

ПРИЛОЖЕНИЕ 1

РУКОВОДСТВО ПО
ЭКСПЛУАТАЦИИ ПЭВМ
АПОГЕЙ — БК01

СИСТЕМА КОМАНД
И ПРОГРАММИРОВАНИЕ
НА АССЕМБЛЕРЕ

В в е д е н и е

Существует много различных языков программирования высокого уровня. Вы, наверное, слышали о таких из них, как БЕЙСИК, ФОРТРАН, ПАСКАЛЬ. Они отличаются от нашего разговорного языка, но достаточно просто и удобно, позволяют описать ход решения задачи. Выучив один из таких языков, Вы сможете "общаться" с ЭВМ самых разных типов, если для данного компьютера разработан соответствующий транслятор. Транслятор "объяснит" компьютеру, что Вы от него требуете, и компьютер выполнит Вашу программу. Программируя на языке высокого уровня, Вы далеки от ЭВМ, не чувствуете ее особенностей: не до конца используете ее возможности. "Виноград" в этом слишком "уголовный": переводчик - транслятор, который скрывает от Вас все, что происходит в ЭВМ, и готов сделать все, чтобы никаких дополнительных знаний, кроме алгоритмического языка, Вам для "разговора" с ЭВМ не понадобилось. Конечно, приятна такая работа, но в этом случае для Вас возможности алгоритмического языка и возможности ЭВМ почти равно и тоже, а что происходит внутри ЭВМ, как она "думает", какими словами "мыслит", решая Вашу задачу, остается для Вас непонятным. Понять же все это Вы сможете, если научитесь программировать на языке АССЕМБЛЕРА.

1. Язык АССЕМБЛЕРА

Что же такое АССЕМБЛЕР? Во-первых, АССЕМБЛЕР - это язык программирования, максимально приближенный к языку, понятному ЭВМ. Каждая команда машинного языка имеет соответствующее символическое представление (инимонику) в языке АССЕМБЛЕРА. Во-вторых, ассемблер - это программа, позволяющая перевести программу, написанную на языке АССЕМБЛЕРА в язык машинных команд. Чтобы разгадать эти два понятия, в случае, когда речь идет о программе, будем писать ассемблер, а вместо слов "язык АССЕМБЛЕРА" - АССЕМБЛЕР.

АССЕМБЛЕР не обещает Вам легкой жизни, так как, кроме языка, понятного ассемблеру, Вам необходимо знать о том, как представляется информация в ЭВМ, что входит в состав ЭВМ.

2. Представление информации в ЭВМ

В персональной ЭВМ "Апогей-БКО1" (далее по тексту - ПЭВМ) для хранения информации используется двоичное представление. Это значит, что каждый элемент информации, хранящийся в ПЭВМ, имеет только два значения: либо "0", либо "1". В ПЭВМ эти значения реализуются в виде уровней напряжения. Причем, "0" представляется низким (0 В), а "1" - высоким (+5 В) уровнями.

В силу двоичного представления информации в ПЭВМ для вычисления используется система счисления по основанию 2, или двоичная система. В этой системе счисления употребляются только две цифры: 0 и 1. В обычной жизни мы используем десятичную систему счисления. В десятичной системе счисления употребляется десять различных цифр от 0 до 9.

Ограничения десятичной системы счисления десятичными цифрами не мешает записывать с их помощью большие значения. Для этого используются многозначные числа, где каждая цифра числа соответствует различной степени 10. Самая правая цифра указывает на число единиц. Следующая влево цифра обозначает число десятков. Следующая - число сотен и т.д. Этому перемещению справа налево соответствует прогрессия: $10^0, 10^1, 10^2$ и т.д. Число 2375 - это в действительности 2 тысячи, 3 сотни, 7 десятков и 5 единиц. На рис.2.1. приведено соответствующее разложение числа 2375.

$$\begin{aligned} 2375 &= 2 \cdot 10^3 + 3 \cdot 10^2 + 7 \cdot 10^1 + 5 \cdot 10^0 = \\ &= 2000 + 300 + 70 + 5 \end{aligned}$$

Рис.2.1. Десятичное представление

Система счисления по основанию 2, или двоичная система, полностью аналогична десятичной системе за исключением того, что разряды числа в ней соответствуют не степеням числа 10, а степеням числа 2. Значения, большие 1 представляются в ней многозначными числами, точно так же, как в системе по основанию 10 представляются значения, большие 9. Любая цифра двоичного числа называется битом. Какой бит в числе соответствует какой-либо степени числа 2. На рис. 2.2. показано, чему равно значение двоичного числа 1011010. Для обозначения того, что приводится двоичное представление числа, в конце его записи мы будем ставить латинскую букву В. Это позволит разграничить двоичные и десятичные числа, так как у последних буква в конце отсутствует.

$$\begin{aligned} 1011010 &= 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = \\ &= 32 + 0 + 8 + 4 + 0 + 2 + 0 = \\ &= 46 \end{aligned}$$

Рис.2.2 Двоичное представление

Этот термин. Максимальное десятичное число, задаваемое восьмизначным двоичным числом, равно 255. То есть один байт может иметь 256 значений (от 0 до 255). Группы из шестнадцати бит или аяв байт называют словом. Слово может иметь 65536 значений (от 0 до 65535).

Каждый байт информации можно рассматривать не как аявчисловое число, а как представление какого-либо символа. В ПЗВМ символы задаются в коде КОИ-7, в котором для представления символа используются только семь младших разрядов байта. Код символа обычно представляется в шестнадцатичной системе счисления (см. табл. 1 и 2 Приложение А). Например, символ "1" имеет код 31H, а код 41H соответствует символу "А". Для перевода шестнадцатичных чисел в десятичные прелагам Вам воспользоваться табл. 5 (Приложение А).

Двоичная арифметика удобна для ПЗВМ, так как она работает только с единицами и нулями. Однако человеческое восприятие требует более компактного представления чисел. В описываемом далее языке АССЕМБЛЕРА мы будем использовать шестнадцатичное представление чисел.

Шестнадцатичное представление чисел основывается на системе счисления по основанию 16. Каждая цифра в таком числе может иметь значение от 0 до 15, а каждый разряд соответствует степени числа 16. Шестнадцатичное представление является удобным методом записи двоичной информации. Каждая шестнадцатичная цифра соответствует четырем битам. Для преобразования двоичного числа в шестнадцатичное надо разбить последовательность бит на тетрады (4 бита), начиная с младших разрядов; затем каждую тетраду заменить соответствующим шестнадцатичным эквивалентом, используя табл. 2.1. Из табл. 2.1. видно, что значения от 10 до 15 в шестнадцатичном представлении записываются с помощью первых шести букв латинского алфавита от А до F. Шестнадцатичные числа помещаются в конце латинской буквой H. Этот способ обозначения полностью соответствует вет записи чисел в языке АССЕМБЛЕРА.

Таблица 2.1.

Десятичное	Двоичное	Шестнадцатичное
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Битом мы условились называть один двоичный разряд; значением которого может быть "либо 0, либо 1". Для удобства вернем специальные названия для некоторых последовательностей бит. Группы из восьми бит обычно называют байтом; и в дальнейшем для обозначения восьмибитовых данных будем использовать

3. Структура компьютера

Любой современный компьютер содержит процессор, память и систему ввода-вывода (см. рис. 3.1). Процессор — это основной элемент компьютера, осуществляющий процесс обработки данных и управляющий этим процессом. Процессор дешифрует (т.е. распознает) и выполняет команды программы, организуя обращение к оперативной памяти и, через систему ввода-вывода к внешним устройствам. Процессор осуществляет управление взаимодействием всех устройств, входящих в состав ЭВМ. Процессор — это "мозг" компьютера. Но для работы "мозга" нужна память! Где храниться бы исходные данные и результаты вычислений. Эти функции выполняет оперативная память. Кроме данных, в оперативной памяти хранятся и программы, которым выполняет процессор. Процессор может считывать информацию из оперативной памяти, а так же записывать туда новую информацию. Но при отключении питания вся хранящаяся там информация исчезает, поэтому в состав компьютера обычно входит постоянное запоминающее устройство (ПЗУ). ПЗУ функционирует аналогично оперативному запоминающему устройству (ОЗУ) с той лишь разницей, что его содержимое всегда остается неизменным. Иначе говоря, ПЗУ предназначено только для считывания хранящихся в нем данных. После отключения питания компьютера ПЗУ сохраняет данные, записанные в него при изготовлении. ПЗУ используется для хранения программ, выполняемых после включения питания компьютера. Система ввода-вывода обеспечивает обмен информацией между процессором, памятью и внешними устройствами (дисплей, клавиатура, кассетный магнитофон и т.п.) и служит для преобразования информации из формы, используемой ЭВМ, в форму, пригодную для восприятия объектом или человеком.

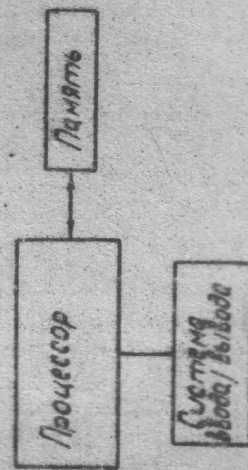


Рис. 3.1. Общая структурная схема ЭВМ

Теперь рассмотрим подробнее структурную схему персонального компьютера "Апогей-БК01", представленную на рис. 3.2. Из данной схемы видно, что микропроцессор (МП), являющийся "мозгом" ПЗВМ, связан со всеми другими компонентами машины посредством трех многопроводных магистралей (шин). Под шиной подразумевается множество линий передачи сигналов, объединенных в группу по своему функциональному назначению. Физически шины реализуются в виде параллельных проводящих полосок печатной платы или в виде связанных в жгут проводов. Так, обмен данными между всеми компонентами ПЗВМ осуществляется по 8-разрядной шине данных (ШД). ШД является двуправленной, то есть, в процессе работы ПЗВМ данные могут передаваться в двух направлениях: из МП в МП. Кроме шин данных в компьютере имеются шина адреса (ША) и шина управления (ШУ). В отличие от шин данных ША и ШУ являются однопроводными, т.е. информация по ним передается только из микропроцессора к остальным компонентам ПЗВМ. Информация, передаваемая по ША, служит для выбора конкретной ячейки памяти (конкретного внешнего устройства), с которой требуется произвести обмен информацией. В ПЗВМ реализована 16-разрядная шина адреса. Таких образцов, МП нашего компьютера имеет возможность адресовать 2¹⁶ (65536) ячеек памяти. Обычно говорят: не 65536 ячеек, а 64К (1К равен 1024 (2¹⁰)). Память представляет собой совокупность пронумерованных ячеек для хранения команд и данных. Номер ячейки — это ее адрес. Каждая ячейка памяти содержит 1 байт (8 бит), что соответствует разрядности (длине) числа, обрабатываемого МП. ШУ состоит из двух линий передачи сигналов ЧТБ-НИК и ЗАПИСЬ, которые определяют направление обмена информацией (чтение из памяти в МП и запись в память из МП). Обращение к внешним устройствам (ВУ) нашего компьютера происходит так же, как к ячейкам памяти: посылку адреса ВУ и адреса ячеек памяти совмещены и находятся в едином адресном пространстве. Распределение адресного пространства ПЗВМ "Апогей-БК01" приведено на рис. 3.3.

Память ПЗВМ состоит из ОЗУ и ПЗУ. ПЗУ содержит программу МОНИТОР и занимает 4 кбайта адресного пространства. Оперативная память ПЗВМ расположена с адреса 0000Н по ВАРН и разделена на три области. Экранная область ОЗУ служит для хранения кодов символов, отображаемых на экране дисплея и занимает 2340 байт. Область рабочих ячеек МОНИТОРА занимает 208 байт и предназначена для хранения служебной информации, необходимой для работы программы МОНИТОР. Третья область ОЗУ находится в полном распоряжении пользователя и служит для хранения его программ и данных. ОЗУ пользователя имеет объем 56,23 Кбайт.

Система ввода-вывода обеспечивает связь ПЗВМ с внешними устройствами ввода-вывода информации (УВВ), без которых работа компьютера была бы бессмысленной. Основными устройствами ввода информации в ПЗВМ являются клавиатура и кассетный маг-

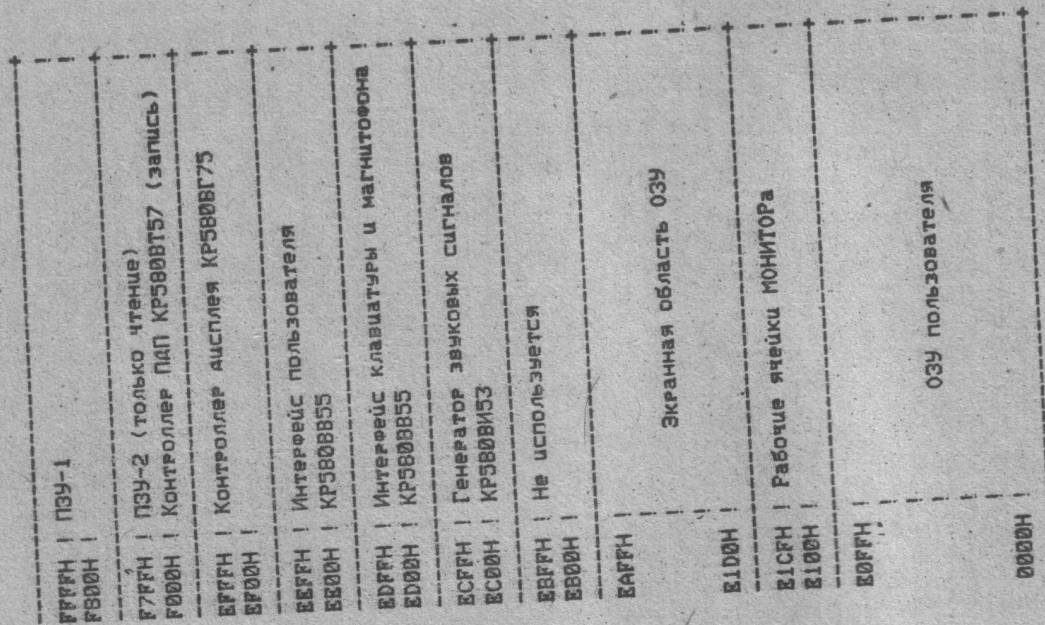


Рис. 3.3. Распределение адресного пространства ПЗМ.

