Struktúra

Struktúra definíció: a struktúra típusát definiálja a későbbi struktúra hívások számára, ezért a memóriában nem jár helyfoglalással.

```
Str_tipus  STRUC   ; struktúra (típus) definíció
...                ; mező (field) definíciók:
...                ;     egyszerű adat definíciós
...                ;     utasítások
Str_tipus  ENDS    ; struktúra definíció vége
```

A **mező** (**field**) definíció csak egyszerű adat definíciós utasítással történhet, ezért struktúra mező nem lehet másik struktúra vagy rekord.

A mezők definiálásakor megadott értékek kezdőértékül szolgálnak a későbbiekben történő struktúra hívásokhoz. A definícióban megadott kezdőértékek közül azoknak a mezőknek a kezdőértéke híváskor felülbírálható, amelyek csak egyetlen adatot tartalmaznak (ilyen értelemben a szöveg konstans egyetlen adatnak minősül). Pl.:

```
S       STRUC            ; struktúra (típus) definíció
F1      db  1,2          ; híváskor nem lehet felülírni
F2      db  10 dup (?)   ;     nem lehet felülírni
F3      db  5            ;     felülírható
F4      db  'a','b','c'  ;     nem lehet felülírni, de
F5      db  'abc'        ;     felülírható
S       ENDS
```


Struktúra hívás: A struktúra definíciójánál megadott **Str_tipus** névnek a műveleti kód részen történő szerepeltetésével hozhatunk létre a definíciónak megfelelő típusú struktúra változókat. A kezdőértékek fölülbírása a kívánt értékek **< >** közötti felsorolásával történik

```
S1    S                ; kezdőértékek a definícióból
S2    S  <,,7,,'FG'>   ; F3 kezdőértéke 7,
                       ; F5-é 'FG '
S3    S  <,,'A'>       ; F3 kezdőértéke 'A' ,
                       ; a többi a definícióból
```

Struktúrából vektort is előállíthatunk, pl.:

```
S_v   S  8 dup (<,,'A'>)
                       ; 8 elemű struktúra vektor
```


Struktúra mezőre hivatkozás: A struktúra változó nevéhez tartozó **OFFSET** cím a struktúra **OFFSET** címét, míg a mező neve a struktúrán belüli címet jelenti. A struktúra adott mezejére úgy hivatkozhatunk, hogy a struktúra és mező név közé **.**-ot írunk, pl.:

```
        MOV     AL,S1.F3
```

A **.** bármely oldalán lehet másfajta cím is, pl.

```
        MOV     BX, OFFSET S1
```

után az alábbi utasítások mind ekvivalensek az előzővel:

```
        MOV     AL,[BX].F3
        MOV     AL,[BX]+F3
        MOV     AL,F3.[BX]
        MOV     AL,F3[BX]
```


A fentiekből az is következik, hogy a mező és struktúra név – ellentétben a magasabb szintű programozási nyelvekkel – szükségképpen **egyedi név**, tehát sem másik struktúra definícióban, sem közönséges változóként nem szerepelhet.

A struktúra vektorokat a hagyományos módon még akkor sem indexezhetjük, ha az index konstans. Pl.

```
        MOV     AL,S_v[5].F3
                ; szintaktikusan helyes, de
```

[5] nem a vektor ötödik elemére mutató címet fogja eredményezni, csupán **5** byte-tal magasabb címet, mint **S_v.F3**. Ha **i** változó, akkor

```
        MOV     AL,S_v[i].F3
                ; szintaktikusan is HIBÁS!
```


Mindkét esetben programmal kell kiszámítani az elem offset-jét, pl. ha **i** word:

```
MOV     AX,TYPE S  ; S hossza byte-okban
                   ;     (l. később)
MUL     i          ; Az indexet 0-tól számoljuk!
MOV     BX,AX      ; az adat nem „lóghat ki” a
                   ; szegmensből (DX=0)
MOV     AL,S_v.F3[BX]
        ; AL ⇐ az i-dik elem F3 mezeje.
```

Rekord

Rekord definíció: Csak a rekord típusát definiálja a későbbi rekord hívások számára.

`Rec_tipus RECORD mező_specifikációk`

Az egyes mező specifikációkat `,`-vel választjuk el egymástól.

Mező specifikáció:

`mező_név:szélesség=kezdőérték`

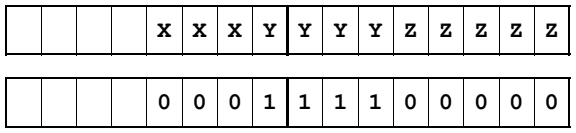
`szélesség` a mező bit-jeinek száma.

Az **`=kezdőérték`** el is maradhat, ha elmarad, az a mező **`0`**-val való inicializálását írja elő.

