리버싱

- 핵심 원리 -

한양대학교 컴퓨터공학부
2014년 1학기
임을규

IA-32 (Intel Architecture 32bit) Register

- Register
  - CPU 내부에 존재하는 다목적 저장 공간
  - 고속 데이터 처리
- IA-32 register의 종류
  - **Basic program execution registers**
  - x87 FPU registers
  - MMX registers
  - XMM registers
  - Control registers
  - Memory management registers
  - Debug registers
  - …

Basic program execution registers

- 4개 그룹
  - General purpose registers (32bit, 8개)
    - EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP
  - Segment registers (16bit, 6개)
    - CS, DS, SS, ES, FS, GS
  - Program status and control register (32bit, 1개)
    - EFLAGS
  - Instruction pointer (32bit, 1개)
    - EIP

General Purpose Registers

- 상수/주소 저장할 때 주로 사용
- 각 레지스터의 이름
  - 산술 연산 관련
    - EAX: Accumulator for operands and results data
      - 추가적으로 함수 리턴 값에 사용됨
    - EBX: pointer to data in the DS segment
    - ECX: Counter for string and loop operations
    - EDX: I/O pointer
  - 그 외의 레지스터
    - EBP: pointer to data on the stack (in the SS segment)
    - ESI: source pointer for string operations
    - EDI: destination pointer for string operations
    - ESP: stack pointer (in the SS segment)
      - 함수가 호출될 때 ESP를 저장. 함수가 리턴되기 직전에 ESP값을 되돌려 줌

Segment Registers

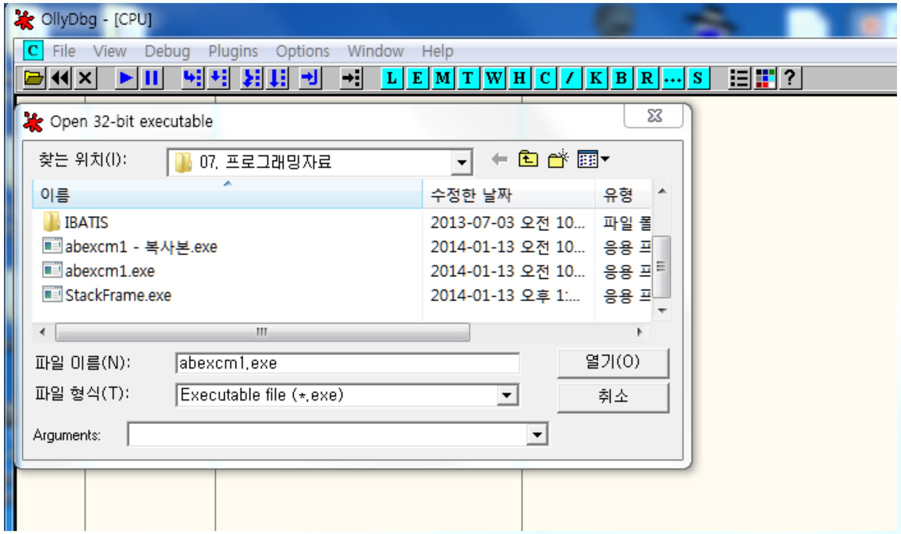
- IA-32 보호 모드에서
  - 세그먼트란 메모리를 조각내어 각 조각마다 시작 주소, 범위, 접근 권한 등을 부여하여 메모리를 보호
- Registers
  - CS: code segment
  - SS: stack segment
  - DS: data segment
  - ES: Extra(data) segment
  - FS: Data segment
  - GS: Data segment

EFLAG: flag register

Intel x86 FLAGS register			
Bit #	Abbreviation	Description	Category
FLAGS			
0	**CF**	**Carry flag**	Status
1	1	Reserved	
2	PF	Parity flag	Status
3	0	Reserved	
4	AF	Adjust flag	Status
5	0	Reserved	
6	**ZF**	**Zero flag**	Status
7	SF	Sign flag	Status
8	TF	Trap flag (single step)	Control
9	IF	Interrupt enable flag	Control
10	DF	Direction flag	Control
11	**OF**	**Overflow flag**	Status
12-13	IOPL	I/O privilege level (286+ only), always 1 on 8086 and 186	System
14	NT	Nested task flag (286+ only), always 1 on 8086 and 186	System
15	0	Reserved, always 1 on 8086 and 186, always 0 on later models	
EFLAGS			
16	RF	Resume flag (386+ only)	System
17	VM	Virtual 8086 mode flag (386+ only)	System
18	AC	Alignment check (486SX+ only)	System
19	VIF	Virtual interrupt flag (Pentium+)	System
20	VIP	Virtual interrupt pending (Pentium+)	System
21	ID	Able to use CPUID instruction (Pentium+)	System
22 ~ 32	0	Reserved	

