения последовательности команд А с помощью щупа выявляется наличие или отсутствие единицы в самом крайнем (нулевом) разряде шкалы, т. е. производится поразрядное логическое умножение чисел, представляющих щуп и шкалу.

При наличии единицы в проверяемом разряде (результат операции не равен нулю) управление передается команде L; при отсутствии единицы (результат операции равен нулю) управление передается команде М. После этого число, представляющее собой щуп, сдвигается на один разряд вправо, в результате чего при следующем выполнении последовательности команд А и выборе направления вычислений, щуп будет проверять наличие единицы в следующем (первом) разряде шкалы и в зависимости от этого осуществлять переход к команде L или М.

Таким образом повторяется весь предусмотренный шкалой цикл переходов. Если требуется повторение всего цикла n раз, то каждый раз после окончания цикла число, представляющее собой щуп, сдвигается влево в исходное положение; n-кратное повторение всего цикла осуществляется при помощи циклических программ обычного типа, примеры которых были рассмотрены выше.

13. ОСОБЕННОСТИ РЕШЕНИЯ МАТЕМАТИЧЕСКИХ ЗАДАЧ НА ЭЛЕКТРОННЫХ ЦИФРОВЫХ МАШИНАХ

1. Контроль вычислений

Контроль вычислений включает в себя, в общем случае, проверку правильности и точности вычислений. Машина может производить вычисления правильно, но с различной точностью. Точность вычислений зависит от количества разрядов, используемых для представления чисел в машине, от выбранного численного метода решения, от характера задачи.

Вопросы оценки точности вычислений на электронных счетных машинах в настоящее время приобрели большое значение и составляют целый раздел численного анализа, значительно развившийся в последние годы.

В настоящей работе эти вопросы рассматриваться не будут, так как они непосредственного отношения к технике программирования и использованию машин не имеют.

Рассмотрим вопросы контроля правильности вычислений. Контроль правильности вычислений включает в себя проверку правильности составления и введения в машину программы вычислений и контроль правильности выполнения программы машиной.

Процесс решения любой задачи на машине складывается из двух этапов:

1. Проверка и отладка программы;
2. Рабочий счет по проверенной программе.

Второй этап, т. е. собственно решение задачи на машине, начинается только после того, как программа полностью проверена

и отлажена; в процессе рабочего счета на машине проверяется только правильность работы самой машины.

Во всех случаях работы машины, как для отладки программы, так и для рабочего счета, машина предварительно тщательно проверяется и пускается в работу только в исправном состоянии.

В процессе работы в машине могут быть как систематические, так и случайные сбои, нарушающие правильность вычислений. Таким образом, контроль правильности вычислений включает в себя три вопроса:

1) проверка исправности машины перед началом счета и обнаружение систематических ошибок;
2) проверка программы и обнаружение и исправление ошибок в программе, так называемая отладка программы;
3) контроль правильности работы машины в процессе рабочего счета и обнаружение случайных ошибок.

Рассмотрим в общих чертах перечисленные вопросы контроля правильности вычислений.

Проверка исправности машины и обнаружение систематических ошибок осуществляется методами профилактического контроля путем задания отдельным устройствам машины и машине в целом утяжеленных режимов работы, а также при помощи специальных контрольных программ, так называемых тест-программ, пропускаемых перед решением основных задач.

Составление достаточно полной системы тестов, обеспечивающей надежную проверку всех устройств машины и в то же время не являющейся слишком громоздкой, представляет собой достаточно сложную задачу,

требующую не только хорошего знания конструкции машины, но и теоретических исследований.

После того как машина проверена и установлено, что она находится в исправном состоянии, приступают к проверке и отладке программы.

Проверка правильности программы может производиться следующими способами:

1. Сравнением значений величин, получаемых в результате вычислений по программе с заранее известными контрольными значениями этих величин. Этот прием является основным приемом проверки правильности программы в процессе ее отладки. Предварительно путем ручных вычислений определяется ряд контрольных значений искомых величин для ряда моментов вычислительного процесса. Заметим, что даже для сложных и больших задач такой ручной подсчет одного варианта обычно не является особенно трудоемким, так как для такого расчета исходные данные и коэффициенты могут быть заменены нулями и единицами, что значительно облегчает ручной счет.

Проверяемую программу вводят в машину вместе с измененными таким же образом исходными данными и коэффициентами, производят пробный счет и сравнивают получающиеся результаты с результатами ручного счета. Иногда нет необходимости в предварительном ручном счете, если для данной задачи известны из каких-нибудь других источников (например, по результатам опытов) контрольные значения функций.

При проверке правильности программы указанным способом нельзя требовать абсолютного совпадения вычисляемых по программе величин с контрольными значениями, так как между ними неизбежны расхождения, обусловленные влиянием ошибок округления, различным количеством разрядов, используемых при ручных и машинных вычислениях, и другими причинами. Учитывая точность, с которой известны контрольные значения, выбираются допустимые величины отклонений между контрольными и вычисляемыми по программе значениями.

2. Введением в исходную систему уравнений задачи дополнительных соотношений между искомыми и заданными величинами таким образом, чтобы при правильном решении задачи это соотношение являлось тождеством.

Например, при интегрировании уравнений движения снаряда, получив значение скорости v, а также значения ее горизонтальной (u) и вертикальной (w) проекции, можно составить контрольное соотношение

$$v = \sqrt{u^2 + w^2} ,$$

сравнивая правую и левую части которого, можно судить о правильности вычислений.

При интегрировании систем обыкновенных дифференциальных уравнений можно ввести дополнительную переменную, представляющую собой известную функцию других переменных, и составить для нее дополнительное дифференциальное уравнение.

Вычисляя эту функцию, с одной стороны, по полученным в результате решения задачи значениям других переменных и, с другой стороны, путем интегрирования дополнительного дифференциального уравнения и сравнивая полученные значения, можно судить о правильности вычислений.

Например, при интегрировании уравнений динамики в качестве такой дополнительной функции можно использовать кинетическую или полную энергию системы.

3. Повторением вычислений одних и тех же функций по различным программам, отличающимся либо составом, либо последовательностью применяемых команд.

Полученные значения функции сравниваются между собой, и если они отличаются на большую величину, чем заданная заранее допустимая величина ошибки, то это свидетельствует о нарушении правильности процесса вычисления. Этот прием является менее выгодным, чем предыдущие, так как требует лишней работы по составлению второй программы, однако, в отдельных случаях это может оказаться проще, чем получение контрольных значений величин ручным счетом или каким-либо другим путем.

Перечисленные способы проверяют, вообще говоря, не только правильность составления программы, но и правильность выбранного численного метода решения задачи, правильность принятых расчетных формул. Поэтому эти способы могут использоваться не только при проверке и отладке программ, но и при проверке рабочего счета в тех случаях, когда возникает сомнение в применимости выбранных методов решения и расчетных формул для тех или иных областей изменения участвующих в вычислениях величин.

Приведенные способы обеспечивают, вообще говоря, только обнаружение самого факта наличия ошибок в программе, но не позволяют установить место и характер ошибок, на что потребуется еще дополнительная, достаточно кропотливая работа по анализу программы.

Для полной проверки программ с указанием места и характера ошибок разработаны специальные проверочные программы.

Сущность работы проверочной программы сводится к следующему.

Проверочная программа поочередно выбирает команды проверяемой программы, переносит их в определенные фиксированные ячейки памяти, анализирует (определяет код) и выполняет их. Результат выполнения некоторой группы команд, взятых из проверяемой программы, автоматически сравнивается с контрольным значением, заранее рассчитанным вручную и внесенным в память.

В случае совпадения значений проверочная программа переходит к проверке следующей группы команд проверяемой программы.

