Eljárások paramétereinek átadási módjai

Az eljárások deklarációjánál nincs mód arra, hogy paramétereket adjunk meg, ezért más, közvetett módon tudunk átadni paramétereket az eljárásoknak. Emlékeztetőül:

```
ELJARAS  PROC
         ...
ELJARAS  ENDP
```

Paraméter átadás regisztereken keresztül

A paraméter átadás egyik legegyszerűbb módja az adatok regisztereken keresztül történő átadása. Ilyenkor kihasználjuk, hogy az általános célú regiszterek tartalma nem változik az eljárások hívásakor, így az eljárás meghívása előtt a regszterekbe tett adatok az eljáráson belül gond nélkül felhasználhatóak.

Az alábbi példa az előadáson is elhangzott skaláris szorzat megoldását mutatja be. Az adatszegmensben definiálásra kerül két vektor, amelyeknek a skaláris szorzatát szeretnénk kiszámítani. Az eljáráshívás előtt ezen vektorok eltolásértékeit beletesszük két tegiszterbe, majd az eljáráson belül ezeket az eltolásokat használjuk az adatok elérésére.

```
.data
    ; elemszám fordítási időben
    ElemSzam EQU SIZEOF Vekt1 / TYPE Vekt1

    Vekt1   DW   1,  2,  3,  4,  5      ; tömb értékek
    Vekt2   DW  24, 23, 22, 21, 20      ;
    Vekt3   DW  90, 91, 92, 93, 94      ;
    Eredm   DD   ?                      ; helyfoglalás eredménynek
```

```
.code
    Main PROC
        MOV  ECX,Elemszam
        MOV  ESI,OFFSET Vekt1
        MOV  EDI,OFFSET Vekt3
        CALL Skal               ; Eljárás meghívása, eredmény
                                ; az EAX regiszterben jön létre
        MOV  Eredm,EAX          ; Mentsük memóriába
        CALL WriteDec           ; És írjuk ki konzolra,
                                ; 1390 lesz az értéke
        CALL CrLf               ; Új sor, kocsi vissza konzolra
        exit
    Main ENDP

    Skal PROC NEAR
        PUSH  EBX
        PUSH  ECX
        PUSH  EDX

        ; Végrehajtó rész kezdete
        PUSH  0             ; Összeg verembe, elfogytak
                            ; a regiszterek!
        JCXZ  kesz          ; Ugrás a kesz címkére,
                            ; ha ECX (= ElemSzam) = 0
    ism:
        MOV   EBX,ECX       ; Vektor index számítása
        DEC   EBX           ; ...
        SHL   EBX,1         ; Index = ( ECX - 1 ) * 2
        MOVZX EAX,WORD PTR [EBX+ESI]; Következő vektor
                            ; elem, WORD PTR kell!
        IMUL  WORD PTR [EBX+EDI]    ; Szorzása másik
                            ; vektor elemével, WORD PTR kell!
        SHL   EDX,16        ; Szorzási átvitel EDX magas
                            ; helyiértékű szavára
        POP   EBX           ; Részösszeg elővétele
        ADD   EBX,EAX       ; Alacsony helyiérték hozzáadása
        ADC   EBX,EDX       ; Magas helyiértékek összeadása
        PUSH  EBX           ; Részösszeg mentése újra
        LOOP  ism           ; Ciklus
    kesz:
```

```
    POP   EAX           ; Eredmény EAX-be kivétele
                        ; Végrehajtó rész vége
    POP   EDX
    POP   ECX
    POP   EBX

    RET
Skal ENDP
```

Paraméter átadás adat szegmensen keresztül

A paraméterek regisztereken keresztül történő átadásával csak annyi probléma van, hogy a regsizterek száma véges. az átadásra ilyenkor használhatjuk az EAX, EBX, ECX, EDX általános regisztereket, az ESI-t, EDI-t, és végszükség esetén akár az EBP-t, de ha hétnél több paramétert kell átadnunk, akkor más megoldást kell találnunk.

Erre adhat egy lehetséges megoldást, ha a paramétereket az eljáráshívás előtt az adat szegmensben helyezzük el, és az eljárás belsejében onnan vesszük ki azokat.

A Skaláris sorzat példa ezen módosulatát mutatja be az alábbi kód.

```
.data
    ; elemszám fordítási időben
    ElemSzam EQU SIZEOF Vekt1 / TYPE Vekt1

    Vekt1   DW   1,  2,  3,  4,  5      ; tömb értékek
    Vekt2   DW  24, 23, 22, 21, 20      ;
    Vekt3   DW  90, 91, 92, 93, 94      ;
    Eredm   DD   ?                      ; helyfoglalás eredménynek

    ; eljárás paraméterek
    param1  DWORD       ?
    param2  DWORD       ?
    param3  DWORD       ?


.code
    Main PROC
        MOV  ECX,Elemszam
        MOV  ESI,OFFSET Vekt1
        MOV  EDI,OFFSET Vekt3
        MOV  param1, ESI
        MOV  param2, EDI
        MOV  param3, ECX
        CALL Skal               ; Eljárás meghívása, eredmény
                                ; az EAX regiszterben jön létre
        MOV  Eredm,EAX          ; Mentsük memóriába
        CALL WriteDec           ; És írjuk ki konzolra,
                                ; 1390 lesz az értéke
        CALL CrLf               ; Új sor, kocsi vissza konzolra
        exit
    Main ENDP

    Skal PROC NEAR
        PUSH  EBX
        PUSH  ECX
        PUSH  EDX

        MOV  ESI, param1
        MOV  EDI, param2
        MOV  ECX, param3
```

```
    ; Végrehajtó rész kezdete
    PUSH  0             ; Összeg verembe, elfogytak
                        ; a regiszterek!
    JCXZ  kesz          ; Ugrás a kesz címkére,
                        ; ha ECX (= ElemSzam) = 0
ism:
    MOV   EBX,ECX       ; Vektor index számítása
    DEC   EBX           ; ...
    SHL   EBX,1         ; Index = ( ECX - 1 ) * 2
    MOVZX EAX,WORD PTR [EBX+ESI]; Következő vektor
                        ; elem, WORD PTR kell!
    IMUL  WORD PTR [EBX+EDI]    ; Szorzása másik
                        ; vektor elemével, WORD PTR kell!
    SHL   EDX,16        ; Szorzási átvitel EDX magas
                        ; helyiérték ű szavára
    POP   EBX           ; Részösszeg elő vétele
    ADD   EBX,EAX       ; Alacsony helyiérték hozzáadása
    ADC   EBX,EDX       ; Magas helyiértékek összeadása
    PUSH  EBX           ; Részösszeg mentése újra
    LOOP  ism           ; Ciklus
kesz:
    POP   EAX           ; Eredmény EAX-be kivétele
                        ; Végrehajtó rész vége
    POP   EDX
    POP   ECX
    POP   EBX

    RET
Skal ENDP
```

