МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО
СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ РСФСР
КУЙБЫШЕВСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Ю. Н. МАЛИЕВ, М. В. КУДРЯВЦЕВ

ПРОГРАММИРОВАНИЕ НА ЭЦВМ «НАИРИ-К»

УЧЕБНОЕ ПОСОБИЕ ПО КУРСУ

«ОСНОВЫ ПРОГРАММИРОВАНИЯ»

Под общей редакцией
доцента Ю. Н. Малиева

КУЙБЫШЕВ
1975

УДК 681.14

Малиев Ю. Н., Кудрявцев М. В. Программирование на ЭЦВМ «Наири-К». Учебное пособие по курсу «Основы программирования». Куйбышевский госуниверситет, 1975.

В учебном пособии излагаются основные понятия и методы программирования для ЭЦВМ «Наири-К».

В первом разделе рассматриваются общие принципы представления и обработки информации в ЭЦВМ, а также порядок подготовки и решения задач.

Во втором разделе (практическая часть) рассматриваются конкретные вопросы программирования и решения задач на ЭЦВМ «Наири-К», разбираются принципы и методика составления рабочих программ в адресных кодах, а также автоматическое программирование на языке АП для этой машины.

Пособие рассчитано на студентов всех факультетов университета и может быть использовано преподавателями кафедр и сотрудниками научно-исследовательских групп в практической работе.

Иллюстр. — 11, табл. — 6, библ. — 7.

ПРЕДИСЛОВИЕ

Учебное пособие по программированию задач для решения на малой вычислительной машине типа «Наири-К» знакомит читателя с математическими и логическими принципами организации вычислений на электронных цифровых вычислительных машинах, системами счисления, употребляющимися при программировании и решении задач, способами представления чисел с фиксированной и плавающей запятой. Рассматриваются как общие методические вопросы составления программ, так и отдельные характерные особенности, возникающие при решении задач на машине.

Изложение материала ведется последовательно, начиная с принципов программирования в адресных кодах, и затем автоматического программирования на алгоритмическом языке АП. Текст иллюстрирован большим количеством примеров и упражнений и содержит рекомендации по использованию наиболее эффективных приемов программирования. Однако следует отметить, что приведенные задачи имеют и другие решения, и внимательный читатель может предложить свои варианты программ, разобраться в причинах расхождений, оценить преимущества и недостатки того или другого способа, а возможно и устранить ошибки.

Пособие предназначено для студентов, изучающих вопросы применения средств вычислительной техники для решения инженерно-технических, научных и статистико-экономических задач, а также может быть использовано для выполнения на ЭЦВМ трудоемких вычислений и расчетов. Пособие может быть использовано при программировании на «Наири-2», т. к. язык АП и система команд для обеих ЭЦВМ (за исключением команд «З», «$З_1$» и некоторых СП) одинаковы.

При подготовке учебного пособия авторы-составители использовали техническое описание машины «Наири-К», свой опыт работы, а также учебные материалы Куйбышевского инженерно-строительного института, Московского государственного университета и Новосибирского института инженеров железнодорожного транспорта.

Основные сведения о программировании

Глава 1

ОБЩИЕ ПРИНЦИПЫ ПРЕДСТАВЛЕНИЯ ИНФОРМАЦИИ В ЭЦВМ

При решении задач на электронных цифровых вычислительных машинах следует учитывать, что вся обрабатываемая информация (как смысловая, так и количественная) представляется внутри машины в числовой форме. Однако, если входная информация может быть в принципе выражена в любой знаковой или числовой системе, то по целому ряду причин более удобным для машинной обработки оказывается представление информации в так называемой двоичной (бинарной) системе. В связи с этим необходимо иметь некоторое понятие о принципах построения и обработки числовой информации в различных системах счисления.

§ 1. СИСТЕМЫ СЧИСЛЕНИЯ

Системой счисления называют способ представления чисел посредством цифровых знаков или символов. Различают позиционные и непозиционные системы счисления.

Разновидностью позиционной системы является римская система, в которой для записи чисел используются буквы латинского алфавита. При этом буква I всегда означает единицу, буква V — 5, X — десять, L — пятьдесят, C — сто, D — пятьсот, M — тысячу и т. д. Например, число 268 записывается в римской системе в виде CCLXVIII. Величина числа получается здесь сложением величин, изображаемых отдельными буквами. По этой причине непозиционные системы иногда называют также аддитивными (от латинского additio — сложение).

Подобные системы счисления неудобны и практически почти не применяются.

В позиционных системах любое число изображается с помощью ограниченного набора знаков (цифр), при этом количественное значение каждой цифры, входящей в запись числа, зависит от ее положения (позиции) в ряду цифр, изображающих число. Количество различных цифр (набор), применяемых в соответствующей системе счисления, называют ее о с н о в а-

нием. Эта величина определяет в то же время «весовое» отношение значений цифр, находящихся в смежных позициях представляемого числа. Например, в десятичной системе счисления используется десять знаков — цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9; основание системы здесь $q=10$.

Таким образом, любое число в какой-либо позиционной системе можно представить в следующей записи:

$$X = \alpha_n q^{n-1} + \alpha_{n-1} q^{n-2} + \ldots + \alpha_i q^{i-1} + \ldots + \alpha_1 q^0 + \alpha_{-1} q^{-1} + \ldots, \quad (1.1)$$

где q — основание системы; i — номер разряда; α_i — цифра, стоящая в i-м разряде; n — количество разрядов целой части числа.

Таблица 1

Системы счисления				
десятичная	восьмеричная	пятеричная	троичная	двоичная
0	0	0	0	0
1	1	1	1	1
2	2	2	2	10
3	3	3	10	11
4	4	4	11	100
5	5	10	12	101
6	6	11	20	110
7	7	12	21	111
8	10	13	22	1000
9	11	14	100	1001
10	12	20	101	1010
11	13	21	102	1011
12	14	22	110	1100
13	15	23	111	1101
14	16	24	112	1110
15	17	30	120	1111

Сокращенная запись имеет вид

$$X = \alpha_n \alpha_{n-1} \ldots \alpha_i \ldots \alpha_1 \alpha_{-1} \alpha_{-2} \ldots \quad (1.2)$$

Например, десятичное число 520, 78 можно записать, как

$$520{,}78 = 5 \cdot 10^2 + 2 \cdot 10^1 + 0 \cdot 10^0 + 7 \cdot 10^{-1} + 8 \cdot 10^{-2}$$

В двоичной системе счисления для изображения чисел используются всего два символа: 0 и I. Например, число пять записывается как 101, а число тринадцать — 1101 и т. д. Поскольку основание системы здесь $q=2$, то можно записывать двоичные числа по формуле (1.1).

Например, двоичное число:

$$1011{,}1_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1}.$$

Аналогичным образом представляются числа и в системах с другими основаниями.

Так, в восьмеричной системе счисления при использовании набора восьми цифр: 0, 1, 2, 3, 4, 5, 6, 7 и основании $q=8$ можно, например, изобразить число $325{,}4_8$, как

$$325{,}4_8 = 3\cdot 8^2 + 2\cdot 8^1 + 5\cdot 8^0 + 4\cdot 8^{-1}.$$

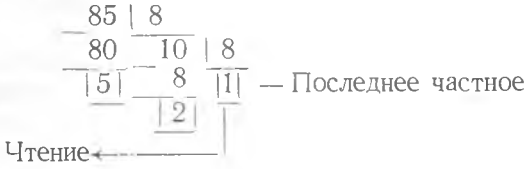
В таблице I для сравнения приводятся записи чисел от нуля до пятнадцати в различных системах счисления.

§ 2. ПЕРЕВОД ЧИСЕЛ ИЗ ОДНОЙ СИСТЕМЫ СЧИСЛЕНИЯ В ДРУГУЮ

При работе на ЭЦВМ может возникнуть необходимость перевода чисел из одной системы счисления в другую. Поскольку на практике главным образом приходится иметь дело с десятичной, восьмеричной и двоичной системами, рассмотрим основные приемы и правила перевода, используемых в этих случаях.

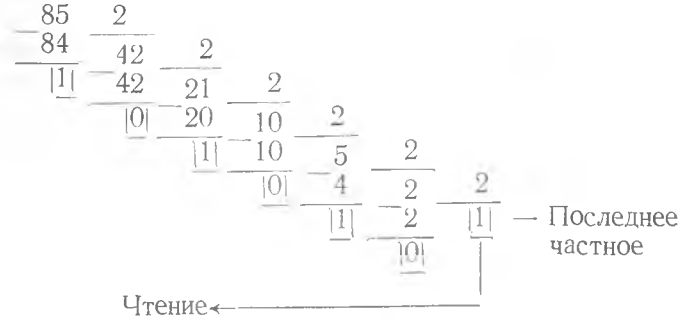
Правила перевода целых чисел. Для перевода целого десятичного числа в другую систему счисления нужно последовательно делить это число и получающиеся частные на новое основание q до тех пор, пока не образуется частное, меньшее q. Полученные при этом остатки, начиная с последнего частного, и будут «цифрами» числа в другой системе счисления.

Пример 1. Десятичное число 85 перевести в восьмеричную систему:



Таким образом: $85_{10} = 125_8$.

Пример 2. Десятичное число 85 перевести в двоичную систему:



Получим: $85_{10}=1010101_2$.

Правило перевода правильных дробей. Для перевода правильной десятичной дроби в другую систему счисления нужно произвести последовательное умножение дробных частей, получающихся произведений на новое основание q. Тогда получающиеся целые части произведений составят изображение дробной части числа в системе с основанием q.

Пример 3. Десятичную дробь 0,3125 перевести в двоичную:

Чтение

$\times$ 0,	3125
	2
0,	$\times$ 6250
	2
1,	$\times$ 2500
	2
0,	$\times$ 5000
	2
1,	0000

Таким образом: $0{,}3125_{10}=0{,}0101_2$.

Правило перевода неправильных дробей. При переводе неправильных дробей отдельно переводят целую и дробную части по рассмотренным выше правилам.

Например, десятичное число 85,3125, переведенное в двоичную систему, будет иметь вид:

$$85{,}3125_{10} = 1010101{,}0101_2.$$

Правило обратного перевода. Перевод чисел из двоичной и восьмеричной систем счисления в десятичную может быть осуществлен достаточно просто с помощью формулы (1.1). При этом исходное число записывают в виде соответствующего аддитивного ряда и выполняют по правилам десятичной системы необходимые действия.

Примеры.

Перевести число $213{,}4_8$ в десятичную систему:

$$213{,}4_8 = 2\cdot 8^2+1\cdot 8^1+3\cdot 8^0+4\cdot 8^{-1} = 139{,}5_{10}$$

Перевести число $1101{,}01_2$ в десятичную систему:

$$1101{,}01_2 = 1\cdot 2^3+1\cdot 2^2+0\cdot 2^1+1\cdot 2^0+0\cdot 2^{-1}+1\cdot 2^{-2} = 13{,}25_{10}$$

Примечание. Для перевода чисел в другие системы счисления можно, в принципе, использовать все рассмотренные правила.

Взаимосвязь восьмеричной и двоичной систем. Благодаря тому, что основание восьмеричной системы кратно целой степени основания двоичной системы $q=8=2^3$, перевод чисел из восьмеричной системы счисления в двоичную и обратно осуще-

Таблица 2

Системы	
восьмеричная	двоичная
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

ствляется чрезвычайно просто. Для этого достаточно заменить каждую восьмеричную цифру числа соответствующей двоичной триадой, то есть трехразрядным двоичным числом (см. таблицу 2).

Например, восьмеричному числу 413_8 будет соответствовать двоичное число

$$413_8 = (\underbrace{100}_{4}\ \underbrace{001}_{1}\ \underbrace{011}_{3})_2.$$

Наоборот, при переводе двоичного числа 101111010 в восьмеричную систему производят соответствующую свертку двоичных триад (справа—налево) в восьмеричные цифры

$$(\underbrace{101}_{5}\ \underbrace{111}_{7}\ \underbrace{010}_{2})_2 = 572_8.$$

Это же правило справедливо и для перевода дробных чисел.

Например, $101011{,}110_2 = 53{,}6_8$.

Практически восьмеричная система часто используется при программировании для записи номеров команд, адресов чисел, кодов операций, исходных данных и ряда других констант. Такая запись удобна тем, что занимает меньше места, чем двоичная и, кроме того, наиболее близка к десятичной. При вводе исходной информации в машину с помощью клавишного устройства восьмеричная запись автоматически преобразуется в двоичную.

§ 3. ДВОИЧНО-КОДИРОВАННЫЕ СИСТЕМЫ

Для удобства ввода числового материала (особенно при вводе больших массивов десятичных чисел) часто используются вспомогательные приемы, основанные на принципе двоичного кодирования десятичных чисел. Этот принцип состоит в том, что каждая из десятичных цифр: 0, 1, 2,..., 9 представляется определенным набором (тетрадой) четырех двоичных цифр. Из некоторого множества способов такого кодирования наиболее часто используется система кодирования 8421, в которой «вес» младшего единичного разряда тетрады равен 1, следующего — 2, и затем — 4 и 8.

Таблица 3

Цифра	Код 8 4 2 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

Таким образом, значение десятичных цифр будет отображаться позициями единиц в соответствующих разрядах дво-

ичных тетрад, как показано в таблице 3. Это дает возможность представить любое десятичное число простым сочетанием эквивалентных двоичных тетрад.

Например, число 2795_{10} в двоично-кодированной записи имеет вид:

$$\overbrace{0010}^{2}\ \overbrace{0111}^{7}\ \overbrace{1001}^{9}\ \overbrace{0101}^{5} =$$

$$= 0010011110010101_{(8421)}.$$

Переход от десятичного вида записи к двоично-кодированному производится автоматически с помощью обычных клавишных устройств, что облегчает ввод десятичных чисел в машину без предварительного перевода их в восьмеричную систему.

Таблица 4

Цифра	Код 5 2 1 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 1
3	0 1 0 1
4	0 1 1 1
5	1 0 0 0
6	1 0 0 1
7	1 0 1 1
8	1 1 0 1
9	1 1 1 1

Однако двоично-кодированное изображение чисел не равнозначно двоичному. В этом нетрудно убедиться, если перевести предыдущее число в двоичную систему счисления:

$$2795_{10} = 101011101011_2.$$

Дальнейший переход от двоично-кодированного вида записи к двоичной системе осуществляется в процессе работы машины по специальной программе перевода. Для вывода результатов из машины на печать производится обратное преобразование к двоично-кодированному виду, который уже техническим путем преобразуется в десятичную форму. Возможны и другие системы кодирования десятичных чисел. Так, например, в машине «Проминь», кроме уже рассмотренной системы, используется представление десятичных цифр в коде 5211 (см. табл. 4), который вследствие неоднозначности отображения десятичных цифр может еще иметь и другой вариант записи.

Что касается способов представления и обработки информации в самой машине, то в большинстве случаев принята обыкновенная двоичная система счисления.

В качестве исключения можно лишь отметить машину «Сетунь», работающую в троичной системе счисления. Для кодирования букв и других знаков также используются различные системы сочетаний из пяти, шести и семи двоичных символов (см., например, второй международный телеграфный код, коды по ГОСТ 10859-64 и др.).

Глава 2

ОСНОВНЫЕ МАТЕМАТИЧЕСКИЕ ОПЕРАЦИИ В ЭЦВМ

§ 1. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ В ДВОИЧНОЙ СИСТЕМЕ СЧИСЛЕНИЯ

Одним из преимуществ двоичной системы счисления является простота выполнения арифметических действий. Сложение производится по следующим правилам:

$$\begin{array}{|l|}\hline 0+0=0 \\ 0+1=1 \\ 1+0=1 \\ 1+1=10 \\ \hline \end{array}$$

(в последнем случае должен быть осуществлен перенос единицы в следующий, старший разряд суммы).

Пример 1. Сложение двух чисел.

Десятичная система	Двоичная система
$+\begin{array}{r}53\\41\\\hline 94\end{array}$	$+\begin{array}{r}110101\\101001\\\hline 1011110\end{array}$

$$94_{10} = 1011110_2.$$

Вычитание производится по следующим правилам:

$$\begin{array}{|l|}\hline 0-0=0 \\ 1-0=1 \\ 1-1=0 \\ 10-1=1 \\ \hline \end{array}$$

Пример 2. Вычитание двух чисел.

Десятичная система	Двоичная система
$\begin{array}{r}53\\-41\\\hline 12\end{array}$	$\begin{array}{r}110101\\-101001\\\hline 001100\end{array}$

$$12_{10} = 1100_2.$$

Умножение производится по следующим правилам:

$$\begin{array}{|l|}\hline 0 \times 0 = 0 \\ 0 \times 1 = 0 \\ 1 \times 0 = 0 \\ 1 \times 1 = 1 \\ \hline \end{array}$$

Как видно, по сравнению с громоздкой десятичной таблицей умножения, двоичная достаточна проста.

Числа умножаются по обычным правилам арифметики, причем процесс умножения может быть осуществлен со сдвигом частных произведений как влево, так и вправо. В первом случае умножение начинается с младшей цифры множителя, во втором — со старшей.

Пример 3. Умножение двух чисел.

Десятичная система

$$\begin{array}{r} 10 \\ \times\,13 \\ \hline 30 \\ +\,10 \\ \hline 130 \end{array}$$

Двоичная система

Сдвиг влево

$$\begin{array}{r} 1010 \\ \times\,1101 \\ \hline 1010 \\ 0000 \\ +\,1010 \\ \hline 1010 \\ \hline 10000010 \end{array}$$

Сдвиг вправо

$$\begin{array}{r} 1010 \\ \times\,1101 \\ \hline 1010 \\ +\,1010 \\ 0000 \\ 1010 \\ \hline 10000010 \end{array}$$

$130_{10} = 10000010_2$.

Из этого примера легко заметить, что при умножении двоичных чисел частное произведение равно множимому (если в соответствующем разряде множителя стоит единица) или нулю (если соответствующий разряд множителя равен нулю). Поэтому умножение фактически состоит из последовательных сдвигов множимого и сложения частных произведений, получающихся в результате сдвигов.

Операция деления выполняется в два действия: умножение и вычитание.

Пример 4. Деление двух чисел.

Десятичная система

$$\begin{array}{r|l} 210 & 15 \\ -\,15 & 14 \\ \hline 60 & \\ -\,60 & \\ \hline 00 & \end{array}$$

Двоичная система

$$\begin{array}{r|l} 11010010 & 1111 \\ -\,1111000 & 1110 \\ \hline 1011010 & \\ -\,111100 & \\ \hline 011110 & \\ -\,11110 & \\ \hline 000000 & \end{array}$$

$14_{10} = 1110_2$.

Таким образом, сложение, вычитание, умножение и деление двоичных чисел выполняется по тем же правилам, что и в обычной арифметике десятичных чисел.

Особенностью умножения и деления двоичных чисел является то обстоятельство, что эти операции сводятся к сложению (или вычитанию) и сдвигу чисел, благодаря чему значительно упрощается конструкция арифметического устройства вычислительных машин.

§ 2. ОПЕРАЦИИ С ДВОИЧНЫМИ КОДАМИ

Осуществление в вычислительных машинах арифметических действий над отрицательными числами связано с рядом неудобств. Например, для выполнения операций сложения и вычитания потребовались бы два отдельных устройства, причем устройство вычитания имело бы довольно сложную конструкцию. Чтобы избежать этих затруднений, применяются прямой, обратный и дополнительный коды, позволяющие операцию прямого вычитания заменить обычным сложением.

Для представления двоичного числа в каком-либо из указанных кодов, его величина должна находиться в интервале $0 \leqslant |X| < 1$, при этом знак числа изображается по правилу

$$«+» \to 0$$
$$«-» \to 1$$

и записывается в разряде целой части числа, который называется знаковым.

Изображение положительных чисел в прямом, обратном и дополнительном кодах совпадает и имеет вид обычного изображения правильной двоичной дроби:

$$X = 0,\ \underbrace{x_1\, x_2\, x_3 \ldots .\, x_n}_{\text{Знак}\quad\text{Мантисса}},$$

то есть $X = [X]_{\text{пр}} = [X]_{\text{обр}} = [X]_{\text{допол.}}$

Например:

$$+0{,}11011 = [0{,}11011]_{\text{пр}} = [0{,}11011]_{\text{обр}} = [0{,}11011]_{\text{доп}}$$

Прямой код. Отрицательное число $X = -0,x_1\, x_2\, x_3 \ldots, x_n$ в прямом коде изображается как

$$[X]_{\text{пр}} = 1,\ x_1\, x_2\, x_3 \ldots\, x_n.$$

Например, отрицательная двоичная дробь записывается в виде

$$-0{,}101011 = [1{,}101011]_{\text{пр}}.$$

Таким образом, прямой код двоичного числа совпадает по изображению с записью самого числа, но в разряде знака стоит 0 и 1 соответственно для положительных или отрицательных чисел.

Нуль в прямом коде имеет два изображения: для «положительного» нуля

$$+0 = [0{,}000\ldots0]_{\text{пр.}},$$

для «отрицательного» нуля

$$-0 = [1{,}000\ldots0]_{\text{пр.}}$$

Обратный код. Чтобы записать отрицательное число в обратном коде, необходимо в знаковом разряде поставить единицу, а в мантиссе нули заменить единицами, а единицы — нулями.

Таким образом, если имеем отрицательное число

$$X = -0,\ x_1\, x_2\, x_3 \ldots x_n,$$

то его изображение в обратном коде будет

$$[X]_{\text{обр}} = 1,\ \bar{x}_1\ \bar{x}_2\ \bar{x}_3\ \bar{x}_4\ \ldots\ \bar{x}_n,$$

где $\bar{x}_i$ есть инвертированное x_i

Например: $-0{,}101011 = [1{,}010100]_{\text{обр}}$.

Нуль в обратном коде также имеет два изображения: для «положительного» нуля

$$+0 = [0{,}000\ldots0]_{\text{обр}},$$

для «отрицательного» нуля

$$-0 = [1{,}111\ldots1]_{\text{обр}}.$$

Дополнительный код. Для того, чтобы получить дополнительный код отрицательного числа, необходимо в знаковом разряде поставить единицу, во всех разрядах мантиссы нули заменить единицами, а единицы — нулями и затем к младшему разряду прибавить единицу.

Таким образом, дополнительный код отрицательного числа

$$X = -0,\ x_1\ x_2\ x_3 \ldots x_n,$$

будет $[X]_{\text{доп}} = 1,\ \bar{x}_1\ \bar{x}_2\ \bar{x}_3 \ldots (\bar{x}_n + 1)$,

где $\bar{x}_i$ есть инвертированное x_i.

Например, если $X = -0{,}110111$, то

$$[X]_{\text{доп}} = 1{,}001000 + 0{,}000001 = 1{,}001001.$$

Нетрудно убедиться в том, что дополнительный код отрицательного двоичного числа есть дополнение этого числа до двух, то есть

$$[X>]_{\text{доп}} = 10 - X\geqslant,$$

где 10 означает два в двоичной системе счисления.

В дополнительном коде нуль имеет только одно значение

$$\pm 0 = [0{,}000\ldots0]_{\text{доп}}$$

Сложение чисел в обратном и дополнительном кодах. Для сложения чисел в обратном или дополнительном коде каждое из слагаемых представляют в соответствующем коде и складывают их обычным порядком. Результат будет выражен в том же коде, в каком были представлены слагаемые.

Если при сложении в обратном коде возникает единица переполнения старше знакового разряда, то она переносится в младший разряд суммы и складывается с ним (циклический перенос). При такой же ситуации в дополнительном коде — единица переполнения отбрасывается.

Рассмотрим следующий пример. Вычислить разность двух чисел: $X=+0{,}101011$ и $Y=+0{,}010110$.

Заменим знак второго числа на противоположный и, представив числа в обратном (или дополнительном) коде, произведем операцию сложения.

Обратный код	Дополнительный код
$[X]_{обр} = 0{,}101011$	$[X]_{доп} = 0{,}101011$
$[Y]_{обр} = 1{,}101001$	$[Y]_{доп} = 1{,}101010$
цикл. перенос $10{,}010100$ $+1$	отбр. $\leftarrow 10{,}010101$
$[Z]_{обр} = 0{,}010101$	$[Z]_{доп} = 0{,}010101$

Следует отметить, что достоверность результата сложения обеспечивается лишь в тех случаях, когда его величина составляет

$$0 \leqslant |X + Y| < 1.$$

Если абсолютная величина суммы окажется больше 1, то в содержание знакового разряда кода суммы войдет значение целой части числа, что приведет к неправильному представлению числа и его знака.

Покажем это на примере. Пусть требуется найти разность чисел:

$$X = +0{,}101010 \quad \text{и} \quad Y = -0{,}110001.$$

Величина: $X-Y=+0{,}101010-(-0{,}110001)=+1{,}011011$.

Если выполнить эту операцию в обратном (или дополнительном) коде, то получим

Обратный код	Дополнительный код
$+\ [X]_{обр} = 0{,}101010$	$+\ [X]_{доп} = 0{,}101010$
$[Y]_{обр} = 0{,}110001$	$[Y]_{доп} = 0{,}110001$
$[Z]_{обр} = 1{,}011011$	$[Z]_{доп} = 1{,}011011$

Единица в знаковом разряде здесь показывает, что получено число отрицательное и равное: в первом случае $Z=-0{,}100100$, во втором случае $Z=-0{,}100101$.

Результат, как видим, получился неверным.

Чтобы избежать в кодовой записи наложения целой части числа на знаковый разряд, часто употребляются так называемые модифицированные коды.

Модифицированные обратный и дополнительный коды. Особенность представления двоичных дробей в модифицированных кодах состоит в том, что на изображение знака отводится два разряда.

Положительные числа вида $X = +0,\ x_1\ x_2\ x_3\ \dots\ x_n$ изображаются в модифицированных обратном и дополнительном кодах одинаково, как:

$$X = [X]^{\text{м}}_{\text{обр}} = [X]^{\text{м}}_{\text{доп}} = 00,\ x_1 x_2 x_3 \dots x_n.$$

Отрицательное число $X = -0,\ x_1 x_2 \dots x_n$ в модифицированном обратном коде изображается в виде

$$[X]^{\text{м}}_{\text{обр}} = 1\bar{0},\ \bar{x}_1 \bar{x}_2 \bar{x}_3 \dots \bar{x}_n = 11,\ \bar{x}_1 \bar{x}_2 \bar{x}_3 \dots \bar{x}_n.$$

Например. $-0{,}10010 = [11{,}011101]^{\text{м}}_{\text{обр}}$.

Отрицательное число $X = -0,\ x_1 x_2 \dots x_n$ в модифицированном дополнительном коде изображается аналогично.

$$[X]^{\text{м}}_{\text{доп}} = 1\bar{0},\ \bar{x}_1 \bar{x}_2 \bar{x}_3 \dots (\bar{x}_n + 1) = 11,\ \bar{x}_1 \bar{x}_2 \bar{x}_3 \dots (\bar{x}_n + 1).$$

Например: $-0{,}100010 = [11{,}011110]^{\text{м}}_{\text{обр.}}$

Во всех рассмотренных случаях при $0 \leqslant |X| < 1$ содержание парных знаковых разрядов будет одинаковым, то есть: 00 (для положительных чисел) и 11 (для отрицательных чисел).

Если двоичная дробь окажется по величине $1 \leqslant |X| < 2$, то содержание знаковых разрядов будет различным.

Так, число $X = +1,\ x_1 x_2 x_3 \dots x_n$ в модифицированных кодах будет изображено как

$$[X]^{\text{м}}_{\text{обр}} = [X]^{\text{м}}_{\text{доп}} = 01,\ x_1 x_2 x_3 \dots x_n.$$

Например: $+1{,}100110 = [01{,}100110]^{\text{м}}_{\text{обр}} = [01{,}100110]^{\text{м}}_{\text{доп}}$.

Отрицательное число $X = -1,\ x_1 x_2 x_3 \dots x_n$ изображается в модифицированном обратном коде

$$[X]^{\text{м}}_{\text{обр}} = \bar{1}\bar{1},\ \bar{x}_1 \bar{x}_2 \bar{x}_3 \dots \bar{x}_n = 10,\ \bar{x}_1 \bar{x}_2 \bar{x}_3 \dots \bar{x}_n,$$

в модифицированном дополнительном коде

$$[X]^{\text{м}}_{\text{доп}} = 1\bar{1},\ \bar{x}_1 \bar{x}_2 \bar{x}_3 \dots (\bar{x}_n + 1) = 10,\ \bar{x}_1 \bar{x}_2 \bar{x}_3 \dots (\bar{x}_n + 1)$$

Таким образом, различное состояние знаковых разрядов (01 или 10) всегда свидетельствует о нарушении условия $0 \leqslant |X| < 1$ и называется переполнением разрядной сетки.

Арифметические операции в модифицированных кодах выполняются аналогично операциям в простых кодах.

Покажем это на примере сложения двух чисел:

$$X = +0{,}101011 \quad \text{и} \quad Y = -0{,}010110$$

Модифицированный обратный код	Модифицированный дополнительный код
$[X]^{м}_{обр} = 00{,}101011$	$[X]^{м}_{доп} = 00{,}101011$
$[Y]^{м}_{обр} = 11{,}101001$	$[Y]^{м}_{доп} = 11{,}101010$
$-100{,}010100$	$\leftarrow 100{,}010101$
$+1$	
$[Z]^{м}_{обр} = 00{,}010101$	$[Z]^{м}_{доп} = 00{,}010101$

Заметим, что удобство использования модифицированных кодов обеспечило их широкое применение в вычислительных машинах.

§ 3. ЛОГИЧЕСКИЕ ОПЕРАЦИИ

При построении схем электронных вычислительных машин и решении многих логических задач широко используется аппарат математической логики. К одному из его разделов относится исчисление высказываний или алгебра логики, основы которой были разработаны во второй половине прошлого столетия английским математиком Джорджем Булем.

Под высказыванием понимается любое утверждение, о котором можно сказать, что оно является либо истинным, либо ложным.

Например: алюминий — металл (истинное), снег — теплый (ложное).

При этом предполагается, что высказываний одновременно истинных и ложных не существует.

Вследствие того, что любое высказывание может иметь только одно из двух значений, истинное значение приравнивают единице, а ложное — нулю. Это обстоятельство позволяет применять двоичную систему счисления для подсчета значений истинности различных высказываний и является весьма ценным для применения в вычислительной технике. Переменную, принимающую всего два дискретных значения, называют логической или булевой.

В алгебре логики каждое отдельное высказывание обозначается соответствующей буквой латинского алфавита, для обозначения простых высказываний используют начальные буквы (A, B, C,...), а для обозначения сложных — конечные (P, Q, R,...). Содержание высказываний обычно учитывается только в процессе присвоения им буквенных символов. В дальнейшем высказывания оцениваются исключительно по их истинности или ложности, а конкретное содержание во внимание не принимается.

Над простыми высказываниями можно выполнять логические операции и в результате получать сложные высказывания. Связь между теми и другими устанавливается с помощью ло-

гических формул, состоящих из обозначений, высказываний и символов логических операций. Формулы позволяют также выяснить значение истинности сложного высказывания по значениям составляющих его простых. В этом смысле сложное высказывание является логической функцией $F(x_1, x_2, ..., x_n)$ от независимых логических аргументов, которыми могут служить простые высказывания.

Рассмотрим некоторые логические операции, встречающиеся при программировании и используемые в ряде алгоритмических языков.

Логическое сложение. Данная логическая операция обозначается знаком V (читается как ИЛИ) и называется дизъюнкцией.

Выражение P=AVB означает: «P есть A или B».

Правила логического сложения приведены в следующей таблице:

$0 \vee 0 = 0$
$0 \vee 1 = 1$
$1 \vee 0 = 1$
$1 \vee 1 = 1$

Из таблицы видно, что сложное высказывание P будет истинным в тех случаях, когда истинным является хотя бы одно из составляющих его простых высказываний. Иначе говоря, эта операция является разрешающей.

Логическое сложение обычно используется для формирования кодов чисел, в частности, для приформирования к некоторым кодам единиц в любых разрядах.

Например, если требуется к произвольному коду $X=10010101$ приформировать единицы в трех старших разрядах, то его следует логически сложить с константой $Y=11100000$.

$$\begin{array}{ll} & X = 10010101 \text{ — исходный код} \\ \vee & Y = 11100000 \text{ — константа} \\ \hline & Z = 11110101 \text{ — сформированный код.} \end{array}$$

Логическое умножение. Эта логическая операция обозначается знаком $\wedge$ (читается, как И) и называется конъюнкцией.

Выражение P=A∧B обозначает: «P есть A и B». Правила логического умножения приведены в следующей таблице:

$0 \wedge 0 = 0$
$0 \wedge 1 = 0$
$1 \wedge 0 = 0$
$1 \wedge 1 = 1$

В этом случае результирующее высказывание P истинно только тогда, когда одновременно истинны все составляющие его простые высказывания.

Логическое умножение в отличие от обыкновенного выполняется поразрядно и часто применяется для выделения части данного кода. При этом достаточно логически умножить двоичный код исходного числа на константу, содержащую единицы в тех разрядах, которые нужно выделить, то есть сохранить в коде результата.

Например:

$$\begin{array}{l} X = 10010101 \text{ — исходный код} \\ \wedge\ Y = 11110000 \text{ — константа} \\ \hline Z = 10010000 \text{ — результат.} \end{array}$$

Отрицание. Логическая операция отрицания обозначается как $\overline{A}$ (читается: «Не A») и называется также инверсией. Эта операция записывается, как $P=\overline{A}$ и воспроизводит следующие соотношения:

$$\begin{array}{|l} \overline{0}=1 \\ \overline{1}=0 \end{array}$$

Из таблицы видно, что если $A=I$, то $\overline{A}=0$.

Равнозначность Эта логическая операция обозначается $\sim$ или $\equiv$ (читается: равнозначно или эквивалентно). Истинность выражения $P=A\sim B$ или $P=A\equiv B$ определяется таблицей

$$\begin{array}{|l} 0\sim 0=1 \\ 0\sim 1=0 \\ 1\sim 0=0 \\ 1\sim 1=1 \end{array}$$

Следовательно, сложное высказывание $P=A\sim B$ истинно только в тех случаях, когда составляющие его простые высказывания имеют одинаковые значения истинности.

Поразрядное сложение. Эта логическая операция, называемая также сложением по модулю 2, записывается как $P=A\oplus B$ и может быть получена из операций отрицания и равнозначности. Здесь результирующее высказывание (в противоположность предыдущему случаю) будет истинным лишь при условии, что составляющие его простые высказывания имеют противоположные значения истинности:

$$\begin{array}{|l} 0\oplus 0=0 \\ 0\oplus 1=1 \\ 1\oplus 0=1 \\ 1\oplus 1=0 \end{array}$$

Операция поразрядного сложения используется для сравнения кодов чисел, при этом критерием сравнимости чисел является результат операции, равный нулю.

Например:

$$\begin{array}{r} X = 10010101 \\ \oplus\ Y = 11010100 \\ \hline Z = 01000001 \end{array}$$ — числа не равны

$$\begin{array}{r} X = 10010101 \\ \oplus\ Y = 10010101 \\ \hline Z = 00000000 \end{array}$$ — числа равны

Остальные, не рассматриваемые здесь логические операции, равно как и булевы функции, могут быть в принципе получены из приведенных методами суперпозиции.

Глава 3

ФОРМЫ ПРЕДСТАВЛЕНИЯ ЧИСЕЛ В ЭЦВМ

При отображении числовых величин в разрядной сетке вычислительной машины необходимо точно знать положение запятой, отделяющей целую часть числа от дробной. Возможны два подхода к разрешению этой задачи: либо отвести для запятой постоянное место, либо по мере надобности менять ее положение. В соответствии с этим в ЭЦВМ применяют две формы представления чисел: представление чисел с фиксированной запятой («естественная форма») и представление чисел с плавающей запятой («нормальная форма»).

§ 1. МАШИНЫ С ФИКСИРОВАННОЙ ЗАПЯТОЙ

При представлении чисел в форме с фиксированной запятой положение запятой закрепляется в определенном месте относительно разрядов числа и сохраняется неизменным для всех чисел, изображаемых в данной разрядной сетке машины.

В большинстве случаев запятая фиксируется перед старшим разрядом. Это означает, что в такой разрядной сетке могут отображаться только лишь числа по модулю меньшие единицы

$$0 \leqslant |X| < 1.$$

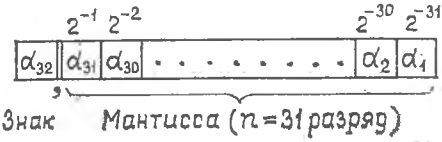


Рис. 1.

Если отвести для знака числа еще один разряд, то можно для записи двоичных чисел использовать их представление в прямом коде.

