Makró definíció tartalmazhat blokk ismétlést, és blokk ismétlés is tartalmazhat makró definíciót vagy makró hívást.

Pl.: A bit léptető és forgató utasítás kiterjesztésnek egy újabb megoldása:

```
; makrót definiáló blokkismétlés
        IRP     OP,<RCR,RCL,ROR,ROL,SAR,SAL>
OP&S    MACRO   OPERANDUS,N
        mov     cl, N
        OP      OPERANDUS,cl
        ENDM
        ENDM
```

Ennek a megoldásnak előnye, hogy nem kell külön meghívnunk a külső makrót az egyes utasításokkal, mert ezt elvégzi helyettünk az **IRP** blokk ismétlés.


```
; makrót definiáló blokkismétlés
        IRP     OP,<RCR,RCL,ROR,ROL,SAR,SAL>
OP&S    MACRO   OPERANDUS,N
        mov     cl, N
        OP      OPERANDUS,cl
        ENDM
        ENDM
```

hatása:

```
RCRS    MACRO   OPERANDUS,N
        mov     cl, N
        RCR     OPERANDUS,cl
        ENDM
RCLS    MACRO   OPERANDUS,N
        ...
```


Szegmens, szegmens csoport

sz_név **SEGMENT** *aling_type combine_type 'osztály'*

. . . szegmens

sz_név **ENDS**

sz_név a szegmens (szelet) neve.

A fordító az azonos nevű szegmens szeleteket úgy tekinti, mintha folyamatosan, egyetlen szegmens szeletbe írtuk volna.

Az azonos nevű szegmens szeletek paraméterei egy modulon belül nem változhatnak.

A szerkesztő egy memória szegmensbe szerkeszti az azonos nevű szegmenseket.

aling_type (illesztés típusa): a szerkesztőnek szóló információ. Azt mondja meg, hogy a szegmens szelet milyen címen kezdődjön:

BYTE 1-gyel,

WORD 2-vel,

DWORD 4-gyel,

PARA 16-tal,

PAGE 256-tal osztható címen.

Akkor van jelentősége, ha a szegmens szelet egy másik modulban lévő ugyanilyen nevű szegmens szelet folytatása.

combine_type (kombinációs típus): a szerkesztőnek szóló üzenet. Lehet:

PUBLIC: (alapértelmezés) az azonos nevű szegmens szeletek egymás folytatásaként szerkesztendők.

COMMON: az azonos nevű szegmens szeletek azonos címre szerkesztendők. Az így keletkező terület hossza megegyezik a leghosszabb ilyen szegmens szelet hosszával. A **COMMON** hatása csak különböző modulokban megírt szegmens szeletekre érvényesül.

MEMORY: a szerkesztő ezt a szegmenst az összes többi szegmens fölé fogja szerkeszteni, mindig a program legmagasabb címre kerülő része (a Microsoft **LINK** programja ezt nem támogatja).

STACK: a stack részeként szerkesztendő a szegmens szelet, egyebekben megegyezik a **PUBLIC**-kal. Amennyiben van **STACK** kombinációs típusú szegmens a programban, akkor **SS** és **SP** úgy inicializálódik, hogy **SS** az utolsó **STACK** kombinációs típusú szegmensre mutat, **SP** értéke pedig ennek a szegmensnek a hossza.

AT kif: a **kif** sorszámú paragrafusra kerül a szegmens szelet. Alkalmas lehet pl. a port-okhoz kapcsolódó memória címek szimbolikus definiálására.

A szegmens osztály legtöbbször **CODE**, **DATA**, **CONSTANT**, **STACK**, **MEMORY**.

Beágyazott (nested) szegmensek

```
message   MACRO
          LOCAL     symbol
ADAT      SEGMENT PARA PUBLIC 'DATA'
symbol    DB   &text
          DB   13,10,"$"
ADAT      ENDS
          mov  ah, 09h
          mov  dx, OFFSET symbol
          int  21h
          ENDM
          . . .
KOD       SEGMENT PARA PUBLIC 'COCE'
          . . .
          message "Please insert disk"
```


Az **ASSUME** utasítás az assembler-t informálja arról, hogy a címzésekhez a szegmens regisztereket milyen tartalommal használhatja, más szóval, hogy melyik szegmens regiszter melyik szegmensnek a szegmens címét tartalmazza (melyik szegmensre mutat):

ASSUME sz_reg1:sz_név1[, sz_reg2:sz_név2 ...]

ASSUME sz_reg1:sz_név1[, sz_reg2:sz_név2 ...]

Az **ASSUME** utasításban felsorolt szegmenseket „aktív"-nak nevezzük.

Az **ASSUME** utasítás nem gondoskodik a szegmens regiszterek megfelelő tartalommal történő feltöltéséről! Ez a programozó feladata!

Az **ASSUME** utasítás hatása egy-egy szegmens regiszterre vonatkozóan mindaddig érvényes, amíg egy másik **ASSUME** utasítással mást nem mondunk az illető regiszterről.

A **GROUP** utasítással csoportosíthatjuk a szegmenseinket:

G_nev GROUP S_név1[, S_név2...]

