Kódgenerálás II.
(alprogramok, memóriakezelés)

Fordítóprogramok előadás (A,C,T szakirány)

2008. őszi félév

Alprogramok fordítása

- függvény, eljárás, metódus
- paraméterátadási formák
  - érték szerint, hivatkozás szerint...
- tagfüggvények

Függvény, eljárás

- *függvény:*
  - csak „bemenő" paraméterek
  - visszatérési érték
  - nincs mellékhatása

Példa: függvény

```
int duplaja( int n )
{
  return 2*n;
}
```

- *eljárás:*
  - „ki- és bemenő" paraméterek
  - jellemzően nincs visszatérési érték

Példa: eljárás

```
void duplaz( int & n )
{
  n *= 2;
}
```

- *C, C++, Java, ...*: nincs ilyen megkülönböztetés

Alprogramok írása assemblyben

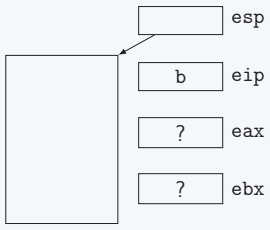
1. paraméter nélküli alprogramok
2. paraméterátadás
3. lokális változók

A `call` és `ret` utasítások

- `call Címke`
  - az eip regiszter tartalmát a verembe teszi
    - ez a call utáni utasítás címe
    - visszatérési címnek nevezzük
  - átadja a vezérlést a `Címke` címkéhez
    - mint egy ugró utasítás
- `ret`
  - kiveszi a verem legfelső négy bájtját és az `eip` regiszterbe teszi
    - mint egy pop utasítás
    - a program a veremben talált címnél folytatódik

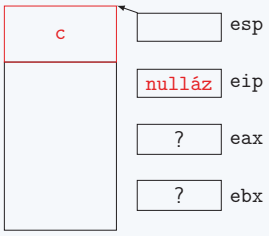
Példa: paraméter nélküli alprogram

```
a:       ...
b:       call nulláz
c:       ...

nulláz:  mov eax,0
d:       mov ebx,0
e:       ret
```



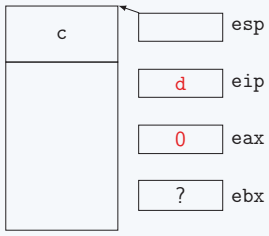
Példa: paraméter nélküli alprogram

```
a:      ...
b:      call nulláz
c:      ...

nulláz: mov eax,0
d:      mov ebx,0
e:      ret
```



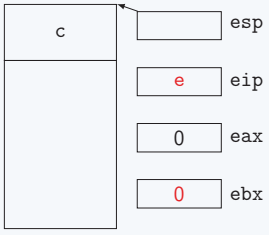
Példa: paraméter nélküli alprogram

```
a:      ...
b:      call nulláz
c:      ...

nulláz: mov eax,0
d:      mov ebx,0
e:      ret
```



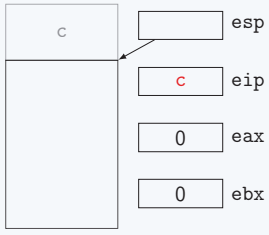
Példa: paraméter nélküli alprogram

```
a:      ...
b:      call nulláz
c:      ...

nulláz: mov eax,0
d:      mov ebx,0
e:      ret
```



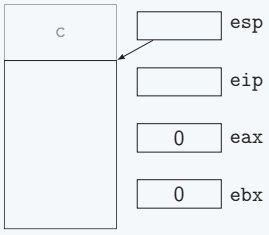
Példa: paraméter nélküli alprogram

```
a:      ...
b:      call nulláz
c:      ...

nulláz: mov eax,0
d:      mov ebx,0
e:      ret
```



Példa: paraméter nélküli alprogram

```
a:      ...
b:      call nulláz
c:      ...

nulláz: mov eax,0
d:      mov ebx,0
e:      ret
```

