**Makró és blokk ismétlés**

Makró definíció:

```
M_név   MACRO   [fpar1[,fpar2...]]
                        ; makró fej (kezdet)
...                     ; makró törzs
        ENDM            ; makró vége
```

**fpar1,fpar2...** formális paraméterek vagy egyszerűen paraméterek.

A makró definíció nem lesz része a lefordított programnak, csupán azt határozza meg, hogy később mit kell a makró hívás helyére beírni (makró kifejtés, helyettesítés).

A makró törzsön belül előfordulhat makró hívás és másik makró definíció is.



Makró hívás:

**M_név [apar1[,apar2...]]**

**apar1,apar2...** aktuális paraméterek/argumentumok.

A műveleti kód helyére írt **M_név** hatására a korábban megadott definíció szerint megtörténik a makró helyettesítés, más néven makró kifejtés. Ez a makró törzs bemásolását jelenti, miközben az összes paraméter összes előfordulása a megfelelő argumentummal helyettesítődik. A helyettesítés szövegesen történik, azaz minden paraméter – mint szöveg – helyére a megfelelő argumentum – mint szöveg – kerül.

A helyettesítés nem rekurzív.

Makró hívás argumentuma nem lehet makró hívás.

Az argumentumnak megfelelő formális paraméternek lehet olyan előfordulása, amely a későbbiek során makró hívást eredményez.



Dupla szavas összeadás: **(DX:AX)⇐(DX:AX)+(CX:BX)**

Eljárás deklaráció:

```
EDADD   PROC    NEAR
        ADD     AX,BX
        ADC     DX,CX
        RET
EDADD   ENDP
```

Makró definíció:

```
MDADD   MACRO
        ADD     AX,BX
        ADC     DX,CX
        ENDM
```



Ha a programban valahol dupla szavas összeadást kell végezzünk, akkor hívnunk kell az eljárást illetve a makrót:

Eljárás hívás:

```
CALL    EDADD
```

Makró hívás:

```
MDADD
```

Futás közben felhívásra kerül az **EDADD** eljárás

Fordítás közben megtörténik a makró helyettesítés:

```
ADD     AX,BX
ADC     DX,CX
```

Futás közben ez a két utasítás kerül csak végrehajtásra.



Látható, hogy eljárás esetén kettővel több utasítást kell végrehajtanunk, mint makró esetében (**CALL EDADD** és **RET**).

Még nagyobb különbséget tapasztalunk, ha **(CX:BX)** helyett paraméterként kívánjuk megadni az egyik összeadandót:



Eljárás deklaráció:

```
EDADD2  PROC    NEAR
        PUSH    BP
        MOV     BP,SP
        ADD     AX,4[BP]
        ADC     DX,6[BP]
        POP     BP
        RET     4
EDADD   ENDP
```

Eljárás hívás:

Ha **SI** az összeadandónk címét tartalmazza, akkor a felhívás:

```
PUSH    2[SI]
PUSH    [SI]
CALL    EDADD2
```

Futás közben végrehajtásra kerül a paraméter átadás, az eljárás hívás, az eljárás: összesen 9 utasítás

Makró definíció:

```
MDADD2  MACRO   P
IFB     <P>
        ADD     AX,BX
        ADC     DX,CX
ELSE
        ADD     AX,P
        ADC     DX,P+2
ENDIF
        ENDM
```

Makró hívás:

```
MDADD2 [SI]
```

Fordítás közben a hívás az

```
ADD     AX,[SI]
ADC     DX,[SI]+2
```

utasításokra cserélődik, futás közben csak ez a két utasítás kerül végrehajtásra.

```
MDADD2
```

hatása:

```
ADD     AX,BX
ADC     DX,CX
```

Most sem része a makró definíció a lefordított programnak.



A fenti példában rövid volt az eljárás törzs, és ehhez képest viszonylag hosszú volt a paraméter átadás és átvétel. Ilyenkor célszerű a makró alkalmazása.

De ha a program sok helyéről kell meghívnunk egy hosszabb végrehajtandó programrészt, akkor általában célszerűbb eljárást alkalmazni.



Paraméter másutt is előfordulhat a makró törzsben, nemcsak az operandus részen, pl.:

```
PL      macro p1,P2
        mov   ax,p1
        P2    p1
        endm

        PL    Adat, INC
```

hatása:

```
        mov   ax,Adat
        INC   Adat
```



A **&**, **%**, **!** karakterek továbbá a **<>** és **;;** speciális szerepet töltenek be makró kifejtéskor.

