


# Учебный курс. Часть 14. Режимы адресации

Автор: xrnd | Рубрика: [Учебный курс](#) | 22-04-2010 |  
 [Распечатать запись](#)

Режимы адресации — это различные способы указания местоположения операндов. До этой части в учебном курсе использовались только простые режимы адресации: операнды чаще всего находились в регистрах или в переменных в памяти. Но в процессоре Intel 8086 существуют также более сложные режимы, которые позволяют организовать работу с массивами, структурами, локальными переменными и указателями. В этой части я расскажу о всех возможных режимах адресации и приведу примеры их использования.

## 1. Неявная адресация

Местоположение операнда фиксировано и определяется кодом операции. Примеры:

```
cbw  
mul al
```

Команда [CBW](#) всегда работает с регистрами AX и AL, а у команды [MUL](#) фиксировано положение первого множителя и результата. Такой режим адресации делает машинную команду короткой, так как в ней отсутствует указание одного или нескольких операндов.

## 2. Непосредственная адресация

При непосредственной адресации значение операнда является частью машинной команды. Понятно, что в этом случае операнд представляет собой константу. Примеры:

```
mov al,5  
add bx,1234h  
mov dx,a
```

Обратите внимание, что в третьей строке в DX помещается *адрес* метки или переменной *a*, а вовсе не значение по этому адресу. Это особенность синтаксиса FASM. По сути адрес метки тоже является числовой константой.

### 3. Абсолютная прямая адресация

В машинной команде содержится адрес операнда, находящегося в памяти. Пример:

```
mov dx,[a]
```

Вот тут уже в DX помещается значение из памяти по адресу *a*. Сравните с предыдущим пунктом. Квадратные скобки обозначают обращение по адресу, указанному внутри этих скобок.

### 4. Относительная прямая адресация

Этот режим используется в командах передачи управления. В машинной команде содержится смещение, которое прибавляется к значению указателя команд IP. То есть указывается не сам адрес перехода, а на сколько байтов вперед или назад надо перейти. Пример:

```
metka:  
...  
loop metka
```

У такого режима адресации два преимущества. Во-первых, машинная команда становится короче, так она содержит не полный адрес, а только смещение. Во-вторых, такой код не зависит от адреса, по которому он размещается в памяти.

## 5. Регистровая адресация

Операнд находится в регистре. Пример:

```
add ax, bx
```

## 6. Косвенная регистровая (базовая) адресация

Адрес операнда находится в одном из регистров BX, SI или DI. Примеры:

```
add ax, [bx]  
mov dl, [si]
```

Размер операнда в памяти здесь определяется размером первого операнда. Так как AX — 16-разрядный регистр, то из памяти берётся слово по адресу в BX. Так как DL — 8-разрядный регистр, то из памяти берётся байт по адресу в SI. Это правило верно и для других режимов адресации.

## 7. Косвенная регистровая (базовая) адресация со смещением

Адрес операнда вычисляется как сумма содержимого регистра BX, BP, SI или DI и 8- или 16-разрядного смещения. Примеры:

```
add ax, [bx+2]  
mov dx, [array1+si]
```

В качестве смещения можно указать число или адрес метки. О размере смещения не беспокойтесь — компилятор сам его

определяет и использует нужный формат машинной команды.

## 8. Косвенная базовая индексная адресация

Адрес операнда вычисляется как сумма содержимого одного из базовых регистров ВХ или ВР и одного из индексных регистров SI или DI. Примеры:

```
mov ax,[bp+si]  
add ax,[bx+di]
```

Например, в одном из регистров может находиться адрес начала массива в памяти, а в другом — смещение какого-то элемента относительно начала. А вообще, всё зависит от вашей фантазии 😊

## 9. Косвенная базовая индексная адресация со смещением

Адрес операнда вычисляется как сумма содержимого одного из базовых регистров ВХ или ВР, одного из индексных регистров SI или DI и 8- или 16-разрядного смещения. Примеры:

```
mov al,[bp+di+5]  
mov bl,[array2+bx+si]
```