На рис. 1 в качестве примера показана 32-разрядная сетка для представления чисел с фиксированной запятой. Здесь, над клетками разрядной сетки, указан «вес» каждого двоичного разряда.

Таким образом, в данной разрядной сетке могут быть представлены числа в диапазоне

$$2^{-31} \leqslant |X| \leqslant (1-2^{-31})$$

или, в общем случае,

$$2^{-n} \leqslant |X| \leqslant (1-2^{-n}), \qquad (3.1)$$

где n — количество разрядов мантиссы (дробной части числа).

Следует отметить, что если в процессе вычислений получится число, большее чем $(1-2^{-n})$, то произойдет «переполнение» разрядной сетки и нарушение содержимого знакового разряда. Поэтому для своевременной блокировки вычислений в сумматоре машин часто добавляется еще один знаковый разряд и используется представление чисел в каком-либо модифицированном коде. Пример такой разрядной сетки приведен на рис. 2.

К недостаткам использования машин с фиксированной запятой относится необходимость масштабирования исходных величин и связанная с этим трудность определения границ изменения значений результатов. Кроме того, при вводе чисел в машину следует учитывать относительную погрешность их представления, которая будет тем больше, чем меньше величина числа. По этим причинам в некоторых машинах рассматриваемого типа вводят искусственный программный режим работы с плавающей запятой.

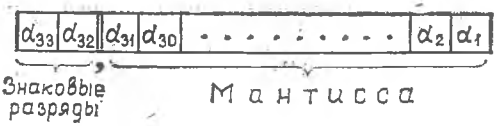


Рис. 2.

§ 2. МАШИНЫ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

В машинах с плавающей запятой все числа представляются в так называемой нормальной форме

$$X = M_x \cdot q^{\pm p}, \tag{3.2}$$

где q — основание системы счисления;

p — целое число, называемое порядком.

Величина M_x называется мантиссой числа X и для нее должно соблюдаться условие

$$q^{-1} \leqslant M_x < 1, \tag{3.3}$$

при котором число X называется нормализованным.

Так, например, десятичное число $X = +25{,}379$ следует записать как

$$X = +0{,}25379 \cdot 10^{+2},$$

а двоичное число

$$Y = -1101{,}01 \text{ соответственно} —$$

$$Y = -0{,}110101 \cdot 2^{+100}.$$

Числа, по модулю меньшие единицы, записываются аналогично. Например, число $X = +0{,}00495_{10}$ представляется как

$$X = +0{,}495 \cdot 10^{-2}.$$

Таким образом, знак и величина порядка ($\pm p$) указывают действительное положение запятой.

Разрядная сетка машины с плавающей запятой должна в этом случае содержать участки, изображенные на рис. 3. При этом знаки числа и порядка по-прежнему изображаются в соответствии с правилом: «+»⟶0; «—»⟶1.

Нормализация чисел производится при вводе чисел в машину, при выполнении арифметических действий и осуществляется с помощью особой операции, которая называется сдвиг. Сущность этой операции заключается в том, что все разряды мантиссы сдвигаются так, чтобы абсолютная величина ее попала внутрь интервала (3.3). При этом, если сдвиг происходит вправо, то порядок увеличивается на столько единиц, на сколько разрядов произошел сдвиг мантиссы. При сдвиге влево величина порядка уменьшается на соответствующее число единиц.

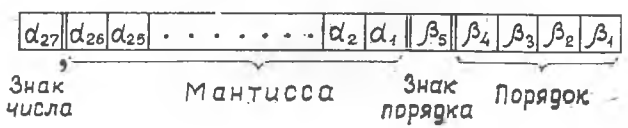


Рис. 3.

Например, ненормализованное двоичное число

$$X = +0{,}00101 \cdot 2^{-110}$$

после нормализации будет равно

$$X = +0{,}10100 \cdot 2^{-1000}$$

(здесь $p = -110 - 10 = -1000$).

Арифметические операции в машинах с плавающей запятой выполняются в несколько приемов. Так, при сложении чисел сначала уравниваются порядки слагаемых, а затем складываются мантиссы. Порядком суммы является общий порядок слагаемых.

Уравнивание порядков заключается в том, что меньший порядок числа увеличивается до большего и при этом соответственно изменяется мантисса, то есть происходит денормализация числа. Полученная сумма (при необходимости) нормализуется.

Например, необходимо сложить два нормализованных числа

$$X = +0{,}101101 \cdot 2^{+101}$$

$$Y = +0{,}100110 \cdot 2^{+11}.$$

Увеличиваем порядок числа Y на две единицы

$$Y = +0{,}00100110 \cdot 2^{+101}.$$

Производим сложение:

$$\begin{array}{r|l} X = +0{,}101101 & \cdot 2^{+101} \\ + \quad Y = +0{,}001001 & 10 \cdot 2^{+101} \\ \hline Z = +0{,}110110 & 10 \cdot 2^{+101} \end{array}$$

Если при денормализации младшие разряды выходят за пределы разрядной сетки, то они теряются.

Умножение чисел в машинах с плавающей запятой производится также в несколько приемов: определяется знак произведения, перемножаются мантиссы сомножителей, складываются (алгебраически) порядки сомножителей: нормализуется результат (если необходимо).

Диапазон представляемых чисел в машинах в плавающей запятой значительно шире, чем в машинах с фиксированной запятой. Кроме того, отпадает необходимость в подборе масштабных коэффициентов для исходных величин, что значительно облегчает программирование и расширяет круг решаемых задач.

Однако в машинах с плавающей запятой также возможно переполнение разрядной сетки, которое заключается теперь в том, что в результате какой-либо операции возникает число, имеющее порядок с большей разрядностью, чем допустимая в машине. Возможно также появление машинных нулей,

то есть нормализованных чисел, имеющих порядок, меньший самого малого порядка, представимого в машине.

Машины с плавающей запятой конструктивно более сложны и стоимость их выше.

Глава 4

ПОДГОТОВКА И РЕШЕНИЕ ЗАДАЧ НА ЭЦВМ

Для решения задачи на ЭЦВМ необходимо, прежде всего, подготовить ее таким образом, чтобы весь процесс вычислений был согласован с возможностями используемой машины и особенностями соответствующих устройств ввода и вывода информации. В свою очередь, процедура решения также требует учета некоторой специфики машинной обработки информации, организации решения, влияния погрешностей и т. п. Поэтому вначале рассмотрим методические вопросы.

§ 1. МЕТОДИКА ПОДГОТОВКИ И РЕШЕНИЯ ЗАДАЧ

Весь процесс подготовки и решения задач на ЭЦВМ удобно подразделить на ряд характерных этапов.

Математическая формулировка задачи. На этом этапе устанавливается в окончательном виде математическое описание задачи (формулы, зависимости и т. п.), необходимые условия, ограничения, пределы изменения переменных и т. д.

Выбор численного метода. Численные математические методы позволяют свести решение самых разнообразных задач к последовательному выполнению четырех арифметических действий. В большинстве случаев одна и та же задача может быть решена несколькими численными методами. На этом этапе оцениваются возможные численные методы решения и выбирается тот из них, который наилучшим образом обеспечивает выполнение требований поставленной задачи.

Указанные два этапа не имеют непосредственного отношения к программированию, и поэтому подробно не рассматриваются.

Разработка решающего алгоритма. В процессе выполнения этого этапа устанавливается необходимая последовательность арифметических и логических действий, с помощью которых может быть реализован выбранный численный метод.

Разработка решающего алгоритма является важным моментом в подготовке структуры (плана) будущей программы.

Практически может быть составлено несколько алгоритмов решения одной и той же задачи. При этом подбирается такой вариант, который обеспечивает наиболее эффективное использование машины. Следует отметить, что алгоритм всегда разрабатывается в обобщенном виде, то есть с употреблением такого языка, который наглядно отображает структуру алгоритма в целом, независимо от конкретного типа используемой для решения машины.

Составление структурной блок-схемы программы. На основе разработанного алгоритма и для удобства программирования целесообразно составлять так называемую структурную блок-схему программы. Такая блок-схема является наглядно-графическим изображением логического построения процесса решения задачи, определяет детальный план и хронологический порядок работы машины. При этом блок-схема помогает представить и зрительно охватить все логические связи и переходы, осуществляемые в процессе вычислений.

Программирование. После того, как составлена логическая схема программы, нужно перевести ее на язык конкретной вычислительной машины. Для этого вначале планируют размещение в ячейках памяти всех исходных величин и констант. Затем, используя кодовую характеристику операций, составляют логическую последовательность команд, которые подетально описывают каждый шаг машины. Такая совокупность команд и называется п р о г р а м м о й.

Необходимо заметить, что составление программы — трудоемкий и сложный процесс. Он является источником большого количества ошибок и поэтому требует от исполнителя напряженного внимания.

Отладка программы. Прежде чем решать задачу, необходимо разработанную программу опробовать на машине с тем, чтобы выявить возможные ошибки, допущенные на предыдущих этапах. Предварительно вручную решается упрощенный вариант задачи, называемый контрольным или отладочным, а затем уже производится отладка.

Наиболее простым методом отладки является контроль выполнения всех команд программы в однотактном режиме и сопоставление результатов с данными контрольного варианта. Однако этот метод неэкономичен, так как приводит к большому расходу машинного времени и практически используется лишь для проверки отдельных сложных участков программы.

Широко используется метод отладки программ в режиме контрольных остановов. При этом программа расчленяется на ряд более или менее самостоятельных участков. В конце каждого контролируемого участка предусматривается останов машины. Полученные результаты сравниваются с результатами отладочного варианта. Контрольные остановы осуществляются либо

с помощью специальных команд, либо с клавиатуры пульта управления машины.

Для экономии времени, затрачиваемого на отладку программ, используются специальные программы, которые обеспечивают выдачу на печать необходимой информации. Наибольший эффект дает автоматизация процесса отладки с помощью так называемых отладочных программ. Сущность работы отладочной программы заключается в имитации работы машины в однотактном режиме. Последовательно исполняются все команды программы и выдается на печать необходимая информация о каждой команде.

Решение задачи на машине. При решении задачи на машине необходимо принять меры, гарантирующие правильность выполнения вычислений. Даже если программа составлена без ошибок, еще нельзя быть уверенным в правильности результатов, поскольку сама машина в процессе счета может допускать как систематические, так и случайные ошибки. Систематические ошибки возникают в результате неисправности отдельных устройств и элементов машины. Случайные ошибки (сбои) могут появиться вследствие резких изменений напряжения в цепях питания, воздействия внешних магнитных полей, перепадов температуры, сотрясений пола и т. п.

Систематические ошибки обнаруживаются и устраняются в процессе проверки исправности машины с помощью специальных тестовых программ.

Случайные ошибки могут быть исключены различными программными методами. Чаще всего используется метод двойного и тройного счета. В этом случае задача на машине решается два или три раза и производится визуальное сличение результатов. Гарантией правильности вычислений будет совпадение любых двух решений.

Для проверки правильности ввода в машину исходной информации используется также контрольное суммирование. Оно заключается в том, что при вводе информации логическим образом суммируются как команды, так и числовые величины. Полученный результат сравнивается с контрольной суммой, вычисленной заранее. Совпадение этих величин и служит гарантией правильности ввода. Этот метод особенно эффективен при необходимости неоднократного ввода и решения задачи.

Практически может быть использована и комбинация приведенных способов. Кроме того, существует также метод аппаратного контроля исправности машины и другие частные методы.

§ 2. РАЗРАБОТКА РЕШАЮЩЕГО АЛГОРИТМА

Понятие алгоритма принадлежит к числу наиболее употребляемых в современной вычислительной математике и используется

в качестве основы при разработке процесса решения задач на ЭЦВМ.

В настоящее время под алгоритмом принято понимать некоторую совокупность элементарных правил, предписывающих выполнение при определенных условиях тех или иных операций для решения задач некоторого данного типа. Тем самым исходная информация должна однозначно определять результат работы алгоритма.

По А. А. Маркову «... в математике под алгоритмом понимается точное предписание, определяющее вычислительный процесс, ведущий от варьируемых исходных данных к искомому результату».

Нужно отметить, что приведенные формулировки не являются точным математическим определением понятия алгоритма, они скорее толкуют значение этого слова, поясняя его смысл. И в то же время отражают то понятие алгоритма, которое стихийно складывалось и применялось в математике с древнейших времен.

Таким образом, в общем случае, процесс применения алгоритма рассматривается как потенциально осуществимый процесс, ведущий после конечного (хотя бы и очень большого) числа шагов к искомому результату. Большое значение при этом имеет язык, с помощью которого описывается алгоритм; используемые понятия и терминология должны быть такими, чтобы исполнитель алгоритма, во-первых, его понял, а во-вторых, имел бы технические возможности выполнить указания, содержащиеся в алгоритме.

Для примера рассмотрим задачу вычисления значений функции

$$y = f(x) = \sin\left(\frac{\pi}{2} \cdot \frac{x^2}{1+x^2}\right)$$

в интервале $x_0 \leqslant x \leqslant x_1$ для $x = x_0,\ x_0 + h,\ x_0 + 2h, \ldots,\ x_0 + nh,$ где $h > 0$ и $n = E\left(\frac{x_1 - x_0}{h}\right)^* + 1.$

Если обозначить $y_i = f(x_0 + ih)$, где $i = 0, 1, 2, \ldots, n$, то алгоритм для решения этой задачи можно записать следующим образом:

1. Принять x равным x_0 (для краткости можно записать, как $x := x_0$).
2. Принять i равным 0 ($i := 0$).
3. Вычислить $y_i = \sin\left(\frac{\pi}{2} \cdot \frac{x^2}{1+x^2}\right)$.
4. Дать x новое значение, полученное из старого прибавлением h ($x := x + h$).

---

* $E(A)$ означает целую часть числа A.

5. Если после этого стало $x>x_1$, то перейти к п. 8, в противном случае выполнить п. 6.

6. Дать i новое значение, полученное из старого прибавлением единицы $(i:=i+1)$.

7. Перейти к выполнению п. 3.

8. Остановиться.

Предписание, содержащееся в п. 3, фактически рассчитано на исполнителя, знакомого с термином sin и числом π, располагающего тригонометрическими таблицами или иными известными ему способами нахождения синусов. Несоблюдение хотя бы одного из условий сделает алгоритм невыполнимым.

Однако при помощи небольшого изменения алгоритма мы можем ориентировать его и на менее квалифицированного исполнителя, если зададим значение числа $\pi=3{,}1416$, а синус представим разложением в ряд с точностью, допустим, до трех членов ряда

$$\sin z = z - \frac{z^3}{6} + \frac{z^5}{120}.$$

Новый вариант алгоритма запишется так:

1. $x := x_0$.
2. $i := 0$.
3. $z := \frac{3{,}1416}{2} \cdot \frac{x^2}{1+x^2}$.
4. $y_i := z - \frac{z^3}{6} + \frac{z^5}{120}$.
5. $x := x + h$.
6. Если после этого стало $x>x_1$, то перейти к п. 9, в противном случае выполнить п. 7.
7. $i:=i+1$.
8. Перейти к выполнению п. 3.
9. Остановиться.

Не следует однако увлекаться особенной детализацией алгоритма: это можно осуществлять и на более поздних стадиях подготовки задачи к счету. В алгоритме иногда достаточно сослаться на название метода, который предлагается использовать в конкретной реализации.

Алгоритм, не перегруженный излишними подробностями, помогает составителю обозреть и осознать свой собственный замысел, увидеть места, подлежащие корректировке, дает возможность обсудить его со специалистами. Здесь очень важно найти разумный компромисс между желанием сделать алгоритм весьма общим, и с другой стороны, выполнить его простым и быстро исполняемым на вычислительной машине.

Таким образом, довольно отчетливо выступают следующие черты решающих алгоритмов*:

Детерминированность алгоритма. Требуется, чтобы метод вычисления можно было сообщить другому лицу в виде конечного числа указаний о действиях на отдельных стадиях вычисления. Причем, вычисления согласно этим указаниям, не зависят от произвола вычисляющего лица и представляют собою детерминированный процесс, который может быть в любое время повторен и выполнен с тем же успехом и другим лицом.

Массовость алгоритма. Алгоритм — это единое предписание, определяющее вычислительный процесс, который может начинаться от различных исходных данных и ведет во всех случаях к соответствующему результату.

Иными словами, алгоритм решает не одну лишь индивидуальную задачу, а некоторую серию однотипных задач.

Наконец, при разработке решающего алгоритма следует учитывать некоторые особенности, связанные с требуемой точностью вычислений, стоимостью всех работ и общим расходом времени на решение задачи.

§ 3. СОСТАВЛЕНИЕ БЛОК-СХЕМЫ ПРОГРАММЫ

Для облегчения и удобства программирования широко используется графическая запись структуры алгоритмов в виде так называемых блок-схем. При этом способе записи алгоритм представляется в виде последовательности арифметических и логических блоков, каждому из которых соответствует определенный этап решения задачи. Блоки изображаются в виде прямоугольных или овальных (ромбических) контуров, внутри которых записывается краткое содержание данного этапа вычислений или формула, по которой эти вычисления осуществляются. Блоки соединяются стрелками, указывающими связи между различными этапами.

Рассмотрим принцип построения блок-схемы решения задачи на примере, разобранном в § 2, гл. 4.

Пусть требуется составить блок-схему программы вычисления значений функции

$$y = f(x) = \sin\left(\frac{\pi}{2} \cdot \frac{x^2}{1+x^2}\right)$$

в интервале $x_0 \leqslant x \leqslant x_1$ для $x = x_0,\ x_0 + h,\ x_0 + 2h, \ldots,\ x_0 + nh$.

В соответствии с разработанным ранее алгоритмом блок-схема будет иметь вид, показанный на рис. 4.

---

* Трахтенброт Б. А. Алгоритмы и машинное решение задач. М., Физматгиз, 1960.

Приведем еще один пример построения блок-схемы программы вычисления корней квадратного уравнения

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

Построение блок-схемы целесообразно начать с проверки условия равенства нулю коэффициента «a». Если $a=0$, то уравнение имеет один корень $x_1 = -c/b$. Если $a \neq 0$, то вначале вычисляется величина $D=b^2-4ac$ и проверяется на условие неотрицательности. Если $D \geqslant 0$, то вычисляются действительные корни уравнения

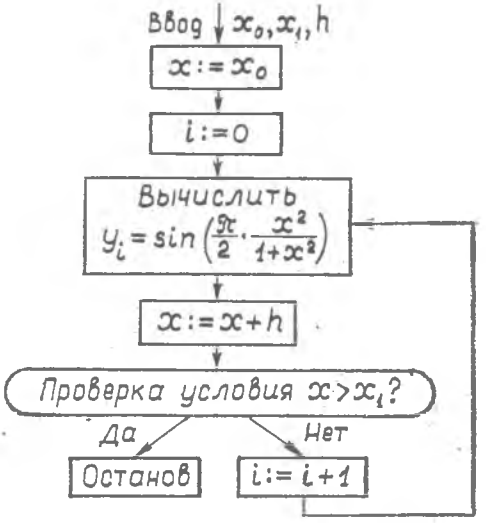


Рис. 4.

$$x_{1,2} = \frac{-b \pm \sqrt{D}}{2a},$$

если $D<0$, то вычисляются мнимые корни

$$x_{1,2} = -\frac{b}{2a} \pm i\frac{\sqrt{|D|}}{2a}.$$

Блок-схема программы приведена на рис. 5.

Достоинством блок-схемы является наглядность алгоритма вычислительного процесса. Кроме того, при больших и сложных блок-схемах программирование отдельных групп блоков может быть поручено различным лицам, что значительно ускоряет составление общей машинной программы.

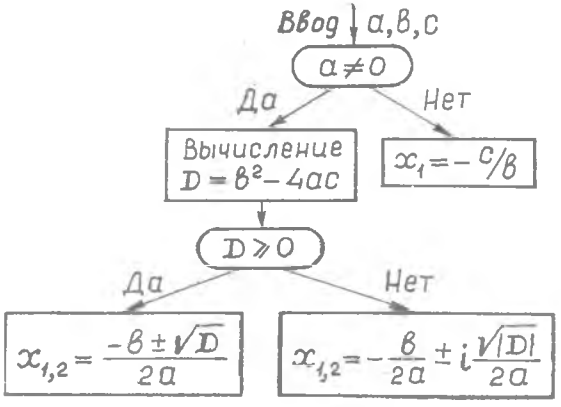


Рис. 5.

Программирование для ЭЦВМ «НАИРИ-К» в адресных кодах

Глава 5

ОСНОВНЫЕ СВЕДЕНИЯ О МАШИНЕ «НАИРИ-К»*

Малая универсальная электронная вычислительная машина «Наири» предназначена для решения весьма широкого круга математических задач, возникающих при инженерных и экономических расчетах и в научных исследованиях. Благодаря тому, что кроме обычного, адресного программирования возможен ввод задач на языке, приближенном к математическому с использованием автоматического программирования, значительно облегчается и ускоряется процесс подготовки задач для решения.

ЭЦВМ «Наири» является *двухадресной* программно-управляемой машиной с возможностью следующих режимов работы:

а) режим работы с программой, представленной в записи на адресном языке;

б) режим работы с использованием автоматического программирования на языке «АП»;

в) режим работы простой счетной машины.

Диапазон представления чисел и модификация команд определены из того, что ячейки ОЗУ состоят из 36 двоичных разрядов. Входные величины могут быть представлены в виде:

десятичных чисел с фиксированной запятой с записью в форме

$$X = \pm 0,\ x_1 x_2 x_3 \ldots x_n.$$

Диапазоны представления чисел в этом виде

$$-2 \leqslant X \leqslant -2^{-34},\ \text{если}\ X < 0;$$

$$2^{-34} \leqslant X \leqslant 2(1 - 2^{-35}),\ \text{если}\ X > 0;\ X = 0;$$

десятичных чисел с плавающей запятой с записью в обычной форме, например $X = +25{,}374$, с точностью до 8 значащих цифр в диапазоне $10^{-19} < |X| < 10^{19}$, точнее

$$-2^{63} \leqslant X \leqslant -2^{-64}\left(\frac{1}{2} + 2^{-27}\right)\ \text{при}\ X < 0,$$

$$\frac{1}{2} \cdot 2^{-64} \leqslant X \leqslant 2^{63}(1 - 2^{-27})\ \text{при}\ X > 0;\ X = 0;$$

целых десятичных чисел с точностью до 11 значащих цифр.

Внутри машины числа представляются в двоичном дополнительном коде.

---

* Далее вместо названия «Наири-К» будем писать «Наири».

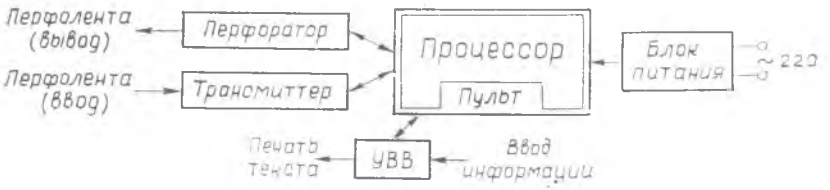


Рис. 6.

Средняя скорость вычислений для операций с фиксированной запятой ~1,5 тыс. операций/сек, а для операций с плавающей запятой ~600 операций/сек.

Память машины состоит из оперативного запоминающего устройства (ОЗУ), выполненного на ферритовых сердечниках, емкостью 4096 ячейки и долговременного запоминающего устройства (ДЗУ) на оксиферах емкостью 16384 ячейки, в котором постоянно хранятся обслуживающие и стандартные программы математического обеспечения машины.

Состав машины

В состав машины входят следующие устройства (рис. 6):

процессор с пультом управления;

печатающее устройство ввода—вывода (УВВ) на базе электрофицированной пишущей машинки «Консул-254»;

выходной ленточный перфоратор ПЛ-80/8;

трансмиттер FS-1500 для ввода информации с перфоленты;

блок питания машины.

Устройства ввода-вывода информации обеспечивают следующие режимы работы.

В режиме связи с процессором:

ввод информации с клавиатуры УВВ или трансмиттера;

вывод информации на печать или на перфоленту;

одновременный вывод информации на печать и на перфоленту.

В автономном режиме:

печать информации на УВВ с одновременной перфорацией;

распечатка информации с перфоленты на УВВ;

реперфорация ленты;

печать на УВВ, как на обычной пишущей машине.

Управление этими режимами производится с клавиатуры пульта управления, а также с пульта управления УВВ.

Скорость печати — 8-10 знаков в секунду.

Скорость считывания с перфоленты до 100 знаков в секунду.

Скорость перфорации до 75 знаков в секунду.

При вводе информации через трансмиттер используется бумажная шестидорожечная перфолента шириной 23 мм. Посредине ленты проходит синхронизирующая дорожка пробивок малого диаметра.

Рис. 7.

Для записи цифр на каждой строке используются четыре дорожки, читаемые справа налево (рис. 7). Запись производится в двоичной системе счисления. Каждой цифровой дорожке приписывается свой «вес» («1», «2», «4», «8»). Таким образом, для примера на рис. 7, на первой строке записана «1», на второй — «5», на третьей — «7» и т. д. Само число будет при этом записано в группе строк.

Все остальные символы записываются различными кодовыми сочетаниями пробивок на всех шести дорожках. Для отделения одной информационной записи от другой служат маркерные пробивки на 5-й дорожке, которые образуются при нажатии на УВВ клавиши «возврат каретки».

Функциональная схема процессора представлена на рис. 8.

Арифметическое устройство служит для выполнения арифметических и логических операций над числами и командами.

Устройство управления предназначено для автоматического управления последовательностью операций (заданных программой) в процессе решения задачи. В том числе:

автоматического ввода программы вычислений и исходных данных в машину;

выборки команд и чисел из ОЗУ и ДЗУ;

непосредственного управления операциями;

выдачи окончательных результатов из машины на печать или перфорацию;

контроля и управления работой машины оператором.

Пульт управления содержит сигнальные табло для индикации режимов работы и группы неоновых ламп для сигнализа-

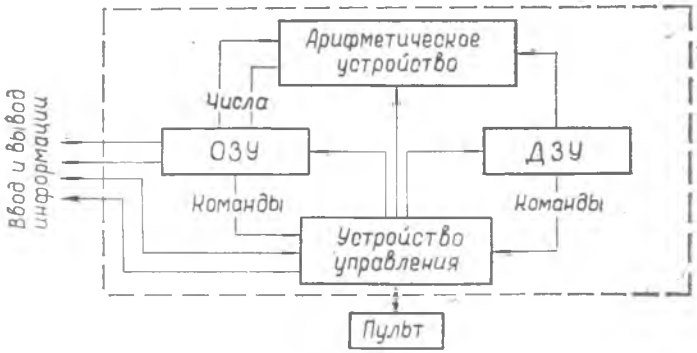


Рис. 8.

ции состояния триггеров различных регистров машины. На пульте расположены также кнопки и клавиши управления режимами работы. Основные из них имеют следующее назначение.

«Пуск 1» — клавиша начального пуска. Подготавливает все устройства машины к началу работы.

«Пуск 2» — клавиша повторного пуска. Используется для продолжения вычислений, если в процессе выполнения программы произошла остановка.

«Останов» — клавиша для остановки машины.

«Выдача памяти» — клавиша, используемая при выводе на печать содержимого некоторой группы ячеек памяти.

«Связь — Автономный» — клавиша обеспечивает работу устройств ввода — вывода в режиме связи с машиной или в автономном режиме, то есть их работу как независимых устройств.

Клавиши «Шаговый» и «Полуавтоматический» предусматривают пооперационное исполнение команд при каждом нажатии клавиши «Пуск 2».

Остальные клавиши используются для обеспечения различных режимов работы устройств ввода — вывода, отладочных работ и перехода к решению тестовых задач.

На панели пульта расположены также две кнопки включения и выключения машины.

Емкость оперативного запоминающего устройства (ОЗУ) составляет 4096 тридцати шести разрядных ячеек, в том числе 7 ячеек с фиксированными адресами: a_1, a_2, ..., a_7.

Условно ОЗУ разбито на две «страницы» (секции):

I страница — адреса 0÷2047,

II страница — адреса 2048÷4088.

Обращение к ячейкам соответствующей страницы обеспечивается подготовительными командами записи (*«з»* и *«з$_1$»*) путем формирования задающего признака β (см. описание служебных команд).

Долговременное запоминающее устройство (ДЗУ) имеет емкость 16384 ячейки памяти, по 36 двоичных разрядов каждая.* С ДЗУ возможен только односторонний обмен информацией — чтение. Записывать информацию в ДЗУ при решении задач нельзя.

В ДЗУ размещены программы псевдоопераций, стандартные программы решения некоторых типовых задач, транслятор для работы машины в режиме автоматического программирования, служебные и вспомогательные программы.

---

* Ячейки с 0—2047 имеют по 72 двоичных разряда.

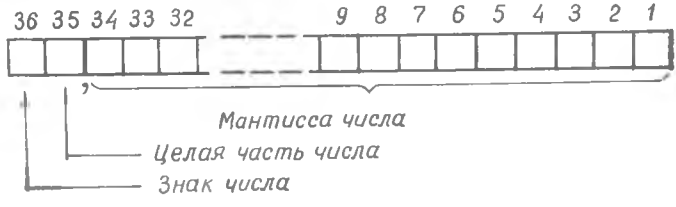
Глава 6

ПРЕДСТАВЛЕНИЕ ЧИСЕЛ В МАШИНЕ «НАИРИ»

ЭЦВМ «Наири» может оперировать с числами, представленными в форме с фиксированной или плавающей запятой, целыми, комплексными и длинными. Внутри машины эти числа изображаются в двоичном дополнительном коде. При этом разрядная сетка, состоящая из 36 двоичных разрядов, используется в соответствующих форматах.

§ 1. ЗАПИСЬ ЧИСЕЛ С ФИКСИРОВАННОЙ ЗАПЯТОЙ

При записи чисел в форме с фиксированной запятой (дробные числа) их абсолютная величина должна быть меньше двух $|X|<2$. В этом случае разрядная сетка имеет следующий формат:



Числа здесь изображаются в виде

$$\alpha_{36}\cdot 2^{1} + \alpha_{35}\cdot 2^{0} + \alpha_{34}\cdot 2^{-1} + \ldots + \alpha_{2}\cdot 2^{-33} + \alpha_{1}\cdot 2^{-34},$$

где α_i принимает значение 0 или 1.

Запятая фиксируется между разрядами α_{35} и α_{34}

α_{35} — разряд целой части числа,

α_{36} — знаковый разряд,

причем знак «+» представляется нулем, а знак «—» — единицей.

Размер разрядной сетки определяет диапазон изменения чисел и точность их представления.

Так, максимальное (по модулю 2) машинное число равно:

$$|X|_{\text{макс}} = 1,\overbrace{111\ldots11}^{n} = 2\cdot(1-2^{-(n+1)}),$$

а минимальное, отличное от нуля, машинное число равно

$$|X|_{\text{мин}} = \overbrace{0{,}000\ldots01}^{n} = 2^{-n},$$

где n — количество разрядов мантиссы числа.

Таким образом, диапазон машинных чисел определится как

$$2^{-n} \leqslant |X| \leqslant 2(1-2^{-(n+1)}).$$

Абсолютная ошибка представления чисел будет $|\Delta| \leqslant 0{,}5 \cdot 2^{-n}$.

§ 2. ЗАПИСЬ ЧИСЕЛ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

Если исходные числа представляются, в общем случае, неправильной дробью, то они записываются в машине в нормализованной форме

$$X = M_x \cdot 2^{\pm p}, \text{ при } 2^{-1} \leqslant M_x < 1,$$

где M_x — мантисса числа,

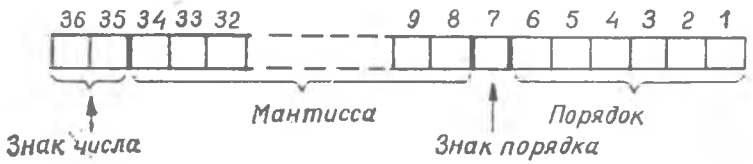
p — порядок числа.

Например:

$$X = -6{,}625_{10} = -110{,}101_2 = -0{,}110101 \cdot 2^{+11},$$

где $p = +11_2$ (3) указывает фактическое местоположение запятой.

Разрядная сетка машины будет иметь следующий формат:



Здесь максимальное (по модулю 2) число можно представить

$$|X|_{\max} = 0{,}\overbrace{111\ldots11}^{n} \cdot 2^{+\overbrace{11\ldots1}^{m}} = [1-2^{-n}] \cdot 2^{(2^m-1)},$$

минимальное число

$$|X|_{\min} = 0{,}\overbrace{100\ldots00}^{n} \cdot 2^{-\overbrace{11\ldots1}^{m}} = 2^{-1} \cdot 2^{-(2^m-1)} = 2^{-2^m},$$

и диапазон машинных чисел записывается как

$$2^{-2^m} \leqslant |X| \leqslant [1-2^{-n}] \cdot 2^{(2^m-1)},$$

где n — количество разрядов мантиссы,

m — количество разрядов порядка.

Максимальная величина относительной ошибки при записывании числа в ячейке ЭЦВМ равна:

$$|\delta|_{\max} = \frac{0{,}5 \cdot 2^{-n}}{2^{-1}} = 2^{-n}.$$

§ 3. ЗАПИСЬ ЦЕЛЫХ ЧИСЕЛ

Целые числа представляются в машине в следующей форме:

$$\alpha_{36} \cdot 2^{35} + \alpha_{35} \cdot 2^{34} + \ldots + \alpha_1 \cdot 2^0,$$

то есть их можно рассматривать как числа с фиксированной запятой, помещенной справа от первого разряда. Здесь α_{36} характеризует знак числа. Если $\alpha_{36} = 1$, то число отрицательное.

§ 4. ЗАПИСЬ КОМПЛЕКСНЫХ ЧИСЕЛ

Комплексные числа записываются в двух последовательных ячейках. В первой из них записывается действительная часть числа в форме с плавающей запятой, а в следующей — мнимая часть, также в форме с плавающей запятой.

§ 5. ЗАПИСЬ ДЛИННЫХ ЧИСЕЛ

Исходные числа можно записывать аналогично комплексным — в двух последовательных ячейках, при этом целая часть размещается в первой ячейке, а дробная часть — в следующей. Такие числа называются длинными. Обозначив двоичные разряды целой и дробной частей соответственно через α_l и β_l, длинное число можно представить в виде

$$\alpha_{36} \cdot 2^{35} + \alpha_{35} \cdot 2^{34} + \ldots + \alpha_1 \cdot 2^0 + \beta_{34} \cdot 2^{-1} + \ldots + \beta_1 \cdot 2^{-34},$$

причем всегда $\beta_{36} = 0$ и $\beta_{35} = 0$.

Следует заметить, что представление нецелых чисел в машине, вообще говоря, является приближенным. Например, десятичная дробь 0,1 в двоичной системе изображается периодической дробью и не может быть точно представлена конечным числом разрядов. Эту особенность следует учитывать при оценке точности решения задач.

Глава 7
СИСТЕМА КОМАНД ЭЦВМ «НАИРИ»

В связи с тем, что ЭЦВМ «Наири» является двухадресной машиной, каждая команда программы вычислений обычно содержит два адреса (A_1 и A_2) исходных чисел, над которыми производится та или иная операция. При этом результат операции, в большинстве случаев, записывается по второму адресу и в фиксированную ячейку a_2. В некоторых командах содержимое a_2 может быть использовано как одно из исходных чисел. Такие команды фактически становятся уже трехадресными (с адресами a_2, A_1, A_2).

Предусмотрена также возможность автоматического формирования команд или исполнение их в зависимости от определенного условия (условные и безусловные команды).

Все команды машины «Наири» разделяются на два класса, одному из которых соответствуют «машинные операции», другому — «псевдооперации». «Машинная операция» —это команда, выполняемая машиной по микропрограммному принципу 8. «Псевдооперация» — это подпрограмма, записанная в ДЗУ в машинных командах и выполняющая то или иное математическое действие. Всякая программа в конечном виде состоит из последовательности машинных операций.

Каждая команда во внешнем ее представлении изображается некоторой комбинацией символов, знаков и числовых параметров (см. таблицу команд в Приложении 1), и состоит из кода операции (Коп), адресной части (A_1 и A_2) и признака модификации (Θ).

Например: $cA_1\Theta A_2$ или *уп* $A_1\Theta A_2$ и т. п.

Код машинной операции записывается буквой, либо буквой с индексом. Например: *с* — сложение, u_1 — переход.

Коды псевдоопераций записываются двумя буквами. Например, *уп* — умножение чисел с плавающей запятой.

Параметр A_1 в зависимости от модификации команды может быть либо адресом (ОЗУ или ДЗУ), либо целым положительным числом.

Признак модификации команды (Θ) обозначается одной из букв: *н, к, п, л.*

Параметр A_2 всегда является вторым адресом ОЗУ.

