РАБОТА № 4

ЭМУЛЯЦИЯ КОМАНД МИКРОПРОЦЕССОРА i8086

Цель работы: ознакомление с принципами микропрограммной эмуляции ЭВМ с программным управлением, микропрограммирование машинных команд микропроцессора i8086 (МП-86).

1. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1) по материалам раздела 2 ознакомиться с принципами микропрограммной эмуляции;
2) по материалам раздела 3 ознакомиться с архитектурой МП-86 и основными этапами выполнения машинных команд в МП-86;
3) по материалам раздела 4 ознакомиться с режимами адресации (регистровый, непосредственный и относительный) и соответствующими форматами команд;
4) в соответствии с вариантом задания разработать алгоритм, написать программу на языке ассемблера, закодировать и ввести ее в оперативную память (ОП);
5) составить список операций, входящих в программу решения задачи;
6) пользуясь рекомендациями раздела 5, разработать микропрограммы операций и разместить их в микропрограммной памяти;
7) составить и ввести в модель таблицу преобразования адресов;
8) ввести исходные данные и установить указатель команды IP на начало программы;
9) отладить программу в режиме МИКРОКОМАНДА;
10) выполнить программу на различных наборах исходных данных в режимах КОМАНДА и АВТОМАТ, фиксируя изменения состояния модели после выполнения каждой команды.

2. ПРИНЦИПЫ МИКРОПРОГРАММНОЙ ЭМУЛЯЦИИ

Эмуляцией в общем случае называют метод приспособления одних ЭВМ для решения задач, подготовленных для других машин. Под микропрограммной эмуляцией понимается выполнение

микропрограммируемым процессором операций программно-управляемой ЭВМ. В нашем случае ПЭВМ эмулирует микропрограммируемый процессор, а тот, в свою очередь, эмулирует некоторое подмножество команд МП-86.

В предыдущих работах составлялись микропрограммы решения задач в целом. Теперь речь идет о микропрограммировании отдельных операций, решение же задачи происходит на программном уровне.

Реализация ЭВМ с заданной системой команд на микропрограммируемых микропроцессорных наборах является часто встречающейся задачей проектирования.

3. АРХИТЕКТУРА ПРОЦЕССОРА И ОСНОВНЫЕ ЭТАПЫ ВЫПОЛНЕНИЯ МАШИННЫХ КОМАНД

В состав процессора входят (рис. 1) регистр команд RGK, арифметико-логическое устройство (АЛУ), устройство управления (УУ) и регистры, доступные программисту на уровне команд программы. Эти регистры можно разделить на четыре группы:

а) регистры общего назначения (РОНы). РОНы разделены на две половины – старшую и младшую. В некоторых командах РОНы специализированны. Это отражается в их названиях:

- аккумулятор (AX) – главный рабочий регистр, команды, относящиеся к нему, являются самыми короткими;
- база (BX) – кроме общего назначения, этот регистр используется также для адресации операндов в памяти;
- счетчик (CX) – в некоторых командах содержимое CL или CX используется как счетчик.

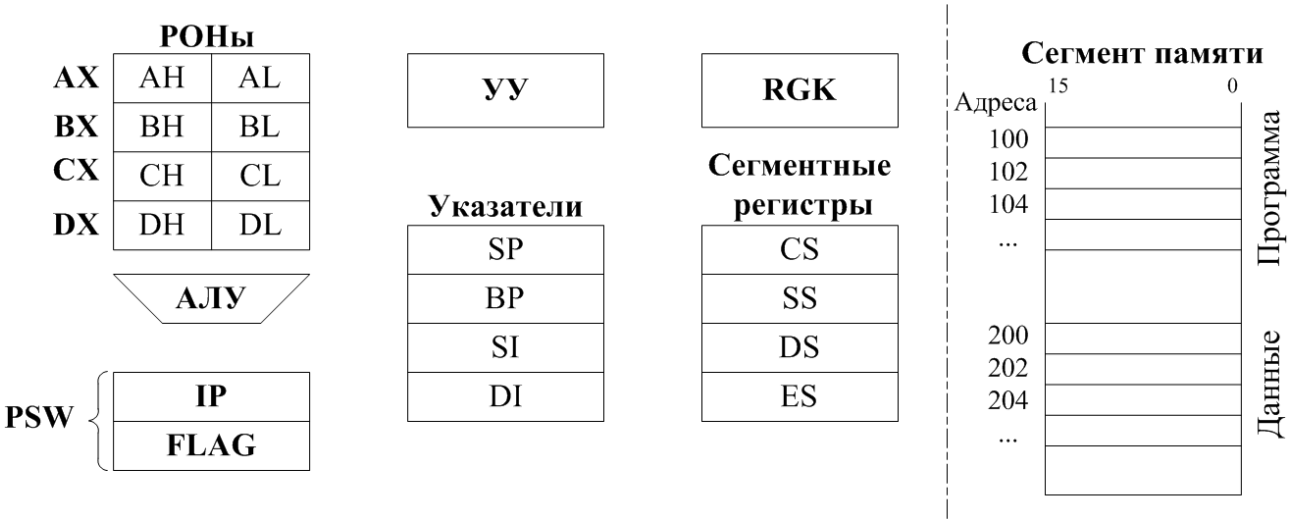

Рис. 1. Архитектура микропроцессора

б) указательные (SP – указатель стека, BP – указатель базы) и индексные (SI – индекс источника, DI – индекс получателя) регистры – это 16-битные регистры, которые используются для хранения адресов, а также участвуют в командах как РОНы;