## Пример программы

Допустим, имеется массив 32-битных целых чисел со знаком. Количество элементов массива хранится в 16-битной переменной без знака. Требуется вычислить среднее арифметическое элементов массива и сохранить его в 32-битной переменной со знаком. Я намеренно использовал разные режимы адресации, хотя тоже самое можно написать проще.

```

1  use16                                ;генерировать 16-битный код
2  org 100h                             ;Программа начинается с адреса 100h
3
4      sub ax,ax                         ;AX = 0
5      cwd                               ;DX = 0
6      mov si,ax                         ;SI = 0 - смещение элемента от начала
7      mov bx,array                      ;Помещаем в BX адрес начала массива
8      mov di,n                          ;Помещаем в DI адрес n
9      mov cx,[di]                       ;CX = n
10 lp1:
11      add ax,[bx+si]                   ;Прибавление младшего слова
12      adc dx,[bx+si+2]                 ;Прибавление старшего слова
13      add si,4                          ;Увеличиваем смещение в SI на 4
14      loop lp1                         ;Команда цикла
15
16      idiv word[di]                    ;Делим сумму на количество элементов
17      cwd                               ;DX:AX = AX
18      mov word[m],ax                   ;\ Сохраняем
19      mov word[m+2],dx                 ;/ результат
20
21      mov ax,4C00h                     ;\
22      int 21h                          ;/ Завершение программы
23 ;-----
24 n      dw 10
25 array dd 10500,-7500,-15000,10000,-8000
26      dd 6500,11500,-5000,10500,-20000
27 m      dd ?

```

## Упражнение

Объявите в программе два массива 16-битных целых со знаком. Количество элементов массивов должно быть одинаковым и храниться в 8-битной переменной без знака. Требуется из последнего элемента второго массива вычесть первый элемент первого, из предпоследнего — вычесть второй элемент и т.д. Результаты можете выкладывать в комментариях.

[Следующая часть »](#)

# Комментарии:

RoverWWorm  
11-05-2010 19:58

```
use16  
org 100h
```

```
mov di,8  
mov si,0  
movzx cx,[n]
```

```
raznost:  
mov ax,[array1+si]  
mov bx,[array2+di]  
sub bx,ax  
add si,2  
sub di,2
```

```
loop raznost
```

```
mov ax,4c00h  
int 21h
```

```
;_____
```

```
n db 5  
array1 dw 2500,-4500,1200,-4300,-1111  
array2 dw -3200,5400,-1111,-3200,2222
```

[\[Ответить\]](#)

[xrnd](#)  
11-05-2010 23:52

Rover, по сути всё верно, только у тебя результат вычитания не сохраняется нигде 😊 Обращение к элементам массивов и

цикл написаны правильно. Я имел в виду, что из элементов второго массива вычитается и там остаётся результат вычитания.

Можно вместо

```
mov bx,[array2+di]
```

```
sub bx,ax
```

написать:

```
sub [array2+di],ax
```

[\[Ответить\]](#)

RoverWWWorm

12-05-2010 07:42

Да я подумал, что не обязательно результат в переменной сохранять.

Ну а так я рад, что цикл верный.

[\[Ответить\]](#)

Dim

23-11-2010 00:38

Помогите выполнить задание:

Вычисление суммы двух чисел, используя регистровую косвенную адресацию со смещением.

[\[Ответить\]](#)

[xrnd](#)

25-11-2010 02:06

Для косвенной адресации со смещением адрес в памяти будет вычисляться как сумма содержимого регистра и какого-то числа.

Например, в регистре может находиться начальный адрес массива, а смещение будет равно количеству байтов от начала массива до определенного элемента.

Не знаю точно вашего задания, но приведу пример сложения с использованием такой адресации. Пусть есть массив байтов и надо вычислить сумму 2-го и 3-го элемента (считая с нуля) и сохранить её на место 4-го элемента.

Складывать два числа в памяти процессор не умеет, поэтому надо сначала загрузить первый операнд из памяти, потом прибавить второй и затем сохранить результат в память.

```
;Массив из 6 байтов  
arr db 5,7,8,-2,4,0  
  
;Код  
...  
mov bx,arr      ;BX=адрес начала массива в памяти  
mov al,[bx+2]   ;AL=2-й элемент массива  
add al,[bx+3]   ;Прибавление 3-го элемента массива  
mov [bx+4],al   ;Сохранение результата  
...
```

[\[Ответить\]](#)

argir

05-12-2010 18:05

У меня получилось так:

```
use16  
org 100h  
mov al,[n]  
mov bl,2  
mul bl  
mov si,ax ;смещение на конец массива  
mov di,0 ;нет смещения  
movzx cx,[n]
```



```
lp1:
sub si,2
mov ax,[array2+si]
mov bx,[array1+di]
sub ax,bx
mov word[array3+di],ax ;сохраняем результат в третьем
массиве
add di,2
loop lp1
```

```
mov ax,4C00h
int 21h
```

```
;_____
```

```
n db 10
```

```
array1 dw 10500,-7500,-15000,10000,-8000
dw 6500,11500,-5000,10500,-20000
```

```
array2 dw 511,12345,-789,-987,6543,5555,222,-9988,5543,-100
```

```
array3 dw ?
```

[\[Ответить\]](#)

[xrnd](#)

06-12-2010 13:37

Всё правильно, хорошая программа.

