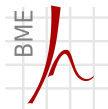


# Programok működése dióhéjban

## Intel x68



Híradástechnikai Tanszék

Izsó Tamás

2015. október 30.

# Section 1

## Fordítás

# authenticate.c

```
1 // cl /GS- authenticate.c
2 #include <stdio.h>
3
4 int verify_password(char* username, char* password) {
5     /* Csak a próba kedvéért! */
6     if( strcmp(username, "tomi") == 0 &&
7         strcmp(password, "csillag") == 0 ) return 1;
8     return 0;
9 }
10
11 void log(const char* fmt, const char* msg) {
12     fprintf(stderr, fmt, msg );
13 }
```

# authenticate.c

```
15 int authenticate(char *username, char *password) {
16     int authenticated;
17     char buffer[1024];
18     authenticated = verify_password(username, password);
19     if(authenticated == 0)    {
20         sprintf(buffer,
21             "password is incorrect for user %s\n",
22             username);
23         log("%s", buffer);
24     }
25     return authenticated;
26 }
27
28 main(int argc, char* argv[] ) {
29     int res;
30     if( argc < 3 ) return 100;
31     res = authenticate( argv[1], argv[2] );
32     printf("Password : %s\n", res ? "OK" : "No!" );
33     return 0;
34 }
```

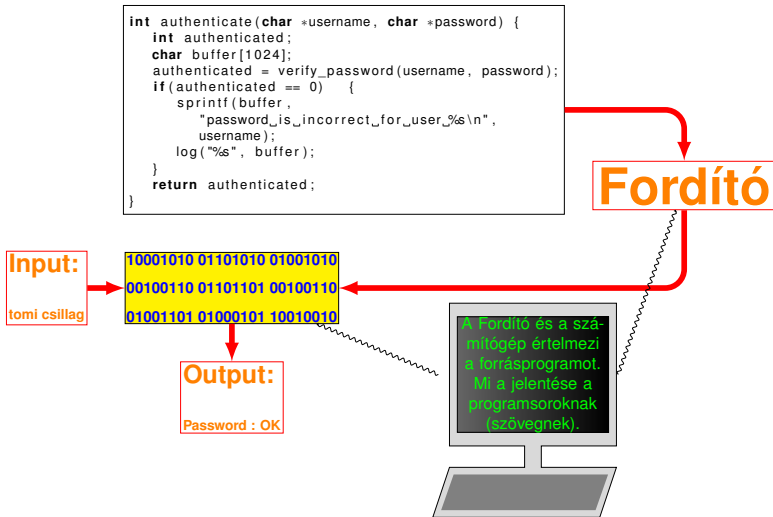
# Kérdés

**Az előző program biztonságos ?**



**Attól függ, hogy  
miként fut le a program.**

# Program értelmezése



# Bináris futtatható program



# Támadás megtervezése

A támadó célja a számítógép felett átvenni a felügyeletet.

Megvalósítás:

- 1** A támadást végrehajtó kód bejuttatása a számítógépre.  
Kivitelezés:
  - A támadó input ablakba beírja a gépi kódú programot.
  - A támadó preparált film, kép, mp3 fájl segítségével juttatja be a kártékony programot.
  - stb.
- 2** A támadó eltéríti a program futását, és ráadja a vezérlést a bejuttatott kódra.
- 3** A kártékony program célja elérése érdekében használja a feltört számítógép erőforrásához.



# Kulcskérdések

- Rendszeresen frissített operációs rendszer esetén a bejuttatott kód csak az operációs rendszerhíváson keresztül érheti el az erőforrásokat, például a fájlokat, hálózatot, stb.<sup>1</sup> Ehhez ismerni kell a DLL (.so) fájl formátumát.
- A támadónak ismerni kell a processzor működését és a gépi utasításokat. Ismerni kell, hogy hol vannak azok a pontok, ahol az utasítás számláló regiszter kapcsolatba lép a memóriával, vagy indirekten módon befolyásolható az értéke.

---

<sup>1</sup>Különböen ismernie kellene még hivatalosan nem publikált úgynevezett *Zero-Day Vulnerability*-t.

# Mit kell tudni a támadás kivitelezéséhez

- a processzor felépítését és működését
  - bájt-sorrend;
  - regiszterek;
  - utasítások;
- operációs rendszert
  - végrehajtható <sup>2</sup> fájl formátumát<sup>3</sup>;
  - dinamikusan linkelt függvények hívási mechanizmusa;
  - rendszer szintű kivételkezelés (SEH);
  - fontosabb operációs rendszerhívások;
- fordító által generált kód működését
  - stack működése;
  - heap szerkezete;
  - függvényhívás megvalósítása;
  - virtuális függvények megvalósítása;
  - programozási nyelvben megvalósított kivételkezelés (exception handler);

---

<sup>2</sup>Windows esetén PE (Portable Executable), Linuxon ELF formátum

<sup>3</sup>A DLL/shared object is ilyen formátumú fájl.

# Trojan.Cryptowall

*„Nearly all men can stand adversity, but if you want to test a man's character, give him power.”*

— Abraham Lincoln

Terjedése:

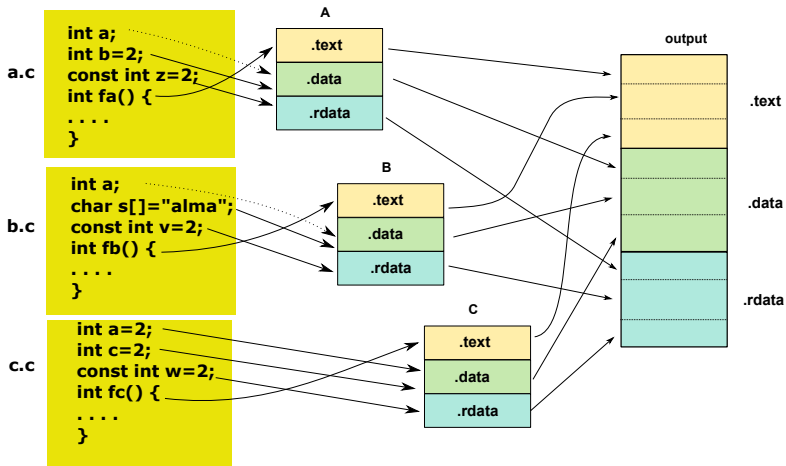
- levélben;
- Exploit Kits
  - Oracle Java SE Remote Java Runtime Environment Code Execution Vulnerability (CVE-2012-0507)
  - Adobe Flash Player Buffer Overflow Vulnerability (CVE-2014-0515)
  - Adobe Flash Player and AIR Unspecified Heap Based Buffer Overflow Vulnerability (CVE-2014-0556)
- Malware utján.

Hatás: Titkosítja a fájlokat 2048-bit-es RSA algoritmussal.

