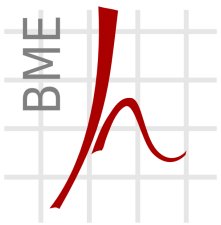
Stack – if – switch – while

Kódvisszafejtés.



Híradástechnikai Tanszék

Izsó Tamás

2012. november 8.

Section 1

stack

__cdecl hívási konvenció IA-32

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

```
_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
```

```
_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
```

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

__fastcall hívási konvenció

```
@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
```

```
_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
```

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

Változó típusú paraméterekátadása 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 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'
```

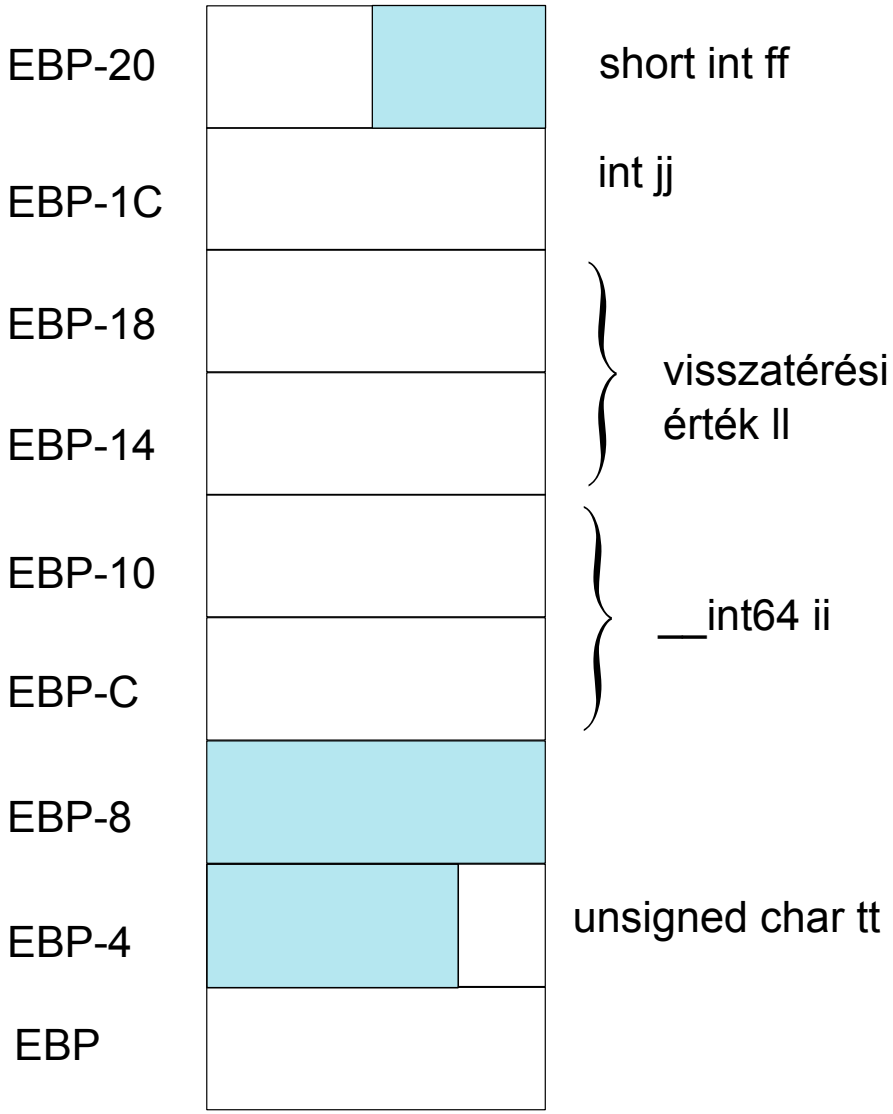
adat méretét módosító prefix

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+var_C]
0000003A 50                 push    eax
0000003B 8B 4D F0           mov     ecx, [
0000003E 51                 push    ecx
0000003F 0F B7 55 E0        movzx   edx
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+var_18]
00000059 51                 push    ecx
0000005A 68 00 00 00 00     push    offset $SG2478
0000005F E8 4C 00 00 00     call    _pri
00000064 83 C4 0C           add     esp,
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
00000044 E8 2D 00 00 00
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]
0000004F 89 55 EC             mov     [ebp
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

```c
#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

```c
#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;
}
```

A makró szóhatárra igazítást végez!

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 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 **EBX** bázisregisztert. Magyarázzuk meg a **lea ecx**,[**esp**+0**Ch**] 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

```c
#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
00401060 2C 10 40 00        off_401060      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 – 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
```

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 Intel

```
_fib:
  00000000 56              push    esi
  00000001 56              push    esi
  00000002 8B 44 24 0C     mov     eax,dword ptr [esp+0Ch]
  00000006 83 F8 01        cmp     eax,1
  00000009 76 20           jbe     0000002B
  0000000B 89 3C 24        mov     dword ptr [esp],edi
  0000000E 33 D2           xor     edx,edx
  00000010 B9 01 000000    mov     ecx,1
  00000015 BE 02 000000    mov     esi,2
  0000001A 8B F8           mov     edi,eax
  0000001C 46              inc     esi
  0000001D 8D 04 0A        lea     eax,[edx+ecx]
  00000020 8B D1           mov     edx,ecx
  00000022 8B C8           mov     ecx,eax
  00000024 3B F7           cmp     esi,edi
  00000026 76 F4           jbe     0000001C
  00000028 8B 3C 24        mov     edi,dword ptr [esp]
  0000002B 59              pop     ecx
  0000002C 5E              pop     esi
  0000002D C3              ret
  0000002E 66 90           xchg    ax,ax
```

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

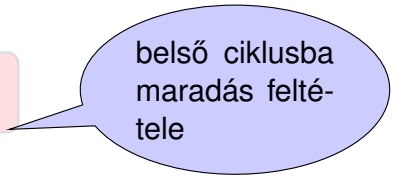
Dupla ciklus

```
_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
```



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 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 pt
  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
```

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
```

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;
}
```