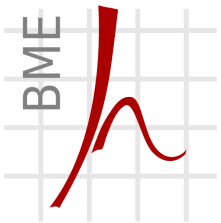
Stack – Vezérlés szerkezet – Adat

Kód visszafejtés.



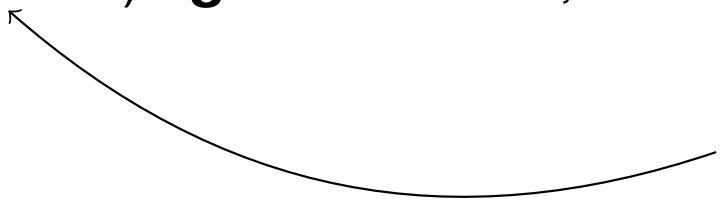
Híradástechnikai Tanszék

Izsó Tamás

2012. október 18.

Rekurzívan pásztázó disassembler összezavarása (obfuscation)

```
int func(int a, int b)
{
    int c=a;
    int d = ~0 ^ c;
    if( ~ c != d ) goto  Junk;
    goto L;
    Junk:
    _asm {
        _emit 0x0f;
    }
    L:
    c= a + b;
    return c-6;
}
```



Átlátszó feltétel

Ollydebug output

```
00401000   55           push ebp
00401001   8B EC        mov ebp,esp
00401003   83 EC 08     sub esp,8
00401006   8B 45 08     mov eax,dword ptr ss:[ebp+8]
00401009   89 45 F8     mov dword ptr ss:[ebp-8],eax
0040100C   8B 4D F8     mov ecx,dword ptr ss:[ebp-8]
0040100F   83 F1 FF     xor ecx,FFFFFFFF
00401012   89 4D FC     mov dword ptr ss:[ebp-4],ecx
00401015   8B 55 F8     mov edx,dword ptr ss:[ebp-8]
00401018   F7 D2        not edx
0040101A   3B 55 FC     cmp edx,dword ptr ss:[ebp-4]
0040101D   74 02        je short 00401021
0040101F   EB 02        jmp short 00401023
00401021   EB 01        jmp short 00401024
00401023   0F           db 0F
00401024   8B 45 08     mov eax,dword ptr ss:[ebp+8]
00401027   03 45 0C     add eax,dword ptr ss:[ebp+0C]
0040102A   89 45 F8     mov dword ptr ss:[ebp-8],eax
0040102D   8B 45 F8     mov eax,dword ptr ss:[ebp-8]
00401030   83 E8 01     sub eax,1
00401033   8B E5        mov esp,ebp
00401035   5D           pop ebp
00401036   C3           retn
```

jól elkülönítette az adatot a kódtól.

IDA disassembler output

```
00000000 55                    push    ebp
00000001 8B EC                 mov     ebp, esp
00000003 83 EC 08              sub     esp, 8
00000006 8B 45 08              mov     eax, [ebp+arg_0]
00000009 89 45 F8              mov     [ebp+var_8], eax
0000000C 8B 4D F8              mov     ecx, [ebp+var_8]
0000000F 83 F1 FF              xor     ecx, 0FFFFFFFFh
00000012 89 4D FC              mov     [ebp+var_4], ecx
00000015 8B 55 F8              mov     edx, [ebp+var_8]
00000018 F7 D2                 not     edx
0000001A 3B 55 FC              cmp     edx, [ebp+var_4]
0000001D 74 02                 jz      short loc_21
0000001F EB 02                 jmp     short loc_23
00000021                       loc_21:
00000021 EB 01                 jmp     short near ptr loc_23+1
00000023                       loc_23:
00000023 0F 8B 45 08+          jnp     near ptr 4503086Eh
00000029 0C 89                 or      al, 89h
0000002B 45                    inc     ebp
0000002C F8                    clc
0000002D 8B 45 F8              mov     eax, [ebp+var_8]
00000030 83 E8 06              sub     eax, 6
00000033 8B E5                 mov     esp, ebp
00000035 5D                    pop     ebp
00000036 C3                    retn
```

hibás visszafejtés

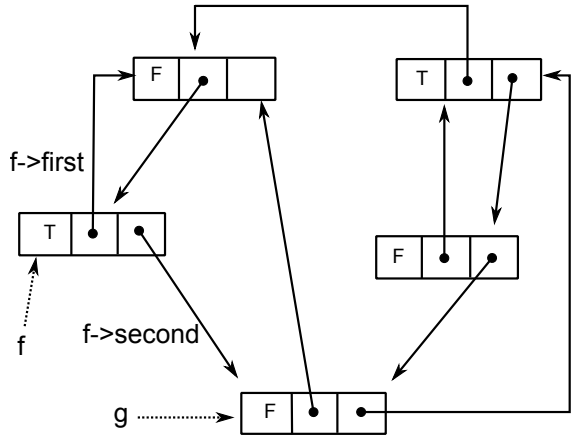
Átlátszó feltétel

Ha feltételes elágazásnál tudjuk, hogy az egyik ág sohasem következhet be, átlátszó feltételről (opaque predicate) beszélünk.

```
if ( i+1 < i ) goto Label;
```

Az előző feltételt könnyű felismerni, és a felesleges kódot kiszűrni. A jó *átlátszó feltétel* értéke fordítási időben ismert, de nehéz észrevenni, hogy a feltétel értéke konstans, annak ellenére, hogy a benne szerepet játszó tagok változnak Ezzel a rekurzív disassemblert megzavarhatjuk, és a visszafejtés idejét megnövelhetjük.

Átlátszó feltétel gráffal megvalósítva



- fordítási időben ismert, hogy a programban adott pontján lévő feltétel igaz vagy hamis;
- fordítási időben ismert, hogy a pointerek hova mutatnak;
- a gráf menetközben változhat.

```
if( f->first->value ) goto Label;
if( f->second == g ) goto Label;
```

Visszatérési cím manipulálása

- A függvény a stack-en lévő visszatérési címet megváltoztatja, így nem a hívás után folytatódik a program.
- A visszatérési érték kiszámításához használhatunk hash táblát, ami a stackre letett visszatérési cím alapján megadja, hogy mennyivel odébb kell folytatni a programot.
- a jump utasításokat is helyettesíthetjük ilyen trükkös függvényhívással.

Visszatérési cím manipulálására példa

```
#include <stdio.h>
void __stdcall[1] f(int k) {
    int* pk=&k; /*stack-en lévő paraméter címe */
    pk--;       /* visszatérési cím */
    *pk+=9;     /* ret addr += 9 */
}
int main() {
  int a=1;
  f(4);
  a++;[2]  /* eax <- a, eax <- eax+1, a <- eax; 9 byte */
  /* ide térünk vissza */
  printf("%d\n", a);
  return 0;
}
```

[1] stack kezelés következő órán

[2] optimalizálás nélkül fordítva

Miért olyan okos az Ollydebug

Különböző heurisztikákat alkalmaz.

- 12 menetben analizálja a kódot;
- utasítás kezdet tesztelése, nem lehet olyan helyen, ami a program memóriában elfoglalt helyétől függ, nem lehet olyan ugrás, aminek a címe kimutat a címtartományból, stb.
- kettőnél többször történik ugrás egy területre, az 0.99 valószínűséggel kód,
- függvényhívásnál a paraméterek elmentése, lokális változók helyfoglalás, stb. árulkodó lehet.
- 2200 rendszerhívást ismer;
- 7800 konstans szimbolikus nevét felismeri;
- for, while, stb vezérlésszerkezetet felismeri;

Gépi program értelmezése

Program visszafejtésnél nem az a cél, hogy a gépi utasítások alapján rekonstruáljuk az eredeti magas szintű nyelven íródott programot, hanem az, hogy megértsük a program működésének a logikáját.

Programot visszafejteni nem egy bonyolult dolog, a lényeg, hogy felismerjük azokat az utasítás mintákat, amit a fordító generál. Ugyanakkor ez sok gyakorlást igényel.

Section 1

Stack kezelés

Stack feladata

Eljárás paraméterei és lokális változóinak az élettertama az eljárás futási idejére korlátozódik. A fordító ezeknek a változóknak csak a függvény hívásától a visszatéréséig foglal helyet, fordítóírók terminológiájával élve az aktivációs rekord létrehozásával. IA-32 architekturán ez a stacket jelenti.