нителей. Для подключения этих устройств используется программ-
ируемый периферийный адаптер (ПАП) КР580ВТ57. Эта микросхема
имеет три 8-разрядных программируемых канала ввода-вывода: А,
В и С. Через настроенный на вывод канал А осуществляется ска-
нирование (опрос) клавиатуры, которая представляет собой мат-
рицу нормально разомкнутых контактов. Информация о нажатых в
данный момент клавишах поступает в ПЗМ через канал В; рабо-

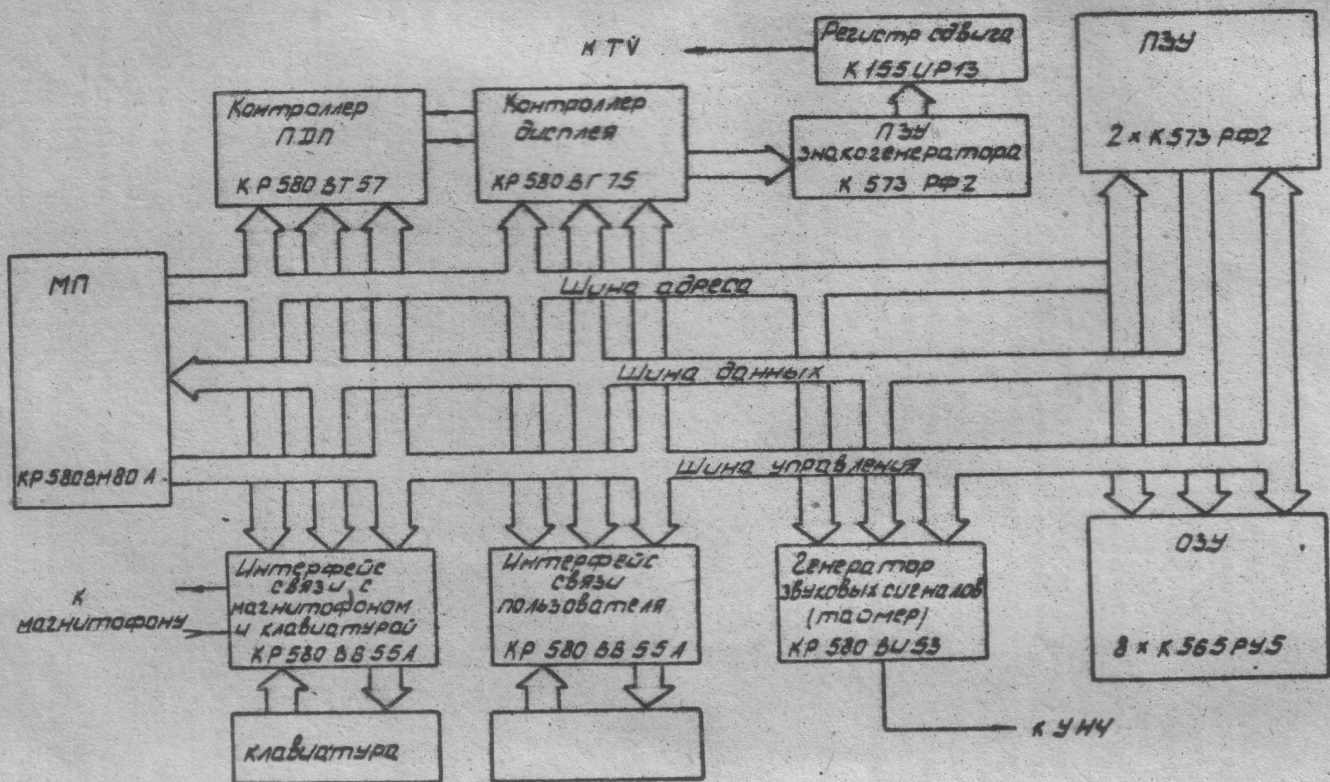


Рис. 3.2. Структурная схема ПЭВМ „Апогей - ВК01“

тающий на ввод. Канал С разделен на старший и младший подканалы. Старший подканал, работающий в режиме ввода, служит для приема информации с матричного и клавиш "РУС/ЛАТ", "УС" и "СС". Управление светодiodом "РУС" и вывод информации на магнитофон осуществляется через младшую половину канала С. Подключение ППА к компьютеру и его программирование подробно описаны в работе [1, 2]. Основным устройством вывода информации является дисплей (телевизионный приемник); на экране которого отображаются тексты программ и результаты вычислений. Формирование телевизионного сигнала осуществляется контроллером дисплея, который построен на БИС КР580ВГ75. Изображение на экране дисплея формируется засветкой отдельных точек телевизионного раstra. Графическое представление всех возможных символов хранится в специальном ПЗУ знакогенератора. Под каждым знакоместо на экране дисплея отводится матрица точек раstra размером 6x8. Адаптивно-цифровая информация отображается на экране 25-ю строками по 64 знакоместа в каждой; причем каждому знакоместу соответствует определенная ячейка экранного ОЗУ ПЗМ. Для вывода символа в заданное место на экране достаточно записать его код в соответствующую ячейку экранной области ОЗУ. Чтобы изображение постоянно присутствовало на экране дисплея, необходимо в течение развертки кадра телевизионного кадра периодически, синхронно с перемещением луча, вызвать последовательно все коды из экранной области на адресные входы ПЗУ знакогенератора. Эта функция выполняется БИС КР580ВГ75 совместно с контроллером прямого доступа к памяти (ПАП) КР580ВТ57. Для ускорения процесса пересылки кодов символов из экранного ОЗУ в контроллер дисплея используется режим ЛМП. Это означает, что коды символов из экранного ОЗУ передаются в контроллер дисплея без участия микропроцессора. Во время прямого доступа к памяти БИС КР580ВТ57 вместо МП формирует сигналы на шинах адреса и управления. Подробное описание работы БИС КР580ВГ75 и БИС КР580ВТ57 приведено в [3] и [4] соответственно. Для формирования звуковых сигналов используется БИС программируемого таймера КР580ВМ53. Эта БИС содержит три независимых канала, работающих в режиме делителей частоты. Путем программирования коэффициента деления можно получить на выходе канала заданной БИС сигнал звуковой частоты в диапазоне от 20 Гц до 20 кГц. Подробнее с режимами работы и программированием БИС КР580ВМ53 можно ознакомиться в [1, 5]. Для подключения различных внешних устройств предназначен дополнительный ППА. Информация о работе БИС КР580ВМ53, ее программировании и подключении внешних устройств дается в последнем разделе описания.

4. Программная модель ПЗМ

АСЕМБЛЕР ориентирован на определенный тип ЗМ, поэтому при изучении этого языка недостаточно знать общую структурную схему ПЗМ. Необходимо перейти к так называемой программной (регистровой) модели ПЗМ. Программная модель ПЗМ "Апогей-БКО1", построенная на основе 8-разрядного однокристального МП КР580ВМ80А, показана на рис. 4.1. Такая модель включает только те элементы ПЗМ, к которым можно обращаться программно на языке АСЕМБЛЕРА.

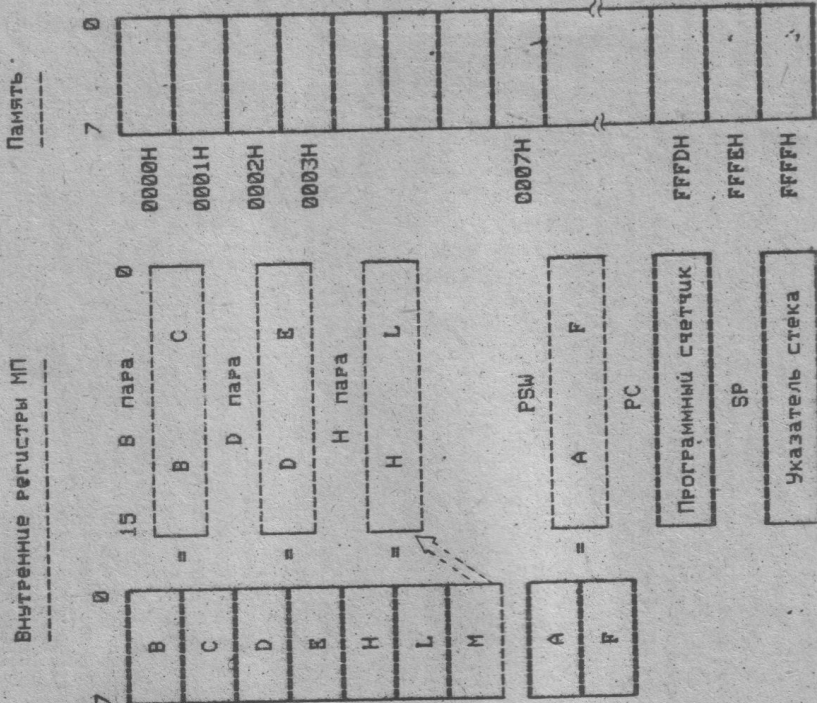


Рис. 4.1. Программная модель ПЗМ

Основным элементом программной модели является регистр, т.е. некоторое физическое устройство, способное воспринимать, хранить и выдавать информацию. Разрядность всех регистров равна или кратна длине слова микропроцессора (8 бит). Память

ЗВМ можно рассматривать как совокупность регистров, имеющих адреса. Функционирование ПЗВМ (выполнение программы) можно рассматривать как передачу данных между регистрами МП, памяти и устройств ввода-вывода.

Программная модель включает внутренние регистры микропроцессора; находящиеся внутри самого МП, и память. Регистры устройств ввода-вывода гарятся как ячейки памяти и отдельно на рис.4.1, не показаны.

В состав микропроцессора КР580 входят восемь 8-разрядных и два 16-разрядных внутренних регистров. 16-разрядные регистры служат для хранения адреса команды РС и указателя стека SP. В 8-разрядных регистрах могут временно храниться и обрабатываться данные, адреса и специальные признаки - флажки. Все регистры являются программно доступными; их содержимое можно модифицировать или использовать в качестве операндов в различных командах. Операции с регистрами занимают немного места в ОЗУ и выполняются намного быстрее операций с памятью. Поэтому, когда это возможно, использование регистров для временного хранения данных во всех случаях предпочтительнее хранения данных в оперативной памяти.

Кроме того, имеются программно недоступные регистры: 16-разрядный регистр адреса; 8-разрядный регистр команды и 8-разрядные регистры W, Z для временного хранения при выполнении команды.

Наиболее важным регистром микропроцессора является аккумулятор, обозначаемый А. Этот 8-разрядный регистр используется почти во всех арифметических и логических операциях. В операциях загрузки, записи в память и во многих других, аккумулятор в отличие от других регистров, может быть савинут, инвертирован, косвенно загружен или записан в память по адресу, также загружен прямо или записан в память по адресу, явно указанному в команде.

В арифметических и логических операциях один из двух операндов находится в аккумуляторе. В него же помещается и результат, стирая исходное содержимое аккумулятора. В операциях сравнения из А вычитается второй операнд и в соответствии с результатом устанавливаются флаги МП, но результат в аккумуляторе не записывается и его содержимое не меняется.

Аккумулятор и регистр F образуют 16-разрядный регистр, называемый словом состояния процессора и обозначаемый PSW. О нем подробнее будет сказано ниже.

Кроме аккумулятора имеется шесть 8-разрядных регистров, называемых регистрами общего назначения (РОН): В, С, D, Е, H, L. Каждый из этих регистров может обрабатываться независимо один от другого, но имеется возможность объединения (конкатенации) регистров В и С, D и Е, H и L в 16-разрядные регистры: пары, к которым можно обращаться по наименьшему старшему регистру, соответственно В, D и H, Пары ВС, DE и HL имеют

индивидуальные особенности, которые определяют при программировании выбор регистров для тех или иных целей.

Один из регистров микропроцессора, называемый косвенным регистром М (Memory - память), доступен при программировании, но на самом деле регистром не является. В действительности, при обращении к регистру М, берется содержимое ячейки памяти, адрес которой находится в регистровой паре HL. Использование регистра М позволяет в бинарных арифметических и логических операциях вторым операндом косвенно задавать содержимое ячейки памяти. Прямая адресация памяти в арифметических и логических операциях невозможна.

Регистровая пара HL существенно выделяется среди других пар. Кроме косвенной регистровой адресации через ссылку на регистр М, пара HL может быть прямо загружена из памяти или записана в память, при этом в команду явно указывается адрес ячейки ОЗУ в которую записывается регистр L (регистр H записывается в ячейку памяти, адрес которой на 1'ольше указанного). Содержимое регистровой пары HL может быть обменено с содержимым пары DE и с вершиной стека, ее содержимое можно переслать в указатель стека и в счетчик команды, можно сложить с содержимым пар регистров DE и BC, а также умножить, т.е. савинуть влево на один разряд.

Регистровая пара DE выделяется тем, что может быть обменена одной командой с парой регистров HL, поэтому ее удобно использовать для временного хранения содержимого пары HL, хранения второго адреса, передачи параметров подпрограммам.

Регистровую пару BC используют в основном для счетчиков и в качестве рабочих регистров для временного хранения данных. Команда состоит из двух частей: кода операции, которое определяет выполняемую операцию, и поля операнда - данные, участвующие в операции. В поле операнда могут быть данные, но чаще там находится адрес регистра (ячейки памяти), содержимого которого операнда. Команда минимальной длины занимает один байт, максимальной - 3 байта.

Шестнадцатеричный регистр PC (program counter) называется счетчиком команд (программным счетчиком) и используется для хранения текущего адреса программной памяти. При каждом обращении к программной памяти производится автоматическое увеличение содержимого PC на единицу.

Выполнение команд в ПЗВМ, как в любой вычислительной машине, состоит из следующих этапов: выборка команды; считывание операнда и собственно выполнение операции. В своем очередном из этапов включает в себя несколько более мелких шагов. Рассмотрим выполнение 3-байтной команды.

На первом шаге содержимое программного счетчика по ША передается в память и из памяти производится выборка первого байта команды - им всегда является код операции. По шине данных код операции передается в МП, загружается в регистр команд и хранится в нем в течение времени выполнения команды.

Одновременно с этим производится инкремент РС, который указывает на следующую ячейку памяти.

По коду операции микропроцессор определяет алину команды и осуществляет действия по выборке второго и третьего байтов команды. Для этого содержимое РС по ША передается в память, осуществляется считывание второго байта команды и перемещение по ША в младшие биты регистра адреса памяти (регистр Z).

Одновременно производится инкремент РС, который будет указывать на следующую ячейку программной памяти. Аналогично в старшие биты регистра адреса памяти (регистр W) считывается третий байт команды. Выборка команды завершена, а в РС сортирован адрес следующей команды.

Обычно команды выбираются из программной памяти последовательно от одной команды к другой, следующей за ней. Микропроцессор может изменить последовательность выборки команды, исполнив команду, которая записывает новое значение в РС. Таковыми командами являются команды перехода, вызова и возврата из подпрограмм, а также некоторые другие.

Шестнадцатиразрядный регистр SP - это указатель стека. Он служит для организации в любом месте ОЗУ участка памяти, называемого стеком. Стек - это особый образом организованная область оперативной памяти, выделяемая программистом для временного хранения информации. В стек, например, помещается адрес возврата при вызове подпрограмм, в стек временно хранятся данные из регистров, так как запись содержимого пары регистров и извлечение их из стека осуществляется однобайтовыми командами. В стек или из стека могут быть переданы только пары регистров. Запись в стек и извлечение из стека производится через SP.

Указатель стека SP (Stack Pointer) адресует последний занятую ячейку ("верхушку") стека. Это единственный адресный регистр, который обеспечивает автоуменьшение (уменьшение адреса до выполнения команды) и автоприращение (увеличение адреса после выполнения команды). Перед загрузкой каждого байта в стек производится декремент (уменьшение содержимого на 1), а после извлечения из стека каждого байта выполняется инкремент указателя стека. При записи в стек слово помещается в следующую свободную ячейку стека, а при считывании из стека - извлекается последнее поступившее в него слово. Данные записываемые в стек посланиями, при обращении к стеку извлекаются первыми. Такая организация хранения данных сокращенно обозначается Lifo (Last In, First Out - последний поступивший удаляется первым). До использования стека необходимо инициализировать указатель стека SP, т.е. загрузить в него максимальный адрес области ОЗУ, выделенной для стека. Доступ к SP производится через пару регистров HL.