Ch6 abex's crackme #1 분석_1

- Abex's crackme1 dbg


OllyDbg - [CPU]
C File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
Open 32-bit executable
열기(O)
취소

찾는 위치(I): 07. 프로그래밍자료

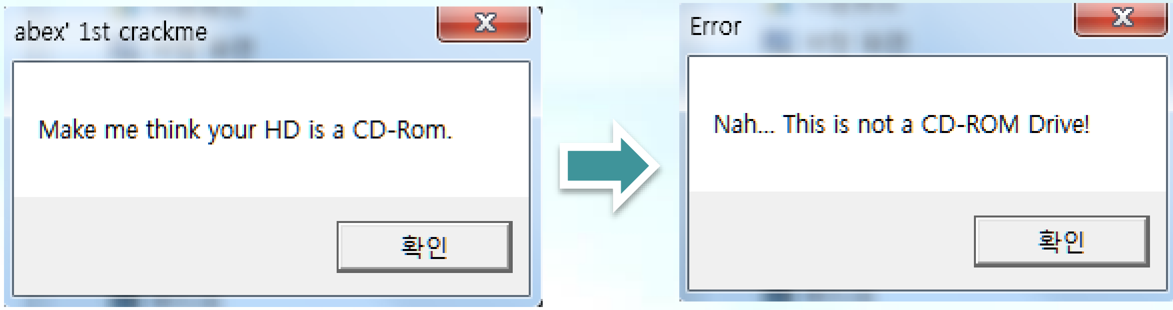
이름	수정한 날짜	유형
IBATIS	2013-07-03 오전 10...	파일 폴
abexcm1 - 복사본.exe	2014-01-13 오전 10...	응용 프
abexcm1.exe	2014-01-13 오전 10...	응용 프
StackFrame.exe	2014-01-13 오후 1:...	응용 프

파일 이름(N): abexcm1.exe

파일 형식(T): Executable file (*.exe)

Arguments:

- 실행결과



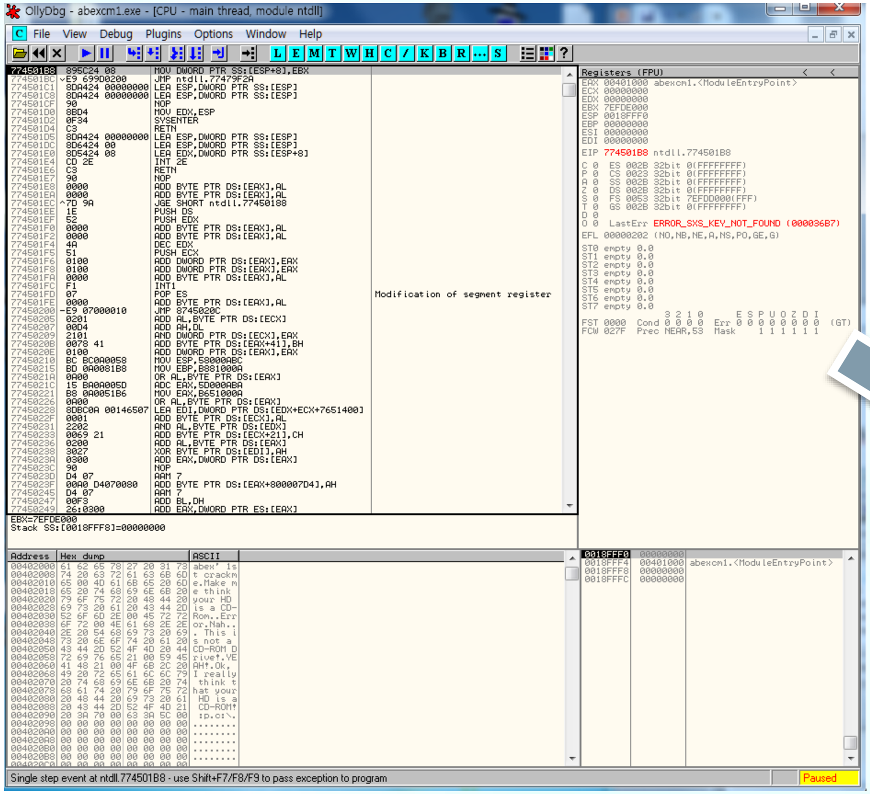
abex' 1st crackme

Make me think your HD is a CD-Rom.

