В. Ф. АНИКЕЕНКО
Б. М. КИСЕЛЕВ
В. И. УБИЙКОНЬ

Программирование на микроЭВМ

В. Ф. АНИКЕЕНКО
Б. М. КИСЕЛЕВ
В. И. УБИЙКОНЬ

Программирование на микроЭВМ

СПРАВОЧНОЕ ПОСОБИЕ

Минск "Вышэйшая школа" 1987

ББК 32.973-01я2
А67
УДК 681.3-181.48.06 (035.5)

Рецензенты: *Л.М. Саликов*, доктор технических наук, профессор;
А.Ф. Терпугов, доктор физико-математических наук, профессор

Аникеенко В.Ф. и др.

А67 Программирование на микроЭВМ: Справ. пособие /В.Ф. Аникеенко, Б.М. Киселев, В.И. Убийконь. – Мн.: Выш. шк., 1987. – 190 с.: ил.

Приводятся подробные сведения, необходимые для программирования на микроЭВМ "Электроника Д3-28" на машинно-ориентированном языке и алгоритмическом языке БЭЙСИК. Описывается сопряжение указанной ЭВМ с внешними устройствами пользователей. Даются тексты прикладных БЭЙСИК-программ для решения наиболее часто встречающихся инженерных и научных задач. Содержатся основные сведения о программировании на микроЭВМ ДВК-1 и "Искра 226".

Для студентов вузов, инженеров и других пользователей микроЭВМ. Может быть полезным учащимся средних специальных учебных заведений, ПТУ и старших классов средней школы.

А $\frac{2405000000-100}{М304(03)-87}$ 43–87

ББК 32.973-01я2

ПРЕДИСЛОВИЕ

В настоящее время широкое распространение получили микроЭВМ. Они проникают практически во все области человеческой деятельности. Малогабаритные, дешевые и простые в эксплуатации, микроЭВМ по своим техническим возможностям нередко не уступают ЭВМ более высокого класса. Они используются для автоматизации проектирования, создания информационно-справочных систем, в управлении технологическими процессами и т.д. В то же время микроЭВМ обладают достаточно широкими вычислительными возможностями для применения их в качестве ЭВМ индивидуального пользования и решения научных и инженерно-технических задач, легко сопрягаются с различной аппаратурой и большими ЭВМ.

Для программирования на микроЭВМ созданы алгоритмические языки ФОРТРАН, БЭЙСИК, ПАСКАЛЬ, ФОКАЛ и др. Большую популярность, особенно у начинающих программистов, получил алгоритмический язык БЭЙСИК, отличающийся простотой программирования и удобством работы в диалоговом режиме.

Быстрое расширение областей применения микроЭВМ открыло доступ к ним многим пользователям, в том числе и не являющимся профессиональными программистами. В связи с этим большое значение придается изданию соответствующей учебной и справочной литературы.

Данное справочное пособие представляет собой обобщение опыта практической работы авторов по обучению студентов, школьников и учителей средних общеобразовательных школ программированию на микроЭВМ. Оно предназначено для пользователей микроЭВМ "Электроника Д3-28", ДВК-1, "Искра 226" и позволяет сравнить возможности для программирования на всех этих микроЭВМ.

Основная часть пособия посвящена программированию на микроЭВМ "Электроника Д3-28". Это вызвано тем, что в настоящее время данная вычислительная машина получила широкое распространение, но специальная литература о работе с ней не издавалась. Эта ЭВМ относится к классу управляющих, относительно просто сопрягается с нестандартным оборудованием и широко применяется для автоматизации научных исследований и управления технологическими процессами. В то же время функциональная завершенность микроЭВМ "Электроника Д3-28", наличие достаточно широкого набора машинных команд, большая точность вычислений и реализация на ней алгоритмических языков высокого уровня привлекли внимание многочисленных пользователей для решения задач средней сложности.

В пособии приводятся сведения, необходимые для составления и отладки программ на машинно-ориентированном языке микроЭВМ "Электроника Д3-28" и на алгоритмическом языке БЭЙСИК. Отдельный параграф посвящен подключению к данной микроЭВМ внешних устройств пользователя.

Программирование на микроЭВМ ДВК-1 и "Искра 226" изложено кратко. Приведен основной справочный материал, необходимый даже опытным программистам. Более подробные сведения о работе с этими микроЭВМ читатель может найти в имеющейся литературе.

Начинающим программистам рекомендуем начать изучение книги с гл. 2. Наиболее элементарные сведения, например что такое двоичная система счисления и т. п., приведены в краткой форме, так как они имеются практически в любом учебном пособии по программированию.

Изложение материала иллюстрируется примерами с решениями, справочными таблицами, текстами программ.

В приложениях содержатся: таблицы команд машинно-ориентированного языка микроЭВМ "Электроника Д3-28" в алфавитном порядке и в порядке возрастания значений их кодов; таблицы встроенных математических функций (для этой же микроЭВМ) и кодов символов КОИ-7 (набор 2); формальное описание языка БЭЙСИК (вариант 3А) и таблица его операторов; перечень сообщений об ошибках при выполнении БЭЙСИК-программ на микроЭВМ "Электроника Д3-28", ДВК-1 и "Искра 226". Кроме того, приведены тексты программ, обслуживающих устройство сопряжения с объектами (принципиальная схема которого рассмотрена в этом пособии), проверки магнитных лент, а также ряд прикладных БЭЙСИК-программ с результатами счета на тестовых задачах для решения часто встречающихся математических задач: безусловной оптимизации, решения систем линейных и обыкновенных дифференциальных уравнений, обращения матриц, вычисления интегралов и т. д.

Изложенный материал не исчерпывает всех вопросов, связанных с работой на микроЭВМ "Электроника Д3-28", ДВК-1 и "Искра 226", в связи с ограниченным объемом книги. Вместе с тем необходимо отметить, что происходит непрерывный процесс совершенствования и развития технических и программных средств указанных микроЭВМ. Так, например, выпускаемые в настоящее время микроЭВМ "Электроника Д3-28" имеют объем ОЗУ 128 К, на них реализованы транслятор алгоритмического языка ФОРТРАН и расширенная версия БЭЙСИК-интерпретатора. Семейство микроЭВМ ДВК пополнилось моделями ДВК-2, ДВК-3, ДВК-4 с развитым набором внешних устройств. Для этих вычислительных машин разработана операционная система ОС ДВК с алгоритмическими языками БЭЙСИК, ФОРТРАН, ПАСКАЛЬ. Но, несмотря на это, авторы надеются, что данное справочное пособие

будет полезно пользователям, работающим и на этих микроЭВМ, так как приведенный материал является базовым.

Авторы выражают благодарность рецензентам: доктору технических наук профессору Л.М. Саликову и доктору физико-математических наук профессору А.Ф. Терпугову – за ценные советы и замечания, способствовавшие улучшению книги, а также доктору физико-математических наук профессору А.А.Кураеву, доктору технических наук профессору А.Н.Останину, сотрудникам кафедры вычислительной техники Гомельского политехнического института (заведующий кафедрой – кандидат технических наук В.А.Шлотгауэр), В.К.Шуклину и С.Н.Лукашевичу – за помощь в работе при подготовке книги к изданию.

Все замечания и пожелания просьба присылать по адресу: 220048, Минск, проспект Машерова, 11, издательство "Вышэйшая школа".

Авторы

СПИСОК СОКРАЩЕНИЙ

АЛУ – арифметическо-логическое устройство
ВВ – регистр входного байта "ВВОД"
ВК – клавиша "ВОЗВРАТ КАРЕТКИ"
ВЫВ – регистр выходного байта "ВЫВОД"
ГМД – гибкий магнитный диск (дискета)
Д3-28 – микроЭВМ "Электроника Д3-28"
ИИ – информационный импульс магнитной ленты
КП – контрольная сумма программы
МД – магнитный диск
МЛ – магнитная лента
НГМД – накопитель на гибких магнитных дисках
НМЛ – накопитель на магнитной ленте
ОЗУ – оперативное запоминающее устройство
ОК – область каталога
ОП – индикатор "ОШИБКА ПРОГРАММНАЯ"
ОПП – область программы пользователя
ПЗУ – постоянное запоминающее устройство
ПЛ – перфолента
ПМ – пишущая машинка
ПС – клавиша "ПЕРЕВОД СТРОКИ"
ПУ – периферийное устройство
РОН – регистр общего назначения
СИ – синхроимпульс магнитной ленты
СИМ – синхроимпульс машинный
СИП – синхроимпульс периферийного устройства
СК – счетчик команд
СПП – система подготовки программ
УВВ – устройство ввода/вывода
УК – указатель каталога
УПВ – код уровня внешнего прерывания
УПР – регистр выходного байта "УПРАВЛЕНИЕ"
УСО – устройство связи с объектами
BD – регистр базового адреса данных
BP – регистр базового адреса программы
PC – программный счетчик
SP – указатель стека

1. ПРОГРАММИРОВАНИЕ НА МАШИННО-ОРИЕНТИРОВАННОМ ЯЗЫКЕ МИКРОЭВМ "ЭЛЕКТРОНИКА Д3-28"

1.1. УСЛОВНЫЕ ОБОЗНАЧЕНИЯ И МНЕМОНИКА КОМАНД

Приведенная в данном справочном пособии система команд соответствует системе команд микроЭВМ "Электроника Д3-28" с объемом ОЗУ 32 К (1 К=1024 байтов).

Команды машинно-ориентированного языка Д3-28 кодируются двумя или четырьмя шестнадцатеричными цифрами. Для облегчения восприятия и запоминания команд наряду с шестнадцатеричным кодом команд используются их мнемокоды в ассемблированном виде. *Мнемокод* представляет собой сокращение английских слов, обозначающих основные действия, выполняемые командой.

Перечень мнемокодов с необходимыми пояснениями приводится ниже.

ABGE (addition one and BGE) – прибавление единицы и ветвление, если больше или равно;
ADD (addition) – сложение;
AND (and) – логическое умножение (конъюнкция);
ANS (analis) – анализ;
ATOI (alteration to integer) – преобразование в целочисленное;
BBIC (branch if bit is clear) – ветвление, если бит "очищен" (равен нулю);
BBIS (branch if bit is set) – ветвление, если бит установлен (равен единице);
BEQ (branch if equal) – ветвление, если равно;
BEV (branch if even) – ветвление, если четно;
BGE (branch if greater or equal) – ветвление, если больше или равно;
BHIS (branch if higher or same) – ветвление, если больше или тождественно;
BKEY (branch if key) – ветвление, если клавиша;
BLT (branch if less than) – ветвление, если меньше чем;
BMER (branch if machine error) – ветвление, если машинная ошибка;
BMI (branch if minus) – ветвление, если минус;
BNE (branch if not equal) – ветвление, если не равно;
BPER (branch if programming error) – ветвление, если программная ошибка;
BPL (branch if plus) – ветвление, если плюс;
BR (branch) – ветвление безусловное;
BSA (branch if same) – ветвление, если тождественно;
CAP (cartesian in polar) – декартовы в полярные (преобразование координат);
CLR (clear) – очистка (обнуление);
CMD (comande) – команда;
COM (complement) – инвертирование;
DEG (degree) – градус;
DIG (digital) – цифра (десятичная);
DIV (divide) – деление;
E (exponent) – экспонента (показатель степени);
END (end) – конец;
GO (go) – пуск;
INP (input) – ввод;
INT (integer) – целое;
INV (invert) – обратная величина;
JMM (jump to mark) – передача управления метке;

LOAD (load) – загрузка;
MARK (mark) – метка;
MOV (move) – пересылка;
MUL (multiply) – умножение;
NEG (negative) – отрицание;
OR (or) – логическое сложение (дизъюнкция);
OUT (output) – вывод;
POC (polar in cartesian) – полярные в декартовы (преобразование координат);
POINT (point) – точка;
PRINT (print) – вывод на печать;
QRT (quadrate) – квадрат;
RES (residue) – остаток;
RTI (return from interrupt) – возврат из прерывания;
RTII (RTI imaginary) – псевдовозврат из прерывания;
RTS (return from subroutine) – возврат из подпрограммы;
RTSI (RTS imaginary) – псевдовозврат из подпрограммы;
SAVE (save) – запись;
SOB (subtract one and branch) – вычитание единицы и ветвление;
SQR (square) – квадратный корень;
SUB (subtract) – вычитание;
SWA (swap) – обмен;
VER (verify) – контроль;
WAIT (wait) – ожидание;
XOR (exlusive OR) – исключающее ИЛИ.

При программировании на Д3-28 пользователю предоставлена возможность доступа к любому из 32 768 байтов ОЗУ, называемых *ячейками памяти.* Адресация ячеек начинается с нуля. Адрес отдельного байта может обозначаться или десятичным пятизначным числом, или равным ему шестнадцатеричным четырехразрядным числом. Так, например, в шестнадцатеричном представлении текущий адрес программы хранится в программном счетчике PC, располо-

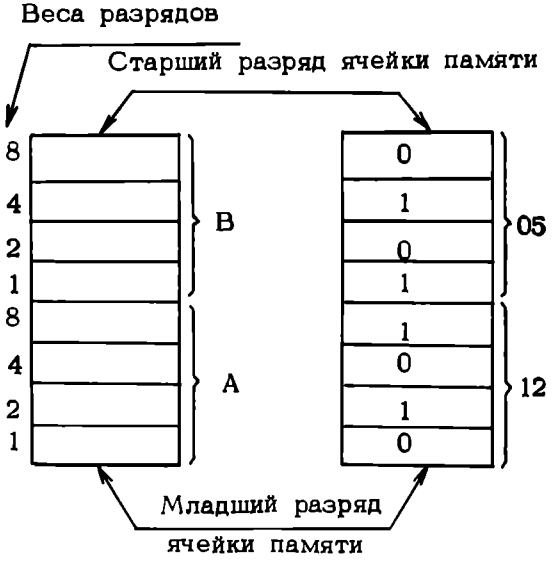


Рис. 1.1

женном в служебной зоне ОЗУ в регистре R15. В режиме работы "В" этот адрес индицируется в регистре Y индикатора Д3-28 в шестнадцатеричном виде. Для представления одной шестнадцатеричной цифры здесь и далее используются две десятичные (00, 01, 02, ..., 09), а числа A, B, C, D, E, F заменены на 10, 11, 12, 13, 14, 15. Для удобства чтения шестнадцатеричных чисел шестнадцатеричные цифры разделены точкой, например 03.12.05.15. Последняя запись равнозначна общепринятой: 3C5F . В дальнейшем адреса ячеек записываются в шестнадцатеричном виде, а рядом в скобках повторяются их десятичные значения. Шестнадцатеричным представлением адреса удобно пользоваться при его чтении или записи в регистры служебной зоны.

Десятичное представление адреса обычно применяется при установке программного счетчика на нужный адрес с клавиатуры Д3-28. При этом используются или клавиша [НШ], или команда 12 13 (JMP @X).

Содержимое одной ячейки памяти кодируется двумя шестнадцатеричными цифрами, условно обозначаемыми В А. Одна шестнадцатеричная цифра кодируется четырьмя двоичными разрядами байта, называемыми *тетрадой.* Расположение разрядов и размещение в ячейке ОЗУ команды 05 12 (END) показаны на рис. 1.1. Здесь цифрами 1, 2, 4, 8 указаны веса разрядов тетрад, с учетом которых шестнадцатеричное значение, например тетрады А, можно представить следующим образом:

$$(A) = 8p_4 + 4p_3 + 2p_2 + 1p_1 ,$$

где p_1 , p_2 , p_3 , p_4 – двоичное значение разрядов тетрад, причем p_1 – младший, а p_4 – старший разряды.

1.2. ОРГАНИЗАЦИЯ ПАМЯТИ Д3-28

В ОЗУ Д3-28 необходимо различать две зоны: *рабочую* и *служебную.* Ячейки служебной зоны в отличие от ячеек рабочей зоны в дальнейшем именуются *регистрами.*

Рабочая зона занимает адреса с 0.00.00.00 (00000_{10}) по 7.14.15.15 (32511_{10}). В этой зоне размещаются программы пользователя и данные.

Служебная зона занимает последние 256 байтов ОЗУ с адреса 7.15.00.00 (32512_{10}) по адрес 7.15.15.15 (32767_{10}). В некоторых регистрах этой зоны хранится информация, необходимая для нормальной работы Д3-28; часть свободных регистров может быть использована программистом. Большое число команд, использующих регистры служебной зоны, значительно расширяет возможности программиста при составлении программ в машинных командах.

Структура служебной зоны ОЗУ показана на рис. 1.2. С адреса 7.15.00.00 (32512_{10}) по адрес 7.15.01.15 (32543_{10}) расположены шестнадцать двухбайтных регистров R00, R01, ..., R15.

Каждый R-регистр состоит из двух байтов ОЗУ с соседними адресами. Байт с меньшим адресом является старшим. Регистры R08, R09, ..., R15 состоят из шестнадцати однобайтных регистров S00, S01, ..., S15.

Четыре регистра R04, R05, R06, R07 образуют восьмибайтный регистр RR, который используется в командах 12 06 (MOV X, RR) и 12 07 (MOV RR, X).

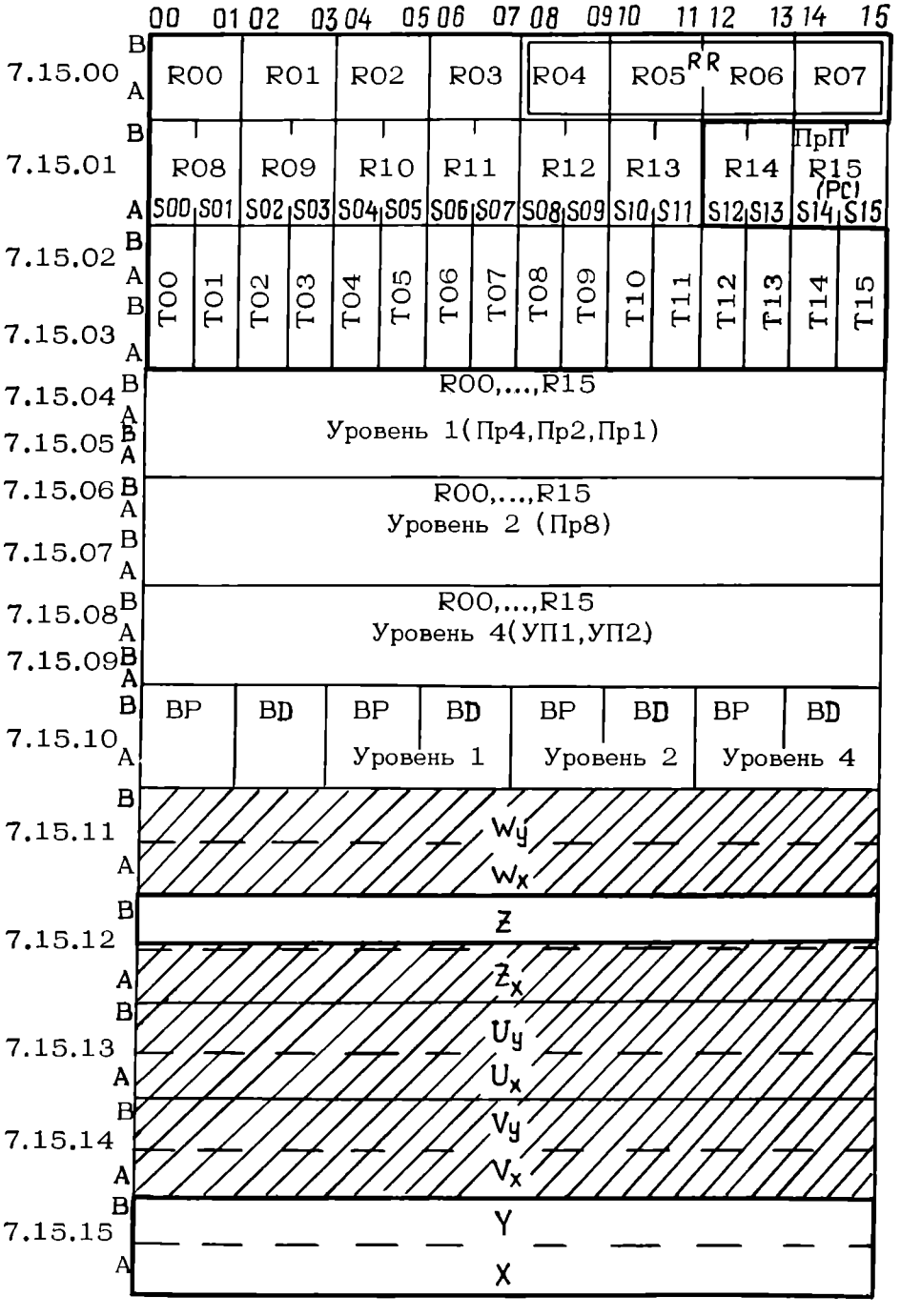

00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
7.15.00
7.15.01
7.15.02
7.15.03
7.15.04
7.15.05
7.15.06
7.15.07
7.15.08
7.15.09
7.15.10
7.15.11
7.15.12
7.15.13
7.15.14
7.15.15
B
A
R00 R01 R02 R03 R04 R05 RR R06 R07
R08 R09 R10 R11 R12 R13 R14 R15 ПрП (PC)
S00 S01 S02 S03 S04 S05 S06 S07 S08 S09 S10 S11 S12 S13 S14 S15
T00 T01 T02 T03 T04 T05 T06 T07 T08 T09 T10 T11 T12 T13 T14 T15
R00,...,R15
Уровень 1(Пр4,Пр2,Пр1)
R00,...,R15
Уровень 2 (Пр8)
R00,...,R15
Уровень 4(УП1,УП2)
BP BD BP BD BP BD BP BD
Уровень 1
Уровень 2
Уровень 4
Wy
Wx
Z
Zx
Uy
Ux
Vy
Vx
Y
X

Рис. 1.2

В остальном эти регистры ничем не отличаются от регистров R00, R01, R02, R03. С адреса 7.15.02.00 (32544_{10}) по адрес 7.15.03.15 (32575_{10}) расположены шестнадцать двухбайтных регистров T00, T01, ..., T15. Каждый Т-регистр состоит из двух байтов ОЗУ с адресами, различающимися на 16. Байт с меньшим адресом является старшим.

Область служебной зоны с адресами с 7.15.04.00 (32576_{10}) по 7.15.09.15 (32671_{10}) используется для запоминания содержимого регистров R00, R01,..., R15 при внешних прерываниях программы (более подробно см. в § 1.12).

Начиная с адреса 7.15.10.00 (32672_{10}), расположен двухбайтный регистр BP, содержащий базовый адрес программ. Содержимое регистра BP указывает на адрес ОЗУ, в котором хранится нулевой шаг программы пользователя.

Начиная с адреса 7.15.10.04 (32674_{10}), расположен двухбайтный регистр BD , содержащий базовый адрес данных. Содержимое регистра BD указывает на адрес ОЗУ, с которого начинается нулевой десятичный регистр.

При внешних прерываниях значения BP и BD переписываются в регистры с адресами с 7.15.10.04 (32676_{10}) по 7.15.10.15 (32687_{10}).

В области служебной зоны (на рис. 1.2 она заштрихована) расположены регистры, используемые микропрограммами Д3-28. Эти регистры являются для программиста запрещенными, так как изменение их содержимого может привести к непредсказуемым последствиям.

Команды десятичного формата используют следующие регистры:

X – расположен в частях A адресов байтов с 7.15.15.00 (32752_{10}) по 7.15.15.15 (32767_{10});

Y – расположен в частях B адресов байтов с 7.15.15.00 (32752_{10}) по 7.15.15.15 (32767_{10});

Z – расположен в частях B адресов байтов с 7.15.12.00 (32704_{10}) по 7.15.12.15 (32719_{10}).

Ряд регистров служебной зоны ОЗУ имеет специальное назначение. В регистре R15 хранится программный счетчик PC с признаком ПрП в старшем разряде старшего байта. Признак ПрП принимает значение 1 при работе Д3-28 по программе.

В регистре R14 хранится ряд служебных признаков и масок прерываний. Структура регистров R14 и R15 представлена на рис. 1.3, где ПрЦ – признак формирования десятичного числа; Пр0 – признак останова; N – маска внутреннего прерывания; ПрФ – признак подмикропрограмм; ПрН – признак внутреннего прерывания; УПВ – код уровня внешнего прерывания; Пр8, Пр4, Пр2, Пр1 – биты маски соответствующих сигналов внешних прерываний; ПрЖ – признак команды WAIT; ПрМ – признак ненулевой маски внешних прерываний; УП1, УП2 – биты маски внешних прерываний от ПМ; ПрП – признак работы по программе; PC – программный счетчик: (PC) = = A3.A2.A1.A0.

В регистре R13 хранится указатель стека SP.

При включении Д3-28 после автоматического выполнения микропрограммы начального запуска в регистры заносятся числа в соответствии с табл. 1.1.

Если в течение примерно 650 000 тактов после включения Д3-28 от ПУ поступит синхроимпульс СИП, то Д3-28 микропрограммно переходит к выполнению команды 04 09 *d* (GR1 < адрес ПУ >). При этом регистр ВЫВ устанавливается в соответствии с поступившим в Д3-28 по СИПу кодом, ПрГ ← 1,

Таблица 1.1

Регистр	Назначение	Устанавливаемое значение	
		шестнадцатеричное	десятичное
R13	Указатель стека SP	7.13.00.00	32000
R14	Регистр признаков и масок	0.00.00.00	–
R15	Программный счетчик, признак ПрП РС	0.00.00.00	0
BP	Базовый адрес программ	0.00.00.00	0
BD	Базовый адрес данных	0.04.00.00	1024
X	Десятичный регистр X	–	0
Y	Десятичный регистр Y	–	0
Z	Десятичный регистр Z	–	0

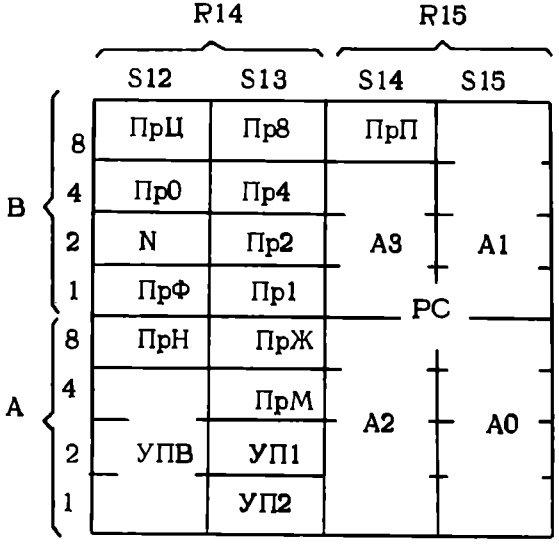


Рис. 1.3

ПрП ← 0. Все коды, посланные с ПУ, воспринимаются Д3-28 как команды, и функция клавиатуры передается ПУ. Если СИП не поступает, то устройство Д3-28 выходит на индикацию при УПР = 0.

1.3. РАБОТА С КЛАВИАТУРЫ Д3-28

В Д3-28 возможна работа в одном из четырех режимов: "Р", "В", "ПВ", "П". Режим работы задается нажатием одноименной клавиши и сопровождается свечением расположенных рядом индикаторов. При включении Д3-28 автоматически устанавливается режим "Р".

Режим "Р" ("РАБОТА") является основным. В этом режиме производит-

ся выполнение записанных в ОЗУ программ и всех команд, вводимых непосредственно с клавиатуры Д3-28, работа с НМЛ.

В режиме "В" ("ВВОД") производятся ввод программ в ОЗУ Д3-28, проверка и изменение текстов программ. В этом режиме введенная с клавиатуры команда не выполняется, а только записывается в ОЗУ по адресу, индицируемому в регистре Y в шестнадцатеричном виде.

Для работы в режимах "ПВ" ("ПЕЧАТЬ И ВВОД") и "П" ("ПЕЧАТЬ") необходимо подключить к Д3-28 печатающее устройство. Тогда в режиме "ПВ" будут реализованы режим "В" и одновременный вывод на печать очередной введенной в ОЗУ команды программы. В режиме "П" производится автоматический вывод на печать текста программы после установки нужного шага и нажатия клавиши [S] . Печатание приостанавливается при выводе на печать шагов программ с номерами, кратными 50, или при выводе кода 05 12 (END). Продолжить печатание можно нажатием клавиши [S] ("ПУСК"), а остановить его в любом месте – нажатием клавиши [Ш] ("ШАГ").

В режиме "Р" на индикацию выводится содержимое регистров X и Y. В режимах "В", "ПВ" и "П" на индикацию выводятся: в регистр Y – содержимое программного счетчика PC в шестнадцатеричном виде; в регистр X – номер шага программы в десятичном виде (пятизначное число) и код команды, записанной на этом шаге.

В одной группе с клавишами [Р], [В], [ПВ], [П] расположены и другие клавиши, не имеющие своего кода в системе команд Д3-28. Все они предназначены для так называемых *непрограммируемых операций.* Эти операции выполняются как в режиме "Р", так и в режиме "В" и предназначены для облегчения поиска нужного адреса в ОЗУ, просмотра и внесения изменений в записанную в ОЗУ программу, ее пошагового выполнения, останова, а также установки Д3-28 в исходное состояние (сброса).

При нажатии клавиши [КП] ("КОНТРОЛЬ ПРОГРАММЫ") в регистр X помещается контрольная сумма программы, равная арифметической сумме кодов программы от нулевого шага до шага, на котором записана команда 05 12 (END). При этом PC указывает адрес, на котором расположена команда 05 12 (END). Если этой команды в ОЗУ нет, включается индикатор "ОП", индикатор регистров X и Y мигает.

Контрольная сумма индицируется в режиме "Р", адрес команды 05 12 (END) – в режиме "В".

Клавиша [НШ] ("НАЙТИ ШАГ") используется для установки PC на нужный адрес ОЗУ. Для этого после нажатия на нее необходимо набрать число, соответствующее десятичному адресу ячейки памяти. Если адрес не является пятизначным числом, необходимо дополнить его слева нулями. Значение адреса не должно превышать 32 767. Ввод цифр после нажатия клавиши [НШ] не изменяет содержимого регистров X и Y , если набирается не более пяти цифр.

Нажатие клавиши [Ш] ("ШАГ") в режиме "В" увеличивает на единицу значение PC и используется для пошагового просмотра программы, записанной в ОЗУ. Нажатие клавиши [Ш] в режиме "Р" вызывает выполнение одной команды, записанной на текущем адресе ОЗУ, что используется для пошагово-

го выполнения программы. Значение РС увеличивается на единицу при выполнении одношаговой команды, и на два при выполнении двухшаговой. Если Д3-28 уже работает автоматически по программе, нажатием клавиши [Ш] производится останов работы на текущем адресе программы.

Клавиша [НМ] ("НАЙТИ МЕТКУ") используется для поиска начала участка программы с нужной меткой. Для этого после нажатия клавиши [НМ] вводится код нужной метки. Поиск производится от нулевого шага программы до кода команды 05 12 (END) или до конца области ОЗУ. Если данная метка в ОЗУ есть, РС устанавливается на первый после кода метки адрес. Если заданной метки в ОЗУ нет, включается индикатор "ОП", индикатор регистров X и Y мигает.

Клавиша [ПШ] ("ПРИБАВИТЬ ШАГ") служит для "раздвигания" текста программы, записанной в ОЗУ, с целью вставки пропущенного шага. Сдвиг содержимого ОЗУ производится от адреса, равного текущему значению РС, до адреса, на котором записана команда 05 12 (END). На текущем шаге записывается код 05 14 (GO). Адрес команды 05 12 (END) не должен превышать значения указателя стека SP. В противном случае сдвиг не происходит, включается индикатор "ОП", индикатор регистров X и Y мигает.

Клавиша [ИШ] ("ИСКЛЮЧИТЬ ШАГ") служит для исключения из текста программы одного шага. При этом происходит сдвиг содержимого ОЗУ от адреса, равного (РС) + 1, до адреса, на котором записана команда 05 12 (END).

Операции "ИШ" и "ПШ" выполняются и в режиме "Р", и в режиме "В". Необходимо помнить, что использование клавиш [ПШ] и [ИШ] в режиме "В" изменяет число шагов программы и может нарушить адресацию внутри программы при использовании команд безусловного перехода (например, команд 14 02 B2 A2 (BR. $-d$), 14 03 B2 A2 (BR. $+d$) и др.).

Кроме перечисленных клавиш непрограммируемых операций, на клавиатуре Д3-28 расположена клавиша [ЗЛ] ("ЗАПИСАТЬ НА ЛЕНТУ"), также не имеющая своего кода. Нажатие клавиши [ЗЛ] вызывает запись на МЛ содержимого ОЗУ от нулевого шага до шага, на котором расположена команда 05 12 (END). При отсутствии этой команды в памяти Д3-28 происходит запись на МЛ всего содержимого ОЗУ, после чего включается индикатор "ОМ".

Индикаторы "ОМ", "ОП" и "ПУ" расположены в левой затемненной части индикаторной панели Д3-28. Индикатор "ОМ" – верхний, индикатор "ПУ" – нижний.

Индикатор "ОМ" сигнализирует о сбое, произошедшем при считывании с МЛ. Он выключается клавишами [С], [ШН] или командой 14 14 B2 A2 (BMER. $+d$). При необходимости индикатор "ОМ" можно включить командой 04 12 12 10 (SMER).

Индикатор "ОП" включается при выполнении на Д3-28 некорректной операции (деление на нуль и др.), неправильной адресации (указано недопустимое значение адреса и пр.). Кроме того, индикатор "ОП" можно включить командой SETPER, задаваемой кодами 04 12 14 i, где i принимает значения от 08 до 15 (включительно), что позволяет отмечать в программах некоррект-

ные ситуации. Выключается индикатор клавишами [С], [ШН], командой 05 10 (BPER) или клавишей [ОП].

Индикатор "ПУ" включается при выполнении команд обмена с периферийными устройствами. Выключается он после выполнения команд обмена или нажатия клавиш [С], [ШН].

При однократном нажатии клавиши [ШН] ("ШАГ НАЗАД") содержимое программного счетчика PC уменьшается на единицу.

Клавиша [С] ("СБРОС") служит для установки Д3-28 в исходное состояние. При ее нажатии выполняются следующие действия: SP ← 7.13.00.00, PC ← (BP), X ←0, Y ← 0, обнуляются все признаки и маски, отключается НМЛ, Д3-28 выходит в режим индикации.

Код любой команды можно ввести с клавиш прямого кодирования, расположенных в верхней части клавиатуры Д3-28. Составляющую B кода команды набирают на четырех клавишах [80], [40], [20], [10] с индикаторами. Эти клавиши имеют веса 8, 4, 2 и 1 соответственно. Включение клавиши подтверждается свечением индикатора. Для выключения клавиши надо нажать на нее повторно.

Составляющая A кода команды задается нажатием одной из клавиш [00], [01], ..., [15].

Код B A команды вводится в Д3-28 только после нажатия одной из клавиш [00], [01], ..., [15].

В целях облегчения работы на Д3-28 на клавиатуре имеется ряд символьных клавиш для ввода наиболее часто используемых команд. Принятые при этом условные обозначения приведены в табл. 1.2.

В режиме "Р" команды, вводимые с клавиатуры Д3-28, сразу выполняются и их коды в ОЗУ не записываются. Таким образом, можно производить не-

Таблица 1.2

Символ на клавише	Название команды	Мнемокод	Код команды
M	Метка	MARK	04 08
СЛ	Считывание с МЛ	LOADP	05 13
▷	Поиск метки	JMM	04 07
S	Пуск	GO	05 14
E	Ввод порядка	E	07 10
ЗН	Знак X	NEG X	07 11
◊	Печать	PRINT	04 11
СК	Сброс X	CLR X	07 15
ОП	Анализ индикатора "ОП"	BPER	05 10
↑	Пересылка Y ← (X)	MOV X,Y	06 04
↓	Пересылка X ← (Y)	MOV Y,X	06 05
↑↓	Обмен (X) ⇄ (Y)	SWA X,Y	06 06
ВП	Вызов из памяти	MOV C,X	04 05
ЗП	Запись в память	MOV X,C	04 04
π	Вызов числа π в регистр X	PI	06 09

сложные расчеты непосредственно с пульта, используя для хранения промежуточных результатов ячейки памяти. Кроме показательных и логарифмических функций, можно использовать и тригонометрические (см. прил. 3). Для этого необходимо включить индикатор над клавишей [80] и выключить его над другими клавишами этой группы. После этого, руководствуясь надписями на трафарете, расположенными над клавишами [00], [01], ..., [15], надо выбрать необходимую клавишу и нажать ее. Для всех функций (кроме 08 08 (САР) и 08 09 (РОС)) аргумент должен находиться в регистре X. При вычислении таких функций содержимое регистра Y и ячеек памяти не изменяется. Результат заносится в регистр X.

При выполнении арифметических операций в регистрах X и Y сначала в регистр Y заносится число, стоящее слева от знака арифметического действия, затем в регистр X – число, стоящее справа от этого знака, после чего нажимают соответствующую клавишу арифметического действия. В регистре Y индицируется результат действия, содержимое регистра X не изменяется.

При попытке выполнить некорректную операцию (деление на нуль, вычисление функции от недопустимого значения аргумента и т. п.) включается индикатор "ОП" и мигает индикатор регистров X и Y. Устранить сбой можно нажатием клавиши [ОП] и при необходимости повторить ввод правильно.

1.4. РАЗМЕЩЕНИЕ И АДРЕСАЦИЯ ДАННЫХ В ОЗУ

В Д3-28 в зависимости от количества занимаемых байтов и формата записи данных в ОЗУ различают четыре типа ячеек:

1) однобайтные (типа регистров S00, ..., S15);
2) двухбайтные (типа регистров R00, ..., R15 или T00, ..., T15);
3) восьмибайтные (типа регистра RR);
4) десятичные (типа регистров X, Y, Z).

Для работы с каждым типом ячеек имеется отдельная группа команд. Кроме того, есть команды перезаписи информации из двухбайтных и восьмибайтных ячеек в десятичные и наоборот.

При обработке данных содержимое десятичных регистров воспринимается как десятичные числа с плавающей запятой, двухбайтных – как целые шестнадцатеричные числа со знаком, расположенным в старшем бите старшего байта.

Форматы записи десятичных и шестнадцатеричных чисел представлены на рис. 1.4 и 1.5 соответственно, где m_1, m_2, ..., m_{12} – разряды мантиссы числа; p_1, p_2 – разряды порядка числа; зн.m, зн.p – знаки мантиссы и порядка числа; i – адрес байта в ОЗУ; n_0, n_1, n_2, n_3 – разряды шестнадцатеричного числа; зн.n – знак шестнадцатеричного числа.

По аналогии с форматом регистров X, Y, Z в ОЗУ организуются десятичные ячейки CD. Адресация CD в командах задается десятичными номерами, начиная с нуля.

Например, для записи содержимого регистра X в двенадцатую десятичную ячейку необходимо выполнить команду

04 04 00 12 (MOV X, 012)

Регистры X,У,Z

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
8 4 2 1	зн.m	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}	m_{11}	m_{12}	зн.p	p_2	p_1

Регистр RR

	08	09	10	11	12	13	14	15
B 8 4 2 1	m_1	m_3	m_5	m_7	m_9	m_{11}	зн.m	p_2
A 8 4 2 1	m_2	m_4	m_6	m_8	m_{10}	m_{12}	зн.p	p_1

Рис. 1.4

Регистр R

	i	i + 1
B 8 4 2 1	зн. n	n_1
	n_3	
A 8 4 2 1	n_2	n_0

Регистр T

B 8 4 2 1	зн. n	i
	n_3	
A 8 4 2 1	n_2	
B 8 4 2 1	n_1	i + 16
A 8 4 2 1	n_0	

Рис. 1.5

Если в нулевую и первую десятичные ячейки записать соответственно числа 123456789 и $-0.777888 \cdot 10^{-4}$, они будут размещены в ОЗУ в следующем виде:

Старшие адреса байтов	Младшие адреса байтов																Номера ячеек
00.04.00	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	
B	00	01	02	03	04	05	06	07	08	09	00	00	00	00	00	09	000
A	01	07	07	07	08	08	08	00	00	00	00	00	00	01	00	04	001

Пример дан для BD = 00.04.00.00 (1024_{10}).

Как видно, две соседние десятичные ячейки занимают 16 байтов памяти. Причем десятичные ячейки с четными номерами размещаются в разрядах B байтов памяти, а ячейки с нечетными – в разрядах A.

Просмотреть записанные числа на индикаторе можно следующим образом. Нажимают последовательно клавиши [B], [НШ], [0], [1], [0], [2], [4]. На индикацию будет выведен знаковый байт чисел 00 01. Нажимая клавишу [Ш], можно последовательно просмотреть все разряды обоих чисел: 01 07, 02 07, 03 07,...

Формат записи чисел в восьмибайтные ячейки полностью совпадает с форматом записи чисел в восьмибайтный регистр RR служебной зоны (см. рис. 1.4).

И в десятичную, и в восьмибайтную ячейки помещается любое число от $-0.999999999999 \cdot 10^{-99}$ до $+0.999999999999 \cdot 10^{99}$. Но в отличие от десятичной ячейки восьмибайтная занимает восемь последовательно расположенных байтов, что упрощает пересылку десятичного числа из одного места ОЗУ в другое и организацию стека для данных.

Двухбайтные ячейки используются в основном для хранения шестнадцатеричных чисел и адресов. Формат записи чисел в двухбайтные ячейки полностью совпадает с форматом записи чисел в двухбайтные регистры R служебной зоны. Диапазон хранимых чисел: от 00.00.00.00 до 15.15.15.15 в шестнадцатеричном виде или от –32767 до +32767 в десятичном. Числа могут быть только целые.

Однобайтные ячейки подобны регистрам S служебной зоны. Адресуется ячейка заданием адреса байта в ОЗУ. Максимальное шестнадцатеричное число в однобайтной ячейке 15.15 (или 255 в десятичном представлении).

В табл. 1.3 даны все методы адресации, применяемые в Д3-28. Их использование проиллюстрировано на примерах, приведенных в табл. 1.4.

Обозначение метода адресации всегда присутствует в мнемокоде команд Д3-28, причем обычно после мнемокода первым записывается источник, а после запятой – приемник информации.

При вычислении адреса ячейки по косвенно-регистровому методу с индексированием (BD) адрес ячейки является суммой кодов выбранного регистра и регистра BD. Знаковые разряды участвуют в сложении как старшие разряды чисел. Перенос из знакового разряда теряется, индикатор "ОП" при этом не включается. Учитывая это, в команде можно указать на ячейку, адрес которой меньше (BD).

При косвенно-регистровом десятичном методе с индексированием (BD)'

Таблица 1.3

Метод адресации	Обозначение	Месторасположение операнда
Регистровый		
десятичный	X, Y, Z	В одном из регистров X, Y, Z соответственно
восьмибайтный	RR	В регистре RR (R04, R05, R06, R07)
по регистру R	R	В регистре R
по регистру T	T	В регистре T
по регистру S	S	В регистре S
Косвенно-регистровый десятичный с индексированием (BD)	@Y	В десятичной ячейке CD, номер которой равен (Y) относительно (BD)'
Косвенно-регистровый	(R)	В восьмибайтной ячейке, адрес которой равен (R)
Косвенно-регистровый с индексированием (BD)	@R	В ячейке ОЗУ, адрес которой равен (R) + (BD)
Автоинкрементный	(R) +	В двухбайтной ячейке, адрес которой равен (R); после выполнения операции (R) увеличивается на 2
Автодекрементный	– (R)	До операции содержимое регистра R уменьшается на 2; операнд в двухбайтной ячейке, адрес которой равен новому (R)
Непосредственный	#*d* или #*e*	Входит в код команды, *d* – второй байт команды, *e* – старшая тетрада второго байта команды
Прямой с индексированием	C	В десятичной ячейке, номер которой равен $(10 \cdot B2 + A2)_{10}$ относительно (BD)'

Таблица 1.4

Мнемокод	Код B1	A1	B2	A2	Метод адресации источника	Тип ячейки (регистра) источника	Метод адресации приемника	Тип ячейки (регистра) приемника	Выполняемая операция
1	2				3	4	5	6	7
MOV X,Y	06	04			X	Десятичная	Y	Десятичная	Y ← (X)
MOV Y, Z	12	14			Y	”	Z	”	Z ← (Y)
MOV X,RR	12	06			X	”	RR	Восьмибайтная	RR ←(X)
MOV X,R08	04	13	11	08	X	”	R	Двухбайтная	R08←INT(X)
MOV R03,T11	14	12	03	11	R	Двухбайтная	T	Формат T-регистра	T11←(R03)

1	2		3		4	5	6	7	
MOV S09,S00	11	12	09	00	S	Однобайтная	S	Однобайтная	S00←(S09)
MOV@Y,X	05	05			@Y	Десятичная	X	Десятичная	X←(CD(Y))
MOV X,(R09)	04	12	02	09	X	"	(R)	Восьмибайтная	(R09)←(X)
MOV S01,@R07	09	12	01	07	S	Однобайтная	@R	Однобайтная	((R07)+ +(BD))←(S01)
MOV (R11)+, R08	10	15	08	11	(R)+	Двухбайтная	R	Двухбайтная	R08← ((R11));
MOV R10 –(R07)	10	12	10	07	R	"	(–R)	"	(R07)–2; (R07)←(R10)
MOV #0007, S03	13	03	00	07	*#d*	–	S	Однобайтная	S03←00 07
MOV X,074	04	04	07	04	X	Десятичная	C	Десятичная	CD 074←(X)

недопустимо указание ячейки, большей 4095, из-за непредсказуемости результатов вычисления адреса. Так же, как и в случае использования косвенно-регистрового метода с индексированием (BD), может быть указана ячейка, адрес которой меньше (BD).

1.5. РАЗМЕЩЕНИЕ И АДРЕСАЦИЯ ПРОГРАММ В ОЗУ

Программа на машинно-ориентированном языке Д3-28 представляет собой последовательность выполняемых на Д3-28 команд. Одна команда может занимать один или два байта ОЗУ. В зависимости от этого различают *одношаговые* или *двухшаговые* команды. Одношаговые команды обозначаются B1 A1, двухшаговые – B2 A2 B1 A1.

Каждый шаг программы имеет свой адрес. Адресация шагов производится относительно базового адреса программ BP, который определяет, где в ОЗУ находится нулевой шаг программы.

При нажатии клавиши [C] в программный счетчик PC заносится значение (BP).

Выполнение команд считывания и записи на МЛ, вычисления КП и некоторых команд перехода производится относительно значения BP, что нужно учитывать при программировании.

При включении Д3-28 в регистр BP автоматически записывается число 00.00.00.00. При необходимости значение BP можно изменить, выполнив команду 04 13 06 i (MOV Ri,BR), предварительно определив Ri. С помощью этой команды в ручном режиме с пульта можно восстановить необходимое значение BP, разрушенное в результате сбоя.

При вводе программы необходимо следить за тем, чтобы она не пересекалась с областью данных, ограниченной снизу значением BD. Если это произойдет, то при записи чисел в десятичные ячейки с начальными номерами будет уничтожаться текст программы, записанный в данной области. Для того чтобы это не произошло, можно записать в BD адрес, больший адреса последнего шага программы.

1.6. КОМАНДЫ ОБРАБОТКИ ДАННЫХ В ДЕСЯТИЧНЫХ РЕГИСТРАХ И ЯЧЕЙКАХ

Десятичное число, как правило, вводится в Д3-28 через регистр X. Число в регистре X формируется командами, приведенными в табл. 1.5.

Командой E числу в регистре X присваивается нулевой порядок. Если после E вводятся цифры, то две последние из них определяют величину порядка. Если команда NEG X вводится до команды E, она изменяет знак мантиссы, если после, – то знак порядка числа.

Для выполнения пересылок и обмена данными между десятичными ячейками используются команды, приведенные в табл. 1.6.

В командах, использующих косвенно-регистровый десятичный метод адресации с индексированием (BD)', номер адреса ячейки не должен превышать значения целой части выражения (32511 – (BD))/8 во избежание непредсказуемых результатов. Если номер ячейки меньше 0 или больше 9999, включается индикатор "ОП". После выполнения команд пересылок предыдущее содержимое ячейки или регистра приемника теряется.

Запись CDC означает десятичную ячейку CD с номером C; запись Y ← (X) – пересылку содержимого регистра X в регистр Y ; запись (Y) ← (X) – пересылку содержимого регистра X в десятичную ячейку с номером, указанным в регистре Y; запись X ← ((Y)) – пересылку в регистр X содержимого ячейки с номером, указанным в регистре Y.

В десятичных ячейках можно производить четыре арифметических действия: сложение, вычитание, умножение, деление. Команды арифметических операций и команда вызова остатка приведены в табл. 1.7.

Таблица 1.5

Мнемокод	Код B1 A1	B2 A2	Выполняемая операция
DIG *e*	07 *e*		Ввод десятичной цифры в регистр X ($0 \leq e \leq 9$)
E	07 10		Сброс порядка (X), начало формирования нового порядка (X)
NEG X	07 11		Изменение знака мантиссы (X) или порядка (X)
POINT	07 12		Установка запятой (десятичной точки)
CLR X	07 15		Очистка (X)
ADD #10,E	04 12	07 00	Прибавление 10 к порядку (X)
ADD #*e*,E	04 12	07 *e*	Прибавление *e* к порядку (X) ($1 \leq e \leq 9$)
SUB #10,E	04 12	04 00	Вычитание 10 из порядка (X)
SUB #*e*,E	04 12	04 *e*	Вычитание *e* из порядка (X) ($1 \leq e \leq 9$)

Таблица 1.6

Мнемокод	Код B1 A1	B2 A2	Выполняемая операция	Индексация
MOV X,C	04 04	C	CDC ← (X)	(BD)′
MOV X,Y	06 04		Y ← (X)	
MOV X,@Y	05 04		(Y) ← (X)	(BD)′
MOV Y,C	04 14	C	CDC ← (Y)	(BD)′
MOV Y,X	06 05		X ← (Y)	
MOV Y,Z	12 14		Z ← (Y)	
MOV C,X	04 05	C	X ← (CDC)	(BD)′
MOV @Y,X	05 05		X ← ((Y))	(BD)′
MOV C,Y	04 15	C	Y ← (CDC)	(BD)′
MOV Z,Y	12 15		Y ← (Z)	
SWA X,C	04 06	C	(CDC) ⇄ (X)	(BD)′
SWA X,Y	06 06		(Y) ⇄ (X)	
SWA X,@Y	05 06		(X) ⇄ ((Y))	(BD)′

Таблица 1.7

Мнемокод	Код B1 A1	B2 A2	Выполняемая операция	Индексация
ADD X,C	04 00	C	CDC ← (CDC)+ (X)	(BD)′
ADD X,Y	06 00		Y ← (Y)+ (X)	
ADD X,@Y	05 00		Y ← ((Y))+ (X)	(BD)′
SUB X,C	04 01	C	CDC ← ((CDC)) – (X)	(BD)′
SUB X,Y	06 01		Y ← (Y)–(X)	
SUB X,@X	05 01		(Y) ← ((Y))–(X)	(BD)′
MUL X,C	04 02	C	CDC ← (CDC)·(X)	(BD)′
MUL X,Y	06 02		Y ← (Y)·(X)	
MUL X,@Y	05 02		(Y) ← ((Y))·(X)	(BD)′
DIV X,C	04 03	C	CDC ← (CDC)/(X)	(BD)′
DIV X,Y	06 03		Y ← (Y) / (X)	
DIV X,@Y	05 03		(Y) ← ((Y))/(X)	(BD)′
RES	07 14		X ← остаток	

На Д3-28 может быть выполнено вычисление ряда математических функций. Для этого в каждом конкретном случае достаточно ввести только одну команду. Перечень этих команд приведен в прил. 3.

Все функции, за исключением САР и РОС, в качестве аргумента используют содержимое регистра X. Результат вычислений также помещается в ре-

гистр X. Для функций CAP и POC, предназначенных для преобразования координат из декартовой системы в полярную и обратно, аргументы занимают как регистр X, так и регистр Y.

В тригонометрических функциях угол задается в радианах. Точность вычисления этих функций не ниже половины младшего разряда мантиссы при значениях аргумента $0 \leqslant x \leqslant 2\pi$.

1.7. КОМАНДЫ ОБРАБОТКИ ДАННЫХ В ОДНО-, ДВУХ- И ВОСЬМИБАЙТНОМ ФОРМАТАХ

В однобайтных ячейках арифметические действия не выполняются. В них производится только пересылка содержимого и обмен им, а также все основные логические операции: логическое умножение ($\wedge$), логическое сложение ($\vee$), исключающее ИЛИ ($\forall$), дополнение до кода 15 15 ($^{-}$). Указанные команды приведены в табл. 1.8 и 1.9.

Таблица 1.8

Мнемокод	Код B1 A1	B2 A2	Выполняемая операция	Индексация
MOV @Ri,Sj	09 13	j i	Sj ← ((Ri))	(BD)
MOV Si,Sj	11 12	i j	Sj ← (Si)	
MOV Si,@Rj	09 12	i j	(Rj) ← (Si)	(BD)
MOV #*d*,Si	13 i	*d*	Si ← *d* (*d* – код второго байта команды)	
SWA Si,Sj	11 14	i j	(Si) ⇄ Sj)	
SWA Si,@Rj	09 14	i j	(Si) ⇄ ((Rj))	(BD)
SWA Si	04 12	09 i	Обмен тетрад в Si	

Таблица 1.9

Мнемокод	Код B1 A1	B2	A2	Выполняемая операция	Индексация
OR Si,Sj	11 11	i	j	Sj ← (Si)∨(Sj)	
OR Si,@Rj	09 11	i	j	(Rj) ← ((Rj))∨(Si)	(BD)
AND Si,Sj	11 08	i	j	Sj ← (Si)∧(Sj)	
AND Si,@Rj	09 08	i	j	(Rj) ← ((Rj))∧(Si)	(BD)
XOR Si,Sj	11 09	i	j	Sj ← (Si)∀(Sj)	
XOR Si,@Rj	09 09	i	j	(Rj) ← ((Rj))∀(Si)	(BD)
COM Sj	11 10	B2	j	Sj ← ($\overline{Sj}$)	
COM @Rj	09 10	B2	j	(Rj) ← (($\overline{Rj}$))	(BD)

Таблица 1.10

Мнемокод	Код B1 A1	Код B2 A2	Выполняемая операция	Тип ячейки: источника	Тип ячейки: приемника
MOVD Ri,X	04 13	04 i	$X \leftarrow (Ri)_{10}$ (перевод в десятичную систему счисления)	Двухбайтная	Десятичная
MOVH X,Ri	04 13	12 i	$Ri \leftarrow INT(X)_{16}$ (перевод в шестнадцатеричную систему счисления)	Десятичная	Двухбайтная
MOV Ri,X	04 13	03 i	$X \leftarrow (Ri)$	Двухбайтная	Десятичная
MOV X,Ri	04 13	11 i	$Ri \leftarrow INT(X)$	Десятичная	Двухбайтная
MOV RR,X	12 07		$X \leftarrow (RR)$	Восьмибайтная	Десятичная
MOV X,RR	12 06		$RR \leftarrow (X)$	Десятичная	Восьмибайтная
MOV (Ri),X	04 12	03 i	$X \leftarrow ((Ri))$	Восьмибайтная	Десятичная
MOV X,(Ri)	04 12	02 i	$(Ri) \leftarrow (X)$	Десятичная	Восьмибайтная
MOV (Ri),Y	04 12	01 i	$Y \leftarrow ((Ri))$	Восьмибайтная	Десятичная
MOV Y,(Ri)	04 12	00 i	$(Ri) \leftarrow (Y)$	Десятичная	Восьмибайтная

Данные, хранимые в двухбайтных ячейках и регистрах R00,...,R15, представляют собой целые шестнадцатеричные числа со знаком.