в) сегментные регистры (CS – кода, DS – данных, SS – стека, ES – дополнительных данных). Их необходимость вызвана способом организации памяти. В общем пространстве памяти (1МБ) в любом месте произвольно находятся 4 сегмента памяти емкостью 64 Кб. Эти сегменты называются:

- сегмент кода (программы), содержащий команды программы;
- сегмент данных, содержащий данные, обрабатываемые программой;
- сегмент стека, содержащий стек;
- сегмент дополнительных данных (экстра-сегмент).

Сегментные регистры содержат начальные адреса этих 4-х сегментов, точнее, 16 старших бит адреса сегмента, а четыре младших бита считаются нулевыми;

г) регистр состояния PSW. Он состоит из программного счетчика PC (указатель IP) и регистра флажков (FLAG). IP содержит адрес *следующей* команды, выполняемой МП-86. Регистр флажков содержит 9 флажков, 3 из которых являются управляющими и 6 - арифметические, которые отражают признаки результата. Рассмотрим некоторые их флажков:

- SF – фиксирует знаковый бит результата;
- ZF – фиксирует нулевой результат;
- PF – фиксирует четное число единиц результата (флажок паритета);
- CF – фиксирует перенос (заем) из старшего бита результата.

АЛУ необходимо для выполнения указанных в КОП команд арифметических и логических операций над данными.

УУ осуществляет в соответствии с КОП команды выработку внутренних и внешних управляющих сигналов, необходимых для выполнения текущей команды.

RGK содержит код *текущей* команды на время ее дешифрации и выполнения.

В прототипе МП-86 полный (физический) адрес содержит 20 бит и равен сумме сегментного регистра, сдвинутого на 4 бита влево (сегментный адрес), и 16-битного эффективного адреса (смещение). В модели считаем, что данные и программа находятся в одном сегменте (CS=0000H, DS=0000H).

В общем виде порядок выполнения команд (программы на машинном языке) в процессоре представлен на рис. 2. В регистре IP записан начальный адрес программы в ОП. Микропроцессор обращается к ОП и осуществляет выборку команды по адресу из IP. Затем загружает ее в регистр команд RGK и дешифрирует ее с одновременным инкрементом IP для адресации следующей по порядку команды. Если команда не является командой перехода, то выполняем ее и переходим к следующей по порядку команде. Команды выполняются последовательно до тех пор, пока не встретится команда перехода. Если встречается команда безусловного перехода, то изменяется естественный порядок следования команд путем замещения содержимого IP адресом, определяемым самой командой перехода. Команды условных переходов замещают или не замещают содержимое IP в зависимости от результатов предыдущих команд, т.е. от состояния регистра FLAG. Если, например, после команды «вычитания» находится команда «перехода по нулю», переход осуществляется, если в регистре FLAG флажок ZF=1. Если же ZF=0, то переход не производится. Когда реализован переход, начинается новая последовательность команд с адреса, к которому осуществлен переход.