Error

Nah... This is not a CD-ROM Drive!

Ch6 abex’s crackme #1 분석_2


OllyDbg - abexcm1.exe - [CPU - main thread, module ntdll]
C File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S

```
774501B8   895C24 08        MOV DWORD PTR SS:[ESP+8],EBX
774501BC  vE9 699D0200      JMP ntdll.77479F2A
774501C1   8DA424 00000000  LEA ESP,DWORD PTR SS:[ESP]
774501C8   8DA424 00000000  LEA ESP,DWORD PTR SS:[ESP]
774501CF   90               NOP
774501D0   8BD4             MOV EDX,ESP
774501D2   0F34             SYSENTER
774501D4   C3               RETN
774501D5   8DA424 00000000  LEA ESP,DWORD PTR SS:[ESP]
774501DC   8D6424 00        LEA ESP,DWORD PTR SS:[ESP]
774501E0   8D5424 08        LEA EDX,DWORD PTR SS:[ESP+8]
774501E4   CD 2E            INT 2E
774501E6   C3               RETN
774501E7   90               NOP
774501E8   0000             ADD BYTE PTR DS:[EAX],AL
774501EA   0000             ADD BYTE PTR DS:[EAX],AL
774501EC  ^7D 9A            JGE SHORT ntdll.77450188
774501EE   1E               PUSH DS
774501EF   52               PUSH EDX
774501F0   0000             ADD BYTE PTR DS:[EAX],AL
774501F2   0000             ADD BYTE PTR DS:[EAX],AL
774501F4   4A               DEC EDX
774501F5   51               PUSH ECX
774501F6   0100             ADD DWORD PTR DS:[EAX],EAX
774501F8   0100             ADD DWORD PTR DS:[EAX],EAX
774501FA   0000             ADD BYTE PTR DS:[EAX],AL
774501FC   F1               INT1
774501FD   07               POP ES                                  Modification of segment register
774501FE   0000             ADD BYTE PTR DS:[EAX],AL
77450200  -E9 07000010      JMP 8745020C
77450205   0201             ADD AL,BYTE PTR DS:[ECX]
77450207   00D4             ADD AH,DL
77450209   2101             AND DWORD PTR DS:[ECX],EAX
7745020B   0078 41          ADD BYTE PTR DS:[EAX+41],BH
7745020E   0100             ADD DWORD PTR DS:[EAX],EAX
77450210   BC BC0A0058      MOV ESP,58000ABC
77450215   BD 0A0081B8      MOV EBP,B881000A
7745021A   0A00             OR AL,BYTE PTR DS:[EAX]
7745021C   15 BA0A005D      ADC EAX,5D000ABA
77450221   B8 0A0051B6      MOV EAX,B651000A
77450226   0A00             OR AL,BYTE PTR DS:[EAX]
77450228   8DBC0A 00146507  LEA EDI,DWORD PTR DS:[EDX+ECX+7651400]
7745022F   0001             ADD BYTE PTR DS:[ECX],AL
77450231   2202             AND AL,BYTE PTR DS:[EDX]
77450233   0069 21          ADD BYTE PTR DS:[ECX+21],CH