Восьмибайтные ячейки и регистр RR (состоит из регистров R04, R05, R06, R07) используются для хранения десятичных чисел.

Имеется ряд команд пересылки данных между десятичным регистром X и двухбайтными регистрами и ячейками. При таких пересылках производится преобразование формата данных, а в некоторых командах – и преобразование системы счисления. Все эти команды, а также команды пересылки данных между регистрами X, Y и восьмибайтными ячейками или регистрами X, Y и регистром RR приведены в табл. 1.10. В них индексация не производится. Кроме того, в таблице указаны форматы данных приемника и источника.

При пересылке десятичного числа из регистра X в двухбайтные регистры и ячейки пересылается только целая часть числа, а дробная отбрасывается, причем значение целой части X не должно превышать 32 767.

В регистры R08,...,R12 шестнадцатеричные цифры удобно вводить через регистры S00,...,S09 с помощью команд 13 i *d* (MOV#*d*, Si). При таком способе загрузка выполняется примерно в 10 раз быстрее, чем через регистр X. Естественно, десятичные числа должны быть предварительно переведены в шестнадцатеричную систему счисления. Для этого можно ввести в регистр X десятичное число, выполнить с клавиатуры Д3-28 в режиме "Р" команду 04 13 12 08 (MOVH X,R08) и прочитать в регистрах S00 и S01 в режиме "В" шестнадцатеричный эквивалент числа.

Ввод чисел в регистры R00,...,R07, в которых нет возможности прямого доступа к каждому байту, можно производить пересылкой их из регистров R08,..., R13.

Группа команд пересылок и обмена для двухбайтных ячеек приведена в

Таблица 1.11

Мнемокод	Код B1 A1	B2 A2	Выполняемая операция	Индексация
MOV BD,Ri	04 13	13 i	Ri ← (BD)	
MOV BP,Ri	04 13	14 i	Ri ← (BP)	
MOV Ri,BD	04 13	05 i	BD ← (Ri)	
MOV Ri,BP	04 13	06 i	BP ← (Ri)	
MOV Ri,Rj	11 04	i j	Rj ← (Ri)	
MOV Ri,Tj	14 12	i j	Tj ← (Ri)	
MOV Ri,-(Rj)	10 12	i j	Rj ← (Rj)−2;(Rj)←(Ri) (запись в стек)	
MOV Ri,@Rj	09 04	i j	(Rj) ← (Ri)	(BD)
MOV @Ri,Rj	09 05	j i	(Rj) ← ((Ri))	(BD)
MOV (Ri)+,Rj	10 15	j i	(Rj) ← ((Ri)); Ri ← (Ri)+2 (вызов из стека)	
MOV Ti,Rj	14 13	i j	Rj ← (Ti)	
SWA Ri,Rj	11 06	i j	(Ri) ⇄ (Rj)	
SWA Ri,@Rj	09 06	i j	(Ri) ⇄ ((Rj))	(BD)
SWA Ri	04 12	08 i	Обмен байтов в Ri	

табл. 1.11. Среди них есть команды, адресуемые к регистрам BP и BD. Команды записи в стек и считывания из стека в качестве указателя стека используют любой выбранный регистр R00, ..., R12. Следует отличать указатель стека, используемый программистом в указанных командах, от указателя стека Д3-28, хранящегося в регистре R13. (Регистр R13 используется микропрограммами Д3-28 в ряде команд безусловного перехода и при начальном запуске микро-ЭВМ загружается адресом 07.13.00.00.)

Группа команд арифметических операций и простейших функций для двухбайтных ячеек приведена в табл. 1.12.

Особую группу команд в Д3-28 составляют команды обработки символьных последовательностей. При выполнении этих команд содержимое ячеек

Таблица 1.12

Мнемокод	Код B1 A1	B2 A2	Выполняемая операция	Индексация
ABC Ri	04 13	08 i	Ri ← |(Ri)|	
CLR Ri	04 13	10 i	Ri ← 0	
NEG Ri	04 13	09 i	Ri ← −(Ri)	
ADD Ri,Rj	11 00	i j	Rj ← (Rj)+ (Ri)	
ADD Ri,@Rj	09 00	i j	(Rj)←((Rj))+ (Ri)	(BD)
ADD #*e*,Ri	10 00	*e* i	Ri ← (Ri)+ *e*	
MUL Ri,Rj	11 02	i j	Rj ← (Rj)·(Ri)	
MUL Ri,@Rj	09 02	i j	(Rj) ← ((Rj))·(Ri)	(BD)
SUB Ri,Rj	11 01	i j	Rj ← (Rj) − (Ri)	
SUB Ri,@Rj	09 01	i j	(Rj)←((Rj)) − (Ri)	(BD)
SUB #*e*, Ri	10 01	*e* i	Ri ← (Ri) − *e*	

ОЗУ Д3-28 рассматривается как последовательность кодов символов и их адреса не индексируются. Под символами понимаются коды цифр, букв латинского и русского алфавитов, знаков в соответствии с ГОСТ 13052–74. Таблица кодов символов приведена в прил. 4. Эти коды могут вводиться в Д3-28 с внешнего устройства – клавиатуры дисплея или ПМ типа "Consul". Ниже подробно описаны все команды этой группы.

1. Команда

10 03 B2 A2 (ATOI *d*)

предназначена для преобразования числа, записанного в ОЗУ в кодах ГОСТ 13052–74 с нулевым дополнительным разрядом, в целое десятичное число в кодах Д3-28 и записи его в регистр X.

Считывание числа из ОЗУ начинается с адреса, записанного в R01. Если очередной считанный символ является цифрой, он заносится в регистр X; если это символ с кодом B2 A2, указанным в команде ATOI *d*, он пропускается и считывается следующий символ. При считывании первого кода, отличного от B2 A2 и от кода цифр, или кода тринадцатой цифры преобразование завершается, и этот код записывается в регистр S03. В регистр R01 записывается адрес, следующий за адресом кода, занесенного в S03. Если считанное число начинается с нулей, то после команды ATOI необходимо выполнить нормализацию содержимого регистра X с помощью команды 12 09 (NORM).

2. По команде

10 04 B2 A2 (NSN *d*)

начиная с адреса, записанного в регистре R01, производится поиск первого, не равного B2 A2 кода. Найденный код записывается в S03, а в R01 – адрес кода, следующего за ним.

3. Команда

10 05 B2 A2 (NSS *d*)

аналогична NSN *d*, но по ней производится поиск кода, равного B2 A2.

Верхняя граница поиска в командах NSN *d* и NSS *d* не определена, поэтому при необнаружении искомого кода эти команды могут зациклиться, что необходимо учитывать при их использовании.

4. По команде

10 06 i j (ANS Si, Sj)

производится анализ содержимого регистра Si. Результат анализа в соответствии с табл. 1.13 заносится в регистр Sj.

5. Команда

10 07 i j (PRIOR Si, Rj)

аналогична команде ANS Si, Sj. Результат анализа регистра Si в соответствии с табл. 1.14 заносится в регистр Rj.

Использование ряда команд обработки данных одно-, двух- и восьмибайтного форматов проиллюстрировано в приведенных ниже примерах.

Пример 1.1. Занести в регистр S03 код 04 01.

Команда: 13 03 04 01 MOV #04 01, S03

Пример 1.2. Занести код 05 12 в ячейку ОЗУ с адресом A = 03.04.05.06 (13398_{10}).

Таблица 1.13

(Si)	Группа кодов ГОСТ 13052–74	(Sj)
03 00–03 09	Цифры	00 01
04 01–05 10	Латинские буквы	00 03
06 00–07 15	Русские буквы и забой	00 05
00 00–02 00	Служебные знаки	00 09
08 00–11 15		00 13
12 00–15 15		00 11
Прочие коды		00 07

Таблица 1.14

(Si)	Символ ГОСТ 13052–74	(RJ)
02 08	(	00 02 00 01
02 09	)	00 01 00 03
02 10	*	00 04 00 05
02 11	+	00 03 00 07
02 13	–	00 03 00 11
02 15	/	00 04 00 15
05 14	¬	00 05 00 13
Прочие коды		00 00 00 09

Пусть (BD) = 00.04.00.00 (1024_{10}). Применим для пересылки кода команду 09 12 ij (MOV Si,@Rj), использовав регистры S04 и R08. Поскольку адресация ячейки ОЗУ в этой команде производится относительно BD, то в R08 необходимо предварительно занести адрес, равный разности абсолютного адреса A ячейки ОЗУ и установленного значения BD:

$$(R08) = A - (BD) = 03.04.05.06 - 00.04.00.00 = 03.00.05.06\ (12374_{10}).$$

Учитывая, что двухбайтный регистр R08 состоит из двух однобайтных регистров S00 и S01, получаем такую последовательность команд:

```
13 04  05 12  MOV # 05 12, S04
13 00  03 00  MOV # 03 00, S00
13 01  05 06  MOV # 05 06, S01
09 12  04 08  MOV  S04,@R08
```

Пример 1.3. Обменять тетрады в однобайтном регистре S02.
Пусть (S02) = 15 03.
Команда: 04 12 09 02 SWA S02
Результат: (S02) = 03 15.
Пример 1.4. Обнулить старший разряд байта, расположенного в регистре S03.
Пусть (S03) = 14 03 (11100011_2).
Команды: 13 00 07 15 MOV # 07 15, S00
11 08 00 03 AND S00, S03
Результат: (S00) = 07 15 (01111111_2),
(S03) = 06 03 (01100011_2).

Пример 1.5. Переслать содержимое регистра X в восьмибайтную ячейку рабочей зоны ОЗУ с адресом, указанным в регистре R10.

Пусть

(R10) = 03.05.02.00 (13600_{10}),

(X) = –.123456789876 E 32.

Команда: 04 12 02 10 MOV X, (R04)

Результат. Начиная с адреса 03.05.02.00 (13600_{10}) ОЗУ Д3-28, расположены восемь байтов, соответствующих переданному значению регистра X:

01 02
03 04
05 06
07 08
09 08
07 06
01 00
03 02

Расположение разрядов числа соответствует обозначениям рис. 1.4.

Пример 1.6. Переслать содержимое регистра R10 в двухбайтную ячейку ОЗУ Д3-28 с адресом, равным содержимому регистра X.

Пусть

(R10) = 07.05.06.08,

(BD) = 00.04.00.00 (1024_{10}),

(X) = 16 (00.00.01.00).

Команды: 04 13 12 11 MOV X, R11
09 04 10 11 MOV R10, @R11

Результат. Начиная с адреса A = (X) + (BD) = 00.00.01.00 + + 00.04.00.00 = 00.04.01.00 (1040_{10}), расположены последовательно два байта: 07 05 и 06 08, соответствующие переданному содержимому регистра R10.

Пример 1.7. Установить новое значение базового адреса данных BD равным 00.15.14.13, сохранив его предыдущее значение в регистре R02.

Команды: 04 13 13 02 MOV BP, R02
13 00 00 15 MOV #00 15, S00
13 01 14 13 MOV #14 13, S01
04 13 05 08 MOV R08, BD

1.8. КОМАНДЫ УСЛОВНЫХ И БЕЗУСЛОВНЫХ ПЕРЕХОДОВ

Последовательность выполнения команд программы могут изменять команды условных и безусловных переходов.

В *командах условного перехода* проверяется выполнение какого-либо заданного командой условия (соотношение содержимого регистров или ячеек, состояние индикаторов, сравнение с заданным кодом и др.). Это условие может выполняться или не выполняться. Если условие выполняется, нормальная последовательность команд нарушается. Переход происходит относительно адреса, на котором записан первый шаг команды условного перехода. Этот адрес обозначается точкой. Величина изменения адреса при переходе может присутствовать или отсутствовать в коде команды. Если она не указана, переход производится на адрес .+3 для одношаговой и .+ 4 для двухшаговой команды условного перехода. Если условие не выполняется, нормальная последователь-

ность вычислений по программе не изменяется, и происходит переход к очередной команде программы, т. е. на адрес .+ 1 для одношаговых команд и .+ 2 для двухшаговых, где точка обозначает адрес первого шага команды перехода.

Таблица 1.15

Мнемокод	Код B1	A1	B2	A2	Условие ветвления	Изменение (PC) при выполнении условия ветвления
BEQ Y,X	05	09			(Y)=(X)(проверка условия (Y) – (X) = 0)	.+3
BEQZ X	04	12	06	11	(X)=0 (по первому разряду мантиссы)	.+4
BEQZ Y	04	12	04	11	(Y)=0 (по первому разряду мантиссы)	.+4
BGE Y,X	05	07			(Y)≥(X) (проверка условия: (Y)–(X) – положительно)	.+3
BLT Y,X	05	08			(Y)<(X) (проверка условия: (Y)–(X) – отрицательно)	.+3
BMI X	04	12	07	10	(X) отрицательно (по ненулевому знаковому разряду)	.+4
BMI Y	04	12	05	10	(Y) отрицательно (по ненулевому знаковому разряду)	.+4
BNEZ X	04	12	07	11	(X) ≠ 0 (по первому разряду мантиссы)	.+4
BNEZ Y	04	12	05	11	(Y)≠0 (по первому разряду мантиссы)	.+4
BPL X	04	12	06	10	(X) положительно (по знаковому разряду)	.+4
BPL Y	04	12	04	10	(Y) положительно (по знаковому разряду)	.+4
BSA Y,X	12	04			Код (Y) равен коду (X) (последовательное совпадение всех разрядов)	.+3

Таблица 1.16

Мнемокод	Код B1	A1	B2	A2	Условие ветвления	Изменение (PC) при выполнении условия ветвления	Индексация
ABGE Ri,Rj	14	09	i	j	Rj ← (Rj) + 1; (Ri) ≥ (Rj)	.+4	
BSAZ Ri, + *e*	11	03	*e*–1	i	(Ri) = 0	.+*e*, при *e*=1 PC← .+2	
BSAZ @Ri.+*e*	09	03	*e*–1	i	((Ri)) = 0	.+*e*, при *e*=1 PC←.+2	(BD)
BSA Ri,Rj	11	07	i	j	(Ri)=(Rj)	.+4	
BSA Ri,@Rj	09	07	i	j	(Ri)=((Rj))	.+4	(BD)
BGE Ri,Rj	14	10	i	j	(Ri)≥(Rj)	.+4	
SOBZ Ri.+*e*	14	08	*e*–1		Ri←(Ri)–1; (Ri)=0	.+*e*, при *e*=1 PC←.+2	

Обычно сразу после команды условного перехода ставится команда безусловного перехода для передачи управления ветви программы, обрабатывающей невыполнение проверяемого условия.

В табл. 1.15–1.18 приведены следующие команды условного перехода: по анализу десятичных регистров (табл. 1.15); по анализу операндов двухбайтного формата (табл. 1.16); по анализу операндов однобайтного формата (табл. 1.17); по битам и сигналам (табл. 1.18).

Команды BSA Y,X и BEQ Y,X (см. табл. 1.15) различаются по выполнению. Команда BSA Y,X проверяет поразрядное совпадение кодов всех шестнадцати разрядов регистров Y и X. Для нее числа –0 и +0 не равны. По команде BEQ Y,X выполняется проверка на нуль первого разряда разности (Y)–(X). В командах BEQZ и BNEZ анализ на нуль (Y) или (X) производится по первому разряду мантиссы.

При выполнении команд SOBZ Ri,*e* и ABGE Ri,Rj (см. табл. 1.16) может включаться индикатор "ОП", если анализируемый регистр принимает отрица-

Таблица 1.17

Мнемокод	Код B1	A1	B2	A2	Условие ветвления	Изменение (PC) при выполнении условия ветвления	Индексация
BSA Si,Sj	11	15	i	j	(Si)=(Sj)	.+4	
BSA Si,@Rj	09	15	i	j	(Si)=((Rj))	.+4	(BD)
BEV @Ri.+*e*	14	07	*e*–1	i	В байте по адресу (Ri) количество единичных бит четное	.+*e*, при *e*=1 PC←.+2	(BD)
BNS #*d*,S1	10	08	*d*		(S01)≠*d*	.+4	
BNS #*d*,S3	10	09	*d*		(S03)≠*d*	.+4	
BHIS Si,Sj	10	02	i	j	(Si)>(Sj)	.+4	

Таблица 1.18

Мнемокод	Код B1	A1	B2	A2	Условие ветвления	Изменение (PC) при выполнении условия ветвления	Индексация
BBIC i @Rj	14	15	i	j	Бит с номером i в байте по адресу (Rj) равен нулю (B2<8)	.+4	(BD)
BBIS i,@Rj	14	15	i +8	j	Бит с номером i в байте по адресу (Rj) равен единице (B2 >B8)	.+4	(BD)
BKEY .+*d*	14	11	B2	A2	П4=1 (специальная клавиша ПМ)	.+16B2+A2+1	
BMER .+*d*	14	14	B2	A2	ОМ=1, после анализа ОМ ← 0	.+16B2+A2+1	
BPER	05	10			ОП=1, после анализа ОП←0	.+3	

тельное значение. В этих командах изменение на единицу содержимого регистра Ri производится до его анализа.

Команды SOBZ Ri,*e* и ABGE Ri,Rj удобно использовать для организации циклических вычислений на Д3-28.

В командах BBIC i, @Rj и BBIS i,@Rj (см. табл. 1.18) биты анализируемого байта нумеруются от 0 до 7 соответственно от старшего к младшему.

Сигнал $\overline{П4}$, анализируемый командой 14 11 B2 A2 (BKEY .+*d*), поступает в Д3-28 от внешнего устройства. В частности, он может быть установлен нажатием третьей слева клавиши, расположенной на передней нижней части ПМ "Consul 260.1".

Таблица 1.19

Мнемокод	Код B1	A1	B2	A2	Выполняемые операции
BR .–*d*	14	02	B2	A2	PC ← .–*d* , где *d* = 16·B2 + A2 – 1
BR .+*d*	14	03	B2	A2	PC ← .+*d*, где *d* = 16·B2 + A2 + 1
JMM *d*	04	07	*d*		PC ← адрес команды, следующей за MARK *d*; ПрП ← 1
JMP @X	12	13			PC ← (BP) + $(X)_{16}$
JMP @Ri	04	13	00	i	PC ← (BP) + (Ri)
JMTT *d*	10	10	*d*		PC ← (7.13.*d*); ПрП ← 1
JMTF *d*	10	11	*d*		PC ← (7.14.*d*); ПрП ← 1
JSM B1 A1	00	A1 или 01 A1			R13 ← (R13)–2; (R13) ← ПрП, (PC); Z ← (Y); PC ← адрес команды, следующей за MARK B2 A2, где код B2 A2 команды MARK совпадает с кодом B1 A1 команды JSM; ПрП ← 1
JSM B1 A1	02	A1 или 03 A1			R13 ← (R13)–2; (R13) ← ПрП, (PC); PC ← адрес команды, следующей за MARK B2 A2, где код B2 A2 команды MARK совпадает с кодом B1 A1 команды JSM; ПрП ← 1
JSM @Ri	04	13	01	i	R13 ← (R13)–2; (R13) ← ПрП, (PC); PC ← (BP) + + (Ri); ПрП ← 1
JSTT *d*	10	13	*d*		R13 ← (R13)–2; (R13) ← ПрП, (PC); PC ← (7.13.*d*); ПрП ← 1
JSTF *d*	10	14	*d*		R13 ← (R13) –2; (R13) ← ПрП, (PC); PC ← (7.14.*d*); ПрП ← 1
RTS	05	11			ПрП, PC ← ((R13)) ; PC ← (PC) + ПрП; R13 ← (R13) + 2
RTSI	12	12			R13 ← (R13) + 2
RTSGO	04	12	07	12	ПрП, PC ←((R 13));PC ← (PC) +ПрП; R13 ← (R13)+2; ПрП←1; ПрП←0; УПР←0
TRAP	12	05			Эквивалентно JSTF 0000

Команды безусловного перехода бывают трех типов.

1. Переход по значению второго байта команды. Адрес перехода равен адресу первого шага команды плюс (минус) значение второго байта команды в шестнадцатеричном виде.

2. Переход по метке. Адресом перехода является адрес, следующий за командой метки 04 08 (MARK) с заданным кодом второго шага. Поиск производится от адреса, равного (BP), до адреса, на котором записана команда 05 12 (END), или до адреса, равного (SP). При необнаружении искомой метки включается индикатор ОП, индикатор регистров X и Y мигает.

3. Переход по содержимому регистра. Адрес перехода равен (BP) + $(X)_{16}$ или (BP) + (Ri) в зависимости от команды.

4. Переход по содержимому таблицы. (*Таблицами* называются области ОЗУ объемом по 256 байтов, расположенные с адресов 7.13.00.00 (32000_{10}) и 7.14.00.00 (32256_{10}).) Адрес перехода равен содержимому ячейки 7.13.*d* или 7.14.*d*, где *d* – код второго байта команды.

При выполнении команд безусловного перехода вычисленное значение адреса не должно превышать 7.15.15.15 (32767_{10}) и быть отрицательным при вводе адреса из регистра X.

Безусловные переходы могут выполняться с запоминанием или без запоминания адреса обращения. Команды с запоминанием используются при обращении к подпрограммам, после возврата из которых работа основной программы продолжается далее с адреса, следующего за адресом команды обращения. Для запоминания адреса обращения используется стек с указателем, расположенным в регистре R13.

Для возврата из подпрограммы обычно используется команда 05 11 (RTS).

Команды безусловных переходов и организации подпрограмм приведены в табл. 1.19.

Время выполнения команды перехода по метке 04 07 B2 A2 (JMM B2 A2) прямо пропорционально отдаленности расположения метки от начала программы, поэтому ее использование обычно оправдано только на стадии отладки программы. После отладки программы в целях сокращения времени ее выполнения желательно заменить команды перехода по меткам на команды перехода с непосредственной адресацией 14 03 B2 A2 (BR.+*d*),14 02 B2 A2 (BR.–*d*) или другие (в зависимости от величины перехода).

1.9. КОМАНДЫ УПРАВЛЕНИЯ НМЛ

В Д3-28 запись /считывание информации с МЛ производится двумя способами. При первом способе используются команды 05 13 (LOADP), 12 02 (LOADX) и 12 03 (SAVEX), в которых длина записываемого/считываемого блока произвольная.

Командой 05 13 (LOADP) производится загрузка информации с МЛ в ОЗУ, начиная с адреса, установленного в РС перед выполнением команды; командой 12 02 (LOADX) осуществляется загрузка информации с МЛ в ОЗУ, начиная с адреса, записанного в десятичном виде в регистре X; командой 12 03 (SAVEX) выполняется запись информации из ОЗУ на МЛ, начиная с адреса, записанного в десятичном виде в регистре X, по адрес, на котором записана команда 05 12 (END).

Для записи программ на МЛ с адреса, равного содержимому РС, по адрес с командой 05 12 (END) используется клавиша ЗЛ .

При втором способе информация записывается/считывается с МЛ блоками одинаковой длины (по 256 байтов). Для этого используются команды 04 12 11 i (SAVER i) и 04 12 10 i (LOADR i).

При первом способе записи/считывания не требуется предварительное выполнение команд для задания режима работы узлов НМЛ. При втором способе необходимо предварительное выполнение ряда команд для установки НМЛ в рабочее состояние. Этот способ позволяет более гибко использовать НМЛ и повышать надежность записи/считывания информации на МЛ за счет того, что блоки можно записывать/считывать при непрерывном движении МЛ и без отрыва магнитной головки от ее поверхности. Блоки можно дублировать программным путем, что значительно повышает надежность работы с НМЛ.

Начало МЛ определяется с помощью фотодатчика. Для его работы необходимо, чтобы в начале и конце МЛ имелись ракорды – прозрачные участки ленты длиной 15–20 см.

Ниже приведен примерный порядок программной записи массива информации на МЛ, когда запись требуется произвести с начала МЛ.

1. Командой 12 00 (REW) МЛ приводится в движение для перемотки.

2. Командой 04 12 12 13 (CRFS) в цикле опрашивается завершение выполнения команды перемотки.

3. Командами 04 12 12 02 (LAMP) и 04 12 12 11 (ELMG) после завершения перемотки включаются лампочка и электромагнит.

4. Задается пауза, равная примерно 100 мс, для надежной установки магнитной головки и прижимного ролика в рабочее положение. Для задания пауз используется команда 04 12 14 02 (PAUSER). Длительность задаваемой паузы определяется содержимым регистра R10 и равна 35 + (R10) · 10 тактов. В зависимости от модификации Д3-28 длительность одного такта равна 1 или 2 мкс.

5. Командой 04 12 12 03 (TRTAP) включаются двигатели протяжки и подмотки МЛ.

6. Командой 04 12 12 04 (SAVC) или 04 12 12 12 (SAVS) включается стирание МЛ. Эта операция необходима для очистки начального участка МЛ после "выборки" ракорда.

7. Командой 04 12 12 05 (CLDRS) в цикле опрашивается ракорд. После обнаружения конца ракорда задается пауза, равная примерно 300 мс.

После "выборки" ракорда лампочку можно выключить командой 04 12 12 00 (STTAP). Так как эта команда, кроме лампочки, отключает двигатели и электромагнит, после нее необходимо повторить команды 04 12 12 11 (ELMG), 04 12 12 03 (TRTAP) и 04 12 12 12 (SAVS) (или же 04 12 12 04 (SAVC)).

8. Все блоки программы последовательно формируются в ОЗУ и командами 04 12 11 i(SAVER i) записываются на МЛ.

9. После записи последнего блока торможение МЛ рекомендуется производить с помощью такой последовательности команд:

1) 04 12 12 00 (STTAP);

2) 04 12 12 11 (ELMG);

3) пауза, равная примерно 100 мс;

4) 04 12 12 03 (TRTAP);

5) 04 12 12 00 (STTAP).

Для последующей записи информации на данную МЛ в случае, когда ее уже не надо перематывать в исходное состояние, последовательность действий может быть такой.

1. Командой 04 12 12 11 (ELMG) включается электромагнит.

2. Задается пауза, равная примерно 100 мс.

3. Командой 04 12 12 03 (TRTAP) включаются двигатели протяжки.

4. Командой 04 12 12 04 (SAVC) или 04 12 12 12 (SAVS) включается стирание МЛ.

5. Задается пауза записи межблочного промежутка и разгона двигателей, равная примерно 150 мс.

6. Командами 04 12 12 11 i (SAVER i) формируются в ОЗУ и записываются последовательно все блоки массива.

При формировании в ОЗУ блоков рекомендуется использовать буфер ввода/вывода на МЛ длиной 256 байтов, а выводимый блок оформлять в следующем виде:

Идентификатор	Контрольная сумма блока	Выводимая информация
N байтов	2 байта	256 -- (N + 2) байтов

Такое оформление блока позволяет: автоматически находить записи по их именам; дублировать блоки при записи; создавать открывающую и закрывающую записи для выводимых массивов команд программ или данных.

Для считывания с МЛ (если необходимо предварительно перемотать ее в исходное состояние) рекомендуется следующий порядок действий.

1. Командой 12 00 (REW) МЛ перематывается в исходное состояние.

2. Командой 04 12 12 13 (CRFS) в цикле опрашивается завершение выполнения команды перемотки.

3. Командами 04 12 12 02 (LAMP) и 04 12 12 11 (ELMG) после завершения перемотки включаются лампочка и электромагнит.

4. Задается пауза, равная примерно 300 мс.

5. Командой 04 12 12 05 (CLDRS) опрашивается ракорд. После обнаружения конца ракорда задается пауза, равная примерно 300 мс. Лампочку для продления срока ее службы желательно выключить командой STTAP. Так как при выполнении этой команды отключаются также электромагнит и двигатель НМЛ, то их необходимо сразу включить командами ELMG и TRTAP и задать паузу (примерно 100 мс) для разгона двигателей.

6. Командой 04 12 10 i (LOADR i) считывается с МЛ и загружается в ОЗУ с адреса (Ri) первый блок и т. д. После считывания каждого блока анализируются его идентификатор и контрольная сумма. В результате анализа этот блок может быть оставлен в ОЗУ или нет, если он дублирует ранее считанный или считан с ошибкой.

7. Если считан последний блок, то МЛ останавливают с помощью следующей последовательности команд:

1) 04 12 12 00 (STTAP);

2) 04 12 12 11 (ELMG);

3) пауза, равная примерно 100 мс;

4) 04 12 12 03 (TRTAP);

5) 04 12 12 00 (STTAP).

Команды записи/считывания на МЛ и команды управления НМЛ приведены в табл. 1.20.

Многообразие команд записи/считывания позволяет составлять эффективные программы работы с МЛ, несмотря на простоту НМЛ, используемого в Д3-28.

Таблица 1.20

Мнемокод	Код B1	A1	B2	A2	Выполняемые операции
CLDRS	04	12	12	05	Опрос ракорда: S00= {0, если есть ракорд; 1, если нет ракорда}
CRFS	04	12	12	13	Опрос перемотки: S00={0, если есть перемотка; 1, если нет перемотки}
ELMG	04	12	12	11	Включение электромагнита
INFS	04	12	12	07	Опрос информационных импульсов (ИИ) : S00←ИИ
FORW	04	12	12	09	Перемотка вперед
LAMP	04	12	12	02	Включение лампы
LOADP	05	13			Загрузка в ОЗУ с МЛ; начальный адрес загрузки (PC) + ПрП
LOADR i	04	12	10	i	Загрузка в ОЗУ с МЛ блока в 256 байтов; начальный адрес загрузки (Ri)
LOADX	12	02			Загрузка в ОЗУ с МЛ; начальный адрес загрузки (BP) + $(X)_{16}$
REW	12	00			Перемотка назад
SAVC	04	12	12	04	Запись нулевого бита
SAVER i	04	12	11	i	Запись из ОЗУ на МЛ блока в 256 байтов с адреса (Ri)
SAVEX	12	03			Запись из ОЗУ на МЛ от адреса (BP) + $(X)_{16}$ до адреса, на котором записан END
SAVS	04	12	12	12	Запись единичного бита
SMER	04	12	12	10	ОМ ← 1
SNCS	04	12	12	06	Опрос синхроимпульсов (СИ) : S00←СИ
STTAP	04	12	12	00	Останов двигателей, отключение лампы фотодатчика и электромагнита
TRTAP	04	12	12	03	Включение двигателей, отключение записи
WTRT	04	12	14	03	Поиск паузы, превышающей (R10) · 10 тактов

1.10. КОМАНДЫ КОНТРОЛЯ, УПРАВЛЕНИЯ И ОТЛАДКИ ПРОГРАММ

Для контроля записанных в ОЗУ программ используются специальные команды, вычисляющие контрольные суммы всех кодов команд контролируемой программы.

Контроль программ основан на том, что каждой программе можно поставить в соответствие некоторое число, равное сумме кодов составляющих ее команд. Это число называется *контрольной суммой программы.* Значение КП для разных программ лежит в широких пределах и редко повторяется, что позволяет использовать КП для идентификации программ при их вводе, например с МЛ.

По команде 12 01 (VERX) в регистр X заносится десятичная сумма частей B и A кодов ОЗУ, записанных от адреса, равного $(X)_{16}$ + (BP), до адреса, на котором расположена команда 05 12 (END). Код команды 05 12 (END) в КП не входит.

Если при вычислении КП по команде 12 01 (VERX) в ОЗУ, начиная от адреса $(X)_{16}$ + (BP) до адреса 7.15.00.00, отсутствует команда 05 12 (END), включается индикатор "ОП" и мигает индикатор регистров X, Y.

По команде 14 04 B2 A2 (VERR B2 A2) в регистры R00 и R01 заносится шестнадцатеричная сумма кодов ОЗУ, записанных с адреса (RB2) + (BD) по адрес (RA2) + (BD). Здесь регистры R00 и R01 используются как один четырехбайтный регистр, старший разряд которого расположен в байте с меньшим адресом.

По команде 14 05 B2 A2 (VEX B2 A2) в регистр S00 заносится контрольный байт, являющийся суммой по модулю 2 всех байтов ОЗУ с адреса (RB2) + + (BD) по адрес (RA2) + (BD).

Кроме этих команд, вычисление КП можно выполнить с помощью клавиши [КП] (см.§ 1.3).

К группе служебных команд управления программой относятся команды, представленные в табл. 1.21.

Команда 05 14 (GO) производит запуск программы с адреса, равного значению программного счетчика PC. Команда обычно вводится нажатием клавиши [S] клавиатуры Д3-28. Одна или несколько команд 05 14 (GO), записан-

Таблица 1.21

Мнемокод команды	Код команды
GO	05 14
STOP	05 15
MARK B2 A2	04 08 B2 A2
END	05 12
CMD Ri	04 13 02 i
PAUSE	04 12 06 15
PAUSER	04 12 14 02

ных в середине программы, не производят никаких действий и являются "пустыми".

Команда 05 15 (STOP) выполняет останов Д3-28. После останова программный счетчик PC указывает на адрес, следующий за командой STOP.

Команда 04 08 B2 A2 (MARK B2 A2) записывается в начале того участка программы, которому в дальнейшем может быть передано управление. Код B2 A2 называется *меткой* и является идентификатором участка программы. Код 04 08 команды вводится нажатием клавиши [М] . Поиск метки может осуществляться одной из команд:

04 07 B2 A2 (JMM B2 A2)
00 A1 или 01 A1 (JSM B1 A1)
02 A1 или 03 A1 (JSM B1 A1)

В зависимости от выбранной команды передача управления помеченному участку ОЗУ может быть произведена как простой безусловный переход (команда JMM B2 A2) или как обращение к подпрограмме (команда JSM B1 A1). Во втором случае для возврата из подпрограммы в конце ее необходимо поставить команду 05 11 (RTS).

Команда 05 12 (END) производит останов подпрограммы, но при этом в отличие от команды 05 15 (STOP) содержимое регистра BP заносится в PC. Команда 05 12 (END) считается логическим концом программы. Она является ограничителем сверху при записи на МЛ, поиске метки, вычислении КП посредством клавиши [КП] и по команде 12 01 (VERX), при редактировании программы клавишами [ИШ] и [ПШ] .

По команде 04 13 02 i (CMD Ri) выполняется команда, код которой записан в регистре Ri. Причем если команда одношаговая, то ее код должен быть записан в старшем байте Ri.

По команде 04 12 06 15 (PAUSE) приостанавливается работа Д3-28, на индикацию в течение приблизительно 0,7 с выводится содержимое регистров X и Y , после чего Д3-28 продолжает работать по программе.

По команде 04 12 14 02 (PAUSER) длительность паузы определяется содержимым регистра R10 и равна 35 + (R10)·10 тактов. При выполнении команды индикатор не светится.

При отладке программы обычно возникает необходимость ее редактирования, т. е. изменения текста введенной в ОЗУ программы. При редактировании количество шагов программы может быть уменьшено с помощью клавиши [ИШ] или увеличено с помощью клавиши [ПШ] . Работа с этими клавишами описана в § 1.3.

Для контроля промежуточных значений в нужных местах программы можно записывать команды останова 05 15 (STOP) или паузы 04 12 06 15 (PAUSE). После останова по команде 05 15 (STOP) можно продолжать выполнение программы по шагам (используя клавишу [Ш]) или в программном режиме (вводя клавишей [S] команду 05 14 (GO)).

1.11. РАБОТА С ПЕРИФЕРИЙНЫМИ УСТРОЙСТВАМИ

Любое устройство, подключенное к разъемам ВВОД/ВЫВОД, ПЕЧАТЬ, ПЛ/ФС, по отношению к Д3-28 называется *периферийным.*

Для работы с ПУ, подключенными к разъему ВВОД/ВЫВОД, используются следующие регистры и сигналы.

Регистр "ВВ" ("ВВОД") служит для приема информации от ПУ. Его разряды обозначаются: $\overline{\text{Вв в8}}$, $\overline{\text{Вв в4}}$, $\overline{\text{Вв в2}}$, $\overline{\text{Вв в1}}$, $\overline{\text{Вв а8}}$, $\overline{\text{Вв а4}}$, $\overline{\text{Вв а2}}$, $\overline{\text{Вв а1}}$.

Регистр "ВЫВ" ("ВЫВОД") служит для вывода информации из Д3-28. Его разряды обозначаются: $\overline{\text{Y82}}$, $\overline{\text{Y42}}$, $\overline{\text{Y22}}$, $\overline{\text{Y12}}$, $\overline{\text{X82}}$, $\overline{\text{X42}}$, $\overline{\text{X22}}$, $\overline{\text{X12}}$.

Регистр "УПР" ("УПРАВЛЕНИЕ") служит для задания адреса выбираемого ПУ. Его разряды обозначаются: $\overline{\text{Y83}}$, $\overline{\text{Y43}}$, $\overline{\text{Y23}}$, $\overline{\text{Y13}}$, $\overline{\text{X83}}$, $\overline{\text{X43}}$, $\overline{\text{X23}}$, $\overline{\text{X13}}$.

Сигналы $\overline{\text{СИМ}}$, Вв, $\overline{\text{СИП}}$ служат для синхронизации обмена с ПУ.

Сигналы Пр1, Пр2, Пр4, Пр8 используются ПУ для прерывания работы Д3-28 (см. § 1.12).

Сигнал В.Пр.П используется при вводе программы периферийным устройством.

Команды ввода/вывода Д3-28 приведены в табл. 1.22. (Отметим, что в 3-й колонке таблицы приведено число синхроимпульсов СИП, необходимых для завершения команды. При подаче команды с клавиатуры (РС) не изменяется.)

Все команды различаются по следующим признакам:

1) вводу или выводу информации;
2) способу адресации ПУ;
3) наличию или отсутствию адресных четырехбайтных передач перед вводом/выводом информации;
4) количеству принимаемых или передаваемых байтов;
5) способу указания адреса ОЗУ, с которого производится ввод/вывод;
6) наличию или отсутствию в передаче контрольного байта;
7) времени ожидания ответа от ПУ;
8) сохранению или изменению состояния регистра УПР после выполнения команды.

Адресация ПУ может производиться по одному из 256 состояний регистра УПР. Не рекомендуется использовать следующие значения УПР, применяемые для работы со стандартным оборудованием и в некоторых командах Д3-28:

00 00 – работа со встроенной клавиатурой;
00 02 – адресные передачи;
00 04 и 00 05 – команды 04 09 B2 A2 (GR1 B2 A2) и 04 10 B2 A2 (GR2 B2 A2);
08 00, 09 00, 10 00 – работа с графопостроителем;
12 00 – ввод с фотосчитывающего устройства;
13 00 – ввод с ПМ;
14 00 – вывод на ПМ;
15 00 – вывод на перфоратор.

Адрес ПУ в зависимости от используемой команды ввода/вывода можно задавать несколькими способами:

1) командами выбора ПУ 04 09 B2 A2 (GR1 B2 A2) или 04 10 B2 A2 (GR2 B2 A2);

Таблица 1.22

Мнемокод	Код B1 A1	B2 A2	Количество байтов по каналу	Выполняемые операции	Примечание
1	2	3	4	5	6
INPS *d*	15 00	*d*	(S09)	УПР ← *d*; прием в ОЗУ с начального адреса (R10)+(BD) (S09) байтов	Время ожидания ответа ПУ (СИП) не ограничено; по окончании приема УПР ← 0; PC ← .+2
INPR *d*	15 04	*d*	(R12)	УПР ← *d*; прием в ОЗУ с начального адреса (R10)+(BD) (R12) байтов	
INPAS *d*	15 08	*d*	4+(S09)	УПР ← 00 02; ВЫВ ← 00 00; (S02); (S03); 00 00; УПР ← *d*; прием в ОЗУ с начального адреса (R10)+(BD) (S09) байтов	
INPAR	15 12	*d*	4+(R12)	УПР ← 00 02; ВЫВ ← 00 00; (S02); (S03); 00 00; УПР ← *d*; прием в ОЗУ с начального адреса (R10)+(BD) (R12) байтов	
INPSV *d*	15 02	*d*	(S09)+1	УПР ← *d*; прием в ОЗУ с начального адреса (R10)+(BD) (S09) байтов; прием контрольного байта	Время ожидания ответа ПУ (СИП) не ограничено; по окончании приема УПР ← 0, PC ← .+2. Если побитная сумма по модулю 2 всех принятых байтов не равна 15 15, то ОП ← 1; контрольный байт в ОЗУ не записывается
INPRV *d*	15 06	*d*	(R12)+1	УПР ← *d*; прием в ОЗУ с начального адреса (R10)+(BD) (R12) байтов; прием контрольного байта	
INPASV	15 10	*d*	4+(S09)+1	УПР←00 02; ВЫВ←00 00; (S02); (S03); 00 00; УПР ← *d*; прием в ОЗУ с начального адреса (R10)+(BD) (S09) байтов; прием контрольного байта	
INPARV *d*	15 14	*d*	4+(R12)+1	УПР ← 00 02; ВЫВ ← 00 00; (S02); (S03); 00 00; УПР ← *d*; прием в ОЗУ с начального адреса (R10)+(BD) (R12) байтов; прием контрольного байта	
INPO *d*	14 00	*d*	1	УПР ← *d*; прием в ОЗУ по адресу (R10)+(BD) одного байта	Время ожидания ответа ПУ (СИП) не ограничено; по окончании приема УПР ← ←0; PC←.+2

1	2	3	4	5	6
INPOWC	04 12	14 04	1 (при поступлении ответа ПУ)	Прием в S03 одного байта при неизменном коде (УПР)	Время ожидания ответа ПУ (СИП) не превышает (R10)·10 тактов; PC←.+2, если ответ ПУ в указанное время не поступит; PC←.+4, если ответ ПУ в указанное время поступит
INPOWS	04 12	14 06	1 (при поступлении ответа ПУ)	УПР←(S02); прием в S03 одного байта; по окончании приема УПР←0	
OUTS *d*	15 01	*d*	(S09)	УПР ← *d*; передача из ОЗУ с начального адреса (R10)+(BD) (S09) байтов	Время ожидания ответа ПУ (СИП) не ограничено; по окончании передачи УПР←0; PC←.+2
OUTR *d*	15 05	*d*	(R12)	УПР ← *d*; передача из ОЗУ с начального адреса (R10)+(BD) (R12) байтов	
OUTAS *d*	15 09	*d*	4+(S09)	УПР ← 00 02; ВЫВ ← 00 00; (S02); (S03); 08 00; УПР ← *d*; передача из ОЗУ с начального адреса (R10)+(BD) (S09) байтов	
OUTAR *d*	15 13	*d*	4+(R12)	УПР ← 00 02; ВЫВ ← 00 00; (S02); (S03); 08 00; УПР ← *d*; передача из ОЗУ с начального адреса (R10)+(BD) (R12) байтов	
OUTSV *d*	15 03	*d*	(S09)+1	УПР ← *d*; передача из ОЗУ с начального адреса (R10) + (BD) (S09) байтов; передача контрольного байта	Время ожидания ответа ПУ (СИП) не ограничено; по окончании передачи УПР←0; PC←.+2; контрольный байт дополняет побитную сумму информационных байтов до кода 15 15
OUTRV *d*	15 07	*d*	(R12)+1	УПР ← *d*; передача из ОЗУ с начального адреса (R10)+(BD) (R12) байтов; передача контрольного байта	
OUTASV *d*	15 11	*d*	4+(S09)+1	УПР←00 02; ВЫВ←00 00; (S02); (S03); 08 00; УПР←*d*; передача из ОЗУ с начального адреса (R10)+(BD) (S09) байтов; передача контрольного байта	

Окончание табл. 1.22

1	2	3	4	5	6
OUTARV *d*	15 15	*d*	4+(R12)+1	УПР ← 00 02; ВЫВ ← 00 00; (S02); (S03); 08 00; УПР ← *d*; передача из ОЗУ с начального адреса (R10) + (BD) (R12) байтов; передача контрольного байта	
OUTO *d*	14 01	*d*	1	УПР ← *d*; передача из ОЗУ одного байта с адреса (R10) + (BD)	Время ожидания ответа ПУ (СИП) не ограничено; по окончании передачи УПР←0; PC←.+2
OUTOWC	04 12	14 05	1 (при поступлении ответа ПУ)	Передача (S03) при неизменном коде (УПР)	Время ожидания ответа ПУ (СИП) (R10)·10 тактов; PC←.+2, если ответ ПУ в указанное время не поступил; PC←.+4, если ответ ПУ в указанное время поступил
OUTOWS	04 12	14 07	1 (при поступлении ответа ПУ)	УПР←(S02); передача (S03); по окончании передачи УПР←0	

2) состоянием регистра УПР, предварительно установленным командой 04 12 14 00 (LNCN);

3) состоянием регистра УПР, равным предварительно записанному в регистр S02 коду;

4) состоянием регистра УПР, равным значению второго байта команды ввода/вывода.

При использовании команд 04 09 B2 A2 (GR1) или 04 10 B2 A2 (GR2) выбор ПУ осуществляется в соответствии с содержимым регистра УПР (00 04 или 00 05) и регистра ВЫВ, равного B2 A2. Здесь B2 A2 задает адрес ПУ. После выбора функции клавиатуры передаются ПУ, и все посланные им коды воспринимаются Д3-28 как команды. Из-за низкого быстродействия эти команды выбора ПУ следует применять только для передачи функций клавиатуры выносному пульту.

Для установки регистра УПР в нужное состояние используется команда 04 12 14 00 (LNCN). При ее выполнении УПР ← (S02).

В некоторых командах перед вводом/выводом информации выполняется так называемая *адресная передача,* которая используется для задания начального адреса в памяти ПУ, начиная с которого будет производиться прием или передача информации. Значение этого адреса должно быть занесено до выполнения команды в регистр R09, состоящий из однобайтных регистров S02 и S03.

Независимо от адреса, выбранного ПУ, адресная передача ведется при УПР = 00 02, после чего осуществляется ввод/вывод информации при УПР = = < адрес ПУ > .

Порядок обмена с адресной передачей показан на рис. 1.6, причем в командах ввода d = 00 00, а в командах вывода d = 08 00.

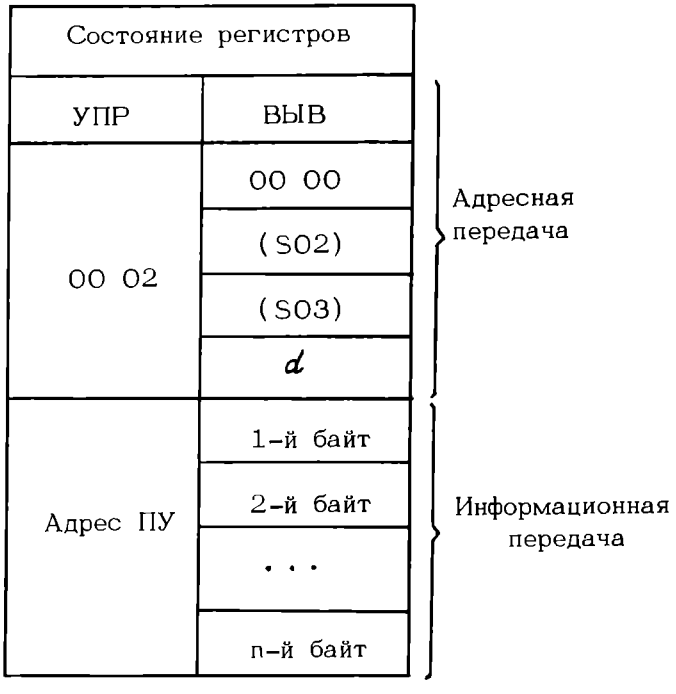


Рис. 1.6

Четыре байта адресной передачи не входят в число передаваемых или принимаемых по команде байтов. Количество принимаемых или передаваемых байтов для разных команд может быть равно (R12), (S09) или только единице.

Адрес ОЗУ, начиная с которого производится ввод/вывод информации, в большинстве команд равен (R10) и индексируется (BD). В остальных командах для ввода/вывода используется регистр S03.

При выполнении некоторых команд производится контроль ввода/вывода. В этом случае при выводе после заданного числа байтов передается контроль-

ный байт, дополняющий сумму переданных байтов по модулю 2 до кода 15 15. После ввода заданного числа байтов проверяется, совпадают ли переданный и подсчитанный контрольные байты. При несовпадении включается индикатор "ОП". Контрольный байт в ОЗУ не записывается.

При обмене с ПУ Д3-28 вырабатывает сигнал СИМ, на который ПУ должно при обработке очередного байта ответить сигналом СИП. Время ожидания этого сигнала в одних командах не ограничено, а в других не может быть больше (R10)·10 тактов. Если в указанное время СИП не поступает в Д3-28, то PC ← .+4, и выполняются очередные команды программы.

После выполнения одних команд ввода/вывода регистр УПР сохраняет свое состояние, после других – обнуляется (см. табл. 1.22).

1.12. ОБСЛУЖИВАНИЕ ПРЕРЫВАНИЙ

В Д3-28 могут обрабатываться внутренние и внешние прерывания. Для их разрешения или запрета служат соответственно однобайтная маска N и шестибайтная маска M, состоящая из бит УП2, УП1, Пр8, Пр4, Пр2, Пр1. Расположение бит масок прерывания показано на рис. 1.3. Прерывание разрешается, если соответствующий бит маски установлен в единичное состояние. При включении Д3-28 или нажатии клавиши $\boxed{C}$ N ← 0 и M ← 0.

Внутреннее прерывание происходит после выполнения команды, вызвавшей включение индикатора "ОП" и разрешенной маски внутреннего прерывания N , если в это время не производится обработка внутреннего прерывания. При обработке внутреннего прерывания все другие прерывания запрещены.

Установка маски N осуществляется командой 04 12 07 14 (MOVX, N). При этом значение N становится равным младшему биту первого разряда регистра X.

При ОП = 1, N = 1 в регистре T00 запоминается текущее значение PC, управление передается программе, обрабатывающей прерывание, индикатор "ОП" устанавливается в нулевое состояние. Адрес программы обработки прерывания равен 0.00. (0.00.00.08), т. е. старший байт адреса нулевой, а младший равен содержимому ячейки 0.00.00.08 ОЗУ Д3-28.

При ОП = 1, N = 0 в T0 запоминается адрес сбоя (адрес первой команды программы, вызвавшей включение индикатора "ОП"). Выполнение программы продолжается, "ОП" остается включенным.

Внешнее прерывание вызывается сигналами УП2, УП1, Пр4, Пр8, Пр2, Пр1.

Установка маски M производится командой 04 13 07 i (MOV Si, M). При этом битам маски соответствуют шесть младших разрядов регистра Si.

Значение маски M можно переслать в регистр Si с помощью команды 04 13 15 i (MOV M, Si).

С состоянием маски M связан признак ненулевой маски внешних прерываний ПрM. При M = 0 ПрM = 0. Если хотя бы один бит маски M установлен в единичное состояние, то ПрM = 1. При ПрM = 0 сигналы прерывания не опрашиваются вообще. Если ПрM = 1, то в программном режиме после выполнения каждой команды (за исключением команд безусловного перехода с мнемоникой BR и команд формирования числа в регистре X) опрашиваются сигналы прерывания, что удлиняет время выполнения команд на 7 тактов.

Сигналы внешних прерываний разделены на три уровня:

УП2, УП1 – уровень 4;

Пр8 – уровень 2;

Пр4, Пр2, Пр1 – уровень 1.

При обработке сигнала прерывания в УПВ (см. рис. 1.3) записывается его уровень. В этом случае прерывание обработки прерывания может произойти только по сигналу более высокого уровня или по внутреннему прерыванию.

Сигнал УПВ устанавливается в нулевое состояние при нажатии клавиши [C] и восстанавливает свое значение при возврате из прерывания, а также при псевдовозврате по команде 12 10 (RTII).

Сигналы УП2 и УП1 на внешний разъем не выведены. Они формируются в ДЗ-28 при обмене с ПМ типа "Consul". Сигнал УП2 устанавливается в единичное состояние, если на ПМ нажата кодовая клавиша, за исключением клавиш перевода регистров. Сигнал УП1 устанавливается в единичное состояние, если ПМ готова к печати символа.

Если для ввода/вывода на ПМ использовать прерывающие сигналы УП2 и УП1, то при отсутствии информации с ПМ или во время печатания очередного символа в основной программе могут производиться какие-либо действия. Ввод символа будет осуществляться только по сигналу УП2, а вывод – по УП1.

Анализ сигналов УП1 и УП2 при работе с ПМ не является обязательным, так как для работы с ПМ, как и с любым ПУ, можно использовать команды ввода/вывода. Но использование сигналов УП1, УП2 сокращает время ожидания ответа от ПУ и увеличивает быстродействие программы.

Начальные адреса программ обработки прерываний могут находиться только в первых 256 байтах ОЗУ, так как старший байт их адреса нулевой, а младший равен содержимому ячеек с 0.00.00.02 по 0.00.00.07 соответственно для сигналов УП2, УП1, Пр8, Пр4, Пр2, Пр1. При одновременном поступлении нескольких сигналов одинакового уровня первым опрашивается сигнал, для которого начальный адрес прерывающей программы записан в ячейке с меньшим адресом.

Перед передачей управления программе, обрабатывающей внешнее прерывание, происходит перезапись содержимого регистров R00, R01, ..., R15, BP, BD в другую область служебной зоны ОЗУ (см. рис. 1.2).

Адреса прерывающих программ и адреса запоминания регистров R00, R01, ..., R15, BP, BD указаны в табл. 1.23.

Ожидание поступления сигналов внешнего прерывания может быть осуществлено командой 12 08 (WAIT), устанавливающей признак ПрЖ (см. рис. 1.3) в единичное состояние. По этой команде сигналы прерывания опрашиваются в цикле через каждые 8 тактов. Предварительно должны быть подготовлены соответствующие биты маски М.

Возврат из прерывания, как внутреннего, так и внешнего, может быть осуществлен командой 12 11 (RTI). При возврате из внутреннего прерывания PC ← (T00) и снова разрешается внутреннее прерывание. При возврате из внешнего прерывания в регистры R00, R01, ..., R15, BP и BD переписывается содержимое области служебной зоны, соответствующей данному уровню прерывания.

Таблица 1.23

Причина прерывания	Бит регистра S, соответствующий биту M	Начальный адрес прерывающей программы	Уровень прерывания	Адреса запоминания (R00), (R01),...,(R15)	(BP)	(BD)
УП2 УП1	в2 в1	0.00. (0.00.00.02) 0.00. (0.00.00.03)	4	7.15.08.00,..., 7.15.09.15	7.15.10.14, 7.15.10.15	7.15.10.12, 7.15.10.13
Пр8	а8	0.00. (0.00.00.04)	2	7.15.06.00,..., 7.15.07.15	7.15.10.10, 7.15.10.11	7.15.10.08, 7.15.10.09
Пр4 Пр2 Пр1	а4 а2 а1	0.00. (0.00.00.05) 0.00. (0.00.00.06) 0.00. (0.00.00.07)	1	7.15.04.00,..., 7.15.05.15	7.15.10.06, 7.15.10.07	7.15.10.04, 7.15.10.05

Примечание. Время реакции на внешнее прерывание – 230 тактов плюс время выполнения последовательности команд, между которыми анализ прерываний не производится.

Если прерывание наступило при выполнении команды 12 08 (WAIT), то по команде 12 11 (RTI) управление передается следующей за WAIT команде.

Выход из прерывания может быть осуществлен командой псевдовозврата 12 10 (RTII), действие которой для внутреннего прерывания заключается в восстановлении его разрешения, а для внешнего – в восстановлении (УПВ) из той ячейки, в которой он запоминался.

2. ПРОГРАММИРОВАНИЕ НА АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ БЭЙСИК

2.1. ОБЩИЕ СВЕДЕНИЯ О ЯЗЫКЕ БЭЙСИК

Язык БЭЙСИК был разработан в 1965 г. по заказу фирмы "General Electric" и предназначался для широкого круга пользователей ЭВМ, не владевших ранее приемами программирования. Он быстро завоевал популярность, так как, несмотря на свою простоту, позволял решать довольно сложные задачи. Важное достоинство языка – приспособленность к работе в режиме диалога с ЭВМ, что особенно удобно для начинающих программистов. В нашей стране БЭЙСИК получил широкое распространение. Им снабжены практически все мини- и микроЭВМ, а также многие терминальные системы к большим ЭВМ.