Stack működésének a részei

- aktív regiszter értékének ideiglenes tárolása;
- paraméterek átadása;
- visszatérési cím elmentése;
- lokális adatok helyének a biztosítása;

A függvények hívásához különböző hívási konvekciókat használhatunk.

Stackkezelő utasítások

```
push eax
```

```
sub esp, 4
mov dword ptr [esp], eax
```

```
pop eax
```

```
mov eax, dword ptr [esp]
add esp, 4
```

Call és return

Eljárás hívás (kódszegmens nem változik)

1. **push** EIP.
2. **jmp** func

Visszatérés a hívóhoz **ret** n:

1. **pop** EIP.
2. **add ESP**, n

C példaprogram

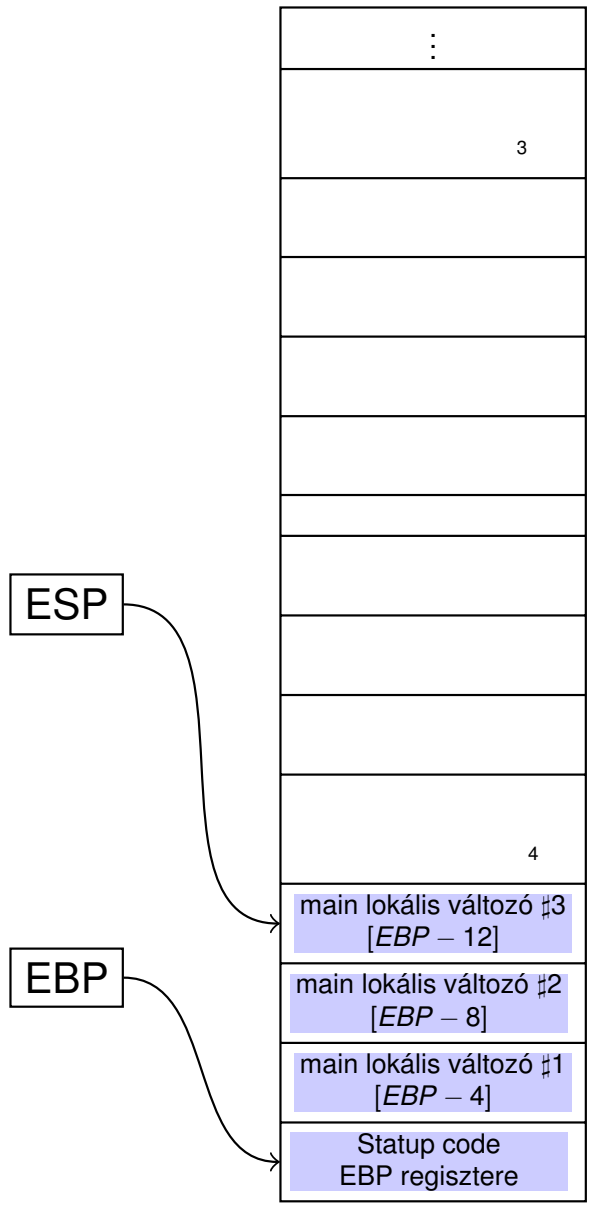
```c
int func(int i, int j, int k )
{
    double r;
    int l=0;
    l = i;
    l +=j;
    l +=k;
    return l;
}

int main()
{
    int a=1;
    int b=2;
    int c=3;
    a=func(a,b,c);
    return 0;
}
```

__cdecl hívási konvenció, hívó feladata

```
_main:
    push    ebp
    mov     ebp,esp
    sub     esp,0Ch
    mov     dword ptr [ebp-4],1
    mov     dword ptr [ebp-8],2
    mov     dword ptr [ebp-0Ch],3
    mov     eax,dword ptr [ebp-0Ch]
    push    eax
    mov     ecx,dword ptr [ebp-8]
    push    ecx
    mov     edx,dword ptr [ebp-4]
    push    edx
    call    _func
    add     esp,0Ch
    mov     dword ptr [ebp-4],eax
    xor     eax,eax
    mov     esp,ebp
    pop     ebp
    ret
```

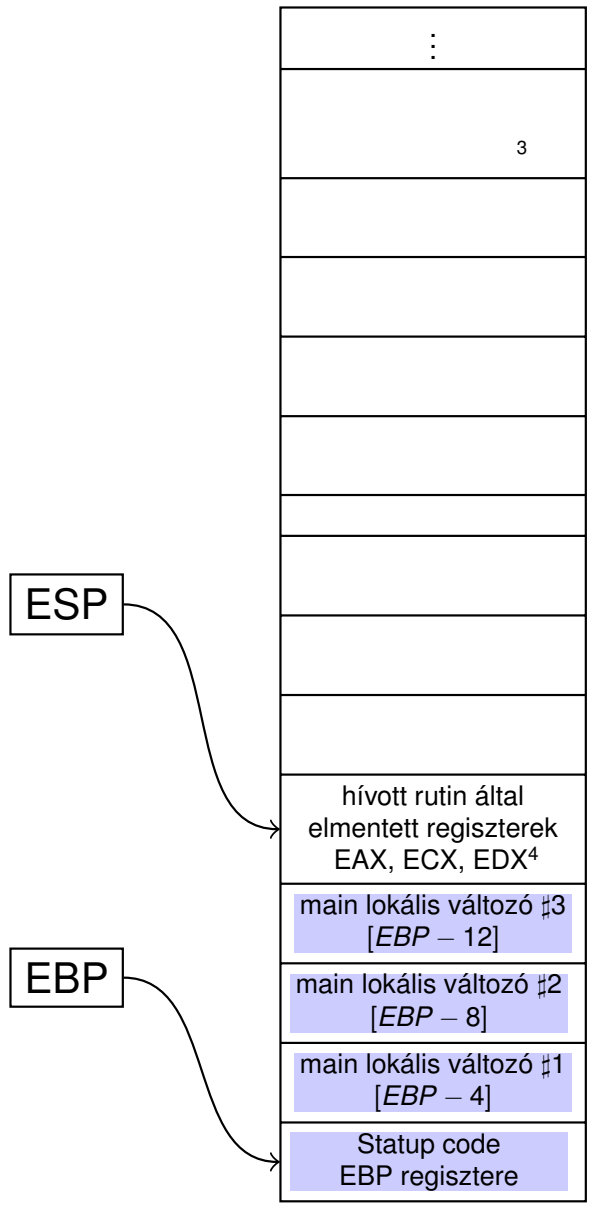
[3] csak ha szükséges



__cdecl hívási konvenció, hívó feladata

```
_main:
    push    ebp
    mov     ebp,esp
    sub     esp,0Ch
    mov     dword ptr [ebp-4],1
    mov     dword ptr [ebp-8],2
    mov     dword ptr [ebp-0Ch],3
    mov     eax,dword ptr [ebp-0Ch]
    push    eax
    mov     ecx,dword ptr [ebp-8]
    push    ecx
    mov     edx,dword ptr [ebp-4]
    push    edx
    call    _func
    add     esp,0Ch
    mov     dword ptr [ebp-4],eax
    xor     eax,eax
    mov     esp,ebp
    pop     ebp
    ret
```