В случае несовпадения проверочная программа выдает на печать номера первой и последней команд проверяемой группы, контрольное и вычисленное значения результата и, внеся в соответствующую ячейку памяти вместо вычисленного (неправильного) значения контрольное (правильное) значение, переходит к проверке группы следующих команд.

Таким образом, в результате работы проверочной программы будут выявлены все группы команд проверяемой программы, в которых произошло несовпадение контрольных и вычисленных значений, что позволит сравнительно просто обнаружить ошибку в программе. Эта ошибка будет обнаружена сразу, если она относится к командам, выполняющим арифметические действия (достаточно внимательно прочитать эти команды). Если же ошибка допущена в командах переадресации или в логических условиях, то обнаружение ее потребует некоторого анализа выданного из машины контрольного материала.

Перейдем к рассмотрению вопроса о методах контроля правильности работы машины в процессе рабочего счета. Как уже упоминалось выше, при решении этого вопроса следует исходить из двух условий:

а) машина в начальный момент работы является совершенно исправной;

б) программа полностью проверена и не содержит ошибок. Задача состоит в том, чтобы постоянно в процессе рабочего счета контролировать правильность работы машины с целью выявления случайных ошибок.

Основным методом контроля правильности вычислений при этих условиях является так называемый метод контрольного суммирования. Этот метод применяется для контроля правильности ввода программы в машину, правильности выполнения вычислений по отдельным участкам программы и всей программе в целом, правильности передачи данных между запоминающими устройствами. Сущность этого метода заключается в следующем: после выполнения определенной части (или иногда всех) вычислений в программе должно быть предусмотрено суммирование некоторых групп чисел, или даже всех чисел, находящихся в памяти. При этом суммируются не только обычные числа, представляющие собой величины, участвующие в задаче, но и коды команд, специальные и условные числа. Для выполнения такого формального суммирования используется специальная команда контрольного суммирования, приведенная в системе команд. Полученное значение контрольной суммы запоминается, после чего весь предыдущий этап расчета повторяется с начала до того момента, когда была получена контрольная сумма. По достижении этого момента вторично производится контрольное суммирование и два значения контрольной суммы сравниваются для проверки их совпадения.

В случае их совпадения этап расчета считается правильным и машина переходит к следующему этапу вычислений, в случае их несовпадения производится повторный расчет предыдущего этапа и получается третье значение контрольной суммы, которое сравнивается с предыдущими двумя значениями. Если последняя контрольная сумма совпала с одной из предыдущих, то последний расчет считается правильным и происходит переход к следующим этапам вычислений.

Если же третье значение контрольной суммы не совпало ни с одним из двух предыдущих, то расчет считается неправильным. Для такого случая в программе должна быть предусмотрена остановка машины.

После такой остановки необходимо при помощи специальных тест-программ проверить машину, выяснить и устранить неисправности и затем повторить расчет.

Для того чтобы не повторять расчет с самого начала, в процессе вычислений предусматриваются периодические выводы во внешний накопитель всей информации, находящейся в памяти, вместе с соответствующими контрольными суммами, подсчитанными для этих моментов. При обнаружении ошибки расчет повторяется только начиная с того места, в котором последний раз было получено правильное значение контрольной суммы.

Иногда в программе предусматривается прекращение расчета сразу после обнаружения несовпадения двух контрольных сумм, не получая третьей. Это бывает целесообразно при решении таких сравнительно небольших задач, решаемых за короткое время, как приведенная ниже задача о перемножении матриц, когда вероятность исчезновения случайной причины сбоев машины в течение короткого промежутка времени невелика.

Последнее соображение говорит о том, что контрольное повторное суммирование целесообразно производить лишь после сравнительно больших этапов вычислений.

При помощи контрольного суммирования проверяется также ввод программы и исходных данных в машину. Для этого ввод информации производится дважды с выполнением контрольного суммирования всего содержимого памяти после первого и после повторного ввода и со сравнением полученных контрольных сумм.

При переписи данных из памяти во внешний накопитель для каждой переписываемой группы чисел также подсчитывается значение контрольной суммы, которое также запоминается вместе с этой группой чисел. Для проверки правильности переписи переписанные данные вводятся из внешнего накопителя обратно в память, и для них снова подсчитывается контрольная сумма.

В случае совпадения значений двух контрольных сумм перепись считается выполненной правильно. Также проверяется правильность передачи данных из накопителя в память, из одного места памяти в другое, выдача данных в выходное устройство (при помощи повторного ввода).

Достоинством метода контрольного суммирования является надежность проверки правильности вычислений, так как возможность получения одновременно двух взаимно компенсирующихся ошибок маловероятна. Особенно ценен метод контрольного суммирования при проверке действий с большим количеством данных, так как при этом не требуется сохранять все данные, полученные при первом расчете, для сравнения их с данными повторного расчета. Достаточно сохранить лишь одно число — контрольную сумму, что обеспечивает значительную экономию емкости памяти.

2. Выбор численных методов решения

Появление быстродействующих электронных счетных машин оказывает существенное влияние на развитие численных методов решения задач. К численным методам, используемым для вычислений на быстродействующих машинах, предъявляются некоторые принципиально новые требования.

Общий подход при этом характеризуется стремлением использовать наиболее простые и однотипные расчетные формулы, а требуемую точность обеспечить за счет увеличения объема вычислений. Например, при численном интегрировании обыкновенных дифференциальных уравнений необходимая точность получается путем уменьшения шага аргумента, а в случае уравнений с частными производными — за счет уменьшения шага сетки.

Однако, несмотря на большие возможности электронных счетных машин, такой подход к выбору численных методов при решении особенно сложных и больших по объему вычислений задач встречает два ограничения.

Во-первых, при значительном увеличении объема вычислений существенную роль начинают играть ошибки округления. Хотя в быстродействующих машинах специально для уменьшения влияния ошибок округления предусматривается значительно большее количество разрядов, чем обычно применяется в ручных расчетах, все же при больших по количеству операций вычислениях и большое количество разрядов может оказаться недостаточным. Для ограничения ошибок округления придется применять более крупный шаг интегрирования и более совершенные расчетные формулы.

Второе ограничение связано с тем, что значительное измельчение шагов сетки, например, при решении сложных задач математической физики приводит к резкому увеличению объема численного материала, который должен храниться в памяти машины, имеющей ограниченную емкость. При этом играет роль также и увеличение машинного времени, имеющее место при решении сложных задач с помощью простейших формул.

Известны случаи, когда отдельные исключительно большие по объему вычислений задачи требовали для своего решения нескольких дней непрерывной работы машины.

В связи с этим можно ожидать, что развитие и применение быстродействующих счетных машин, а также непрерывное усложнение задач, предъявляемых к решению на машинах, потребует разработки новых более сильных методов численного решения задач, которые бы сочетали в себе удобство программирования с высокой точностью вычислений.

Рассмотрение принципов построения и действия автоматической цифровой вычислительной машины позволяет сформулировать некоторые общие требования к численным методам решения задач на этих машинах.

1. Метод должен обеспечивать циклический ход решения задачи. Это означает, что процесс решения должен сводиться к многократному повторению расчетов по одним и тем же формулам с подстановкой в эти формулы при каждом повторном цикле новых числовых данных. Циклический характер вычислительного процесса позволяет, как это было показано выше, построить циклическую изменяемую программу, значительно меньшую по объему, чем эквивалентная ей развернутая. Компактность программ обеспечивает уменьшение загрузки памяти и сокращение машинного времени на ввод программы. Компактные циклические программы сокращают также объем ручной подготовительной работы по оформлению программ и подготовке их к вводу в машину; при этом резко уменьшается вероятность ошибок в программе, связанных с техническим оформлением (опись, пропуски и т. п.). Однако в циклических программах, благодаря усложнению их логической структуры, чаще, чем в развернутых программах, бывают ошибки принципиального характера (неправильная переадресация, отсутствие восстановления и т. д.).

