

Режимы адресации микроспроцессора i8088

Понятие режима адресации

1. Команды микропроцессора оперируют данными, которые указаны в операндах команд.
2. Данные могут находиться:
 - в самой команде
 - в регистре, который явно или неявно указан в команде
 - в ячейке памяти, адрес которой явно или неявно указан в команде

Под режимом адресации будем понимать способ, который использует микропроцессор для определения месторасположения данных, которые необходимо обработать в команде

Микропроцессор автоматически определяет метод адресации по тому, как записана команда

Режимы адресации

1. Непосредственная
2. Регистровая
3. Прямая
4. Косвенная регистровая
5. Прямая по базе
6. Прямая с индексированием
7. Прямая по базе с индексированием.

Режимы (3 -7) режимы адресации памяти.

Непосредственная адресация

Применяется , когда в качестве операнда команды используется константа (может использоваться только во втором операнде команды).

В этом случае данное извлекается непосредственно из команды.

Пример

Mov cx, 500 ; переслать значение 500 в
; регистр cx

K equ 500; определить константу k=500
mov cx,k ; переслать значение
; константы k регистр cx

Особенности непосредственной адресации

1. Длина непосредственного операнда зависит от длины первого операнда.

mov ah, 0123h ; ошибка

2. Если длина первого операнда превышает длину непосредственного операнда, то МП увеличивает длину непосредственного операнда до длины первого

Mov cx,500 ; cx=0000 0001 1111 0100

3. В программе на ассемблере допустимо указывать отрицательные значения

**Mov cl,-30 ; cx= 0000 0000 1110 0010 в ;
дополнительном коде**

Регистровая адресация

Применяется, когда в команде указаны регистры МП.

В этом случае МП выбирает данные из регистра или помещает данные в регистр

**Mov ax, cx ; переслать ;
содержимое ; регистра cx
в ; регистр ax**

Длина регистров должна быть одинаковой.

Mov ax, cl ; ошибка

Режимы адресации памяти

Применяются для выборки данных из оперативной памяти (пересылки данных в оперативную память).

При этом способе адресации в команде ассемблера указывается **исполнительный адрес** месторасположения данных в памяти.

Исполнительный адрес данных определяет адрес первого байта обрабатываемых данных.

Длина обрабатываемых данных определяется записью команды.

Исполнительный адрес может быть задан:

- либо в виде указателя на ячейку памяти

Сегментный адрес: смещение

- либо указанием только смещения ячейки памяти. В этом случае используется сегментный адрес по умолчанию (в разных случаях – разный).

Прямая адресация

Применяется в том случае, когда исполнительный адрес ячейки задан непосредственно в команде.

Основной способ - указание имени ячейки памяти и, возможно, сдвига.

В этом случае исполнительный адрес ячейки интерпретируется как смещение внутри текущего сегмента данных. Смещение нужной ячейки памяти вычисляется как сумма смещения имени внутри сегмента, в котором расположена ячейка памяти и, если задан, сдвига.

Сдвиг измеряется в байтах.

Пример

Field dw? ; Определить имя field как ;
ячейку памяти длиной слово в ;
сегменте

Mov ax,field+3 ; занести в регистр ax
; содержимое ячейки ; памяти из
текущего ; сегмента данных,
смещение ; которой равно смещению
; ячейки **field+3 байта.**

Пример

```
A dw 0001h  
   dw 0002h
```

```
Mov ax, a+1 ; ax=?
```

```
Mov bx, a+2 ; bx=?
```

Результат

Содержимое памяти (шестнадцатеричный вид)

01000200

A+1=0002, ax=0200

Bx=0002

Косвенная регистровая адресация

Применяется, когда исполнительный адрес ячейки памяти является смещением и записан в регистрах bx , bp , si , di , или вычисляется как сумма содержимого одного из базовых (Bx , bp) и одного из индексных (si , di) регистров.

Если используется регистр bx и явно не указано иное, то считается, что ячейка памяти находится в текущем сегменте данных.

Если используется регистр bp и явно не указано иное, то считается, что ячейка памяти находится в текущем сегменте стека.

Длина ячейки памяти определяется из записи команды

Правила записи

Mov ax, [bx]

Занести в регистр ax содержимое ячейки памяти длиной слово из текущего сегмента данных. Величина смещения (в байтах) находится в регистре bx.

Mov ax, [bx+si]

Mov ah, [si] ; используется ячейка памяти длиной байт

Не допускается совместное использование двух базовых и двух индексных регистров.

Как получить смещения?