В связи с тем что на БЭЙСИКе нет единого стандарта, существует множество версий этого языка [19, 28, 30] различного уровня сложности. Ниже описан язык БЭЙСИК (вариант 3А), который реализован на микроЭВМ "Электроника Д3-28". Все количественные ограничения, которые будут встречаться, справедливы для этой версии языка. В других версиях они могут быть иными.

БЭЙСИК-программа состоит из последовательности пронумерованных строк. Каждая строка может содержать не более 100 символов. Нумерация строк производится от 1-й до 7999-й. Соседним строкам при первичном наборе программы рекомендуется давать номера, отличные друг от друга на 5 или 10, чтобы при отладке программы можно было вставить между ними другие строки. Каждая строка может содержать один или несколько операторов (предложений языка). В последнем случае операторы разделяются двоеточием.

Например:

а) 15 LET A=B+C
   20 LET E=A * SIN(X)

б) 15 LET A=B+C: LET E=A * SIN(X)

Во фрагменте "а" каждая строка содержит по одному оператору, а во фрагменте "б" – два. При вычислениях оба фрагмента дадут одинаковый результат.

Оператор должен начинаться и оканчиваться в одной строке, поэтому большие выражения необходимо записывать с помощью нескольких операторов.

Общая форма записи всех операторов приведена в прил. 5, а список операторов – в прил. 6.

Программу обычно заканчивают оператором END.

Приведем пример текста БЭЙСИК-программы.

Пример 2.1. Вычислить значение функции $y = a + b$.

Возможный вариант программы:

```
10 INPUT A,B
20 LET Y=A+B
30 PRINT Y
40 END
```

В произвольных местах строки могут быть записаны пробелы. Все они, кроме заключенных в апострофы, игнорируются. Однако ключевые слова БЭЙСИКа должны записываться без пробелов.

Например:

GOTO, GOSUB, NEXT – правильная запись;
GO TO, GO SUB, NE XT – неправильная запись.

2.2. РЕЖИМЫ РАБОТЫ БЭЙСИК-ИНТЕРПРЕТАТОРА

В зависимости от комплекта внешних устройств Д3-28 БЭЙСИК-программа может вводиться различными способами. Ниже будет рассматриваться ввод с клавиатуры дисплея. Порядок подготовки ЭВМ к работе с дисплеем описан в § 3.2. Здесь мы только отметим, что появление на экране дисплея текста "ГОТОВ:" или просто символа ":" указывает на готовность ЭВМ к работе с БЭЙСИК-интерпретатором.

Интерпретатор – это программа, которая осуществляет перевод текста БЭЙСИК-программы в машинные команды ЭВМ. (Подробно работа интерпретатора рассмотрена в гл. 3.) Интерпретатор позволяет работать в непосредственном режиме и режиме ввода программ.

Режим ввода программ рассмотрим на примере 2.1. Набор текста каждой строки на клавиатуре дисплея завершается нажатием клавиши [ПС] ("ПЕРЕВОД СТРОКИ"), при этом текст строки помещается в ОЗУ. Набираемый текст отображается на экране дисплея. Для устранения неправильно набранных символов используется клавиша забоя [ЗБ] . Каждое ее нажатие стирает последний символ в уже набранном тексте строки, что позволяет затем ввести вместо неправильных символов правильные. Изменить строку, уже записанную в ОЗУ Д3-28, можно повторным набором исправленной строки. Для исключения строки из программы надо набрать ее номер и нажать клавишу [ПС] . Вначале обучения рекомендуется пользоваться только этими приемами исправления текста, так как есть разница между стиранием информации на экране дисплея и в ОЗУ Д3-28. (Подробнее о редактировании текста см. в § 4.4.) После редактирования строки или всей программы рекомендуется проверить внесение изменений. Для этого следует набрать команду LIST и нажать клавишу [ПС] . На экране дисплея появится текст, который находится в ОЗУ.

Все операторы и команды БЭЙСИКа вводятся нажатием клавиши [ПС] , поэтому в дальнейшем [ПС] будем опускать.

Запуск программы производится набором на клавиатуре дисплея команды RUN. При этом выполнение программы начинается со строки с минимальным номером. Например, при наборе текста программы из примера 2.1 и команды RUN на экране дисплея появится символ "?". После этого надо набрать два числа, разделенных запятой, например 2, 3. Тогда на экране дисплея появится текст:

5.000000000
ОСТАНОВ В СТРОКЕ 40

Непосредственный режим отличается тем, что нумерация строки не производится и после нажатия клавиши [ПС] набранный оператор или команда выполняются сразу. При этом набранный текст в ОЗУ не сохраняется.

Например:

LET A=3: LET B=4: LET C=5: PRINT (A+B)*C

После выполнения указанных действий на экране дисплея появится число

3. 500000000E 01

Режимы работы БЭЙСИК-интерпретатора можно чередовать в любом порядке, что очень удобно при отладке программ.

2.3. АЛФАВИТ ЯЗЫКА. ЗАПИСЬ ЧИСЕЛ И ПЕРЕМЕННЫХ

Алфавит языка БЭЙСИК состоит из 26 заглавных букв латинского алфавита, цифр от 0 до 9, символов

.	,	:	;	+	–	*	/	¬	>	<	=	(	)	#	␣ (пробел)

В заключенном в апострофы тексте можно использовать все символы клавиатуры дисплея, кроме двоеточия и символов забоя.

Для обозначения арифметических и логических операций используются символы: ¬ – возведение в степень; * – умножение; / – деление; + – сложение; – – вычитание; < – меньше; > – больше; = – равно; >< или <> – не равно; < = или =< – меньше или равно; > = или > = – больше или равно.

В БЭЙСИКе используется лишь один тип чисел – вещественный и две формы их записи: обычная (только вместо запятой ставится точка) и экспоненциальная. В последнем случае десятичная точка может быть поставлена в любом месте числа, но необходимо соответственно изменить и его порядок. Ниже приведены примеры возможных форм записи чисел на языке БЭЙСИК. (Буква Е используется вместо основания 10.)

Число	Возможные формы записи на БЭЙСИКе	
	обычная	экспоненциальная
3,457	3.457	3.457 E0
$-5\cdot10^4$	–50000	–5E4
$3{,}8\cdot10^{-5}$	0.000038	3.8 E–5
6	6	.6E1

Количество цифр в числе должно быть не более 12, его модуль не должен превышать $(1-10^{-12})\cdot10^{99}$, а порядок записывается целым числом. Константа π задается набором на клавиатуре символов "#PI".

Для обозначения переменных в БЭЙСИКе используются имена, состоящие из одного или двух символов, первый из которых обязательно должен быть буквой латинского алфавита, второй – цифрой, например A, B3, P6.

Переменным в процессе вычислений могут присваиваться различные значения.

2.4. АРИФМЕТИЧЕСКИЕ ВЫРАЖЕНИЯ И ОПЕРАТОР LET

При записи арифметических выражений используются символы операций, которые приведены в § 2.3. Последовательность выполнения действий естественная, слева направо. Для изменения ее используются круглые скобки в их обычном толковании.

Так, например, формула $\frac{ax^2 + bx + c}{d - 2{,}5}$ на БЭЙСИКе запишется следующим образом:

```
(A*X¬2+B*X+C)/(D-2.5)
```

Для присваивания переменным значений используется оператор присваивания LET, общая форма записи которого

НС LET $\alpha=\beta$

где НС – номер строки; α – имя переменной; β – арифметическое выражение.

Например:

```
50 LET A=(B+C)¬4
85 LET B6 = A2
```

В первой строке переменной A присваивается значение арифметического выражения $(B+C)^4$, во второй – переменной B6 присваивается значение переменной A2.

Ключевое слово LET можно не набирать на клавиатуре дисплея, так как его автоматически вставляет интерпретатор. Отметим, что для БЭЙСИКа (вариант 3) ЭВМ Д3-28 это правило не выполняется.

Использование термина "равно" при записи символа "=" не всегда корректно. Так, например, запись J=J+1 ошибочна с точки зрения математики, а в программировании она обычна и означает, что берется содержимое ячейки памяти, отведенной для переменной J, к нему прибавляется единица и результат помещается в ту же ячейку. Старое значение J при этом стирается. Вследствие вышесказанного предпочтение отдается термину "присваивание".

В арифметических выражениях можно использовать математические функции. Наиболее часто встречающиеся из них предусмотрены в БЭЙСИКе, их список приведен в прил. 3. Функция задается именем и указанием аргумента, который заключается в скобки. Аргументом может быть имя переменной или любое арифметическое выражение. Такие функции называют также *стандартными* (или *встроенными*).

Например, SIN(X¬2), COS(COS(Y)). Второе выражение показывает, что стандартные функции допускают рекурсию, т. е. обращение к самим себе. Программист может вводить собственные функции (более подробно об этом см. в § 2.9).

2.5. ПОНЯТИЕ О ВВОДЕ/ВЫВОДЕ

Рассмотрим простейшие приемы обмена информацией с ЭВМ (более подробно о вводе/выводе см. в § 2.11–2.13).

Для ввода значений переменных, которые являются исходными данными, служит оператор INPUT. Общая форма записи оператора:

НС INPUT 'текст' список

где НС – номер строки; 'текст'–произвольный текст, заключенный в апостро-

фы, который может быть опущен; "список" – имена переменных, разделенные запятыми.

При выполнении оператора INPUT вычисления приостанавливаются. На экран выводится текст, а при отсутствии текста и апострофов – символ "?". Программист последовательно вводит константы или арифметические выражения, разделяя их запятыми. Общее количество констант и арифметических выражений должно быть равно числу переменных в списке оператора INPUT. Ввод должен быть завершен нажатием клавиши ПС . Если вводится арифметическое выражение, то соответствующей переменной в операторе INPUT будет присвоен его результат.

Например:

```
10 INPUT 'ВВЕДИТЕ K,N 'K,N
20 INPUT Y
```

При выполнении первого оператора на экран выводится сообщение:

```
ВВЕДИТЕ K,N
```

После ввода двух чисел, разделенных запятой, начнет выполняться второй оператор INPUT, на экран выводится символ "?", после чего необходимо ввести значение Y .

Для вывода значений переменных и результатов вычислений служит оператор PRINT. Одна из форм записи оператора:

```
НС PRINT 'текст' список, 'текст' список,...
```

где 'текст' – произвольный текст, заключенный в апострофы; "список" – имена разделенных запятыми переменных, значения которых необходимо вывести на экран дисплея.

Например:

```
10 A=2.58:B=-.00289
20 PRINT A,B
30 END
```

На экран выводятся значения переменных A и B:

```
2.580000000      -2.890000000E-03
```

Изменим оператор в строке 20. Получим:

```
10 A=2.58: B=-.00289
20 PRINT 'A='A,'B='B
30 END
```

В этом случае на экран дисплея будут выведены следующие значения:

```
A=2.580000000      B=-2.890000000E-03
```

2.6. ЗАВЕРШЕНИЕ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Выполнение программы прекращается по достижении строки с максимальным номером, либо строк, содержащих операторы END и STOP, либо при наличии некорректных вычислений.

Оператор END служит для останова выполнения программы. Для этого можно использовать и оператор STOP, так как в БЭЙСИКе они эквивалентны.

Однако в целях обеспечения последующего использования программы (трансляция на других ЭВМ и т. п.) рекомендуется внутри ее записывать оператор STOP, а оканчивать программу оператором END.

2.7. ОПЕРАТОРЫ ПЕРЕДАЧИ УПРАВЛЕНИЯ

Последовательность выполнения операторов определяется нумерацией строк. Для изменения естественной последовательности выполнения операторов используются операторы безусловной и условной передачи управления: GOTO,ON и IF соответственно.

Общая форма записи оператора GOTO:

HC GOTO HC1

где HC – номер строки; HC1 – номер строки, которой передается управление.

Оператор

GOTO 65

означает переход к строке с номером 65.

В непосредственном режиме оператор GOTO можно использовать для начала выполнения программы или продолжения ее с любой строки. При этом следует помнить, что предыдущее состояние программы может оказать влияние на результат.

Оператор ON будет рассмотрен в § 2.14.

Оператор IF записывается в виде

IF $\alpha \otimes \beta$ THEN HC

где α , β – имена переменных или арифметические выражения; $\otimes$ – символ операции отношения; HC – номер строки, к которой переходит управление при выполнении условия $\otimes$. Если в программе нет строки с указанным HC, происходит переход к строке с ближайшим бо́льшим номером. Вместо слова THEN может стоять слово GOTO. После слова THEN вместо HC может быть записан любой другой исполнимый оператор БЭЙСИКа, их может быть несколько (сколько позволит строка). В этом случае при соблюдении условия $\otimes$ выполнится записанный (записанные) оператор и управление будет передано следующей строке.

Например:

```
85 IF X > = X1+8 THEN 30
```

Если X=10 и X1=0, управление будет передано 30-й строке, а если X1=5, – то строке, следующей за 85-й строкой.

Оператор IF позволяет строить циклические программы, т. е. такие, в которых однотипные вычисления повторяются многократно.

Пример 2.2. Вычислить произведение $1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$ (n!).

Возможный вариант программы:

```
10 INPUT N
20 I = 0
30 K= 1
40 I = I+1
50 K= K*1
60 IF I<N THEN 40
```

```
70 PRINT 'ФАКТОРИАЛ' N' = 'K
80 END
```

Участок программы от 40-й строки до 60-й включительно образует цикл, который повторяется до тех пор, пока значение I не станет равным N. При каждом повторении (шаге) значение переменной I увеличивается на единицу. Значение переменной K на каждом шаге является факториалом текущего значения переменной I. В результате выполнения этой программы на экране дисплея, например при N=5, появится текст:

ФАКТОРИАЛ 5.000000000 = 1.200000000E 02

Пример 2.3. Составить таблицу значений функции $y = \sin x + \cos x$.

Введем следующие обозначения: X – начальное значение аргумента, X1 – его конечное значение, H – шаг изменения аргумента.

Возможный вариант программы:

```
10 INPUT X,X1, H
20 IF X > X1 THEN 80
30 Y = SIN(X) + COS(X)
40 PRINT X, Y
60 X = X + H
70 GOTO 20
80 END
```

Цикл. Количество повторений определяется шагом H изменения аргумента X и пределом изменения аргумента X1.

Если X=0, X1=6,28, H=0,1, то первые пять пар значений X и Y будут выведены на экран в виде:

```
 .000000000        1.000000000
1.000000000E-01    1.094837582
2.000000000E-01    1.178735909
3.000000000E-01    1.250856696
4.000000000E-01    1.310479336
. . . . . . . . . . . . . . . . . .
```

Простейшие формы операторов LET, GOTO, IF, INPUT, PRINT дают программисту возможность написать любую программу, но для того чтобы она была "стройной и красивой", используется ряд других операторов, которые рассмотрены ниже.

2.8. МАССИВЫ

Наряду с простыми переменными в БЭЙСИКе используются переменные, объединенные в группу с одним именем. Эта группа называется *массивом*. В математике примерами таких групп являются векторы и матрицы.

Например, вектор A размерностью 10 содержит 10 элементов a_i – составляющих вектора. Матрица B размером 4×4 содержит 16 элементов b_{ij}. В обоих случаях индексы определяют порядковый номер элемента.

Основная цель введения массивов – упрощение массовой обработки данных. Так, для ссылки на конкретный элемент массива достаточно указать его имя и порядковый номер (индекс).

В БЭЙСИКе допускаются одномерные и двумерные массивы. Имена массивов записываются по правилам записи имен переменных. Элементы массива определяются именем массива и индексом, заключенным в скобки. В рассматриваемом варианте БЭЙСИКа элементы одномерного, а также строки и столбцы двумерного массивов нумеруются, начиная с нуля.

Например: A (0,0) – 1-й элемент двумерного массива A; M5 (K,L) – элемент массива M5, стоящий на пересечении $(k+1)$-й строки и $(l+1)$-го столбца.

В БЭЙСИКе не различаются нулевые элементы массивов и одноименные простые переменные, например A (0, 0) = A. Так как индексы могут принимать различные значения, необходимо заранее указать их предельные величины, чтобы ЭВМ могла распределить для элементов массива требуемый объем памяти. Однако общее количество элементов одного массива не должно превышать 256.

Описание массива производится в операторах DIM или COM с указанием в скобках максимальных для данной задачи значений индексов. (Некоторые особенности применения этих операторов описаны также в § 2.10, 2.18.)

Например:

```
 5 DIM A(6,9),C(8),B2(20)
10 COM P2(10),E(6,6)
```

Операторы DIM и COM должны описывать массивы до использования последних в вычислениях. Индексы массива могут записываться в виде констант, переменных и арифметических выражений. В последнем случае индексом считается целая часть вычисленного значения выражения.

Например: A (K/Y), при K=8 и Y=3 A (K/Y) соответствует A (2).

В последующих примерах для удобства чтения будем считать, что элементы массивов нумеруются, начиная с единицы. Аналогично поступают многие программисты, если их не ограничивает объем ОЗУ.

Пример 2.4. Элементам массива A размерностью 5 присвоить значения 1, 2, 3, 4, 5 соответственно, вычислить их сумму и вывести ее на печать.

```
10 DIM A(5)
20 INPUT A(1),A(2),A(3),A(4),A(5)
30 S=0: I=0
40 I=I+1
50 S=S+A (I)          ]   Вычисление суммы.
60 IF I<5 THEN 40     ]
70 PRINT 'S='S
80 END
```

Результат вычислений будет выведен в следующем виде:

```
S= 1.500000000E 01
```

Размерность массива можно задавать переменной, которая должна быть предварительно определена.

Например:

```
10 INPUT N
20 DIM A(N)
```

При выполнении оператора DIM или COM всем описанным в них элементам массивов автоматически присваиваются нулевые значения.

При программировании допускается повторное определение массивов, а также определение массива, имеющего имя ранее определенной простой переменной, но предварительно необходимо ликвидировать использованные области DIM и COM. Это можно осуществить с помощью операторов CLEARD для области DIM и CLEARC для области COM.

Пример 2.5. Элементам массива A присвоить значения, равные их номерам, и вывести эти значения на печать. Затем изменить размер массива и повторить вычисления.

Возможный вариант программы:

```
 10 INPUT N
 20 DIM A(N)
 25 LET I= 0
 30 LET I=I+1
 40 LET A(I)=I
 50 PRINT A(I)
 60 IF I<N THEN 30
 70 CLEARD              ]  Ликвидация области DIM.
 80 INPUT M
 90 DIM A(M)            ]  Повторное использование имени массива A.
 95 LET I=0
100 LET I=I+1
110 LET  A(I)=I
120 PRINT A(I)
130 IF I<M THEN 100
140 END
```

Если N=3 и M=4, то результатом вычислений будет:

```
1.000000000
2.000000000
3.000000000
1.000000000
2.000000000
3.000000000
4.000000000
```

Если операторы DIM и CLEARD заменить операторами COM и CLEARC соответственно, результат не изменится.

Следует помнить, что использование операторов DIM и COM в любых циклах запрещено и приводит к сообщению об ошибке. При необходимости построения такого цикла можно, например, использовать конструкцию с GOTO.

Отметим, что в расширенных версиях БЭЙСИКа существуют специальные операторы для действий над массивами.

2.9. ФУНКЦИИ ПОЛЬЗОВАТЕЛЯ

Наряду со стандартными функциями (см. § 2.4) в БЭЙСИКе можно использовать заранее определенные *функции пользователя.*

Определение функции пользователя аналогично заданию обычной функциональной зависимости $y = f(x)$. Пусть имеется выражение

$$f(x) = ax^2 + bx + c, \qquad (2.1)$$

где a, b, c – постоянные коэффициенты; x – формальный аргумент (формальный потому, что мы можем вместо него подставить любое число или арифметическое выражение). Вполне естественны записи:

$$f(y) = ay^2 + by + c;$$

$$f(4) = a \cdot 16 + b \cdot 4 + c;$$

$$f(a+b) = a(a+b)^2 + b(a+b) + c.$$

Выражение (2.1) дает общий вид функциональной зависимости, и, подставляя вместо формального аргумента x конкретные (фактические) аргументы (в данном случае $y, 4, a+b$), выполняем вычисления согласно формуле (2.1). Отметим, что формальный и фактический аргументы независимы.

Функции пользователя должны иметь имя из трех латинских букв, первые две из которых FN. Общая форма определения функции пользователя имеет вид

DEF FN$\alpha(\beta)=\gamma$

где FNα – имя функции; α – любая буква латинского алфавита; β – имя аргумента (формальный аргумент); γ – арифметическое выражение.

Функция пользователя должна быть определена один раз до ее использования. Формальный аргумент может быть только один: переменная.

Например:

```
DEF FNA(Y)=Y¬3+Y¬2+5
```

Эта запись определяет функцию с именем FNA, зависящую от формального аргумента Y, которая соответствует выражению $y^3 + y^2 + 5$. Имя формального аргумента Y автономно, т. е. переменные с такими же именами могут использоваться в программе и при этом не будут связаны с аргументом.

Пример 2.6. Вычислить значение функции $z = \dfrac{a^2+b}{c^2+b}(\cos^2 a+b)\ (b+25)$.

Возможный вариант программы:

```
10 DEF FNP(X)=X*X+B
20 INPUT' ВВЕДИТЕ ПЕРЕМЕННЫЕ A,B,C 'A,B,C
30 Z=FNP(A)*FNP(COS(A))*FNP(5)/FNP(C)
40 PRINT 'Z='Z
50 END
```

Если A=1, B=2, C=3, то на экране дисплея после выполнения программы появится следующий текст:

```
ВВЕДИТЕ ПЕРЕМЕННЫЕ A,B,C 1,2,3
Z= 1.687691392E 01
```

Арифметическое выражение, стоящее справа от знака равенства в операторе DEF , может содержать любые определенные к моменту вычисления функции пользователя, кроме самой определяемой функции. Это выражение может включать или не включать формальный аргумент.

Например:

```
DEF FNB(X)=X*X-3
DEF FNA(Z) = (SIN(X) +COS(Z))*FNB(5)
DEF FNC(H)=SIN(X)+COS(X)
```

Неформальные переменные, например переменная B в примере 2.6, должны быть определены к моменту использования функции.

Функции пользователя допускают рекурсии.

Например:

```
10 DEF FNA(X)=X+2
```

```
20 PRINT FNA(FNA(2))
30 END
```

Отметим, что в расширенных версиях БЭЙСИКа можно определять функции многих переменных.

2.10. ЦИКЛЫ

Циклические вычисления, т. е. такие, при которых выполнение некоторых участков программы повторяется многократно, уже рассматривались в § 2.7, где они определялись с помощью оператора IF (см. примеры 2.3–2.5). Для упрощения организации циклов с заданным числом повторений могут использоваться специальные операторы FOR и NEXT.

Начало цикла записывается в виде

FOR $\alpha=\beta$ то γ STEP δ

где α – имя управляющей переменной; β, γ – арифметические выражения, задающие соответственно начальное и конечное значения α; δ – арифметическое выражение, определяющее приращение управляющей переменной при очередном выполнении операторов цикла.

Оператор NEXT завершает цикл и записывается в виде

NEXT α

где α – имя управляющей переменной цикла, записанной в операторе FOR.

В приведенной конструкции вычисления выполняются в такой последовательности, что участок программы, заключенный между операторами FOR и NEXT, повторяется многократно. Сначала α принимает значение β и выполняются операторы, следующие за оператором FOR. Когда вычислительный процесс доходит до оператора NEXT, значение α изменяется на величину δ, управление передается оператору, следующему за оператором FOR, и вычисления повторяются. Так продолжается до тех пор, пока значение α остается меньше значения γ (при положительном δ) или больше его (при отрицательном δ). В противном случае управление передается следующему после NEXT оператору, т. е. происходит выход из цикла.

Пример 2.7. Составить таблицу значений функции $y = \sin x + \cos x$.

Пусть X – начальное значение аргумента, X1 – его конечное значение, H – шаг изменения аргумента (ср. с примером 2.3).

Возможный вариант программы:

```
10 INPUT X,X1,H
20 FOR A=X TO X1 STEP H
30 Y=SIN(A)+COS(A)
40 PRINT A,Y
60 NEXT A
70 END
```

A – управляющая переменная, она же используется как аргумент.

Результаты вычислений будут такие же, как в примере 2.3.

Если значение γ такое, что при $\alpha = \beta$ справедливо условие выхода из цикла, операторы внутри цикла не выполняются.

Если в операторе FOR отсутствует слово STEP, то БЭЙСИК-интерпретатор устанавливает шаг, равный единице.

Пример 2.8. Вычислить $n!$

Возможный вариант программы:

```
10 INPUT N
20 K=1
30 FOR I=1 TO N
40 K=K*I
50 NEXT I
60 PRINT 'ФАКТОРИАЛ 'N; '='K
70 END
```

Результаты вычислений будут такие же, как в примере 2.2.

Задание нулевого шага приводит к сообщению об ошибке. При выходе из цикла значение управляющей переменной (I в примере 2.8) равно последнему, использованному в теле цикла, ее значению (N в примере 2.8). С помощью операторов GOTO и IF можно выходить из цикла раньше, чем он закончится по изменении управляющей переменной. Вход в цикл в обход оператора FOR недопустим. Если цикл не выполняется ни разу, за управляющей переменной сохраняется начальное значение.

Внутри одного цикла может быть другой, который называется *вложенным*. Он должен полностью содержаться внутри первого цикла. Допускается до семи уровней вложения.

Пример 2.9. Вычислить произведение двух квадратных матриц $C = A \times B$, использовав формулу $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$, где $i = \overline{1, n}$; $j = \overline{1, n}$.

Пусть

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}; \quad B = \begin{bmatrix} 10 & 11 & 12 \\ 13 & 14 & 15 \\ 16 & 17 & 18 \end{bmatrix}.$$

Такие значения элементов матриц взяты для удобства ввода.

Возможный вариант программы:

```
 10 DIM A(3,3),B(3,3),C(3,3)
 15 LET K=0
 20 FOR I=1 TO 3
 25 FOR J=1 TO 3
 30 LET K=K+1: LET A(I,J)=K: LET B(I,J)=K+9
 35 NEXT J
 40 NEXT I
 50 FOR I=1 TO 3
 60 FOR J=1 TO 3
 70 LET C(I,J)=0
 80 FOR K=1 TO 3
 90 LET C(I,J)=C(I,J)+A(I,K)*B(K,J)
100 NEXT K
110 NEXT J
120 NEXT I
130 FOR J=1 TO 3
140 PRINT C(J,1), C(J,2), C(J,3)
150 NEXT J
160 END
```

В этих двух циклах вычисляются значения элементов матриц A и B.

3-й цикл. 2-й цикл. 1-й цикл.

Цикл для вывода результата на экран.

На экране появятся результаты в следующем виде:

```
8.400000000E 01   9.000000000E 01   9.600000000E 01
2.010000000E 02   2.160000000E 02   2.310000000E 02
3.180000000E 02   3.420000000E 02   3.660000000E 02
```

Отметим, что в расширенных версиях БЭЙСИКа имеются операторы цикла с выходом по условию.

2.11. ВВОД ДАННЫХ

Кроме рассмотренных ранее операторов INPUT и LET (см. § 2.4, 2.5), значения переменным можно присваивать, используя блок данных.

Блоком данных называется последовательность записанных в ОЗУ чисел. Запись производится с помощью оператора DATA. Операторов DATA в программе может быть несколько. Общий вид этого оператора:

DATA $\alpha, \beta, \gamma, \ldots$

где α , β , γ ,... – числа или арифметические выражения (в последнем случае используются результаты этих выражений) в количестве не менее одного.

Например:

```
10 X=3: Y=7
20 DATA 2,X+Y,8.2
30 A=X+ Y
50 DATA 64,7
```

Оператор в строке 20 записывает в блок данных три числа: 2, 10, 8.2. Оператор в строке 50 дополняет блок еще двумя числами: 64 и 7. В результате в блоке данных будет записана последовательность чисел: 2, 10, 8.2, 64. 7.

Оператор DATA должен быть единственным или последним в строке. Считывание данных из блока производится оператором READ. Общая форма записи этого оператора:

READ $\alpha, \beta, \gamma, \ldots$

где α, β , γ ,... – имена переменных или элементов массивов, которым присваиваются значения из блока данных.

При выполнении оператора READ из блока данных последовательно слева направо считываются числа, и их значения присваиваются переменным, указанным в операторе READ, в порядке их следования. В программе может быть несколько операторов READ.

Например:

```
10 DATA 2,6,4,-8
20 DATA -7,9
60 READ X(5,8)
70 READ Y,Z(4)
80 READ A2,A3,A4
```

В результате выполнения указанных операторов READ элементу массива X(5, 8) будет присвоено значение 2, переменной Y – значение 6, элементу массива Z(4) – значение 4, переменным A2, A3, A4 – значения –8, –7, 9 соответственно.

Пример 2.10. Вычислить сумму элементов массива A1 (10) (ср. с примером 2.4).

Возможный вариант программы:

```
 10 DIM A1(10)
 20 DATA 0,1,-1,2,4.5,-.8E-12
 30 DATA -3,-0.5,-1,8.E-13
 40 FOR I=1 TO 10
 50 READ A1(I)
 60 NEXT I
 70 S=0
 80 FOR I=1 TO 10
 90 S=S+A1(I)
100 NEXT I
110 PRINT 'S='S
120 END
```

Присваивание элементам массива А численных значений.

Вычисление суммы.

После выполнения программы на экране дисплея появится строка

```
S= 2.000000000
```

Участок программы после 30-й строки можно сократить, объединив циклы:

```
35 S=0
40 FOR I=1 TO 10
50 READ A1(I)
60 S=S+A1 (I)
70 NEXT I
80 PRINT 'S='S
90 END
```

С блоком данных непосредственно связан хранящийся в ОЗУ ДЗ-28 так называемый *указатель блока данных.* Перед считыванием первого числа он указывает на него в блоке данных, а после этого – на второе число и т.д. Если в блоке данных чисел больше, чем имен переменных в операторе READ, то после его выполнения указатель блока данных указывает на число, следующее за последним считанным. При выполнении очередного оператора READ считывание чисел из блока данных начинается с того места, где остановился указатель. Если в блоке данных чисел меньше, чем переменных в операторах READ, выдается сообщение об ошибке.

При использовании операторов DATA и READ всегда нужно помнить о состоянии указателя, который изменяет свое состояние по мере считывания чисел из блока данных.

В любом месте программы указатель блока данных можно возвратить в начальное состояние командой RESTORE, после чего считывание снова начнется с первого числа блока.

Например:

```
 10 DATA 3,4,5,6,9,8
 20 READ X,Y,Z
100 RESTORE
110 READ Y,Y,Z
```

В 100-й строке фрагмента программы указатель блока данных устанавливается в исходное положение, а в 110-й строке переменные Y и Z принимают значения 4 и 5 соответственно. При отсутствии оператора RESTORE переменные Y и Z приняли бы значения 9 и 8.

2.12. УПРАВЛЕНИЕ ПЕЧАТЬЮ. ФОРМАТЫ ПЕЧАТИ

Печать (под этим понимается и вывод на экран дисплея) результатов вычислений, пояснительных текстов и графиков осуществляется с помощью оператора PRINT (см. § 2.5).

Числа, переменные, элементы массивов и арифметические выражения при последовательной записи в списке оператора PRINT должны быть обязательно записаны с разделителями. Текст может записываться без разделителей. *Разделителями списка* являются запятая и точка с запятой. Они же служат для управления печатью.

Если в качестве разделителя используется запятая, то вся строка печати длиной в 100 позиций условно разделяется на 5 зон по 20 позиций. Каждая запятая вызывает перемещение к началу следующей зоны, т. е. значение первой переменной начинает печататься с 1-й позиции, второй переменной – с 21-й, третьей – с 31-й и т. д. Если в текущей строке не хватает места, печать продолжается в следующей строке.

Например:

```
10 DIM A(3),B(3),C(3)
20 FOR I=1 TO 3
30 A(I)=I: B(I)=I+3: C (I)=I+6
40 NEXT I
50 FOR I=1 TO 3
60 PRINT A(I),B(I),C(I)
70 NEXT I
80 END
```

Вычисление элементов массивов A,B,C.

В результате на экране дисплея появится текст:

```
1.000000000    4.000000000    7.000000000
2.000000000    5.000000000    8.000000000
3.000000000    6.000000000    9.000000000
```

Для более компактной печати в качестве разделителя используется точка с запятой. При этом после печатания значения переменной происходит сдвиг по строке печати вправо на одну позицию.

Если 60-ю строку в приведенном выше фрагменте программы заменить строкой

```
60 PRINT A(I); B(I); C(I)
```

на экран дисплея будет выведен следующий результат:

```
1.000000000 4.000000000 7.000000000
2.000000000 5.000000000 8.000000000
3.000000000 6.000000000 9.000000000
```

Интервалы в четыре позиции между числами вызваны резервированием места для печатания порядка числа. Подобным образом выходные данные будут располагаться, если разделять элементы списка текстом в апострофах.

Например:

```
10 A=5: B=8: C=-2
20 PRINT 'A='A'B='B'C='C
30 END
```

На экране дисплея появится текст:

A= 5.000000000 B= 8.000000000 C= –2.000000000

Список оператора PRINT может содержать спецификации формата, которые существенно расширяют возможности управления печатью. *Форматом* называется порядок размещения символов в данных.

В БЭЙСИКе реализован формат печати трех видов, которые задаются в списке оператора PRINT спецификацией, расположенной между восклицательными знаками.

По спецификации !E! число выводится в форме с плавающей запятой, т. е. знак числа, точка, двенадцать цифр мантиссы, символ E, знак порядка и две цифры порядка:

.XXXXXXXXXXXXE YY

Знак "+" во всех формах вывода не печатается. Если порядок числа равен нулю, вместо E YY выводятся пробелы.

Например:

```
10 LET A=625
20 PRINT !E! 'КОРЕНЬ ИЗ' A'='SQR(A)
30 END
```

На экран дисплея будет выведен следующий результат:

КОРЕНЬ ИЗ .625000000000E 03= .250000000000E 02

По спецификации !n_1.n_2 !на печать выводится знак числа, n_1 цифр до десятичной точки, n_2 цифр после десятичной точки:

$$\underbrace{\pm XXX...X}_{n_1}.\underbrace{XXX...X}_{n_2}$$

Незначащие нули в целой части числа заменяются пробелами. Если число не может быть напечатано в заданном формате, то в позициях, отведенных для этого числа, будут выведены символы "*". Каждое из значений n_1 и n_2 не должно превышать числа 9.

Например:

```
10 LET A=42.81: B=–587.4: C=2348566.01: D=5467.9
20 PRINT !3.2! A,B,C
30 PRINT D
40 END
```

На экран дисплея будет выведен следующий результат:

```
42.81        –587.40        *******
*******
```

По спецификации !Fn_1 .n_2! на печать выводится знак числа, n_1 цифр до десятичной точки, n_2 цифр после десятичной точки, буква E, знак порядка и две цифры порядка:

XXX...X.XXX...XE YY

Незначащие нули и буква E при нулевом порядке числа заменяются пробелами. Любое число в этом формате занимает $n_1 + n_2 + 6$ позиций.

Например:

```
10 LET A=14
20 PRINT !F2.4! A*A
30 END
```

На экран дисплея будет выведен следующий результат:

```
19.6000E 01
```

Если спецификация не определена программистом, то после запуска БЭЙСИК-интерпретатора автоматически задается формат !F1.9.!. Эта информация хранится в определенном месте ОЗУ Д3-28 (в регистре Т14), поэтому, если указать в операторе PRINT другой формат, он запишется на место старого и сохранится для всех операторов PRINT до тех пор, пока не будет указан новый формат.

Например:

```
10 A=1: B=2: C=3: E=4
20 PRINT !2.1!A
30 PRINT B,C
40 PRINT !E! 'ПРОВЕРКА'
50 PRINT E
60 END
```

После выполнения программы на экране дисплея появится следующий текст:

```
 1.0
 2.0          3.0
ПРОВЕРКА
 .400000000000E 01
```

Пример 2.11. Напечатать таблицу значений функции $y = \sin x + \cos x$.

Возможный вариант программы:

```
 10 DIM Y(100),X2(100)
 20 INPUT X,X1,H
 30 K=(X1-X)/H                         ] Вычисление количе-
 40 FOR I=1 TO K                        ства шагов цикла.
 50 Y(I)=SIN(X)+ COS(X)
 60 X2(I)=X                             В массиве Y хранят-
 70 X=X+H                               ся значения функции.
 80 NEXT I                              В массиве X2 хранят-
 90 PRINT ' ТАБЛИЦА ЗНАЧЕНИЙ ФУНКЦИЙ'   ся значения аргумента.
100 PRINT 'НОМЕР ШАГА', 'АРГУМЕНТ', 'ФУНКЦИЯ'
110 FOR I=1 TO K
120 PRINT !3.0!' 'I,!F1.5!X2 (I),Y(I)   Вывод всех вычис-
130 NEXT I                              ленных значений.
140 END
```

При X=0, X1=6,28, H=0,1 имеем такие же результаты, как в примерах 2.3, 2.7. Они будут выведены на экран дисплея в следующем виде:

```
     ТАБЛИЦА ЗНАЧЕНИЙ ФУНКЦИИ
НОМЕР ШАГА          АРГУМЕНТ          ФУНКЦИЯ
     1              .00000            1.00000
     2              1.00000E-01       1.09484
.........................................................
```

Отметим, что в расширенных версиях БЭЙСИКа имеются операторы для вывода массивов.

2.13. ВЫВОД ГРАФИКОВ НА ПЕЧАТЬ. ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ УПРАВЛЕНИЯ ПЕЧАТЬЮ

Оператор PRINT можно использовать для печатания графиков. Для этого в списке оператора PRINT необходимо указать ключевое слово TAB, затем в круглых скобках арифметическое выражение и любой символ, заключенный в апострофы, которым будет изображаться график. Вывод начинается с позиции, координата которой равна целой части значения выражения, записанного для слова TAB. Значения арифметического выражения не должны выходить из диапазона 0–100.

Пример 2.12. Напечатать график функции $y = x^2$, где x изменяется от –7 до 7 с шагом 2.

Возможный вариант программы:

```
10 FOR X=–7 TO 7 STEP 2
20 PRINT TAB (X¬2)'*'
30 NEXT X
40 END
```

На экране появится изображение:

```
                                                        *
                                    *
                  *
*
*
                  *
                                    *
                                                        *
```

Чтобы график был наглядным, обычно приходится вводить масштабные коэффициенты, преобразующие диапазон значений функции в диапазон числа символов в строке используемого внешнего устройства.

Пример 2.13. Напечатать график функции $y = \sin x$, где x изменяется от 0 до 2π с шагом 0,1.

Возможный вариант программы:

```
10 K=#PI*2
15 INPUT M
20 FOR X=0 TO K STEP .1
30 PRINT TAB(M+M*SIN(X))'*'
40 NEXT X
50 END
```

Масштабный коэффициент M устанавливает диапазон значений функции от –M до M, а слагаемое M смещает нулевую ось на M позиций от левого края экрана. Если M=40, синусоида займет по ширине весь экран, а если M=20, то левую половину экрана.

После вывода на печать элементов списка оператора PRINT на дисплей, как правило, подается команда перевода строки. Она не выполняется, если:

1) список завершен разделителями запятая или точка с запятой, а также спецификацией формата; 2) последним элементом списка вывода является слово TAB с сопутствующим выражением.

Пустой оператор PRINT также вызывает перевод строки.

Например:

```
10 A=1: B=2: C=3: D=4: E=5: F=6
20 PRINT A,B
30 PRINT C,
40 PRINT D;
50 PRINT E
60 PRINT TAB(20)
70 PRINT F
80 END
```

В результате на экран дисплея будут выведены следующие результаты:

```
1.000000000     2.000000000
3.000000000     4.000000000     5.000000000
                6.000000000
```

Если в приведенном выше фрагменте программы строку 60 заменить строкой

```
60 PRINT TAB(20) '*'
```

то перевод строки уже будет осуществляться, на экране дисплея появится текст:

```
1.000000000     2.000000000
3.000000000     4.000000000     5.000000000
                *
6.000000000
```

Отметим, что в расширенных версиях БЭЙСИКа имеются специальные операторы для вывода графической информации.

2.14. СЛОЖНЫЕ ВЕТВЛЕНИЯ

Для организации ветвления программы по многим направлениям используется оператор ON. Общий вид этого оператора:

ON выражение

где "выражение" – любое арифметическое выражение, причем целая часть значения последнего является номером строки, к которой осуществляется переход. При отсутствии в программе строки с полученным номером управление передается строке с ближайшим бо́льшим номером. Значение выражения должно лежать в интервале 1–7999.

Например:

```
 5 FOR I=40 TO 20 STEP -10
10 ON I
20 PRINT 'БЭЙСИК',
25 GOTO 50
30 PRINT 'ИЗУЧАЕМ',
35 GOTO 50
40 PRINT 'МЫ',
50 NEXT I
60 END
```

В результате на экран дисплея будет выведен следующий результат:

МЫ ИЗУЧАЕМ БЭЙСИК

Отметим, что в расширенных версиях БЭЙСИКа оператор ON имеет структуру, близкую к вычисляемому GOTO в ФОРТРАНе.

2.15. ПОДПРОГРАММЫ

Часто встречающиеся однотипные вычисления можно оформлять в виде *подпрограмм*. Это экономит труд программиста, так как вместо многократного программирования одинаковых участков записываются лишь обращения к подпрограмме, в которой содержится такой участок.

Для организации подпрограмм используют оператор обращения к подпрограмме GOSUB и оператор возврата RETURN. Общая форма записи оператора:

```
GOSUB HC
```

где HC – номер строки, с которой начинается подпрограмма.

Оператор GOSUB передает управление строке, номер которой указан после имени оператора. При этом запоминается адрес обращения к подпрограмме.

Подпрограмма, которая начинается со строки, указанной в операторе GOSUB, должна оканчиваться оператором RETURN. При выполнении оператора RETURN управление передается оператору, следующему за оператором обращения к подпрограмме.

Пример 2.14. Вычислить величины $l!$, $m!$, $k!$ и сформировать матрицу A размерности n, где $a_1=1!$, $a_2=2!$, ..., $a_n=n!$.

Возможный вариант программы:

```
 5 INPUT K,L,M,N
10 DIM A(N)
15 LET J=K
20 GOSUB 90
25 PRINT 'K! ='B
30 LET J=L
35 GOSUB 90
40 PRINT 'L! ='B
45 LET J=M
50 GOSUB 90
55 PRINT 'M! ='B
60 FOR I=1 TO N
65 LET J=I
70 GOSUB 90
75 LET A(I)=B
80 PRINT 'A('! 2.0! I') ='!F1.9! A(I)
85 NEXT I
87 STOP
90 LET B=1
95 FOR C=1 TO J
100 LET B=B*C
105 NEXT C
110 RETURN
115 END
```

Вычисление и печать K!.

Вычисление и печать L!

Вычисление и печать M!.

Вычисление и печать элементов массива A.

Подпрограмма, вычисляющая B=J! .

Если K=2, L=3, M=4, а N=10, на экран дисплея будет выведен следующий результат:

```
K!= 2.000000000
L!= 6.000000000
M!= 2.400000000E 01
A( 1)= 1.000000000
A( 2)= 2.000000000
. . . . . . . . . . . . . .
A(10)= 3.628800000E 06
```

Приведенная программа верно работает и без оператора STOP, но в этом случае выводится сообщение № 42 об ошибке, на которое можно не обращать внимание.

Если в программе нет строки с указанным после GOSUB номером, управление передается строке с ближайшим бо́льшим номером.

Внутри подпрограммы можно использовать переход к другой подпрограмме, которая называется в этом случае *подпрограммой низшего уровня*. Максимальное количество таких уровней 16.

Пример 2.15. Пусть в процессе вычислений необходимо многократно определять значения функции

$$f(x) = \begin{cases} a \text{ при } x \geqslant 0; \\ e \text{ при } x = 0, \end{cases}$$

где a, e – натуральные числа. Кроме того, в программе встречаются вычисления $(f(x))!$.

Возможный вариант программы:

```
 5 INPUT K,M,X,P,D
10 LET A=K: LET E=M
15 GOSUB 70
20 PRINT ! 2.0.'F('X')='! F1.9! J;' ГДЕ A='K' E='M
25 LET A=P: LET E=D
30 GOSUB 40
35 PRINT ! 2.0! '(F('X'))! ='F1.9! B;' ГДЕ A='P' E='D
37 STOP
40 LET B=1
45 GOSUB 70
50 FOR C=1 TO J
55 LET B=B*C
60 NEXT C
65 RETURN
70 IF X>=0 THEN 85
75 LET J=E
80 GOTO 90
85 LET J=A
90 RETURN
95 END
```

Первая подпрограмма, вычисляющая B=J!. Для вычисления J эта подпрограмма в 45-й строке обращается ко второй подпрограмме.

Вторая подпрограмма, вычисляющая

$$J = \begin{cases} A \text{ при } X \geqslant 0; \\ E \text{ при } X < 0. \end{cases}$$

Эта подпрограмма имеет 2-й уровень по отношению к первой.

При K=2, M=3, X=–1, P=4, D=5 на экран дисплея будет выведен следующий текст:

```
F(-1)= 3.000000000        ГДЕ A= 2.000000000     E= 3.000000000
(F(-1))=  1.200000000E 02 ГДЕ A= 4.000000000     E= 5.000000000
```

2.16. КОММЕНТАРИИ

В любом месте БЭЙСИК-программы можно записывать *строки-комментарии,* которые начинаются оператором REM. Все дальнейшие символы строки не обрабатываются интерпретатором. Текст строки не должен содержать символов забоя и апострофов.

Строки-комментарии используются для пояснения текста программы. Так, в примере 2.15 первую подпрограмму можно выделить в тексте программы:

```
38 REM ПОДПРОГРАММА ВЫЧИСЛЕНИЯ (F(X))!
40 B=1
45 GOSUB 70
50 FOR C=1 TO J
55 B=B*C
60 NEXT C
65 RETURN
```

2.17. ПРЕКРАЩЕНИЕ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Запланированный останов программы задается операторами STOP и END (см. § 2.6). Прервать выполнение программы, используя клавиатуру дисплея, можно следующими способами:

1) удерживая нажатой клавишу [СУ] (для дисплеев 15ИЭ200×140-017 – клавишу [УПР]), нажать клавишу [П/Р];

2) нажать клавишу [АР1] и после печати сообщения об останове отпустить ее.

Для продолжения счета по программе с точки прерывания нужно одновременно нажать клавиши [СУ] и [Б/В], а затем клавишу [ОЧС].

В процессе обмена информацией между ЭВМ и дисплеем удобно прерывать программу, отпустив клавишу [ЛИН] ("ЛИНИЯ СВЯЗИ"), а продолжать, – нажав эту клавишу.

2.18. ИСПОЛЬЗОВАНИЕ НМЛ

Язык БЭЙСИК позволяет использовать магнитную ленту как для хранения данных, так и для хранения текстов программ или их сегментов (отдельных частей). Программы и данные помещаются на МЛ в виде записей по 256 байтов с двукратным дублированием. На одной стороне кассеты может быть размещено около 400 записей.

При записи текста программы на МЛ БЭЙСИК-интерпретатор формирует программный файл, состоящий из следующих записей:

Заголовок файла	Записи текста программы (256 байтов)	256 байтов...256 байтов	Закрывающая запись файла

Под *файлом* понимается организованный каким-либо образом набор данных, в том числе программа или ее сегменты.

В заголовке программного файла записывается его имя. Для записи программного файла используется оператор SAVE. Общая форма записи оператора:

SAVE 'α' НС1, НС2

где α – имя файла – произвольный текст; НС1, НС2 – соответственно первая и последняя строки сегмента программы, который будет записан на МЛ.

Например:

SAVE 'ФАЙЛ 1'10,400 – оператор записывает программный файл с именем ФАЙЛ 1, содержащий строки программы с 10-й по 400-ю;

SAVE 5,250 – оператор записывает безымянный файл, содержащий строки с 5-й по 250-ю;

SAVE – оператор записывает на МЛ всю БЭЙСИК-программу, находящуюся в оперативной памяти.

При наличии в операторе SAVE одного номера строки на экран будет выведено сообщение об ошибке.

Оператор SAVE END помещает на МЛ служебную запись, которая является признаком конца ленты (не физического конца, а конца программных файлов). Этот признак служит ограничителем при считывании программы или данных.

Пример 2.16. Сохранить текст программы на МЛ в программном файле под именем: "ВЫЧИСЛЕНИЕ ФАКТОРИАЛА ЧИСЛА".

Программу для вычисления N! возьмем из примера 2.8:

```
10 SAVE 'ВЫЧИСЛЕНИЕ ФАКТОРИАЛА ЧИСЛА'
20 INPUT N
30 K=1
40 FOR I=1 TO N
50 K=K*I
60 NEXT I
70 PRINT 'ФАКТОРИАЛ 'N,'='K
80 SAVE END
```

Программа работает следующим образом: после команды RUN выполняется оператор в 10-й строке, который записывает на МЛ имя файла и исходный текст; затем – следующие операторы, вычисляющие N!;последним выполняется оператор в 80-й строке, который записывает признак конца МЛ.

Рассмотренный пример решается несколько иначе:

```
10 INPUT N
20 K=1
30 FOR I=1 TO N
40 K=K*I
50 NEXT I
60 PRINT 'ФАКТОРИАЛ 'N'='K
70 END
SAVE 'ВЫЧИСЛЕНИЕ ФАКТОРИАЛА ЧИСЛА'
SAVE END
```

Здесь команда RUN не набирается и выполнение программы не производится. Двумя последними операторами набранный текст записывается в программный файл с именем

"ВЫЧИСЛЕНИЕ ФАКТОРИАЛА ЧИСЛА", который завершается записью признака конца МЛ.

Если нужно вернуть МЛ в исходное состояние (перемотать), пользуются оператором REWIND.

Считывание с МЛ и загрузка в ОЗУ программных файлов производится оператором LOAD. Общая форма записи этого оператора:

LOAD 'a' HC1, HC2

где a – имя файла; HC1, HC2 – номера строк.

Форма использования оператора LOAD аналогична форме использования оператора SAVE.

Если имя в операторе LOAD не задано, производится загрузка очередного программного файла. Сравнение имен заданного и считываемого файлов ведется по шести первым символам имени (включая пробелы).

При отсутствии в записи оператора LOAD номеров строк считывается вся программа. Задание только одного номера строки считается ошибкой.

Например, необходимо прочитать с МЛ текст программы из примера 2.16 (при условии, что он записан на МЛ).

Для этого достаточно в непосредственном режиме набрать команды:

REWIND
LOAD' ВЫЧИСЛЕНИЕ ФАКТОРИАЛА ЧИСЛА'

Первый оператор осуществит перемотку МЛ к ее началу, второй поместит требуемый текст в оперативную память. Текст прочитанной программы можно вызвать на экран дисплея командой LIST.

Оператор LOAD в этом примере можно записать короче:

LOAD 'ВЫЧИСЛ'

При выполнении оператора LOAD HC1,HC2 производятся следующие действия:

1) из текста программы исключаются строки с HC1 по HC2;

2) начиная с текущего положения МЛ, ищется заголовок программного файла с заданным именем;

3) текст записанной программы загружается в ОЗУ;

4) по прочтении закрывающей записи файла управление передается строке, следующей за строкой оператора LOAD (если этот оператор использовался в программном режиме) с ближайшим бо́льшим номером.

Если при считывании файла возникают ошибки, снова ищется заголовок программы, что позволяет дублировать программу внутри одного файла.

Если программа на МЛ не найдена до записи "КОНЕЦ ЛЕНТЫ", выполнение оператора LOAD завершается.

Имя загруженного файла может быть напечатано оператором PRINT, содержащим в списке ключевое слово OPEN; при этом перед именем будет напечатан символ "P".

Например, на МЛ в файле с именем ПРОБА записан текст:

```
20 INPUT N
30 LET K=1
40 FOR I=1 TO N
```

```
50 LET K=K*I
60 NEXT I
70 PRINT 'ФАКТОРИАЛ 'N; '='K
```

Составим следующую программу:

```
10 LOAD 'ПРОБА' 20, 70
80 PRINT OPEN
90 END
```

Эти операторы можно выполнить и в непосредственном режиме.

При запуске программы для N=5 получим:

```
ФАКТОРИАЛ 5.000000000   =1.200000000E 02
P ПРОБА
```

Если загружается несколько файлов, то ключевое слово OPEN выводит на печать имя последнего считанного с МЛ файла.

Пример 2.17. Пусть необходимо прочитать имена файлов, записанных на МЛ. Для этого можно воспользоваться следующей программой:

```
 5 INPUT N
10 FOR K=1 TO N
15 LOAD 40,300
20 PRINT OPEN
30 NEXT K
```

Пусть на МЛ записаны два безымянных файла и два файла с именами АСА и П231. При N=4 после завершения программы на экран дисплея будет выведен следующий текст:

```
P
P
P ACA
P П231
```

Напомним, что для данного примера в операторе LOAD первый номер строки должен быть больше 30

При записи числовых данных на МЛ формируется блок данных, состоящий из следующих записей:

...	Начало блока	Записи данных	Конец блока	...

Запись блока данных производится оператором

```
DATA SAVE
```

содержащим список данных. Элементами списка могут быть арифметические выражения и массивы. Элементы списка разделяются запятыми.

Например: оператор

```
DATA SAVE P( )
```

записывает на МЛ блок данных, содержащих массив P; оператор

```
DATA SAVE B(3), SIN(X+Y)*Z,B( ),C
```

записывает в файл на МЛ третий элемент массива B, результат арифметического выражения, все элементы массива B и переменную C.

Блоки можно объединять в файлы данных. Файл данных организуется путем записи заголовка оператором

DATA SAVE OPEN 'имя файла'

и конца файла данных оператором

DATA SAVE END

Например:

```
20 DATA SAVE OPEN 'ПРИМЕР'
30 DATA SAVE X( )
40 DATA SAVE Y( )
50 DATA SAVE Z( )
60 DATA SAVE END
```

Файл данных под именем "ПРИМЕР" состоит из трех блоков, содержащих массивы X, Y, Z соответственно.

Если бы вместо строк 30–50 была записана строка

DATA SAVE X( ), Y( ), Z( )

то файл состоял бы из одного блока, содержащего все три массива.

Считывание данных с МЛ и присваивание переменным прочитанных значений производится оператором

DATA LOAD список,

где "список" содержит арифметические выражения, массивы, переменные.

Например, оператор

DATA LOAD A,B(2,7),C( )

считывает с МЛ и присваивает переменной A, элементу массива B (2,7) и элементам массива C последовательно расположенные значения из загруженного с МЛ блока данных. Считывание блока данных производится с текущего положения МЛ.

Выполнение рассматриваемого оператора завершается в следующих случаях:

1) после считывания очередного блока всем элементам списка DATA LOAD присвоены значения;

2) прочитана запись "КОНЕЦ ФАЙЛА ДАННЫХ";

3) прочитана запись "КОНЕЦ ЛЕНТЫ".

Поиск на МЛ первого блока файла данных производится оператором

DATA LOAD 'a'

где a – имя файла данных.

Сравнение имен производится по первым шести символам. Если заголовок файла данных с нужным именем не найден, то выполнение оператора завершается по прочтении записи "КОНЕЦ ЛЕНТЫ".

Очередной заголовок может быть прочитан оператором

DATA LOAD OPEN

Имя файла может быть напечатано оператором PRINT со служебным словом OPEN. При этом перед именем будет выведен символ "D".

Пример 2.18. Вычислить элементы массива A (у которого $a_i = i$), записать их значения на МЛ в файл "ПРИМЕР", ленту перемотать, изменить значения двух элементов массива A, присвоить значения элементам массива A из файла данных "ПРИМЕР".