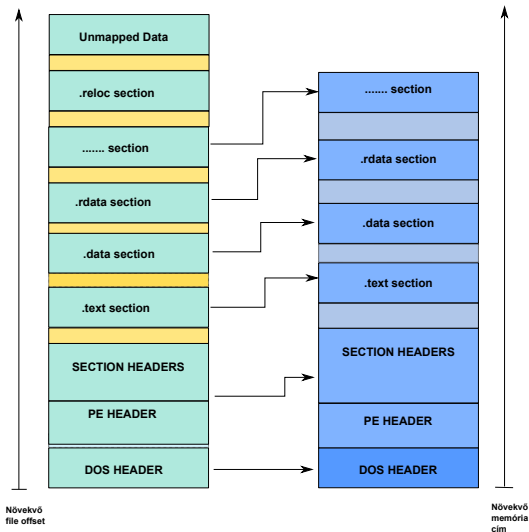
## Section 2

# PE/COFF

# Program szegmensek linkelése



# Exe file leképzése a memóriába



# Portable Executable fájl

- DOS operációs rendszerrel kompatibilis, figyelmeztető üzenetet ad, ha nem futtatható DOS alatt a program. A PE fájl eleje megegyezik a MZ végrehajtható fájlformátummal.
- DLL hívást támogatja. Az exe és a DLL azonos szerkezettű. A DLL fájlokat sokszor más kiterjesztéssel használják, pl. OCX , CPL
- Alpha, MIPS és .NET MSIL végrehajtható fájlformátuma is egyben.
- 64 bites programok hasonló formában vannak tárolva, egyes helyeken a 32 bitet 64 bites adatok váltották fel. PE32+ elnevezést kapta.

# Relative Virtual Address (RVA)

A futtatható fájlt a loader nem másolja be a memóriába, hanem az egyes szekciókat memory mapped file-ként kezeli. A fájlban tárolt szekciók 512-vel (0x200) osztható byte határon kezdődnek, míg a memóriában új lapra kerülnek, ahol egy lap 4Kbyte (vagy 8 Kbyte). Ezért a fájl adott tartalma a fájl elejétől más távolságra van, mint amikor ugyanezt az adatot a memóriába a betöltési ponthoz viszonyítjuk.

## RVA kiszámítása

A memóriában lévő címek a betöltés helyétől függetlenül vannak megadva. Ha egy cím a 0x4010DA címen szerepel és a program képe a 0x400000 címtől kezdve lett betöltve, akkor a relatív virtuális cím (RVA):

$$\text{RVA} = 0x4010DA - 0x400000 = 0x10DA$$



# Exe file dumpja

00000000:	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....DOS HEADER	00000440:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000010:	88 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00		00000450:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000020:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000460:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000030:	00 00 00 00 00 00 00 00 00 00 00 00 C0 00 00 00		00000470:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000040:	06 1F 8A 0E 00 8A 09 00 21 88 03 4C 00 23 5A 68	is.program.canno	00000480:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000050:	69 73 20 70 72 6F 67 72 61 60 20 63 61 4E 5F 20	t.be.run.in.dos.	00000490:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000060:	74 20 62 65 20 72 75 6E 20 69 6E 20 40 6F 6E 6D	mode	000004A0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000070:	60 6F 64 65 2E 00 0A 24 00 00 00 00 00 00 00 00		000004B0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000080:	09 76 0C 80 04 18 8F 8D 04 18 8F 8D 04 18 8F	rich	000004C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000090:	84 AC 8B 8F 8E D4 18 8F 8D 04 19 8F 8D 04 18 8F		000004D0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000000A0:	84 AC 9B 8F 8C D4 18 8F 84 AC 89 8F 8C D4 18 8F		000004E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000000B0:	72 69 63 68 8D 04 18 8F 00 00 00 00 00 00 00 00	PE.....P	000004F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000000C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000500:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000000D0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000510:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000000E0:	00 02 00 00 00 00 00 00 00 10 00 00 00 00 00 00		00000520:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000000F0:	00 20 00 00 00 00 40 00 00 10 00 00 00 02 00 00		00000530:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000100:	05 00 00 00 00 00 00 00 05 00 00 00 00 00 00 00		00000540:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000110:	00 30 00 00 00 04 00 00 00 00 00 03 00 00 84	OPTIONAL HEADER	00000550:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000120:	00 00 10 00 00 10 00 00 00 10 00 00 10 00 00 00		00000560:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000130:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000570:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000140:	0C 20 00 00 28 00 00 00 00 00 00 00 00 00 00 00	IMAGE DATA DIRECTORY	00000580:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000150:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000590:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000160:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		000005A0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000170:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		000005B0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000180:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		000005C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000190:	00 00 00 00 00 00 00 00 20 00 00 0C 00 00 00 00	text	000005D0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000001A0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		000005E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000001B0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		000005F0:	74 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000001C0:	7A 00 00 00 00 10 00 00 00 02 00 00 00 04 00 00	.rdata	00000600:	4C 20 00 00 40 20 00 00 00 00 00 00 34 20 00 00	
000001D0:	00 00 00 00 00 00 00 00 00 00 00 00 20 00 00 50		00000610:	00 00 00 00 00 00 00 00 00 52 20 00 00 20 00 00	R
000001E0:	7E 72 64 61 74 61 00 5C 00 00 00 00 20 00 00 00		00000620:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000001F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000630:	00 00 00 4C 20 00 00 40 20 00 00 00 00 00 00 00	Function
00000200:	00 00 00 00 40 00 00 40 00 00 00 00 00 00 00 00		00000640:	01 00 46 75 6E 63 74 69 6F 6E 00 00 00 41 64	Function
00000210:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000650:	64 00 63 61 6C 63 2E 64 6C 6C 00 00 00 00 00 00	d.calc.dll
00000220:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000660:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000230:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000670:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000240:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	SECTION TABLE	00000680:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000250:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000690:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000260:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		000006A0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000270:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		000006B0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000280:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		000006C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000290:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		000006D0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000002A0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		000006E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000002B0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		000006F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000002C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000700:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000002D0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.rdata	00000710:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000002E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	adatok	00000720:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000002F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000730:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000300:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000740:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000310:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000750:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000320:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000760:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000330:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000770:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000340:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000780:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000350:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		00000790:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000360:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		000007A0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000370:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		000007B0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000380:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		000007C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00000390:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		000007D0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000003A0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.text	000007E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000003B0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	adatok	000007F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
000003C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00				
000003D0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00				
000003E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00				
000003F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00				
00000400:	55 88 EC 51 6A 04 6A 03 FF 15 00 20 40 03 83 C4	U. Qj. j			
00000410:	08 89 45 FC 88 45 FC 50 FF 15 04 20 40 03 84 C4	..E. E. P			
00000420:	04 88 01 00 00 88 83 00 C3 00 00 00 00 00 00 00				
00000430:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00				

# DOS HEADER tartalma

- Két érdemi részt tartalmaz:
  - 1 e\_magic értéke "MZ" (Mark Zbikowski)
  - 2 e\_lfanew file offset, a PE header-re mutat.
- DOS\_HEADER után kis DOS program következnek.
- A PE rész közvetlenül ez után, 8-cal osztható címre igazítva található.