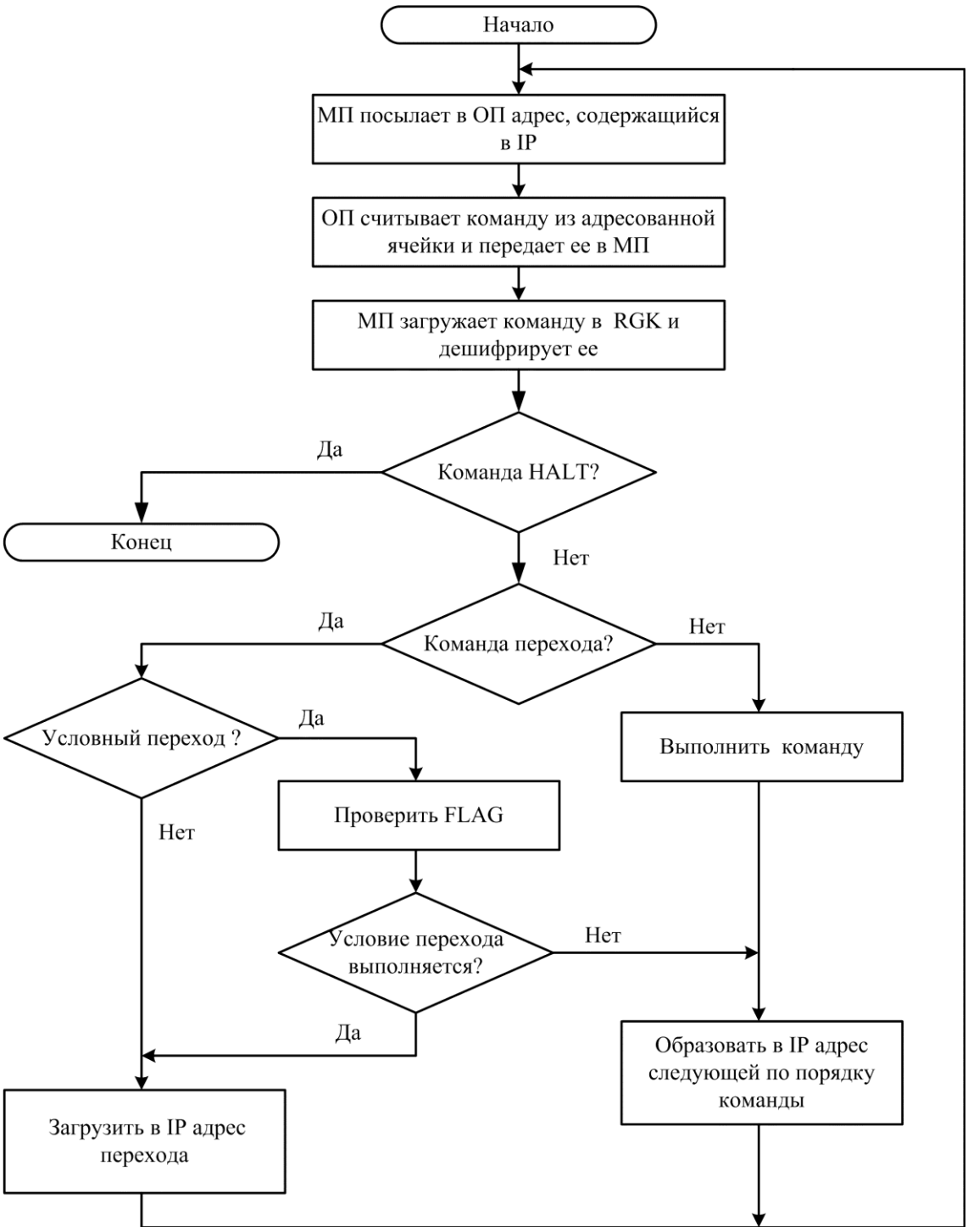

Рис. 2. Последовательное выполнение команд

4. ФОРМАТЫ КОМАНД И ОПИСАНИЕ ОПЕРАЦИЙ

4.1. Пересылочные, арифметико-логические команды и команды сдвигов

В модели реализуется лишь часть команд МП-86. Отсутствуют операции над байтами, десятичные операции и некоторые другие.

Форматы пересылочных и арифметико-логических команд приведены на рис. 3. Обозначения типов форматов не являются стандартными для МП-86 и введены для удобства описания. Длина команды изменяется от 1 до 6 байтов. Штриховой линией выделены необязательные байты, наличие или отсутствие которых зависит от способа адресации и длины непосредственного операнда.

Обозначения полей команд:

КОп - код операции;

reg - адрес регистра (данных или указателей);

mod - режим адресации;

r/m - при mod ≠ 11 - указатель регистров, участвующих в формировании адреса, при mod=11 - адрес регистра;

dispL, dispH - младший и старший байты смещения;

addrL, addrH - младший и старший байты адреса;

dataL, dataH - младший и старший байты непосредственного данного.

Операции, задаваемые различными типами команд:

R – операция над регистром reg;

AR – операция над регистром reg и аккумулятором AX;

RM – операция над регистром r/m или словом памяти;

RRM – операция над регистром reg и регистром r/m или словом памяти;

ARM – операция над AX и регистром r/m или словом памяти;

AM – операция над AX и словом памяти [addr];

AI – операция над AX и непосредственным одно- или двухбайтовым операндом;

RI – операция над регистром reg и непосредственным операндом;

RMI – операция над регистром r/m или словом памяти и непосредственным операндом.

В данной работе для команд типов RM, RRM, RMI используется только регистровая адресация (см. п.4.2). Другие способы адресации рассматриваются в следующей работе.

Пересылочные, арифметико-логические команды и сдвиги описываются в табл. 1.

