Kifejezés

A kifejezés szimbólumokból és konstansokból épül fel az alább ismertetendő műveletek segítségével. Kifejezés az utasítások, pszeudo utasítások operandus részére írható.

A kifejezés **értékét a fordítóprogram határozza meg**, és az utasítás a kiszámított értéket alkalmazza operandusként.

Szimbólumok értéke konstansok esetében természetesen a konstans értéke, címkék, változók esetében a hozzájuk tartozó cím – és nem a cím tartalma!

Kifejezés

A kifejezés értéke nemcsak számérték lehet, hanem minden, ami az utasításokban megengedett címzési módok valamelyikének megfelel. Pl. **[BX]** is kifejezés, és értéke a **BX** regiszterrel történő indirekt hivatkozás, ehhez természetesen a fordító programnak nem kell ismernie **BX** értékét (**BX** tartalmát).

Természetesen előfordulhat, hogy egy kifejezés egyik szintaktikus helyzetben megengedett, a másikban nem, pl.:

```
mov     ax,[BX] ; [BX] megengedett
mul     [BX]    ; [BX] hibás, de
mul     WORD PTR [BX]  ; megengedett
```

Egy kifejezés akkor **megengedett**, ha az értéke **fordítási időben meghatározható, és az adott szintaktikus helyzetben alkalmazható**, azaz az adott utasítás lehetséges címzési módja megengedi.

A megengedett kifejezés értékeket az egyes utasítások ismertetése során megadtuk.

A műveletek csökkenő precedencia szerinti sorrendben:

1. **( )** és **[ ]** (zárójelek) továbbá **< >**: míg a **( )** zárójel pár a kifejezés kiértékelésében csupán a műveletek sorrendjét befolyásolja, addig a **[ ]** az indirekció előírására is szolgál. Ha a **[ ]** -en belüli kifejezésre nem alkalmazható indirekció, akkor a **( )** -lel egyenértékű;
   - **LENGTH változó**: a **változó**-hoz tartozó adat terület elemeinek száma;
   - **SIZE változó**: a **változó**-hoz tartozó adat terület hossza byte-okban;
   - **WIDTH R/F**: az **R** rekord vagy az **F** (rekord) mező szélessége bitekben;
   - **MASK F**: az **F** (rekord) mező bitjein **1**, másutt **0**;

LENGTH változó: a **változó**-hoz tartozó adat terület elemeinek száma; pl.:

```
v       dw      20 dup (?)
rec     record  x:3,y:4
table   dw      10 dup (1,3 dup (?))
str     db      "12345"
```

esetén:

```
        mov     ax,LENGTH v       ; ax ⇐ 20
        mov     ax,LENGTH rec     ; ax ⇐ 1
        mov     ax,LENGTH table   ; ax ⇐ 10
                          ; a ( ) belseje ignorálva!
        mov     ax,LENGTH str     ; ax ⇐ 1
                          ; str egy elem
```


SIZE változó: a **változó**-hoz tartozó adat terület hossza byte-okban; pl.:

```
v       dw      20 dup (?)
rec     record  x:3,y:4
table   dw      10 dup (1,3 dup (?))
str     db      "12345"
```

esetén:

```
        mov     ax,SIZE v         ; ax ⇐ 40
        mov     ax,SIZE rec       ; ax ⇐ 1
        mov     ax,SIZE table     ; ax ⇐ 20
                          ; a ( ) belseje ignorálva!
        mov     ax,SIZE str       ; ax ⇐ 1
                          ; str bájtos
```

WIDTH R/F: az **R** rekord vagy az **F** (rekord) mező szélessége bitekben, **MASK F**: az **F** (rekord) mező bitjein **1**, másutt **0** ; pl.:

```
v       dw      20 dup (?)
rec     record  x:3,y:4
table   dw      10 dup (1,3 dup (?))
str     db      "12345"
```

esetén:

```
        mov     ax,WIDTH rec    ; ax ⇐ 7
        mov     ax,WIDTH x      ; ax ⇐ 3
        mov     ax,MASK x       ; ax ⇐ 70H
```