# DOS header dumpja

00000000:	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	MZ.....	DOS HEADER
00000010:	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00	.....	
00000020:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	
00000030:	00 00 00 00 00 00 00 00 00 00 00 00 C0 00 00 00	.....	
00000040:	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	.....L.Th	
00000050:	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F	is.program.canno	
00000060:	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	t.be.run.in.DOS.	
00000070:	6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	mode.....	
00000080:	C9 B5 76 DC 8D D4 18 8F 8D D4 18 8F 8D D4 18 8F	..V.....	NT HEADER
00000090:	84 AC 8B 8F 8F D4 18 8F 8D D4 19 8F 8F D4 18 8F	.....	
000000A0:	84 AC 9B 8F 8C D4 18 8F 84 AC 89 8F 8C D4 18 8F	Rich.....	
000000B0:	52 69 63 68 8D D4 18 8F 00 00 00 00 00 00 00 00	PE.L.....P....	
000000C0:	50 45 00 00 4C 01 02 00 9C 0B A8 50 00 00 00 00	.....	
000000D0:	00 00 00 00 E0 00 03 01 0B 01 09 00 00 02 00 00	.....	
000000E0:	00 02 00 00 00 00 00 00 00 10 00 00 00 10 00 00	.....	

# NT HEADER

```
typedef struct _IMAGE_NT_HEADERS {  
    DWORD Signature;    // "PE"  
    IMAGE_FILE_HEADER FileHeader;  
    IMAGE_OPTIONAL_HEADER32 OptionalHeader;  
} IMAGE_NT_HEADERS32, *PIMAGE_NT_HEADERS32;
```

# NT header dumpja

```

00000000: 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ.....
00000010: B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .....
00000020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 C0 00 00 00 .....
00000040: 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 .....L..Th
00000050: 69 73 20 70 70 6F 67 72 61 6D 20 63 61 65 65 65 .....is.program.canno
00000060: 74 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....t.be.run.in.DOS.
00000070: 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 .....mode.....
00000080: C9 8B 8F 8E D4 18 8F 8D D4 18 8F 8D D4 18 8F .....v.....
00000090: 84 AC 8B 8F 8E D4 18 8F 8D D4 18 8F 8D D4 18 8F .....
000000A0: 84 AC 9B 8F 8C D4 18 8F 8D D4 18 8F 8D D4 18 8F .....
000000B0: 52 69 63 68 8D D4 18 8F 00 00 00 00 00 00 00 00 .....Rich..
000000C0: 50 45 00 00 4C 01 02 00 9C 0B A8 50 00 00 00 00 .....PE..L.....P....
000000D0: 00 00 00 00 E0 00 03 01 0B 01 09 00 00 02 00 00 .....
000000E0: 00 02 00 00 00 00 00 00 00 10 00 00 00 10 00 00 .....
    
```

**DOS HEADER**

**NT HEADER**

Intel 386 or later processors and compatible processors

signature

2 section időbélyeg

szimbólumok száma

opcionális header mérete

szimbólumtáblára mutató pointer

IMAGE\_FILE\_EXECUTABLE\_IMAGE  
IMAGE\_FILE\_RELOCS\_STRIPPED

# Optional header dumpja

The diagram shows a memory dump of the PE Optional Header. Annotations include:

- .data mérete**: points to the first four bytes of the first row (50 45 00 00).
- .bbs mérete**: points to the next four bytes (01 02 00 00).
- PE32**: points to the byte 09.
- program belépési pontja RVA**: points to the byte 0B.
- .text mérete**: points to the last four bytes of the first row (00 00 00 00).
- Kód címe RVA**: points to the byte 00 in the second row.

Callouts on the right side identify sections:

- NT HEADER**: points to the first row.
- IMAGE OPTIONAL HEADER**: points to the first row.
- IMAGE DATA DIRECTORY**: points to the first row.

A table below the dump explains the fields in the first row:

Field	Value
adat címe RVA	50 45 00 00
image betöltési címe	01 02 00 00
címhatárra igazítás	09 0B A8 50
file offset határra igazítás	00 00 00 00

```

000000C0: 50 45 00 00 01 02 00 00 09 0B A8 50 00 00 00 00 PE...L.....P...
000000D0: 00 00 00 00 E0 00 03 01 0B 01 09 00 00 02 00 00
000000E0: 00 02 00 00 00 00 00 00 00 10 00 00 00 10 00 00
000000F0: 00 20 00 00 00 00 40 00 00 10 00 00 00 02 00 00
00000100: 05 00 00 00 00 00 00 00 05 00 00 00 00 00 00 00
00000110: 00 30 00 00 00 04 00 00 00 00 00 00 00 03 00 84
00000120:
00000130:
00000140: 0C 20 00 00 28 00 00 00 00 00 00 00 00 00 00 00
00000150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000160: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000190: 00 00 00 00 00 00 00 00 00 20 00 00 0C 00 00 00
000001A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001B0: 00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00
    
```

# SECTION TABLE

A section táblák a fordítás során keletkezett kód, adat, konstans adat stb. területek információit tartalmazza. Egy section adatai:

- neve;
- helye a file-ban (blokkhatáron kezdődnek);
- section helye a memóriában (laphatáron kezdődnek);
- memóriavédelmi beállítások (írható, olvasható, futtatható, stb.);
- méret;
- relokációs adatok száma és azok helye a file-ban;
- debug-os fordítás esetén kapcsolat a forrásszöveggel (sorok száma);

# SECTION TABLE dumpja

opcionális header mérete + opcionális header kezdete

section tábla kezdete

NT HEADER

IMAGE OPTIONAL HEADER

IMAGE DATA DIRECTORY

data

SECTION TABLE

.rdata adatok

.text adatok

```

000000c0: 50 45 00 00 4c 01 02 00 9c 08 a8 50 00 00 00 00 PE...L...P...
000000d0: 00 00 00 00 ed 00 03 01 0b 01 09 00 00 02 00 00
000000e0: 00 02 00 00 00 00 00 00 00 10 00 00 00 10 00 00
000000f0: 00 20 00 00 00 00 40 00 00 10 00 00 00 02 00 00
00000100: 05 00 00 00 00 00 00 00 05 00 00 00 00 00 00 00
00000110: 00 30 00 00 04 00 00 00 00 00 00 03 00 00 84
00000120: 00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00
00000130: 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00
00000140: 0c 20 00 00 28 00 00 00 00 00 00 00 00 00 00 00
00000150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000160: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000190: 00 00 00 00 00 00 00 00 20 00 00 0c 00 00 00 00
000001a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001b0: 00 00 00 00 00 00 00 00 0e 74 65 78 74 00 00 00
000001c0: 2a 00 00 00 00 10 00 00 00 02 00 00 00 04 00 00
000001d0: 00 00 00 00 00 00 00 00 00 00 00 00 20 00 00 60
000001e0: 2e 72 64 61 74 61 00 00 5c 00 00 00 00 20 00 00
000001f0: 00 02 00 00 06 00 00 00 00 00 00 00 00 00 00 00
00000200: 00 00 00 40 00 00 40 00 00 00 00 00 00 00 00 00
00000210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000220: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000230: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000240: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000250: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000260: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000270: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000280: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000290: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000300: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000310: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000320: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000330: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000340: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000350: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000360: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000370: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000380: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000390: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000003f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000400: 55 88 ec 51 6a 04 6a 03 ff 15 00 20 40 00 83 c4 U...q...j
00000410: 08 89 45 fc 88 45 fc 50 ff 15 04 20 40 00 83 c4 ...E...P
00000420: 04 88 01 00 00 00 88 e5 5d c3 00 00 00 00 00 00
00000430: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    
```



