А. В. Карпов

ИЗУЧЕНИЕ СПОСОБОВ АДРЕСАЦИИ И ГРУППЫ КОМАНД ПЕРЕДАЧИ ДАННЫХ МИКРОПРОЦЕССОРА i8086

Методические указания к лабораторной работе

Волгоград 2006

УДК 681.325

Изучение способов адресации и группы команд передачи данных микропроцессора i8086: Методические указания к лабораторной работе/ Сост. А. В. Карпов. -- Волгоград, 2006.-- 14 с.

Приведено описание лабораторной работы “Изучение способов адресации и группы команд передачи данных микропроцессора (МП) i8086” по курсу “Вычислительные машины, системы и сети”, в которой изучаются способы адресации, операнды и команды передачи данных МП i8086. Издание предназначено для студентов дневной, вечерней и заочной форм обучения специальности 2102.

Цель работы

Целью настоящей работы является изучение способов адресации и группы команд передачи данных микропроцессора (МП) i8086.

1 Общие сведения о системе команд

Система команд 16-разрядного МП i8086 является расширенным множеством системы команд 8-разрядного МП i8080, в которой сохранено большинство команд МП i8080 за исключением редко используемых команд условных вызовов и возвратов.

Команды основных операций МП i8086 задают два операнда, одним из которых является содержимое регистра или ячейки памяти, а другим содержимое регистра или соответствующего поля в теле команды (так называемый непосредственный операнд). Результат операции может записываться на место операнда в регистре или ячейке памяти.

Система команд МП использует непосредственную, прямую, регистровую и косвенную адресацию. В МП могут выполняться арифметические операции над 8- и 16-разрядными числами со знаком и без знака, над упакованными и неупакованными двоично-кодированными десятичными числами. Для простых операций обработки последовательностей данных (цепочек) имеются однобайтные команды, для которых посредством префикса можно задавать число повторений. Имеются команды передачи управления двух типов, осуществляющие переходы соответственно внутри текущего программного сегмента и к произвольному сегменту, который при этом становится текущим. Кроме того, для управления режимом работы МП имеется несколько команд, с помощью которых можно изменять состояние МП: сброс и установка отдельных признаков в регистре FLAGS, ожидание, блокировка шины, пошаговый режим, останов и так далее.

Способы адресации

Используемые в МП способы адресации ориентированы на эффективную реализацию программ, написанных на языке высокого уровня и оперирующих с массивами данных. Так, например, к любому элементу массива можно обеспечить доступ в режиме косвенной адресации через регистры BX, SI и смещение. При этом в регистре BX располагается указатель начала записи, смещение определяет начальный адрес массива, а в регистре SI формируется индекс.

В МП реализуются следующие способы адресации: непосредственная, регистровая, прямая, косвенная и некоторые их комбинации.

Непосредственная адресация. Операнд содержится в теле команды. В двухоперандных командах имеется возможность определять операции над непосредственным операндом и содержимым регистра. Необходимо отметить, что МП не имеет команд загрузки непосредственного операнда в сегментные регистры или в стек. Пример: запись значения 0FFEh в регистр AX:

MOV AX, 0FFEh.

Регистровая адресация. Этот способ адресации определяет, что операнд находится в одном из регистров общего назначения. В двухоперандных командах задаются два регистра. Пример: пересылка данных из регистра AX в регистр BX:

MOV BX, AX.

Режимы адресации операнда в памяти

Все режимы адресации операнда в памяти получаются всего из нескольких элементов, комбинируемых различными путями. Общий вид операндов в памяти выглядит следующим образом:

[базовый регистр]+[индексный регистр]+[смещение]

где базовый регистр - это регистр BX или BP, индексный регистр - регистр SI или DI, а смещение - любая 16-битовая константа, включая имена переменных и выражения. Каждый раз, когда выполняется команда, использующая операнд в памяти, МП i8086 эти три компоненты складывает. Каждая из трех частей операнда в памяти является необязательной.

