4. ПРОГРАММИРОВАНИЕ МИКРОПРОЦЕССОРНЫХ СИСТЕМ

4.1. Основные принципы программирования микропроцессорных систем

Микропроцессор может выполнять определенный набор действий, комбинируя которые, можно для него составлять различные программы работы вычислительной системы. Идентификатором определенного действия микропроцессора является двоичный код-команда. После ввода такого кода в микропроцессор устройство управления последнего расшифровывает код и обеспечивает выполнение заданной операции.

Набор кодов и соответствующих им операций, который может быть реализован данным микропроцессором, называется системой команд микропроцессора. Рассмотренные выше микропроцессоры имеют жесткую систему команд, которая целиком определяется логикой его устройства управления.

Каждая команда из набора команд идентифицирует некоторую операцию, выполняемую микропроцессором. В общем случае команда должна содержать два вида информации: указание на характер операции и указание на объекты данной операции (операнды). Первая часть команды является кодом операции (КОП), вторая часть команды является адресной частью.

Использование двоичных кодов для написания команд при программировании заменяется использованием буквенных кодов – мнемоник, которые гораздо более понятны человеку и облегчают процесс программирования.

При использовании мнемонических кодов код операции КОП задается буквенным сочетанием (от двух до четырех букв), которое является сокращением или аббревиатурой описания операции (на английском языке). Для записи мнемонических кодов используются заглавные буквы латинского алфавита, например

MOV – переслать,

ADD – сложить,

SUB – вычесть,

DCR – уменьшить на единицу и т.д.

Для указания числовой информации в команде могут использоваться двоичная, десятичная или шестнадцатеричная системы счисления. Последняя используется наиболее часто.

Любая информация в микроЭВМ представляется двоичным кодом. Логический нуль реализуется низким уровнем напряжения, логическая единица – высоким. МикроЭВМ работает с кодами определенной разрядности - машинными словами. Единицей информации, как правило, принят

восьмиразрядный код — один байт информации. Байтом информации можно представить $2^8 = 256$ различных состояний или десятичные числа от 0 до 255.

Двоичный код может выражать некоторую числовую или логическую информацию. Двоичными кодами в ЭВМ представляются обрабатываемые данные, адреса и команды программы.

Языки программирования

Последовательность действий микроЭВМ определяется программой, для написания которой используются различные языки, подразделяющиеся на машинно-ориентированные и алгоритмические. Из алгоритмических языков широко известны БЕЙСИК, ФОРТРАН, СИ, АДА. Машинно-ориентированным языком является АССЕМБЛЕР. В операторах машинно-ориентированного языка учитываются особенности той ЭВМ, для которой он предназначен. Таким образом, языки АССЕМБЛЕРА для разных ЭВМ различны, хотя эти различия в большинстве случаев невелики.

Программа на языке АССЕМБЛЕРА состоит из отдельных команд, определяющих содержание шагов программы. Естественным для ЭВМ является представление команд программ в виде двоичных кодов. Программа, написанная в вида последовательности двоичных кодов команд, называется объектной. Недостаток ее - сложность написания и контроля ввиду трудности восприятия человеком длинных двоичных кодов.

Язык АССЕМБЛЕРА позволяет написать программу с использованием буквенных аббревиатур для обозначения выполняемых операций. Буквенные коды команд называют также мнемоническими. Программа, написанная с помощью мнемонических кодов, называется исходной, не может быть непосредственно введена в память микроЭВМ и должна быть преобразована в двоичные коды, то есть в объектную программу. Это преобразование может быть выполнено человеком с использованием таблиц соответствия мнемокодов и двоичных кодов. Однако чаще такое преобразование выполняется с использованием ЭВМ и специальной программы - транслятора. Программа-транслятор для преобразования исходных программ, написанных на языке АССЕМБЛЕРА, в объектную программу называется АССЕМБЛЕРОМ.

Система команд микропроцессора включает следующие основные группы:

а) команды пересылок данных между регистрами, между регистрами и памятью, между регистрами и портами ввода-вывода;

б) команды арифметических операций над данными;

в) команды логических операций над данными;

г) команды передачи управления или перехода;

д) специальные команды.

Команда, написанная на языке АССЕМБЛЕРА, может быть разделена на четыре части:

МЕТКА | ОПЕРАЦИЯ | ОПЕРАНД | КОММЕНТАРИЙ

Назначение частей команды:

Метка служит для выделения данной команды в последовательности команд (например, с целью последующего обращения к ней) и представляет собой набор некоторых символов, образующих имя команды.

Операция описывает характер выполняемой микроЭВМ операции и выражается некоторым мнемокодом (кодом операции КОП), например: *ADD* - сложить, *SUB*- вычесть и т.д.

Операнд - данные, представляющие собой объект операции, реализуемой ЭВМ в ходе выполнения программы. Операнд может непосредственно содержать данные для выполняемой операции или указывать местонахождение данных (содержать адрес данных).

Комментарий - словесное пояснение выполняемой операции для упрощения пользования программой человеком. ЭВМ комментарий не обрабатывается.

Ниже приведен фрагмент программы циклового управления движениями руки манипулятора, составленной на АССЕМБЛЕРЕ 8-разрядного микропроцессора КР580ВМ80А.

```
МЕТКА   КОП   ОПЕРАНД     КОММЕНТАРИЙ
        LXI    H, 0400H    Адрес РГУ в регистровую пару HL
        MVI    M, 8BH      Управляющее слово в РГУ
BEGIN:  LXI    H, 0403H    Адрес канала А в регистровую пару HL
        MVI    M, 00H      Выключения распределителей К1 и К2
        LXI    H, 0402H    Адрес канала SB в регистровую пару HL
LOOP1:  CALL   ONESEC      Вызов подпрограммы задержки времени
        MOV    A, M        Ввод кода состояния привода
        XRI    05H         Проверка исходного состояния датчиков
        JNZ    LOOP1       Ожидание исходного состояния привода
        LXI    H, 0403H    Адрес канала А в регистровую пару HL
        MVI    M, 01H      Включение распределителя К1
        LXI    H, 0402H    Адрес канала В в регистровую пару
LOOP2:  CALL   ONESEC      Вызов подпрограммы задержки времени
```

Программа начнёт работать после включения системы управления и подачи на микропроцессорный модуль (рис. 3.29) сигналов "Сброс" R и готовность Г. Работа программы будет продолжаться до выключения системы или до снятия сигнала Г, после чего микропроцессор перейдет в состояние ожидания. Режим работы адаптера КР580ИК55 задается программным путем засылкой в его регистр управления РГУ кода 8BH, что соответствует режиму 0, программированию канала А на вывод и каналов В и С на ввод данных. Для ввода-вывода использованы команды обращения к памяти, что обусловлено выбранным способом адресации портов

ввода-вывода. Приведенную программу следует рассматривать как один из наиболее простых вариантов.

Система программирования микропроцессора

Чтобы сделать процесс разработки программ более эффективным, используется компьютерная система разработки программ. Она представляет собой вычислительную систему (центральным элементом которой является микроЭВМ), оснащенную набором системных программ. Процесс под-