Можно немного оптимизировать — для вычитания не обязательно помещать второй операнд в регистр. Вместо

```
mov bx,[array1+di]
sub ax,bx
```

можно вычитать сразу:

```
sub ax,[array1+di]
```

[\[Ответить\]](#)

argir

06-12-2010 23:31

Спасибо. Учту.

[\[Ответить\]](#)

[Борис](#)

22-12-2010 23:03

Добрый вечер, спасибо за уроки, вот мое решение,

```
use16 ;gen 16-bit cod
```

```
org 100h ;begin 100h
```

```
;-----Init RON-----
```

```
mov bl, 2
```

```
mov al, [n]
```

```
mul bl
```

```
sub ax, 2
```

```
mov si, ax
```

```
mov di, 0
```

```
;-----Loop-----
```

```
movzx cx,[n]
```

```
loop_array:
```

```
mov ax,[ar2+si]
```

```
sub ax,[ar1+di]
```

```
mov [res+di], ax
```

```
add di, 2
```

```
sub si, 2
```

```
loop loop_array
```

```
exit_progr:
```

```
mov ax,4C00h ;\
```

```
int 21h ;/ end
```

```
;-----
```

```
n db 6
```

```
ar1 dw 10,-400,60,1,0,8
ar2 dw 0,8,300,0,18,500
res rw 6
```

[\[Ответить\]](#)

[xrnd](#)

23-12-2010 19:56

Хорошее решение. Написано без ошибок.

Есть более быстрые способы умножения на 2, чем командой MUL. Лучше всего, конечно, использовать сдвиг влево на 1 бит. Но об этом написано дальше, в [18-й части](#).  
Ещё можно сложить число с самим собой 😊

```
movzx ax,[n]
add ax,ax    ;AX = 2 * AX
```

[\[Ответить\]](#)

Гость

13-01-2011 19:39

```
use16
org 100h
movzx cx,[n] ; количество элементов
movzx ax,[n] ;
mov si,2 ; смещение
mul si ; смещение в конец массива +2
mov si, ax; si = смещение в конец массива +2
х:
;на первом шаге цикла
mov di,cx ;cx=5 ; сдвиг с шагом 1
add di,cx ;di=5+5=10 ; сдвиг с шагом 2
mov ax,[array2+di-2]; , последней элемент имеет смещение +8
```

```
mov bx,si; сохраняем si , так как она изменяется
sub si,di ;si-di =0 ; первый элемент
mov dx,[array1+si]; здесь si не дойдёт до 10 по этому нет -2
sub ax, dx ; из последнего элемента второго массива вычесть
первый элемент ;первого и сохраняем в ax
mov [array1+si],ax ; по идеи наверно запишет результат в
первый элемент масива
mov si,bx ;востонавливаем si
loop x
mov ax,4C00h
int 21h
;-----
n db 5
array1 dw 1001,1002,1003,1004,-1005
array2 dw 1005,1005,1005,1005,-1005
```

[\[Ответить\]](#)

[xrnd](#)

15-01-2011 01:43

Программа написана правильно, хотя немного коряво 😊

Значение в DI удваивается сложением, а в SI — умножением.  
Лучше использовать сложение.

Можно было в цикле увеличивать значение SI на 2 и уменьшать значение DI тоже на 2. У тебя какое-то сложное вычисление смещений — на каждом шаге цикла считается заново, да ещё с сохранением SI.

[\[Ответить\]](#)

A13R42

23-02-2011 16:19

Привет!

Блин, опять у меня коряво получилось(( но вроде бы работает  
;\_\_\_\_\_

```
use16
```

```
org 0x100
```

```
mov si,0
```

```
movzx cx,[n]
```

```
movzx bx,[n]
```

```
add bx,bx
```

```
lp:
```

```
mov ax,[fstarr+si]
```

```
mov dx,[scdarr+bx-2]
```

```
sub ax,dx
```

```
mov word[m+si],ax
```

```
add si,2
```

```
sub bx,2
```

```
loop lp
```

```
mov ax,0x4c00
```

```
int 0x21
```

```
;_____
```

```
n db 5
```

```
fstarr dw 5,6,7,3,-4
```

```
scdarr dw -56,3,55,1,0
```

```
m dw ?
```

[\[Ответить\]](#)

[xrnd](#)

25-02-2011 22:41

Почему коряво 😊 Хорошо получилось, ошибок нет.