2. **.** (pont): struktúra mezőre hivatkozásnál használatos;

3. **:** mező szélesség (rekord definícióban) és explicit szegmens megadás (segment override prefix). Az explicit szegmens megadás az automatikus szegmens regiszter helyett más szegmens regiszter használatát írja elő, pl.:

```
        mov     ax, ES:[BX]
                ; ax ⇐ (ES:BX) címen lévő szó
```

Nem írható felül az automatikus szegmens regiszter az alábbi esetekben:

- **CS** program memória címzésnél,
- **SS** stack referens utasításokban (**PUSH**, **POP**, **...**),
- **ES** string kezelő utasításban **DI** mellett, de az **SI**-hez tartozó **DS** átírható.

4.

- **típus PTR cím:** (típus átdefiniálás) ahol **típus** lehet **BYTE**, **WORD**, **DWORD**, **QWORD**, **TBYTE**, illetve **NEAR**, **FAR** (előre hivatkozás esetén fontos) és **PROC**. Pl.:

```
        MUL     BYTE PTR [BX]  ; a [BX] címet
                               ; byte-osan kell kezelni
```

- **OFFSET kifejezés:** a **kifejezés OFFSET** címe (a szegmens kezdetétől számított távolsága byte-okban);
- **SEG kifejezés:** a **kifejezés** szegmens címe (abban az értelemben, ahogy a szegmens regiszterben szokásos tárolni, tehát valós üzemmódban a szegmens tényleges kezdőcímének **16**-oda);

- **TYPE változó:** az elemek hossza byte-okban, ha változó, de

```
TYPE string = 1,
TYPE konstans = 0,
TYPE NEAR címke = -1,
TYPE FAR címke = -2

JMP    (TYPE cím) PTR [BX]
   ; NEAR vagy FAR ugrás [BX]-re attól függően,
   ; hogy cím közeli vagy távoli címke
```

- **... THIS típus:** a program szöveg adott pontján adott típusú szimbólum létrehozása;

Pl.:

```
ADATB       EQU     THIS BYTE
            ; BYTE típusú változó, helyfoglalás nélkül
ADATW       dw      1234H
            ; ez az adat ADATB-vel byte-osan érhető el
. . .
  mov   al,ADATW            ; hibás, de
  mov   al,BYTE PTR ADATW   ; al ⇐ 34H, helyes
  mov   al,ADATB            ; al ⇐ 34H, helyes
  mov   ah,ADATB+1          ; ah ⇐ 12H, helyes
```

Emlékeztetünk arra, hogy szavak tárolásakor az alacsonyabb helyértékű byte kerül az alacsonyabb címre!

5.

- **LOW kifejezés:** egy szó alsó (alacsonyabb helyértékű) byte-ja;
- **HIGH kifejezés:** egy szó felső (magasabb helyértékű) byte-ja;

Pl.:

```
ADATW   dw      1234H
        mov     al,LOW ADATW    ; al ⇐ 34H
        mov     ah,HIGH ADATW   ; ah ⇐ 12H
```

6. Előjelek:

- **+ :** pozitív előjel;
- **– :** negatív előjel;

7. Multiplikatív műveletek:
   - *** :** szorzás;
   - **/ :** osztás;
   - **MOD:** (modulo) a legkisebb nem negatív maradék, pl.:
     ```
     mov     al,20 MOD 16   ; al ⇐ 4
     ```
   - **kifejezés SHL lépés:**
     kifejezés léptetése balra **lépés** bittel;
   - **kifejezés SHR lépés:**
     kifejezés léptetése jobbra **lépés** bittel;

   lépés is lehet kifejezés!

