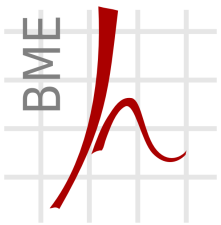
Stack kezelés és vezérlésszerkezetek visszafejtése

Kód visszafejtés.



Híradástechnikai Tanszék

Izsó Tamás

2016. november 24.

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. Intel X86 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 **call** (kódszegmens nem változik)

1. **push** <return address>
2. **jmp** func

Visszatérés a hívóhoz **ret** n:

1. **add esp**, n
2. **jmp dword ptr** [**esp**−n][1]

[1] n a letett paraméterek stacken elfoglalt mérete byte-ban.

Stack pointer munkában

Stack felszabadítása nagyobb egységekben

```
pop eax                ≡      add esp, 12
pop eax
pop eax
```

Stack lefoglalás nagyobb egységekben

```
sub esp,400
```

Stack megtekintés

```
mov eax, [esp+4]
```

C példaprogram

```
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;
}
```

A paraméterek átadásához helyre van szükség

C példaprogram

```
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;
}
```

Át kell venni a paramétereket.

C példaprogram

```
int func(int i, in
{
    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;
}
```

lokális adatoknak helyet kell biztosítani.

C példaprogram

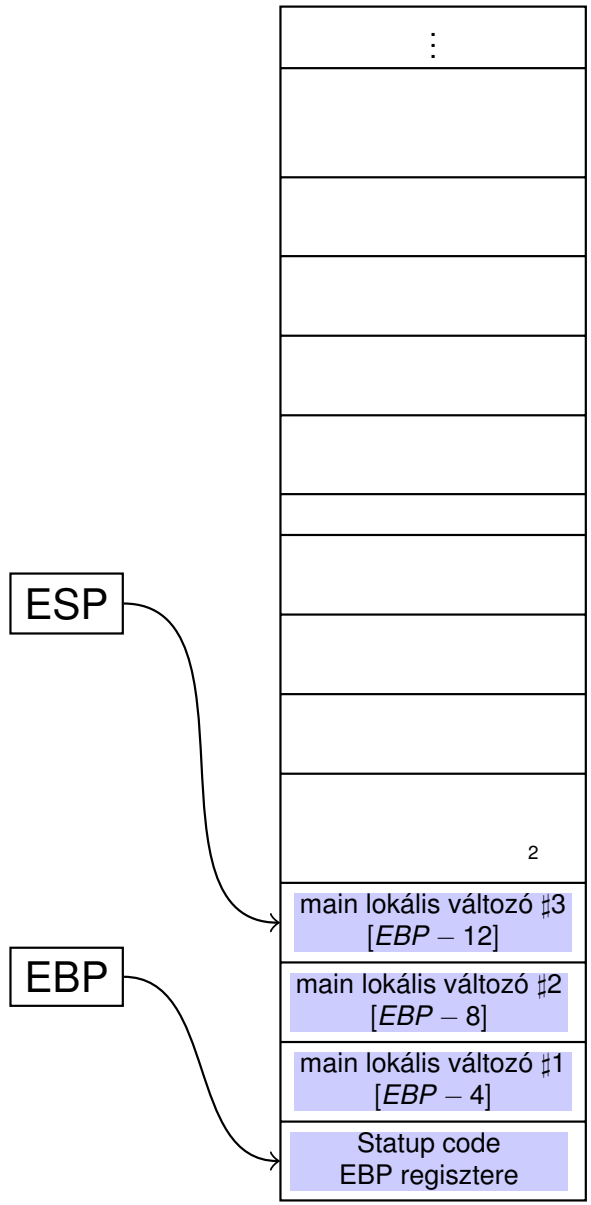
```
int func(int i, int j, int k )
{
    double r;
    int l=0;
    l = i;
    l +=j;
    l +=k;
    return l;
}

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

Vissza kell adni az értéket.

__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
```

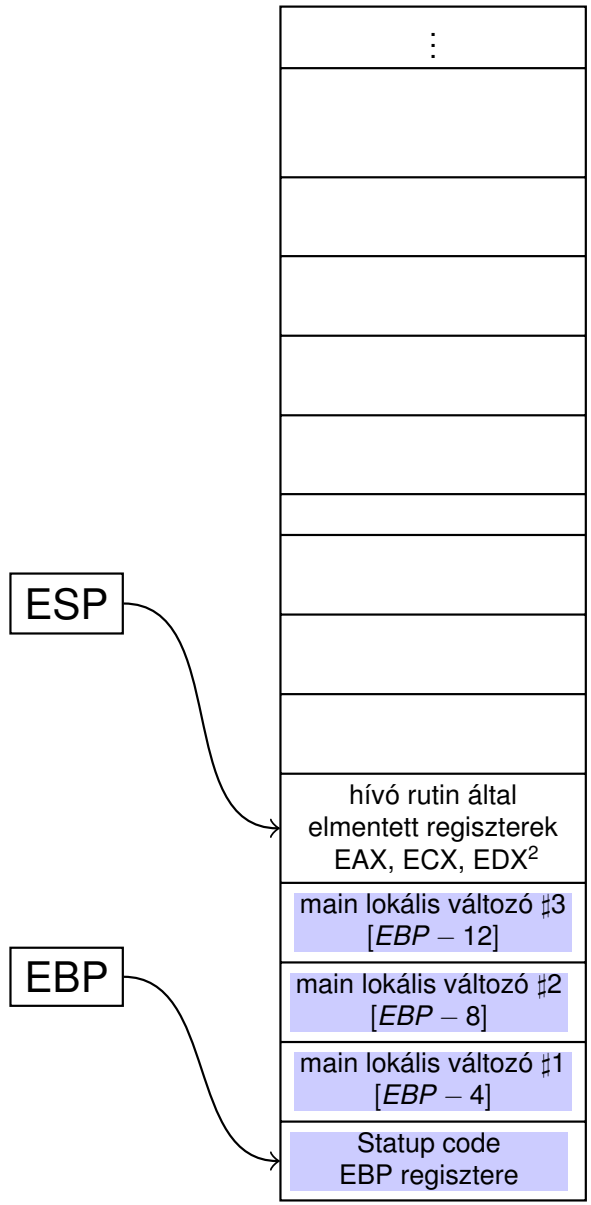
[2]csak ha szükséges



__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
```

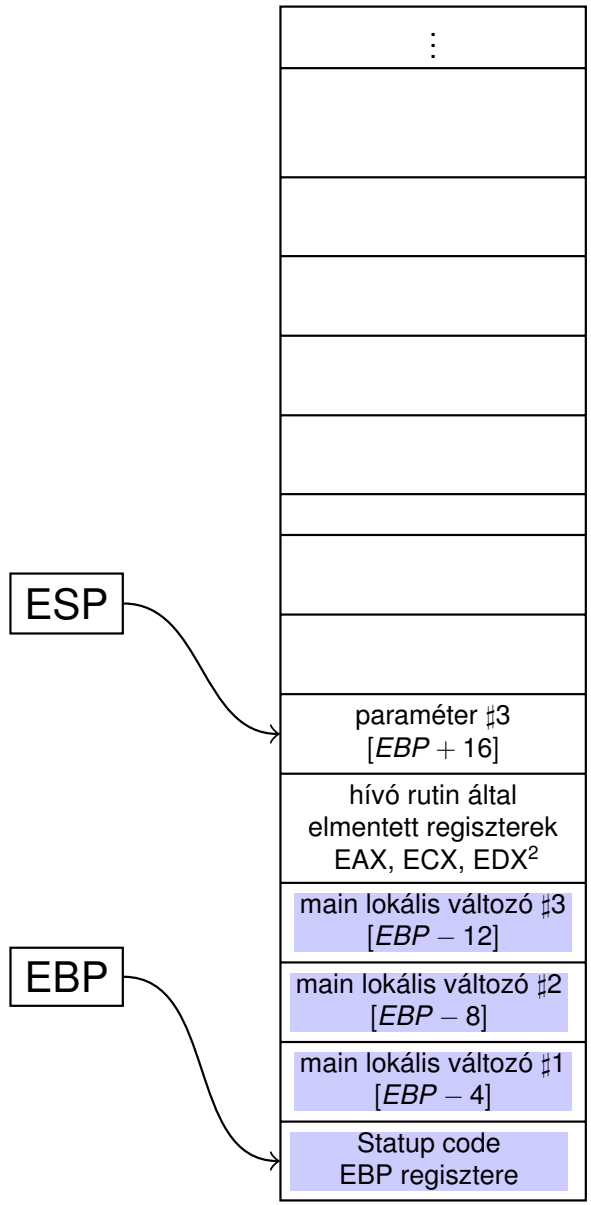
[2]csak ha szükséges



__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
```

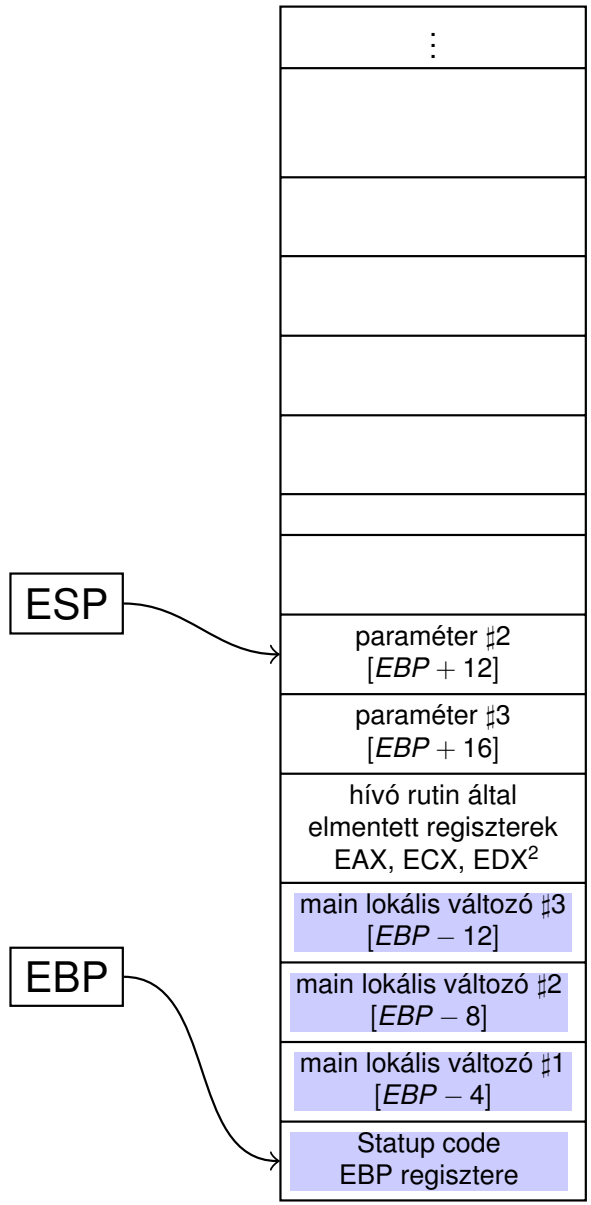
[2] csak ha szükséges



__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
```

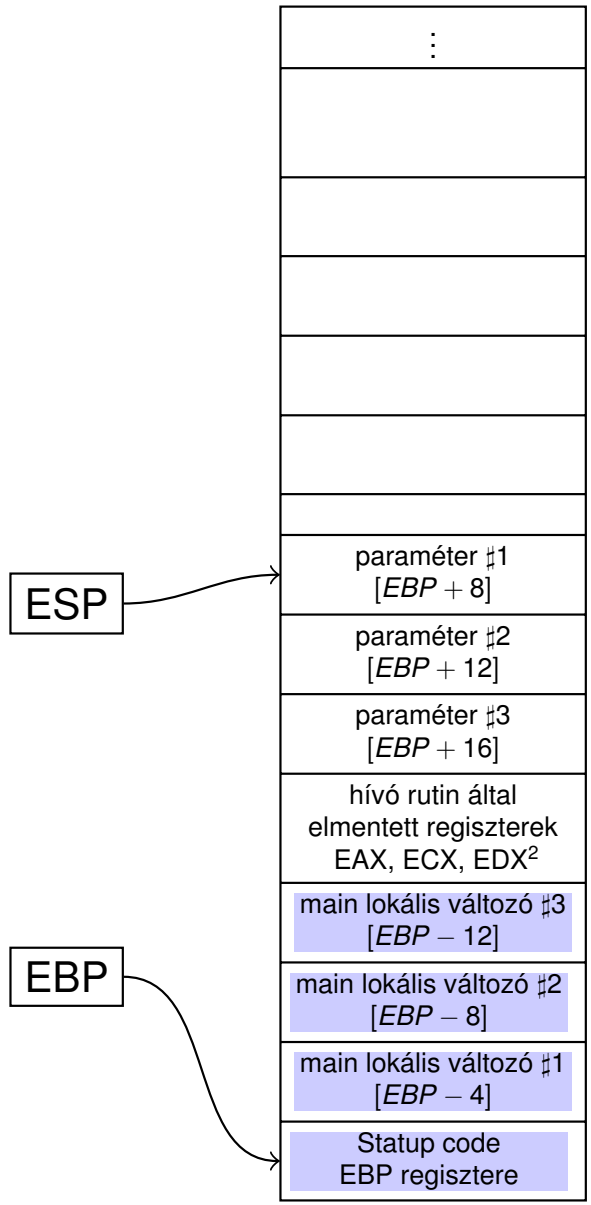
[2] csak ha szükséges



__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
```

