Eljárások paramétereinek átadási módjai

Az eljárások deklarációjánál nincs mód arra, hogy paramétereket adjunk meg, ezért más, közvetett módon tudunk átadni paramétereket az eljárásoknak. Az eddigi példákban többnyire előre beállítottuk valamelyik regiszter értékét, amit az eljáráson belül felhasználtunk.
Emlékeztetőül:

```
ELJARAS  PROC  NEAR | FAR
         ...
ELJARAS  ENDP
```

Paraméter átadás szabványos adatterületen

A paraméter átadásának egyik lehetséges módja, hogy az adatszegmensen belül elhelyezzük a paramétereket, amelyet az eljárás felhasznál.

A következő példa egy téglalap kerületét számolja ki, melynek oldalai a és b. (Az a és a b érték előjel nélküli bájt.) Az eredményt `AX`-ben kapjuk.

```
ADAT     SEGMENT
a        db 15
b        db 32
ADAT     ENDS
```

```
; a kod szegmens
         ...
; eljaras deklaracio
KERULET  PROC NEAR
         MOV AL, a      ; AL = a
         MOV DL, 2      ; DL = 2
         MUL DL         ; AX = 2*a
         MOV BL, b      ; BL = b
         XOR BH,BH      ; BX = b
         ADD AX,BX      ; AX = AX + BX
         ADD AX,BX      ; ez mar a kerulet
         ; lehetne cserevel es megegy szorzassal is
         RET            ; visszaterunk az eljarashivashoz
KERULET  ENDP
         ...

; a kodreszlet az eljarashivassal
         PUSH AX        ; elmentjuk a korabbi ertekeket
         PUSH BX
         CALL KERULET   ; eljarashivas
         POP BX         ; visszamentjuk BX-et
         ...            ; felhasznaljuk a keruletet
         POP AX         ; kivesszuk a regi AX-et is
         ...
```

Ha a paraméter átadás az adatszegmensben történik, akkor érdemes az eljáráshívás előtt a használandó regiszterek értékét a verembe menteni, majd onnan visszamenteni az eljáráshívás után.

Paraméter átadás vermen keresztül

A paramétereket vermen keresztül is átadhatjuk, nem csak adatterületen. Az eljáráshívás előtt a verembe kell menteni ezeket az értékeket. Vegyük az előző példát, de most a veremben történő paraméter átadással.

```
ADAT    SEGMENT
a       db 15
b       db 32
ADAT    ENDS
```

```
; a kod szegmens
        ...
; eljaras deklaracio
KERULET PROC NEAR
        PUSH BP         ; mentjuk BP erteket
        MOV BP, SP      ; legyen BP a veremmutato erteke
        MOV AX, 6[BP]   ; ez az a parameter
        MOV BX, 4[BP]   ; ez a b parameter
        SHL AX,1        ; AX = AX * 2
        ADD AX,BX       ; AX = AX + BX
        ADD AX,BX       ; ez mar a kerulet
        ; lehetne cserevel es megegy szorzassal is
        POP BP         visszamentjuk BP erteket
        RET 4           ; visszaterunk az eljarashivashoz
                        ; tobbe nem kell sem a sem b a veremben
                        ; nem kerulnek ki a verembol
                        ; csak a veremmutato csokken
KERULET ENDP
        ...

; a kodreszlet az eljarashivassal
        PUSH AX         ; elmentjuk a korabbi ertekeket
        PUSH BX
        MOV AL, a       ; AL = a
        XOR AH,AH       ; AX = a
        MOV BL, b       ; BL = b
        XOR BH,BH       ; BX = a
        PUSH AX         ; a-t a verembe rakjuk
        PUSH BX         ; b-t a verembe rakjuk
        CALL KERULET    ; eljarashivas
        ; a visszatérési cím is bekerül a verembe
        ...
```

Másik példa: egy `osszead` eljárásás, amely a veremben kapja meg a két operandusát

```
          ...
          mov ax, 2d
          push ax
          mov ax, 4d
          push ax
          call osszead
          add sp, 4
          ...
osszead   proc
          push bp          ;bp mentese
          mov bp, sp       ;verem teteje
          push bx          ;bx mentese

          mov bx, [bp+4]   ;2. parameter
          mov ax, [bp+6]   ;1. parameter
          add ax, bx       ;osszeadas

          pop bx           ;bx visszatoltese
          pop bp           ;bp visszatoltese
          ret
osszead   endp
```

A vermen keresztüli paraméterátadásnak akkor van igazán előnye a regiszterekben illetve az adatterületen átadott paraméterekkel szemben, ha sok paramétert kell átadni, valamint rekurzív eljárások paramétereinek átadásánál (lásd 8. előadás: Fibonacci sorozat).