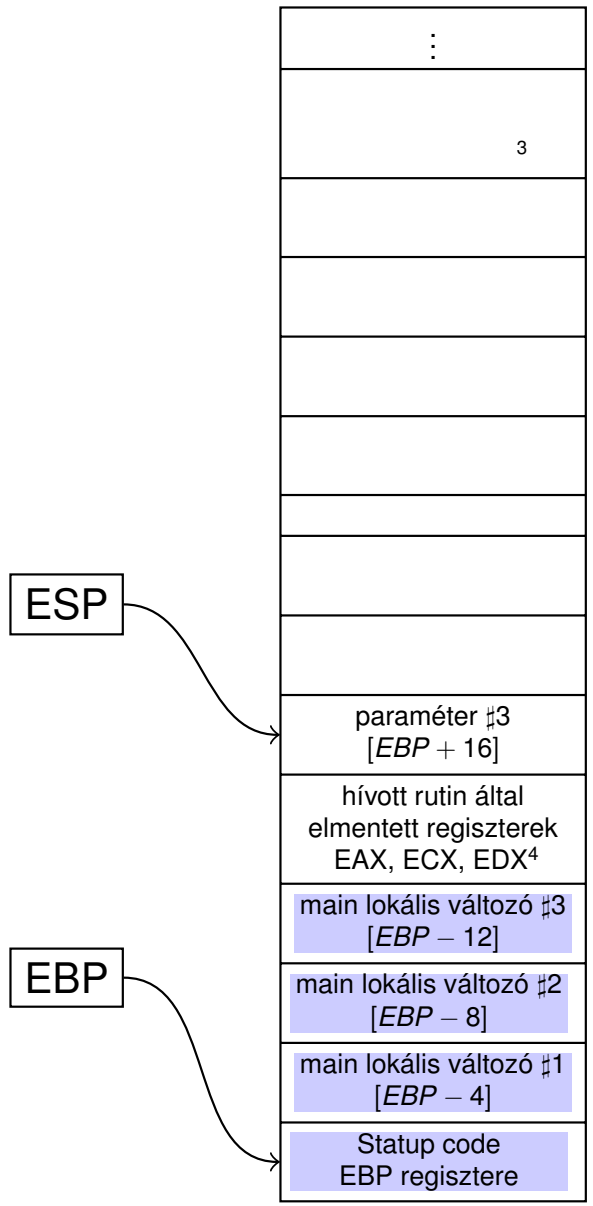
[3] csak ha szükséges



__cdecl hívási konvenció, hívó feladata

```
_main:
    push    ebp
    mov     ebp,esp
    sub     esp,0Ch
    mov     dword ptr [ebp-4],1
    mov     dword ptr [ebp-8],2
    mov     dword ptr [ebp-0Ch],3
    mov     eax,dword ptr [ebp-0Ch]
    push    eax
    mov     ecx,dword ptr [ebp-8]
    push    ecx
    mov     edx,dword ptr [ebp-4]
    push    edx
    call    _func
    add     esp,0Ch
    mov     dword ptr [ebp-4],eax
    xor     eax,eax
    mov     esp,ebp
    pop     ebp
    ret
```

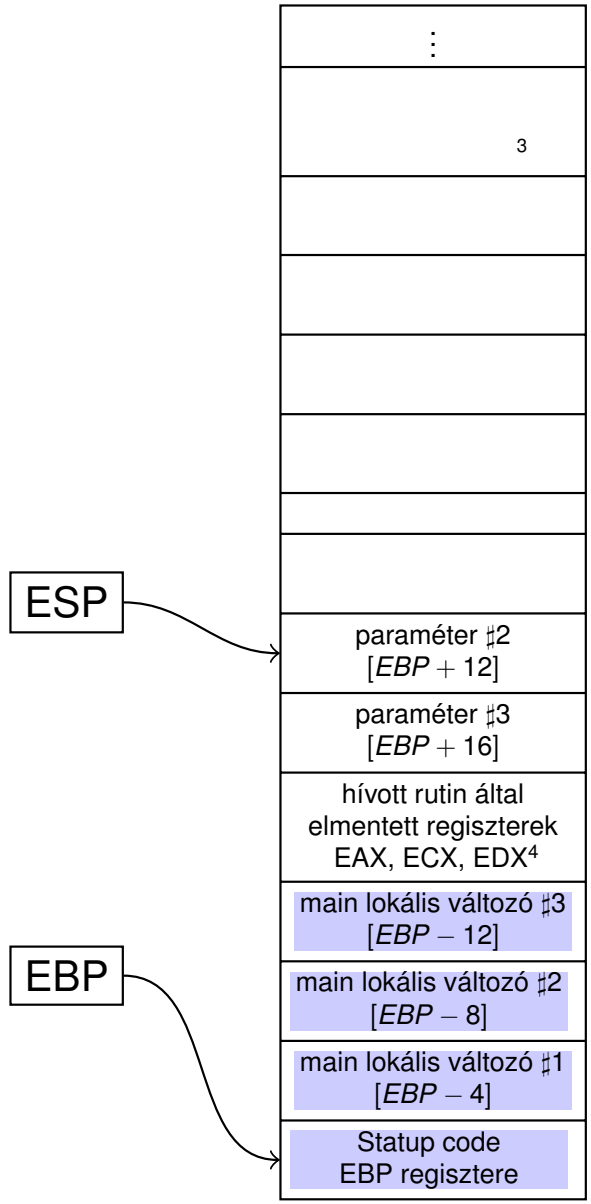
[3] csak ha szükséges



__cdecl hívási konvenció, hívó feladata

```
_main:
    push    ebp
    mov     ebp,esp
    sub     esp,0Ch
    mov     dword ptr [ebp-4],1
    mov     dword ptr [ebp-8],2
    mov     dword ptr [ebp-0Ch],3
    mov     eax,dword ptr [ebp-0Ch]
    push    eax
    mov     ecx,dword ptr [ebp-8]
    push    ecx
    mov     edx,dword ptr [ebp-4]
    push    edx
    call    _func
    add     esp,0Ch
    mov     dword ptr [ebp-4],eax
    xor     eax,eax
    mov     esp,ebp
    pop     ebp
    ret
```

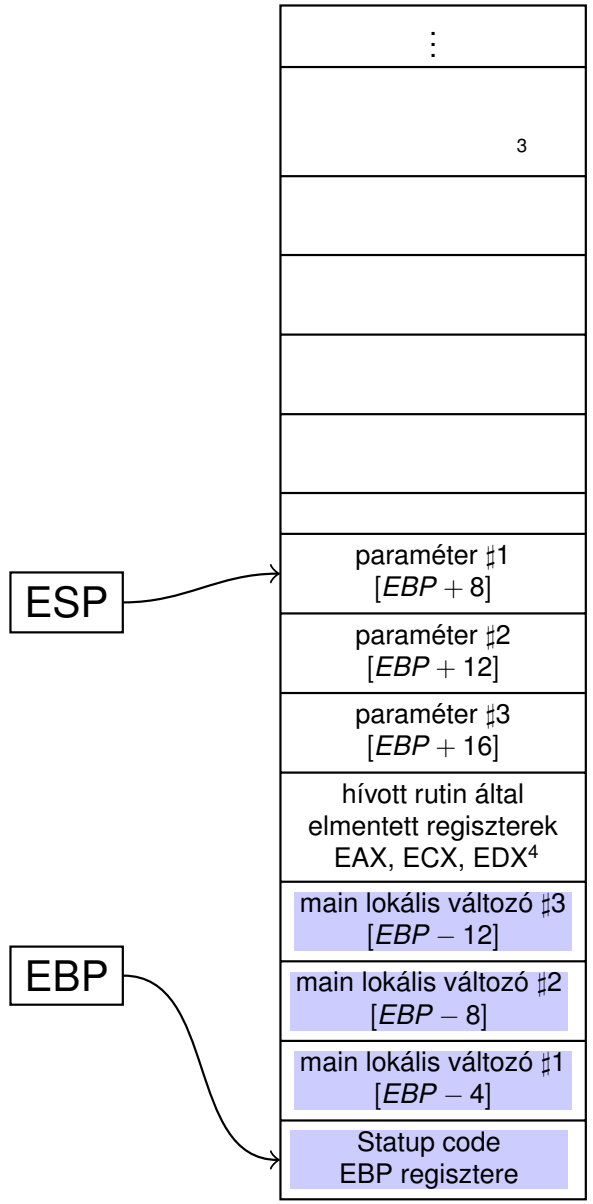
[3] csak ha szükséges



__cdecl hívási konvenció, hívó feladata

```
_main:
    push    ebp
    mov     ebp,esp
    sub     esp,0Ch
    mov     dword ptr [ebp-4],1
    mov     dword ptr [ebp-8],2
    mov     dword ptr [ebp-0Ch],3
    mov     eax,dword ptr [ebp-0Ch]
    push    eax
    mov     ecx,dword ptr [ebp-8]
    push    ecx
    mov     edx,dword ptr [ebp-4]
    push    edx
    call    _func
    add     esp,0Ch
    mov     dword ptr [ebp-4],eax
    xor     eax,eax
    mov     esp,ebp
    pop     ebp
    ret
```