Возможный вариант программы:

```
10 DIM A(5)
20 FOR I=1 TO 5
30 LET A(I)=I
40 NEXT I
45 PRINT A(1), A(5)
50 DATA SAVE OPEN' ПРИМЕР '
60 DATA SAVE A( )
70 DATA SAVE END
80 REWIND
90 LET A(1)=-1: LET A(5)=-5
95 PRINT A(1),A(2)
100 DATA LOAD 'ПРИМЕР'
105 PRINT OPEN
110 DATA LOAD A( )
115 PRINT A(1),A(5)
120 END
```

В результате на экран дисплея будет выведен следующий результат:

```
  1.000000000        5.000000000
 -1.000000000       -5.000000000
 D ПРИМЕР
  1.000000000        5.000000000
```

Программные файлы, отдельные блоки и файлы данных при считывании с МЛ можно пропустить с помощью оператора SKIP. Для пропуска заданного числа блоков данных после ключевого слова SKIP записывается выражение, определяющее количество пропускаемых блоков.

Например, оператор SKIP 2 осуществляет пропуск двух блоков данных. Этот оператор выполняется путем считывания записей с МЛ и подсчета количества записей "КОНЕЦ БЛОКА".

Пропуск заданного числа файлов задается записью буквы F после выражений, определяющих количество файлов.

Например, оператор SKIP 3F осуществляет пропуск трех файлов.

Анализ различных вариантов завершения действия операторов LOAD, DATA LOAD, SKIP можно производить операторами:

IF END THEN – анализирует запись "КОНЕЦ ЛЕНТЫ";
IF ENDF THEN – анализирует запись "КОНЕЦ ФАЙЛА ДАННЫХ".

Если при считывании информации с МЛ прочитана соответственно запись "КОНЕЦ ЛЕНТЫ" или "КОНЕЦ ФАЙЛА ДАННЫХ", то управление передается оператору, следующему за словом THEN , или строке с указанным номером.

Например, если данные записаны аналогично примеру 2.18, можно составить следующую программу:

```
10 INPUT N
20 DIM A(N)
30 DATA LOAD A( )
40 IF ENDF THEN 70
50 PRINT ! 2.0! 'ПРОЧИТАНО'N'ЭЛЕМЕНТОВ,A ('N')='! F1.9! A(N)
60 STOP
70 PRINT 'ПРОЧИТАНО ПЯТЬ ЭЛЕМЕНТОВ A (5)='A(5)
90 END
```

Если при выполнении программы задать N=8, на экране дисплея появится текст:

ПРОЧИТАНО ПЯТЬ ЭЛЕМЕНТОВ МАССИВА A (5) = 5.000000000

Если задать N = 4, то имеем:

ПРОЧИТАНО 4 ЭЛЕМЕНТОВ, A (4) = 4.000000000

Анализ записи "КОНЕЦ ЛЕНТЫ" позволяет записать более рациональную, чем в примере 2.17, программу для считывания имен файлов на МЛ:

```
7990 LOAD 1,7000
7992 IF END THEN 7999
7994 PRINT OPEN
7995 GOTO 7990
7999 END
```

В приведенной программе не требуется указывать число файлов.

2.19. НЕКОТОРЫЕ ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ

Здесь мы рассмотрим другие формы оператора CLEAR и приведем некоторые сведения об операторах PRINT, DIM и COM, которые по методическим соображениям не были указаны ранее.

Оператор

```
CLEARP HC1,HC2
```

позволяет удалить из ОЗУ строки программы с номерами с HC1 по HC2. Оператор

```
CLEAR HC1,HC2
```

выполняет те же функции, но вызывает еще и перемотку МЛ. Если в этих двух операторах указан один номер строки, то удаляется только эта строка. Если номера строк отсутствуют, то оператор CLEARP игнорируется, а CLEAR вызывает только перемотку МЛ.

Оператор LIST можно использовать для вывода не только всего текста программы, но и ее фрагментов, если использовать формы записи LIST HC1, HC2 и LIST HC.

В область DIM помещаются массивы и простые переменные, а в область COM – массивы и адреса функций пользователя.

Оператор CLEARF служит для сброса указателя уровней циклов, а оператор CLEARS – для сброса указателей уровня подпрограмм.

Для вывода информации на ПМ используется оператор

```
PRINT#1
```

по тем же правилам, которые справедливы для оператора PRINT. Указатель устройства вывода информации сохраняет свое значение подобно указателю формата. Поэтому, чтобы вернуть вывод на экран дисплея, необходимо использовать оператор

```
PRINT#0
```

2.20. ИСПОЛЬЗОВАНИЕ ПЕРФОЛЕНТЫ

Для работы с перфолентой (ПЛ) используются операторы SAVE, LOAD, DATA SAVE, DATA LOAD. Правила использования аналогичны правилам ра-

боты с МЛ. Отличие состоит в том, что для ввода/вывода на ПЛ после названий операторов нужно указать символ #1.

Например:

LOAD #1, DATA SAVE #1 C( )

Заметим, что для перевода выполнения операций ввода/вывода на МЛ можно указать символ #0 либо опустить его.

Вывод программ на ПЛ осуществляется оператором SAVE #1, а считывание с ПЛ – оператором LOAD #1. В обоих случаях передается вся программа.

Вывод данных на ПЛ производится оператором

DATA SAVE #1 список

а считывание с ПЛ – оператором

DATA LOAD #1 список

Перфорация ленты выполняется в кодах КОИ-7 (см. прил. 4). Разделителями строк служат символы ПС или ВК. Программа должна быть завершена символом ПФ (код 00 12).

Числовые данные на ПЛ разделяются запятой, а числа представляются в любом допустимом для БЭЙСИКа виде. Блок данных на ПЛ должен завершаться символом ПФ.

ЭВМ выводит на ПЛ данные по формату !E!.

3. РАБОТА С БЭЙСИК-ИНТЕРПРЕТАТОРОМ

3.1. ОБЩИЕ СВЕДЕНИЯ ОБ ИНТЕРПРЕТАТОРЕ

БЭЙСИК-интерпретатор – это программа, составленная в машинных командах и предназначенная для преобразования операторов БЭЙСИК-программы в машинные команды с последующим их выполнением. Такие программы-переводчики называют *трансляторами.*

Трансляторы подразделяют на интерпретаторы и компиляторы. *Компиляторы* преобразовывают программы, составленные на алгоритмических языках высокого уровня, например ФОРТРАНе или ПЛ/1, в программы на машинно-ориентированном языке и размещают их на временное или постоянное хранение в ОЗУ либо на внешних носителях информации. *Интерпретатор* также преобразовывает операторы языка высокого уровня, например БЭЙСИКа, в машинные команды, но не размещает их на временное или постоянное хранение. Если какой-либо оператор БЭЙСИК-программы должен выполняться k раз, то БЭЙСИК-интерпретатор k раз будет преобразовывать этот оператор в машинные команды. Интерпретаторы занимают меньший объем ОЗУ, чем компиляторы, но выполняют программы пользователей дольше.

БЭЙСИК-интерпретатор записан на МЛ и поставляется в комплекте с Д3-28. Для обеспечения работы интерпретатора его необходимо загрузить в ОЗУ Д3-28. При выключении Д3-28 текст БЭЙСИК-интерпретатора в ОЗУ не сохраняется.

В настоящее время существует несколько вариантов БЭЙСИК-интерпретаторов. Они различаются по объему занимаемой памяти и своим возможностям. Все версии интерпретаторов устроены в принципе однотипно. В дальнейшем будем отмечать конкретно, для какой версии рассмотрен пример.

3.2. ЗАГРУЗКА И ЗАПУСК ИНТЕРПРЕТАТОРА

Рассматриваемая в данном параграфе последовательность действий программиста предполагает использование специальной служебной программы, которая существенно облегчает процесс загрузки БЭЙСИК-интерпретатора в ОЗУ Д3-28 (текст этой служебной программы и пояснения по ее использованию приведены в § 4.6). Отметим, что указанная программа может работать с разными версиями интерпретатора, у которых различаются контрольные суммы кодов команд. С 45-го по 50-й шаг надо записывать КП для загружаемой версии. На МЛ должен быть ракорд (прозрачный участок длиной 15–20 см, расположенный в самом начале МЛ и в конце ее). На МЛ записи располагаются в следующем порядке:

< СЛУЖЕБНАЯ ПРОГРАММА > , < СЛУЖЕБНАЯ ПРОГРАММА > , < СЛУЖЕБНАЯ ПРОГРАММА>,< БЕЙСИК-ИНТЕРПРЕТАТОР>,<БЭЙСИК-ИНТЕРПРЕТАТОР> , <БЭЙСИК-ИНТЕРПРЕТАТОР> .

Для подготовки Д3-28 к работе с алгоритмическим языком БЭЙСИК надо выполнить следующие действия:

1) подключить к Д3-28 внешние устройства;

2) выключить дисплей, если он был включен;

3) включить тумблер [СЕТЬ], расположенный на задней стенке Д3-28 справа. При этом в регистрах X и Y Д3-28 должны высвечиваться нули. Если это не так, необходимо выключить Д3-28 и не менее чем через 10–15 с включить ее еще раз;

4) вставить в лентопротяжное устройство НМЛ кассету с БЭЙСИК-интерпретатором. Для этого необходимо*:

а) нажать до упора на крышку лентопротяжного механизма и отпустить ее; крышка вместе с кассетным блоком должна выдвинуться;

б) вставить кассету в пазы кассетного блока до упора таким образом, чтобы видимая часть МЛ была сверху;

в) осторожно нажимая на крышку, проследить, чтобы зубцы обеих катушек кассеты свободно вошли в соответствующие пазы осей приемного и подающего валов НМЛ.

Отметим, что, *если зубцы кассеты не вошли в пазы валов НМЛ, не стоит прикладывать чрезмерных усилий, закрывая крышку лентопротяжного механизма. Это приводит к поломке кассеты. При несовпадении пазов и зубцов следует провернуть вал кратковременным нажатием на клавишу* [◀◀] *или* [▶▶] *, находящуюся под лентопротяжным механизмом. Затем повторить п. "в"*;

г) убедившись, что кассета установлена правильно, нажать на крышку с обеих сторон до ее фиксации;

5) нажать клавишу [P] Д3-28;

6) нажать клавишу [C] Д3-28, перемотать МЛ в исходное состояние командой 12 00 (REW), для чего необходимо:

а) нажимая клавиши [80], [40], [20], [10] Д3-28, добиться свечения индикаторов только над клавишами [80] и [40];

б) нажать клавишу [00], расположенную правее клавиш, указанных в п. "а";

7) нажать клавишу [СЛ] Д3-28. При этом происходит кратковременное считывание служебной программы. Если данная служебная программа прочиталась с ошибкой, на что указывает мигание регистров X и Y Д3-28, необходимо нажать клавишу [C] Д3-28 и повторить п. 7;

8) нажать клавишу [S] Д3-28. При этом происходит загрузка в ОЗУ Д3-28 БЭЙСИК-интерпретатора. Индикация погашена;

9) после останова МЛ в регистрах X и Y высвечивается КП команд БЭЙСИК-интерпретатора, что сигнализирует о завершении его считывания в ОЗУ;

---

*В некоторых моделях Д3-28 возможна установка лентопротяжных механизмов с другим способом установки кассеты.

10) включить клавишу СЕТЬ , расположенную на передней панели дисплея (здесь имеется в виду дисплей 15ИЭ-00-013);

11) нажать клавишу С Д3-28;

12) нажать и зафиксировать в нижнем положении клавиши ЛИН , ДУП и РЕД дисплея;

13) нажать клавишу S Д3-28. При этом происходит запуск БЭЙСИК-интерпретатора. Индикация регистров X и Y гаснет, включается индикатор ПУ внизу слева от блока индикации Д3-28. После запуска интерпретатора на экране дисплея появляется текст:

БЭЙСИК Д3-28, ВАРИАНТ 3А
СНИМИТЕ КАССЕТУ!

14) снять кассету с БЭЙСИК-интерпретатором; дать подтверждение нажатием клавиши ПС дисплея. На экран дисплея выводится текст:

НОМЕРА ВНЕШНИХ ПОДПРОГРАММ

15) ввести с клавиатуры дисплея список номеров внешних подпрограмм*, нажать клавишу ПС дисплея. На экране дисплея появляется текст:

ПОСТАВЬТЕ КАССЕТУ С ПОДПРОГРАММАМИ < список >

16) поставить кассету с внешними подпрограммами и нажать клавишу ПС . Происходит загрузка в ОЗУ Д3-28 текстов внешних подпрограмм. Если при этом внешние подпрограммы не загружаются, необходимо нажать клавишу ШН , перемотать командой 12 00 (REW) МЛ к началу и повторить все действия, начиная с п. 13;

17) при отсутствии заранее подготовленных внешних подпрограмм в машинных командах Д3-28 нажать клавишу ПС . На экран дисплея выводится сообщение:

ГОТОВ
:

которое указывает на готовность микроЭВМ к работе с алгоритмическим языком БЭЙСИК.

Загрузка других версий интерпретатора такая же, за исключением того, что может быть запрошен номер внешнего устройства, которое в данное время подключено к Д3-28.

3.3. РАСПРЕДЕЛЕНИЕ ПАМЯТИ ИНТЕРПРЕТАТОРОМ

Программа БЭЙСИК-интерпретатора состоит из собственно интерпретатора, таблицы адресов внутренних подпрограмм и программы начального диало-

* Работа с внешними подпрограммами и правила их оформления приведены в § 3.6. Если нет внешних подпрограмм, то пп. 15,16 надо пропустить и перейти сразу к п. 17.

га. Распределение ОЗУ непосредственно после загрузки с МЛ интерпретатора с КП=132107 показано на рис. 3.1.

При запуске интерпретатора управление передается программе начального диалога, которая вначале переписывает таблицы адресов внутренних подпрограмм и устанавливает нужные значения в регистрах служебной зоны. Далее у программиста запрашиваются сведения об используемых внешних подпрограммах (если они есть). Во время начального диалога внешние подпрограммы программиста загружаются последовательно, друг за другом, в свободную зону ОЗУ, начиная с адреса, указанного в регистре R03. Например, для интерпретатора с КП=132107 этот адрес равен 2.03.00.00 (08963), для интерпретатора с КП=157071 – 2.07.00.06 (09990).

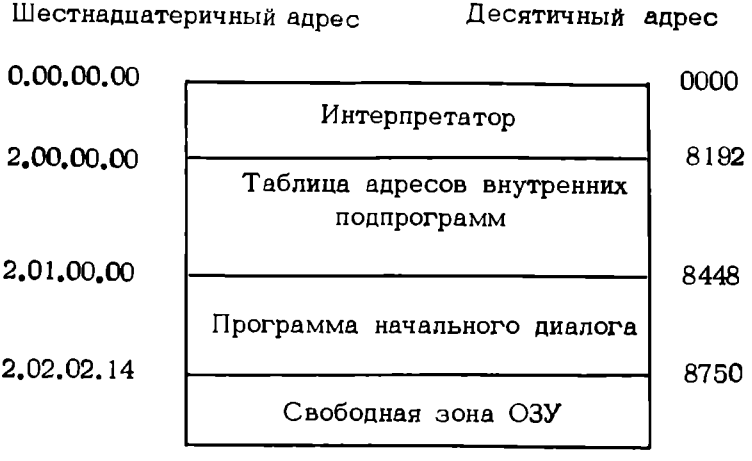


Рис. 3.1

После загрузки внешних подпрограмм таблица адресов внутренних подпрограмм и программа начального диалога стираются, а на их место переписываются внешние подпрограммы, начиная с адреса 2.00.00.00 (08192) для интерпретатора с КП=132107.

Начальные адреса внешних подпрограмм записываются в таблицу, нижней границей которой является адрес, указанный в ячейке 7.13.08.14 (32140) таблицы внутренних подпрограмм.

Каждой внешней подпрограмме в таблице соответствует ячейка из четырех байтов. Два старших байта занимает номер внешней подпрограммы, а два младших байта содержат ее начальный адрес.

Указатель начала программы пользователя, расположенный в ячейке 7.13.08.12 (32138), устанавливается на адрес первого свободного шага после внешних подпрограмм, а начальное состояние указателя стека (R13), соответствующее верхней границе таблицы адресов внешних подпрограмм, записывается в ячейку 7.13.11.10 (32186).

Если внешних подпрограмм нет, указатель начала программы пользователя устанавливается равным 2.00.00.00 (08192) для интерпретатора с

КП=132107, а начальное состояние указателя стека – 7.11.00.00 (31488). Распределение ОЗУ после начального диалога показано на рис. 3.2.

В своей работе интерпретатор использует некоторые регистры. Информация об используемых интерпретатором регистрах приведена в табл. 3.1.

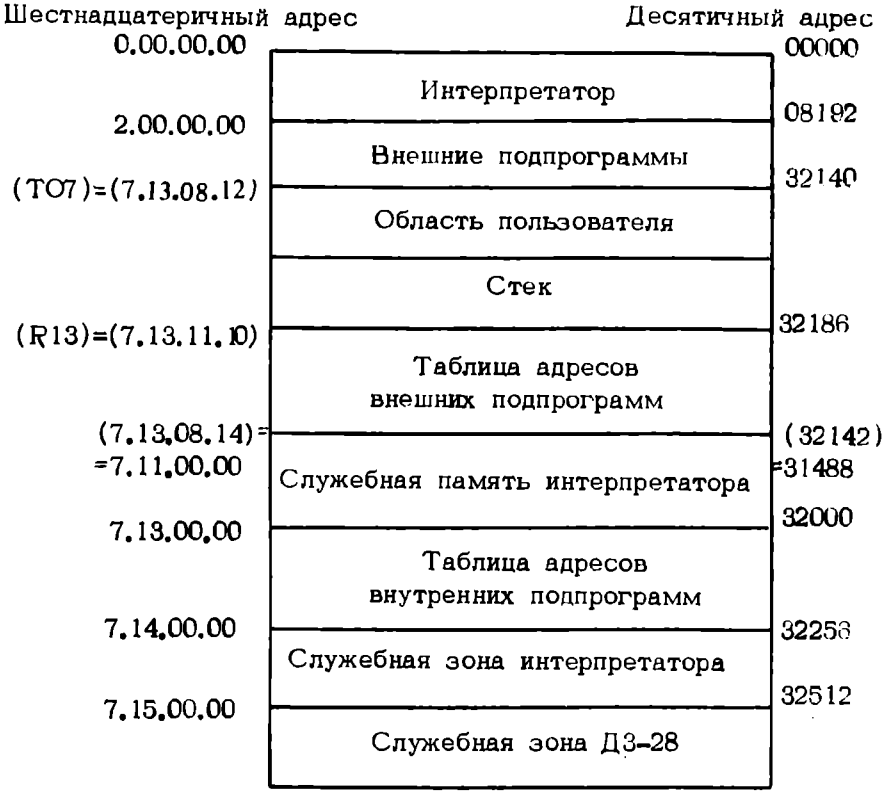


Рис. 3.2

Рассмотрим более подробно область пользователя. В ней хранится текст (тексты) исходной программы, переменные, массивы, указатели функций пользователя. Структура этой области приведена на рис. 3.3.

В области COM размещаются переменные и массивы, объявленные оператором COM, указатели функций пользователя FN. Расширение области производится вставкой данных в ее начало.

В области DIM размещаются массивы, объявленные оператором DIM, простые (неиндексированные) переменные, определенные любыми операторами, кроме операторов COM. Расширение области производится вставкой данных в конец ее.

Служебная память интерпретатора состоит из буфера МЛ (адреса с 7.11.00.00 по 7.11.15.15) и буфера ввода строки (адреса с 7.12.00.00 по 7.12.15.15).

Таблица 3.1

Регистр	Назначение регистра
R00	Константа нуль (0)
R01	Текущий адрес в тексте БЭЙСИК-программы
R02	Константа два (2)
R03	Указатель границы области пользователя
R13	Указатель стека
S03	Очередной символ текста программы
T01	Номер текущей строки
T02	Начало областей данных
T03	Конец области COM
T04	Указатель блока DATA в тексте программы
T05	Счетчик позиций печати
T06	Флаг прерывания
T07	Начало программы пользователя
T08	Указатель стека подпрограмм типа GOSUB
T09	Указатель стека циклов FOR-NEXT
T10	Режим интерпретатора
T11	Текущий номер записи на МЛ
T14	Формат печати PRINT
T15	Счетчик строк (для контроля сохранности интерпретатора)

Регистр

(T07)	Текст исходной программы
(T02)	Область COM
(T03)	Область DIM
(R03)	Резервная зона
(R13)	Указатель стека

Рис. 3.3

Служебная зона интерпретатора состоит из стека подпрограмм (адреса с 7.14.00.00 по 7.14.03.15), стека циклов (адреса с 7.14.04.00 по 7.14.14.15) и двух десятичных ячеек генератора случайных чисел (адреса с 7.14.15.00 по 7.14.15.15).

На рис. 3.4 приведена таблица имен внутренних подпрограмм. В каждой клетке таблицы записано имя внутренней подпрограммы. Над именем помещен адрес той ячейки ОЗУ, с которой начинается текст указанной подпрограммы в шестнадцатеричной системе счисления, а под именем подпрограммы ука-

7.13.00.00

(32000)	00	02	04	06	08	10	12	14
00								
01								
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
12	0115 1108 LET 8120	0003 1006 READ 934	0103 0206 GOTO 4902	0102 1510 GOSUB 4858	0002 0210 RETURN 554	0105 1014 IF 5540	0101 1008 FOR 4520	0102 0909 NEXT 4761
13	0104 1502 INPUT 5362	0010 1410 DATA 2794	0106 0704 DIM 5748	0106 0614 COM 5742	0012 1100 PRINT 3248	0110 0506 LOAD 6742	0109 1102 SAVE 6578	
14	0107 0610 CMD 5994		0115 0306 SKIP 7990			0015 1007 REWIND 4007	0104 0703 DEF 5235	0002 0008 END 520
15	0002 0008 STOP 520	0015 1504 LIST 4084	0010 1506 REM 2806	0103 0614 RUN 4974	0010 0302 RESTORE 2610	0105 0508 CLEAR 5464	0107 0402 CALL 5954	0003 0412 ON 844

Рис. 3.4

зан этот же адрес в десятичной системе счисления. Таблица имен приведена частично, поскольку в различных интерпретаторах она отличается в деталях.

3.4. ПРЕДСТАВЛЕНИЕ ИСХОДНОЙ ПРОГРАММЫ И РАЗМЕЩЕНИЕ ДАННЫХ В ОЗУ

Операторы БЭЙСИК-программ размещаются в ОЗУ Д3-28, начиная с адреса, указанного в регистре Т07, причем в памяти записываются не операторы и ключевые слова, а только их коды (табл. 3.2). Пробелы, цифры, другие символы кодируются в соответствии с прил. 4.

Элементы массивов, объявленные оператором DIM или COM, и простые (неиндексированные) переменные, объявленные только оператором COM, хранятся в области данных в виде записей однотипного формата. Каждая запись состоит из имени (два байта), определителя (два байта) и соответствующего числа восьмибайтных десятичных ячеек, необходимых для хранения значений элементов массивов, или одной восьмибайтной десятичной ячейки (для хранения простой переменной).

Имя массива или простой переменной может состоять из буквы латинского алфавита и цифры. Если цифра отсутствует, ее заменяют кодом 00 00. Определитель записи состоит из двух байтов, в которых содержатся в шестнадцатеричной системе счисления максимальные значения индексов массивов. В старшем байте записан первый индекс.

Рассмотрим примеры размещения в ОЗУ одномерного массива X5 (3), двумерного массива У (2,2) и простой переменной А. Значения элементов массива X5: X5 (1) = –20; X5 (2) = 100; X5 (3) = 0,01. Значения элементов массива Y: Y (1,1) = –1; Y (1,2) = 8; Y (2,1) = 4,5; Y (2,2) = 7. Простая переменная А = –0,006. На рис. 3.5–3.7 приведены примеры размещения этих данных в ОЗУ: на рис. 3.5 – запись простой переменной А; на рис. 3.6 – запись массива X5 (3); на рис. 3.7 – запись массива Y (2,2).

Таблица 3.2

Ключевое слово	Код	Ключевое слово	Код	Ключевое слово	Код
CALL	05 12	RETURN	12 08	LOG(	10 08
CLEAR	15 10	REWIND	14 10	NEXT	12 14
CMD	14 00	RND(	10 12	ON	15 14
COM	13 06	RUN	15 06	OPEN	11 06
COS(	09 02	SKIP	14 04	PRINT	13 08
DATA	13 02	SAVE	13 12	READ	12 02
DEF	14 12	SGN(	08 08	REM	15 04
DEG(	08 00	SIN(	08 02	RESTORE	15 08
DIM	13 04	HSN(	08 10	STEP	11 02
END	14 14	HTN(	08 12	STOP	15 00
EXP(	10 02	IF	12 10	SQR(	09 08
EXT(	10 10	INPUT	13 00	TAB	11 10
FOR	12 12	INT(	10 04	#PI	10 14
FN	11 04	LET	12 00	TAN(	08 04
GOSUB	12 06	LGT(	10 06	THEN	11 08
GOTO	12 04	LIST	15 02	TO	11 00
HCS(	09 10	LOAD	13 10		

0401	0000	Имя A
0000	0000	Определитель 0,0
0000	0000	
0006	0000	
0000	0000	
0101	0003	

Рис. 3.5

0508	0305	Имя X5
0003	0000	Определитель 3,0
0000	0000	X5(0)=X5
0000	0000	
0000	0000	
0000	0000	
0002	0000	X5(1)
0000	0000	
0000	0000	
0100	0002	
0100	0000	X5(2)
0000	0000	
0000	0000	
0000	0002	
0001	0000	X5(3)
0000	0000	
0000	0000	
0001	0001	

Рис. 3.6

В область данных, объявленную оператором COM, заносятся записи-указатели функций пользователя. Каждая запись состоит из шести байтов: имени функции пользователя (два байта), формального параметра (два байта) и адреса определения функции в тексте программы (два байта). В качестве последнего принят абсолютный адрес первого символа, следующего за знаком равенства в операторе DEF. Формат записи-указателя функции пользователя DEF FNA(Z1)=SIN(Z1)+Z1¬2 приведен на рис. 3.8, а структура служебной записи в стеке FOR-NEXT – на рис. 3.9.

0509	0000	Имя Y
0002	0002	Определитель 2,2
Y(0,0)		
Y(0,1)		
Y(0,2)		
Y(1,0)		
Y(1,1)		
Y(1,2)		
Y(2,0)		
Y(2.1)		
Y(2,2)		

Рис. 3.7

Иногда требуется найти начальный адрес размещения БЭЙСИК-программы в ОЗУ. Ранее отмечалось, что в регистре T07 находится начальный адрес программы пользователя (см. табл. 3.1). Рассмотрим более детально, как определить начальный адрес программ пользователя. Предположим, что в ОЗУ занесена БЭЙСИК-программа

```
20 INPUT X
30 LET Y=X*X
40 PRINT X,Y
50 STOP
```

Пусть в ОЗУ загружен БЭЙСИК-интерпретатор с КП=157071. На клавиатуре Д3-28 надо набрать следующую последовательность кодов машинных команд:

```
14 13  07 04 MOV T07,R04
04 13  04 04 MOVD R04,X
12 13        JMP @X
```

В регистре X индицируется номер начального адреса размещения БЭЙСИК-программы в ОЗУ Д3-28. Переходя в режим "В", можно просмотреть данный текст БЭЙСИК-программы. Строка 20 закодируется в следующем виде:

```
09994  03 02    2
09995  03 00    0
09996  13 00    INPUT
09997  02 00    ПРОБЕЛ
09998  05 08    X
09999  00 10    ПС
```

Левый столбец цифр означает адрес ОЗУ, следующие 4 цифры – коды символов в соответствии с табл. 3.2 и прил. 4. Справа показаны соответствующие БЭЙСИК-операторы и разделители.

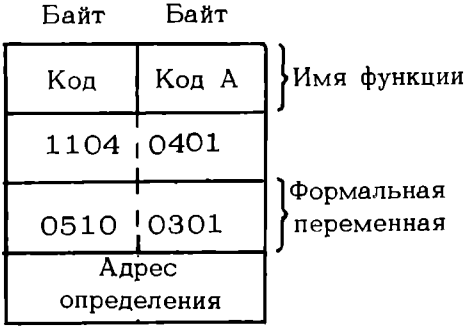


Рис. 3.8

Рис. 3.9

3.5. ВНУТРЕННИЕ ПОДПРОГРАММЫ ИНТЕРПРЕТАТОРА

В состав интерпретатора входят такие внутренние подпрограммы, которые пользователи могут применять для облегчения работы, например с внешними подпрограммами. Эти внутренние подпрограммы описаны ниже.

Внутренние подпрограммы позволяют производить обмен значениями между переменными БЭЙСИК-программы и регистром X Д3-28, дают возможность внешним подпрограммам использовать стек интерпретатора и т. д.

Обращение ко всем внутренним подпрограммам осуществляется одинаково:

10 13 B2 A2 (ISTT B2 A2)

где B2 A2 – код подпрограммы. Все внутренние подпрограммы рассчитаны на (BD)=0 и (BP)=0. Перечислим эти подпрограммы и укажем правила их использования.

1. Подпрограмма 10 13 06 12 (ИМЯ ПЕРЕМЕННОЙ) считывает из текста БЭЙСИК-программы имя переменной, начиная с адреса, указанного в регистре R01, и заносит это имя в регистр R09. Если адрес регистра R01 указывает не на букву латинского алфавита, на экран дисплея выводится сообщение об ошибке с номером 15. После выполнения подпрограммы в регистр R01 заносится адрес первого символа, следующего за именем переменной. Подпрограмма использует регистры R08 и R09.

2. Подпрограмма 10 13 04 14 (ВЫЗОВ ПЕРЕМЕННОЙ) ищет в области данных переменную или массив с именем, указанным в регистре R09, и вызывает в регистр X значение указанной переменной. Если переменной с указанным именем в области данных нет, на печать или экран дисплея выводится сообщение об ошибке с номером 123. Регистр R01 перед обращением к подпрограмме должен указывать на адрес первого символа, следующего за именем переменной. После выполнения подпрограммы в регистр S03 заносится первый символ, следующий за именем переменной или за закрывающей скобкой (при задании элемента массива). Регистр R01 указывает на следующий адрес. Подпрограмма использует все регистры.

3. Подпрограмма 10 13 06 08 (АДРЕС ЭЛЕМЕНТА) ищет в области данных переменную или массив с именем, указанным в регистре R09. Если переменной или массива с указанным именем в области данных нет, в регистр R08 заносится нуль. Регистр R01 перед обращением к подпрограмме должен указывать на адрес символа, следующего в БЭЙСИК-программе за именем переменной. Если за именем переменной следует элемент массива, производится считывание индексных выражений и вычисление адреса указанного элемента. Уменьшенное на 2 значение адреса элемента массива заносится в регистр R08, первый символ после закрывающей скобки – в регистр S03. Регистр R01 указывает на следующий адрес в программе. Если ищется неиндексированная переменная, в регистр R08 заносится адрес ее определителя, в регистр S03 – первый символ после имени переменной, а регистр R01 указывает на следующий адрес в программе. При поиске элементов массива используются все регистры, а при поиске простой переменной – регистры R04, R08, R09, R10, R11, R12.

4. Подпрограмма 10 13 00 06 (ПОИСК ПЕРЕМЕННОЙ) считывает из текста БЭЙСИК-программы имя переменной, начиная с адреса, указанного в регистре R01, ищет в области данных переменную или массив с указанным именем и заносит в регистр R08 адрес определителя этой переменной. Если переменной или массива с указанным именем в области данных нет, в регистр R08 заносится нуль. Если задан элемент массива, то в регистр R08 заносится уменьшенный на 2 адрес указанного элемента массива. После выполнения подпрограммы в регистр R10 заносится имя переменной, в регистр S03 – символ, следующий за именем простой переменной или за закрывающей скобкой для элементов массива, а содержимое регистра R01 указывает на следующий адрес. Подпрограмма использует все регистры.

5. Подпрограмма 10 13 06 14 (ФОРМИРОВАНИЕ ПЕРЕМЕННОЙ) формирует в области данных ОЗУ новую неиндексированную переменную с именем, указанным в регистре R10. После выполнения подпрограммы указатель границы области данных пересчитывается. В регистр R03 заносится (R03)+12. Значение сформированной переменной и ее определитель принимаются равными нулю. Подпрограмма использует регистры R08 и R10.

6. Подпрограмма 10 13 01 12 (ЗАПИСЬ ЗНАЧЕНИЯ) записывает в восьмибайтную ячейку по адресу (R08)+2 число из регистра X. После выполнения подпрограммы в регистр R08 заносится (R08)+8. Регистр R08 указывает на адрес последней двухбайтной ячейки, занятой записанным числом. Подпрограмма использует регистры R04, R05, R06, R07, R08.

7. Подпрограмма 10 13 02 14 (ЗАПИСЬ В СТЕК) записывает в стек, указателем которого является регистр R13 , число из регистра X. Число представляется в восьмибайтном десятичном формате. Указатель стека при этом пересчитывается. В регистр R13 заносится значение (R13)–8. Подпрограмма использует регистры R04, R05, R06, R07, R10. Эту подпрограмму применяют, когда при выполнении внешних подпрограмм необходимо запомнить какие-то промежуточные значения, а в служебной зоне нет свободных регистров. Необходимо помнить, что после выхода из внешней подпрограммы значение указателя стека должно быть восстановлено таким же, как и при входе в нее.

8. Подпрограмма 10 13 02 00 (ВЫЗОВ ИЗ СТЕКА) вызывает в регистр X число из стека. Указатель стека находится в регистре R13. Число в стеке должно быть записано в восьмибайтном десятичном формате. Для записи в стек следует использовать подпрограмму 10 13 02 14. После выполнения подпрограммы указатель стека пересчитывается. В регистр R13 заносится значение (R13)+8. Подпрограмма использует регистры R04, R05, R06, R07, R10.

9. Подпрограмма 10 13 00 10 (ВВОД СТРОКИ) принимает посимвольно информацию с клавиатуры основного устройства ввода/вывода или же с ПЛ в зависимости от состояния младшего байта регистра T10. Если младший байт равен единице, ввод осуществляется с ПЛ, если нулю,– то с основного устройства ввода/вывода (дисплея или ПМ "Consul 260.1"). Прием информации производится в буфер ввода, начинающийся с адреса 7.12.00.00 (31744). Если принят код 00 13 (ВК) или 00 10 (ПС), прием заканчивается. Длина вводимой строки не должна превышать длины буфера, т. е. 256 байтов. Старший бит принятого кода (контроль на четность) обнуляется при приеме. Подпрограмма использует регистры R09, R10, R12.

10. Подпрограмма 10 13 06 06 (ВЫВОД СИМВОЛОВ) выводит посимвольно информацию из ОЗУ, начиная с адреса, указанного в регистре R10, и по адрес, на котором записан код 00 00. Если младший байт регистра T10 равен нулю, вывод информации производится на дисплей или ПМ "Consul 260.1", если нет, – на ПЛ.

11. Подпрограмма 10 13 07 12 (ВЫЧИСЛИТЬ АРИФМЕТИЧЕСКОЕ ВЫРАЖЕНИЕ) производит вычисление арифметического выражения, записанного в ОЗУ в кодах ПМ "Consul 260.1" (ГОСТ 13052–74), без дополнения до четности. Перед обращением к подпрограмме в регистре R01 должен быть указан адрес первого символа вычисляемого арифметического выражения. Все переменные и функции пользователя, используемые в арифметическом выражении, должны быть определены до обращения к подпрограмме вычисления. Результат вычислений заносится в регистр X. После окончания вычислений в регистр S03 заносится первый символ, не входящий в арифметическое выражение, а содержимое регистра R01 указывает на адрес следующего за выражением символа. Подпрограмма использует все регистры.

12. Подпрограмма 10 13 11 14 (ОБЪЕМ ОЗУ) вызывает в регистр R09 указатель объема ОЗУ, который равен 07 00 для ОЗУ 32 К. Подпрограмма использует регистр R12.

3.6. РАБОТА С ВНЕШНИМИ ПОДПРОГРАММАМИ

Внешние подпрограммы в машинных командах позволяют организовать связь с УСО, что нельзя сделать средствами только алгоритмического языка БЭЙСИК. Для уменьшения времени счета некоторые фрагменты алгоритма могут быть оформлены внешними подпрограммами.

Внешние подпрограммы вызываются оператором CALL:

CALL < номер подпрограммы > [< параметры >]
< номер подпрограммы > – целое положительное число в диапазоне от 1 до 7999,
< параметры > – требуемые параметры.

Внешние подпрограммы загружаются в ОЗУ в процессе начального диалога и состоят из двух блоков. Первый блок длиной 8 байтов должен содержать номер подпрограммы (число в десятичной системе счисления) длиной 2 байта, контрольную сумму команд, полученную командой 12 01 (VERX), записанную в шестнадцатеричной системе счисления, длиной 2 байта, количество шагов во внешней подпрограмме, записанное в шестнадцатеричной системе счисления, длиной 2 байта, и коды 00 00 00 00 05 12. Выход из внешней подпрограммы должен осуществляться командой 05 11 (RTS).

Перед выполнением команды 05 11 в регистр S03 должен быть записан (или сохранен) код завершающего оператор CALL символа 00 10 (ПС) или 03 10 (:), а в регистр R01 – адрес следующего символа текста программы.

Для поддержания работоспособности интерпретатора при использовании внешних подпрограмм пользователь должен заботиться о сохранности интерпретатора и содержимого используемых им регистров служебной зоны ОЗУ. Примеры применения внешних подпрограмм приведены в § 4.7.

БЭЙСИК-интерпретатор позволяет вставлять в текст БЭЙСИК-программы

фрагменты в машинных командах. Это обеспечивается применением операторов CMD, общий формат которых одинаков и имеет вид

CMD {< код команды >} [[,]...]

Оператор CMD удобно применять для выполнения операций, не предусмотренных алгоритмическим языком БЭЙСИК, например для очистки экрана дисплея от ненужной информации, передвижения маркера по экрану дисплея, организации вывода символов. Например, символ ":" нельзя вывести на печать средствами языка БЭЙСИК, а применение оператора CMD дает такую возможность. Один оператор CMD должен занимать одну строку.

Рассмотрим один из возможных вариантов очистки экрана дисплея с применением оператора CMD. Чтобы выполнить эту операцию, надо использовать следующие машинные команды:

Одна строка { < номер строки > CMD 1304, 0015, 1302, 1501, 0412, 1400, 1303, 0012, 0412, 1405, 0514, 0514, 0514, 0514

Приведенные здесь машинные команды описаны в гл. 1.

Для перемещения маркера вдоль любой из 24 строк можно применить следующие машинные команды:

Одна строка { < номер строки > CMD 1304, 0001, 1302, 1501, 1303, XXXX, 0412, 1407, 0701, 0515, 0514, 0514, 0514, 0514

Символы XXXX обозначают код клавиш (для дисплея 15ИЭ-00-013). Они приведены в табл. 3.3.

Таблица 3.3

Клавиша	Код	Клавиша	Код
↖	0008	↑	0112
→	0109	↓	0113
←	0110	Очистка экрана вправо вниз от маркера	0115

Рассмотренные машинные команды перемещают маркер на одну позицию вправо или влево (если используется код 0109 или 0110 соответственно). Если эти машинные команды вставить в цикл FOR-NEXT, маркер будет перемещаться на y позиций ($y < 80$).

Например:

```
 5 LET Y=10
10 FOR I=1 TO Y
20 CMD 1304, 0001, 1302, 1501, 1303, 0109, 0412, 1407, 0701, 0515, 0514
30 NEXT I
40 END
```

Маркер продвинется на десять позиций вправо.

Движение маркера по вертикали можно осуществить средствами языка БЭЙСИК.

Например, перемещение маркера на 5 позиций вниз (точнее, пропуск пяти строк с маркером в левой позиции):

```
 5 LET X=5
10 FOR J=1 TO X
20 PRINT
30 NEXT J
40 END
```

Для вывода на экран дисплея, например, символа ":" 79 раз надо применить машинные команды:

```
 5 LET K=79
10 FOR I =1 TO K
15 CMD 1304, 1515, 1302, 1501, 0412, 1400, 1303, 0310, 0412, 1405, 0514
20 NEXT I
40 END
```

В пособии [22] приведена программа построения гистограммы для датчика случайных чисел HASARD. Аналогичную программу можно выполнить на Д3-28, только имя датчика случайных чисел, равномерно распределенных на интервале 0–1, будет RND.

Если на отрезке $[a, b]$ генерируются случайные числа, то, разбив его на p равных частей (обычно $p = 5-8$), называемых иногда "классами", подсчитаем количество чисел, попавших в k-й класс ($k = 1-p$). Для этого определим переменную j следующим образом: $j =$ целое$\left\{(x-a)\ \frac{1}{h}\right\} + 1$, где $h = (b-a)/p$.

Например, если случайное число равно 15,7, а отрезок $[a, b]$ равен [10, 20], то при $p = 5$ имеем: $j =$ целое $\left\{(15{,}7-10)\ \frac{10}{5}\right\} + 1 = 3$.

В приведенной ниже программе с классами p связан массив чисел T, количество элементов которого равно числу классов, а значения его элементов пропорциональны количеству чисел, попавших в тот или иной класс. В 98-й строке происходит масштабирование переменной V. Гистограмму образуют выводимые на экран дисплея символы ":". При числе классов $p = 5$ и 100 реализациях случайных чисел на интервале 0–1 программа работает 37 с.

```
  1 PRINT'*************************************'
  2 PRINT'****  ПОСТРОЕНИЕ ГИСТОГРАММЫ   ******'
  3 PRINT'*************************************'
 20 INPUT'ГРАНИЦЫ ИНТЕРВАЛА ДАТЧИКА СЛ.ЧИСЕЛ? 'A,B
 30 INPUT'ЧИСЛО КЛАССОВ И ЧИСЛО РЕАЛИЗАЦИЙ?  'P,N
 40 DIM T(20)
 50 FOR I=0 TO 20 :LETT(I)=0:NEXT I
 60 LETH=P/(B-A):LETI=1:LETZ=0
 70 LETX=RND(Z):LETZ=123456789:LETJ=INT((X-A)*H)+1
 75 LETT(J)=T(J)+1:LETI=I+1
 80 IF I<N THEN 70
 86 PRINT:PRINT
 90 FOR I=1 TO P:PRINT T(I):NEXT I
 91 PRINT'                        ОТНОСИТЕЛЬНАЯ ЧАСТОТА'
 92 PRINT'                0-------------------------------->>'
 95 FOR I=1 TO P
 98 LETV=T(I)/N*60:LETV1=T(I)/N
 99 PRINT !F1.3!V1;
100 PRINT'  I';
105 FOR J=1 TO V
106 CMD1304,1515,1302,1501,0412,1400,1303,0213,0412,1405
110 NEXT J:PRINT'*'
115 NEXT I
118 FOR I=1 TO 4:PRINT'              I ':NEXT I
119 PRINT'              Y'
120 END
```

С помощью команды CMD на экран дисплея можно выводить контуры различных фигур: прямоугольников, треугольников, квадратов.

Ниже приведена программа, которая выводит на экран дисплея контур квадрата, используя символ "*".

```
 10 REM ***********************************************
 20 REM ******   ПРИМЕНЕНИЕ КОМАНДЫ CMD          ******
 30 REM ***********************************************
 40 CMD1304,0001,1302,1501,1303,0008,0412,1407,0701,0515,0514
 50 CMD1304,0001,1302,1501,1303,0012,0412,1407,0701,0515,0514
 70 FOR K=1 TO 8
 80 PRINT
 90 NEXT K
100 PRINT TAB (25)' ';
110 FOR K=1 TO 30
120 GOSUB 900
130 NEXT K
140 FOR K=1 TO 10
150 GOSUB 900
160 GOSUB 910
170 GOSUB 920
180 NEXT K
190 FOR K=1 TO 30
200 GOSUB 900
210 GOSUB 920
220 GOSUB 920
250 NEXT K
260 FOR K=1 TO 10
270 GOSUB 900
280 GOSUB 930
290 GOSUB 920
300 NEXT K
310 STOP
900 CMD 1302,1501,1303,0210,1304,1515,0412,1407,0701,0515,0514
905 RETURN
910 CMD 1302,1501,1303,0113,1304,0001,0412,1407,0701,0515,0514
915 RETURN
920 CMD 1302,1501,1303,0110,1304,0001,0412,1407,0701,0515,0514
925 RETURN
930 CMD 1302,1501,1303,0112,1304,0001,0412,1407,0701,0515,0514
935 RETURN
940 END
```

4. ПРАКТИЧЕСКАЯ РАБОТА С МИКРОЭВМ "ЭЛЕКТРОНИКА Д3-28"

4.1. РАБОТА С НМЛ

Для успешной работы на микроЭВМ пользователь должен овладеть рядом практических приемов, позволяющих ускорять его работу и "выходить" из казалось бы безвыходных ситуаций, например при сбое микроЭВМ.

Рассмотрим два часто встречающихся при программировании на Д3-28 случая (все применяемые при этом операторы были описаны в § 2.18).

I. Пусть произведена запись информации (текста БЭЙСИК-программы) на МЛ. Для проверки сделанной записи необходимо выполнить считывание с МЛ в ОЗУ, не стирая старой информации. Считаем, что кассета с МЛ установлена в НМЛ; запись на МЛ сделана. Рекомендуется произвести следующие действия:

1) командой 12 00 (REW) перемотать МЛ к началу;

2) командой LOAD 'имя программы' НС3,НС4 загрузить БЭЙСИК-программу (если проверяется запись на МЛ текста БЭЙСИК-программы) в ОЗУ к уже имеющейся программе. Номера строк НС3 и НС4 могут быть произвольными, но обязательно должно выполняться неравенство НС4 > НС3 > > НС2, причем НС2 – номер последней строки программы, находящейся в ОЗУ. После дозагрузки нового текста с МЛ к старому тексту их можно просмотреть с помощью команды LIST;

3) стереть добавленный текст из ОЗУ командой CLEARP НС2, НС2+1 (запись НС2+1 означает конкретное число, т. е. не предполагает суммирования);

4) восстановить строку программы с номером НС2 повторным ее набором на клавиатуре дисплея и записью в ОЗУ.

II. Иногда в процессе работы с БЭЙСИК-интерпретатором возможна ошибочная запись БЭЙСИК-строки, начинающейся не с цифры. Непосредственно обратиться к этой строке и стереть ее нельзя. В таких случаях рекомендуется выполнить следующие действия:

1) командой CLEARP НС1,НС2 стереть фрагмент БЭЙСИК-программы от НС1 до НС2, где НС1 и НС2 – номера строк, соседних со стираемой строкой;

2) восстановить стертые строки с номерами НС1 и НС2, набрав их повторно с клавиатуры дисплея;

3) командой LIST вывести текст БЭЙСИК-программы на экран дисплея и убедиться в том, что неверная строка стерта.

Часто требуется распечатать имена БЭЙСИК-программ, записанные на МЛ. Для этого рекомендуется записать в ОЗУ текст служебной программы, которая распечатает имена (ср. с текстом программы из примера 2.17):

```
1 LOAD 4,7899
2 PRINT OPEN
3 GOTO 1
```

Если печать имен БЭЙСИК-программ осуществляется на ПМ "Consul", то оператор во 2-й строке надо заменить на следующий:

2 PRINT #1 OPEN

Перед загрузкой информации с МЛ в ОЗУ рекомендуется командами CLEARC и CLEARD очищать области COM и DIM. Такая очистка обеспечивает нормальную скорость считывания БЭЙСИК-программ и данных с МЛ в ОЗУ.

Как правило, на МЛ информация записывается многократно, поэтому рекомендуется перед каждой новой записью очищать МЛ от ранее сделанных записей. Часто возникает ситуация, когда БЭЙСИК-интерпретатор уже загружен в ОЗУ данной микроЭВМ и надо очистить МЛ для последующей записи информации, уже имеющейся в ОЗУ в данный момент. Очистку МЛ лучше всего делать на этой же микроЭВМ. Нажимая клавишу [ШН] на клавиатуре Д3-28, диалог временно приостанавливают. Затем по содержимому регистра R03 находят начало свободной зоны ОЗУ, записывают туда следующую программу в машинных командах

```
04 12  12 11  (ELMG)
04 12  12 03  (TRTAP)
04 12  12 04  (SAVC)
05 15         (STOP)
```

и выполняют ее. Когда МЛ очистится, надо нажать клавишу [С] на клавиатуре Д3-28, командой 12 00 (REW) перемотать МЛ к началу, восстановить диалог, последовательно нажимая клавиши [ШН] , [▷] , [М] .

В процессе редактирования на Д3-28 могут произойти сбои, после которых необходимо произвести перезапуск БЭЙСИК-интерпретатора. Для этого надо нажать клавишу [ШН] на клавиатуре Д3-28 и передать управление подпрограмме повторного старта интерпретатора командой

```
04 07  04 08  JMM 04 08
```

БЭЙСИК-интерпретатор можно запустить командой

```
04 07  04 07  JMM 04 07
```

При этом на экран дисплея выводится сообщение

```
ГОТОВ
:
```

В таком случае связь с находившейся в ОЗУ БЭЙСИК-программой теряется.

4.2. ПРОВЕРКА КАЧЕСТВА МАГНИТНЫХ ЛЕНТ

Для хранения программ и данных в Д3-28 используются компакт-кассеты, применяемые для звуковой записи в кассетных магнитофонах. В связи с тем что эти кассеты не предназначены специально для записи цифровой информации, необходимо производить проверку пригодности их использования на микроЭВМ. Предварительная разбраковка МЛ значительно повышает надежность считывания записанной на нее информации. Надежность повышается так-

же, если при работе с МЛ даже хорошего качества перед записью новой информации очистить ее от старых записей.

Текст программы, проверяющей качество МЛ, приведен в прил. 8. Контрольная сумма программы КП=2691,количество шагов 200. Программа работает в четырех режимах. Выбор режима производится командами, вводимыми с клавиш прямого кодирования в режиме "Р":

00 00	JSM 00 00	– очистка МЛ;
00 02	JSM 00 02	– запись на МЛ контрольных блоков заданной длины;
00 04	JSM 00 04	– считывание с МЛ контрольных блоков и подсчет количества сбоев;
00 15	JSM 00 15	– вывод на индикацию результатов проверки МЛ.

Порядок работы с программой следующий. После загрузки с нулевого адреса ОЗУ программы проверки, имеющей КП=2691, в НМЛ устанавливается проверяемая кассета. При необходимости предварительной очистки МЛ в Д3-28 вводится команда 00 00 (JSM 00 00). МЛ перематывается в исходное состояние, и производится ее очистка. После очистки МЛ вновь перематывается в исходное состояние и останавливается. МикроЭВМ Д3-28 переходит в режим индикации. Этот режим может использоваться перед проверкой МЛ, а также для очистки МЛ, уже использовавшихся в работе.

Проверка МЛ производится в два приема. Сначала на нее последовательно записываются тестовые блоки одинаковой длины, задаваемой при запуске программы записи. Затем выполняется считывание этих блоков и анализ всех возможных сбоев и ошибок. После завершения считывания вызываются результаты проверки.

Для записи на МЛ тестовых блоков в Д3-28 необходимо ввести команду 00 02 (JSM 00 02). При этом осуществляется запуск программы, индикация регистра X гаснет. С цифровых клавиш Д3-28 вводится требуемая длина тестового блока, задаваемая целым положительным числом от 1 до 31 488. Нажимается клавиша [S] ("ПУСК"). Если (X)=0, длина тестового блока автоматически устанавливается равной 10 000. Затем в ОЗУ, начиная с адреса 00.04.00.00 (1024_{10}), формируется запись заданной длины, состоящая из кодов 05 05. На МЛ многократно выводится записанный участок ОЗУ. По достижении конца МЛ производится отключение НМЛ и останов Д3-28. После этого необходимо перемотать МЛ к началу. Для считывания и контроля записей в Д3-28 вводится команда 00 04 (JSM 00 04). Последовательно считываются записанные блоки. При включении индикаторов "ОМ", "ОП" или несовпадении контрольных сумм считанного и тестового блоков регистрируются сбои.

После останова МЛ в крайнем положении следует установить Д3-28 в исходное состояние клавишей [C]. В Д3-28 вводится команда 00 15 (JSM 00-15). В регистре Y будет индицироваться число записанных тестовых блоков, в регистре X – число сбоев.

Проверку МЛ рекомендуется начинать записью блоков большой длины (10 000 байтов и более). Для уточнения качества МЛ длину блоков следует уменьшить в несколько раз. Проверяемая лента должна иметь ракорд.

4.3. ОТЛАДКА И РЕДАКТИРОВАНИЕ БЭЙСИК-ПРОГРАММ

Как уже отмечалось в гл. 2, каждая строка БЭЙСИК-программы нумеруется целыми числами, лежащими в диапазоне от 1 до 7999. Удобно выбирать номера соседних строк таким образом, чтобы они различались на 5 или 10 и при необходимости можно было вставить между ними любую новую строку. Для этого надо сначала набрать номер вставляемой строки, который не должен совпадать с уже имеющимися номерами, а затем – вставляемый текст. После нажатия клавиши [ПС] новая строка будет автоматически помещена в уже набранную программу с соблюдением нумерации строк в возрастающем порядке.

Если требуется стереть всю строку, надо набрать ее номер и нажать клавишу [ПС] на клавиатуре дисплея. Можно набирать номер стираемой строки и записывать на ее место другой текст.

Для стирания неверно набранного, последнего в набираемой строке символа надо нажать клавишу [ЗБ] на клавиатуре дисплея, после чего набрать требуемый символ.

Чтобы прервать выполнение программы, нужно нажать одновременно клавиши [СУ] и [П Р], при этом на экране дисплея (имеется в виду дисплей типа 15ИЭ-00-013) появится сообщение:

¬ Р
ОСТАНОВ В СТРОКЕ<номер строки>

Для продолжения счета по программе надо набрать оператор GOTO<НС1>, где НС1 – номер строки, в которой произошло прерывание, и нажать клавишу [ПС] .

Приостановить вывод информации на экран дисплея можно, отжимая клавишу [ЛИН] на клавиатуре дисплея. Для продолжения вывода информации необходимо снова нажать клавишу [ЛИН] .

Для контроля процесса вычислений удобно в отлаживаемую программу вставлять оператор PRINT или STOP. С помощью оператора PRINT рекомендуется выводить промежуточные значения переменных и элементов массивов. Оператор STOP останавливает выполнение программы. Продолжить ее выполнение можно, набрав оператор GOTO<номер строки> , в которой произошел останов, и нажав клавишу [ПС] .

При обнаружении интерпретатором синтаксических ошибок или ошибок счета на экран дисплея выводится сообщение

ОШИБКА<номер ошибки>В СТРОКЕ<номер строки>

Для ошибок, возникающих при работе в непосредственном режиме, и ошибок в строке данных, вводимых оператором INPUT , указывается нулевой номер строки.

Оператором PRINT в непосредственном режиме можно вывести на экран дисплея любые значения переменных и элементов массивов, полученных к мо-

менту останова по оператору STOP (или по прерыванию). В прил. 7 приведены сообщения БЭЙСИК-интерпретатора об ошибках.

Ошибки с номерами до 99-го вызывают останов программы и переход в непосредственный режим, при ошибках с номерами 121–128 выполнение программы продолжается после вывода сообщения об ошибке на экране дисплея.

Ошибочные символы в строке БЭЙСИК-программы можно исправить на экране дисплея и записать в ОЗУ на место неверной строки. Для этого необходимо выполнить следующие действия:

1) отжать клавишу [ЛИН] ;

2) стереть символ ":" перед редактируемой строкой (если он есть); это достигается подведением маркера клавишей [←] к символу ":" и нажатием клавиши "пробел";

3) исправить ошибочные символы в строке;

4) клавишей [→] установить маркер за последним символом строки;

5) отжать клавишу [РЕД], нажать клавишу [СУ] и, кратковременно удерживая ее, нажать клавишу [Ц/С]. При этом в конце исправляемой строки запишется мигающий символ С;

6) нажать клавишу [РЕД] ;

7) нажать клавишу [ПРС] ;

8) нажать клавишу [ЛИН] ; через несколько секунд на экран дисплея выведется символ":" .