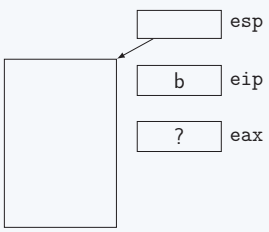


Paraméterek átadása

- a paramétereket a verembe kell tenni a `call` utasítás előtt
- C stílusú paraméterátadás: fordított sorrendben tesszük a verembe
  - az utolsó kerül legalulra
  - az első a verem tetejére
- az ejárásból való visszatérés után a *hívó* állítja vissza a vermet
- visszatérési érték: az eax regiszterbe kerül

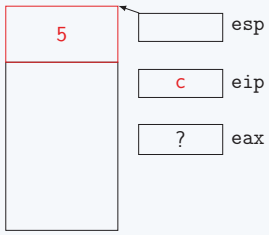
Példa: paraméterátadás (1. változat)

```
a:        ...
b:        push dword 5
c:        call duplája
d:        add esp,4
e:        ...

duplája:  mov eax,[esp+4]
f:        add eax,[esp+4]
g:        ret
```



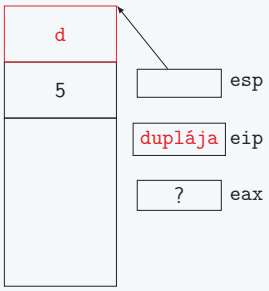
Példa: paraméterátadás (1. változat)

```
a:        ...
b:        push dword 5
c:        call duplája
d:        add esp,4
e:        ...

duplája:  mov eax,[esp+4]
f:        add eax,[esp+4]
g:        ret
```



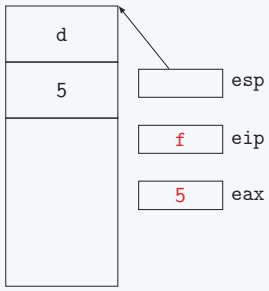
Példa: paraméterátadás (1. változat)

```
a:        ...
b:        push dword 5
c:        call duplája
d:        add esp,4
e:        ...

duplája:  mov eax,[esp+4]
f:        add eax,[esp+4]
g:        ret
```



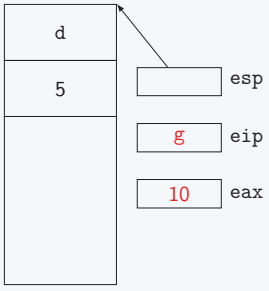
Példa: paraméterátadás (1. változat)

```
a:        ...
b:        push dword 5
c:        call duplája
d:        add esp,4
e:        ...

duplája:  mov eax,[esp+4]
f:        add eax,[esp+4]
g:        ret
```



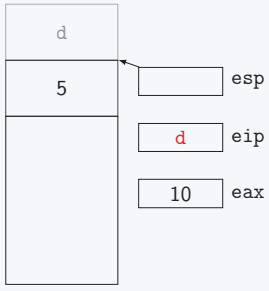
Példa: paraméterátadás (1. változat)

```
a:        ...
b:        push dword 5
c:        call duplája
d:        add esp,4
e:        ...

duplája:  mov eax,[esp+4]
f:        add eax,[esp+4]
g:        ret
```



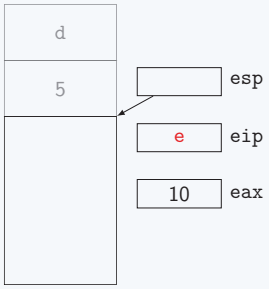
Példa: paraméterátadás (1. változat)

```
a:        ...
b:        push dword 5
c:        call duplája
d:        add esp,4
e:        ...

duplája:  mov eax,[esp+4]
f:        add eax,[esp+4]
g:        ret
```



Példa: paraméterátadás (1. változat)

```
a:        ...
b:        push dword 5
c:        call duplája
d:        add esp,4
e:        ...

duplája:  mov eax,[esp+4]
f:        add eax,[esp+4]
g:        ret
```



Paraméterátadás – 1. változat

- hivatkozás a paraméterekre (ha mindegyik 4 bájtos):
  1.: [esp+4] 2.: [esp+8] ...