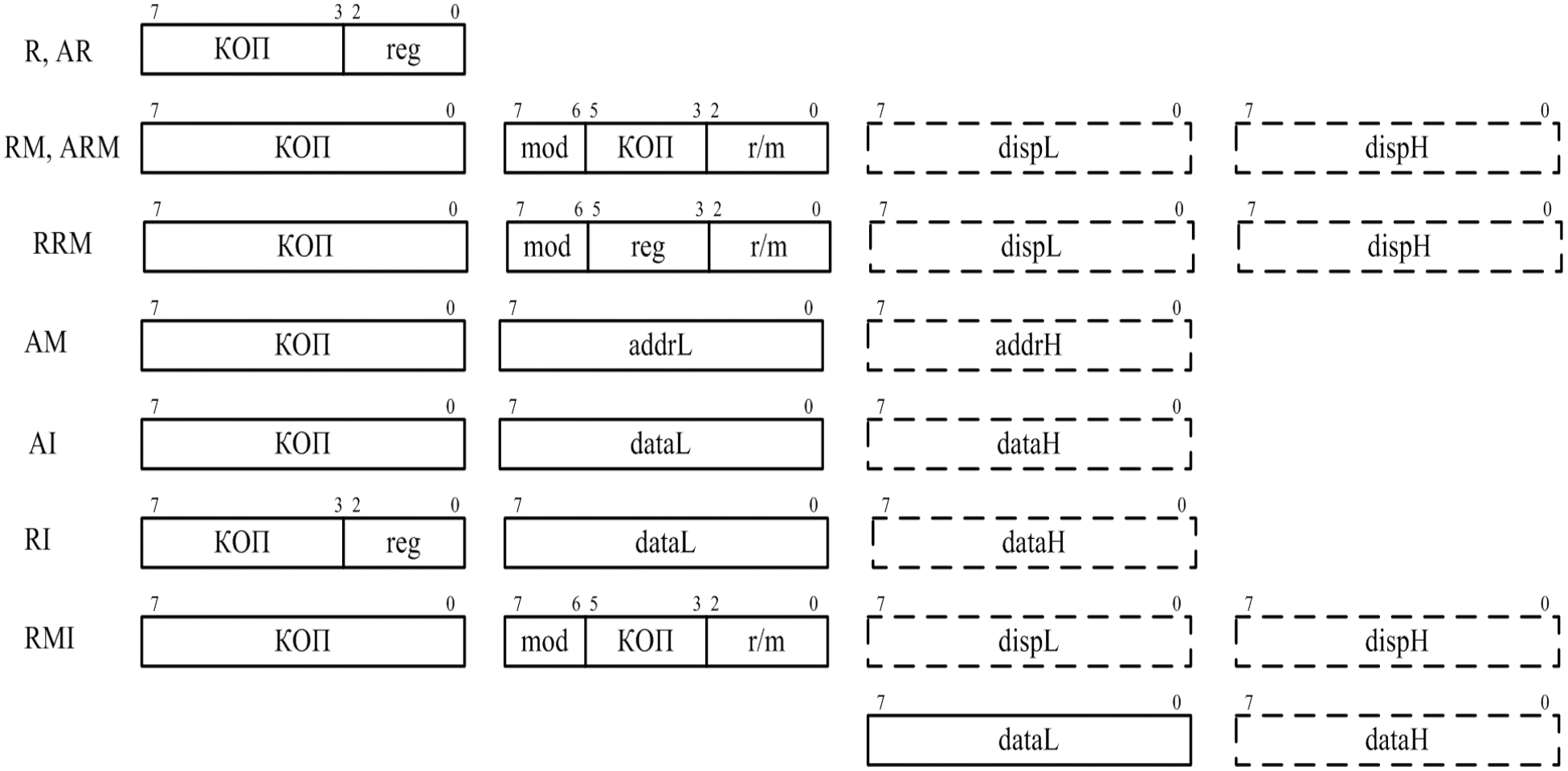

Рис. 3. Форматы пересылочных и арифметико-логических команд

Пересылочные, арифметико-логические команды и сдвиги Таблица 1

Операция	**Тип**	**Байт 1** 7654 3210	**Байт 2** 76 543 210	**Флажки** S Z N C
MOV – передать MOV dst, src; dst:=(src)	RRM RMI RI AM	1000 10d1 1100 0111 1011 1reg 1010 00d1	mod reg r/m mod reg r/m	- - - -
XCHG – обменять XCHG dst, src; (dst)<->(src)	RRM AR	1000 0111 1001 0reg	mod reg r/m mod reg r/m	- - - -
ADD – сложить ADD dst, src; dst:=(dst) + (src)	RRM RMI AI	0000 00d1 1000 00s1 0000 0101	mod reg r/m mod 000 r/m	X X X X
SUB – вычесть SUB dst, src; dst:=(dst) - (src)	RRM RMI AI	0010 10d1 1010 00s1 0010 1101	mod reg r/m mod 000 r/m	X X X X
INC – инкремент INC src; (src):=src+1	RM R	1111 1110 0100 0reg	mod 000 r/m	X X X X
DEC – декремент INC src; (src):=src-1	RM R	1111 1111 0100 1reg	mod 000 r/m	X X X X
CMP – сравнить CMP dst, src; (dst) - (src)	RRM RMI AI	0011 10d1 1011 00s1 0011 1101	mod reg r/m mod 000 r/m	X X X X
AND – объединить по И: AND dst, src; dst:=(dst) & (src)	RRM RMI AI	0010 00d1 1000 0001 0010 0101	mod reg r/m mod 100 r/m	X X 0 0
TEST – проверить (И без записи результата): TEST dst, src; (dst) & (src)	RRM RMI AI	1000 0101 1111 0111 1010 1001	mod reg r/m mod 100 r/m	X X 0 0
OR – объединить по ИЛИ OR dst, src; dst:=(dst) v (src)	RRM RMI AI	0000 10d1 1000 0001 0000 1101	mod reg r/m mod 001 r/m	X X 0 0
XOR – сложение по mod2 XOR dst, src; dst:=(dst) ⊕ (src)	RRM RMI AI	0011 00d1 1000 0001 0011 0101	mod reg r/m mod 001 r/m	X X 0 0
SHL – сдвиг логический влево	RM	1101 00v1	mod 100 r/m	X X X X
SHR – логический сдвиг вправо	RM	1101 00v1	mod 101 r/m	X X X X