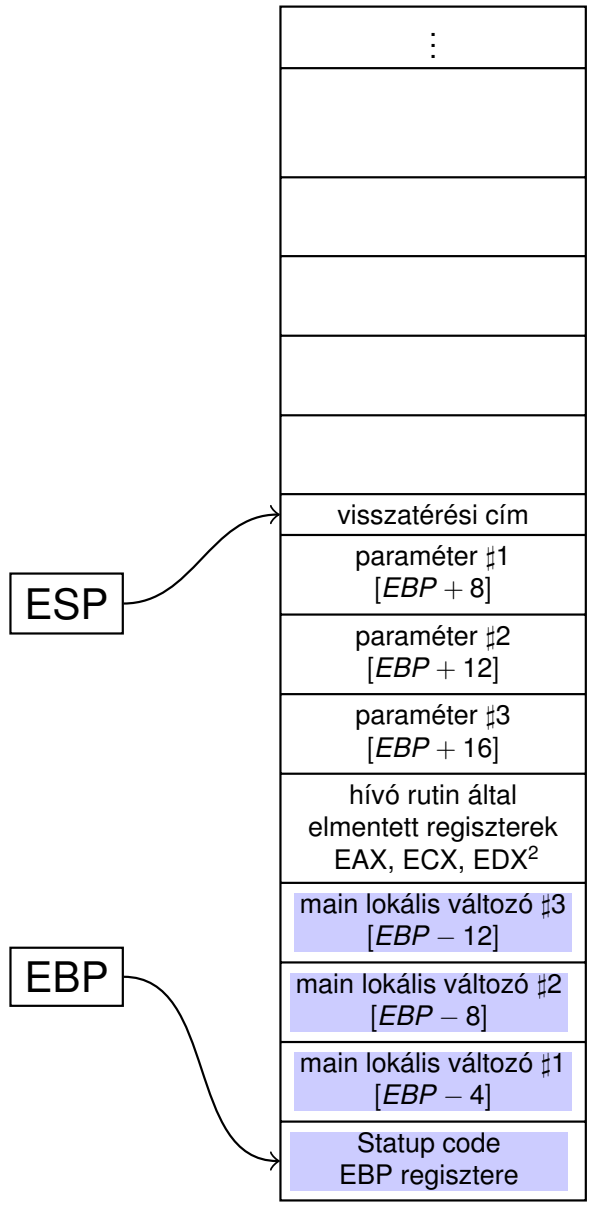
[2]csak ha szükséges



__cdecl hívási konvenció, hívó feladata

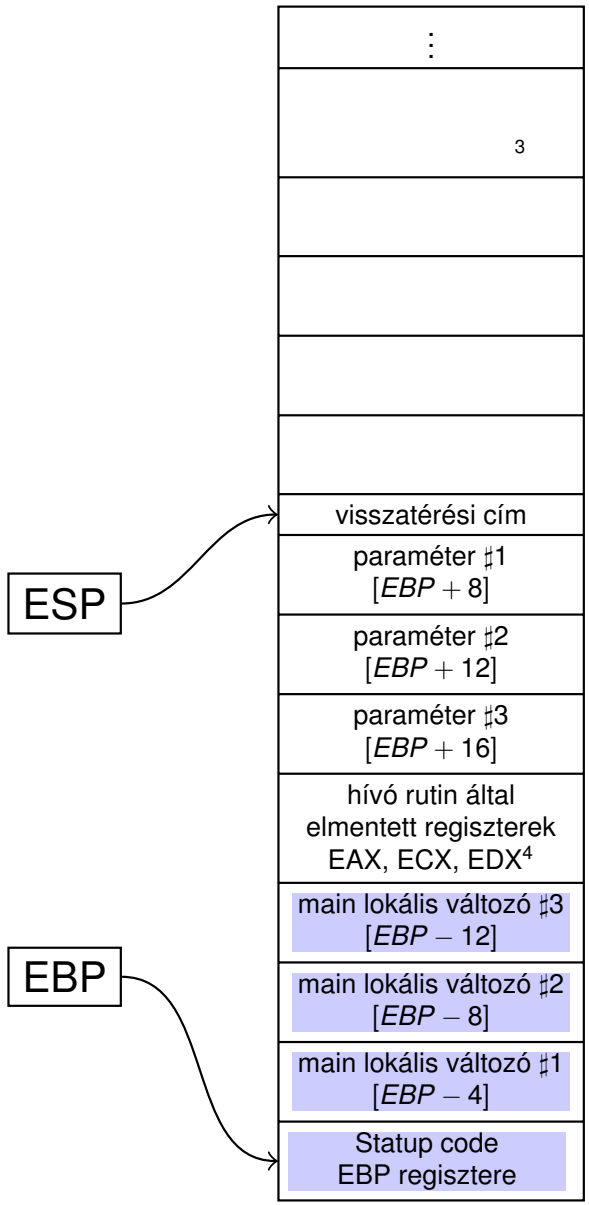
```
_main:
    push    ebp
    mov     ebp,esp
    sub     esp,0Ch
    mov     dword ptr [ebp-4],1
    mov     dword ptr [ebp-8],2
    mov     dword ptr [ebp-0Ch],3
    mov     eax,dword ptr [ebp-0Ch]
    push    eax
    mov     ecx,dword ptr [ebp-8]
    push    ecx
    mov     edx,dword ptr [ebp-4]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

[2] csak ha szükséges



__cdecl hívási konvenció, hívott feladata

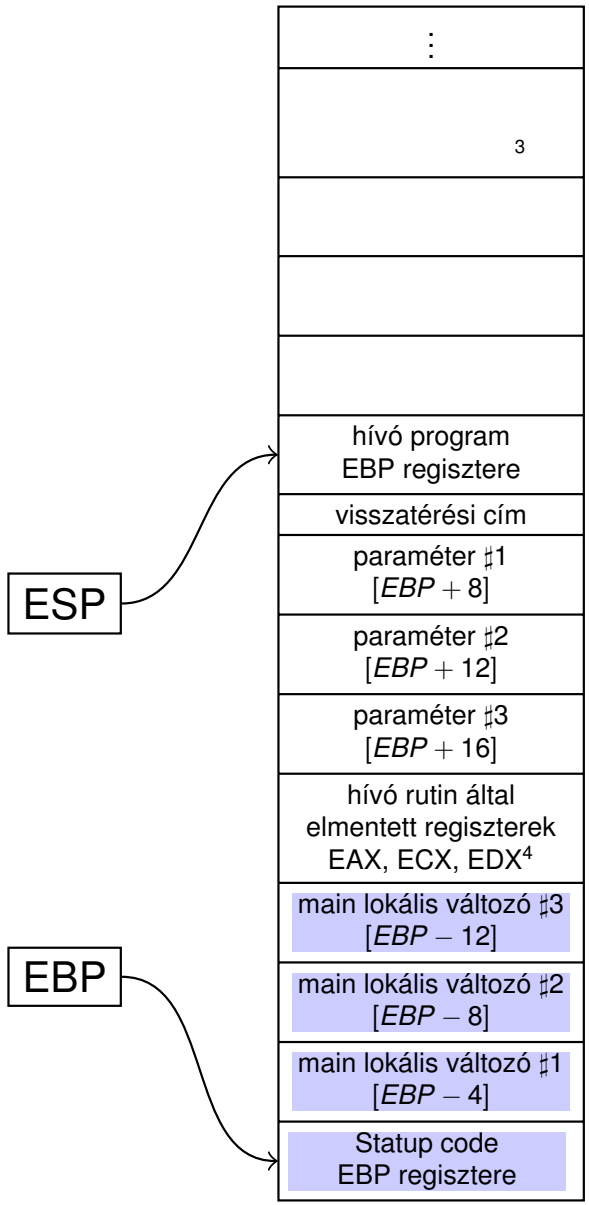
```
_func:
    push    ebp
    mov     ebp,esp
    sub     esp,10h
    mov     dword ptr [ebp-4],0
    mov     eax,dword ptr [ebp+8]
    mov     dword ptr [ebp-4],eax
    mov     ecx,dword ptr [ebp-4]
    add     ecx,dword ptr [ebp+0Ch]
    mov     dword ptr [ebp-4],ecx
    mov     edx,dword ptr [ebp-4]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



[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
```

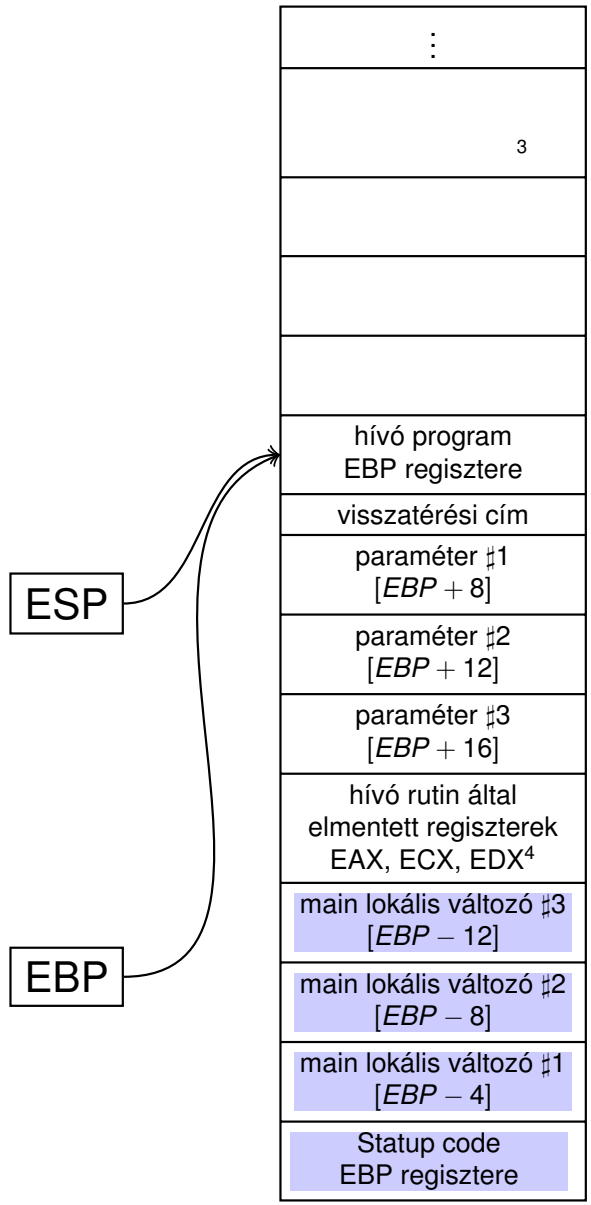


[3] csak ha szükséges

__cdecl hívási konvenció, hívott feladata

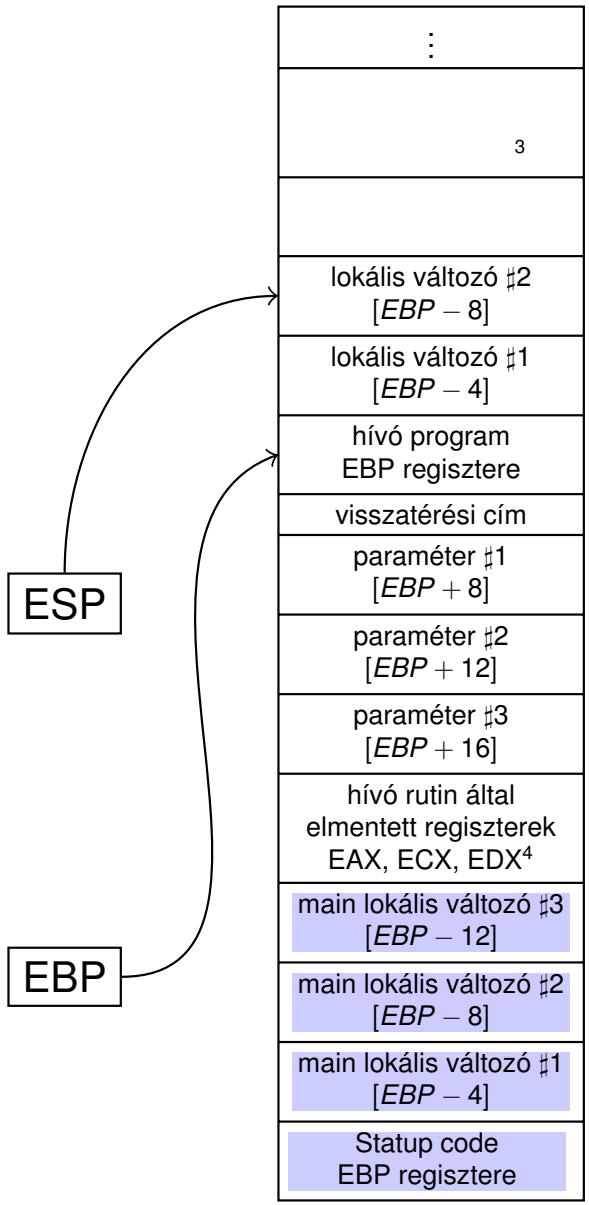
```
_func:
    push    ebp
    mov     ebp,esp
    sub     esp,10h
    mov     dword ptr [ebp-4],0
    mov     eax,dword ptr [ebp+8]
    mov     dword ptr [ebp-4],eax
    mov     ecx,dword ptr [ebp-4]
    add     ecx,dword ptr [ebp+0Ch]
    mov     dword ptr [ebp-4],ecx
    mov     edx,dword ptr [ebp-4]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

[3]csak ha szükséges



__cdecl hívási konvenció, hívott feladata

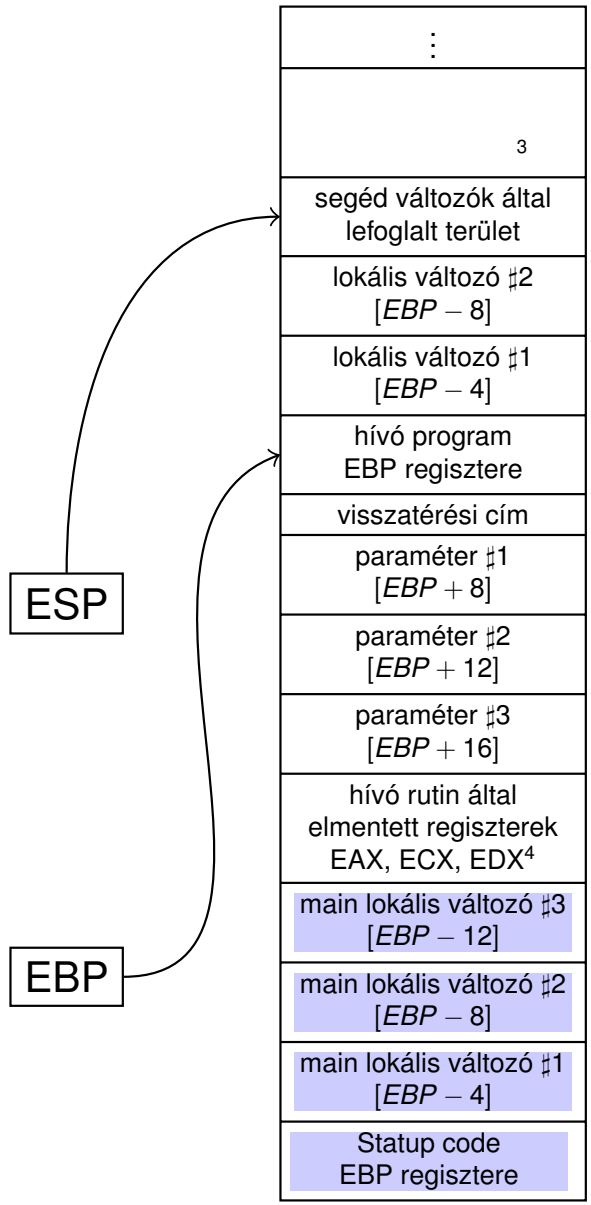
```
_func:
    push    ebp
    mov     ebp,esp
    sub     esp,10h
    mov     dword ptr [ebp-4],0
    mov     eax,dword ptr [ebp+8]
    mov     dword ptr [ebp-4],eax
    mov     ecx,dword ptr [ebp-4]
    add     ecx,dword ptr [ebp+0Ch]
    mov     dword ptr [ebp-4],ecx
    mov     edx,dword ptr [ebp-4]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



[3] csak ha szükséges

__cdecl hívási konvenció, hívott feladata

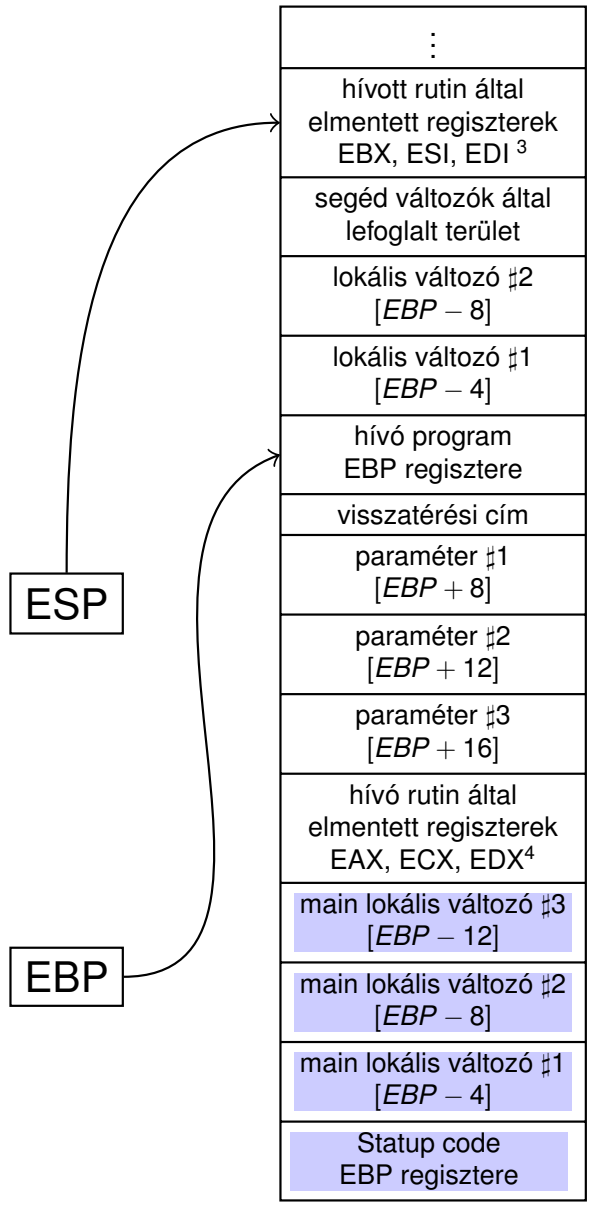
```
_func:
    push    ebp
    mov     ebp,esp
    sub     esp,10h
    mov     dword ptr [ebp-4],0
    mov     eax,dword ptr [ebp+8]
    mov     dword ptr [ebp-4],eax
    mov     ecx,dword ptr [ebp-4]
    add     ecx,dword ptr [ebp+0Ch]
    mov     dword ptr [ebp-4],ecx
    mov     edx,dword ptr [ebp-4]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



[3] csak ha szükséges

__cdecl hívási konvenció, hívott feladata

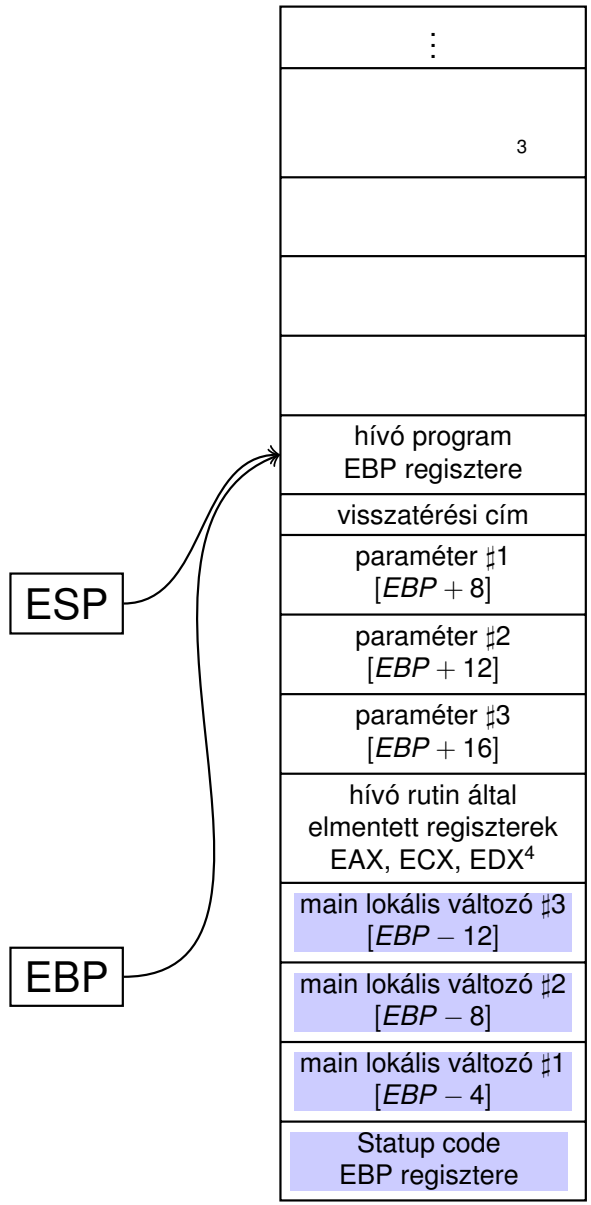
```
_func:
    push    ebp
    mov     ebp,esp
    sub     esp,10h
    mov     dword ptr [ebp-4],0
    mov     eax,dword ptr [ebp+8]
    mov     dword ptr [ebp-4],eax
    mov     ecx,dword ptr [ebp-4]
    add     ecx,dword ptr [ebp+0Ch]
    mov     dword ptr [ebp-4],ecx
    mov     edx,dword ptr [ebp-4]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



[3] csak ha szükséges

__cdecl hívási konvenció, hívott feladata

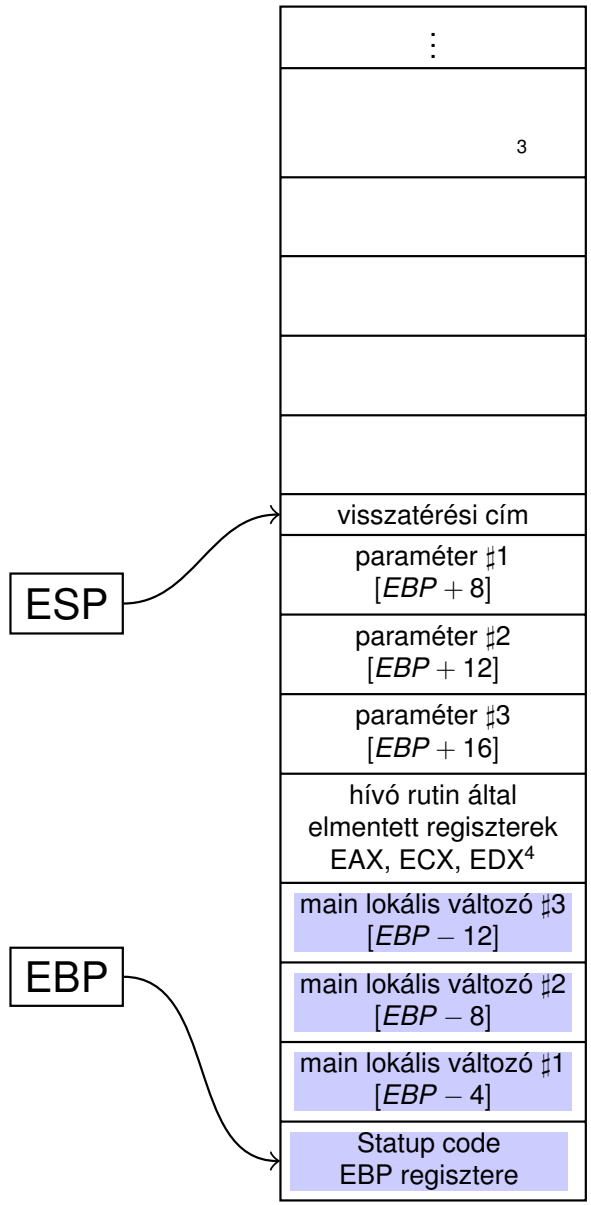
```
_func:
    push    ebp
    mov     ebp,esp
    sub     esp,10h
    mov     dword ptr [ebp-4],0
    mov     eax,dword ptr [ebp+8]
    mov     dword ptr [ebp-4],eax
    mov     ecx,dword ptr [ebp-4]
    add     ecx,dword ptr [ebp+0Ch]
    mov     dword ptr [ebp-4],ecx
    mov     edx,dword ptr [ebp-4]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