A fenti módon definiált eljárás már tetszőleges számú vektor eltolását képes megkapni, és tetszőleges vektorokon képes műveleteket végezni.

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 paraméter átadás legfejlettebb módja a paraméterek vermen keresztül történő átadása. Ilyenkor az eljáráshívás előtt a verembe kell menteni ezeket az értékeket.

Ebben az helyzetben az adatok átadása az alábbi lépésekkel oldható meg.

- A paramétereket elhelyezzük a veremben, és a sorrendjüket feljegyezzük.
- Meghívjuk az eljárás.
- AZ eljárás hívás elején az EBP korábbi értékét elmentjük, és készítünk bele egy másolatot az ESP-ről. Ekkor az EBP egy fix pontként fogja mutatni, hogy az eljárás indulásakor hol volt a verem teteje.
- Az eljárás törzsén belül elvégezzük a számítást. A paramétereket az EBP-hez képest relatív címzéssel tudjuk megtalálni.
- A vermet továbbra is használhatjuk rendeltetés szerűen (akár újabb eljárásáok számára a paraméterek átadására is), mivel az ESP szabad, a paraméterek elérésére az EBP-t használjuk.
- Az eljárás végén visszaállítjuk az EBP korábbi értékét. Ha a hívó kód hasonlóan a vermen keresztül kapta a paramétereit, akkor ezzel visszaállítjuk a paraméterlistáját.
- Visszatérünk.
- A bent maradt paramétereket kitakarítjuk a veremből.

A skaláris szorzat vermet alkalmazó megoldása az elábbi kódban található.

```
.data
    ; elemszám fordítási időben
    ElemSzam  EQU SIZEOF Vekt1 / TYPE Vekt1

    Vekt1     DW   1,  2,  3,  4,  5    ; tömb értékek
    Vekt2     DW  24, 23, 22, 21, 20    ;
    Vekt3     DW  90, 91, 92, 93, 94    ;
    Eredm     DD   ?                    ; helyfoglalás eredménynek

.code
    Main PROC
        MOV  EAX,Elemszam ; Paraméterek verembe helyezése
```

```
    PUSH EAX
    MOV  EAX,OFFSET Vekt1
    PUSH EAX
    MOV  EAX,OFFSET Vekt3
    PUSH EAX
                    ; Verembe került eddig:
                    ; N értéke, Vekt1 címe, Vekt3 címe
                    ; ( 3 * 4 = 12 bájt )
                    ; CALL hatására a verembe kerül még
                    ; a visszatérési cím is (4 bájt)!
    CALL Skal       ; Eljárás meghívása,
                    ; eredmény az EAX regiszterben jön
    ADD  ESP,12     ; Verem takarítás: 3 * 4 bájt
                    ; (A RET már kivette a címet!)
    MOV Eredm,EAX   ; Eredményt mentsük memóriába
    CALL WriteDec   ; És írjuk ki konzolra
    CALL CrLf       ; Új sor, kocsi vissza konzolra
    exit
Main ENDP

Skal PROC NEAR
    PUSH EBP ; EBP mentése
    MOV  EBP,ESP ; Veremmutató bázis regiszterbe töltése
                    ; verem címzéshez

    PUSH ESI ; Regiszterek mentése
    PUSH EDI ; ESP változik, de EBP a belépéskori offszet
                    ; marad!
    PUSH EBX
    PUSH ECX
    PUSH EDX
                    ; A verem tartalma jelenleg
                    ; (alulról felfelé):
                    ;  Paraméterek:
                    ;    N értéke, Vekt1 címe, Vekt3 címe,
                    ;  Visszatérési cím (NEAR címke),
                    ;  Mentett regiszterek:
                    ;  EBP, ESI, EDI, EBX, ECX, EDX

    MOV  EDI,[EBP+8] ; Paraméterek átvétele a veremből
    MOV  ESI,[EBP+12]
```

```
    MOV  ECX,[EBP+16]

    ; Végrehajtó rész kezdete

    ... ; Nincs változás, ugyanaz mint előzőleg

    ; Végrehajtó rész vége

    POP EDX
    POP ECX
    POP EBX
    POP EDI
    POP ESI
    POP EBP

    RET
Skal 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.

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. Kiindulás forrásként használhatod a `Tombok.asm` állományt.

2. Írj rekurzív kódot a Fibonacci számsorozat értékeinek kiszámtítására!