Paraméterátadás – 1. változat

- hivatkozás a paraméterekre (ha mindegyik 4 bájtos):
  1.: [esp+4] 2.: [esp+8] ...
- probléma: Időnként használni akarjuk a vermet az alprogram belsejében.
  Például:
  - egyes regiszterek elmentése
  - lokális változóknak helyfoglalás
- Ilyenkor változik a paraméterekre való hivatkozás módja!

Példa: veremhasználat alprogram belsejében

```
alprogram: ; itt [esp+4] az első paraméter
           push ecx
           ; itt már [esp+8]
           pop ecx
           ; itt megint [esp+4]
           ret
```


Paraméterátadás – 2. változat

- Megoldási ötlet: Használjuk az ebp regisztert az esp helyett:
  - az alprogram elején ebp megkapja esp értékét
  - közben esp akárhogy változhat
  - a paramétereket mindig ugyanúgy érjük el:
    [ebp+4], [ebp+8], ...

Példa: ebp használata a paraméterek eléréséhez

```
alprogram: mov ebp,esp
           ; az első paraméter: [ebp+4]
           push ecx
           ; ugyanúgy [ebp+4]
           pop ecx
           ; továbbra is [ebp+4]
           ret
```


Paraméterátadás – 2. változat

- probléma: Alprogramhívás egy alprogram belsejében: elállítja ebp értékét!

Alprogramhívás alprogramban

```
egyik: mov ebp,esp
       ; első paraméter: [ebp+4]
       call masik   ; ez elállítja ebp-t
       ; itt ebp-t már nem tudjuk használni
       ret

masik: mov ebp,esp   ; itt állítjuk el ebp-t
       ; ...
       ret
```


Paraméterátadás – 3. (végső) változat

- Megoldás: Mentsük el ebp értékét az alprogram elején a verembe és állítsuk vissza a végén!

Alprogramhívás alprogramban

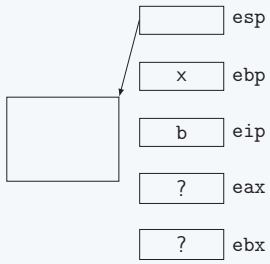
```
egyik: push ebp
       mov ebp,esp
       ; első paraméter: [ebp+8]
       call masik   ; ez megőrzi ebp értékét
       ; itt is [ebp+8] az első paraméter
       pop ebp
       ret

masik: push ebp      ; itt mentjük el ebp-t
       mov ebp,esp
       ; ...
       pop ebp       ; és itt állítjuk vissza
       ret
```

Paraméterátadás – 3. (végső) változat

- Hivatkozás a paraméterekre: [ebp+8], [ebp+12], ...
  - [ebp]: az ebp elmentett értéke
  - [ebp+4]: a visszatérési cím
- Akár rekurzív is lehet az alprogram...

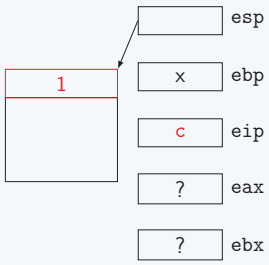
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
        ret
```



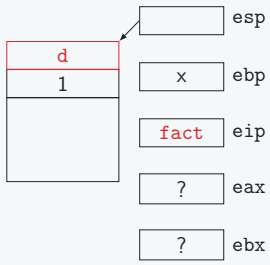
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
        ret
```



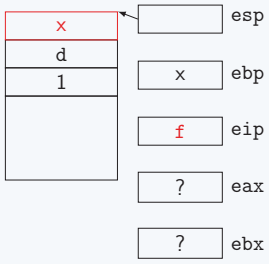
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
        ret
```



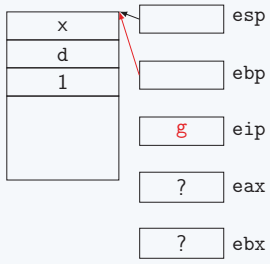
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
        ret
```