Tekintsük azt a példát, amikor két téglalapról kell eldönteni, hogy átfedőek-e vagy sem. A téglalapok a bal alsó és a jobb felső koordinátáival adottak, ezeket a `t1x1`, `t1x2`, `t1y1`, `t1y2`, `t2x1`, `t2x2`, `t2y1` és `t2y2` előjeles szavas változók tárolják. Nincs annyi regiszterünk, hogy a 8 változót mind külön regiszterben tároljuk. Feltételezzük, hogy a programban van két kiíró eljárás, amely átfedés esetén kiírja, hogy "A ket teglalap atfedo." illetve ha nincs átfedés, akkor a "Nincs atfedes." sztringet írja a képernyőre. Oldaluk illetve sarkuk mentén érintkező téglalapok nem tekintendők átfedőknek.

```
  MOV AX, t1x1          ; az 1. teglalap bal also sarkanak x koordinataja
  PUSH AX
  MOV AX, t1y1          ; az 1. teglalap bal also sarkanak y koordinataja
  PUSH AX
  MOV AX, t1x2          ; az 1. teglalap jobb felso sarkanak x koordinataja
  PUSH AX
  MOV AX, t1y2          ; az 1. teglalap jobb felso sarkanak y koordinataja
  PUSH AX
  MOV AX, t2x1          ; a 2. teglalap bal also sarkanak x koordinataja
  PUSH AX
  MOV AX, t2y1          ; a 2. teglalap bal also sarkanak y koordinataja
  PUSH AX
  MOV AX, t2x2          ; a 2. teglalap jobb felso sarkanak x koordinataja
  PUSH AX
  MOV AX, t2y2          ; a 2. teglalap jobb felso sarkanak y koordinataja
  PUSH AX

  CALL atfedest_vizsgal
  ; a visszatérési cím is bekerül a verembe
  ...

atfedest_vizsgal PROC
  PUSH BP               ; BP erteket menteni kell
  MOV BP, SP            ; a veremmutato erteket eltaroljuk BP-ben
  PUSH AX               ; mentjuk AX erteket
  PUSH BX               ; mentjuk BX erteket
  MOV AX, [BP+14]       ; AX = t1x2
  MOV BX, [BP+10]       ; BX = t2x1
  CMP AX, BX            ; ha t1x2 ≤ t2x1,
  JLE nincs_atfedes     ; akkor nincs atfedes
  MOV AX, [BP+ 6]       ; AX = t2x2
  MOV BX, [BP+18]       ; BX = t1x1
  CMP AX, BX            ; ha t2x2 ≤ t1x1,
  JLE nincs_atfedes     ; akkor nincs atfedes
  MOV AX, [BP+16]       ; AX = t1y1
  MOV BX, [BP+ 4]       ; BX = t2y2
  CMP AX, BX            ; ha t1y1 ≥ t2y2,
  JGE nincs_atfedes     ; akkor nincs atfedes
  MOV AX, [BP+ 8]       ; AX = t2y1
  MOV BX, [BP+12]       ; BX = t1y2
  CMP AX, BX            ; ha t2y1 ≥ t1y2,
  JGE nincs_atfedes     ; akkor nincs atfedes
  ; atfedes van
  CALL kiir_atfedo
  JMP vege              ; ugras a vegere
nincs_atfedes:
  CALL kiir_nem_atfedo
```

(SS:SP)		BX
+2		AX
+4		BP
+6	BP+ 2	visszat. cím
+8	BP+ 4	t2y2
+10	BP+ 6	t2x2
+12	BP+ 8	t2y1
+14	BP+10	t2x1
+16	BP+12	t1y2
+18	BP+14	t1x2
+20	BP+16	t1y1
+22	BP+18	t1x1

```
vege:   POP BX          ; visszamentesek
        POP AX
        POP BP
        RET 16          ; kiuritjuk a parametereket a verembol
atfedest_vizsgal ENDP
        ...
```

Feladatok

1. Írjunk egy eljárást, amely kiszámolja egy sorozat átlagát! Az eljárásnak 2 paramétere legyen (melyeket a veremben kap): az első a sorozat offszetje, a második a sorozat hossza. Kiindulási forrás a gyakorlati munkához: `atlag.asm`

2. Írjunk eljárást, ami 2 n hosszúságú vektor skalárszorzatát határozza meg! A paramétereket a vermen keresztül kapja! (Teljes forráskód: `10skal.asm`)