Знак условия ($=, \neq, >, \geqslant, <, \leqslant, !, \infty$) пишется только в условных командах после второго адреса.

Признак формирования *(Ф)* записывается в конце команды знаком «+» только в том случае, если команда должна формироваться.

Адреса и числовые параметры во всех случаях записываются в десятичной системе счисления.

§ 1. МОДИФИКАЦИИ КОМАНД

В ЭЦВМ «Наири» существует несколько модификаций и соответственно несколько форматов команд (см. таблицу 5).

Для безусловных команд используются четыре модификации (по признаку Θ): *н, к, п, л.*

1. В модификации «*н*» (накопитель) параметр A_1, занимающий разряды 12÷25, является адресом ОЗУ или ДЗУ. Если $A_1 \leqslant 2047$, то это есть адрес ОЗУ. Если $A_1 \geqslant 2048$, то это — адрес ДЗУ. Над содержимым адресов A_1 и A_2 (или a_2) выполняется действие, указанное в коде операции. Результат записывается по второму адресу и в a_2.

Например, команда: *с*143*н*29 — означает, что нужно сложить содержимое 29 и 143 ячеек, результат записать в 29-ю ячейку.

2. В модификации «*к*» (короткий параметр) целое число A_1, записанное в 12÷25 разрядах команды, выделяется, сдвигается вправо на 11 разрядов и операция выполняется над полученным числом и содержимым ячейки A_2 (или a_2).

Например, требуется уменьшить число 10, записанное в 300-й ячейке, на единицу. Команда будет такой:

*в*1*к*300.

Результат: 10—1=9 получим в ячейке 300 и в ячейке a_2.

3. Модификация «*п*» (правый параметр) аналогична предыдущей, но здесь целое число A_1 записывается в 8÷25 разрядах команды и, следовательно, может иметь большую величину (до 262143 включительно), чем в модификации «*к*» (максимальное число при этом 16383). Зато адрес A_2 здесь не должен превышать 127.

Например, требуется сложить число 25, записанное в ячейке 90 с числом 162255. Команда будет иметь вид:

*с*162255*п*90.

Результат: 162255+25=162280 будет записан в ячейке 90 и a_2.

4. В модификации «*л*» (левый параметр) целое число A_1 записанное в 8÷25 разрядах команды, выделяется, сдвигается влево на 11 разрядов и операция выполняется над полученным числом и содержимым ячейки A_2 (или a_2).

Например, требуется к содержимому 125-й ячейки, в которой находится целое число 4 (единица в 3-м разряде), приформировать единицу в 19-м разряде. Команда в этом случае имеет вид:

*с*1*л*125.

После исполнения команды сложения, результат $2^{18}+2^2=$ $=262144+4=262148$, получим в ячейке 125 и a_2.

МОДИФИКАЦИИ КОМАНД ЭЦВМ «НАИРИ»

Представление числового содержания двоичных разрядов сумматора	131072	65536	32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
разряды	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Безусловные команды

Модификация	Признак			Код операции	Ф	Параметр A_1	A_2 – адрес ОЗУ
Ж	0	0	0			A_1 – адрес ОЗУ или ДЗУ	$A_2 \leq 1023$
К	0	0	1			$A_1 \Rightarrow$ в 1÷14 разр. сумматора	$A_2 \leq 1023$
П	0	1	0			$A_1 \Rightarrow$ в 1÷18 разряды сумматора	$A_2 \leq 127$
Л	0	1	1			$A_1 \Rightarrow$ в 19÷36 разряды сумматора	$A_2 \leq 127$

Условные команды

Модификация	Признак			Код операции	Ф	Параметр A_1	m	Условие	A_2
Жу	1	0	0			A_1 – адрес ОЗУ или ДЗУ	$m \leq 15$	z	$A_2 \leq 15$
Пу	1	1	0			$A_1 \Rightarrow$ в 1÷14 разр. сумматора	$m \leq 15$	z	$A_2 \leq 15$
Лу	1	1	1			$A_1 \Rightarrow$ в 19÷32 разр. сумматора	$m \leq 15$	z	$A_2 \leq 15$

ПРЕДСТАВЛЕНИЕ ЧИСЕЛ С ФИКСИРОВАННОЙ ЗАПЯТОЙ
(дробные числа)

разряды	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	Знак	Ц.Ч.	Мантисса числа																																	

ПРЕДСТАВЛЕНИЕ ЧИСЕЛ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

разряды	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	Зн.1	Зн.2	Мантисса M_x																											Зн.Р	Порядок Р					

Для условных команд во внешнем коде после второго адреса пишется символ условия $z(\neq, =, >, \geqslant, <, \leqslant, !, \infty)$ и адрес ячейки m, содержимое которой проверяется на выполнение условия.

Например: $cA_1\Theta A_2zm$.

Здесь адреса A_2 и m не должны превышать 15, то есть $A_2 \leqslant 15$ и $m \leqslant 15$. Формат разрядной сетки для условных команд с модификациями *«н, п, л»*, приведен в таблице 5.

При исполнении такой команды вначале проверяется, удовлетворяет ли содержимое ячейки m заданному условию z. Если условие выполнено, команда исполняется, в противном случае машина переходит к следующей команде.

Проверка заданного условия производится путем сравнения содержимого ячейки m с нулем, то есть: $(m)=0$, $(m)\neq 0$, $(m)>0$, $(m)\geqslant 0$, $(m)<0$, $(m)\leqslant 0$. Условие, обозначенное знаком «!», считается выполненным, если на пульте управления нажата клавиша «Вариант». Условие «∞» выполняется при переполнении разрядной сетки.

Например, команда: $c34н12\geqslant 5$ исполняется следующим образом. Если в пятой ячейке находится неотрицательное число, то производится сложение содержимого 12 и 34 ячеек и результат записывается в 12 ячейку и a_2. Если же в пятой ячейке окажется число отрицательное, то эта команда пропускается и машина переходит к следующей команде.

Если в команде не указывается второй адрес — A_2 (например, в командах управления), — знак условия пишется непосредственно после признака модификации Θ. Если в условной команде не указывать адрес m проверяемой ячейки, то на выполнение условия z будет проверяться содержимое ячейки a_2, в которой хранится результат предыдущей операции.

Например, команда $c34н12\neq$ выполняется, если число в ячейке a_2 — ненулевое.

Модификации условных команд *«н, п, л»* исполняются аналогично соответствующим модификациям безусловных команд.

§ 2. ФОРМИРОВАНИЕ КОМАНД

При выполнении команды с признаком формирования (+) машина суммирует двоичный код команды с кодом, находящимся в первой ячейке ОЗУ. Вновь полученный код представляет собой исполнительную команду, которая и выполняется машиной. В запоминающем устройстве команда с признаком формирования сохраняется в первоначальном виде.

Пусть, к примеру, требуется в команде умножения: $y120н50$ в одном случае увеличить на единицу второй адрес, в другом — первый адрес, и в третьем — первый и второй адреса. Для этого нужно исходной команде приписать признак формирования:

*у*120*н*50+ и организовать в предварительно очищенной первой ячейке ОЗУ соответствующие константы формирования.

Практически это можно сделать следующим образом:

1 случай	2 случай	3 случай
. . . . .	. . . . .	. . . . .
*с*1*п*1	*с*2048*п*1	*с*2049*п*1
. . . . .	. . . . .	. . . . .
. . . . .	. . . . .	. . . . .
*у*120*н*50+	*у*120*н*50+	*у*120*н*50+

В первом случае к содержимому (нулю) первой ячейки прибавляется единица в младшем разряде. При исполнении программы эта единица складывается с младшим разрядом второго адреса и исходная команда будет исполняться как: *у*120*н*51.

Во втором случае к содержимому первой ячейки прибавляется единица в 12-м разряде (число 2048), соответствующем младшему разряду первого адреса. Тогда исходная команда будет исполняться как *у*121*н*50.

Наконец, в третьем случае, к содержимому первой ячейки будут прибавлены единицы в 1-м и 12-м разрядах (число 2049) и исполнительная команда будет *у*121*н*51.

§ 3. УСЛОВНЫЕ ОБОЗНАЧЕНИЯ

Для удобства дальнейшего изложения введем следующие обозначения:

(A) — содержимое ячейки A памяти машины;

Θ — обобщенный символ признака модификации (*н*, *к*, *п*, *л*);

$[A_1]_Θ$ — содержание параметра A_1 в зависимости от модификации;

Например: $[A_1]_н = (A_1)$, $[A_1]_к = A_1^{14}$ — числовой код в разрядах 1÷14 и т. д.;

$[A_1]_Θ^п$ — число в форме с плавающей запятой, равное целому числу $[A_1]_Θ$. При этом A_1 не меньше нуля;

$(A)^к$ — комплексное число в ячейках A и A+1;

$(A)^д$ — длинное число в ячейках A и A+1;

$\overrightarrow{(A)}$ — сдвиг вправо кода (A);

$\overleftarrow{(A)}$ — сдвиг влево кода (A);

→A — засылка результата операции в ячейку с адресом A;

$→A^д$ — запись длинного числа в ячейки A и A+1;

$→A^к$ — запись комплексного числа в ячейки A и A+1.

Таким образом, например, содержание команды $cA_1ΘA_2$ (сложение) может быть представлено в символической форме, как

$$(A_2) + [A_1]_Θ → A_2,\ a_2.$$

Представляется также целесообразным объединить команды по характеру операций в следующие группы:

арифметические операции;

операции посылки;

операции управления;
операции вычисления элементарных функций;
логические операции;
операции над кодами;
служебные операции.

Полная таблица команд ЭЦВМ «Наири-К» приведена в приложении 1.

§ 4. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ

Операции с числами в форме с фиксированной запятой. Для выполнения арифметических действий над числами с фиксированной запятой предусмотрены две группы команд:

I c, c_1, $в$, $в_1$, $у$, $у_1$, $д$, $д_1$, $в_4$,

II c_2, c_3, $в_2$, $в_3$, $у_2$, $у_3$, $д_2$, $д_3$.

Эти группы аналогичны и отличаются лишь тем, что в командах I группы не предусмотрен анализ результата на переполнение разрядной сетки. Если результат не удовлетворяет условию: $-2<x<2$, машина продолжает считать, выдавая неверные результаты. В командах II группы результат анализируется на переполнение следующим образом. Если абсолютная величина результата больше или равна единице, то адрес следующей команды запоминается в ячейке 2046 и управление передается по адресу 2047.

Например, требуется сложить число X, находящееся в ячейке 50, и число Y, находящееся в ячейке 30, и в случае переполнения разрядной сетки перейти к программе, начинающейся с 300 ячейки.

Для этого необходимо предварительно в 2047 ячейке образовать команду «$и300п$».

Пусть команда сложения находится в ячейке 200:

$$200 : c_2 30 н 50.$$

Если при выполнении операции произойдет переполнение, то в ячейку 2046 посылается номер 201 и управление передается по адресу 2047, а затем — 300.

Если при переполнении требуется остановить машину, то в ячейку 2047 нужно записать команду «$к0н$».

В командах «c, c_2, $в$, $в_2$, $у$, $у_2$, $д$, $д_2$» действия выполняются над числами (A_2) и $[A_1]_Э$ и результат записывается в ячейки A_2 и a_2. В командах «c_1, c_3, $в_1$, $в_3$, $у_1$, $у_3$, $д_1$, $д_3$» операции производятся над числами (a_2) и $[A_1]_Э$, а результат записывается в ячейки A_2 и a_2, то есть эти команды выполняются как трехадресные. Для команды $в_4$ — вычитание модулей чисел с фиксированной запятой, запись в A_2 не производится.

Следует отметить, что во всех рассмотренных операциях первым операндом является число, записанное по второму адресу A_2 (или a_2), а вторым — число, заданное по $[A_1]_Э$.

Например, команда вычитания $вA_1нA_2$ будет исполняться, как $(A_2) - (A_1) \longrightarrow a_2$, A_2, а команда $в_1A_1ΘA_2$ — как:

$$(a_2) - [A_1] \rightarrow A_2, a_2.$$

Операции с целыми числами. Для выполнения операций сложения и вычитания целых чисел могут быть использованы команды рассмотренных выше групп. Однако, умножение и деление целых чисел осуществляется с помощью псевдоопераций *«ут»* и *«дт»*. При выполнении этих команд коды (A_2) и $[A_1]_Θ$ обрабатываются как целые числа. Результат по абсолютной величине должен быть меньше 2^{35}, в противном случае печатается знак «∞» и машина останавливается. Для команды *«дт»* целая часть результата засылается в A_2, а дробная часть в ячейку 3.

Операции с числами в форме с плавающей запятой. Группа команд *«сп, вп, ов, уп, дп, од, сб, вб, уб; дб, сс, вс, дс, ус, вм»* используется для обработки чисел, представленных в форме с плавающей запятой. Здесь допускаются все рассмотренные ранее модификации и выполняются все арифметические действия. Однако, при выполнении первых шести и последнего из этих операций, первым операндом является число, заданное по $[A_1]_Θ$ вторым — число, записанное по второму адресу A_2. Исключение еще составляют команды обратного вычитания *«ов»* и обратного деления *«од»*. При выполнении группы команд *«сб, вб, уб, дб»* запись по второму адресу не производится. Для группы команд *«сс, вс, ус, дс»* первым операндом является число, записанное в a_2 (сумматоре).

Например, требуется вычесть число 5,2, записанное в ячейке 29, из числа 12,6, записанного в ячейке 30. Команда будет иметь вид:

$$вп30н29.$$

Результат: 12,6—5,2=7,4 получим в ячейке 29 и a_2. Если же использовать для тех же целей команду обратного вычитания, то ее следует записать, как:

$$ов29н30.$$

При использовании в командах рассматриваемой группы модификации *«л»*, число, записанное как параметр A_1, должно быть представлено в форме с плавающей запятой. В модификациях *«к, п»*, число, записанное в A_1, переводится машиной в форму с плавающей запятой.

Если, например, требуется к числу X, записанному в ячейке 100, прибавить 0,5, то последнее в форме с плавающей запятой можно представить разрядным кодом 2^{34}.

Здесь показатель равен нулю, имеется только мантисса. При модификации *«л»* происходит сдвиг влево на 11 разрядов числа, записанного в A_1. Значит, в 23-м разряде кода должна быть записана 1. Поэтому, число 0,5 в форме с плавающей запятой

для модификации *«л»* будет представлено как параметр $A_1 = 32768 = 2^{23}$, а соответствующая команда:

*сп*32768*л*100.

Вычитание модулей (абсолютных величин) чисел (команда *вм*A_1ΘA_2) производится только для чисел в форме с плавающей запятой. При этом из содержимого A_2 вычитается абсолютная величина содержимого A_1 и результат засылается в ячейки A_2 и a_2.

Например, из числа «+25,2», находящегося в ячейке 100 нужно вычесть абсолютную величину числа «—8,2», находящегося в ячейке 105.

Команда имеет вид: *вм*105*н*100.

Результат: 25,2—8,2=17 получим в ячейке 100 и в ячейке a_2.

Операции с комплексными числами. Для выполнения арифметических действий с комплексными числами используются команды *«ск, вк, ук, дк»*. Модификации этих команд имеют иное значение, чем во всех ранее рассмотренных командах. Именно, в модификациях *«к»* и *«п»* параметр A_1 понимается как адрес; в модификации *«н»* параметр A_1 — адрес ячейки, в которой записан адрес первого числа. Таким образом, команда с модификацией *«н»* будет исполняться, как операция второго ранга. Модификация *«л»* для этой группы команд не имеет смысла. В этой связи рекомендуется использовать модификации *«к»* и *«п»* в безусловных командах и модификацию *«п»* — в условных.

При выполнении этих команд отдельно обрабатываются действительные части, находящиеся в ячейках A_1 и A_2 и мнимые части — в ячейках A_1+1 и A_2+1. Результат записывается в ячейках A_2 и A_2+1. При этом действительные и мнимые части исходных чисел и результата выражаются в форме с плавающей запятой.

Операции с длинными числами. Для выполнения операций над длинными числами используются команды *«сд, вд, уд, дд»*, которые также рекомендуется употреблять в модификациях *«к»* и *«п»*. Значение модификаций в этих командах такое же, как и в командах для комплексных чисел. Если истинный результат x не удовлетворяет условию

$$2^{-34} \leqslant |x| < 2^{34},$$

то печатается ∞ и машина останавливается.

Например, по команде: *сд*50*к*60 производится сложение длинных чисел, находящихся в ячейках 50, 51 и 60, 61; результат записывается в ячейки 60, 61.

§ 5. ОПЕРАЦИИ ПОСЫЛКИ

К этой группе относятся команды передачи числа: *«п»*, *«$п_1$»*, *«от»* и *«оп»*.

1. По команде: *п*A_1*н*A_2 код числа из ячейки A_1, записывает-

ся по адресу A_2 и a_2. Код числа в ячейке A_1 сохраняется. Команды с модификациями $\Theta = к, п, л$ применяются для формирования кодов в ячейке A_2. Например, команда: $п2048кA_2$ засылает единицу в 12 разряд ячейки A_2; в остальных разрядах будут нули. Команда: $п1023кA_2$ засылает единицы в $1 \div 10$ разряды ячейки A_2 и нули во все остальные разряды.

Команды: $п0пA_2$ или $п0кA_2$ применяются для очистки ячейки A_2.

Если команда имеет вид: $пA_1\Theta$, то передача происходит только в ячейку a_2. Например, команда: $п0п$ служит для очистки ячейки a_2.

2. По команде: $п_1 0 \Theta A_2$ код числа из ячейки a_2 передается в A_2.

3. По команде: $отA_1\Theta A_2$ длинное число переписывается из ячеек A_1 и A_1+1 в ячейки A_2 и A_2+1.

4. По команде: $опA_1\Theta A_2$ происходит передача числа с обратным знаком из A_1 в форме с плавающей запятой в A_2 и a_2.

§ 6. ОПЕРАЦИИ УПРАВЛЕНИЯ

Команды группы «$и$, $и_1$, $и_2$, $и_3$, $е$, $е_1$, $е_2$, $е_3$, $х$» дают возможность изменять порядок выполнения программы, а также осуществлять пропуск такта.

1. **Переход (передача управления) без возврата** «$и$» (команда $иA_1\Theta A_2$. Команда аналогична команде $и_1$. При исполнении команды сначала «$и(СчК)_п \rightarrow A_2$, затем младшие 14 разрядов кода $[A_1]_\Theta$ передаются в счетчик команд (СчК). Таким образом, эта команда осуществляет переход к команде, находящейся в ячейке с адресом $[A_1]_\Theta$. Содержимое ячейки a_2 сохраняется.

Например, команда: $и100п$ передает управление по адресу 100, а команда: $и100н$ передает управление по адресу, который записан в младших 14 разрядах ячейки 100. Команда: $и150п<$ условная, она выполняется следующим образом: если число в ячейке a_2 (результат предыдущей операции) отрицательное, то производится переход к команде по адресу 150, в противном случае исполняется следующая команда.

2. **Переход с возвратом** «$и_1$» (команда $и_1A_1\Theta A_2$). Эта команда удобна при выходе на подпрограмму. При исполнении команды управление передается по адресу, указанному в коде $[A_1]_\Theta$ [14], а в ячейку с адресом A_2 записывается команда: $иNп$, где N — адрес команды, следующей за командой: $и_1A_1\Theta A_2$. В конце подпрограммы можно написать команду: $иA_2п$.

Например, если в 120 ячейке записана команда:

$$120 : и_1 200к40,$$

то сначала в ячейке 40 сформируется команда «$и121п$», в счетчик команд запишется число 200 и машина передаст управление в ячейку 200.

С помощью команды «u_1» можно несколько раз выходить на одну и ту же подпрограмму с автоматическим возвратом к основной программе.

3. **Безусловный переход** «u_2». Команда имеет вид

$$u_2 A_1 \Theta$$

и применяется, в основном, в ДЗУ при выходе из псевдоопераций.

4. **Переход по ключу** «u_3» (команда $u_3 A_1 \Theta$). Эта команда выполняется, если на пульте управления нажата клавиша «Ключ». В остальном она аналогична команде «u». Чаще всего ее используют в длинных программах в случае необходимости сделать разветвление или узнать промежуточный результат вычислений и т. п. В случае 3, 4 содержимое (a_2) сохраняется.

5. **Формирование командного адреса** «e» (команда $e A_1 \Theta A_2$). При исполнении этой команды происходит формирование нужного командного адреса путем суммирования текущего содержимого счетчика команд с кодом $[A_1]_\Theta$ [14]. Результат записывается в ячейки A_2, a_2, a_3. Если $[A_1]_\Theta = 0$, то в A_2, a_2 и a_3 заносится содержимое счетчика команд. Напомним, что в счетчике команд находится число, на единицу большее адреса исполняемой команды, так как после передачи очередной команды из запоминающего устройства в регистр команд содержимое счетчика команд увеличивается на единицу.

6. **Относительный безусловный переход** «e_1» (команда $e_1 A_1 \Theta A_2$). При исполнении этой команды содержимое счетчика команд суммируется с кодом $[A_1]_\Theta$ [14] и результат засылается в счетчик команд. Иными словами эта операция будет означать: «перейти через столько-то команд» или «вернуться назад на столько-то адресов». Чаще всего эта команда используется в модификациях «n» или «$к$», но можно использовать и модификацию «$н$». Особенностью команды «e_1» является то, что в ней не указываются конкретные адреса, а задается лишь относительный переход на столько-то адресов «вперед» или «назад». Поэтому содержимое команды «e_1» не зависит от места программы в памяти машины. В A_2 записывается команда возврата «u(СчК)n», содержимое a_2 сохраняется.

Предположим, что команда $e_1 A_1 n$ записана в ячейке с адресом $К_i$ и требуется сделать переход к команде с адресом $К_n$. Если переход делается «вперед», то есть $К_n > К_i$, то A_1 подсчитывается по формуле

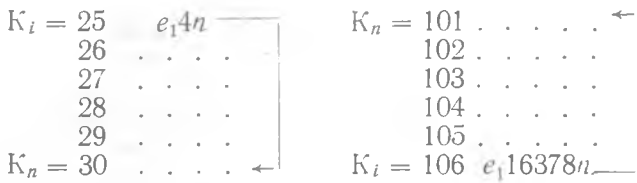
$$A_1 = К_n - К_i - 1.$$

Если переход делается «назад», то есть $К_n < К_i$, то A_1 подсчитывается по формуле

$$A_1 = 16383 - (К_i - К_n).$$

Здесь число 16383 представляет максимальное содержимое (емкость) счетчика команд.

Например:



7. **Относительный условный переход по набору** «e_2» (команда $e_2A_1\Theta A_2$). При исполнении этой команды сначала проверяется совпадает ли $[A_1]_\Theta$ с содержимым сумматора a_2, то есть предыдущим результатом. Если (a_2) совпадает с $[A_1]_\Theta$, то есть $[A_1]_\Theta \oplus (a_2) = 0$, то в счетчик команд засылается A_2+ (СчК) — результат, полученный сложением A_2 и содержимым счетчика команд. Если (a_2) не совпадает с $[A_1]_\Theta$, то в счетчик команд засылается номер следующей по порядку команды. Эта команда может использоваться для выхода из циклов, для разветвления программы. Например, если нам нужно 10 раз в цикле просчитать какую-то формулу, мы можем написать программу следующим образом.

Пусть программа начинается с 200-й ячейки:

200) $n0\kappa300$
201) $n0\kappa301$
202) $cn10\kappa301$
. . . . . .
. . . . . .
к+0) $cn1\kappa300$
к+1) e_2301n1
к+2) $u203\kappa$
. . . . . .

Здесь в ячейках 203÷к описаны вычисления, производимые в цикле. (к+2) команда делает переход к повторному вычислению (зацикливает). (к+1) команда производит выход из цикла. Команды в ячейках 200 и 201 устанавливают нули в 300 и 301 ячейках. В 202 ячейке находится команда, задающая длину цикла.

8. **Относительный условный переход по разряду** «e_3» (команда $e_3A_1\Theta A_2$). Команда осуществляет переход при условии, что имеется единичка в соответствующем разряде предыдущего результата, то есть в a_2. Сначала происходит логическое умножение содержимого a_2 на $[A_1]_\Theta$, затем результат сравнивается на совпадение с $[A_1]_\Theta$. Если есть совпадение, то в счетчик (СчК) засылается сумма содержимого A_2 и счетчика команд, в противном случае выполняется следующая команда.

В случаях 7, 8 скачок перехода не больше 2048, так как $A_2 < 2048$. При выполнении команд содержимое a_2 сохраняется.

К командам управления также относятся команды записи «*з*» и «$з_1$».

9. **Операция** «*з*» (команда $зA_1ΘA_2$) служит подготовительной для последующих команд при обращении к 2 странице ОЗУ. Последующие команды не должны быть псевдооперациями. Для характеристики обращения к I или II странице служит фиксированная ячейка a_7. В соответствующих разрядах a_7 записывается признак обращения $\beta_i (i=1, 2, 3)$. Если $\beta_i=0$, то мы работаем с первой страницей, если $\beta_i=1$, то со 2-й. Здесь β_1 (26 разряд a_7) — признак модификации адреса выбираемой команды, β_2 (27 разряд a_7) — признак модификации $[A_1]_н$ выбранной команды и β_3 (28 разряд a_7) — признак модификации адреса A_2 выбранной команды.

Все последующие команды будут модифицироваться, если $\beta_i=1$ $(i=1, 2, 3)$. Модификация команд происходит до тех пор, пока β_i $(i=1, 2, 3)$ не станут равными нулю.

Таким образом, при $\beta_i=1$ команда выбирается из адреса (СчК)+2048 [для (СчК) <2048].

При $\beta_2=1$ содержимое читается из адреса $[A_1]_н+2048$.

При $\beta_3=1$ результат записывается по адресу A_2+2048.

Если операция условная, то при $\beta_1=1$ номер команды выбирается независимо от условия со 2-й страницы.

Если же $\beta_2=1$, то условие проверяется с содержимым ячейки $(m \leqslant 15)$. При $\beta_3=1$ условие проверяется с содержимым сумматора или содержимым ячейки $m+2048$ $(m \leqslant 15)$. Например, если в 28 разряде ячейки a_7 стоит 1, то команда *сп*100*н*101> >13 выполняется при условии, что число, записанное в 2061 (=2048+13) больше 0. При этом складываются содержимые 100-й и 2149-й ячеек и результат посылается в 2149-ю ячейку и в a_2.

По команде $зA_1ΘA_2$ в фиксированную ячейку a_7 засылается $[A_1]_Θ$, а команда восстановления содержимого a_7 засылается в A_2. Здесь Θ рекомендуется брать «*н*», «*л*». Например, 1 в 28 разряд ячейки a_7 засылается так: *з*512*л*100. В ячейке 100 будет записано: $з_1(a_7)л$.

10. **Операция** «$з_1$» выполняется так же, как и «*з*», но нет команды восстановления. Команда записывается так: «$з_1A_1Θ$». Значение Θ рекомендуется брать «*н*» и «*л*».

11. **Холостая операция** «*х*» машиной не исполняется (пропускается) и происходит переход к следующей команде. При этом сохраняется результат выполнения предшествующей команды.

Команда имеет вид:

$$х0н$$

и во всех ее разрядах находятся нули.

§ 7. ОПЕРАЦИИ ВЫЧИСЛЕНИЯ ЭЛЕМЕНТАРНЫХ ФУНКЦИЙ

К этой группе операций машины «Наири» относятся:
извлечение квадратного корня ($\sqrt{x}$).

Команда: *кп*$A_1\Theta A_2$,

вычисление натурального логарифма *(lnx)*.

Команда: *ln*$A_1\Theta A_2$,

вычисление десятичного логарифма *(lgx)*.

Команда: *lg* $A_1\Theta A_2$,

вычисление экспоненциальной функции (e^x).

Команда: *ex*$A_1\Theta A_2$,

вычисление тригонометрических функций ($\sin x$, $\cos x$, $\operatorname{tg} x$).

Команда: *sn*$A_1\Theta A_2$, *cs*$A_1\Theta A_2$, tg$A_1\Theta A_2$,
(аргумент задается в радианах),

вычисление обратных тригонометрических функций

($\arcsin x$, $\arccos x$, $\operatorname{arctg} x$).

Команды: *as*$A_1\Theta A_2$, *ac*$A_1\Theta A_2$, *at*$A_1\Theta A_2$,

вычисление гиперболических функций ($\operatorname{sh} x$, $\operatorname{ch} x$, $\operatorname{th} x$).

Команды: *sн* $A_1\Theta A_2$, *cн* $A_1\Theta A_2$, *tн* $A_1\Theta A_2$,

вычисление бесселевых функций ($j_p(x)$, $y_p(x)$).

Команды: δ_j $A_1\Theta A_2$, $\delta_y A_1\Theta A_2$,

вычисление гамма функции $\Gamma(x)$.

Команда: *га* $A_1\Theta A_2$,

нахождение наименьшего числа [по модулю] ($\min\{a_i\}$).

Команда: *mn* $A_1\kappa A_2$,

нахождение наибольшего числа [по модулю] ($\max\{a_i\}$).

Команда: *mx*$A_1\kappa A_2$.

При выполнении всех операций аргумент выбирается из ячейки A_1, а результат посылается в ячейку A_2. Рекомендуется пользоваться модификацией *«н»*, но можно применять и модификации *«к», «п», «л»*. Однако, в этих случаях аргумент нужно записывать как параметр A_1 в форме с плавающей запятой, так как все перечисленные операции выполняются над числами с плавающей запятой. Кроме того, для операций *ln* и *lg* число должно быть нормализованным.

Рассмотрим несколько примеров:

1) вычислить $y=\sqrt{9,5}$, если число 9,5 находится в ячейке 100.

Команда будет иметь вид: *кп*100*н*200.

Результат операции $\sqrt{9,5}=3,082$ получим в ячейке 200;

2) вычислить $y=\sqrt{5}$, причем число 5 представить как параметр A_1 в форме с плавающей запятой.

Запишем $5 = 101,0 \cdot 2^{\circ}$ разрядным кодом: $(2^{17}+2^{15}) \cdot 2^{19}$. Тогда в числовом выражении получим:

$(2^{17}+2^{15})+(2^{4}+2^{1}+2^{\circ}) = (131072+32768)+19=163859$.

Команда будет иметь вид: *кп*163859*п*100.

Результат операций $\sqrt{5}$ получим в ячейке 100;

3) вычислить ln25, если число 25 в форме с плавающей запятой находится в ячейке 30.

Команда будет иметь вид: ln30*н*30.

Результат операции ln25 получим в ячейке 30.

Примечания:

1. В операции $\sqrt{x}$ при отрицательном знаке подкоренного выражения машина печатает: «$x<0$» и останавливается.

2. В операции e^x, если $x>43$ машина печатает знак «∞» и останавливается.

3. В операциях ln$x$ и lg$x$ происходит останов, если аргумент $x\leqslant 0$, при этом печатается: $x\leqslant 0$.

4. В операциях sn и cs, если порядок x больше 27, то печатается $n=$«порядок» и ЭВМ останавливается.

5. В операции tg при $x=\pm\frac{\pi}{2}$ машина печатает «∞» и останавливается.

6. В операциях ac и as соответственно, если $|x|>1$ и $\{0>x$ и $x<1\}$, печатается $x>1$ и машина останавливается.

7. В операциях δ_j и δ_y величина $x(x<750)$ должна быть записана в ячейке A_2, параметр p командами используется как $[A_1]_{\Theta}^{n}$ и может быть положительным и отрицательным числом.

8. В операции *га*, если $x=0$ или целому отрицательному числу, машина печатает ∞ и останавливается.

9. В операциях *mx* и *mn* отыскиваются соответственно максимум и минимум группы чисел и их модулей. Эти числа располагаются последовательно в ячейках от A_1 до A_2.

Результат как min (max) получается в a_2, а min (max) модулей чисел посылается в ячейку 9.

§ 8. ЛОГИЧЕСКИЕ ОПЕРАЦИИ

К числу логических операций, выполняемых машиной «Наири», относятся: логическое сложение «*l*», «l_1», логическое умножение «*л*», «$л_1$» и сложение по модулю 2 «*м*», «$м_1$».

Операции *l, л, м* производятся над кодами (A_2) и $[A_1]_\Theta$, результат записывается в ячейку a_2.

Операции «l_1, $л_1$, $м_1$» производятся над кодами (a_2) и $[A_1]_\Theta$; результат записывается в ячейки A_2 и a_2.

1. Логическое сложение (команда $l[l_1]A_1\Theta A_2$) производится со всеми 36 разрядами кодов чисел по правилам дизъюнкции:

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

Эта операция употребляется для объединения (формирования) кодов.

2. Логическое умножение (команда $л[л_1]A_1\Theta A_2$) производится со всеми 36 разрядами кодов чисел по правилам конъюнкции:

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

Эта операция употребляется для выделения части кода.

3. Сложение по модулю 2 (команда $м[м_1]A_1\Theta A_2$) производится для кодов чисел или команд по правилам поразрядного сложения

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

Эта операция употребляется для сравнения двух кодов. Здесь критерием сравнимости двоичных чисел, складываемых поразрядно (по модулю 2) является результат, равный нулю.

Пример 1. Пусть в ячейках 120 и 121 записаны команды

120) $п2048н0$

121) $х155н100.$

При выполнении команд из ячеек $\alpha+0$ и $\alpha+1$

$\alpha+0)$ $l120н121$

$\alpha+1)$ $n_1 0н122$

в ячейке 122 будет код команды «$п2203н100$». Здесь все становится ясно, если написать команды в двоичном представлении:

120) 000 000 100 000 010 000 000 000 000 000 000 000

121) 000 000 000 000 000 001 001 101 100 001 100 100

122) 000 000 100 000 010 001 001 101 100 001 100 100

Пример 2. Если нужно выделить значение разрядов 12, 7, 6, 5, 4, 3, 2, 1 ячейки 100, то можно использовать команду «$л2175к100$», где $2175=2^{11}+2^6+2^5+2^4+2^3+2^2+2^1+2^0$. Результат получится в ячейке a_2.

Пример 3. Команды сложения по модулю 2 могут быть использованы для сравнения двух кодов. Результат команды «$м$» и «$м_1$» будет нулевым, если коды совпали, в противном случае выделяются несовпадающие разряды кодов.

Пусть в ячейках 300 и 1000 записаны коды:

300) 000 000 110 100 000 000 111 100 000 001 111 001

1000) 000 001 000100 010 001 110 001 000 000 000 000

Результат выполнения команды «*м*300*н*1000» будет в ячейке a_2 и имеет следующую двоичную запись:

000 001 110 000 010 001 001 101 000 001 111 001.

§ 9. ОПЕРАЦИИ НАД КОДАМИ

Сдвиг чисел. Сдвиги кодов чисел производятся операциями «*а*, a_1, *б*, $б_1$, *г*, $г_1$». В этих командах модификация «*л*» не употребляется.

1. *Арифметический сдвиг вправо* (команда $a[a_1]A_1\Theta A_2$). В операции «*а*» сдвигается код, находящийся в ячейке A_2, а в операции a_1 — код, находящийся в ячейке a_2. Результат записывается в ячейки A_2 и a_2. При выполнении операций «*а*» и «a_1» происходит сдвиг вправо на столько разрядов, сколько указано в младших семи разрядах кода $[A_1]_\Theta$. Если сдвигается положительное число, в освободившиеся разряды заносятся нули, если сдвигается отрицательное число — единицы.

2. *Логический сдвиг влево* (команда $б[б_1]A_1\Theta A_2$). В операции «*б*» сдвигается код, находящийся в ячейке A_2, а в операции «$б_1$» — код, находящийся в ячейке a_2. Сдвиг происходит влево на столько разрядов, сколько записано в младших семи разрядах кода $[A_1]$. В освободившиеся разряды заносятся нули. Результат операций *б* и $б_1$ записывается в A_2 и a_2.

Например, пусть в 100 и 355 ячейках записаны коды

100) 000 000 000 000 000 000 000 000 000 001 000

355) 000 000 000 000 100 000 000 000 000 000 001

и пусть выполняются команды

$$\alpha)\ б15к355$$

$$\alpha+1)\ a_1100н356.$$

Первая команда сдвигает на 15 разрядов влево код из ячейки 355. По второй команде производится сдвиг вправо результата предыдущей операции на столько разрядов, сколько записано в ячейке 100. Содержимое 100 равно 8, значит сдвиг производится на 8 разрядов вправо.

Таким образом, в 355 и 356 ячейках получаются следующие коды

355) 000 000 000 000 000 000 001 000 000 000 000 000

356) 000 000 000 000 000 000 000 000 000 010 000 000.

3. *Сдвиг длинного числа:* правый (команда $гA_1\Theta A_2$) и левый (команда $г_1A_1\Theta A_2$).