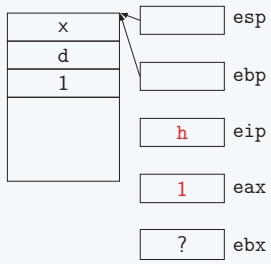
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
        ret
```



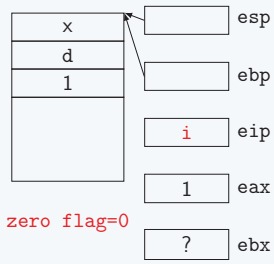
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
        ret
```



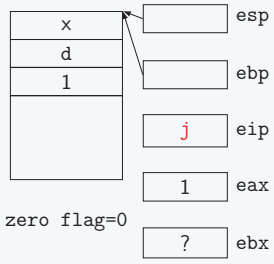
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
        ret
```



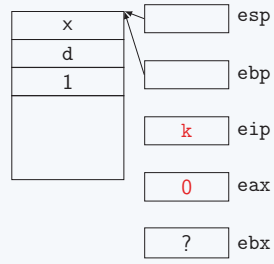
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
        ret
```



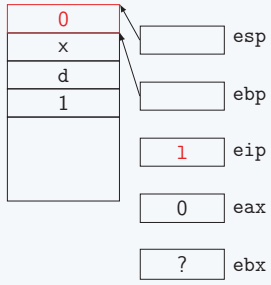
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
        ret
```



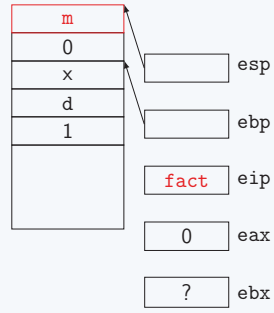
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
        ret
```



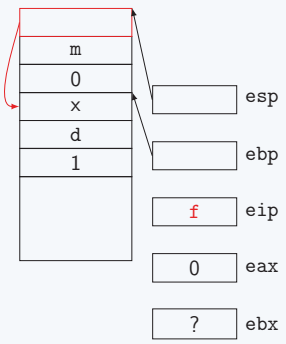
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
        ret
```



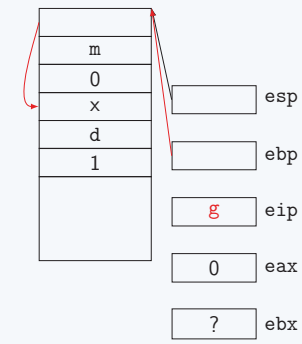
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
        ret
```



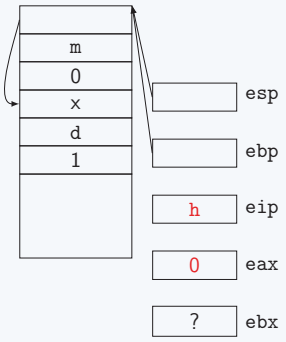
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
        ret
```



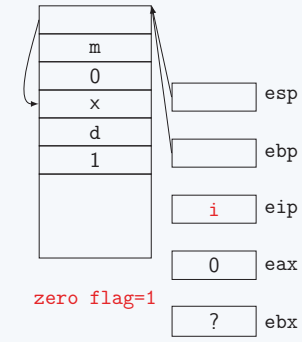
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
        ret
```



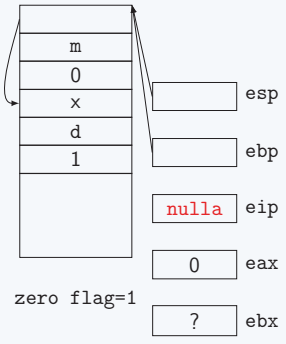
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
        ret
```



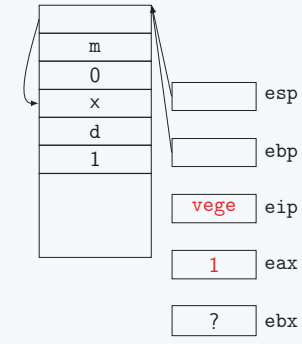
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
        ret
```