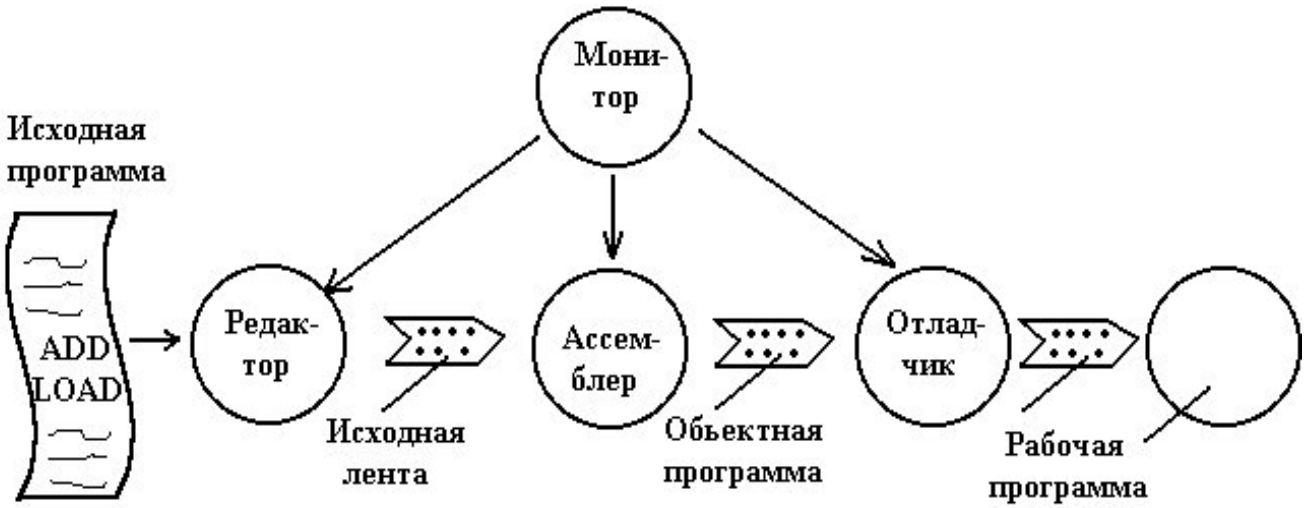

Рис. 4.1. Система программирования микропроцессора

готовки рабочей программы иллюстрируется рис. 4.1. Исходная программа с помощью текстового редактора вводится в ЭВМ и записывается на машинный носитель информации (например, на диск). При вводе исходной программы или при последующей работе с ней осуществляется проверка и редактирование программы с помощью средств редактора системы разработки.

Исходная программа с помощью ассемблера транслируется в объектную программу и записывается системой разработки на носителе в машинных кодах. Далее осуществляется отладка программы с помощью программы - отладчика системы и проверка работы программы. Полученная в результате отладки рабочая программа записывается системой разработки на том или ином носителе либо непосредственно в БИС памяти запоминающего устройства микропроцессора.

4.2. Система команд микропроцессора К580ВМ80

Системой команд называют перечень инструкций команд, который может выполнить данный микропроцессор. Микропроцессоры разных типов имеют свои несколько отличающиеся друг от друга системы команд.

Рассмотрим систему команд однокристального микропроцессора КР580ВМ80А.

Общие правила записи команд. Большинство команд состоит из двух частей. В первой части указывается на те действия, которые должен выполнить микропроцессор. Эта часть команды называется кодом операции.

Вторая часть команды указывает на то, где находятся данные или каков их адрес. Эта часть команды получила название операнд и может быть записана тремя способами:

1. В виде числа (в двоичном, десятеричном или шестнадцатеричном формате)
2. В виде адреса ячейки памяти или порта, в которых находится число.
3. В виде места, где хранится число (например, регистр В)

Из трех перечисленных способов записи чаще всего пользуются адресом, что делает программу более универсальной и пригодной для многих аналогичных случаев.

Встречаются команды, состоящие только из первой части, то есть не имеющие операнда. К таким командам относятся некоторые служебные команды, например EI, HLT.

Чтобы упростить написание и облегчить запоминание команды, ее первую часть операции записывают в виде сокращенных английских слов, указывающие выполненные действия. При этом употребляют сокращение до двух, трех или четырех букв.

Например: команда ввода-вывода IN
команда пересылки MOV
команда вызова подпрограммы CALL

Для определения места, где находятся данные, применяют условные обозначения. Расположение данных в РОН МП обозначают латинской буквой R, под которой понимаются регистры (A, B, C, D, E, H, L). Если данные находятся в ячейке памяти, то пишут букву M.

Когда операнд записывают в виде адреса, то эту запись выполняют при помощи символа ADR или какого-либо условно символического слова, состоящего из четырех-пяти букв (например - MULTI).

Для МП КР580ВМ80 употребляются команды в один, два и три байта.

В однобайтовых командах старшие разряды предназначаются для размещения кода операции, а младшие для указания кода регистров:

1 часть		2 часть					
1	0	1	0	1	0	1	0
Код операции		Код 1 операнда			Код 2 операнда		

В двухбайтовых командах первый байт полностью отводится под код операции, а второй – служит для записи данных (операнда) в виде числа, кода регистров или адреса

Команда	
Операция	Операнд
1 байт	2 байт

8 бит второго байта позволяют записать наибольшее число 25510 При необходимости ввода чисел, превышающих 255 используют трех байтовую команду, у которой первый байт – код операции, а второй и третий байты предназначены для размещения шестнадцатиразрядных двоичных чисел или шестнадцатиразрядного адреса ячейки памяти.

Код операции	Операнд	Операнд
1 байт	2 байт	3 байт

При этом имеется возможность задания чисел до 65535.

При написании команд очень важно правильно выбрать способ адресации данных который требовал бы наименьший объем памяти для размещения данных и наименьшее время исполнения команды. В МП КР 580 ВМ 80 существуют следующие способы адресации:

1. прямой;
2. регистровый;
3. косвенный;
4. непосредственный;
5. неявный.

1. При прямой адресации адрес места, где хранятся данные помещают сразу же за кодом операции. Этим адресом может быть номер порта устройства ввода-вывода. Прямая адесация наиболее простая, но в тоже время и наиболее неэкономичная из-за достаточно длинных командных слов, приводящих к увеличению объема памяти.

2. При регистровой адресации вслед за кодом операции указывают код регистра, где хранятся данные. Этот код занимает три разряда для каждого регистра, поэтому команды получаются однобайтовые и выполняются довольно быстро.

3. При косвенной адресации указывают код регистров, в которых находится адрес ячейки памяти, содержащий данные. Косвенный способ адресации позволяет компактно адресовать пространство памяти при помощи коротких однобайтовых команд.

4. При непосредственной адресации данные в виде восьмиразрядного двоичного числа непосредственно размещают во втором байте команды. Если число имеет длину более восьми разрядов, то его записывают во втором и в третьем байтах. При этом младшие разряды числа заносят во второй байт, а старшие – в третий. Команды с непосредственной адресацией обозначают буквой I, написанной в конце мнемонического слова операции, а само слово уменьшают на одну букву. Например, вместо MOV пишут MVI, вместо ADD – ADI, и вместо ORA – ORI.

5. При неявной (подразумеваемой) адресации один из операндов должен находится во внутреннем регистре микропроцессора, чаще всего в регистре А, называемом аккумулятором. Поэтому нет необходимости дополнительно сообщать его адрес. Команды с неявной адресацией получаются более короткими, что сокращает их написание и выполнение. Неявный способ адресации часто встречается в командах МП-580, который устроен так, что все логические и арифметические операции выполняются только через аккумулятор, когда один из операндов должен находится в аккумуляторе еще до выполнения команды.

Существует также смешанный способ адресации. Например, команда CALL использует косвенную адресацию и прямую.

Система команд МП-580 насчитывает более 240 команд. Многие из них повторяют свое мнемоническое название, отличаясь друг от друга только операндом. Например команда MOV повторяется более 60 раз, указывая при помощи новых операндов разные места пересылки данных. Поэтому основных (ключевых) команд с непохожими мнемоническими наименованиями имеется около 70. из них более половины команд встречается при программировании очень редко. Таким образом для начала достаточно запомнить 20-30 мнемонических названий операций, чтобы приступить к написанию простейших команд на машинном языке или языке ассемблера.

Все команды МП-580 можно условно разделить на группы:

1. команды пересылок;
2. команды арифметических операций;
3. команды логических операций;
4. команды перехода;
5. специальные команды.

4.2.1. Команды пересылок.

Команды пересылки чаще других встречаются при составлении программ и поэтому на них следует обратить особое внимание. Каждая пересылка данных всегда связанна с местом, куда они пересылаются, а именно с приемником данных и тем местом, откуда эти данные поступают – источником данных. При написании операнда условились сначала указывать приемник, а затем источник. Наименование приемника и источника разделяют запятой.

К командам пересылок относятся следующие команды:

1.MOV

Данная команда пересылок позволяет обменивать данные между РОН и памятью. Синтаксис этой команды может выглядеть, как:

1.1 MOV R1, R2.
1.2 MOV R, M.
1.3 MOV M, R.

где R1, R2, R – регистры общего назначения.

M - ячейка памяти.

При обмене данными между внутренними регистрами A,B,C,D,E,H,L команда MOV повторяется 49 раз, указывая в своем операнде различные сочетания регистров (A и B, A и C, A и E, A и L, B и A и т. д.)