Ранее уже упоминалось, что слово состояния процессора PSW (processor status word) содержит в себе аккумулятор (старший байт) и регистр флагов F (младший байт). В регистре F задей-

ствовано 5 флагов (разрядов), которые устанавливаются и сбрасываются в результате выполнения арифметических и логических операций (см. рис. 4.2):

S (sign) - флаг знака; совпадает со значением старшего бита результата. '0' в старшем бите соответствует положительному числу, '1' - отрицательному;

Z (zero) - флаг нуля; устанавливается при нулевом результате;

AC (auxiliary carry) - флаг вспомогательного переноса; устанавливается при возникновении переноса из третьего бита в четвертый, т.е. из младшей тетрады в старшую;

P (parity) - флаг паритета; устанавливается при наличии в результате четного числа единиц;

CY (carry) - флаг переноса; устанавливается при наличии переноса из старшего бита результата (в операциях вычитания флаг становится флагом заема).

7	6	5	4	3	2	1	0
S	Z	0	AC	0	P	1	CY

Рис. 4.2. Регистр флагов.

Флаги всегда устанавливаются и сбрасываются автоматически после выполнения очередной команды, влияющей на флаги, в зависимости от результата операции. При этом флаг считается установленным, если разряд флага принимает значение 1, и сброшенным, если значение разряда 0. Состояние флагов проверяется при выполнении команды условного перехода, при условном вызове и возврате из подпрограмм. Довольно трудно понять и запомнить, при каких операциях какие флаги устанавливаются и сбрасываются. Определить как команды действуют на флаги можно с помощью табл. 3 (приложение А).

5. РЕЖИМЫ АДРЕСАЦИИ

В поле операнда всех команд, каким-либо образом определяются данные, участвующие в операции, или операнды. Способ определения операнда называется режимом адресации.

В микропроцессоре КР580 используется четыре основных режима адресации: непосредственная, прямая, регистровая и косвенная (см. рис. 5.1.).

При непосредственной адресации необходимые данные находятся в самой команде. 8-разрядный операнд занимает второй байт <B2> команды. В КР580 опускается непосредственно адресовывать 16-разрядные слова, причем младшие разряды слова содержатся во втором байте <B2> команды, а старшие - в третьем байте <B3>.

Наиболее простой и неэкономичной является прямая адресация: в поле операнда содержится полный 16-битный адрес ячейки памяти. Младшим байтом адреса является <B2>, старшим - <B3>. С помощью прямой адресации можно обратиться к любой ячейке адресного пространства.

При регистровой адресации в команде указывается регистр (пара регистров), содержание которого является операндом. Адрес регистра занимает всего 3 бита, поэтому команды с регистровым режимом адресации являются однобайтовыми и выполняются достаточно быстро.

Косвенная адресация отличается от регистровой лишь тем, что в регистровой паре содержится не данные, а 16-разрядный адрес ячейки памяти, в которой находятся эти данные. Основным указателем памяти служит N-пара; адресуемая как R0N с адресом 110. В командах загрузки и запоминания аккумулятора с качеством указателей памяти могут выступать пары B и D.

Кроме описанных режимов адресации во многих 1-байтных командах микропроцессора КР580 применяется неявная адресация. Здесь операнд находится в определенном внутреннем регистре МП и его адресовывать не нужно. Для большинства команд с неявной адресацией таким регистром является аккумулятор A. Например, все команды сдвига МП КР580 оперируют с содержимым аккумулятора. Во всех бинарных (с двумя операндами) операциях преобразования данных подразумевается, что одним из операндов является содержимое аккумулятора и что результат операции загрузится в аккумулятор. При загрузке и запоминании данных в стеке, операнд содержится в указателе стека SP (стековая адресация).

Одним из недостатков микропроцессора КР580 является отсутствие индексной адресации, которую удобно применять для организации циклов. Заметим, что при необходимости индексную адресацию можно реализовать программно.

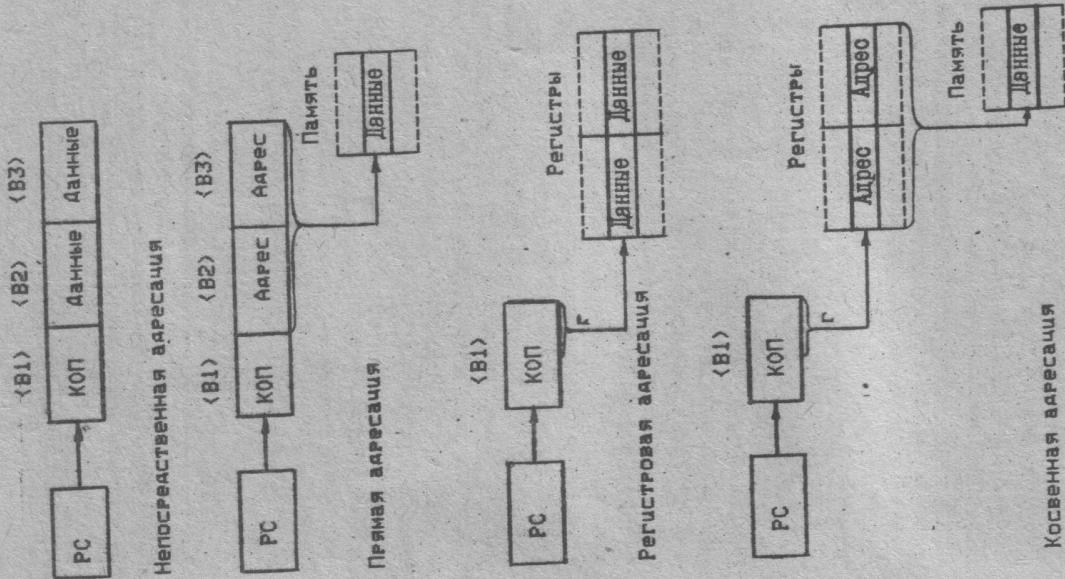


Рис: 5.1. Режимы адресации

6. Структура языка АССЕМБЛЕР

Язык АССЕМБЛЕР является машинно-зависимым и отражает аппаратные особенности микро-ЭВМ, для которой он используется. Близость к микро-ЭВМ проявляется тем, что в языке АССЕМБЛЕР используется mnemonic-символьное обозначение машинных команд, учитывается ее внутренняя структура и форма представления чисел.

Программа, написанная на АССЕМБЛЕРЕ состоит из последовательности команд, называемых также операторами (причем каждый оператор порождает одну машинную команду). Обычно оператор языка АССЕМБЛЕР занимает одну строку текста программы и содержит четыре поля:

Метка	Мнемоника	Операнды	Комментарий
-------	-----------	----------	-------------

В Ассемблере АССМ "МИКРОН" ни одно из перечисленных полей ассемблерной строки не является обязательным, но пустая строка отмечается как ошибочная, хотя и не влечет за собой никаких отрицательных последствий. При записи каждой команды необходимо придерживаться определенных правил.

Поле метки, как правило, определяет символическое наименование 16-разрядного адреса или численной константы. Если меткой помечается команда, то ей присваивается значение адреса первого байта команды. Это присвоение делается программой ассемблера при каждой трансляции текста программы. Длина метки ограничена шестью символами; первым из которых обязательно должна быть буква. За последним символом метки следует двоеточие. Метка не должна содержать пробелов и знаков пунктуации, не должна совпадать с наименованиями регистров и именами кодов операций.

Поскольку метки ассоциируются с адресами памяти, их нельзя аббревиовать. Аргументы слов или кодов символическое наименование может появиться в поле метки только 1 раз, но один и тот же оператор допускается помечать несколькими метками.

На число меток в программе ограничений нет, но надо иметь в виду, что все метки включаются ассемблером в таблицу меток, которая располагается в ОЗУ сразу же после текста программы. На каждую метку в таблице отводится 8 байт: 6 - на имя метки; 2 - на ее значение; поэтому большое число меток в программе потребует большей памяти при трансляции и, естественно, увеличит общее время трансляции программы за счет поиска меток в таблице. Поэтому следует вводить метки только для тех команд, которым необходимо передавать управление.

В поле именника содержится символическое описание выполняемой командой, которое заменяет числовое значение кода операции. Мнемоника команд, как правило, представляет собой аббревиатуру английского прилагательного и отражает основную функцию команд; например:

MOV (MOVE) - передать, переслать;
 ACI (Add with Carry Immediate) - сложение с переносом непосредственного;

LDA (Load Accumulator Direct) - прямая загрузка аккумулятора;

JNZ (Jump if Not Zero) - перейти, если не ноль.

Некоторые мнемоники явно определяют функцию команд; например, команда RCH, означает передачу содержимого регистра пары HL в программный счетчик PC. Поле мнемоники должно быть отделено от поля операндов хотя бы одним пробелом.

В поле операнда содержатся данные, с которыми следует выполнять операцию. В качестве операндов могут быть адреса памяти, внутренние регистры микропроцессора, числовые и символьные константы. Некоторые команды оперируют определенными внутренними регистрами, которые определяют коды операций; и поэтому поле операндов остается пустым. Например:

Метка	Мнемоника	Операнды	Комментарий
	SMA		! Инвертирование A
	RLC		! Сдвиг A влево

Регистры и регистровые пары задаются соответствующим символическим наименованием. Каждый регистр имеет двоичный адрес (табл. 6.1.).

Таблица 6.1.

Регистр	B	C	D	E	H	L	M	A
Адрес	0001001101010111001101110111							

Наименование регистровой пары совпадает с наименованием старшего регистра и определяется, что из них имеется в виду, можно только по коду операции.

Операндом в команде может быть односимвольная константа. В этом случае символ закладывается в апостроф, а при трансляции в команду вторым байтом команд помещается его шестнадцатеричный код. Например:

Метка	Мнемоника	Операнды	Комментарий
	MVI	A, 'T'	! Загрузка в A кода 'T'
			! Буквы T

Численные непосредственные данные и адреса в поле операндов можно записать в десятичной или шестнадцатеричной системе счисления. Это может быть также метка, которая должна быть определена в поле метки какого-либо оператора. Использование

ние неопределенных меток считается ошибкой.

Численные данные, адреса и метки в поле операнда могут быть аргументами простых арифметических выражений. Допускается два арифметических действия: сложение и вычитание. Результат выражения определяется ассемблером (программой) и подставляется в команду при трансляции. Например:

Метка	Мнемоника	Операнды	Комментарий
LXI	H, 116H-3BH	;	Загрузка числа 216
		;	в пару HL
JMP	START+5	;	переход безусловный
		;	по адресу на 5
		;	больше START

Поле комментария необходимо для лучшего понимания программы. Знаком начала комментария считается символ '!'. При трансляции комментарий полностью игнорируется, поэтому в нем можно помещать любой поясняющий текст. Если комментарий занимает всю ассемблерную строку, то в первой позиции строки должен быть символ '!'.
Число пробелов, разделяющих поля ассемблерной строки, зависит только от объема ОЗУ и привычек программиста.

7. Система команд

Совокупность команд, которые может выполнять микропроцессор, называется системой команд данного МП. Базовая система команд МП КР580 включает 78 простых команд. Однако многие базовые команды порождают несколько различных кодов операций, поэтому общее число команд составляет 244.

Все команды по их функциональному назначению можно разбить на следующие пять групп: группа команд перемещения данных; арифметические команды; логические команды; команды переходов и команд управления и работы со стеком.

Приведем обозначения сокращений, используемых в дальнейшем при описании команд.

Символ	Значение
A	Аккумулятор (регистр A)
addr	Значение адреса - 16 бит
data	8-разрядное число (данные)
data 16	16-разрядное число (данные)
r1, r2	Один из регистров A, B, C, D, E, H, L
DDD, SSS	Набор битов, представляющий один из регистров A, B, C, D, E, H, L (DDD - назначение, SSS - источник).
DDD	Регистр
(SSS)	
111	A
000	B
001	C
010	D
011	E
100	H
101	L
rr	Одна из пар регистров: B представляет пару BC; D представляет пару DE; H, представляет пару HL; SR - 16-разрядный указатель стека;
RR	Набор бит, представляющий одну из пар регистров B, D, H, SF

рандом; в приращки; задаваемый первым операндом.
Отметим: что ни одна из команд пересылки не изменяет содержания флагов микропроцессора.

MOV r1,r2 (Move register). Передача между регистрами.
(r1)←(r2). Содержимое регистра 2 передается в регистр 1.

01DDSSSS

MOV r1,M (Move from Memory). Передача из памяти.
(r)←((H)(L)). Содержимое памяти, адрес которой находится в паре HL, передается в регистр r.

01DD0110

MOV M,r (Move to Memory). Передача в память.
((H)(L))←(r). Содержимое регистра r передается в память, адрес которой находится в паре HL.

01110SSS

MVI r, data (Move immediate). Передача непосредственная.
(r)←(байт2). Содержимое байта 2 команды передается в регистр r.

00DD0110
данные

MVI M, data (Move to memory immediate). Передача в память непосредственная.
((H)(L))←(байт2). Содержимое байта 2 команды передается в память, адрес которой указан в паре HL.

00110110
данные

LXI rp, data 16 (Load register pair immediate). Загрузить непосредственно пару регистров.
(rh)←(байт3); (r)←(байт2). Байт 3 командой передается в старший регистр (rh) регистровой пары rp, а байт 2 - в младший регистр (r) той же пары.

RP Пара регистров

BC
DE
HL
SP

rh Первый (старший) регистр данной пары
r1 Второй (младший) регистр данной пары

PC 16-разрядный счетчик команд (PC и PC1) представляют соответственно в старших и в младших бит).

SP 16-разрядный указатель стека (SPH, SPL) представляют соответственно в старших и в младших бит)

() Содержимое ячейки памяти или регистра, имя которого или адрес заключены в скобки

← Передача

^ Логическое И
+ или ИСКЛЮЧАЮЩЕЕ
- ИЛИ

+ Сложение
- Вычитание

↔ Обмен
↔ Обратный код (инверсия)

NNN Абсолютное представление (000 - 111) номера повторного запуска от 0 до 7

7.1. Команды перемещения данных

Команды перемещения копируют данные из одного регистра в другой (пересылка данных); из ОЗУ - в регистр (загрузка); из регистра - в ячейку памяти (запоминание); а также пересылают данные, содержащиеся в команде в регистр или ячейку памяти. В командах перемещения данных всегда указывается два операнда: источник и приемник данных. Первые обозначаются символом S (source - источник); вторые - символом D (destination - место назначения). Аккумулятор и регистровая пара HL могут быть указаны неявно. Если в команде два операнда, то перемещение происходит из источника, задаваемого вторым опе-

00RP0001
мл. байт данных
ст. байт данных

LDA addr (Load accumulator direct). Прямая загрузка аккумулятора.

(A) ← ((байт 3) (байт 2)). Содержимое ячейки памяти, адрес которой определен байтами 2 и 3 команд, передается в аккумулятор (регистр A).

00111010
мл. байт адреса
ст. байт адреса

STA addr (Store accumulator direct). Прямое размещение содержимого аккумулятора.

((байт 3) (байт 2)) ← (A). Содержимое аккумулятора передается в память, адрес которой указан байтами 2 и 3 команд.

00110010
мл. байт адреса
ст. байт адреса

LHL addr (Load H and L direct). Загрузить H и L прямо.

((L) ← ((байт 3) (байт 2))) (H) ← ((байт 3) (байт 2) + 1). Содержимое ячейки памяти, адрес которой определяется байтами 2 и 3 команд, передается в регистр L. Содержимое следующего байта памяти передается в регистр H.

00101010

мл. байт адреса
ст. байт адреса

SHLD addr (Store H and L direct). Поместить H и L прямо. ((байт 3) (байт 2)) ← (L); ((байт 3) (байт 2) + 1) ← (H). Содержимое регистра L передается в ячейку памяти, адрес которой определяется байтами 2 и 3 команд. Содержимое регистра H передается в следующий байт памяти.