Прямая адресация. Адрес нужной ячейки памяти записывается в команде. Пример: операнд variab является адресом памяти и при выполнении этой команды происходит обращение за данными напрямую в ячейку памяти с адресом variab:

MOV AX, [variab]

Квадратные скобки, в которые заключаются непосредственные адреса, являются необязательными. То есть команды:

MOV AL, [memvar]

и

MOV AL, memvar

одинаковы.

Косвенная адресация. Адрес (смещение) может содержаться в регистрах BX, BP, SI и DI.

Адресация с использованием базовых регистров. Этот способ адресации используется для обращения к элементам различных структур данных, когда величина смещения известна во время ассемблирования, а базовый адрес структуры определяется в процессе выполнения программы.

Косвенная адресация с базированием. Пример:

MOV DX, [BX]

Квадратные скобки показывают, что в качестве операнда источника должна быть использоваться ячейка ОЗУ, на которую указывает регистр BX, а не сам регистр BX.

Прямая адресация с базированием. Адрес операнда определяется как сумма содержимого указанного базового регистра и смещения содержащегося в теле команды. Пример:

MOV DX, variab[BX]

Адресация с использованием индексных регистров. В МП данный способ адресации операнда реализуется с использованием команд прямой и косвенной адресацией и двухбайтным смещением. При этом в теле команды содержится двухбайтный “базовый адрес”, а в индексном регистре -- индекс.

Косвенная индексная адресация. Пример:

MOV AX, [DI]

Прямая адресация с индексацией. Пример:

MOV DX, variab[SI]

Адресация с индексацией и базированием. В командах, использующих данный способ адресации, адрес операнда равен сумме значений, содержащегося в регистрах. По командам с индексно-базовой адресацией обеспечивается эффективный доступ к элементам памяти, так как этот метод адресации позволяет вычислять в процессе выполнения команды не только базовый (начальный) адрес структуры данных, но и индекс элемента внутри структуры данных.

Косвенная адресация с индексацией и базированием. Пример:

MOV AX, [BX+SI]

Прямая адресация с индексацией и базированием. Пример:

MOV DX, variab[SI+BX]

Знак плюс (+), используемый внутри квадратных скобок, имеет специальное значение. Во время ассемблирования Ассемблер складывает все постоянные значения (константы) внутри квадратных скобок. Базовый регистр, индексный регистр и смещение складываются вместе МП при выполнении команды. Таким образом, Ассемблер складывает константы во время ассемблирования, а МП складывает содержимое базового регистра, индексного регистра и смещения во время действительного выполнения команды.

Операнды

Операнды указывают Ассемблеру, какие регистры, параметры, ячейки памяти и т. д. нужно связать с каждым вхождением команды или директивы. Число операндов зависит от конкретной команды или директивы. Возможные операнды включают в себя регистры, константы, метки, переменные в памяти и текстовые строки.

Регистровые операнды

Регистровые операнды являются наиболее часто используемыми в командах операндами. Регистры могут использоваться в качестве источника или приемника и при некоторых обстоятельствах могут даже содержать адрес, на который нужно выполнить переход. Имеются некоторые команды, в которых можно использовать только регистровые операнды.

Операнды-константы

Часто в операндах требуется использовать постоянное значение. Постоянные значения можно задавать в двоичном, восьмеричном или шестнадцатеричном представлении, а также в десятичном виде:

```
SUB AL, 01000001b
```

или

```
SUB AL, 41h
```

или

```
SUB AL, 65
```

В качестве постоянных операндов (операндов-констант) можно использовать также символы, поскольку символ представляет собой определенное значение. Например:

```
SUB AL, 'A'
```

Операнды-константы никогда не могут при использовании двух операндов располагаться слева, так как невозможно использовать константу в качестве операнда-приемника (это противоречит определению константы, как неизменяемой величины). Процессор i8086 накладывает на использование констант некоторые ограничения. Например, невозможно занести значение-константу в стек (это ограничение касается только процессоров 8086/8088). Чтобы занести в стек значение 5, необходимо выполнить две инструкции:

```
MOV AX, 5
PUSH AX
```

Выражения

Постоянные выражения можно использовать там же, где допускается использование постоянных значений (констант). Ассемблер поддерживает полное вычисление выражений, включая вложенные скобки, арифметические, логические операции и операции отношения, а также множество операций, предназначенных для таких целей, как выделение для меток сегмента и смещения и определение размера переменных в памяти.