[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
```

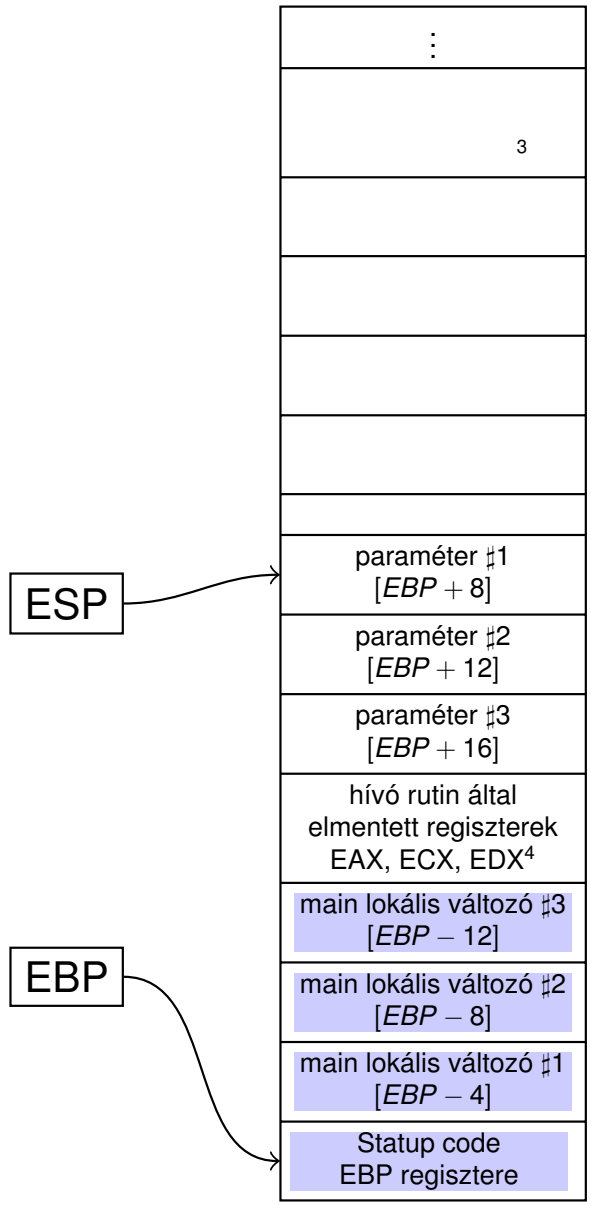


[3] csak ha szükséges

__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
```

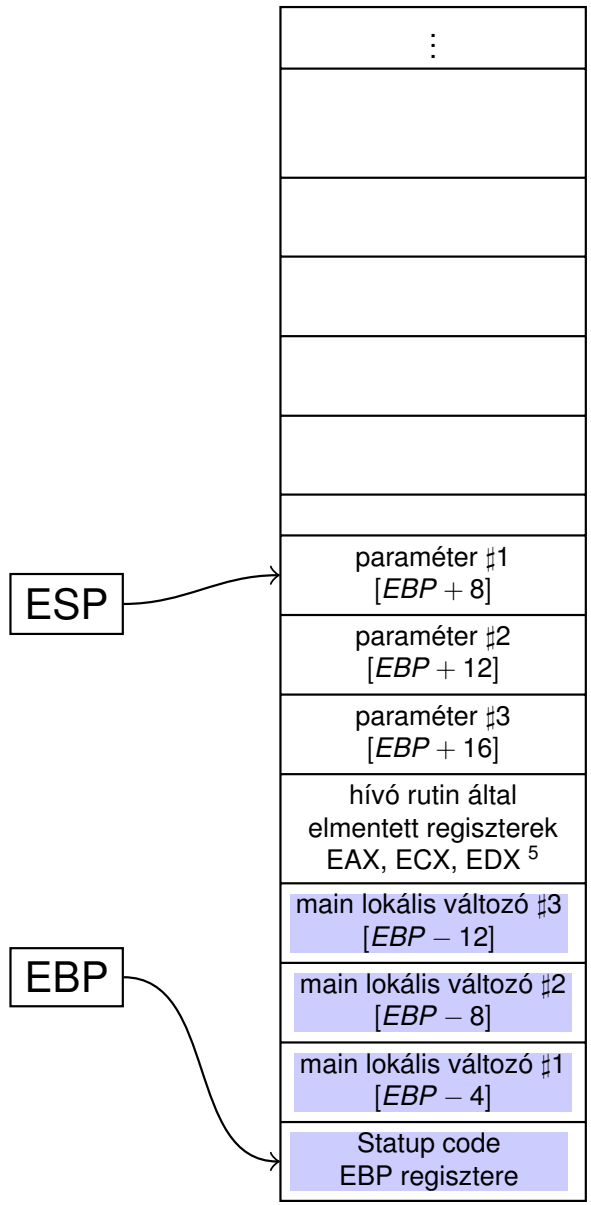
[3] csak ha szükséges



__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
```

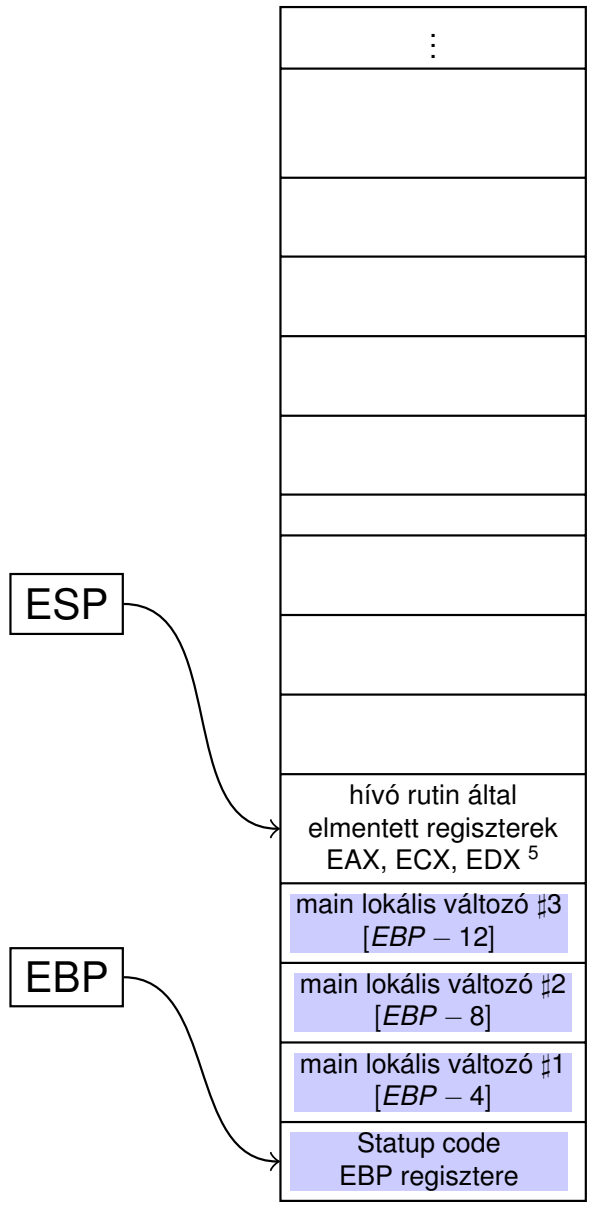
[5] csak ha szükséges



__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
```

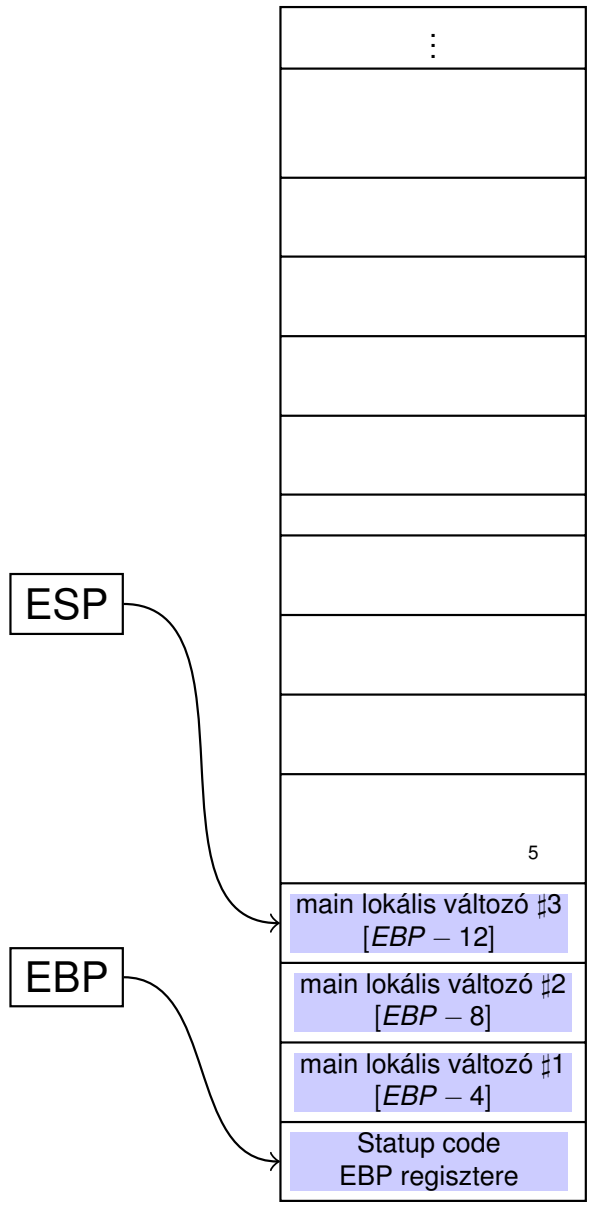
[5] csak ha szükséges



__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
```

[5]csak ha szükséges



__cdecl hívási konvenció Intex X86

1. Hívó feladata a hívás előtt
   - szükség esetén elmenteni az EAX, ECX EDX-t;
   - átadni a függvény paramétereit;
   - meghívni a függvényt (call).
2. Hívott feladata az eljárás elején
   - elmenteni az EBP regisztert;
   - a lokális adatoknak helyet foglalni;
   - szükség esesetén elmenteni az EBX, ESI és EDI-t.
3. Hívott feladata a visszatérés előtt
   - az EAX regiszterbe betenni a visszatérési értéket;
   - visszaállítani az elmentett EBX, ESI, EDI regisztereket;
   - ESP stack pointert az EBP által mutatott helyre állítani;
   - az elmentett stack báziscím visszaállítása az EBP-be;
   - visszatérni a hívóhoz (ret).
4. Hívó feladata a hívás után
   - felszabadítani a függvény paramétereit;
   - az EAX-ben lévő visszatérési érték elmentése;
   - az EAX, ECX, EDX regiszterek visszaállítása.

Egyéb hívási konvenciók

```
int __stdcall func(int i, int j, int k)
{
.......
}

int __fastcall func(int i, int j, int k)
{
.......
}
```

__stdcall hívási konvenció

```
_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@12
  mov dword ptr [ebp-4],eax
  xor eax,eax
  mov esp,ebp
  pop ebp
  ret
```

```
_func@12:
  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 0Ch
```

Paramétereket a hívott szabadítja fel. Gyorsabb a __cdecl-nél, de nem lehet változó paraméterszámú függvényt írni.

__fastcall hívási konvenció

```
_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 edx,dword ptr [ebp-8]
  mov ecx,dword ptr [ebp-4]
  call @func@12
  mov dword ptr [ebp-4],eax
  xor eax,eax
  mov esp,ebp
  pop ebp
  ret
```

```
@func@12:
  push ebp
  mov ebp,esp
  sub esp,18h
  mov dword ptr [ebp-18h],edx
  mov dword ptr [ebp-14h],ecx
  mov dword ptr [ebp-4],0
  mov eax,dword ptr [ebp-14h]
  mov dword ptr [ebp-4],eax
  mov ecx,dword ptr [ebp-4]
  add ecx,dword ptr [ebp-18h]
  mov dword ptr [ebp-4],ecx
  mov edx,dword ptr [ebp-4]
  add edx,dword ptr [ebp+8]
  mov dword ptr [ebp-4],edx
  mov eax,dword ptr [ebp-4]
  mov esp,ebp
  pop ebp
  ret 4
```

Hasonlít az __stdcall-ra, de pár paramétert regiszterben ad át.

Terminológia

- Függvény prológus (előszó) a függvény elején azok az utasítások, amelyek elmentik a használt regisztereket, és előkészítik a stack keretet.

```
push ebp
ebp = esp
esp = esp - <frame space>
```

- Függvény epilógus (utószó) a függvény végén azok az utasítások, amelyek törlik a stack keretet, és visszaállítják az elmentett regisztereket.

```
leave       // esp = ebp, pop ebp
ret         // pop eax, jmp eax
```

Változó típusú paraméterek átadása 1.

```
#include <stdio.h>
__int64 myfunc(short int, __int64, int, unsigned char);

void main()  {
        short int ff = 1024;
        __int64 ii = 1000;
        int jj = 200;
        unsigned char bb = 'H';
        __int64 ll = myfunc(ff, ii, jj, bb);
        printf("%lld\n", ll );
};

__int64 myfunc(short s, __int64 i, int j, unsigned char b) {
        __int64 ll;
        ll = s + i + j + b;
        return ll;
};
```

Paraméterek átadása 2a. (IDA output)

```
00000006 _main              proc near
00000006     var_20             = word ptr -20h
00000006     var_1C             = dword ptr -1Ch
00000006     var_18             = dword ptr -18h
00000006     var_14             = dword ptr -14h
00000006     var_10             = dword ptr -10h
00000006     var_C              = dword ptr -0Ch
00000006     var_1              = byte ptr -1
00000006 55                     push    ebp
00000007 8B EC                  mov     ebp, esp
00000009 83 EC 20               sub     esp, 20h
0000000C B8 00 04 00 00         mov     eax, 400h
00000011 66 89 45 E0            mov     [ebp+var_20], ax
00000015 C7 45 F0 E8 03 00 00   mov     [ebp+var_10], 3E8h
0000001C C7 45 F4 00 00 00 00   mov     [ebp+var_C], 0
00000023 C7 45 E4 C8 00 00 00   mov     [ebp+var_1C], 0C8h
0000002A C6 45 FF 48            mov     [ebp+var_1], 48h ; 'H'
```

Paraméterek átadása 2a. (IDA output)

```
00000006 _main                proc near
00000006     var_20           = word ptr -20h
00000006     var_1C           = dword ptr -1Ch
00000006     var_18           = dword ptr -18h
00000006     var_14           = dword ptr -14h
00000006     var_10           = dword ptr -10h
00000006     var_C            = dword ptr -0Ch
00000006     var_1            = byte ptr -1
00000006 55                   push    ebp
00000007 8B EC                mov     ebp, esp
00000009 83 EC 20             sub     esp, 20h
0000000C B8 00 04 00 00       mov     eax, 400h
00000011 66 89 45 E0          mov     [ebp+var_20], ax
00000015 C7 45 F0 E8 03 00 00 mov     [ebp+var_10], 3E8h
0000001C C7 45 F4 00 00 00 00 mov     [ebp+var_C], 0
00000023 C7 45 E4 C8 00 00 00 mov     [ebp+var_1C], 0C8h
0000002A C6 45 FF 48          mov     [ebp+var_1], 48h ; 'H'
```

lokális adatokhoz konstans definiciók

Paraméterek átadása 2a. (IDA output)