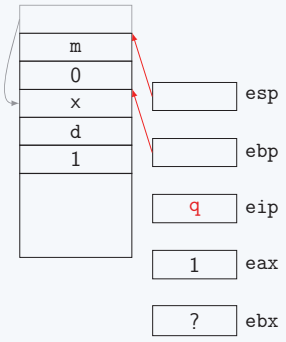
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
        ret
```



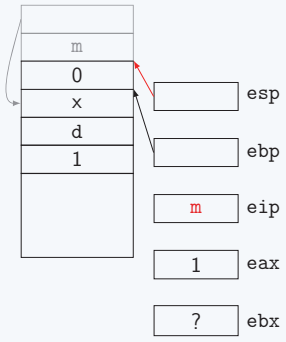
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
q:      ret
```



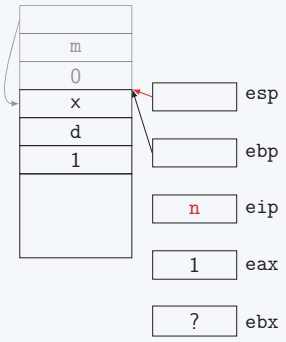
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
q:      ret
```



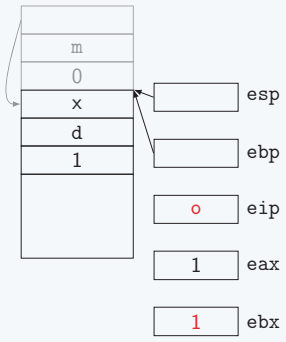
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
q:      ret
```



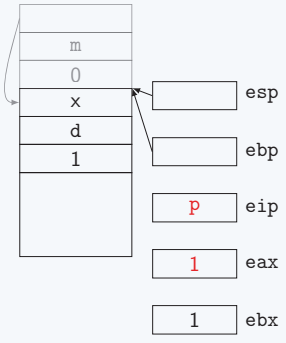
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
q:      ret
```



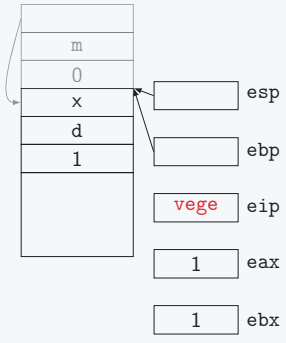
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
q:      ret
```



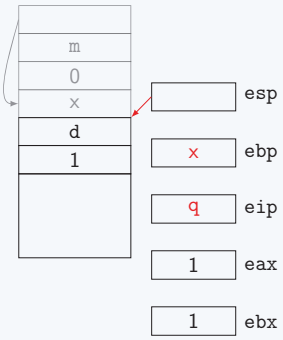
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
q:      ret
```



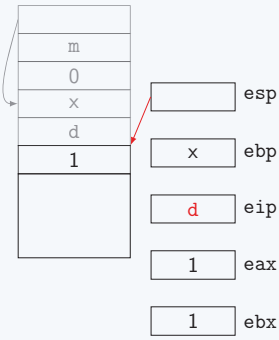
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
q:      ret
```



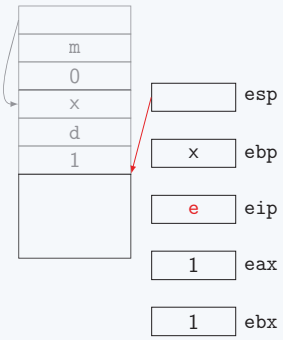
Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
q:      ret
```



Paraméterátadás (3. változat)