2. Метод должен сводить решение к элементарным операциям. Например, методы, дающие решения в виде тригонометрических рядов, будут являться нерациональными, так как частая операция обращения к таблицам тригонометрических функций может значительно увеличить время решения задачи.

Малая эффективность введения в машину таблиц каких-либо функций объясняется тем, что если значения функций хранятся во внутренней памяти машины, эта память, обладающая ограниченной емкостью, оказывается перегруженной. Если же значения функций хранятся во внешнем накопителе совместно со многими другими данными, то на считывание функций уходит много времени и общая высокая скорость работы машины не используется полностью.

3. Метод должен обеспечивать возможность контроля правильности решения задачи. Желательно, чтобы формулы, дающие точное решение данной конкретной задачи могли быть продублированы или же дополнены приближенными формулами.

4. При выборе метода решения задачи необходимо обращать внимание на обеспечение благоприятных условий для повышения точности работы машины. В этом отношении нежелательны методы, использующие многократное сложение чисел, сильно отличающихся друг от друга своими порядками, и многократное вычитание близких по величине чисел.

б. Весьма удобными для использования на электронных цифровых машинах являются методы, обеспечивающие самоисправляемость ошибок. Таким свойством обладают итерационные методы. При этом появление случайной ошибки в вычислениях часто совершенно не оказывает влияния на точность окончательных результатов, а лишь приводит к увеличению числа итераций, необходимых для достижения заданной точности решения. Однако этот вопрос должен особо исследоваться в каждом конкретном случае, так как иногда слишком большая случайная ошибка может увести итерационный процесс в сторону, в результате чего будет получено не искомое решение, а другое (из числа возможных).

Численные методы решения математических задач с точки зрения их использования на автоматических цифровых вычислительных машинах могут быть разделены на три группы.

1. Прямые методы, позволяющие получить окончательное решение задачи путем определенной постоянной последовательности вычислений через заранее известное количество шагов вычислений. Сюда относится решение линейных алгебраических уравнений методом исключения, вычисление определенных интегралов, решение обыкновенных дифференциальных уравнений и др.

2. Методы последовательных приближений (итерационные), при которых заранее неизвестно потребное количество шагов вычислений.

Эти методы вследствие присущей им цикличности процесса особенно пригодны для использования на машине при условии, что применяемый итерационный процесс достаточно быстро сходится.

Процесс вычислений прекращается тогда, когда разность между двумя последовательными решениями становится меньше наперед заданного малого числа.

3. Вероятностные методы, известные под названием

методов Монте—Карло, при которых решение задачи заменяется определением математического ожидания ряда случайных величин и определением законов их распределения.

При этих методах с помощью электронной цифровой машины моделируется тот случайный процесс, который описывается решаемыми уравнениями.

Методы вероятностные в настоящее время усиленно разрабатываются применительно к электронным счетным машинам.

Следует заметить, что для обеспечения возможности применения электронных счетных машин для решения задач вероятностными методами эти машины должны обладать возможностью вырабатывать случайные числа или функции. Это реализуется в машинах либо заданием таблицы случайных чисел, либо введением в состав машины дополнительного устройства — датчика случайных чисел.

В последнее время стали использоваться при машинных вычислениях так называемые псевдослучайные числа, которые вырабатываются самой машиной по специальной программе с помощью небольшой исходной таблицы случайных чисел.

3. Введение в машину заданных функций

Вопрос о вводе в машину заданных функций имеет большое значение для эффективности решения задач на машине.

В большинстве электронных цифровых машин не предусматривается возможность непосредственного введения в машину графиков функций; использование больших таблиц функций в машинах оказывается не вполне эффективным. Эти таблицы функций не могут быть помещены во внутреннюю память ввиду ее ограниченной емкости, а выборка из таблиц, помещенных во внешнем накопителе, значительно увеличивает время решения.

Наиболее удобным с точки зрения простоты программирования является представление функций аналитическими зависимостями, например, в виде степенных рядов или непрерывных дробей. При этом удобны для использования лишь быстро сходящиеся степенные ряды и непрерывные дроби при непременном условии, что связанные с их применением промежуточные вычисления осуществляются просто и единообразно. Для вычисления значений функций широко применяются итерационные методы, например, такие, как метод Ньютона для вычисления функций, заданных алгебраическими уравнениями.

В случае необходимости введение в машину таблиц функций может быть осуществлено следующими приемами:

1. Большие по объему таблицы функций записываются на магнитные ленты или барабаны внешних накопителей в виде отдельных групп чисел. Каждая группа может содержать либо всю таблицу, либо ее часть в зависимости от объема таблицы. Для обращения к таблице, размещенной во внешнем накопителе, в программе заблаговременно ставится команда, обеспечивающая установку в рабочее положение нужной зоны накопителя. При обзоре таблица передается в память машины группами и обзор группы, переданной в память, выполняется по специальной подпрограмме. Использование таблиц функций, записанных во внешнем накопителе, сопряжено с большой потерей времени и является наименее желательным приемом введения функций.

2. Небольшие по объему таблицы могут быть записаны непосредственно в память машины.

Обзор таблицы, записанной в памяти, осуществляется при помощи подпрограммы; поиск нужного значения функции может происходить путем контроля знака разности текущего и заданного значения аргумента. Перемена знака служит сигналом для передачи найденного значения функции в фиксированную ячейку памяти.

Для вычисления табличных функций во внешний накопитель или в память машины вводятся, помимо таблиц, интерполяционные коэффициенты для вычисления промежуточных значений функций и программы интерполяционных вычислений функций для заданного порядка интерполяции.

С целью уменьшения количества данных, вводимых в память машины, целесообразно пользоваться более редкой сеткой таблицы и соответственно интерполяционными формулами более высоких порядков.

Если значения аргумента изменяются с постоянным шагом и таблица располагается в памяти в последовательно возрастающих ячейках, то может быть составлена программа, которая обеспечит непосредственное нахождение функции по заданному значению аргумента. Наиболее просто выполняется выборка из таблиц в том случае, когда текущие значения аргумента совпадают с табличными.

К у с о ч н о - п о л и н о м и а л ь и а я а п п р о к с и м а ц и я ф у н к ц и й

Рациональным способом введения в машину функций, задаваемых таблицами или графиками, является представление их в виде аналитических зависимостей, точно или приближенно соответствующих данным функциям.

Во многих случаях удобно применять для этой цели так называемую кусочно-полиномиальную аппроксимацию, т. е. представление функций полиномами по участкам изменения независимого переменного.

Коэффициенты полиномов должны быть вычислены и введены заранее в память машины, после чего вычисление значений функций производится в машине автоматически. При этом получается большая экономия места в памяти машины, так как вместо всей таблицы в нее заносится небольшое количество коэффициентов аппроксимирующих полиномов.

Помимо аппроксимации экспериментальных функций, полученных в виде таблицы или графиков, зачастую для введения в машину может оказаться удобной также замена различных трансцендентных и специальных функций приближенными аналитическими

Выражениями, с помощью которых вычисление на машине осуществляется проще.

Эта замена может быть эффективно применена, если известен интервал изменения аргумента и требуемая точность представления функций на этом интервале.

Иногда при решении систем обыкновенных дифференциальных уравнений замена отдельных аналитических функций ($\sin x$, e^x, $\ln x$ и др.) определяющими их дифференциальными уравнениями (уравнениями, решением которых являются эти функции) может упростить вычисления.