```
00000006 _main                  proc near
00000006      var_20                = word ptr -20h
00000006      var_1C                = dword ptr -1Ch
00000006      var_18                = dword ptr -18h
00000006      var_14                = dword ptr -14h
00000006      var_10                = dword ptr -10h
00000006      var_C                 = dword ptr -0Ch
00000006      var_1                 = byte ptr -1
00000006 55                         push    ebp
00000007 8B EC                      mov     ebp, esp
00000009 83 EC 20                   sub     esp, 20h
0000000C B8 00 04 00 00             mov     eax, 400h
00000011 66 89 45 E0                mov     [ebp+var_20], ax
00000015 C7 45 F0 E8                mov     [ebp+var_10], 3E8h
0000001C C7 45 F4 00                mov     [ebp+var_C], 0
00000023 C7 45 E4 C8 00 00 00       mov     [ebp+var_1C], 0C8h
0000002A C6 45 FF 48                mov     [ebp+var_1], 48h ; 'H'
```

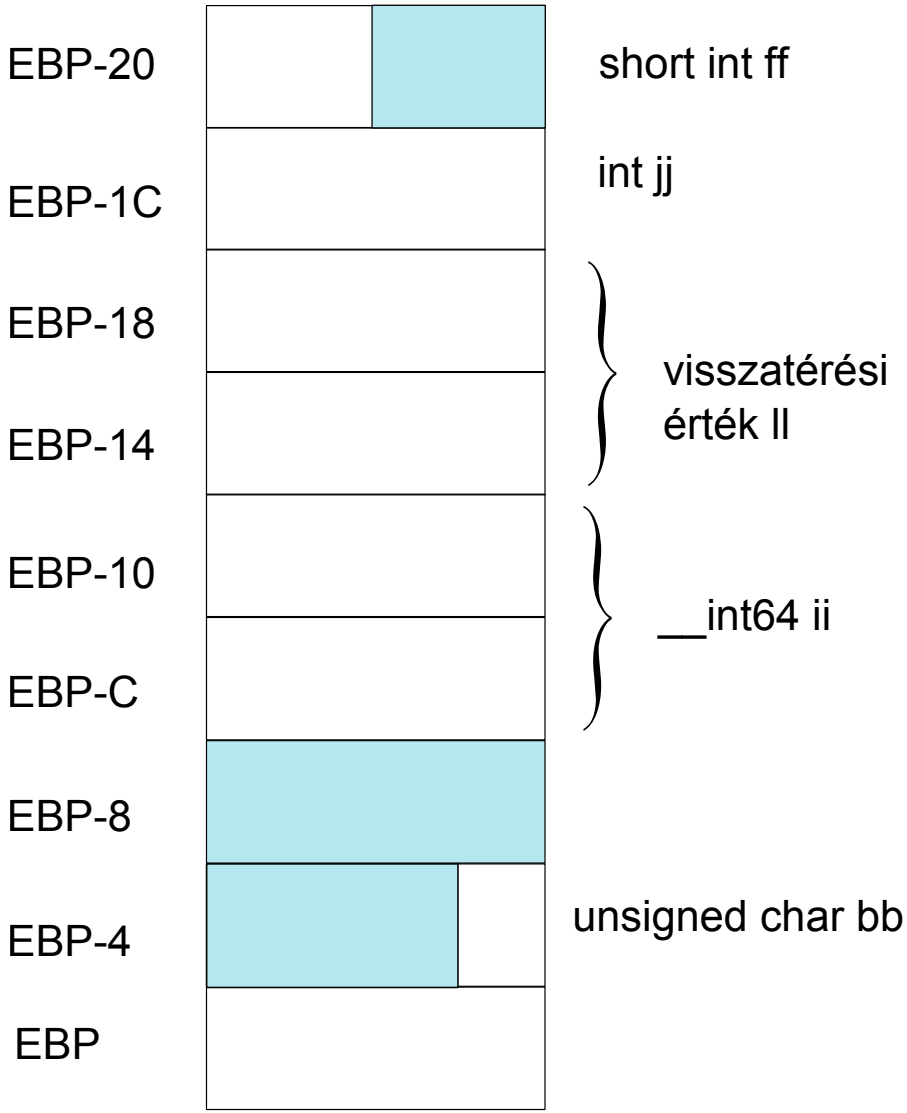
adat méretét módosító prefix

Paraméterek átadása 2a. (IDA output)

```
00000006 _main                  proc near
00000006      var_20                = word ptr -20h
00000006      var_1C                = dword ptr -1Ch
00000006      var_18                = dword ptr -18h
00000006      var_14                = dword ptr -14h
00000006      var_10                = dword ptr -10h
00000006      var_C                 = dword ptr -0Ch
00000006      var_1                 = byte ptr -1
00000006 55                           push    ebp
00000007 8B EC                        mov     ebp, esp
00000009 83 EC 20                     sub     esp, 20h
0000000C B8 00 04 00 00               mov     eax, 400h
00000011 66 89 45 E0                  mov     [ebp+var_20], ax
00000015 C7 45 F0 E8 03 00 00         mov     [ebp+var_10], 3E8h
0000001C C7 45 F4 00 00 00 00         mov     [ebp+var_C], 0
00000023 C7 45 E4 C8 00 00 00         mov     [ebp+var_1C], 0C8h
0000002A C6 45 FF 48                  mov     [ebp+var_1], 48h ; 'H'
```

lokális adatok inicializálása

Lokális adatok címhatárra igazítása



Változó típusú paraméterek átadása 2b.

```
0000002E 0F B6 4D FF          movzx   ecx, [ebp+var_1]
00000032 51                   push    ecx
00000033 8B 55 E4             mov     edx, [ebp+var_1C]
00000036 52                   push    edx
00000037 8B 45 F4             mov     eax, [ebp+var_C]
0000003A 50                   push    eax
0000003B 8B 4D F0             mov     ecx, [ebp+var_10]
0000003E 51                   push    ecx
0000003F 0F B7 55 E0          movzx   edx, [ebp+var_20]
00000043 52                   push    edx
00000044 E8 2D 00 00 00       call    _myfunc
00000049 83 C4 14             add     esp, 14h
0000004C 89 45 E8             mov     [ebp+var_18], eax
0000004F 89 55 EC             mov     [ebp+var_14], edx
00000052 8B 45 EC             mov     eax, [ebp+var_14]
00000055 50                   push    eax
00000056 8B 4D E8             mov     ecx, [ebp+var_18]
00000059 51                   push    ecx
0000005A 68 00 00 00 00       push    offset $SG2478
0000005F E8 4C 00 00 00       call    _printf
00000064 83 C4 0C             add     esp, 0Ch
00000067 33 C0                xor     eax, eax
00000069 8B E5                mov     esp, ebp
0000006B 5D                   pop     ebp
0000006C C3                   retn
0000006C                      _main   endp
```

Változó típusú paraméterek átadása 2b.

```
0000002E 0F B6 4D FF        movzx   ecx, [ebp+var_1]
00000032 51                 push    ecx
00000033 8B 55 E4           mov     edx, [ebp+var_1C]
00000036 52                 push    edx
00000037 8B 45 F4           mov     eax, [ebp+var_C]
0000003A 50                 push    eax
0000003B 8B 4D F0           mov     ecx, [ebp+var_10]
0000003E 51                 push    ecx
0000003F 0F B7 55 E0        movzx   edx, [ebp+var_20]
00000043 52                 push    edx
00000044 E8 2D 00 00 00     call    _myfunc
00000049 83 C4 14           add     esp, 14h
0000004C 89 45 E8           mov     [ebp+var_18], eax
0000004F 89 55 EC           mov     [ebp+var_14], edx
00000052 8B 45 EC           mov     eax, [ebp+var_14]
00000055 50                 push    eax
00000056 8B 4D E8           mov     ecx, [ebp+var_18]
00000059 51                 push    ecx
0000005A 68 00 00 00 00     push    offset $SG2478
0000005F E8 4C 00 00 00     call    _printf
00000064 83 C4 0C           add     esp, 0Ch
00000067 33 C0              xor     eax, eax
00000069 8B E5              mov     esp, ebp
0000006B 5D                 pop     ebp
0000006C C3                 retn
0000006C                    _main   endp
```

paraméterek átadása

Változó típusú paraméterek átadása 2b.

```
0000002E 0F B6 4D FF        movzx   ecx, [ebp+var_1]
00000032 51                 push    ecx
00000033 8B 55 E4           mov     edx, [ebp+var_1C]
00000036 52                 push    edx
00000037 8B 45 F4           mov     eax, [ebp+va
0000003A 50                 push    eax
0000003B 8B 4D F0           mov     ecx, [ebp+va
0000003E 51                 push    ecx
0000003F 0F B7 55 E0        movzx   edx, [ebp+var_20]
00000043 52                 push    edx
00000044 E8 2D 00 00 00     call    _myfunc
00000049 83 C4 14           add     esp, 14h
0000004C 89 45 E8           mov     [ebp+var_18], eax
0000004F 89 55 EC           mov     [ebp+var_14], edx
00000052 8B 45 EC           mov     eax, [ebp+var_14]
00000055 50                 push    eax
00000056 8B 4D E8           mov     ecx, [ebp+var_18]
00000059 51                 push    ecx
0000005A 68 00 00 00 00     push    offset $SG2478
0000005F E8 4C 00 00 00     call    _printf
00000064 83 C4 0C           add     esp, 0Ch
00000067 33 C0              xor     eax, eax
00000069 8B E5              mov     esp, ebp
0000006B 5D                 pop     ebp
0000006C C3                 retn
0000006C                    _main   endp
```

1 byte-os paraméter a stacken 4 byte-ot foglal. Felső byte-okat a movzx nullával tölti fel.

Változó típusú paraméterek átadása 2b.

```
0000002E 0F B6 4D FF        movzx   ecx, [ebp+var_1]
00000032 51                 push    ecx
00000033 8B 55 E4           mov     edx, [ebp+var_1C]
00000036 52                 push    edx
00000037 8B 45 F4           mov     eax, [ebp+var_C]
0000003A 50                 push    eax
0000003B 8B 4D F0           mov     ecx, [ebp+var_10]
0000003E 51                 push    ecx
0000003F 0F B7 55 E0        movzx   edx, [ebp+var_20]
00000043 52                 push    edx
00000044 E8 2D 00 00 00     call    _myfunc
00000049 83 C4 14           add     esp, 14h
0000004C 89 45 E8           mov     [ebp+var_18], eax
0000004F 89 55 EC           mov     [ebp+var_14], edx
00000052 8B 45 EC           mov     eax, [ebp+var_14]
00000055 50                 push    eax
00000056 8B 4D E8           mov     ecx, [ebp+v
00000059 51                 push    ecx
0000005A 68 00 00 00 00     push    offset $SG2478
0000005F E8 4C 00 00 00     call    _printf
00000064 83 C4 0C           add     esp, 0Ch
00000067 33 C0              xor     eax, eax
00000069 8B E5              mov     esp, ebp
0000006B 5D                 pop     ebp
0000006C C3                 retn
0000006C                    _main   endp
```

Visszatérési érték elmentése.

Változó típusú paraméterek átadása 2b.

```
0000002E 0F B6 4D FF          movzx   ecx, [ebp+var_1]
00000032 51                   push    ecx
00000033 8B 55 E4             mov     edx, [ebp+var_1C]
00000036 52                   push    edx
00000037 8B 45 F4             mov     eax, [ebp+var_C]
0000003A 50                   push    eax
0000003B 8B 4D F0             mov     ecx, [ebp+var_10]
0000003E 51                   push    ecx
0000003F 0F B7 55 E0          movzx   edx, [ebp+var_20]
00000043 52                   push    edx
00000044 E8 2D 00 00 00       call    _myfunc
00000049 83 C4 14             add     esp, 14h
0000004C 89 45 E8             mov     [ebp+var_18], eax
0000004F 89 55 EC             mov     [ebp+var_14], edx
00000052 8B 45 EC             mov     eax, [ebp+var_14]
00000055 50                   push    eax
00000056 8B 4D E8             mov     ecx, [ebp+var_18]
00000059 51                   push    ecx
0000005A 68 00 00 00 00       push    offset $SG2478
0000005F E8 4C 00 00 00       call    _printf
00000064 83 C4 0C             add     esp, 0Ch
00000067 33 C0                xor     eax, eax
00000069 8B E5                mov     esp, ebp
0000006B 5D                   pop     ebp
0000006C C3                   retn
0000006C                      _main   endp
```

Eredmény kiíratása.

Változó típusú paraméterek átadása 2b.

```
0000002E 0F B6 4D FF          movzx   ecx, [ebp+var_1]
00000032 51                   push    ecx
00000033 8B 55 E4             mov     edx, [ebp+var_1C]
00000036 52                   push    edx
00000037 8B 45 F4             mov     eax, [ebp+var_C]
0000003A 50                   push    eax
0000003B 8B 4D F0             mov     ecx, [ebp+var_10]
0000003E 51                   push    ecx
0000003F 0F B7 55 E0          movzx   edx, [ebp+var_20]
00000043 52                   push    edx
00000044 E8 2D 00 00 00       call    _myfunc
00000049 83 C4 14             add     esp, 14h
0000004C 89 45 E8             mov     [ebp+var_18], eax
0000004F 89 55 EC             mov     [ebp+var_14], edx
00000052 8B 45 EC             mov     eax, [ebp+var_14]
00000055 50                   push    eax
00000056 8B 4D E8             mov     ecx, [ebp+var_18]
00000059 51                   push    ecx
0000005A 68 00 00 00 00       push    offset $SG2478
0000005F E8 4C 00 00 00       call    _printf
00000064 83 C4 0C             add     esp, 0Ch
00000067 33 C0                xor     eax, eax
00000069 8B E5                mov     esp, ebp
0000006B 5D                   pop     ebp
0000006C C3                   retn
0000006C                      _main   endp
```

Egy vagy két paraméter?

Változó számú paraméterátadás C nyelven

```
#include <stdio.h>

void kiir(const char* stype, ... ) {
    char* pargs = (char*)&stype;
    pargs += sizeof( stype );
    for( ; *stype; stype++ ) {
        switch( *stype )    {
            case 'c' :
                printf("%c\n", *( (pargs+=4)-4) );
                break;
            case 'i' :
                printf("%d\n", *(int*)( (pargs+=4)-4) );
                break;
            case 'f' :
                printf("%f\n", *(double*)( (pargs+=8)-8) );
                break;
        }
    }
}

int main() {
    float f=3.14;
    double lf=2.71;
    kiir("cifcf",'A', 125, f, 'X', lf );
    return 0;
}
```

Karakter esetén is 4 byte kerül a stackre!

Nekünk kell gondoskodni, hogy float paraméterátadásnál is double kerül a stackre!

Változó számú paraméterátadás C nyelven

```
#include <stdio.h>
#include <stdarg.h>

void kiir(const char* stype, ... ) {
    va_list ap;
    va_start(ap, stype );
    for( ; *stype; stype++ ) {
        switch( *stype )        {
            case 'c' :
                printf("%c\n", va_arg(ap,char) );
                break;
            case 'i' :
                printf("%d\n", va_arg(ap,int) );
                break;
            case 'f' :
                printf("%f\n", va_arg(ap,double) );
                break;
        }
    }
}

int main() {
    float f=3.14;
    double lf=2.71;
    kiir("cifcf",'A', 125, f, 'X', lf );
    return 0;
}
```

Függvény obfuscation

```
void funcret(void) {
  int x=0xbabfa;

 __asm push ecx   ;save registers
 __asm push ebx   ; return addr = edx
 __asm push edx   ; return addr = edx
 __asm mov  ebx,esp ;save ebp, esp
 __asm mov  esp,ebp
 __asm pop  ebp ; save old %ebp
 __asm pop  ecx ; save return addr
 __asm lea  edx,lab0 ; edx = addr of lab0:
 __asm push edx   ; return addr = edx
 __asm ret
 __asm _emit 0x0F ; off-by-one byte
 lab0:  ;
 __asm push ecx ; restore ret addr
 __asm push ebp ; restore old ebp
 __asm mov  ebp,esp ; restore ebp, esp
 __asm mov  esp,ebx
 __asm pop  ebx
 __asm pop  ecx

/* other stuff can be done here that won't be disassembled */
  printf("x=%d\n",x);
}
```

Függvény obfuscation IDA

```
; void __cdecl funcret()
public ?funcret@@YAXXZ
?funcret@@YAXXZ proc near
var_4 = dword ptr -4
  push   ebp
  mov    ebp, esp
  push   ecx
  push   ebx
  mov    [ebp+var_4],0BABFAh
  push   ecx
  push   ebx
  push   edx
  mov    ebx, esp
  mov    esp, ebp
  pop    ebp
  pop    ecx
  lea    edx,$lab0$3851
  push   edx
  ret    n
```

```
?funcret@@YAXXZ end
    db 0Fh
;---------------
 $lab0$3851:
    push   ecx
    push   ebp
    mov    ebp, esp
    mov    esp, ebx
    pop    ebx
    pop    ecx
    mov    eax, [ebp-4]
    push   eax
    push   offset $SG3852; "x=%d\n"
    call   _printf
    add    esp, 8
    pop    ebx
    mov    esp, ebp
    pop    ebp
    ret    n
```

Section 2

if

Maximum kiválasztás C nyelven

```c
#include <stdio.h>

int main()
{
    int a, b, c, max;
    scanf("%d %d %d", &a, &b, &c);
    if( a > b )
        if( a > c ) max = a;
        else max = c;
    else
        if( b > c ) max = b;
        else max = c;

    printf("%d", max );

    return 0;
}
```

Maximum kiválasztás nem optimalizálva (beolvasás)