# IMAGE DATA DIRECTORY

IMAGE DATA directory megadja a program futásánál speciális szerepet betöltő adatok. Ezeknek a helye valamelyik section-ban van.

- 16 darab bejegyzés van, amiből 15-öt használnak.
- Egy bejegyzés az adat helyét és méretét tartalmazza.

# IMAGE DATA 16 bejegyzése

Illusztráció!

0	Export Table	dll .edata Section
1	Import Table	dll .idata
2	Resource Table	.rsrc Section
3	Exception Table	.pdata Section
4	Certificate Table	
5	Base Relocation Table	.reloc Section
6	Debug	.debug Section
7	Architecture Specific Data	reserved 0
8	Global Ptr	
9	TLS Directory	
10	Load Configuration Directory	
11	Bound Import	
12	IAT	import address table
13	Delay Load Import Descriptors	
14	CLR Runtime descriptor	
15	reserved	0

# IMPORT DIRECTORY DATA dumpja

```

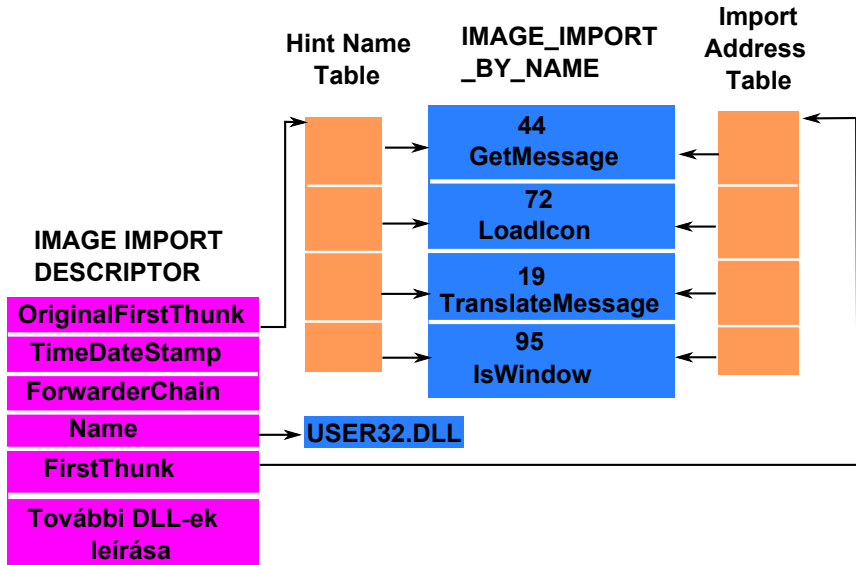
000000c0: 50 45 00 00 4c 01 02 00 9c 0b a8 50 00 00 00 00 PE...L...P...
000000d0: 00 00 00 00 e0 00 03 01 08 01 09 00 00 02 00 00
000000e0: 00 02 00 00 00 00 00 00 10 00 00 00 10 00 00
000000f0: 00 20 00 00 00 40 00 00 10 00 00 00 02 00 00
00000100: 05 00 00 00 00 00 05 00 00 00 00 00 00 00 00
00000110: 00 30 00 00 00 04 00 00 00 00 03 00 00 84
00000120: 00 00 10 00 00 10 00 00 00 10 00 00 10 00 00
00000130: 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00
00000140: 0c 20 00 00 28 00 00 00 00 00 00 00 00 00 00
00000150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000160: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001c0: 2a 00 00 00 10 00 00 00 02 00 00 00 04 00 00
000001d0: 00 00 00 00 00 00 00 00 00 00 20 00 00 60
000001e0: 7e 72 64 61 74 61 00 00 5c 00 00 00 20 00 00
000001f0: 00 02 00 00 06 00 00 00 00 00 00 00 00 00 00
00000200: 00 00 00 00 40 00 00 00 40 00 00 00 00 00 00
00000210: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000220: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000230: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000240: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000250: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000260: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000270: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000280: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000290: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000002f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000300: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000310: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000320: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000330: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000340: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000500: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000510: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000520: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000530: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000540: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000550: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000560: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000570: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000580: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000590: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000005a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000005b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000005c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000005d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000005e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000005f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000600: 4c 20 00 00 40 20 00 00 00 00 00 00 34 20 00 00
00000610: 00 00 00 00 00 00 00 00 00 00 52 20 00 00 20 00 00
00000620: 00 00 00 00 00 00 00 00 00 00 50 00 00 00 00 00
00000630: 00 00 00 00 4c 20 00 00 40 20 00 00 00 00 00 00
00000640: 01 00 46 75 6e 63 74 69 6f 6e 00 00 00 41 64
00000650: 64 00 63 61 6c 63 2e 54 6c 6c 00 00 00 00 00 00
00000660: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000670: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000680: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000690: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000006a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000006b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000006c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000006d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000006e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000006f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000700: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000710: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000720: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000730: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000740: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000750: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000760: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000770: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000780: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    
```

Annotations in the image:

- NT HEADER**: Points to the PE...L...P... header.
- IMAGE OPTIONAL HEADER**: Points to the IMAGE...L...P... header.
- IMAGE DATA DIRECTORY**: Points to the IMAGE...D...Y... directory.
- SECTION TABLE**: Points to the .text, .data, and .rdata sections.
- IMPORT SECTION**: Points to the IMPORT SECTION RVA and MÉRLET fields.
- SECTION TABLE**: Points to the .text, .data, and .rdata sections.
- .rdata adatok**: Points to the .rdata section.
- adatok méret**: Points to the Size of Image field.
- mérete a fájlban**: Points to the RVA field.
- helye a fájlban**: Points to the RVA field.

Látható, hogy a konstans adatszegmensben kapott helyet.