Например команда MOV A,B позволяет переслать 8-битные данные из регистра B в аккумулятор A, команда MOV H,E – из регистра E в регистр H, т.е. пересылка осуществляется из R2 в R1. При этом используется регистровый способ адресации и команда является однобайтовой в которой:

0	1	Код R1	Код R2
Код операции		Коды регистров	

Команды MOV R, M и MOV M, R позволяет обменивать данные между памятью и внутренними регистрами МП. Т. к. адрес ячейки памяти является шестнадцатиразрядным то перед выполнением этой команды в регистровую пару HL необходимо поместить адрес ячейки памяти в (из) которую пересылаются данные. В этом случае по данной команде МП пересылает данные из(в) ячейки памяти, адрес которой находится в регистровой паре HL в(из) данный регистр общего назначения.

Следующая команда пересылок:

2.MVI R,K

где R – регистр общего положения

K – произвольное 8-битное число

Команда предназначена для размещения в каком либо регистре РОН МП произвольного числа К. последняя буква мнемоники команды I указывает на то что в команде участвуют непосредственные данные. Команда MVI R,K является двухбайтовой. В первом байте указывается код команды и код регистра приемника, во втором байте указывается непосредственно число К.

Например: по команде

MVI B, 12H

Число 12 записанное в шестнадцатеричном формате записывается в регистр В.

3. MVI M, K

Данная команда позволяет записать с ячейку памяти восьмиразрядное число К. При этом адрес ячейки памяти находится в регистровой паре HL. Например: необходимо в ячейке памяти М с адресом OFFOH записать число 17H.

Для этого необходимо выполнить следующие команды:

MVI H, OFH
MVI L, FOH
MVI M, 17H

Следующая команда

4.LXI P, K

Команда LXI P, K в отличии от команды MVI R,K позволяет загружать шестнадцатиразрядное, а не восьмиразрядное число К, причем запись осуществляется в регистровую пару Р, а не в регистр R.

Операнд Р обозначает одну из регистровых пар МП 580 BC, DE или HL, причем последние в этом случае обозначаются как:

BC – B
DE –D
HL – H

Команда LXI P, K – трехбайтовая.

Например: по команде LXI H, FFOOH, в регистровую пару HL загружается число FFOOH.

5. LDA ADR

С помощью команды LDA ADR можно загрузить аккумулятор А содержимым ячейки памяти с адресом ADR.

Команда LDA ADR – трехбайтовая.

Например: по команде LDA 2FF2H в аккумулятор A записывается число, находящееся в памяти по адресу 2FF2H.

6. STA ADR

Команда STA ADR, в отличии от команды LDA ADR позволяет переслать содержимое аккумулятора в ячейку памяти с адресом ADR.

7. LDAX P	8.STAX P

Команды LDAX и STAX выполняют те же операции что и команды LDA и STA, т.е. пересылают содержимое ячейки памяти M, в аккумулятор и обратно. Отличие этих команд заключается в том что при выполнении команд LDA ADR и STA ADR адрес ячейки памяти находится во 2 и 3 байтах самой команды, а при выполнении команд LDAX и STAX адрес ячейки памяти находится в регистой паре P. Причем регистовой парой может быть только пара BC и DE. В регистровой паре HL размещать такой адрес нельзя. Команды LDAX и STAX – однобайтовые и эти команды в основном используются в том случае, когда по условию выполнения программы адрес ячейки памяти M по тем или иным причинам уже находится в регистровой паре BC или DE.

10. IN

Команда IN N служит для вода данных в аккумулятор A из устройства ввода имеющего номер N. В этой команде приемником данных вида является аккумулятор, поэтому он и не указывается в операнде.

11. OUT

Команда OUT N служит для ввода данных из аккумулятора в порт вывода с номером N. Команды IN и OUT являются двухбайтовыми. Первый байт отводится для кода операции, а второй для записи адреса порта вывода-ввода.

12. PUSH	13. POP

По этим командам записывается (по команде PUSH) и считывается (по команде POP) в(из) стек содержимое либо регистровой пары B, D или H, либо слово состояния и содержимое аккумулятора. Начальный адрес об-

ласти памяти, отведенной для стека, находится в указателе стека SP микропроцессора.

Например: необходимо поместить в указатель стека содержание регистровой пары BC.

Для этого помещаем в указатель стека SP начальный адрес области памяти для стека LXI SP, ADR. Затем PUSH B.

14. XCHG

Осуществляется обмен содержимым регистровых пар DE и HL.

15. XTHL

Происходит обмен содержимого двух первых ячеек стека с регистровой парой HL.

16. SPHL

Пересылается содержимое регистровой пары в указатель стека SP.

17. LHLD

По команде LHLD ADR в регистр L передается содержимое первой ячейки памяти, адрес которой записан во второй и третий байты команды. В регистр H передается содержимое второй ячейки памяти, адрес которой на единицу больше адреса первой ячейки.

18. SHLD

Команда обратная команде LHLD ADR, когда загружаются две соседние ячейки памяти содержимым регистров H и L.

4.2.2. Команды арифметических операций.

Команды арифметических операций выполняются только через аккумулятор. При этом один из операндов должен находиться в аккумуляторе еще до выполнения команды. Таким образом, складывая два числа, необходимо одно из них поместить в аккумулятор, а второе можно загрузить в регистр R или в ячейку памяти M. Только после этого можно выполнять операции арифметики.

После выполнения команды арифметической операции результат вычисления автоматически помещается в аккумулятор, а число, которое там было стирается.

МП-К580 выполняет команды сложения и вычитания, а операции умножения и деления осуществляются по особым подпрограммам с помощью набора команд сложения, сдвига и других преобразований в соответствии с теми правилами, которые были рассмотрены нами в разделе арифметики двойного счисления.

К группе команд арифметических операций относятся также команды, уменьшающие или увеличивающие на 1 содержимое регистров РОН или ячеек памяти.

19. ADD R
20. ADD M

Команда ADD R служит для сложения числа, содержащегося в аккумуляторе с числом, размещенным в регистре R. Сумма, которая образуется после выполнения этой команды, помещается в аккумулятр, а число, находившееся там стирается.

Команда ADD R – однобайтовая.

Пример:

Сложить два десятичных числа 13 и 7. и результат поместить в ячейке памяти F1FOH

```
MVI A, 13D
MVI B, 7D
ADD B
STA F1FO
```

Команда ADDM является разновидностью команды ADD R, отличающейся тем, что число содержащееся в аккумуляторе складывается с числом находящимся в ячейке памяти М. адрес этой ячейки памяти находится в регистровой паре HL.

Команды ADD R и ADD M при выполнении операции арифметического сложения влияют на все флаги регистра признаков.

Так при сложении чисел может возникать:

1. единица переноса, которая записывается в признак С.
2. при сложении двух двоично-десятичных чисел единица в дополнительном признаке переноса АС.
3. признак знака S.
4. признак нуля 2.
5. признак четности Р.

21. ADC

Команда ADC R и ADC M аналогична командам ADD R и ADD M с отличием в том, что сложение двух чисел осуществляется с учетом признака переноса. Единица переноса прибавляется к результату вычислений в младшем разряде.

22.ADI K
23.ACI K

Эти команды позволяют сложить число К с числом находящемся в аккумуляторе. Для этого во второй байт команды нужно записать двоичное выражение числа К, после выполнения команды сумма вычислений остается в аккумуляторе.

Команды различаются тем что в ACI K число К складывается с учетом признака переноса.

24. SUB	26. SUI
25. SBB	27. SBI

Данные команды выполняют операции вычитания. По своей структуре и особенностям действий они аналогичны операциям сложения.

SUB – ADD
SBB – ADC
SUI – ADI
SBI – ACI

Все, что было сказано о командах сложения, действительно и для команд вычитания. Таким образом по команде SUB R выполняется вычитание из содержимого аккумулятора числа расположенного в регистре R, по команде SUB M выполняется вычитание числа расположенного в ячейке памяти M, а по команде SUI K – вычитание числа К, записанного во втором байте команды.

Единственное отличие команд вычитания от команд сложения стоит в том, что в аккумуляторе должно находиться уменьшаемое число. В это же время в регистре R, в ячейке памяти M или во втором байте команды должно размещаться вычитаемое.

28. INR R	30. DCR R
29. INR M	31. DCR M

Команда INR R позволяет увеличить на единицу содержимое регистра R, а команда INR M – увеличить на единицу содержимое ячейки памяти M.

аналогично команда DCR R позволяет уменьшить на единицу содержимое регистра R, а команда DCR M – уменьшить содержимое ячейки памяти M.