При исполнении команды сдвигается длинное число, находящееся в ячейках A_2 и A_2+1. Сдвиг происходит на число разрядов, указанное в младших семи разрядах кода $[A_1]_\Theta$. При выполнении правого сдвига «*г*» младшие разряды кода (A_2), вышедшие за разрядную сетку ячейки A_2, занимают освободившиеся старшие разряды ячейки A_2+1, начиная с 34, причем 35

и 36 разряды ячейки A_2+1 не используются. При левом сдвиге *«$г_1$»* старшие разряды кода (A_2+1), вышедшие за 34 разряд, занимают освободившиеся младшие разряды ячейки A_2.

Старшие разряды, вышедшие за разрядную сетку ячейки A_2 при левом сдвиге, и младшие разряды, вышедшие за разрядную сетку ячейки A_2+1 при правом сдвиге, теряются. При правом сдвиге в освободившиеся разряды записываются нули, если число было положительное, и единицы в противном случае.

Результат операции записывается в ячейки A_2 и A_2+1.

Нормализация чисел. 1. *Нормализация чисел «н», «$н_1$»* обеспечивается командами: *н*$A_1\Theta A_2$ и *$н_1$*$A_1\Theta A_2$.

Нормализованными считаются числа, имеющие вид: 00,1... или 11,0... Число нуль считается нормализованным числом.

Числа, имеющие вид:

01,0...; 01,1...; 10,0...; 10,1...; 11,10...; 00,01...,

считаются ненормализованными, причем в первых четырех примерах показаны случаи нарушения нормализации влево, а в последних двух — случаи нарушения нормализации вправо. Нарушение нормализации обнаруживается анализом 36, 35 и 34 разрядов сумматора. При нормализации числа происходит сдвиг числа вправо или влево на столько разрядов, сколько необходимо для получения нормализованного числа.

Командой *«н»* нормализуется код $[A_1]_\Theta$, а командой *«$н_1$»* — содержимое ячейки a_2. Нормализованное число записывается в ячейку A_2, а число сдвигов — в младшие разряды ячейки a_2 в дополнительном коде.

Пусть, например, в ячейке 100 записано число с фиксированной запятой: $0{,}25_{10}=(00{,}0100...)_2$. По команде *н*100*н*200 это число нормализуется и результат: $(00{,}1000...)_2=0{,}5_{10}$ получим в ячейке 200.

2. *Нормализация длинных чисел «нд»* (команда *нд*$A_1\Theta A_2$) употребляется для перехода от длинных чисел к числам в форме с плавающей запятой. По этой команде длинное число, записанное в ячейках A_1 и A_1+1, приводится к виду: $M_x\cdot 2^p$ и записывается в ячейку A_2.

3. *Денормализация чисел с плавающей запятой «дн»* (команда *дн*$A_1\Theta A_2$) предназначена для перехода от числа в форме с плавающей запятой к длинным числам. По этой команде код $[A_1]_\Theta$ рассматривается как число, заданное в форме с плавающей запятой. Выделяются целая и дробная части числа, затем целая часть записывается в ячейку A_2, а дробная — в ячейку A_2+1. Если порядок исходного числа больше 34, печатается знак «∞» и машина останавливается.

§ 10. СЛУЖЕБНЫЕ ОПЕРАЦИИ

Операции этой группы используются, в основном, для автоматического ввода и вывода информации из машины. Сюда относятся команды чтения, печати и остановки машины.

Чтение. *Операция чтения «ч»* (команда $ч0\Theta A_2$) используется при вводе информации с внешних устройств ОЗУ и записывается в любой модификации. При выполнении этой команды код вводимого символа записывается в ячейку A_2.

Например, если в ячейке 100 записана команда:

$$100 : ч0п120,$$

машина, дойдя до этого адреса, останавливается и ждет ввода символа (на панели управления появляется сигнал «Ком. ввод»). При наборе на клавиатуре печатающего устройства нужного символа (цифры, буквы, знака) в ячейку 120 запишется код, отображающий этот символ.

Печать. 1. *Операции печати символа «о, o_1»* (команды $оA_1\Theta$ и $o_1 0\Theta$). Эти операции предназначены для печати символа (цифры, буквы, индекса и др.). В машине «Наири» каждому символу печатающего устройства соответствует код, называемый внутренним кодом символа. Эти коды находятся в ячейках ДЗУ с 2176 по 2275. В приложении № 3 приведена таблица символов и их адресов.

При выполнении команды $оA_1н$ машина напечатает тот символ, внутренний код которого находится в ячейке с адресом A_1.

При выполнении команды $o_1 0н$, 12 младших разрядов содержимого ячейки, номер которой равен $(a_2)+2^{11}+2^7=(a_2)+2176$, печатается как один символ.

В этих командах рекомендуется использовать модификацию *«н»*. При других модификациях число $[A_1]_\Theta$ должно быть кодом того символа, который требуется напечатать. При выполнении команд *«о,o_1»* содержимое ячейки a_2 сохраняется.

Приведем следующий пример. Пусть требуется напечатать слово «проект». Для этого составляется группа команд:

$о2249н$	соотв. печати	$п$
$о2229н$	»	$р$
$о2226н$	»	$о$
$о2227н$	»	$е$
$о2248н$	»	$к$
$о2230н$	»	$т$

2. *Печать чисел с плавающей запятой «пп»* (команда $ппA_1\Theta q$). По этой команде содержимое $[A_1]_\Theta$ печатается в виде десятичного числа. Возможны два варианта печати в зависимости от числа q.

1) $0<q\leqslant 31$. В этом случае число печатается с q знаками после запятой и с одним пробелом в конце.

Например, требуется напечатать число 0,25, полученное в результате какой-либо операции в ячейке 40. Команда имеет вид:

$$ппn40n3.$$

Здесь $q=3$ определяет количество знаков после запятой. Таким образом, будет напечатано число 0,250.

2) $q=64+r$, где r — количество цифр после запятой и $0 \leqslant r \leqslant 31$. Пусть t — количество цифр числа до запятой и $r+t<15$. В этом случае число печатается с r знаками после запятой и делается столько пробелов, чтобы число вместе с пробелами занимало интервал в 14 машинописных знаков. В таком виде команда употребляется, когда требуется печатать результаты в несколько столбцов (колонок). Если $r+t \geqslant 15$, то печатается число с r десятичными цифрами после запятой и делается пробел.

Пусть, например, требуется вывести на печать в две колонки массивы значений x и y, расположенные в ячейках, начиная с 300 и 400 соответственно.

Программа вывода будет иметь вид:

```
. . . . . .
о2220н )
о2250н )      печать: x=
пп300н516 +
о2214н )
о2250н )      печать: y=
пп400н6 +
. . . . . .
```

и обеспечит печать по форме:

$x = 10{,}0625$	$y = 0{,}327028$
$x = 10{,}1250$	$y = 0{,}401596$
. . . . .	. . . . .
. . . . .	. . . . .

14 разрядов

3. *Печать дробных чисел «пч»* (команда $пчA_1\Theta q$). При выполнении этой команды код $[A_1]_\Theta$ рассматривается как число с фиксированной запятой, переводится в десятичную систему и печатается с q знаками после запятой. Рекомендуется модификация $\Theta=н$.

4. *Печать длинных чисел «пд»* (команда $пдA_1\Theta q$). Команда работает, как *«пп»*, при этом содержимое ячеек A_1 и A_1+1 печатается в виде десятичного числа с q знаками после запятой, $q=31$. Используются модификации $\Theta=к$, $п$.

5. *Печать целых чисел «пт»* (команда $птA_1\Theta$). В этой команде $[A_1]_\Theta$ рассматривается как целое число. Команда работает со всеми модификациями и в зависимости от них печатается содержимое или число A_1.

6. *Печать команд «пк»* (команда $пкA_I\Theta$). При выполнении этой операции содержимое A_I печатается как команда (во внешнем коде). Используется модификация $\Theta = н$.

Например, надо напечатать команду *п*10*п*35, находящуюся в ячейке 100.

Команда имеет вид: $пк100н$ и выдает на печать текст: *п*10*п*35.

7. *Печать содержимого «пс»* (команда $псA_I\Theta$). По этой команде код $[A_I]_\Theta$ печатается в виде двоичного числа, то есть набора единиц и нулей.

8. *Печать цифровых индексов «пи»* (команда $пиA_I\Theta$). По этой команде код $[A_I]_\Theta$ рассматривается как целое число, переводится в десятичную систему, печатается в виде числового индекса и делается пробел. Должно быть выполнено условие: $0 \leqslant [A_I]_\Theta < 1000$.

9. *Печать комплексных чисел «пм»* (команда $пмA_1\Theta q$). В этой команде выводится на печать число $a+bi$, где a есть содержимое ячейки $[A_1]_\Theta^{14}$, а b — содержимое ячейки $[A_1]_\Theta^{14}+1$ с q десятичными знаками после запятой.

Останов машины осуществляется командой $кA_1\Theta$ или $к_1 0\Theta$. По команде «*к*» машина останавливается и в сумматор передается код $[A_I]_\Theta$. Этот код можно прочитать на пульте управления в нижнем ряду лампочек. По команде «$к_1$» машина останавливается и в сумматор передается код, находящийся в ячейке a_2.

§ 11. ВВОД И ВЫВОД ИНФОРМАЦИИ

Начальный ввод. Перед решением задачи в память машины необходимо ввести исходные числовые величины и программу вычислений. Для первичной обработки и расшифровки поступающих данных служит специальная программа: «Дешифрация исходной информации» (ДИИ), постоянно находящаяся в ДЗУ. Эта программа начинает работать при нажатии на клавишу «Пуск-1».

При вводе чисел или команд в ОЗУ машины нужно задать начальный адрес ОЗУ, начиная с которого должна происходить запись, и признак информации — букву, определяющую вид информации:

к — для ввода команд;
в — для ввода восьмеричных чисел;
п — для ввода чисел с плавающей запятой;
т — для ввода целых чисел;
ч — для ввода дробных чисел;
д — для ввода длинных чисел;
м — для ввода комплексных чисел.

Затем вводят числа или команды, делая после каждого из них пробел или возврат каретки. По окончании ввода информационного массива одного вида печатают букву «*ш*». При этом машина снова переходит к началу программы ДИИ. После этого можно задать начальный адрес нового массива и новый признак информации.

Например, для ввода группы команд в ОЗУ начиная с 31 ячейки, записывают:

31*к*
. . . . . } Группа
. . . . . } команд
. . . . . }
*к*0*н*
ш

Если при вводе была допущена ошибка, то ее можно исправить следующим образом.

1. Если ошибка при вводе числа или команды обнаружена до того, как был сделан пробел или возврат каретки, нужно «испортить» код, вводя какой-либо несоответствующий символ. Машина такой код не примет и напечатает знак «*». После этого число или команду вводят снова.

2. Если ошибочный код обнаружен после того, как был сделан пробел или возврат каретки, надо напечатать букву «*ш*», адрес данного кода и признак информации. После этого снова произвести ввод.

Выдача памяти. Режим «Выдача памяти» предназначен для контроля содержимого ячеек памяти при отладке программ, а также для вывода программ на печать или на перфоленту. Для перехода к этому режиму следует нажать клавишу «Выдача памяти» или же нажать клавишу «Пуск-1», напечатать буквы «*вв*». Машина сделает «*вк*», «*пс*» и остановится на команде ввод, загорается табло «Ком. ввод». Затем печатают код, указывающий начальный адрес массива, признак информации, количество ячеек массива, признак формы выдачи и делают возврат каретки. После этого машина выдает на печать содержимое указанного количества ячеек памяти.

Например, код 120*к*10 означает: содержимое десяти последовательных ячеек, начиная со 120, рассматривать как команды, перевести во внешний код и напечатать. При выводе сначала печатаются номера адресов.

Код 150*п*20*п* означает: содержимое двадцати последовательных ячеек, начиная со 150, рассматривать как числа с плавающей запятой, перевести в десятичную систему и напечатать (без адресов).

Если количество ячеек массива больше 50, то после каждых 50 ячеек машина делает шесть переводов строки и останавливается, если нажата клавиша «Вариант». Продолжение вывода производится нажатием клавиши «Пуск-2».

В машине предусмотрена возможность прервать решение задачи, перейти к режиму «Выдача памяти», сделать необходимые выдачи и вернуться к решаемой задаче. Для перехода от решения к выдаче памяти следует нажать клавишу «Полуавтоматический», снова восстановив ее. После окончания операций по выводу интересующей информации, чтобы вернуться к решению задачи, нужно напечатать буквы «*уу*», и машина продолжит решение с того адреса, на котором было прервано решение задачи.

Выдача массивов на перфоленту. Для выдачи содержимого памяти на перфоленту требуется предварительно заправить ленту в перфоратор и нажать клавишу «Работа перфоратора» на пульте управления ЭПМ. Затем нажать клавиши «Выдача памяти» (при нажатой клавише «Вывод на печать») и «Вывод на перфорацию». Далее ввести с ЭПМ первый адрес, признак информации, количество кодов и признак формы выдачи — букву «*п*» или «*пк*». Если массив не нужно вывести на печать, то отжать клавишу «Вывод на печать», выполнить возврат каретки и нажать клавишу «Пуск-2». Машина перфорирует содержимое заданного массива следующим образом: в начале ленты будут пробиты начальный адрес массива и признак информации, затем — содержимое требуемого количества ячеек. Адреса ячеек не перфорируются. В конце массива перфорируется буква «*ш*».

Глава 8

ПРОГРАММИРОВАНИЕ

При составлении программ решения задач на ЭЦВМ «Наири» следует иметь в виду, что не все ячейки памяти машины (ОЗУ) можно использовать для размещения в них констант, переменных или команд, поскольку некоторые группы ячеек используются во время работы стандартных программ машины (в том числе программ псевдоопераций, тестов и др.). В приложении 4 указаны программы, размещенные в ДЗУ машины и в приложении 2 приведены используемые ими ячейки ОЗУ. Все остальные ячейки практически являются свободными.

Таким образом, при программировании можно всегда использовать в качестве рабочих ячейки с 13 по 15 включительно и с 29 по 1994, исключая из последних те немногие, что описаны в приложении 3.

В отдельных случаях можно резервировать и ячейки, которые оказываются свободными за счет неиспользуемых стандартных программ. Например, если в программу не входят операции «*дт*», можно использовать ячейку 10, а если не используется операция «*пм*», — то ячейку 12 и т. д.

Другая особенность программирования связана с тем, что ЭЦВМ «Наири» является двухадресной машиной и результат большинства операций записывается в ячейку ОЗУ по второму адресу. Прежнее содержимое этой ячейки автоматически стирается. Таким образом, чтобы избежать случайной потери нужной информации, рекомендуется выделить несколько ячеек ОЗУ (например, № 13÷15), которые можно будет использовать в качестве операционных и накопительных.

Наконец, так же, как и при программировании для любых ЭЦВМ, следует стремиться к экономичности программ, то есть составлять наиболее короткие и быстродействующие алгоритмы и программы.

Ниже рассматриваются особенности построения прямых, разветвляющихся и циклических программ для ЭЦВМ «Наири».

§ 1. ПРЯМЫЕ ПРОГРАММЫ

Наиболее простым случаем является программирование алгебраических выражений, вычисление которых сводится к последовательному выполнению указанных в формуле операций.

Рассмотрим следующий пример.

Пример 1. Требуется вычислить значение y

$$y = \frac{\sin x - x^4}{e^x + 2{,}5}$$

для некоторой заданной величины x.

Вычисления в данном случае целесообразно проводить в форме с плавающей запятой, поскольку, кроме константы «2,5», в формулу входят функции $\sin x$ и e^x, определение которых удобно реализуется псевдооперациями.

Разместим исходные данные в следующих ячейках ОЗУ:

29	30
2,5	x

Программу же будем записывать начиная, например, с 40 ячейки.

Рассмотрим вначале два варианта построения программы вычисления знаменателя.

I вариант

40 *ех*30*н*30 в ячейке 30 образуется значение e^x
41 *сп*30*н*29 в ячейке 29 получается знаменатель

2 вариант

40 *ех*30*н*10 в ячейке 10 образуется e^x
41 *сп*29*н*10 в ячейке 10 получается знаменатель

В первом варианте допущены следующие характерные ошибки. При выполнении первой команды результат операции (e^x) записывается в 30 ячейку, прежнее содержимое которой (то есть x) автоматически стирается. Однако значение x еще необходимо для вычисления $\sin x$ и x^4. Аналогичная ошибка содержится и во второй команде *сп*30*н*29, где стирается константа 2,5 и в 29 ячейку записывается значение знаменателя. Здесь, несмотря на то, что эта константа в дальнейших вычислениях не используется, целесообразно было бы ее сохранить, так как при повторных вычислениях функции y для других значений аргумента x константу потребуется заново вводить в 29 ячейку.

Второй приведенный вариант является правильным, поскольку в этом случае для вычислений и хранения промежуточных результатов используется операционная ячейка 10.

Учитывая рассмотренные выше ошибки, составим теперь программу вычисления числителя.

42	*п* 30*н*11	запись значения x в 11 ячейку
43	*уп*11*н*11	в 11 ячейке образуется x^2
44	*уп*11*н*11	в 11 ячейке образуется x^4
45	*sn*30*н*12	в 12 ячейке образуется $\sin x$
46	*вп*12*н*11	в 11 ячейке получается числитель

Для вычисления величины y потребуется только одна команда

47 *од*10*н*11 вычисление и запись y в 11 ячейку.

Если нужно напечатать полученное значение y и затем прекратить вычисления, то добавляются следующие команды:

48 *пп* 11*п*5 из 11 ячейки выводится на печать значение y в виде числа с плавающей запятой с пятью цифрами после запятой,

49 *о*2274*н* возврат каретки печатающего устройства в исходное положение и перевод строки,

50 *к*0*н* останов машины

Вся программа будет иметь вид:

№ ком.	Команды	Пояснения
40	ех30*н*10	$e^x \rightarrow$ «10»
41	*сп*29*н*10	$e^x + 2{,}5 \rightarrow$ «10»
42	*п*30*н*11	$x \rightarrow$ «11»
43	*уп*11*н*11	$x^2 \rightarrow$ «11»
44	*уп*11*н*11	$x^4 \rightarrow$ «11»
45	*sn*30*н*12	$\sin x \rightarrow$ «12»
46	*вп*12*н*11	$\sin x - x^4 \rightarrow$ «11»
47	*од*10*н*11	Вычисление y и запись в «11»
48	*пп*11*н*5	Печать значения y

49	*о2274н*	Возврат каретки
50	*к0п*	Останов.

Если потребуется повторить вычисления $y=f(x)$ для других значений аргументов, то их можно воспроизвести, записывая каждый раз x_i в 30 ячейку и запуская машину с команды № 40.

Следует отметить, что в исходных формулах может несколько раз встречаться одно и то же выражение. В этом случае программа окажется более экономичной, если вычислить такое выражение один раз и, записав в операционную ячейку, использовать его в дальнейшем как известную величину.

Пример 2. Требуется вычислить значение y по формуле

$$y=(a+b+c)^2+(a-b)/(a+c)-(a+c)$$

Здесь выражение $(a+c)$ встречается три раза и поэтому удобнее вычислить его в начальной части программы.

Разместим исходные величины в следующих ячейках ОЗУ:

30	31	32
a	b	c.

Программа

№ ком.	Команды	Пояснения
50	*п30н10*	$a \rightarrow$ «10»
51	*сп32н10*	$(a+c) \rightarrow$ «10»
52	*п10н11*	$(a+c) \rightarrow$ »11»
53	*сп31н11*	$(a+b+c) \rightarrow$ «11»
54	*уп11н11*	$(a+b+c)^2 \rightarrow$ «11»
55	*п30н12*	$a \rightarrow$ «12»
56	*ов31н12*	$(a-b) \rightarrow$ «12»
57	*од10н12*	$(a-b)/(a+c) \rightarrow$ «12»
58	*сп11н12*	$(a+b+c)^2+(a-b)/(a+c) \rightarrow$ «12»
59	*ов10н12*	$y \rightarrow$ «12»
60	*пп12н4*	Печать значения y
61	*о2274н*	Возврат каретки
62	*к0п*	Останов

Из программы видно, что операционные ячейки 10, 11 и 12 выполняют также функции накопительных.

Если исходные величины (a, b и c) рассмотренной задачи будут представлены только в виде целых чисел, то программу целесообразно составлять с использованием машинных операций. Результат получим также в виде целого числа.

Программа

№ ком.	Команды	Пояснения
50	*п30н13*	$a \rightarrow$ «13»

51	*с32н13*	$(a+c) \rightarrow$ «13»
52	*п13н11*	$(a+c) \rightarrow$ «11»
53	*с31н11*	$(a+b+c) \rightarrow$ «11»
54	*ут11н11*	$(a+b+c)^2 \rightarrow$ «11»
55	*п30н12*	$a \rightarrow$ «12»
56	*в31н12*	$(a-b) \rightarrow$ «12»
57	*дт13н12*	$(a-b)/(a+c) \rightarrow$ «12»
58	*с11н12*	$(a+b+c)^2+(a-b)/(a+c) \rightarrow$ «12»
59	*в13н12*	$y \rightarrow$ «12»
60	*пт12н4*	Печать значения y
61	*о2274н*	Возврат каретки
62	*к0п*	Останов

В этом случае нельзя использовать 10 ячейку в качестве операционной, так как она будет занята в псевдооперации деления целых чисел *«дт»*.

§ 2. РАЗВЕТВЛЯЮЩИЕСЯ ПРОГРАММЫ

Чаще всего разветвление вычислительных процессов связано с необходимостью выбора вида расчетных формул, в зависимости от выполнения некоторых критериальных условий, определяющихся в ходе вычислений. В соответствии с этим, при составлении программы, нужно предусмотреть несколько возможных направлений расчетов (ветвей), отвечающих заданным условиям.

Переход к какой-либо ветви вычислений выполняется посредством команд с условием, которые позволяют изменить последовательность хода программы в зависимости от получаемых результатов. В машине «Наири» имеется 8 условий: $=$; $\neq$; $>$; $\geqslant$; $<$; $\leqslant$; ∞; !, причем сравнение результатов производится для содержимого ячейки m ($m \leqslant 15$). Кроме того, могут быть использованы и команды безусловной передачи управления, например, в случае перехода в некоторый, общий для ряда ветвей, блок.

Рассмотрим следующий пример.

Пример 3. Требуется вычислить значение функции $f(x, y)$ по одной из двух формул при следующих условиях:

$$f(x, y) = \begin{cases} yt^2/s - x, & \text{если } x - y = 0, \\ x \cdot \sqrt{s + yt^2}, & \text{если } x - y \neq 0, \end{cases}$$

причем предполагается, что величины x и y заранее неизвестны и будут получены только в процессе решения некоторой предыдущей части задачи. Однако определены рабочие ячейки, в которых они будут находиться

50	51	52	53
x	y	t	s

Программу разместим в ОЗУ, начиная с ячейки 100.

Программа

	Команды	Пояснения
100	*п*52*н*10	Вычисление yt^2
101	*уп*10*н*10	и запись в 10 ячейку
102	*уп*51*н*10	
103	*п*50*н*11	
104	*ов*51*н*11	$(x-y) \rightarrow$ «11»
—105	*и*110*п* =	Переход на вычисление $f(x, y)$
106	*сп*53*н*10	Вычисление $x \mid \overline{s+yt^2}$
107	*кп*10*н*10	и запись в ячейку 10
108	*уп*50*н*10	
—109	*и*112*п*	Переход к команде 112
→110	*од*53*н*10	Вычисление $yt^2/s - x$
111	*ов*50*н*10	и запись в 10 ячейку
→112	*пп*10*н*5	Печать $f(x, y)$
113	*о*2274*н*	Возврат каретки
114	*к*0*п*	Останов

Как видно, в программе используются две команды, позволяющие изменить последовательность вычислений: это команды 105 и 109. Причем, команда 105 является командой перехода с условием, а вторая — командой безусловного перехода.

Приведем некоторые пояснения.

Команды 100÷102 обеспечивают вычисление общего для обеих ветвей выражения yt^2 и записывают его значение в 10 ячейку. С помощью команд 103 и 104 в ячейке 11 образуется значение разности $(x-y)$. Команда *и*110*п* = начинает исполняться с проверки условия равенства нулю результата предыдущего действия. В случае невыполнения условия (т. е. $x-y \neq 0$) машина пропускает команду 105 и переходит к следующей за ней группе команд 106÷108. Эта группа производит вычисление функции $f(x, y) = x\sqrt{s+yt^2}$. Следующая за ней команда безусловного перехода *и*112*п* передает управление на выполнение заключительных операций (печать, останов). Если же значение $(x-y) = 0$, то команда 105 передает управление в ячейку 110. Начиная с команды, расположенной в этой ячейке, машина вычисляет функцию $f(x, y) = yt^2/s - x$, печатает результат и останавливается.

Итак, программа, как и собственно задача, имеет две ветви, переход к вычислениям которых определяется командой *и*110*п* = в соответствии с заданным условием.

В том случае, когда ветвей программы больше двух, выбор направления хода вычислений осуществляется аналогичным образом.

§ 3. ЦИКЛИЧЕСКИЕ ПРОГРАММЫ

Вычислительные процессы и программы, содержащие многократно повторяющиеся участки, называются циклическими. Структурно такие программы обычно состоят из нескольких характерных блоков (рис. 9), обеспечивающих условия для повторения вычислительных циклов. В зависимости от назначения и способа организации циклических процессов их можно подразделить на простые циклы, циклы с переадресацией, кратные циклы, итерационные циклы.

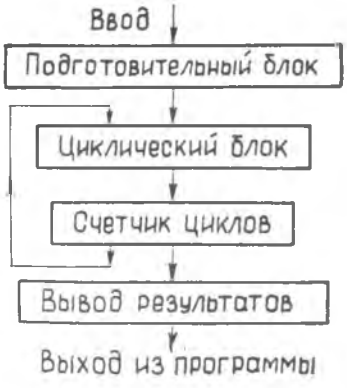


Рис. 9.

Следует отметить, что циклические вычислительные процессы с точки зрения их организации являются весьма экономичными, так как позволяют выполнять большое количество машинных вычислений при относительно небольших программах.

Рассмотрим основные принципы построения характерных циклических программ.

Простой цикл. Наиболее часто простые циклы используются при вычислении ограниченных рядов и цепных дробей, факториалов, табулировании функций и т. д.

Рассмотрим методику составления простейших циклических программ.

Пример 4. Требуется вычислить сумму членов степенного ряда.

$$s = x + 2x^2 + 3x^3 + 4x^4 + 5x^5 + 6x^6 + 7x^7$$

при заданном числовом значении x.

Исходные данные разместим в ячейках:

30	31
x	1

Программа

Команды		Пояснения
50	$n31n10$	Запись 1 в 10 ячейку
51	$n31n11$	Запись 1 в ячейку 11
52	$n0n12$	Очистка рабочей ячейки 12
53	$n7n13$	Запись 7 (числа циклов) в 13 ячейку

		1 цикл	2 цикл
→54	*уп*30*н*10	$x \cdot 1 \to$ «10»	$x \cdot x \to$ «10»
55	*п*10*н*14	$x \to$ «14»	$x^2 \to$ «14»
56	*уп*11*н*14	$1 \cdot x \to$ «14»	$2x^2 \to$ «14»
57	*сп*14*н*12	$0 + x \to$ «12»	$x + 2x^2 \to$ «12»
58	*сп*31*н*11	$1 + 1 \to$ «11»	$1 + 2 \to$ «11»
59	*в*1*п*13		
—60	*и*54*п*≠	Счетчик циклов	
61	*пп*12*н*5		
62	*о*2274*н*	Печать результата и останов машины	
63	*к*0*п*		

В этой программе команды 50÷53 являются подготовительными. Первые две из них засылают единицу в операционные ячейки 10 и 11. Команда 52 очищает ячейку 12, а команда 53 засылает число 7 (количество циклов) в ячейку 13, которая будет служить обратным счетчиком циклов.

Циклический блок представляет собой группу команд, последовательно вычисляющую значения членов ряда и их сумму. В этой группе команда 57 обеспечивает накопление суммы членов ряда в ячейке 12, а команда 58 вычисляет очередные коэффициенты.

Счетчик циклов образуется командой *в*1*п*13, которая в каждом цикле уменьшает содержимое счетчика (ячейки 13) на единицу, и командой *и*54*п*≠, передающей управление на повторение циклов, пока содержимое счетчика не станет равным нулю. После этого машина переходит к командам 61÷63 печати результата, полученного в ячейке 12, и остановки.

Однако следует заметить, что программа для рассматриваемого случая будет экономичнее, если использовать другой, более оригинальный алгоритм.

Программа

Команды		Пояснения	
50	*п*31*н*10	Запись 1 в 10 ячейку	
51	*п*0*п*11		
52	*п*0*п*12	Очистка ячеек 11 и 12	
53	*п*7*п*13	Запись 7 (числа циклов) в 13 ячейку	
		1 цикл	2 цикл
→54	*уп*30*н*10	$x \cdot 1 \to$ «10»	$x \cdot x \to$ «10»
55	*сп*10*н*11	$x + 0 \to$ «11»	$x^2 + x^2 \to$ «11»
56	*сп*11*н*12	$0 + x \to$ «12»	$x + 2x^2 \to$ «12»
57	*уп*30*н*11	$x \cdot x \to$ «11»	$x \cdot 2x^2 \to$ «11»
58	*в*1*п*13		
—59	*и*54*п*≠	Счетчик циклов	
60	*пп*12*н*5		
61	*о*2274*н*	Печать результата и останов машины	
62	*к*0*п*		

Для лучшего понимания программы рекомендуется самостоятельно рассмотреть еще 1÷2 цикла ее работы.

В общем эта программа получается более «быстрой», уменьшается количество команд и занятых операционных ячеек.

Приведем еще один пример решения задачи с использованием простого цикла.

Пример 5. Составить программу табулирования функции

$$F(x) = \cos(cx+p)$$

для неотрицательных c, p и аргумента x, изменяющегося в интервале от 2,5 до 5 с шагом $h=\Delta x=0{,}5$.

Число циклов здесь определяется по формуле:

$$n = \frac{b-a}{h} + 1,$$

где $(b-a)$ — разность между конечным и начальным значением переменной x,

h — шаг изменения значений переменных x,

n — количество вычислений значения функции $F(x)$

В нашем случае

$$n = \frac{5-2{,}5}{0{,}5} + 1 = 6.$$

Разместим исходные данные в ячейках:

30	31	32	33
2,5	c	p	0,5

Программа

	Команды		Пояснения	
50	*п6п13*	Запись числа циклов в ячейку 13		
51	*п30н11*	Засылка начального значения x_1 в яч. 11		
		1 цикл	2 цикл	
52	*п11н12*	$x_1 \rightarrow$ «12»	$x_2 \rightarrow$ «12»	
53	*уп31н12*	$c \cdot x_1 \rightarrow$ «12»	$c \cdot x_2 \rightarrow$ «12»	
54	*сп32н12*	$(cx_1+p) \rightarrow$ «12»	$(cx_2+p) \rightarrow$ «12»	
55	*cs12н12*	$\cos(cx_1+p) \rightarrow$ «12»	$\cos(cx_2+p) \rightarrow$ «12»	
56	*сп33н11*	$(x_1+\Delta x)=x_2 \rightarrow$ «11»	$(x_2+\Delta x)=x_3 \rightarrow$ «11»	
57	*пп12н5*	Печать значения $F(x)$		
58	*о2274н*			
59	*в1п13*	Счетчик циклов		
60	*и52п≠*			
61	*к0п*	Останов машины		

В рассматриваемой задаче при построении программы можно обойтись и без счетчика циклов, если критерием окончания циклических вычислений принять условие $x_{i+1} > x_{кон}$. Для это-

го нужно исключить подготовительную команду $n6n13$, ввести в состав исходных данных значение $x_{кон}$ (в ячейке 34) и перед проверкой условия $x_{i+1} > x_{кон}$ ввести команду сохранения x_{i+1} в ячейке 12. Далее, команду $в1n13$ заменить на $ов34н12$, а в команде условного перехода изменить адрес и условие $\neq$ на $\leqslant$.

Программа будет иметь следующий вид.

Программа

	Команды	Пояснения
50	$n30н11$	$x_1 \to$ «11»
→51	$n11н12$	$x_i \to$ «12»
52	$уп31н12$	$c \cdot x_i \to$ «12»
53	$сп32н12$	$(cx_i + p) \to$ «12»
54	$cs12н12$	$\cos(cx_i + p) \to$ «12»
55	$сп33н11$	$(x_i + \Delta x) \to$ «11»
56	$пп12н5$	
57	$о2274н$	Печать $F(x)$
58	$n11н12$	$x_{i+1} \to$ «12»
59	$ов34н12$	$(x_{i+1} - x_{кон}) \to$ «12»
60	$и51n\leqslant$	Переход при $(x_{i+1} - x_{кон}) \leqslant 0$
61	$к0n$	Останов машины

Необходимость использования в циклических программах команд с автоматически изменяющимися адресами возникает в тех случаях, когда приходится при повторении циклов поочередно обращаться к величинам, расположенным в некоторой группе ячеек памяти. Такие случаи встречаются, например, при суммировании массивов чисел, при операциях с матрицами и табличными величинами и т. п.

Для решения подобных задач на машине «Наири» используется первая ячейка ОЗУ, которая называется регистровой ячейкой формирования команд. Признаком формирования команды является наличие знака «+» в конце команды. В этом случае такая команда перед исполнением складывается с кодом, находящимся в данный момент в первой ячейке ОЗУ. Вновь полученная команда будет выполняться машиной вместо записанной в программе. Последовательно изменяя теперь содержимое первой ячейки получим возможность автоматической переадресации в соответствующих командах.

Рассмотрим несколько характерных примеров.

Пример 6. Составить программу вычисления суммы

$$s = a_1b_1 + a_2b_2 + a_3b_3 + a_4b_4 + a_5b_5.$$

Исходные величины находятся в ячейках:

30	31	32	33	34	35	36	37	38	39
a_1	b_1	a_2	b_2	a_3	b_3	a_4	b_4	a_5	b_5

Программа

	Команды	Пояснения	
		1 цикл	2 цикл
50	*п*0*п*1	Очистка регистровой ячейки 1	
51	*п*0*п*11	Очистка накопительной ячейки 11	
52	*п*5*п*13	Запись числа циклов в ячейку 13	
→53	*п*30*н*12+	$a_1 \rightarrow$ «12»	$a_2 \rightarrow$ «12»
54	*уп*31*н*12+	$a_1 \cdot b_1 \rightarrow$ «12»	$a_2 b_2 \rightarrow$ «12»
55	*сп*12*н*11	$0+a_1 b_1 \rightarrow$ «11»	$a_1 b_1 + a_2 b_2 \rightarrow$ «11»
56	*с*4096*к*1	Изменение содержимого 1 ячейки	
57	*в*1*п*13	Счетчик циклов	
→58	*и*53*п*≠		
59	*пп*11*н*5	Печать значения s	
60	*о*2274*н*		
61	*к*0*п*	Останов машины	

Циклический блок программы работает следующим образом. По команде *п*30*н*12+ машина сначала анализирует признак формирования «+» и складывает команду с содержимым ячейки 1. Поскольку в этой ячейке первоначально находился ноль, то после сложения сама команда пока не изменится и в 12 операционную ячейку будет записано число a_1 из ячейки 30. Аналогично, по команде *уп*31*н*12+ производится перемножение содержимого 31 и 12 ячеек с записью произведения $a_1 b_1$ в ячейку 12. По команде *сп*12*н*11 производится сложение содержимого 12 и 11 ячеек, с образованием в 11 ячейке суммы $S=0+a_1 b_1$.

По команде *с*4096*к*1 производится сложение числа 4096 в виде целого с содержимым первой ячейки. Так как в первой ячейке ранее был 0, то после выполнения команды в этой ячейке будет число 4096, выражаемое в двоичной системе как 1000000000000, что представляет при модификации «*к*» число 2 в адресе A_1.

Счетчик циклов работает обычным образом. При повторном выполнении циклического блока в командах *п*30*н*12+ и *уп*31*н*12+ адрес A_1 увеличится на 2 единицы. Вновь сформированные команды *п*32*н*12 и *уп*33*н*12 образуют в ячейке 12 произведение $a_2 b_2$ и т. д.

Если потребуется увеличить количество складываемых произведений, то нужно, не изменяя самой программы, лишь увеличить число циклов.

Пример 7. Составить программу пересылки массива чисел из одной группы ячеек памяти в другую.

Предположим, что массив состоит из 36 чисел, расположенных первоначально в 200÷235 ячейках ОЗУ. Переслать же его нужно в группе ячеек, начиная с 800.