14. ПРИМЕРЫ ПРОГРАММ

П р и м е р 1

Программа для перемножения двух квадратных матриц порядка *n*

Как известно, элементы матрицы произведения C двух квадратных A и B определяются по формуле

$$c_{ik} = \sum_{j=1}^{n} a_{ij} \times b_{jk} \quad (i,\ k = 1, 2, \ldots n).$$

Процесс вычислений распадается на три циклических процесса в соответствии с наличием трех переменных индексов i, j, k. Первый внутренний цикл соответствует процессу вычисления отдельного элемента матрицы; второй

цикл, включающий в себя первый цикл, соответствует процессу вычисления всех элементов очередной строки и третий, внешний цикл, соответствует вычислению всех строк матрицы.

Считаем, что емкость памяти достаточно велика, чтобы разместить полностью две исходные матрицы, матрицу — произведение, программу и все вспомогательные величины.

Наметим план размещения исходных данных и результатов вычислений в памяти машины.

Элементы матрицы A располагаются последовательно по строкам в ячейках с номерами от a+l до $a+n^2$.

Элементы матрицы B располагаются последовательно по столбцам в ячейках с номерами от b+1 до $b+n^2$ и элементы матрицы C [будут располагаться последовательно по срокам в ячейках с номерами от c+1 до $c+n^2$.

Разница в порядке размещения элементов исходных матриц A и B связана, как это будет видно из дальнейшего, с некоторым упрощением порядка преобразования адресов команд.

Команды программы будем располагать последовательно в ячейках с номерами K+1, K+2... План размещения величин во внутренней памяти приведен в табл. IV. 14, а программа решения задачи — в табл. IV. 15. Блок-схема программы показана на рис. IV.4

В программе не указаны команды, относящиеся к переводу исходных данных в двоичную систему и размещению этих данных в памяти машины, так как эти команды будут составлять отдельную программу ввода. Составление программы ввода весьма просто и мы на этом останавливаться не будем. этом останавливаться не будем.

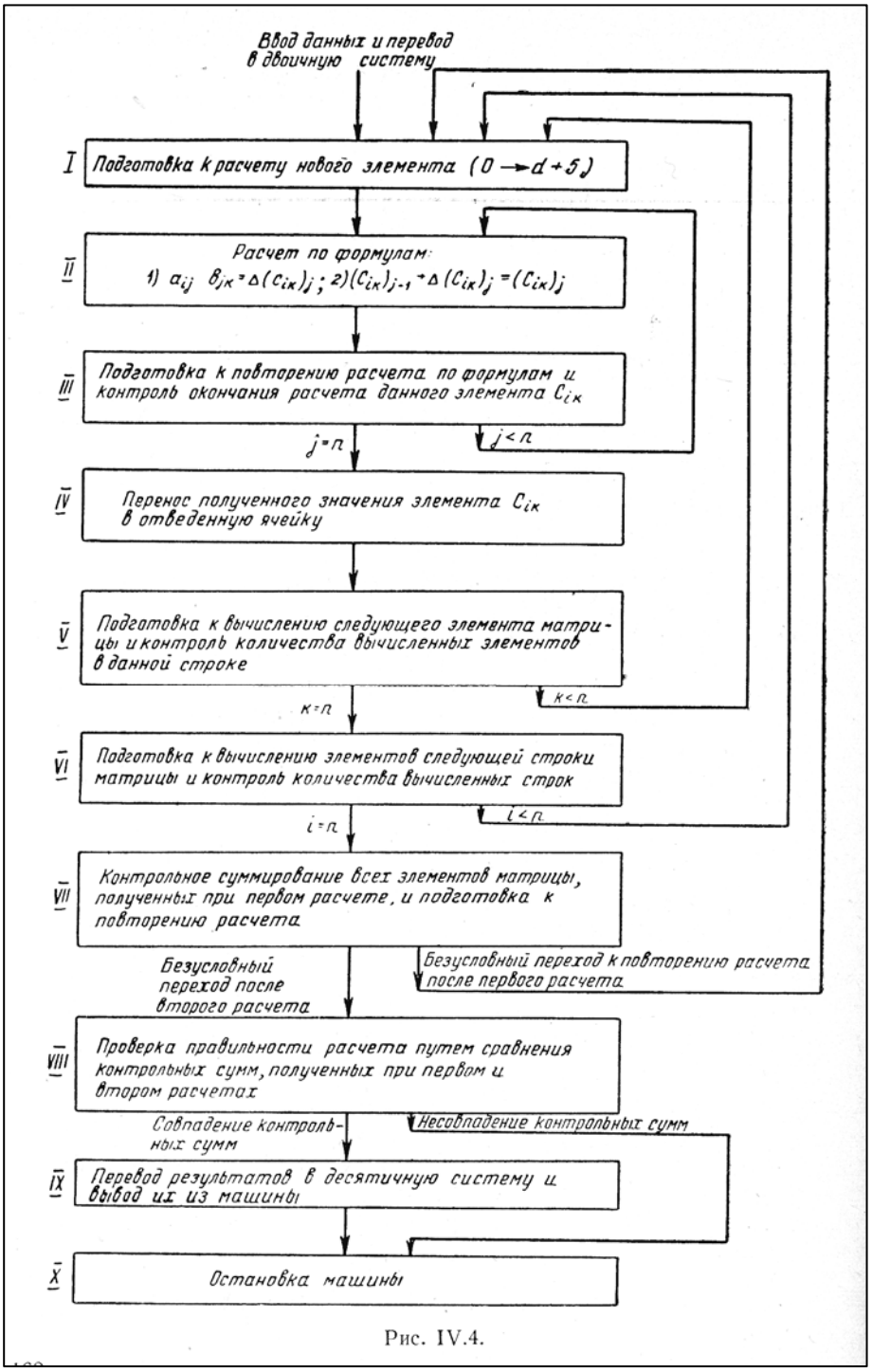

Рис. IV.4.

Таблица IV.14

План размещения величин во внутренней памяти машины

Ячейки	Величины	Пояснения
$d+1$	0	Постоянные величины
$d+2$	1	
$d+3$	n	
$d+4$	$\Delta(c_{ik})_j$	Рабочие ячейки, используемые в процессе вычислений для хранения промежуточных результатов
$d+5$	$(c_{ik})_j$	
$d+6$	$i_0=0$	Изменяемые параметры. В начальный момент (при вводе данных в машину) эти параметры равны нулю
$d+7$	$j_0=0$	
$d+8$	$k_0=0$	
$d+9$	0001 0001 0000 000	Условные числа для изменения и восстановления адресов в командах
$d+10$	0000 0000 0001 000	
$d+11$	(0001 0000 0000 000)$\cdot n$	
$d+12$	(0000 0001 0000 000)$\cdot n^2$	
$d+13$	(0001 0000 0000 000)$\cdot n^2$	
$d+14$	(0000 0000 0001 000)$\cdot n^2$	
$d+15$	1	Единица, необходимая для контроля количества повторений расчетов
$d+16$	—	Ячейки для хранения контрольных сумм, полученных при первом и втором расчетах
$d+17$	—	
$a+1$	a_{11}	Элементы a_{ij} матрицы A. Зависимость адресов элементов матрицы A от параметров: $<a_{ij}>=a+j+n(i-1)$
$a+2$	a_{12}	
$a+n$	a_{1n}	
$a+n+1$	a_{21}	
$a+n^2$	a_{nn}	
$b+1$	b_{11}	Элементы b_{jk} матрицы B. Зависимость адресов элементов матрицы B от параметров: $<b_{jk}>=b+j+n(k-1)$
$b+2$	b_{21}	
$b+n$	b_{n1}	
$b+n+1$	b_{12}	
$b+n^2$	b_{nn}	
$c+1$	c_{11}	Элементы c_{ik} матрицы C. Зависимость адресов элементов матрицы C от параметров: $<c_{ik}>=c+k+n(i-1)$
$c+2$	c_{12}	
$c+n$	c_{1n}	
$c+n+1$	c_{21}	
$c+n^2$	c_{nn}	