SAR – арифметический сдвиг вправо	RM	1101 00v1	mod 111 r/m	X X X X

Используемые обозначения:

src – источник;

- dst – приемник;
- бит s задает длину непосредственного операнда - слово (s=0) или байт (s=1);
- бит d задает функцию регистра reg - источник (d=0) или приемник (d=1);
- бит v управляет сдвигами - на 1 бит (v=0) или на число битов, заданное в регистре CX (v=1).

Состояния флажков обозначены следующим образом:

0 (1) – установка в нуль (единицу);

X – установка в соответствии с результатом операции;

“ – “ – флажок не модифицируется.

4.2. Регистровый способ адресации

При регистровом способе адресации mod=11, а поле r/m, как и поле reg, определяет номер регистра (см. табл. 2).

Пример кодирования команды MOV DX, SI; DX := SI

	КОП	mod	reg	r/m
формат RRM	1000 1011	11	010	110
			DX	SI

Таким образом, команда занимает одно слово в ОП – 8BD6h.

Кодирование регистров Таблица 2

reg и r/m	**Регистр**	**reg и r/m**	**Регистр**
000	AX	100	SP
001	CX	101	BP
010	DX	110	SI
011	BX	111	DI

Вторым операндом может выступать *непосредственный* операнд (константа).

Пример кодирования команды ADD DX, 0ABCh; DX:= DX + 0ABCh

	КОП	mod	КОП	r/m	data L	data H
формат RMI	1000 0011	11	000	010 DX	BCh	0Ah

Таким образом, команда занимает два слова в ОП – 83C2h и 0ABCh.

4.3. Команды переходов и циклов

Команды циклов, условных и безусловных переходов имеют формат, представленный на рис. 4. Первый байт команды занимает КОП, второй байт команды называется смещением и считается целым со знаком в дополнительном коде. Для команд переходов и циклов применяется относительная адресация (рис. 5). Байт смещения (dispL), которое представлено как целое в дополнительном коде, расширяется до слова и прибавляется к указателю команд.

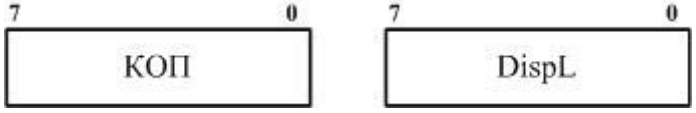

Рис. 4. Формат команд переходов и циклов

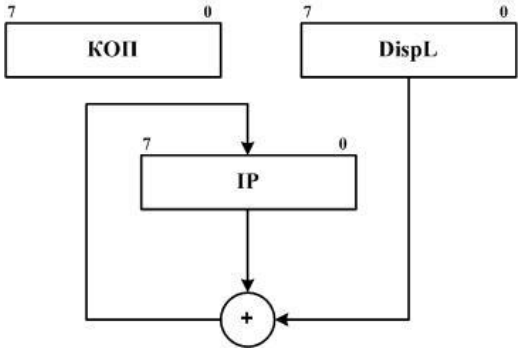

Рис. 5. Относительная адресация

Этот режим дает диапазон переходов от -128 до +127 байт, *относительно первого байта следующей команды*. Программист указывает не значение адреса, а смещение, которое необходимо прибавить к IP, чтобы получить адрес перехода.

В команде безусловного перехода адрес перехода заносится в IP (IP:=(IP)+dispL). В командах условного перехода адрес перехода заносится в IP только при выполнении условия перехода.

В командах циклов в качестве счетчика используется регистр CX. Вначале производится декремент CX, если CX≠0, то передается

управление в начало цикла, т.е. IP:=(IP)+dispL, иначе цикл завершен и выполняется следующая по порядку команда.

Команды переходов и циклов описаны в табл. 3.

5. РЕКОМЕНДАЦИИ ПО МИКРОПРОГРАММИРОВАНИЮ

5.1. Адресация регистров

Регистры, явно определяемые командой, адресуются ее полями reg1 (биты 2-0 первого байта), reg2 (биты 5-3 второго байта) или r/m (биты 2-0 второго байта). Регистры, определяемые неявно, и рабочие регистры адресуются полями A и B микрокоманды. Источники адресов РЗУ задаются полями MA/MB.

