9. Tétel

a. Adatátadás alprogramok között, makrók létrehozása, fordítása:

Adatátadás alprogramok között:

Adatátadás alprogramok között(paraméter átadás): Például van egy C főprogramunk és egy Assembly szubrutinunk, valamilyen módon adatot kell hogy cseréljenek, adatot kell tudniuk átadni egymásnak. Az assembly példaprogramjainkban az adatok átadására regisztereket használtunk. A C programok azonban a vermet(stack-et) használják a paraméterek átadására. Azzal, hogy megadjuk az assembler-nek, hogy C-hez írunk alprogramot, néhány beépített automatizmus egyszerűsíti a program írását.

A paraméterátadás egyszerűbb megértéséhez nézzünk egy egyszerű példát:

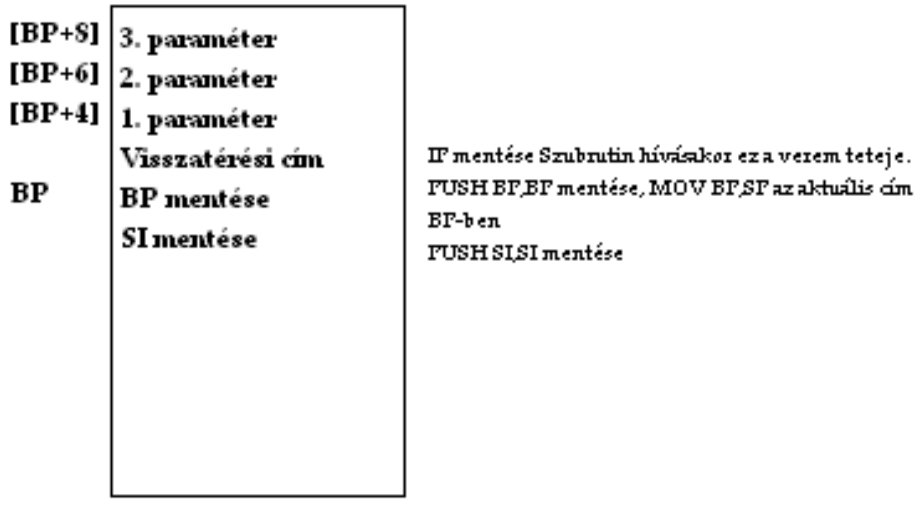
Legyen a C programunk a következő:

```
Main()
{
int param_1, param_2, param_3;
/*Meghívjuk a szubrutin függvényt 3 paraméterrel */
subrutine(param_1,param_2,param_3);
}
```

A subrutine függvény hívásakor a következő történik:

- A jobb szélső paramétertől kezdve betölti a paramétereket a verembe (Legutoljára az első paraméter menti.)
- Menti a visszatérési címet a verembe
- Átadja a vezérlést a függvénynek

Vizsgáljuk meg egy verem képét:



A verem az alacsonyabb memóriacím felé növekszik.
Amennyiben a BP mentése után azonnal betöltjük a veremmutatót(SP) BP-be, utána már szabadon menthetünk bármennyi adatot a stack-re, az már nem befolyásolja a BP-ben elmentett cím és a paraméter távolságát. Azaz az első paraméter mindig azonos ofszetre lesz BP-ben tárolt címtől, MODEL SMALL esetében 4. Mivel távoli (FAR) hívás esetén a CS (kódszegmens) is bekerül a stack-be, ezért itt az ofszet 6 lesz.
A paraméterek által elfoglalt stack nagysága függ a paraméter típusától, amit a szubrutinban is figyelembe kell vennünk. Vagyis a paraméterek címe(az első kivételével) függ attól, hogy milyen típusúak(hány bájtosak).

Mivel MASM generálja- a paraméterátadáshoz szükséges összes – utasítást, a fent említett direktívák használatakor nem kell törődnünk azzal, hogy az utasításokat megfelelő sorrendben írjuk, és azzal sem, hogy a paraméterek melyik veremcímen találhatók.

Érdekesség ként jegyeznénk meg, hogy a paraméterek fordított sorrendbe történő tárolását lehetővé teszi, hogy kevesebb paramétert használjunk a függvényben, mint amennyivel azt meghívtuk.

Nézzünk kétparaméteres adatátadásra példaprogramot. Írjunk egy képernyő-kurzort mozgató rutint**. Hivatkozzunk a már elkészített függvényekre**.