Таблица IV.15

Программа для перемножения двух **квадратных матриц**

№№ п.п.	Команды						Пояснения
1	2	3	4	5	6	7	8
$K+1$	Ф	$d+1$	$d+1$	$d+5$	0	07	I блок
$K+2$	М	$a+1$	$b+1$	$d+4$	0	05	II блок
$K+3$	С	$d+5$	$d+4$	$d+5$	0	01	
$K+4$	С	$d+7$	$d+2$	$d+7$	0	01	III блок
$K+5$	C_1	$K+2$	$d+9$	$K+2$	0	02	
$K+6$	Ср	$d+7$	$d+3$	0000	0	12	
$K+7$	Е	$K+8$	$K+2$	0000	0	13	
$K+8$	Ф	$d+5$	$d+1$	$c+1$	0	07	IV блок
$K+9$	C_1	$K+8$	$d+10$	$K+8$	0	02	V блок
$K+10$	С	$d+8$	$d+2$	$d+8$	0	01	
$K+11$	B_2	$K+2$	$d+11$	$K+2$	0	11	
$K+12$	Ф	$d+1$	$d+1$	$d+7$	0	07	

№№ п.п.	Команды						Пояснения	
1	2	3	4	5	6	7	8	
$K+13$	Ср	$d+8$	$d+3$	0000	0	12		
$K+14$	Е	$K+15$	$K+1$	0000	0	13		
$K+15$	С	$d+6$	$d+2$	$d+6$	0	01		
$K+16$	$С_1$	$K+2$	$d+11$	$K+2$	0	02		
$K+17$	Ф	$d+1$	$d+1$	$d+8$	0	07	VI блок	
$K+18$	$В_2$	$K+2$	$d+12$	$K+2$	0	11		
$K+19$	Ср	$d+6$	$d+3$	0000	0	12		
$K+20$	Е	$K+21$	$K+1$	0000	0	13		
$K+21$	Ф	$d+1$	$d+1$	$d+6$	0	07		
$K+22$	$В_2$	$K+2$	$d+13$	$K+2$	0	11		
$K+23$	$В_2$	$K+8$	$d+14$	$K+8$	0	11		
$K+24$	К	$c+1$	n^2	$d+16$	0	34	VII блок	
$K+25$	$С_1$	$K+24$	$d+10$	$K+24$	0	02		
$K+26$	В	$d+15$	$d+2$	$d+15$	0	03		
$K+27$	Е	$K+1$	$K+28$	0000	0	13		
$K+28$	Ср	$d+16$	$d+17$	0000	0	12	VIII блок	Сравнение результатов двух расчетов
$K+29$	Е	$K+30$	$K+32$	0000	0	13		
$K+30$	$Пр_2$	$c+1$	n^2	$c+1$	0	33	IX блок	Перевод из двоичной системы в двоично-десятичную и вывод из машины
$K+31$	$П_4$	$c+1$	n^2	0000	0	21		
$K+32$	Я	0000	0000	0000	0	15	X блок	Остановка

Следует сделать пояснения относительно зависимости адресов команд $K+2$ и $K+8$ от переменных параметров *i, j, k.* Команда $K+2$ определяет умножение двух очередных элементов a_{ij} и $b_{jк}$ матриц А и В и направление результата в постоянную ячейку d+4. В ней изменяются первый и второй адреса. В качестве первого адреса в этой команде указывается адрес очередного элемента матрицы А и в качестве второго адреса — адрес очередного элемента матрицы *B.*

В общем случае при вычислении *j*-го члена в сумме $\sum_{i=1}^{n} a_{ij}b_{jk}$, определяющей элемент c_{ik} матрицы С, первый адрес будет равен $a+j+n(i-1)$, а второй адрес равен $b+j+n(k—1)$. Таким образом, при переходе к перемножению следующих элементов матрицы *A* (по строке) и матрицы *B* (по столбцу) оба адреса должны увеличиваться на единицу. После вычисления одного элемента c_{ik} адреса будут иметь вид:

первый адрес

$$a+n+n\,(i—1)+1;$$

второй адрес

$$b+n+n\,(k—1)+1.$$

Для перехода к вычислению следующего элемента по строке матрицы *C* нужно уменьшить на *n* значение индекса *j* и увеличить на единицу значение индекса *k.* При этом первый адрес команды $K+2$ уменьшится на *n* единиц, а второй адрес останется без изменения, так как уменьшение *j* на *n* единиц и увеличение *K* на 1 единицу взаимно компенсируются. При переходе к вычислению следующей строки матрицы *C* необходимо уменьшить на *n* единиц значение *k* и увеличить на единицу значение *i.* При этом первый адрес увеличивается на *n* единиц, а второй адрес уменьшается на n^2 единиц. Эти изменения предусмотрены соответствующими командами

$$(K+5),\ (K+11),\ (K+18)$$

и специальными числами

$$(d+9),\ (d+11),\ (d+12).$$

Для восстановления первоначального значения первого адреса после вычисления *n*-ой строки необходимо уменьшить его на n^2 единиц, для чего используется команда $K+22$ и условное число $(d+13)$.

Команда $K+8$ служит для переноса очередного значения c_{ik}, получаемого в постоянной ячейке $d+5$, в соответствующую ячейку. Третий адрес в этой команде изменяется и имеет зависимость от индексов *i, k* вида

$$c+k+n(i—1).$$

При переходе к вычислению следующего элемента c_{ik} в данной строке матрицы индекс k увеличивается на единицу и третий адрес команды $K + 8$ увеличивается на единицу. Это выполняется командой $K + 9$ с помощью условного числа (d+10).

При переходе к вычислению элементов следующей строки матрицы необходимо уменьшить на n единиц индекс k и увеличить на единицу индекс i, т. е. при этом третий адрес команды $K + 8$ не изменяется.

Для восстановления начального значения третьего адреса в $K + 8$ после окончания вычисления n строки матрицы C необходимо уменьшить его на n^2 единиц, что осуществляется командой $K + 23$ с помощью условного числа (d+14).

Остановимся на последних этапах программы.

Команда $K + 24$ обеспечивает контрольное суммирование значений элементов матрицы C; значение контрольной суммы получается в ячейке $d + 16$. Команда $K + 25$ увеличивает третий адрес команды $K + 24$ на единицу с тем, чтобы при повторении расчета контрольная сумма получилась в ячейке $d + 17$.

Команды $K + 26$ и $K + 27$ обеспечивают отсылку к повторению расчета. Команда $K + 28$ производит сравнение значений двух контрольных сумм. При их совпадении расчет считается правильным и команда $K + 29$ передает управление командам $K + 30$, $K + 31$, $K + 32$, обеспечивающим перевод данных в двоично-десятичную систему счисления и вывод из машины.

При несовпадении контрольных сумм команда $K + 29$ передает управления команде $K + 32$ для остановки машины. Несовпадение контрольных сумм свидетельствует о ненормальной работе машины (так как программа была предварительно проверена и отлажена и сомнений не вызывает). В этом случае после остановки машина должна быть проверена при помощи специальных тест-задач с целью выявления причин ненормальной работы и их устранения. После чего решение задачи должно быть повторено вновь.

Пример 2

Программа вычисления определенного интеграла $\int_a^b f(x)dx$

$$f(x) = \begin{cases} \frac{\sin x}{x} & \text{при } x \neq 0, \\ 1 & \text{при } x = 0. \end{cases}$$

Определенная таким образом $f(x)$ является непрерывной на любом отрезке [a, b] и, следовательно, ее можно интегрировать.

Отрезок [a, b] разделим на n равных частей и введем обозначение $h = \frac{b-a}{n}$ (h – шаг интегрирования).