Программа

	Команды	Пояснения
50	*п*0*п*1	Очистка регистровой ячейки 1
51	*п*36*п*10	Запись числа циклов в ячейку 10
→52	*п*200*и*800+	Циклический блок
53	*с*2049*к*1	
54	*в*1*п*10	Счетчик циклов
—55	*и*52*п*≠	
56	*к*0*п*	Останов машины

В циклическом блоке при каждом его исполнении адреса A_1 и A_2 команды 52 увеличиваются на единицу, благодаря чему и происходит пересылка чисел из одной последовательной группы ячеек в другую.

Кратные циклы чаще всего встречаются в тех случаях, когда вычисляемая функция является зависимой не только от собственного аргумента, но и от значения одной или нескольких других переменных, также изменяющихся в некотором интервале. Сюда относится, например, вычисление функции нескольких переменных, решение задач матричной алгебры и др. При этом образуется как бы система из вложенных один в другой циклов (внутренних и внешних)

Рассмотрим решение следующей задачи.

Пример 8. Составить программу вычисления функции

$$y=(z^2-x)\cdot\sin x,$$

при изменении переменной x в интервале $2\leqslant x\leqslant 4$ с шагом $\Delta x=0{,}5$ для каждого из значений другой переменной, начиная от $z_н=3$ и до $z_к=4{,}2$ при шаге $\Delta z=0{,}4$.

Разместим исходные данные в следующих ячейках:

30	31	32	33
$x_н$	Δx	$z_н$	Δz
2	0,5	3	0,4

Определим количество шагов для внутреннего цикла

$$n_1=\frac{x_к-x_н}{\Delta x}+1=\frac{4-2}{0{,}5}+1=5$$

и, соответственно, для внешнего:

$$n_2=\frac{z_к-z_н}{\Delta z}+1=\frac{4{,}2-3}{0{,}4}+1=4.$$

Присвоим переменным x и z индексы i и j, то есть:

$$x_i\ (i=0,1,2,3,4)\ \text{ и }\ z_j\ (j=0,1,2,3).$$

Программа

Команды	Пояснения
50 *n*0*n*10	Очистка рабочей ячейки 10 (для $\sum \Delta z$)
51 *n*4*n*11	Запись числа внешних циклов в ячейку 11
→52 *n*0*n*12	Очистка рабочей ячейки 12 (для $\sum \Delta x$)
53 *n*5*n*13	Запись числа внутренних циклов в 13 яч.
→54 *n*30*н*14	$x_н \rightarrow$«14»
55 *cn*12*н*14	$x_н + i\Delta x = x_i \rightarrow$«14»
56 *n*32*н*15	$z_н \rightarrow$«15»
57 *cn*10*н*15	$z_н + j\Delta z = z_j \rightarrow$«15»
58 *уn*15*н*15	$z_j^2 \rightarrow$«15»
59 *ов*14*н*15	$(z_j^2 - x_i) \rightarrow$«15»
60 *sn*14*н*14	$\sin x_i \rightarrow$«14»
61 *уn*14*н*15	$(z_j^2 - x_i)\sin x_i \rightarrow$«14»
62 *nn*15*н*5	Печать значений *y*
63 *o*2274*н*	Приращение шага $(i+1)\Delta x \rightarrow$«12»
64 *cn*31*н*12	Счетчик внутренних циклов
65 *в*1*n*13	Приращение шага $(j+1)\Delta z \rightarrow$«10»
—66 *и*54*n*≠	Счетчик внешних циклов
67 *cn*33*н*10	Останов машины
68 *в*1*n*11	
—69 *и*52*n*≠	
70 *к*0*n*	

В этой программе команды *n*0*n*10 и *n*0*n*12 служат для очистки рабочих ячеек с целью последующего накопления в них суммы шагов изменения переменных z и x (то есть $\Sigma\Delta z$ и $\Sigma\Delta x$).

Группа команд циклического блока 54÷61 вычисляет текущее значение функции y, причем команды 54, 55 подготавливают значение переменной x_i, а команды 56, 57 — значение переменной z_j.

Команда 64 служит для накопления суммы шагов $\Sigma\Delta x$ переменной x, а команда 67 — накопления шагов $\Sigma\Delta z$ переменной z. В процессе вычислений машина повторяет выполнение внутреннего циклического блока для каждого из значений z_j до тех пор, пока $z_j \leqslant z_к$, после чего останавливается. Для того, чтобы отделить интервалом получаемые группы значений функции y можно после команды 66 или 67 включить команду перевода строки: *o*2268*н* (или *o*2274*н*).

Итерационный цикл. Вычисления методами последовательных приближений всегда сопряжены с необходимостью проверки условия достижения заданной точности расчетов. Поэтому, каждый очередной цикл (каждая итерация) должен заканчиваться сравнением полученной ε^1 и требуемой ε точности вычислений. Повторение циклов, естественно, должно продолжаться до тех пор, пока не станет $\varepsilon^1 \leqslant \varepsilon$. Примерами итераци-

онных процессов могут служить вычисления по рекуррентным формулам, а также вычисление сходящихся рядов.

Пример 9. Составить программу вычисления функции y по рекуррентной формуле

$$y_{i+1} = \frac{1}{3}\left(\frac{x}{y_i^2} + 2y_i\right)$$

с точностью ε для значений $x > 0$.

Если принять начальное приближение функции за y_0, то итерации следует повторять до выполнения условия

$$|y_{i+1} - y_i| \leqslant \varepsilon.$$

Разместим исходные величины в ячейках:

30	31	32	33	34
x	y_0	2	3	ε

Программа

	Команды	Пояснения
50	*п31н12*	Запись $y_0 = y_i$ в 12 ячейку
51	*п12н11*	Сохранение y_i в ячейке 11
52	*уп12н12*	$y_i \to$ «12»
53	*оп30н12*	$x/y_i^2 \to$ «12»
54	*п32н13*	$2 \to$ «13»
55	*уп11н13*	$2y_i \to$ «13» Вычисление y_{i+1}
56	*сп13н12*	$(x/y_i^2 + 2y_i) \to$ «12»
57	*од33н12*	$y_{i+1} \to$ «12»
58	*вп12н11*	$(y_{i+1} - y_i) \to$ «11»
59	*вм34н11*	Проверка условия
60	*и51п>*	$\lvert y_{i+1} - y_i \rvert \leqslant \varepsilon$
61	*пп12н5*	Печать значения y_{i+1}
62	*о2274н*	
63	*к0п*	Останов машины

В этой программе командой 50 засылается начальное значение функции y_0 в ячейку 12. Команда 51 обеспечивает сохранение y_i в ячейке 11. Команды 52÷57 производят вычисление очередного приближения y_{i+1} с записью в операционную ячейку 12. Командой 58 определяется достигнутая (актуальная) точность вычислений $\varepsilon' = y_{i+1} - y_i$, а команды 59 и 60 сравнивают ее с заданной ε. При этом вычисляется разность модулей $|y_{i+1} - y_i| - |\varepsilon|$ и, если эта разность > 0, управление передается команде 51 на повторение цикла. При достижении заданной точности вычислений команда *и51п>* не исполняется (так как разность модулей становится $\leqslant 0$) и полученное значение y_{i+1} выдается на печать.

Примечание. Чтобы специально не вводить простые константы «2» и «3» в ячейки памяти, можно заменить команду 54:

*п*32*н*13 на *п*1049*п*13 и команду 57: *од*33*н*12 на *од*1561*п*12, где число 1049 представляет «2», а число 1561 — «3» в форме с плавающей занятой и в виде «правого параметра». Команды 54 и 55 можно также заменить одной командой *уп*2*п*13, а команду 57 следующей командой: *од*3*п*12. Здесь «2» и «3» приводятся к плавающему виду, потом выполняется соответствующая операция.

Пример 10. Составить программу вычисления знакопеременного ряда

$$s = x - \frac{x^2}{2!} + \frac{x^3}{3!} - \frac{x^4}{4!} + \ldots \frac{(-1)^{n+1} x^n}{n!} \pm \ldots$$

с заданной степенью точности ε.

Здесь вычисление членов ряда необходимо производить до выполнения условия $\left|\frac{x^n}{n!}\right| \leqslant \varepsilon$, после чего вычислительный процесс будет считаться оконченным.

Поместим исходные величины в следующие ячейки:

30	31	32	33
x	1	−1	ε

В рабочей ячейке 13 будем накапливать сумму вычисленных членов ряда, а для формирования членов ряда с соответствующим знаком используем множитель «−1» из ячейки 32.

Программа

Команды	Пояснения	1 цикл	2 цикл
50 *п*31*н*11	Запись «1» в ячейку 11		
51 *п*32*н*12	Запись «−1» в ячейку 12		
52 *п*0*п*13	Очистка рабочей ячейки 13		
→53 *уп*30*н*12		$-1 \cdot x \to$ «12»	$x \cdot x \to$ «12»
54 *од*11*н*12		$-x/1 \to$ «12»	$x^2/1 \cdot 2 \to$ «12»
55 *уп*32*н*12		$(-1) \cdot (-x) = x \to$ «12»	$(-1)(x^2/2 \cdot 1) = -x^2/2! \to$ «12»
56 *п*12*н*14		$x \to$ «14»	$-x^2/2! \to$ «14»
57 *сп*12*н*13		$(0+x) \to$ «13»	$(x - x^2/2!) \to$ «13»
58 *сп*31*н*11		$(1+1) \to$ «11»	$(1+2) \to$ «11»
59 *вм*33*н*14	Проверка условия		
←60 *и*53*п*>	$\left\lvert \frac{x^n}{n!} \right\rvert \leqslant \varepsilon$		
61 *пп*13*н*5			
62 *о*2274*н*	Печать значения s		
63 *к*0*п*	Останов машины		

В этой программе команда 55 производит изменение знака очередного члена ряда, команда 57 обеспечивает накопление суммы членов ряда, а команда 58 подготавливает следующий множитель знаменателя. Команды 59 и 60 проверяют условие достижения заданной точности вычислений и если очередной член ряда $|x^n/n!| > \varepsilon$, то управление передается команде 53 на

повторение цикла. По окончании вычислений полученный результат s выдается на печать.

§ 4. ПРОГРАММИРОВАНИЕ В ОТНОСИТЕЛЬНЫХ АДРЕСАХ

Кроме рассмотренных выше принципов программирования с обращением к конкретным адресам ячеек памяти машины можно использовать также и методы составления программ с относительной адресностью. Необходимость в этом может возникнуть, например, в тех случаях, когда заранее не определено расположение и величина массива исходных данных задачи, место размещения в памяти самой программы, а также при обращениях к стандартным программам и отдельным программным блокам, находящимся в разных местах памяти машины.

Рассмотрим следующий пример.

Пример 11. Требуется переслать массив чисел (матрицу) из одной группы ячеек ОЗУ в другую с произвольными начальными адресами.

Построим два варианта такой программы.

Первый вариант ориентируем на относительность начального и конечного расположения массива, а второй — на относительность расположения самой программы.

Распределим данные о массиве (в виде целых чисел) в следующих ячейках:

30 — начальный адрес α_0 массива до пересылки (например, ячейка 70);

31 — начальный адрес β_0 массива после пересылки (например, ячейка 200);

32 — количество элементов массива (например, $n=12$).

Программу первого варианта расположим в некоторых ячейках ОЗУ, начиная с 50, а программу второго варианта — в условных, начиная с $a+00$.

I Вариант

50	*n*0*n*11	П. б.
51	*n*32*н*12	
→52	*n*30*н*1	Ц. б.
53	*c*11*н*1	
54	*б*11*n*1	
55	*n*0*н*13+	
56	*n*31*н*1	
57	*c*11*н*1	
58	*n*13*н*0+	
59	*c*1*n*11	
60	*в*1*n*12	С. ц.
—61	*u*52*n*>	
62	*к*0*n*	

II Вариант

a+00	*n*0*n*11	П. б.
a+01	*n*32*н*12	
→*a*+02	*n*30*н*1	Ц. б.
a+03	*c*11*н*1	
a+04	*б*11*n*1	
a+05	*n*0*н*13+	
a+06	*n*31*н*1	
a+07	*c*11*н*1	
a+08	*n*13*н*0+	
a+09	*c*1*n*11	
a+10	*в*1*n*12	С. ц.
—*a*+11	$e_1$16374>	
a+12	*к*0*n*	

Программа 1 варианта будет работать следующим образом.

Подготовительный блок. Команда *п*0*п*11 очищает рабочую ячейку 11, а команда *п*32*н*12 записывает количество элементов массива в ячейку 12, которая будет служить счетчиком циклов.

Циклический блок. По команде *п*30*н*1 в регистровую ячейку 1 записывается содержимое 30 ячейки, то есть начальный адрес массива до его пересылки (в нашем случае $\alpha_0=70$).

По команде *с*11*н*1 производится сложение содержимого 11 и 1 ячеек, а так как в 11 ячейке первоначально находился 0, то в 1 ячейке остается число 70.

Команда *б*11*п*1 производит сдвиг содержимого 1 ячейки на 11 разрядов влево. Таким образом, число 70 переносится в младшие разряды первого адреса A_1 (см. разрядную сетку команд).

Команда *п*0*н*13+ сначала складывается с содержимым 1 ячейки, а затем уже исполняется как команда *п*70*н*13. При этом первый элемент пересылаемого массива помещается в рабочую ячейку 13.

По команде *п*31*н*1 в регистровую ячейку 1 записывается содержимое 31 ячейки, то есть новый начальный адрес пересылаемого массива (нашем случае $\beta_0=200$).

По команде *с*11*н*1 производится сложение содержимого 11 и 1 ячеек, и в ячейку 1 записывается число 200.

Команда *п*13*н*0+ сначала складывается с содержимым 1 ячейки и в результате исполняется как команда *п*13*н*200. При этом первый элемент массива из рабочей ячейки 13 переносится в ячейку 200.

По команде *с*1*п*11 содержимое 11 ячейки увеличивается на единицу и тем самым подготавливается приращение адресов: 70+1, 70+2.. и 200+1, 200+2,... в последующих циклах.

Счетчик циклов в этой программе работает обычным образом. В каждом цикле из содержимого 12 ячейки вычитается единица и когда все циклы будут выполнены, ее содержимое станет равным нулю. При этом команда *и*52*п*> выведет машину из циклического режима.

Если теперь потребуется переслать массив, состоящий не из 12 чисел, а из 100 и начальные адреса соответствующих групп ячеек будут равны: $\alpha_0=300$ и $\beta_0=500$, то следует лишь изменить содержимое ячеек 30, 31 и 32 не внося никаких исправлений в саму программу.

Второй вариант программы в принципе не отличается от первого. Только в блоке счетчика циклов команда условного перехода *и*52*п*> заменяется командой **относительного условного перехода** $e_1$16374*п*>. В этом случае (если результат предыдущей операции >0) адрес перехода формируется посредством операции: $K_п=(a+11)+16374-16383=(a+2)$, то есть после команды $(a+11)$ машина перейдет к выполнению команды $(a+2)$.

Пример 12. Вычислить среднее арифметическое значение некоторого массива неотрицательных чисел, расположенных в произвольной группе последовательных ячеек памяти (α_0, α_1, α_2,...).

При составлении программы заметим, что величина массива может быть различной, а для работы программы лишь фиксируется начальный адрес массива и в конце его записывается любое отрицательное число (например, —1).

Отведем ячейку 40 для записи начального адреса α_0 массива чисел.

Программа

Команды	Пояснения
$a+00$ $n0n10$	Очистка рабочих ячеек
$a+01$ $n0n12$	
$a+02$ $n40н1$	Запись начального адреса в регистр. ячейку 1
$a+03$ $б11n1$	Сдвиг начального адреса в A_1
$a+04$ $n0н13+$	Запись числа из $A_{1ис}$ в ячейку 13
$a+05$ $e_14n<$	Проверка чисел на знак
$a+06$ $cn13н12$	Накопление суммы чисел
$a+07$ $c2048n1$	Прибавление «1» к A_1 (содержимому) ячей-
$a+08$ $cn1n10$	ки 1 (10)
$a+09$ $e_116378n$	Передача управления к $a+04$
$a+10$ $од10н12$	Вычисление среднего арифметического
$a+11$ $пп12н5$	Печать результата
$a+12$ $о2274н$	
$a+13$ $к0n$	Останов машины

В этой программе по команде $n40н1$ в регистровую ячейку 1 заносится начальный адрес α_0 массива чисел. Командой $б11n1$ этот адрес сдвигается влево в младшие разряды A_1 ячейки 1.

Команда $n0н13+$ сначала складывается с содержимым ячейки 1 и первый раз исполняется, как команда $n\alpha_0н13$. При этом очередной элемент массива из исполнительного адреса $A_{1ис} = (\alpha_0+0)$ переписывается в рабочую ячейку 13. В следующем цикле $A_{1ис} = \alpha_0+1 = \alpha_1$ и т. д.

Команда $e_14n<$ проверяет очередной элемент на знак. Если этот элемент будет иметь отрицательный знак, то управление передается команде на окончание вычислений: $a+10$, в ином случае исполняется следующая команда.

По команде $cn13н12$ в рабочей ячейке 12 будет накапливаться сумма чисел (элементов) массива.

По команде $c2048n1$ к содержимому ячейки 1 в каждом цикле прибавляется единица в 12-м разряде (2048), то есть формируется исполнительный адрес $A_{1ис} = \alpha_{i+1}$.

По команде $cn1n10$ к содержимому ячейки 10 в каждом цикле прибавляется единица в форме с плавающей запятой, благо-

даря чему происходит счет числа слагаемых элементов. Форма с плавающей запятой выбрана здесь из тех соображений, что сумма чисел должна быть разделена на количество элементов, которое еще не вычислено.

Команда $e_1 16378n$ передает управление по адресу $a+04$ на повторение циклов.

По команде *од*10*н*12 полученная в ячейке 12 полная сумма делится на число элементов данного массива, то есть вычисляется среднее арифметическое значение.

Последние три команды осуществляют печать результата и останов машины.

§ 5. НЕКОТОРЫЕ ПРИЕМЫ РАБОТЫ С 2-й СТРАНИЦЕЙ ОЗУ

Вторая страница ОЗУ занимает ячейки 2048÷4089, к которым можно обращаться только с помощью команд: «$з_1 A_1 \Theta$» и «$зA_1 \Theta A_2$». В зависимости от наличия единицы в 26, 27, 28 разрядах фиксированной ячейки a_7 происходит модификация выполняемой команды. Возможны три модификации, которые могут иметь место все вместе или по отдельности. Это — модификация номера выполняемой команды; модификация первого адреса выполняемой команды; модификация второго адреса и в условных командах еще номера ячейки m ($m \leqslant 15$). О ячейке m смотри в начале главы 7.

С помощью операций «*з*» и «$з_1$» можно выполнить три вида действий:

1) записать массивы (команд, чисел, подпрограмм) на 2-ую страницу;

2) считывать элементы вышеуказанных массивов и производить с ними все машинные операции за исключением псевдоопераций;

3) выполнять команды, записанные на 2-ой странице;

Рассмотрим с помощью примеров как это делается.

1. *Запись на 2-ую страницу.* Пусть массив расположен в ячейках $\alpha \div \alpha_1$ первой страницы и информация об этом массиве записана начиная с 33-й ячейки:

33	34	35
α	β_0	N

где N — число ячеек от α до α_1, β — начало массива ячеек во второй странице, $\beta_0 = \beta - 2048$.

Тогда, например, можно пользоваться нижеследующей программой, которая использует в качестве рабочих ячейки 33÷37, а сама может записываться с ячейки γ.

Программа

Команды	Пояснения
γ+ 0) *пαп*33	Засылка исходных констант в ячейки хранения 33, 34, 35 как целых чисел
γ+ 1) $п\beta_0 п$34	
γ+ 2) *пNп*35	
γ+ 3) *п*33*н*36	пересылка констант α и N в рабочие ячейки 36 и 37
γ+ 4) *п*35*н*37	
γ+ 5) *б*11*п*36	формирование команды «$x\alpha н\beta_0$» и засылка ее в ячейку 1
γ+ 6) *с*34*н*36	
γ+ 7) *п*36*н*1	
γ+ 8) $з_1$512*л*	запись 1 в 28 разряде ячейки a_7
γ+ 9) *п*0*н*+	операция пересылки
γ+10) $з_1$0*л*	запись 0 в ячейку a_7
γ+11) *с*2049*к*1	добавление по «1» к 1 и 2-му адресам команды «$x\alpha н\beta_0$» в ячейке 1.
γ+12) *в*1*п*37	счетчик цикла с командой передачи управления
γ+13) $е_2$16378*ч*=1	
γ+14) *к*0*п*	останов

По команде (γ+9) происходит модификация команды по содержимому индекс-ячейки I, так как имеется «+» — признак модификации. Команда при первом счете в цикле будет иметь вид «$п\alpha н\beta_0$». Дальше команда выполняется в зависимости от содержимого ячейки a_7. Так как 28 разряд ячейки a_7 ненулевой, то содержимое ячейки α пересылается в ячейку $2048+\beta_0$. В дальнейшем происходит циклическая пересылка содержимых ячеек $\alpha+i$ в ячейки $2048+\beta_0+i$, где $i=1, 2,..., N$.

Если команда (γ+9) была бы условная, например, «*п*0*н*0>13+», то перед выполнением команды «$п\alpha н\beta_0$>13» проверяется неравенство «>» между содержимым ячейки 2048+13 и нулем. Если неравенство выполняется, то содержимое ячейки α пересылается в ячейку $2048+\beta_0$.

2. *Считывание с 2 страницы.* Для считывания можно пользоваться программой, написанной выше. При этом надо произвести следующие изменения:

а) под α надо понимать адрес начало массива на 2 странице, уменьшенный на 2048;

б) под β_0 понимается адрес, начиная с которого помещается массив на 1 странице.

в) команда γ+8 заменяется на команду «$з_1$256*л*», то есть происходит запись 1 в 27 разряд, и нули в остальные разряды ячейки a_7.

Аналогично можно производить машинные операции с числами, находящимися на 2-ой странице.

Пример. Пусть надо сложить два числа с фиксированной запятой, находящиеся в ячейках 2048+100 и 2048+101 и резуль-

тат послать в ячейку 102. Программа написана начиная с ячейки α 1 страницы ОЗУ.

$\alpha+0)$ $з_1768л$	запись 1 в 27 и 28 разряды ячейки a_7,
$\alpha+1)$ $с100н101$	сложение содержимого 2148 и 2149 ячеек,
$\alpha+2)$ $з_1256л$	запись 1 в 27 разряд ячейки a_7,
$\alpha+3)$ $п101н102$	пересылка содержимого 2149 ячейки в 102,
$\alpha+4)$ $з_10л$	очистка ячейки a_7.
$\alpha+5)$ $к0п$	останов.

3. *Выполнение команд, записанных на 2 странице.* Допустим в ячейках $(2048+\beta_0)\div(2048+\beta_1)$ записана подпрограмма (команды или команда) и в ячейке $(2048+\beta_1)$ записана команда «$з_10л$». Если где-то в программе мы должны обращаться к этой подпрограмме, то это можно сделать нижеследующим образом. Программу пишем с ячейки α, используя ячейки с 33 в качестве рабочих.

Рабочие константы записаны в ячейках 33÷35:

33	34	35
$з_1128л$	β_0	β_1

Здесь в 33 ячейке записана команда «$з_1128л$», в 34 и 35 ячейках как целые числа записаны β_0 и β_1.

Программа

Команды	Пояснения
$\gamma+$ 0) $п34н36$	пересылка β_0 и β_1 в рабочие ячейки 36 и 37
$\gamma+$ 1) $п35н37$	
$\gamma+$ 2) $в1к36$	получение β_0-1 и β_1+1 в 36 и 37 ячейках, 38 и 39 ячейках соответственно
$\gamma+$ 3) $с1к37$	
$\gamma+$ 4) $п36н38$	
$\gamma+$ 5) $п37н39$	
$\gamma+$ 6) $б11к38$	сдвиг β_0-1 на место $[A_1]_н$
$\gamma+$ 7) $п38н1$	в 1 ячейку записывается «$x(\beta_0-1)н0$»
$\gamma+$ 8) $п0н40+$	пересылка содержимого (β_0-1) ячейки в 40 яч.
$\gamma+$ 9) $б11к39$	сдвиг β_1+1 на место $[A_1]_н$
$\gamma+10)$ $п39н1$	в 1 ячейку записывается «$x(\beta_1+1)н0$»
1) $п0н41+$	пересылка содержимого (β_1+1) ячейки в 41 яч.
2) $п36н1$	получение кода «$з_1128л$» в ячейке β_0-1
3) $п33н+1$	
4) $п38н1$	формирование кода $x(\beta_0-1)н(\beta_1+1)$
5) $с37н1$	в 1 ячейке
6) $и0к+$	команда перехода
7) $п36н1$	
8) $п40н+1$	восстановление содержимого ячеек (β_0-1) и (β_1+1)
9) $п37н1$	
$\gamma+20)$ $п41н+1$	
1)	

Программа может помещаться в любом свободном месте, имеет рабочие ячейки 33÷41. Если нельзя использовать эти свободные ячейки, то в программу нужно ввести исправления. Команды $\gamma \div (\gamma+5)$ позволят получить значения β_0 и β_1 в ячейках 36 и 37, значения β_0-1 и β_1+1 в ячейках 36 и 37, 38 и 39. По команде $\gamma+6$ сдвигается влево на 11 разрядов код β_0-1 и по следующей команде результат записывается в 1 ячейку. Команда $\gamma+8$ сохраняет содержимое ячейки β_0-1 в ячейке 40, так как потом в эту ячейку записывается новая команда. Командами $(\gamma+9) \div (\gamma+11)$ содержимое ячейки β_1+1 пересылается в 41 ячейку. По команде $\gamma+12$ в 1 ячейке получаем число β_0-1. Команда $\gamma+13$ перед выполнением модифицируется и будет выполняться команда «$n33н(\beta_0-1)$». При ее исполнении в ячейку β_0-1 записывается код «$з_1128л$». По команде $\gamma+14$ в первую ячейку посылается код «$x(\beta_0-1)н0$» и по следующей команде этот код складывается с содержимым ячейки 37, в результате в ячейке 1 получается код
«$x(\beta_0-1)\ н(\beta_1+1)$».

По команде $\gamma+16$ происходит переход к выполнению подпрограммы на 2-ой странице. Перед выполнением команда имеет вид «$и\ (\beta_0-1)\ к(\beta_1+1)$». При этом в (СчК) посылается (β_0-1), но прежнее значение (СчК) запоминается, то есть в ячейку (β_1+1) записывается команда «$и$(СчК)$п$». При выполнении программы ЭВМ выполняет (β_0-1) команду, то есть записывается 1 в 26 разряд ячейки a_7. Все последующие команды выполняются с 2 страницы до тех пор, пока в 26 р. ячейки a_7 не станет 0 или не встретится команда останова. В ячейке β_1+2048 записаны команда «$з_10л$». Таким образом, следующая команда берется из ячейки β_0+1 с 1 страницы. Но там записана команда возврата в подпрограмму. Следующая команда будет выполняться с номером $\gamma+17$. В ячейках $(\gamma+17) \div (\gamma+20)$ записаны команды восстановления содержимого ячеек β_0-1 и β_1+1.

Для выполнения одной команды с 2 страницы можно все упростить.

Пример. Выполнить команду, записанную в ячейку 2248=2048+200. Предполагается, что в следующей ячейке записана команда «$з_10л$». В рабочую ячейку 33 запишем команду «$з_1128л$». Тогда программу можно записать, например с 100 ячейки:

100	$п33н199$	запись кода «$з_1128л$» в ячейку 199
101	$и_1199к202$	переход к исполнению 199 команды
102	$к0п$	конец.

Здесь предполагается, что в ячейке 199 ничего не записано. При этом исполняются только две команды с 2 страницы, первая из 2248 ячейки и вторая из 2249 ячейки.

§ 6. ОТЛАДКА ПРОГРАММЫ НА МАШИНЕ

Перед решением задачи на машине по вновь составленной программе необходимо предусмотреть проверку правильности работы программы и обеспечить контроль качества решения. Целью отладки программы является выявление ошибок в алгоритме задачи и в самой программе. Перед отладкой программы необходимо вручную, на хороших числах, произвести поблочный расчет (начиная с внутренних блоков) и фиксировать получаемые промежуточные результаты.

Отладка состоит в последовательном выполнении вычислений по отдельным участкам или блокам и выдаче, после каждого этапа вычислений, промежуточных результатов и содержимого некоторых рабочих ячеек памяти. Сравнивая результаты машинного и ручного счета, можно судить о правильности программы. Обычно, после выполнения каждого блока программы производится контрольный останов машины. Это можно осуществить с цомощью команды *к0п* или с пульта управления в режиме «Останов по адресу».

Рассмотрим некоторые особенности выполнения этих операций.

Остановка по адресу. При использовании этого режима машина останавливается по заданному адресу и на сумматоре можно прочесть результат исполнения предыдущей команды.

Останов по адресу выполняется следующим образом. Предположим, требуется остановить машину на адресе 204 и узнать результат выполнения 203 команды. Для этого, прежде всего, число 204 нужно перевести в двоичную систему. Такой перевод можно сделать с помощью машины следующими двумя командами:

$$\begin{array}{ll} \alpha+00) & п204к1020 \\ \alpha+01) & к0н. \end{array}$$

где ячейка 1020 используется в качестве рабочей. Затем, в режиме «Выдача памяти», вывести на печать содержимое 1020 ячейки в двоичном коде.

Далее производятся следующие операции.

1. На клавиатуре ОПА (останов по адресу) пульта управления набирается в двоичном коде число 204.

2. Нажимается клавиша «Останов по адресу» на панели сигнализации.

3. Производится запуск программы с нужного адреса. Для этого печатают: N*и*, где N — начальный адрес проверяемого блока, а символ «*и*» — команда исполнения программы с заданного адреса.

4. После остановки машины на 204 адресе проверяется результат выполнения 203 команды.

5. Если результат получился верным, то нажатием клавиши «Пуск-2» можно продолжать вычисление другого блока с заранее предусмотренным остановом по следующему адресу. Если же интерес представляет не только результат одной команды отлаживаемого блока, но и некоторые другие, то после нажатия клавиши «Выдача памяти» можно проверить содержимое соответствующих рабочих ячеек. После этого, для продолжения работы программы надо нажать клавишу «Пуск-1» и напечатать адрес следующего блока: *Ми.*

6. Если результат не верен, то в отлаживаемом блоке допущена ошибка и нет смысла дальше продолжать вычисления, пока не сделаны необходимые исправления.

Останов по команде к0п. При использовании для отладки программы команды *к0п* (или *к0н*) производятся следующие операции.

1. Определяется команда, которой завершается вычисление проверяемого блока.

2. Следующую за ней команду заменяют на «*к0п*».

3. Выполняется программа данного блока. По окончании вычислений на панели сигнализации должна загореться лампа «Конец».

4. В режиме «Выдача памяти» производят сравнение результатов вычислений с данным ручного счета.

5. Если блок работает правильно, то команду «*к0п*» исправляют на ту, которая была ранее, и с нее продолжают вычисления. В противном случае надо разбить блок на части и повторить проверку для более точной локализации ошибки.

6. После отладки одного блока переходят к проверке других аналогичным способом.

В некоторых случаях машина может не пройти до конца отлаживаемый блок. Причиной тому может быть следующее.

1. Машина не выходит из цикла, совершая бесконечные вычисления, то есть зацикливается.

2. Машина останавливается при переполнении разрядной сетки.

3. Машина останавливается по имеющемуся в программе останову вне отлаживаемого блока.

4. Машина останавливается по технической неисправности.

Зацикливание может происходить при неверном изменении счетчика цикла. Это легко проверить остановом по адресу команды, содержащей проверку условия окончания цикла или его формирования.

Переполнение разрядной сетки возникает, если величина получаемых результатов превышает возможности машины.

Остановка вне блока происходит при неверном выполнении передачи управления. В этом случае машина будет выполнять команды вне проверяемого блока или вообще вне программы.

Контрольное суммирование. При неоднократном использова-

нии достаточно сложных программ, для проверки правильности их ввода в машину, рекомендуется использовать метод контрольного суммирования команд. Как правило, контрольное суммирование команд производится для больших программ, которые вводятся значительное время и при этом возможен какой-либо сбой в машине или в системе устройств ввода. Процедура контрольного суммирования выполняется с помощью специальной вспомогательной подпрограммы, пример которой мы здесь приводим. Для использования этой подпрограммы нужно, прежде всего, разделить отлаженную рабочую программу на участки, состоящие из равного количества команд, для того, чтобы в дальнейшем суммирование кодов команд производилось по этим группам с выдачей сумм команд каждого участка в виде целых чисел.

Рассмотрим следующий пример. Пусть требуется произвести контрольное суммирование 120 команд по 30 команд в каждой сумме, начиная со 140 ячейки.

Исходные данные размещаем (в виде целых чисел) в следующем порядке:

30 — первый адрес суммируемой программы (в нашем случае число 140);

31 — количество команд, входящих в одну контрольную сумму (в данном случае — 30);

32 — количество контрольных сумм (в данном случае $n=120/30=4$).

Подпрограмму контрольного суммирования можно разместить в любом незанятом участке ОЗУ.

Программа КΣ

Команды		Пояснения
$a+00$	$n32н15$	Подготовительный блок
$a+01$	$n30н1$	
$a+02$	$б11п1$	
$\rightarrow a+03$	$п0п14$	Очистка рабочей ячейки 14
$a+04$	$п31н12$	Запись числа внутр. циклов в 12 ячейке
$\rightarrow a+05$	$с0н14+$	Образование контр. Σ
$a+06$	$с2048п1$	Увеличение адреса A_1 регистр. ячейки на 1
$-a+07$	$в1п12$	Счетчик внутренних циклов
$a+08$	$e_i16380п\neq$	
$a+09$	$п1н13$	Подготовка печати начального адреса контр. Σ группы команд
$a+10$	$а11п13$	
$a+11$	$в31н13$	
$a+12$	$пт13н$	Печать начальн. адреса контр. Σ
$a+13$	$о2270н$	Образование двух пробелов
$a+14$	$о2270н$	
$a+15$	$пт14н$	Печать контр. Σ группы команд
$a+16$	$о2274н$	Возврат каретки и интервал
$a+17$	$о2268н$	

$a+18$	$в1n15$	
$a+19$	$e_1 16367n\neq$	Счетчик внешних циклов
$a+20$	$к0n$	Останов машины

При решении задачи на машине отлаженная программа вводится в машину, за ней вводится подпрограмма контрольного суммирования. После работы подпрограммы на печать будут выданы контрольные суммы, которые следует сравнить с числами контрольного суммирования, приведенными в инструкции к пользованию данной программой. Если числа совпадают, то программа введена без ошибок.

§ 7. ИСПОЛЬЗОВАНИЕ СТАНДАРТНЫХ ПРОГРАММ

В состав математического обеспечения машины «Наири-К» входит несколько стандартных программ, предназначенных для решения некоторых типовых задач. К ним относятся:

СП вычисления определенного интеграла методом Симпсона;

СП интегрирования систем n обыкновенных дифференциальных уравнений 1-го порядка методом Рунге—Кутта;

СП умножения двух матриц;

СП обращения матриц методом пополнения А. П. Ершова (приближенный метод);

СП обращения матриц методом Жордана;

СП вычисления определителя матрицы методом «главных элементов»;

СП решения системы n линейных алгебраических уравнений с m неизвестными методом наименьших квадратов ($m<n$);

СП решения системы линейных алгебраических уравнений n-го порядка методом «главных элементов»;

СП «симплекс—метод» для решения задач линейного программирования;

СП интерполяции полиномами по итеративной формуле Эйткина;

СП вычисления полиномов Лежандра;

СП для отладки программ, написанных в машинных адресах.

Все эти программы постоянно находятся в ДЗУ. Для решения задачи по стандартной программе требуется лишь ввести исходные числовые данные и некоторые вспомогательные параметры (например, порядок обращаемой матрицы и т. д.). Все стандартные программы составлены для работы с числами, представленными в форме с плавающей запятой.

Каждая из перечисленных программ может быть использована и автономно (как отдельная программа) и как подпрограмма. Если поставленную задачу можно полностью решить

по одной из стандартных программ, — ее используют как автономную. В более общем случае, когда стандартная программа включается, как составная часть, в основную программу, — она используется в качестве подпрограммы.

Кроме стандартных программ имеется возможность использования в качестве подпрограмм некоторых блоков транслятора АП. Например, подпрограммы: «Построение графиков», «Раскрытие формул» и др.

Автоматическое программирование

Глава 9

ОПИСАНИЕ ЯЗЫКА «АП»

Решение задач на электронных цифровых вычислительных машинах обычно связано с достаточно трудоемким процессом программирования в адресных кодах конкретной машины, знания системы команд, логики и других особенностей ЭЦВМ. В связи с этим в настоящее время все более широкое применение получают различные алгоритмические языки, позволяющие в значительной степени упростить процесс программирования и приблизить форму записи программы к обычному математическому языку.