```
_main:
   00000000  55                    push    ebp
   00000001  8B EC                 mov     ebp,esp
   00000003  83 EC 10              sub     esp,10h
   00000006  8D 45 F4              lea     eax,[ebp-0Ch]
   00000009  50                    push    eax
   0000000A  8D 4D F8              lea     ecx,[ebp-8]
   0000000D  51                    push    ecx
   0000000E  8D 55 FC              lea     edx,[ebp-4]
   00000011  52                    push    edx
   00000012  68 00 00 00 00        push    offset $SG2472
   00000017  E8 00 00 00 00        call    _scanf
   0000001C  83 C4 10              add     esp,10h
```

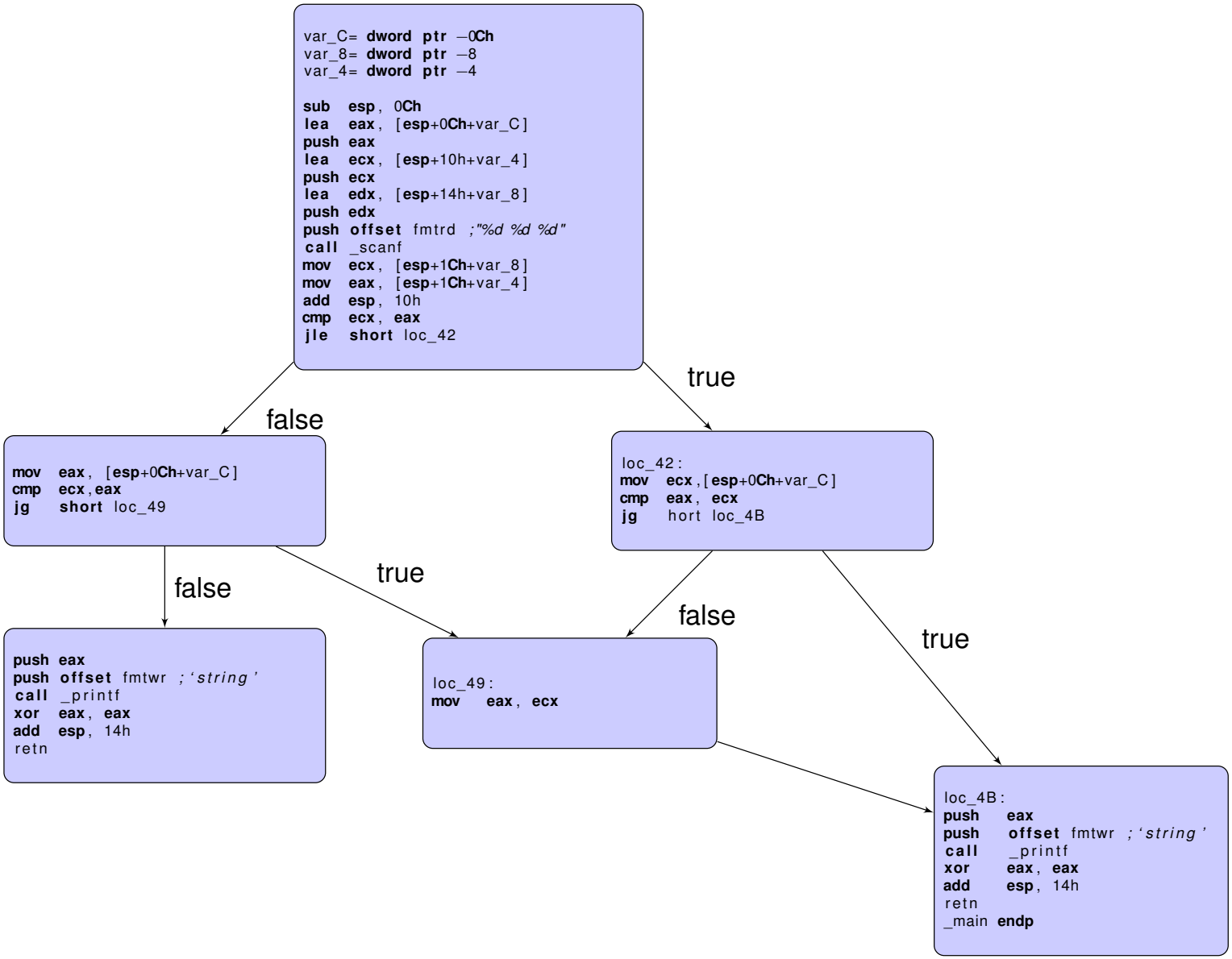
Maximum kiválasztás nem optimalizálva (döntés)

```
0000001F 8B 45 FC       mov   eax,dword ptr [ebp-4]
00000022 3B 45 F8       cmp   eax,dword ptr [ebp-8]
00000025 7E 18          jle   0000003F
00000027 8B 4D FC       mov   ecx,dword ptr [ebp-4]
0000002A 3B 4D F4       cmp   ecx,dword ptr [ebp-0Ch]
0000002D 7E 08          jle   00000037
0000002F 8B 55 FC       mov   edx,dword ptr [ebp-4]
00000032 89 55 F0       mov   dword ptr [ebp-10h],edx
00000035 EB 06          jmp   0000003D
00000037 8B 45 F4       mov   eax,dword ptr [ebp-0Ch]
0000003A 89 45 F0       mov   dword ptr [ebp-10h],eax
0000003D EB 16          jmp   00000055
0000003F 8B 4D F8       mov   ecx,dword ptr [ebp-8]
00000042 3B 4D F4       cmp   ecx,dword ptr [ebp-0Ch]
00000045 7E 08          jle   0000004F
00000047 8B 55 F8       mov   edx,dword ptr [ebp-8]
0000004A 89 55 F0       mov   dword ptr [ebp-10h],edx
0000004D EB 06          jmp   00000055
0000004F 8B 45 F4       mov   eax,dword ptr [ebp-0Ch]
00000052 89 45 F0       mov   dword ptr [ebp-10h],eax
```

Maximum kiválasztás nem optimalizálva (kiírás)

```
00000055  8B 4D F0          mov     ecx,dword ptr [ebp-10h]
00000058  51                push    ecx
00000059  68 00 00 00 00    push    offset $SG2479
0000005E  E8 00 00 00 00    call    _printf
00000063  83 C4 08          add     esp,8
00000066  33 C0             xor     eax,eax
00000068  8B E5             mov     esp,ebp
0000006A  5D                pop     ebp
0000006B  C3                ret
```

Maximum kiválasztás nem optimalizálva (BB)



Maximum kiválasztás optimalizálva (beolvasás)

```
_main:
   00000000  83 EC 0C              sub     esp,0Ch
   00000003  8D 04 24              lea     eax,[esp]
   00000006  50                    push    eax
   00000007  8D 4C 24 0C           lea     ecx,[esp+0Ch]
   0000000B  51                    push    ecx
   0000000C  8D 54 24 0C           lea     edx,[esp+0Ch]
   00000010  52                    push    edx
   00000011  68 00 00 00 00        push    offset $SG2472
   00000016  E8 00 00 00 00        call    _scanf
   0000001B  8B 4C 24 14           mov     ecx,dword ptr [esp+14h]
   0000001F  8B 44 24 18           mov     eax,dword ptr [esp+18h]
   00000023  83 C4 10              add     esp,10h
```

Figyelem!

Vegyük észre, hogy a kód nem haszálja az EBP frame pointert. Magyarázzuk meg a lea ecx,[esp+0Ch] utasításokat.

Maximum kiválasztás optimalizálva (if-ek)

```
00000026 3B C8              cmp     ecx,eax
00000028 7E 18              jle     00000042
0000002A 8B 04 24           mov     eax,dword ptr [esp]
0000002D 3B C8              cmp     ecx,eax
0000002F 7F 18              jg      00000049
00000031 50                 push    eax
00000032 68 00 00 00 00     push    offset $SG2479
00000037 E8 00 00 00 00     call    _printf
0000003C 33 C0              xor     eax,eax
0000003E 83 C4 14           add     esp,14h
00000041 C3                 ret
00000042 8B 0C 24           mov     ecx,dword ptr [esp]
00000045 3B C1              cmp     eax,ecx
00000047 7F 02              jg      0000004B
00000049 8B C1              mov     eax,ecx
0000004B 50                 push    eax
0000004C 68 00 00 00 00     push    offset $SG2479
00000051 E8 00 00 00 00     call    _printf
00000056 33 C0              xor     eax,eax
00000058 83 C4 14           add     esp,14h
0000005B C3                 ret
```

Maximum kiválasztás összetett logikai kifejezéssel

```
#include <stdio.h>

int main()
{
    int a, b, c, max;
    scanf("%d %d %d", &a, &b, &c);
    if( a >= b && a >= c ) max = a;
    else if( b >= a && b >= c ) max = b;
    else max = c;

    printf("%d", max );
    return 0;
}
```

Maximum kiválasztás össztett logikai kifejezéssel optimalizálva

```
_main:
                                        ...
                                        ...
0000001C 8B 44 24 10        mov         eax,dword ptr [esp+10h]
00000020 8B 4C 24 14        mov         ecx,dword ptr [esp+14h]
00000024 8B 54 24 18        mov         edx,dword ptr [esp+18h]
00000028 83 C4 10           add         esp,10h
0000002B 3B C1              cmp         eax,ecx
0000002D 7E 04              jl          00000033
0000002F 3B C2              cmp         eax,edx
00000031 7F 0C              jge         0000003F
00000033 3B C8              cmp         ecx,eax
00000035 7E 06              jl          0000003D
00000037 3B CA              cmp         ecx,edx
00000039 8B C1              mov         eax,ecx
0000003B 7F 02              jge          0000003F
0000003D 8B C2              mov         eax,edx
                                        ...
                                        ...
```

If feltétel ugrás nélkül

A jump lassítja a futást, ezért a Microsoft C, ha lehet kerüli az ugró utasítás használatát.

```
#include <stdio.h>

int main()
{
    int a,  c;
    scanf("%d", &a);
    if( a > 0  ) c = 1000;
    else c = 10;

    printf("%d", c );
    return 0;
}
```

If feltétel ugrás nélkül

```
_main:
   00000000 51                    push    ecx
   00000001 8D 04 24              lea     eax,[esp]
   00000004 50                    push    eax
   00000005 68 00 00 00 00        push    offset $SG2478
   0000000A E8 00 00 00 00        call    _scanf
   0000000F 33 C0                 xor     eax,eax
   00000011 39 44 24 08           cmp     dword ptr [esp+8],eax
   00000015 0F 9E C0              setle   al  ;Set byte if less or equal (al = 0 or 1)
   00000018 48                    dec     eax ;eax ~0 or 0
   00000019 25 DE 03 00 00        and     eax,3DEh ; eax 990 or 0
   0000001E 83 C0 0A              add     eax,0Ah ; eax 1000 or 10
   00000021 50                    push    eax
   00000022 68 00 00 00 00        push    offset $SG2479
   00000027 E8 00 00 00 00        call    _printf
   0000002C 33 C0                 xor     eax,eax
   0000002E 83 C4 14              add     esp,14h
   00000031 C3                    ret
```

Vegyük észre, hogy a letett paramétereket a program a scanf függvényhívás után nem rögtön szabadítja fel, hanem csak a főprogram végén add esp,14h.

Section 3

switch

Switch utasítás

```
enum state { one=2, two, three, four, five } stat;

int foo(enum state  stat)
{
   int  a=0;
   switch( stat )
   {
   case one :
      a=10;
      break;
   case two :
      a=3;
      break;
   case three :
      a=8;
      break;
   case four :
      a=2;
      break;
   case five :
      a=4;
      break;
   }
   return a;
}
```

switch – Microsoft C – nem optimalizált

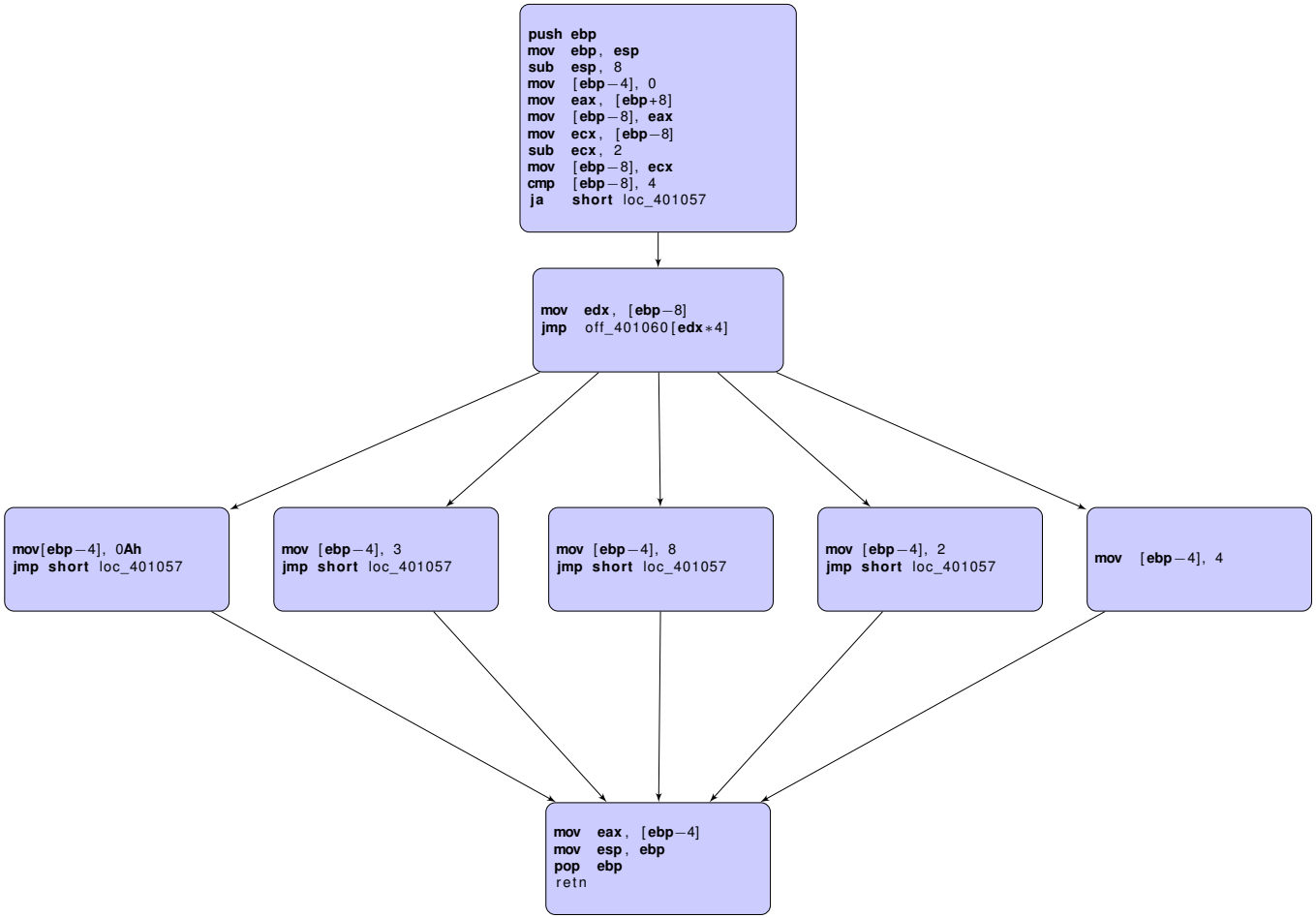
```
00401000 55                   push    ebp
00401001 8B EC                mov     ebp, esp
00401003 83 EC 08             sub     esp, 8
00401006 C7 45 FC 00000000    mov     [ebp-4], 0
0040100D 8B 45 08             mov     eax, [ebp+8]
00401010 89 45 F8             mov     [ebp-8], eax
00401013 8B 4D F8             mov     ecx, [ebp-8]
00401016 83 E9 02             sub     ecx, 2
00401019 89 4D F8             mov     [ebp-8], ecx
0040101C 83 7D F8 04          cmp     [ebp-8], 4
00401020 77 35                ja      short loc_401057
00401022 8B 55 F8             mov     edx, [ebp-8]
00401025 FF 24 95 60104000    jmp     off_401060[edx*4]
0040102C loc_40102C:
0040102C C7 45 FC 0A000000    mov     [ebp-4], 0Ah
00401033 EB 22                jmp     short loc_401057
00401035 loc_401035:
00401035 C7 45 FC 03000000    mov     [ebp-4], 3
0040103C EB 19                jmp     short loc_401057
0040103E    loc_40103E:
0040103E C7 45 FC 08000000    mov     [ebp-4], 8
00401045 EB 10                jmp     short loc_401057
00401047    loc_401047:
00401047 C7 45 FC 02000000    mov     [ebp-4], 2
0040104E EB 07                jmp     short loc_401057
00401050    loc_401050:
00401050 C7 45 FC 04000000    mov     [ebp-4], 4
```

switch – Microsoft C – nem optimalizált (folyt.)

```
00401057    loc_401057:
00401057 8B 45 FC            mov     eax, [ebp-4]
0040105A 8B E5               mov     esp, ebp
0040105C 5D                  pop     ebp
0040105D C3                  retn
0040105E 8B FF          ; szóhatárra igazítás
off_401060
00401060 2C 10 40 00         dd offset loc_40102C
00401064 35 10 40 00         dd offset loc_401035
00401068 3E 10 40 00         dd offset loc_40103E
0040106C 47 10 40 00         dd offset loc_401047
00401070 50 10 40 00         dd offset loc_401050
00401074 00 00 00 00 00 00 00 00+          ;align 200h
```

- ugrási tábla a kódban;
- kitöltő byteok ret után mov edi,edi máshol nullák.

switch – Microsoft C – nem optimalizált (BB)



switch – Microsoft C – optimalizált

```
arg_0           = dword ptr  4
00401000        mov     ecx, [esp+arg_0]
00401004        add     ecx, 0FFFFFFFEh
00401007        xor     eax, eax
00401009        cmp     ecx, 4          ; switch 5 cases
0040100C        ja      short locret_401032 ; default
0040100E        jmp     off_401034[ecx*4] ; switch jump
00401015 loc_401015:
00401015        mov     eax, 0Ah        ; case 0x0
0040101A        retn
0040101B loc_40101B:
0040101B        mov     eax, 3          ; case 0x1
00401020        retn
00401021 loc_401021:
00401021        mov     eax, 8          ; case 0x2
00401026        retn
00401027 loc_401027:
00401027        mov     eax, 2          ; case 0x3
0040102C        retn
0040102D loc_40102D:
0040102D        mov     eax, 4          ; case 0x4
00401032 locret_401032:
00401032        retn                    ; default
00401032 start          endp
00401033                align 4
  ; jump table for switch statement
00401034 off_401034     dd offset loc_401015
00401034                dd offset loc_40101B
00401034                dd offset loc_401021
00401034                dd offset loc_401027
00401034                dd offset loc_40102D
```

switch – Intel C – optimalizált

```
00401000    arg_0       = dword ptr  4
00401000     mov     eax, [esp+arg_0]
00401004     lea     eax, [eax-2]     ; switch 5 cases
00401007     cmp     eax, 4
0040100A     ja      short loc_401036 ; default
0040100C     jmp     ds:off_402000[eax*4] ; switch jump
00401013 loc_401013:
00401013     mov     eax, 4           ; case 0x6
00401018     jmp     short locret_401038
0040101A loc_40101A:
0040101A     mov     eax, 2           ; case 0x5
0040101F      jmp     short locret_401038
00401021 loc_401021:
00401021     mov     eax, 8           ; case 0x4
00401026     jmp     short locret_401038
00401028 loc_401028:
00401028     mov     eax, 3           ; case 0x3
0040102D     jmp     short locret_401038
0040102F loc_40102F:
0040102F      mov     eax, 0Ah         ; case 0x2
00401034      jmp     short locret_401038
00401036 loc_401036:
00401036      xor     eax, eax         ; default
00401038 locret_401038:
00401038     retn
00401038 start    endp
00401039     db 0Fh, 1Fh, 80h
0040103C    align 200h
```

Switch utasítás nem folytonos értékekre

```
extern void f1();
extern void f2();
extern void f3();

void foo(int i)
{
    switch (i)
    {
        case 1: f1(); break;
        case 2: f2(); break;
        case 5: f1(); break;
        case 7: f2(); break;
        case 10: f1(); break;
        case 11: f2(); break;
        case 12: f2(); break;
        case 17: f1(); break;
        case 18: f1(); break;

        default: f3();
    }
}
```

Switch utasítás nem folytonos értékekre

```
. . .
 mov    eax, [ebp+arg_0]
 mov    [ebp+var_4], eax
 mov    ecx, [ebp+var_4]
 sub    ecx, 1
 mov    [ebp+var_4], ecx
 cmp    [ebp+var_4], 11h ;
 ja     short $LN1       ;default case
 mov    edx, [ebp+var_4]
 movzx  eax, ds:$LN15[edx]
 jmp    ds:$LN16[eax*4] ; switch jump
. . . . . .
$LN2:
 call   _f1
 jmp    short loc_6E

$LN1:
 call   _f3   ;default case
```

```
$LN16   dd offset $LN10
        dd offset $LN9
        dd offset $LN8
        dd offset $LN7
        dd offset $LN6
        dd offset $LN5
        dd offset $LN4
        dd offset $LN3
        dd offset $LN2
        dd offset $LN1

$LN15   db  0, 1, 9, 9
        db  2, 9, 3, 9
        db  9, 4, 5, 6
        db  9, 9, 9, 9
        db  7, 8
```

Switch utasítás nem folytonos értékekre

Visual C ilyenkor kétszintű switch táblát használ.