Интеграл $\int_a^b f(x)dx$ представим в виде суммы n интегралов

$$\int_a^b f(x)dx = \int_a^{a+h} f(x)dx + \int_{a+h}^{a+2h} f(x)dx + \int_{a+2h}^{a+3h} f(x)dx + \ldots + \int_{b-h}^{b} f(x)dx$$

Значение интеграла на каждом шаге будем вычислять по формуле Симпсона

$$\int_{x_i}^{x_i+h} f(x)dx = \frac{h}{6}\left[f(x_i) + 4f(x_i + \frac{h}{2}) + f(x_i + h) \right]$$

При составлении программы будем придерживаться следующего порядка вычислений:

1. Вычислим $\int_a^{a+h} f(x)dx$ и $S_1 = S_0 + \int_a^{a+h} f(x)dx$, ($S_0 = 0$)
2. Вычислим $\int_{a+h}^{a+2h} f(x)dx$ и $S_2 = S_1 + \int_{a+h}^{a+2h} f(x)dx$
3. .....
4. .....

n. Вычислим $\int_{b-h}^{b} f(x)dx$ и $S_n = S_{n-1} + \int_{b-h}^{b} f(x)dx$

$$S_n \approx \int_a^b f(x)dx$$

Затем для контроля правильности работы машины повторяем расчет с тем же шагом. Результаты первого и второго расчетов сравниваем по команде сравнения (на полное совпадение по всем разрядам). Совпадение результатов считаем признаком правильности работы машины.

Точность вычислений зависит от величины шага интегрирования (при уменьшении шага точность повышается).

Первоначально шаг выбирается произвольно. Проверку правильности выбора шага будем производить следующим образом.

После вычисления интеграла с шагом h производим вычисление интеграла с шагом $h/2$ (по той же программе). Если абсолютная величина разности результатов, полученных с шагами h и $h/2$, невелика, т.е. $|S_n(h) - S_n(h/2)| \leq \varepsilon$, где ε выбирается в зависимости от требуемой точности, то в качестве результата выдается значение $S_n(h/2)$. Если же эта разность велика (больше ε), то снова уменьшаем шаг вдвое и сравниваем по ε уже результаты, полученные с шагами $h/2$ и $h/4$ и т. д.

План размещения данных в памяти показан в табл. IV.16.

Таблица IV.16

Размещение исходных данных в памяти машины

Номера ячеек	d+1	d+2	d+3	d+4	d+5	d+6	d+7	d+8	d+9	d+10	d+11	d+12
Данные	1	4	1/2	$f(a)$	a	b	k	k	$h/2$	$h/6$	ε	N

Примечания:
a и b — пределы интегрирования,
k — любое положительное число, отличное от нуля (для счета циклов),

В ячейку d+12 первоначально помещается некоторое большое число N, такое, чтобы заведомо выполнялось условие $|N - S_n(h)| > \varepsilon$, где $S_n(h)$ — значение интеграла, полученное при первом просчете. Перед вычислением с более мелким шагом в ячейку d+12 заносится предыдущий результат из ячейки c+7 (см. табл. IV.17, команда K+30).

Таблица IV.17

Программ

№№ п/п.	Команды						Пояснения
1	2	3	4	5	6	7	8
K+ 1	$П_5$	d+4	0002	c+3	0	22	I блок
K+ 2	Ф	0000	0000	c+7	0	07	
K+ 3	$П_5$	c+2	0002	c+1	0	22	II блок
K+ 4	С	c+4	d+9	c+4	0	01	
K+ 5	Ср	c+4	0000	0000	0	12	III блок
K+ 6	Е	K+7	K+9	0000	0	13	
K+ 7	Ф	d+1	0000	c+3	0	07	IV блок
K+ 8	Е	K+11	K+11	0000	0	13	
K+ 9	Sin x	c+4	0000	c+3	0	31	V блок
K+10	Д	c+3	c+4	c+3	0	25	
K+11	В	0000	d+7	d+7	0	03	VI блок
K+12	Е	K+13	K+3	0000	0	13	
K+13	М	c+2	d+2	c+2	0	05	VII блок
K+14	С	c+3	c+2	c+2	0	01	
K+15	С	c+2	c+1	c+1	0	01	
K+16	М	c+1	d+10	c+1	0	05	
K+17	С	c+1	c+7	c+7	0	01	
K+18	В	c+4	d+6	0000	0	03	VIII блок
K+19	Е	K+20	K+3	0000	0	13	
K+20	$П_5$	c+6	0002	c+5	0	22	IX блок
K+21	В	0000	d+8	d+8	0	03	
K+22	Е	K+23	K+1	0000	0	13	
K+23	Ср	c+5	c+7	0000	0	12	X блок
K+24	Е	K+25	K+34	0000	0	13	
K+25	В	d+12	c+7	c+6	0	04	XI блок
K+26	$В_1$	d+11	c+6	0000	0	04	
K+27	Е	K+32	K+28	0000	0	13	
K+28	М	d+9	d+3	d+9	0	05	XII блок
K+29	М	d+10	d+3	d+10	0	05	
K+30	Ф	c+7	0000	d+12	0	07	
K+31	Е	K+1	K+1	0000	0	13	
K+32	$Пр_2$	c+7	0001	c+6	0	33	XIII блок Перевод в двоично-десятичн. систему и вывод из машины
K+33	$П_4$	c+6	0001	0000	0	21	
K+34	Я	c+5	c+7	0000	0	15	XIV блок Остановка машины

Рабочие ячейки — (c+1) — (c+7). Окончательный результат получается в ячейке c+6.

Назначение отдельных этапов программы и порядок их выполнения указаны в блок-схеме программы (рис. IV.5).