В ЭЦВМ «Наири» имеется система автоматизации программирования. Это означает, что программа, написанная на языке «АП», автоматически транслируется, переводится в программу, написанную в машинных командах. Язык «АП» очень близок к обычной математической записи выражений и условий. Таким образом, весь трудоемкий процесс составления рабочей программы передается самой машине, а инженеры, конструкторы и исследователи могут выполнять самые разнообразные вычисления не изучая адресного программирования. Алгоритм задачи записывается на этом языке в виде последовательности простых предложений, называемых операторами.

В языке «АП» могут быть выделены три основные структурные единицы:

— набор основных символов;
— слова;
— предложения.

Под основными символами языка понимаются неделимые знаки, из них строят слова.

Слова — это наименьшие смысловые единицы языка, они получаются из символов установлением их в каком-либо порядке. Из слов строятся предложения.

§ 1. АЛФАВИТ

Основными символами языка являются:

1) буквы русские

а, б, в, г, д, е, и, к, л, м, н, о, р, с, у, х, ч, ш, ы, ж, з, й, ь, ф, ц, щ, э, ю, я;

2) латинские строчные буквы *f, l, m, s, t, z, n, i, j, k;*
3) константа $\pi=3{,}141592653$;
4) подстрочные буквы $_{i,\ j,\ k}$;
5) цифры 0, 1, 2, 3, 4, 5, 6, 7, 8, 9;
6) надстрочные цифры $^{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9}$;
7) подстрочные цифры $_{0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9}$;
8) арифметические операции $+, -, \times, /$;
9) операции отношения $=, \neq, >, <, \geqslant, \leqslant$;
10) разделители: «,» — запятая, «(», «)» — скобки;
11) знак абсолютного значения «|»;
12) знак звездочек $*$.
13) знак бесконечности ∞;
14) знак пробел $\_$
15) знак возврата каретки «*в. к.*» или «↑».

Буквы, указанные в 1) и 2), используются для обозначения переменных величин, функций и названий операторов.

Подстрочные буквы и цифры $_{i,\ j,\ k,\ 0,\ 1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9}$ используются только для обозначения индексов и их значений. Например, a_i; c_{ij}; $в_{500}$; $б_{100,2}$ являются, переменными с индексами.

Надстрочные цифры используются для обозначения показателя степени. Например, a^{13}; $25{,}7^2$; а степень $x^{3/4}$ записывают в виде: $\exp((1/4)\ln x^3)$ или $\exp(0{,}75\ln x)$ или $\exp 0{,}75\ln x$.

Знак $=$ служит для присвоения левой части значения правой части, а также используется как символ равенства. Символ «|» употребляется для обозначения абсолютной величины. Например, $y=|x|$.

Символ * используется при построении графиков для пометки точек графика. Этот же символ используется для отмечания синтаксической ошибки при вводе программы в машину. Символ ∞ используется для пометки синтаксических ошибок и ошибок в случае переполнения содержимого ячеек.

Символ «*в. к.*» или «↑» используется в качестве признака конца оператора.

Запись арифметических выражений обычно математическая, с некоторыми дополнительными особенностями. В языке используются постоянные и переменные величины. Ниже излагаются основные особенности записи выражений на языке АП.

§ 2. ЧИСЛОВЫЕ КОНСТАНТЫ

Числа могут использоваться целые и действительные, записанные в десятичной форме. Например, 1500; —5; —7,12. Константы целого типа представлены в машине точно (в форме с фиксированной запятой), а константы действительного типа — приближенно (в форме с плавающей запятой). Целые числа могут находиться в интервале $|x|\leqslant 0{,}34359738367\cdot 10^{11}$ и записываться 11 цифрами. Числа в десятичной записи могут

лежать в интервале $0{,}922337203 \cdot 10^{-19} < |x| < 0{,}922337203 \cdot 10^{19}$ и $x=0$ и записываться 9 цифрами так: *целая часть* (если есть), *запятая, дробная часть.*

Целые (без знака) числа от 1 до 232 могут также использоваться в качестве меток операторов. Однако общее количество числовых констант, встречающихся в программе, и меток не должно превышать 232.

Знак «+» перед положительными числами обязательно опускается.

§ 3. ПРОСТЫЕ ПЕРЕМЕННЫЕ

Простая переменная — это любая из букв алфавита, за исключением букв *ж, з, й, ь, ф, ц, щ, э, ю, я, ѓ,* подстрочных букв $_{i,\ j,\ к}$ и строчных букв *i, j, к, п.* Строчные буквы *i, j, к, п* называются переменными **целого** типа, подстрочные буквы $_{i,\ j,\ к}$ — **индексами.** Тип обозначаемой величины определяется самой переменной. Переменные *i, j, к, п* принимают только неотрицательные целочисленные значения. Например, 125; 999. В зависимости от значений строчных букв *i, j, к* определяются значения индексов $_{i,\ i,\ к}$. Все остальные переменные являются переменными действительного типа, которые принимают значения, представленные в машине в форме нормализованного числа.

§ 4. ПЕРЕМЕННЫЕ С ИНДЕКСАМИ

Если к простой переменной припишем один или два индекса, то получим **переменную с индексами.** Например: $в_i$; $с_{i8}$; $l_к$; $б_{10.2}$; $б_{105}$ и т. д. Переменные с индексами являются переменными действительного типа, т. е. всегда будут рациональными числами.

Простая переменная $а$ называется **идентификатором** переменной с индексами $а_{ij}$.

Если в программу, написанную на языке автоматического программирования входят переменные с индексами, то перед тем оператором, где впервые встречаются переменные с индексами, должны быть указаны максимальные значения индексов, а также идентификаторы, которые должны снабжаться индексом (индексами) (описательная часть массива). Минимальные значения индексов считаются равными нулю.

Например, если в описательной части массива или иначе в «шапке» стоит запись: $i=3 \_\_ а$

$$j=2 \_\_ к=4 \_\_ в \_\_ с,$$

то это означает, что мы имеем дело с величинами $а_0$, $а_1$, $а_2$, $а_3$ и с величинами

$в_{00}$,	$в_{01}$,	$в_{02}$,	$в_{03}$,	$в_{04}$,		$с_{00}$,	$с_{01}$,	$с_{02}$,	$с_{03}$,	$с_{04}$,
$в_{10}$,	$в_{11}$,	$в_{12}$,	$в_{13}$,	$в_{14}$,		$с_{10}$,	$с_{11}$,	$с_{12}$,	$с_{13}$,	$с_{14}$,
$в_{20}$,	$в_{21}$,	$в_{22}$,	$в_{23}$,	$в_{24}$;		$с_{20}$,	$с_{21}$,	$с_{22}$,	$с_{23}$,	$с_{24}$.

§ 5. ФУНКЦИИ

В АП-программе могут быть использованы следующие функции: **sin, cos, tg, arcsin, arccos, arctg, ln, lg, exp,]** (квадратный корень), **нс** (гиперболический косинус), **нs** (гиперболический синус), **нt** (гиперболический тангенс) и **га** (гамма-функция Гаусса). Как значение аргумента, так и значение функции должно быть действительного типа.

§ 6. АРИФМЕТИЧЕСКОЕ ВЫРАЖЕНИЕ

Различают два типа арифметических выражений: целый и действительный. Их определения:

а) любая постоянная или переменная действительного (целого) типа есть арифметическое выражение действительного (целого) типа;

б) если А — переменная целого типа и n — натуральное число (переменная целого типа), то $A \pm n$ арифметическое выражение целого типа первого рода (второго рода);

в) если А и В — арифметические выражения действительного типа, γ — натуральное число, то A+(B), A(B), A/(B), A^{γ}, A—(B), —(A) арифметические выражения действительного типа, если первый символ В не есть знак «—», то скобки можно опустить;

г) если первый символ А не есть знак «—», то —А также арифметическое выражение действительного типа;

д) если А — арифметическое выражение действительного типа, то нsA, нсA, нtA, гаA, sinA, cosA, tgA, arcsinA, arccosA, arctgA, expA, lnA, lgA, $\sqrt{A}$ также являются арифметическими выражениями действительного типа.

Например,

$$\sqrt{(\sin \cos x - 1)} - \text{нs}\ \lg\ \ln \sqrt{(\text{га}(x - b_{ij}))}.$$

§ 7. ПОРЯДОК ВЫПОЛНЕНИЯ ОПЕРАЦИЙ

Если в выражении старшинство действий не полностью определено скобками, то действия выполняются в следующем порядке:

1) возведение в степень;
2) умножение и деление;
3) сложение и вычитание.

Если скобки опущены, то в ряде последовательных действий одинакового порядка выражение раскрывается так, что скобки группируются слева. Например, *абсд* понимается как *(((аб)с)д)*; выражение *a—x—c+e* понимается как *(((a—x)—c)+e)*. Исключением является тот случай, когда среди умножений и делений после знака деления «/» имеется хотя бы одна операция

деления или умножения. В этом случае выражение понимается так, как будто скобки группируются справа. Например, выражение xa_{ij}/e понимается как $((xa_{ij})/e)$, а выражение *тд/абн* как $(m(д/a(бн))))$.

Скобки употребляются только круглые и не более 31 скобок, подряд открывающихся или закрывающихся.

Выражения, заключенные в круглые скобки, вычисляются в первую очередь. Если выражение, содержащее скобки, само заключено в круглые скобки, то вычисления производятся, начиная с внутренних скобок.

Например, в выражении $a(x-b/(x+c)$ сначала находятся сумма $x+c$, затем частное $y=\frac{b}{x+c}$ затем разность $z=x-y$ и, наконец, произведение $a\cdot z$.

При составлении арифметических выражений необходимо следить за тем, чтобы число закрытых скобок было равно числу открытых.

§ 8. НЕКОТОРЫЕ ОСОБЕННОСТИ ЗАПИСИ АРИФМЕТИЧЕСКИХ ВЫРАЖЕНИЙ

В АП-программе должны быть символы только из алфавита и должны употребляться только по назначению. Нельзя использовать в АП-программе символы: *ж, з, й, ь, ф, ц, щ, э, ю, я, ѣ*. При записи арифметических выражений знак умножения опускается; таким образом, выражение xy понимается как $x\times y$, а $5{,}1a$ как $5{,}1\times a$. Произведение двух чисел, например, 5 *умножить на* 2,71 записывается как 5(2,71) или (5)(2,71).

Для деления употребляется только знак /. Пример использования знака: $\frac{a+b}{c+x-y}$ записывается как $(a+b)/(c+x-y)$. Скобки появляются и при использовании функции $\sqrt{}$. Например, вместо $\sqrt{a^2-xy}$ пишется $\sqrt{(a^2-xy)}$.

Для расшифровки информации АП-программы **каждой функции должен предшествовать символ «пробел»** — ⎵, при отсутствии символа «пробел» ЭВМ будет рассматривать поступившие символы как сомножители произведения. Так выражение $xy\cdot 1nд$ записывается как xy⎵$1nд$.

Нельзя писать рядом два знака операции, например, $x=y/-a$, можно $x=y/(-a)$. Аргументом функции является все то, что следует за символом функции до знака «)», «+», «—» или «↑» (возврат каретки), если только при этом количества левых и правых скобок, заключенных между символом функции и одним из знаков «)», «+», «—» или «↑» равны. Например, выражение $\sqrt{}$(⎵sin⎵cos x—1)+⎵нs⎵lg⎵lnt⎵$\sqrt{}$(⎵ra$(x-b_{if})$)↑ понимается как $((\sqrt{}((\text{⎵sin}(\text{⎵cos}\,x))-1))+\text{⎵нs}(\text{⎵lg}(\text{⎵ln}(t\text{⎵}\sqrt{}(\text{⎵ra}(x-b_{if}))))))$; выражение ⎵sin⎵cos $3a^2(x+y)\oplus$ пони-

мается как _sin(_ cos(3a²(x+y))) ⊕, где знак ⊕ есть один из символов «)», «+», «—» или «↑».

Если y переменная и А-арифметическое выражение, то «y=А» — есть **формула**. Запись «y=А» означает: присвоить значение выражения А переменной y.

Ниже приводятся примеры правильной и неправильной записи на языке АП соответствующих арифметических выражений.

ПРИМЕРЫ

Выражение	Верно	Неверно
$\frac{a^5}{y}\cos x^2$	(a⁵/y) _cos x² (a⁵ _cos x²)/y	a⁵/y cos x² a⁵ cos x²/y
$\frac{x}{\sin^2 x+1}$	x/((_sin x)² + 1)	x/(sin x)² + 1 x/(sin² x + 1)
$\frac{m^{3/4}}{y^2+1}$	(_exp 0,75 _ln m)/(y²+1)	m^0,75/(y²+1)
$\ln\left(\frac{1}{\sqrt{s-32,5}}\cdot t^2\right)$	_ln t²/_√(s − 32,5)	ln 1/√(s — 32,5)) t²
$e^{0,3x}\cdot\operatorname{arctg} z$	(_exp 0,3x) _arctg z	_exp 0,3x _arctg z

Примечание. В записи арифметических операторов для удобства ввода желательно использовать обозначения переменных одного регистра печатающего устройства (верхнего — русский шрифт или нижнего — латинский шрифт).

Напишите на языке АП следующие арифметические выражения:

1. $\operatorname{tg}^2\frac{a+b}{c^2-1}-\frac{3}{2}c\ln\frac{a}{5}$;
2. $\frac{(2\pi)^{1/2}e^{-x}}{x}$;
3. $\sin a\cdot\cos\left(\frac{\ln b+x^2}{ac}\right)$;
4. $\sqrt{a^2+\frac{1}{\sqrt{y^2+1}}}\cdot c$.

Глава 10

ОПЕРАТОРЫ

Служебный оператор — это некоторое указание, написанное в виде слова или группы слов, данное машине для обработки информации, поступающей за этим указанием. Фактически каждый оператор настраивает машину на прием, анализ и составление программы для определенной совокупности символов

данного оператора. Операторы записываются в той последовательности, какую потребует решаемая задача.

Наименования операторов подобраны так, что первые две буквы названия оператора определяют данный оператор, а остальные буквы добавляются самой машиной.

В языке АП используются следующие 18 операторов:

допустим...
вставим...
вычислим...
если...
идти к...
введем...
печатаем...
храним...
начертим...
программа...
спросим...
массив...
интервал...
останов...
кончаем...
исполним...
сотрем...
лента....

Каждому оператору программы, за исключением операторов «*массив*», «*исполним*», «*сотрем*» и «*лента*» присваивается соответствующий порядковый номер (метка).

Рассмотрим особенности употребления этих операторов.

§ 1. ОПЕРАТОРЫ ПРИСВАИВАНИЯ

1. Оператор **«допустим...»** служит для присваивания переменным конкретных числовых значений, а также значений других переменных и их модулей. Например,
«*допустим д=р __ a=5 __ i=7*».

Допускаются соотношения следующих видов.

Для переменных типа «целые» *(i, j, k, n)*:

$$\text{буква} = \begin{cases} \text{число целое} \\ \text{буква.} \end{cases}$$

Для переменных типа «действительные».

$$\text{буква или (адрес)} = \begin{cases} \text{число} \\ \text{буква} \\ |\text{буква}| — \text{абсолютное значение} \\ -|\text{буква}| — \text{абсолютное значение со знаком «—»} \\ \text{(адрес)} \end{cases}$$

Слева под буквой понимается переменная любого типа, в том числе с индексами.

Справа же под буквой понимается переменная без индексов, т. е. там нельзя употреблять переменные с индексами.

Знак «=» рассматривается здесь как символ присвоения. Таким образом, по смыслу, значение, указанное правой частью, присваивается левой части.

Например: *допустим* $x = 3{,}527\_i = 4\_y = c$
допустим $a_{ij} = б\_m = |t|\_s = -|z|$
допустим $(300) = 27{,}4$

В последнем случае в ячейку № 300 будет записано число 27,4.

Знак «—» перед целочисленной переменной в записи данного оператора употреблять нельзя. Допускается запись в левой и правой частях формулы переменных разных типов, и тип правой части согласуется с типом левой. Например, если написано $a = i\_к = в$, то значение i превращается в число, представленное в форме с плавающей запятой и присваивается переменной a; значение $в$, представленное в форме с плавающей запятой, преобразуется так, что его целая часть присваивается $к$.

Заметим, что модульные скобки используются только в операторе «*допустим*».

Чаще всего этот оператор употребляется в начале программы для определения значений переменных, участвующих в вычислениях.

2. Оператор **«вставим...»** служит для изменения значений переменных без индексов на заданную величину и, в общем случае, имеет вид:

$$«вставим\ q = q \pm c»,$$

где c — числовая константа без знака. При этом значения переменных i, j, $к$ и n можно изменять только на целое число.

Например: *вставим* $x = x + 2{,}5\_y = y - 0{,}5\_i = i + 1$. Здесь значение переменной x увеличивается на 2,5, переменной y — уменьшается на 0,5, а значение индекса i увеличивается на единицу.

В записи рассмотренных операторов может быть несколько присваиваний в одной строке, отделяемых друг от друга пробелами.

Найдите ошибки в записи приведенных ниже операторов присваивания:

вставим $y = y - a\_z_i = z_i + 1$
вставим $i = i + 0{,}5\_j = j - к$.

§ 2. ВЫЧИСЛИТЕЛЬНЫЙ ОПЕРАТОР

Оператор **«вычислим...»** служит для организации вычислений по формулам, составленным из символов и по правилам языка АП и, в общем случае, имеет вид:

$$«вычислим\ q = Q\_q_1 = Q_1\_q_2 = Q_2 ...»,$$

где q, q_1, q_2 ... — переменные действительного типа,

Q, Q_1, Q_2 ... — арифметические выражения,
$q = Q$ — формула.

Например: а) «*вычислим* $y = (ac^2 + \sin x)/x$».

При этом сначала вычисляется числовое значение правой части и полученный результат записывается в ячейку, соответствующую переменной y.

б) «*вычислим* $x = x + a$»

В этом случае старое значение x складывается с a и записывается в ячейку хранения x.

Следует учесть, что перед исполнением данного оператора все переменные правой части должны быть численно определены.

Заметим также, что в формулы нельзя включать переменные из множества $\{i, j, к, n\}$.

Рассматриваемый оператор может быть задан и в виде: «*вычислим q*», где переменная q должна быть определена в программе раньше в другом операторе «*вычислим*» и q должна стоять в левой части формулы в этом операторе.

Найдите ошибки в записи приведенных ниже вычислительных операторов:

а) $к = a \ln z - 2{,}5\sqrt{a}/z$

б) $y = \cos 2\pi/ - \ln(t+1)$

в) $f_i = a \sin x_n - в \cos x_i$

г) $y = y + \lg(x^a + 1)$

д) $s = m \exp a - n \exp в$

е) $z = a \sin(\omega t + \varphi)$

ж) $e = ((a + b)/c) \lg(x - 1))/x.$

§ 3. ОПЕРАТОРЫ ПЕРЕХОДОВ

Алгоритм задачи содержит всегда несколько операторов, которые нумеруются в той последовательности, как они должны исполняться. Для изменения естественного порядка выполнения алгоритма служат операторы переходов.

1. Оператор безусловного перехода **«идти к...»** употребляется для безусловной передачи управления к любому оператору программы. Встретив оператор «*идти к N*», машина переходит к исполнению оператора с номером N.

2. Оператор **«если A Z 0 идти к N»** употребляется для условной передачи управления. Здесь A — арифметическое выражение любого типа, а Z — один из знаков операций отношения $=$; $\neq$; $<$; $\leqslant$; $>$; $\geqslant$. Если заданное условие выполняется,—управление передается оператору с номером N, при невыполнении — следующему по порядку.

3. Оператор *«если»* можно использовать еще при нажатой клавише «вариант». Он записывается так: **«если ! идти к N».** Тогда при нажатой клавише «вариант» управление передается оператору с меткой N.

Примеры: а) *если* $c-6{,}5\leqslant 0$ *идти к* 3;

б) *если* $i-10\neq 0$ *идти к* 7;

в) *если* $y-x=0$ *идти к* 9.

В программу оператор «если» может быть включен следующим образом.

Пример. Присвоить переменной y то значение, которое имеет бо́льшая из величин a и $в$, и любое из них, если они равны.

Решение:

```
4 . . . . . .
5 если a — в ⩾ 0 идти к 8
6 допустим y = в
7 идти к 9
8 допустим y = a
9 (продолжение программы).
```

§ 4. ОПЕРАТОРЫ ВВОДА И ВЫВОДА ИНФОРМАЦИИ

В эту группу входят следующие операторы.

1. Оператор ввода информации имеет вид: **«введем** *<список вводимых переменных>»*.

Например: *«введем a, $в_i$, x»*. При этом в список нельзя включать переменные из множества $\{i, j, к, n\}$. В списке переменных могут быть простые переменные и переменные с индексами.

При исполнении этого оператора машина останавливается и ожидает ввода числовых значений переменных, которые следует вводить в той же последовательности, что и в списке. По окончании ввода, после нажатия клавиши «возврат каретки», продолжается исполнение программы. Ввод данных осуществляется с печатающего устройства или с перфоленты.

2. Оператор **«печатаем...»** служит для печати переменных с заданным числом знаков после запятой.

Например: *«печатаем с n знаками x, y_i, c_{ij}»*. Здесь $1\leqslant n\leqslant 9$, где n указывает количество печатаемых цифр после запятой. В этом случае в одной строке машина напечатает

$x=<$число$>_\, y_i=<$число$>_\, c_{ij}=<$число$>$.

В одной строке можно печатать столько переменных, сколько умещается на бумаге ЭПМ.

В списке переменных могут быть переменные любого типа.

Примечание. Целочисленные переменные печатаются без учета заданной точности.

Пример. Оператор: «*печатаем с 5 знаками* $a\_\_в\_\_i$» осуществляет вывод информации вида

$$a = 0{,}24376 \_\_\_\_в = 27{,}30591 \_\_\_\_ i = 6.$$

3. Оператор **«храним...»** означает, что машина должна запомнить заданное количество (n) значений указанных переменных, которые будут использованы для машинного построения графика.

Например: *храним* 12__ *y*;
храним 49 __*t*__*s*.
означает, что в первом случае нужно запомнить 12 значений переменной y, а во втором — 49 пар значений переменных t и s. Здесь $t = t(x)$ и $s = s(x)$ и t, s, x — переменные действительного типа (без индексов).

4. Оператор **«начертим...»** используется ЭВМ для построения 1 или 2 графиков. Например:
начертим 1 *гр*;
начертим 2 *гр*.

§ 5. ОПЕРАТОР ИСПОЛЬЗОВАНИЯ ПОДПРОГРАММ

Оператор **«программа».**

Данный оператор служит для обращения к выполнению программ-процедур (подпрограмм), записанных в оперативной памяти или хранящихся в ДЗУ. Оператор может записываться в двух видах:

а) «*программа N*», где N — неотрицательное целое число, адрес команды, с которой необходимо начинать выполнение произвольной нестандартной программы.

Выход из подпрограммы осуществляется по команде «*и*(69+N)*н*», где N — метка оператора.

б) «*программа* <*код программы-процедуры*> (<*список фактических параметров*>)», где <*код программы-процедуры*> — наименование стандартных подпрограмм и состоит из двух букв; <*список фактических параметров*> состоит из простых переменных, идентификаторов (букв) переменных с индексами и констант. Элементы списка фактических параметров, последовательность их записывания, и сам список определяются данной стандартной подпрограммой. Таким образом, каждая подпрограмма имеет свой список фактических параметров и свою форму записи элементов списка. Элементы списка отделяются друг от друга пробелом «__».

Все программы-процедуры составлены в кодах, к ним можно обращаться также и при ручном программировании в кодах.

Ниже приводится библиотека стандартных подпрограмм с их кодами и списками фактических параметров:

1. Решение системы n линейных алгебраических уравнений с n неизвестными

$$су\,(a\_\_n\_\_в\_\_),$$

где a — идентификатор переменной с двумя индексами, матрица имеет порядок $(n+1)\times(n+2)$, $в$ — идентификатор переменной с одним индексом, массив $в$ имеет n элементов для корней, n — конкретное число. Коэффициенты исходной системы задаются как $n\times(n+1)$ элемента массива a.

2. Решение системы n линейных алгебраических уравнений с m неизвестными методом наименьших квадратов

$$нк\,(a\_\_m\_\_в\_\_n\_\_),$$

где a — идентификатор переменной с двумя индексами, матрица a имеет порядок $(n+1)\times(m+1)$; $в$ — идентификатор переменной с одним индексом, $в$ имеет m элементов для корней. Коэффициенты системы задаются как $n\times(m+1)$ элемента матрицы a. Здесь m и n — конкретные числа.

3. Интерполирование функции полиномом степени n

$$ип\,(a\_\_n\_\_в\_\_с\_\_д\_\_),$$

где a и $в$ — идентификаторы переменных с одним индексом. Массивы a и $в$ имеют по n элементов для значений аргумента и функции соответственно, $с$ — аргумент, простая переменная действительного типа или число, $д$ — функция (простая переменная), искомая для значения $с$, n — конкретное число.

4. Обращение матрицы порядка n:

методом пополнения Ершова (приближенный метод) $ом\,(a\_\_n\_\_)$, где a — идентификатор переменной с двумя индексами, матрица a имеет порядок $(n+2)\times n$. Исходная матрица задается как $n\times n$ элемента в a. Здесь n — конкретное число.

методом Жордана $мо\ (a\_\_n\_\_в\_\_)$,

где a — идентификатор переменной с двумя индексами, матрица a имеет порядок $(2n+3)\times n$, а матрица $в$ — $n\times n$. Исходная матрица задается как $n\times n$ элемента в a, результат получается в $в$. Здесь n — конкретное число.

5. Вычисление определителя матрицы порядка n

$$во\,(a\_\_n\_\_в\_\_),$$

где a — идентификатор переменной с двумя индексами. Матрица a имеет порядок $n\times n$ для исходной матрицы, $в$ — простая переменная и служит для результата, n — конкретное число.

6. Умножение матриц

$$ум\,(a\_\_в\_\_с\_\_m\_\_n\_\_p\_\_),$$

где a, $в$, $с$ — идентификаторы переменных с одним или двумя индексами в зависимости от m, n и p;

Матрица a имеет порядок $m\times n$, $в$ — $n\times p$, матрица $с$ — $m\times p$. Исходные матрицы задаются как массивы a и b, результат получается в массиве $с$. Здесь m, n, p — конкретные числа.

7. Вычисление интеграла

$$ил(а\_\_в\_\_с\_\_д\quad е\_\_и\_\_),$$

где *а, в, с* — простые переменные (аргумент, функция и результат соответственно); *д, е, и* — простые переменные и константы действительного типа (нижний, верхний пределы интегрирования и точность интегрирования соответственно).

8. Решение системы n обыкновенных дифференциальных уравнений первого порядка методом Рунге—Кутта:

с постоянным шагом интегрирования

$$рк(а\_\_п\_\_в\_\_с\_\_д\_\_е\_\_и\_\_);$$

с автоматическим выбором шага

$$ду(а\_\_п\_\_в\_\_с\_\_д\_\_е\_\_и\_\_р\_\_).$$

Здесь *а* — идентификатор переменной с одним индексом, содержит $3n$ элемента. Начальные значения функций задаются как первые n элементов этого массива, *в* — идентификатор переменной с одним индексом, массив *в* имеет n элементов и служит для обозначения значений производных функций, *д* — аргумент, простая переменная. Величины *с, е, и, р* — простые переменные, константы действительного типа и служат соответственно как нижний предел, верхний предел, шаг интегрирования и точность вычисления. Здесь n — конкретное число. В случае *ду* массив *а* имеет $6n$ элемента.

Ниже приводятся примеры использования СП для вычисления интеграла с точностью 0,00005 в интервале (0,1) от функции $y=x^2\sin x+\sqrt{x}-\cos\pi x$ и СП для вычисления определителя и обращения матрицы.

Пример 1.

ап 4—7—1973 *г*1 ≪*наири*≫
1__ *допустим* $x=0$__ $a=0$__ $в=1$__ $e=0{,}00005$
2__ *вычислим* $y=x^2$__ $\sin x+$ __$\sqrt{x}-$ __$\cos\pi x$
3__ *программа* *ил*(*x*__*y*__*c*__*a*__*в*__*e*__)
4__ *печатаем с* 9 *знаками с*
5__ *кончаем*
исполним 1
$c=0{,}889891125$ — полученное значение интеграла.

Пример 2.

Исходная матрица имеет вид:

$$a_{ij}=\begin{pmatrix}1&2&3&4\\2&3&4&3\\3&4&3&2\\4&3&2&1\end{pmatrix}$$

ап 4—7—1973 *г* 2 ≪*наири*≫

$i = 3$__$k = 3$__*а с*
$k = 10$__$j = 3$__*в*
1__ *допустим* $i = 0$
2__ *допустим* $j = 0$
3__ *введем* a_{ij}
4__ *вычислим* $в_{ij} = a_{ij}$
5__ *вставим* $j = j + 1$
6__ *если* $j - 4 < 0$ *идти к* 3
7__ *вставим* $i = i + 1$
8__ *если* $i - 4 < 0$ *идти к* 2
9__ *программа во*(*а*__4__*х*__)
10__ *печатаем с* 9 *знаками* x
11__ *программа мо*(*в*__4__*с*__)
12__ *интервал* 3
13__ *допустим* $i = 0$
14__ *допустим* $j = 0$
15__ *печатаем с* 9 *знаками* c_{ij}
16__ *вставим* $j = j + 1$
17__ *если* $j - 4 < 0$ *идти к* 15
18__ *вставим* $i = i + 1$
19__ *если* $i - 4 < 0$ *идти к* 14
20__ *кончаем*
исполним 1
1 2 3 4
2 3 4 3
3 4 3 2
4 3 2 1

Решение:

x = 20,0000000476 — определитель матрицы
$c_{0'0}$ = 0,099999997
$c_{0'1}$ = 0,000000000
$c_{0'2}$ = — 0,499999992
$c_{0'3}$ = 0,599999994
$c_{1'0}$ = 0,000000000
$c_{1'1}$ = — 0,499999992
$c_{1'2}$ = 0,999999985
$c_{1'3}$ = — 0,499999988
$c_{2'0}$ = — 0,499999992
$c_{2'1}$ = 0,999999992
$c_{2'2}$ = — 0,499999992
$c_{2'3}$ = — 0,000000001
$c_{3'0}$ = 0,599999994
$c_{3'1}$ = — 0,499999992
$c_{3'2}$ = — 0,000000003
$c_{3'3}$ = 0,099999999.

§ 6. СЛУЖЕБНЫЕ ОПЕРАТОРЫ

1. Оператор **«спросим»** дает возможность запросить машину о текущем значении каких-либо переменных. Он имеет вид *«спросим»*.

Если в ходе решения задачи требуется проверить значение ряда параметров или величин, надо нажать клавишу «Вариант». При этом машина, как только встретит оператор *«спросим»*, останавливается, готовая к запросу. Далее необходимо напечатать переменную, значение которой проверяется, вместе со знаком =, после чего будет выдан ответ. Поскольку в программе может содержаться несколько операторов *«спросим»*, то перед остановкой машина печатает порядковый номер этого оператора.

Для продолжения решения задачи нужно восстановить клавишу «Вариант» и нажать клавишу «Возврат каретки».

Если клавиша «Вариант» не нажата, то оператор *«спросим»* пропускается.

2. Оператор **«массив...»** контролирует размещение программы решаемой задачи в оперативной памяти.

Этот оператор может быть использован в двух вариантах:

а) *«массив к сост М команд»* предусмотрен для выдачи на печать количества машинных команд, составленных по АП до момента ввода этого оператора;

б) *«массив в нач* 370» обеспечивает вывод на печать машинной программы, составленной по АП-программе до ввода этого оператора.

Номера команд не указываются, поскольку начало программы строго зафиксировано адресом 370. Оператор *«массив...»* вводится без порядкового номера.

3. Оператор **«сотрем».** Этот оператор имеет вид *«сотрем* с N» и пишется без метки.

В программе, составленной на языке АП, этим оператором пренебрегаются (стираются) введенные операторы, начиная с оператора меткой N. Результирующая программа этих операторов стирается из памяти засылкой адреса первой команды результирующей программы N-го оператора в ячейку 64. Стирание происходит сразу после нажатия клавиши «в.к.».

4. Оператор **«лента»** служит для организации перфорации ленты. Может задаваться в двух видах:

а) *«лента повт реш* N *оп»*, где N — количество операторов в АП-программе. Получим перфоленту для повторного решения задачи;

б) *«лента прод реш* N *оп»,* где N — то же, что и в а). После ввода полученной перфоленты можно продолжить решение задачи.

5. Оператор **«интервал** *m»* обеспечивает пропуск указанного количества строк (*m*) для отделения таблиц печатаемых величин.

Например, *«интервал 6»* означает, что следует пропустить 6 строк перед следующим печатаемым массивом.

6. Оператор **«останов»** употребляется главным образом для отладки задач. Он дает возможность остановить вычисления в нужный момент, то есть выполняет функцию останова по адресу. Нажатием клавиши «Пуск 2» решение задачи продолжается.

7. Оператор **«кончаем»** указывает на конец вычислений. Им завершается операторная программа.

8. Оператор **«исполним N»** является командой пуска машины для исполнения программы, начиная с оператора с номером N. Решение задачи начинается при нажатии клавиши «Возврат каретки». При этом машина проверяет, поместятся ли рабочая программа, переменные и т. д. в оперативной памяти. Если рабочая информация не помещается, машина печатает *«не в силах»*. Оператор *«исполним»* вводится без порядкового номера в самом конце программы.

Глава 11

СОСТАВЛЕНИЕ ОПЕРАТОРНЫХ ПРОГРАММ

§ 1. СТРУКТУРА ПРОГРАММ НА ЯЗЫКЕ «АП»

При решении задачи на ЭВМ выявляется алгоритм, который затем расписывается на языке АП как последовательность операторов. Каждая программа, записанная на языке АП, должна заканчиваться оператором *«исполним»*.

Если при решении данной задачи используются переменные с индексами, то перед первым вхождением идентификатора переменной с индексами в АП-программу задается описательная часть массива. В описательной части массива задаются индексы, их максимальные значения и идентификаторы (буквы), которые должны снабжаться индексами. Таким образом, перед первым использованием какой-нибудь буквы с индексами мы обязательно должны задать, в каких пределах изменяются индексы у данной переменной.

Возможны следующие три вида структур программ, написанных на языке АП:

I	II	III
описательная часть массива	операторы *исполним*	операторы
операторы		описательная часть массива

I	III
оператор	операторы
описательная часть	. . . . .
массива	оператор
операторы	*исполним*
. . . . .	
оператор	
исполним	

Все операторы за исключением «*лента*», «*массив*», «*сотрем*», «*исполним*» начинаются с числовой метки — порядкового номера оператора.

Первым всегда выполняется оператор, номер которого указывается в операторе «*исполним*». Затем операторы выполняются в естественном порядке возрастания их номеров (меток). Этот порядок нарушается в случае, если встретились операторы передачи управления («*идти*», «*если*», «*программа*»). Если мы оператором «*программа*» обращаемся к СП, то порядок выполнения операторов не нарушается.

Программа в ходе решения может быть прервана по следующим причинам:

1) выполнился оператор «*останов*»;
2) выполнился оператор «*кончаем*» (означает естественный конец программы);
3) выполнился оператор «*спросим*» (при нажатой клавише «вариант»);
4) остановка с помощью пульта управления;
5) сбой машины или переполнение.

В случаях (1—4) значения всех вычисленных к этому моменту времени переменных сохраняются и могут использоваться при дальнейшем решении задачи. В случае (5) дальнейшее решение к правильному результату не приводит. В этом случае нужно выяснить подробно причину останова и возможно повторно решить задачу.

Рассмотрим некоторые примеры построения операторных программ для различных вычислительных процессов.

§ 2. ЦИКЛИЧЕСКИЕ ПРОЦЕССЫ

Как известно, в циклических вычислительных процессах предусматривается многократное повторение некоторой группы операций, причем обычно задается либо число повторений циклов, либо условие их окончания. Такие процессы можно организовать с помощью достаточно эффективных и экономичных операторных программ.

Пример 1. Требуется вычислить значения некоторой функции $f(x)$ для $x = 0; 0{,}3\ 0{,}6; 0{,}9; 1{,}2; 1{,}5$.