00100010
мл. байт памяти
ст. байт памяти

LDAH gp (Load accumulator indirect). Загрузить аккумулятор косвенно.

(A) ← ((gp)). Содержимое ячейки памяти, адрес которой определяется парой регистров gp, передается в регистр A. Могут использоваться только пары gp = B (регистры B и C) или gp = D (регистры D и E).

00RP1010

STAX gp (Store accumulator indirect). Загрузить аккумулятор косвенно.

((gp) ← (A). Содержимое аккумулятора A передается в ячейку памяти, адрес которой содержится в паре регистров gp. Могут использоваться только пары gp = B (регистры B и C) или gp = D (регистры D и E).

00RP0010

XCHG (Exchange H and L with D and E). Обмен H и L с D и E. (H) ↔ (D); (L) ↔ (E). Содержимое регистров H и L обменивается с содержимым регистров D и E.

11101011

7.2. Арифметические команды

Арифметические команды предназначены для выполнения операций сложения, сложения с переносом, вычитания, вычитания с заемом, инкрементирования, декрементирования, десятичной коррекции аккумулятора.

Эти команды оперируют с данными в памяти и регистрах и непосредственными данными. Положительные данные представляются собой целое двоичное число без знака с диапазоном изменения от 0 до 255. В операциях с отрицательными числами используется формат представления чисел со знаком. В этом формате старший бит (7) является знаковым S. Он указывает, является ли число положительным (+) или отрицательным (-). При 0 в знаковом бите число положительно; при 1 - отрицательно. Представление положительных чисел в этом формате такое же как и в беззнаковом формате, но наибольшим положительным числом является 127. Отрицательные числа представляются в так называемом дополнителном коде. Преобразование двоичного отрицательного числа в абсолютный код осуществляется следующим образом: сначала инвертируется каждый бит этого числа и затем к полученному результату прибавляется 1.

В арифметических операциях одним из неадресуемых operands является содержимое аккумулятора. Результат загружается в аккумулятор и в соответствии с полученным результатом производится модификация всех флагов микропроцессора, кроме указанных ниже исключений.

ADD r (Add register). Сложение содержимого регистра (A) ← (A) + (r). Содержимое регистра r складывается с содержимым аккумулятора.

10000555

ADD M (Add memory). Сложение данных памяти (A) ← (A) + ((H) (L)). Содержимое памяти, адрес которой указан в регистрах H и L, складывается с содержимым аккумулятора.

10000110

ADI data (Add immediate). Непосредственное сложение. (A) ← (A) + (байт 2). Содержимое байта 2 команды складывается с содержимым аккумулятора.

11000110

данные

Если результат сложения двух чисел больше 255, то возникает перенос из старшего разряда и устанавливается в единицу флаг переноса CY. Содержимое флага переноса может участвовать в операциях сложения в качестве третьего слагаемого.

ADC r (Add register with carry). Прибавление содержимого регистра r и переноса.

(A) ← (A) + (r) + (CY). Содержимое регистра r и флага переноса CY (бит переполнения) складывается с содержимым аккумулятора.

10001555

ADC M (Add memory with carry). Прибавление содержимого памяти и переноса.

(A) ← (A) + ((H) (L)) + (CY). Содержимое ячейки памяти, адресом которой является содержимое пары регистров HL, и флага переноса складывается с содержимым аккумулятора.

10001110

ACI data (Add immediate with carry). Непосредственное сложение с учетом переноса.

(A) ← (A) + (байт 2) + (CY). Содержимое 2 байта команды и флага переноса складывается с содержимым аккумулятора.

11001110

данные

Команды вычитания очень похожи на команды сложения: в аккумуляторе должно находиться уменьшаемое. Флаг переноса CY в операции вычитания устанавливается, если уменьшаемое меньше вычитаемого, т.е. он превращается в флаг заема.

SUB r (Subtract register). Вычитание содержимого регистра.

(A) ← (A) - (r). Содержимое регистра r вычитается из содержимого аккумулятора.

10010555

SUB M (Subtract memory). Вычитание содержимого памяти.
 $(A) \leftarrow (A) - ((H)(L))$. Содержимое ячейки памяти: адрес которой является содержимым регистровой пары HL, вычитается из аккумулятора.

10010110

SUI data (Subtract, immediate). Непосредственное вычитание.
 $(A) \leftarrow (A) - (\text{байт 2})$. Содержимое 2 байта команд вычитается из аккумулятора.

11010110

данные

SBB r (Subtract register with borrow). Вычитание содержимого регистра и заема.
 $(A) \leftarrow (A) - (r) - (CY)$. Содержимое регистра r и флага CY вычитается из содержимого аккумулятора.

10011555

SBB M (Subtract memory with). Вычитание содержимого памяти и заема.
 $(A) \leftarrow (A) - ((H)(L)) - (CY)$. Содержимое ячейки памяти, адресом которой является содержимое пары регистров HL, и флага переноса CY вычитается из аккумулятора.

10011110

SBI data (Subtract immediate with borrow). Непосредственное вычитание данных и заема.
 $(A) \leftarrow (A) - (\text{байт 2}) - (CY)$. Содержимое 2 байта команд и флага CY вычитаются из аккумулятора A.

11011110

данные

Наличие в системе команд микропроцессора команда сложения с переносом и команда вычитания с заемом позволяет достаточно просто реализовать программно сложение (вычитание) целых положительных чисел с любой точностью. Для этого каждое число

представляется 2, 3, 4 и более байтами. Примеры таких программ можно найти в литературе по программированию для микропроцессоров. Учтите, что сложение (вычитание) надо начинать с младших байтов, которые складываются без учета флага переноса. При сложении (вычитании) старших байтов содержимое флага CY учитывается. Операнды должны находиться в памяти.

Две специальные арифметические команды - инкремент (INR) и декремент (DCR) - позволяют соответственно увеличить или уменьшить на 1 содержимое любого 8-разрядного регистра или ячейки памяти. Команды INR и DCR устанавливают все флаги, за исключением флага переноса CY.

INR r (Increment register). Инкремент содержимого регистра r.
 $(r) \leftarrow (r) + 1$. Содержимое регистра r увеличивается на 1.

00000100

INR M (Increment memory). Инкремент содержимого памяти.
 $((H)(L)) \leftarrow ((H)(L)) + 1$. Содержимое ячейки памяти, адрес которой содержится в паре регистров HL, увеличивается на 1.

00110100

DCR r (Decrement register). Декремент содержимого регистра r.
 $(r) \leftarrow (r) - 1$. Содержимое регистра r уменьшается на 1.

00000101

DCR M (Decrement memory). Декремент содержимого памяти.
 $((H)(L)) \leftarrow ((H)(L)) - 1$. Содержимое памяти, адрес которой содержится в паре регистров HL, уменьшается на 1.

00110101

Для увеличения и уменьшения на 1 содержимого любой регистровой пары и указателя стека применяются команды INX и DCX. Необходимо запомнить, что командам INX и DCX не изменяют состояние флагов.

INX r (Increment register pair). Инкремент пары регистров r.
 $((rh)(rl)) \leftarrow ((rh)(rl)) + 1$. Содержимое пары регистров r увеличивается на 1. Не устанавливаются никакие флаги!

00RP0011

DCX gr (Decrement register pair): декремент пары регистров.
 $(rh)(rl) \leftarrow (rh)(rl) - 1$. Содержимое пары регистров gr уменьшается на 1. Не устанавливаются никакие флаги!

00RP1011

В микропроцессоре КР580 имеется несколько команд, выполняющих сложение 16-разрядных чисел. Аккумулятор в выполнении этих команд не участвует. Роль аккумулятора выполняет регистровая пара HL. Ее содержимое можно сложить с содержимым регистровых пар HL, DE, BC и с содержимым указателя стека SP. Результат такого сложения помещается в регистр HL.

DAD gr (Add register pair to H and L). Сложить содержимое пары регистров с содержимым пары HL.
 $(H)(L) \leftarrow (H)(L) + (rh)(rl)$. Содержимое пары регистров gr складывается с содержимым пары HL. Устанавливается только флаг CY. Он устанавливается в 1, если есть перенос при сложении с указанной точностью; если нет - сбрасывается.

00RP1101

Для несложной обработки больших массивов числовых данных удобно ввести формат абсолютного кодированных десятичных чисел или BCD чисел. В этом формате байт содержит две десятичные цифры (тетрады) в коде B421. Сложение BCD чисел выполняется в два этапа:

Операнды складываются как абсолютные числа!
 Производится коррекция в обмен случаев неправильного промежуточного результата.

DAA (Decimal adjust accumulator). Десятичная коррекция аккумулятора. 8-разрядное число в аккумуляторе преобразуется из шестнадцатичной системы в два 4-разрядных абсолютных десятичных числа по следующему правилу.

1. Если значение младшей тетрады аккумулятора больше 9 или установлен флажок ментетрадного переноса AC=1, то к содержимому аккумулятора прибавляется 6.

2. Если значение старшей тетрады аккумулятора больше 9 или установлен флаг переноса CY, то 6 прибавляется к значению старшей тетрады аккумулятора.

00100111

Следует подчеркнуть, что команда DAA не корректирует результат абсолютного вычитания.

7.3. Логические команды

Логические операции в отличие от арифметических являются поразрядными, т.е. выполняются независимо для каждого бита операнда. Независимый операнд находится в аккумуляторе. Результат операции тоже записывается в аккумулятор и по полученному результату модифицируются состояния всех флагов; кроме флага переноса. Флаг переноса CY никогда не может быть установлен в единицу и поэтому принудительно сбрасывается.

ANA r (AND register). И регистр.
 $(A) \leftarrow (A) \wedge (r)$. Содержимое регистра r подвержено операции логического И с содержимым аккумулятора.

10100555

ANA M (AND memory). И память.
 $(A) \leftarrow (A) \wedge (H)(L)$. Содержимое ячейки памяти, адрес которой находится в регистрах HL, подвержено логическому И с содержимым аккумулятора.

10100110

ANI data (AND immediate). Непосредственное И.
 $(A) \leftarrow (A) \wedge (\text{байт } 2)$. Содержимое 2 байта команды подвержено логическому И с содержимым регистра A.

11100110

данные

При выполнении операции логического И (команды) каждый бит аккумулятора устанавливается в 1 только в том случае, если оба бита операндов были 1.

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

XRA r (Exclusive OR register). ИЛИ ИСКЛЮЧАЮЩЕЕ РЕГИСТРА.
(A) $\rightarrow$ (A) $\vee$ (r). ИЛИ ИСКЛЮЧАЮЩЕЕ ВЫПОЛНЯЕТСЯ С СОДЕРЖИМЫМ РЕГИСТРА r И АККУМУЛЯТОРА.

101015SS

XRA M (Exclusive OR memory). ИЛИ ИСКЛЮЧАЮЩЕЕ ПАМЯТИ.
(A) $\rightarrow$ (A) $\vee$ (H)(L). ИЛИ ИСКЛЮЧАЮЩЕЕ ВЫПОЛНЯЕТСЯ С СОДЕРЖИМЫМ ЯЧЕЙКИ ПАМЯТИ, АДРЕС КОТОРОЙ УКАЗАН ПАРОЙ HL И СОДЕРЖИМЫМ АККУМУЛЯТОРА.

10101110

XRI data (Exclusive immediate). ИЛИ ИСКЛЮЧАЮЩЕЕ НЕПОСРЕДСТВЕННО.
(A) $\rightarrow$ (A) $\vee$ (байт 2). ИЛИ ИСКЛЮЧАЮЩЕЕ ВЫПОЛНЯЕТСЯ С СОДЕРЖИМЫМ БАЙТА 2 КОМАНДЫ И СОДЕРЖИМЫМ АККУМУЛЯТОРА.

11101110
данные

Команды ИСКЛЮЧАЮЩЕГО ИЛИ ПРОИЗВОДЯТ ПОРАЗЯНОЕ СЛОВАНИЕ ДВУХ ОПЕРАНДОВ ПО мод 2. Если значения соответствующих бит операндов совпадают, то значение бита результата будет 0, иначе - 1.

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

Данные команды позволяют выборочно инвертировать биты аккумулятора. Команда XRI 80H, например, будет формировать чередующуюся последовательность 0 и 1 в старшем разряде аккумулятора, если ее выполнять многократно, а команда XRA A очистить (обнулить) аккумулятор.

CMP r (Compare register). Сравнить регистр.
(A) - (r). Содержимое регистра r вычитается из содержимого аккумулятора. Содержимое аккумулятора не изменяется. Флаги устанавливаются, как при вычитании. Флаг нуля Z устанавливается в 1, если есть единицы; если (A) = (r). CY устанавливается в 1, если (A) < (r).

10115SS

Данные команды применяются для выделения, обнуления или проверки значения определенных бит аккумулятора с помощью маски, которая является вторым байтом команды. Например, команда ANI 0FH выделяет 4 младших бита аккумулятора, обнуляя 4 старших бита.

ORA r (OR register). ИЛИ регистр.
(A) $\rightarrow$ (A) $\vee$ (r). ИЛИ ВЫПОЛНЯЕТСЯ С СОДЕРЖИМЫМ РЕГИСТРА r И АККУМУЛЯТОРА.

10110SSS

ORA M (OR memory). ИЛИ память.
(A) $\rightarrow$ (A) $\vee$ (H)(L). ИЛИ ВЫПОЛНЯЕТСЯ С СОДЕРЖИМЫМ ЯЧЕЙКИ ПАМЯТИ, АДРЕС КОТОРОЙ СОДЕРЖИТСЯ В РЕГИСТРОВОЙ ПАРЕ HL И СОДЕРЖИМЫМ АККУМУЛЯТОРА.

10110110

ORI data (OR immediate). ИЛИ непосредственно.
(A) $\rightarrow$ (A) $\vee$ (байт 2). ИЛИ ВЫПОЛНЯЕТСЯ С СОДЕРЖИМЫМ БАЙТОМ КОМАНДЫ И АККУМУЛЯТОРА.

11110110
данные

При выполнении операции логического ИЛИ (аппозиция) бит аккумулятора будет нулевым только в том случае, если оба бита операндов были нулевыми.

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

Эти команды применяются для того, чтобы установить в единицу определенные разряды аккумулятора или чтобы скопировать содержимое аккумулятора из нескольких полей. Например, команда ORI 55H устанавливает в единицу 0, 2, 4 и 6 биты аккумулятора, не изменяя значения других битов. Команда ORA A не изменяет содержимого аккумулятора, но по его содержимому модифицирует флаги нуля Z, знака S и паритета P.

СМР И (Compare memory). Сравнить память.
 $(A) \leftarrow (M)(L)$. Содержимое ячейки памяти; адрес которой является содержимым пары HL; вычитается из содержимого аккумулятора; которое не изменяется.

10111110

СРІ data (Compare immediate). Непосредственное сравнение.
 $(A) \leftarrow (байт\ 2)$. Содержимое 2 байта команды вычитается из аккумулятора. Содержимое аккумулятора не изменяется.

11111110
 данные

При выполнении команда сравнения флаги устанавливаются: как при вычитании. Флаг нуля 2 устанавливается в 1, если сравниваемые значения равны, и в 0, если они не равны. Флаг переноса CY устанавливается в 1, если содержимое аккумулятора (A) меньше (I) сравниваемого значения. Команды сравнения более чашбны, чем команды вычитания (SBB, SBI, SUB или SUI); т. е. при сравнении аккумулятор сохраняется для последующих операций.