Редактирование строки завершено. Командой LIST проверить это.

Исправление фрагмента БЭЙСИК-программы на экране дисплея и передача его с экрана дисплея в ОЗУ осуществляется точно так же, как и исправление одной строки, с тем отличием, что первая строка редактируемого текста клавишей [↑] подводится под верхний срез экрана дисплея, а мигающий символ С записывается в конце последней строки редактируемого фрагмента. Вместо клавиши [ПРС] надо нажимать клавишу [ПРД] .

4.4. РАБОТА С ДИСПЛЕЕМ 15ИЭ-00-013

Дисплей 15ИЭ-00-013 (в дальнейшем дисплей) обеспечивает выполнение следующих функций:

1) отображение информации в алфавитно-цифровом виде;

2) набор и редактирование алфавитно-цифровой информации с помощью клавиатуры;

3) обмен информацией с внешними устройствами Д3-28.

Дисплей обеспечивает индикацию характера взаимодействия (диалоговый, автономный) его с Д3-28 с помощью индикаторов ДУП (дуплексный), ЛИН (линия) и индикацию состояния регистров ввода: латинского (русского) и нижнего (верхнего). В нем реализована звуковая сигнализация нажатия клавиш. Дисплей обеспечивает повторение выдачи данных с частотой 9 Гц при

продолжительности нажатия клавиши свыше 2 с, поэтому нажатие на клавишу должно быть кратковременным.

В верхней части клавиатуры расположены световые индикаторы режимов работы дисплея и установленных регистров клавиатуры, ниже индикаторов – ряд клавиш управления режимов работы дисплея. С правой стороны находится группа клавиш с изображением стрелок. Это клавиши управления перемещением маркера (курсора) по экрану дисплея. В центре расположена основная группа клавиш с нанесенными на них символами, цифрами, буквами латинского и русского алфавитов, клавиши ввода, а также клавиши переключения регистров [ЛАТ], [РУС], [ВР], [НР]. Правее клавиш ввода находится группа цифровых клавиш, дублирующих цифровые клавиши ввода. После включения дисплея в сеть примерно через 15–18 с на экране дисплея в правом верхнем углу появляется служебная строка (25-я строка, так как на экран дисплея выводятся 24 строки информации). Изображение 25-й служебной строки на экране дисплея приведено на рис. 4.1.

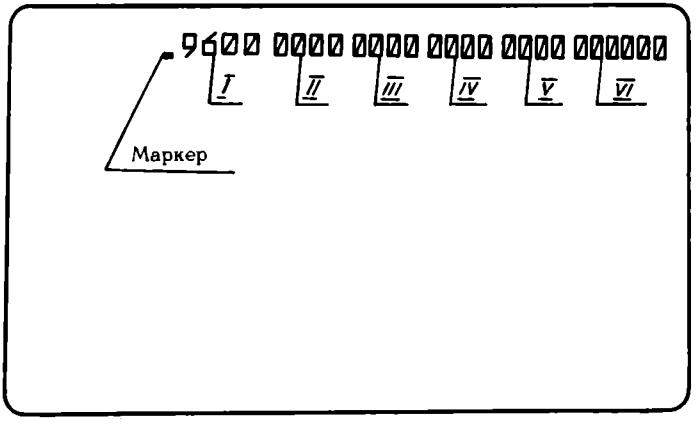


Рис. 4.1

Рассмотрим назначение шести групп цифр служебной строки:

I группа показывает текущую скорость обмена по интерфейсу "20 мА токовая петля" (бит/с). При включении дисплея автоматически устанавливается скорость 9600 бит/с. Пользователь может установить одну из скоростей: 4800, 2400, 1200, 600, 300, 150 и 75 бит/с;

II группа (рассматриваем ее последовательно слева направо) включает: сдвиг (0 – есть, 1 – нет); автоповтор клавиатуры (0 – есть, 1 – нет); резерв 1; вид маркера (0 – вид 1, 1 – вид 2);

III группа содержит:звуковой сигнал после 72-го знака в строке (0 – есть, 1 – нет); звуковой сигнал клавиатуры (0 – есть, 1 – нет); систему команд* (0 – № 1, 1 – № 2); синхронизацию обмена (0 – нет, 1 – есть);

---

*В процессе работы дисплея с внешними устройствами пользователь может изменять содержимое данного разряда

IV группа включает: резерв 3; фиксацию маркера в крайних положениях строки* (0 – нет, 1 – есть); авто ПС, ВК** (0 – нет, 1 – есть); резерв 2;

V группа: тип контроля (0 – нечетность, 1 – четность); контроль (0 – нет, 1 – есть); авто ПС,ВК при передаче (0 – есть, 1 – нет); индикация 25-й строки (0 – есть, 1 – нет);

VI группа включает: десятки часов; часы; десятки минут; минуты; десятки секунд; секунды.

В группах цифр II–V при включении дисплея устанавливаются нули. Для изменения содержания какой-либо группы цифр надо отжать клавишу [СДВ], подвести маркер, нажимая клавишу [→] или [←], нажать клавишу [↓], снова нажать клавишу [СДВ].

Клавиша [ОЧС] предназначена для очистки экрана дисплея без изменения 25-й служебной строки, клавиша [СБР] – для сброса логики дисплея, очистки экрана и установления 25-й строки в исходное состояние. Если нажать клавишу [СДВ] в момент диалога с Д3-28, на экран дисплея выведется сообщение

ОШИБКА 1 В СТРОКЕ 0

4.5. СЛУЖЕБНЫЕ ПРОГРАММЫ В МАШИННЫХ КОДАХ

Для облегчения процесса загрузки БЭЙСИК-интерпретатора в ОЗУ Д3-28 удобно использовать служебную программу, текст которой приводится в табл. 4.1.

Таблица 4.1

Номер шага	Команда	Мнемонический код
00000	13 00 00 00	MOV #00 00,S00
00002	13 01 01 08	MOV #01 08,S01
00004	13 02 07 05	MOV #07 05,S02
00006	13 03 03 00	MOV #03 00,S03
00008	13 04 00 00	MOV #00 00,S04
00010	13 05 01 02	MOV #01 02,S05
00012	10 15 11 08	MOV (R08)+,R11
00014	10 12 11 09	MOV R11,–(R09)
00016	10 00 04 09	ADD #04,R09
00018	14 08 03 10	SOBZ R10 00022
00020	14 02 00 09	BR 00012
00022	07 03	DIG 3
00023	07 10	E
00024	07 05	DIG 5
00025	12 13	JMP @X
00026	07 08	DIG 8
00027	04 13 11 00	MOV X,R00
00029	07 00	DIG 0
00030	12 02	LOADX

* Для системы команд № 1 фиксация маркера в крайних положениях строк не производится.

**Только для системы команд № 2.

Окончание табл. 4.1

Номер шага	Команда	Мнемонический код
00031	14 14 00 05	BMER 00037
00033	12 01	VERX
00034	05 10	BPER 00037
00035	14 03 00 08	BR 00044
00037	14 08 03 00	SOBZ R00 00041
00039	14 03 00 02	BR 00042
00041	12 00	REW
00042	14 02 00 14	BR 00029
00044	06 04	MOV X,Y
00045	07 01	DIG 1
00046	07 05	DIG 5
00047	07 07	DIG 7
00048	07 00	DIG 0
00049	07 07	DIG 7
00050	07 01	DIG 1
00051	05 09	BEQ, Y,X 00054
00052	14 02 01 08	BR 00029
00054	05 15	STOP
00055	14 02 00 02	BR 00054
00057	05 12	END

Таблица 4.2

Номер шага	Команда	Мнемонический код
25000	04 13 11 01	MOV X,R01
25002	10 05 07 01	NSS 07 01
25004	04 13 04 01	MOVD R01,X
25006	05 15	STOP
25007	14 02 00 06	BR 25002

Программа загружается в ОЗУ Д3-28 с нулевого шага. При запуске она переписывает себя, начиная с адреса 30 000, последовательно загружает с нулевого адреса БЭЙСИК-интерпретатор, производит программный контроль правильности загрузки, отсутствия сбоев. Данная программа может загружать в ОЗУ любую версию БЭЙСИК-интерпретатора. Необходимо записать КП загружаемой версии на адреса с 00045 по 00051. В табл. 4.1 записана КП=157071 БЭЙСИК-интерпретатора. Если при считывании произошло 8 сбоев, производится перемотка МЛ в начало кассеты, и интерпретатор вновь загружается.

Для поиска номера шага, на котором записан код некоторого символа, удобно использовать служебную программу, текст которой приведен в табл. 4.2.

Программа может загружаться с любого адреса в ОЗУ. В табл. 4.2 она записана с адреса 25000. Программа ищет в ОЗУ с адреса, равного (R01), код, записанный во втором байте команды

NSS 10 05 0701 (NSS 0701)

Если указанный код найден, программа останавливается. В регистре X при этом индицируется номер шага, на котором записан данный код. Для дальнейшего поиска кода по программе она запускается нажатием клавиши [S].

В процессе работы возможно случайное выключение дисплея. После его включения в сеть связь между дисплеем и Д3-28 нарушается. Возобновить ее можно следующими командами, набираемыми непосредственно на клавиатуре Д3-28:

1) вновь включить дисплей в сеть;

2) нажать клавишу [ШН] ;

3) на клавиатуре Д3-28 набрать последовательно коды:

```
13 02  15 07 MOV # 15 07,S02
13 03  15 15 MOV # 15 15,S03
04 12  14 07 OUTOWS
04 07  04 08 JMM 04 08
```

На экране дисплея появляется символ ":", что свидетельствует о возобновлении связи с Д3-28. Все БЭЙСИК-программы, занесенные в ОЗУ до момента потери связи, сохраняются.

4.6. СОПРЯЖЕНИЕ МИКРОЭВМ "ЭЛЕКТРОНИКА Д3-28" С ВНЕШНИМИ УСТРОЙСТВАМИ

МикроЭВМ "Электроника Д3-28", обладая большим набором команд ввода/вывода и относительной простотой сопряжения с нестандартным оборудованием, может успешно использоваться для автоматизации экспериментов и управления процессами, которые не требуют высокого быстродействия ЭВМ.

Сопряжение Д3-28 с внешними устройствами вызывает необходимость создания определенных технических и программных средств.

Технические средства должны обеспечивать параллельный восьмиразрядный ввод/вывод информации, выбор устройства по заданному адресу, синхронизацию обмена.

Сопряжение выполняется обычно с использованием сигналов, выведенных на разъем Д3-28 ВВОД/ВЫВОД. Расположение сигналов на этом разъеме и дополнительная информация о них приведены в табл. 4.3. Временные диаграммы сигналов во время вывода и ввода информации при обмене внешних устройств с Д3-28 приведены на рис. 4.2 и 4.3 соответственно, где τ_2 – время обработки информации ПУ.

На рис. 4.4 представлена принципиальная электрическая схема одного из вариантов построения УСО. Она выполняет следующие функции:

1) при включении питания УСО устанавливает в нулевое состояние триггеры DD1, DD2 и DD16;

2) дешифрирует состояние шины "$\overline{\text{УПР}}$" от 14 01 до 14 08, вырабатывая при этом сигнал $\overline{\text{СИП}}$ и восемь сигналов на выходах элемента DD3, используемых для выбора устройств;

Таблица 4.3

Сигнал	Контакт разъема ВВОД/ВЫВОД	Назначение сигнала	Активный уровень	Примечание
$\overline{X13}$	Б9			Младший разряд
$\overline{X23}$	Г9			
$\overline{X43}$	Г7			
$\overline{X83}$	Г14	Шина "УПРАВЛЕНИЕ"	Низкий	
$\overline{Y13}$	Г13			
$\overline{Y23}$	Б13			
$\overline{Y43}$	Г8			
$\overline{Y83}$	Б8			Старший разряд
$\overline{X12}$	Б1			Младший разряд
$\overline{X22}$	Г2			
$\overline{X42}$	Г3			
$\overline{X82}$	Г4	Шина "ВЫВОД"	Низкий	
$\overline{Y12}$	Б5			
$\overline{Y22}$	Г6			
$\overline{Y42}$	Б10			
$\overline{Y82}$	Г11			Старший разряд
$\overline{Вва1}$	Г1			Младший разряд
$\overline{Вва2}$	Б2			
$\overline{Вва4}$	Б3			
$\overline{Вва8}$	Б4	Шина "ВВОД"	Низкий	
$\overline{Ввв1}$	Г5			
$\overline{Ввв2}$	Б6			
$\overline{Ввв4}$	Г10			
$\overline{Ввв8}$	Б11			Старший разряд
$\overline{СИМ}$	Б7		Низкий	
$\overline{СИП}$	Г15	Синхронизация обмена	"	
Вв	Г12		Высокий	
$\overline{Пр1}$	Б14			
$\overline{Пр2}$	Б15	Внешнее прерывание	Низкий	
$\overline{Пр4}$	Г16			
$\overline{Пр8}$	Б16			
$\overline{ВПрП}$	Б12	Ввод программы из внешнего устройства	Низкий	

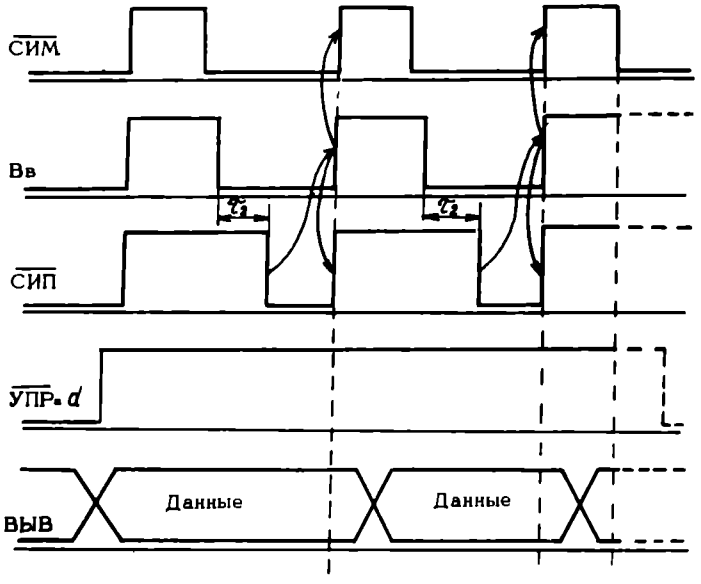

СИМ
Вв
τ2
τ2
СИП
УПР=d
Данные
Данные
ВЫВ

Рис. 4.2

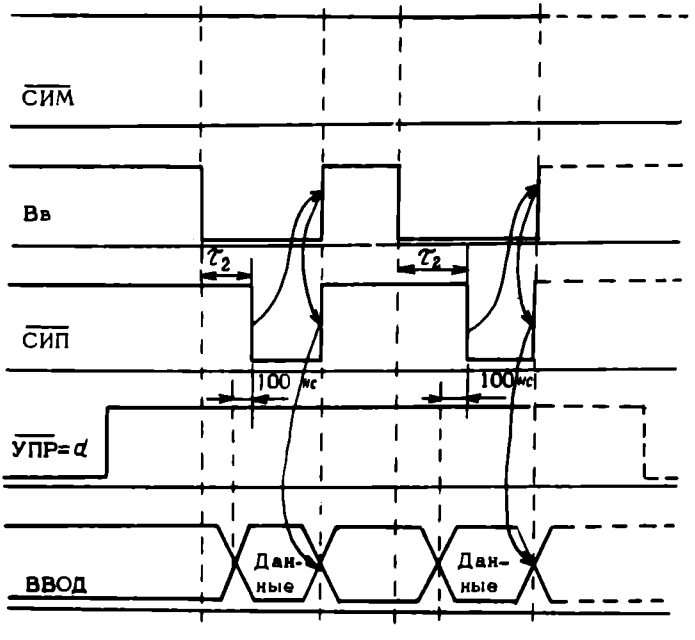

СИМ
Вв
τ2
τ2
СИП
100 мс
100 мс
УПР=d
Дан-
ные
Дан-
ные
ВВОД

Рис. 4.3

3) выводит в регистр на элементах DD1 и DD2 байт данных при УПР=14 01;

4) устанавливает в единичное или нулевое состояние триггер DD16 при УПР=14 02 и УПР=14 03 соответственно.

5) вводит в ДЗ-28 данные при подключении источника данных к шине "ВВОДПУ", состоящей из сигналов Ввв8ПУ,..., Вва1ПУ;

6) обеспечивает обмен информацией между ДЗ-28 и дисплеем при отсутствии обмена с УСО и подключении дисплея к шине "ВВОДД" и линиям СИПД,

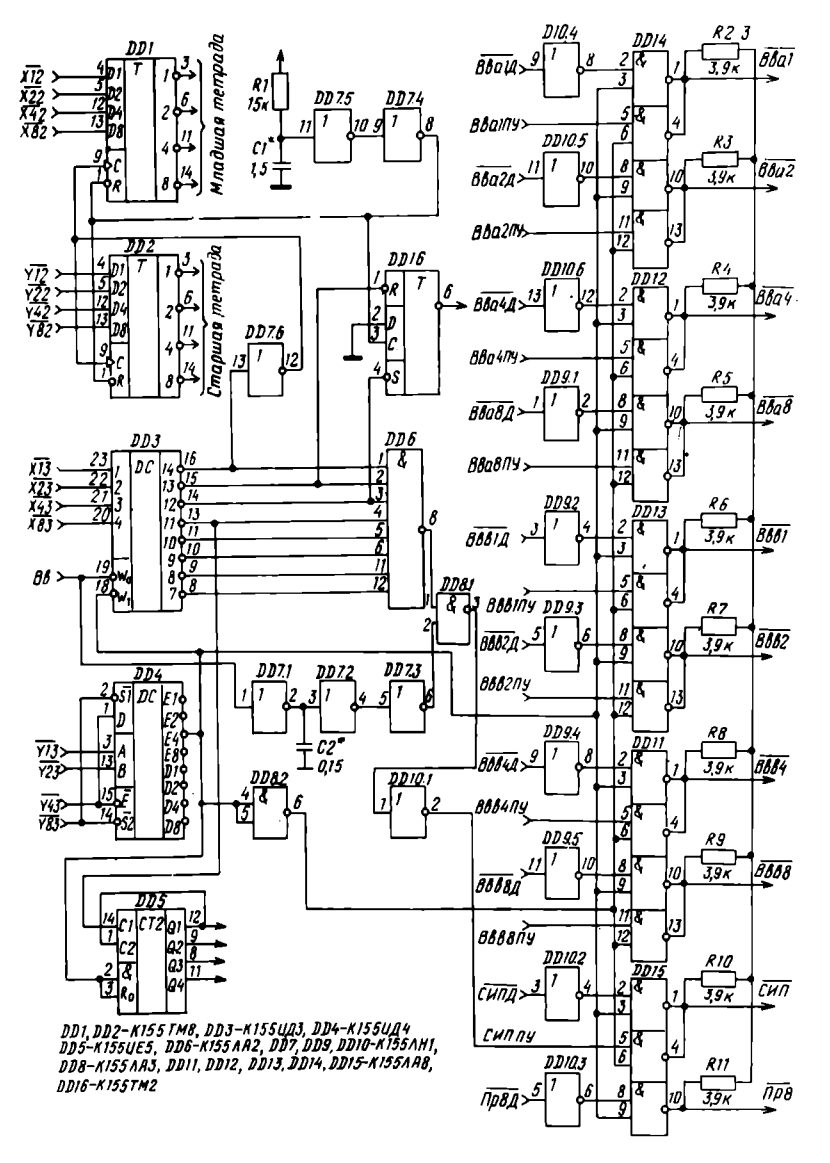


Рис. 4.4

$\overline{Пр8Д}$. При этом сигналы шины "$\overline{ВЫВ}$" $\overline{Y82}$,...,$\overline{X12}$ подаются в дисплей параллельно с УСО без дополнительной развязки. Кроме того, в дисплей подается сигнал СИМ, не обозначенный на схеме.

При необходимости ввода в Д3-28 более одного байта за одну передачу (например, при считывании результатов измерений цифрового прибора) разряды приборов подключаются к входам цифровых коммутаторов, например К155КП1, К155КП2, на адресные входы которых подаются выходные сигналы счетчика DD5.

Тексты двух внешних подпрограмм, обслуживающих рассмотренную схему при работе Д3-28 с БЭЙСИК-интерпретатором, приведены в прил. 12.

Подпрограмма с номером 7 предназначена для вывода в регистр на триггерах DD1, DD2 байта данных. Общая запись обращения к подпрограмме:

CALL 7, < имя переменной >

Переменной перед обращением должно быть присвоено значение от 0 до 255.

Подпрограмма с номером 21 предназначена для установки DD16 в состояние "1" или "0" и одновременного ввода в Д3-28 байта данных через шину "$\overline{ВВОД}$". Общая запись обращения к подпрограмме:

CALL 21, < имя переменной1 > , < имя переменной2 >

Первой переменной перед обращением должно быть присвоено значение 0 или 1. Второй переменной после выполнения подпрограммы будет присвоено значение введенного байта.

Проиллюстрируем использование указанных подпрограмм в БЭЙСИК-программе на примере.

Пример 4.1. Вывести в регистры DD1, DD2 УСО значение переменной A, установить триггер DD16 в единичное состояние, ввести в Д3-28 один байт данных, присвоить его значение переменной B и вывести значение B на дисплей.

Возможный вариант программы:

```
120 A=176
130 CALL 7,A
140 T=1
150 CALL 21,T,B
160 PRINT B
```

В подпрограммах проверяется правильность записи списка переменных в операторах CALL,а также осуществляется проверка на допустимое значение выводимых переменных. При обнаружении ошибок на экран дисплея выводятся соответствующие сообщения и производится выход из подпрограммы.

Если при обращении к подпрограмме 21 вторая переменная в операторе CALL ранее не была определена, то в подпрограмме производится формирование этой переменной в ОЗУ Д3-28.

5. ПРОГРАММИРОВАНИЕ НА МИКРОЭВМ "ИСКРА 226"

5.1. СОСТАВ И ОСНОВНЫЕ ХАРАКТЕРИСТИКИ

В состав микроЭВМ "Искра 226" входят (в зависимости от ее исполнения): процессор интерпретирующий диалоговый; дисплей с клавишным устройством; печатающее устройство "ROBOTRON 1154"; накопитель на гибком магнитном диске; кассетный накопитель на МЛ; графопостроитель.

Основными характеристиками рассматриваемой микроЭВМ являются: объем ОЗУ 64 К (128 К); разрядность числа – 13 разрядов; среднее время выполнения операций (в секундах): арифметических – не более 0,001, вычисление элементарных функций – не более 0,05, извлечение квадратного корня – не более 0,002; наличие библиотеки стандартных программ по математике и статистике, перечень которых приведен в табл. 5.1.

МикроЭВМ "Искра 226" позволяет: вводить программы и данные с клавиатуры или внешних устройств; редактировать программы; выводить тексты программ и результаты вычислений на экран дисплея, печатающее устройство и на другие внешние устройства и т. д.

Устройство внешней памяти (НГМД) поставляется в двух вариантах: носитель информации – гибкий магнитный диск (дискета) – устанавливается горизонтально (в НГМД "Искра 005-50") и вертикально (в НГМД "Искра 005-51"). На одну сторону дискеты можно записать примерно 256 К информации. В НГМД "Искра 005-51" информация размещается на обеих сторонах дискеты. В описанных НГМД имеется по два дисковода: верхний (левый) обозначается R , нижний (правый) – F. Порядок включения НГМД несколько различен для разных вариантов исполнения.

Рассмотрим последовательность действий при подготовке НГМД "Искра 005-50" к работе:

1) включить клавишу питания, при этом загорится индикаторная лампочка;

2) нажать кнопку вывода магнитной головки на предельную дорожку диска и выждать около 10 с (до щелчка);

3) открыть дверцу нужного дисковода и вставить дискету в дисковод так, чтобы наклейка на дискете располагалась в верхнем углу;

4) закрыть дверцу дисковода и нажать кнопку вывода магнитной головки на предельную дорожку диска.

Опишем последовательность действий при подготовке НГМД "Искра 005-51" к работе:

1) включить клавишу питания, при этом загорится индикаторная лампочка;

2) поднять вверх ручку дверцы и открыть дверцу;

3) вставить дискету в дисковод так, чтобы наклейка на ней располагалась в верхнем правом углу;

Таблица 5.1

Имя программы	Назначение
МАТМ 1	Корни квадратного уравнения
МАТМ 2	Корни полинома
МАТМ 3	Нахождение корней методом деления отрезка пополам
МАТМ 4	Действительные корни полинома
МАТМ 5	Расчет интеграла методом Симпсона
МАТМ 6	Расчет интеграла методом Ромберга
МАТМ 7	Метод Рунге–Кутта
МАТМ 8	Квадратурная формула Гаусса
МАТМ 9	Производная
МАТМ 10	Обращение матрицы
МАТМ 11	Обращение матрицы. Модифицированный метод Гаусса–Жордана
МАТМ 12	Собственные числа и собственные векторы
МАТМ 13	Операции над векторами
МАТМ 14	Векторный анализ
МАТМ 15	Решение системы линейных уравнений (метод исключения неизвестных)
МАТМ 16	Сложение и вычитание матриц, умножение матриц на скалярную величину
МАТМ 17	Матричное умножение
МАТМ 18	Решение системы линейных уравнений. Итерационный метод Гаусса–Зейделя
МАТМ 19	Линейное программирование
МАТМ 20	Комплексный определитель
МАТМ 21	Прямые и обратные гиперболические функции
МАТМ 22	Тригонометрические и гиперболические функции комплексного аргумента
МАТМ 23	Преобразование углов I
МАТМ 24	Преобразование углов II
МАТМ 25	Тригонометрический полином
МАТМ 26	Решение плоских треугольников
МАТМ 27	Преобразование координат
МАТМ 28	Вычисление площади многоугольника
МАТМ 29	Линейная интерполяция
МАТМ 30	Интерполяция Лагранжа
МАТМ 31	Наибольший общий делитель двух целых чисел
МАТМ 32	Разложение числа на простые множители
МАТМ 33	Размещения и сочетания
МАТМ 34	Логарифм В по основанию А
МАТМ 35	Уравнение второй степени I
МАТМ 36	Обращение уравнения второй степени
МАТМ 37	Уравнение второй степени II
МАТМ 38	Арифметика комплексных чисел
МАТМ 39	Гипергеометрическая функция
МАТМ 40	Квадратный корень комплексного числа
МАТМ 41	Бесселевы функции
МАТМ 42	Гамма-функция
МАТМ 43	Анализ Фурье

Имя программы	Назначение
МАТМ 44	Анализ Фурье (функция задана таблично)
СТАТ 1	Линейная регрессия Y= A+BX
СТАТ 2	Множественная линейная регрессия
СТАТ 3	Регрессия порядка N
СТАТ 4	Экспоненциальная регрессия $Y=Ae^{BX}$
СТАТ 5	Геометрическая регрессия $Y=AX^{B}$
СТАТ 6	Линейная корреляция
СТАТ 7	Корреляционная матрица
СТАТ 8	Однофакторный дисперсионный анализ
СТАТ 9	Двухфакторный дисперсионный анализ
СТАТ 10	Дисперсионный анализ – латинские квадраты
СТАТ 11	Хи-квадрат критерий и распределение Хи-квадрат
СТАТ 12	Хи-квадрат анализ
СТАТ 13	Т-критерий Стьюдента
СТАТ 14	Двухвыборочный критерий Вилкоксона
СТАТ 15	Критерий Манна–Уитни
СТАТ 16	Нормальная плотность распределения и функция распределения
СТАТ 17	Отрицательное биномиальное распределение
СТАТ 18	Биномиальное распределение
СТАТ 19	Распределение Пуассона
СТАТ 20	F-значение
СТАТ 21	Т-значение
СТАТ 22	Случайное нормальное отклонение
СТАТ 23	Среднее значение, дисперсия, среднее квадратичное отклонение I
СТАТ 24	Среднее значение, дисперсия, среднее квадратичное отклонение II
СТАТ 25	Геометрическое среднее и среднее квадратичное отклонение
СТАТ 26	Кросс-ковариация процессов
СТАТ 27	Автоковариация процесса
СТАТ 28	Надежность системы
СТАТ 29	Интеграл вероятности ошибки
СТАТ 30	Формула Тальборта
СТАТ 31	Формула Мэннинга
СТАТ 32	Падение напора в трубе
СТАТ 33	Уравнение Бернулли
СТАТ 34	Напряжение от коробления из-за температурного перепада
СТАТ 35	Давление из-за поверхностных нагрузок
СТАТ 36	Балка
СТАТ 37	Срок эксплуатации нефтяной скважины
СТАТ 38	Полное сопротивление цепи
СТАТ 39	Характеристическое сопротивление и ЭДС генератора
СТАТ 40	Уравнение "Эрланга В"

4) закрыть дверцу дисковода, нажав ее ручку вниз.

При включении НГМД автоматически устанавливается защита записи. Снять защиту можно, нажав соответствующую кнопку. Включение микроЭВМ "Искра 226" и загрузка БЭЙСИК-интерпретатора осуществляются в следующем порядке:

1) включить процессор тумблером, расположенным на боковой панели слева. Через 50–60 с на экране дисплея появится надпись:

ЗАГРУЗЧИК

:_

2) включить и подготовить к работе НГМД;

3) на клавиатуре дисплея последовательно набрать информацию, нажимая клавиши [LOAD], [F], [#], [0], [CR/LF] для дисковода F или клавиши [LOAD], [R], [#], [0], [CR/LF] для дисковода R.

Если на экране дисплея появляется сообщение о сбое, п.3 надо повторить, набирая вместо символа 0 символы 1,2 или 3;

4) после загрузки интерпретатора на экран дисплея выводится сообщение

BASIC 01.15.15.12

5) для запуска интерпретатора необходимо нажать на клавиатуре дисплея клавиши [RUN], [1], [CR/LF]. При этом на экран дисплея выводится сообщение:

READY

:_

Символ ":" указывает на наличие диалога с микроЭВМ. Дальнейшая работа возможна в непосредственном и программируемом режимах. В первом случае счет происходит непосредственно после набора БЭЙСИК-строки без номера и при нажатии клавиши [CR/LF]. Во втором случае текст БЭЙСИК-программы вначале записывается в ОЗУ и по команде RUN выполняется.

В непосредственном режиме строка может быть исправлена, если не была нажата клавиша [CR/LF]. Для этого применяются клавиши [BACKSPACE] и [LINE ERASE]. Первая клавиша стирает посимвольно текст, начиная с последнего введенного символа, вторая стирает всю введенную строку. Строку можно отредактировать, нажимая клавишу [EDIT]. На экране дисплея вместо символа ":" появляется символ "*". Клавишей [DELETE] стирается символ в позиции, под которой находится курсор, а часть строки справа от курсора сдвигается влево на один символ. Клавишей [INSERT] раздвигается редактируемая строка для вставки символа. Клавишей [ERASE] стирается часть строки, начиная с местоположения курсора. Отредактированная строка выполняется после нажатия клавиши [CR/LF]. В режиме "EDIT" могут использоваться клавиши [BACKSPACE] и [LINE ERASE].

Ввод текста БЭЙСИК-программы с клавиатуры дисплея начинается с ввода номера строки одним из следующих двух способов: 1) нажатием клавиши [STMT NUMBER]; 2) нажатием соответствующих цифровых клавиш. При

первом нажатии клавиши [STMT NUMBER] вводимой строке присваивается номер 10. Очередным нажатием этой клавиши новой вводимой строке присваивается номер, на 10 больший номера предыдущей строки. Перед вводом текста новой БЭЙСИК-программы ОЗУ очищается нажатием клавиш [CLEAR] и [CR/LF] . Введенный текст можно просмотреть на экране дисплея, нажав клавишу [LIST] или клавиши [LIST] и [S] (для постраничного вывода на экран дисплея). Для удаления строки необходимо набрать ее номер и нажать клавишу [CR/LF] . Для редактирования строки сначала надо перевести ее в режим редактирования (набрать ее номер и затем нажать клавишу [EDIT]). При этом около номера строки появляется символ "*". Нажатием клавиши [RECALL] текст редактируемой строки выводится на экран дисплея. Процесс редактирования аналогичен вышеописанному.

Пошаговое выполнение программы осуществляется после нажатия клавиши [HALT/STEP] , а переход снова в автоматический режим счета – клавиши [CONTINUE] .

Для использования стандартных подпрограмм надо подготовить ЭВМ к работе (если она не была готова), установить дискету с надписью "Математика" или "Статистика", загрузить текст стандартной программы, нажимая клавиши [LOAD] , [D] , [C] <[F] или [R] >["] < имя программы> ["] , [CR/LF] , и запустить программу на выполнение нажатием клавиш [RUN] и [CR/LF] , после чего отвечать на запросы данной программы.

5.2. АЛГОРИТМИЧЕСКИЙ ЯЗЫК "БЭЙСИК" ДЛЯ МИКРОЭВМ "ИСКРА 226"

Операторы языка БЭЙСИК. Алгоритмический язык БЭЙСИК для микроЭВМ "Искра 226" отличается от рассмотренного в гл. 2 варианта БЭЙСИКа для Д3-28. В прил. 10 дан список сообщений об ошибках интерпретатора, а ниже приведен перечень ключевых слов операторов БЭЙСИКа микроЭВМ "Искра 226".

К *основным операторам* относятся: CLEAR, COM, COM CLEAR, DATA, DEFFN, DEFFN' , DIM, FOR, ¤GIO', GOSUB, GOSUB' , GOTO, IF-THEN, LET, NEXT, ON, ON ERROR, READ, REM, RENUMBER, RESTORE, RETURN, RETURN CLEAR, RUN, SELECT, STOP, TRACE.

Операторами преобразования данных являются: ADD, AND, BIN, BOOL, CONVERT, PACK, ROTATE, UNPACK.

Операторы ввода/вывода: BACKSPACE, DATALOAD, DATARESAVE, DATALOAD BT, DATASAVE, DATASAVE BT, ¤GIO' HEXPRINT, IF END THEN, IMAGE, INPUT, KEYIN, LINPUT, LIST, LOAD, PRINT, PRINTUSING, REWIND, SAVE, SKIP.

К *матричным операторам* относятся: MAT=, MAT+, MAT–, MAT( )* , MAT* , MAT TRN, MAT INV, MAT REDIM, MAT IDN, MAT CON, MAT ZER, MAT READ, MAT INPUT, MAT PRINT.

Операторы сортировки: MAT CONVERT, MAT COPY, MAT SORT, MAT MOVE, MAT SEARCH, MAT MERGE.

К *дисковым операторам* относятся: DATALOAD DC, DATALOAD DC OPEN, DATASAVE DC, DATASAVE DC CLOSE, DATASAVE DC OPEN, DBACKSPACE, DSKIP, LIMITS, LIST DC, LOAD DC, MOVE, MOVE END, SAVE DC, SCRATCH, SCRATCH DISK, VERIFY, DATALOAD BA, DATALOAD DA, DATASAVE BA, DATASAVE DA, LOAD DA, SAVE DA.

Операторы преобразования данных: ¤TRAN, ¤PACK, ¤UNPACK.

Графическим оператором является оператор PLOT.

Операторы обработки графики: ¤COPY, DOT, DDOT, DRAW, DDRAW, NDOT, NDRAW, FRAME, LABEL, ¤LET, ¤MOVE, ¤OPEN, ORIGIN, NPLOT, ¤PEN, SCALE, SCRATCH, TRANSFORM, TURN, WINDOW.

Рассмотрим теперь более подробно только те операторы алгоритмического языка БЭЙСИК для "Искры 226", которые отсутствуют в варианте 3А языка БЭЙСИК Д3-28, описанного в гл. 2. Основная форма записи каждого рассматриваемого оператора приводится на метаязыке Бэкуса (аналогично прил. 5).

Основные операторы. Оператор DEFFN′ и имеющаяся клавиатура специальных функций позволяют программисту создавать свою собственную клавиатуру для той или иной программы. Общая форма записи оператора:

$$\text{DEFFN}' \ \langle \text{целое число} \rangle \ \left\{ \begin{array}{l} \text{"} \langle \text{цепочка знаков} \rangle \text{"} \\ [(\langle \text{переменная} \rangle [, \langle \text{переменная} \rangle] \ldots)] \end{array} \right\}$$

где

$$\langle \text{целое число} \rangle :: = \left\{ \begin{array}{l} \text{от 00 до 255 – для ссылок на внешние подпрограммы;} \\ \text{от 00 до 31 – для номеров клавиш специальных} \\ \text{функций;} \end{array} \right\}$$

< переменная > – формальный аргумент; " < цепочка знаков >" – любая последовательность знаков, заключенных в кавычки.

Некоторые строки в БЭЙСИК-программе могут встречаться несколько раз. В подобных случаях удобно иметь клавиатуру, позволяющую работать с такими строками, нажимая только одну клавишу.

Оператор DEFFN′, за которым следует целое число и список переменных (или без списка переменных), обозначает начало подпрограммы. Обращение к этой подпрограмме осуществляется оператором GOSUB′, форма записи которого следующая:

$$\text{GOSUB}' \ \langle \text{целое число} \rangle \ \left[\left(\left\{ \begin{array}{l} \text{"} \langle \text{цепочка знаков} \rangle \text{"} \\ \langle \text{символьная переменная} \rangle \\ \langle \text{арифметическое выражение} \rangle \end{array} \right\} \ldots \right) \right]$$

Оператор DEFFN′ должен быть первым в строке, в непосредственном режиме он применяться не может.

Рассмотрим примеры применения оператора DEFFN′.

Оператор

```
10 DEFFN′05 "CARD"
```

определяет специальную функцию – текст "CARD" с помощью клавиши $\boxed{5}$.

Подпрограмма DEFFN′180 вычисляет выражение $\sqrt{a^2 - bc}$ при значениях переменных $a = 10$, $b = 5$, $c = 3$. Она может быть оформлена и вызвана следующим образом:

```
10 INPUT "ВВЕДИТЕ ИСХОДНЫЕ ДАННЫЕ A,B,C"A,B,C
20 GOSUB' 180
30 PRINT Y,A,B,C
35 STOP
40 DEFFN' 180
50 LETY=A*A-B*C
60 IF Y < 1E-12 THEN 80
70 LET Y=SQR(Y)
75 RETURN
80 PRINT "КОРЕНЬ ОТРИЦАТЕЛЬНЫЙ"
90 STOP
100 END
```

Значения переменных A, B, C можно передать через список параметров. При этом операторы с номерами 20 и 40 изменятся следующим образом:

```
20 GOSUB' 180 (10,5,3)
40 DEFFN' 180 (A,B,C)
```

Оператор

$$\text{ON} < \text{арифметическое выражение} > \begin{Bmatrix}\text{GOSUB}\\ \text{GOTO}\end{Bmatrix} < \text{номер строки} > [\{,<\text{номер строки}>\}...]$$

предназначен для организации в БЭЙСИК-программе разветвляющихся переходов.

Например, при K=4 в операторе

10 ON K GOTO 9, 2, 15, 70, 3, 6

произойдет переход на оператор с номером 70, стоящий в списке на 4-м месте.

Оператор SELECT предназначен для выбора адреса устройства ввода/вывода, определения продолжительности паузы после вывода строки и установления единицы измерения величины углов тригонометрических функций. Общая форма записи оператора:

SELECT < параметр > [{, < параметр > }...]

где

$$< \text{параметр} > :: = \begin{cases} \text{CI xxx } [(<\text{длина}>)] \\ \text{CO xxx } [(<\text{длина}>)] \\ \text{TAPE xxx } [(<\text{длина}>)] \\ \\ \# < \text{номер массива} > \text{xxx } \left[\begin{Bmatrix}\text{F}\\ \text{R}\end{Bmatrix}\right] \\ \text{LIST xxx}[(<\text{длина}>)] \\ \text{PRINT xxx}[(<\text{длина}>)] \\ \text{PLOT xxx}[(<\text{длина}>)] \\ \\ \text{DISK xxx}\left[\begin{Bmatrix}\text{F}\\ \text{R}\end{Bmatrix}\right][(<\text{длина}>)] \\ \text{P}[<\text{цифра}>] \\ \text{D} \\ \text{R} \\ \\ \text{G} \end{cases}$$

xxx — адрес устройства ввода/вывода, задаваемый числом в шестнадцатеричной системе счисления (в табл. 5.2 показано соответствие между устройствами ввода/вывода и адресами, записанными в десятичной, шестнадцатеричной и двоичной системах счисления); < длина > — целое число, определяющее длину строки; < номер массива > — логический номер массива, приписываемый внешнему устройству ввода/вывода; < цифра > определяет величину задержки появления на экране дисплея новой строки. Величина задержки может изменяться от 1/6 с до 11/2 с (от P1 до P9); буквы D, R, G устанавливают единицы измерения величин углов соответственно в градусах, радианах и градах (400 град = 2π рад = 360 °).

Таблица 5.2

Устройство ввода/вывода	Адрес устройства ввода/вывода		
	десятичный	шестнадцатеричный	двоичный
Клавиатура	0	000	000000
	1	001	000001
	2	002	000010
	3	003	000011
Дисплей	4	004	000100
	5	005	000101
	6	006	000110
	7	007	000111
Кассетный магнитофон	8	008	001000
	9	009	001001
	10	00A	001010
	11	00B	001011
Печатающее устройство	12	00C	001100
	13	00D	001101
	14	00E	001110
	15	00F	001111
Графопостроитель	20	014	010100
	21	015	010101
	22	016	010110
	23	017	010111
НГМД	24	18	011000
	25	19	011001
	26	1A	011010
Устройство считывания с ПЛ	40	28	101000
	41	29	101001
	42	2A	101010
	43	2B	101011

Например, оператор SELECT TAPE 100 присваивает кассетному магнитофону с адресом 100 функции консольного кассетного устройства; SELECT P9 устанавливает задержку в 11/2 с; SELECT R выбирает в качестве единицы измерения углов в тригонометрических функциях радиан.

Оператор RENUMBER нумерует заново все строки БЭЙСИК-программы. Общая форма записи оператора:

RENUMBER [<номер строки НС1>][,<номер строки НС2>][,<целое число>]

Нумеруются все строки, начиная со строки с номером НС1 по строку с номером НС2 с шагом, равным третьему параметру в списке параметров. Если НС1 не указан,то перенумеровываются все строки программы. Если шаг перенумерации не указан, нумерация строк автоматически осуществляется с шагом 10. Номер строки НС2 определяет номер, присваиваемый строке, с которой начинается новая нумерация. Если НС2 не задан, он определяется самой программой и равен шагу (третьему параметру в списке параметров или числу 10). Все обращения к номерам строк в операторах GOTO, GOSUB, IF-THEN , PRINTUSING корректируются автоматически в соответствии с новой нумерацией строк.

Оператор TRACE предназначен для пошаговой отладки БЭЙСИК-программы с выдачей информации на печать. Общая форма записи оператора:

TRACE[OFF]

Режим отладки действует от оператора TRACE до оператора TRACE OFF. Информация на печать выдается в следующих случаях:

1) переменной в процессе выполнения БЭЙСИК-программы присваивается новое значение; формат выдачи информации:

< переменная >=<новое значение>

2) происходит передача управления; формат выдачи информации:

TRANSFER TO < номер строки>

Оператор COM CLEAR предназначен для переопределения общих массивов (переменных), ранее объявленных оператором COM, в необщие массивы (переменные) или для переопределения необщих массивов (переменных), ранее объявленных оператором DIM , в общие. Общая форма записи оператора:

$$\text{COM CLEAR}\ \left[\left\{\begin{array}{l}\text{< простая цифровая переменная >}\\ \text{< простая символьная переменная >}\\ \text{< метка массива >}\end{array}\right\}\right]$$

Оператор¤ GIO' обеспечивает написание фрагментов БЭЙСИК-программы в машинных кодах (командах) на автокоде ТЭМПО. Общая форма записи оператора:

$$\text{¤GIO'}\left\{\begin{array}{l}\text{< символьная переменная >}\\ \text{< символьная константа >}\\ \text{< метка символьного массива >}\end{array}\right\}\left[\left\{\begin{array}{l}\text{< переменная >}\\ \text{< метка массива >}\end{array}\right\}\right]$$

Оператор ON ERROR предназначен для программной обработки ошибок. Общая форма записи оператора:

$$\text{ON ERROR}\ \left[\text{< }\begin{array}{c}\text{символьная}\\ \text{переменная 1}\end{array}\text{ >, < }\begin{array}{c}\text{символьная}\\ \text{переменная 2}\end{array}\text{ > GOTO < НС >}\right]$$

где <символьная переменная 1> – код ошибки; <символьная перемен-

ная 2 > указывает на номер строки, в которой произошла ошибка при выполнении программы.

Оператор RETURN CLEAR предназначен для стирания адреса возврата из подпрограммы в ОЗУ микроЭВМ. Общая форма записи оператора:

RETURN CLEAR

Операторы преобразования данных. Операторы преобразования данных выполняют различные логические и математические операции с битовой структурой символьной переменной.

Оператор ADD предназначен для двоичного сложения символьных переменных. Общая форма записи оператора:

$$\text{ADD[C]}\left(\text{<символьная переменная>},\left\{\begin{matrix}\text{xx}\\ \text{<символьная переменная>}\end{matrix}\right\}\right)$$

где xx – шестнадцатеричная цифра; C – параметр, указывающий на необходимость переноса бита переполнения в следующий (левый) байт.

Операторы AND, OR и XOR выполняют логические операции соответственно И, ИЛИ и исключающее ИЛИ над символьными переменными. Общая форма записи оператора:

$$\left\{\begin{matrix}\text{AND}\\ \text{OR}\\ \text{XOR}\end{matrix}\right\}\left(\text{<символьная переменная>},\ \left\{\begin{matrix}\text{xx}\\ \text{<символьная переменная>}\end{matrix}\right\}\right)$$

где xx – шестнадцатеричная цифра.

Оператор BIN используется для преобразования целой части арифметического выражения в двоичное число. Общая форма записи оператора:

BIN(<символьная переменная>) = <арифметическое выражение>

где значение выражения находится в диапазоне от 0 до 255. Оператор BIN преобразует целую часть арифметического выражения в символ – один байт, который интерпретируется как двухцифровое шестнадцатеричное число, и присваивает это значение первому символу символьной переменной.

Оператор BOOL позволяет производить одну из 16 логических операций над битовой структурой двух символьных переменных. Общая форма записи оператора:

$$\text{BOOL X}\left(\text{<символьная переменная>},\left\{\begin{matrix}\text{xx}\\ \text{<символьная переменная>}\end{matrix}\right\}\right)$$

Результат действия оператора BOOL присваивается первой символьной переменной. Шестнадцатеричная цифра X определяет код логической операции, которая должна быть выполнена. В табл. 5.3 приведены значения цифры X и соответствующие логические операции.

Оператор CONVERT предназначен для преобразования символьной переменной в число и присвоения значения этого числа цифровой переменной. Общая форма записи оператора:

CONVERT <символьная переменная> TO <цифровая переменная>

Оператор CONVERT может производить и обратное преобразование, т. е. преобразовывать числовое значение арифметического выражения в символьный код по заданному формату и присваивать значение этого кода символьной переменной. Общая форма записи такого оператора:

Таблица 5.3

Значение цифры X	Логическая операция	Значение цифры X	Логическая операция
0	$\equiv 0$	8	$X \wedge Y$
1	$\overline{X \vee Y}$	9	$X \oplus Y$
2	$\overline{X} \wedge \overline{Y}$	A	Y
3	$\overline{X}$	B	$X \rightarrow Y$
4	$\overline{X \rightarrow Y}$	C	X
5	$\overline{Y}$	D	$X \vee \overline{Y}$
6	$X \oplus Y$	E	$X \vee Y$
7	$\overline{X \wedge Y}$	F	$\equiv 1$

CONVERT < арифметическое выражение > TO < символьная переменная > (<формат>)

где

<формат>::=[{± }][#...][.][#...][∧∧∧∧]

Оператор INIT приравнивает все символы переменных или массивов какому-нибудь заданному коду. Общая форма записи оператора:

INIT ({ xx / "символ" / < символьная переменная > }) { < символьная переменная> / < идентификатор символьного массива> } [, { < символьная переменная> / < идентификатор символьного массива> } ...]

где xx – шестнадцатеричная цифра; < идентификатор символьного массива>::=<имя символьного массива > ( ).

Оператор PACK упаковывает численные значения в байты символьных переменных или массивов. Это сокращает объем ОЗУ, необходимый для хранения неупакованных данных. Общая форма записи оператора:

PACK (<формат>) { <символьная переменная> / <идентификатор символьного массива> } FROM { <идентификатор цифрового массива> / <арифметическое выражение> }

где <формат>::= [{± }] [#...] [.] [#...] [∧∧∧∧] ; число знаков # не превосходит 13; < идентификатор массива>::=<имя массива ( )>, например X¤( ).

С упакованными числами нельзя производить вычислений, поэтому предварительно их надо распаковать оператором UNPACK, общая форма записи которого:

UNPACK (<формат>) { <символьная переменная> / <идентификатор символьного массива> } TO { <цифровая переменная> / <идентификатор цифрового массива> }

где значения параметров такие же, что и в операторе PACK.

Оператор ROTATE предназначен для d-кратного циклического сдвига влево каждого разряда символьной переменной; при этом старшие биты становятся на место младших. Перестановка производится во всех символах заданной переменной, включая все пробелы. Общая форма записи оператора:

ROTATE (<символьная переменная>, d)

где d – цифра от 1 до 7.

Операторы ввода/вывода. Большинство приведенных здесь операторов предназначены для работы с МЛ.

Оператор BACKSPACE предназначен для перемотки МЛ назад на заданное число файлов или блоков. Общая форма записи оператора:

$$\text{BACKSPACE}\left[\left\{\begin{matrix}\#n,\\ /xxx,\end{matrix}\right\}\right]\left\{\begin{matrix}BEG\\ n\\ nF\end{matrix}\right\}$$

где #n – логический номер массива, приписанный внешнему устройству; /xxx – адрес кассетного устройства (код); xxx – число в шестнадцатеричной системе счисления; BEG – перемотка МЛ назад к началу файла (установка ее после заглавия); n – перемотка МЛ назад на n логических блоков; nF – перемотка МЛ назад на n файлов; если n = 1, то осуществляется возврат к началу текущего файла (головка считывания устанавливается перед заглавием).

Оператор DATA RESAVE предназначен для обновления логических записей. Общая форма записи оператора:

$$\text{DATA RESAVE}\left[\left\{\begin{matrix}\#n,\\ /xxx,\end{matrix}\right\}\right]\left\{\begin{matrix}\text{OPEN}\,\text{"<имя>"}\\ \text{<список аргументов>}\end{matrix}\right\}$$

где оператор OPEN предназначен для замены старого имени логических записей на новое (1–8 знаков);

$$\text{<список аргументов>}::=\left\{\begin{matrix}\text{"<цепочка знаков>"}\\ \text{<символьная переменная>}\\ \text{<выражение>}\\ \text{<метка массива>}\end{matrix}\right\}$$

Оператор DATALOAD BT считывает с МЛ физическую запись, состоящую из 100 или 256 байтов, и присваивает содержимое записи заданному символьному массиву. Если параметр N не задан, то автоматически устанавливается N = 256. Общая форма записи оператора:

$$\text{DATALOAD BT}\,[(N=\text{<выражение>})]\left[\left\{\begin{matrix}\#n\\ /xxx\end{matrix}\right\},\right]\text{<идентификатор символьного массива>}$$

где N – число байтов в физической записи, равное 100 или 256; < идентификатор символьного массива>::=<имя символьного массива> ( ).

Оператор DATASAVE предназначен для копирования содержимого МЛ. Общая форма записи оператора:

$$\text{DATASAVE BT}[R][(N=\text{<выражение>},[H])]\left[\left\{\begin{matrix}\#n\\ /xxx\end{matrix}\right\},\right]\ \text{<идентификатор символьного массива>}$$

где R – параметр возврата; H – метка заглавной записи (описание остальных параметров было приведено ранее). При указании параметра H на МЛ записывается временная метка, которая указывает на заголовок записи. Параметр R

служит для перезаписи. Перед выполнением новой записи МЛ автоматически возвращается на одну запись назад.

Оператор HEXPRINT предназначен для вывода на печать значений символьной переменной или символьного массива в шестнадцатеричном коде (пробелы тоже печатаются). Общая форма записи оператора:

$$\text{HEXPRINT}\left\{\begin{array}{l}\text{<символьная переменная>}\\ \text{<идентификатор символьного массива>}\end{array}\right\}\left[\theta\left\{\begin{array}{l}\text{<символьная переменная>}\\ \text{<идентификатор символьного массива>}\end{array}\right\}\cdots\right][\{\theta\}]$$

где θ – запятая или точка с запятой.

Если после элемента или идентификатора символьного массива стоит запятая, следующий элемент печатается в начале следующей строки. Если на указанном месте стоит точка с запятой, следующий элемент печатается в той же строке.

Оператор

IF END THEN <номер строки>

предназначен для управления считыванием файлов данных с МЛ.

Рассмотрим пример. Значения элементов массивов X и Y загружаются в ОЗУ с МЛ. Если выполнение оператора IF END THEN завершилось по прочтении записи "КОНЕЦ ЛЕНТЫ", то после оператора IF END THEN выполнится оператор с номером 30, стоящий после ключевого слова, иначе произойдет переход к строке с номером 10.

Возможный вариант программы:

```
 5 DIM X(20), Y(50)
10 DATALOAD X( ),Y( )
20 IF END THEN 30
25 GOTO 10
30 PRINT X(1),Y(1)
40 END
```

Оператор PRINTUSING предназначен для вывода результатов почти в любом желаемом формате, когда программисту не хватает стандартных форм печати (зонной или компактной). Общая форма записи оператора:

PRINTUSING <номер строки> [, {<элемент>θ} ...][;]

где <номер строки> указывает строку, в которой задан формат для печати; θ – запятая или точка с запятой;

$$\text{<элемент>}::=\left\{\begin{array}{l}\text{<выражение>}\\ \text{<символьная переменная>}\\ \text{"<цепочка знаков>"}\end{array}\right\}$$

Оператор PRINTUSING взаимодействует с оператором IMAGE, общая форма записи которого следующая:

%θ [{fθ} ...]

где θ::=<цепочка знаков> (за исключением знака #) или пробелы.

Оператор IMAGE поставляет оператору PRINTUSING спецификацию формата f для вывода строки на печать:

<спецификация формата>::=[±][≠[.]...][.#...] [ΛΛΛΛ]

Рассмотрим пример. Числовой массив данных A(3,3) вначале формируется в цикле FOR-NEXT, затем выводится на печать с применением операторов PRINTUSING и IMAGE.

Возможный вариант программы:

```
 5 DIM A(3,3)
10 L=0.002
15 FOR I=1 TO 3
20 FOR J=1 TO 3
25 L=L+2
30 A(I,J)=L
35 PRINTUSING 40,A(I,J);
40 %+##.###
50 NEXT J
60 PRINT
70 NEXT I
75 PRINT
80 STOP
```

В результате выполнения программы на экран дисплея или на печатающее устройство будет выведен результат:

```
 +2,002   +4.002   +6.002
 +8.002  +10.002  +12.002
+14.002  +16.002  +18.002
```

Оператор

KEYIN<символьная переменная>, <номер строки НС1>, <номер строки НС2>

проверяет, готово ли внешнее устройство к выдаче информационного символа. При готовности внешнего устройства этот символ вводится и помещается в первом байте указанной символьной переменной, затем осуществляется переход к строке с номером НС1. При нажатии одного из ключей специальных функций в первый байт символьной переменной помещается код данного ключа в шестнадцатеричной системе счисления, и осуществляется переход к строке с номером НС2. Если внешнее устройство не готово к выдаче информации, то БЭЙСИК-программа продолжает выполняться со следующего оператора.