Команды INR R и DCR R однобайтовые. Перед выполнением команд INR M и DCR M, в которых участвуют ячейка памяти M, адрес ячейки памяти M нужно разместить в регистровой паре HL. Иногда указанные операции называют инкрементированием, а операцию уменьшения декрементированием.

При составлении программ команды INR и DCR чаще всего используют в сочетании с командами логических операций и переходов.

32. DAD P

Команда DAD P позволяет складывать два шестнадцатиразрядных числа. Одно из этих чисел помещают в регистровую пару HL, а другое в регистровую пару P (BC, DE или SP). Результат сложения размещается в регистровой паре HL. Команда DAD P однобайтовая.

Пример: сложить два числа 1263H и 0284H.

Решение:
```
LXI H, 1263H
LXI B, 0284H
DAD B
```

33.DAA

Команда DAA служит для десятичной коррекции при выполнении особого способа двоично-десятичного сложения, при котором числа из двоичного кода переводятся в двоично-десятичный формат.

4.2.3. Команды логических операций.

Команды логических операций во многом схожи с командами арифметических операций. Они так же оперируют с двумя восьмиразрядными двоичными числами, одно из которых должно находится в аккумуляторе, а второе – регистре R, в ячейке памяти M или во втором байте команды.

К группе логических команд условились относить так же операции сравнивания двух чисел и сдвига числа, находящегося в аккумуляторе.

Логические действия выполняются над числами поразрядно, то есть нулевой разряд одного числа логически сравнивается с нулевым разрядом другого числа, и результат сравнения располагается в нулевом разряде аккумулятора. Затем такое же сравнивание производится над первыми разрядами чисел, потом над вторыми и т.д. до последних седьмых разрядов.

33. ANA R

34. ANA M
35. ANI K

Команды ANA R и ANI K выполняют логическое умножение (операцию И) над двумя восьми разрядными двоичными числами, одно из которых находится в аккумуляторе, а второе в регистре R (ANA R) или в памяти (ANA M) или во втором байте команды ANI K. Перед выполнением команды ANA M адрес ячейки памяти, в которой хранится число должен быть занесен в регистровую пару HL.

Команды ANA R и ANA M – однобайтовые, а команда ANI K – двухбайтовая.

Результат операции И во всех трех командах размещается в аккумуляторе.

36. ORA R
37. ORA M
38. ORI K

Эти команды выполняют логическое сложение (операцию ИЛИ). По своей структуре и порядку выполнения они полностью соответствуют командам логического умножения

ORA R – ANA R
ORA M – ANA M
ORI K – ANI K

39. XRA R
40. XRA M
41. XRI K

Эти три команды выполняют логическую операцию ИСКЛЮЧАЮЩЕЕ ИЛИ. Эти команды полностью соответствуют структуре и правилам выполнения уже рассмотренных команд И и ИЛИ.

Довольно часто команду XRA A используют для очистки аккумулятора, т.е. для записи в его разряды всех нулей. Так например эту операцию можно выполнить с помощью двухбайтовой команды

MVI A, OOH

А если использовать команду XRA A то она займет в памяти один байт.

Пусть в аккумуляторе было записано восьмиразрядное число 01010101 В. операция XRA A осуществляет следующее действие:

01010101
+
<u>01010101</u>
00000000

Таким образом, в аккумуляторе записано число ООН.

42. CMA

Команда CMA выполняет логическую операцию НЕ. В результате этой операции все единицы во всех разрядах числа заменяются на нули, а все нули – на единицы, т.е. число инвертируется.

43. CMP R
44. CMP M
45. CPI K

С помощью этих трех команд выполняют сравнение двух чисел, одно из которых находится в аккумуляторе, а второе в регистре R, либо в ячейке памяти M, либо во втором байте команды.

Сравнение выполняется путем внутреннего вычитания содержимого регистра R, ячейки памяти M или числа K из содержимого аккумулятора. Результат вычитания определяется по состоянию триггеров Z и AC. Если сравниваемые числа равны друг другу, то триггер Z устанавливается в 1, а если не равны, то сбрасывается в нуль. Когда сравниваемое число больше того, которое находится в аккумуляторе, то срабатывает триггер AC, устанавливаясь в единицу, а если меньше, то устанавливается в нуль.

Команда CMP по своему действию соответствует команде вычитания SUB, с той лишь разницей, что результат вычислений после выполнения команды CMP не размещается в аккумуляторе, не изменяет своего значения

46. RAL	48. RAR
47. RLC	49. RRC

Эти четыре команды выполняют операцию сдвига числа находящегося в аккумуляторе. Операцией сдвига называется перемещение каждого разряда числа на одну позицию влево (RAL и RLC) или вправо (RAR и RRA).

В операциях сдвига принимает участие триггер C регистра признаков F, расположенный левее старших разрядов аккумулятора и представляет собой как бы продолжение аккумулятора.

Команды сдвига применяются для выполнения умножения деления и некоторых преобразований данных, поступающих в микропроцессор.

50. STC

Перед выполнением некоторых логических и арифметических операций требуется предварительная установка в единицу триггера переноса С, входящего в состав регистра признаков.

В каком бы положении не находился триггер С после команды STC, он устанавливается в единицу.

4.2.4. Команды переходов.

К командам перехода в МП-580 относятся команды безусловного перехода, когда по команде происходит переход по указанному адресу, расположенному во втором и третьем байтах команды и команды условного перехода, когда переход по адресу осуществляется в зависимости от состояния специальных триггеров условий.

Команды безусловного перехода.

51.JMP ADR

По команде JMP ADR выполняется безусловный переход к выполнению программы, адрес которой указывается во втором и третьем байтах команды. Для возвращения к прежней программе в конце новой программы необходимо снова записать JMP ADR, указав в ней адрес того места программы, с которого начался переход.

52. PCHL

Команда PCHL осуществляет безусловный переход к выполнению программы, адрес которой находится в регистровой паре HL. Команда PCHL – однобайтовая.

Перед выполнением команды необходимо в регистровую пару HL поместить начальный адрес новой программы.

53. CALL ADR	54. RET

Команда CALL ADR осуществляет безусловный переход к выполнению подпрограммы по адресу, указанному во втором и третьем байтах команды, а команда RET – безусловное возвращение к основной программе после выполнения подпрограммы. С этой целью адрес возврата размещается в ячейках памяти стека.

Команды условного перехода.

Напомним, что команды условного перехода осуществляют переход в зависимости от условия, записанного в команде.

Так, например, в языках высокого уровня команда условного перехода выглядит так:

IF< условие >GO TO ADR

Выполняется команда следующим образом: если условие, записанное в операторе истинно, то выполняется оператор, следующий за условием, а если условие ложно, то выполняется следующий за оператором IF оператор.

Точно так же осуществляется выполнение команды перехода в МП-580.

53. J(F) ADR

По команде J(F) ADR осуществляется условный переход к выполнению программы, адрес которой записан во втором и третьем байтах команды, если условие F выполняется. Если условие F не выполняется, то перехода не происходит, а продолжается выполнение команд основной программе. Различают восемь отдельных команд условного перехода J(F) ADR учитывающих признаки конкретного триггера, входящего в состав регистра F:

JNZ ADR – если не нулевой результат (z=0)
JN ADR – если нулевой результат (z=1)
JP ADR – если число положительное (s=0)
JM ADR – если число отрицательное (s=1)
JPO ADR – если нечетный результат (p=0)
JPE ADR – если четный результат (p=1)
JNC ADR – если нет переноса (c=0)
JC ADR – если есть перенос (c=1)

Другой разновидностью команды условного перехода является команда вызова подпрограммы при выполнении условия, учитывающего признаки триггеров.

54. C(F) ADR	55. R(F)

Команды C(F) ADR и R(F) аналогичны командам CALL ADR и RET и отличаются от них только тем, что переход к подпрограмме и возвращение из нее осуществляется в зависимости от состояния триггеров регистра признаков. Различают восемь команд C(F) ADR и R(F):

CNZ ADR и RNZ, если z=0
CZ ADR и RZ, если z=1
CP ADR и RP, если s=0

CM ADR и RM, если s=1
CPO ADR и RPO, если p=0
CPE ADR и RPE, если p=1
CNC ADR и RNC, если c=0
CC ADR и RC, если c=1

56. RSTX

Команда RSTX осуществляет переход к одной из восьми специальных подпрограмм режима прерывания, расположенных по фиксированному адресу X.