**&** (helyettesítés operátor):

- ha a paraméter – helyettesített – értéke része egy szónak;
- idézeten belüli helyettesítés:

```
errgen      macro  y, x
err&y       db     'Error &y: &x'
            endm

            errgen 5, <Unreadable disk>
```

hatása:

```
err5        db  'Error 5: Unreadable disk'
```



**<>** (literál szöveg operátor): Ha aktuális paraméter szóközt vagy **,**-t is tartalmaz. Az előző példa **<>** nélkül:

```
        errgen 5, Unreadable disk
```

kifejtve:

```
err5    db      'Error 5: Unreadable'

adat    macro   p
        db      p
        endm

        adat    <'abc',13,10,0>
        adat    'abc',13,10,0
```

kifejtve:

```
        db      'abc',13,10,0
        db      'abc'
```



**!** (literál karakter operátor): Az utána következő karaktert makró kifejtéskor közönséges karakterként kell kezelni. Pl.: a korábbi **errgen** makró

```
        errgen 103, <Expression !> 255>
```

hívásának hatása:

```
err103 db  'Error 103: Expression > 255'
```

de

```
        errgen 103, <Expression > 255>
```

hívásának hatása:

```
err103 db  'Error 103: Expression '
```

**%** (kifejezés operátor): Az utána lévő argumentum (kifejezés is lehet) értéke – és nem a szövege – lesz az aktuális paraméter. Pl.:

```
sym1    equ     100
sym2    equ     200
txt     equ     'Ez egy szöveg'

kif     macro   exp, val
        db      "&exp = &val"
        endm

        kif     <sym1+sym2>, %(sym1+sym2)
        kif     txt, %txt

        db      "sym1+sym2 = 300"
        db      "txt = 'Ez egy szöveg'"
```



Az alábbi példa a **%** használatán kívül a makró törzsön belüli makró hívást is bemutatja:

```
s       =       0
ErrMsg  MACRO   text
s       =       s+1
        Msg     %s,text
        ENDM

Msg     MACRO   sz,str
msg&sz  db      str
        ENDM
```



```
s       =       0
ErrMsg  MACRO   text
s       =       s+1
        Msg     %s,text
        ENDM
```

```
Msg     MACRO   sz,str
msg&sz  db      str
        ENDM
```

```
        ErrMsg 'syntax error'
```

makró hívás hatására bemásolásra kerül (**.LALL** hatására látszik a listán) az

```
s       =       s+1
        Msg     %s,'syntax error'
```

szöveg. **s** értéke itt **1**-re változik. Újabb makró hívás (**Msg**). A **%s** paraméter az **s** értékére (**1**) cserélődik, majd kifejtésre kerül ez a makró is, ebből kialakul:

```
msg1    db      'syntax error'
```



```
s       =       0
ErrMsg  MACRO   text
s       =       s+1
        Msg     %s,text
        ENDM
```

```
Msg     MACRO   sz,str
msg&sz  db      str
        ENDM
```

Egy újabb hívás és hatása:

```
        ErrMsg 'invalid operand'
msg2    db      'invalid operand'
```



**;;** (makró kommentár): A makró definíció megjegyzéseinek kezdetét jelzi. A **;;** utáni megjegyzés a makró kifejtés listájában nem jelenik meg.



```
LOCAL   c1[,c2...]
```

**c1**, **c2**, **...** minden makró híváskor más, **??xxxx** alakú szimbólumra cserélődik, ahol **xxxx** a makró generátor által meghatározott hexadecimális szám. A **LOCAL** operátort közvetlenül a makró fej utáni sorba kell írni.

```
KOPOG   macro   n
        LOCAL   ujra
        mov     cx,n
ujra:   KOPP
        loop    ujra
        endm
```

Ha a programban többször hívnánk a **KOPOG** makrót, akkor a **LOCAL** operátor nélkül az **ujra** címke többször lenne definiálva.

Makró definíció belsejében lehet másik makró definíció is. A belső makró definíció csak a külső makró meghívása után jut érvényre, válik láthatóvá. Pl.:

```
shifts     macro      OPNAME ; makrót
                          ; definiáló makró
OPNAME&S   MACRO      OPERANDUS,N
           mov        cl, N
           OPNAME     OPERANDUS,cl
           ENDM
           endm
```



```
shifts     macro      OPNAME ; makrót
                          ; definiáló makró
OPNAME&S   MACRO      OPERANDUS,N
           mov        cl, N
           OPNAME     OPERANDUS,cl
           ENDM
           endm
```

Ha ezt a makrót felhívjuk pl.:

```
           shifts ROR
```

akkor a

```
RORS       MACRO   OPERANDUS,N
           mov     cl, N
           ROR     OPERANDUS,cl
           ENDM
```

makró definíció generálódik.



```
RORS       MACRO   OPERANDUS,N
           mov     cl, N
           ROR     OPERANDUS,cl
           ENDM
```

Mostantól meghívható a **RORS** makró is, pl.:

```
           RORS    AX, 5
```

aminek a hatása:

```
           mov     cl, 5
           ROR     AX,cl
```