```
a:      ...
b:      push dword 1
c:      call fact
d:      add esp,4
e:      ...
fact:   push ebp
f:      mov ebp,esp
g:      mov eax,[ebp+8]
h:      cmp eax,0
i:      je nulla
j:      dec eax
k:      push eax
l:      call fact
m:      add esp,4
n:      mov ebx,[ebp+8]
o:      mul ebx
p:      jmp vege
nulla:  mov eax,1
vege:   pop ebp
q:      ret
```



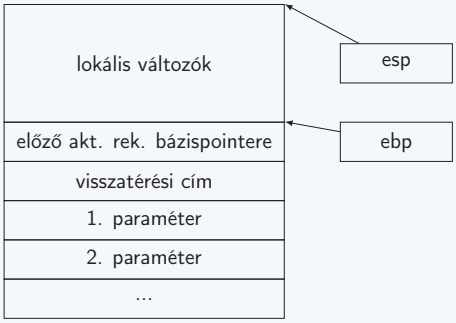
Paraméterátadás – megjegyzések

- minden éppen futó alprogram-példányhoz tartozik egy rész a veremből
  - aktivációs rekord / verem-keret (stack frame)
- az ebp regiszter segítségével érhetők el az aktuális aktivációs rekord részei
  bázis pointer
- az ebp-ből indulva egy láncolt listára vannak felfűzve az aktivációs rekordok
  - ez teszi lehetővé az eggyel korábbi aktivációs rekordhoz való visszatérést az alprogram végén

Lokális változók

- a lokális változók a verem tetejére kerülnek
- hivatkozás a lokális változókra (ha mindegyik 4 bájtos):
  - `[ebp-4]`, `[ebp-8]`, ...
- az esp a lokális változók területe „fölé” mutat
  - lehet veremműveleteket és alprogram-hívásokat végrehajtani a lokális változók felülírása nélkül

Az aktivációs rekord felépítése



Alprogramdefiníciók fordítása

Alprogramok sémája

```
alprogram: push ebp
           mov ebp,esp
           sub esp,'a lokális változók mérete'
           ; az alprogram törzse
vége:      add esp,'a lokális változók mérete'
           pop ebp
           ret
```

- egyedi címkéket kell generálni
- a lokális változók összmérete a törzs szintetizált attribútuma lehet

Lokális változók definíciójának fordítása

- összesíteni kell az alprogramban használt lokális változókat
  - ha mindet az elején kell definiálni, akkor könnyű
  - fel kell venni őket a szimbólumtáblába
  - mindegyikhez ki kell számolni, hogy hol fog elhelyezkedni: `[ebp-?]`
- az összméretükre szükség lesz az alprogram `sub esp,'a lokális változók mérete'` soránál

Lokális változók kiértékelése

- kiegészítjük a kifejezéskiértékelés fordítását egy új esettel: „ha a kifejezés egyetlen lokális változó...”
- a szimbólumtáblából kiolvassuk a pozícióját: p
- a generálandó kód:

Hivatkozás lokális változóra

```
mov eax,[ebp-p]
```


A `return` utasítás fordítása

A '`return` *kifejezés*' utasítás kódja

```
; a kifejezés kiértékelése eax-be
jmp vége
```

- itt is tudni kell az alprogram végét jelző címkét (hasonló problémák, mint a `break` esetén)

Alprogramhívás fordítása

Alprogramok sémája

```
; utolsó paraméter kiértékelése eax-be
push eax
; ...
; 1. paraméter kiértékelése eax-be
push eax
call alprogram
add esp,'a paraméterek összhossza'
```


Paraméterátadás

- érték szerint:
  - a paraméterértékeket másoljuk a verembe
  - ha az alprogram módosítja, az nem hat az átadott változóra
- hivatkozás szerint
  - az átadandó változóra mutató pointert kell a verembe tenni
  - az alprogramban a lokális változó kiértékelése is módosul:

```
mov eax,[ebp+p]
mov eax,[eax]
```

Memóriakezelés

- statikus memóriakezelés:
  - a `.data` vagy `.bss` szakaszban
  - globális vagy statikusnak deklarált változók
  - előre ismerni kell a változók méretét, darabszámát
- dinamikus memóriakezelés
  - blokk-szerkezethez kötődő, lokális változók: *verem*
  - tetszőleges élettartamú változók: *heap memória*

A *heap* memória

- két alapvető művelet: *allokálás* és *felszabadítás*
- az operációs rendszer vagy egy programkönyvtár végzi
- a szabad és foglalt területeket nyilván kell tartani
  - allokáláskor valamilyen stratégia szerint egy szabad területet kell lefoglalni
  - a felszabadított memóriát hozzá kell tenni a szabad területhez
- assemblyből lehet hívni a *C* `malloc` és `free` függvényeit