Команда RSTX является однобайтовым вариантом команды перехода с возвратом. Адрес очередной команды определяется тремя разрядами X содержащимися в коде операции:

1	1	Код X	1	1	1
Код операции RST		Код перехода			

Например RST7 → 1111 1111

F F

Так например в ЭВМ при нажатии на клавиши в клавиатуре в МП поступает запрос на прерывание. МП останавливает выполнение основной программы и считывает из внешнего для него устройства – клавиатуры код команды RST X, сформированный самой клавиатурой. После чего МП приступает к выполнению подпрограммы обслуживания прерывания, адрес которой находится в коде команды RST X.

4.2.5. Специальные команды.

Эта группа команд не производит каких-либо действий над данными. Команды не имеют операнда, а только код операции. Большинство из них является служебными командами изменяющими режим работы МП.

57. EI	58. DI

Команда EI разрешает прерывание и устанавливает триггера разрешения прерывания и связанную с ним выходную линию МП в состояние 1.

Команда DI запрещает прерывание устанавливает соответственный триггер и линию в состояние 0.

В общем случае прерывание является крайней ситуацией для МП. Иногда они нужны , иногда нежелательны, поэтому управление МП командами EI и DI позволяет эффективно использовать возможности МП.

59. NOP

Команда NOP является пустой операцией. Код этой операции ООН. МП не выполняет здесь каких либо полезных действий по обработке данных, а просто затрачивает некоторое время для различных служебных переключений во время четырех машинных тактов. Зная длительность машинного такта, можно с помощью некоторого количества команд NOP получить нужную задержку по времени выполнения программы.

Команду NOP часто используют для пространственного заполнения программы. Если появляется необходимость ввести в программу дополнительную команду, то ее размещают взамен команд NOP, записанных в нужном месте программы

60. HLT

По команде HLT микропроцессор останавливает выполнение очередной команды и переходит в режим ожидания, вывести его из этого состояния можно, подав сигнал сброса или прерывания.

4.3. Система команд К1810ВМ86

Микропроцессор ВМ86 обладает развитой системой команд, включающей операции умножения и деления. Эти команды манипулируют байтами, словами, строками, 2/10-числами в упакованном и распакованном форматах. На уровне языка ассемблера в систему команд ВМ86 входит система команд ВМ80. Однако система команд ВМ86 имеет ряд недостатков, которые устранены в новом, высокоинтегрированном однокристальном МП 80186. Системы команд МП 80186 и ВМ86 полностью совместимы на уровне объектного кода. Вместе с тем в состав команд МП 80186 много команд, которых не было в ВМ86.

В параграфе приводится набор команд нового МП 80186. Те команды, которых нет в составе команд ВМ86, выделены знаком «*». Таким образом, можно получить представление о дальнейшем усовершенствовании и расширении системы команд ВМ86.

Систему команд МП ВМ86 удобно разбить на шесть групп: пересылки, арифметической обработки, логической обработки, обработки строк,

передачи управления, управления процессором. Рассмотрим их более подробно.

Основной формат двухадресной команды OP dst, src имеет вид

code(dw) mod red r/m disp(mod)

Один из операндов кодируется полем reg и, следовательно, должен быть регистром. Второй операнд кодируется полями mod r/m и может быть как ячейкой памяти, так и регистром. Какой из операндов dst или src является регистром, определяет признак направления d в машинной инструкции: dst = reg, при d=l, src = reg, при d = 0. В зависимости от mod может потребоваться дополнительная 8- или 16-разрядная адресная компонента disp(mocl), следующая непосредственно за постбайтом. Данный формат допускает операции как с байтами (w = 0), так и со словами (w==l).

В формате не предусмотрена возможность кодирования непосредственной адресации. Поэтому для команд вида OP dst, data применяется специальный формат

Таблица 4.1

Мнемоника	Формат		
MOV dst, src	lOOOlOdw	mod reg r/m	
MOV dst, data	1 10001 Iw	mod 0 r/m	data (w)
MOV reg, data	lOllwreg	data (w)	
MOV ace, offset	10 10000 w	offset	
MOV offset, ace	lOSOOOlw	offset	
MOV seg, src	8E	mod 0 seg r/m	
MOV dst. seg	8C	mod 0 seg r/m	
PUSH . src	FF	mod 6 r/m	
PUSH reg	01010 reg		
PUSH seg	OOOse-gllO		
PUSH data*	011010SO	data (s)	
PUSHA*	60		
POP src	8F	mod 0 r/m	
POP reg	01011 reg		
POP seg	OOOseglll		
POP A*	61		j
XCHG dst reg	lOOOOHw	mod reg r/m	
XCHG AX, reg	10010 reg		
IN ace, ports	HlOOlOw	ports	
TN ace, DX	HlOliOw		
OUT ports, ace	lllOOllw	ports	
OUT DX. ace	lllOlllw		
XLAT	D7		

LEA reg, src	8D	mod reg r/m	
LDS reg, src	C5	mod reg r/m	
LES reg, src	C4	mod reg r/m	
LAHF	9F		
SAHE	9E		
PUSIIF	9C		
POPF	9D		

Примечание. В команде MOV seg, src seg^l (CS), в команде POP seg seg^l (CS), в команде XCHG AX, reg reg^O (AX), в командах LEA reg, src LDS reg, src LES reg, src mod + 11 (reg)

code (w/sw) mod code r/m disp (mod) data (w/sw)

Существуют также часто используемые короткоформатные варианты этих двухадресных команд. Операции с аккумулятором AL или AX: OP ace, data имеют формат

code(w) data(w)

Команды пересылки. Группа команд пересылки представлена в табл 4.1. Наиболее мощная команда этой группы MOV dst, src предусматривает использование в качестве источника src или приемника dst одного из регистров адресов/данных AL — BH (w = = 0) или AX —SP (w=l):

MOVdst,reg;признакd=0.

MOV reg, dst; признак d=l.

Второй операнд определяется одним из восьми способов адресации.

Направление пересылки определяется признаком d в машинной команде.

Постбайт не обеспечивает поддержку непосредственного способа адресации. По этой причине введен специальный формат команды

MOV dst, data

где data — адресное выражение типа CONSTANT. В обоих случаях пересылке подлежат либо байт (w = 0), либо слово (w=l). Указателем типа данных в первом случае служит мнемоника регистра, во втором — тип dst.

Для реализации эффективной работы с наиболее часто применяемыми операциями загрузки регистров в состав машинных команд пересылки введены короткоформатные варианты

MOV reg, data MOV ace, offset MOV offset, ace

Хотя эти команды и являются частными случаями вышеприведенных, но их формат, по крайней мере на один байт короче первых, что *дает* возможность построить более эффективный код. Указателем типа данных служит имя регистра, использованного в командах.

Примерами команды пересылки могут служит следующие:

MOV AX, WORD PTR [BX] MOV BP, OFFSET BUF MOV BYTE PTR[BX],0

В составе набора имеются две команды загрузки сегментных регистров и считывания их содержимого. Команда загрузки CS не определена, т. е. нет возможности выполнить прямую загрузку сегментного регистра CS. Это объясняется тем, что смена селектора кодового сегмента без изменения содержимого IP — смещения внутри кодового сегмента — приведет к практически не имеющей смысла передаче управления. По этой причине загрузка CS допускается только в командах межсегментной передачи управления. Следует отметить, что загрузка сегментных регистров непосредственными данными отсутствует. Для выполнения такой операции требуется двухшаговая процедура

MOV AX, SEG VAR MOV DS, AX,

обеспечивающая загрузку селектора переменной VAR в регистр DS.

Команды PUSH и POP организуют доступ к системному стеку стандартного типа. При этом пересылке подлежат только слова. Команды обеспечивают загрузку и выборку из стека операндов, хранимых как в памяти, так и в регистровой области МП. В последнем случае допускается более короткий формат кодирования. Операция загрузки CS из стека независимо от IP не определена по тем же причинам, что и ранее.

К недостаткам команд группы пересылок ВМ86 следует отнести отсутствие возможности загрузки непосредственных данных в стек. Необходимость такой загрузки возникает довольно часто при использовании стека для передачи параметров. В рамках системы команд ВМ86 для этого требуется двушаговая процедура

MOV AX, data PUSH AX

Этот недостаток устранен в новом МП 80186, в систему команд которого введена команда PUSH data.