Поскольку шаг изменения аргумента здесь постоянный: $\Delta x = 0{,}3$, то программу построим следующим образом:

```
        an
1__ допустим x = 0                    (вк)
2__ вычислим y = f(x)                 (вк)
3__ печатаем с 5 знаками x__y         (вк)
4__ вставим x = x + 0,3               (вк)
5__ если x—1,51<0 идти к 2            (вк)
6__ кончаем
исполним 1
```

Теоретически можно было бы использовать в условном операторе 5 отношение $\neq$ (т. е. «*если* $x-1{,}5\neq 0$ *идти к* 2»). Однако, это нежелательно, поскольку число 0,3 в двоичной системе записывается бесконечной периодической дробью и потому не имеет в машине точного представления. Если сложить пять раз число $\sim 0{,}3$, то точного значения величины 1,5 не получится. Следовательно, условие $\neq$ всегда выполнено и машина не выйдет из цикла.

Поэтому к употреблению условий вида « $=$ » и « $\neq$ » нужно подходить весьма осторожно.

Пример 2. Требуется вычислить сумму членов степенного ряда

$$s = \sum_{i=1}^{12} x^i = x + x^2 + x^3 + \ldots + x^{12}$$

при заданном числовом значении x.

В программах этого типа повторение циклов обеспечивается операторами счета циклов и условной передачи управления. Предполагается два варианта построения операторной программы.

I вариант

```
        an
1 допустим x=2 s=0 n=0 a=1
2 вычислим a = ax
3 вычислим s = s + a
4 вставим n = n + 1
5 если n — 12 < 0 идти к 2
6 печатаем с 5 знаками s
7 кончаем
исполним 1
```

II вариант

```
        an
1 допустим x = 2 s = 0 n = 0
2 вычислим s = (s + 1)x
3 вставим n = n + 1
4 если n — 12 < 0 идти к 2
5 печатаем с 5 знаками s
6 кончаем
исполним 1
```

В первом варианте оператор 2 вычисляет в каждом цикле очередной член ряда x^i и записывает его значение в ячейку хранения переменной a (то есть $a=x^i$). Оператор 3 вычисляет и накапливает сумму x^i путем прибавления очередных значений $a=x^i$ к величине накопленной до этого суммы s_{i-1} (начиная с нулевой $s_0=0$), то есть $s_i=s_{i-1}+s_i$ или $s=s+a$. Операторы 4 и 5 обеспечивают повторение циклов и их окончание при $n=12$.

Во втором варианте оператор 2 вычисляет значения очередных членов ряда с одновременным накоплением суммы по фор-

муле, соответствующей модифицированной схеме Горнера. Операторы 4 и 3 выполняют адекватные функции счетчиков циклов.

Пример 3. Составить программу табулирования функции

$$y = \frac{4\sqrt{1 - ax^2}}{1 + \sqrt{a^2 + x^2}},$$

зависящей от параметра a на отрезке $0{,}1 \leqslant x \leqslant 0{,}9$ с шагом $h = \Delta x = 0{,}2$. Табулирование производится для трех значений параметра $a = 0{,}15;\ 0{,}35;\ 0{,}54$.

В задачах такого рода удобно использовать оператор ввода. При каждом останове машины по этому оператору очередное значение параметра a вводится с клавиатуры печатающего устройства.

```
                an
1__ введем a                                              (вк)
2__ допустим x = 0,1                                      (вк)
3__ вычислим y = 4(__)(1 - ax^2))/(1 + __)(a^2 + x^2))    (вк)
4__ печатаем с 5 знаками x__y                             (вк)
5__ вставим x = x + 0,2                                   (вк)
6__ если x - 0,91 < 0 идти к 3                            (вк)
7__ интервал 2                                            (вк)
8__ идти к 1                                              (вк)
исполним 1
```

Пример 4. Составить программу вычисления двух функций:

$$y = (x^2 - 1)(x - 2)\sin\frac{2\pi x}{3}$$

и

$$z = (x^2 - 1)(x - 2)\cos\frac{2\pi x}{3}$$

с построением их графиков в интервале изменения аргумента $(-1 \leqslant x \leqslant 2)$ с шагом $h = \Delta x = 0{,}0625$.

Для построения графиков определяем число шагов по формуле:

$$n = \frac{b - a}{h} + 1 = \frac{2 - (-1)}{0{,}0625} + 1 = 49$$

Программа

```
an
1__ допустим x = -1
2__ вычислим y = (x^2 - 1)(x - 2)__ sin 2πx/3
3__ вычислим z = (x^2 - 1)(x - 2)__ cos 2πx/3
4__ печатаем с 5 знаками x__y__z__
5__ храним 49 y__z
6__ вставим x = x + 0,0625
```

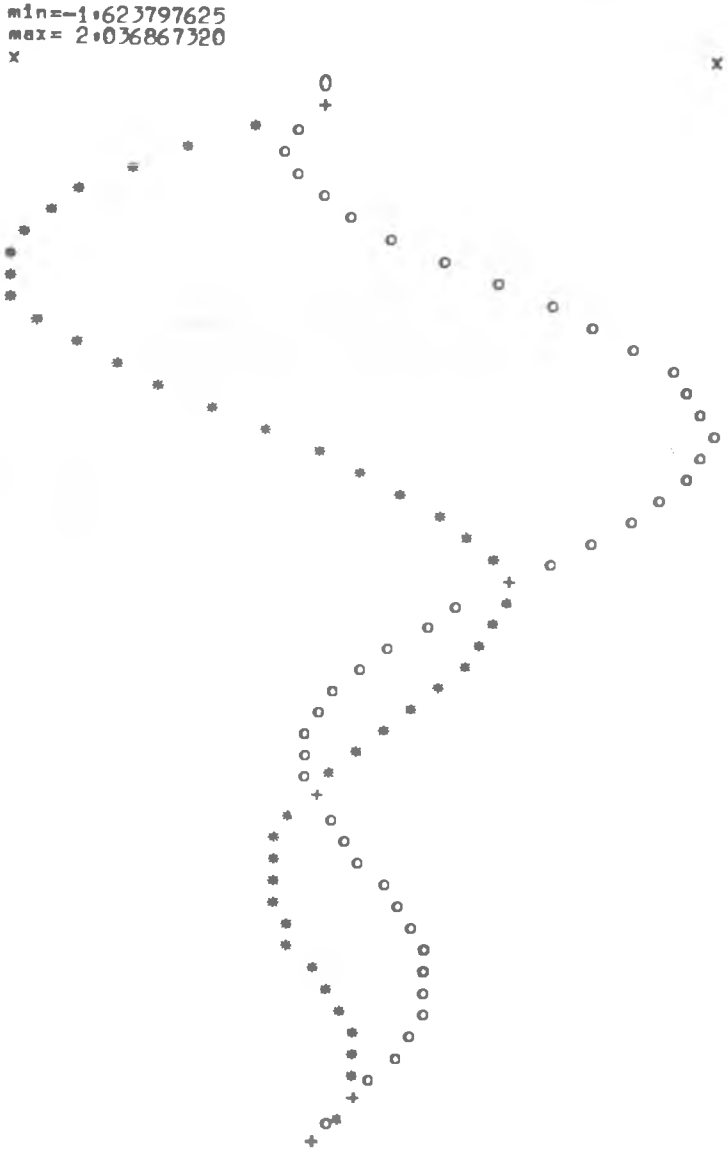


Рис. 10.

7__ *если* $x-2\leqslant 0$ *идти к* 2 (вк)
8__ *начертим* 2 *гр* (вк)
9__ *кончаем*
исполним 1

Решение

$x = -1{,}00000$	$y = 0{,}00000$	$z = 0{,}00000$
$x = -0{,}93750$	$y = -0{,}32863$	$z = -0{,}13612$
$x = -0{,}87500$	$y = -0{,}65086$	$z = -0{,}17439$
. . . . .	. . . . .	. . . . .
$x = 1{,}93750$	$y = 0{,}13655$	$z = 0{,}10477$
$x = 2{,}00000$	$y = 0{,}00000$	$z = 0{,}00000$

§ 3. КРАТНЫЕ ЦИКЛИЧЕСКИЕ ПРОЦЕССЫ

При решении некоторых задач приходится строить сложные циклические программы, в которых имеется два или более «вложенных» друг в друга циклов. Для организации таких процессов образуют сначала внутренние, а затем внешние циклы. Подобные случаи встречаются в задачах группировки, комбинаторики, в операциях с матрицами и т. п. Рассмотрим следующий пример.

Пример 5. Вычислить объем тела, изображенного на рис. 11, если сверху оно ограничено поверхностью:

$$z = \frac{x}{2}\ln(y+2) + \frac{y}{3}\ln(x+3) + 5.$$

Задачу решить путем численного интегрирования с шагом $h=0{,}1$ по обеим осям, при $a=4$ и $b=6$.

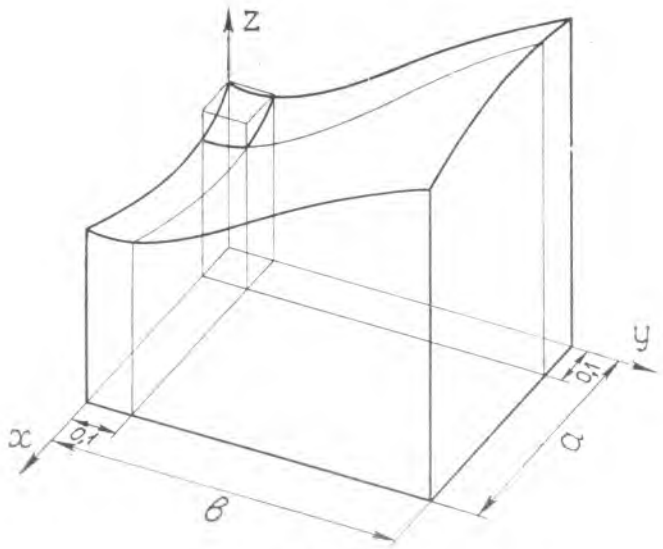


Рис. 11.

Объем заданного тела выражается формулой

$$u = \int_0^a \int_0^b z\,dx\,dy.$$

Рассечем тело на ряд слоев толщиной $h = 0{,}1$ плоскостями, параллельными координатной плоскости $z0y$. В свою очередь, рассекая такой слой плоскостями, параллельными $x0z$, получим ряд столбиков (параллелепипедов) с высотой z. Если основание каждого столбика будет иметь площадь $h^2 = 0{,}01$, то его объем составит $0{,}01\,z$.

Суммируя объемы всех столбиков, получим приближенный объем тела. Сложение будем осуществлять перемещаясь вначале по 1-му слою, затем по 2-му и т. д. до $x = 4$.

Программа

an

```
1__ допустим x = 0__ u = 0
2__ допустим y = 0
3__ вычислим z = (x/2)__ ln(y + 2) + (y/3)__ ln(x + 3) + 5
4__ вычислим u = u + 0,01z
5__ вставим y = y + 0,1
6__ если y — 6 <0 идти к 3
7__ вставим x = x + 0,1
8__ если x — 4 < 0 идти к 2
9__ печатаем с 3 знаками u
10__ кончаем
исполним 1
```

Здесь внутренний цикл вычисления объема слоя образуется операторами $3 \div 6$, а внешний цикл вычисления полного объема — операторами $2 \div 8$. Величина полного объема накапливается оператором 4 как значение u.

§ 4. ИТЕРАЦИОННЫЕ ПРОЦЕССЫ

Рассмотрим примеры построения операторных программ итерационных вычислительных процессов, сущность которых состоит в том, что искомая величина y находится путем последовательных приближений, когда результаты предыдущих вычислений используются как исходные данные для последующих. При этом, каждое последующее значение y_{i+1} по сравнению с предыдущим y_i ближе к истинному значению искомой величины y (для сходящихся процессов). Вычисления ведутся до получения заданной степени точности, то есть до выполнения условия

$$| y_{i+1} - y_i | \leqslant \varepsilon.$$

Пример 6. Решить нелинейное уравнение

$$x = 3{,}5\,\frac{x}{1+x^2}\cdot \ln 38{,}5x^2$$

методом простой итерации с точностью $e = 0{,}0001$.

ап

1__ *допустим* $x = 10$__ $e = 0{,}0001$
2__ *вычислим* $y = (3{,}5x$__ $\ln 38{,}5x^2)/(1 + x^2)$
3__ *вычислим* $m = y - x$
4__ *допустим* $x = y$__ $б = |m|$
5__ *если* $б - e > 0$ *идти к* 2
6__ *печатаем с* 5 *знаками* x
7__ *кончаем*
исполним 1

Выделение модуля разности двух приближений в операторе 4 производится в связи с тем, что получаемые значения m могут оказаться отрицательными или знакопеременными.

При вычислении сходящихся рядов определение момента окончания вычислений можно произвести аналогичным образом, то есть путем сравнивания величины каждого очередного члена ряда с заданной степенью точности. Операторные программы в этих случаях удобнее всего строить на основе рекуррентных соотношений, которые позволяют вычислять очередной член ряда через предыдущие.

Пример 7. Составить программу вычисления ряда

$$y = f(x) = \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} + \ldots$$

для значения $x = 0{,}5$ с точностью вычислений $e = 0{,}00001$.

Исходя из отношений двух соседних членов ряда

$$a_{n+1} : a_n = \frac{x^{n+1}}{(n+1)!} : \frac{x}{n!} = \frac{x}{n+1},$$

получим выражение:

$$a_n = a_{n-1}\cdot \frac{x}{n},$$

которое удобно использовать при составлении операторной программы.

ап

1__ *допустим* $y = 0$__ $a = 1$__ $в = 1$__ $x = 0{,}5$__ $e = 0{,}00001$
2__ *вычислим* $c = ax/в$
3__ *вычислим* $y = y + c$
4__ *вставим* $в = в + 1$
5__ *допустим* $a = c$__ $б = |c|$
6__ *если* $б - e > 0$ *идти к* 2

7__ *печатаем с 5 знаками у*
8__ *кончаем*
исполним 1

§ 5. РАЗВЕТВЛЯЮЩИЕСЯ ПРОЦЕССЫ

Разветвляющиеся вычислительные процессы наиболее характерны для случаев зависимости расчетных формул от входящих в них переменных. При этом, на определенном этапе расчетов, соотношение полученных значений переменных и заданных критериальных условий будет определять дальнейший ход вычислений, использование тех или иных функциональных зависимостей. Подобного рода процессы встречаются также в теории игр, при решении тактических задач и т. п.

В операторных программах разветвления обычно осуществляются с помощью условных операторов, позволяющих проверять отношения вида: $=$, $\neq$, $>$, $\geqslant$, $<$, $\leqslant$.

Рассмотрим примеры

Пример 8. Вычислить значение функции $f(x,y)$ по одной из двух формул

$$f(x,\ y)=\begin{cases}\sqrt{y-x/в}, & \text{если } x+y\geqslant в,\\ (x-y/в)^2, & \text{если } x+y<в,\end{cases}$$

при изменении x в интервале $-5\leqslant x\leqslant+5$ и значениях $y=x^2+вx$ и $в=3{,}5$.

ап

1__ *допустим* $в=3{,}5$
2__ *введем* x
3__ *вычислим* $y=x^2+вx$
4__ *вычислим* $а=x+y$
5__ *если* $а-в<0$ *идти к* 9
6__ *вычислим* $д=\sqrt{}(y-x/в)$
7__ *печатаем с* 5 *знаками* $д$
8__ *идти к* 2
9__ *вычислим* $д=(x-y/в)^2$
10__ *идти к* 7
исполним 1

Здесь оператор 5 определяет направление вычислений в зависимости от актуального состояния отношения $x+y<в$. Оператором 2 вводятся значения x в указанном интервале.

Пример 9. Составить программу вычисления величины $s=f(z)+z$, где $z=t^3+mt$ и функция

$$f(z)=\begin{cases}\sqrt{m/z}-1{,}5, & \text{если } z>0,\\ 1+\cos^2 z, & \text{если } -1<z\leqslant 0,\\ m(z^2+1), & \text{если } z\leqslant -1,\end{cases}$$

при изменении t в интервале $-1{,}5 \leqslant t \leqslant 1{,}5$ с шагом $h = \Delta t = 0{,}1$ и $m = 0{,}5$.

an

1__ *допустим* $t = -1{,}5$__ $m = 0{,}5$__ $l = -1$
2__ *вычислим* $z = t^3 + mt$
3__ *если* $z > 0$ *идти к* 11
4__ *если* $z - l > 0$ *идти к* 9
5__ *вычислим* $d = m(z^2 + 1)$
6__ *вычислим* $s = d + z$
7__ *печатаем с* 5 *знаками* t__z__s
8__ *идти к* 13
9__ *вычислим* $d = 1 + (\_\cos z)^2$
10__ *идти к* 6
11__ *вычислим* $d = \_\sqrt{m/z - 1{,}5}$
12__ *идти к* 6
13__ *вставим* $t = t + 0{,}1$
14__ *если* $t - 1{,}5 \leqslant 0$ *идти к* 2
15__ *кончаем*
исполним 1

Здесь операторы 6 и 7 общие для всех вычислительных ветвей. Оператор 13 осуществляет требуемое изменение аргумента t.

Рассмотрим теперь пример операторной программы с обращением к вспомогательной нестандартной программе, составленной в адресных кодах машины.

Пример 10. Вычислить скалярное произведение двух десятимерных векторов $\bar{a}$ и $\bar{в}$.

Координаты вектора $\bar{a}$ размещаем в ячейках 800÷809, а вектора $\bar{в}$ — в ячейках 810÷819 оперативного накопителя.

800*n*10*n*	810*n*10*n*
1,00000	—1,00000
1,50000	2,00000
2,00000	—3,00000
2,50000	4,00000
3,00000	—5,00000
3,50000	6,00000
4,00000	—7,00000
4,50000	8,00000
5,00000	—9,00000
5,50000	10,00000.

Нестандартную вспомогательную программу разместим в ячейках 710÷714. Эта программа будет поочередно засылать значения координат a_i и $в_i$ в ячейки 42 и 34, соответствующие буквам a и $в$.

```
710к5
710 п700н1
711 п800н42 +
712 п810н34 +
713 с2048к700
714 и71н.
```

Вычисление суммы $\sum_{i=0}^{9} a_i \; в_i$ произведем по операторной программе

ап

```
1__ допустим s = 0__ (700) = 0__ i = 0
2__ программа 710
3__ вычислим s = s + ав
4__ вставим i = i + 1
5__ если i − 10 < 0 идти к 2
6__ печатаем с 5 знаками s
7__ кончаем
исполним 1.
```

При размещении нестандартной программы и числовых массивов в оперативном накопителе нужно следить за тем, чтобы они не попали в ячейки, используемые транслятором АП при обработке операторной программы.

§ 6. ОПЕРАЦИИ С МАССИВАМИ

В языке АП для обозначения переменных (действительного типа) можно использовать буквы с одним или двумя индексами. Совокупность переменных, обозначенных одной и той же буквой с индексами, называется массивом, а каждая переменная — компонентой (элементом) массива. Массив переменных, обозначенных буквой с одним индексом, называется одномерным. Массив переменных, обозначенных буквой с двумя индексами — двумерным.

В описательной части массива для программы, включающей переменные с индексами, обязательно проводится описание массивов по следующей форме.

Для одномерных массивов:

<индекс-буква> = <макс. значение первого индекса> <список букв>.

Для двумерных массивов:

<индекс буква> = <макс. значение первого индекса> (пробел) <индекс-буква> = <макс. значение второго индекса> (пробел) <список букв>.

Здесь индекс-буква — одна из букв *i*, *j*, *к* нижнего регистра печатающего устройства. Если количества элементов в массивах одинаковы, то максимальное значение индекса задается

общим для всех букв, с учетом, что минимальное значение индекса равно нулю.

Примеры.

Описание: $i=8\_\_a\ в\ c$ — означает, что в программе будут использоваться следующие массивы переменных:

$$a_0,\ a_1 \dots,\ a_8$$
$$в_0,\ в_1, \dots,\ в_8$$
$$c_0,\ c_1, \dots,\ c_8.$$

Описание: $i=3\_\_\ j=5\_\_\ a\ в\ c$ — представляет описание следующих массивов переменных:

$$a_{00},\ a_{01}, \dots,\ a_{05}\quad в_{00},\ в_{01}, \dots,\ в_{05}\quad c_{00},\ c_{01}, \dots,\ c_{05}$$
$$a_{10},\ a_{11}, \dots,\ a_{15}\quad в_{10},\ в_{11}, \dots,\ в_{15}\quad c_{10},\ c_{11}, \dots,\ c_{15}$$
$$a_{20},\ a_{21}, \dots,\ a_{25}\quad в_{20},\ в_{21}, \dots,\ в_{25}\quad c_{20},\ c_{21}, \dots,\ c_{25}$$
$$a_{30},\ a_{31}, \dots,\ a_{35}\quad в_{30},\ в_{31}, \dots,\ в_{35}\quad c_{30},\ c_{31}, \dots,\ c_{35}.$$

Индекс-буквы в описательной части массива служат только для выделения памяти для переменных с индексами. В дальнейшем на использование целочисленных переменных это не сказывается, поэтому индексам мы можем присваивать другие значения, а переменным присваивать другие индексы.

Например:

an

$i=3\_\_\ j=5\ \_\_a\ в\ c$
$1\_\_$ *допустим* $к=1\_\_\ j=2\_\_\ i=5\_\_\ n=6$
$2\_\_$ *вычислим* $a_{jк}=\_\_\sqrt{(в_{ji}+2c_{kj})}$.

Основные правила операций с массивами

При составлении операторных программ с использованием массивов переменных нужно учитывать следующие правила и условия.

1. Буквы, обозначающие в программе переменные с индексами, нельзя употреблять без индексов в формулах.

2. Если в программе используются переменные с двумя индексами, то в одном из операторов «*допустим...*» переменной n следует придать значение, на единицу большее максимального значения второго индекса. Эта засылка необходима для правильного исполнения программы и должна быть выполнена до первого вызова переменной с индексами.

3. Во всех операторах, за исключением операторов «*допустим*», «*вводим*», «*печатаем*», «*спросим*», допускаются лишь буквенные индексы при переменных. Поэтому для использования конкретной переменной с индексами следует предварительно в операторах присваивания придать необходимые числовые значения тем переменным (i, j, $к$), которыми будут обозначены соответствующие индексы.

Например, произведение: $c_{5,7}=a_{5,6}\cdot b_{6,5}$ можно вычислить следующей группой операторов:

N__ *допустим* $i=5$__ $j=6$__ $к=7$
N+1__ *вычислим* $c_{iк}=a_{ij}\,b_{ji}$

4. Ввод массива переменных можно произвести с помощью оператора *«введем...»*.

Например, если требуется ввести массивы переменных a_i и b_i ($i=0, 1, 2, ...9$), то можно использовать типовую программу ввода:

1__ *допустим* $i=0$
2__ *введем* a_i__ b_i
3__ *вставим* $i=i+1$
4__ *если* $i-9\leqslant 0$ *идти к* 2
5__ . . . . . .

5. Для вывода на печать переменные с индексами включаются в содержание оператора *«печатаем...»*.

Например: *«печатаем с 5 знаками* a_i *и* b_i*»*.

При выводе значений переменных последние печатаются с текущими значениями индексов.

6. При использовании переменных с двумя индексами, цифровые индексы необходимо разделять запятой, буквенные индексы запятой не разделяются. Например $a_{5,6}$; c_{ji}.

Рассмотрим примеры операторных программ.

Пример 11. Составить программу табулирования функции

$$y_i=\sin\frac{\pi}{2}\,(x_i^2-1)$$

при дискретно заданных значениях аргумента $x_i=x_0, x_1, x_2, ..., x_9$.

ап

$i=9$__ $x\ y$
1__ *допустим* $i=0$
2__ *введем* x_i
3__ *вычислим* $y=$__$\sin\pi(x_i{}^2-1)/2$
3__ *печатаем с 5 знаками* y_i
5__ *вставим* $i=i+1$
6__ *если* $i-9\leqslant 0$ *идти к* 2
7__ *кончаем*
исполним 1.

Пример 12. Вычислить сумму вида: $s=\sum\limits_{i=0}^{7} a_i\cdot a_{i+2}$. Поскольку в языке АП нельзя использовать сложные индексы, то $i+2$ заменяем на j.

an

$i = 9$__a
1__ *допустим* $i = 0$
2__ *введем* a_i
3__ *вставим* $i = i + 1$
4__ *если* $i - 10 < 0$ *идти к* 2
5__ *допустим* $i = 0$__ $j = 2$__ $s = 0$
6__ *вычислим* $s = s + a_i a_j$
7__ *вставим* $i = i + 1$__$j = j + 1$
8__ *если* $i - 8 < 0$ *идти к* 6
9__ *печатаем с* 5 *знаками* s
10__ *кончаем*
исполним 1

Первые четыре оператора этой программы обеспечивают ввод переменных $a_0, a_1, \ldots, a_9$. Операторы $5 \div 8$ образуют цикл вычисления суммы $\Sigma a_i a_{i+2}$. Оператором 5 переменным i и j придаются начальные значения 0 и 2. В операторе 7 переменные i и j изменяются таким образом, что значение переменной j всегда на две единицы больше, чем значение i.

Пример 13. Составить программу вычисления следа квадратной матрицы.

По определению след матрицы равен сумме диагональных элементов $\Sigma_{i=j} a_{ij}$.

Примем $A = \begin{Vmatrix} a_{00}, & a_{01}, & a_{02}, & a_{03}, & a_{04} \\ a_{10}, & a_{11}, & a_{12}, & a_{13}, & a_{14} \\ a_{20}, & a_{21}, & a_{22}, & a_{23}, & a_{24} \\ a_{30}, & a_{31}, & a_{32}, & a_{33}, & a_{34} \\ a_{40}, & a_{41}, & a_{42}, & a_{43}, & a_{44} \end{Vmatrix}$

С помощью первой части программы организуем ввод элементов матрицы (операторы $1 \div 7$).

an

$i = 4$__ $j = 4$__a
1__ *допустим* $i = 0$__ $n = 5$ (см. указание 2)
2__ *допустим* $j = 0$
3__ *введем* a_{ij}
4__ *вставим* $j = j + 1$
5__ *если* $j - n < 0$ *идти к* 3
6__ *вставим* $i = i + 1$
7__ *если* $i - n < 0$ *идти к* 2
8__ *допустим* $i = 0$__ $m = 0$
9__ *вычислим* $m = m + a_{ii}$
10__ *вставим* $i = i + 1$
11__ *если* $i - n < 0$ *идти к* 9

12_ *печатаем с* 5 *знаками т*
13_ *кончаем*
исполним 1

Заметим, что в операторе 9 переменная a взята с двумя одинаковыми индексами. Так как суммируются только диагональные элементы матрицы, значения первого и второго индексов при элементах суммы $\sum_{i=j} a_{ij}$ совпадают, поэтому их можно обозначить одной и той же буквой, например i.

Отметим, что к индексированным переменным прибегают и при нехватке алфавита языка АП для обозначения элементов исходного математического текста.

§ 7. ВВОД ИНФОРМАЦИИ И ИСПРАВЛЕНИЕ ОШИБОК

Программу на языке АП и числовые данные можно вводить непосредственно с ЭПМ или с перфоленты. Учитывая быстроту ввода, желательным является ввод с перфоленты. Перфолента с информацией подготавливается на ТА РТА-2 или на специальном устройстве.

Рассмотрим ввод программы с ЭПМ. Первоначально 1 раз в день производится ввод даты в ОЗУ в следующем порядке:

1. Нажимаем «Пуск-1».
2. Набираем на ЭПМ символ *ч*; ЭВМ делает «*вк*», «*пс**».
3. Набираем: число, символ «—», месяц, символ «—», год, символ «*г*»;
4. После этого ЭВМ делает «*вк*», «*пс*» и остановится на команде ввода. Дата получается в ячейке 1994. Например, нажимаем «Пуск-1», *ч* 30-06-1973 *г*.

Чтобы ввести программу, необходимо:

1. Нажать клавишу «Пуск-1»;
2. Набрать буквы «*ап*» на клавиатуре ЭПМ, при этом машина печатает число, месяц, год и делает «*вк*» и «*пс*»;
3. Набрать описательную часть массива, если есть переменные с индексами, то есть печатать: «*индекс-буква*», — ЭВМ печатает знак равенства, печатаем максимальное значение индекса, нажимаем клавишу «пробел», обозначаемый _ (если есть второй индекс, то для него делается то же самое), печатаем идентификаторы (без «_»), к которым относится индекс и производим возврат каретки «*вк*».

Например, для переменных a_{ij} и b_{ij} печатаем

$$i = 3\_\ j = 5\_ав«вк»$$

---

* «*пс*» — перевод строки.

4. Набрать порядковый номер оператора, сделать пробел, затем печатать первые 2 буквы названия оператора (остальное допечатывает ЭПМ) и информацию, идущую после названия оператора. В конце, если нужно, производится возврат каретки. Остальные операторы вводятся аналогично. После набора «*исполним* N» и «*вк*» ЭВМ переходит к решению задачи.

5. При вводе числовых констант по оператору «*введем*» после набора каждой константы необходимо делать пробел или производить возврат каретки.

При составлении программы на языке АП, подготовке перфоленты и наборе программы на ЭПМ может быть полезной сводная таблица записи операторов. Слева операторы написаны в общей форме, обычно употребляющейся. Справа даны те же операторы, набранные на ЭПМ или пробитые на перфоленте с указанием только тех символов операторов и их содержимого, которые набирались программистом на ЭПМ (или пробивались на перфоленте).

Замечания:

1. Перед набором символов на ЭПМ, указанных в правой стороне таблицы, необходимо нажать клавишу соответствующего регистра;

2. Расшифровка обозначений:

N — номер оператора. Целое неотрицательное число (1÷÷232);

M — некоторое целое неотрицательное число;

$i_{и-б}$ — означает, что нажимается клавиша индекс-буквы;

$i_{и}$ — означает, что нажимается клавиша целочисленного переменного;

КПП — обозначает код программы-процедуры (см. в главе 2 оператор «*программа*»);

3. В конце операторов, у которых отсутствует «*вк*», клавиша «*вк*» не нажимается;

4. В операторах «*допустим*», «*вычислим*», «*печатаем*», «*вставим*» можно писать список формул и выражений. Количество выражений и формул в списке не ограничено каким-то определенным числом и определяется лишь емкостью ОЗУ. Между формулами списка ставится __ , а в операторе «*вычислим*» — —.

В таблице 6 приведены образцы записи и перфорации элементов программ на языке АП.

Исправление ошибок при вводе программы в ЭВМ

Транслятор с языка АП принимает и расшифровывает программу пооператорно. Каждый введенный оператор анализируется различными блоками транслятора в зависимости от символов, содержания операторов и т. д. Если символы не поддаются анализу, т. е. их нельзя обработать как правильные, то ЭВМ

Сводная таблица записи операторов режима пЛП и «шапки» *Таблица* 6

Запись	Печать
ап	*ап*
$i = 10\ a$	$i_{н.б}$ 10—a ↑
$к = 3\ в\ с$	$к_{н.б}$ 3—$в\ с$ ↑
$l = 2\ \ j = 3\ б\ д$	$l_{н.б}$ 2—$j_{н.б}$ 3—$бд$ ↑
N *допустим* $z = 3{,}5\ e = u\ л = (1024)\ n = \lvert л \rvert$	N—*до* $z = 3{,}5$—$e = u$—$л = (1024)$—$n = \lvert л \rvert$—
$m = -\lvert m \rvert\ (201) = 5{,}1\ (202) = z\ (203) = (1025)$	$m = -\lvert m \rvert$—$(201) = 5{,}1$—$(202) = z$—$(203) = (1025)$—
$(127) = \lvert e \rvert\ (2046) = -\lvert u \rvert\ a_i = 36{,}1\ c_j = z$	$(127) = \lvert e \rvert$—$(2046) = -\lvert u \rvert$—$a_i = 36{,}1$—$c_j = z$—
$в_к = (100)\ к = 2\ j = i\ б_{ji} = 7{,}15\ б_{iк} = x$	$в_к = (100)$—$к_{н.} = 2$—$j_{н.} = i_{н.}$—$б_{ji} = 7{,}15$—$б_{iк} = x$—
$д_{jк} = (250)\ б_{1,2} = y\ в_3 = z\ x = i\ j = y$	$д_{jк} = (250)$—$б_{1,2} = y$—$в_3 = z$—$x = i_{н.}$—$j_{н.} = y$ ↑
N *вычислим* $z = f(x,\ y,\ a_i,\ б_{кj}, \ldots)$	N— *вы* $z = f(x,\ y,\ a_i,\ б_{kl}, \ldots)$— —
$a_i = f(x,\ y,\ a_j,\ б_{кj}, \ldots)$	$a_j = f(x,\ y,\ a_i,\ б_{кj}, \ldots)$— —
$д_{jк} = f(x,\ y,\ a_j,\ б_{кj}, \ldots)$	$д_{jк} = f(x,\ y,\ a_i,\ б_{кj}, \ldots)$ ↑
N *вычислим* y	N— *вы* y ↑
N *вставим* $x = x \pm 0{,}5\ к = к \pm 4$	N—*вс* $x = x \pm 0{,}5$—$к = к \pm 4$ ↑
N *введем* $a\ в_i\ x_{ij}\ б_{1,2}$	N—*вв* a—$в_i$—x_{ij}—$б_{1,2}$ ↑
N *печатаем с 9 знаками* $a_к$ *ы* $j\ в_{ij}\ c_1$	N—*пс* 9 $a_к$—*ы*—j—$в_{ij}$—c_1 ↑
N *программа* КПП	N— *пр* КПП
N *программа* М	N— *пр* М ↑
N *если* $i - 10 = 0$ *идти к* М	N—*ес* $i - 10 = $ М ↑

Запись	Печать
N *если* $f(x, y, a_{1j}, б_{kj}, \ldots) \geqslant 0$ *идти к* M	N—*ес* $f(x, y, a_{1j}, б_{kj}, \ldots) \geqslant$ M↑
N *если* ! *идти к* M	N—*ес*! M↑
N *идти к* M	N—*ид* M↑
N *интервал* M	N—*ин* M↑
N *спросим*	N—*сп*
N *храним* M *y z*	N—*хр* M—*y*—*z*↑
N *храним* M *t*	N—*хр* M—*t*↑
N *начертим* 1 *гр*	N—*на* 1
N *начертим* 2 *гр*	N—*на* 2
N *кончаем*	N—*ко*
N *останов*	N—*ос*
сотрем с N	*со* N↑
исполним N	*ис* N↑
лента повт реш N *оп*	*ле о* N—
лента прод реш N *оп*	*ле р* N—
массив к сост M команд	*ма к*
массив в нач 370	*ма в*

печатает признак ошибки и останавливается. Все предыдущие операторы сохраняются правильными, а в ошибочном операторе надо исправить ошибку.

В случае обнаружения ошибок при вводе программ с электрической пишущей машинки ЭВМ печатает следующие символы:

а) ∞_*;
б) $_*$;
в) ∞ .

При вводе же с перфоленты печатается* и ЭВМ останавливается.

Случай а) говорит о синтаксической ошибке; нужно ввести правильную смысловую информацию, начиная с предыдущего пробела.

Символы* и ∞ печатаются вслед за неправильно введенной формулой.

Печать символа ∞ означает, что формулу нужно ввести заново. Для устранения ошибки надо допечатать ∞. Происходит возврат каретки и перевод строки ЭПМ, после этого нужно заново ввести правильный оператор целиком.

Печать символа* означает, что ЭВМ исключает только последний символ и готова заменить его другим.

Вычислитель имеет возможность исправить некоторые несинтаксические ошибки при вводе информации. Если возврат каретки еще не сделан, то 1) букву можно заменить любым другим символом, напечатав после буквы символ*; 2) символы операций /, $\times$, +, —, можно заменить любым другим символом операции, напечатав символ*; 3) если оператор и его содержимое напечатаны неправильно, то набирается $\infty\infty$ и вводится правильный оператор с той же меткой.

Если возврат каретки уже сделан и мы обнаружили ошибку или машина фиксирует ошибку в только что введенном операторе, то необходимо нажать клавишу «Пуск-1», набрать «*па*» и правильный оператор с той же меткой.

Пример:

N— *вычислим* $y=x/*-a\_\_\_z=1{,}25,*$.

В первом случае знак «/» набрали вместо «—», потом напечатав $_*$ поставили необходимый знак «—»; во втором случае неправильно был набран символ «,» и ЭВМ напечатала $_*$ и не восприняла символ «,». Это — синтаксическая ошибка.

При работе с индексированными переменными может получиться такая ошибка: ЭПМ печатает *ni. Здесь индекс принял или отрицательное значение или значение большее чем 1000. Ошибка выясняется просматриванием значений индексов.

