**Direct-offset Operands:**

You can add a displacement to the name of a variable, creating a direct-offset operand.

**Example:**

.data

arrayB BYTE 10h,20h,30h,40h,50h

.code

Mov al,arrayB ; AL = 10h

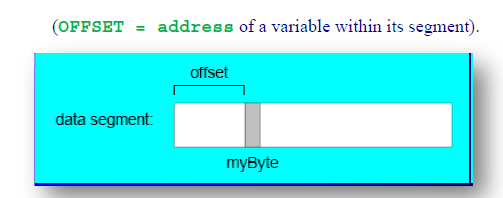
Mov al,[arrayB+1] ; AL = 20h

Mov al,[arrayB+2] ; AL = 30h

**OFFSET Operator:**

The OFFSET operator returns the offset of a data label.

return the distance of variable from the start of the segment



**OFFSET :**

If a street is a base address, the number of a house in the street is the offset.  
Number 21 is the 21st house on the street.

**Syntax:**

MOV reg32, OFFSET mem; reg32 points to count

Example:

.data

bVal BYTE ?

wVal WORD ?

dVal DWORD ?

dVal2 DWORD ?

• If bVal is located at offset 00404000h, we would get:

Mov esi, OFFSET bval ; ESI = 00404000

Mov esi, OFFSET wVal ; ESI = 00404001

Mov esi, OFFSET dVal ; ESI = 00404003

Mov esi, OFFSET dVal2 ; ESI = 00404007

**ALIGN DIRECTIVES**

The ALIGN directive aligns a variable on a byte, word, double word, or paragraph boundary.

**Syntax**:

ALIGN bound (where bound is either 1, 2 or 4)

**example**

.DATA

bVal BYTE ? ; 00404000

ALIGN 2

wVal WORD ? ; 00404002

bVal2 BYTE ? ; 00404004

ALIGN 4

dVal DWORD ? ; 00404008

dVal2 DWORD ? ; 0040400C

**.CODE**

**MAIN PROC**

MOV EAX,OFFSET bVal

MOV EBX,OFFSET wVal

MOV ECX,OFFSET bVal2

MOV EDX,OFFSET dVal

MOV EDI,OFFSET dVal2

**PTR Operator:**

You can use the PTR operator to override the declared size of an operand.

Assembly instructions require operands to be the same size. However, it may be required at

some point to operate on data in a size other than that originally declared. This can be done

with the PTR operator (override a variable's default size).

val32 DWORD 12345678h

Suppose want to move 5678 to AX?

o Can NOT do mov ax, val32 ; since mismatched sizes

**Example:**

.data

val32 DWORD 12345678h

.code

mov ax, word ptr val32 ; AX = 5678H

mov dx, word ptr val32+2 ; DX = 1234H

**Example:**

.DATA

num DWORD 12345678H

.CODE

MAIN PROC

mov eax,0

mov ebx,0

mov ecx,0

mov edx,0

mov ax, WORD PTR [num] ; 5678

mov Bx, WORD PTR [num+1] ; 3456

mov Cx, WORD PTR [num+2] ; 1234

mov Dx, WORD PTR [num+3] ; 0012

mov bl, BYTE PTR NUM ; 78

**Can use it to move smaller to larger too…**

**Example**

.data

wordList WORD 5678h, 1234h

.code

moveax, DWORD PTR wordlist ;12345678

**TYPE Operator:**

The TYPE operator returns the size, in bytes, of a single element of a variable.

**Syntax:**

MOV reg16, TYPE mem

**Example 1:**

.data

var1 BYTE ? ; TYPE var1 = 1

var2 WORD ? ; TYPE var2 = 2

var3 DWORD ? ; TYPE var3 = 4

var4 QWORD ? ; TYPE var4 = 8

**Example 2:**

.data

var1 BYTE 20h

var2 WORD 1000h

var3 DWORD ?

var4 BYTE 10, 20, 30, 40, 50

msg BYTE ‘File not found’, 0

.code

Mov ax, type var1 ; AX = 0001

Mov ax, type var2 ; AX = 0002

Mov ax, type var3 ; AX = 0004

Mov ax, type var4 ; AX = 0001

movax, type msg ; AX = 0001

**LENGTHOF Operator:**

The LENGTHOF operator counts the number of individual elements in a variable that has been defined using DUP.

**Syntax:**

MOV reg16 , LENGTHOF mem

**Example:**

.data

val1 WORD 1000h

val2 SWORD 10, 20, 30

array WORD 32 DUP(0)

array2 WORD 5 DUP(3 DUP(0))

message BYTE ‘File not found’, 0

.code

movax, LENGTHOF val1 ; AX = 1

movax, LENGTHOF val2 ; AX = 3

movax, LENGTHOF array ; AX = 32

movax, LENGTHOF array2 ; AX = 15

movax, LENGTHOF message ; AX = 15

**SIZEOF Operator:**

The SIZEOF operator returns the number of bytes an array takes up. It is similar in effect to multiplying LENGTHOF with TYPE.

**Syntax:**

MOV reg16/32 , SIZEOF mem

**Example:**

intArray WORD 32 DUP(0) ; SIZEOF = 64

.data

intArray WORD 32 DUP(0)

.code

MAIN PROC

Mov eax,SIZEOFintArray ; returns 64 = 32 \* 2

**Example**

.data

array\_1 WORD 40 DUP (5)

num DWORD 4, 5, 6, 7, 8, 9, 10, 11

warray WORD 40 DUP (40 DUP (5))

.code

MAIN PROC

Mov eax,SIZEOF array\_1 ;40\*2=80

Mov eBx,SIZEOF num ;8\*4=32

Mov eCx,SIZEOF warray ;40\*40\*2 =3200

**Example:**

.data

array\_1 BYTE 4 DUP ("STACK")

warray BYTE 40 DUP (40 DUP ("STACK"))

.code

MAIN PROC

Mov eax,SIZEOF array\_1 ; 4\*5=20

Mov eCx,SIZEOF warray ;40\*40\*5\*1=8000