Охват командами PUSH и POP всего набора регистров, включая сегментные, обеспечивает быстрое переключение выборочного контекста МП. Однако процесс сохранения и восстановления полного контекста МП, необходимый при переключениях задач и обработки прерываний, требует довольно длительного времени. Поэтому в состав команд 80186 введены две новые команды PUSHA (PUSH All) и POPA (POP All), обеспечивающие быстрое переключение контекста. По команде PUSHA в стек последовательно загружаются содержимое всех восьми регистров адресов/данных. Загрузка выполняется в следующем порядке: AX, CX, DX, BX, начальное значение SP, BP, S1 и DI. Команда POPA восстанавливает регистры в обратной последовательности. При этом выталкиваемое из стека значение SP теряется из-за ненадобности.

В составе группы команд пересылки ВМ86 находится команда обмена XCHG с форматами двух типов. Один из них обеспечивает обмен данными между регистром и памятью/регистром. Допускается обмен, как словами, так и байтами. Другой более короткий формат может быть применен

для кодирования операции обмена между аккумулятором AX и регистром reg (reg AX).

В МП ВМ86 используется изолированный ВВ. Для обмена данными с пространством ВВ имеются две команды ввода IN и две команды вывода OUT. Первая пара команд IN и OUT работает с коротким 8-разрядным адресом, обеспечивающим доступ к первым 256 портам ВВ с младшими адресами. При обращении к портам со старшими адресами реализуется второй вариант команд IN и OUT с косвенной регистровой адресацией, когда регистр DX служит 16-разрядным указателем порта. Во всех случаях роль источника или приемника данных выполняет аккумулятор, порт при этом может быть как 8-, так и 16-разрядным. Примеры команд ВВ:

IN AX, 32H OUT DX, AL

Специальная команда XLAT осуществляет замену содержимого AL байтом из 256-байтовой таблицы преобразования, адресуемой регистром BX. Индекс таблицы определяется первоначальным значением AL. Классическим примером использования этой команды служит преобразование кодов.

По команде LEA в регистр reg загружается исполнительный адрес (смещение) операнда src. Операнд источника должен быть расположен в памяти. Эта команда применяется для предварительной настройки соответствующих регистров на объекты. Например, команда XLAT требует предварительной настройки регистра BX, которая может быть выполнена следующим образом:

LEA BX, TAB

где TAB — переменная, обозначающая массив из 256 байт.

Две другие команды обеспечивают пересылку 32-разрядных слов, обычно интерпретируемых как полные указатели seg:offset логического пространства адресов. В команде LDS селектор загружается в регистр DS, тогда как команда LES предусматривает загрузку селектора в дополнительный сегментный регистр ES. Смещение offset всегда загружается в один из регистров reg. Одновременная загрузка полного логического адреса в пару SS:SP отсутствует. Для этого используется пара команд

MOV SS, WORD PTR PTRADDR + 2 MOV SP, WORD PTR PTRADDR

После команды MOV ss, src прерывания не проверяются, что обеспечивает согласование значений SS и SP. При другой последовательности инициализации стека необходимо предусмотреть защиту от возможного появления запросов на прерывания, переход на процедуры обслуживания которых может разрушить содержимое некоторых ячеек памяти из-за несогласованности SS и SP. Этот недостаток устранен в 32-разрядном МП 80386, в состав команд которого введена команда LSS, реализующая загрузку полного логического адреса в пару SS:reg.

Команда LAHF копирует содержимое младшего байта флажкового регистра F в регистре AH. Команда SAHF выполняет обратную пересылку. Эти команды предназначены для обеспечения совместимости BM86 и BM80 на уровне ассемблера, так как дают возможность рассматривать регистр AX как аналог регистра PSW микропроцессора BM80.

Таблица 4.2

Мнемоника	Формат		
ADD dst, src	OOOOOOdw	mod reg r/m	
ADD dst, data	1 OOOOOs w	mod 0 r/m	data(sw)
ADD ace, data	00000 lOw	data (w)	
ADC dst, src	OOOlOOdw	mod reg r/m	
ADC dst, data	lOOOOOsw	mod 2 r/m	data(sw)
ADC ace, data	0001 OlOw	data (w)	
INC dst	lllllllw	mod 0 r/m	
INC reg	OlOOOreg		
SUB dst, src	OOlOlOdw	mod reg r/m	
SUB dst, data	lOOOOOsw	mod 5 r/m	data(sw)
SUB ace, data	00101 lOw	data (w)	
SBB dst, src	OOOllOdw	mod reg r/m	
SBB dst, data	lOOOOOsw	mod 3 r/m	data(sw)
SBB ace, data	0001 HOw	data (w)	
DEC dst	lllllllw	mod 1 r/m	
DEC reg	OlOOlreg		
CMP dst, src	OOlllOdw	mod reg r/m	
CMP dst, data	lOOOOOsw	mod 7 r/m	data(sw)
CMP ace, data	OOllllOw	data (w)	
NEG dst	llllOllw	mod 3 r/m	
AAA	37	4	
DAA	32		
AAS	3F		
DAS	2F		
MUL src	llllOllw	mod 4 r/m	
IMUL src	llllOllw	mod 5 r/m	
IMUL reg, src, data*	011010s!	mod reg r/m	data(s)
DIV src	llllOllw	mod 6 r/m	
IDIV src	llllOllw	mod 7 r/m	
AAM	D4	OA	
AAD	D5	07	
CBW	98		
CWD	99		
BOUND reg, src*	62	mod reg r/m	

Примечание. В команде BOUND reg, src moduli Команды PUSHF и POPF являются частным случаем опе-ации доступа к системному стеку. Они расширяют возмож-ости операции загрузки и считывания из стека на флажковый егистр F. В системе команд BM86 отсутствует какая-либо оманда по установке/сбросу флажка трассировки TF. Эта перация может быть выполнена только с помощью команды OPF.

Все команды пересылки, за исключением SAHF и POPF, не зменяют состояния флажкового регистра.

Команды арифметической обработки. Представленная в табл. 4.2 группа содержит ряд типичных для МП команд арифметической обработки:

ADD Сложение
ADC Сложение с переносом
SUB Вычитание
SBB Вычитание с заем ом
CMP Арифметическое сравнение
TNC Увеличение на 1
DEC Уменьшение на 1
NEG Изменение знака

Все эти команды кодируются стандартными для BM86 способами. Они работают как со словами, так и с байтами. Операнды команд могут быть либо целым без знака, либо целым со знаком. В последнем случае применяется дополнительный код представления чисел. В данном случае различие между беззнаковой целочисленной арифметикой и со знаком состоит не в способе выполнения операций, а в интерпретации содержимого операндов и флажков признаков результата. Флажки устанавливаются типовым образом.

Набор команд сложения и вычитания расширен одноадресными операциями умножения и деления целых со знаком и без него:

MUL Умножение без знака
IMUL Умножение со знаком
DIV Деление без знака
IDIV Деление со знаком

Операции могут выполняться со словами и с байтами, что определяется типом src. Источником одного из операндов и приемником результата служит неявно адресуемый 8-, 16- или 32-разрядный аккумулятор, в качестве которого используется AL, AX или пара DX:AX соответственно. При этом DX представляет старшую часть аккумулятора. Операции умножения и деления выполняются по схеме

accN • srcN->acc2N acc2N:srcN->acc2N, N = 8 или 16

После выполнения операции деления в младшей части аккумулятора хранится N-разрядное частное, а в старшей — N-разрядный остаток. При делении на нуль генерируется прерывание типа 0. При выполнении опера-

ций деления и умножения могут быть полезны две вспомогательные команды расширения чисел со знаком:

CBW Расширение байта AL в слово AX

CWD Расширение слова AX в двойное слово DX:AX

В состав арифметической группы введены команды для поддержки 2/10-арифметики в упакованном формате:

DAA Десятичная коррекция AL при сложении

DAS Десятичная коррекция AL при вычитании

а также неупакованном, например в коде КОИ-7:

AAA

Коррекция AL в коде КОИ-7 при сложении

AAS

Коррекция AL в коде КОИ-7 при вычитании

AAM

Коррекция AL в коде КОИ-7 при умножении

AAD

Коррекция AL в коде КОИ-7 при делении

При выполнении десятичной арифметики предполагается, что исходные данные уже представлены в соответствующем формате. Для представления конечного результата в коде КОИ-7 необходимо с каждой распакованной 2/10-цифрой, еще находящейся в аккумуляторе, выполнить действие

OR AL, ЗОН

К недостаткам арифметической группы можно отнести отсутствие операций умножения и деления на литерал. Этот недостаток частично устранен в МП 80186.