§ 8. РАСПРЕДЕЛЕНИЕ ПАМЯТИ ПРИ РАБОТЕ АП ТРАНСЛЯТОРА

ОЗУ используется следующим образом:

Массив А (32÷55) — для переменных без индексов;

Массив Б (56÷59) — для целочисленных переменных:

яч.	сим.	яч.	сим.	яч.	сим.	яч.	сим.
32	*т*	40	*н*	48	*е*	56	*i*
33	*с*	41	*х*	49	*l*	57	*j*
34	*в*	42	*а*	50	*р*	58	*к*
35	*у*	43	*б*	51	*т*	59	*п*
36	*д*	44	*г*	52	*ш*	60	*ф*
37	*л*	45	*ч*	53	*s*	61	*ц*
38	*м*	46	*и*	54	*t*	62	*щ*
39	*з*	47	*о*	55	*ы*	63	*f*

Если не использовать целочисленные переменные, то в ячейках 56, 57, 58, 59 будут соответственно записываться переменные *ж, э, й, ь,* которые могут быть использованы в программе, составленной на языке АП.

Массив В (69÷68+*r*) — здесь организуются и записаны команды передачи управления результирующим программам соответствующих операторов, где *r* — количество операторов (*r*<232).

Массив Г (300÷300—*p*) — для хранения числовых констант действительного типа, где *p* — число констант (*p*<233).

Массив Д (370÷369+N) — для результирующей программы, где N — количество команд в отранслированной программе.

Массив Е (370+N÷370+N+M) — для хранения значений функции (или 2-х функций), необходимых при построении графика (двух графиков), где M — количество точек, задаваемое оператором *«храним»*.

Массив Ж (1940÷1941—*s*) — для переменных с индексами.

Массив З (10—31, 59—68, 1996, 1997, 2047) — рабочие ячейки транслятора АП и результирующей программы.

Массив И (2012—2041) — для хранения информации о формулах (кода переменной, которой обозначена формула, начального и конечного адресов программы соответствующей формулы).

Приложения

код операции				операция	команда	содержание команды	примечание	
I	II	I	II					
				Арифметические операции				
C	C_2	01	11	сложение	$c[c_2]\ A_1\theta A_2$	$(A_2)+[A_1]_\theta \Rightarrow A_2, a_2$	Для команд типа I результат не анализируется и истинен, если он $-2 \le X \le 2$	Для команд типа II результат не анализируется на переполнение, если его модуль ≤ 1 по модулю, то ПУ по адресу 2047. В 2046 запоминается номер следующей команды.
C_1	C_3	41	51		$c_1[c_3]\ A_1\theta A_2$	$(a_2)+[A_1]_\theta \Rightarrow A_2, a_2$		
B	B_2	02	12	вычитание	$B[B_2]\ A_1\theta A_2$	$(A_2)-[A_1]_\theta \Rightarrow A_2, a_2$		
B_1	B_3	42	52		$B_1[B_3]\ A_1\theta A_2$	$(a_2)-[A_1]_\theta \Rightarrow A_2, a_2$		
У	$У_2$	03	13	умножение	$У[У_2]\ A_1\theta A_2$	$(A_2)\times[A_1]_\theta \Rightarrow A_2, a_2$		
$У_1$	$У_3$	43	53		$У_1[У_3]\ A_1\theta A_2$	$(a_2)\times[A_1]_\theta \Rightarrow A_2, a_2$		
Д	$Д_2$	04	14	деление	$Д[Д_2]\ A_1\theta A_2$	$(A_2):[A_1]_\theta \Rightarrow A_2, a_2$		Для команд типа II: если $\|(A_2)\| \ge \|[A_1]_\theta\|$ то ПУ на 2047
$Д_1$	$Д_3$	44	54		$Д_1[Д_3]\ A_1\theta A_2$	$(a_2):[A_1]_\theta \Rightarrow A_2, a_2$		
$б_4$		26		вычитание модулей	$б_4\ A_1\theta A_2$	$\|(A_2)\|-\|(A_1)_\theta\| \Rightarrow a_2$		
ут		100		умножение	$ут\ A_1\theta A_2$	$(A_2)\times[A_1]_\theta \Rightarrow A_2$	для целых чисел	если модуль рез-та $\ge 2^{35}$, выдача ∞ и останов.
дт		144		деление	$дт\ A_1\theta A_2$	$[(A_2):[A_1]_\theta] \Rightarrow A_2$		при $[A_1]_\theta = 0$. Выдача ∞ и останов. $\{(A_2):[A_1]_\theta\} \Rightarrow 3$
Сп		101		сложение	$сп\ A_1\theta A_2$	$[A_1]_\theta^n+(A_2) \Rightarrow A_2, a_2$	Для чисел с плавающей запятой	Если порядок результата $>б3$, то печатается ∞ и останов.
Вп		102		вычитание	$вп\ A_1\theta A_2$	$[A_1]_\theta^n-(A_2) \Rightarrow A_2, a_2$		
ов		143		обратное вычитание	$ов\ A_1\theta A_2$	$(A_2)-[A_1]_\theta^n \Rightarrow A_2, a_2$		
Уп		103		умножение	$уп\ A_1\theta A_2$	$[A_1]_\theta^n\times(A_2) \Rightarrow A_2, a_2$		Если делитель равен 0, то печатается ∞ и останов.
дп		104		деление	$дп\ A_1\theta A_2$	$[A_1]_\theta^n:(A_2) \Rightarrow A_2, a_2$		
од		121		обратное деление	$од\ A_1\theta A_2$	$(A_2):[A_1]_\theta^n \Rightarrow A_2, a_2$		
сб		145		сложение (без записи)	$сб\ A_1\theta A_2$	$[A_1]_\theta^n+(A_2) \Rightarrow a_2$		
вб		147		вычитание (без записи)	$вб\ A_1\theta A_2$	$[A_1]_\theta^n-(A_2) \Rightarrow a_2$		
уб		151		умножение (без записи)	$уб\ A_1\theta A_2$	$[A_1]_\theta^n\times(A_2) \Rightarrow a_2$		
дб		153		деление (без записи)	$дб\ A_1\theta A_2$	$[A_1]_\theta^n:(A_2) \Rightarrow a_2$		
сс		146		сложение (с сумматором)	$сс\ A_1\theta A_2$	$(a_2)+[A_1]_\theta^n \Rightarrow A_2, a_1$		
вс		150		вычитание (из сумматора)	$вс\ A_1\theta A_2$	$(a_2)-[A_1]_\theta^n \Rightarrow A_2, a_2$		
ус		152		умножение (на сумматор)	$ус\ A_1\theta A_2$	$(a_2)\times[A_1]_\theta^n \Rightarrow A_2, a_2$		
дс		154		деление сумматора	$дс\ A_1\theta A_2$	$(a_2):[A_1]_\theta^n \Rightarrow A_2, a_2$		
Вм		122		вычитание (модулей)	$Вм\ A_1\theta A_2$	$\|(A_2)\|-\|[A_1]_\theta^n\| \Rightarrow A_2, a_2$		

ск	135	сложение	ск $A_1 \theta A_2$	$(A_1)^к + (A_2)^к \Rightarrow A_2^к$	Для комплексных чисел
вк	136	вычитание	вк $A_1 \theta A_2$	$(A_2)^к - (A_1)^к \Rightarrow A_2^к$	
ук	137	умножение	ук $A_1 \theta A_2$	$(A_1)^к \times (A_2)^к \Rightarrow A_2^к$	
дк	140	деление	дк $A_1 \theta A_2$	$(A_1)^к : (A_2)^к \Rightarrow A_2^к$	
сд	112	сложение	сд $A_1 \theta A_2$	$(A_2)^д + [A_1]_\theta^д \Rightarrow A_2^д$	Для длинных чисел
вд	113	вычитание	вд $A_1 \theta A_2$	$(A_2)^д - [A_1]_\theta^д \Rightarrow A_2^д$	
уд	114	умножение	уд $A_1 \theta A_2$	$(A_2)^д \times [A_1]_\theta^д \Rightarrow A_2^д$	
дд	115	деление	дд $A_1 \theta A_2$	$(A_2)^д : [A_1]_\theta^д \Rightarrow A_2^д$	
		ОПЕРАЦИИ ПОСЫЛКИ			
п	10	ПЕРЕДАЧА ЧИСЕЛ	п $A_1 \theta A_2$	$[A_1]_\theta \Rightarrow A_2, a_2$	
$п_1$	50		$п_1$ 0 θA_2	$(a_2) \Rightarrow A_2, a_2$	реком. $\theta = к, п$
оп	155	обратная передача	оп $A_1 \theta A_2$	$-[A_1]_\theta^п \Rightarrow A_2, a_2$	
от	142	отсылка длинн. числа	от $A_1 \theta A_2$	$[A_1] и [A_1+1] \Rightarrow A_2 и A_2+1$	реком. $\theta = н$
		ОПЕРАЦИИ УПРАВЛЕНИЯ			
и	20	ПЕРЕХОД с возвратом	и $A_1 \theta$	$[A_1]_\theta^{14} \Rightarrow СчК$	(a_2) сохран.
$и_1$	60		$и_1$ $A_1 \theta A_2$	$[A_1]_\theta^{14} \Rightarrow СчК$ и "и $(СчК)_п$" $\Rightarrow \Lambda$	
$и_2$	40	безусловный	$и_2$ $A_1 \theta$	$[A_1]_\theta^{14} \Rightarrow СчК$	Вых. из псевдоп., (a_2) сохран.
$и_3$	25	по ключу	$и_3$ $A_1 \theta$	$[A_1]_\theta^{14} \Rightarrow СчК$	При вкл. „КЛЮЧ", (a_2) сохран.
е	22	Формирование ком. адреса	е $A_1 \theta A_2$	$(СчК) + [A_1]_\theta \Rightarrow A_2, a_2, a_3$	реком. $\theta = п, к, н$
$е_1$	62	относительный безусловн. переход	$е_1$ $A_1 \theta A_2$	$(СчК) + [A_1]_\theta \Rightarrow СчК$	
$е_2$	64	переход по набору	$е_2$ $A_1 \theta A_2$	если $(a_2) \oplus [A_1]_\theta = 0$, то $(СчК) + A_2^{14} \Rightarrow СчК$, если $(a_2) \oplus [A_1]_\theta \neq 0$, то $(СчК) + 1 \Rightarrow СчК$	
$е_3$	65	переход по разряду	$е_3$ $A_1 \theta A_2$	если $\{(a_2) \wedge [A_1]_\theta\} \oplus [A_1]_\theta = 0$, то $(СчК) + A_2^{14} \Rightarrow СчК$, иначе $(СчК) + 1 \Rightarrow СчК$	
з	27	Запись β в a_7	з $A_1 \theta A_2$	$[A_1]_\theta \Rightarrow a_7$, "$з_1 (a_7) \wedge$" $\Rightarrow a_2$	реком. $\theta = н, п$.
$з_1$	67	Запись β в a_7	$з_1$ $A_1 \theta A_2$	$[A_1]_\theta \Rightarrow a_7$	
х	00	холостая операция	х 0 θ	$СчК \Rightarrow СчК$	(a_2) сохран.

ВЫЧИСЛЕНИЕ ФУНКЦИЙ

					Действия производятся только над числами с плавающей запятой	
кп	105	$\sqrt{X}$	кп $A_1\theta A_2$	$\sqrt{[A_1]_\theta} \Rightarrow A_2$		При $[A_1]_\theta < 0$ печатается $X<0$ и останов.
ln	107	ln X	ln $A_1\theta A_2$	$ln\,[A_1]_\theta \Rightarrow A_2$		При $[A_1]_\theta \leq 0$ печатается $X \leq 0$ и останов.
lg	116	lg X	lg $A_1\theta A_2$	$lg\,[A_1]_\theta \Rightarrow A_2$		
ex	133	e^X	ex $A_1\theta A_2$	$ex\,[A_1]_\theta \Rightarrow A_2$		Если $[A_1]_\theta > 43$, то печатается ∞ и останов.
sn	111	sin X	sn $A_1\theta A_2$	$sin\,[A_1]_\theta \Rightarrow A_2$		Если порядок $[A_1]_\theta > 27$, то печатается п="порядок" и останов.
cs	126	cos X	cs $A_1\theta A_2$	$cos\,[A_1]_\theta \Rightarrow A_2$		
tg	120	tg X	tg $A_1\theta A_2$	$tg\,[A_1]_\theta \Rightarrow A_2$		Если $[A_1]_\theta = \pm \pi/2$, то печатает. ∞ и останов.
as	123	arcsin X	as $A_1\theta A_2$	$arcsin\,[A_1]_\theta \Rightarrow A_2$		Если $[A_1]_\theta > 1$, то печатает. $X>1$ и останов.
ac	124	arccos X	ac $A_1\theta A_2$	$arccos\,[A_1]_\theta \Rightarrow A_2$		Если $\lvert[A_1]\rvert > 1$, то печатает. $X>1$ и останов.
at	125	arctg X	at $A_1\theta A_2$	$arctg\,[A_1]_\theta \Rightarrow A_2$		
sh	156	SH X	sh $A_1\theta A_2$	$sh\,[A_1]_\theta \Rightarrow A_2$		
ch	157	CH X	ch $A_1\theta A_2$	$ch\,[A_1]_\theta \Rightarrow A_2$		
th	160	tH X	th $A_1\theta A_2$	$th\,[A_1]_\theta \Rightarrow A_2$		
бj	162	jP(x)	бj $A_1\theta A_2$	$j_{[A_1]_\theta}^{n}((A_2)) \Rightarrow A_2$		
бy	163	yp(x)	бy $A_1\theta A_2$	$y_{[A_1]_\theta}^{n}((A_2)) \Rightarrow A_2$		
Га	166	Г(x)	Га $A_1\theta A_2$	$\Gamma([A_1]_\theta) \Rightarrow A_2$		
mn	167	нахожд. наим. число (по модулю)	mn $A_1 K A_2$	экстремальное значение $m \Rightarrow a2$; экстремальное значение по модулю $n \Rightarrow 9$	Числа расположены в адресах A_1, A_1+1, A_1+2, ..., A_2	
mx	170	нахожд. наиб. число (по модулю)	mx $A_1 K A_2$			

ЛОГИЧЕСКИЕ ОПЕРАЦИИ

l	15	логическое сложение	l $A_1\theta A_2$	$(A_2) \vee [A_1]_\theta \Rightarrow a_2$	Операции выполняются над всеми 36-ю разрядами
l_1	55		l_1 $A_1\theta A_2$	$(a_2) \vee [A_1]_\theta \Rightarrow A_2, a_2$	
л	16	логическое умножение	л $A_1\theta A_2$	$(A_2) \wedge [A_1]_\theta \Rightarrow a_2$	
$л_1$	56		$л_1$ $A_1\theta A_2$	$(a_2) \wedge [A_1]_\theta \Rightarrow A_2, a_2$	
м	17	сложение по модулю 2	м $A_1\theta A_2$	$(A_2) \oplus [A_1]_\theta \Rightarrow a_2$	
$м_1$	57		$м_1$ $A_1\theta A_2$	$(a_2) \oplus [A_1]_\theta \Rightarrow A_2, a_2$	

операции над кодами					
a	05	Арифметич. сдвиг вправо	$a\ A_1\theta A_2$	$(\overrightarrow{A_2})$ на $[A_1]_\theta^* \Rightarrow A_2, a_2$	$\theta \neq Л$
a_1	45		$a_1\ A_1\theta A_2$	$(\overrightarrow{a_2})$ на $[A_1]_\theta^* \Rightarrow A_2, a_2$	$\theta \neq Л$
$б$	06	Логический сдвиг влево	$б\ A_1\theta A_2$	$(\overleftarrow{A_2})$ на $[A_1]_\theta^* \Rightarrow A_2 a_2$	$\theta \neq Л$
$б_1$	46		$б_1\ A_1\theta A_2$	$(\overleftarrow{a_2})$ на $[A_1]_\theta^* \Rightarrow A_2, a_2$	
$г$	23	Правый арифметич. сдвиг длинн. числа	$г\ A_1\theta A_2$	$(\overrightarrow{A_2})^q$ на $[A_1]_\theta^* \Rightarrow A_2^q$	$\theta \neq Л$
$г_1$	63	Левый логический сдвиг длинн. числа	$г_1\ A_1\theta A_2$	$(\overleftarrow{A_2})^q$ на $[A_1]_\theta^* \Rightarrow A_2^q$	
$н$	07	нормализация	$н\ A_1\theta A_2$	$[A_1]_\theta$ норм $\Rightarrow A_2$	
$н_1$	47		$н_1\ 0\theta A_2$	(a_2) норм $\Rightarrow A_2$	
$н_q$	106	Нормализация длинн. числа	$н_q\ A_1\theta A_2$	$[A]_\theta^q$ норм $\Rightarrow A_2$	$\theta = п, к, н$
$qн$	134	Денормализация числа	$qн\ A_1\theta A_2$	$[A_1]_\theta \Rightarrow A_2$ и A_2+1	
служебные операции					
0	21	печать символов	$0\ A_1\theta$	$[A_1]_\theta^{12} \Rightarrow$ печать	(a_2) сохран
0_1	61		$0_1\ 0\theta$	$[(a_2)+2^{11}+2^7]^{12} \Rightarrow$ печ.	
$ч$	24	чтение	$ч\ 0\theta A_2$	$\langle P_2 ВУ\rangle \Rightarrow A_2, a_2$	Реком. $\theta = п$
$пп$	110	печать чисел: с плавающей запятой	$пп\ A_1\theta q$	$[A_1]_\theta \Rightarrow$ печать	q = требуемое кол-во дес. цифр после запятой; $0 \le q \le 81$; при $q \ge 64$ печ. $k-1$ знаков и $14-k$ пробелов; $\theta = Н, К, П$
$пч$	127	дробных	$пч\ A_1\theta q$	$[A_1]_\theta \Rightarrow$ печать	реком. $\theta = Н$
$пq$	117	длинных	$пq\ A_1\theta q$	$[A_1]_\theta^q \Rightarrow$ печать	
$пт$	130	целых	$пт\ A_1\theta$	$[A_1]_\theta^q \Rightarrow$ печать	$[A_1]_\theta$ рассматривается как целое число переводится и печатается $\theta = Н$
$пм$	161	комплексных	$пм\ A_1\theta q$	$[A_1]_\theta^{14}$ и $[A]_\theta^{14}+1 \Rightarrow$ печ	печатается $a+ва$, где $a=[A_1]_\theta^{14}$, $в=([A_1]_\theta^{14}+1)$ $\theta = Н, К, П$
$пк$	132	печать команд	$пк\ A_1\theta$	$[A_1]_\theta \Rightarrow$ печать	$[A_1]_\theta$ печатается как команда во внеш. коде $\theta = Н$
$пс$	131	печать содержим	$пс\ A_1\theta$	$[A_1]_\theta \Rightarrow$ печать	$[A_1]_\theta$ печатается как 36-ю разрядный набор 1 и 0 $\theta = Н, К, П$
$пи$	141	печать индекса	$пи\ A_1\theta$	$[A_1]_\theta \Rightarrow$ печать	$[A_1]_\theta$ переводится в дес. сист., $0 \le [A_1]_\theta < 1000$
$К$	37	Останов	$к\ A_1\theta$	$[A_1]_\theta \Rightarrow$ см и ост	
$К_1$	77		$К_1\ 0\theta$	$(a_2) \Rightarrow$ см и ост.	Реком. $\theta = п$

Использование рабочих ячеек ОЗУ программами, записанными в ДЗУ

№	Виды операций	Используемые ячейки
1	$\ln x$; $\lg x$; $\sqrt{x}$; $\arcsin x$; $\arccos x$; $\operatorname{arctg} x$; $\operatorname{sh} x$	3÷7
2	e^x; га (x); *сп*; *вп*; *дп*; *уп*; *сб*; *вб*; *уб*; *дб*; *сс*; *вс*; *ус*; *дс*; *вм*; *ов*; *од*; *оп*; *сд*; *вд*; *жд*; *пт*; *от*; *пи*	3÷8
3	$\sin x$; $\cos x$; $\operatorname{tg} x$; *дп*; *пк*; $\min\limits_i \{x_i\}$; $\max\limits_i \{x_i\}$	3÷9
4	$\operatorname{ch} x$; $\operatorname{th} x$	3÷11
5	*пс*	3,4
6	*пм*	1÷12
7	*дт*	3÷5; 8÷10
8	*ут*; *пч*	3÷9; 16; 17
9	*пд*	3÷9; 16; 17; 20; 21
10	*пп*	3÷9; 16; 17; 20÷26
11	*уд*	3÷9; 16; 17; 20÷23; 25
12	*дд*	3÷9; 16; 17; 20÷25
13	*ск*; *вк*	3; 4; 16; 17; 20÷23
14	*ук*	3; 4; 16; 17; 20÷23; 25; 26
15	*дк*	3; 4; 9; 16; 17; 20÷23; 25; 26
16	$I_p(x)$	2÷9; 16; 17; 22; 24; 25
17	$У_p(x)$	2÷9; 16; 17; 22; 24; 25
18	ДИИ (дешифрация исходной информации)	1÷9; 16÷26; 28; 2010; 2011; 1994; 2043; 2044; 2045
19	ВВ (выдача памяти)	2; 3; 4; 2000; 2001
20	ОП (отладочная программа)	2; 1941÷1966; 2041÷2047
21	СС (счетный режим)	2; 16; 17; 22÷26; 28; 1998÷2003
22	ТАП (транслятор автоматического программирования)	10÷369; 1996; 1997; 2047
23	ИЛ (вычисление определенного интеграла)	14; 15; 31; 1971÷1983
24	РК, ДУ (решение системы дифференциальных уравнений)	5; 8; 9; 13; 35; 54; 1971÷1991; 1940÷(1941—$4\cdot n$) при РК 1940÷(1941—$7\cdot n$) при ДУ
25	ВО (вычисление определителя)	15; 31; 32; 1971÷1986; n^2+2n
26	МО (обращение матрицы методом Жордана)	10; 31; 32; 1971÷1986; 1992÷2002; $3n^2+3n$
27	ОМ (обращение матрицы)	15; 31; 1971÷1986; $3n^2$
28	УМ (умножение матриц)	15; 31; 1971÷1986; $mn+np+mp$
29	НК (решение системы уравнений методом наименьших квадратов)	14; 15; 31; 32; 1971÷1991; $(n+1)\times(m+1)$
30	СУ (решение системы уравнений методом главных элементов)	15; 31; 32; 1971÷1991; n^2+3n
31	СМ (симплекс-метод)	10÷43; $45+m(n+4)+3(n+1)$
32	ИП (интерполяция)	14; 15; 31; 1971÷1986; $4(n+1)$
33	Вычисление полиномов Лежандра	2; 7; 9; 16÷23

Коды знаков и служебных клавиш, записанные в ДЗУ

Адрес восьм.	Адрес десятич.	Символ	α_{13}	α_{12}	α_{11}	α_{10}	α_9	α_8	α_7	α_6	α_5	α_4	α_3	α_2	α_1
ЧИСЛА															
4200	2176	0	0	1	0	1	0	0	0	0	0	0	0	0	0
4201	2177	1	0	1	0	1	0	0	0	0	0	0	0	0	1
4202	2178	2	0	1	0	1	0	0	0	0	0	0	0	1	0
4203	2179	3	0	1	0	1	0	0	0	0	0	0	0	1	1
4204	2180	4	0	1	0	1	0	0	0	0	0	0	1	0	0
4205	2181	5	0	1	0	1	0	0	0	0	0	0	1	0	1
4206	2182	6	0	1	0	1	0	0	0	0	0	0	1	1	0
4207	2183	7	0	1	0	1	0	0	0	0	0	0	1	1	1
4210	2184	8	0	1	0	1	0	0	0	0	0	1	0	0	0
4211	2185	9	0	1	0	1	0	0	0	0	0	1	0	0	1
СТЕПЕНИ															
4212	2186	0	0	1	0	1	1	0	0	0	0	0	0	0	0
4213	2187	1	0	1	0	1	1	0	0	0	0	0	0	0	1
4214	2188	2	0	1	0	1	1	0	0	0	0	0	0	1	0
4215	2189	3	0	1	0	1	1	0	0	0	0	0	0	1	1
4216	2190	4	0	1	0	1	1	0	0	0	0	0	1	0	0
4217	2191	5	0	1	0	1	1	0	0	0	0	0	1	0	1
4220	2192	6	0	1	0	1	1	0	0	0	0	0	1	1	0
4221	2193	7	0	1	0	1	1	0	0	0	0	0	1	1	1
4222	2194	8	0	1	0	1	1	0	0	0	0	1	0	0	0
4223	2195	9	0	1	0	1	1	0	0	0	0	1	0	0	1
4224	2196	,	0	1	0	1	1	0	0	0	0	1	0	1	0
4225	2197	/	0	1	0	1	1	0	0	0	0	1	0	1	1

Адрес восьм.	Адрес десятич.	Символ	α_{13}	α_{12}	α_{11}	α_{10}	α_9	α_8	α_7	α_6	α_5	α_4	α_3	α_2	α_1
		И Н Д Е К С Ы													
4226	2198	*i*	0	1	0	1	1	0	0	1	1	1	1	0	0
4227	2199	*j*	0	1	0	1	1	0	0	0	0	1	1	0	0
4230	2200	*к*	0	1	0	1	1	0	0	1	0	0	1	1	0
4231	2201	0	0	1	0	1	1	0	0	1	1	1	1	1	0
4232	2202	1	0	1	0	1	1	0	0	1	1	0	1	1	1
4233	2203	2	0	1	0	1	1	0	0	1	1	0	0	0	1
4234	2204	3	0	1	0	1	1	0	0	1	0	1	1	0	0
4235	2205	4	0	1	0	1	1	0	0	1	0	1	0	0	0
4236	2206	5	0	1	0	1	1	0	0	1	1	0	0	1	0
4237	2207	6	0	1	0	1	1	0	0	1	1	1	0	1	1
4240	2208	7	0	1	0	1	1	1	0	1	0	0	0	0	1
4241	2209	8	0	1	0	1	1	0	0	1	1	1	1	0	1
4242	2210	9	0	1	0	1	1	0	0	0	1	1	0	0	1
		Б У К В Ы													
4243	2211	*п*	0	1	0	1	1	0	0	1	0	0	1	0	0
4244	2212	*с*	0	1	0	1	0	0	0	1	1	0	0	1	0
4245	2213	*в*	0	1	0	1	0	0	0	0	1	1	0	0	0
4246	2214	*у*	0	1	0	1	0	0	0	1	1	0	0	0	0
4247	2215	*д*	0	1	0	1	0	0	0	0	0	1	1	0	0
4250	2216	*л*	0	1	0	1	0	0	0	1	1	0	0	0	1
4251	2217	*м*	0	1	0	1	0	0	0	1	1	0	1	1	0
4252	2218	*г*	0	1	0	1	1	0	0	1	1	0	1	0	1
4253	2219	*н*	0	1	0	1	0	0	0	1	0	0	1	0	0

Адрес восьм.	Адрес десятич.	Символ	α_{13}	α_{12}	α_{11}	α_{10}	α_9	α_8	α_7	α_6	α_5	α_4	α_3	α_2	α_1
4254	2220	*x*	0	1	0	1	0	0	0	1	0	1	0	0	1
4255	2221	*a*	0	1	0	1	0	0	0	0	0	1	0	1	0
4256	2222	*б*	0	1	0	1	0	0	0	0	0	1	0	1	1
4257	2223	*г*	0	1	0	1	0	0	0	1	1	1	1	0	0
4260	2224	*ч*	0	1	0	1	0	0	0	1	1	0	1	0	0
4261	2225	*и*	0	1	0	1	0	0	0	1	1	1	0	1	1
4262	2226	*o*	0	1	0	1	0	0	0	1	0	1	0	1	1
4263	2227	*e*	0	1	0	1	0	0	0	1	0	0	1	1	0
4264	2228	*l*	0	1	0	1	1	0	0	1	0	0	1	1	1
4265	2229	*p*	0	1	0	1	0	0	0	1	0	1	0	0	0
4266	2230	*m*	0	1	0	1	0	0	0	1	1	1	0	1	0
4267	2231	*ш*	0	1	0	1	0	0	0	1	0	0	0	1	0
4270	2232	*s*	0	1	0	1	1	0	0	1	1	1	0	0	0
4271	2233	*t*	0	1	0	1	1	0	0	1	1	1	0	0	1
4272	2234	*ы*	0	1	0	1	0	0	0	1	1	1	1	0	1
4273	2235	*ж*	0	1	0	1	0	0	0	1	1	1	1	1	0
4274	2236	*з*	0	1	0	1	0	0	0	1	1	0	1	1	1
4275	2237	*й*	0	1	0	1	0	0	0	1	0	0	0	0	1
4276	2238	*ь*	0	1	0	1	0	0	0	0	1	1	0	0	1
4277	2239	*ф*	0	1	0	1	0	0	0	1	0	1	1	1	1
4300	2240	*ц*	0	1	0	1	0	0	0	1	1	0	0	1	1
4301	2241	*щ*	0	1	0	1	1	0	0	1	1	0	1	1	0
4302	2242	*f*	0	1	0	1	1	0	0	1	0	1	0	1	1
4303	2243	*я*	0	1	0	1	1	0	0	1	0	1	1	1	0
4304	2244	*ю*	0	1	0	1	1	0	0	1	0	1	1	1	1
4305	2245	*э*	0	1	0	1	1	0	0	1	1	0	0	0	0
4306	2246	*i*	0	1	0	1	1	0	0	1	0	1	1	0	1
4307	2247	*j*	0	1	0	1	1	0	0	1	0	0	0	1	1
4310	2248	*к*	0	1	0	1	0	0	0	1	0	1	1	0	0
4311	2249	*n*	0	1	0	1	0	0	0	1	0	1	1	1	0

Адрес восьм.	Адрес десятич.	Символ	α_{13}	α_{12}	α_{11}	α_{10}	α_9	α_8	α_7	α_6	α_5	α_4	α_3	α_2	α_1
		УСЛОВИЯ													
4312	2250	$=$	0	1	0	1	0	0	0	1	0	0	0	0	0
4313	2251	$\neq$	0	1	0	1	1	0	0	1	1	0	0	1	1
4314	2252	$>$	0	1	0	1	1	0	0	1	1	1	0	1	0
4315	2253	$\geqslant$	0	1	0	1	1	0	0	1	1	0	1	0	0
4316	2254	$<$	0	1	0	1	1	0	0	1	0	0	0	1	0
4317	2255	$\leqslant$	0	1	0	1	1	0	0	1	0	0	0	0	0
4320	2256	!	0	1	0	1	1	0	0	1	0	1	0	1	0
4321	2257	∞	0	1	0	1	1	0	0	1	0	0	1	0	1
		ОПЕРАЦИИ													
4322	2258	$+$	0	1	0	1	0	0	0	1	0	0	0	1	1
4323	2259	$-$	0	1	0	1	0	0	0	1	1	1	0	0	0
4324	2260	$\times$	0	1	0	1	0	0	0	1	1	1	0	0	1
4325	2261	/	0	1	0	1	0	0	0	1	0	0	1	1	1
4326	2262	$\surd$	0	1	0	1	0	0	0	1	1	0	1	0	1
		СИМВОЛЫ													
4327	2263	(	0	1	0	1	0	0	0	1	0	1	0	1	0
4330	2264	)	0	1	0	1	0	0	0	1	0	0	1	0	1
4331	2265	,	0	1	0	1	0	0	0	1	0	1	1	0	1
4332	2266	*	0	1	0	1	1	0	0	0	1	1	0	0	0
4333	2267	I	0	1	0	1	1	0	0	1	0	1	0	0	1
		СЛУЖЕБНЫЕ КОДЫ													
4334	2268	*пер. стр.*	0	1	0	1	0	0	0	0	1	0	0	0	1
4335	2269	*возвр. кар.*	0	1	0	1	0	0	0	0	1	0	0	0	0
4336	2270	*проб.*	0	1	0	1	0	0	0	0	1	0	1	1	1
4337	2271	ч. л.	0	1	0	1	0	0	0	0	1	0	0	1	1
4340	2272	в. р.	0	1	0	1	1	0	0	0	1	0	1	0	0
4341	2273	н. р.	0	1	0	1	0	0	0	0	1	0	1	1	0
4342	2274	*воз. кар.*	0	1	0	1	0	0	0	0	1	0	0	0	0
4343	2275	кр. л.	0	1	0	1	0	0	0	0	1	0	0	1	0

Размещение в ДЗУ псевдоопераций и СП

0.	*ут*	4192—4232	31.	сн x	7480—7509
1.	$\sqrt{}$	4240—4286	32.	th x	7512—7557
2.	*нд*	4288—4325	33.	*пм*	7896—7911
3.	ln x	4328—4390	34.	$j_{\text{р}}(x)$	9600—9847
		5025—5029			10139—10199—*с*
		4108—4117—*с*	35.	$y_{\text{р}}(x)$	9848—10138
4.	*пп*	4392—4599			10200—10206
		4144—4187—*с*			10211—10214
5.	sin x	4600—4686			10139—10199—*с*
		4119—4131—*с*	36.	$\Gamma(x)$	7912—7958
		4104—*с*			10147—10158—*с*
6.	*сд*	4688—4713	37.	*тп, тх*	3888—3926
7.	*вд*	4720—4753	38.	ДИИ	2368—3493
8.	*уд*	4760—4838	39.	ВВ	3600—3885
9.	*дд*	4840—5018	40.	ОП	14336—16321
10.	lg x	5024—5087	41.	ИЛ	7562—7643
		4331—4332	42.	РК и ДУ	7648—7879
		4108—4118—*с*			15656—15840
11.	*пд*	5088—5142			8184—8185
		4233—4235	43.	УМ	6496—6595
12.	tg x	5144—5257	44.	ОМ	6598—6599
		4119—4131—*с*			6321—6483
		4104—*с*			8072—8078
13.	arcsin x	5264—5351	45.	МО	15856—15970
14.	arccos x	5352—5443	46.	ВО	6596—6597
		4132—4143—*с*			6144—6304
15.	arctg x	5448—5541	47.	НК	7510—7511
16.	cos x	5544—5618			6624—6633
		4119—4131—*с*			6642—6653
		4104—*с*			6655—6737
17.	*пч*	5624—5658	48.	СУ	6961—6962
18.	*пт*	5664—5686			6640—6654
19.	*пс*	5696—5706			6738—6960
20.	*пк*	3510—3599			6490—6492
21.	exp (x)	5712—5783			7473—7478
		4097—4107—*с*			7467—7472
22.	*дн*	5784—5833			6921—6924
23.	*ск*	5840—5859			6305—6318
24.	*вк*	5864—5883	49.	СМ	7348—7349
25.	*ук*	5888—5913			6968—7347
26.	*дк*	5920—5957			6404—6495
27.	*пи*	5960—6008			394
28.	*от*	6016—6027	50.	ИП	7558—7559
29.	*дт*	6032—6094			7352—7466
30.	sh x	6096—6125	51.	Полиномы Лежандра	7960—8059
		6126—6130—*с*			

Примечание: *с* — означает ячейки для констант.

ЛИТЕРАТУРА

1. *Жарков Г. М., Рыжков А. Б.* Программирование для цифровой вычислительной машины «Наири», М., изд-во МГУ, 1973.

2. *Жоголев Е. А., Трифонов Н. П.* Курс программирования. М., «Наука», 1971.

3. *Лавров С. С.* Введение в программирование, М., «Наука», 1973.

4. *Прохоров В. И., Погорелко И. А., Яковлев В. А.* Основы программирования для электронных цифровых вычислительных машин, М., «Высшая школа», 1967.

5. *Синельник А. К., Сыромятникова Г. В., Сафонов С. Ф.* Справочное пособие по программированию задач строительной механики на ЭЦВМ «Наири», Куйбышев, 1971.

6. *Пыринов Б. В., Красильников В. В.* Программирование для электронной вычислительной машины «Наири», ч. I (учебное пособие). Новосибирский институт инженеров железнодорожного транспорта, Новосибирск, 1969.

7. *Феоктистов В. П.* Решение инженерных задач на универсальной вычислительной машине «Наири», Изд. 2-е, М., 1968.

8. *Ицкович И. А.* Программирование для ЭВМ «Наири», М., Статистика, 1975.

ОГЛАВЛЕНИЕ

Автоматическое программирование

Юрий Николаевич Малиев, Михаил Васильевич Кудрявцев

ПРОГРАММИРОВАНИЕ НА ЭЦВМ «НАИРИ-К»

Учебное пособие по курсу «Основы программирования»

Редактор *И. С. Колышева*

Техн. редактор *Т. В. Телепегина*

Корректор *И. М. Чулкова*

Подписано в печать 6/VIII 1975 г. ЕО01358. Объем 8,5 печ. л. Тир. 1000 экз. Формат бумаги $60\times90^{1}/_{16}$. Цена 70 коп. Куйбышевский государственный университет, г. Куйбышев, ул. Потапова, 64/163. Заказ № 5816.

Типография изд-ва «Волжская коммуна», г. Куйбышев, пр. К. Маркса, 201.