77450236   0200             ADD AL,BYTE PTR DS:[EAX]
77450238   3027             XOR BYTE PTR DS:[EDI],AH
7745023A   0300             ADD EAX,DWORD PTR DS:[EAX]
7745023C   90               NOP
7745023D   D4 07            AAM 7
7745023F   00A0 D4070080    ADD BYTE PTR DS:[EAX+800007D4],AH
77450245   D4 07            AAM 7
77450247   00F3             ADD BL,DH
77450249   26:0300          ADD EAX,DWORD PTR ES:[EAX]
EBX=7EFDE000
Stack SS:[0018FFF8]=00000000
```

```
Registers (FPU)
EAX 00401000 abexcm1.<ModuleEntryPoint>
ECX 00000000
EDX 00000000
EBX 7EFDE000
ESP 0018FFF0
EBP 00000000
ESI 00000000
EDI 00000000

EIP 774501B8 ntdll.774501B8

C 0  ES 002B 32bit 0(FFFFFFFF)
P 0  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 7EFDD000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SXS_KEY_NOT_FOUND (000036B7)

EFL 00000202 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
                3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

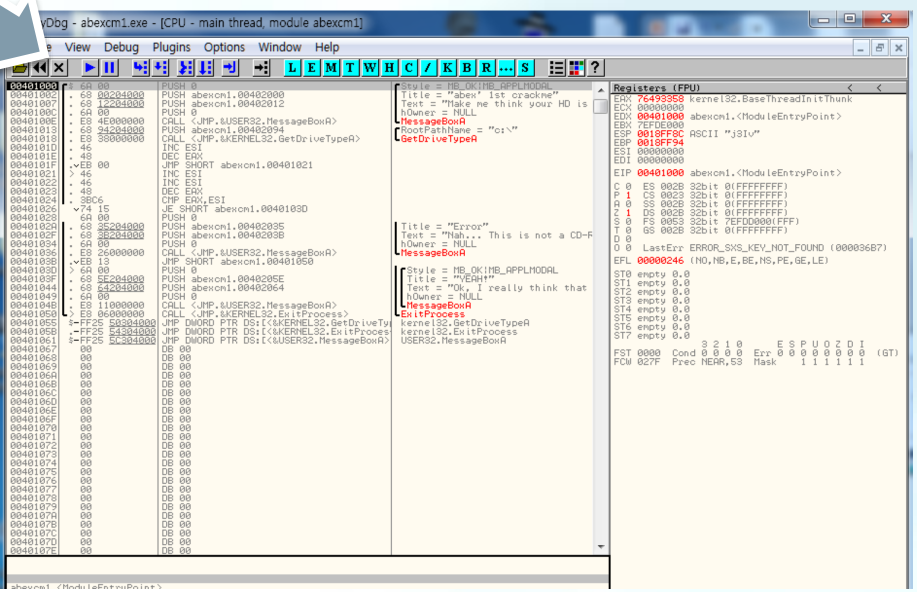
```
Address  Hex dump                 ASCII
00402000 61 62 65 78 27 20 31 73 abex' 1s
00402008 74 20 63 72 61 63 6B 6D t crackm
00402010 65 00 4D 61 6B 65 20 6D e.Make m
00402018 65 20 74 68 69 6E 6B 20 e think
00402020 79 6F 75 72 20 48 44 20 your HD
00402028 69 73 20 61 20 43 44 2D is a CD-
00402030 52 6F 6D 2E 00 45 72 72 Rom..Err
00402038 6F 72 00 4E 61 68 2E 2E or.Nah..
00402040 2E 20 54 68 69 73 20 69 . This i
00402048 73 20 6E 6F 74 20 61 20 s not a
00402050 43 44 2D 52 4F 4D 20 44 CD-ROM D
00402058 72 69 76 65 21 00 59 45 rive!.YE
00402060 41 48 21 00 4F 6B 2C 20 AH!.Ok,
00402068 49 20 72 65 61 6C 6C 79 I really
00402070 20 74 68 69 6E 6B 20 74  think t
00402078 68 61 74 20 79 6F 75 72 hat your
00402080 20 48 44 20 69 73 20 61  HD is a
00402088 20 43 44 2D 52 4F 4D 21  CD-ROM!
00402090 20 3A 70 00 63 3A 5C 00  :p.c:\.
00402098 00 00 00 00 00 00 00 00 ........
004020A0 00 00 00 00 00 00 00 00 ........
004020A8 00 00 00 00 00 00 00 00 ........
004020B0 00 00 00 00 00 00 00 00 ........
004020B8 00 00 00 00 00 00 00 00 ........
```

```
0018FFF0   00000000
0018FFF4   00401000  abexcm1.<ModuleEntryPoint>
0018FFF8   00000000
0018FFFC   00000000
```

Single step event at ntdll.774501B8 - use Shift+F7/F8/F9 to pass exception to program

Paused

F9-RUN(실행)


yDbg - abexcm1.exe - [CPU - main thread, module abexcm1]
View Debug Plugins Options Window Help
L E M T W H C / K B R ... S