Оператор

LINPUT[<символьная константа>,] {<символьная переменная> / <метка символьного массива>}

предназначен для вызова значения указанной символьной переменной (метки символьного массива) с последующим присвоением ей откорректированного значения. Оператор заканчивает действие после нажатия клавиши [CR/LF]. Первоначальное значение символьной переменной можно вызвать на экран дисплея до нажатия клавиши [CR/LF]. Для этого надо нажать подряд клавиши [LINE], [ERASE], [EDIT], [RECALL].

Матричные операторы. В алгоритмическом языке БЭЙСИК для микроЭВМ "Искра 226" есть несколько матричных операторов, которые позволяют работать с матрицами: вводить их, складывать, вычитать, умножать, обращать, обнулять, а также осуществлять другие операции. Применение матричных операторов (см. табл. 5.4) увеличивает в 8–10 раз скорость выполнения матричных операций. При обращении к матричным операторам указывается ключевое слово MAT с последующей мнемонической записью требуемой матричной операции.

Таблица 5.4

№ п/п	Операция	Описание	Пример
1	MAT =	Массив=массив	MAT X=Y
2	MAT+	Массив=массив+массив	MAT A=B+C
3	MAT–	Массив=массив–массив	MAT A=B–C
4	MAT ( )*	Массив= (скаляр) *массив	MAT C= (4)*B
5	MAT*	Массив=массив *массив	MAT C=A*B
6	MAT TRN	Массив=транспонированный массив	MAT A= TRN(C)
7	MAT INV,D	Массив=обратная матрица, определитель	MAT A=INV(X),D
8	MAT REDIM	Переопределение размера массива	MAT REDIM A(X,Y)
9	MAT IDN	Матрица=единичная матрица	MAT C= IDN
10	MAT CON	Каждый элемент массива=1	MAT X=CON
11	MAT ZER	Каждый элемент массива=0	MAT A=ZER
12	MAT READ	Массив=значение величин в операторе READ	MAT READ A
13	MAT INPUT	Ввод элементов массива с клавиатуры	MAT INPUT A
14	MAT PRINT	Печать элементов массива	MAT PRINT A

Умножение двух матриц происходит по правилам линейной алгебры, т. е. при перемножении исходных матриц A и B получаем матрицу C, элементы которой $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$. Матричная операция умножения может быть реализована самим пользователем в следующей программе:

```
10 FOR I=1 TO M
20 FOR J=1 TO M
30 S=0.
40 FOR K=1 TO M
50 S=S+A (I,K)*B (K,J)
60 NEXT K
70 C(I,J)=S
80 NEXT J
90 NEXT I
```

Операторы сортировки. Операторы сортировки предназначены для быстрого поиска, перемещения и обработки данных. В табл. 5.5 приведены эти операторы и пояснено действие и назначение каждого из них.

Дисковые операторы. Дисковые операторы используются для обмена информацией с дискетой. При описании дисковых операторов мы следуем руководству [30].

Все используемые в работе дискеты должны быть форматизованы. В процессе форматизации дискета подразделяется на секторы. Секторам присваи-

Таблица 5.5

№ п/п	Оператор	Назначение
1	MAT CONVERT	Преобразует значения входного числового массива в формат, удобный для сортировки, и запоминает их в выходном символьном (цифровом) массиве
2	MAT COPY	Осуществляет побайтный перенос данных из входного алфавитно-цифрового массива в выходной массив
3	MAT MERGE	Выполняет часть операции объединения, которая часто используется при сортировке
4	MAT MOVE	Берет все или часть данных из входного алфавитно-цифрового массива и помещает их в качестве элементов в другой массив, начиная с заданного элемента выходного массива
5	MAT SEARCH	Просматривает элементы входного массива, ищет группы символьных данных, удовлетворяющих заданным соотношениям, и запоминает положение каждой такой группы в выходном массиве
6	MAT SORT	Берет элементы входного массива и создает массив индексов

вается адрес, и организуется информация, которая необходима при текущей работе с дискетой. Свободная зона сектора заполняется нулями, т. е. стирается предыдущее содержание дискеты.

При описании дисковых операций используются следующие обозначения: F – фиксированный диск; R – съемный диск; T – фиксированный или съемный диск (в зависимости от типа устройства, указанного в адресе); #n – номер файла (целое число или переменная, имеющая значения от 0 до 6); xxx – адрес устройства (в шестнадцатеричной системе счисления); ¤ – контроль "чтение после записи"; P – параметр, задающий защиту файла (программа не может быть распечатана или снова записана).

Оператор DATALOAD DC считывает данные из каталогизированного файла и последовательно присваивает их значения переменным и (или) массивам из списка аргументов. Общая форма записи оператора:

DATALOAD DC [#n,] <список аргументов>

где

$$<\text{список аргументов}> ::= \left\{\begin{array}{l}<\text{символьная переменная}>\\<\text{числовая переменная}>\\<\text{название массива}>()\end{array}\right\}[,\{\ldots$$

При каждом выполнении оператора DATALOAD DC считывается очередная логическая запись. Массивы заполняются по строкам. Если значения присвоены не всем переменным из списка, считывается следующая логическая запись.

Оператор DATALOAD DC OPEN предназначен для открытия каталогизируемого файла. Общая форма записи оператора:

$$\text{DATALOAD DC OPEN}\ \left\{\begin{array}{l}F\\R\\T\end{array}\right\}\ [\#n,]\left\{\begin{array}{l}\text{TEMP}<\text{выражение1}>,<\text{выражение2}>\\<\text{"имя"}>\end{array}\right\}$$

где "имя" – имя открывающего каталогизированного файла; TEMP – временный рабочий файл с адресом начального сектора < выражение1 > и адресом конечного сектора < выражение2 > .

Оператор DATASAVE DC OPEN предназначен для резервирования сектора для вновь создаваемого файла и записи об этом файле в зону указателя каталога (УК). Общая форма записи оператора:

$$\text{DATASAVE DC OPEN}\left\{\begin{matrix}F\\R\\T\end{matrix}\right\}[¤][\#n,]\left\{\begin{matrix}\left\{\begin{matrix}<\text{имя1}>\\<\text{выражение}>\end{matrix}\right\},<\text{имя2}>\\ \text{TEMP},<\text{выражение1}>,<\text{выражение2}>\end{matrix}\right\}$$

где <имя1> – имя каталогизированного файла, вместо которого каталогизируется текущий файл <имя2> (файл <имя1> должен быть заранее отмечен оператором SCRATCH); <выражение> – количество секторов, резервируемых для каталогизируемого файла (последний сектор каждого файла отводится для служебной информации); TEMP – параметр, указывается при каталогизации временного рабочего файла; <выражение1>, <выражение2> задают начальный и конечный секторы временного файла.

Оператор

$$\text{DATASAVE DC}[¤][\#n,]\left\{\begin{matrix}\text{END}\\<\text{список аргументов}>\end{matrix}\right\}$$

где

$$<\text{список аргументов}>::=\left\{\begin{matrix}<\text{символьная переменная}>\\<\text{выражение}>\\<\text{числовая переменная}>\\<\text{название массива}>()\\<\text{цепочка знаков}>\end{matrix}\right\}\left[,\left\{\begin{matrix}\cdots\end{matrix}\right\}\right]$$

записывает на дискету все данные списка аргументов в виде одной логической записи. Каждое числовое значение занимает 9 байтов, каждая символьная переменная – столько байтов, сколько объявлено, плюс еще один байт. По параметру END записывается метка конца файла и изменяется запись в УК. Каталогизированный файл всегда должен завершаться меткой конца.

Оператор

$$\text{DATASAVE DC CLOSE}\left[\begin{matrix}\#n\\\text{ALL}\end{matrix}\right]$$

закрывает один (#n) или все (ALL) открытые файлы. Если параметры не указаны, закрывается только текущий открытый файл.

Оператор DBACKSPACE устанавливает значение текущего адреса равным значению адреса начального сектора первой после метки начала файла логической записи. Общая форма записи оператора:

$$\text{DBACKSPACE}[\#n,]\left\{\begin{matrix}\text{BEG}\\<\text{выражение}>[S]\end{matrix}\right\}$$

где <выражение> задает количество логических записей или секторов (если указан параметр S), которое следует пропустить.

Оператор

$$\text{DSKIP}\,[\#n,]\left\{\begin{matrix}\text{END}\\ \langle\text{выражение}\rangle\,[\,S\,]\end{matrix}\right\}$$

где <выражение> – количество пропускаемых логических записей, пропускает количество записей, которое указано в параметре <выражение>. При указании параметра S пропускается соответствующее число секторов. Оператор DSKIP END устанавливает текущий адрес на метку конца файла, после чего к файлу можно добавлять новые данные.

Оператор LIMITS осуществляет доступ к адресам начального и конечного секторов. Общая форма записи оператора:

$$\text{LIMITS}\ \left\{\begin{matrix}F\\ R\\ T\end{matrix}\right\}\,[\#n,][\langle\text{имя}\rangle,]\,\langle\text{переменная1}\rangle,\langle\text{переменная2}\rangle,\langle\text{переменная3}\rangle$$

где <переменная1> – начальный адрес файла (числовая переменная); <переменная2> – конечный адрес файла (числовая переменная); <переменная3>– количество секторов, использованных для файла, или текущий адрес файла (числовая переменная).

Если в операторе LIMITS задано <имя>, то информация о начальном и конечном адресах файлов берется из УК. В таком случае <переменная3> есть общее количество занятых этим файлом секторов. Если <имя> не задано, информация берется по номеру файла из таблицы устройства. Тогда <переменная3> есть текущий адрес сектора (только для открытого файла). Чтобы определить количество использованных секторов, каждую запись данных следует заканчивать оператором DATASAVE DC END.

Оператор

$$\text{LIST DC}\ \left\{\begin{matrix}F\\ R\\ T\end{matrix}\right\}\left[\begin{matrix}\#n,\\ /xxx,\end{matrix}\right]\left[\left\{\begin{matrix}\langle\text{символьная переменная}\rangle\\ \langle\text{символьная константа}\rangle\end{matrix}\right\}\right]$$

выдает на дисплей или печатающее устройство следующую информацию из УК: количество секторов в УК; адрес очередного доступного сектора в области каталога (ОК); адрес последнего использованного сектора в ОК. Для каждого каталогизированного файла указываются: имя; состояние (файл отмечен – S; не отмечен – пробел); тип (программный файл (P) или файл данных (D)); адреса начального и конечного секторов; текущее количество использованных секторов. Печать каталога заканчивают нажатием клавиши $\boxed{\text{HALT/STEP}}$.

Оператор

$$\text{MOVE}\ \left[\begin{matrix}\#n,\\ /xxx,\end{matrix}\right]\left\{\begin{matrix}FR\\ RF\end{matrix}\right\}$$

переписывает весь каталог с фиксированного диска на съемный (FR) или со съемного на фиксированный (RF). При этом файлы, отмеченные оператором SCRATCH удаляются, а остальные сдвигаются в свободные секторы. Для выполнения оператора требуется 1024 байта свободной ОЗУ.

Оператор

$$\text{MOVE END}\left\{\begin{matrix}F\\R\\T\end{matrix}\right\}\left[\begin{matrix}\#n,\\/xxx,\end{matrix}\right]=<\text{выражение}>$$

применяется для увеличения или уменьшения ОК. Значение <выражение> задает адрес сектора, который является новой границей ОК.

Оператор SAVE DC записывает на дискету программы или их части. В УК заносятся имя, тип и адрес начального сектора файла. Имена файлов в ОК не должны повторяться . Общая форма записи оператора:

$$\text{SAVE DC}\left\{\begin{matrix}F\\R\\T\end{matrix}\right\}[¤]\left[\begin{matrix}(<\text{выражение}>)\\<\text{"имя1"}>\end{matrix}\right]\left[\begin{matrix}\#n,\\/xxx,\end{matrix}\right][P]<\text{"имя2"}>[<\text{номер строки 1}>][,<\text{номер строки 2}>]$$

где <выражение> – количество секторов, резервируемых дополнительно; "имя1" – имя программы, вместо которой на дискету записывается новая, имеющая "имя2" (допускается "имя1"="имя2"); "имя2" – символьная переменная или цепочка знаков в кавычках (содержит от 1 до 8 символов); <номер строки1>, <номер строки2> – номера первой и последней строк соответственно, которые должны быть записаны на дискете.

Чтобы записать программу на дискету после того, как в ОЗУ загружена защищенная программа, необходимо выполнить команду CLEAR без параметров или отключить и вновь включить питание.

Оператор

$$\text{SCRATCH}\left\{\begin{matrix}F\\R\\T\end{matrix}\right\}\left[\begin{matrix}\#n,\\/xxx,\end{matrix}\right]<\text{"имя"}>[,<\text{"имя"}>]...]$$

используется для отметки файлов, которые после этого не могут быть загружены в ОЗУ оператором DATALOAD DC или оператором LOAD DC. Новые файлы могут быть записаны в секторы, занятые такими файлами. При выполнении оператора MOVE отмеченные файлы удаляются из ОК.

Оператор SCRATCH DISK формирует каталог, т. е. отводит место для УК и ОК на выбранной дискете, начиная с нулевого сектора. Остальные секторы УК и ОК заполняются нулями. Общая форма записи оператора:

$$\text{SCRATCH DISK}\left\{\begin{matrix}F\\R\\T\end{matrix}\right\}\left[\begin{matrix}\#n,\\/xxx,\end{matrix}\right][LS=<\text{выражение1}>,]\ \text{END}=<\text{выражение2}>$$

где LS – параметр, задающий количество секторов для УК; <выражение1> – целое число или выражение (1–255) (если параметр LS не указан, ему автоматически присваивается значение 24); <выражение2> – целое число, не превосходящее 9791 и зависящее от типа применяемого НГМД; END – параметр, задающий адрес последнего сектора в ОК. ОК можно расширить командой MOVE END. Размер УК не может быть изменен без реорганизации всего каталога.

Оператор

$$\text{VERIFY}\left\{\begin{matrix}F\\R\\T\end{matrix}\right\}\left[\begin{matrix}\#n,\\/xxx,\end{matrix}\right][(<\text{выражение1}>,<\text{выражение2}>)]$$

контролирует правильность занесения информации на дискету с сектора <выражение1> по сектор <выражение2>. Если начальный и конечный адреса не указаны, то проверяется вся ОК.

Абсолютная адресация секторов. При *абсолютной адресации секторов* программы и данные записываются и считываются с обращением к адресу начального сектора файла. После этого автоматически устанавливается адрес следующего доступного сектора.

Рассмотрим операторы, применяемые в режиме абсолютной адресации. При их описании используются следующие обозначения: <адрес сектора> – выражение или символьная переменная, значение которой указывает адрес начального сектора; L – числовая переменная, которой после выполнения оператора присваивается адрес следующего доступного сектора; L¤ – переменная (длиной 2 байта), которой присваивается двоичный адрес следующего доступного сектора; DA – параметр, задающий режим абсолютной адресации; BA – параметр, задающий режим абсолютной адресации и занесение по физическим записям.

Оператор

$$\text{DATALOAD BA}\left\{\begin{matrix}\text{F}\\ \text{R}\\ \text{T}\end{matrix}\right\}\left[\begin{matrix}\#\text{n},\\ /\text{xxx},\end{matrix}\right](\langle\text{адрес сектора}\rangle,\left\{\begin{matrix}\text{L}\\ \text{L¤}\end{matrix}\right\})\langle\begin{matrix}\text{название символьного}\\ \text{массива}\end{matrix}\rangle()$$

считывает один сектор с указанного диска и последовательно засылает 256 байтов в массив. Если массив не может вместить 256 байтов, то выдается сообщение об ошибке.

Оператор

$$\text{DATALOAD DA}\left\{\begin{matrix}\text{F}\\ \text{R}\\ \text{T}\end{matrix}\right\}\left[\begin{matrix}\#\text{n},\\ /\text{xxx},\end{matrix}\right](\langle\text{адрес сектора}\rangle,\left\{\begin{matrix}\text{L}\\ \text{L¤}\end{matrix}\right\})\langle\text{список аргументов}\rangle$$

где

$$\langle\text{список аргументов}\rangle ::= \left\{\begin{matrix}\langle\text{символьная переменная}\rangle\\ \langle\text{цифровая переменная}\rangle\\ \langle\text{название массива}\rangle()\end{matrix}\right\}\left[,\left\{\cdots\right\}\right]$$

считывает логическую запись и последовательно присваивает считанные значения переменным и (или) массивам из списка переменных (массивы заполняются построчно). Если список аргументов не заполнен, считывается следующая логическая запись. Если при выполнении оператора в записи не хватает данных для присвоения всем переменным из списка, то переменные, стоящие в нем последними, сохраняют прежние значения.

Оператор

$$\text{DATASAVE BA}\left\{\begin{matrix}\text{F}\\ \text{R}\\ \text{T}\end{matrix}\right\}[¤]\left[\begin{matrix}\#\text{n},\\ /\text{xxx},\end{matrix}\right](\langle\text{адрес сектора}\rangle,\left\{\begin{matrix}\text{L}\\ \text{L¤}\end{matrix}\right\})\langle\begin{matrix}\text{название символьного}\\ \text{массива}\end{matrix}\rangle()$$

используется для записи неформатизованных данных. Каждый оператор записывает один сектор без контрольной информации. Если массив содержит бо-

лее 256 байтов, то записываются только первые 256 байтов. Информацию, записанную оператором DATASAVE BA, можно прочитать только с помощью оператора DATALOAD BA.

Оператор DATASAVE DA записывает на дискету одну логическую запись, состоящую из одного или нескольких секторов. Общая форма записи оператора:

$$\text{DATASAVE DA}\left\{\begin{matrix}F\\R\\T\end{matrix}\right\}[\text{¤}]\left[\begin{matrix}\#n,\\/xxx,\end{matrix}\right](<\text{адрес сектора}>,\left\{\begin{matrix}L\\L\text{¤}\end{matrix}\right\})\left\{\begin{matrix}\text{END}\\<\text{список аргументов}>\end{matrix}\right\},$$

где

$$<\text{список аргументов}>::=\left\{\begin{matrix}<\text{ символьная переменная}>\\"<\text{цепочка знаков}>"\\<\text{ выражение}>\\<\text{ название массива}>(\,)\end{matrix}\right\}\left[,\left\{\begin{matrix}\\\dots\end{matrix}\right\}\right]$$

Значения из списка аргументов записываются последовательно, массивы – построчно. Требуемый объем памяти для переменных такой же, как при использовании оператора DATASAVE DC.

Оператор

$$\text{LOAD DA}\left\{\begin{matrix}F\\R\\T\end{matrix}\right\}\left[\begin{matrix}\#n,\\/xxx,\end{matrix}\right](<\text{адрес сектора}>,\left\{\begin{matrix}L\\L\text{¤}\end{matrix}\right\})[<\text{номер строки1}>][,<\text{номер строки2}>]$$

загружает программу, которая записана на дискете, начиная с указанного адреса сектора.

Оператор

$$\text{LOAD DA}\left\{\begin{matrix}F\\R\\T\end{matrix}\right\}\left[\begin{matrix}\#n,\\/xxx,\end{matrix}\right](<\text{адрес сектора}>[,\left\{\begin{matrix}L\\L\text{¤}\end{matrix}\right\}])[<\text{номер строки 1}>][,<\text{номер строки 2}>]$$

стирает из памяти строки программы с позиции <номер строки1> по позицию <номер строки2> и на их место загружает новую программу.

Оператор

$$\text{SAVE DA}\left\{\begin{matrix}F\\R\\T\end{matrix}\right\}\text{¤}\left[\begin{matrix}\#n,\\/xxx,\end{matrix}\right](<\text{адрес сектора}>,\left\{\begin{matrix}L\\L\text{¤}\end{matrix}\right\})[<\text{ НС1}>][,<\text{НС2}>]$$

записывает программу или часть ее на дискету.

Графический оператор. Оператор PLOT предназначен для вывода символьной и (или) графической информации на устройство вывода для графической информации. Общая форма записи оператора:

$$\text{PLOT}[\text{выражение0}]<[\text{ выражение1}],[\text{ выражение2}]\left\{\begin{matrix}U\\D\\R\end{matrix}\right\}>[<\dots>]$$

где "выражение0" задает при построении количество повторений, заданных в треугольных скобках (если "выражение0" отсутствует, то его значение принимается равным 1; диапазон допустимых значений 1–1000); "выражение1" задает приращение Δx по координате x с шагом 0,0635 мм (если "выражение1" отсутствует, то его значение принимается равным 0; диапазон допустимых значений от –1000 до 1000); "выражение2" задает приращение Δy по координате y с шагом 0,0635 мм (если "выражение2" отсутствует, то его значение принимается равным 0; диапазон допустимых значений от –1000 до1000 (используется це-

лая часть выражения)); U – пишущий узел при обработке значений Δx и Δy, определенных параметрами "выражение1", "выражение2", будет поднят; D – пишущий узел будет опущен, т. е. пишущий элемент построит отрезок прямой, соответствующий приращениям по обеим координатам, заданным параметрами "выражение1" и "выражение2" (аргумент D необязательный, т. е. отсутствие D будет воспринято так же, как и его наличие); R – пишущий узел поднимется и возвратится в исходное положение. Аргументы оператора задают перемещение пишущего узла в поднятом или опущенном положении на величину, соответствующую приращениям Δx и Δy относительно текущего положения узла.

6. ПРОГРАММИРОВАНИЕ НА МИКРОЭВМ ДВК-1

6.1. ОБЩИЕ СВЕДЕНИЯ

МикроЭВМ ДВК-1 конструктивно выполнена в виде алфавитно-цифрового дисплея 15ИЭ-00-013 со встроенной одноплатной микроЭВМ "Электроника НМС 11100.1".

Методы адресации и мнемоника машинных команд ДВК-1 совпадают с системой команд мини-ЭВМ "Электроника-60", СМ ЭВМ и др. В области программирования эти ЭВМ различаются лишь количеством команд. Так, в ДВК-1 отсутствуют команды для арифметики с плавающей запятой. Программирование на ЭВМ "Электроника-60", СМ-4, PDP/11 подробно изложено в работах [8, 12, 29, 38]. Ниже приведены лишь некоторые сведения по включению ДВК-1 и список команд.

Версия языка БЭЙСИК для микроЭВМ ДВК-1 соответствует ленточной версии БЭЙСИКа мини-ЭВМ "Электроника-60". Программа-интерпретатор языка БЭЙСИК хранится в ПЗУ. Отметим, что эта версия языка близка к БЭЙСИКу микроЭВМ "Электроника Д3-28". Основные различия состоят в том, что отсутствует оператор COM, а для очистки строк программы вместо оператора CLEAR используется оператор DELETE. Упрощены операторы INPUT и PRINT: первый не предусматривает вывод на экран дисплея имен переменных, а второй не имеет возможности устанавливать различный формат вывода. Другие отличия касаются использования внешних программ и периферийного оборудования. Сообщения об ошибках приведены в прил. 9.

Опишем последовательность действий при подготовке микроЭВМ ДВК-1 к работе:

1) подключить микроЭВМ к сети переменного тока напряжением 220 В, частотой 50 Гц;

2) включить тумблер "СЕТЬ", расположенный на задней стенке микроЭВМ. При этом загораются индикатор режима "ДВК" и индикаторная лампочка "СЕТЬ";

3) нажать клавиши [ДУП], [ЛИН], [РЕД] на клавиатуре дисплея;

4) установить режим работы дисплея. Скорость обмена по последовательному каналу должна быть равна 9600 бит/с;

5) нажать клавишу [ПИТАНИЕ]. Клавиша [ПРОГРАММА] должна быть в отжатом положении. На экране дисплея появится сообщение, заканчивающееся символом @, после чего можно набирать программу в машинных кодах;

6) для работы с БЭЙСИК-интерпретатором необходимо набрать адрес 140000, нажать клавиши [ПРОГРАММА] и [С]. На экране появится сообщение:

@ 0 БЭЙСИК ДВК НЦ

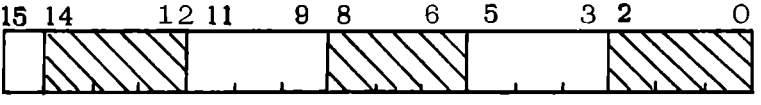
Нажать клавишу [ВК]. На экране появится сообщение: "ЖДУ".

6.2. КРАТКИЕ СВЕДЕНИЯ О СИСТЕМЕ КОМАНД

Как известно, основной единицей ОЗУ является бит. Биты объединяются в байты, а байты – в слова. Каждое слово состоит из 16 бит, пронумерованных от 0 до 15 справа налево. Биты с номерами от 0 до 7 составляют младший байт, с номерами от 8 до 15 – старший байт. Адресом слова, по соглашению, является адрес младшего байта. Для удобства и сокращения записи содержимого слова используют двоично-восьмеричную форму представления чисел.

Например, слово 1010011100101110 запишется как 123456.

Ниже дано графическое представление формата слова:



Центральный процессор состоит из арифметико-логического устройства и набора шестнадцатиразрядных регистров (R). Существует восемь регистров общего назначения (РОН), обозначаемых R0, R1,..., R7. Регистр R7 является также счетчиком команд (PC), содержащим адрес следующей команды, которая будет выполняться. Регистр состояния процессора (PS) содержит набор кодов условий, значения которых зависят от результата последней выполненной команды. Имеются регистры, связанные с периферийными устройствами.

Ниже приведен справочный материал по системе команд и адресации ДВК-1. Обозначения на рисунках соответствуют обозначениям в таблицах.

Введем следующие обозначения (символ "::=" означает "определяется как"):

к кодам команд

■ ::=0 для слова/1 для байта;
SS ::=поле источника (6 бит);
DD ::=поле приемника (6 бит);
R ::=общий регистр (3 бита), смещение от 0 до 7;
XXX ::=смещение (8 бит) от –128 до +127;
N ::=число (3 бита);
NN ::=число (6 бит);

к операциям

( ) ::=указатель содержимого;
s ::=содержимое источника;
d ::=содержимое приемника;
r ::=содержимое регистра;
← ::=пересылка;
X ::=относительный адрес;
% ::=регистр;

к логическим операциям

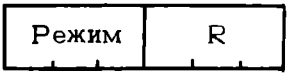
∧ ::=И;
∨ ::=ИЛИ;
⊻ ::=исключающее ИЛИ
¬ ::=НЕТ;

к кодам условий

* ::=устанавливается по результату;
– ::=не изменяется;
1 ::=устанавливается единица;
0 ::=устанавливается нуль.

Например, общая запись команды очистки: ■050DD, а конкретная может быть 005003, что означает: занесение нуля в R3.

Система адресации включает различные способы образования адреса и широко использует регистры общего назначения. Формат поля РОН следующий:



Адрес операнда в команде задается полем из 6 бит, а при индексной адресации – еще и адресным словом из 16 бит. В двухоперандных командах используется два поля и может быть два адресных слова. Методы адресации приведены в табл. 6.1. (Отметим, что последние 4 режима адресации заданы в регистре 7.)

Таблица 6.1

Метод адресации (режим)		Символика	Описание
Восьмеричный код	Наименование		
0	Регистровый	R	(R)::=операнд, т. е. регистр R содержит операнд
1	Косвенно-регистровый	(R)	(R)::=адрес
2	Автоинкрементный	(R)+	(R)::=адрес; (R)+(1 или 2)
3	Косвенно-автоинкрементный	@(R)+	(R)::=адрес адреса; (R)+2
4	Автодекрементный	–(R)	(R)–(1 или 2); (R)::=адрес
5	Косвенно-автодекрементный	@–(R)	(R)–2; (R)::=адрес адреса
6	Индексный	X(R)	(R)+X::=адрес
7	Косвенно-индексный	@X(R)	(R)+X::=адрес адреса
2	Непосредственный	#A	Операнд A следует за командой
3	Абсолютный	@#A	Адрес операнда A следует за командой
6	Относительный	A	(PC)+4+X ::=адрес
7	Косвенно-относительный	@A	(PC)+4+X ::=адрес адреса

Используемые в ДВК-1 команды сведены в табл. 6.2–6.8 по своему функциональному назначению и в табл. 6.9 по адресам. Перед некоторыми таблицами приведены графические иллюстрации форматов команд.

Одноадресные команды даны в табл. 6.2. Формат таких команд:

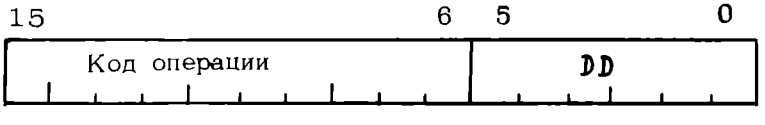


Таблица 6.2

Команда			Операция	Признак			
Мнемоника	Код	Наименование		N	Z	V	C
		Общие					
CLR(B)	▪050DD	Очистка слова (байта)	0	0	1	0	0
COM (B)	▪051DD	Дополнение слова (байта)	$\neg d$	*	*	0	1
INC(B)	▪052DD	Увеличение слова (байта)	$d+1$	*	*	*	–
DEC(B)	▪053DD	Уменьшение слова (байта)	$d-1$	*	*	*	–
NEG(B)	▪054DD	Отрицание слова (байта)	$-d$	*	*	*	*
TST(B)	▪057DD	Проверка слова (байта)	d	*	*	0	0
		Циклические и сдвига					
ROR(B)	▪060DD	Циклический сдвиг вправо	$\rightarrow C,d$	*	*	*	*
ROL(B)	▪061DD	Циклический сдвиг влево	$C,d\leftarrow$	*	*	*	*
ASR(B)	▪062DD	Арифметический сдвиг вправо	$d/2$	*	*	*	*
ASL(B)	▪063DD	Арифметический сдвиг влево	$2d$	*	*	*	*
SWAB	0003DD	Перестановка байтов		*	*	*	0
		Для работы с удлиненными операндами					
ADC(B)	▪055DD	Прибавление переноса	$d+C$	*	*	*	*
SBC(B)	▪056DD	Вычитание переноса	$d-C$	*	*	*	*
SXT	0067DD	Расширение знака	0 или 1	–	*	*	–

Двухадресные команды приведены в табл. 6.3. Формат таких команд:

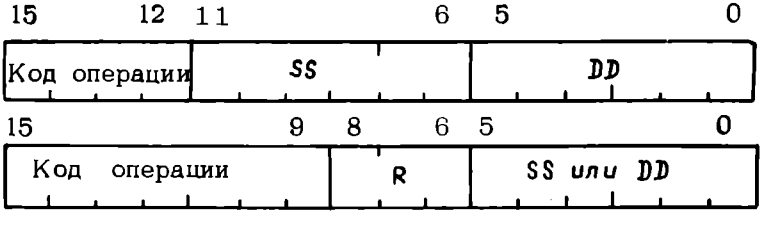
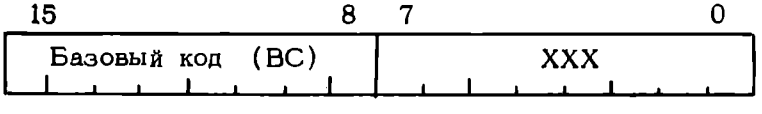


Таблица 6.3

Команда			Операция	Признак			
Мнемоника	Код	Наименование		N	Z	V	C
		Общие					
MOV(B)	■1SSDD	Пересылка	$d \leftarrow s$	*	*	0	–
CMP(B)	■2SSDD	Сравнение	$s - d$	*	*	*	*
ADD	06SSDD	Сложение	$d \leftarrow s + d$	*	*	*	*
SUB	16SSDD	Вычитание	$d \leftarrow d - s$	*	*	*	*
		Логические					
BIT(B)	■3SSDD	Проверка байтов (И)	$s \wedge d$	*	*	0	–
BIC(B)	■4SSDD	Очистка байтов	$d \leftarrow (\neg s) \wedge d$	*	*	0	–
BIS(B)	■5SSDD	Установка бит (ИЛИ)	$d \leftarrow s \vee d$	*	*	0	–
		Регистровые					
MUL	070RSS	Умножение	$r \leftarrow r \times s$	*	*	0	*
DIV	071RSS	Деление	$r \leftarrow r/s$	*	*	*	*
ASH	072RSS	Арифметический сдвиг		*	*	*	*
ASHC	073RSS	Сдвиг в двойном регистре		*	*	*	*
XOR	074RDD	Исключающее ИЛИ	$d \leftarrow r \forall d$	*	*	0	–

Команды ветвления приведены в табл. 6.4. Формат таких команд следующий:



Код перехода определяется как BC+XXX. Если условия перехода выполняются, то (PC) ← (PC)+2*XXX. При выполнении сложения PC указывает на слово, следующее за командой перехода, т. е. фактически увеличенное на 2.

Таблица 6.4

Команда			Условие перехода	Признак
Мнемоника	Базовый код	Наименование		
		Ветвления		
BR	000400	Переход безусловный	Всегда существует	
BNE	001000	Переход, если не равно 0	≠0	Z=0
BEQ	001400	Переход, если равно 0	=0	Z=1
BPL	100000	Переход, если плюс	+	N=0
BMI	100400	Переход, если минус	–	N=1
BVC	102000	Переход, если нет переполнения		V=0
BVS	102400	Переход, если переполнение		V=1
BCC	103000	Переход, если переноса не было		C=0
BCS	103400	Переход, если был перенос		C=1
		Ветвления с учетом знаков операндов		
BGE	002000	Переход, если больше или равно	≥0	N∨V=0
BLT	002400	Переход, если меньше нуля	<0	N∨V=1
BGT	003000	Переход, если больше нуля	>0	Z∨(N∨V)=0
BLE	003400	Переход, если меньше или равно	≤0	Z∨(N∨V)=1
		Ветвления без учета знаков операндов		
BHI	101000	Переход, если выше	>	C∨Z=0
BLOS	101400	Переход, если ниже или одинаково	≤	C∨Z=1
BHIS	103000	Переход, если выше или одинаково	≥	C=0
BLO	103400	Переход, если ниже	<	C=1

Команды переходов через большие участки памяти, команды прерываний и разные команды описаны в табл. 6.5–6.7 соответственно.

Таблица 6.5

Команда			Примечания
Мнемоника	Код	Наименование	
JMP	0001DD	Переход безусловный	PC ← *d*
JSR	004RDD	Переход к подпрограмме	Используется один и тот же регистр R
RTS	00020R	Возврат из подпрограммы	
MARK	0064NN	Сброс содержимого стека	Используется в работе RTS
SOB	077RNN	Вычитает 1 и осуществляет переход, если ≠0	(R)–1; затем, если (R) ≠ ≠ 0, то PC ←PC – 2×NN

Таблица 6.6

Команда			Примечания
Мнемоника	Код	Наименование	
EMT	104000–104377	Командное прерывание (не для общего пользования)	PC=30, PS=32
TRAP	104400–104777	Командное прерывание	PC=34, PS=36
BPT	000003	Специальное прерывание по точке останова программы	PC=14, PS=16
IOT	000004	Специальное прерывание для операции ввода/вывода	PC=20, PS=22
RTI	000002	Возврат из прерывания	
RTT	000006	Возврат из прерывания	Запрещает трассировку (бит T в PS)

Таблица 6.7

Мнемоника	Код	Наименование
HALT	000000	Останов
WAIT	000001	Ожидание прерывания
RESET	000005	Сброс устройств
NOP	00240	Нет операции

Команды установки кодов условий даны в табл. 6.8. Формат таких команд:

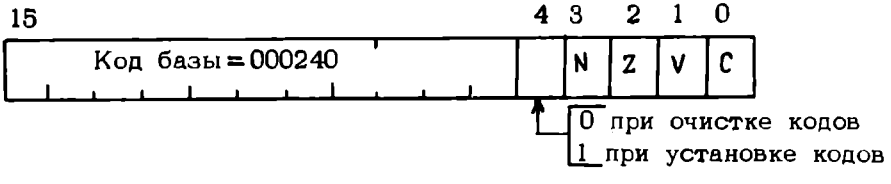
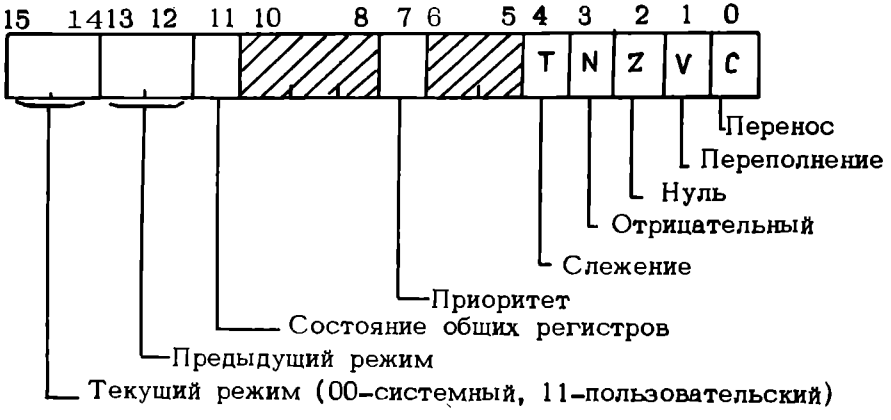


Таблица 6.8

Команда			Признак			
Мнемоника	Код	Наименование	N	Z	V	C
CLC	000241	Очистка C	–	–	–	0
CLV	000242	Очистка V	–	–	0	–
CLZ	000244	Очистка Z	–	0	–	–
CLN	000250	Очистка N	0	–	–	–
CCC	000257	Очистка всех битов условий	0	0	0	0
SEC	000261	Установка C	–	–	–	1
SEV	000262	Установка V	–	–	1	–
SEZ	000264	Установка Z	–	1	–	–
SEN	000270	Установка N	1	–	–	–
SCC	000277	Установка всех битов условий	1	1	1	1

Перечислим адреса некоторых регистров.

1. Слово состояния процессора PS=177 776. Его формат:



2. Общие регистры (используются только консолью):

R0 – 177 700, R4 – 177 704,
R1 – 177 701, R5 – 177 705,
R2 – 177 702, R6 – 177 706,
R3 – 177 703, R7 – 177 707.

3. Регистр консоли и дисплея – 177 560.

Список кодов команд в порядке возрастания дан в табл. 6.9.

Таблица 6.9

Код	Мнемоника	Код	Мнемоника	Код	Мнемоника
00 00 00	HALT	00 55 DD	ADC	10 34 XXX	BCS,BLO
00 00 01	WAIT	00 56 DD	SBC	10 40 00	
00 00 02	RTI	00 57 DD	TST	.	EMT
00 00 03	BPT	00 60 DD	ROR	.	
00 00 04	IOT	00 61 DD	ROL	10 43 77	
00 00 05	RESET	00 62 DD	ASR	10 44 00	
00 00 06	RTT	00 63 DD	ASL	.	TRAP
00 00 07	Не используется	00 64 NN	MARK	.	
00 00 77		00 65 00		10 47 77	
00 01 DD	JMP	.	Не используется	10 50 DD	CLRB
00 02 0R	RTS	.		10 51 DD	COMB
00 02 10	Не используется	.		10 52 DD	INCB
.		00 66 77		10 53 DD	DECB
.		00 67 DD	SXT	10 54 DD	NEGB
.		00 70 00	Не используется	10 55 DD	ADCB
00 02 27		.		10 56 DD	SBCB
00 02 30	Не используется	.		10 57 DD	TSTB
.		.		10 60 DD	RORB
.		00 77 77		10 61 DD	ROLB
.		01 SS DD	MOV	10 62 DD	ASRB
00 02 37		02 SS DD	CMP	10 63 DD	ASLB
00 02 40	NOP	03 SS DD	BIT	10 64 00	Не используется
00 02 41	Коды условий	04 SS DD	BIC	.	
.		05 SS DD	BIS	.	
.		06 SS DD	ADD	10 77 77	
.		07 0R SS	MUL	11 SS DD	MOVB
00 02 77		07 1R SS	DIV	12 SS DD	CMPB
00 03 DD	SWAB	07 2R SS	ASH	13 SS DD	BITB
00 04 XXX	BR	07 3R SS	ASHC	14 SS DD	BICB
00 10 XXX	BNE	07 4R DD	XOR	15 SS DD	BISB
00 14 XXX	BEQ	07 50 00	Не используется	16 SS DD	SUB
00 20 XXX	BGE	.		17 00 00	Не используется
00 24 XXX	BLT	.		.	
00 30 XXX	BGT	.		.	
00 34 XXX	BLE	07 67 77		17 77 77	
00 4R DD	JSR	07 7R NN	SOB		
00 50 DD	CLR	10 00 XXX	BPL		
00 51 DD	COM	10 04 XXX	BMI		
00 52 DD	INC	10 10 XXX	BHI		
00 53 DD	DEC	10 14 XXX	BLOS		
00 54 DD	NEG	10 20 XXX	BVC		
		10 24 XXX	BVS		
		10 30 XXX	BCC,BHIS		

За рядом кодов внутренних прерываний закреплены постоянные ячейки. Перечислим их:

000 – зарезервированный;

004 – прерывание по ошибке обращения к каналу;

010 – прерывание по резервной команде;

014 – прерывание по Т-разряду;

020 – прерывание по команде IOT;

024 – прерывание по нарушению питания;

030 – прерывание по команде EMT;

034 – прерывание по команде TRAP.

Абсолютный загрузчик осуществляет загрузку памяти с начального адреса XXX500. В зависимости от объема памяти XXX принимает значения: 4 К – 017; 8 К – 037; 12 К – 057; 16 К – 077; 20 К – 117 ; 24 К – 137; 28 К (и больше) – 157.

Проиллюстрируем написание программы, ее ввод и выполнение на следующем примере.

Пример 6.1. Занести в массив пять чисел: 0,1,2,3,4, найти их сумму, предусмотреть занесение в тот же массив следующих пяти чисел: 5,6,7,8,9 и продолжить суммирование. Массив расположить с адреса 1000, а суммирование произвести в ячейке с адресом 2000. Программу занести с адреса 3000.

Текст программы с комментариями приведен в табл. 6.10.

Таблица 6.10

Адрес	Код	Мнемоника	Описание
3000	005000	CLR RO	Занести 0 в R0
3002 3004	005037 002000	A: CLR @#2000	Занести 0 по адресу 2000
3006 3010	012701 001000	MOV 1000,R1	В R1 занести 1000
3012 3014	012702 000005	MOV 5,R2	В R2 занести 5
3016	010011	B: MOV R0,(R1)	Переслать содержимое R0 по адресу в R1 (т.е. 1000)
3020 3022	062137 002000	ADD (R1)+, @2000	Сложить содержимое адреса R1 с содержимым адреса 2000 и адрес в R1 увеличить на 2
3024	005200	INC R0	Увеличить на 1 содержимое R0
3026	077205	SOB R2,.–10 (или SOB R2,B)	Увеличить на 1 содержимое R0, вычесть 1 из R2 и перейти на 8 команд назад (или выполнить то же и перейти к метке B)
3030	000000	HALT	Останов
3032	000765	BR A	Перейти к метке A

При вводе программы с клавиатуры дисплея набираются два левых столбца табл.6.10. При этом набором текста "3000/" открывается первая ячейка программы. После ввода наклонной черты машина выведет содержимое ячейки 3000. Далее следует набрать 005000 и нажать клавишу ПС . Номера следующих ячеек и их содержимое будут выведены автоматически. Пользователю необходимо лишь набирать команду и нажимать клавишу ПС . Нули слева в кодах команд можно опускать.

Для запуска программы следует набрать на клавиатуре дисплея "3000 G". При этом должна быть нажата клавиша ПРОГРАММА . Результат можно посмотреть, открыв ячейку 2000, т. е. набрав на клавиатуре: "2000/".

Отметим, что новые модели ДВК полностью содержат приведенную выше систему команд. Как уже указывалось в начале главы, подробное описание архитектуры микроЭВМ ДВК-1,команд и примеров составления программ можно найти в упомянутой там литературе, а также в других источниках, посвященных семействам ЭВМ типа СМ, "Электроника-60", PDP/11.

ПРИЛОЖЕНИЯ

1. Команды машинно-ориентированного языка микроЭВМ "Электроника Д3-28" в алфавитном порядке

Мнемокод	Код B1 A1	Код B2 A2	Индексация	Время выполнения	Примечание
ABGE Ri,Rj	14 09	i j		55	
ABS Ri	04 13	08 i		28	
ABS X	06 07			23	
ACS	08 06			55500	Функция в радианах
ADD Ri, Rj	11 00	i j		62	
ADD Ri, @Rj	09 00	i j	(BD)	73	
ADD X,C	04 00	B2 A2	(BD)′	750	C = 10·B2+A2
ADD X,Y	06 00			600	
ADD X,@Y	05 00		(BD)′	770	
ADD #10,E	04 12	07 00		54	
ADD #*e*,E	04 12	07 *e*		54	$1 \leq e \leq 9$
ADD #*e*,Ri	10 00	*e* i		33	
AHC	08 14			49000	
AHS	08 13			49000	
AHT	08 15			28500	
AND Si,Sj	11 08	i j		37	
AND Si, @Rj	09 08	i j	(BD)	48	
ANS Si,Sj	10 06	i j		31	
ASN	08 05			50300	Функция в радианах
ATN	08 07			15500	То же
ATOI *d*	10 03	*d*		200	
BBIC i,@Rj	14 15	i j	(BD)	49	
BBIS i, @Rj	14 15	i+8j	(BD)	49	
BEQ Y,X	05 09			680	
BEQZ X	04 12	06 11		32	
BEQZ Y	04 12	04 11		32	
BEV @Ri.+*e*	14 07	*e*−1 i	(BD)	54	
BGE Ri,Rj	14 10	i j		50	
BGE Y,X	05 07			680	
BHIS Si,Sj	10 02	i j		36	
BKEY.+ *d*	14 11	B2 A2		22	*d* = 16·B2 + A2 + 1
BLT Y, X	05 08			680	
BMI X	04 12	07 10		32	
BMI Y	04 12	05 10		32	
BNS #*d*,S1	10 08	*d*		28	

Мнемокод	Код B1 A1	Код B2 A2	Индексация	Время выполнения	Примечание
BNS #*d*,S3	10 09	*d*		28	
BNEZ X	04 12	07 11		32	
BNEZ Y	04 12	05 11		33	
BMER .+*d*	14 14	B2 A2		22	*d* = 16·B2 + A2 + 1
BPER	05 10			24	
BPL X	04 12	06 10		32	
BPL Y	04 12	04 10		32	
BR .+*d*	14 03	B2 A2		20	*d* = 16 · B2 + A2 + 1
BR .–*d*	14 02	B2 A2		22	*d* = 16 · B2 + A2 – 1
BSA Y,X	12 04			51	
BSA Si,Sj	11 15	i j		37	
BSA Si, @Rj	09 15	i j	(BD)	48	
BSAZ @Ri .+*e*	09 03	*e*–1 i	(BD)	58	
BSAZ Ri .+*e*	11 03	*e*–1 i		47	
BSA Ri,Rj	11 07	i j		49	
BSA Ri, @Rj	09 07	i j	(BD)	60	
CAP	08 08			64600	Угол в радианах
CLDRS	04 12	12 05		31	
CLR Ri	04 13	10 i		33	
CLR X	07 15			66	
CMD Ri	04 13	02 i		16+*t* команды	
COM Si	11 10	B2 i		35	B2 – любое
COM @Ri	09 10	B2 i	(BD)	46	То же
COS	08 03			62500	Аргумент в радианах
CRFS	04 12	12 13		31	
DEG	08 01			4700	
DIG *e*	07 0*e*			123 для мантиссы, 40 для порядка	0 ≤ *e* ≤ 9
DIV X,C	04 03	C	(BD)′	4650	C = 10 · B2+A2
DIV X,Y	06 03			4500	
DIV X, @Y	05 03		(BD)′	4670	
ELMG	04 12	12 11		28	
END	05 12			22	
E	07 10			36	
EXP	06 14			23000	
EXT	06 13			18500	
FORW	04 12	12 09		23	Время до включения перемотки
GO	05 14			31	
GR1 *d*	04 09	*d*		47	УПР=00 04; *d*= < адрес ПУ >
GR2 *d*	04 10	*d*		47	УПР=00 05; *d*= < адрес ПУ >
HCS	08 11			32800	

Мнемокод	Код B1 A1	Код B2 A2	Индексация	Время выполнения	Примечание
HSN	08 10			32700	
HTN	08 12			32800	
INFS	04 12	12 07		31	
INPS *d*	15 00	*d*	(BD)	65+K(15+τ)	K – количество вводимых в ОЗУ байтов; τ – время ожидания ответа ПУ (не ограничено)
INPAS *d*	15 08	*d*	(BD)	81+4τ+K(15+τ)	
INPAR *d*	15 12	*d*	(BD)	92+4τ+K(21+τ)	
INPARV *d*	15 14	*d*	(BD)	100+5τ+K(27+τ)	
INPASV *d*	15 10	*d*	(BD)	89+5τ+K(21+τ)	
INPR *d*	15 04	*d*	(BD)	76+K(21+τ)	
INPRV *d*	15 06	*d*	(BD)	84+τ+K(27+τ)	
INPSV *d*	15 02	*d*	(BD)	73+τ+K(21+τ)	
INPO *d*	14 00	*d*	(BD)	54+τ	
INPOWC	04 12	14 04		39+τ	τ ≤ (R10)·10
INPOWS	04 12	14 06		42+τ	τ ≤ (R10)·10
INT	06 08			65	
INV	06 15			4600	
JMM *d*	04 07	*d*	(BP)	26+4K	K – количество просмотренных байтов
JMP @Ri	04 13	00 i	(BP)	47	
JMP @X	12 13		(BP)	166	
JSM B1 A1	B1 A1		(BP)	125+4K	B1 ≤ 1, K – количество просмотренных байтов
JSM B1 A1	B1 A1		(BP)	58+4K	2<B1≤3, K – количество просмотренных байтов
JMTT *d*	10 10	*d*		39	
JMTF *d*	10 11	*d*		39	
JSR @Ri	04 13	01 i	(BP)	79	
JSTT *d*	10 13	*d*		63	
JSTF *d*	10 14	*d*		63	
LAMP	04 12	12 02		29	
LGT	06 10			13550	
LNCN	04 12	14 00		35	
LOADP	05 13			260000+2250K+ $+\tau_{рак}$	K – количество считываемых байтов; $\tau_{рак}$ – время выборки ракорда
LOADRi	04 12	10 i		516000min	
LOADX	12 02		(BP)	260000+2250K+ $+\tau_{рак}$	K – количество считываемых байтов; $\tau_{рак}$ – время выборки ракорда
LOG	06 11			18000	
MARK *d*	04 08	*d*		23	
MOV C,X	04 05	C	(BD)′	225	C = 10·B2 + A2
MOV C,Y	04 15	C	(BD)′	225	C = 10·B2 + A2

Мнемокод	Код B1 A1	Код B2 A2	Индексация	Время выполнения	Примечание
MOV BD,Ri	04 13	13 i		33	
MOV BP,Ri	04 13	14 i		33	
MOV M,Si	04 13	15 i		32	
MOV Ri,BD	04 13	05		37	
MOV Ri,BP	04 13	06 i		37	
MOVD Ri,X	04 13	04 i		400	
MOV Ri,X	04 13	03 i		71	
MOV Ri,Rj	11 04	i j		51	
MOV Ri,Tj	14 12	i j		33	
MOV Ri,–(Rj)	10 12	i j		51	
MOV X,N	04 12	07 14		40	
MOV X,Ri	04 13	11 i		60	
MOV X,RR	12 06			115	
MOV X,Y	06 04			91	
MOV X,(Ri)	04 12	02 i		134	
MOV Y,(Ri)	04 12	00 i		134	
MOV X,@Y	05 04		(BD)′	243	
MOV Y,C	04 14	C	(BD)′	225	C = 10·B2 + A2
MOV Y,X	06 05			91	
MOV Y,Z	12 14			68	
MOV @Y,X	05 05		(BD)′	243	
MOV Z,Y	12 15			68	
MOV #*d*,Si	13 i	*d*		26	
MOV Ri, @Rj	09 04	i j	(BD)	62	
MOV RR,X	12 07			89	
MOV @Ri,Rj	09 05	j i	(BD)	59	
MOV (Ri)+, Rj	10 15	j i		51	
MOV @Ri,Sj	09 13	j i	(BD)	47	
MOV (Ri),X	04 12	03 i		111	
MOV (Ri),Y	04 12	01 i		111	
MOV Si,M	04 13	07 i		32	
MOV Si,Sj	11 12	i j		36	
MOV Si, @Rj	09 12	i j	(BD)	47	
MOV Ti,Rj	14 13	i j		33	
MOVH X,Ri	04 13	12 i		170	
MOV X,C	04 04	C	(BD)′	225	C = 10·B2 + A2
MUL Ri,Rj	11 02	i j		420	
MUL Ri, @Rj	09 02	i j	(BD)	431	
MUL X,C	04 02	C	(BD)′	4650	C = 10·B2 + A2
MUL X,Y	06 02			4500	
MUL X,@Y	05 02		(BD)′	4670	
NEG Ri	04 13	09 i		28	
NEG X	07 11			30	
NORM	12 09			220	
NSS *d*	10 05	*d*		44+15K	K – количество просмотренных байтов
NSN *d*	10 04	*d*		44+15K	
OR Si,Sj	11 11	i j		38	
OR Si,@Rj	09 11	i j	(BD)	49	

Мнемокод	Код B1 A1	Код B2 A2	Индексация	Время выполнения	Примечание
OUTS *d*	15 01	*d*	(BD)	65+K (14+τ)	К – количество выводимых из ОЗУ байтов; τ – время ожидания ответа ПУ (СИП)
OUTAS *d*	15 09	*d*	(BD)	81+4τ+K (14+τ)	
OUTAR *d*	15 13	*d*	(BD)	92+4τ+K (28+τ)	
OUTARV *d*	15 15	*d*	(BD)	100+5τ+K (26+τ)	
OUTASV *d*	15 11	*d*	(BD)	89+5τ+K (28+τ)	
OUTR *d*	15 05	*d*	(BD)	75+K (20+τ)	
OUTRV *d*	15 07	*d*	(BD)	84+τ+K (26+τ)	
OUTSV *d*	15 03	*d*	(BD)	73+τ+K (20+τ)	
OUTO *d*	14 01	*d*	(BD)	55+τ	
OUTOWC	04 12	14 05		39+τ	τ ≤(R10)·10
OUTOWS	04 12	14 07		48+τ	τ ≤(R10)·10
PAUSE	04 12	06 15		479292	
PAUSER	04 12	14 02		35+ (R10) 10	
PI	06 09			53	
POC	08 09			72000	В радианах
POINT	07 12			77	
PRINT *d*	04 11	*d*		200+19τ	τ – время ожидания ответа ПМ
PRINT #*d*	14 06	*d*		37+τ	
PRIOR Si,Rj	10 07	i j		38	
QRT	07 13			4660	
RAD	08 00			9200	
RES	07 14			95	
REW	12 00			20	Время до включения перемотки
RTI	12 11			170	
RTII	12 10			30	
RTS	05 11			50	
RTSGO	04 12	07 12		65	
RTSI	12 12			28	
SAVERi	04 12	11 i		516000 min	
SAVEX	12 03		(BP)	320000+2250K+ $+\tau_{рак}$	К – количество записываемых байтов; $\tau_{рак}$ – время выборки ракорда
SAVC	04 12	12 04		28	
SAVS	04 12	12 12		28	
SMER	04 12	12 10		29	
SQR	06 12			25000	
SIN	08 02			67000	Аргумент в радианах
SNCS	04 12	12 06		31	
SOBZ Ri. +*e*	14 08	*e*–1 i		39	
SPCMD 05 12	04 12	05 12		36	
SPCMD 05 13	04 12	05 13		36	
SPCMD 05 14	04 12	05 14		65	
SPCMD 05 15	04 12	05 15		4500	
SPCMD 06 12	04 12	06 12		439	
SPCMD 06 13	04 12	06 13		453	
SPCMD 06 14	04 12	06 14		36	
SPCMD 07 13	04 12	07 13		34	