A kifejezésben előforduló **műveleti jelek** (**SHL**, **SHR**, és a később előforduló **NOT**, **AND**, **OR**, és **XOR**) nem tévesztendők össze a velük azonos alakú **műveleti kódokkal**: az előbbieket a fordító program, az utóbbiakat a futó program hajtja végre!

8. Additív műveletek:
   - **+ :** összeadás;
   - **- :** kivonás;

9. Relációs operátorok (igaz=-1, hamis=0): általában feltételes fordítással kapcsolatban fordulnak elő
   - **EQ : =** `// -1 EQ 0FFFFFFFFH  igaz`
   - **NE : ≠** `// -1 NE 0FFFFFFFFH  hamis`
   - **LT : <**
   - **LE : ≤**
   - **GT : >**
   - **GE : ≥**

   33 bites argumentumok!
   1 GT -1 igaz
   1 GT 0FFFFFFFFH hamis

10. **NOT :** bitenkénti negálás;

11. **AND :** bitenkénti és művelet;

12. Bitenkénti vagy és kizáró vagy művelet:
   - **OR :** bitenkénti vagy művelet;
   - **XOR :** bitenkénti kizáró vagy művelet;

13. **SHORT** : 8 bites relatív címzés kikényszerítése;

Pl.:

```
        JMP SHORT L
. . .
L:      . . .
```


Feltételes fordítás

A fordító programok általában – így az assembler is – feltételes fordítási lehetőséget biztosít. Ez azt jelenti, hogy a program bizonyos részeit csak abban az esetben fordítja le, ha – a fordítóprogram számára ellenőrizhető – feltétel igaz illetve hamis.

```
IFxx        feltétel
...         ; lefordul, ha a feltétel igaz
ELSE        ; el is maradhat
...         ; lefordul, ha a feltétel hamis
ENDIF
```



```
IF      kifejezés  ; igaz, ha
                   ; kifejezés≠0
IFE     kifejezés  ; igaz, ha
                   ; kifejezés=0
```

Pl.:

```
IF      debug GT 20
        call    debug1
ELSE
        call    debug2
ENDIF
```

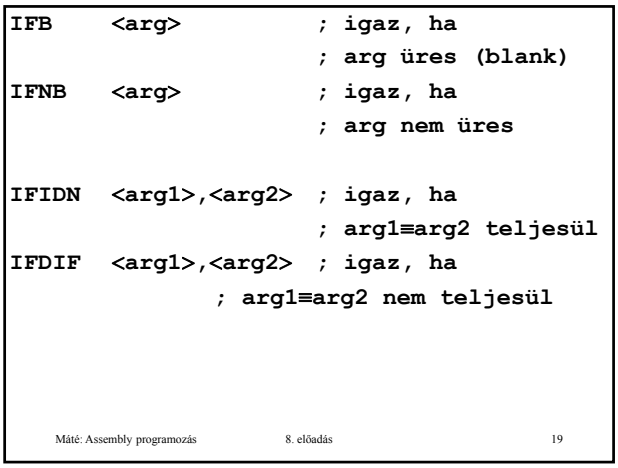


```
IF1                ; igaz a fordítás
                   ; első menetében
IF2                ; igaz a fordítás
                   ; második menetében

IFDEF  Szimbólum   ; igaz, ha Szimbólum
                   ; definiált
IFNDEF Szimbólum   ; igaz, ha Szimbólum
                   ; nem definiált
```

Pl. Csak akkor definiáljuk **buff**-t, ha a hossza ismert:

```
IFDEF  buff_len
buff   db  buff_len dup (?)
ENDIF
```



```
IFB     <arg>           ; igaz, ha
                        ; arg üres (blank)
IFNB    <arg>           ; igaz, ha
                        ; arg nem üres

IFIDN   <arg1>,<arg2>   ; igaz, ha
                        ; arg1≡arg2 teljesül
IFDIF   <arg1>,<arg2>   ; igaz, ha
                ; arg1≡arg2 nem teljesül
```