BİL 362 Mikroişlemciler: 8086 Çevirici Diline Giriş

Ahmet Burak Can
abc@hacettepe.edu.tr

Çalıştırılabilir Programlar

- 8086 mimarisinde iki tür çalıştırılabilir program vardır:
  - .COM programı: Kod, veri ve yığıtı içeren tek bir bölütten oluşur.
    - Küçük yardımcı program (“utility program”) veya yerleşik program (“resident program”) olarak kullanıma uygundur.
  - .EXE programı: Kod, veri ve yığıt için ayrı bölütlerden oluşur.
    - Daha ciddi ve büyük kapsamlı programlar için kullanıma uygundur.

Assembly Dilinin Temel Bileşenleri

- Sabitler ve ifadeler (“constants and expressions”)
- Açıklamalar (“comments”)
- Ayrılmış sözcükler (“reserved words”)
- Belirteçler (“identifiers”)
- Deyimler (“statements”)
- Anımsatıcılar (“mnemonics”) ve işlenenler (“operands”)
- Derleyici komutları (“directives”)

Tamsayı Sabitler

[{ + | - }] *rakamlar* [*taban*]

- İkili, onlu, onaltılı veya sekizli rakamlar
- Taban karakterleri:
  - h – onaltılı (“hexadecimal”)
  - d – onlu (“decimal”)
  - b – ikili (“binary”)
  - o – sekizli (“octal”)
  - r – kodlanmış gerçel (“encoded real”)
- Örnekler:
  - 30d, 6Ah, 42, 1101b
  - Harf ile başlayan onaltılının önüne “0” konur: 0A5h

Tamsayı İfadeler

- İşleçler ve öncelikleri:

Operator	Name	Precedence Level
( )	parentheses	1
+,-	unary plus, minus	2
*,/	multiply, divide	3
MOD	modulus	3
+,-	add, subtract	4

- Örnekler:

Expression	Value
`16 / 5`	3
`-(3 + 4) * (6 - 1)`	-35
`-3 + 4 * 6 - 1`	20
`25 mod 3`	1

Gerçel Sabitler

[{ + | - }] *tamsayı.[tamsayı] [üs]*

- Örnekler:
  - 2.
  - +3.0
  - -44.2E + 05
  - 26.E5
- Kodlanmış gerçel (“encoded real”)
  - IEEE kayan-noktalı biçiminde, onaltılı tabanda ifade edilmiş gerçel sayıdır.
    - Örnek:

      0011 1111 1000 0000 0000 0000 0000 0000 (ikili)

      3 F 8 0 0 0 0 0 (kodlanmış gerçel)

      ⇨ 3F800000r

Karakter ve Dizgi Sabitleri

- Karakterler tek veya çift tırnak içine alınır.
  - 'A', "x"
  - ASCII karakter = 1 bayt
- Dizgiler tek veya çift tırnak içine alınır.
  - "ABC"
  - 'xyz'
  - Her karakter 1 bayt yer tutar.

Açıklamalar (“Comments”)

- Kodun anlaşılmasını kolaylaştırmak için açıklamalar program satırları içine eklenmelidir.
  - Tek-satır açıklamalar
    - Noktalı virgül (“;”) ile başlar.
      - Örnek: `ADD AX, BX ; AX = AX + BX`
  - Çok-satır açıklamalar
    - COMMENT bildirimini takip eden, programcının seçtiği bir karakterle devam eder ve yine programcının seçtiği aynı karakterle sona erer.
      - Örnek:

```
COMMENT !       Bu satır açıklamadır.
                Bu satır da açıklamadır !
```

Ayrılmış Sözcükler (“Reserved Words”)

- Komut anımsatıcılar (instruction mnemonics”): İşlemci tarafından işletilir.
  - Örnek: MOV, ADD, …
- Derleyici talimatları (“directives): Derleyiciye bilgi sağlar.
  - Örnek: END, SEGMENT, …
- Özellikler (“attributes”): İşlenenler için büyüklük ve kullanımı tanımlar.
  - Örnek: BYTE, WORD, …
- İşleçler (“operators”): İfadelerin içinde kullanılır.
  - Örnek: FAR, SIZE, …
- Öntanımlı semboller (“predefined symbols”): Programa bilgi döndürür.
  - Örnek: @data, @model, …