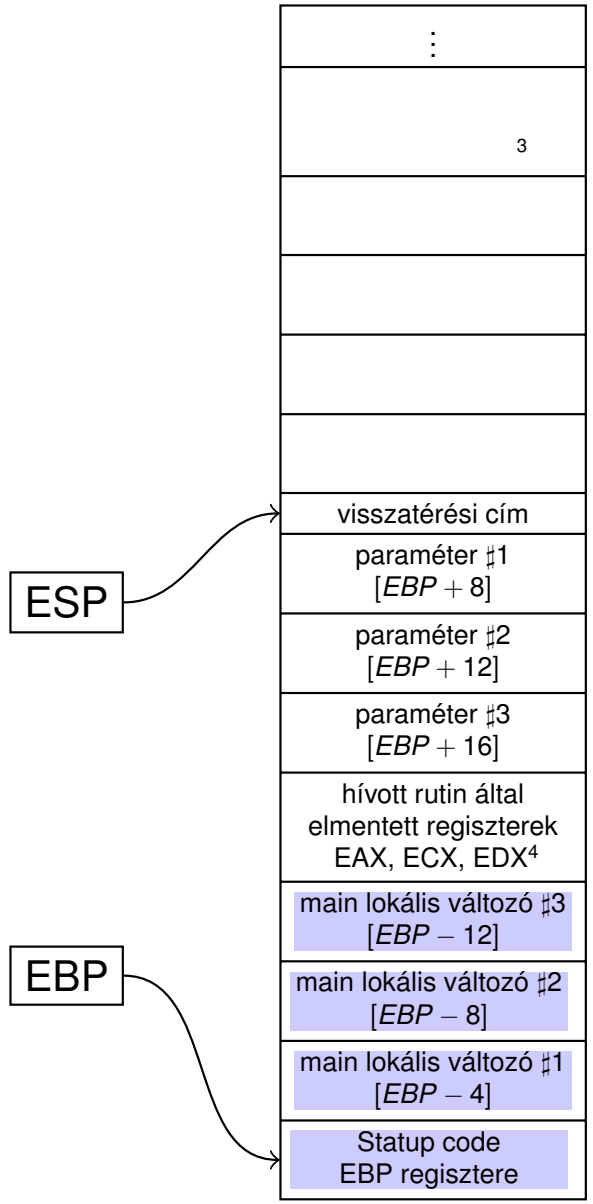
[3] csak ha szükséges



__cdecl hívási konvenció, hívó feladata

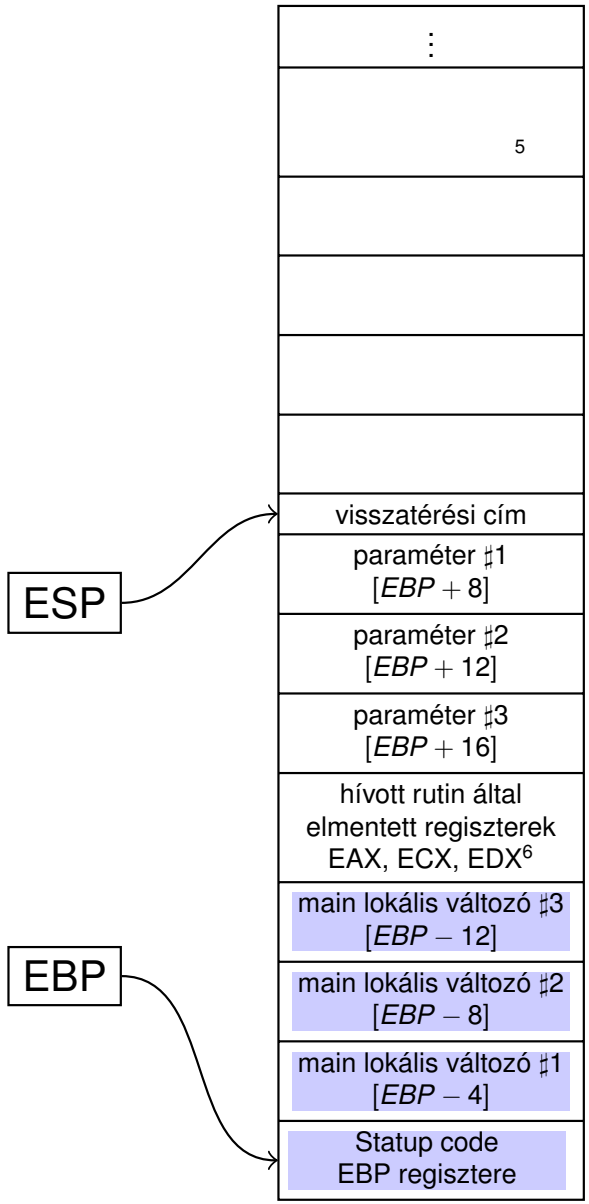
```
_main:
    push    ebp
    mov     ebp,esp
    sub     esp,0Ch
    mov     dword ptr [ebp-4],1
    mov     dword ptr [ebp-8],2
    mov     dword ptr [ebp-0Ch],3
    mov     eax,dword ptr [ebp-0Ch]
    push    eax
    mov     ecx,dword ptr [ebp-8]
    push    ecx
    mov     edx,dword ptr [ebp-4]
    push    edx
    call    _func
    add     esp,0Ch
    mov     dword ptr [ebp-4],eax
    xor     eax,eax
    mov     esp,ebp
    pop     ebp
    ret
```

[3] csak ha szükséges



__cdecl hívási konvenció, hívott feladata

```
_func:
    push    ebp
    mov     ebp,esp
    sub     esp,10h
    mov     dword ptr [ebp-4],0
    mov     eax,dword ptr [ebp+8]
    mov     dword ptr [ebp-4],eax
    mov     ecx,dword ptr [ebp-4]
    add     ecx,dword ptr [ebp+0Ch]
    mov     dword ptr [ebp-4],ecx
    mov     edx,dword ptr [ebp-4]
    add     edx,dword ptr [ebp+10h]
    mov     dword ptr [ebp-4],edx
    mov     eax,dword ptr [ebp-4]
    mov     esp,ebp
    pop     ebp
    ret
```

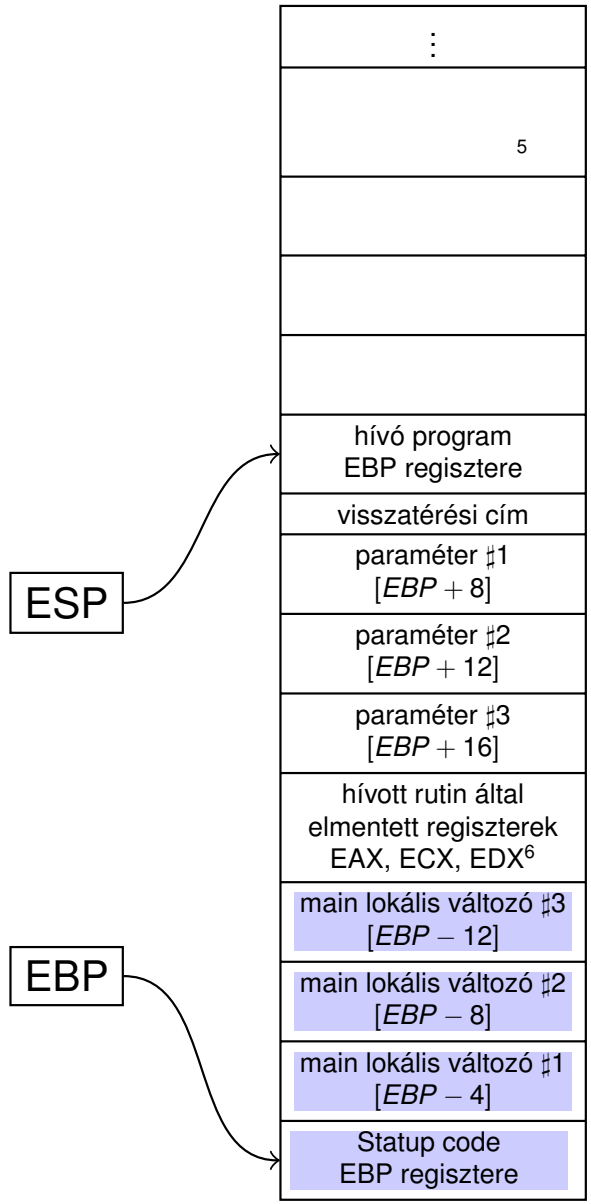


[5] csak ha szükséges

__cdecl hívási konvenció, hívott feladata

```
_func:
    push    ebp
    mov     ebp,esp
    sub     esp,10h
    mov     dword ptr [ebp-4],0
    mov     eax,dword ptr [ebp+8]
    mov     dword ptr [ebp-4],eax
    mov     ecx,dword ptr [ebp-4]
    add     ecx,dword ptr [ebp+0Ch]
    mov     dword ptr [ebp-4],ecx
    mov     edx,dword ptr [ebp-4]
    add     edx,dword ptr [ebp+10h]
    mov     dword ptr [ebp-4],edx
    mov     eax,dword ptr [ebp-4]
    mov     esp,ebp
    pop     ebp
    ret
```