Az egy csoportba sorolt szegmenseket a szerkesztő a memória egy szegmensébe helyezi. Ha ilyenkor az **ASSUME** utasításban a csoport nevét adjuk meg, és ennek megfelelően állítjuk be a bázis regisztert, akkor a csoport minden szegmensének minden elemére tudunk hivatkozni.
Ilyenkor egy változó **OFFSET**-je és effektív címe (**EA**) nem feltétlenül egyezik meg.


```
GRP       GROUP          ADAT1,ADAT2
ADAT1     SEGMENT        para public 'data'
A         dw             1111h
...
ADAT1     ENDS

ADAT2     SEGMENT        para public 'data'
W         dw             2222h
...
ADAT2     ENDS

code  segment      para public 'code'
      ASSUME       CS:code, DS:GRP
      ASSUME       SS:stack, ES:nothing
...
```



```
GRP   GROUP  ADAT1,ADAT2
ADAT1 SEGMENT ...
A     dw     1111h
...
ADAT1 ENDS
ADAT2 SEGMENT ...
W     dw     2222h
...
ADAT2 ENDS
```

```
code  segment ...
      ASSUME CS:code, DS:GRP
      ASSUME SS:stack, ...
      ...
```

```
      MOV SI,OFFSET W ; SI ⇐ W offset-je: 0
; az ADAT2 szegmens elejétől mért távolság
      MOV     AX,[SI]  ; AX ⇐ 1111h,
; de!!!
      LEA     SI, W    ; SI ⇐ effektív címe:
; a GRP szegmens csoport elejétől mért távolság
      MOV     AX,[SI]  ; AX ⇐ 2222h.
```

Globális szimbólumok

A több modulból is elérhető szimbólumok.

A globális szimbólumok teszik lehetővé, hogy a programjainkat modulokra bontva készítsük el. Az egyes modulok közötti kapcsolatot a globális szimbólumok jelentik.

Globális szimbólumok

Ha egy szimbólumot globálissá kívánunk tenni, akkor **PUBLIC**-ká kell nyilvánítanunk annak a modulnak az elején, amelyben a szimbólumot definiáljuk:

```
PUBLIC    sz1[, sz2...]
```

Azokban a modulokban, amelyekben más modulban definiált szimbólumokat is használni szeretnénk, az ilyen szimbólumokat **EXTRN**-né kell nyilvánítanunk:

```
EXTRN sz1:típus1[, sz2:típus2...]
```


INCLUDE utasítás

```
INCLUDE     File_Specifikáció
```

hatására az assembler az **INCLUDE** utasítás helyére bemásolja az utasítás paraméterében specifikált file szövegét.

Az **INCLUDE**-olt file-ok is tartalmazhatnak **INCLUDE** utasítást.

Ha makró definícióinkat a **MyMacros.i** file-ba kívánjuk összegyűjteni, akkor célszerű ezt a file-t így elkészítenünk:

```
IFNDEF   MyMacros_i
MyMacros_i   = 1
...      ;; makró, struktúra definíciók
...      ;; EXTRN szimbólumok
ENDIF
```

Ekkor a **MyMacros.i** file legfeljebb egyszer kerül bemásolásra, mert az összes további esetben a feltételes fordítás feltétele már nem teljesül.

A **.**-ot _-sal helyettesítettük!

A legtöbb include file-ban ezt a konvenciót alkalmazzák.

Lista vezérlési utasítások

```
TITLE       cím
```

A fordítás során keletkező lista minden oldalán megjelenik ez a **cím**. Egy modulon belül csak egyszer alkalmazható.

```
SUBTITLE    alcím
```

Többször is előfordulhat egy modulon belül. A program lista minden oldalán – a cím alatt – megjelenik az utolsó **SUBTITLE** utasításban megadott **alcím**.


```
PAGE        [op1][,op2]
```

Paraméter nélkül lapdobást jelent.

Ha egyetlen paramétere van és az egy **+** jel, akkor a fejezet sorszámát növeli eggyel, és a lapszámot **1**-re állítja.

Ettől eltérő esetekben **op1** az egy lapra írható sorok (**10≤op1≤255**), **op2** az egy sorba írható karakterek számát jelenti (**60≤op2≤132**).

Ha valamelyik paramétert nem adjuk meg, akkor természetesen nem változik a korábban beállított értéke.

A sorok száma kezdetben **66**, a karaktereké **80**.

A **TITLE**, a **SUBTITLE** és a **PAGE** egy elkészült programcsoport végső papír-dokumentációjának jól olvashatóságát segíti.

A programfejlesztés során ritkán készítünk program listákat.

NAME név

A modul nevét definiálhatjuk vele. A szerkesztő ezt a nevet fogja használni. Ha nem szerepel **NAME** utasítás a modulban, akkor a **TITLE** utasítással megadott cím a modul neve. Ha ez sincs, akkor a file nevének első **6** karaktere lesz a modul neve.

COMMENT határoló_jel szöveg határoló_jel

Segítségével több soros kommentárokat írhatunk. Az assembler a **COMMENT** utáni első látható karaktert tekinti **határoló_jel**-nek, és ennek a jelnek az újabb előfordulásáig minden kommentár. Nyilvánvaló, hogy a kommentár belsejében nem szerepelhet **határoló_jel**.

%OUT szöveg

Amikor ehhez az utasításhoz ér a fordítóprogram, akkor a paraméterként megadott **szöveg**-et kiírja a képernyőre.

.RADIX számrendszer_alapja

Ha programban egy szám nem tartalmaz számrendszer jelölést, akkor az illető számot ebben a számrendszerben kell érteni (alapértelmezésben decimális).