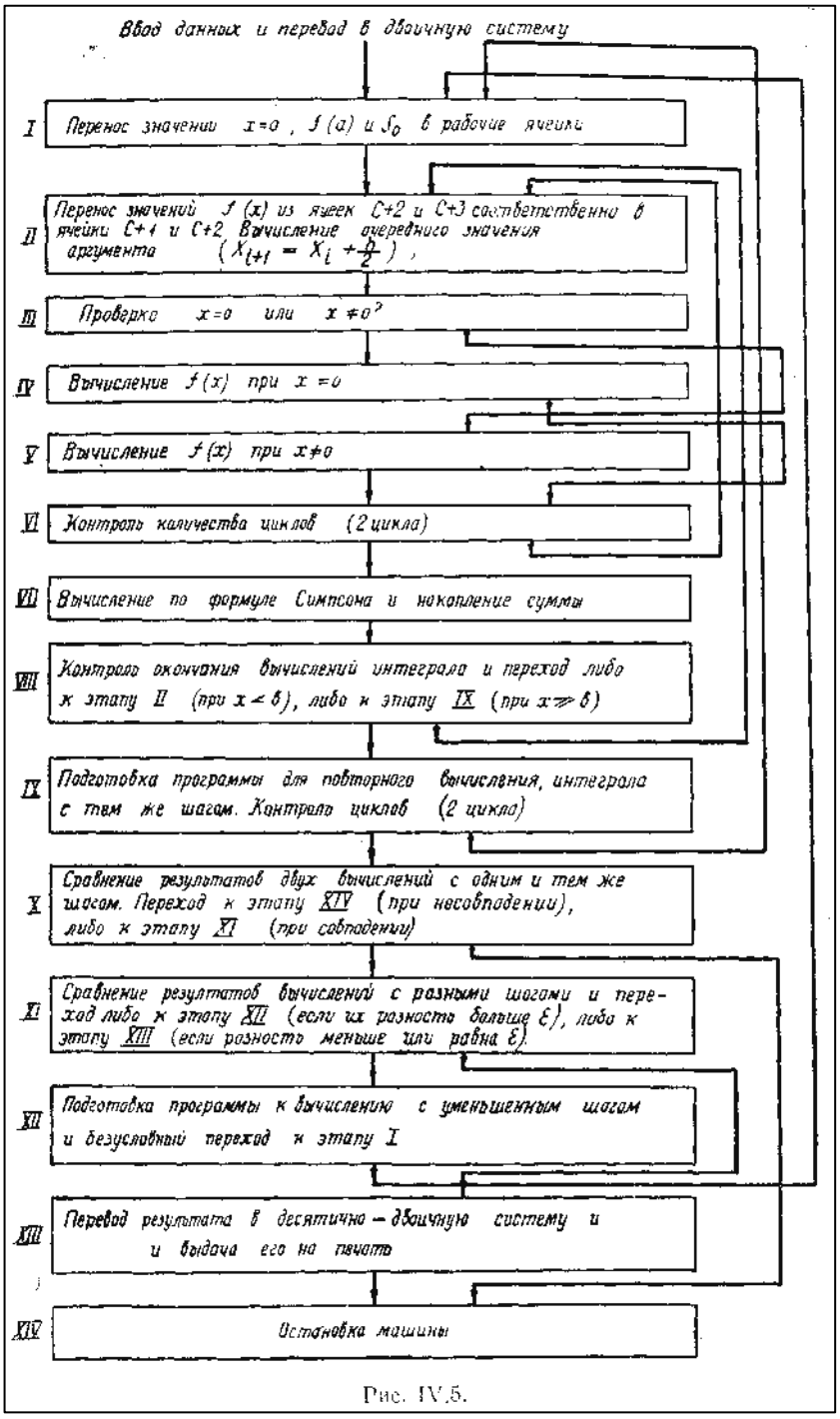

Рис. IV.5.

Дополнительно поясним только назначение и работу команд $K + 3$ и $K + 20$ и счетчиков циклов (этапы VI и IX).

Для вычисления интеграла на каждом шаге по формуле Симпсона требуется 3 значения *f(x),* которые мы будем размещать в ячейках внутренней памяти следующим образом:

c+1	c +2	c + 3
$f(x_i)$	$f(x_i+h/2)$	$f(x_i + h)$

При вычислении *f(x)* ее значения всегда помещаются в одну и ту же ячейку c + 3. Команда K+3 передвигает значения *f(x)* из ячейки c + 3 в ячейку c + 2, а из ячейки c + 2 в ячейку c + 1, причем вначале происходит перенос из ячейки c + 2 в ячейку c + 1, а затем из ячейки c + 3 в ячейку c + 2 (чтобы предыдущее значение $f(x)$ не стиралось последующим).

В целях контроля правильности работы машины вычисление интеграла с одним и тем же шагом производится дважды, после чего результаты сравниваются. Значение интеграла всегда получается в одной и той же ячейке $c+7$. Поэтому перед вторым вычислением значение интеграла нужно перенести из ячейки c + 7 в другую ячейку, чтобы оно не пропало при посылке нового числа в ячейку c + 7. Это делается с помощью команды K + 20,

которая производит перенос чисел из ячейки $c + 6$ в ячейку $c + 5$ и из ячейки $c + 7$ в ячейку $c + 6$.

В целях упрощения программа построена так, что команда $K + 20$ при двойном просчете выполняется 2 раза, после чего получается следующее размещение результатов:

$c+5$	$c+6$	$c+7$
S_n	S_n	S^*_n

S_n —значение интеграла, полученное при первом просчете;

S^*_n—значение интеграла, полученное при втором просчете.

Счетчик циклов блока VI обеспечивает двукратное выполнение блоков II - VI. Он работает следующим образом. По команде *(К* + 11) = В 0000 *d+7 d+7* содержимое ячейки *d*+7 вычитается из нуля и результат направляется в ту же ячейку *d* + 7. При каждом выполнении этой команды знак результата меняется (абсолютная величина остается неизменной) и по команде *(К* + 12) = *Е К*+13 *К*+3 0000 управление передается то команде *К*+3 (когда результат отрицательный), то команде *К*+13 (когда результат положительный). Первоначально в ячейку *d* + 7 помещается любое положительное число, например, 1. После выполнения блоков II - V выполняется команда *К* + 11, которая производит вычитание 0 — 1 = — 1. Результат получается отрицательный, и управление будет передано команде *К* + 3. Блоки II - V выполняются еще раз, после чего снова выполняется команда *К* + 11.

Результат теперь будет положительный 0 — (—1) = 1, и управление будет передано команде *К* + 13.

Счетчик циклов блока IX (команды *К* + 21 и *К* + 22) обеспечивает двукратное выполнение блоков I - IX и работает аналогичным образом.

15. ВЫЧИСЛИТЕЛЬНЫЕ ВОЗМОЖНОСТИ СОВРЕМЕННЫХ ЭЛЕКТРОННЫХ ЦИФРОВЫХ МАШИН

1. Типовые задачи, эффективно решаемые на машинах

Электронные счетные машины с программным управлением предназначаются для производства большого количества арифметических и логических операций в заданной последовательности, обеспечивающих численное решение математических задач. Так как численное решение почти любой математической задачи может быть сведено к последовательному выполнению четырех арифметических действий и выборке из таблиц, то эти машины являются машинами универсального назначения. Однако, несмотря на принципиальную возможность решения на электронных цифровых машинах любой задачи, практически не все задачи одинаково удобно решаются на машине.

Большие вычислительные и расчетные работы по характеру вычислительного процесса могут быть двух типов:

1) последовательно циклические вычисления;
2) параллельно циклические вычисления.

Примером последовательно циклического процесса является численное интегрирование обыкновенных дифференциальных уравнений, например, методами Рунге — Кутта, Адамса — Штермера и др.

Вычисление значений функции *y,* определяемой уравнением $y'=f(x,y)$, ведется по шагам для последовательно возрастающих значений аргумента. Для подсчета каждого последующего шага в качестве исходных данных используются данные, полученные на предыдущем шаге вычислений. При этом вид расчетных формул остается, как правило, постоянным для всех шагов вычислений; меняются только подставляемые в формулы величины. Для таких последовательно циклических вычислений характерным является небольшое количество входных и выходных данных и большой объем промежуточных данных. Эти процессы очень хорошо реализуются на автоматических счетных машинах и дают наибольший эффект в использовании машины, так как при этом затрата машинного времени и ручного труда на подготовку и ввод исходных данных в машину и вывод результатов решений получаются незначительными по сравнению с величиной машинного времени, используемого непосредственно для вычислений.

Менее выгодными для решения на автоматических машинах являются параллельно циклические задачи. Примером такой задачи может служить перевод большого количества данных, полученных опытным путем из одной системы измерения в другую. При этом необходимо многократно выполнять простой расчет над большим количеством исходных данных и получать большое количество результатов на выходе. Расчет ведется циклично, т. е. по одним и тем же формулам, но в отличие от первого случая здесь основная масса времени и труда будет тратиться не на вычисления, а на подготовку данных, ввод их в машину и на вывод результатов. Автоматическая машина в таких задачах будет использована недостаточно эффективно.

Помимо этого основного соображения, относящегося к характеру самого вычислительного процесса, при постановке задач для решения на машине играет роль сложность решаемых задач и требуемая точность вычислений.

Необходимо учитывать величину машинного времени, потребного для решения задачи. Действительно возможны случаи, когда задачи настолько просты, что могут быть успешно решены с помощью обычных ручных вычислительных средств. Естественно, что решение подобных задач на сложной вычислительной машине, потребляющей десятки киловатт электроэнергии и обслуживаемой несколькими десятками людей, нецелесообразно.