```
i2 = i;
i2 = i2-1;
if i2 > 17 goto Default_Label;
goto table2[4*table1[i2]];
```

Ha egy érték sem folytonos, és távol vannak egymástól, akkor a fordító bináris kereséssel valósítja meg az elágazásokat.

Forrás: Zuoliu Ding, Something You May Not Know About the Switch Statement in C/C++

Section 4

while

Fibonacci

```
#include <stdio.h>

int fib(unsigned int m)
{   int f0, f1, f2, i;
    f0 = 0;
    f1 = 1;
    if (m <= 1)
        f2 = m;
    else
    {   for (i=2; i<=m; i++)
        {   f2 = f0 + f1;
            f0 = f1;
            f1 = f2;
        }
    }
    return f2;
}

int main(int argc, char* argv[] ) {
   printf("%d\n", fib(argc) );

   return 0;
}
```

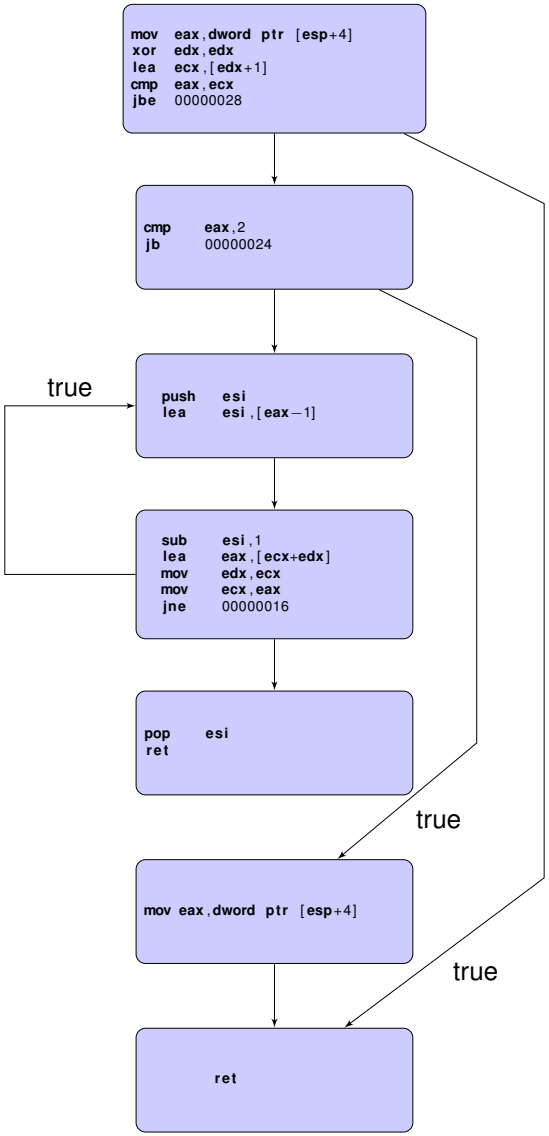
Ciklus — Fibonacci MSVC

```
_fib:
00000000 8B 44 24 04     mov     eax,dword ptr [esp+4]
00000004 33 D2           xor     edx,edx
00000006 8D 4A 01        lea     ecx,[edx+1]
00000009 3B C1           cmp     eax,ecx
0000000B 76 1B           jbe     00000028
0000000D 83 F8 02        cmp     eax,2
00000010 72 12           jb      00000024
00000012 56              push    esi
00000013 8D 70 FF        lea     esi,[eax-1]
00000016 83 EE 01        sub     esi,1
00000019 8D 04 11        lea     eax,[ecx+edx]
0000001C 8B D1           mov     edx,ecx
0000001E 8B C8           mov     ecx,eax
00000020 75 F4           jne     00000016
00000022 5E              pop     esi
00000023 C3              ret
00000024 8B 44 24 04     mov     eax,dword ptr [esp+4]
00000028 C3              ret
```

Fibonacci-t hívó főprogram MSVC

```
_main:
00000000  8B 4C 24 04     mov     ecx,dword ptr [esp+4]
00000004  33 D2           xor     edx,edx
00000006  8D 42 01        lea     eax,[edx+1]
00000009  3B C8           cmp     ecx,eax
0000000B  76 2B           jbe     00000038
0000000D  83 F9 02        cmp     ecx,2
00000010  72 22           jb      00000034
00000012  56              push    esi
00000013  8D 71 FF        lea     esi,[ecx-1]
00000016  83 EE 01        sub     esi,1
00000019  8D 0C 10        lea     ecx,[eax+edx]
0000001C  8B D0           mov     edx,eax
0000001E  8B C1           mov     eax,ecx
00000020  75 F4           jne     00000016
00000022  5E              pop     esi
00000023  51              push    ecx
00000024  68 00000000     push    offset $AA@
00000029  E8 00000000     call    _printf
0000002E  83 C4 08        add     esp,8
00000031  33 C0           xor     eax,eax
00000033  C3              ret
00000034  8B 4C 24 04     mov     ecx,dword ptr [esp+4]
00000038  51              push    ecx
00000039  68 00000000     push    offset $AA@
0000003E  E8 00000000     call    _printf
00000043  83 C4 08        add     esp,8
00000046  33 C0           xor     eax,eax
00000048  C3              ret
```

Fibonacci BB



Dupla ciklus

```
_set:
  00000000 55                   push ebp
  00000001 8B EC                mov  ebp,esp
  00000003 83 EC 08             sub  esp,8
  00000006 C7 45 FC 00000000    mov  dword ptr [ebp-4],0
  0000000D EB 09                jmp  00000018
  0000000F 8B 45 FC             mov  eax,dword ptr [ebp-4]
  00000012 83 C0 01             add  eax,1
  00000015 89 45 FC             mov  dword ptr [ebp-4],eax
  00000018 83 7D FC 64          cmp  dword ptr [ebp-4],64h
  0000001C 7D 37                jge  00000055
  0000001E C7 45 F8 00000000    mov  dword ptr [ebp-8],0
  00000025 EB 09                jmp  00000030
  00000027 8B 4D F8             mov  ecx,dword ptr [ebp-8]
  0000002A 83 C1 01             add  ecx,1
  0000002D 89 4D F8             mov  dword ptr [ebp-8],ecx
  00000030 83 7D F8 64          cmp  dword ptr [ebp-8],64h
  00000034 7D 1D                jge  00000053
  00000036 8B 55 FC             mov  edx,dword ptr [ebp-4]
  00000039 6B D2 64             imul edx,edx,64h
  0000003C 03 55 F8             add  edx,dword ptr [ebp-8]
  0000003F 8B 45 FC             mov  eax,dword ptr [ebp-4]
  00000042 69 C0 90 01 00 00    imul eax,eax,190h
  00000048 03 45 08             add  eax,dword ptr [ebp+8]
  0000004B 8B 4D F8             mov  ecx,dword ptr [ebp-8]
  0000004E 89 14 88             mov  dword ptr [eax+ecx*4],edx
  00000051 EB D4                jmp  00000027
  00000053 EB BA                jmp  0000000F
  00000055 8B E5                mov  esp,ebp
  00000057 5D                   pop  ebp
  00000058 C3                   ret
```

Dupla ciklus

```
_set:
  00000000 55                   push ebp
  00000001 8B EC                mov  ebp,esp
  00000003 83 EC 08             sub  esp,8
  00000006 C7 45 FC 00000000    mov  dword ptr [ebp-4],0
  0000000D EB 09                jmp  00000018
  0000000F 8B 45 FC             mov  eax,dword ptr [ebp-4]
  00000012 83 C0 01             add  eax,1
  00000015 89 45 FC             mov  dword ptr [ebp-4],eax
  00000018 83 7D FC 64          cmp  dword ptr [ebp-4],64h
  0000001C 7D 37                jge  00000055
  0000001E C7 45 F8 00000000    mov  dword ptr [ebp-8],0
  00000025 EB 09                jmp  00000030
  00000027 8B 4D F8             mov  ecx,dword ptr [ebp-8]
  0000002A 83 C1 01             add  ecx,1
  0000002D 89 4D F8             mov  dword ptr [ebp-8],ecx
  00000030 83 7D F8 64          cmp  dword ptr [ebp-8],64h
  00000034 7D 1D                jge  00000053
  00000036 8B 55 FC             mov  edx,dword ptr [ebp-4]
  00000039 6B D2 64             imul edx,edx,64h
  0000003C 03 55 F8             add  edx,dword ptr [ebp-8]
  0000003F 8B 45 FC             mov  eax,dword ptr [ebp-4]
  00000042 69 C0 90 01 00 00    imul eax,eax,190h
  00000048 03 45 08             add  eax,dword ptr [ebp+8]
  0000004B 8B 4D F8             mov  ecx,dword ptr [ebp-8]
  0000004E 89 14 88             mov  dword ptr [eax+ecx*4],edx
  00000051 EB D4                jmp  00000027
  00000053 EB BA                jmp  0000000F
  00000055 8B E5                mov  esp,ebp
  00000057 5D                   pop  ebp
  00000058 C3                   ret
```

ciklusváltozó inicializálása

Dupla ciklus

```
_set:
  00000000 55                   push ebp
  00000001 8B EC                mov  ebp,esp
  00000003 83 EC 08             sub  esp,8
  00000006 C7 45 FC 00000000    mov  dword ptr [ebp-4],0
  0000000D EB 09                jmp  00000018
  0000000F 8B 45 FC             mov  eax,dword ptr [ebp-4]
  00000012 83 C0 01             add  eax,1
  00000015 89 45 FC             mov  dword ptr [ebp-4],eax
  00000018 83 7D FC 64          cmp  dword ptr [ebp-4],64h
  0000001C 7D 37                jge  00000055
  0000001E C7 45 F8 00000000    mov  dword ptr [ebp-8],0
  00000025 EB 09                jmp  00000030
  00000027 8B 4D F8             mov  ecx,dword ptr [ebp-8]
  0000002A 83 C1 01             add  ecx,1
  0000002D 89 4D F8             mov  dword ptr [ebp-8],ecx
  00000030 83 7D F8 64          cmp  dword ptr [ebp-8],64h
  00000034 7D 1D                jge  00000053
  00000036 8B 55 FC             mov  edx,dword ptr [ebp-4]
  00000039 6B D2 64             imul edx,edx,64h
  0000003C 03 55 F8             add  edx,dword ptr [ebp-8]
  0000003F 8B 45 FC             mov  eax,dword ptr [ebp-4]
  00000042 69 C0 90 01 00 00    imul eax,eax,190h
  00000048 03 45 08             add  eax,dword ptr [ebp+8]
  0000004B 8B 4D F8             mov  ecx,dword ptr [ebp-8]
  0000004E 89 14 88             mov  dword ptr [eax+ecx*4],edx
  00000051 EB D4                jmp  00000027
  00000053 EB BA                jmp  0000000F
  00000055 8B E5                mov  esp,ebp
  00000057 5D                   pop  ebp
  00000058 C3                   ret
```

ciklusváltozó növelése

Dupla ciklus

```
_set:
  00000000 55                   push ebp
  00000001 8B EC                mov  ebp,esp
  00000003 83 EC 08             sub  esp,8
  00000006 C7 45 FC 00000000    mov  dword ptr [ebp-4],0
  0000000D EB 09                jmp  00000018
  0000000F 8B 45 FC             mov  eax,dword ptr [ebp-4]
  00000012 83 C0 01             add  eax,1
  00000015 89 45 FC             mov  dword ptr [ebp-4],eax
  00000018 83 7D FC 64          cmp  dword ptr [ebp-4],64h
  0000001C 7D 37                jge  00000055
  0000001E C7 45 F8 00000000    mov  dword ptr [ebp-8],0
  00000025 EB 09                jmp  00000030
  00000027 8B 4D F8             mov  ecx,dword ptr [ebp-8]
  0000002A 83 C1 01             add  ecx,1
  0000002D 89 4D F8             mov  dword ptr [ebp-8],ecx
  00000030 83 7D F8 64          cmp  dword ptr [ebp-8],64h
  00000034 7D 1D                jge  00000053
  00000036 8B 55 FC             mov  edx,dword ptr [ebp-4]
  00000039 6B D2 64             imul edx,edx,64h
  0000003C 03 55 F8             add  edx,dword ptr [ebp-8]
  0000003F 8B 45 FC             mov  eax,dword ptr [ebp-4]
  00000042 69 C0 90 01 00 00    imul eax,eax,190h
  00000048 03 45 08             add  eax,dword ptr [ebp+8]
  0000004B 8B 4D F8             mov  ecx,dword ptr [ebp-8]
  0000004E 89 14 88             mov  dword ptr [eax+ecx*4],edx
  00000051 EB D4                jmp  00000027
  00000053 EB BA                jmp  0000000F
  00000055 8B E5                mov  esp,ebp
  00000057 5D                   pop  ebp
  00000058 C3                   ret
```

ciklusba maradás feltétele

Dupla ciklus

```
_set:
  00000000 55                    push ebp
  00000001 8B EC                 mov  ebp,esp
  00000003 83 EC 08              sub  esp,8
  00000006 C7 45 FC 00000000     mov  dword ptr [ebp-4],0
  0000000D EB 09                 jmp  00000018
  0000000F 8B 45 FC              mov  eax,dword ptr [ebp-4]
  00000012 83 C0 01              add  eax,1
  00000015 89 45 FC              mov  dword ptr [ebp-4],eax
  00000018 83 7D FC 64           cmp  dword ptr [ebp-4],64h
  0000001C 7D 37                 jge  00000055
  0000001E C7 45 F8 00000000     mov  dword ptr [ebp-8],0
  00000025 EB 09                 jmp  00000030
  00000027 8B 4D F8              mov  ecx,dword ptr [ebp-8]
  0000002A 83 C1 01              add  ecx,1
  0000002D 89 4D F8              mov  dword ptr [ebp-8],ecx
  00000030 83 7D F8 64           cmp  dword ptr [ebp-8],64h
  00000034 7D 1D                 jge  00000053
  00000036 8B 55 FC              mov  edx,dword ptr [ebp-4]
  00000039 6B D2 64              imul edx,edx,64h
  0000003C 03 55 F8              add  edx,dword ptr [ebp-8]
  0000003F 8B 45 FC              mov  eax,dword ptr [ebp-4]
  00000042 69 C0 90 01 00 00     imul eax,eax,190h
  00000048 03 45 08              add  eax,dword ptr [ebp+8]
  0000004B 8B 4D F8              mov  ecx,dword ptr [ebp-8]
  0000004E 89 14 88              mov  dword ptr [eax+ecx*4],edx
  00000051 EB D4                 jmp  00000027
  00000053 EB BA                 jmp  0000000F
  00000055 8B E5                 mov  esp,ebp
  00000057 5D                    pop  ebp
  00000058 C3                    ret
```

ciklus törzse

Dupla ciklus

```
_set:
  00000000 55                   push ebp
  00000001 8B EC                mov  ebp,esp
  00000003 83 EC 08             sub  esp,8
  00000006 C7 45 FC 00000000    mov  dword ptr [ebp-4],0
  0000000D EB 09                jmp  00000018
  0000000F 8B 45 FC             mov  eax,dword ptr [ebp-4]
  00000012 83 C0 01             add  eax,1
  00000015 89 45 FC             mov  dword ptr [ebp-4],eax
  00000018 83 7D FC 64          cmp  dword ptr [ebp-4],64h
  0000001C 7D 37                jge  00000055
  0000001E C7 45 F8 00000000    mov  dword ptr [ebp-8],0
  00000025 EB 09                jmp  00000030
  00000027 8B 4D F8             mov  ecx,dword ptr [ebp-8]
  0000002A 83 C1 01             add  ecx,1
  0000002D 89 4D F8             mov  dword ptr [ebp-8],ecx
  00000030 83 7D F8 64          cmp  dword ptr [ebp-8],64h
  00000034 7D 1D                jge  00000053
  00000036 8B 55 FC             mov  edx,dword ptr [ebp-4]
  00000039 6B D2 64             imul edx,edx,64h
  0000003C 03 55 F8             add  edx,dword ptr [ebp-8]
  0000003F 8B 45 FC             mov  eax,dword ptr [ebp-4]
  00000042 69 C0 90 01 00 00    imul eax,eax,190h
  00000048 03 45 08             add  eax,dword ptr [ebp+8]
  0000004B 8B 4D F8             mov  ecx,dword ptr [ebp-8]
  0000004E 89 14 88             mov  dword ptr [eax+ecx*4],edx
  00000051 EB D4                jmp  00000027
  00000053 EB BA                jmp  0000000F
  00000055 8B E5                mov  esp,ebp
  00000057 5D                   pop  ebp
  00000058 C3                   ret
```

belső ciklusváltozó inicializálása

Dupla ciklus

