**Лабораторная работа № 1 «Изучение способов адресации и группы команд передачи данных микропроцессора i8086»**

**Цель работы**

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

**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.

**Способы адресации**

Используемые в МП i8086 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 i8086 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 i8086 не имеет команд загрузки непосредственного операнда в сегментные регистры или в стек. Пример: запись значения FFFEh в регистр AX:

MOV AX, FFFEh.

**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.

**Способы адресации операнда в памяти**

Позволяют реализовать следующие два основных вида адресации: прямую и косвенную.

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

[базовый регистр]+[индексный регистр]+[смещение]
где базовый регистр ‑ это BX или BP, индексный 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 инструкции:

MOV AL, [memvar]
и

MOV AL, memvar
выполняют одни и те же действия.

**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. Индексная адресация очень удобна для работы с массивами. Это связано с возможностью масштабирования содержимого индексного регистра.

Косвенная индексная адресация. Пример:

MOV AX, [DI]

Прямая адресация с индексацией. Пример:

MOV DX, variab[SI]

Адресация с использованием индексных и базовых регистров. В командах, использующих данный способ адресации, адрес операнда равен сумме значений, содержащегося в индексном и базовом 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]

Примечания.

1. Другие способы адресации получаются комбинацией рассмотренных.

2. 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 i8086 при выполнении команды. Таким образом, Ассемблер складывает константы во время ассемблирования, а МП i8086 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. Команды или директивы могут совсем не иметь операндов, иметь один или два операнда. 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). МП i8086 накладывает на использование констант некоторые ограничения. Например, невозможно занести значение‑константу в стек (это ограничение касается только МП i8086/8088). Чтобы занести в стек значение 5, необходимо выполнить две инструкции:

MOV AX, 5

PUSH AX

**Порт ввода-вывода**

Операндом является порт ввода-вывода из адресного пространства ввода-вывода. Физически порту ввода-вывода соответствует аппаратный регистр, доступ к которому осуществляется с помощью команд IN и OUT. Регистры, адресуемые с помощью порта ввода-вывода, могут иметь разрядность 8 или 16 бит, но для конкретного порта разрядность регистра фиксирована. Команды IN и OUT работают с фиксированной номенклатурой объектов. В качестве источника данных или получателя применяются регистры AL или AX. Выбор регистра определяется разрядностью порта. Номер порта может задаваться непосредственным операндом в командах IN и OUT или значением в регистре DX.

**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 регистр 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 регистр 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 регистра 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:

TempVar DW ?

...

MOV TempVar, AX

SUB AX, TempVar

**Команды передачи данных**

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

**IN (INput byte or word from port)**

**Ввод байта или слова**

Признаки не меняются.

Команда: IN аккумулятор, порт

Логика: аккумулятор=[порт]

Команда IN вводит 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).

Примечание. Аппаратная часть не использует порты от 00F8h до 00FFh для ввода/вывода, поскольку они зарезервированы для контроля за внешним процессором и для других возможных расширений МП в будущем.

**LAHF (Load AH register from register Flags)**

**Загрузка регистра AH из регистра FLAGS**

Признаки не меняются.

Команда: LAHF

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Логика: | биты регистра AH: | 7 | 6 | 4 | 2 | 0 |
|   | биты регистра признаков FLAGS: | SF | ZF | AF | PF | CF |

Команда LAHF копирует пять признаков МП i8086 (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 (Load pointer into DS)**

**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 (Load Effective Address)**

**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 (Load pointer into ES)**

**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, а младшее слово ‑ в регистр получатель. В качестве операнда получатель может выступать любой 16‑битный регистр общего назначения.

**MOV (MOVe byte or word)**

**Пересылка байта или слова**

Признаки не меняются.

Команда: MOV получатель, источник

Логика: получатель:=источник

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

**OUT (OUT byte or word to port)**

**Вывод байта или слова в порт**

Признаки не меняются.

Команда: OUT порт, аккумулятор

Логика: [порт]=аккумулятор

Команда OUT выводит 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 (POP word from the stack)**

**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 (рис. 1).



*Рис. 1. Принцип работы команды* POP

**POPF (POP Flags register from the stack)**

**Выборка слова из стека в регистр FLAGS**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Признаки: | OF | DF | IF | TF | SF | ZF | AF | PF | CF |
|   | r | r | r | r | r | r | r | r | r |

Команда: POPF

Логика: 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 (PUSH word onto stack)**

**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 источник в новую вершину стека (рис. 2).



*Рис. 2. Принцип работы команды*PUSH

Примечание. Если ***источник*** указывает на байт, в стек пересылается целое слово. МП i80286 и i80386 перешлют в стек не те же значения, что МП i8086/8088, если использовать команду PUSH SP. МП i80286 и i80386 перешлют старое значение SP, а МП i8086/8088 ‑ новое значение SP в вершину стека.

