Gyakori assembly hibák

Lentebb az assembly programozás kis-, és nagy-ZH –k tipikus és gyakori hibáit gyűjtik össze. Ezekre érdemes lehet jobban odafigyelni.

Előjeles és előjel nélküli számok kiterjesztése.

Sokan keverik az előjeles és előjel nélküli számok kiterjesztését 8-ról 16 bitre. A helyes megoldás:

He előjel nélküli számot kell kiterjeszteni, akkor két lépés van:

1. A számot bele kell tenni egy általános célú regiszter (AX, BX, CX, DX) alsó byte-jába.
2. A regiszter felső byte-ját ki kell nullázni.

Ha előjeles számot kell kiterjeszteni, akkor:

1. Bele kell azt tenni az AL regiszterbe (AX alsó byte-ja). De ez csak az AL-el működik.
2. CBW utasítás. Fontos, hogy nincs paramétere! A kiterjesztett szám az AX-ben lesz.

Például: Adott egy adat szegmens:

```
ADAT SEGMENT PARA PUBLIC ’DATA’
    A       db  42      ; egy előjel nélküli byte
    B       db  -5      ; egy előjeles byte
ADAT ENDS
```

- „A” előjel nélküli kiterjesztése:

```
MOV AL, A
XOR AH, AH
```

- „B” előjeles kiterjesztése:

```
MOV AL, B
CBW
```

Szavas és byte-os adatok mozgatása.

Az adatok mozgatásakor, és az adatszegmensből való olvasáskor figyelni kell rá, hogy („db”-vel definiált) byte-os, vagy („dw”-vel definiált) szavas adatot mozgatunk-e.

- A byte-ot fél regiszterbe kell mozgatni (AL, AH, BL, BH, CL, CH, DL, DH).
- Szavas adatot teljes 16 bites regiszterbe kell tenni (AX, BX, CX, CX).

Például: Adott egy adat szegmens:

```
ADAT SEGMENT PARA PUBLIC ’DATA’
    A       db  42      ; egy előjel nélküli byte
    C       dw  -5      ; egy előjeles szó
ADAT ENDS
```

- MOV AL, A ; byte a byte-ba, ok ☺
- MOV CH, A ; byte a byte-ba, ok ☺

- MOV BX, B ; szó a szóba, ok ☺
- MOV AX, A ; byte a szóba, nem jó ☹
- MOV BL, B ; szó a byte-ba, nem jó ☹

Egyszerű „if”-ek megvalósítása.

Az assembly-ben a program másképpen van strukturálva, mint a korábbi nyelveknél. A parancsok végrehajtása szekvenciálisan megy, a processzor alapesetben egymás után hajtja végre az utasításokat a kódban. Ha „if”-et szeretnénk, akkor nem egy blokkot definiálunk, ami lefut, ha egy feltétel teljesül, hanem megírjuk a feltételez tartozó kódot, és ha nem akarjuk, hogy lefusson, akkor egy feltételes ugrással kihagyjuk.

Például:
AX = abs(AX) – abszolút érték számítás

Jó megoldás	Rossz megoldás
CMP AX, 0 JGE címke NEG AX címke:	CMP AX, 0 JL címke címke: NEG AX
Magyarázat: A feltételesen lefuttatandó utasítás a „NEG AX” vagyis AX előjelének megváltoztatása, ha az érték negatív volt. A helyes kódban megfordul a dolog, és a negációt kell kihagyni, ha az AX már amúgy is pozitív.	Magyarázat: A C-s logika azt mondaná, hogy a negálást kell végrehajtani, ha a szám negatív. Ugorjunk hát rá egy feltételes ugrással. Csak az assembly nem így működik. Ennél a megoldásnál A JL-el ugrunk a negációra, ha a szám negatív volt. Ha pedig a szám pozitív, és akkor a processzor továbblép a JL utáni utasításra, ami szintén a negáció. Tehát az mindenképpen lefut, függetlenül attól, hogy pozitív vagy negatív volt-e a szám.

Utasítások paraméterezése.

Sok gond szokott lenni az utasítások paraméterezésével. Általában a nem egyszerű kétoperandusú utasításoknál jön ez elő.

A legjellemzőbb példák:

- MUL:
  - Egy operandusú, a másik operandus mindig az AL vagy az AX
  - Az operandus nem lehet konstans
- DIV:
  - Ugyanaz, mint a MUL-nál
- CBW:
  - Nincs operandusa. Az AL-t terjeszti ki AX-be, másik regiszterrel nem működik.
- Ugró utasítások (JMP, JNE, JG, stb.):
  - Egy paraméterük van, ami egy címke. A program a megadott címkéről fog tovább futni ha az adott ugrás végbemegy.
  - A flags-regiszter tartalma alapján ugrik, vagy nem. Az, hogy a feltétel teljesül-e attól függ, hogy a korábbi utasítások (általában egy CMP) hogyan állították be a flags-et.

16 bites számok osztása

16 bites (szavas) számok osztásánál figyelni kell rá, hogy az osztandó a DX:AX regiszterpárban lesz. Ezért nem elég csak beletenni az osztót az AX-be, de azt ki is kell terjeszteni duplaszóvá.

Pl.:

Az adat szegmens:

```
ADAT SEGMENT PARA PUBLIC ’DATA’
    A       dw  42      ; egy előjeles szó
    B       dw  -5      ; egy előjeles szó
ADAT ENDS
```

„A / B” kiszámítása:

```
MOV AX, A
CWD                     ; AX kiterjesztése DX:AX-be
IDIV B                  ; utána jöhet az osztás.
```