A C főprogram:

```
Main()
{
int x,y;          /*kurzor pozíció*/
x=40;
y=25;
clear_screen();
goto_xy(x,y);
write_string(„Ez egy szöveg”);
}
```

Az assembly rutin:

```
PUBLIC goto_xy
Goto_xy PROC x:WORD, y: WORD
 MOV DH,BYTE PTR(y)                ;Kurzor oszlop koordinátája
 MOV DL,BYTE PTR(x)
MOV BH,0                           ;Képernyő oldalszámának beállítása
MOV AH,2                           ;Kurzorpozíció funkció
INT 10h
RET
Goto_xy ENDP
```

Figyeljük meg, hogy a paraméterek sorrendje megegyezik a hívási sorrenddel. Tehát először az x-et, majd az y-t deklaráltuk. Programunkban egy új különlegességgel találkozunk: BYTE PTR(y)

Híváskor az x és y változókat egészként, paraméter átvételkor pedig word-ként deklaráltuk, ami két-két bájtot jelent. A kurzor címzéséhez pedig egy bájtos adatra van szükség. Így a fordításkor: MOV DH, WORD PTR[PB+4] utasítást kapnánk. Ez azonban az assemblerben nem megengedett. Nem mozgathatunk közvetlen két bájtos adatot egy bájtos helyre. Használnunk kell a BYTE PTR direktívát. Ebben az esetben az utasítás: MOV DH,BYTE PTE WORD PTR[PB +4] lenne. Ezt viszont az assembler nem tudja kezelni. Ezt a problémát úgy tudjuk kiküszöbölni, hogy zárójelezünk. Tehát: MOV DH,BYTE PTE(WORD PTR[PB+4]). Ez pedig visszaírva az eredeti programlistánkba megfelel: MOV DH,BYTE PTE(y) utasításnak. Az assembler ezeket a programozást könnyítő átalakításokat makrók segítségével végzi. Ilyen makró tulajdonképpen az x és y is.

Függvényérték visszaadása: C-ben a függvény érték visszaadásra a következő regisztereket használjuk:

- AL – bájt hosszúságú adatnál,
- AX – szó esetén,
- DX:AX – duplaszavas adatnál.

Szemléltetésül írjunk egy rutint, amely segítségével bekérünk egy karaktert a billentyűzetről.

A C program:

```
Main()
{
clear_screen();
goto_xy(40,25);
write_string(„Ez egy szöveg”);
while (read_key()!=’ ’);
}
```

Assembler rutin:

```
PUBLIC read_key
Read_key PROC
XOR AH,AH
INT 16h
OR AL,AL
JZ ext_kod
XOR AH,AH
RET
Ext_kod:
MOV AL,AH
MOV AH,1
RET
Read_key ENDP
```

Makrók

Gyakori utasítássorozatok egy általunk kitalált utasítással (macro) kiválthatók.
Felhasználásához először deklarálni kell (speciális kezdő- és záró utasítással). A makró törzse

tartalmazza az utasításokat. Az első utasításban megadjuk a nevét, esetleges paramétereit. Felhasználásakor elég csak a nevét leírni. A fordítóprogram a makró előfordulásakor behelyettesíti a törzset a makróutasítás helyére.
Assembly felhasználás: name MACRO - utasítások – ENDM

b. Háttértár kezelése:

Egy assembly programban az INT 25h megszakítás segítségével lehet egy lemezről egy blokkot beolvasni. Ez nem karakteresen történik, hanem blokkokban, ehhez csak a forrás és cél címet, valamint a beolvasandó blokkok számát kell megadni. Egy floppy esetében egy blokk mérete 512 byte, így ennyi helyet kell lefoglalni hozzá az adatszegmensben.

```
.MODEL SMALL
Space EQU " "
.STACK
.DATA?
  Block DB 512 DUP(?)
.CODE
main proc
  MOV AX, Dgroup                ;DS beállítása
  MOV DS,AX
  LEA BX, BLOCK                 ;DS:BX memóriacímre tölti a blokkot
  MOV AL,0                      ;Meghajtó száma
  MOV CX,1                      ;Beolvasott blokkok száma
  MOV DX,0                      ;Lemezolvasás kezdőblokkja
  INT 25h                       ;Beolvasás
  POPF                          ;A veremben tárolt jelzőbitek törlése
  XOR DX,DX                     ;Kiírandó adatok kezdőcíme DS:DX
  CALL write_block
  MOV AH,4Ch                    ;Kilépés
main endp
```

c. Szintaxisfa, levezetési fa:

Legyen G=(N,T,P,S) tetszőleges környezetfüggetlen grammatika. G-beli szintaxis (levezetési) fának nevezünk minden olyan fagráfot, amelynek a csúcsait az $N \cup T$ halmaz elemei jelölik.
Egy mondat szintaxisfája:
A gyökérhez az S kezdőszimbólum tartozik
A levelei a grammatika terminális szimbólumai
A többi pontja a nemterminális szimbólumok halmaza

Egy mondatforma szintaxisfája:
A gyökérhez az S kezdőszimbólum tartozik
A levelei a grammatika terminális és nemterminális szimbólumai
A többi pontja a nemterminális szimbólumok halmaza

A képen látható szintaxisfa a következő környezetfüggetlen grammatikához tartozik: G=[{S,A}, {if, then, else, for, do, a, b, c}, {S→if b then A, S→if b then A else S, S→a, A→ for c do S, A→a}, S].