Belirteçler (“Identifiers”)

- Referans etmek istediğiniz program öğeleri için kullanılır.
  - İsim (“name”)
    - Örnek: **COUNTER** DB 0
  - Etiket (“label”)
    - Örnek:

      EKLE: ADD BL, 25
- 1-247 karakter (rakamlar dahil), büyük/küçük harfe duyarlı değil
- İçerebileceği karakterler:
  - Alfabetik karakterler : A…Z, a…z
  - Rakamlar (ilk karakter olamaz) : 0…9
  - Özel karakterler : Soru işareti (“?”), alt çizgi (“_”), dolar işareti (“$”), salyangoz işareti (“@”), nokta işareti (“.” ilk karakter olamaz)

Deyimler (“Statements”)

- Komut anımsatıcılar (“instruction mnemonics”)
  - Derleyici makine koduna çevirir ve işletir.
  - Örnek: MOV, ADD, …
- Talimatlar (“directives”)
  - Derleyiciye belirli bir işlemin yapılmasını bildirir; işletilmez.
  - Örnek: Veri tanımlama

Örnek:

	Belirteç	İşleç	İşlenen
Bildirim:	COUNT	DB	1
Komut:	L30:	MOV	AX, 0

Komutlar

- Assembler tarafından makine koduna derlenir.
- İşlemci tarafından koşturulur.
- *[label:] anımsatıcı işlenen(ler) [; açıklama]*
  - Etiket (“label”) -- seçimli
  - Anımsatıcı (“mnemonic”) -- zorunlu
  - İşlenen (“operand”) -- komuta bağlı olarak gerekli
  - Açıklama (“comment”) -- seçimli

Komut Anımsatıcılar ve İşlenenler

- Komut anımsatıcılar
  - Örnek: MOV, ADD, SUB, MUL, INC, DEC
- İşlenenler
  - Sabit (“constant”)
  - Sabit ifade (“constant expression”)
  - Yazmaç (“register”)
  - Bellek (veri etiketi – “data label”)

Komut Biçimleri: Örnekler

- İşlenen yoksa:
  - stc ; Elde (“Carry”) bayrağını 1 yap
- Bir işlenen varsa:
  - inc eax ; yazmaç
  - inc myByte ; bellek
- İki işlenen varsa:
  - add ebx, ecx ; yazmaç, yazmaç
  - sub myByte, 25 ; bellek, sabit
  - add eax, 36 * 25 ; yazmaç, sabit ifade

Komut İşlenenleri

Örnek:

```
wordx DW 0          ; wordx'i sözcük olarak tanımla
…
mov cx, wordx       ; wordx'in içeriğini cx yazmacına kopyala
mov cx, 25          ; cx yazmacına 25 yaz
mov cx, bx          ; bx yazmacının içeriğini cx yazmacına kopyala
mov cx, [bx]        ; bx ile adreslenen bellek içeriğini cx yazmacına kopyala
```


Derleyici Talimatları (“Directives”) - 1

- Kaynak programın derlenmesi ve listelenmesi için deyimlerdir; sadece derleme sırasında iş görürler ve makine koduna çevrilmezler.
- Assembler tarafından tanınan ve yanıt verilen komutlardır.
  - Intel komut setinin bir parçası değildir.
  - Kodu ve veri alanlarını tanımlamak, bellek modelini seçmek, yordamları tanımlamak, vb. için kullanılır.
  - Büyük/küçük harfe duyarlı değildir.
- Farklı derleyiciler (“Assemblers”) farklı bildirimlere sahiptir.

Derleyici Talimatları (“Directives”) - 2

- En yaygın kullanılan derleyici talimatları:
  - **SEGMENT**: Bölüt tanımlamak için kullanılır.
  - **PROC**: Yordam tanımlamak için kullanılır.
  - **END**: Kodun sonunu belirtir.
    - ENDP yordam sonunu, ENDS ise bölüt sonunu belirtir.
  - **ASSUME**: Derleyiciye hangi bölütün hangi amaçla kullanılacağını belirtmek için kullanılır.
    - .EXE programlarında veri bölütünün adresi DS yazmacına atanmalıdır.