Например, во фрагменте программы:

```
MemVar DB 0
NextVar DB ?
...
MOV AX, SEG MemVar
MOV DS, AX
MOV BX, OFFSET MemVar+(3*2)-5
MOV BYTE PTR [BX], 1
```

операция ***SEG*** используется для загрузки постоянного значения сегмента, в котором находится ***MemVar***, и копирования этого значения из регистра AX в DS. Далее в этой программе используется сложное выражение, включающее в себя операции *, +, - и ***OFFSET***, при вычислении которого получается значение ***OFFSET MemVar+1***, которое представляет собой ни что иное, как адрес ***NextVar***. Наконец, для выбора байтовой операции при сохранении константы 1 в ячейке, на которую указывает регистр BX (что представляет собой ***NextVar***), используется операция ***BYTE PTR***.

Примечание. При вычислении всех выражений должно получаться значение-константа. ***OFFSET MemVar*** - это значение-константа, представляющее собой смещение переменной ***MemVar*** в ее сегменте. В то время как сохраненное в переменной ***MemVar*** значение может изменяться, сама переменная ***MemVar***, никуда не перемещается.

Так как значения-константы точно известны, Ассемблер может вычислять состоящие из постоянных значений выражения так же, как он ассемблирует исходный код. Для Ассемблера выражение ***OFFSET MemVar+2*** совершенно аналогично выражению 5+2. Поскольку все элементы выражения неизменяемы и определены во время ассемблирования, выражение можно свести к одному значению-константе.

В выражениях могут использоваться следующие операции:

<>, (), LENGTH, MASK, SIZE, WIDTH, HIGH, LOW, OFFSET, PTR, SEG, THIS, TYPE, *, /, MOD, SHL, SHR +, -, EQ, GE, GT, LE, LT, NE, NOT, AND, OR, XOR, LARGE, SHORT, SMALL, TYPE

Операнды-метки

Во многих командах в качестве операндов можно использовать метки. При указании их в соответствующих операциях метки могут использоваться для получения постоянных значений (констант).

Например:

```
MemWord DW 1
...
MOV AL, SIZE MemWord
```

Здесь значение 2 (размер в байтах переменной в памяти ***MemWord***) помещается в AL. В данном контексте метка может становиться частью выражения, как уже показано в предыдущем разделе.

Метки могут также использоваться в качестве целевых операндов в операциях ***CALL*** и ***JMP***. Например, во фрагменте программы:

```
CMP AX, 100
JA IsAbove
...
IsAbove:
```

команда ***JA*** используется для перехода по адресу, заданному операндом ***IsAbove***, если значение AX превышает 100. Здесь метка используется в качестве константы, задавая адрес перехода.

Метки можно использовать в качестве операндов почти также, как используются регистры, то есть как операнд-источник или операнд-приемник в командах работы с данными. Пример:

```
TampVar DW ?
…
MOV TempVar, AX
SUB AX, TempVar
```

Группы команд МП К1810

По функциональному признаку все множество команд МП можно разделить на следующие группы: команды передачи данных, команды арифметических операций, команды логических операций и сдвига, команды операций с цепочками, команды передачи управления и команды управления МП.

Команды передачи данных

По командам данной группы выполняются операции передачи четырех типов: общие, аккумуляторные, адресные и признаковые.

IN. Ввод байта или слова

Признаки не меняются.

Команда: IN *аккумулятор*, *порт*

Логика: *аккумулятор*=[*порт*].

IN передает байт или слово из заданного *порта* в регистр AL или AX. Адрес *порта* может определяться как непосредственным байтовым значением (в диапазоне 0000h-00FFh), так и значением из регистра DX (от 0100h и выше).

LAHF. Загрузка AH из регистра флагов

Признаки не меняются.