```
_set:
  00000000 55                   push ebp
  00000001 8B EC                mov  ebp,esp
  00000003 83 EC 08             sub  esp,8
  00000006 C7 45 FC 00000000    mov  dword ptr [ebp-4],0
  0000000D EB 09                jmp  00000018
  0000000F 8B 45 FC             mov  eax,dword ptr [ebp-4]
  00000012 83 C0 01             add  eax,1
  00000015 89 45 FC             mov  dword ptr [ebp-4],eax
  00000018 83 7D FC 64          cmp  dword ptr [ebp-4],64h
  0000001C 7D 37                jge  00000055
  0000001E C7 45 F8 00000000    mov  dword ptr [ebp-8],0
  00000025 EB 09                jmp  00000030
  00000027 8B 4D F8             mov  ecx,dword ptr [ebp-8]
  0000002A 83 C1 01             add  ecx,1
  0000002D 89 4D F8             mov  dword ptr [ebp-8],ecx
  00000030 83 7D F8 64          cmp  dword ptr [ebp-8],64h
  00000034 7D 1D                jge  00000053
  00000036 8B 55 FC             mov  edx,dword ptr [ebp-4]
  00000039 6B D2 64             imul edx,edx,64h
  0000003C 03 55 F8             add  edx,dword ptr [ebp-8]
  0000003F 8B 45 FC             mov  eax,dword ptr [ebp-4]
  00000042 69 C0 90 01 00 00    imul eax,eax,190h
  00000048 03 45 08             add  eax,dword ptr [ebp+8]
  0000004B 8B 4D F8             mov  ecx,dword ptr [ebp-8]
  0000004E 89 14 88             mov  dword ptr [eax+ecx*4],edx
  00000051 EB D4                jmp  00000027
  00000053 EB BA                jmp  0000000F
  00000055 8B E5                mov  esp,ebp
  00000057 5D                   pop  ebp
  00000058 C3                   ret
```

belső ciklusváltozó növelése

Dupla ciklus

```
_set:
  00000000 55                   push ebp
  00000001 8B EC                mov  ebp,esp
  00000003 83 EC 08             sub  esp,8
  00000006 C7 45 FC 00000000    mov  dword ptr [ebp-4],0
  0000000D EB 09                jmp  00000018
  0000000F 8B 45 FC             mov  eax,dword ptr [ebp-4]
  00000012 83 C0 01             add  eax,1
  00000015 89 45 FC             mov  dword ptr [ebp-4],eax
  00000018 83 7D FC 64          cmp  dword ptr [ebp-4],64h
  0000001C 7D 37                jge  00000055
  0000001E C7 45 F8 00000000    mov  dword ptr [ebp-8],0
  00000025 EB 09                jmp  00000030
  00000027 8B 4D F8             mov  ecx,dword ptr [ebp-8]
  0000002A 83 C1 01             add  ecx,1
  0000002D 89 4D F8             mov  dword ptr [ebp-8],ecx
  00000030 83 7D F8 64          cmp  dword ptr [ebp-8],64h
  00000034 7D 1D                jge  00000053
  00000036 8B 55 FC             mov  edx,dword ptr [ebp-4]
  00000039 6B D2 64             imul edx,edx,64h
  0000003C 03 55 F8             add  edx,dword ptr [ebp-8]
  0000003F 8B 45 FC             mov  eax,dword ptr [ebp-4]
  00000042 69 C0 90 01 00 00    imul eax,eax,190h
  00000048 03 45 08             add  eax,dword ptr [ebp+8]
  0000004B 8B 4D F8             mov  ecx,dword ptr [ebp-8]
  0000004E 89 14 88             mov  dword ptr [eax+ecx*4],edx
  00000051 EB D4                jmp  00000027
  00000053 EB BA                jmp  0000000F
  00000055 8B E5                mov  esp,ebp
  00000057 5D                   pop  ebp
  00000058 C3                   ret
```

belső ciklusba maradás feltétele

Dupla ciklus

```
_set:
  00000000 55                   push ebp
  00000001 8B EC                mov  ebp,esp
  00000003 83 EC 08             sub  esp,8
  00000006 C7 45 FC 00000000    mov  dword ptr [ebp-4],0
  0000000D EB 09                jmp  00000018
  0000000F 8B 45 FC             mov  eax,dword ptr [ebp-4]
  00000012 83 C0 01             add  eax,1
  00000015 89 45 FC             mov  dword ptr [ebp-4],eax
  00000018 83 7D FC 64          cmp  dword ptr [ebp-4],64h
  0000001C 7D 37                jge  00000055
  0000001E C7 45 F8 00000000    mov  dword ptr [ebp-8],0
  00000025 EB 09                jmp  00000030
  00000027 8B 4D F8             mov  ecx,dword ptr [ebp-8]
  0000002A 83 C1 01             add  ecx,1
  0000002D 89 4D F8             mov  dword ptr [ebp-8],ecx
  00000030 83 7D F8 64          cmp  dword ptr [ebp-8],64h
  00000034 7D 1D                jge  00000053
  00000036 8B 55 FC             mov  edx,dword ptr [ebp-4]
  00000039 6B D2 64             imul edx,edx,64h
  0000003C 03 55 F8             add  edx,dword ptr [ebp-8]
  0000003F 8B 45 FC             mov  eax,dword ptr [ebp-4]
  00000042 69 C0 90 01 00 00    imul eax,eax,190h
  00000048 03 45 08             add  eax,dword ptr [ebp+8]
  0000004B 8B 4D F8             mov  ecx,dword ptr [ebp-8]
  0000004E 89 14 88             mov  dword ptr [eax+ecx*4],edx
  00000051 EB D4                jmp  00000027
  00000053 EB BA                jmp  0000000F
  00000055 8B E5                mov  esp,ebp
  00000057 5D                   pop  ebp
  00000058 C3                   ret
```

belső ciklus törzse

Dupla ciklus optimalizálva

```
_set:
  00000000 8B 4C 24 04          mov  ecx,dword ptr [esp+4]
  00000004 33 D2                xor  edx,edx
  00000006 56                   push esi
  00000007 33 C0                xor  eax,eax
  00000009 8D A4 24 00000000    lea  esp,[esp+00000000h]
  00000010 8D 34 02             lea  esi,[edx+eax]
  00000013 89 31                mov  dword ptr [ecx],esi
  00000015 40                   inc  eax
  00000016 83 C1 04             add  ecx,4
  00000019 83 F8 64             cmp  eax,64h
  0000001C 7C F2                jl   00000010
  0000001E 83 C2 64             add  edx,64h
  00000021 81 FA 10270000       cmp  edx,2710h
  00000027 7C DE                jl   00000007
  00000029 5E                   pop  esi
  0000002A C3                   ret
```

Dupla ciklus optimalizálva

```
_set:
  00000000 8B 4C 24 04          mov  ecx,dword ptr [esp+4]
  00000004 33 D2                xor  edx,edx
  00000006 56                   push esi
  00000007 33 C0                xor  eax,eax
  00000009 8D A4 24 00000000    lea  esp,[esp+00000000h]
  00000010 8D 34 02             lea  esi,[edx+eax]
  00000013 89 31                mov  dword ptr [ecx],esi
  00000015 40                   inc  eax
  00000016 83 C1 04             add  ecx,4
  00000019 83 F8 64             cmp  eax,64h
  0000001C 7C F2                jl   00000010
  0000001E 83 C2 64             add  edx,64h
  00000021 81 FA 10270000       cmp  edx,2710h
  00000027 7C DE                jl   00000007
  00000029 5E                   pop  esi
  0000002A C3                   ret
```

belső ciklus

Dupla ciklus optimalizálva

```
_set:
  00000000 8B 4C 24 04          mov  ecx,dword ptr [esp+4]
  00000004 33 D2                xor  edx,edx
  00000006 56                   push esi
  00000007 33 C0                xor  eax,eax
  00000009 8D A4 24 00000000    lea  esp,[esp+00000000h]
  00000010 8D 34 02             lea  esi,[edx+eax]
  00000013 89 31                mov  dword ptr [ecx],esi
  00000015 40                   inc  eax
  00000016 83 C1 04             add  ecx,4
  00000019 83 F8 64             cmp  eax,64h
  0000001C 7C F2                jl   00000010
  0000001E 83 C2 64             add  edx,64h
  00000021 81 FA 10270000       cmp  edx,2710h
  00000027 7C DE                jl   00000007
  00000029 5E                   pop  esi
  0000002A C3                   ret
```

külső ciklus

Dupla ciklus optimalizálva

```
_set:
  00000000 8B 4C 24 04          mov   ecx,dword ptr [esp+4]
  00000004 33 D2                xor   edx,edx
  00000006 56                   push  esi
  00000007 33 C0                xor   eax,eax
  00000009 8D A4 24 00000000    lea   esp,[esp+00000000h]
  00000010 8D 34 02             lea   esi,[edx+eax]
  00000013 89 31                mov   dword ptr [ecx],esi
  00000015 40                   inc   eax
  00000016 83 C1 04             add   ecx,4
  00000019 83 F8 64             cmp   eax,64h
  0000001C 7C F2                jl    00000010
  0000001E 83 C2 64             add   edx,64h
  00000021 81 FA 10270000       cmp   edx,2710h
  00000027 7C DE                jl    00000007
  00000029 5E                   pop   esi
  0000002A C3                   ret
```

ESI regisztert a hívott rutinnak kell elmenteni

Dupla ciklus optimalizálva

```
_set:
  00000000 8B 4C 24 04          mov  ecx,dword ptr [esp+4]
  00000004 33 D2                xor  edx,edx
  00000006 56                   push esi
  00000007 33 C0                xor  eax,eax
  00000009 8D A4 24 00000000    lea  esp,[esp+00000000h]
  00000010 8D 34 02             lea  esi,[edx+eax]
  00000013 89 31                mov  dword ptr [ecx],esi
  00000015 40                   inc  eax
  00000016 83 C1 04             add  ecx,4
  00000019 83 F8 64             cmp  eax,64h
  0000001C 7C F2                jl   00000010
  0000001E 83 C2 64             add  edx,64h
  00000021 81 FA 10270000       cmp  edx,2710h
  00000027 7C DE                jl   00000007
  00000029 5E                   pop  esi
  0000002A C3                   ret
```

10000

Dupla ciklus optimalizálva

```
_set:
  00000000 8B 4C 24 04          mov  ecx,dword ptr [esp+4]
  00000004 33 D2                xor  edx,edx
  00000006 56                   push esi
  00000007 33 C0                xor  eax,eax
  00000009 8D A4 24 00000000    lea  esp,[esp+00000000h]
  00000010 8D 34 02             lea  esi,[edx+eax]
  00000013 89 31                mov  dword ptr [ecx],esi
  00000015 40                   inc  eax
  00000016 83 C1 04             add  ecx,4
  00000019 83 F8 64             cmp  eax,64h
  0000001C 7C F2                jl   00000010
  0000001E 83 C2 64             add  edx,64h
  00000021 81 FA 10270000       cmp  edx,2710h
  00000027 7C DE                jl   00000007
  00000029 5E                   pop  esi
  0000002A C3                   ret
```

Mi ez ?

Előző program kódja

```
void set( int matrix[][100] )
{
    int i, j;
    for( i=0; i<100; i++)
        for(j=0; j<100; j++)
          matrix[i][j] = 100*i+j;
}
```

Section 5

Ciklus optimalizálás

Ciklus kiértékelése fordítási idő alatt

```
int main() {
    int i=0, s=0, k=5;
    for( i=0; i<k; i++) s = s+i;
    printf("%d\n", s );
    return 0;
}
```

```
_main:
  00000000 6A 0A              push 0Ah
  00000002 68 00 00 00 00     push offset ??_C@_03PMGGPEJJ@?$CFd?6?$AA@
  00000007 E8 00 00 00 00     call _printf
  0000000C 83 C4 08           add  esp,8
  0000000F 33 C0              xor  eax,eax
  00000011 C3                 ret
```

Section 6

x87 processzor

x87 processzor regiszterei

Olvasmány!

- regiszterek
  - 8 darab regiszter
  - stack szervezés
  - 32, 64 és 80 bites float számokat tartalmazhat
- önálló flag regiszter
- a stack tartalmát függvényhívások között megőrzi

x87 processzor adatmozgató utasítások

Olvasmány!

- FLD – Load Floating Point
- FST – Store Floating Point
- FSTP – Store Floating Point and Pop
- FXCH – Exchange Register Contents
- FILD – Load Integer

x87 processzor – fontosabb konstansok

Olvasmány!

- FLDZ Load +0.0
- FLD1 Load +1.0
- FLDPI Load π
- FLDL2T Load $log_2(10)$
- FLDL2E Load $log_2(e)$
- FLDLG2 Load $log_{10}(2)$
- FLDLN2 Load $log_e(2)$

x87 alaputasítások

Olvasmány!

- FADD/FADDP Add floating point
- FIADD Add integer to floating point
- FSUB/FSUBP Subtract floating point
- FISUB Subtract integer from floating point
- FSUBR/FSUBRP Reverse subtract floating point
- FISUBR Reverse subtract floating point from integer
- FMUL/FMULP Multiply floating point
- FIMUL Multiply integer by floating point
- FDIV/FDIVP Divide floating point
- FIDIV Divide floating point by integer
- FDIVR/FDIVRP Reverse divide
- FIDIVR Reverse divide integer by floating point
- FABS Absolute value
- FCHS Change sign
- FSQRT Square root

x87 reverse utasítás értelmezése

Olvasmány!

FSUB:

$$ST(0) \leftarrow ST(0) - ST(i)$$
$$ST(i) \leftarrow ST(i) - ST(0)$$

FSUBR:

$$ST(0) \leftarrow ST(i) - ST(0)$$
$$ST(i) \leftarrow ST(0) - ST(i)$$

Másodfokú egyenletmegoldó program

Olvasmány!

```
int masodfoku(double a, double b, double c,
              double* res1, double *res2)
{
    double disc;
    disc = b*b - 4*a*c;
    if( disc < 0 ) return 0;
    *res1 = (-b+sqrt(disc))/(2*a);
    *res2 = (-b-sqrt(disc))/(2*a);
    return 1;
}
```

Másodfokú egyenletmegoldó gépi kódja

Olvasmány!

```
_masodfoku:
  0000  fld     qword ptr [esp+0Ch]  st: b
  0004  fld     st(0)                st: b, b
  0006  fmul    st,st(1)             st: b*b, b
  0008  fld     qword ptr [esp+4]    st: a, b*b, b
  000C  fld     qword ptr [esp+14h]  st: c, a, b*b, b
  0010  fmul    st,st(1)             st: c*a, a, b*b, b
  0012  fmul    qword ptr [__real_4] st: 4*c*a, a, b*b, b
  0018  fsubp   st(2),st             st: a, b*b-4*a*c, b
  001A  fldz                         st: 0, a, b*b-4*a*c, b
  001C  fcomp   st(2)                st: a, disc, b
  001E  fnstsw  ax                   ax <- status word
  0020  test    ah,41h
  0023  jne     0000002E             0 <= disc
  0025  fstp    st(1)                st: disc, b
  0027  xor     eax,eax
  0029  fstp    st(1)                st: b
  002B  fstp    st(0)                st:
  002D  ret
  002E  fxch    st(1)                st: disc, a, b
  0030  mov     eax,dword ptr [esp+1Ch]
  0034  fsqrt                        st: sqrt(disc), a, b
  0036  mov     ecx,dword ptr [esp+20h]
  003A  fxch    st(1)                st: a, sqrt(disc), b
  003C  fadd    st(0),st             st: 2*a, sqrt(disc), b
  003E  fld     st(1)                st: sqrt(disc), 2*a, sqrt(disc), b
  0040  fsub    st,st(3)             st: sqrt(disc)-b, 2*a, sqrt(disc), b
  0042  fdiv    st,st(1)             st: (sqrt(disc)-b)/ (2*a), 2*a, sqrt(disc), b
  0044  fstp    qword ptr [eax]      st: 2*a, sqrt(disc), b
  0046  fxch    st(2)                st: b, sqrt(disc), 2*a
  0048  fchs                         st: -b, sqrt(disc), 2*a
  004A  mov     eax,1
  004F  fsubrp  st(1),st             st: -b-sqrt(disc), 2*a /reverse !!! rp
  0051  fdivrp  st(1),st             st: -b-sqrt(disc)/ 2*a
  0053  fstp    qword ptr [ecx]
  0055  ret
```

Másodfokú egyenletmegoldó gépi kódja – S

Olvasmány!