Хотя неплохо бы для массива m зарезервировать 10 байт а не 2.

Вычитать значение можно без предварительной загрузки в

регистр:  
sub ax,[scdarr+bx-2]

[\[Ответить\]](#)

annihilator  
03-05-2011 15:04

use16 ;Генерировать 16-битный код  
org 100h ;Программа начинается с адреса 100h

```
mov di, 0
movzx si, [n]
mov cx, si
dec si
add si, si
```

```
lb:
mov ax, word[a+si]
sub word[b+di], ax
sub si, 2
add di, 2
loop lb
```

```
mov ax,4C00h ;\
int 21h ;/ Завершение программы
;-----
n db 5
a dw 10500,-7500,-15000,10000,-8000
b dw 6500,11500,-5000,10500,-20000
```

[\[Ответить\]](#)

[xrnd](#)  
06-05-2011 00:25

Решение правильное 😊

Для команд mov и sub можно не указывать размер операнда (word).

Он и так понятен, поскольку другой операнд — 16-битный регистр AX.

[\[Ответить\]](#)

annihilator

07-05-2011 14:07

а ну да, точно.... привычка...

[\[Ответить\]](#)

vanilla\_ratty

11-08-2011 15:32

Приветствую тебя, о, великий Гуру! Помоги найти спасение заблудшей в ассемблере душе. Проверь задание по теме. Это вроде как компилироваться соглашается, но почему-то есть ощущение, что налепил ошибок. И ещё вопрос не по теме: тэгом/ВВ-кодом 'PRE' можно пользоваться, чтобы кучей пробелов выравнивать текст, как в блокноте?

use16

org 100h

movzx di,[n]

mov si,0

mov cx,di

ccl:

mov ax,[array1+si]

mov bx,[array2+di]

sub bx,ax

mov [array2],bx

inc si

```
dec di  
loop ccl
```

```
mov ax,4c00h  
int 21h
```

```
n db 4  
array1 dw 3334,-4445,222,-7000  
array2 dw 8889,-7654,555,-3000
```

[\[Ответить\]](#)

[xrnd](#)

17-09-2011 16:56

Ахахаха!

Молись, постись, программируй на ассемблере.

Здесь работает тэг `<pre lang="asm">`.

Ошибка в том, что `dw` занимает в памяти 2 байта, а `INC/DEC` прибавляет/вычитает единицу.

Надо писать:

```
add si,2  
sub di,2
```

Изначально смещение в `DI` тоже задано неправильно.

Нулевой элемент имеет смещение 0 от начала массива.

Первый элемент имеет смещение 2.

Второй элемент — 4

Третий последний элемент — его адрес на 6 больше, чем адрес начала массива.

Нужно в `DI` поместить  $(n-1)*2$ . Например, так:

```
movzx cx,[n]  
mov di,cx
```



```
dec di
add di,di ; x2
```

[\[Ответить\]](#)

алекс

04-03-2012 19:50

use16

org 100h

sub ax,ax ;обнуляем ax

mov bx,array1 ;помещаем в bx адрес начала первого массива

mov bp,array2 ;помещаем в bp адрес начала второго массива

movzx cx,[n] ;помещаем в счетчик цикла n

mov si,8 ;начальное смещение второго массива

mov di,ax ;начальное смещение первого массива

cycle:

mov ax,[bp+si] ;в ax последний элемент второго массива

sub ax,[bx+di] ;вычитаем их ax первый элемент первого м.

mov [array3+si],ax ;сохраняем результат

sub si,2 ;уменьшаем смещение для второго массива

add di,2 ;увеличиваем смещение для первого масс.

loop cycle

mov ax,4c00h

int 21h ;завершение программы

array1 dw 10,15,20,25,30

array2 dw 60,50,40,30,20

array3 rw 5

n db 5

[\[Ответить\]](#)

Ренат

15-03-2012 22:40

Почему абсолютно нигде не написано когда используются непосредственная, прямая и косвенная адресации.