```
00401000  $ 6A 00           PUSH 0                                   Style = MB_OK|MB_APPLMODAL
00401002  . 68 00204000     PUSH abexcm1.00402000                    Title = "abex' 1st crackme"
00401007  . 68 12204000     PUSH abexcm1.00402012                    Text = "Make me think your HD is
0040100C  . 6A 00           PUSH 0                                   hOwner = NULL
0040100E  . E8 4E000000     CALL <JMP.&USER32.MessageBoxA>           MessageBoxA
00401013  . 68 94204000     PUSH abexcm1.00402094                    RootPathName = "c:\"
00401018  . E8 38000000     CALL <JMP.&KERNEL32.GetDriveTypeA>       GetDriveTypeA
0040101D  . 46              INC ESI
0040101E  . 48              DEC EAX
0040101F  .vEB 00           JMP SHORT abexcm1.00401021
00401021  > 46              INC ESI
00401022  . 46              INC ESI
00401023  . 48              DEC EAX
00401024  . 3BC6            CMP EAX,ESI
00401026   v74 15           JE SHORT abexcm1.0040103D
00401028    6A 00           PUSH 0
0040102A  . 68 35204000     PUSH abexcm1.00402035                    Title = "Error"
0040102F  . 68 3B204000     PUSH abexcm1.0040203B                    Text = "Nah... This is not a CD-R
00401034  . 6A 00           PUSH 0                                   hOwner = NULL
00401036  . E8 26000000     CALL <JMP.&USER32.MessageBoxA>           MessageBoxA
0040103B  .vEB 13           JMP SHORT abexcm1.00401050
0040103D  > 6A 00           PUSH 0                                   Style = MB_OK|MB_APPLMODAL
0040103F  . 68 5E204000     PUSH abexcm1.0040205E                    Title = "YEAH!"
00401044  . 68 64204000     PUSH abexcm1.00402064                    Text = "Ok, I really think that
00401049  . 6A 00           PUSH 0                                   hOwner = NULL
0040104B  . E8 11000000     CALL <JMP.&USER32.MessageBoxA>           MessageBoxA
00401050  > E8 06000000     CALL <JMP.&KERNEL32.ExitProcess>         ExitProcess
00401055  $-FF25 50304000   JMP DWORD PTR DS:[<&KERNEL32.GetDriveTy  kernel32.GetDriveTypeA
0040105B  .-FF25 54304000   JMP DWORD PTR DS:[<&KERNEL32.ExitProces  kernel32.ExitProcess
00401061  $-FF25 5C304000   JMP DWORD PTR DS:[<&USER32.MessageBoxA>  USER32.MessageBoxA
00401067    00              DB 00
00401068    00              DB 00
00401069    00              DB 00
0040106A    00              DB 00
0040106B    00              DB 00
0040106C    00              DB 00
0040106D    00              DB 00
0040106E    00              DB 00
0040106F    00              DB 00
00401070    00              DB 00
00401071    00              DB 00
00401072    00              DB 00
00401073    00              DB 00
00401074    00              DB 00
00401075    00              DB 00
00401076    00              DB 00
00401077    00              DB 00
00401078    00              DB 00
00401079    00              DB 00
0040107A    00              DB 00
0040107B    00              DB 00
0040107C    00              DB 00
0040107D    00              DB 00
0040107E    00              DB 00
abexcm1.<ModuleEntryPoint>
```

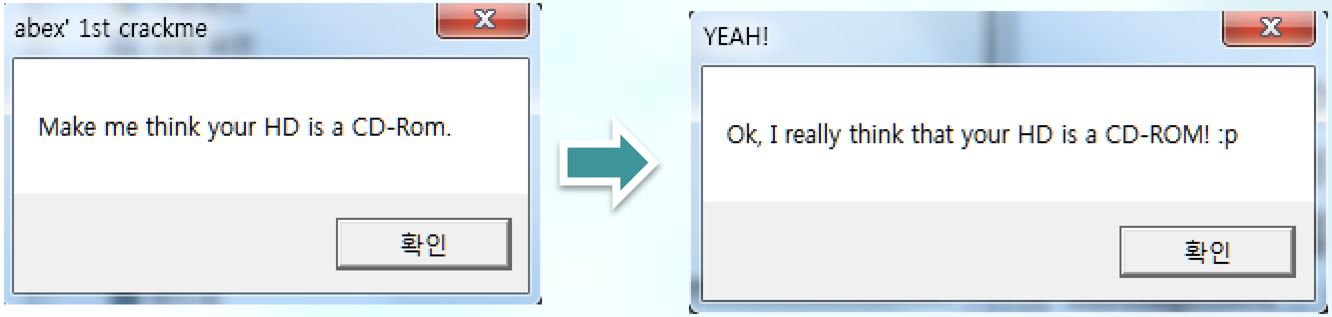
```
Registers (FPU)
EAX 76493358 kernel32.BaseThreadInitThunk
ECX 00000000
EDX 00401000 abexcm1.<ModuleEntryPoint>
EBX 7EFDE000
ESP 0018FF8C ASCII "j3Iv"
EBP 0018FF94
ESI 00000000
EDI 00000000