Derleyici Talimatları: Örnek

```
STACK SEGMENT
        ...
STACK ENDS
;-------------------------------------------------
DATASEG SEGMENT
        ...
DATASEG ENDS
;-------------------------------------------------
CODESEG SEGMENT
MAIN    PROC    FAR
        ASSUME  SS:STACK, DS:DATASEG, CS:CODESEG
        MOV     AX, DATASEG         ; Veri bölütü DATASEG
        MOV     DS, AX

        ...                         ; Komutlar

MAIN    ENDP                        ; Yordam sonu
CODESEG ENDS                        ; Kod bölütü sonu
        END     MAIN                ; Program sonu
```


Basitleştirilmiş Bölüt Talimatları - 1

- Kod, yığıt ve veri bölütlerinin tanımlarını kolaylaştırmak için eklenmiştir.
- Öncelikle .MODEL talimatı ile kullanılacak bellek modeli seçilir.
  - Small (Kod: Tek , 64k; Veri: Tek, 64k)
  - Medium (Kod: Sınır yok; Veri: Tek, 64k)
  - Compact (Kod: Tek , 64k; Veri: Sınır yok)
  - Large (Kod: Sınır yok; Veri: Sınır yok)
  - Huge (Kod: Sınır yok; Veri: Sınır yok)
    - Huge içerisinde 64k’dan büyük değişkenler tanımlanabilir.

Basitleştirilmiş Bölüt Talimatları - 2

- Model belirlendikten sonra bölütler tanımlanır.
  - .STACK [büyüklük] : Yığıt
  - .DATA : Veri
  - .CODE : Kod
- Bölüt sonlarını belirtmeye gerek yoktur.
- ASSUME talimatı yazmaya gerek yoktur, kendisi eklenir.

Basitleştirilmiş Bölüt Talimatları: Örnek

```
.MODEL SMALL
.STACK 64                       ; Yığıt bölütünü tanımla
.DATA                           ; Veri bölütünü tanımla
FLDD DW     215
FLDE DW     125
FLDF DW     ?
;-------------------------------------------------------------------------------
.CODE                           ; Kod bölütünü tanımla
MAIN        PROC    FAR
            MOV     AX, @data   ; Veri bölütünün adresini
            MOV     DS, AX      ; DS yazmacına koy

            MOV     AX, FLDD    ; AX yazmacına 0215 değerini koy
            ADD     AX, FLDE    ; AX'e 0125'i ekle
            MOV     FLDF, AX    ; Sonucu FLDF'de sakla

            MOV     AX, 4C00H   ; Komutların sonu
            INT     21H
MAIN        ENDP                ; Yordam sonu
            END     MAIN        ; Program sonu
```


Ön-tanımlı Veri Tipleri (Tamsayı)

- BYTE, SBYTE
  - 8-bit işaretsiz ve işaretli tamsayı
- WORD, SWORD
  - 16-bit işaretsiz ve işaretli tamsayı
- DWORD, SDWORD
  - 32-bit işaretsiz ve işaretli tamsayı
- QWORD
  - 64-bit tamsayı
- TBYTE
  - 80-bit tamsayı

Ön-tanımlı Veri Tipleri (Gerçel Sayı)

- REAL4
  - 4-bayt IEEE kısa gerçel sayı
- REAL8
  - 8-bayt IEEE uzun gerçel sayı
- REAL10
  - 10-bayt IEEE genişletilmiş gerçel sayı

Veri Tanımlama İfadesi

- Veri tanımlama ifadesi, değişken için bellekte yer ayrılmasını sağlar.
- Sözdizim:

[isim] talimat ilklendirici [, ilklendirici] . . .

```
value1 BYTE 10
```

Veri Tanımlama: BYTE ve SBYTE

- 8-bit tamsayılar (veya çift karakterler) için yer açar

```
value1 DB 'A'                ; karakter sabit
value2 DB 0                  ; en kucuk isaretsiz bayt
value3 DB 255                ; en buyuk isaretsiz bayt
value4 DB -128               ; en kucuk isaretli bayt
value5 DB +127               ; en buyuk isaretli bayt
value6 DB ?                  ; ilklendirilmemis bayt
```