Команды переходов и циклов Таблица 3

Мнемо-ника	Код	Операция	Коды условия
JMP	1110 1011	Безусловный переход	-
JNZ	0111 0011	Переход по неравенству нулю	Z=0
JZ	0111 0100	Переход по равенству нулю	Z=1
JNS	0111 1001	Переход по плюсу	S=0
JS	0111 1000	Переход по минусу	S=1
JNO	0111 0001	Переход по непереполнению	V=0
JO	0111 0000	Переход по переполнению	V=1
JNC	0111 1011	Переход по переносу	C=0
JC	0111 1010	Переход по отсутствию переноса	C=1
JNL	0111 1101	Переход, если больше или равно	S⊕V=0
JL	0111 1100	Переход, если меньше	S⊕V=1
JNLE	0111 1111	Переход, если больше	Z∨(S⊕V)=0
JLE	0111 1110	Переход, если меньше или равно	Z∨(S⊕V)=1
JNBE	0111 0111	Переход, если больше (без знака)	C∨Z=0
JBE	0111 0110	Переход, если меньше или равно (без знака)	C∨Z=1
LOOP	1110 0010	Зациклить	CX≠0
LOOPZ	1100 0001	Зациклить, пока нуль или равно	Z=1 и CX≠0
LOOPNZ	1110 0000	Зациклить, пока нуль или не равно	Z=0 и CX≠0
JCXZ	1110 0011	Перейти, если CX=0	CX=0
HALT	1111 1111	Останов	

5.2. Выборка и дешифрация команды

Процесс выборки команды зависит от ее длины и значения младшего бита адреса. Для упрощения выборки команд примем, что

все они выровнены по границе слова и содержат четное число байтов. При этом возможно появление в программе “холостых” байтов. Основные этапы выборки и дешифрации команды:

```
ARAM:=IP;              {адрес команды}
ЧтОП; RGK:=RGR;        {чтение команды}
IP:=IP+2;              {инкремент IP}
Decode;                {дешифрация}
```

Для записи из RGR в регистр команд используется запись в регистр **E** РЗУ. При этом занесение в RGK происходит автоматически.

Микропрограмма выборки команды начинается с адреса 0. В последней МК записывается инструкция JMAP схемы УПМ для перехода к нужной микропрограмме по коду операции (CHA=2). В конце микропрограммы исполнения команды записывается инструкция JZ для перехода к выборке следующей команды (CHA=0).

5.3. Команды пересылки и преобразования данных

Микропрограммы операций типов ARM, RM, и RRM составляются с учетом способа адресации (поля mod и r/m), который рассматривается как часть кода операции. В данной работе mod=11, а поле r/m задает номер регистра и в код операции не входит. При микропрограммировании команд с непосредственным операндом помните, что он записан в одном или двух дополнительных байтах и адресуется IP. Однобайтовый операнд может быть записан в младшем байте RGK. В этом случае он выбирается без дополнительного обращения к памяти.

Если команды выровнены по границе слова, однобайтовый непосредственный операнд всегда записан во втором байте RGK, а двухбайтовый - в следующем слове памяти.

Для расширения знака однобайтового операнда используется соответствующая функция сдвигателя.

Для команд формата AM адрес записан во втором и третьем байтах команды. При обращении к памяти следует учитывать четность или нечетность адреса.

5.4. Команды переходов, циклов и останова

В микропрограммах условных переходов анализируются флажки, зафиксированные в RFD. При выполнении условия нужно запрограммировать расширение знака младшего байта команды и прибавление его к IP. Для сокращения затрат времени анализ

условия и расширение знака задаются одной МК. Аналогично, но без анализа условия выполняется короткий безусловный переход.

Операции циклов отличаются от условных переходов манипуляциями над счетчиком CX и анализом его состояния.

Команда останова HALT реализуется записью кода 5 в поле JFI.

6. КОДИРОВАНИЕ ПРОГРАММЫ И ТАБЛИЦЫ ПРЕОБРАЗОВАНИЯ АДРЕСОВ

Команды кодируются в соответствии с их форматами и кодами операций, приведенными в разделе 4. При кодировании переходов и циклов обратите особое внимание на правильность вычисления смещения. Ввод программы осуществляется в произвольную область памяти при условии, что адреса команд не превышают 3FE.

Таблица преобразования адресов устанавливает соответствие между кодами операций и начальными адресами соответствующих микропрограмм. Эта таблица является моделью преобразователя начальных адресов PA, который физически реализуется на ПЗУ или ПЛМ.

В нашей модели принято, что код операции включает в себя биты, определяющие режим адресации операндов. Биты RGK, не относящиеся к коду операции, не влияют на значение начального адреса.

Таблица состоит из двух столбцов, в первый из которых записываются адреса микропрограмм, а во второй - соответствующий код операции. Безразличные биты в коде операций заменяются символами ‘X‘. Если введено менее 16 двоичных цифр или X-в, то в конец кода записываются символы X.

7. ПРИМЕР МИКРОПРОГРАММИРОВАНИЯ

В качестве примера рассмотрим поиск наименьшего из чисел, записанных в регистрах CX, DX, BX, и занесение этого числа в AX. Алгоритм решения данной задачи был приведен в лабораторной работе № 2.

Программа поиска приведена в табл. 4.