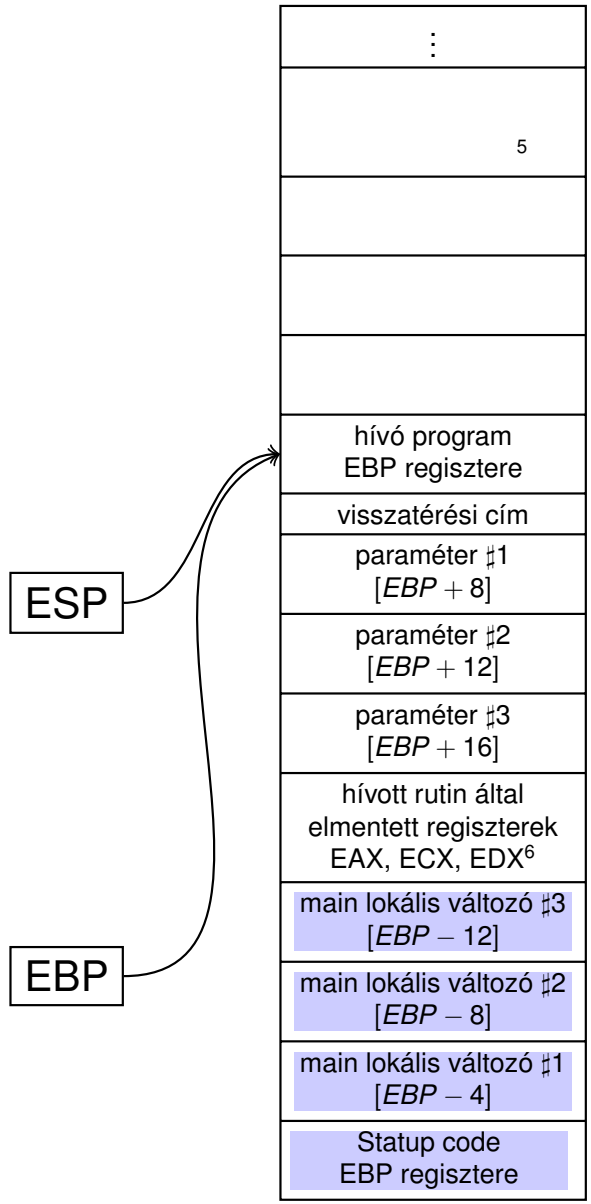
[5] csak ha szükséges



__cdecl hívási konvenció, hívott feladata

```
_func:
    push    ebp
    mov     ebp,esp
    sub     esp,10h
    mov     dword ptr [ebp-4],0
    mov     eax,dword ptr [ebp+8]
    mov     dword ptr [ebp-4],eax
    mov     ecx,dword ptr [ebp-4]
    add     ecx,dword ptr [ebp+0Ch]
    mov     dword ptr [ebp-4],ecx
    mov     edx,dword ptr [ebp-4]
    add     edx,dword ptr [ebp+10h]
    mov     dword ptr [ebp-4],edx
    mov     eax,dword ptr [ebp-4]
    mov     esp,ebp
    pop     ebp
    ret
```

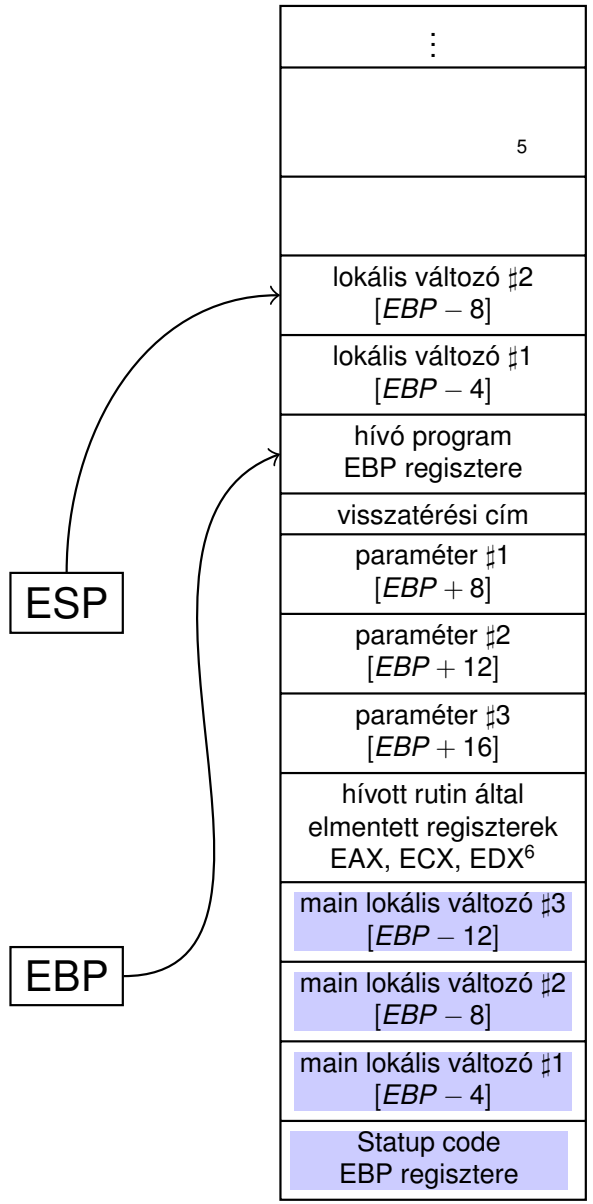
[5] csak ha szükséges



__cdecl hívási konvenció, hívott feladata

```
_func:
    push    ebp
    mov     ebp,esp
    sub     esp,10h
    mov     dword ptr [ebp-4],0
    mov     eax,dword ptr [ebp+8]
    mov     dword ptr [ebp-4],eax
    mov     ecx,dword ptr [ebp-4]
    add     ecx,dword ptr [ebp+0Ch]
    mov     dword ptr [ebp-4],ecx
    mov     edx,dword ptr [ebp-4]
    add     edx,dword ptr [ebp+10h]
    mov     dword ptr [ebp-4],edx
    mov     eax,dword ptr [ebp-4]
    mov     esp,ebp
    pop     ebp
    ret
```