# IMPORT TABLE



# IMPORT TABLE

00000560:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000570:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000580:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000590:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000005A0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000005B0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000005C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000005D0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000005E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000005F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000600:	4C 20 00 00 40 20 00 00 00 00 00 00 34 20 00 00	L.....
00000610:	00 00 00 00 00 00 00 00 52 20 00 00 00 20 00 00	.....R.....
00000620:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000630:	00 00 00 00 4C 20 00 00 40 20 00 00 00 00 00 00	.....L.....
00000640:	01 00 46 75 6E 63 74 69 6F 6E 00 00 00 00 41 64	..Function...Ad
00000650:	64 00 63 61 6C 63 2E 64 6E 15C 00 00 00 00 00 00	d.calc.dll.....
00000660:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
Ordinal 0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000670:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000680:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000690:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000006A0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000006B0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000006C0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000006D0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000006E0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000006F0:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000700:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....

**.rdata  
adatok**

**OriginalFirstThunk**

**Name (RVA)**

**Name (RVA)**

**FirstThunk**

**Importált  
szimbólum  
(függvény)  
neve**

# IMPORT TABLE – program indítása után a memóriában

```

00402000  00 10 00 10|30 10 00 10|00 00 00 00|34 20 00 00| ....0.....4 ..
00402010  00 00 00 00|00 00 00 00|52 20 00 00|00 20 00 00| .....R ... ..
00402020  00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00| .....
00402030  00 00 00 00|4C 20 00 00|40 20 00 00|00 00 00 00| ....L ... .....
00402040  01 00 46 75|6E 63 74 69|6F 6E 00 00|00 00 41 64| ..Function....Ad
00402050  64 00 63 61|6C 63 2E 64|6C 6C 00 00|00 00 00 00| d.calc.dll.....

```

# DLL-ben lévő függvény hívása

Illusztráció!

00401000	55	<b>PUSH EBP</b>
00401001	8BEC	<b>MOV EBP,ESP</b>
00401003	51	<b>PUSH ECX</b>
00401004	6A 04	<b>PUSH 4</b>
00401006	6A 03	<b>PUSH 3</b>
00401008	FF15 00204000	<b>CALL DWORD PTR DS:[ &lt;&amp;calc.Add &gt;]</b>
0040100E	83C4 08	<b>ADD ESP,8</b>
00401011	8945 FC	<b>MOV DWORD PTR SS:[ LOCAL.1 ],EAX</b>
00401014	8B45 FC	<b>MOV EAX,DWORD PTR SS:[ LOCAL.1]</b>
00401017	50	<b>PUSH EAX</b>
00401018	FF15 04204000	<b>CALL DWORD PTR DS:[ &lt;&amp;calc.Function &gt;]</b>
0040101E	83C4 04	<b>ADD ESP,4</b>
00401021	B8 01000000	<b>MOV EAX,1</b>
00401026	8BE5	<b>MOV ESP,EBP</b>
00401028	5D	<b>POP EBP</b>
00401029	C3	<b>RETN</b>

## DLL tartalma

Illusztráció!

```

10001000 55          PUSH EBP
10001001 8BEC        MOV EBP,ESP
10001003 8B45 08     MOV EAX,DWORD PTR SS:[EBP+8]
10001006 0345 0C     ADD EAX,DWORD PTR SS:[EBP+0C]
10001009 5D          POP EBP
1000100A C3          RETN

10001010 55          PUSH EBP
10001011 8BEC        MOV EBP,ESP
10001013 8B45 08     MOV EAX,DWORD PTR SS:[EBP+8]
10001016 2B45 0C     SUB EAX,DWORD PTR SS:[EBP+0C]
10001019 5D          POP EBP
1000101A C3          RETN

10001020 55          PUSH EBP
10001021 8BEC        MOV EBP,ESP
10001023 8B45 08     MOV EAX,DWORD PTR SS:[EBP+8]
10001026 0FAF45 0C  IMUL EAX,DWORD PTR SS:[EBP+0C]
1000102A 5D          POP EBP
1000102B C3          RETN

10001030 55          PUSH EBP
10001031 8BEC        MOV EBP,ESP
10001033 8B45 08     MOV EAX,DWORD PTR SS:[EBP+8]
10001036 50          PUSH EAX
10001037 68 00C00010 PUSH OFFSET 1000C000 ; ASCII "Printf from DLL %d "
1000103C E8 7F000000 CALL 100010C0
10001041 83C4 08     ADD ESP,8
10001044 5D          POP EBP
10001045 C3          RETN

```



# Importált függvények címének feloldása

- A program a dll-ben lévő **Add()** függvényt a 00401008 címen a **CALL** [00402000] utasítással hívja meg. Az indirekten hívott függvény címét az Import Address Table (IAT) tárolja.
- A fordító a `__declspec(dllimport) int Add(int,int)`; függvény *tárolási osztály módosító* hatására generál optimálisabb kódot.
- A loadernek a betöltésnél csak az IAT-t kell megváltoztatni.
- A függvényeket nemcsak név, hanem sorszám (ordinal number) alapján is meg lehet hívni.

## Dll Export függvények feltérképezése

- Program kezdetének a megállapítása.
- DOS header-ben lévő érték alapján a NT header kezdetének az elérése.
- Opcional header-ből a export tábla kezdő és végcímének a kiszámítása (RVA).
- Függvénynevek, függvényekhez tartozó sorszámok és címek kiolvasása.

Ha a memóriába bejuttatott program nem tudja a rendszerhívások memória címét, akkor így tudja azokat kikeresni.

# Irodalom

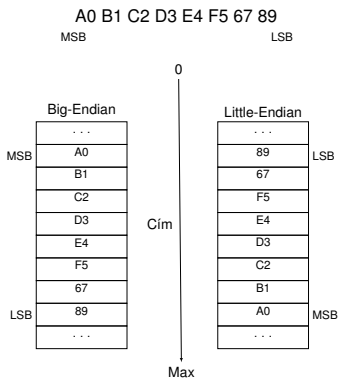
- 1** Matt Pietrek An In-Depth Look into the Win32 Portable Executable File Format <http://msdn.microsoft.com/en-us/magazine/bb985992.aspx>
- 2** Matt Pietrek Peering Inside the PE: A Tour of the Win32 Portable Executable File Format <http://msdn.microsoft.com/en-us/magazine/ms809762.aspx>
- 3** Microsoft PE and COFF Specification. <http://msdn.microsoft.com/en-us/library/windows/hardware/gg463119.aspx>

## Section 3

### X86 utasítás

# Byte-sorrend

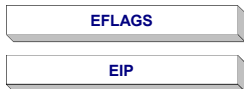
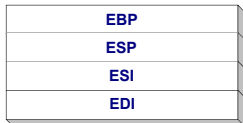
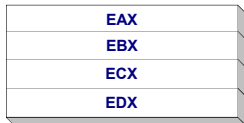
A byte-sorrend megadja hogy a 2, 4, 8 byte-os adatok milyen sorrendben helyezkednek el a memóriában. A byte-on belül a bitek sorrendje változatlan.



# X86 regiszterek

- 8 darab 32 bites regiszter;
- 6 darab szegmens regiszter;
- statusz regiszter EFLAGS
- utasításszámláló EIP

## 32 bites általános célú regiszterek



## szegmens regiszterek



# Regiszterek speciális szerepe egyes utasításokban

Speciális de nem kizárólagos felhasználása a regisztereknek.