```
_masodfoku:
  00000000: F2 0F 10 5C 24 04   movsd   xmm3,mmword ptr [esp+4]
  00000006: 66 0F EF C9         pxor    xmm1,xmm1
  0000000A: F2 0F 10 05000000   movsd   xmm0,mmword ptr [_2il0floatpacket.5]
  00000012: F2 0F 59 C3         mulsd   xmm0,xmm3
  00000016: F2 0F 10 64 24 0C   movsd   xmm4,mmword ptr [esp+0Ch]
  0000001C: 0F 28 D4            movaps  xmm2,xmm4
  0000001F: F2 0F 59 44 24 14   mulsd   xmm0,mmword ptr [esp+14h]
  00000025: F2 0F 59 D4         mulsd   xmm2,xmm4
  00000029: F2 0F 5C D0         subsd   xmm2,xmm0
  0000002D: 66 0F 2F CA         comisd  xmm1,xmm2
  00000031: 76 03               jbe     00000036
  00000033: 33 C0               xor     eax,eax
  00000035: C3                  ret
  00000036: F2 0F 51 D2         sqrtsd  xmm2,xmm2
  0000003A: F2 0F 58 DB         addsd   xmm3,xmm3
  0000003E: 0F 28 C2            movaps  xmm0,xmm2
  00000041: 66 0F EF C9         pxor    xmm1,xmm1
  00000045: 8B 44 24 1C         mov     eax,dword ptr [esp+1Ch]
  00000049: F2 0F 5C C4         subsd   xmm0,xmm4
  0000004D: F2 0F 58 E2         addsd   xmm4,xmm2
  00000051: F2 0F 5E C3         divsd   xmm0,xmm3
  00000055: 8B 54 24 20         mov     edx,dword ptr [esp+20h]
  00000059: F2 0F 5C CC         subsd   xmm1,xmm4
  0000005D: F2 0F 5E CB         divsd   xmm1,xmm3
  00000061: F2 0F 11 00         movsd   mmword ptr [eax],xmm0
  00000065: B8 01 00 00 00      mov     eax,1
  0000006A: F2 0F 11 0A         movsd   mmword ptr [edx],xmm1
  0000006E: C3                  ret
```

Section 7

Adatok és típusok

Típus visszaállítás

```
_sum:
    mov   edx,dword ptr [esp+4]
    mov   eax,dword ptr [edx]
    add   eax,0Ah
    ret
    lea   esi,[esi+00000000h]
```

Továbbra sem ismert a visszatérési érték típusa, de azt tudjuk, hogy 32 bites numerikus típus.

Típus visszaállítás

```
_sum:
    mov   edx,dword ptr [esp+4]
    mov   eax,dword ptr [edx]
    add   eax,0Ah
    ret
    lea   esi,[esi+00000000h]
```

????? sum(?????) {

Továbbra sem ismert a visszatérési érték típusa, de azt tudjuk, hogy 32 bites numerikus típus.

Típus visszaállítás

```
_sum:
    mov   edx,dword ptr [esp+4]
    mov   eax,dword ptr [edx]
    add   eax,0Ah
    ret
    lea   esi,[esi+00000000h]
```

```
????? sum( ????? ) {
    dword t1 = args1;
```

Továbbra sem ismert a visszatérési érték típusa, de azt tudjuk, hogy 32 bites numerikus típus.

Típus visszaállítás

```
_sum:
    mov   edx,dword ptr [esp+4]
    mov   eax,dword ptr [edx]
    add   eax,0Ah
    ret
    lea   esi,[esi+00000000h]
```

```
????? sum( dword args1 ) {
        dword t1 = args1;
```

Továbbra sem ismert a visszatérési érték típusa, de azt tudjuk, hogy 32 bites numerikus típus.

Típus visszaállítás

```
_sum:
    mov   edx,dword ptr [esp+4]
    mov   eax,dword ptr [edx]
    add   eax,0Ah
    ret
    lea   esi,[esi+00000000h]
```

```
????? sum( dword args1 ) {
        dword t1 = args1;
        dword t2 = *(dword*)t1;
```

Továbbra sem ismert a visszatérési érték típusa, de azt tudjuk, hogy 32 bites numerikus típus.

Típus visszaállítás

```
_sum:
    mov   edx,dword ptr [esp+4]
    mov   eax,dword ptr [edx]
    add   eax,0Ah
    ret
    lea   esi,[esi+00000000h]
```

```
????? sum( dword* args1 ) {
        dword t1 = args1;
        dword t2 = *(dword*)t1;
```

Továbbra sem ismert a visszatérési érték típusa, de azt tudjuk, hogy 32 bites numerikus típus.

Típus visszaállítás

```
_sum:
    mov   edx,dword ptr [esp+4]
    mov   eax,dword ptr [edx]
    add   eax,0Ah
    ret
    lea   esi,[esi+00000000h]
```

```
????? sum( dword* args1 ) {
        dword t1 = args1;
        dword t2 = *(dword*)t1;
        t2 = t2 +10;
```

Továbbra sem ismert a visszatérési érték típusa, de azt tudjuk, hogy 32 bites numerikus típus.

Típus visszaállítás

```
_sum:
    mov   edx,dword ptr [esp+4]
    mov   eax,dword ptr [edx]
    add   eax,0Ah
    ret
    lea   esi,[esi+00000000h]
```

```
????? sum( dword* args1 ) {
        dword t1 = args1;
        dword t2 = *(dword*)t1;
        t2 = t2 +10;
        return t2;
}
```

Továbbra sem ismert a visszatérési érték típusa, de azt tudjuk, hogy 32 bites numerikus típus.

Típus visszaállítás

```
_sum:
    mov   edx,dword ptr [esp+4]
    mov   eax,dword ptr [edx]
    add   eax,0Ah
    ret
    lea   esi,[esi+00000000h]
```

```
dword sum( dword* args1 ) {
        dword t1 = args1;
        dword t2 = *(dword*)t1;
        t2 = t2 +10;
        return t2;
}
```

Továbbra sem ismert a visszatérési érték típusa, de azt tudjuk, hogy 32 bites numerikus típus.

Globális adatok inicializálása

```
char c;
int  i;
short int h;
int p1;
int p2;
int p3;
void init()
{
    c = 'A';
    h = 10;
    i = 1024;
    p1 = 0x401000;
    p2 = 0x401003;
    p3 = 0x401005;
}
```

Globális adatok inicializálása

```
00401000                                        public start
00401000                                start   proc near
00401000 B8 0A 00 00 00                         mov   eax, 0Ah
00401005
00401005                                loc_401005:
00401005 C6 05 00 20 40 00 41                   mov   byte_402000, 41h
0040100C 66 A3 02 20 40 00                      mov   word_402002, ax
00401012 C7 05 04 20 40 00 00 04 00 00          mov   dword_402004, 400h
0040101C C7 05 08 20 40 00 00 10 40 00          mov   dword_402008, offset start
00401026 C7 05 0C 20 40 00 03 10 40 00          mov   dword_40200C, 401003h
00401030 C7 05 10 20 40 00 05 10 40 00          mov   dword_402010, offset loc_401005
0040103A C3                                     retn
0040103A                                start   endp
```

```
00402000 ??                             byte_402000     db ?
00402001 ??                                             align 2
00402002 ?? ??                          word_402002     dw ?
00402004 ?? ?? ?? ??                    dword_402004    dd ?
00402008 ?? ?? ?? ??                    dword_402008    dd ?
0040200C ?? ?? ?? ??                    dword_40200C    dd ?
00402010 ?? ?? ?? ??                    dword_402010    dd ?
00402010                                _data           ends
```

Globális adatok inicializálása

- karakter típusú adatra: mov byte_402000, 41h
- short int típusú adatra: mov word_402002, ax
- int típusú adatra: mov dword_402004, 400h
- Ha a kódszegmensben az adott címen utasítás van, akkor az IDA *beugrik*. PI.
  - mov dword_402008, offset start
  - mov dword_402010, offset loc_401005

Strukturák visszaadása visszatérési értékként

```
#include <stdio.h>
typedef struct A_struct  {
    int x;
    int y;
    int z;
} A;

A init_A(int value) {
    A v;
    v.x = value;
    v.y = 20;
    v.z = 30;

    return v;
}

int main() {
    A r;
     r = init_A(5);
    printf("(%d,%d,%d)",r.x,r.y,r.z);
    return 0;
}
```

Strukturák visszaadása visszatérési értékként

```
#include <stdio.h>
typedef struct A_struct  {
    int x;
    int y;
    int z;
} A;

A init_A(int value) {
    A v;
    v.x = value;
    v.y = 20;
    v.z = 30;

    return v;
}
int main() {
    A r;
     r = init_A(5);
    printf("(%d,%d,%d)",r.x,r.y,r.z);
    return 0;
}
```

Lokális struktúra inicializálása.

Strukturák visszaadása visszatérési értékként

```
#include <stdio.h>
typedef struct A_struct  {
    int x;
    int y;
    int z;
} A;

A init_A(int value) {
    A v;
    v.x = value;
    v.y = 20;
    v.z = 30;

    return v;
}

int main() {
    A r;
     r = init_A(5);
    printf("(%d,%d,%d)",r.x,r.y,r.z);
    return 0;
}
```

Visszatérési érték 0. paraméterként kerül átadásra.

Strukturák visszaadása visszatérési értékként

```
#include <stdio.h>
typedef struct A_struct  {
    int x;
    int y;
    int z;
} A;

A init_A(int value) {
    A v;
    v.x = value;
    v.y = 20;
    v.z = 30;
    return v;
}

int main() {
    A r;
     r = init_A(5);
    printf("(%d,%d,%d)",r.x,r.y,r.z);
    return 0;
}
```

Lokális struktúra átmásolása a paraméterként megadott területre.

Strukturák visszaadása visszatérési értékként

```
#include <stdio.h>
typedef struct A_struct  {
    int x;
    int y;
    int z;
} A;

A init_A(int value) {
    A v;
    v.x = value;
    v.y = 20;
    v.z = 30;

    return v;
}

int main() {
    A r;
    r = init_A(5);
    printf("(%d,%d,%d)",
    return 0;
}
```

Visszaadott érték átmásolása r-be.

Strukturák visszaadása visszatérési értékként

```
0000000C _init_A    proc near
0000000C            push    ebp
0000000D            mov     ebp, esp
0000000F            sub     esp, 0Ch
00000012            mov     eax, [ebp+0Ch]
00000015            mov     [ebp - 0Ch], eax
00000018            mov     [ebp - 8], 14h
0000001F            mov     [ebp- 4], 1Eh
00000026            mov     ecx, [ebp+8]
00000029            mov     edx, [ebp- 0Ch]
0000002C            mov     [ecx], edx
0000002E            mov     eax, [ebp - 8]
00000031            mov     [ecx+4], eax
00000034            mov     edx, [ebp - 4]
00000037            mov     [ecx+8], edx
0000003A            mov     eax, [ebp + 8]
0000003D            mov     esp, ebp
0000003F            pop     ebp
00000040            retn
00000040 _init_A        endp
```

Strukturák visszaadása visszatérési értékként

```
0000000C _init_A    proc near
0000000C            push    ebp
0000000D            mov     ebp, esp
0000000F            sub     esp, 0Ch
00000012            mov     eax, [ebp+0Ch]
00000015            mov     [ebp - 0Ch], eax
00000018            mov     [ebp - 8], 14h
0000001F            mov     [ebp- 4], 1Eh
00000026            mov     ecx, [ebp+8]
00000029            mov     edx, [ebp- 0Ch]
0000002C            mov     [ecx], edx
0000002E            mov     eax, [ebp - 8]
00000031            mov     [ecx+4], eax
00000034            mov     edx, [ebp - 4]
00000037            mov     [ecx+8], edx
0000003A            mov     eax, [ebp + 8]
0000003D            mov     esp, ebp
0000003F            pop     ebp
00000040            retn
00000040 _init_A    endp
```

v változó inicializálása.

Strukturák visszaadása visszatérési értékként

```
0000000C _init_A    proc near
0000000C            push    ebp
0000000D            mov     
0000000F            sub     
00000012            mov     
00000015            mov     
00000018            mov     
0000001F            mov     [ebp-4], 1Eh
00000026            mov     ecx, [ebp+8]
00000029            mov     edx, [ebp- 0Ch]
0000002C            mov     [ecx], edx
0000002E            mov     eax, [ebp - 8]
00000031            mov     [ecx+4], eax
00000034            mov     edx, [ebp - 4]
00000037            mov     [ecx+8], edx
0000003A            mov     eax, [ebp + 8]
0000003D            mov     esp, ebp
0000003F            pop     ebp
00000040            retn
00000040 _init_A    endp
```

v változó tartalmának az átmásolása a 0. paraméterben megadott helyre.

Strukturák visszaadása visszatérési értékként

```
_main        proc near
0000004C             push    ebp
0000004D             mov     ebp, esp
0000004F             sub     esp, 3
00000052             and     esp, 0FFFFFFF8h
00000055             add     esp, 4
00000058             sub     esp, 18h
0000005B             lea     eax, dword ptr [ebp - 10h]
0000005E             mov     dword ptr [esp], eax
00000061             mov     dword ptr [esp+4], 5
00000069             call    _init_A
0000006E             add     esp, 8
00000071             add     esp, 0FFFFFFF0h
00000074             mov     dword ptr [esp], offset aDDD ; "(%d,%d,%d)"
0000007B             mov     eax, dword ptr [ebp - 10h]
0000007E             mov     dword ptr [esp+4], eax
00000082             mov     eax, dword ptr [ebp - 0Ch]
00000085             mov     dword ptr [esp+8], eax
00000089             mov     eax, dword ptr [ebp - 8h]
0000008C             mov     dword ptr [esp+0Ch], eax
00000090             call    _printf
00000095             add     esp, 10h
00000098             mov     dword ptr [ebp - 4h], eax
0000009B             mov     eax, 0
000000A0             leave
000000A1             retn
000000A1 _main       endp
```

Strukturák visszaadása visszatérési értékként

```
_main        proc near
0000004C             push    ebp
0000004D             mov     ebp, esp
0000004F             sub     esp, 3
00000052             and     esp, 0FF
00000055             add     esp, 4
00000058             sub     esp, 18h
0000005B             lea     eax, dword ptr [ebp - 10h]
0000005E             mov     dword ptr [esp], eax
00000061             mov     dword ptr [esp+4], 5
00000069             call    _init_A
0000006E             add     esp, 8
00000071             add     esp, 0FFFFFFF0h
00000074             mov     dword ptr [esp], offset aDDD ; "(%d,%d,%d)"
0000007B             mov     eax, dword ptr [ebp - 10h]
0000007E             mov     dword ptr [esp+4], eax
00000082             mov     eax, dword ptr [ebp - 0Ch]
00000085             mov     dword ptr [esp+8], eax
00000089             mov     eax, dword ptr [ebp - 8h]
0000008C             mov     dword ptr [esp+0Ch], eax
00000090             call    _printf
00000095             add     esp, 10h
00000098             mov     dword ptr [ebp - 4h], eax
0000009B             mov     eax, 0
000000A0             leave
000000A1             retn
000000A1 _main       endp
```

r struktúra címének a stackre helyezése.

Strukturák visszaadása visszatérési értékként

```
_main        proc near
0000004C             push    ebp
0000004D             mov     ebp, esp
0000004F             sub     esp, 3
00000052             and     esp, 0FFFFFFF8h
00000055             add     esp, 4
00000058             sub     esp, 18h
0000005B             lea     eax, dword ptr [ebp - 10h]
0000005E             mov     dword ptr [esp], eax
00000061             mov     dword ptr [esp+4], 3
00000069             call    _init_A
0000006E             add     esp, 8
00000071             add     esp, 0FFFFFFF0h
00000074             mov     dword ptr [esp], offset aDDD ; "(%d,%d,%d)"
0000007B             mov     eax, dword ptr [ebp - 10h]
0000007E             mov     dword ptr [esp+4], eax
00000082             mov     eax, dword ptr [ebp - 0Ch]
00000085             mov     dword ptr [esp+8], eax
00000089             mov     eax, dword ptr [ebp - 8h]
0000008C             mov     dword ptr [esp+0Ch], eax
00000090             call    _printf
00000095             add     esp, 10h
00000098             mov     dword ptr [ebp - 4h], eax
0000009B             mov     eax, 0
000000A0             leave
000000A1             retn
000000A1 _main       endp
```

r változó értékének a stackre helyezése a kiíratáshoz.

Strukturák visszaadása visszatérési értékként (optimalizált 1.)

```
00000000 _main          proc near
00000000 var_80         = dword ptr -80h
00000000                push    ebp
00000001                mov     ebp, esp
00000003                and     esp, 0FFFFFF80h
00000006                sub     esp, 80h
0000000C                push    3
0000000E                call    ___intel_new_proc_init
00000013                stmxcsr [esp+84h+var_80]
00000018                or      [esp+84h+var_80], 8000h
00000020                ldmxcsr [esp+84h+var_80]
00000025                push    1Eh[6]
00000027                push    14h
00000029                push    5
0000002B                push    offset ??_C@_0L@A@?$CJ?$AA@ ; "(%d,%d,%d)"
00000030                call    _printf
00000035                xor     eax, eax
00000037                mov     esp, ebp
00000039                pop     ebp
0000003A                retn
0000003A _main          endp
```

[6] Inline hívás + optimalizálás.

Strukturák visszaadása visszatérési értékként (optimalizált 1.)

```
00000000 _main          proc near
00000000 var_80         = dword ptr -80h
00000000                pu
00000001                mo
00000003                ar                FFFFF80h
00000006                su
0000000C                pu
0000000E                call    ___intel_new_proc_init
00000013                stmxcsr [esp+84h+var_80]
00000018                or      [esp+84h+var_80], 8000h
00000020                ldmxcsr [esp+84h+var_80]
00000025                push    1Eh[6]
00000027                push    14h
00000029                push    5
0000002B                push    offset ??_C@_0L@A@?$CJ?$AA@ ; "(%d,%d,%d)"
00000030                call    _printf
00000035                xor     eax, eax
00000037                mov     esp, ebp
00000039                pop     ebp
0000003A                retn
0000003A _main          endp
```

Flush to Zero – SSE lebegőpontos számítás gyorsítása.

[6] Inline hívás + optimalizálás.

Strukturák visszaadása visszatérési értékként (optimalizált 1.)

```
00000000 _main          proc near
00000000 var_80         = dword ptr -80h
00000000                push    ebp
00000001                mov     ebp
00000003                sub     esp, 84h
00000006                stmxcsr dword ptr [esp+84h+var_80]
0000000C                mov     eax, [esp+84h+var_80]
0000000E                or      eax, 8000h
00000013                mov     [esp+84h+var_84], eax
00000018                ldmxcsr dword ptr [esp+84h+var_80]
00000020                push    1Eh[6]
00000025                push    14h
00000027                push    5
00000029                push    offset ??_C@_0L@A@?$CJ?$AA@ ; "(%d,%d,%d)"
0000002B                call    _printf
00000030                xor     eax, eax
00000035                mov     esp, ebp
00000037                pop     ebp
00000039                retn
0000003A _main          endp
```

A fordító tudja, hogy az init_A() függvény mit csinál, ezért tudja, hogy milyen értékeket kell átadni a struktúramezők kiíratásához. Az init_A() függvényt nem hívja meg, és az r struktúrát sem inicializálja.

[6] Inline hívás + optimalizálás.

Strukturák visszaadása visszatérési értékként (optimalizált 2.)

```
00000050 _init_A    proc near
00000050 arg_0      = dword ptr  8
00000050 arg_4      = dword ptr  0Ch
00000050            push    edi
00000051            mov     ecx, 14h
00000056            mov     eax, [esp+arg_0]
0000005A            mov     edi, 1Eh
0000005F            mov     edx, [esp+arg_4]
00000063            mov     [eax], edx
00000065            mov     [eax+4], ecx
00000068            mov     [eax+8], edi
0000006B            pop     edi
0000006C            retn
0000006C _init_A        endp
```