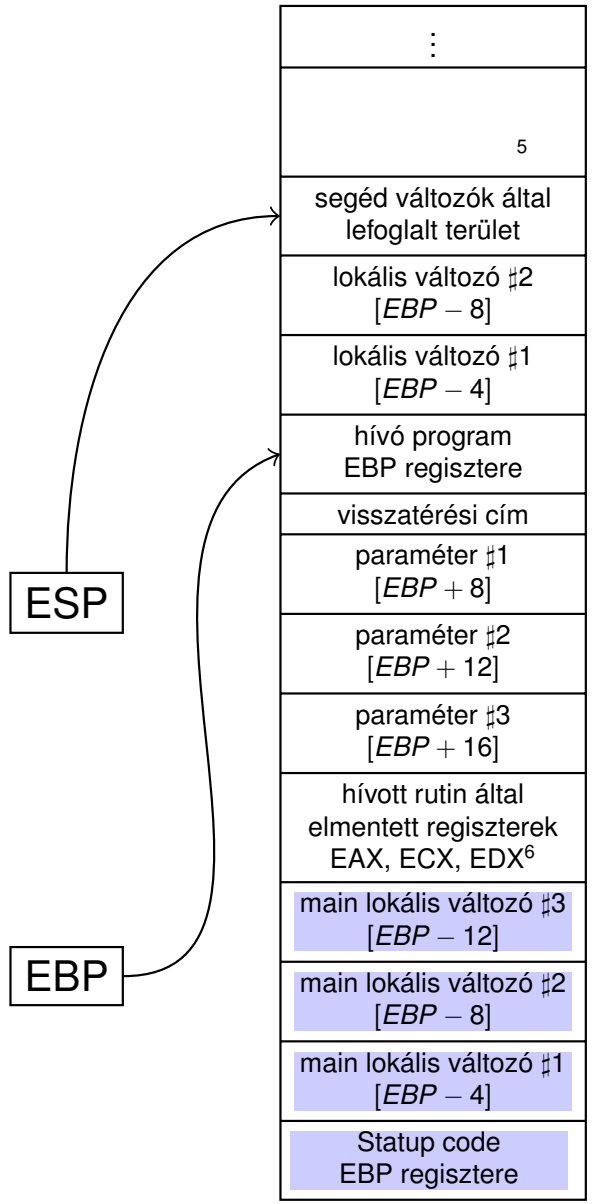
[5] csak ha szükséges



__cdecl hívási konvenció, hívott feladata

```
_func:
    push    ebp
    mov     ebp,esp
    sub     esp,10h
    mov     dword ptr [ebp-4],0
    mov     eax,dword ptr [ebp+8]
    mov     dword ptr [ebp-4],eax
    mov     ecx,dword ptr [ebp-4]
    add     ecx,dword ptr [ebp+0Ch]
    mov     dword ptr [ebp-4],ecx
    mov     edx,dword ptr [ebp-4]
    add     edx,dword ptr [ebp+10h]
    mov     dword ptr [ebp-4],edx
    mov     eax,dword ptr [ebp-4]
    mov     esp,ebp
    pop     ebp
    ret
```

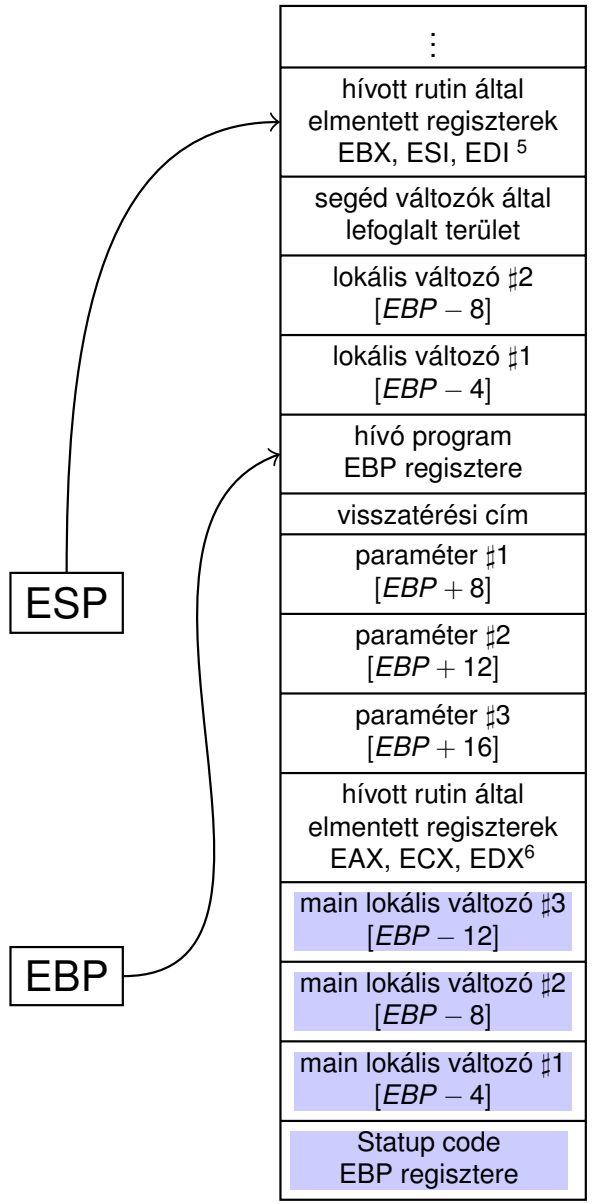
[5] csak ha szükséges



__cdecl hívási konvenció, hívott feladata

```
_func:
    push    ebp
    mov     ebp,esp
    sub     esp,10h
    mov     dword ptr [ebp-4],0
    mov     eax,dword ptr [ebp+8]
    mov     dword ptr [ebp-4],eax
    mov     ecx,dword ptr [ebp-4]
    add     ecx,dword ptr [ebp+0Ch]
    mov     dword ptr [ebp-4],ecx
    mov     edx,dword ptr [ebp-4]
    add     edx,dword ptr [ebp+10h]
    mov     dword ptr [ebp-4],edx
    mov     eax,dword ptr [ebp-4]
    mov     esp,ebp
    pop     ebp
    ret
```

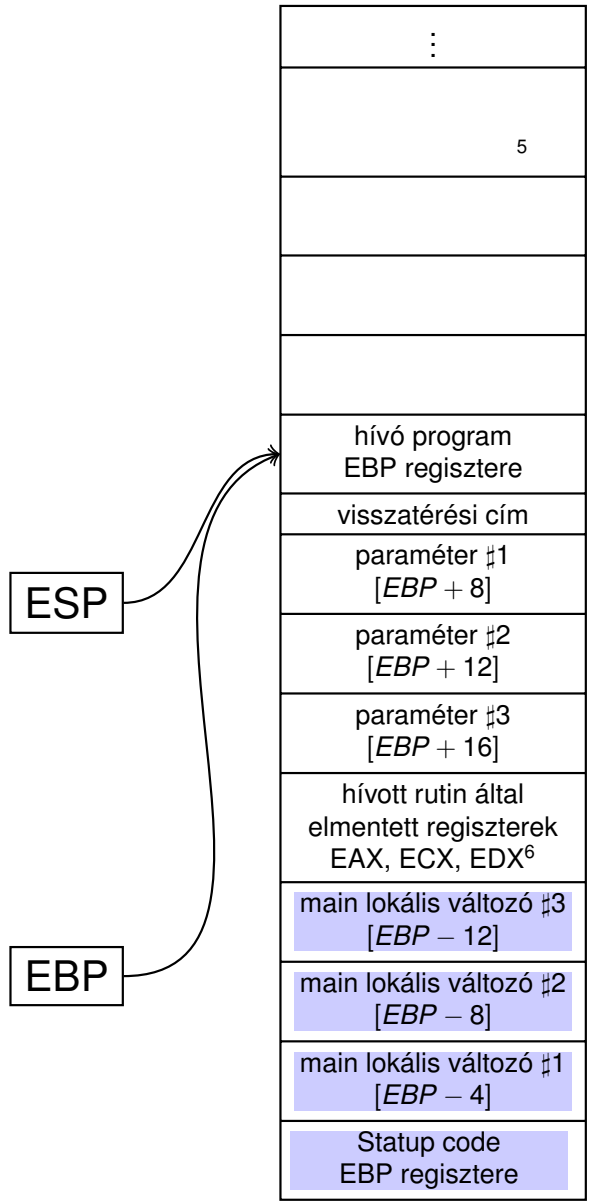
[5] csak ha szükséges



__cdecl hívási konvenció, hívott feladata

```
_func:
    push    ebp
    mov     ebp,esp
    sub     esp,10h
    mov     dword ptr [ebp-4],0
    mov     eax,dword ptr [ebp+8]
    mov     dword ptr [ebp-4],eax
    mov     ecx,dword ptr [ebp-4]
    add     ecx,dword ptr [ebp+0Ch]
    mov     dword ptr [ebp-4],ecx
    mov     edx,dword ptr [ebp-4]
    add     edx,dword ptr [ebp+10h]
    mov     dword ptr [ebp-4],edx
    mov     eax,dword ptr [ebp-4]
    mov     esp,ebp
    pop     ebp
    ret
```