Команда: LAHF

Логика: биты регистра AH: 7 6 4 2 0

биты регистра признаков FLAGS: S Z A P C

Команда LAHF копирует пять признаков МП (признаки знака, нулевого результата, вспомогательного переноса, четности и переноса) в биты регистра AH с номерами 7, 6, 4, 2, 0 соответственно. Сами признаки при выполнении этой команды не меняются.

LDS. Загрузка указателя с использованием DS

Признаки не меняются.

Команда: LDS *получатель*, *источник*

Логика: DS=[*источник*]

получатель=[*источник*+2]

Команда LDS загружает в два регистра 32-битный указатель, расположенный в памяти по адресу *источник*. При этом старшее слово заносится в сегментный регистр DS, а младшее слово -- в регистр *получатель*. В качестве операнда *получатель* может выступать любой 16-битный регистр общего назначения.

LEA. Загрузка исполнительного адреса

Признаки не меняются.

Команда: LEA *получатель*, *источник*

Логика: *получатель*=Адрес[*источник*]

Команда LEA присваивает значение смещения (offset) операнда *источник* (а не его значение!) операнду *получатель*. Операнд *источник* должен быть ссылкой на память, а в качестве операнда *получатель* может выступать любой 16-битный регистр общего назначения.

Примечание. Эта команда имеет то преимущество по сравнению с использованием оператора OFFSET в команде MOV, что операнду *источник* можно иметь индексы. Например, следующая строка не содержит ошибок:

```
LEA BX, TABLE[SI],
```

в то время, как строка

```
MOV BX, OFFSET TABLE[SI]
```

ошибочна, так как оператор OFFSET вычисляется во время ассемблирования, а указанный адрес не будет известен до тех пор, пока программа не будет запущена на счет.

LES. Загрузка указателя с использованием ES

Признаки не меняются.

Команда: LES *получатель*, *источник*

Логика: ES=[*источник*]

получатель=[*источник*+2]

Команда LES загружает в два регистра 32-битный указатель, расположенный в памяти по адресу *источник*. При этом старшее слово заносится в сегментный регистр ES, регистр DS, а младшее слово -- в регистр *получатель*. В качестве операнда *получатель* может выступать любой 16-битный регистр общего назначения.

MOV. Пересылка (байта или слова)

Признаки не меняются.

Команда: MOV *получатель*, *источник*

Логика: *получатель=источник*

MOV пересылает в операнд *получатель* байт или слово, находящееся в операнде *источник*.

OUT. загрузка в порт

Признаки не меняются.

Команда: OUT *порт*, *аккумулятор*

Логика: [*порт*]=*аккумулятор*

OUT передает байт или слово из регистра AL или AX в заданный *порт*. Адрес *порта* может определяться как непосредственным байтовым значением (в диапазоне 0000h-00FFh), так и значением из регистра DX (от 0100h и выше).

POP. выборка слова из стека

Признаки не меняются.

Команда: POP *получатель*

Логика: *получатель*=[SP]

SP=SP+2

Команда POP пересылает слово из вершины стека в операнд *получатель*, затем увеличивает указатель стека SP на 2, чтобы он указывал на новую вершину стека.

POPF. пересылка слова из стека в регистр FLAGS

Признаки:	O	D	I	T	S	Z	A	P	C
	r	r	r	r	r	r	r	r	r

Команда: POPF

Логика: Регистр FLAGS=[SP]
SP=SP+2

Команда POPF пересылает слово из вершины стека в регистр FLAGS, изменяя значения всех признаков, затем увеличивает указатель стека SP на 2, чтобы он указывал на новую вершину стека.

PUSH. загрузка слова в стек

Признаки не меняются.
Команда: PUSH *источник*
Логика: SP=SP-2,
[SP]=*источник*

Команда PUSH уменьшает значение указателя стека SP на 2, затем пересылает операнд в новую вершину стека. Операндом *источник* не может быть 8-битный регистр.

Примечание. Даже если *источник* указывает на байт, в стек пересылается слово. МП i80286 и i80386 перешлют в стек не те же значения, что МП i8086/i8088, если использовать команду PUSH SP. МП i80286 и i80386 перешлют старое значение SP, а i8086/i8088 -- новое значение SP в вершину стека.