**PUSHF (PUSH Flags register onto stack)**

**Загрузка содержимого регистра FLAGS в стек**

Признаки не меняются.

Команда: PUSHF

Логика: SP=SP-2

[SP]=FLAGS

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

**SAHF (Store AH register into register Flags)**

**Сохранение содержимого регистра AH в регистре FLAGS**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Признаки: | OF | DF | IF | TF | SF | ZF | AF | PF | CF |
|   |   |   |   |   | r | r | r | r | r |

Команда: SAHF

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Логика: | биты регистра признаков FLAGS: | SF | ZF | AF | PF | CF |
|   | 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 (eXCHanGe)**

**Обмен значениями**

Признаки не меняются.

Команда: XCHG операнд 1, операнд 2

Логика: операнд 1 <-> операнд 2

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

**XLAT (transLATe byte from table)**

**Кодирование байта в регистре 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 байт от начала таблицы\_преобразований.

Примечание. ***Таблица\_преобразований*** может содержать менее 256 байт. Операнд ***таблица\_преобразований***, является необязательным, поскольку указатель таблицы должен быть загружен в регистр BX еще до начала выполнения команды.

**Порядок выполнения работы**

1. Введите в микроЭВМ программу №1:

MOV DX, x00h[**[См. вариант]**](http://i8086.ucoz.net/publ/metodicheskie_ukazanija_k_laboratornym_rabotam/lr1/2-1-0-2#_ftn1)

IN AL, DX

MOV BL, AL

IN AL, DX

ADD AL, BL    ;команда сложения двух операндов AL:=AL+BL

MOV DX, x01h[**[См. вариант]**](http://i8086.ucoz.net/publ/metodicheskie_ukazanija_k_laboratornym_rabotam/lr1/2-1-0-2#_ftn1)

OUT DX, AL

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

Таблица 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Мнемокод** | **Программно-доступные регистры** | **Регистр признаков**FLAGS | **Порт**x00h[**[См. вариант]**](http://i8086.ucoz.net/publ/metodicheskie_ukazanija_k_laboratornym_rabotam/lr1/2-1-0-2#_ftn1) | **Порт**x01h[**[См. вариант]**](http://i8086.ucoz.net/publ/metodicheskie_ukazanija_k_laboratornym_rabotam/lr1/2-1-0-2#_ftn1) |
| AL (AX) | BL (BX) | DX |
|   |   |   |   |   |   |   |

Замените в программе № 1 четвертую строку, содержащую вторую команду ввода из порта на следующую строку:

MOV AL, xx**[[См. вариант]](http://i8086.ucoz.net/publ/metodicheskie_ukazanija_k_laboratornym_rabotam/lr1/2-1-0-2%22%20%5Cl%20%22_ftn1)**

Исследуйте выполнение программы по командам. После выполнения каждого шага программы анализируйте содержимое всех программно‑доступных регистров. Результаты занесите в таблицу 1.

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.

**Содержание отчета**

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

**Контрольные вопросы**

1. Перечислите способы адресации МП i8086.

2. Объясните механизм формирования физического адреса для МП i8086.

3. Перечислите типы операндов МП i8086.

4. Перечислите команды (с указанием их формата и логики) МП i8086, относящиеся к группе команд передачи данных.

**Варианты заданий к лабораторной работе**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Предпоследняя цифра варианта** | **Первое исследуемое число** | **Порт ввода** |   | **Последняя цифра варианта** | **Второе исследуемое число** | **Порт вывода** |
| 0 | 13h | 100h |   | 0 | D4h | 301h |
| 1 | 76h | 100h |   | 1 | 4Ah | 301h |
| 2 | 51h | 300h |   | 2 | 9Fh | 101h |
| 3 | BCh | 300h |   | 3 | 64h | 101h |
| 4 | 5Eh | 100h |   | 4 | D5h | 301h |
| 5 | B1h | 100h |   | 5 | 25h | 301h |
| 6 | 94h | 300h |   | 6 | 7Ch | 101h |
| 7 | D1h | 300h |   | 7 | 9Ch | 101h |
| 8 | B4h | 100h |   | 8 | A8h | 301h |
| 9 | 49h | 100h |   | 9 | D2h | 301h |