СМА (Complement accumulator). Инвертировать аккумулятор.
 $(A) \leftarrow (\bar{A})$. Содержимое аккумулятора инвертируется (все бит 0 становятся 1, все бит 1 становятся 0). Никакие флаги не устанавливаются.

00101111

Команда СМА использует, например, для образования дополнительного кода путем выполнения последовательности из двух команд: СМА, INR A.

СМС (Complement carry). Инвертировать перенос CY.
 $(CY) \leftarrow (\bar{CY})$. Инвертируется содержимое флага переноса CY. Никакие прочие флаги не устанавливаются.

00111111

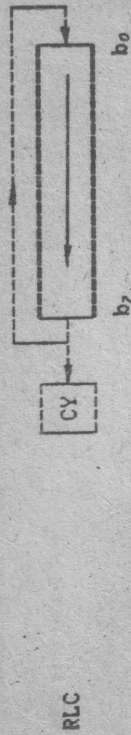
СТС (Set carry). Установить перенос.
 $(CY) \leftarrow 1$. Во флаге переноса CY устанавливается 1. Никакие

другие флаги не устанавливаются.

00110111

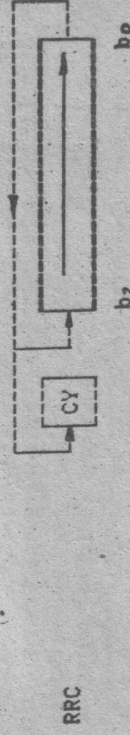
К логическим командам относятся также четыре команды сдвига аккумулятора на один разряд влево или вправо. Различают циклический сдвиг аккумулятора и сдвиг аккумулятора через перенос. Все команды сдвига устанавливают только флаг переноса CY.

RLC (Rotate left in carry). Сдвиг влево в перенос.
 $(b_{n+1}) \leftarrow (b_n); (b_0) \leftarrow (b_7); (CY) \leftarrow (b_7)$. Содержимое аккумулятора сдвигается на одну позицию влево. Младший бит b_0 и CY принимают значения вытесненного бита, т.е. старшего бита b_7 .



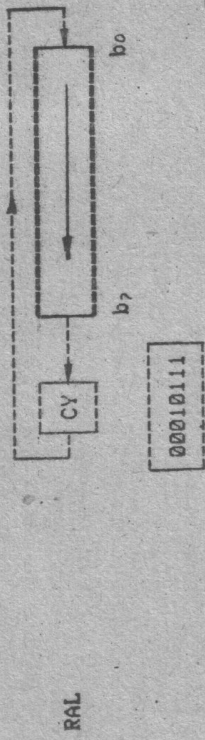
11111110

RRC (Rotate right in carry). Сдвиг вправо в перенос.
 $(b_n) \leftarrow (b_{n+1}); (b_7) \leftarrow (b_0); (CY) \leftarrow (b_0)$. Содержимое аккумулятора сдвигается на одну позицию вправо. Старший бит b_7 и CY принимают значения вытесненного бита, т.е. бывшего младшего бита b_0 .

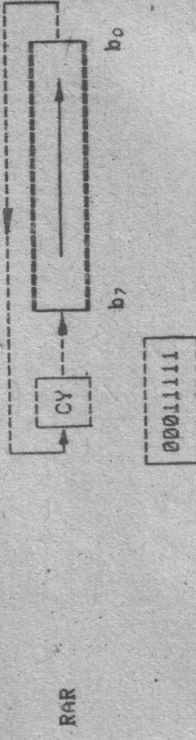


00001111

RAL (Rotate left through carry). Циклический сдвиг влево через перенос.
 $(b_{n+1}) \leftarrow (b_n); (CY) \leftarrow (b_7); (b_0) \leftarrow (CY)$. Содержимое аккумулятора сдвигается влево циклически вместе с CY. В младшем бите b_0 устанавливается содержимое CY, а в CY - вытесненное содержимое старшего бита b_7 .



RAR (Rotate right through carry). Циклический сдвиг вправо через перенос.
 $(b_n) \rightarrow (b_{n+1})$; $(CY) \rightarrow (b_0)$; $(b_7) \rightarrow (CY)$. Содержимое аккумулятора сдвигается вправо циклически вместе с CY. В старшем бите b_7 устанавливается содержимое CY, а в CY - вытесненное содержимое бывшего младшего бита b_0 .



7.4. Команды переходов

Эта группа команд позволяет изменить последовательность выполнения команд. Сюда входят команды безусловного и условного переходов; команды вызова подпрограммы и возврата из них и команды повторного запуска (рестарта).

Команды безусловного перехода (передачи управления) всегда изменяют порядок выполнения команд. Команды условного перехода, называемые также командами ветвления, осуществляют передачу управления только при удовлетворении некоторого условия (CCC); заанного в коде операции. Если условие не удовлетворяется, то выполняется сдвиг по порядку команд. Таким условием является проверка содержимого любого из четырех флагов микропроцессора: CY, Z, S и P, указанных в коде операции. Сами команды перехода на флаги не влияют. Содержимое флагов, проверяемое в командах перехода, устанавливается при выполнении предшествующих арифметических и логических команд.

CCC	Условие
000	NZ - не ноль (Z=0).
001	Z - ноль (Z=1).
011	NC - нет переноса (CY=0).

Результат не нулевой.
 Результат нулевой.

- 011 C - есть перенос (CY=1).
 - 100 PO - нечетность (P=0).
 - 101 PE - четность (P=1).
 - 110 P - плюс (S=0).
 - 111 M - минус (S=1).
- (Parity odd).
 (Parity even).
 Результат положительный.
 Результат отрицательный.

JMP addr (Jump). Переход.
 $(PC) \rightarrow (байт\ 3)(байт\ 2)$. Управление передается команде, адрес которой установлен в байтах 3 и 2 текущей команды.

1100011
мл. байт адреса
ст. байт адреса

J condition addr (Conditional jump). Условное ветвление.
 Если (CCC), то $(PC) \rightarrow (байт\ 3)(байт\ 2)$. Если установленное условие выполняется, то управление передается команде, адрес которой установлен в байтах 3 и 2 текущей команды; если нет - управление продолжается последовательно.

1100010
мл. байт адреса
ст. байт адреса

Всего команд ветвления восемь: для флага Z - JZ и JNZ
 S - JM и JP
 P - JPE и JPO
 CY - JC и JNC

Подпрограммы представляют собой некоторую последовательность команд, часто используемых при выполнении какой-либо программы. При выполнении вызова подпрограммы необходимо иметь все меры для возврата в точку вызова после выполнения подпрограммы. В программировании для МП подпрограммы применяются очень часто и в системе команд микропроцессора имеются

(CCC), то (PC) $\leftarrow$ ((SP)); (PC) $\leftarrow$ ((SP) + 1); (SP) $\leftarrow$ (SP) + 2. Если данное условие удовлетворено, то выполняются действия, определенные командой возврата RET; если нет - управление следует последовательно.

11CCC000

В командах условных вызовов и возвратов проверяется состояние указанного в коде операции флага. Имеются команды, проверяющие единичное и нулевое значения каакого из четырех флагов:

- флаг Z: вызовы CZ, CNZ; возвраты RZ, RNZ;
- флаг CY: вызовы CC, CNC; возвраты RC, RNC;
- флаг S: вызовы CS, CNS; возвраты RS, RNS;
- флаг P: вызовы CP, CM; возвраты RP, RM.

В системе комана микропроцессора имеется особая 1-байтная команда вызова, предназначенная для обработки прерываний и введения контрольных точек (разрывов) при отлаке программы.

RET (Restart). Повторный запуск (рестарт).

((SP) - 1) $\leftarrow$ (PCh); ((SP) - 2) $\leftarrow$ (PC1); (SP) $\leftarrow$ (SP) - 2; (PC) $\leftarrow$ 8 X (NNN). Текущее содержимое программного счетчика запоминается в стеке: старший байт - в ячейке по адресу на 1 меньше содержимого SP, младший байт - в ячейке, адрес которой меньше на 2 единицы регистра SP. Содержимое SP аккрементуется на 2. Управление передается команде, адресом которой является 8 раз содержимое NNN (8X NNN).

11NNN111

Счетчик комана PC после команды рестарт содержит:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	N	N	N	0	0	0

где NNN - уровень запроса на прерывание.

Следовательно, в зависимости от значений NNN микропроцессор переходит к одной из восьми ячеек, находящихся в первых адресах памяти: 0000, 0008, 0010, ..., 0038.

Особенной командой безусловной передачи управления без возврата является команда RCHL; в результате выполнения кото-

большая группа комана, упрощающая вызов и возврат из подпрограмм.

CALL addr (Call). Вызов.

((SP) - 1) $\leftarrow$ (PCh); ((SP) - 2) $\leftarrow$ (PC1); (SP) $\leftarrow$ (SP) - 2; (PC) $\leftarrow$ (байт 3)(байт 2). Содержимое счетчика комана PC запоминается в стеке (старшие 8 бит адреса следующей команы - по адресу на 1 меньше содержимого регистра SP; младшие 8 бит - по адресу на 2 меньше SP). Содержимое регистра SP уменьшается на 2. В PC заносится адрес первой команы программы, указанный в байтах 3 и 2 текущей команы.

11001101
мл. байт адреса
ст. байт адреса

С condition addr (Conditional call). Условный вызов.

Если (CCC), то ((SP) - 1) $\leftarrow$ (PCh); ((SP) - 2) $\leftarrow$ (PC1); (SP) $\leftarrow$ (SP) - 2; (PC) $\leftarrow$ (байт 3)(байт 2). Если заданные условия выполняются, то выполняются действия, определенные в команде CALL; если нет - управление продолжается последовательно.

11CCC100
мл. байт адреса
ст. байт адреса

RET (Return). Возврат.

((PC1) $\leftarrow$ ((SP)); (PCh) $\leftarrow$ ((SP) + 1); (SP) $\leftarrow$ (SP) + 2. Содержимое ячейки памяти, адрес которой определен в указателе стека SP, передается в младший байт регистра PC. Содержимое ячейки памяти, адрес которой на 1 больше содержимого SP, передается в старший байт счетчика комана PC. Содержимое регистра, SP увеличивается на 2.

11001001

R condition (Conditional return). Возврат условный. Если

адресным памяти адрес, которой больше на 1 содержимого указателя стека. Значение SP при этом не изменяется.

11100011

SPHL (Move HL to SP). Передать HL в SP.
(SP) ← (H)(L). Содержимое регистров H и L (16 бит) передается в регистр SP.

11111001

IN port (Input). Ввести данные.
(A) ← (данные). Данные, помещенные на 8-разрядную двунаправленную шину определенным портом, передаются в регистр A.

11011011
ПОРТ

OUT port (Output). Вывести данные.
(данные) → (A). Содержимое регистра A помещается на 8-разрядную двунаправленную шину данных для передачи в определенный порт.

11010011
ПОРТ

Ранее уже упоминалось, что в персональном компьютере "Апогей-БК01" адреса внешних устройств (I/O) размещены с ячейками памяти и находятся в едином адресном пространстве. Поэтому все команды с обращением к памяти, содержащие адреса I/O, превращаются в команды ввода - вывода: команды загрузки осуществляют ввод, а команды запоминания - вывод.

BI (Enable interrupt). Разрешение прерываний.
Эта команда устанавливает внутренний триггер разрешения прерываний в состояние "1" (INTB=1) и используется в нашем компьютере для перехода в режим работы с расширенной псевдографикой (поаклируется знакогенератор, коды символов которого представлены в табл. 2 приложения А).

11111011

DI (Disable interrupt). Запрет прерывания.
Эта команда сбрасывает внутренний триггер прерывания в исходное состояние (INTB=0) и возвращает работу в основной алгоритм - цифровой режим (коды символов этого знакогенератора приведены в табл. 1 приложения А).

11110011

HLT (Halt). Останов. Команда вызывает прекращение выполнения программы и переводит микропроцессор в состояние останова.

01110110

NOP (No op). Нет операции. Не выполняются никакие действия, кроме инкрементирования содержимого программного счетчика PC.

Краткое изложение системы команд приведено в табл. 3 Приложения А. Колонка 1 содержит именовку команд, колонка 2 описывает действие команд, число байт, необходимых для размещения данной команды в памяти; указано в колонке 3; колонка 4 показывает на какие флаги команда оказывает влияние в зависимости от результата операции (*); колонка "КОЛИЧЕСТВО ТАКТОВ" приводит необходимые длительности выполнения команд. Зная количество тактов и частоту генератора синхронизации 1,78 МГц, можно точно определить время выполнения команды или любого сегмента программы.

Коды операций команд МП КР580ВМ80 в шестнадцатеричной записи приведены в табл. 4 Приложения А. Например, коду 27H соответствует команда десятичной коррекции DAA, а команда SBB M имеет код 9BH.

7.6. Псевдокоманды

Для удобства пользователя в АСSEMBLЕРЕ существуют директивы (псевдокоманды), которые являются указаниями программисту о выполнении определенных действий в процессе ассемблирования. Директивы определяют порядок ассемблирования, размещают в памяти информацию, присваивают численные значения символическим наименованиям, резервируют память и выполняют другие функции.

Директивы поочередно стандартному формату операторов АСSEMBЛЕРА, но содержимое их полей имеет некоторые особенности. Набор директив определяется используемой программой ассемблера, в данном случае АССМ "МИКРОН".

Директива ORG (начало)

Определяет ячейку памяти, в которую будет загружаться первый байт следующей команды или байт данных. По умолчанию в самом начале программы подразумевается наличие директивы ORG с нулевым операндом.

Метка Мнемоника Операнды Комментарий
ORG 100H

Задает адрес начала области памяти, в которой будет работать программа, но после трансляции.

Директива END (конец)

Информировать программу - ассемблер о окончании физического конца программы. Директива END может быть только одна и находится в последней строке программы.

Метка Мнемоника Операнды Комментарий
END

Метка в директивах ORG и END является обязательным элементом.

Директива EQU (присвоить)

При выполнении этой директивы программа - ассемблер присваивает значение выражения символическому наименованию, находящемуся в поле метки.

Метка Мнемоника Операнды Комментарий
<имя> EQU <выражение> ; Формат директивы EQU

Заметим, что каждое символическое наименование может появиться в поле метки только одной директивы EQU. При программировании следует сгруппировать все директивы EQU в начале или конце программы. Приведем пример:

Метка Мнемоника Операнды Комментарий
BUF : EQU 0E1D0H ; Значение BUF равно
; E1D0H (начало экранного буфера)
SCR : EQU 0E2C2H ; SCR присваивается значение E2C2H (начало видной части экрана).

Директива DB (определить байт)

Применяется для инициализации данных и резервирования памяти.

Метка Мнемоника Операнды Комментарий
(метка:) DB <список> ; Формат директивы DB

Операнда директивы DB может быть последовательностью выражений, разделенных запятыми и имеющих 8-битные значения либо цепочкой символов, заключенных в апострофы. При выполнении директивы DB значения выражений и код символов запоминаются в смежных ячейках памяти, начинающихся после ячейки предыдущей команды. Приведем пример:

Метка Мнемоника Операнды Комментарий
DATA: DB 5,7,9,13 ; Запоминаются четыре значения

SIMV: DB 'PRIVET' ; Запоминается шесть символов

Директива DW (определить слово - 2 байта)

Имеет такой же формат, как и директива DB, но в поле операнда содержится список выражений, имеющих 16-битные значения. Например:

Метка Мнемоника Операнды Комментарий
ADDR : DW 05A6H ; (ADDR)=A6H,
; (ADDR+1)=05H

Директива DS (определить память)

Метка Мнемоника Операнды Комментарий
(метка:) DS <выражение> ; Формат директивы DS

Вычисленное значение выражения из поля операнда определяет число ячеек (байт) памяти, резервируемых для запоминания данных. Адрес следующего оператора равен сумме адреса оператора, находящегося перед директивой DS и значения выражения директивы DS. Примеры использования директивы DS:

Метка Мнемоника Операнды Комментарий
HAND: DS 16 ; Резервируется 16 ячеек
RESERV: DS 64 ; Резервируется 64 ячейки

8. ОПИСАНИЕ ПАКЕТА ПРОГРАММ "МИКРОН"

Предлагаемый в программном обеспечении ПЗВИ "Аполгей-БКО1" пакет программ "МИКРОН" состоит из Редактора текстов (ED) и транслирующей программы Ассемблера (ASM).