[\[Ответить\]](#)

Марина

05-11-2014 09:09

Описать каждую строчку(что она означает) и определить условие задачи...Пожалуйста

a 100

mov si,200

mov di,210

mov bx,220

mov dx,0

F:mov cx,10

mov dl,[di]

mov al,[si]

cbw

cmp ah,al

jA L

mov [di],al

mov al,0

mov [bx],si

inc bx

inc bx

inc dx

L:inc si

inc di

loop F

hlt

[\[Ответить\]](#)

Sam

14-01-2015 03:32

;Не подскажите, как оптимизировать программу?)

use16 ;Генерировать 16-битный код

org 100h ;Программа начинается с адреса 100h

mov si,0 ; Итератор

movzx cx,[n] ; Счетчик цикла

mov di,cx ; Смещение массива b+1

add di,di

l1:

mov bx,b

mov ax,[bx+di-2]

mov bx,a

sub ax,[bx+si]

mov bx,x

mov [bx+si],ax

add si,2

loop l1

mov ax,4C00h ;\

int 21h ;/ Завершение программы

;

---

n db 5

a dw 1,2,3,4,5

b dw 10,10,10,10,10

x dw 5 dup(?)

[\[Ответить\]](#)

Sam

14-01-2015 03:36

Цикл следует немного подкорректировать:

```
.....  
add si,2  
sub di,2 // изменения  
loop l1  
.....
```

[\[Ответить\]](#)

Александр

27-10-2016 00:55

```
use16;  
org 100h  
mov di,array1  
mov si,array2  
mov bx,[si+6]  
mov cl,[n]  
lp:  
sub bx,[di]  
mov word[arrey3],bx  
sub si,2  
add di,2  
loop lp
```

```
mov ax, 4C00h
```

```
int 21h
```

```
;  
_____
```

```
n db 3
```

```
array1 dw 2222,3333,-4444
```

```
array2 dw -6666,-7777,8888
```

```
arrey3 rw 3
```

[\[Ответить\]](#)

Александр  
27-10-2016 01:18

Андрей. скажи пожалуйста — могут ли регистры SI и DI, в которые помещается адрес массива, являться одновременно счетчиком?

PS: mov bx,[si+6] нужно в начало цикла поместить. наверное.

[\[Ответить\]](#)

Александр  
27-10-2016 23:20

```
use16;
org 100h
mov si,array2
mov di,array1
mov cl,[n]
```

```
lp: mov bx,[si+4]
sub bx,[di]
mov word[arrey3],bx
sub si,2
add di,2
loop lp
```

```
mov ax, 4C00h
int 21h
;_____
```

```
n db 3
array1 dw 2222,3333,-4444
array2 dw -6666,-7777,8888
arrey3 rw 3
```

Все! 😊 Разобрался. Окончательный вариант. В турбодебагере все сходится. Спасибо!

[\[Ответить\]](#)

limbo

17-12-2016 13:50

use16

org 100h

mov bx,massive

mov si,0

sub dx,dx

mov cx,0005h

cycle:

add dx,cx

loop cycle

mov ax,4c00h

int 21h

massive dw 1,2,3,4,5

[\[Ответить\]](#)

fast rivet

17-12-2016 16:05

use16

org 100h

mov cx,0005h

mov si,0000h

lop:

mov di,cx

mov ax,[mass2 + di]

sub ax,[mass1 + si]

mov bx,mass3

add bx,si

```
mov bx,ax
inc si;
loop lop
mass1 dw 14,26,3,46,5
mass2 dw 1,2,3,47,58
mass3 dw 5 dup(0)
```

[\[Ответить\]](#)

Елизар  
12-12-2017 05:03

Опять подниму комментарии. Я всё ещё продвигаюсь  
Огромное спасибо за ваши уроки =)

```
use16 ;Генерировать 16-битный код
org 100h ;Программа начинается с адреса 100h
movzx ax,[n]
mov cx,2
mul cx
mov cx,ax
mov si,0
lp:
dec cx
mov di,cx
dec di
mov dx,[arr1+si]
sub dx,[arr2+di]
mov [arr1+si],dx
add si,2
loop lp

mov ax,4C00h ;\
int 21h ;/ Завершение программы
;_____
arr1 dw 1,2,3,4,5,6,7,8
```

arr2 dw 1,2,3,4,5,6,7,8  
n db 8


[\[Ответить\]](#)

## Ваш комментарий

Имя \*

Почта (скрыта) \*

Сайт



**Добавить**

☐ Уведомить меня о новых комментариях по email.

☐ Уведомлять меня о новых записях почтой.