**EAX** akkumulátor regiszter, szorzáshoz osztáshoz;

**ECX** counter regiszter (ciklusszámlálásra);

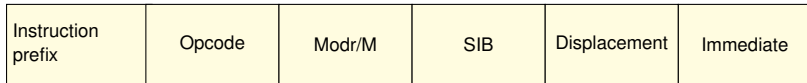
**ESP** stack pointer;

**ESI** string műveletek esetén a forrás memóriaterületet indexeli;

**EDI** string műveletek esetén a cél memóriaterületet indexeli;

**EBP** bázis pointer a stack kezeléshez.

# x86 utasítás formátum



Opcionálisan adható 4 csoportba sorolható prefix

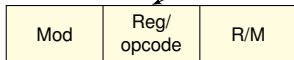
Műveleti kód  
1-,2-,3 byte

1 byte, ha szükséges

1 byte, ha szükséges

Displacement (eltolás) 1-,2-,4 byte

Immediate (közvetlen) adat 1-,2-,4 byte





# Példa X86 utasításokra 1.

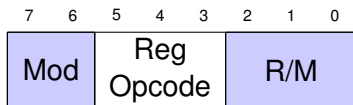
**add eax , ecx**

$eax = eax + ecx;$

**sub esi , 10**

$esi = esi - 10;$

# ModR/M



A *ModR/M* mező a műveletben résztvevő regisztereket vagy memória címzés típusát azonosítja.

- A *mod* és az *r/m* mező összevonva 5 bites, azaz 32 lehetséges értéket vehet fel. Ebből 8 érték regisztert címez, 24 pedig a címzési módot adja meg.
- A *reg/opcode* vagy regiszter cím, vagy 3 bit még hozzáadódik az utasításhoz, ami plusz információt hordoz.
- Az *r/m* mező a regiszter típusú operandust adja meg, vagy a *mod* mezővel kombinálva a címzési módot írja le.

Némely *ModR/M* értékhez még egy byte címzési információt kell elhelyezni, ez a *SIB* byte.

# SIB mező



- skálafaktor
- index
- bázis

Néhány címzési módnál a *ModR/M* vagy ha van *SIB* byte után találjuk az eltolás (displacement) értéket. Ez az érték 1, 2, vagy 4 byte-os lehet.

# Címszámítás a SIB alapján

$$\begin{array}{c} \textit{base} \\ \left[ \begin{array}{c} \textit{EAX} \\ \textit{EBX} \\ \textit{ECX} \\ \textit{EDX} \\ \textit{ESP} \\ \textit{EBP} \\ \textit{ESI} \\ \textit{EDI} \end{array} \right] \end{array} + \begin{array}{c} \textit{index} \\ \left( \begin{array}{c} \textit{EAX} \\ \textit{EBX} \\ \textit{ECX} \\ \textit{EDX} \\ \textit{EBP} \\ \textit{ESI} \\ \textit{EDI} \end{array} \right) \end{array} * \begin{array}{c} \textit{scale} \\ \left( \begin{array}{c} 1 \\ 2 \\ 4 \\ 8 \end{array} \right) \end{array} + \begin{array}{c} \textit{displacement} \\ \left[ \begin{array}{c} \textit{nincs} \\ 8 - \textit{bit} \\ 16 - \textit{bit} \\ 32 - \textit{bit} \end{array} \right] \end{array}$$

$$\textit{offset} = \textit{base} + (\textit{index} * \textit{scale}) + \textit{displacement}$$

# ModR/M számítása

Illusztráció!

000	AL	AX	EAX	MM0	XMM0
001	CL	CX	ECX	MM1	XMM1
010	DL	DX	EDX	MM2	XMM2
011	BL	BX	EBX	MM3	XMM3
100	AH	SP	ESP	MM4	XMM4
101	CH	BP	EBP	MM5	XMM5
110	DH	SI	ESI	MM6	XMM6
111	BH	DI	EDI	MM7	XMM7

regiszterek számozása

	MOD	11			
	R/M			000	
	REG		001		
ModR/M		11	001	000	= C8H
			ECX	EAX	

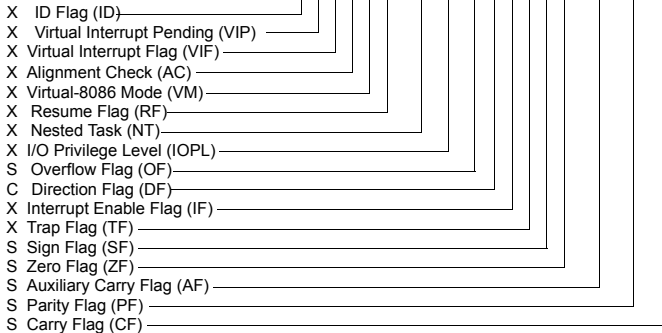
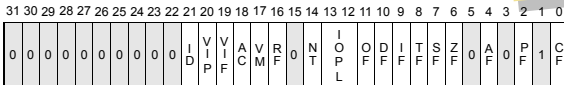
## Példa X86 utasításokra 2.

Memória címzési módok:

<b>mov esi , [10128CF4]</b>	<b>esi = *(unsigned int *) 0x10128CF4;</b>
<b>mov edx , [ecx+0A8h]</b>	<b>esi = *(unsigned int *) (ecx+0x0A8);</b>
<b>mov edx , [ecx+eax*4]</b>	<b>esi = *(unsigned int *) (ecx+eax*4)</b>
<b>mov edx , [ebp+esi*4-74h]</b>	<b>edx = *(unsigned int *) (ebp+esi*4 - 0x74);</b>

# Állapotregiszter

Illusztráció!



- S Indicates Status Flag
- C Indicates a Control Flag
- X Indicates a System Flag



Foglalt bitek

# Vezérlésátadó utasítások

Illusztráció!

Utasítás	Feltétel	Leírás
jmp	1	feltétel nélkül
je	ZF = 1	= vagy 0
jne	ZF = 0	≠
js	SF = 1	< 0
jns	SF = 0	≥ 0
jg	((SF XOR OF) OR ZF) = 0	>
jge	(SF XOR OF) = 0	≥
jl	(SF XOR OF) = 1	<
jle	((SF XOR OF) OR ZF) = 1	≤
ja	(CF OR ZF) = 0	>
jb	CF = 1	<



# Vezérlésátadó utasítások

Illusztráció!