Pl.:

`R RECORD X:3,Y:4=15,Z:5`

Az **`R`** rekord szavas (12 bit), a következőképpen helyezkedik el egy szóban:



Rekord hívás: A rekord definíciójánál megadott névnek a műveleti kód részen történő szerepeltetésével hozhatunk létre a definíciónak megfelelő típusú rekord változókat. A kezdőértékek fölülbírálása a kívánt értékek **`< >`** közötti felsorolásával történik.

```
R1      R   < >     ; 01E0H, kezdőértékek a
                    ;    definícióból
R2      R   <,,7>   ; 01E7H, X, Y kezdőértéke a
                    ;    definícióból, Z-é 7
R3      R   <1,2>   ; 0240H, X kezdőértéke 1, Y-é 2,
                    ;    Z-é a definícióból
```

Rekordból vektort is előállíthatunk, pl.:

```
R_v     R   5 dup (<1,2,3>) ; 0243H,
                    ; 5 elemű rekord vektor
```


Rekord mezőre hivatkozás

A mező név olyan konstansként használható, amely azt mondja meg, hány bittel kell jobbra léptetnünk a rekordot, hogy a kérdéses mező az 1-es helyértékre kerüljön.

`MASK` és **`NOT MASK`** operátor

```
; AX ⇐ R3 Y mezeje a legalacsonyabb helyértéken
    MOV     AX,R3       ; R3 szavas rekord!
    AND     AX,MASK Y   ; Y mezőhöz tartozó bitek
                        ;    maszkolása
    MOV     CL,Y        ; léptetés előkészítése
    SHR     AX,CL       ; kész vagyunk.
```

`SAR` nem lenne korrekt: nem biztos, hogy az Y mező nem tartalmazza az előjel bitet.

Kifejezés

Egy művelet operandusa lehet konstans, szimbólum vagy kifejezés.

Konstans

A konstans lehet numerikus vagy szöveg konstans.

A numerikus konstansok decimális, hexadecimális, oktális és bináris számrendszerben adhatók meg. A számrendszert a szám végére írt **`D`**, **`H`**, **`O`** illetve **`B`** betűvel választhatjuk ki.

`.RADIX n ; 2 ≤ n ≤ 16 , n` decimális

A szöveg konstansokat a **`DB`** utasításban ″ vagy ′ jelek között adhatjuk meg.

Szimbólum

A szimbólum lehet szimbolikus konstans, változó név vagy címke.

Szimbolikus konstans: Az **`=`** vagy az **`EQU`** pszeudo utasítással definiálható. Szimbolikus szöveg konstans csak **`EQU`**-val definiálható. A szimbolikus konstans a program szövegnek a definíciót követő részében használható, értékét a használat helyét megelőző utolsó definíciója határozza meg.

Ha egy szimbólumot **`EQU`**-val definiálunk, akkor ezt a szimbólumot a modulban másutt nem definiálhatjuk!

```
S       =       1       ; S értéke 1
N       EQU     14      ; N értéke 14
        MOV     CX,N    ; CX ⇐ 14
ISM:
S       =       S+1 ; S értéke ezután 2, függetlenül
                ; attól, hogy hányadszor fut a ciklus
        MOV     AX,S    ; AX ⇐ 2
        LOOP    ISM
N       =       5       ; hibás
N       EQU     5       ; hibás
S       =       5       ; helyes
S       EQU     5       ; hibás
```


Szimbolikus konstansként használhatjuk a **$** jelet (helyszámláló), melynek az értéke mindenkor a program adott sorának megfelelő **OFFSET** cím. A helyszámláló értékének módosítására az **ORG** utasítás szolgál, pl.:

```
ORG     $+100H ; 100H byte kihagyása
               ; a memóriában
```


Címke: Leggyakoribb definíciója, hogy valamelyik utasítás előtt a sor első pozíciójától **:** -tal lezárt azonosítót írunk. Az így definiált címke **NEAR** típusú. Címke definícióra további lehetőséget nyújt a **LABEL** és a **PROC** pszeudo utasítás:

```
ALFA:   ...     ; NEAR típusú

BETA    LABEL FAR ; FAR típusú
GAMMA:  ...     ; BETA is ezt az utasítást
                ; címkézi, de GAMMA NEAR típusú
```


Az eljárás deklarációt a **PROC** pszeudo utasítással nyitjuk meg. A címke rovatba írt azonosító az eljárás neve és egyben a belépési pontjának címkéje. Az eljárás végén az eljárás végét jelző **ENDP** pszeudo utasítás előtt meg kell ismételnünk ezt az azonosítót, de az ismétlés nem minősül címkének. Az eljárás címkéje aszerint **NEAR** vagy **FAR** típusú, hogy maga az eljárás **NEAR** vagy **FAR**. Pl.:

```
A       PROC            ; NEAR típusú
        ...
B       PROC    NEAR    ; NEAR típusú
        ...
C       PROC    FAR     ; FAR típusú
        ...
```


Címkére vezérlés átadó utasítással hivatkozhatunk, **NEAR** típusúra csak az adott szegmensből, **FAR** típusúra más szegmensekből is.

Változó: Definíciója adat definíciós utasításokkal történik. Néha (adat) címkének is nevezik.

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
```