1. Командой `Lea`
2. Операцией `Offset`

`Field db '1234'` ; определение переменной
в сегменте данных и присвоение ей
значения `'1234'`

`Lea bx, field` ; В регистр `bx` занести
смещение имени `field` в
текущем сегменте данных

`Mov ax, [bx]` ; `ax='1234'`

Операция offset

`Mov bx, offset field ; bx=смещению field`

`Mov ax,0`

`Mov [bx], ax ; Что сделает эта команда?`

Прямая адресация с базированием

Применяется в том случае, если исполнительный адрес задан в виде смещения и определяется:

- Суммой содержимого базового регистра (bx, bp) и сдвига (в байтах);
- Суммой содержимого базового регистра (bx, bp) и смещения ячейки памяти, заданной символическим именем;
- Суммой содержимого базового регистра (bx, bp), смещения ячейки памяти, заданной символическим именем, и сдвига (в байтах);

Пример

- `Lea bx, field` ; занести в `bx` смещение `field` ; ячейки
- `Mov bp, bx`
- `Mov dx, [bx+1]`
- `Mov ax, table[bp+1]` ; операция в ; сегменте данных
- `Mov table[bx+1], ax`

Занести в ячейку памяти длиной слово и со смещением, равным сумме смещения ячейки `table`, содержимого регистра `bx` и `1` в сегменте данных, значение, содержащееся в регистре `ax`.

Прямая адресация с индексированием

Применяется в том случае, если исполнительный адрес задан в виде смещения и определяется:

- Суммой содержимого индексного регистра (si, di) и сдвига (в байтах);
- Суммой содержимого индексного регистра (si, dip) и смещения ячейки памяти, заданной символическим именем;
- Суммой содержимого индексного регистра (si, di), смещения ячейки памяти, заданной символическим именем, и сдвига (в байтах);

Пример

- `Lea si, field ; занести в si смещение ; ячейки field`
- `Mov dx, [si+1]`
- `Mov ax, table[si+1] ;`
- `Mov table[si+1], ax`

Занести в ячейку памяти длиной слово и со смещением, равным сумме смещения ячейки `table`, содержимого регистра `si` и 1 в сегменте данных, значение, содержащееся в регистре `ax`.

Эти методы адресации удобны для доступа к последовательно расположенным ячейкам памяти (элементам таблицы)

Table db 100 dup(0); Задать байтовую
; область ; памяти (таблицу)
; длиной 100 байт

mov si, 1

Mov ah,table[si] ; занести в ah первый
; байт таблицы

Mov si,2

Mov ah,table[si] ; 2 элемент

Прямая адресация с базированием и индексированием

Применяется в том случае, если исполнительный адрес задан в виде смещения и определяется:

- Суммой содержимого индексного регистра (si, di), базового регистра (bx, bp) и сдвига (в байтах);
- Суммой содержимого индексного регистра (si, di), базового регистра (bx, bp) и смещения ячейки памяти, заданной символическим именем;
- Суммой содержимого индексного регистра (si, di), базового регистра (bx, bp), смещения ячейки памяти, заданной символическим именем, и сдвига (в байтах);

Пример

- `Mov dx, [bx+si+1]`
- `Mov ax, table[si+bx] ;`
- `Mov table[bx+si+1], ax`
- `Mov table[bp+si+1],ax`

Занести в ячейку памяти длиной слово и со смещением, равным сумме смещения ячейки `table`, содержимого регистра `si`, содержимого регистра `bx` (`bp`) и `1` в сегменте данных, значение, содержащееся в регистре `ax`.

Удобно применять для доступа к элементам двумерных массивов

Важное замечание

Правила определения сегмента, в котором находится ячейка памяти.

1. Если исполнительный адрес формируется с использованием регистра `bp` без использования имен ячеек памяти, то сегментный адрес ячейки памяти извлекается из регистра `SS` (текущий сегмент стека)
2. В командах обработки строк регистр `di` адресует текущий дополнительный сегмент (сегментный адрес извлекается из регистра `es`)
3. В остальных случаях ячейка памяти находится в текущем сегменте данных (сегментный адрес извлекается из регистра `ds`)

Явное указание сегмента

Для явного указания сегмента
используется имя сегментного регистра.

```
Mov bp, es:[bx] ; операция в текущем  
; дополнительном ;  
сегменте
```

```
Mov cs:[bp], dx
```

Возможные варианты написания команд

Mov ax, [bx-1]

Mov ,ax, table[si]+1

Mov ax, table[bx][si+1]

Mov ax, table[bx][di]-1