Редактор ED предназначен для редактирования любого текста непосредственно на экране дисплея; исправления ошибок, удаление и ввод новых символов и целых фрагментов текста.

Транслирующая программа Ассемблер ASM переводит мнемоническое наименование команд языка АССЕМБЛЕР в машинные коды. Текст программы под управлением Редактора ED с клавиатуры или магнитной ленты заносится в начало текстового буфера ОЗУ компьютера, адрес которого равен - 2000H. Затем с помощью ASM эту программу транслируют и полученные в результате трансляции машинные коды распределяются в специальной области трансляции с адреса 1100H. Областью трансляции можно воспользоваться для отладки оттранслированной программы или перенести ее директивный Т МИНИТОРА для этой цели в другую область.

Работа с пакетом программ начинается с запуска Редактора директивной Б МИНИТОРА в результате чего на экране появляется надпись "ED *МИКРОН*". (при первом запуске появляется сообщение об ошибке, не пугайтесь, Редактор ED сам восстановит нормальный режим работы). Задание адреса поочередным нажатием на клавишу "AP2" и клавишу какой-либо буквы латинского алфавита или символа (далее поочередное нажатие двух клавиш будем именовать директивной и обозначать в виде алгебраической суммы) например, AP2+H). По этой директиве очистится экран и появится запрос Редактора NEW?. Если ответить Y, то это приведет к очистке текстового буфера в памяти компьютера и переводу Редактора в режим ввода строки (в начале первой строки появится псевдоадресный символ "4"); в противном случае установится режим редактирования (его отличительный признак - символ "#").

Режим ввода использует для ввода текста с клавиатуры компьютер, причем длина строки ограничена 63 символами. За восемь позиций до конца строки генерируется звуковой сигнал, предупреждающий о том, что для продолжения ввода надо будет перейти на новую строку нажатием клавиши "BK" (точно так же Редактор отреагирует на попытку ввода недопустимых при каких-либо ситуациях директив). Набор строки завершают нажатием на клавишу "BK", в результате чего эта строка пересылается в текстовый буфер. Допущенную при вводе ошибку легко исправить, сместив курсор назад до нужного места. Устранив ошибку, его можно вернуть в любую позицию в строке. После нажатия на "BK" в текстовой буфер заносится символ под которым находится курсор и все символы, находящиеся влево от него. Для перехода из режима ввода строки в режим редактирования достаточно нажать на клавишу "STR" (если "STR" нажимается до нажатия на

"BK", то набранная строка не попадет в текстовый буфер).

Режим редактирования позволяет оперативно просмотреть введенный текст, построчно или фрагментами (по 24 строки). Очередной фрагмент текста выводит на экран директивой AP2+H. Для удобства восприятия текста в начале каждого нового фрагмента всегда отображаются две строки предыдущего. Директива AP2+H служит для просмотра фрагментов в обратном порядке. К началу или концу фрагмента можно вернуться, дав Редактору соответствующую директиву AP2+B или AP2+E. Для перемещения курсора к началу первой строки текущего фрагмента пользуются клавишей "1", а внутри него клавишами "A", "E", "4", "H". При попытке сместить курсор за пределы экрана вверх или вниз текст сдвигается на одну строку. Смещение курсора за его нижнюю границу приводит к автоматическому переходу Редактора в режим ввода строки, т.е. для продолжения ввода достаточно нажать на клавиши "AP2+E" и "4".

Чтобы найти какой-либо фрагмент текста совсем необязательно просматривать его весь: на этот случай в Редакторе предусмотрена возможность автоматического поиска заданной группы символов. Для этого после задания директив AP2+L вводят группу символов, которые необходимо отыскать, и нажимают на клавишу "BK" - на экране появится фрагмент текста, начиная со строки, в которой они впервые встретились. Воспользовавшись директивой AP2+R, можно найти и все последующие фрагменты с заданными символами. Если в тексте заданной группы символов нет, то прозвучит звуковой сигнал и восстановится режим редактирования. Если вы забыли сообщить Редактору какие символы необходимо искать и нажмете клавишу "BK" - на экране восстановится последняя из отображавшихся страниц текста.

Наиболее часто при редактировании возникает необходимость вставить в строку один или несколько новых символов, например, записать их взамен старых или ввести дополнительно. Для этого в Редакторе предусмотрен режим автоматической разбивки символов в строке. Включают авторазбивку директивой AP2+F4, а выключают - AP2+F2. Отдельные символы удаляют из строки установкой курсора под соответствующими знаками и нажатием на клавишу "F2", а освобождают для пропущенного - клавишей "F4". При вводе дополнительных символов необходимо поминуть об ограничении на их количество в строке.

В режиме редактирования в текст можно вставлять отдельные псевдоадресные символы, нажав предварительно на клавишу "AP2", "...". В режиме ввода строки этого делать не нужно - любые символы можно вводить непосредственно.

Чтобы вставить в текст одну или несколько новых строк, к началу следующей за ним строки подают курсор и сообщают Редактору о своих намерениях директивой AP2+A. Если же надо вставить перед первой строкой текста, то вначале нажимают на клавишу "1", а уж затем вводят директиву AP2+A. В результате весь текст, следующий за помеченной строкой, сдвигается с эк-

экран протокола трансляции, представляющего собой строки исходного текста программы; первая строчка в шестнадцатичной системе выводится коды ошибок; адреса размещения команд и данных в ОЗУ компьютера и машинные коды транслируемой программы;

2. - после трансляции программы на дисплее отображается перечень встретившихся в ней меток в алфавитном порядке (для латинского алфавита) и их шестнадцатичные адреса;

3. - исходный текст программы транслируется в машинные коды; и выводится сообщение о результатах трансляции; число ошибок, встретившихся в тексте, и два шестнадцатичных числа. Первое из них - адрес конца оттранслированной программы в той области, где она должна работать; второе, ограниченное символами '/' - в области трансляции. Это связано с тем, что в результате трансляции машинные коды программы всегда располагаются в области трансляции независимо от адресов, в которых они должны работать. Поэтому, если начальный адрес транслируемой программы отличен от адреса начала области трансляции (1100H), то первая отладка или запуском необходимо ее поместить в рабочую область, определяемую псевдокомандой ORG в начале программы.

Отметим, что выполнение директив 1 и 2 автоматически сопровождается выполнением директивы 3.

При переименовании программы с помощью директивы МОНИТОРА следует помнить, что первый адрес области трансляции известен и всегда равен 1100H. Поэтому директива МОНИТОРА для переименования машинных кодов программы в рабочую область должна выглядеть следующим образом:

T1100; (адрес конца программы в области трансляции); (адрес, заданный псевдокомандой ORG).

Обычно, если нет каких-либо специальных особенностей, в начале текста программы псевдокоманда ORG 1100H можно не ставить. В этом случае рабочая область оттранслированной программы совпадает с областью трансляции. После отладки в одной области программы можно перетранслировать для использования в любой другой, изменив лишь начальный адрес, задаваемый псевдокомандой ORG.

Трансляцию программы можно прервать нажатием на клавиши 'УС' и 'С'. Выход из транслятора ASSM в МОНИТОР можно, нажав на клавиши 'УС' и 'Е' (для этого можно использовать также клавиши '0' или любую другую) с кодом символа меньше 31H). Управление МОНИТОРУ переключается по адресу 0F875H, что позволяет обойти процедуру очистки экрана.

При нажатии на клавишу 'СТР' управление передается Реактору. Если при трансляции объем ОЗУ компьютера окажется недостаточным, то она прекратится и на экране появится сообщение "МАЛО ПАМЯТИ".

Во время трансляции ASSM анализирует также синтаксис исходного текста программы и при обнаружении ошибок выводит их-

рана (но не из текстового буфера) и Редактор переходит в режим ввода строк. Выход из этого режима можно нажатием на клавишу 'СТР'.

Для удаления фрагмента текста курсор помещают в начало его первой строки и нажимают на клавиши 'AP2', 'D'. Затем манипулируя клавишами '↑', '↓' или 'AP2' и '↑', перемещают курсор до строки, перед которой заканчивается удаляемый фрагмент текста (помеченная строка будет сохранена), и вновь вводят директиву AP2+D. Если данный фрагмент решено оставить, необходимо нажать на клавишу 'СТР'.

Имеющийся в памяти компьютера текст можно записать на магнитофон; воспользовавшись директивой AP2+0. В ответ на нее Редактор запрашивает имя текста; под которым он будет записан на магнитную ленту. Указав имя (впрочем, этого можно и не делать) и включив магнитофон в режим записи, нажимают 'BK'.

Для приема текста с магнитной ленты в память ПЗМ вводят директиву AP2+I, а затем, в ответ на запрос Редактора - имя нужного текста. После этого вклмкнут магнитофон на воспроизведение и нажимают на клавишу 'BK'. Если имя не указать, то с магнитной ленты в текстовый буфер будет введен первый встретившийся на ленте текст. По окончании ввода на экране отображается начальный фрагмент текста.

Редактор может самостоятельно сравнить записанный на ленту текст с имеющимся в памяти компьютера. Для этого надо нажать на клавиши 'AP2', 'У' и ввести текст с магнитной ленты. Если тексты не идентичны, то на экране появится сообщение "ОШИБКА", а если полностью совпадут - их начальный фрагмент.

Редактор позволяет компоновать текст из нескольких фрагментов, которые в этом случае вводят директивой AP2+M. Любую директиву работы с магнитофоном можно отменить, нажав на клавишу 'СТР'.

Введенную программу можно сразу же оттранслировать, дав управление программе Ассемблер директивой 'СТР', в ответ на которую на экране появится надпись 'ASSM #ИМЯПРОГ' (из режима ввода она должна быть отлаана дважды; редактирование - один раз). При необходимости из Редактора можно перейти в МОНИТОР, нажав клавишу 'УС' и удерживая ее на 'Е' (далее эту ситуацию будем записывать 'УС' и 'Е').

Требования, предъявляемые к программам при использовании транслятора 'ASSM #ИМЯПРОГ' с языка АССЕМБЛЕР, рассмотрены в описании структуры языка АССЕМБЛЕР (см. раздел 6 данного описания). Для удобства ввода программы в память компьютера пользуйтесь клавишей 'TAB' (табуляция). При нажатии на нее курсор смещается в ближайшую правую позицию; номер которой кратен восьми, что позволяет правильно форматировать текст программы.

Трансляция начинается после нажатия на одну из следующих клавиш:

1. - программа транслируется с одновременным вводом на

формации о них в виде соответствующих кодов вместе с протоколом трансляции. Приведем перечень обнаруживаемых ошибок:

- 01 - двойное определение метки!
- 02 - метка не была определена ранее!
- 04 - использована несуществующая именная команда!
- 06 - неправильно определен операнд!
- 10 - в имени метки применен недопустимый символ.

Если в одной строке обнаружено несколько ошибок, выводится результирующий код, равный сумме кодов ошибок.

Директивы Редактора ED и Ассемблера ASSM приваждены соответственно в табл. 8.1 и 8.2.

Для размещения пакета программы "МИКРОН" необходимо 4 Кбайт памяти; причем и Редактор ED, и Ассемблер ASSM могут работать независимо один от другого. Первый расположен в памяти с адресом 0000H по 07FFH, второй - с 0800H по 0FFFH. Служебная область ограничена ячейками 1000H и 10FFH; стек располагается в ОЗУ с адреса 0E00FFH. Текст, подлежащий загрузке с помощью Редактора располагается в буфере с адреса 2000H.

Пакет программы "МИКРОН" реализован на базе материала, опубликованного в журнале "Радио" 7 за 1987 г., с небольшими изменениями, связанными с отличительными особенностями архитектуры ПЗМ "Апогей - БК01":

- 1) адрес начала текстового буфера - 2000H;
- 2) стек располагается в ОЗУ с адреса 00FFH;
- 3) выход в МОНИТОР по адресу F875H;
- 4) звуковой сигнал подается 4 раза вместо 25.

Таблица 8.1

ДИРЕКТИВА	ВЫПОЛНЯЕМЫЕ ДЕЙСТВИЯ
BO	Запуск Редактора по директиве МОНИТОРА НАЧАЛО РАБОТЫ При ответе "Y" на запрос "HEM?" : очистка экрана, очистка текстового буфера и установка режима ввода строки
CTP	Выход в Ассемблер
UC и E	Выход в МОНИТОР
AP2+0	РАБОТА С МАГНИТОФОНОМ Вывод текста на магнитофон
AP2+1	Ввод текста с магнитофона
AP2+V	Сравнение текста, хранимого в ОЗУ с введенным с МЛ
AP2+H	Ввод дополнительного фрагмента текста к уже имеющемуся в ОЗУ
CTP	Отмена любой директивы работы с магнитофоном
BK	РЕЖИМ ВВОДА СТРОК Ввод в память набранной строки
AP2+E	Перемещение курсора для исправления ошибки
AP2+A	Переход к концу текста
AP2+H	Ввод новой строки
AP2+I	Просмотр фрагментов текста
AP2+J	Просмотр фрагментов текста в обратном порядке
→ J ←	Перемещение курсора к началу первой строки
→ J ←	Перемещение курсора к месту редактирования
AP2+H, BK	ПОИСК ГРУППЫ СИМВОЛОВ Ввод группы символов Отображение фрагмента текста, в котором впервые встретилась заданная группа символов
AP2+R	Отображение следующего фрагмента текста с заданной группой символов

9. М О Н И Т О Р

Продолжение таблицы 8.1

ИСПРАВЛЕНИЕ ОШИБОК	
AP2+F4	Включение автораздвижки символов
AP2+F2	Выключение автораздвижки символов
F2	Удаление символа
F4	Освобождение места для пропущенного символа
AP2+V	Ввод одного псевдографического символа
УДАЛЕНИЕ ФРАГМЕНТА ТЕКСТА	
↑, ↓, AP2+↓	Перемещение курсора к началу первой строки удаляемого фрагмента текста
AP2+D	Маркировка первой строки удаляемого фрагмента текста
↑, ↓, AP2+↑	Перемещение курсора к началу строки, предшествующей последней удаляемой строке
AP2+D	Удаление фрагмента текста
СТР	Отмена любой директивы удаления фрагмента текста

В ПЗУ объемом 4 Кбайта записана управляющая программа МОНИТОР, инициализирующая все программируемые БИС и обеспечивающая работу клавиатуры, дисплея и интерфейса с кассетным магнитофоном. Кроме того МОНИТОР поддерживает диалог с пользователем, который вводит с клавиатуры определенные директивы и на экране дисплея получает сообщения о результатах их выполнения. (Синтаксис задания директив и их описание дано в руководстве по эксплуатации).

В состав МОНИТОРА входит стандартный набор подпрограмм ввода - вывода, к которым программа пользователя может обращаться; собираемая соглашения об обмене информацией. Это обеспечивает совместимость программного обеспечения аналогичных компьютеров при условии, что в программах собираются все правила вызова подпрограмм.