В МП 80186 возможности команды IMUL расширены. Введено 16-разрядное умножение на литерал, имеющее два варианта записи:

IMUL reg, data IMUL reg, src, data

Схема выполнения этих команд соответственно имеет вид

reg- data -> reg src • data -> reg

Нетрудно заметить, что первый вариант является специальным случаем второго, когда src = reg.

В МП 80186 также предусмотрена команда проверки границ BOUND, благодаря которой упрощается работа с массивами данных. Для использования этой команды индекс массива помещается в один из регистров reg, а границы массива— в два соседних слова памяти (сначала нижняя, а затем верхняя). Команда проверяет вхождение индекса в интервал, отмеченный границами (граничные точки входят в интервал). В случае выхода за границы генерируется прерывание типа 5.

Команды логической обработки. Наиболее часто используемые операции логической обработки включены в набор команд МП ВМ86 (табл. 4.3). Среди них:

AND Логическое И;
OR Логическое ИЛИ;
XOR Исключающее ИЛИ;
TEST Логическое сравнение;
NOT Инверсия.

Таблица 4.3

Команда	Формат		
AND dst, src	•OOlOOOdw	mod reg r/m	
AND dst, data	JOOOOOOw	mod 100 r/m	data (w)
AND ace, data	00 100 1 Ow	data (w)	
TEST *rcg,* src	1 00001 Ow	mod reg r/m	
TEST dst, data	HllOllw	mod 000 r/m	data (w)
TEST ace, data	IQlOlOOw	data (w)	
OR dst, src	0000 IQdw	mod reg r/m	
OR dst, data	lOOOOOOw	mod 001 r/m	data (w)
OR ace, data	0000 i lOw	data (w)	
XOR dst, src	OOllOOdw	mod reg r/m	
XOR dst, data	lOOOOOOw	mod 110 r/m	data (w)
XOR ace, data	001 101 Ow	data (w)	
NOT dst	llllOilw	mod 010 r/m	
ROL dst, 1/CL	HOlOOvw	mod 000 r/m	
ROR dst, 1/CL	HOlOOvw	mod 00 i r/m	
RCL dst, 1/CL	UOlOOvw	mod ftO r/m	
RCR dst. 1/CL	HOlOOvw	mod Oil r/m	
SHL/SAL dst, 1/CL	HOlOOvw	mod 100 r/m	
SHR dst, 1/CL	HOlOOvw	mod 101 r/m	
SAR dst, 1/CL	HOlOOvw	mod 1 1 1 r/m	
ROL dst, cut*	HOOOOOw	mod 000 r/m	cnt
ROR dst, cnt*	HOOOOOw	mod 001 r/m	cnt
RCL dst, cnt*	1 lOOOOOw	mod 010 r/m	cnt
RCR dst, cnt*	HOOOOOw	mod Oil r/m	cnt
SHL/SAL dst, cnt*	llOOOOOw	mod 100 r/m	cnt
SHR dst, cnt*	HOOOOOw	mod 101 r/m	cnt
SAR dst, cut*	llOOOOOw	mod 1 1 1 r/m	cut

Для кодирования команд применены стандартные для ВМ86 способы. Возможности адресации здесь те же, что и у команд арифметической обработки. Операции с аккумулятором имеют укороченный формат.

В рассматриваемую группу включен полный набор команд логического и арифметического сдвига на одну или несколько позиций вправо (влево), что определяется вторым операндом. Когда второй операнд равен единице (v = 0), команды реализуют обычные сдвиги вправо (влево) на одну позицию. Когда в качестве второго операнда используется символ CL(v = 1), „команды образуют группу параметрических сдвигов на число позиций, определяемое содержимым CL. В составе операций сдвига следующие команды:

SAR, SAL Арифметический сдвиг
SHR, SHL Логический сдвиг
ROR, ROL Циклический сдвиг
RCR, RCL Расширенный сдвиг через CF

Среди недостатков группы команд логической обработки можно отметить отсутствие команд сдвига по литералу и команд поразрядной обработки.

Таблица 4.4

Мнемоника		Формат
MOVS	type PTR [DI], seg:type PTR [SI]	WlOOlOw
MOVSB		A4
MOVSW		A5
CMPS	type PTR [DT], scg:type PTR [SI]	lOlOOHw
CMPSB		A6
CMPSW		A7
SCAS	type PTR [DI]	lOlOlllw
SCASB		AE
SCASW		AF
LODS	segrtype PTR [SI]	10101 lOw
LODSB		AC
LODSW		AD
STOS	type PTR [DI]	lOlOlOlw
STOSB		AA
STOSW		AB
INS	type PTR [DI], DX	OllOllOw
TNSB		6C
INSW		6D
OUTS	DX, seg:type PTR [SI]	OllOlllw
OUTSB		6E
OUTSW		6F
REP	minstr	F2
REPE/REPZ	cinstr	F3
REPNE/REPNZ	cinstr	F2

Первый недостаток устранен в МП 80186, в состав команд которого введен полный набор операций сдвига по литералу. Команды, манипулирующие битами, будут реализованы только в МП 80386.

Команды обработки строк. В системе команд ВМ86 существует ряд средств по организации процедур обработки строк. Сюда входят базовые команды и префиксы повторения, кодировка которых приведена в табл. 4.4.

Базовые строковые операции используют регистр SI для адресации элемента исходной строки и регистр DI для указания на элемент строки-результата. Последняя всегда располагается в сегменте дополнительных данных. Элемент строки может быть байтом или словом. Во время выполнения операции указатели операндов SI и DI автоматически модифицируются на один или два в зависимости от типа элемента. Направлением модификации управляет флажок DF. Если DF = 0, то реализуется увеличение указателей, если DF=1—уменьшение.

В группе имеется пять базовых команд:

MOVC Пересылка элемента строки

CMPS Сравнение элементов строк методом вычитания [SI]—[DI]

SCAS Сравнение элемента строки с содержимым аккумулятора по схеме асе — [DI]

LODS Загрузка элемента строки в аккумулятор

STOS Передача содержимого аккумулятора в результирующую строку

Операции CMPS и SCAS устанавливают ряд признаков результата, сам результат не возвращают. Размер операнда определяется типом адресных выражений

OP WORD PTR[DI],WORD PTR [SI]

Существует вариант использования модификатора имени команды (суффикс B или W) для указания размера операндов, например MOVSB, LODSW. В этом случае адресные выражения не нужны и, следовательно, отсутствует возможность замены сегмента, в котором расположена строка-источник.

Любая из базовых команд может иметь префикс повторения, показывающий, что базовая операция будет повторена несколько раз. При этом регистр CX применяется в качестве счетчика числа повторений. Выход из цикла реализуется по нулевому значению CX. Проверка счетчика повторений на нуль выполняется перед каждой базовой, операцией. Это означает, что нулевое начальное значение счетчика CX не вызовет никаких действий.

Существует несколько вариантов префикса повторений:

REP-повторить;

REPE/REPZ - повторить, пока равно;

REPNE/REPNZ - повторить, пока не равно.

Первый префикс используется для повторения команд пересылки minstrMOVS, STOS известное число раз. Он реализует итерацию

while CX O do

begin CX: = CX-1; minstr end

Четыре других варианта префикса служат для повторения команд сравнения cinstr: CMPS, SCAS. В отличие от первого эти префиксы осуществляют выход из цикла при CX = 0 и ZF = 1 или ZF = O соответственно. Алгоритм их работы можно определить следующим образом:

while CX/0 do

begin CX: = CX-1; cinstr

if ZF = 0/1 then exit end

Такой префикс дает возможность сравнивать цепочки, находить конкретные символы в них, выполнять другие практически важные операции.

Более сложные процедуры обработки строк реализуются с помощью команд управления циклами типа LOOP, которые будут рассмотрены ниже. Комбинируя базовые операции с префиксами повторения и командами управления итерациями, можно построить мощные и эффективные процедуры символьной обработки. При этом очень полезной может оказаться команда трансляции XLAT, преобразующая коды символов и разбивающая символы на различные классы.