Программа поиска наименьшего числа Таблица 4

Адрес команды	Метка	Мнемоника		16-ричный код
10		MOV	AX, CX	8BC1
12		CMP	AX, DX	3BC2
14		JLE	Min1	7E02
16		MOV	AX, DX	8BC2
18	Min1:	CMP	AX, BX	3BC3
2A		JLE	Min2	7E02
2C		MOV	AX, BX	8BC3
2E	Min2:	HALT		FF00

7.1. Кодирование команд

Кодирование команд MOV AX, CX, CMP AX,DX и HALT приведено в табл. 5. Аналогично кодируются MOV AX, DX, CMP AX, BX и MOV AX, BX.

Рассмотрим кодирование команды JLE Min1. Код операции берем из табл. 3, КОП = 0111 1110. Смещение – это значение, которое необходимо прибавить к IP, чтобы получить адрес перехода. У нас IP=16 (IP содержит адрес *следующей* команды), адрес перехода равен 18, следовательно, смещение Disp=18–16=02h. Таким образом, код команды JLE Min1 равен 0111111000000010b = 7E02h. Аналогично кодируется JLE Min2.

Кодирование команд MOV, CMP и HALT Таблица 5

Команда	Формат	Код команды в двоичном виде				Код команды в 16-ричном виде
MOV AX, CX	RRM	КОП	mod	reg	r/m	8BC1
		1000 1011	11	000	001	
				AX	CX	
CMP AX, DX	RRM	КОП	mod	reg	r/m	3BC2
		0011 1011	11	000	010	
				AX	DX	
HALT		1111 1111	0000 0000			FF00

7.2. Кодирование микропрограмм

7.2.1. Микропрограмма выборки команд

Микропрограмма выборки команды приведена в табл. 6. Адрес очередной команды определяется указателем IP. Микропрограмма выборки команд начинается с нулевого адреса. В таблице приведены поля микрокоманды, значения которых не совпадают со значениями по умолчанию.

Микропрограмма выборки команд Таблица 6

Адрес МК	Операция	Поле	Значение	Функция
00	ARAM := IP IP := IP + 2	B WM SRC ALU DST CONST	C 3 5 3 4 2	IP ARAM:= RGB CONST, RGB Сложение Запись в РЗУ
01	Чтение ОП RGK := RGR Дешифрация	MEM B DST CHA	5 E 1 2	Чтение слова RGK РЗУ[B]:= RGR JMAP

7.2.2. Микропрограммы операций

Микропрограммы операций MOV, CMP, JLE и HALT (табл. 7) записаны начиная с адресов 03, 05, 07 и 0A соответственно. Адреса регистров в операциях MOV и CMP заданы в полях reg и r/m регистра команд. В таблице приведены поля микрокоманды, значения которых не совпадают со значениями по умолчанию.

Микропрограммы операций Таблица 7

Адрес МК	Операция	Поле	Значение	Функция
03	**MOV reg2, r/m**	MA MB DST CHA	3 2 4 0	r/m reg2 (reg во втором байте RGK) Запись в РЗУ JZ – переход к выборке команды

05	**CMP reg2, r/m**	MA MB ALU CCX F CHA	3 2 1 1 1 0	r/m reg2 S-R-C0+1 C0=1 Фиксация флажков JZ – переход к выборке команды
07	**JLE** Проверка условия по фиксированным флажкам Расширение знака Условный переход	B ALU SH DST CC JFI SHA CONST	E 4 E 4 6 2 3 0009	RGK S + C0 Расширение знака Запись в РЗУ JLE Фиксированные флажки Условный переход Адрес перехода
08		CHA	0	JZ – переход к выборке команды
09	IP := IP + RGK	A B ALU DST CHA	E C 3 4 0	RGK IP R+S+C0 Запись в РЗУ JZ– переход к выборке команды
0A	**HALT**	JFI	5	Останов

7.3. Кодирование таблицы преобразования адресов

Кодирование таблицы преобразования адресов производится в соответствии с рекомендациями в разделе 6. Для нашего примера преобразование адресов описывается в табл. 8. В командах MOV, CMP безразличными являются биты 0-5 второго байта команды. В командах JLE и HALT безразличными являются все биты второго байта команды.

Преобразование адресов Таблица 8

Начальный адрес	**Код операции**			
03	1000	1011	11XX	XXXX
05	0011	1011	11XX	XXXX
07	0111	1110	XXXX	XXXX
0A	1111	1111	XXXX	XXXX

7.4. Порядок выполнения команды

На рис. 6 представлен порядок выполнения команды CMP AX, DX в микропроцессоре. Микропрограмма начинает выполняться с адреса 00 (СМК=0000). По этому адресу записана микропрограмма выборки команд, а в IP находится адрес команды CMP AX, DX. Основные этапы выборки команд:

- в регистр адреса ОП – ARAM передаем значение из IP и увеличиваем IP на 2;
- производим чтение ОП по заданному адресу и прочитанное значение заносим в регистр команд – RGK;
- выполняем дешифрацию команды: по КОП в таблице преобразования адресов определяем начальный адрес микропрограммы выполнения команды CMP reg2, r/m и полученный адрес заносим в СМК.