Мнемокод	Код B1 A1	B2 A2	Индексация	Время выполнения	Примечание
SPCMD 07 15	04 12	07 15		4548	
STOP	05 15				
STTAP	04 12	12 00		28	
SUB Ri,Rj	11 01	i j		65	
SUB Ri,@Rj	09 01	i j	(BD)	76	
SUB X,Y	06 01			600	
SUB X,C	04 01	C	(BD)′	750	C = 10·B2 + A2
SUB X,@Y	05 01		(BD)′	770	
SUB #10,E	04 12	04 00		54	
SUB #*e*,E	04 12	04 *e*		54	$1 \leq e \leq 9$
SUB #*e*,Ri	10 01	*e* i		33	
SWA Ri	04 12	08 i		39	
SWA Ri,Rj	11 06	i j		52	
SWA Ri, @Rj	09 06	i j	(BD)	63	
SWA Si	04 12	09 i		33	
SWA Si,Sj	11 14	i j		39	
SWA Si, @Rj	09 14	i j	(BD)	50	
SWA X,C	04 06	C	(BD)′	274	C = 10·B2 + A2
SWA X,Y	06 06			140	
SWA X,@Y	05 06		(BD)	292	
TAN	08 04			32300	Аргумент в радианах
TRAP	12 05			38	
TRTAP	04 12	12 03		28	
VERR B2 A2	14 04	B2 A2	(BD) 99+19K		К – количество контролируемых байтов
VERX	12 01		(BP) 197+24K		
VEX B2 A2	14 05	B2 A2	(BD) 93+10K		
WAIT	12 08			230	
WTRT	04 12	14 03		535+K(R10)·10	К – количество считываемых бит
XOR Si, Sj	11 09	i j		37	
XOR Si, @Rj	09 09	i j	(BD)	48	

2. Команды машинно-ориентированного языка микроЭВМ "Электроника Д3-28" в порядке возрастания значений их кодов

Код	Мнемокод	Код	Мнемокод
00 00	JSM 00 00	04 12 05 14	SPCMD 05 14
...	...	04 12 05 15	SPCMD 05 15
01 15	JSM 01 15	04 12 06 00	–
02 00	JSM 02 00	...	...
...	...	04 12 06 09	–
03 15	JSM 03 15	04 12 06 10	BPL X
04 00 B2 A2	ADD X,C	04 12 06 11	BEQZ X
04 01 B2 A2	SUB X,C	04 12 06 12	SPCMD 06 12
04 02 B2 A2	MUL X,C	04 12 06 13	SPCMD 06 13
04 03 B2 A2	DIV X,C	04 12 06 14	SPCMD 06 14
04 04 B2 A2	MOV X,C	04 12 06 15	PAUSE
04 05 B2 A2	MOV C,X	04 12 07 00	ADD #10,E
04 06 B2 A2	SWA X,C	04 12 07 01	ADD #01,E
04 07 B2 A2	JMM B2 A2	...	...
04 08 B2 A2	MARK B2 A2	04 12 07 09	ADD #09,E
04 09 B2 A2	GR B2 A2	04 12 07 10	BMI X
04 10 B2 A2	GR B2 A2	04 12 07 11	BNEZ X
04 11 B2 A2	PRINT B2 A2	04 12 07 12	RTSGO
04 12 00 A2	MOV Y,(RA2)	04 12 07 13	SPCMD 07 13
04 12 01 A2	MOV (RA2),Y	04 12 07 14	MOV X,N
04 12 02 A2	MOV X,(RA2)	04 12 07 15	SPCMD 07 15
04 12 03 A2	MOV (RA2),X	04 12 08 A2	SWA RA2
04 12 04 00	SUB #10,E	04 12 09 A2	SWA SA2
04 12 04 01	SUB #01,E	04 12 10 A2	LOADR A2
...	...	04 12 11 A2	SAVER A2
04 12 04 09	SUB #09,E	04 12 12 00	STTAP
04 12 04 10	BPL Y	04 12 12 01	–
04 12 04 11	BEQZ Y	04 12 12 02	LAMP
04 12 04 12	–	04 12 12 03	TRTAP
...	...	04 12 12 04	SAVC
04 12 04 15	–	04 12 12 05	CLDRS
04 12 05 00	–	04 12 12 06	SNCS
...	...	04 12 12 07	INFS
04 12 05 09	–	04 12 12 08	–
04 12 05 10	BMI Y	04 12 12 09	FORW
...	...	04 12 12 10	SMER
04 12 05 11	BNEZ Y	04 12 12 11	ELMG
04 12 05 12	SPCMD 05 12		
04 12 05 13	SPCMD 05 13		

Код	Мнемокод	Код	Мнемокод
04 12 12 12	SAVS	05 03	DIV X,@Y
04 12 12 13	CRFS	05 04	MOV X,@Y
04 12 12 14	–	05 05	MOV @Y,X
04 12 12 15	–	05 06	SWA X,@Y
04 12 13 00	–	05 07	BGE Y,X
...	...	05 08	BLT Y,X
04 12 13 15	–	05 09	BEQ Y,X
04 12 14 00	LNCN	05 10	BPER
04 12 14 01	–	05 11	RTS
04 12 14 02	PAUSER	05 12	END
04 12 14 03	WTRT	05 13	LOADP
04 12 14 04	INPOWC	05 14	GO
04 12 14 05	OUTOWC	05 15	STOP
04 12 14 06	INPOWS	06 00	ADD X,Y
04 12 14 07	OUTOWS	06 01	SUB X,Y
04 12 14 08	SETPER	06 02	MUL X,Y
...	...	06 03	DIV X,Y
04 12 14 15	SETPER	06 04	MOV X,Y
04 12 15 00	–	06 05	MOV Y,X
...	...	06 06	SWA X,Y
04 12 15 15	–	06 07	ABS X
04 13 00 A2	JMP @RA2	06 08	INT
04 13 01 A2	JSR @RA2	06 09	PI
04 13 02 A2	CMD RA2	06 10	LGT
04 13 03 A2	MOV RA2,X	06 11	LOG
04 13 04 A2	MOVD RA2, X	06 12	SQR
04 13 05 A2	MOV RA2,BD	06 13	EXT
04 13 06 A2	MOV RA2,BP	06 14	EXP
04 13 07 A2	MOV SA2,M	06 15	INV
04 13 08 A2	ABS RA2	07 00	DIG 0
04 13 09 A2	NEG RA2	07 09	DIG 9
04 13 10 A2	CLR RA2	...	...
04 13 11 A2	MOV X,RA2	07 10	E
04 13 12 A2	MOVH,RA2	07 11	NEG X
04 13 13 A2	MOV BD,RA2	07 12	POINT
04 13 14 A2	MOV BP,RA2	07 13	QRT
04 13 15 A2	MOV M,SA2	07 14	RES
04 14 B2 A2	MOV Y,C	07 15	CLR X
04 15 B2 A2	MOV C,Y	08 00	RAD
05 00	ADD X,@Y	08 01	DEG
05 01	SUB X,@Y	08 02	SIN
05 02	MUL X,@Y	08 03	COS

Код		Мнемокод
08 04		TAN
08 05		ASN
08 06		ACS
08 07		ATN
08 08		CAP
08 09		POC
08 10		HSN
08 11		HCS
08 12		HTN
08 13		AHS
08 14		AHC
08 15		AHT
09 00	B2 A2	ADD RB2,@RA2
09 01	B2 A2	SUB RB2,@RA2
09 02	B2 A2	MUL RB2,@RA2
09 03	B2 A2	BSAZ @RA2.+B2+1
09 04	B2 A2	MOV RB2,@RA2
09 05	B2 A2	MOV @RA2,RB2
09 06	B2 A2	SWA RB2,@RA2
09 07	B2 A2	BSA RB2,@RA2
09 08	B2 A2	AND SB2,@RA2
09 09	B2 A2	XOR SB2,@RA2
09 10	B2 A2	COM @RA2
09 11	B2 A2	OR SB2,@RA2
09 12	B2 A2	MOV SB2,@RA2
09 13	B2 A2	MOV @RA2,SB2
09 14	B2 A2	SWA SB2,@RA2
09 15	B2 A2	BSA SB2,@RA2
10 00	B2 A2	ADD #B2,RA2
10 01	B2 A2	SUB #B2,RA2
10 02	B2 A2	BHIS SB2,SA2
10 03	B2 A2	ATOI B2,A2
10 04	B2 A2	NSN B2,A2
10 05	B2 A2	NSS B2,A2
10 06	B2 A2	ANS SB2,SA2
10 07	B2 A2	PRIOR SB2,RA2
10 08	B2 A2	BNS #B2 A2,S1
10 09	B2 A2	BNS #B2 A2,S3
10 10	B2 A2	JMTT B2 A2
10 11	B2 A2	JMTF B2 A2
10 12	B2 A2	MOV RB2,-(RA2)
10 13	B2 A2	JSTT B2 A2
10 14	B2 A2	JSTF B2 A2
10 15	B2 A2	MOV (RA2)+,RB2
11 00	B2 A2	ADD RB2,RA2
11 01	B2 A2	SUB RB2,RA2
11 02	B2 A2	MUL RB2,RA2
11 03	B2 A2	BSAZ RA2.+B2+1
11 04	B2 A2	MOV RB2,RA2
11 05	B2 A2	MOV RA2,RB2
11 06	B2 A2	SWA RB2,RA2
11 07	B2 A2	BSA RB2,RA2
11 08	B2 A2	AND SB2,SA2
11 09	B2 A2	XOR SB2,SA2
11 10	B2 A2	COM SA2
11 11	B2 A2	OR SB2,SA2
11 12	B2 A2	MOV SB2,SA2
11 13	B2 A2	MOV SA2,SB2
11 14	B2 A2	SWA SB2,SA2
11 15	B2 A2	BSA SB2,SA2
12 00		REW
12 01		VERX
12 02		LOADX
12 03		SAVEX
12 04		BSA Y,X
12 05		TRAP
12 06		MOV X,RR
12 07		MOV RR,X
12 08		WAIT
12 09		NORM
12 10		RTII
12 11		RTI
12 12		RTSI
12 13		JMP @X
12 14		MOV Y,Z
12 15		MOV Z,Y
13 00	B2 A2	MOV #B2,S00
...		...
13 15	B2 A2	MOV #B2 A2,S15
14 00	B2 A2	INPO B2 A2
14 01	B2 A2	OUTO B2 A2
14 02	B2 A2	BR.-16B2-A2+1
14 03	B2 A2	BR .+16B2+A2+1
14 04	B2 A2	VERR B2 A2
14 05	B2 A2	VEX B2 A2

Код		Мнемокод	Код		Мнемокод
14 06	B2 A2	PRONT #B2 A2	15 03	B2 A2	OUTSV B2 A2
14 07	B2 A2	BEV @RA2.+B2+1	15 04	B2 A2	INPR B2 A2
14 08	B2 A2	SOBZ RA2.+B2+1	15 05	B2 A2	OUTR B2 A2
14 09	B2 A2	ABGE RB2,RA2	15 06	B2 A2	INPRV B2 A2
14 10	B2 A2	BGE RB2,RA2	15 07	B2 A2	OUTRV B2 A2
14 11	B2 A2	BKEY .+16B2+A2+1	15 08	B2 A2	INPAS B2 A2
14 12	B2 A2	MOV RB2,TA2	15 09	B2 A2	OUTAS B2 A2
14 13	B2 A2	MOV TB2,RA2	15 10	B2 A2	INPASV B2 A2
14 14	B2 A2	BMER .+16B2+A2+1	15 11	B2 A2	OUTASV B2 A2
14 15	B2 A2	BBIC B2,@RA2	15 12	B2 A2	INPAR B2 A2
14 15	B2 A2	BBIS B2–8,@RA2	15 13	B2 A2	OUTAR B2 A2
15 00	B2 A2	INPS B2 A2	15 14	B2 A2	INPARV B2 A2
15 01	B2 A2	OUTS B2 A2	15 15	B2 A2	OUTARV B2 A2
15 02	B2 A2	INPSV B2 A2			

3. Встроенные математические функции

Мнемокод	Код B1 A1 B2 A2	Содержание	Допустимые значения аргумента
ABS X	06 07	$X \leftarrow \lvert(X)\rvert$	Весь диапазон чисел
ACS	08 06	$X \leftarrow \arccos(X)$	$-1 \leqslant (X) \leqslant 1$
AHC	08 14	$X \leftarrow \mathrm{arch}(X)$	$1 \leqslant (X) \leqslant 0{,}999999999999 \cdot 10^{49}$
AHS	08 13	$X \leftarrow \mathrm{arsh}(X)$	$-629357 \leqslant (X) \leqslant 0{,}999999999999 \cdot 10^{49}$
AHT	08 15	$X \leftarrow \mathrm{arth}(X)$	$-1 < (X) < 1$
ASN	08 05	$X \leftarrow \arcsin(X)$	$-1 \leqslant (X) \leqslant 1$
ATN	08 07	$X \leftarrow \mathrm{arctg}(X)$	Весь диапазон чисел
CAP	08 08	$\begin{cases} Y \leftarrow \sqrt{(X)^2 + (Y)^2}; \\ X \leftarrow \mathrm{arctg}((Y)/(X)) \end{cases}$	$\lvert(X)\rvert < 0{,}1 \cdot 10^{50}$; $\lvert(Y)\rvert < 0{,}1 \cdot 10^{50}$
COS	08 03	$X \leftarrow \cos(X)$	Весь диапазон чисел
DEG	08 01	$X \leftarrow (X)\, 180/\pi$	$\lvert(X)\rvert \leqslant 0{,}999999999999 \cdot 10^{47}$
EXP	06 14	$X \leftarrow e^{(X)}$	$(X) \leqslant 227{,}955924206$
EXT	06 13	$X \leftarrow 10^{(X)}$	$(X) \leqslant 98{,}9999999999$
QRT	07 13	$X \leftarrow (X)^2$	$\lvert(X)\rvert \leqslant 0{,}999999999999 \cdot 10^{49}$
HCS	08 11	$X \leftarrow \mathrm{ch}(X)$	$\lvert(X)\rvert \leqslant 227{,}955924206$

Мнемокод	B1 A1	B2 A2	Содержание	Допустимые значения аргумента
HSN	08 10		X ← sh(X)	|(X)| ⩽ 227,955924206
HTN	08 12		X ← th(X)	|(X)| ⩽ 227,955924206
INT	06 08		Сброс дробной части (X)	Весь диапазон чисел
INV	06 15		X ← 1/(X)	$\|(X)\| > 0{,}1 \cdot 10^{-98}$
LGT	06 10		X ← lg(X)	(X) > 0
LOG	06 11		X ← ln(X)	(X) > 0
NORM	12 09		Нормализация (X)	В разрядах мантиссы и порядка десятичные цифры, в знаковых разрядах мантиссы и порядка нули или единицы
PI	06 09		X ← π	–
POC	08 09		{ Y = (Y)sin(X); X = (Y) cos(X)	(X) – любое, $\|(Y)\| < 0{,}1 \cdot 10^{99}$
RAD	08 00		X ← (X) π/180	Весь диапазон чисел
SQR	06 12		X ← $\sqrt{(X)}$	(X) ⩾ 0
SIN	08 02		X ← sin(X)	Весь диапазон чисел
SPCMD 05 14	04 12	05 14	X ← 180/π	–
SPCMD 05 15	04 12	05 15	X ← π/180	–
TAN	08 04		X ← tg(X)	Весь диапазон чисел

4. Символы КОИ-7 (набор 2)

Шестнадцатеричный код для Д3-28	Шестнадцатеричный код обычный	Символ	Шестнадцатеричный код для Д3-28	Шестнадцатеричный код обычный	Символ	Шестнадцатеричный код для Д3-28	Шестнадцатеричный код обычный	Символ	Шестнадцатеричный код для Д3-28	Шестнадцатеричный код обычный	Символ
00 00	00	ПУС	02 00	20	Пробел	04 00	40	@	06 00	60	Ю
00 01	01	НЗ	02 01	21	!	04 01	41	A	06 01	61	А
00 02	02	НТ	02 02	22	"	04 02	42	B	06 02	62	Б
00 03	03	КТ	02 03	23	#	04 03	43	C	06 03	63	Ц
00 04	04	КП	02 04	24	¤	04 04	44	D	06 04	64	Д
00 05	05	КТМ	02 05	25	%	04 05	45	E	06 05	65	Е
00 06	06	ДА	02 06	26	&	04 06	46	F	06 06	66	Ф
00 07	07	ЗВ	02 07	27	'	04 07	47	G	06 07	67	Г
00 08	08	ВШ	02 08	28	(	04 08	48	H	06 08	68	Х
00 09	09	ГТ	02 09	29	)	04 09	49	I	06 09	69	И
00 10	0A	ПС	02 10	2A	*	04 10	4A	J	06 10	6A	Й
00 11	0B	ВТ	02 11	2B	+	04 11	4B	K	06 11	6B	К
00 12	0C	ПФ	02 12	2C	,	04 12	4C	L	06 12	6C	Л
00 13	0D	ВК	02 13	2D	–	04 13	4D	M	06 13	6D	М
00 14	0E	ВЫХ	02 14	2E	.	04 14	4E	N	06 14	6E	Н
00 15	0F	ВХ	02 15	2F	/	04 15	4F	O	06 15	6F	О
01 00	10	АР1	03 00	30	0	05 00	50	P	07 00	70	П
01 01	11	(СУ1)	03 01	31	1	05 01	51	Q	07 01	71	Я
01 02	12	(СУ2)	03 02	32	2	05 02	52	R	07 02	72	Р
01 03	13	(СУ3)	03 03	33	3	05 03	53	S	07 03	73	С
01 04	14	СТП	03 04	34	4	05 04	54	T	07 04	74	Т
01 05	15	НЕТ	03 05	35	5	05 05	55	U	07 05	75	У
01 06	16	СИН	03 06	36	6	05 06	56	V	07 06	76	Ж
01 07	17	КБ	03 07	37	7	05 07	57	W	07 07	77	В
01 08	18	АН	03 08	38	8	05 08	58	X	07 08	78	Ь
01 09	19	КН	03 09	39	9	05 09	59	Y	07 09	79	Ы
01 10	1A	ЗМ	03 10	3A	:	05 10	5A	Z	07 10	7A	З
01 11	1B	АР2	03 11	3B	;	05 11	5B	[	07 11	7B	Ш
01 12	1C	РФ	03 12	3C	<	05 12	5C	\	07 12	7C	Э
01 13	1D	РГ	03 13	3D	=	05 13	5D	]	07 13	7D	Щ
01 14	1E	РЗ	03 14	3E	>	05 14	5E	¬	07 14	7E	Ч
01 15	1F	РЭ	03 15	3F	?	05 15	5F	_	07 15	7F	ЗБ

5. Формальное описание языка БЭЙСИК (вариант 3А)

Основные обозначения

При описании операторов языка БЭЙСИК используются следующие обозначения:

1) информация, записанная буквами и заключенная в треугольные скобки, задается программистом. Например, общая форма написания оператора безусловной передачи управления: GOTO <номер строки>, а конкретно можно записать в программе: GOTO 25;

2) задание информации, заключенной в квадратные скобки, не обязательно. Например, в операторе [LET] <переменная> = <выражение> ключевое слово LET может быть опущено;

3) из параметров, заключенных в фигурные скобки и размещенных один под другим, должен быть выбран один. Например, по определению,

$$< \text{переменная} > :: = \left\{ \begin{array}{l} < \text{простая переменная} > \\ < \text{индексированная переменная} > \end{array} \right\}$$

т. е. переменная может быть простой либо индексированной;

4) многоточие означает, что заключенные в фигурные скобки параметры могут использоваться неоднократно;

5) символы "::=" означают "определяется как"; символ "|" – "или".

Синтаксис языка БЭЙСИК

< цифра > :: = 0|1|2|3|4|5|6|7|8|9

< шестнадцатеричная цифра > :: = 00|01|02|...|15

< буква > :: = A|B|C|...|Y|Z

< целое > :: = < цифра > [{< цифра >}] ...

< дробная часть > :: = .< целое >

< показатель степени > :: = E{+/−}< цифра > [< цифра >]

$$< \text{число без знака} > :: = \left\{ \begin{array}{l} \left\{ \begin{array}{l} < \text{целое} > \\ < \text{дробная часть} > \\ < \text{целое} > . < \text{дробная часть} > \end{array} \right\} \left[\left\langle \begin{array}{l} \text{показатель} \\ \text{степени} \end{array} \right\rangle \right] \\ \# \text{PI} \end{array} \right\}$$

$$< \text{переменная} > :: = \left\{ \begin{array}{l} < \text{простая переменная} > \\ < \text{индексированная переменная} > \end{array} \right\}$$

< простая переменная > :: = < буква > [< цифра >]

< индексированная переменная > :: = < имя массива > (< целое > [,< целое >])

< имя массива > :: = < буква > [< цифра >]

< массив > :: = < имя массива > ( )

< номер строки > :: = 1|2|3|...|7999

$$< \text{функция} > :: = \left\{ \begin{array}{l} \text{FN} < \text{буква} > \\ < \text{стандартная функция} > \end{array} \right\} (< \text{выражение} >)$$

< стандартная функция > :: = DEG|RAD|SIN|COS|TAN|ASN|ACS|ATN|SGN|SQR|HSN|
HCS|HTN|AHS|AHC|AHT|ABS|EXP|INT|LGT|LOG|EXT|RND

< цепочка знаков > :: = < любая последовательность знаков, кроме апострофов двоеточия и забоев >

< элемент выражения > :: = < число без знака > | < функция > | < переменная >;

< знак арифметической операции > :: = + | − | * | / | ¬

<выражение> :: = {[±]} <элемент выражения> |

< выражение > < знак арифметической операции > < элемент выражения > |

(< выражение >) | (< выражение >) < знак арифметической операции > < элемент выражения > |

(< выражение >) < знак арифметической операции > (< выражение >)

< индекс > :: = < выражение > (значение целой части выражения от 0 до 255)

< код однобайтной команды > :: = < шестнадцатеричная цифра > < шестнадцатеричная цифра >

< код двухбайтной команды > :: = < код однобайтной команды > < код однобайтной команды >

< код команды > :: = < код однобайтной команды > | < код двухбайтной команды >

Операторы языка БЭЙСИК

Операторы описания массивов:

DIM{< имя массива > (< индекс > [,< индекс >])} [,...]

COM{< имя массива > (< индекс > [,< индекс >])}[,...]

Не допускается запись DIM и COM внутри циклов!

Оператор определения функции пользователя:

DEF <имя функции> (< формальный параметр>) = <выражение>

где < имя функции > :: = FN < буква >; (< формальный параметр >) :: = < простая переменная >

Не допускается запись DEF внутри цикла!

Оператор комментариев:

REM < цепочка знаков >

Оператор REM должен быть последним или единственным в строке.

Оператор присваивания:

[LET] < переменная > = < выражение >

Определение блоков данных:

DATA {< выражение > / < переменная >} [,...]

Оператор DATA должен быть последним или единственным в строке.

Оператор

READ {< переменная >} [,...]

присваивает переменным значения из блока данных.

Оператор

RESTORE

устанавливает указатель блока данных в исходное положение.

Оператор ввода:

INPUT [< 'цепочка знаков' >]{< переменная >}[,...]

При выполнении оператора INPUT печатается цепочка знаков или, если цепочки знаков

нет, – вопросительный знак. Числа при вводе разделяются запятой. Ввод должен быть завершен нажатием клавиши [ВК] или [ПС] .

Оператор цикла:

FOR < переменная >=< выражение >TO< выражение > STEP < выражение >

NEXT < переменная >

(FOR – первое предложение цикла, NEXT – последнее).

Операторы передачи управления:

GOTO< номер строки >
ON< выражение >

(целая часть выражения есть номер строки, которой передается управление)

$$\text{IF}\left\{\begin{array}{l}\text{<выражение><знак отношения><выражение>}\\ \text{END[F]}\end{array}\right\}\left\{\begin{array}{l}\text{THEN}\left\{\begin{array}{l}\text{<оператор>}\\ \text{<номер строки>}\end{array}\right\}\\ \text{GOTO}\quad\text{<номер строки>}\end{array}\right\}[,\ldots]$$

где <знак отношения>::=<|>|=|>=|<=|<>|< ; END означает, что проверяется считывание конца МЛ; ENDF означает проверку считывания конца файла данных. Если проверяемое условие выполняется, то управление передается оператору, записанному после оператора THEN.

Оператор вывода:

$$\text{PRINT}\left\{\left[\left[\begin{array}{l}\text{!F <цифра>.<цифра>!}\\ \text{!<цифра>.<цифра>!}\\ \text{!E!}\\ \text{<выражение>}\\ \text{<'цепочка знаков'>}\\ \text{OPEN}\\ \text{TAB<выражение>}\\ ,\\ ;\end{array}\right]\right]\right\}$$

где !E! – машинный формат; !F<цифра> . <цифра>! – формат с плавающей точкой; !<цифра>.<цифра>! – формат с фиксированной точкой.

Операторы вызова и возврата подпрограммы:

GO SUB<номер строки>
RETURN

Первый оператор осуществляет обращение к подпрограмме, второй – выход из подпрограммы.

Операторы останова:

STOP
END

Операторы записи и считывания текста на МЛ и ПЛ:

$$\text{SAVE}\left\{\begin{array}{l}\text{\#1}\\ \text{[\#0] ['<цепочка знаков>'][<номер строки>,<номер строки>]}\\ \text{END}\end{array}\right\}$$

SAVE END

LOAD {#1 / [#0] ['<цепочка знаков>'][<номер строки>,<номер строки>]}

Знак # 1 означает обмен информацией с ПЛ, а знак # 0 (или его отсутствие) – с МЛ.

Операторы записи и считывания файла данных:

DATA SAVE { {OPEN ['<цепочка знаков>'] / END} / {[#0] / #1} {<выражение> / <массив>} [,...] }

DATA LOAD { {[#0] / #1} [{'<цепочка знаков>' / OPEN}] / {[#0] / #1} {<переменная> / <массив>} [,...] }

Оператор пропуска блоков данных и файлов:

SKIP<выражение> [F]

Оператор перемотки МЛ к началу:

REWIND

Оператор очистки памяти:

CLEAR { [P]<номер строки>[,<номер строки>] / C / D / F / S }

где CLEARC – стирание данных в области COM и функций пользователя; CLEARD – стирание данных в области DIM; CLEARF – установка в начальное состояние указателя циклов FOR-NEXT; CLEARS – установка в начальное состояние указателя стека подпрограмм; CLEARP – стирание части программы, определенной указанными номерами (при отсутствии в последнем операторе буквы P будет выполнена дополнительная перемотка МЛ к началу).

Оператор вывода на печать участка текста программы:

LIST[<номер строки>[,<номер строки>]]

Оператор обращения к внешней подпрограмме:

CALL<номер программы>[<параметры>]

где <номер программы> – целое положительное число от 1 до 7999; <параметры> – требуемые внешними программами параметры.

Оператор использования машинных кодов:

CMD{<код команды>}[[,]...]

Оператор запуска программы:

RUN

6. Операторы языка БЭЙСИК (Вариант 3А)

Оператор или ключевое слово	Слово, от которого происходит название оператора	Перевод	Параграф, в котором описан оператор
CALL	CALL	Вызывать	3.7
CLEAR	CLEAR	Очищать	2.8
CMD	COMMAND	Команда	3.6
COM	COMMON	Общий	2.8
DATA	DATA	Данные	2.11
DATASAVE	DATA SAVE	Данные сохранить	2.18
DATALOAD	DATA LOAD	Данные загрузить	2.18
DEF	DEFINE	Определять	2.9
DIM	DIMENSION	Размерность	2.8
END	END	Конец	2.6
FOR	FOR	Для	2.10
GOSUB	GO SUBROUTINE	Идти к подпрограмме	2.15
GOTO	GO TO	Идти к	2.7
IF	IF	Если	2.7
INPUT	INPUT	Ввести	2.5
LET	LET	Пусть	2.4
LIST	LIST	Распечатать	2.2
LOAD	LOAD	Загрузить	2.18
NEXT	NEXT	Следующий	2.10
ON	ON	На	2.14
OPEN	OPEN	Открыть	2.18
PRINT	PRINT	Печатать	2.5
READ	READ	Читать	2.11
REM	REMARK	Комментарий	2.16
RESTORE	RESTORE	Восстановить	2.11
RETURN	RETURN	Возврат	2.15
REWIND	REWIND	Перемотать	2.18
RUN	RUN	Пуск	2.2
SAVE	SAVE	Хранить	2.18
SKIP	SKIP	Перепрыгнуть	2.18
STEP	STEP	Шаг	2.10
STOP	STOP	Останов	2.6
THEN	THEN	Затем	2.7
TO	TO	К	2.10
TAB	TABULATE	Табулировать	2.13

7. Сообщения об ошибках микроЭВМ "Электроника Д3-28"

Номер сообщения	Причина
0	Переполнение памяти, отведенной пользователю
1	Недопустимый оператор
2	Переполнение строки ввода
3	Недопустимый ограничитель в строке
4	Недопустимый номер строки
5	Несоответствие кавычек в предложении
6	Отсутствие открывающей скобки перед аргументом функции
7	Недопустимый оператор LET
10	Неправильная запись индексов
11	Неправильная размерность индекса
12	Несоответствие скобок в выражении
13	Недопустимый элемент выражения
14	Функция пользователя не определена
15	Неправильное имя переменной
20	Неправильная операция отношения
21	Недопустимый оператор IF
22	Недопустимый оператор COM или DIM
23	Недостаточно места для массива DIM
24	Неправильный оператор DEF
25	Нет данных для оператора READ
26	Недопустимый оператор DATA
27	Неправильный формат команд в CMD
30	Неправильный формат в операторе FOR-NEXT
31	Отсутствие NEXT
32	Не было FOR
33	Переполнение стека FOR-NEXT
34	Нулевой шаг FOR
35	Неверный формат оператора PRINT
36	Неверно задан формат печати
37	Недопустимое выражение в операторе TAB
38	Отсутствие открывающей записи в буфере МЛ
40	Номер строки НС2 меньше номера строки НС1
41	Превышение уровня вложенности подпрограмм
42	Оператор RETURN без оператора GOSUB
43	Нет строки для перехода по операторам GOSUB или GOTO
44	Нет внешней подпрограммы с указанным именем
50	Неправильное предложение с оператором обслуживания МЛ
52	Сбой структуры файла

Номер сообщения	Причина
53	Отсутствие в запоминающем устройстве массива при приеме с МЛ
54	Не считан очередной блок данных с МЛ в ОЗУ
55	Считанный с МЛ блок не помещается в ОЗУ
59	При загрузке или записи программы с МЛ указан только один номер строки
121	Недопустимые знаки при вводе по оператору INPUT
122	Недостаточно данных для оператора INPUT
123	Несуществующая переменная
124	Слишком много данных для оператора INPUT
128	Некорректная операция в процессе вычисления

8. Программа проверки магнитных лент

```
00000  05 15              STOP
00001  14 02  00 02       BR 00000
00003  04 08  00 00       MARK 00 00
00005  13 01  00 01       MOV #00 01,S01
00007  00 11              JSM 00 11
00008  13 03  12 05       MOV #12 05,S03
00010  00 12              JSM 00 12
00011  00 10              JSM 00 10
00012  13 01  00 00       MOV #00 00,S01
00014  13 04  15 15       MOV #15 15,S04
00016  04 12  14 02       PAUSER
00018  00 10              JSM 00 10
00019  04 12  12 00       STTAP
00021  00 11              JSM 00 11
00022  05 11              RTS
00023  04 08  00 02       MARK 00 02
00025  13 06  00 04       MOV #00 04,S06
00027  13 07  00 00       MOV #00 00,S07
00029  04 13  05 11       MOV R11,BD
00031  04 13  10 11       CLR R11
00033  07 15              CLR X
00034  05 15              STOP
00035  04 13  12 12       MOVH X,R12
00037  05 10              BPER 00040
00038  14 03  00 03       BR 00042
00040  14 02  00 08       BR 00033
00042  04 12  06 11       BEQZ X  00046
00044  14 03  00 07       BR 00052
00046  13 08  02 07       MOV #02 07,S08
00048  13 09  00 15       MOV #00 15,S09
00050  14 03  00 09       BR 00060
00052  13 00  07 11       MOV #07 11,S00
00054  13 01  00 01       MOV #00 01,S01
00056  14 10  08 12       BGE R08,R12 00060
00058  14 02  01 10       BR 00033
00060  00 11              JSM 00 11
00061  13 01  05 05       MOV #05 05,S01
00063  09 12  01 11       MOV S01,@R11
00065  14 09  12 11       ABGE R12,R11 00069
00067  14 03  00 03       BR 00071
00069  14 02  00 07       BR 00063
00071  13 01  05 12       MOV #05 12,S01
00073  09 12  01 11       MOV S01,@R11
00075  04 13  13 11       MOV BD,R11
00077  04 13  04 11       MOVD R11,X
00079  06 04              MOV X,Y
00080  12 01              VERX
00081  12 06              MOV X,RR
00082  06 05              MOV Y,X
00083  13 01  00 01       MOV #00 01,S01
00085  00 12              JSM 00 12
00086  13 03  12 05       MOV #12 05,S03
00088  00 10              JSM 00 10
00091  13 01  00 01       MOV #00 01,S01
00093  04 13  10 02       CLR R02
```

```
00095  12 03              SAVEX
00096  00 12              JSM 00 12

00097  04 12   12 05      CLDRS
00099  11 15   00 01      BSA S00,S01 00103
00101  14 03   00 05      BR 00107
00103  10 00   01 02      ADD #01,R02
00105  14 02   00 11      BR 00095
00107  04 12   12 00      STTAP
00109  05 15              STOP
00110  04 08   00 04      MARK 00 04
00112  00 11              JSM 00 11
00113  00 11              JSM 00 11
00114  04 13   10 03      CLR R03
00116  04 13   04 11      MOVD R11,X
00118  12 02              LOADX
00119  14 14   01 03      BMER 00139
00121  05 10              BPER 00124
00122  14 03   00 03      BR 00126
00124  14 03   00 14      BR 00139
00126  12 01              VERX
00127  05 10              BPER 00130
00128  14 03   00 03      BR 00132
00130  14 03   00 08      BR 00139
00132  06 04              MOV X,Y
00133  12 07              MOV RR,X
00134  05 09              BEQ Y,X 00137
00135  14 03   00 03      BR 00139
00137  14 02   01 06      BR 00116
00139  10 00   01 03      ADD #01,R03
00141  05 10              BPER 00144
00142  05 14              GO
00143  14 02   01 12      BR 00116
00145  05 15              STOP
00146  04 08   00 10      MARK 00 10
00148  13 02   04 12      MOV #04 12,S02
00150  04 13   02 09      CMD R09
00152  11 15   00 01      BSA S00,S01 00156
00154  14 02   00 05      BR 00150
00156  05 11              RTS
00157  04 08   00 11      MARK 00 11
00159  13 00   00 05      MOV #00 05,S00
00161  12 00              REW
00162  14 08   03 08      SOBZ R08 00166
00164  14 02   00 04      BR 00161
00166  13 01   00 01      MOV #00 01,S01
00168  13 02   04 12      MOV #04 12,S02
00170  13 03   12 13      MOV #12 13,S03
00172  00 10              JSM 00 10
00173  05 11              RTS
00174  04 08   00 12      MARK 00 12
00176  04 12   12 02      LAMP
00178  04 12   12 11      ELMG
00180  04 12   12 03      TRTAP
00182  04 12   12 04      SAVC
00184  13 04   01 00      MOV #01 00,S04
00186  13 05   00 00      MOV #00 00,S05
00188  04 12   14 02      PAUSER
00190  05 11              RTS
00191  04 08   00 15      MARK 00 15
00193  04 13   04 02      MOVD R02,X
00195  06 04              MOV X,Y
00196  04 13   04 03      MOVD R03,X
00198  05 15              STOP
00199  05 12              END
```

9. Сообщения об ошибках микроЭВМ ДВК-1

Код ошибки	Содержание
0	Переполнение памяти, отведенной пользователю
1	Нераспознаваемый оператор
2	Недопустимый оператор GO или GOSUB
3	Недопустимый знак, ограничивающий оператор
4	Оператор RETURN без соответствующего оператора GOSUB
5	Неправильно сформирован индекс
6	Индекс не попадает в интервал от 0 до 255 или превышает максимум, установленный программой
7	Несоответствие скобок в операторе
8	Недопустимый оператор LET
9	Недопустимый знак операции отношения в операторе IF
10	Недопустимый оператор IF
11	Недопустимый оператор PRINT
12	Слишком длинная вводимая строка
13	Неправильная размерность в операторе DIM
14	В ОЗУ недостаточно места для массива
15	Неправильно сформирован оператор DEF
16	Недопустимый номер строки или значение размерности
17	Оператор DIM для ранее описанного или использованного элемента
18	Неправильная переменная в списке оператора INPUT
19	Неправильная переменная в списке оператора READ
20	Данные в списке оператора READ исчерпаны
21	Неправильный формат оператора DATA
22	Недопустимый оператор FOR
23	Оператор FOR не сопровождается соответствующим оператором NEXT
24	Оператор NEXT без оператора FOR
25	Несоответствие кавычек в операторе
26	Неправильно установлена функция EXP
27	Неправильно сформировано выражение (пропущен порядок числа в формате E)

10. Сообщения об ошибках для микроЭВМ "Искра 226"

Номер ошибки	Характер ошибки	Возможная причина ошибки и действия программиста по ее устранению
	Системный сбой	При случайном системном сбое рекомендуется попытаться устранить причину, изменив последовательность операций, приведших к сбою, путем нажатия клавиши RESET и использования оператора CLEAR , либо выключить и еще раз включить микроЭВМ. Если микроЭВМ не удалось вывести из этого состояния и сбой носит постоянный характер, то необходим ремонт
01	Текстовое переполнение	Использован весь доступный пользователю объем ОЗУ. Сократить программу и (или) разбить ее на сегменты
02	Переполнение таблиц	Отведенное в ОЗУ место для запоминания внутренних операционных таблиц и идентификаторов переменных полностью использовано либо глубина вложений и циклов недопустима. Сократить, отредактировать и (или) разбить программу на сегменты
03	Математическая ошибка: переполнение показателя степени; деление на нуль; отрицательный или нулевой аргумент логарифма; отрицательный аргумент функции; недопустимое возведение в степень; недопустимые аргументы функции SIN, COS или TAN	Отредактировать программу или обрабатываемые данные
04	Неопределенный массив или переменная при вводе строки НС	Упомянутый массив (переменная) не был определен должным образом в операторе DIM или COM, т. е. либо вообще не был определен, либо упоминался и как одномерный, и как двухмерный
05	Резерв	
06	Синтаксическая ошибка при вводе с клавиатуры	При вводе очередной строки допущена синтаксическая ошибка. Исправить текст строки
07	Синтаксическая ошибка при вводе с МЛ/MD	При вводе программы с МЛ (MD) допущена синтаксическая ошибка. Исправить текст строки

Номер ошибки	Характер ошибки	Возможная причина ошибки и действия программиста по ее устранению
08	Функция FN не определена	В программе обнаружена ссылка на неопределенную функцию FN. Определить функцию или сослаться на определенную функцию
09	Резерв	
10	Резерв	
11	Неправильный номер строки	Не определена строка, на которую произведена ссылка, либо пользователь пытался продолжить программу с несуществующего номера строки. Исправить текст программы
12	Неправильный оператор	При выполнении программы встретился ошибочный оператор. Исправить текст оператора
13	Резерв	
14	Резерв	
15	Неправильный номер строки иди диапазон номеров строк	В операторах LIST, SAVE, CLEARP либо задан диапазон несуществующих номеров строк, либо нарушена возрастающая последовательность номеров строк. Исправить текст оператора
16	Резерв	
17	Резерв	
18	Недопустимая величина (числа, размерности массива, индекса)	Величина превышает допустимый предел, например размерность больше, чем 7999, либо индекс массива превышает заданную размерность. Исправить текст оператора
19	Резерв	
20	Недопустимый формат числа, порядок числа больше 99	Формат введенного числа неверен. Исправить текст оператора
21	Резерв	
22	Неопределенный массив (переменная), встретившийся при выполнении программы	При просмотре программы, выполняемой микроЭВМ по оператору RUN, в операторах DIM, COM встретился не определенный должным образом массив (переменная), т.е. либо массив упоминался и как одномерный, и как двухмерный,либо вообще не был определен. Исправить текст оператора
23	Отсутствие программы в ОЗУ	Были использованы операторы LIST, SAVE, RUN, а в ОЗУ не оказалось ни одной строки программы. Ввести текст программы
24	Неупорядоченный список данных	Был использован оператор READ в номере строки, но перед этим не был выполнен программный счет по оператору RUN

Номер ошибки	Характер ошибки	Возможная причина ошибки и действия программиста по ее устранению
25	Недопустимая пара операторов GOSUB-RETURN или FOR-NEXT	Для встретившегося в программе оператора RETURN (NEXT) нет сопряженного оператора GOSUB (FOR) или осуществлен переход внутрь подпрограммы (цикла). Исправить текст программы
26	Резерв	
27	Данные исчерпаны или находятся за допустимыми пределами	При выполнении оператора READ данных, перечисленных в операторе DATA, не хватило или данные, указанные в операторе DATA, находились за допустимыми пределами. Исправить текст программы либо исправить оператор RESTORE
28	Резерв	
29	Недопустимый формат данных в операторе INPUT	При вводе данных по оператору INPUT (целой или действительной переменной) число набрано неверно. Ввести заново данные в правильном формате, начиная с ошибочного числа, либо прервать выполнение программы, нажав клавишу [RESET], и начать выполнение программы снова
30	Резерв	
31	Новый номер строки по оператору RENUMBER 9999	При перенумерации программы появился номер строки, превышающий 9999. Исправить текст оператора RENUMBER
32	Резерв	
33	Резерв	
34	Резерв	
35	Резерв	
36	Недопустимый формат данных	Неправильно, например, задан формат числа в естественной форме в операторе CONVERT. Исправить текст оператора
37	Отсутствует оператор IMAGE	В строке, номер которой указан в операторе PRINTUSING, нет оператора IMAGE. Исправить текст программы
38	Резерв	
39	Резерв	
40	Резерв	
41	Недопустимый аргумент функции STR	Аргумент превышает длину переменной. Исправить текст оператора
42	Резерв	

Номер ошибки	Характер ошибки	Возможная причина ошибки и действия программиста по ее устранению
43	Недопустимое присвоение	Недопустимое присвоение при использовании операторов READ, DATALOAD. Исправить текст оператора
44	Программа защищена	Загруженная программа была защищена, следовательно, не разрешена перезапись ее или выдача листинга. Выполнить оператор CLEAR, чтобы выйти из режима защиты, но при этом необходимо учесть, что все программы будут стерты из ОЗУ
45	Длина оператора больше 253	Длина оператора превышает 253 байта. Исправить текст оператора
46	Новый начальный номер в операторе RENUMBER мал	Новый начальный номер строки в операторе RENUMBER нарушает возрастающую последовательность номеров строки программы. Исправить текст оператора
47	Недопустимое задание адреса (физического адреса устройства, логического номера) внешнего устройства	В операторе ввода/вывода неправильно задан физический адрес устройства, либо задан неопределенный логический номер, либо номер превышает допустимое значение. Исправить параметр устройства
48	Нет подпрограммы DEFFN' с указанным номером	В программе отсутствует подпрограмма с указанным номером. Исправить текст программы
49	Неквадратная матрица	Размерности операндов при операции обращения матрицы или равенства единичной матрицы не равны. Исправить размерность массива
50	Матричные операторы несовместимы	Несовместимые размерности операндов в матричных операторах и операторах сортировки. Исправить размерность массива
51	Недопустимый матричный операнд	При матричном умножении и транспонировании использованы одинаковые идентификаторы по обеим сторонам от знака равенства. Исправить текст оператора
52	Сингулярная матрица	Операнд матричного обращения сингулярный и не может быть обращен. Исправить текст оператора
53	Длина распаковываемого массива мала	Длина символьного массива не достаточна для того, чтобы вместить считываемый с МД или МЛ упаковываемый блок информации. Исправить текст программы

Номер ошибки	Характер ошибки	Возможная причина ошибки и действия программиста по ее устранению
54	Недопустимая длина строки в операторе SELECT	В операторе SELECT было задано недопустимое значение параметра < длина строки > . Исправить текст оператора
55	Недопустимый адрес сектора	В операторах COPY, VERIFY нарушена возрастающая последовательность задаваемых номеров секторов или в операторе SCRATCHDISK адрес последнего сектора зоны каталога меньше (или равен) адресу последнего сектора УК. Исправить текст оператора
56	Число превышает формат в операторах PACK, CONVERT	В операторе PACK (CONVERT) данные не соответствуют формату. Исправить текст оператора
57	Недопустимый адрес сектора или недопустимая область построения графика	Адрес сектора в дисковых операторах был задан числом отрицательным или большим, чем 64 000, либо произошел выход координат графики за допустимую область построения. Исправить текст оператора
58	Необобщенная переменная	Переменная в операторе LOAD DA, принимающая значение адреса очередного доступного сектора после загрузки, не является обобщенной. Определить переменную как обобщающую
59	Недопустимая символьная переменная	В операторах ON ERROR, INPUT (для таймера) и в дисковых операторах с абсолютной адресацией длина недостаточна для хранения информации. Изменить длину переменной
60	Файл не открыт или не существует	При обращении к МД был задан номер не открытого или не существующего файла. Изменить номер файла
61	Резерв	
62	Резерв	
63	Резерв	
64	Резерв	
65	Резерв	
66	Резерв	
67	Неправильный тип записи программа/данные	По оператору LOAD программа не была считана или по оператору DATA LOAD не были считаны данные. Выяснить причину, при необходимости исправить текст оператора.

Номер ошибки	Характер ошибки	Возможная причина ошибки и действия программиста по ее устранению
68	Буфер ввода/вывода мал	Программная строка не помещается в буфер устройства по оператору SAVE. Изменить длину строки устройства по оператору SELECT
69	Переменная (массив) не помещается в буфер ввода/вывода	Значение вводимой переменной или массива по оператору DATA SAVE не помещается в буфер устройства. Изменить длину строки устройства по оператору SELECT или изменить размерность переменной (массива)
70	В операторе PRINTUSING нет формата	В операторе PRINTUSING отсутствует формат. Изменить текст оператора
71	Файл уже есть в каталоге	Была попытка записать в каталог файл с именем, уже содержащимся в каталоге. Изменить имя файла
72	Файл вычеркнут	Была попытка обращения к файлу, отмеченному оператором SCRATCH
73	Файл отсутствует в каталоге	Была попытка адресации к файлу с несуществующим именем, либо обращение к файлу данных как к программному файлу, либо попытка открыть программный файл как файл данных. Убедиться в правильном использовании имени файла, а также в том, что вставлен новый МД
74	Адрес сектора вне файла	Был задан неправильный адрес сектора. Исправить текст оператора
75	Конец каталога	Конец каталога попал внутрь области УК, или была сделана попытка сдвинуть конец зоны каталога в пределы, уже занятые другими файлами (по оператору MOVE END), или же в ОК не осталось места для запоминания информации. Исправить текст программы
76	Файл заполнен	Внутри упомянутого каталогизированного файла адресуемого сектора на МД нет. Исправить текст программы
77	Временный файл в пределах каталога	Была предпринята попытка поместить временный файл в ОК. Исправить текст программы
78	УК заполнен	В УК нет места для имени нового файла. Вычеркнуть любой ненужный файл или установить новый МД
79	Резерв	

Номер ошибки	Характер ошибки	Возможная причина ошибки и действия программиста по ее устранению
80	Ошибка ввода/вывода: авария УВВ (дисклента, ...) (01); нарушение последовательности процедур (02); УВВ не готово к обмену (03); резерв (04); защита от записи (05); маркер начала (конца) носителя (06); ошибка по контрольной сумме контрольного считывания после записи (07); авария блоков интерфейсных функциональных (08); резерв (09); ошибка по контрольной сумме при считывании (0A); ошибка процедуры (кодирование формы контрольной суммы обмена процессор интерпретирующий диалоговый – блоки интерфейсные функциональные) (0B); физический блок с указанным адресом не найден (0C); нарушение последовательности интерфейсных команд в процедуре (0D); недопустимый адрес физического блока (0E); ошибочная длина массива данных (0F)	

11. Тексты прикладных БЕЙСИК-программ с результатами счета на тестовых задачах

Существует много разнообразных численных методов решения инженерных и научных задач. В данное пособие включены следующие избранные численные методы и программы: решение системы линейных уравнений методом Гаусса; нахождение собственных значений и собственных векторов методом исчерпывания; обращение матриц; вычисление однократных интегралов методом Симпсона и методом Гаусса с 8 узлами; вычисление двухкратных интегралов методом Симпсона; численное решение обыкновенного дифференциального уравнения методами Рунге–Кутта 4-го порядка и прогноза-коррекции; нахождение безусловного экстремума функции двух переменных модифицированным методом Ньютона–Рафсона; сортировка чисел; линейная интерполяция и интерполяция сплайнами.

Алгоритм решения по каждой программе не приводится, но даны соответствующие ссылки на литературу.

1. *Решение системы линейных уравнений методом Гаусса.* Алгоритм решения приведен, например, в пособиях [1, 2, 4, 9–11, 14, 16–18, 21, 23, 28, 31, 34, 35]. В целях уменьшения ошибки вычислений обычно рекомендуется [23] делать полный или частичный выбор главного элемента. Однако такой выбор увеличивает время счета по программе, что весьма значительно для Д3-28. Небольшой размер системы линейных уравнений (до 15) и высокая точность представления чисел (12 десятичных знаков) в значительной мере компенсируют отказ от выбора главного элемента.

Тестовая задача. Решить систему линейных уравнений:

$$\begin{cases} x_1 + 149999999999\,x_2 = 150000000000; \\ 150000000000\,x_1 + x_2 = 150000000001. \end{cases}$$

Возможный вариант программы:

```
  3 PRINT'*****************************************'
  4 PRINT'**  РЕШЕНИЕ СИСТЕМЫ ЛИНЕЙНЫХ УРАВНЕНИЙ   **'
  5 PRINT'*****************************************'
  6 INPUT'РАЗМЕР МАТРИЦЫ?  'N
 10 DIM A(N,N+2)
 30 PRINT'ВВЕДИТЕ КОЭФФИЦИЕНТЫ МАТРИЦЫ ЛЕВОЙ ЧАСТИ'
 40 PRINT'   (ПО ЭЛЕМЕНТАМ СТОЛБЦОВ) '
100 FOR J=1 TO N
119 FOR I=1 TO N
120 INPUT A(I,J)
130 NEXT I
135 NEXT J
140 PRINT'ВВЕДИТЕ ВЕКТОР ПРАВОЙ ЧАСТИ
142 FOR I=1 TO N
144 INPUT A(I,N+1)
146 NEXT I
150 PRINT:PRINT
210 FORS=1 TO N
220 FOR T=S TO N
230 IF A(T,S)<>0 THEN 240
235 NEXT T
237 PRINT'НЕТ РЕШЕНИЯ
238 GOTO 830
```

```
240 GOSUB 400
250 LETC=1/A(S,S)
260 GOSUB 500
270 FOR T=1 TO N
275 IF T=S THEN 300
280 LETC=-A(T,S)
290 GOSUB 600
300 NEXT T
305 NEXT S
310 GOTO 700
400 FOR J=1 TO N+1
410 FOR J=1 TO N+1
420 LETB=A(S,J)
430 LETA(S,J)=A(T,J)
440 LETA(T,J)=B
450 NEXT J
460 RETURN
500 FOR J=1 TO N+1
520 LETA(S,J)=C*A(S,J)
530 NEXT J
540 RETURN
600 FOR J=1 TO N+1
610 LETA(T,J)=A(T,J)+C*A(S,J)
620 NEXT J
630 RETURN
700 FOR T=1 TO N
710 PRINT'X('!2.0!T;')=';!F1.6!A(T,N+1)
720 NEXT T
730 END
```

Вычисления по программе привели к следующим результатам:

X (1) = 1.000000000

X (2) = 1.000000000

Программа выполнялась 7 с.

2. *Нахождение собственных значений и собственных векторов методом исчерпывания.* Алгоритм решения дан, например, в работах [9, 31]. Исследование полной проблемы собственных значений и собственных векторов приведено, например, в пособиях [1, 2].

Тестовая задача. Вычислить собственные значения матрицы

$$A=10^6\begin{bmatrix}10 & 5 & 6\\ 5 & 20 & 4\\ 6 & 4 & 30\end{bmatrix}.$$

Возможный вариант программы:

```
6499 PRINT'*************************************************'
6500 PRINT'*****        РАСЧЕТ СОБСТВЕННЫХ ЗНАЧЕНИЙ        '
6501 PRINT' И СОБСТВЕННЫХ  ВЕКТОРОВ МЕТОДОМ ИСЧЕРПЫВАНИЯ **'
6502 PRINT'*************************************************'
6503 INPUT'РАЗМЕР МАТРИЦЫ?  'M
6510 DIM H1(M,M),X4(M),R(M)
6511 DIM H2(M,M)
6512 PRINT'ВВЕДИТЕ ИСХОДНУЮ МАТРИЦУ ПО СТОЛБЦАМ'
6513 FOR I=1 TO M:FOR J=1 TO M:INPUT H1(J,I):NEXT J:NEXT I
6515 PRINT'ВВЕДЕНА МАТРИЦА'
6517 FOR I=1 TO M:FOR J=1 TO M:PRINT!E!H1(I,J):NEXT J:NEXT I
```

```
6519 GOTO 6541
6541 LETK2=0
6542 LETX4(1)=1
6545 FOR J7=2 TO M:LETX4(J7)=0:NEXT J7
6546 LETX5=0
6600 FOR I=1 TO 50
6610 FOR I4=1 TO M
6620 LETR(I4)=0
6630 FOR J4=1 TO M
6631 LETR(I4)=R(I4)+H1(I4,J4)*X4(J4)
6632 NEXT J4
6640 NEXT I4
6650 FOR J=1 TO M:LETX4(J)=R(J):NEXT J
6660 REMPRINT'ПОИСК НАИМЕНЬШЕГО ЭЛЕМЕНТА'
6670 LETX6=X4(1)
6671 FOR J7=2 TO M
6672 IF X4(J7)>X6 THEN LETX6=X4(J7)
6673 NEXT J7
6674 GOTO 6700
6680 IF X4(2)>X6 THEN LETX6=X4(2)
6690 IF X4(3)>X6 THEN LETX6=X4(3)
6700 REM PRINT'НОРМАЛИЗАЦИЯ'
6701 FOR J7=1 TO M:LETX4(J7)=X4(J7)/X6:NEXT J7
6702 GOTO 6713
6713 LETX7=X6
6730 IF ABS((X5-X7)/X7)<=0.0001 THEN 6760
6740 LETX5=X7
6750 NEXT I
6760 PRINT'---------------------------------------------'
6761 LETO=0
6762 FOR J7=1 TO M:LETO=O+X4(J7)↑2:NEXT J7
6763 LETO=SQR(O)
6767 FOR J7=1 TO M:LETX4(J7)=X4(J7)/O:NEXT J7
6769 PRINT'ИТЕРАЦИЙ='I3.0II'     СОБСТВЕННОЕ ЗНАЧЕНИЕ='IF1.7IX7
6770 PRINT'                    СОБСТВЕННЫЙ ВЕКТОР '
6772 FOR J7=1 TO M:PRINT'                    'X4(J7):NEXT J7
6790 FOR I2=1 TO M
7000 FOR J2=1 TO M
7010 LETH2(I2,J2)=H1(I2,J2)
7020 NEXT J2
7030 NEXT I2
7040 FOR I2=1 TO M
7050 FOR J2=1 TO M
7060 LETH1(I2,J2)=H2(I2,J2)-X7*X4(I2)*X4(J2)
7070 NEXT J2
7080 NEXT I2
7090 LETK2=K2+1:IF K2<M THEN GOTO 6550
7999 STOP
```

Вычисления по программе привели к следующим результатам:

СОБСТВЕННОЕ ЗНАЧЕНИЕ=3.3712189E 07 ИТЕРАЦИЙ 14

СОБСТВЕННОЕ ЗНАЧЕНИЕ=1.9149068E 07 ИТЕРАЦИЙ 4

СОБСТВЕННОЕ ЗНАЧЕНИЕ=7.1417603E 06 ИТЕРАЦИЙ 4

3. *Обращение матриц.* Алгоритм обращения матриц описан в работах [1, 2, 9, 10, 14, 18]. (Можно применять программу решения системы линейных уравнений методом Гаусса.)

