Assembly program felépítése

Nézzük át részletesen a ***template*** programunkat és elemezzük sorról-sorra (ne felejtsük el, a sorszámozás nem része a programnak). A *template.asm* program elérhető a feltelepített ***Irvine*** programcsomag *Irvine\Examples\ch03* könyvtárában:

```
 1: ; Program template
 2:
 3: .386
 4: .model flat,stdcall
 5: .stack 4096
 6: ExitProcess proto,dwExitCode:dword
 7:
 8: .data
 9:  ; declare variables here
10: .code
11: main proc
12:  ; write your code here
13:
14:  invoke ExitProcess,0
15: main endp
16: end main
```

Az **1.** sor kommentet, tehát olyan szövegeket tartalmaz, mely nem kerül kiértékelésre és végrehajtásra. Mi magunk is helyezhetünk (sőt helyezzünk is) el kommenteket a programunkban, melyet úgy tehetünk meg, hogy egy "**;**" karaktert helyezünk el a kikommentezni kívánt sor vagy sorrészlet elé.

A **3.** sor tartalmazza a ***.386*** direktívát, amely azonosítja a programunkat, mint 32 bites program, hozzáférést biztosítva ezzel a 32 bites regiszterekhez és címekhez.

A **4.** sor választja ki a program memória modelljét, ami jelen esetben ***flat***, és azonosítja a hívás konvenciót (***stdcall***) az eljárásokhoz.

A **5.** sorban "félreteszünk" 4096 bájtot a futási verem (*runtime* ***stack***) számára, amellyel minden programnak rendelkeznie kell.

A **6.** sor deklarál egy prototípust az ***ExitProcess*** függvény számára, ami egy standard Windows service (szolgáltatás). A prototípus áll a függvény nevéből, a ***PROTO*** kulcsszóból, egy vesszőből és az input paraméterek listájából. Az *ExitProcess* bemeneti paraméterének neve ***dwExitCode***. Erre gondolhatunk úgy, mint egy érték visszaadásra a Windows operációs rendszer felé. A *0* visszaadott érték jelenti, hogy a programunk sikeresen lefutott. Bármilyen más egész számérték valamilyen hibakódot

jelöl. Tehát egy assembly programra úgy is gondolhatunk, mint egy szubrutin, vagy egy függvény, amely az operációs rendszer által kerül meghívásra. Amikor a program a végéhez ér, meghívja az *ExitProcess* eljárást és visszaad egy egész számot, ami jelzi az operációs rendszenek, hogy a programunk hogyan működött.

A **8.** sorban lévő ***.data*** direktíva jelöli az adat szegmenst, itt tudunk deklarálni és definiálni különböző változókat a későbbiek folyamán.

A **10.** sorban lévő ***.code*** direktíva jelöli a program kódrészének a kezdetét, azt a részt, amely végrehajtható parancsokat tartalmaz.

Legtöbbször a *.code* után következő sor (jelenleg **11.** sor) tartalmazza a program belépési pontjának a deklarációját, amely konvenciók szerint a ***main*** nevet viseli. Egy program belépési pontja azon legelső parancs helyét határozza meg, melyet a program először fog végrehajtani. (A ***PROC*** kulcsszó jelöli, hogy a main egy eljárás. Ezt később részletesen tárgyaljuk.)

A **15.** sorban lévő ***endp*** direktíva jelöli az eljárás végét. A programunkban lévő eljárás a *main* nevet viselte, tehát az *endp* is ezt a nevet kell, hogy kapja.

Legvégül a **16.** sorban lévő ***end*** direktíva jelöli a program végét és hivatkozik a program belépési pontjára. Ha bármilyen más szöveget helyezünk a programba az **end** direktíva után, ezek az assembler figyelmen kívül hagyja.