Bayt Dizisi Tanımlama

```
list1 DB 10,20,30,40
list2 DB 10,20,30,40
      DB 50,60,70,80
      DB 81,82,83,84
list3 DB ?,32,41h,00100010b
list4 DB 0Ah,20h,'A',22h
```


Dizgi Tanımlama - 1

- Bir dizgi, karakterlerin dizisinden oluşur.
  - Genellikle "null" ile biter ("null-terminated")
- Örnek:

```
str1 DB "Adinizi girin",0
str2 DB 'Hata: Program durduruluyor',0
str3 DB 'A','E','I','O','U'
greeting DB "BIL-220 icin yazilmis "
         DB "deneme programina hosgeldiniz.",0
```


Dizgi Tanımlama - 2

- Tek bir dizgiyi birden çok satırda tanımlamak için, her satırın sonuna virgül (",") koyulur.

```
menu DB "Hesap Kontrol",0dh,0ah,0dh,0ah,
    "1. Yeni hesap yarat",0dh,0ah,
    "2. Mevcut hesabi ac",0dh,0ah,
    "3. Hesaba kredi ver",0dh,0ah,
    "4. Hesabi borclandir",0dh,0ah,
    "5. Cikis",0dh,0ah,
    "Secim> ",0
```

Dizgi Tanımlama - 3

- Satır sonu karakter dizisi:
  - 0Dh = carriage return
  - 0Ah = line feed

```
str1 DB "Adinizi girin:    ",0Dh,0Ah
     DB "Adresinizi girin: ",0

newLine DB 0Dh,0Ah,0
```


DUP İşleci

- Bir dizi ("array") veya dizgi ("string") için yer açar.
- Sözdizimi:

  sayaç DUP (değişken)
  - *sayaç* ve *değişken* sabit veya sabit ifade olmalıdır.

```
var1 DB 20 DUP(0)            ; 20 bayt, hepsinin degeri sifir
var2 DB 20 DUP(?)            ; 20 bayt, ilklendirilmemis
var3 DB 4 DUP("STACK")       ; 20 bayt: "STACKSTACKSTACKSTACK"
var4 DB 10,3 DUP(0),20       ; 5 bayt
```


Veri Tanımlama: WORD ve SWORD

- 16-bit tamsayılar (veya çift karakterler) için yer açar.

```
word1  DW 65535              ; en büyük isaretsiz deger
word2  DW -32768             ; en kucuk isaretli deger
word3  DW ?                  ; ilklendirilmemis, isaretsiz
word4  DW "AB"               ; cift karakter
myList DW  1,2,3,4,5         ; sozcuk dizisi
array  DW  5 DUP(?)          ; ilklendirilmemis dizi
```


Veri Tanımlama: DWORD ve SDWORD

- 32-bit tamsayılar için yer açar.
  - Aşağıda verilen tanımlar MASM standardındadır.

```
val1 DWORD  12345678h            ; isaretsiz
val2 SDWORD -2147483648          ; isaretli
val3 DWORD  20 DUP(?)            ; isaretsiz dizi
val4 SDWORD -3,-2,-1,0,1         ; isaretli dizi
```

Veri Tanımlama: QWORD, TBYTE, Gerçel Sayı

- Dört-sözcük (“quadword”), on-bayt (“tenbyte”) ve gerçel sayılar için yer açar .
  - Aşağıda verilen tanımlar MASM standardındadır.

```
quad1 QWORD  1234567812345678h
val1  TBYTE  100000000123456789Ah
rVal1 REAL4  -2.1
rVal2 REAL8  3.2E-260
rVal3 REAL10 4.6E+4096
ShortArray REAL4 20 DUP(0.0)
```


EQU Talimatı

- Bir sembolü tamsayı veya metin ifade olarak tanımlar.
  - Tekrar tanımlanamaz.

```
Count EQU 100
PI EQU <3.1416>
pressKey EQU <“Devam etmek için bir tusa basin...",0>
.data
prompt BYTE pressKey
```