Тестовая задача. Для матрицы

$$A=\begin{bmatrix} 9 & 0 & 0 & 6 & 6 & 0 \\ 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 & 0 \\ 6 & 0 & 0 & 6 & 4 & 0 \\ 6 & 0 & 0 & 4 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 \end{bmatrix}$$

найти обратную матрицу A^{-1}.

Возможный вариант программы:

```
  2 PRINT'****************************'
  3 PRINT'**** ОБРАЩЕНИЕ МАТРИЦЫ  ****'
  4 PRINT'****************************'
 10 INPUT'РАЗМЕР МАТРИЦЫ?( НЕ ВОЛЕЕ 14)  'N
 12 DIM A(N,N+1),V(N)
 15 PRINT'МАТРИЦУ ВВОДИТЬ ПО СТОЛБЦАМ '
 20 FOR I=1 TO N
 25 FOR S=1 TO N
 26 INPUT A(S,I)
 27 NEXT S
 30 NEXT I
 31 PRINT'==========================='
 32 PRINT'ВВОД МАТРИЦЫ ЗАВЕРШЕН      '
 33 PRINT'==========================='
 34 PRINT:PRINT
 35 FOR S=1 TO N
 40 FOR T=S TO N:IF A(T,S)<>0 THEN 50:NEXT T
 44 PRINT:PRINT
 45 PRINT'МАТРИЦА СИНГУЛЯРНАЯ'
 46 GOTO 520
 47 GOSUB 200
 50 GOSUB 200
 55 LETA(S,S)=1/A(S,S):GOSUB 300
 60 FOR T=1 TO N:IF T=S THEN 70
 65 LETB=-A(T,S):LETA(T,S)=0:GOSUB 400
 70 NEXT T:NEXT S
100 FOR S=N TO 1 STEP -1:IF V(S)=S THEN 140
110 FOR J=1 TO N
120 LETB=A(J,S):LETA(J,S)=A(J,V(S)):LETA(J,V(S))=B
130 NEXT J
140 NEXT S:GOTO 500
200 FOR J=1 TO N
210 LETB=A(S,J):LETA(S,J)=A(T,J):LETA(T,J)=B
220 NEXT J:LETV(S)=T:RETURN
300 FOR J=1 TO N:IF J=S THEN 320
310 LETA(S,J)=A(S,S)*A(S,J)
320 NEXT J:RETURN
400 FOR J=1 TO N
410 LETA(T,J)=A(T,J)+B*A(S,J)
420 NEXT J:RETURN
500 FOR I=1 TO N:PRINT'СТРОКА НОМЕР '!2.0!I
505 PRINT!E!
507 FOR J=1 TO N:PRINT A(I,J),
510 NEXT J:PRINT:NEXT I
520 END
```

Вычисления по программе привели к следующим результатам:

$$A^{-1} = \begin{bmatrix} 0.555556 & 0 & 0 & -0.333333 & -0.333333 & 0 \\ 0 & 0.166667 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.166667 & 0 & 0 & 0 \\ -0.333333 & 0 & 0 & 0.5 & 0 & 0 \\ -0.333333 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.25 \end{bmatrix}$$

4. *Вычисление однократных интегралов методом Симпсона.* Алгоритм решения приведен, например, в работах [1, 3, 14, 23].

Тестовая задача. Вычислить интеграл $\int_{-2}^{+2} e^{-x^2/2} dx$ по методу Симпсона с точностью $\epsilon = 0{,}0001$.

Возможный вариант программы:

```
  4 PRINT'*********************************************'
  5 PRINT'****   ВЫЧИСЛЕНИЕ ИНТЕГРАЛА МЕТОДОМ СИМПСОНА ***'
  6 PRINT'*********************************************'
 10 DEF FNF(X)=EXP(-X*X/2)
 20 INPUT'ВВЕДИТЕ НИЖНИЙ ПРЕДЕЛ ИНТЕГРИРОВАНИЯ 'A
 30 INPUT'ВВЕДИТЕ ВЕРХНИЙ ПРЕДЕЛ ИНТЕГРИРОВАНИЯ 'B
 40 INPUT'ВВЕДИТЕ МАКСИМАЛЬНОЕ ЧИСЛО РАЗВИЕНИЙ 'N9
 50 INPUT'ВВЕДИТЕ ОТНОСИТЕЛЬНУЮ ТОЧНОСТЬ 'E
 51 LETN=4:GOSUB 54
 52 PRINT'ИНТЕГРАЛ='!F1.4!S;' РАЗВИЕНИЙ '16.0!N
 53 STOP
 54 LETS1=1
 55 LETX=A:LETC=1:LETI=1:LETS=0:LETH=(B-A)/N
 60 LETX=X+H:LETS=S+(J+C)*FNF(X):LETC=-C:LETI=I+1
 70 IF I<=N-1 THEN 60
 80 LETS=H/J*(FNF(A)+FNF(B)+S)
 81 PRINT'ОЦЕНКА ИНТЕГРАЛА='!F1.4!S;'РАЗВИЕНИЙ '16.0!N
 82 IF ABS((S-S1)/S)<E THEN RETURN
 85 LETS1=S:LETN=N*2
100 IF N>N9 THEN RETURN
110 GOTO 55
120 STOP
130 END
```

После выполнения программы получены следующие результаты:
ИНТЕГРАЛ=2.3926 ТОЧНОСТЬ=1.0000E–04 РАЗБИЕНИЙ 16

5. *Вычисление однократных интегралов методом Гаусса с 8 узлами.* Алгоритм решения описан, например, в пособиях [1, 3, 14, 18, 22, 23, 34, 35, 37].

Тестовая задача. Вычислить интеграл $\int_0^1 \cos(x + x^5)\, dx$ по методу Гаусса с 8 узлами.

Возможный вариант программы:

```
 10 PRINT '*************************************************'
 20 PRINT '* ВЫЧИСЛЕНИЕ ИНТЕГРАЛА МЕТОДОМ ГАУССА С 8 УЗЛАМИ *'
 30 PRINT '*************************************************'
 40 DIM B(8),T(8)
 50 INPUT' ВВЕДИТЕ НИЖНИЙ ПРЕДЕЛ ИНТЕГРИРОВАНИЯ    'A
 60 INPUT' ВВЕДИТЕ ВЕРХНИЙ ПРЕДЕЛ ИНТЕГРИРОВАНИЯ   'C
 70 DEF FNF(Z)=COS(Z+Z↑5)
 80 FOR K=1 TO 8:READ B(K):NEXTK
 90 FOR K=1 TO 8:READ T(K):NEXTK
100 LETA1=(A+C)*0.5:LETA2=(C-A)*0.5
110 FOR K=1 TO 8:LETX1=A1+A2*T(K)
120 LETG=G+B(K)*FNF(X1):NEXTK
130 LETS=G*A2
140 PRINT ' ИНТЕГРАЛ РАВЕН '!F1.5!S
150 STOP
160 DATA .1012285363,.2223810345,.3137066459
170 DATA .3626837834,.3626837834,.3137066459
180 DATA .2223810345,.1012285363
190 DATA-.9602898565,-.7966664774,-.5255324099
200 DATA-.18344346425,.1834436425,.5255324099
210 DATA .7966664774,.9602898565
220 END
```

Вычисления по программе привели к следующим результатам:
ИНТЕГРАЛ=6.98087E–01

6. *Вычисление двухкратных интегралов методом Симпсона.* Алгоритм решения дан, например, в работах [1, 14].

Тестовая задача. Вычислить интеграл $\int_0^1\int_0^1 e^{x+y}dx\,dy$ по методу Симпсона с точностью $\varepsilon = 0{,}01$.

Возможный вариант программы:

```
  2 PRINT'*********************************************'
  3 PRINT'** ВЫЧИСЛЕНИЕ 2-Х КР.ИНТЕГРАЛА  ПО СИМПСОНУ *'
  5 PRINT'*********************************************'
  6 DIM Z(128)
 20 FOR I=0 TO 128:LETZ(I)=0:NEXT I
 21 INPUT'ВВЕДИТЕ НИЖНИЙ ПРЕДЕЛ ПО ПЕРЕМЕННОЙ  X 'X1
 22 INPUT'ВВЕДИТЕ ВЕРХНИЙ ПРЕДЕЛ ПО ПЕРЕМЕННОЙ X 'X2
 23 INPUT'ВВЕДИТЕ НИЖНИЙЙ ПРЕДЕЛ ПО ПЕРЕМЕННОЙ Y 'Y1
 24 INPUT'ВВЕДИТЕ ВЕРХНИЙ ПРЕДЕЛ ПО ПЕРЕМЕННОЙ Y 'Y2
 25 INPUT'ВВЕДИТЕ ОТН.ТОЧНОСТЬ ВЫЧИСЛЕНИЙ        'E
 30 LETN=2:LETZ(0)=1:LETZ(1)=4:LETZ(2)=1
 50 LETS=0:LETH1=(X2-X1)/N:LETH2=(Y2-Y1)/N:LETI=-1
 60 FOR X=X1 TO X2 STEP H1
 70 LETI=I+1:LETJ=-1
 80 FOR Y=Y1 TO Y2 STEP H2
 90 LETJ=J+1
100 LETF=5*X↑2*Y-2*Y↑3
110 LETS1=Z(I)*Z(J)*F
120 LETS=S+S1
```

```
140 NEXT Y
150 NEXT X
155 LETK=J+1+N*I+I
160 LETU1=H1*H2*S/9
165 PRINT'ОЦЕНКА ИНТЕГРАЛА='!F1.6!U1
170 IF N=2 THEN 190
180 IF ABS((U2-U1)/U1)<E THEN 250
190 FOR I=N TO 0 STEP -1
195 LETZ(N+I)=Z(N+I)+Z(I):NEXT I
200 LETN=N+N:LETU2=U1:GOTO 50
250 PRINT'ИНТЕГРАЛ='!F1.4!U1;' ТОЧНОСТЬ='!1.6!E
260 STOP
270 END
```

После выполнения программы получены следующие результаты:
ИНТЕГРАЛ=2.95 ТОЧНОСТЬ=0.010000

7. *Численное решение обыкновенного дифференциального уравнения методом Рунге–Кутта 4-го порядка.* Алгоритм решения приведен, например, в работах [21, 23, 35].

Тестовая задача. Дано дифференциальное уравнение $y'_x = -0{,}01y$. Известно, что $y(0)=100$. Вычислить $y(100)$ при $h=50$.

Возможный вариант программы:

```
 1 PRINT'*********************************************'
 2 PRINT'****  РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФ.УРАВНЕНИЙ  ****'
 3 PRINT'*********************************************'
 4 PRINT'ВВЕДИТЕ КОД МЕТОДА  1 - МЕТОД ЭЙЛЕРА(МОДИФИЦИРОВАННЫЙ)'
 5 PRINT'                    2 - МЕТОД РУНГЕ-КУТТА 4-ГО ПОРЯДКА'
 6 PRINT'                    3 - МЕТОД ПРОГНОЗА-КОРРЕКЦИИ'
 7 INPUT P9
 8 IF P9=1 THEN 14
 9 IF P9=2 THEN 110
10 IF P9=3 THEN 410
14 PRINT'======  МЕТОД ЭЙЛЕРА(МОДИФИЦИРОВАННЫЙ) ======'
15 PRINT'======  ==================================':PRINT
16 PRINT 'ЗАПИШИТЕ В 20   СТРОКУ ПРАВУЮ ЧАСТЬ ДИФ.УРАВНЕНИЯ
17 PRINT 'НАПРИМЕР     20DEF FNF(X1)=X1+SIN(Y1/2.25)'
18 PRINT 'СЧЕТ НАЧИНАЕМ ПО КОМАНДЕ   GOTO 20'
19 STOP
20 DEF FNF(X1)=X1+SIN(Y1/2.25)
31 PRINT'ВВЕДИТЕ НАЧ.ЗНАЧЕНИЕ X'
32 INPUT X
33 PRINT 'ВВЕДИТЕ НАЧ.ЗНАЧЕНИЕ Y'
34 INPUT Y
35 PRINT 'ВВЕДИТЕ  ШАГ НАРАЩЕНИЯ АРГУМЕНТА X'
36 INPUT H
```

```
 37 PRINT'ВВЕДИТЕ КОНЕЧНОЕ ЗНАЧЕНИЕ АРГУМЕНТА X'
 38 INPUT X9
 39 PRINT'НАЧ.ЗНАЧЕНИЕ X '!F1.6!X
 40 PRINT'НАЧ.ЗНАЧЕНИЕ Y 'Y
 41 PRINT'ШАГ НАРАЩЕНИЯ АРГУМЕНТА X'H
 42 PRINT'КОНЕЧНОЕ ЗНАЧЕНИЕ АРГУМЕНТА 'X9
 43 LETI=0
 44 PRINT'==========================================':PRINT:PRINT
 47 LETX1=X:LETY1=Y:LETP=FNF(X1)
 49 LETX1=X+H*0.5:LETY1=Y+P*H*0.5
 51 LETY=Y+H*FNF(X1)
 60 LETX=X+H
 65 LETI=I+1
 70 PRINT ' ИТЕРАЦИЙ='!4.0!I'      АРГУМЕНТ X='!F1.4!X'  ФУНКЦИЯ Y='Y
 80 IF X<X9 THEN GOTO 47
 90 PRINT'======================================================'
 91 PRINT:PRINT
 92 GOTO 820
 93 STOP
 95 END
110 PRINT 'МЕТОД РУНГЕ-КУТТА 4-ГО ПОРЯДКА'
112 PRINT'==============================':PRINT
114 PRINT 'ЗАПИШИТЕ В 120   СТРОКУ ПРАВУЮ ЧАСТЬ ДИФ.УРАВНЕНИЯ'
116 PRINT 'НАПРИМЕР    120DEF FNA(X1)=X1+SIN(Y1/2.25)'
117 PRINT 'СЧЕТ НАЧИНАЕМ ПО КОМАНДЕ   GOTO 119'
118 STOP
120 DEF FNA(X1)=X1+SIN(Y1/2.25)
130 PRINT'ВВЕДИТЕ НАЧ.ЗНАЧЕНИЕ АРГУМЕНТА'
140 INPUT X
150 PRINT'ВВЕДИТЕ НАЧ.ЗНАЧЕНИЕ ФУНКЦИИ'
160 INPUT Y
170 PRINT'ВВЕДИТЕ ШАГ НАРАЩЕНИЯ АРГУМЕНТА'
180 INPUT H
190 PRINT'ВВЕДИТЕ КОНЕЧНОЕ ЗНАЧЕНИЕ АРГУМЕНТА'
200 INPUT X9
201 PRINT'================================='
210 PRINT'НАЧ.ЗНАЧЕНИЕ АРГУМЕНТА'!F1.6!X
220 PRINT'НАЧ.ЗНАЧЕНИЕ ФУНКЦИИ'Y
230 PRINT 'ШАГ НАРАЩЕНИЯ АРГУМЕНТА'H
240 PRINT'КОНЕЧНОЕ ЗНАЧЕНИЕ АРГУМЕНТА'X9
250 PRINT'==============================='
260 LETH2=H*0.5:LETI=0
270 PRINT'ИТЕРАЦИЯ='!4.0!I'   АРГУМЕНТ='!F1.4!X' ФУНКЦИЯ='Y
280 LETX1=X:LETY1=Y
290 LETA1=FNA(X1)
300 LETX1=X+H2:LETY1=Y+H2*A1
310 LETA2=FNA(X1)
320 LETX1=X+H2:LETY1=Y+H2*A2
330 LETA3=FNA(X1)
340 LETX1=X+H:LETY1=Y+H*A3
350 LETA4=FNA(X1)
360 LETY=Y+H/6*(A1+(A2+A3)*2+A4)
370 LETX=X+H
375 LETI=I+1
380 PRINT'ИТЕРАЦИЯ='!4.0!I'   АРГУМЕНТ='!F1.4!X' ФУНКЦИЯ='Y
390 IF X<X9 GOTO 280
400 GOTO 820
405 STOP
410 PRINT '******  МЕТОД ПРОГНОЗА-КОРРЕКЦИИ ******'
411 PRINT'================================':PRINT
413 PRINT 'ЗАПИШИТЕ В 420   СТРОКУ ПРАВУЮ ЧАСТЬ ДИФ.УРАВНЕНИЯ
415 PRINT 'НАПРИМЕР    420DEF FNE(X5)=X5+SIN(Y5/2.25)'
417 PRINT 'СЧЕТ НАЧИНАЕМ ПО КОМАНДЕ   GOTO 420'
419 STOP
```

```
420 DEF FNE(X5)=X5+SIN(Y5/2.25)
430 PRINT'ВВЕДИТЕ НАЧ.ЗНАЧЕНИЕ X'
440 INPUT X
450 PRINT 'ВВЕДИТЕ НАЧ.ЗНАЧЕНИЕ Y'
460 INPUT Y
470 PRINT 'ВВЕДИТЕ  ШАГ НАРАЩЕНИЯ АРГУМЕНТА X'
480 INPUT H
490 PRINT'ВВЕДИТЕ КОНЕЧНОЕ ЗНАЧЕНИЕ АРГУМЕНТА X'
500 INPUT X9
501 PRINT'ВВЕДИТЕ ТОЧНОСТЬ ВЫЧИСЛЕНИЙ'
502 INPUT E
505 PRINT'==============================='
510 PRINT'НАЧ.ЗНАЧЕНИЕ АРГУМЕНТА'!F1.6!X
520 PRINT'НАЧ.ЗНАЧЕНИЕ ФУНКЦИИ'Y
530 PRINT'ШАГ НАРАЩЕНИЯ АРГУМЕНТА'H
540 PRINT'КОНЕЧНОЕ ЗНАЧЕНИЕ АРГУМЕНТА'X9
542 PRINT'ТОЧНОСТЬ ВЫЧИСЛЕНИЙ'E
550 PRINT'==============================='
560 LETX0=X:LETY0=Y:LETH2=H*0.5
565 LETH2=H*.5
570 LETX5=X0:LETY5=Y0:LETA1=FNE(X5)
580 LETX5=X0+H2:LETY5=Y0+H2*A1
590 LETA2=FNE(X5)
600 LETX5=X0+H2:LETY5=Y0+H2*A2
610 LETA3=FNE(X5)
620 LETX5=X0+H:LETY5=Y0+H*A3
630 LETA4=FNE(X5)
640 LETX1=X0+H:LETY1=Y0+H/6*(A1+(2*(A2+A3)+A4))
650 PRINT'АРГУМЕНТ='!F1.4!X1'  ФУНКЦИЯ='Y1 '   ШАГ    '!2.3!H
660 LETX2=X1+H
670 LETX5=X1:LETY5=Y1
680 LETY2=Y0+2*H*FNE(X5)
690 LETI=1:LETY6=Y1
700 LETX5=X1:LETY5=Y1:LETT1=FNE(X5)
710 LETX5=X2:LETY5=Y2:LETT2=FNE(X5)
720 LETY3=Y1+H2*(T1+T2)
730 IF ABS(Y2-Y3)<E GOTO 800
740 IFI<3 GOTO 790
750 PRINT'ЗА 3  ПРИБЛИЖЕНИЙ СХОДИМОСТИ НЕТ.ШАГ ДЕЛИМ НА 2 '
751 LETH=H*0.5:LETX0=X1:LETY0=Y8:GOTO 565
790 LETY2=Y3:LETI=I+1:GOTO 700
800 PRINT'АРГУМЕНТ='!F1.4!X2' ФУНКЦИЯ='Y2' ЧИСЛО ДЕЛЕНИЙ'!6.0!I
805 LETX0=X1:LETY0=Y1:LETX1=X2:LETY1=Y2:LETH=H*1
810 IFX2<X9 GOTO 660
820 PRINT'РАСЧЕТ ОКОНЧЕН.':PRINT
830 INPUT'ПРОДОЛЖИТЬ РАСЧЕТ ? (1 - ДА, 0 - НЕТ ) 'S7
833 PRINT:PRINT
835 IF S7=1 THEN 1
900 END
```

После вычислений по программе получится следующий результат:
3.681708E 01

8. *Численное решение обыкновенного дифференциального уравнения методом прогноза-коррекции.* Алгоритм решения задачи дан, например, в пособиях [23, 35].

Тестовая задача. Численно решить дифференциальное уравнение $y'_x = -\frac{xy}{1+x^2}$ на отрезке [0; 0,3].

Возможный вариант программы приведен в п. 7.

После вычислений по программе получены результаты:

2.000000, 1.997504, 1.990074, 1.977872, 1.961161, 1.940285, 1.915652

9. *Нахождение безусловного экстремума функции двух переменных, модифицированным методом Ньютона–Рафсона.* Алгоритм решения задачи описан, например, в работах [34, 36, 37]. В программе предусмотрено вычисление градиента и матрицы вторых производных либо аналитически, либо конечно-разностным методом [1, 18].

Тестовая задача. Вычислить минимум функции Розенброка

$$F(x_1, x_2) = 100\,(x_2 - x_1^2)^2 + (1 - x_1)^2$$

при начальном значении вектора аргументов $[-0{,}5\ 0{,}5]^T$ с точностью по определению минимума $\epsilon = 0{,}000001$.

Возможный вариант программы:

```
  1 PRINT'*************************************************'
  2 PRINT'*ВЕЗУСЛОВНАЯ МИНИМИЗАЦИЯ ФУНКЦИИ 2-Х ПЕРЕМЕННЫХ**'
  3 PRINT'*********************************N***************'
  4 LETN=2
  5 DIM A(N,N+1),U(N),V(N),U1(N),V1(N)
  6 DIM X(N),H(2*N),B(N)
  8 DEF FNF(Z1)=100*(Z2-Z1¬2)¬2+(1-Z1)¬2
  9 LETH1=0.01
 12 INPUT'ВВЕДИТЕ НАЧ.ЗНАЧЕНИЕ X(1) 'X(1)
 14 INPUT'ВВЕДИТЕ НАЧ.ЗНАЧЕНИЕ X(2) 'X(2)
 16 INPUT'ВВЕДИТЕ ПРЕДПОЛАГАЕМЫЙ МИНИМУМ ФУНКЦИИ 'F1
 18 INPUT'ВВЕДИТЕ ТОЧНОСТЬ ВЫЧИСЛЕНИЙ ПО ФУНКЦИИ 'E6
 20 INPUT'ВВЕДИТЕ ТОЧНОСТЬ ВЫЧИСЛЕНИЙ ПО ГРАДИЕНТУ 'E5
 22 PRINT'ВВЕДИТЕ ПРИЗНАК ВЫЧИСЛЕНИЯ ГРАДИЕНТА '
 24 PRINT'И МАТРИЦЫ ВТОРЫХ ПРОИЗВОДНЫХ   '
 26 PRINT'      0 - ГРАДИЕНТ И МАТРИЦА ВТОРЫХ '
 27 PRINT'          ПРОИЗВОДНЫХ ОПРЕДЕЛЕНЫ '
 28 PRINT'          АНАЛИТИЧЕСКИ '
 29 PRINT'      1 - ГРАДИЕНТ И МАТРИЦА ВТОРЫХ'
 30 PRINT'          ПРОИЗВОДНЫХ ОПРЕДЕЛЯЮТСЯ'
 31 PRINT'          КОНЕЧНО - РАЗНОСТНЫМ МЕТОДОМ'
 32 INPUT P1
 40 LETZ1=X(1):LETZ2=X(2)
 76 PRINT'НАЧАЛЬНОЕ ЗНАЧЕНИЕ ФУНКЦИИ '1F1.61FNF(Z1)
 78 IF ABS(F1-FNF(Z1))<=E6 THEN 425
 79 IF P1=1 THEN 5060
 89 LETB(1)=-400*X(1)*X(2)+400*X(1)¬3+2*X(1)-2
 90 LETB(2)=200*X(2)-200*X(1)¬2
100 LETN1=SQR(B(1)¬2+B(2)¬2)
120 IF N1<=E5 THEN 425
130 LETH(1)=-400*X(2)+1200*X(1)¬2+2
140 LETH(2)=-400*X(1)
150 LETH(3)=H(2)
160 LETH(4)=200
171 LETB9=-H(1)-H(4):LETC9=H(1)*H(4)-H(2)*H(3)
172 IF (B9¬2-4*C9)<0 THEN 432
177 LETQ1=(-B9+SGN(B9)*SQR(B9¬2-4*C9))/2
179 LETQ2=C9/Q1
184 GOTO 189
```

```
 187 LETH(1)=1:LETH(2)=0:LETH(3)=0:LETH(4)=1
 189 LETT5=H(1)*H(4)-H(2)*H(3)
 191 IF T5<1E-12 THEN IF T5>=-1E-12 THEN 432
 192 IF T5>-1E-12 THEN IF T5<1E-12 THEN 432
 200 LETG1=H(4)/T5:LETG2=-H(2)/T5
 201 LETG3=-H(3)/T5:LETG4=H(1)/T5
 220 LETS1=-(G1*B(1)+G3*B(2)):LETS2=-(G2*B(1)+G4*B(2))
 305 LETI1=0
 310 LETE9=1E-5:LETE1=1E-3:LETA8=0:LETB8=10
 315 LETC=(A8+B8)*0.5
 316 LETZ1=X(1)+S1*C:LETZ2=X(2)+S2*C:LETC3=FNF(Z1)
 320 LETX2=C+E1:LETZ1=X(1)+S1*X2:LETZ2=X(2)+S2*X2
 321 LETX3=FNF(Z1)
 330 IF ABS((C3-X3)/C3)<=E9 THEN 390
 340 IF C3<X3 THEN 360
 350 LETA8=C:LETI1=I1+1:GOTO 370
 360 LETB8=X2:LETI1=I1+1:GOTO 370
 370 IF I1>=20 THEN 390
 380 GOTO 315
 390 LETW=(A8+B8)*0.5
 410 LETX(1)=X(1)+W*S1:LETX(2)=X(2)+W*S2
 415 LETZ1=X(1):LETZ2=X(2):LETI5=I5+1
 419 PRINT'ФУНКЦИЯ ='!F1.6!FNF(Z1);' АРГУМЕНТЫ = 'X(1);X(2)
 420 GOTO 78
 425 PRINT:PRINT
 426 PRINT'ЭКСТРЕМУМ = '!F1.6!FNF(Z1)
 427 PRINT'АРГУМЕНТ X(1) = 'X(1)
 428 PRINT'АРГУМЕНТ X(2) = 'X(2)
 429 PRINT'ИТЕРАЦИЙ    '16.0!I5
 430 PRINT'==========================='
 431 STOP
 432 PRINT'МАТРИЦА ВТОРЫХ ПРОИЗВОДНЫХ ВЫРОЖДЕНА
 433 STOP
5060 FOR I=1 TO N
5070 LETU(I)=X(I)+H
5072 LETV(I)=X(I)-H
5074 LETU1(I)=X(I)+H
5076 LETV1(I)=X(I)-H
5080 FOR L=1 TO N
5082 IF L=I THEN 85
5083 LETU(L)=X(L):LETV(L)=X(L):LETU1(L)=X(L):LETV1(L)=X(L)
5085 NEXT L
5086 LETZ1=X(1):LETZ2=X(2):LETF1=FNF(Z1)
5088 LETZ1=U(1):LETZ2=U(2):LETF2=FNF(Z1)
5090 LETZ1=V(1):LETZ2=V(2):LETF3=FNF(Z1)
5095 LETA(I,N+1)=(F3-F2)/H/2
5100 LETA(I,I)=(F3+F2-2*F1)/H/H
5110 FOR K=1 TO N
5112 IF K=I THEN 5130
5114 LETU(K)=X(K)+H
5115 LETV(K)=X(K)-H
5116 LETU1(K)=X(K)-H
5117 LETV1(K)=X(K)+H
5118 LETZ1=U(1):LETZ2=U(2):LETF4=FNF(Z1)
5120 LETZ1=V(1):LETZ2=V(2):LETF5=FNF(Z1)
5122 LETZ1=U1(1):LETZ2=U1(2):LETF6=FNF(Z1)
5124 LETZ1=V1(1):LETZ2=V1(2):LETF7=FNF(Z1)
5126 LETA(I,K)=(F4+F5-F6-F7)/4/H/H
5130 NEXT K
5140 NEXT I
5141 GOTO 171
```

Вычисления по программе дали следующие результаты:

МИНИМУМ=8.5504294E-07 АРГУМЕНТЫ 0.99912743 0.99822501

Для нахождения минимума функции одной переменной вместо метода дихотомии можно применять "метод золотого сечения".

Возможный вариант программы:

```
  1 PRINT'*********************************************'
  2 PRINT'** ПОИСК МИНИМУМА ФУНКЦИИ ОДНОЙ ПЕРЕМЕННОЙ **'
  3 PRINT'*********************************************'
  4 DEF FNA(X)=100*((0.5-0.2218*X)-(-0.5-0.0383*X)↑2)↑2
  5 DEF FNF(X)=FNA(X)+(1+0.5+0.0383*X)↑2
  6 LETG=0.6180339
 10 INPUT'ВВЕДИТЕ ГРАНИЦЫ ОТРЕЗКА (A-B) 'A,B
 14 INPUT'ВВЕДИТЕ ТОЧНОСТЬ ВЫЧИСЛЕНИЙ  'E
 20 LETI=0
 60 LETT1=A+(B-A)*G:LETT2=B-(B-A)*G
 90 LETY1=FNF(T1)
110 LETY2=FNF(T2)
120 IF Y1<Y2 THEN 200
125 LETB=T1:LETT1=T2:LETY1=Y2:LETT2=B-(B-A)*G
130 LETY2=FNF(T2):LETI=I+1
140 PRINT'ОТРЕЗОК A B  '!F1.4!A;B;!3.0!' ИТЕРАЦИЙ '; I
150 GOTO 210
200 LETA=T2:LETT2=T1:LETY2=Y1:LETT1=A+(B-A)*G
201 LETY1=FNF(T1):LETI=I+1
205 PRINT'ОТРЕЗОК A B  '!F1.4!A;B;!3.0!' ИТЕРАЦИЙ '; I
210 IF(B-A)>E THEN 120
220 LETW=(B+A)*0.5
230 LETF=FNF(W)
235 PRINT'++++++++++++++++++++++++++++++++++++'
240 PRINT'МИНИМАЛЬНОЕ ЗНАЧЕНИЕ ФУНКЦИИ          '!F1.6!F
250 PRINT'АРГУМЕНТ                              '!W
260 PRINT'ИТЕРАЦИЙ                              '!6.0!I
270 PRINT'ТОЧНОСТЬ ВЫЧИСЛЕНИЙ                   '!F1.5!E
275 PRINT'++++++++++++++++++++++++++++++++++++'
280 STOP
290 END
```

10. *Сортировка чисел в порядке возрастания методом Шелла и методом по индексам.* Алгоритм решения приведен, например, в пособиях [20, 22].

Возможный вариант программы:

```
 10 PRINT'**********************'
 20 PRINT'** СОРТИРОВКА ЧИСЕЛ **'
 30 PRINT'**********************'
 40 CLEARD:PRINT:PRINT
 50 PRINT'КОЛИЧЕСТВО ЧИСЕЛ?
 60 INPUT N
 70 DIM A(N)
 80 PRINT '      1 - СОРТИРОВКА ПО ИНДЕКСАМ '
 90 INPUT '      2 - СОРТИРОВКА МЕТОДОМ ШЕЛЛА 'T
100 PRINT 'ВВЕДИТЕ ЧИСЛА '
110 FOR I=1 TO N
120 INPUT A(I)
130 NEXT I
140 IF T=1 THEN 330
150 LETD=1
160 LETD=2*D
170 IF D<=N THEN 160
180 LETD=INT((D-1)/2)
190 IF D=0 THEN 320
200 LETN1=N-D
210 FOR I=1 TO N1
220 LETJ=I
230 LETL=J+D
240 IF A(L)>=A(J) THEN 300
250 LETX=A(J)
260 LETA(J)=A(L)
270 LETA(L)=X
280 LETJ=J-D
290 IF J>0 THEN 230
300 NEXT I
310 GOTO 180
320 GOTO 440
330 FOR I=1 TO N-1
340 LETJ=I+1
350 FOR K=J TO N
360 IF A(I)<=A(K) THEN 400
370 LETB=A(I)
380 LETA(I)=A(K)
390 LETA(K)=B
400 NEXT K
410 NEXT I
420 PRINT'******************'
430 PRINT:PRINT
440 PRINT'*****  ОТВЕТ  *****'
450 PRINT:PRINT
460 FOR I=1 TO N
470 PRINT !4.0!I;'A('I')='!F1.6!A(I)
480 NEXT I
490 PRINT '************************'
500 PRINT 'ЕЩЕ СОРТИРОВАТЬ ЧИСЛА?  '
510 PRINT '     0 - НЕТ '
520 INPUT '     1 - ДА  'T1
530 IF T1=1 THEN 40
540 STOP
550 END
```

Ограничения: количество чисел не должно быть больше 255.

Если необходимо выполнить сортировку чисел в порядке убывания, то в операторе 240 символ "> =" надо заменить на "< =", а в операторе 360 символ "< =" – на "> =".

11. *Линейная интерполяция.* Алгоритм решения задачи описан, например, в работах [1, 14, 18].

Тестовая задача. Найти значения функции Y = F (X), заданной таблично, при N = 4 и X1 = 0.3:

X	0	0.1	0.5	0.725
Y	3.375	2	8	4

Возможный вариант программы:

```
010 PRINT'****************************'
020 PRINT'**  ЛИНЕЙНАЯ ИНТЕРПОЛЯЦИЯ' **'
030 PRINT'****************************'
040 PRINT:PRINT
050 INPUT'СКОЛЬКО АРГУМЕНТОВ ?  'N
060 INPUT'ВВЕДИТЕ X1  'X1
070 DIM X(N),Y(N)
080 FOR I=1 TO N
090 READ X(I),Y(I)
100 NEXT I
110 FOR I=2 TO N
120 IF X(I)>=X1 THEN 140
130 NEXT I
140 LETI=I-1
150 LETY1=Y(I)+(Y(I+1)-Y(I))*(X1-X(I))
160 LETY1=Y1/(X(I+1)-X(I))
170 PRINT'X1='X1;'Y1='Y1
180 DATA 0,3.375,0.1,2,0.5,8,0.725,4
190 END
```

После вычислений по программе получится следующий результат:

Y1 = 5

12. *Интерполяция сплайнами.* Алгоритм решения приведен, например, в пособиях [1, 18, 34, 37].

Тестовая задача. Проинтерполировать функцию Рунге $y = \dfrac{1}{25x^2+1}$ на отрезке [−1; 1] двадцатью кубическими полиномами вида $a_i + (x - x_i)b_i + (x - x_i)^2 c_i + (x - x_i)^3 d_i$, $x_i < x < x_{i+1}$, $i = 1, 2, ..., N$.

Возможный вариант программы:

```
010 PRINT'****************************'
020 PRINT'** ИНТЕРПОЛЯЦИЯ СПЛАЙНАМИ **'
030 PRINT'****************************'
040 INPUT 'ЧИСЛО ИНТЕРВАЛОВ ?      'N
050 LETN1=N+1
060 DIMX(N1),Y(N1)
070 DIM B(N1),C(N1),D(N1)
080 PRINT'ВВОДЯТСЯ ДАННЫЕ
090 PRINT:PRINT
100 LETX=-1.0
110 FOR I=1 TO N1
120 LETY=1/(25*X(I)*X(I)+1)
```

```
130 LETY=1/(25*X(I)*X(I)+1)
140 LETX=X+0.1*I
150 NEXT I
160 GOSUB 190
170 GOSUB 630
180 STOP
190 LETN2=N-1
200 IF N<2 THEN 610
210 IFN<3 THEN 570
220 LETD(1)=X(2)-X(1)
230 LETC(2)=(Y(2)-Y(1))/D(1)
240 FOR I=2 TO N2
250 LETD(I)=X(I+1)-X(I)
260 LETB(I)=2*(D(I-1)+D(I))
270 LETC(I+1)=(Y(I+1)-Y(I))/D(I)
280 LETC(I)=C(I+1)-C(I)
290 NEXT I
300 LETB(1)=-D(1):LETB(N)=-D(N-1)
310 LETC(1)=0:LETC(N)=0
320 IF N=3 THEN 370
330 LETC(1)=C(3)/(X(4)-X(2))-C(2)/(X(3)-X(1))
340 LETC(N)=C(N-1)/(X(N)-X(N-2))-C(N-2)/(X(N-1)-X(N-3))
350 LETC(1)=C(1)*D(1)*D(1)/(X(4)-X(1))
360 LETC(N)=-C(N)*D(N-1)*D(N-1)/(X(N)-X(N-3))
370 FOR I=2 TO N
380 LETW=D(I-1)/B(I-1)
390 LETB(I)=B(I)-W*D(I-1)
400 LETC(I)=C(I)-W*C(I-1)
410 NEXT I
420 LETC(N)=C(N)/B(N)
430 FOR J=1 TO N2
440 LETI=N-J
450 LETC(I)=(C(I)-D(I)*C(I+1))/B(I)
460 NEXT J
470 LETB(N)=(Y(N)-Y(N2))/D(N2)+D(N2)*(C(N2)+2*C(N))
480 FOR I=1 TO N2
490 LETB(I)=(Y(I+1)-Y(I))/D(I)
500 LETB(I)=B(I)-D(I)*(C(I+1)+2*C(I))
510 LETD(I)=(C(I+1)-C(I))/D(I)
520 LETC(I)=3*C(I)
530 NEXT I
540 LETC(N)=3*C(N)
550 LETD(N)=D(N-1)
560 RETURN
570 LETB(I)=(Y(2)-Y(1))/(X(2)-X(1))
580 LETC(1)=0:LETD(1)=0:LETB(2)=B(1)
590 LETC(2)=0:LETD(2)=0
600 RETURN
610 PRINT 'N='!2.0!N;'  СЧЕТ ПРЕКРАЩЕН ! '
620 STOP
630 PRINT'*********************************************'
640 PRINT'ИНТЕРВАЛ КОЭФФ.A  КОЭФФ.B  КОЭФФ.C  КОЭФФ.D *'
650 PRINT'*********************************************'
660 PRINT!2.0!I;'    '!F1.6!Y(I);'    'B;'  'C;'    'D
670 NEXT I
680 PRINT'*********************************************'
690 RETURN
700 END
```

В результате вычислений по программе получаем:

ИНТЕРВАЛ	КОЭФФ. A	КОЭФФ. B	КОЭФФ. C	КОЭФФ. D
1	3.8462−02	7.5623−02	7.4900−02	2.8601−01
2	4.7059−02	9.9183−02	1.6070−01	2.3938−01
3	5.8824−02	1.3850−01	2.3252−01	4.7250−01
4	7.5472−02	1.9918−01	3.7427−01	8.6728−01
5	1.0000−01	3.0006−01	6.3445−01	1.5810+00
6	1.3793−01	4.7438−01	1.1087+00	3.5440+00
7	2.0000−01	8.0244−01	2.1719+00	5.7284+00
8	3.0769−01	1.4087+00	3.8905+00	1.2534+01
9	5.0000−01	2.5628+00	7.6508+00	−3.2789+01
10	8.0000−01	3.1093+00	−2.1859+00	−8.9070+01
11	1.0000+00	2.5373−08	−2.8907+01	8.9070+01
12	8.0000−01	−3.1093+00	−2.1859+00	3.2789+01
13	5.0000−01	−2.5628+00	7.6508+00	−1.2535+01
14	3.0769−01	−1.4087+00	3.8904+00	−5.7280+00
15	2.0000−01	−8.0244−01	2.1720+00	−3.5453+00
16	1.3793−01	−4.7439−01	1.1084+00	−1.5759+00
17	1.0000−01	−2.9999−01	6.3564−01	−8.8595−01
18	7.5472−02	−1.9944−01	3.6985−01	−4.0282−01
19	5.8823−02	−1.3755−01	2.4900−01	−4.9945−01
20	4.7059−02	−1.0273−01	9.9171−02	−4.9945−01

12. Тексты внешних подпрограмм обмена с УСО

ПОДПРОГРАММА НОМЕР 7

ПЕРВЫЙ БЛОК

```
00000  00 00
00001  00 07
00002  00 05
00003  00 13
00004  00 00
00005  06 01
00006  00 00
00007  05 12
```

ВТОРОЙ БЛОК

```
00000  10 09  02 12   BNS #02 12,S3 00004
00002  14 03  01 01   BR 00020
00004  13 00  00 00   MOV #00 00,S00
00006  13 01  03 02   MOV #03 02,S01
00008  11 05  10 15   MOV R15,R10
00010  04 13  08 10   ABS R10
00012  11 00  08 10   ADD R08,R10
00014  10 13  06 06   JSTT 06 06
00016  10 13  03 04   JSTT 03 04
00018  04 07  04 08   JMM 04 08
00020  10 13  06 12   JSTT 06 12
00022  10 13  04 14   JSTT 04 14
00024  06 04          MOV X,Y
00025  07 02          DIG 2
00026  07 05          DIG 5
00027  07 06          DIG 6
00028  05 07          BGE Y,X 00031
00029  14 03  00 07   BR 00037
00031  13 00  00 00   MOV #00 00,S00
00033  13 01  03 15   MOV #03 15,S01
00035  14 02  01 12   BR 00008
00037  06 05          MOV Y,X
00038  04 13  12 11   MOVH X,R11
00040  13 00  15 01   MOV #15 01,S00
00042  13 01  14 01   MOV #14 01,S01
00044  13 04  07 15   MOV #07 15,S04
00046  13 05  01 07   MOV #01 07,S05
00048  13 09  00 01   MOV #00 01,S09
00050  04 13  02 08   CMD R08
00052  13 01  15 07   MOV #15 07,S01
00054  13 07  15 15   MOV #15 15,S07
00056  04 13  02 08   CMD R08
00058  05 11          RTS
00059  06 14   Н
00060  06 05   Е
00061  07 04   Т
00062  02 00
00063  07 10   З
00064  06 01   А
00065  07 00   П
00066  07 01   Я
00067  07 04   Т
00068  06 15   О
00069  06 10   Й
00070  02 00
00071  00 00
00072  06 14   Н
00073  06 05   Е
00074  06 04   Д
```

```
00075  06 15   О
00076  07 00   П
00077  07 05   У
00078  07 03   С
00079  07 04   Т
00080  06 09   И
00081  06 13   М
00082  06 15   О
00083  06 05   Е
00084  02 00
00085  07 10   З
00086  06 14   Н
00087  06 01   А
00088  07 14   Ч
00089  06 05   Е
00090  06 14   Н
00091  06 09   И
00092  06 05   Е
00093  02 00
00094  00 00
00095  00 00
00096  05 12          END
```

ПОДПРОГРАММА НОМЕР 21

ПЕРВЫЙ БЛОК

```
00000  00 00
00001  02 01
00002  00 07
00003  03 09
00004  00 00
00005  08 08
00006  00 00
00007  05 12
```

ВТОРОЙ БЛОК

```
00000  10 09  02 12   BNS #02 12,S3 00004
00002  14 03  01 01   BR 00020
00004  13 00  00 00   MOV #00 00,S00
00006  13 01  05 09   MOV #05 09,S01
00008  11 05  10 15   MOV R15,R10
00010  04 13  08 10   ABS R10
00012  11 00  08 10   ADD R08,R10
00014  10 13  06 06   JSTT 06 06
00016  10 13  03 04   JSTT 03 04
00018  04 07  04 08   JMM 04 08
00020  10 13  06 12   JSTT 06 12
00022  10 13  04 14   JSTT 04 14
00024  13 00  15 01   MOV #15 01,S00
00026  13 04  07 15   MOV #07 15,S04
00028  13 05  01 07   MOV #01 07,S05
00030  13 09  00 01   MOV #00 01,S09
00032  06 04          MOV X,Y
00033  07 00          DIG 0
00034  05 09          BEQ Y,X 00037
00035  14 03  00 03   BR 00039
00037  14 03  00 07   BR 00045
00039  07 01          DIG 1
00040  05 09          BEQ Y,X 00043
00041  14 03  00 15   BR 00057
00043  14 03  00 07   BR 00051
00045  13 01  14 02   MOV #14 02,S01
00047  04 13  02 08   CMD R08
```

```
00049  14 03  00 13   BR 00063
00051  13 01  14 03   MOV #14 03,S01
00053  04 13  02 08   CMD R08
00055  14 03  00 07   BR 00063
00057  13 00  00 00   MOV #00 00,S00
00059  13 01  06 06   MOV #06 06,S01
00061  14 02  03 06   BR 00008
00063  10 09  02 12   BNS #02 12,S3 00067
00065  14 03  00 03   BR 00069
00067  14 02  04 00   BR 00004
00069  13 00  15 00   MOV #15 00,S00
00071  13 01  14 04   MOV #14 04,S01
00073  04 13  02 08   CMD R08
00075  13 06  00 00   MOV #00 00,S06
00077  10 12  11 13   MOV R11,-(R13)
00079  10 13  00 06   JSTT 00 06
00081  10 15  11 13   MOV (R13)+,R11
00083  04 13  03 11   MOV R11,X
00085  11 03  04 08   BSAZ R08 00090
00087  10 13  01 12   JSTT 01 12
00089  05 11          RTS
00090  10 13  06 14   JSTT 06 14
00092  11 05  08 10   MOV R10,R08
00094  14 02  00 08   BR 00087
00096  05 14          GO
00097  05 14          GO
00098  06 14   Н
00099  06 05   Е
00100  07 04   Т
00101  02 00
00102  07 10   З
00103  06 01   А
00104  07 00   П
00105  07 01   Я
00106  07 04   Т
00107  06 15   О
00108  06 10   Й
00109  02 00
00110  00 00
00111  06 14   Н
00112  06 05   Е
00113  06 04   Д
00114  06 15   О
00115  07 00   П
00116  07 05   У
00117  07 03   С
00118  07 04   Т
00119  06 09   И
00120  06 13   М
00121  06 15   О
00122  06 05   Е
00123  02 00
00124  07 10   З
00125  06 14   Н
00126  06 01   А
00127  07 14   Ч
00128  06 05   Е
00129  06 14   Н
00130  06 09   И
00131  06 05   Е
00132  02 00
00133  00 00
00134  00 00
00135  05 12          END
```

ЛИТЕРАТУРА

1. *Бахвалов Н.С.* Численные методы. – М.: Наука, 1975. – 631 с.

2. *Березин И.С., Жидков Н.Л.* Методы вычислений: В 2 т. – М.: Физматгиз, 1959. – Т. 1. – 464 с.

3. *Березин И.С., Жидков Н.Л.* Методы вычислений: В 2 т. – М.: Физматгиз, 1962. – Т. 2. – 639 с.

4. *Бобков В.Я., Городецкий Л.М.* Избранные численные методы решения на ЭВМ инженерных и научных задач. – Мн.: Изд-во Университетское, 1985. – 173 с.

5. *Бронштейн И.Н., Семендяев К.А.* Справочник по математике. – М.: Наука, 1981. – 718 с.

6. *Бутомо И.Д., Самочадин А.В., Усанова О.В.* Программирование на алгоритмическом языке Паскаль для микроЭВМ. – Л.: Изд-во ЛГУ, 1985. – 216 с.

7. *Валикова Л.И., Вигдорчик А.Ю. Воробьев А.Ю., Лукин А.А.* Операционная система СМ ЭВМ РАФОС. – М.: Финансы и статистика, 1984. – 206 с.

8. *Вигдорчик Г.В., Воробьев А.Ю., Праченко В.Д.* Основы программирования на ассемблере для СМ ЭВМ. – М.: Финансы и статистика, 1983. – 256 с.

9. *Воеводин В.В.* Вычислительные основы линейной алгебры. – М.: Наука, 1977. – 303 с.

10. *Воробьева Г.Н., Данилова А.Н.* Практикум по численным методам. – М.: Высш. шк., 1979. – 182 с.

11. Вычислительная техника в инженерных и экономических расчетах / А.В. Петров, В.Е. Алексеев, М.А. Титов и др.; Под ред. А.В. Петрова. – М.: Высш. шк., 1984. – 320 с.

12. *Гилл А.* Программирование на языке ассемблера для PDP-11. – М.: Радио и связь, 1983. – 159 с.

13. *Дамке М.* Операционные системы микроЭВМ. – М.: Финансы и статистика, 1985. – 151 с.

14. *Демидович Б.П., Марон И.А.* Основы вычислительной математики. – М.: Наука, 1966. – 664 с.

15. *Демидович Б.П., Марон И.А., Шувалова Э.З.* Численные методы анализа. – М.: Физматгиз, 1963. – 400 с.

16. *Дьяконов В.П.* Расчет нелинейных и импульсных устройств на программируемых микрокалькуляторах. – М.: Радио и связь, 1984. – 176 с.

17. Инженерные расчеты на ЭВМ/В.А. Троицкий, И.М. Иванова, И.А. Старостин, В.Д. Шелест; Под ред. В.А. Троицкого. – Л.: Машиностроение; Ленингр. отд-ние, 1979. – 257 с.

18. *Калиткин Н.Н.* Численные методы. – М.: Наука, 1978. – 512 с.

19. *Кетков Ю.Л.* Программирование на БЭЙСИКе. – М.: Статистика, 1978. – 157 с.

20. *Кнут Д.* Искусство программирования для ЭВМ: В 3 т. – М.: Мир, 1976–1978. – Т. 1. – 1976. – 734 с.; Т. 2. – 1977. – 729 с.; Т. 3. – 1978. – 843 с.

21. *Корн Г., Корн Т.* Справочник по математике. – М.: Наука, 1974. – 831 с.

22. *Ламуатье Ж.-П.* Упражнения по программированию на Фортране. – М.: Мир, 1978. – 162 с.

23. *Мак-Кракен Д., Дорн У.* Численные методы и программирование на Фортране. – М.: Мир, 1977. – 584 с.

24. *Мейер Б., Бодуэн К.* Методы программирования: В 2 т. – М.: Мир, 1982. – Т. 1. – 356 с.; Т. 2. – 368 с.

25. Микропроцессоры: В 9 кн./ Под ред. Л.Н. Преснухина. – М.: Высш. шк., 1985. – Кн. 9: Сборник примеров и задач. – 94 с.

26. Программирование на языке БЭЙСИК-плюс для СМ-4 / В.П. Семик, Б.Р. Монцибович, Д.П. Непочатых и др. – М.: Финансы и статистика, 1982. – 245 с.

27. *Прокофьев В.А.* Программирование для мини-ЭВМ. – М.: Сов. радио, 1979. – 77 с.

28. Сборник научных программ на Фортране: В 2 кн.: Пер. с англ. под ред. С.Я.Виленкина. – М.: Статистика, 1974. – Кн. 1. – 314 с.; Кн. 2. – 221 с.

29. *Сингер М.* Мини-ЭВМ PDP-11. Программирование на языке ассемблера и организация машины. – М.: Мир, 1984. – 272 с.

30. Система ВАНГ-2200В: Руководство для пользователей. – Рига: Зинатне, 1974. – 254 с.

31. *Уилкинсон Дж., Райнш К.* Справочник алгоритмов на языке Алгол. Линейная алгебра. – М.: Машиностроение, 1976. – 389 с.

32. *Уокерли Дж.* Архитектура и программирование микроЭВМ: В 2 кн. – М.: Мир, 1984. – Кн. 1. – 486 с.; Кн. 2. – 359 с.

33. *Уорт Т.* Программирование на языке БЭЙСИК. – М.: Машиностроение, 1981. – 253 с.

34. *Форсайт Дж., Малькольм М., Моулер К.* Машинные методы математических вычислений. – М.: Мир, 1980. – 279 с.

35. *Хемминг Р.В.* Численные методы. – М.: Наука, 1972. – 490 с.

36. *Химмельблау Д.* Прикладное нелинейное программирование. – М.: Мир, 1975. – 534 с.

37. *Щуп Т.* Решение инженерных задач на ЭВМ. – М.: Мир, 1982. – 235 с.

38. *Экхауз Р., Моррис Л.* Мини-ЭВМ: Организация и программирование. – М.: Финансы и статистика, 1983. – 358 с.

39. *Ярмиш Р., Ярмиш Дж.* Основы программирования на языке ассемблера: В 2 кн.– М: Мир, 1983. – Кн. 1. – 320 с.; Кн. 2. – 567 с.

ОГЛАВЛЕНИЕ

4. Практическая работа с микроЭВМ "Электроника Д3-28"

5. Программирование на микроЭВМ "Искра 226"

6. Программирование на микроЭВМ ДВК-1

Виктор Филиппович Аникеенко
Борис Михайлович Киселев
Владимир Иванович Убийконь

ПРОГРАММИРОВАНИЕ НА МИКРОЭВМ

Зав. редакцией *Е.В. Сукач*
Редактор *М.С. Молчанова*
Мл. редактор *В.М. Кушилевич*
Худож. редактор *Ю.С. Сергачев*
Техн. редактор *Г.А. Лакишик*
Корректоры *В.П. Шкредова, И.И. Ганелес*
Оператор *А.И. Маль*

ИБ № 2484

Подписано в печать 31.07.87. АТ 16727. Формат 60x90 1/16. Бумага офсет. Офсетная печать. Гарнитура Пресс Роман. Усл. печ. л. 12 . Усл. кр.-отт. 12. Уч.-изд. л. 13,04. Тираж 62000 экз. Зак. 645. Цена 1 р. 10 к.

Издательство "Вышэйшая школа" Государственного комитета БССР по делам издательств, полиграфии и книжной торговли. 220048, Минск, проспект Машерова, 11.

Ордена Трудового Красного Знамени полиграфкомбинат МППО им. Я. Коласа. 220005, Минск, ул. Красная, 23.

Отпечатано с оригинала-макета, подготовленного в издательстве "Вышэйшая школа".

Опечатки

Страница	Строка	Напечатано	Следует читать
19	13 сверху	(BD)	$(BD)'$
20	10 и 11 сверху	R0θ ← ((R11));	R08 ← ((R11)); (R11) + 2
43	15 снизу	в T0	в T00
51	2 снизу	50 K = K * 1	50 K = K * I
58	23 сверху	2, 10, 8.2, 64.7	2, 10, 8.2, 64, 7
66	1 снизу	(F(−1)) =	(F(−1))! =
109	8 снизу	¤ GIO′	¤ GIO,
115	4 сверху	X ⊕ Y	$\overline{X \oplus Y}$
115	9 сверху	X V Y	$\overline{X}$ V Y
125	14 снизу	SAVE DA $\left\{\begin{matrix}F\\R\\T\end{matrix}\right\}$ ¤	SAVE DA $\left\{\begin{matrix}F\\R\\T\end{matrix}\right\}$ [¤]
133	9 сверху	I0T	IOT
145	12 и 11 снизу	07 09 DIG 9	... ... 07 09 DIG 9
154	10 и 11 сверху	$\left\{\begin{matrix}[\#0]\\\#1\end{matrix}\right\}$	$\left\{\begin{matrix}[\#0]\\\#1\end{matrix}\right.$

Цена 10 к.

Издательство «Советская школа»