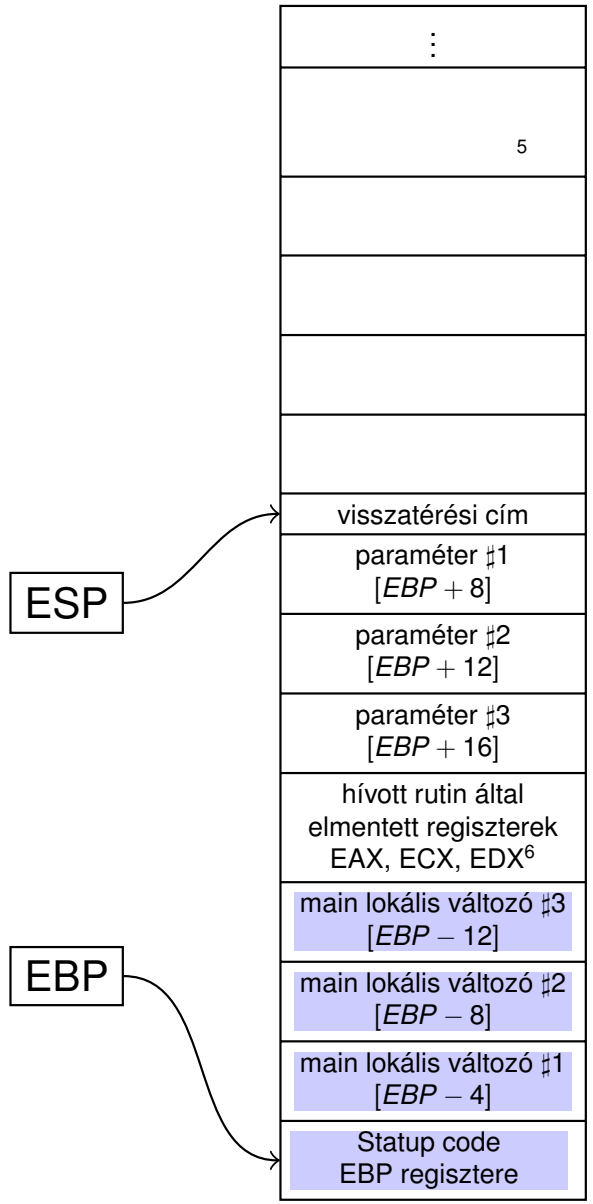
[5] csak ha szükséges



__cdecl hívási konvenció, hívott feladata

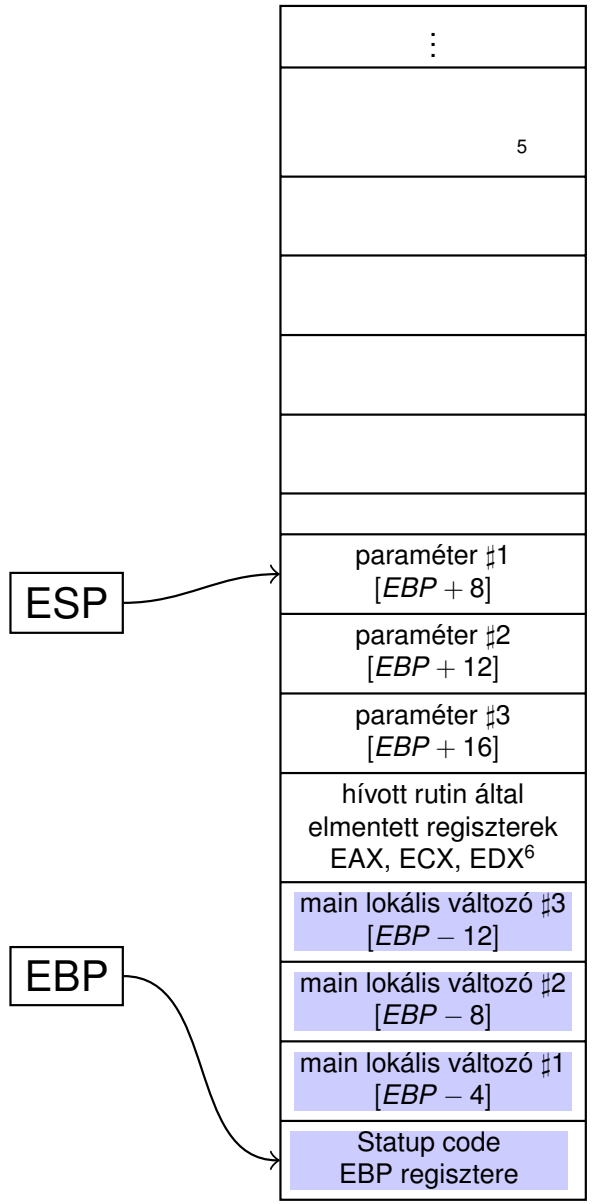
```
_func:
    push    ebp
    mov     ebp,esp
    sub     esp,10h
    mov     dword ptr [ebp-4],0
    mov     eax,dword ptr [ebp+8]
    mov     dword ptr [ebp-4],eax
    mov     ecx,dword ptr [ebp-4]
    add     ecx,dword ptr [ebp+0Ch]
    mov     dword ptr [ebp-4],ecx
    mov     edx,dword ptr [ebp-4]
    add     edx,dword ptr [ebp+10h]
    mov     dword ptr [ebp-4],edx
    mov     eax,dword ptr [ebp-4]
    mov     esp,ebp
    pop     ebp
    ret
```

[5] csak ha szükséges



__cdecl hívási konvenció, hívott feladata

```
_func:
    push    ebp
    mov     ebp,esp
    sub     esp,10h
    mov     dword ptr [ebp-4],0
    mov     eax,dword ptr [ebp+8]
    mov     dword ptr [ebp-4],eax
    mov     ecx,dword ptr [ebp-4]
    add     ecx,dword ptr [ebp+0Ch]
    mov     dword ptr [ebp-4],ecx
    mov     edx,dword ptr [ebp-4]
    add     edx,dword ptr [ebp+10h]
    mov     dword ptr [ebp-4],edx
    mov     eax,dword ptr [ebp-4]
    mov     esp,ebp
    pop     ebp
    ret
```

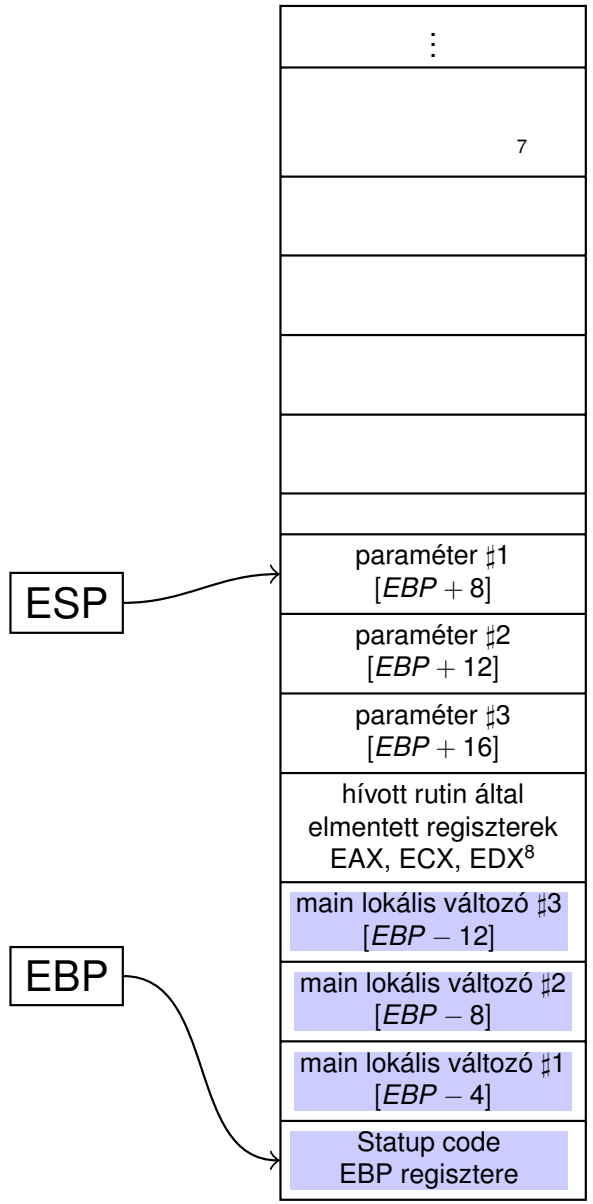


[5] csak ha szükséges

__cdecl hívási konvenció, visszatérés a hívóhoz

```
_main:
    push    ebp
    mov     ebp,esp
    sub     esp,0Ch
    mov     dword ptr [ebp-4],1
    mov     dword ptr [ebp-8],2
    mov     dword ptr [ebp-0Ch],3
    mov     eax,dword ptr [ebp-0Ch]
    push    eax
    mov     ecx,dword ptr [ebp-8]
    push    ecx
    mov     edx,dword ptr [ebp-4]
    push    edx
    call    _func
    add     esp,0Ch
    mov     dword ptr [ebp-4],eax
    xor     eax,eax
    mov     esp,ebp
    pop     ebp
    ret
```