В МП 80186 добавлено две новых базовых команды типа minstr, которые обеспечивают организацию блочного ВВ:

INS Ввод элемента строки Ввод элемента строки

OUTS Вывод элемента строки

Они работают примерно так же, как и команда пересылки элемента строки MOVS, за исключением того, что одним из операндов служит порт, адресуемый DX. Здесь также существуют варианты указания типа элемента с помощью модификатора имени.

Команды передачи управления. В группе команд передачи управления (табл. 4.5) прежде всего следует выделить:

JMP Переход;

CALL Вызов подпрограммы;

RET Возврат из подпрограммы.

Эти команды обеспечивают передачу управления как внутри сегмента (типа NEAR), так и с выходом из него (типа FAR).

Таблица 4.5

Мнемоника		Формат			
JMP	disp 8	E8	dispS		
JMP	disp	E9	disp		
JMP	src!6	FF	mod	100 r/m	

JMP	sel: offset	EA	offset		sel
JMP	src32	FF	mod	101 r/m	
CALL	disp	E8	disp		
CALL	src16	FF	mod	010 r/m	
CALL	sehoffset	9A	offset		sel
CALL	src32	FF	mod	Oil r/m	
near RET		C3			
near RET	n	C2	n		
far RET		CB			
far RET	n	CA	n		
Jcc	dispS				
LOOP	diepS				
LOOP(Z/E)	dispS				
LOOP(NZ/NE)	dispS				
JCXZ	dispS				

Примечание. В командах JMP src16, JMP src32, CALL src!66, CALL src32 od ll .

Поэтому каждая команда имеет несколько кодов и форматов, выбираемых в зависимости от типа операнда exрг.

Существуют четыре разновидности каждой команды в соответствии с четырьмя допустимыми типами операнда exрг: NEAR, PTR, FAR PTR, WORD PTR и DWORD PTR. Они используются для внутрисегментной и межсегментной прямой передачи управления, а также внутрисегментной и межсегментной косвенной передачи управления соответственно. В последнем случае место расположения адреса передачи управления, кодируемого словом или двойным словом, определяется одним из способов адресации с помощью постбайта.

Передача управления внутри текущего сегмента связана с изменением только содержимого IP, тогда как передача управления между сегментами требует изменения содержимого не только IP, но и селектора CS. При вызовах подпрограмм это обстоятельство учитывается также тем, что в первом случае в стеке сохраняется только IP, а во втором — пара CS:IP.

Когда объявляется вход в подпрограмму, ему присваивается тип NEAR или FAR. Явно это делается с помощью специальной директивы объявления PROC [10, 35]:

```
SUB1    PROC  NEAR
RET
SUB1    ENDP SUB2    PROC   FAR
RET SUB2    ENDP
```

Тогда обращение CALL SUB1 должно быть выполнено в NEAR-формате, а CALL SUB2 в FAR-формате. В первом случае RET имеет NEAR-форму, а во втором он реализуется в FAR-форме. Считается, что

подпрограммы типа NEAR необъявленные. Согласование типов подпрограмм, их вызовов и возвратов возлагается на программиста.

Для каждого типа команд возврата существует две ее разновидности:

RET

RET

Здесь—целое, которое используется для увеличения содержимого SP после возврата. Эта команда может быть применена для удаления параметров, записанных в стек выполнением команды вызова.

Команды условной передачи управления, имеющие вид Jcc dispS, могут осуществлять или не осуществлять переход в зависимости от состояния флажков регистра F. Каждая из этих 18 команд опрашивает соответствующую комбинацию флажков для определения условия перехода в соответствии с табл. 1. 7. Если условие не выполняется, то управление передается следующей команде. При выполнении условия управление передается по адресу $ + disp8. Здесь адресное значение disp8 интерпретируется как целое со знаком, обеспечивающее относительную передачу управления в диапазоне —128 -н + 127 байт. Такой тип передачи управления называют коротким (SHORT).

Существует также вариант безусловной команды JMP с относительным адресом. Для явного указания на использование JMP типа SHORT применяется ключевое слово SHORT:

JMP SHORT Ml

Читателю предлагается его сравнить с обычным переходом типа NEAR

JMP Ml

Команды управления итерацией служат для организации программных циклов. Они используют регистр CX в качестве счетчика. Подобно командам условного перехода команды управления итерацией предполагают относительную адресацию в пределах — 128 +127 байт и, следовательно, являются короткими (SHORT):

LOOP disp8;CX«-CX-l;

;if CX^O then IP<-IP + disp8 ;

else exit;

LOOPE/ ;CX«-CX-1;

/LOOPZ ;if CX = 0 and ZF = 1 then IP<-IP + disp8

dips ;

else exit;

LOOPNE/ ;CX*-CX-1;

LOOPNZ ;if CX^O and ZF = 0 then IP<-IP + disp8

Dips;

else exit;

JCXZ dispS ;if CX = 0 then IP+-IP + disp8 ;

else exit;

Последнюю команду целесообразно применять в начале цикла для его обхода по CX = 0, т. е. для исполнения нулевых периодов цикла.

Команды управления процессором. В состав группы (табл. 4.6) входит ряд команд управления прерываниями, которые позволяют программам активизировать процедуры обслуживания прерываний:

INT vect

Таблица 4.6

Команда	Формат
ENTER n.l*	C8 n!6 18
LEAVE*	C9
INT vect	CD vect
INT 3	CC
INTO	CE
IRET	CF
CLC	F8
CMC	F5
STC	F9
CLD	FC
STD	ED
CLI	FA
STI	FB
HLT	F4
NOP	90
LOCK	FO
WAIT	9B
ESC op, ptr	100 11 XXX mod XXX r/m
seg:	OOlsegllO

Если vect = 3, то формируется укороченный однобайтовый вариант инструкции прерывания

INT 3

Еще одна инструкция прерывания

INT 0; if OF=1 then INT 4
else exit;

используется после арифметических или логических команд для активации процедур обслуживания переполнения. По специальной команде выполняется операция выхода из процедуры обслуживания прерывания

IRET IP<-(SP) +CS<-(SP) +F<-(SP)

Группа из семи команд позволяет изменять содержимое флажков CF, DF и IF в регистре F:

CLC CMC STC CLD STD ′CLI STI;CF^NOT CF

Если первые три команды поддерживают обычное управление флажков переноса CF, то две следующие необходимы для указания направления размещения строк. Последние две команды эквивалентны командам запрета и разрешения маскируемых прерываний по входу INTR.

Команда HLT вызывает переход МП в состояние останова. Процессор может быть выведен из данного состояния либо подачей активного уровня на линию RESET, либо при получении запроса на прерывание от внешних средств. Команда HLT является альтернативой бесконечному программному циклу в ситуациях ожидания запроса на прерывания.

Команда NOP не вызывает никаких действий МП.

Префикс LOCK может быть использован в многопроцессорной системе для организации доступа к общему ресурсу, такому, как буфер, указатель, блок данных. В таких системах одновременный доступ к одному и тому же ресурсу со стороны двух и более процессоров может привести к ошибке. Для предотвращения этого в состав МС вводятся специальные системные объекты-семафоры, которые обеспечивают монопольное владение ресурсом одним процессором. Захват ресурса другими процессорами разрешается только после его освобождения.

Пусть байт SEM — семафор типа: 0 — «Свободно»; 1—«Занято». Операция тестирования семафора и его установки в случае состояния «Свободно» может быть модифицирована в операцию считывания семафора с одновременной его установкой в состояние «Занято» и последующим анализом считанного состояния. При этом наиболее важным является этап считывания и установки. Считывание и установка должны выполняться одновременно. Этот этап можно легко осуществить с помощью команды XCHG — обмена состояния семафора с регистром, который предварительно устанавливается в необходимое состояние. Однако фаза исполнения команды XCHG состоит из нескольких циклов обращения к каналу. Для предотвращения захвата магистрали и перехвата семафора в этот период необходимо использовать префикс LOCK. Например:

```
    MOV
AL,1 WAIT:          LOCK
    XCHG
AL,SEM
    TEST
AL,AL
    JNZ
WAIT ; Доступ к ресурсу
    MOV
SEM,0
```

Команда WAIT переводит МП в состояние бесконечного ожидания активного уровня на входе TEST. Эта команда оказывается полезной при синхронизации программы с некоторыми внешними событиями.

Команда ESC предоставляет возможность другому процессору (сопроцессору) получить команду, кодируемую в поле op, а также хранящийся в памяти и кодируемый полем src операнд. При этом функция генерации адреса в соответствии с src возлагается на основной процессор.