

Assembly programozás

Pseudo utasítások

A pseudo utasításokat a fordítóprogram hajtja végre. Ez a végrehajtás fordítás közbeni tevékenységet vagy a fordításhoz szükséges információ gyűjtést jelenthet.

Máté: Assembly programozás

7. előadás

1

Adat definíciós utasítások

Az adatokat általában külön szegmensben szokás és javasolt definiálni iniciálással vagy anélkül.

Az adat definíciós utasítások elé általában azonosítót (változó név) írunk, hogy hivatkozhatunk az illető adatra. Egy-egy adat definíciós utasítással – vesszővel elválasztva – több azonos típusú adatot is definiálhatunk. A kezdőérték – megfelelő típusú – tetszőleges konstans (szám, szöveg, cím, ...) és **kifejezés** lehet. Ha nem akarunk kezdőértéket adni, akkor ? -et kell írunk.

DUP operátor

kifejezés DUP (adat)

Máté: Assembly programozás

7. előadás

2

Egyszerű adat definíciós utasítások

Define Byte (DB):

```
Adat1 db 25 ; 1 byte, kezdőértéke decimális 25
Adat2 db 25H ; 1 byte, kezdőértéke hexadec. 25
Adat3 db 1,2 ; 2 byte (nem egy szó!)
Adat4 db 5 dup (?) ; 5 inicializálatlan byte
Kar db 'a','b','c' ; 3 ASCII kódú karakter
Szoveg db "Ez egy szöveg",13,0AH ; ACSII kódú szöveg és 2 szám
Szov1 db 'Ez is "szöveg"'
Szov2 db "és ez is 'szöveg'"
```

Máté: Assembly programozás

7. előadás

3

Define Word (DW):

```
Szo dw 0742H,452
Szo_cime dw Szo ; Szo offset címe
```

Define Double (DD):

```
Szo_f dd Szo ; Szo távoli ; (segment + offset) címe
```

Define Quadword (DQ)

Define Ten bytes (DT)

Máté: Assembly programozás

7. előadás

4

Összetett adat definíciós utasítások

Struktúra és a rekord.

Először a **típust** kell definiálni. A típus **definíció** nem jelent helyfoglalást. A struktúra illetve rekord konkrét példányai struktúra illetve rekord **hívással** definiálhatók. A struktúra illetve rekord elemi részeit **mezőknek (field)** nevezzük.

A hardver nem ismeri ezeket az adat típusokat, a kezelésükről szoftveresen kell gondoskodni!

Máté: Assembly programozás

7. előadás

5

Struktúra

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

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

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

Máté: Assembly programozás

7. előadás

6

A mezők definiálásakor megadott értékek kezdőértékül szolgálnak a későbbiekben történő struktúra hívásokhoz. A definícióban megadott kezdőértékek közül azoknak a mezőknek a kezdőértéke híváskor felülbírható, 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
```

Máté: Assembly programozás

7. előadás

7

Struktúra hívás: A struktúra definíciójánál megadott **Str_típus** 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
```

Máté: Assembly programozás

7. előadás

8

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 mezőjére 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]
```

Máté: Assembly programozás

7. előadás

9

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!

Máté: Assembly programozás

7. előadás

10

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

```
MOV    AX, TYPE S      ; S hossza byte-okban
                          ; (1. később)
```

```
MUL    i               ; Az indexet 0-tól számoljuk!
```

```
MOV    BX, AX          ; az adat nem „lőghat ki” a
                          ; szegmensből (DX=0)
```

```
MOV    AL, S_v.F3[BX]
                          ; AL ← az i-dik elem F3 mezeje.
```

Máté: Assembly programozás

7. előadás

11

Rekord

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

Rec_típus 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.

Máté: Assembly programozás

7. előadás

12

Pl.:

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

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

				X	X	X	Y	Y	Y	Y	Z	Z	Z	Z	Z

				0	0	0	1	1	1	1	0	0	0	0	0

Máté: Assembly programozás 7. előadás 13

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
```

Máté: Assembly programozás 7. előadás 14

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.

Máté: Assembly programozás 7. előadás 15

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.

Máté: Assembly programozás 7. előadás 16

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!

Máté: Assembly programozás 7. előadás 17

```
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
```

Máté: Assembly programozás 7. előadás 18

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
```

Máté: Assembly programozás

7. előadás

19

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
```

Máté: Assembly programozás

7. előadás

20

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
```

```
. . .
```

Máté: Assembly programozás

7. előadás

21

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

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

Máté: Assembly programozás

7. előadás

22