PUSHF. загрузка содержимого регистра FLAGS в стек

Признаки не меняются.
Команда: PUSHF
Логика: SP=SP-2,
[SP]=регистр FLAGS

Команда PUSHF уменьшает значение указателя стека SP на 2, затем пересылает слово из регистра FLAGS в вершину стека.

SAHF. загрузка регистра AH в регистр флагов

Признаки:	O	D	I	T	S	Z	A	P	C
	r	r	r	r	r	r	r	r	r

Команда: SAHF

Логика: биты регистра признаков FLAGS:	S	Z	A	P	C
биты регистра AH:	7	6	4	2	0

Команда SAHF копирует биты регистра AH с номерами 7, 6, 4, 2 и 0 в регистр FLAGS, заменяя текущие значения признаков знака, нулевого результата, вспомогательного признака переноса, четности и переноса.

XCHG. Обмен значениями

Признаки не меняются.
Команда: XCHG *операнд1*, *операнд2*
Логика: *операнд1*$\leftarrow\rightarrow$ *операнд2*

Команда XCHG обменивает значения своих *операндов*, которые могут быть байтами или словами.

XLAT. Кодирование AL по таблице

Признаки не меняются.
Команда: XLAT [*таблица_преобразований*]

Логика: AL=[BX+AL]

Команда XLAT переводит байт, согласно *таблице_преобразований*. Указатель 256-байтовой таблицы преобразований находится в регистре BX. Байт, который нужно перевести, расположен в регистре AL. После выполнения команды XLAT байт в регистре AL заменяется на байт, смещенный на AL байтов от начала *таблицы_преобразований*.

Примечание. *Таблица_преобразований* может содержать менее 256 байтов. Операнд *таблица_преобразований*, является необязательным, поскольку указатель таблицы должен быть загружен в регистр BX еще до начала выполнения команды.

2 Порядок выполнения работы

2.1 Введите в микроЭВМ программу №1:

```
MOV DX, 300h
IN AL, DX
MOV BL, AL
IN AL, DX
ADD AL, BL  ;команда сложения двух операндов AL:=AL+BL
MOV DX, 301h
OUT DX, AL
```

Исследуйте выполнение программы №1 по командам. После выполнения каждого шага программы анализируйте содержимое всех программно-доступных регистров. Результаты занесите в таблицу.

Мнемокод	Программно-доступные регистры			Регистр признаков FLAGS	Порт 300h (301h)
	AL (AX)	BL(BX)	DX		

Замените в программе № 1 четвертую строку, содержащую вторую операцию ввода из порта поочередно, на следующие строки:

```
MOV AL, 18h
MOV AX, 1234h
```

Исследуйте выполнение программы по командам. После выполнения каждого шага программы анализируйте содержимое всех программно-доступных регистров. Результаты занесите в таблицу.

2.2 Введите в микроЭВМ программу №2:

```
MOV AX, 1234h
MOV BX, 5678h
MOV DX, 0700h
PUSH AX
PUSH BX
PUSH DX
PUSHF
POP AX
POPF
POP DX
POP BX
```

Исследуйте выполнение программы № 2 по командам. После выполнения каждого шага программы анализируйте содержимое всех программно-доступных регистров. Результаты занесите в таблицу.

3 Содержание отчета

Отчет о лабораторной работе должен содержать следующие сведения: 1) цель работы; 2) текст программ; 3) протокол полученных результатов.

4 Контрольные вопросы

1. Перечислите известные Вам способы адресации МП i8086.
2. Объясните механизм формирования физического адреса для МП i8086.
3. Перечислите известные Вам типы операндов МП i8086.
4. Перечислите известные Вам команды (с указанием их формата и логики) МП i8086, относящиеся к группе команд передачи данных.

Составитель Александр Викторович Карпов

ИЗУЧЕНИЕ СПОСОБОВ АДРЕСАЦИИ И ГРУППЫ КОМАНД ПЕРЕДАЧИ ДАННЫХ МИКРОПРОЦЕССОРА i8086

Методические указания к лабораторной работе