При работе МОНИТОР использует ячейки ОЗУ с E100H по E1CFH для хранения служебной информации. Рабочие ячейки МОНИТОРА и их назначения представлены в табл. 9.1.4.

Таблица 8.2.

ДИРЕКТИВА	ВЫПОЛНЯЕМЫЕ ДЕЙСТВИЯ
ВВ00	Запуск Ассемблера по директиве МОНИТОРА
1	Трансляция программы с отображением на экране протокола трансляции
2	Трансляция программы с отображением на экране таблицы меток
3	Трансляция программы с отображением на экране количества ошибок и адресов оттранслированной программы
СТР	Выход в Редактор
УС+E	Выход в МОНИТОР

Примечание. Выполнение директив 1 и 2 автоматически сопровождается выполнением директивы 3.

9.1. СТАНДАРТНЫЕ ПОДПРОГРАММЫ МОНИТОРА ПЗВМ "АПОГЕЙ-БКО1"

МОНИТОР ПЗВМ "АПОГЕЙ-БКО1" состоит из двух частей. Первая часть МОНИТОРА расположена по адресам F800H-F8FFH, и полностью совместима с МОНИТОРОМ ПЗВМ "РАДИО-86РК" по входным точкам и используемым для передачи параметров регистрам. Стандартные подпрограммы первой части МОНИТОРА приведены в таблице 9.1.1.

Таблица 9.1.1.

Адрес	Назначение
F803H	Ввод символа с клавиатуры с ожиданием ввода. Входные параметры: нет. Выходные параметры: А - код введенного символа.
F806H	Чтение байта с магнитофона. Входные параметры: А - признак поиска синхробайта; А=0FFH - чтение с поиском синхробайта; А=0BH - чтение без поиска. Выходные параметры: А - считанный байт.
F809H	Вывод символа на экран. Входные параметры: С - код выводимого символа. Выходные параметры: нет.
F80CH	Запись байта на магнитофон. Входные параметры: С - выводимый байт. Выходные параметры: нет.
F80FH	Не используется.
F812H	Опрос состояния клавиатуры. Входные параметры: нет. Выходные параметры: А - статус клавиатуры; А=0FFH - ни одна клавиша не нажата; А=0FFH - нажата одна из клавиш.
F815H	Вывод байта на экран в шестнадцатеричном виде. Входные параметры: А - код выводимого байта. Выходные параметры: нет.
F818H	Вывод сообщения на экран. Входные параметры: HL - адрес начала сообщения. Выходные параметры: нет.

Продолжение таблицы 9.1.1.

F81BH	Опрос клавиатуры без ожидания. Входные параметры: нет. Выходные параметры: А=0FFH - ни одна клавиша не нажата; иначе А - код нажатой клавиши. Код клавиши "РУС/ЛАТ" = 0FEN.
F81EH	Запрос положения курсора. Входные параметры: нет. Выходные параметры: H - вертикальная координата; L - горизонтальная координата.
F821H	Запрос байта из ячейки курсора. Входные параметры: нет. Выходные параметры: А - код символа из ячейки экранного ОЗУ, в которой установлен курсор.
F824H	Чтение блока с магнитофона. Входные параметры: HL - смещение. Выходные параметры: HL - начальный адрес загрузки; DE - конечный адрес загрузки; BC - контрольная сумма блока; считанная с ленты.
F827H	Запись блока на магнитофон. Входные параметры: HL - начальный адрес блока; DE - конечный адрес блока; BC - контрольная сумма блока. Выходные параметры: нет.
F82AH	Расчет контрольной суммы области памяти. Входные параметры: HL - начальный адрес области; DE - конечный адрес области. Выходные параметры: BC - контрольная сумма.
F82DH	Запуск отображения на экране. Входные параметры: нет. Выходные параметры: нет.
F830H	Запрос верхней границы доступной памяти. Входные параметры: нет. Выходные параметры: HL - адрес установленной верхней границы доступной памяти.
F833H	Установка верхней границы доступной памяти. Входные параметры: HL - адрес верхней границы доступной памяти; устанавливаемый пользователем. Выходные параметры: нет.

Вторая часть МОНИТОРА, расположенная по адресам F000H-F7FFF, содержит дополнительные адрессы, не реализованные в ПЗВМ "Радио-86РК": А; В; С; D; E; F; G; H; I; J; K; L; M; N; O; P; Q; R; S; T; U; V; W; X; Y; Z; AA; AB; AC; AD; AE; AF; AG; AH; AI; AJ; AK; AL; AM; AN; AO; AP; AQ; AR; AS; AT; AU; AV; AW; AX; AY; AZ; BA; BB; BC; BD; BE; BF; BG; BH; BI; BJ; BK; BL; BM; BN; BO; BP; BQ; BR; BS; BT; BU; BV; BW; BX; BY; BZ; CA; CB; CC; CD; CE; CF; CG; CH; CI; CJ; CK; CL; CM; CN; CO; CP; CQ; CR; CS; CT; CU; CV; CW; CX; CY; CZ; DA; DB; DC; DD; DE; DF; DG; DH; DI; DJ; DK; DL; DM; DN; DO; DP; DQ; DR; DS; DT; DU; DV; DW; DX; DY; DZ; EA; EB; EC; ED; EE; EF; EG; EH; EI; EJ; EK; EL; EM; EN; EO; EP; EQ; ER; ES; ET; EU; EV; EW; EX; EY; EZ; FA; FB; FC; FD; FE; FF; FG; FH; FI; FJ; FK; FL; FM; FN; FO; FP; FQ; FR; FS; FT; FU; FV; FW; FX; FY; FZ; GA; GB; GC; GD; GE; GF; GG; GH; GI; GJ; GK; GL; GM; GN; GO; GP; GQ; GR; GS; GT; GU; GV; GW; GX; GY; GZ; HA; HB; HC; HD; HE; HF; HG; HH; HI; HJ; HK; HL; HM; HN; HO; HP; HQ; HR; HS; HT; HU; HV; HW; HX; HY; HZ; IA; IB; IC; ID; IE; IF; IG; IH; II; IJ; IK; IL; IM; IN; IO; IP; IQ; IR; IS; IT; IU; IV; IW; IX; IY; IZ; JA; JB; JC; JD; JE; JF; JG; JH; JI; JJ; JK; JL; JM; JN; JO; JP; JQ; JR; JS; JT; JU; JV; JW; JX; JY; JZ; KA; KB; KC; KD; KE; KF; KG; KH; KI; KJ; KK; KL; KM; KN; KO; KP; KQ; KR; KS; KT; KU; KV; KW; KX; KY; KZ; LA; LB; LC; LD; LE; LF; LG; LH; LI; LJ; LK; LL; LM; LN; LO; LP; LQ; LR; LS; LT; LU; LV; LW; LX; LY; LZ; MA; MB; MC; MD; ME; MF; MG; MH; MI; MJ; MK; ML; MM; MN; MO; MP; MQ; MR; MS; MT; MU; MV; MW; MX; MY; MZ; NA; NB; NC; ND; NE; NF; NG; NH; NI; NJ; NK; NL; NM; NN; NO; NP; NQ; NR; NS; NT; NU; NV; NW; NX; NY; NZ; OA; OB; OC; OD; OE; OF; OG; OH; OI; OJ; OK; OL; OM; ON; OO; OP; OQ; OR; OS; OT; OU; OV; OW; OX; OY; OZ; PA; PB; PC; PD; PE; PF; PG; PH; PI; PJ; PK; PL; PM; PN; PO; PP; PQ; PR; PS; PT; PU; PV; PW; PX; PY; PZ; QA; QB; QC; QD; QE; QF; QG; QH; QI; QJ; QK; QL; QM; QN; QO; QP; QQ; QR; QS; QT; QU; QV; QW; QX; QY; QZ; RA; RB; RC; RD; RE; RF; RG; RH; RI; RJ; RK; RL; RM; RN; RO; RP; RQ; RR; RS; RT; RU; RV; RW; RX; RY; RZ; SA; SB; SC; SD; SE; SF; SG; SH; SI; SJ; SK; SL; SM; SN; SO; SP; SQ; SR; SS; ST; SU; SV; SW; SX; SY; SZ; TA; TB; TC; TD; TE; TF; TG; TH; TI; TJ; TK; TL; TM; TN; TO; TP; TQ; TR; TS; TT; TU; TV; TW; TX; TY; TZ; UA; UB; UC; UD; UE; UF; UG; UH; UI; UJ; UK; UL; UM; UN; UO; UP; UQ; UR; US; UT; UY; UZ; VA; VB; VC; VD; VE; VF; VG; VH; VI; VJ; VK; VL; VM; VN; VO; VP; VQ; VR; VS; VT; VU; VV; VW; VX; VY; VZ; WA; WB; WC; WD; WE; WF; WG; WH; WI; WJ; WK; WL; WM; WN; WO; WP; WQ; WR; WS; WT; WU; WV; WW; WX; WY; WZ; XA; XB; XC; XD; XE; XF; XG; XH; XI; XJ; XK; XL; XM; XN; XO; XP; XQ; XR; XS; XT; XU; XV; XW; XX; XY; XZ; YA; YB; YC; YD; YE; YF; YG; YH; YI; YJ; YK; YL; YM; YN; YO; YP; YQ; YR; YS; YT; YU; YV; YW; YX; YY; YZ; ZA; ZB; ZC; ZD; ZE; ZF; ZG; ZH; ZI; ZJ; ZK; ZL; ZM; ZN; ZO; ZP; ZQ; ZR; ZS; ZT; ZU; ZV; ZW; ZX; ZY; ZZ.

Таблица 9.1.2.

Адрес	Назначение
F003H	Запись байта в формате MSX на магнитофон. Входные параметры: С - выводимый байт. Выходные параметры: нет.
F006H	Чтение байта в формате MSX с магнитофона. Входные параметры: нет. Выходные параметры: А - считанный байт; признак С=1 - ошибка чтения.
F009H	Запись блока в формате MSX на магнитофон. Входные параметры: HL - начальный адрес блока; DE - конечный адрес блока. Выходные параметры: нет.
F00CH	Чтение блока в формате MSX с магнитофона. Входные параметры: HL - начальный адрес блока; DE - конечный адрес блока; Выходные параметры: признак С=1 - ошибка чтения.
F00FH	Запись длинного маркера MSX. Входные параметры: нет. Выходные параметры: нет.
F012H	Запись короткого маркера MSX. Входные параметры: нет. Выходные параметры: нет.
F015H	Чтение длинного маркера. Входные параметры: нет. Выходные параметры: нет.

Если в основной программе возникла необходимость провести какие - либо действия с экраном дисплея (отобразить символ, переместить курсор, стереть символ и т.п.) надо обратиться к подпрограмме вывода символа на экран (по адресу F009H). Для отображения алфавитно - цифрового или псевдографического символа предварительно в регистр С заносится соответствующий

кода символа (см. табл.1 и 2 Приложения А). Если в регистр С записан управляющий код дисплея из табл. 9.1.3, то это вызовет выполнение какой - либо функции, связанной с управлением перемещения курсора. Если перед вызовом подпрограммы вывода символа на экран записать в регистр С микропроцессора код 07H, то компьютер выдаст звуковой сигнал.

Таблица 9.1.3.

Функция	Коды	Ввод с клавиатуры
Курсор влево	00H	"←" или "УС" + "H"
Курсор вправо	10H	"→" или "УС" + "X"
Курсор вверх	10H	"↑" или "УС" + "Y"
Курсор вниз	10H	"↓" или "УС" + "Z"
Возврат каретки	00H	"BK" или "УС" + "M"
Перевод строки	00H	"PS" или "УС" + "J"
Очистка экрана	1FH	"СТР" или "УС" + "3E"
Курсор в начало экрана	00H	"←" или "УС" + "L"
Прямая адресация курсора	10H, 59H, 120H (N строки), 120H (N позиции), 120H (N строки) + 120H (N позиции)	"УС" + "I", "УС" + "AP2", "У", ..., "У"

Чтобы установить курсор в требуемую позицию на экране, необходимо выдать на дисплей последовательность кодов: 10H + 59H + (номер строки + 20H) + (номер позиции + 20H). Строки и позиции на экране отсчитываются от 0, причем нулевой строкой экрана считается самая верхняя строка, а нулевой позицией - самая левая.

Рабочие ячейки программы МОНИТОР
ПЗМ "Апогей - БК01"

Таблица 9.1.4.

Адрес	Назначение ячейки
E100	Адрес ячейки памяти экранного ОЗУ в которой установлен курсор
E101	
E102	Номер позиции курсора в строке (нумерация слева)
E103	Номер строки курсора (нумерация сверху)
E104	Признак обработки ESC/Y последовательности
E105	Признак разрешения опроса клавиатуры
E106	Признак активного регистра (0 - ЛАТ, не 0 - РУС)
E109	Код символа, принятого с клавиатуры
E108	Признак первого опроса и автоповтора
E10D	Хранение содержимого указателя стека SP во время работы подпрограммы чтения байта с МЛ
E114	Хранение содержимого программного счетчика PC
E115	Хранение адресов 'X' и 'G'
E116	Хранение содержимого регистровой пары HL
E117	Хранение адресов 'X' и 'G'
E118	Хранение содержимого регистровой пары BC
E119	Хранение адресов 'X' и 'G'
E11A	Хранение содержимого регистровой пары DE
E11B	Хранение адресов 'X' и 'G'
E11C	Хранение содержимого указателя стека SP
E11D	Хранение адресов 'X' и 'G'
E11E	Хранение регистров A и F для работы
E11F	Хранение адресов 'X' и 'G'
E120	Хранение байта при чтении в формате MSX
E121	Константа чтения для формата MSX

Продолжение таблицы 9.1.4.

Адрес	Назначение ячейки
E122	Хранение константы при чтении или записи файла в формате Микрош
E123	Хранение заданного адреса останова для выполнения директивы 'G'
E124	
E125	Хранение байта из ячейки памяти по адресу, где должен произойти останов
E126	Код операции JMP для выполнения директивы 'G'
E127	Первый параметр директивы
E128	
E129	Второй параметр директивы
E12A	
E12B	Третий параметр директивы
E12C	
E12D	Признак ввода более одного параметра в директиве
E12E	Признак прямого/инверсного чтения с магнитофона
E12F	Константа для чтения формата РК
E130	Константа для записи формата РК
E131	Адрес верхней границы ОЗУ
E132	
E133	Область хранения командной строки МОНИТОРА
E151	
E152	Область хранения вводного с клавиатуры имени файла для формата MSX
E158	
E159	Область хранения считанного с магнитофона имени файла для формата MSX
E15F	

Возможны три режима (0, 1 и 2) работы каналов ППА:

- Режим 0 - простой ввод - вывод информации;
- Режим 1 - стробированный ввод - вывод;
- Режим 2 - режим асинхронной шины.

Остановившись на наиболее используемом режиме работы ППА, режим 0 предусматривает обмен данными с ВУ через каналы А, В и АВА поканала С без управляющих сигналов о готовности к работе и подтверждения обмена. Это означает, что данные, выходящие из микро - ЭВМ по командам запоминания, записываются в соответствующие порты (ячейки памяти) и хранятся там до записи в этот порт новых данных. По линиям каналов они поступают во внешние устройства.

При вводе из внешних устройств данные не фиксируются в портах; а считываются непосредственно в аккумулятор микропроцессора из соответствующего порта (ячейки памяти) при выполнении команды загрузки (ввода).