Далее выполняем непосредственно заданную операцию, для которой операнды определяются не полями A и B микрокоманды, а полями reg и r/m команды, этот факт фиксируется в полях MA и MB микрокоманды.

8. СОДЕРЖАНИЕ ОТЧЕТА

В отчет входят следующие пункты:

1) описание основных этапов выполнения команд в процессоре;
2) схема алгоритма решаемой задачи;
3) программа решаемой задачи на языке ассемблера;
4) кодирование команд;
5) микропрограмма выборки команд;
6) микропрограммы операций;
7) таблица преобразования адресов;
8) результаты решения задачи в режиме КОМАНДА (трасса).

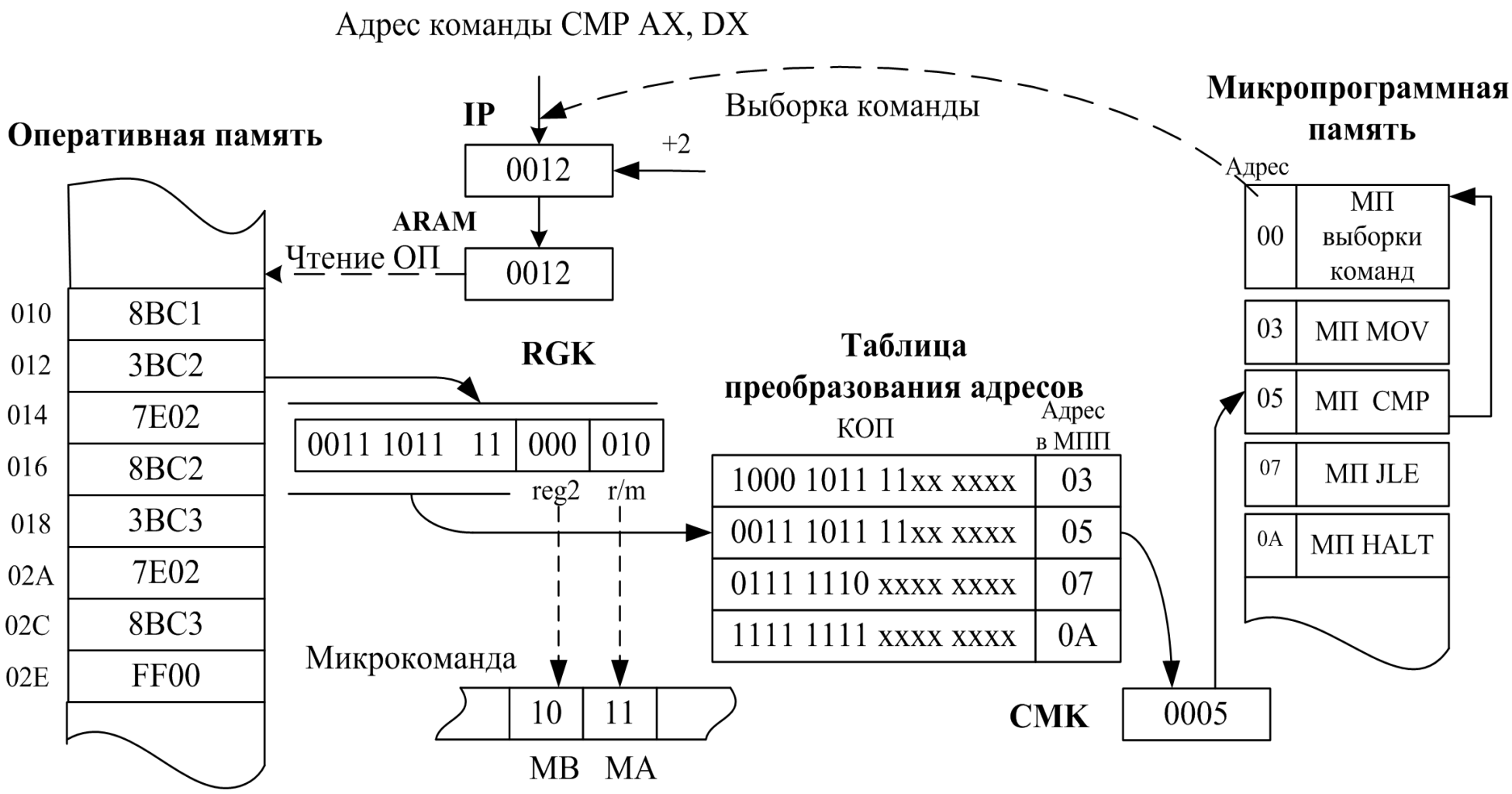

Рис. 6 Выполнение команды CMP AX, DX в микропроцессоре

9. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем состоит микропрограммная эмуляция системы команд?
2. Какова архитектура и программная модель микропроцессора i8086?
3. Каковы основные этапы выполнения машинных команд?
4. Как выполняются команды переходов и циклов? Каковы особенности их микропрограммирования?
5. Каковы особенности микропрограммирования пересылочных, арифметико-логических команд и команд сдвига?
6. Как моделируется преобразователь начального адреса?
7. Составьте микропрограмму выполнения команды по указанию преподавателя.