BİL 362 Mikroişlemciler: PTR, OFFSET işleçleri ve Dizi Adresleme

Ahmet Burak Can
abc@hacettepe.edu.tr

PTR İşleci - 1

Bir veri etiketinin veri tipinin üzerine yazar.

Bir verinin bölümlerine ulaşmada esneklik sağlar.

```
.data
myDouble DWORD 12345678h

.code
mov ax, myDouble                        ; hata – neden?
mov ax, WORD PTR myDouble               ; 5678h yükler
mov WORD PTR myDouble, 4321h            ; 4321h saklar
```


PTR İşleci: Örnek

```
.data
myDouble DWORD 12345678h
```

Çift-sözcük	Sözcük	Bayt	Ofset	
		12	0003	myDouble+ 3
	1234	34	0002	myDouble + 2
		56	0001	myDouble + 1
12345678	5678	78	0000	myDouble

```
mov al, BYTE PTR  myDouble              ; AL = 78h
mov al, BYTE PTR [myDouble+1]           ; AL = 56h
mov al, BYTE PTR [myDouble+2]           ; AL = 34h
mov ax, WORD PTR  myDouble              ; AX = 5678h
mov ax, WORD PTR [myDouble+2]           ; AX = 1234h
```


PTR İşleci - 2

PTR işleci aynı zamanda, küçük tipte tanımlanmış verileri birleştirmek ve daha büyük bir işlenene taşımak için de kullanılır.

İşlemci baytları otomatik olarak ters çevirir.

```
.data
myBytes BYTE 12h,34h,56h,78h

.code
mov ax, WORD PTR [myBytes]              ; AX = 3412h
mov ax, WORD PTR [myBytes+2]            ; AX = 7856h
mov eax, DWORD PTR myBytes              ; EAX = 78563412h
```

Alıştırma

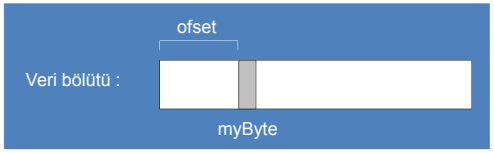
Aşağıdaki işlemlerde her hedef işlenenin değerini gösterin.

```
.data
varB BYTE 65h,31h,02h,05h
varW WORD 6543h,1202h
varD DWORD 12345678h

.code
mov ax, WORD PTR [varB+2]          ; 0502h
mov bl, BYTE PTR varD              ; 78h
mov bl, BYTE PTR [varW+2]          ; 02h
mov ax, WORD PTR [varD+2]          ; 1234h
mov eax, DWORD PTR varW            ; 12026543h
```


OFFSET İşleci - 1

- OFFSET işleci, bir veri etiketinin, ait olduğu bölütün başlangıcına olan uzaklığını, bayt olarak döndürür.



OFFSET İşleci: Örnek

```
.data
bVal BYTE ?
wVal WORD ?
dVal DWORD ?
dVal2 DWORD ?

.code
mov si, OFFSET bVal          ; SI = 0000H
mov si, OFFSET wVal          ; SI = 0001H
mov si, OFFSET dVal          ; SI = 0003H
mov si, OFFSET dVal2         ; SI = 0007H
```


OFFSET İşleci - 2

OFFSET ile döndürülen değer bir imleçtir (“pointer”).

C++

```
; C++ version:
char array[100];
char *p = array;
```

Assembly

```
.data
array BYTE 100 DUP(?)

.code
mov  si, OFFSET array            ; SI is p
```

Dolaylı İşlenenler (“Indirect Operands”) – 1

Bir dolaylı işlenen, dizi veya dizgi olarak tanımlanmış değişkenin adresini tutar.

İmleç (“pointer”) gibidir; ters-referans edilebilir.

```
.data
val1 BYTE 10h,20h,30h

.code
mov si, OFFSET val1
mov al, [si]                    ; ters-referans SI (AL = 10h)

inc si
mov al, [si]                    ; AL = 20h

inc si
mov al, [si]                    ; AL = 30h
```


Dolaylı İşlenenler - 2

Dolaylı adresleme yaparken, bir bellek işleneninin büyüklüğünü net olarak alabilmek için, PTR işlecini kullanmak gerekir.

```
.data
myCount WORD 0

.code
mov si, OFFSET myCount
inc [si]              ; hata: myCount iki bayt uzunlugunda
                      ; bir bayt uzerinde arttirma yeterli olmaz

inc WORD PTR [si]     ; tamam
```


Örnek: Dizi Toplama

Dolaylı işlenenler bir diziyi çevirmek için kullanılabilir.

(Köşeli parantez içindeki yazmacın, dizi tipine uyan bir değerle arttırılması gerekir.)

```
.data
arrayW WORD 1000h,2000h,3000h
.code
    mov si, OFFSET arrayW
    mov ax, [si]
    add si, 2                   ; veya: add si, TYPE arrayW
    add ax, [si]
    add si, 2
    add ax, [si]                ; AX = dizinin toplami
```


Dizinli İşlenenler (“Indexed Operands”)

Bir dizinli işlenen, etkin adresi (“effective address” – EA) oluşturmak için; bir sabiti bir yazmaca ekler.

`[etiket + yazmaç]` *veya* `etiket[yazmaç]`

```
.data
arrayW WORD 1000h,2000h,3000h

.code
    mov si, 0
    mov ax, [arrayW + si]           ; AX = 1000h
    mov ax, arrayW[si]              ; alternatif biçim
    add si, 2
    add ax, [arrayW + si]
```

Dizin Ölçekleme (“Index Scaling”)

Bir dolaylı veya dizinli işlenen, dizi elemanının ofsetine ölçeklenebilir.

Dizin değeri (“index”), dizinin elemanlarının boyu ile çarpılır.

```
.data
arrayB BYTE  0,1,2,3,4,5
arrayW WORD  0,1,2,3,4,5
arrayD DWORD 0,1,2,3,4,5

.code
mov si,4
mov al, arrayB[si * 1]              ; 04
mov bx, arrayW[si * 2]              ; 0004
mov edx, arrayD[si * 4]             ; 00000004
```