Utasítás	Kód (bináris)	Feltétel 3-2-1 bitek	Negálás 0 bit
JZ	01110100	010	0
JNZ	01110101	010	1
JB	01110010	001	0
JNB	01110011	001	1
JG	01111111	111	1
JNG	01111110	111	0

Intel X86 utasításkészlet:

<http://ref.x86asm.net/index.html>

## Section 4

stack

# Stack feladata

Eljárás paramétereit és lokális változóinak az élettartama 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. X86 architektúrán 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]<sup>4</sup>**

---

<sup>4</sup>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
```

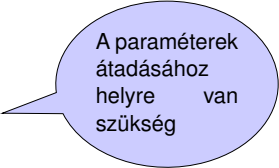
Stack megtekintés

```
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;
}
```

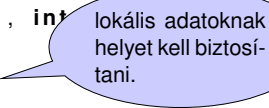


Át kell venni a paramétereket.

# C példaprogram

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

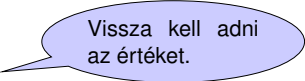
int main()
{
    int a=1;
    int b=2;
    int c=3;
    a=func(a,b,c);
    return 0;
}
```



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;
}
```



Vissza kell adni az értéket.

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

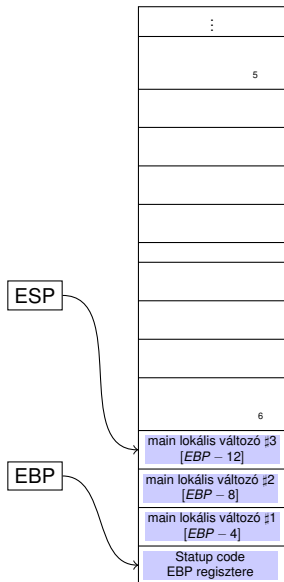
# \_\_cdecl hívási konvenció, hívó feladata

```

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

```

<sup>5</sup>csak ha szükséges



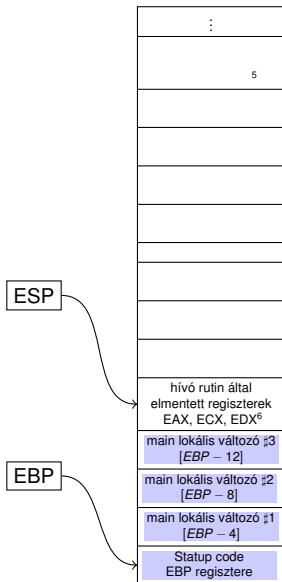
# cdecl hívási konvenció, hívó feladata

```

_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

```

<sup>5</sup>csak ha szükséges



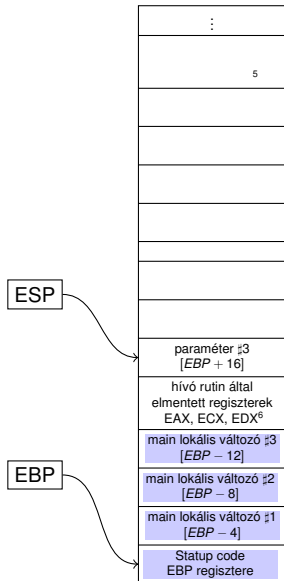
# \_\_cdecl hívási konvenció, hívó feladata

```

_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

```

<sup>5</sup>csak ha szükséges



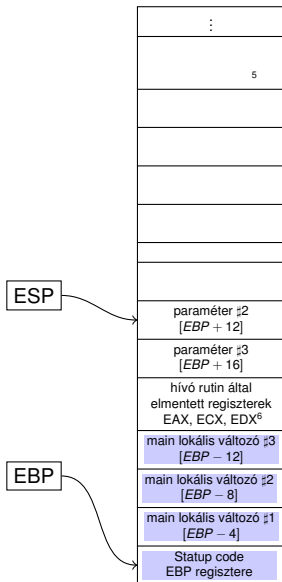
# \_\_cdecl hívási konvenció, hívó feladata

```

_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

```

<sup>5</sup>csak ha szükséges



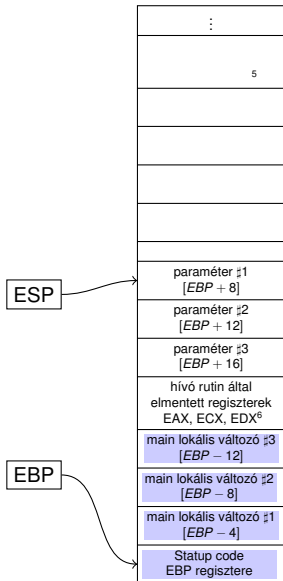
# \_\_cdecl hívási konvenció, hívó feladata

```

_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

```

<sup>5</sup>csak ha szükséges





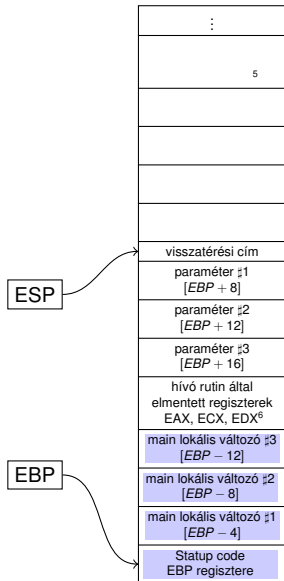
# \_\_cdecl hívási konvenció, hívó feladata

```

_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

```

<sup>5</sup>csak ha szükséges



# \_\_cdecl hívási konvenció, hívott feladata

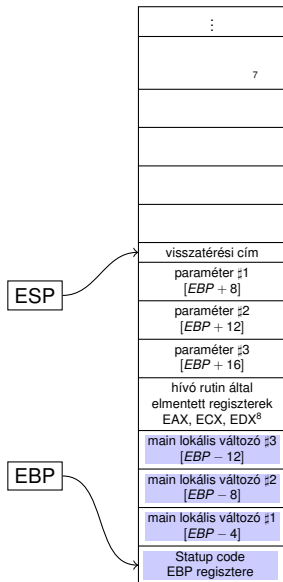
**\_func :**

```

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

```

<sup>7</sup>csak ha szükséges



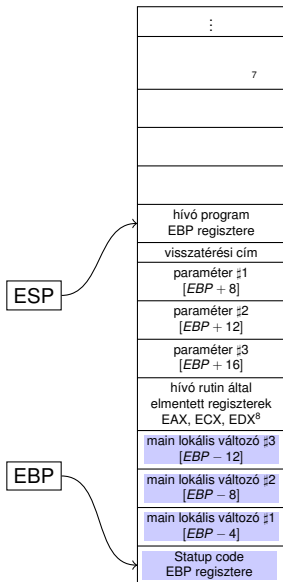
# \_\_cdecl hívási konvenció, hívott feladata

```

_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

```

<sup>7</sup>csak ha szükséges



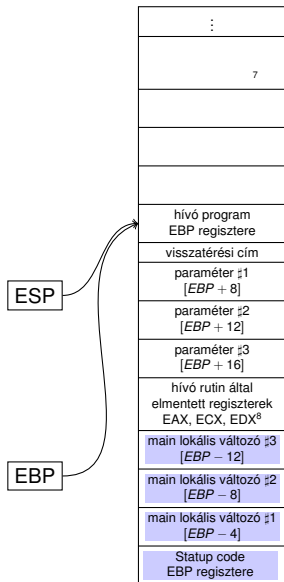
# \_\_cdecl hívási konvenció, hívott feladata