При оценке возможностей автоматических быстродействующих цифровых вычислительных машин и определении наиболее целесообразных областей их применения следует принимать во внимание, помимо большой скорости решения задач, также высокую точность вычислений. Машины оперируют обычно с 10 — 12-значными

числами в десятичной системе счисления и производят от 2000 до 20000 операций в секунду. Учитывая, что примерно половина операций (а иногда и больше) выполняется машиной для вспомогательных целей (преобразование команд, контроль вычислений, передача данных и т. п.) и, кроме того, каждую задачу с целью проверки правильности решений приходится решать дважды, можно ориентировочно считать, что средняя рабочая скорость большой машины в настоящее время характеризуется выполнением 2000 операций в секунду.

Известно, что вычислитель, работающий на настольной счетно-клавишной машине, производит в среднем одну операцию за 20 секунд.

Таким образом, рабочая скорость вычислений машины примерно в 40 000 раз превышает скорость работы вычислителя, работающего на настольной цифровой машине. При этом следует учесть, что вычислитель обычно работает с четырехзначными числами, а в машине вычисления производятся с 10-значными числами.

Практика применения больших вычислительных машин и анализ особенностей их конструкции показывают, что наиболее целесообразно применять эти машины для вычислений, связанных с составлением больших таблиц каких-либо функций, требующих производства массовых расчетов различных вариантов задач по одной и той же программе.

Важными областями использования быстродействующих вычислительных машин являются точные и трудоемкие вычисления, возникающие при применении современных математических методов для исследования различных научных и инженерных проблем. К числу типовых математических задач, эффективно решаемых на автоматических быстродействующих цифровых вычислительных машинах, относятся, например, следующие классы задач.

1. Системы алгебраических уравнений. Эти задачи имеют большое значение в геодезии, астрономии, при исследовании точности работы различных систем автоматического управления, при статистической обработке результатов наблюдений, в строительной механике.

Возможные методы решения систем алгебраических уравнений разделяются на две группы: итерационные методы и методы исключения. Обе группы методов состоят из серий стандартных вычислений и могут быть удобно применены для машины.

2. Системы обыкновенных дифференциальных уравнений. Этот класс математических задач является важным и распространенным в инженерной практике. Широкое применение обыкновенные дифференциальные уравнения находят во внешней и внутренней баллистике, как управляемых, так и неуправляемых снарядов, в аэродинамике, в нелинейной теории колебаний, в теории автоматического регулирования и т. д.

Системы обыкновенных дифференциальных уравнений решаются наиболее удобно применением численных методов, к числу которых относятся широко известные методы Адамса—Штермера, Рунге— Кутта и др. Могут быть применены также итерационные методы, например, итерационный процесс Пикара.

Все эти методы благодаря присущей им цикличности вычислений очень удобны для применения на машине.

3. Дифференциальные уравнения в частных производных эллиптического типа. Широко применяются в гидродинамике, газовой динамике, теории потенциала.

Для решения уравнений с частными производными эллиптического типа наиболее употребителен метод конечных разностей, состоящий в сведении задачи к решению систем линейных алгебраических уравнений.

4. Дифференциальные уравнения в частных производных гиперболического типа (уравнение струны и др.), задачи с начальными условиями (задача Коши.)

Этот класс задач имеет большое значение для газовой динамики, теории ударных волн, радиотехники, оптики, акустики, гидродинамики.

Применяя метод конечных разностей или метод характеристик, можно получить численное решение задач. Возможно также интегральное представление решений, что позволяет свести задачу к нахождению квадратур.

5. Дифференциальные уравнения в частных производных параболического типа. Применяются в различных разделах теоретической физики, в теплотехнике, метеорологии.

Численное решение основано либо на методе конечных разностей, либо на сведении задачи к квадратурам.

6. Задачи о собственных значениях дифференциальных операторов имеют большое значение в квантовой механике, атомной физике, теории колебаний, акустике.

Для численного решения применяются вариационные методы (метод Ритца, метод Галеркина и др.).

7. Интегральные уравнения. Находят широкое применение в различных областях физики, механики, в гидродинамике.

В последнее время интегральные уравнения получили применение для исследования различных процессов методами теории случайных функций.

8. Вычисление корней алгебраических уравнений высокого порядка имеет большое значение для ряда прикладных задач.

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

2. Некоторые специальные расчеты

В последние годы электронные цифровые вычислительные машины начали применяться в ряде новых областей, в частности, для прогнозов погоды, для экономико-статистических и военно-штабных расчетов.

Прогноз погоды.

Применение электронных цифровых машин для составления суточных и месячных прогнозов дает возможность более точно и быстро обрабатывать сведения, получаемые от метеорологических станций, и своевременно доводить прогнозы по большим районам до сведения заинтересованных лиц и учреждений. Использование машин уменьшает возможность появления ошибок в полученных прогнозах и значительно сокращает количество персонала, занятого составлением прогнозов.

В США первые работы по составлению суточных прогнозов погоды с помощью вычислительной машины велись Высшим исследовательским институтом в Принстоне. Для расчетов использовалась универсальная

электронная цифровая машина МАНИАК, которая по формулам, описывающим движение масс воздуха, за 1 час работы давала прогноз погоды за сутки. Обработка данных, получаемых от широкой сети метеорологических станций, расположенных по всей стране, и составление суточного прогноза погоды потребовало бы одночасовой работы 64000 вычислителей, работающих на клавишных автоматах.

Опыты по предсказанию погоды с помощью машины МАНИАК дали возможность предсказать два сильных шторма 5 ноября 1950 г. и 24 ноября 1953 г. За 12 часов до штормов были правильно определены место, время и сила штормов.

На построенных в настоящее время электронных цифровых вычислительных машинах может легко решаться задача предсказания погоды на 24 часа вперед. Наибольшие трудности при применении машин для прогнозов погоды возникают в связи с необходимостью ввода в машину огромного количества данных, для чего необходимы специальные быстродействующие входные устройства.

Экономико-статистические расчеты

В отличие от научно-технических задач, для которых характерными являются длинные последовательности вычислений со сравнительно небольшим количеством исходных данных и результатов вычислений, экономические, коммерческие, банковские, демографические и т. п. расчеты характеризуются большим количеством исходных данных и результатов вычислений и несложными, но чрезвычайно разнообразными вычислениями. Это весьма усложняет программирование, поэтому для решения подобных задач большие электронные вычислительные машины почти не применялись. Однако в последнее время начались работы по применению этих машин для решения коммерческих задач и экономико-статистических расчетов. Например, фирмой Ремингтон Ранд (США) в 1953 г. была построена электронная цифровая машина Дистрибьютон, предназначенная для механизации деловых расчетов. Машина используется для учета заказанных и проданных товаров, обрабатывая 90 тысяч товарных карточек в день. Основной частью машины является магнитный барабан емкостью 39000 ячеек.

В машине имеется 10 входных блоков — клавиатур, на которых одновременно работают 10 операторов. Каждую ночь машина запускается и печатает результаты дневной торговли по всем 8500 наименований товаров.

Большие универсальные машины УНИВАК, ИРА-11ОЗ и ИБМ-701 используются помимо решения научно-технических задач также и для коммерческих расчетов.

Например, на машине ИБМ-701 была решена задача о распределении рабочей силы для выполнения ряда заданий. Вычислительная машина, получив данные о составе специалистов, характере заданий и программу решения, дала меньше чем через час семейство графиков использования рабочей силы.

В таких отраслях, как химическая, нефтяная промышленность, выпускающих много различных продуктов, калькуляция стала настолько сложной, что действительная стоимость продуктов часто остается неизвестной. Для решения большой системы линейных уравнений, определяющей стоимость продукта, потребовалось выполнить 400 000 операций, что было сделано за несколько часов работы машины ИБМ-701.

В 1953 г. в США была построена вычислительная машина, предназначенная для расчетов, связанных с организацией военного тыла, снабжения и перевозок. Машина, названная Лоджистик, находится в Вашингтонском университете и используется научно-исследовательским бюро по организации тыла, снабжения и перевозок.