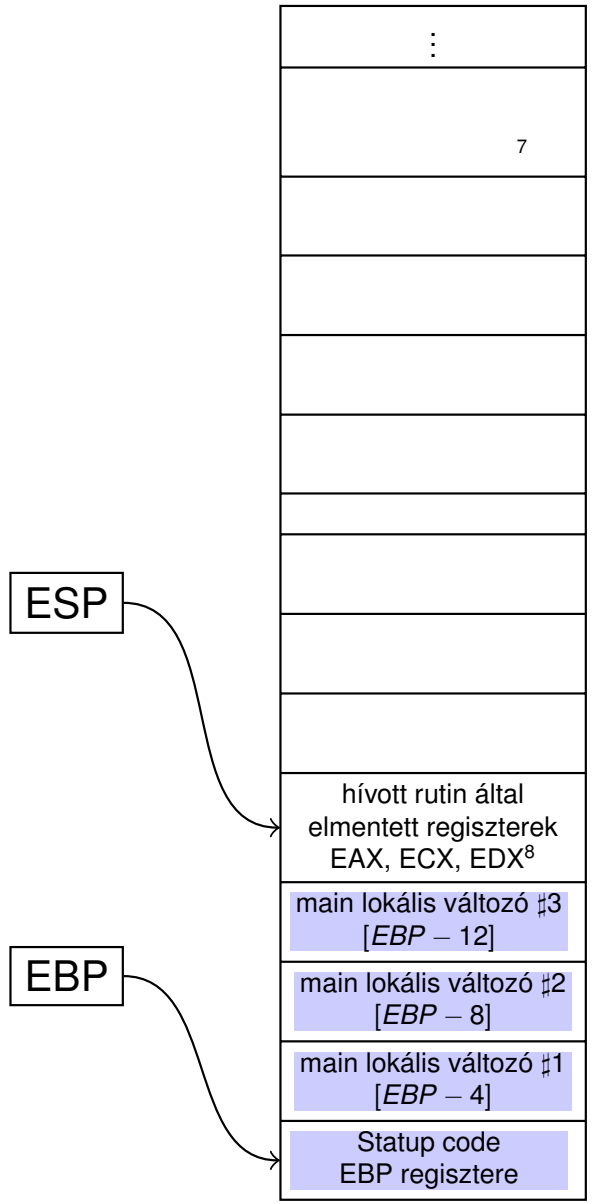
[7] csak ha szükséges



__cdecl hívási konvenció, visszatérés a hívóhoz

```
_main:
    push    ebp
    mov     ebp,esp
    sub     esp,0Ch
    mov     dword ptr [ebp-4],1
    mov     dword ptr [ebp-8],2
    mov     dword ptr [ebp-0Ch],3
    mov     eax,dword ptr [ebp-0Ch]
    push    eax
    mov     ecx,dword ptr [ebp-8]
    push    ecx
    mov     edx,dword ptr [ebp-4]
    push    edx
    call    _func
    add     esp,0Ch
    mov     dword ptr [ebp-4],eax
    xor     eax,eax
    mov     esp,ebp
    pop     ebp
    ret
```

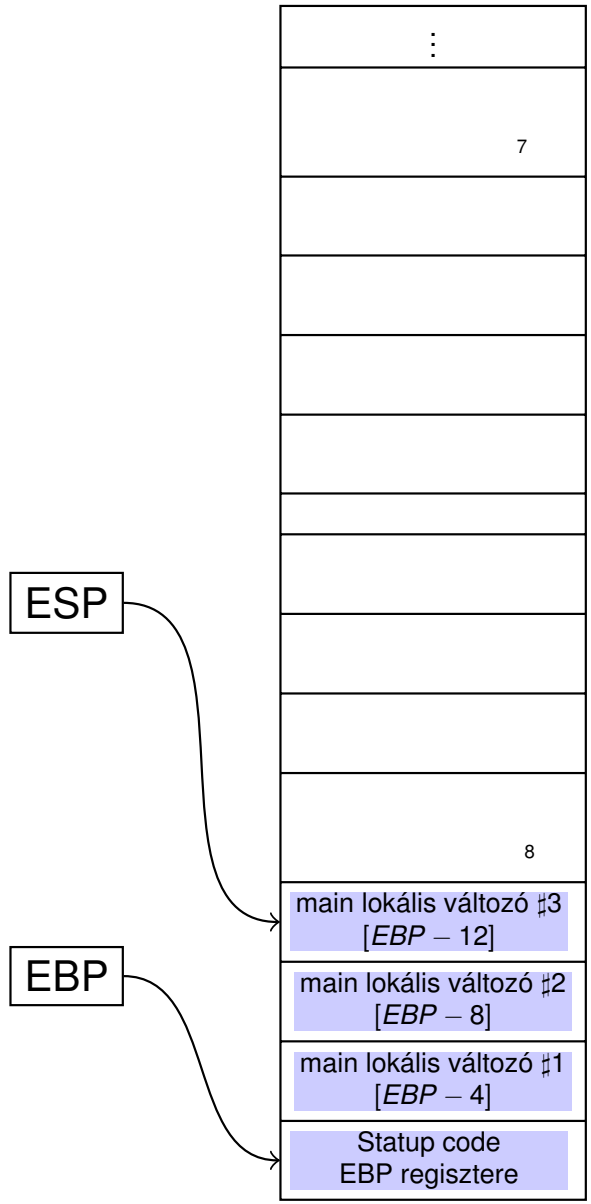
[7] csak ha szükséges



__cdecl hívási konvenció, visszatérés a hívóhoz

```
_main:
    push    ebp
    mov     ebp,esp
    sub     esp,0Ch
    mov     dword ptr [ebp-4],1
    mov     dword ptr [ebp-8],2
    mov     dword ptr [ebp-0Ch],3
    mov     eax,dword ptr [ebp-0Ch]
    push    eax
    mov     ecx,dword ptr [ebp-8]
    push    ecx
    mov     edx,dword ptr [ebp-4]
    push    edx
    call    _func
    add     esp,0Ch
    mov     dword ptr [ebp-4],eax
    xor     eax,eax
    mov     esp,ebp
    pop     ebp
    ret
```

[7] csak ha szükséges



__cdecl hívási konvenció IA-32

1. Hívó feladata a hívás előtt
   - ha szükséges, akkor elmenti EAX, ECX EDX regisztereket;
   - leteszi a hívott függvény paramétereit;
   - függvény meghívása (**call**).
2. Hívott feladata az eljárás elején
   - elmenti az EBP regisztert;
   - lokális adatoknak helyet foglal;
   - ha szükséges, elmenti az EBX, ESI és EDI regisztereket.
3. Hívott feladata a visszatérés előtt
   - az EAX regiszterbe beleteszi a visszatérési értéket;
   - Visszaállítja az elmentett EBX, ESI, EDI regisztereket;
   - ESP stack pointert az EBP által mutatott helyre állítja;
   - EBP-be beteszi a hívó stack keret bázisának a címét
   - visszatér a hívóhoz (**ret**).
4. Hívó feladata a hívás után
   - felszabadítja a függvény paramétereit;
   - EAX-ben lévő visszatérési érték elmentése;
   - EAX, ECX, EDX regiszterek visszaállítása.