EIP 00401000 abexcm1.<ModuleEntryPoint>

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 7EFDD000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SXS_KEY_NOT_FOUND (000036B7)

EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
                3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

Ch6 abex's crackme #1 분석_3

- Windows API Call list
  - MessageBox, GetDriverType
  - MessageBox(실패)
  - MessageBox(성공)
- 레지스터비교
  - EAX와 ESI
  - JE SHORT 이용
    - 분기에서 MessageBox(실패,성공)으로 이동

Ch6 abex's crackme #1 분석_4

- Crack
  - CMP에서 비교한 결과 수행
    - 성공 MessageBox으로 이동함



abex' 1st crackme

Make me think your HD is a CD-Rom.

YEAH!

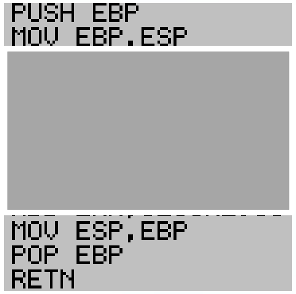
Ok, I really think that your HD is a CD-ROM! :p

Stack Frame

- 스택프레임
  - ESP(스택포인터)가 아닌 EBP(베이스포인터)를 사용
  - 스택내부의 로컬변수, 파라미터, 복귀주소에 접근
- 사용이유
  - ESP의 값은 프로그램이 동작하면 계속 변경됨
  - 상황에 따라 ESP값을 EBP에 넘겨 값을 저장
  - EBP에서는 로컬변수,파라미터,복귀주소 사용할 수 있다.

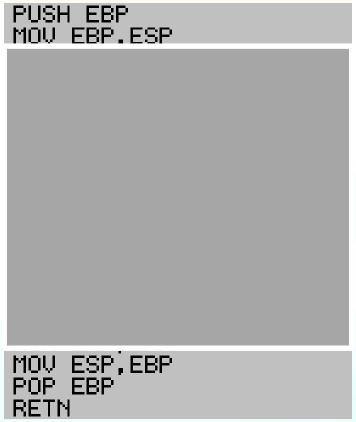
스택 프레임 구조

- 스택프레임 구조



```
PUSH EBP
MOV EBP.ESP
MOV ESP,EBP
POP EBP
RETN
```

add()

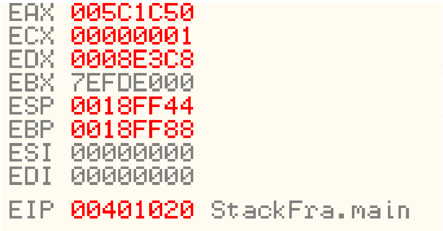


```
PUSH EBP
MOV EBP.ESP
MOV ESP,EBP
POP EBP
RETN
```