```

_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

```

<sup>7</sup>csak ha szükséges



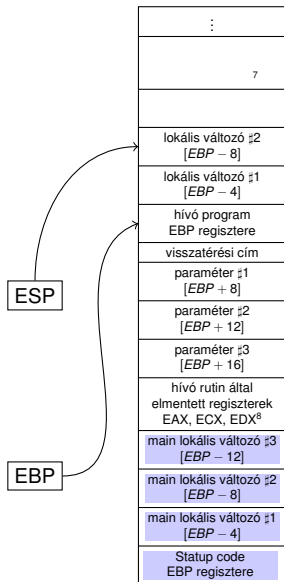
# \_\_cdecl hívási konvenció, hívott feladata

```

_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

```

<sup>7</sup>csak ha szükséges



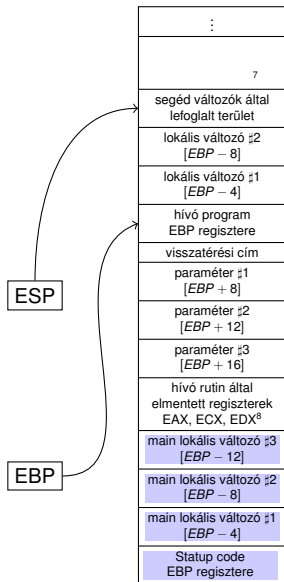
# \_\_cdecl hívási konvenció, hívott feladata

```

_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

```

<sup>7</sup>csak ha szükséges



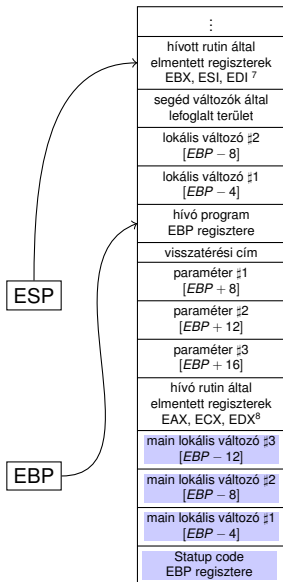
# cdecl hívási konvenció, hívott feladata

```

_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

```

<sup>7</sup>csak ha szükséges



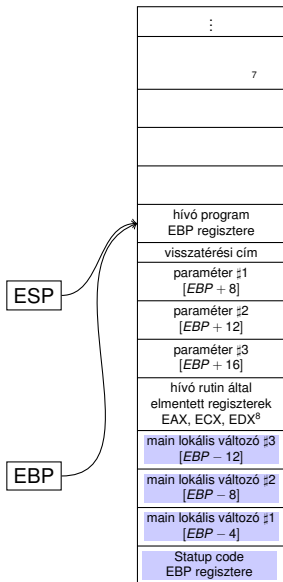
# \_\_cdecl hívási konvenció, hívott feladata

```

_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

```

<sup>7</sup>csak ha szükséges





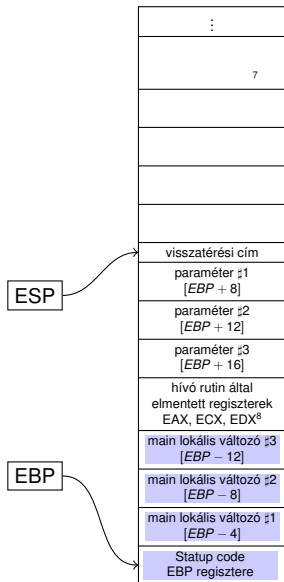
# \_\_cdecl hívási konvenció, hívott feladata

```

_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

```

<sup>7</sup>csak ha szükséges



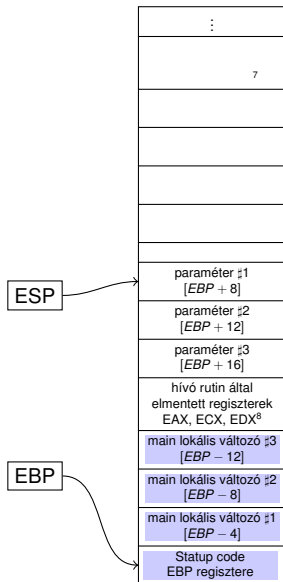
# \_\_cdecl hívási konvenció, hívott feladata

```

_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

```

<sup>7</sup>csak ha szükséges



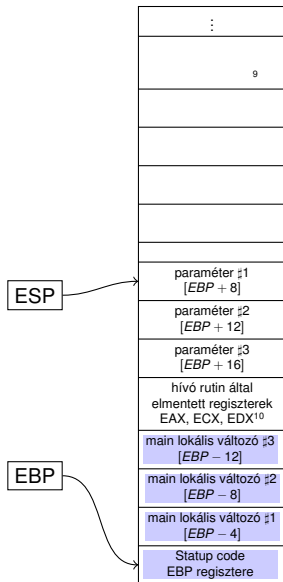
# \_\_cdecl hívási konvenció, visszatérés a hívóhoz

```

_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

```

<sup>9</sup>csak ha szükséges



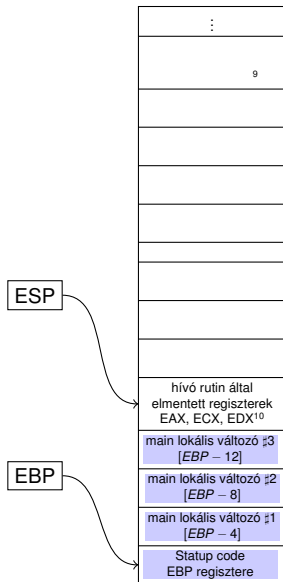
# \_\_cdecl hívási konvenció, visszatérés a hívóhoz

```

_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

```

<sup>9</sup>csak ha szükséges



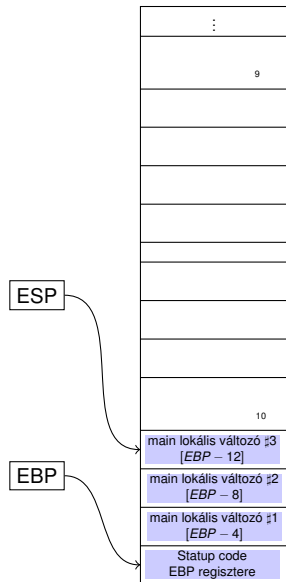
# \_\_cdecl hívási konvenció, visszatérés a hívóhoz

```

_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

```

<sup>9</sup>csak ha szükséges



# \_\_cdecl hívási konvenció 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 esetében 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.

# 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, jump eax**

# Egyéb hívási konvenciók

**stdcall** esetén a hívott rutin szabadítja fel a paramétereknek lefoglalt helyet. Windows API függvények ezt használja.

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

**fastcall** esetén egyes paraméterek a regiszterekbe kerülnek.

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



# \_\_stdcall hívási konvenció

Illusztráció!

```

_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

Illusztráció!

`@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.