Следует отметить, что любой из восьми разрядов порта С может быть установлен в 0 или 1 при помощи специального управляющего слова манипуляции битами (УСМБ), которое записывается в РУС. Формат УСМБ представлен в табл. 10.3.

Таблица 10.3.

0 D6 D5 D4 D3 D2 D1 D0	1	0
Установить разряд в '1'		
Установить разряд в '0'		
Код		
Устанавливаемый разряд порта С		
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1
Значение разрядов D4-D6 произвольно		
Признак установки разряда порта С		

В зависимости от кода, записанного в разрядах D1-D3, выбирается разряд порта С, значение которого будет изменено при записи УСМБ в регистр управляющего слова. Если необходимо изменить состояние нескольких разрядов порта С, то загружают последовательно несколько управляющих слов УСМБ.

Подробнее с режимами работы 1 и 2 БИС КР580ВВ55 можно ознакомиться в [1,2].

При подключении внешних устройств к ПЗВМ через разъем интерфейса пользователя необходимо руководствоваться следующими: к контакту A14 разъем, связанному с источником питания +5В, можно подключать нагрузку, потребляющую не более 100 мА; нагрузочная способность входа - выхода микросхемы КР580ВВ55А - один вход TTL; выхода A15 INTE устанавливается по команде EI и снимается по DI.

Схема подключения внешних устройств к ПЗВМ "Апогей - БК01" через разъем интерфейса пользователя приведена на рис. 10.2.

Для обмена с внешними запоминающими устройствами используется описанная в руководстве по эксплуатации устройства R и дополнительная директива U.

Директива V записывает содержимое указанной области памяти ПЗВМ во внешнее запоминающее устройство, подключаемое к интерфейсу пользователя. Данные передаются по каналу А интервальной БИС КР580ВВ55А: младший байт адреса по каналу В, старший байт - по каналу С. Признаком записи во внешнее ЗУ является наличие сигнала высокого уровня в старшем разряде канала С. Каждый передаваемый байт информации стробится сигналом INTE, снимаемым с центрального процессора и выведенным на разъем интерфейса пользователя.

Формат директивы V

У ADR1, ADR2, ADR3 (BK),

где ADR1 - начальный адрес передаваемой области ОЗУ;

ADR2 - конечный адрес передаваемой области ОЗУ;

ADR3 - начальный адрес внешнего ЗУ.

На примере рассмотрим программу ввода данных из модуля ПЗУ, подключаемого к нашему компьютеру через разъем интерфейса пользователя, в оперативную память компьютера (адреса в R). Программа представлена на рис 10.1.

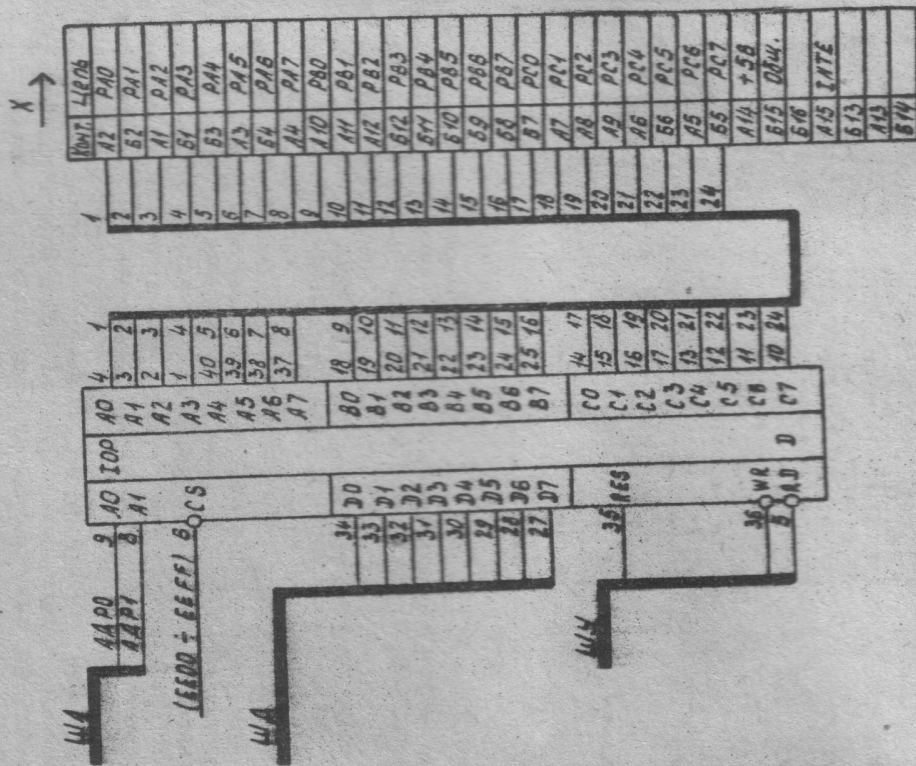
***** Программа пересылки информации *****
 *** из внешнего модуля ППЗУ в оперативную память ПЗВМ ***

Метка	Мнемоника	Операнды	Комментарий
PORT:	EQU	0B800H	Интерфейс пользователя
MONIT:	EQU	0F875H	Выход в МОНИТОР
	LXI	H, PAR1	Загрузить в (H)(L) адрес начала области пересылки ППЗУ
	LXI	D, PAR2	Загрузить в (D)(E) адрес конца области ППЗУ
	LXI	B, PAR3	Загрузить в (B)(C) адрес начала области загрузки ОЗУ
DIRR:	MVI	A, 90H	Настройка портов БИС
	STA	PORT + 3	Записать в регистр управления слово 90H
DIRR0:	SHLD	PORT + 1	Выставить текущий адрес ППЗУ
	LDA	PORT	Прочитать байт из ППЗУ
	STAX	B	Записать считанный байт в ОЗУ
	INX	B	Подготовить следующий адрес ОЗУ
	CALL	COMP	Сравнить (H)(L) и (D)(E)
	JZ	MONIT	Если равны, то выход в МОНИТОР (0F875H)
	INX	H	Иначе - подготовить следующий адрес ППЗУ
	JMP	DIRR0	Продолжить пересылку

*** Подпрограмма сравнения (H)(L) и (D)(E) ***

COMP:	Операнды	Комментарий
MOV	A, H	Сравнить (H) и (D)
CMP	D	Не равны - возврат
RNZ	A, L	Сравнить (L) и (E)
MOV	E	Возврат
CMP	E	
RET		

Рис. 10.1



Э - КР580ВВ55А

Х - ГРПМШ-1-31Ш02

Рис. 10.2

Таблица 2

	0	1	2	3	4	5	6	7
0								
1								
2								
3								
4								
5								
6								
7								
8								
9								
A								
B								
C								
D								
E								
F								

Таблица 3

МНЕМНИКА	ДЕЙСТВИЕ	ДЛИНА КОМАНДЫ	ИЗМЕНЯЕМЫЕ ФЛАГИ				КОЛ-ВО ТАКТОВ
			S	Z	AC	IP	
1	2	3	4	5			
Команды перемещения данных							
MOV r1, r2	(r1) ← (r2)	1				5	
MOV r, M	(r) ← ((H) (L))	1				7	
MOV M, r	((H) (L)) ← (r)	1				7	
MVI r, data	(r) ← (byte 2)	2				7	
MVI M, data	((H) (L)) ← (byte 2)	2				10	
LXI rp, data16	(rp) ← (byte 3); (rh) ← (byte 2)	3				10	
LDA addr	(A) ← ((byte3) (byte2))	3				13	
STA addr	((byte3) (byte2)) ← (A)	3				13	
LHLD addr	(L) ← ((byte3) (byte2)) (H) ← ((byte3) (byte2)+1)	3				16	
SHLD addr	((byte3) (byte2)) ← (H) (byte3) (byte2)+1 ← (L)	3				16	
LDAX rp	(A) ← ((rp))	1				7	
STAX rp	((rp)) ← (A)	1				7	
XCHG	(H) ↔ (D); (L) ↔ (E)	1				4	
Арифметические команды							
ADD r	(A) ← (A) + (r)	1				4	
ADD M	(A) ← (A) + ((H) (L))	1				7	
ADI data	(A) ← (A) + (byte 2)	2				7	
ADC r	(A) ← (A) + (r) + (CY)	1				7	
ADC M	(A) ← (A) + ((H) (L)) + (CY)	1				7	
ACI data	(A) ← (A) + (byte 2) + (CY)	2				7	
SUB r	(A) ← (A) - (r)	1				5	
SUB M	(A) ← (A) - ((H) (L))	1				7	
SUI data	(A) ← (A) - (byte 2)	2				7	
SBB r	(A) ← (A) - (r) - (CY)	1				4	
SBB M	(A) ← (A) - ((H) (L)) - (CY)	1				7	
SBI data	(A) ← (A) - (byte 2) - (CY)	2				7	
INR r	(r) ← (r) + 1	1				4	
INR M	((H) (L)) ← ((H) (L)) + 1	1				10	
DCR r	(r) ← (r) - 1	1				4	
DCR M	((H) (L)) ← ((H) (L)) - 1	1				10	
INX rp	(rh) (r1) ← (rh) (r1) + 1	1				6	
DCX rp	(rh) (r1) ← (rh) (r1) - 1	1				6	
DAD rp	(H) (L) ← (H) (L) + (rh) (r1)	1				10	
DAA	Если нл. 4 бита (A) > 9 или (AC)=1, то (A) ← (A) + 6 Если ст. 4 бита (A) > 9 или (CY)=1, то к ним +6.	1				4	

Продолжение таб. 3

1	2	3	4	5
ЛОГИЧЕСКИЕ КОМАНДЫ				
ANA r	$(A) \wedge (r)$	1	*	4
ANA M	$(A) \wedge (H) (L)$	1	*	7
ANI data	$(A) \wedge (\text{byte } 2)$	2	*	7
XRA r	$(A) \vee (r)$	1	*	4
XRA M	$(A) \vee (H) (L)$	1	*	7
XRI data	$(A) \vee (\text{byte } 2)$	2	*	7
ORA r	$(A) \vee (r)$	1	*	4
ORA M	$(A) \vee (H) (L)$	1	*	7
ORI data	$(A) \vee (\text{byte } 2)$	2	*	7
CMP r	$(A) - (r)$	1	*	4
CMP M	$(A) - (H) (L)$	1	*	7
CPI data	$(A) - (\text{byte } 2)$	2	*	7
RLC	$(b_{n+1}) \rightarrow (b_n); (b_0) \rightarrow (b_7)$ $(CY) \rightarrow (b_7)$	1	*	4
RRC	$(b_n) \rightarrow (b_{n+1}); (b_7) \rightarrow (b_0)$ $(CY) \rightarrow (b_0)$	1	*	4
RAL	$(b_{n+1}) \rightarrow (b_n); (CY) \rightarrow (b_7)$ $(b_0) \rightarrow (CY)$	1	*	4
RAR	$(b_n) \rightarrow (b_{n+1}); (CY) \rightarrow (b_0)$ $(b_7) \rightarrow (CY)$	1	*	4
CMA	$(A) \rightarrow (\bar{A})$	1	*	4
CMC	$(CY) \rightarrow (\bar{CY})$	1	*	4
STC	$(CY) \rightarrow 1$	1	*	4
КОМАНДЫ ПЕРЕХОДОВ				
JMP addr	$(PC) \rightarrow (\text{byte } 3) (\text{byte } 2)$	3	*	10
JC addr	Если $(CY)=1$, то $(PC) \rightarrow (\text{byte } 3) (\text{byte } 2)$. Иначе $(PC) \rightarrow (PC)+3$.	3	*	10
JNC addr	Если $(CY)=0$, то $(PC) \rightarrow (\text{byte } 3) (\text{byte } 2)$. Иначе $(PC) \rightarrow (PC)+3$.	3	*	7
JZ addr	Если $(Z)=1$, то $(PC) \rightarrow (\text{byte } 3) (\text{byte } 2)$. Иначе $(PC) \rightarrow (PC)+3$.	3	*	10
JNZ addr	Если $(Z)=0$, то $(PC) \rightarrow (\text{byte } 3) (\text{byte } 2)$. Иначе $(PC) \rightarrow (PC)+3$.	3	*	7
JM addr	Если $(S)=1$, то $(PC) \rightarrow (\text{byte } 3) (\text{byte } 2)$. Иначе $(PC) \rightarrow (PC)+3$.	3	*	10
JP addr	Если $(S)=0$, то $(PC) \rightarrow (\text{byte } 3) (\text{byte } 2)$. Иначе $(PC) \rightarrow (PC)+3$.	3	*	7
JPE addr	Если $(P)=1$, то $(PC) \rightarrow (\text{byte } 3) (\text{byte } 2)$. Иначе $(PC) \rightarrow (PC)+3$.	3	*	10
JPO addr	Если $(P)=0$, то $(PC) \rightarrow (\text{byte } 3) (\text{byte } 2)$. Иначе $(PC) \rightarrow (PC)+3$.	3	*	7

Продолжение табл. 3

[illegible]

Продолжение табл. 3

1	2	3	4	5
PUSH GP	((SP)-1) ← (rph); ((SP)-2) ← (rpl); (SP) ← (SP)-2	1		11
POP GP	(r1) ← ((SP)); (r1) ← ((SP)+1); (SP) ← (SP)+2	1		10
XTHL	(L) ← ((SP)); (H) ← ((SP)+1)	1		18
SPHL	(SP) ← (H) (L)	1		5
OUT port	(A) ← (data)	2		10
EI	(data) ← (A)	2		4
DI	Разрешение прерываний	1		4
HLT	Запрещение прерываний	1		7
NOP	Останов	1		4
	Нет операции	1		4

Команды управления и работы со стеком

- Примечания:
- * - команда оказывает влияние на данный флаг в зависимости от результата операции.
 - 0 - флаг переноса CY при выполнении логических операций никогда не может быть установлен в единицу и поэтому принудительно сбрасывается (CY=0).

Таблица 4

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	B, L	STAX INX	INR	DCR	MVI	RLC			DAD	LDAX DCX	INR	DCR	MVI	RRC
1		LXI	STAX INX	INR	DCR	MVI	RAL			DAD	LDAX DCX	INR	DCR	MVI	RAR
2		LXI	SHLD INX	INR	DCR	MVI	DAA			DAD	LHLD DCX	INR	DCR	MVI	CMA
3		LXI	STA	INX	INR	DCR	STC			DAD	LDA	DCX	INR	DCR	CMC
4	MOV	B, B	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
5	MOV	B, B	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
6	MOV	B, B	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
7	MOV	B, B	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV	MOV
8	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD	ADD
9	SUB	SUB	SUB	SUB	SUB	SUB	SUB	SUB	SUB	SUB	SUB	SUB	SUB	SUB	SUB
A	ANA	ANA	ANA	ANA	ANA	ANA	ANA	ANA	ANA	ANA	ANA	ANA	ANA	ANA	ANA
B	ORA	ORA	ORA	ORA	ORA	ORA	ORA	ORA	ORA	ORA	ORA	ORA	ORA	ORA	ORA
C	RNZ	POP	JNZ	JMP	CNZ	PUSH	AD1	RET	JZ						
D	RNC	POP	JNC	OUT	CNC	PUSH	AD1		JC	IN	CC				
E	RPO	POP	JPO	XTHL	CPD	PUSH	ANI								
F	RP	POP	JP	DI	CP	PUSH	ORI								

Примечания:

1. N - номер порта ввода-вывода
2. L - двухбайтовый операнда data
3. B - двухбайтовый операнда addr
4. H - однобайтовый операнда data