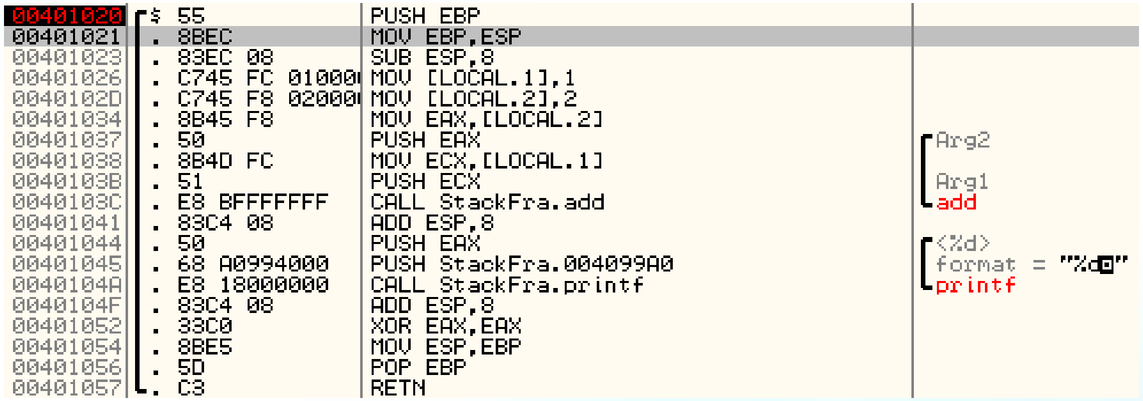
main()

- Push EBP
  - 함수시작 EBP를 사용전 기존 값을 스택에 저장
- MOV EBP, ESP
  - 현재의 ESP를 EBP에 저장
- MOV ESP, EBP
  - ESP를 다시 복원
- POP EBP
  - 처음의 EBP값으로 복원
- RETN
  - 함수종료

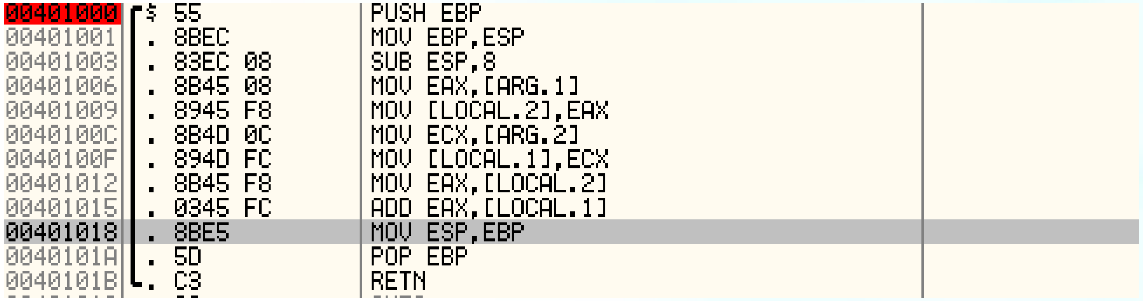
Stack Frame Debugging



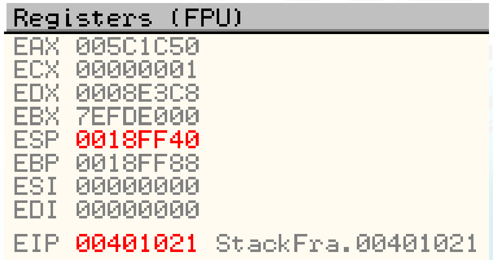
```
EAX 005C1C50
ECX 00000001
EDX 0008E3C8
EBX 7EFDE000
ESP 0018FF44
EBP 0018FF88
ESI 00000000
EDI 00000000
EIP 00401020 StackFra.main
```



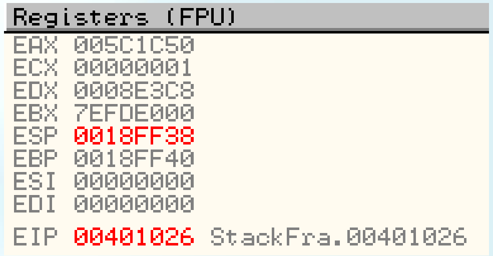
```
00401020 ┌$ 55            PUSH EBP
00401021 │. 8BEC          MOV EBP,ESP
00401023 │. 83EC 08       SUB ESP,8
00401026 │. C745 FC 01000 MOV [LOCAL.1],1
0040102D │. C745 F8 02000 MOV [LOCAL.2],2
00401034 │. 8B45 F8       MOV EAX,[LOCAL.2]
00401037 │. 50            PUSH EAX                     ┌Arg2
00401038 │. 8B4D FC       MOV ECX,[LOCAL.1]            │
0040103B │. 51            PUSH ECX                     │Arg1
0040103C │. E8 BFFFFFFF   CALL StackFra.add            └add
00401041 │. 83C4 08       ADD ESP,8
00401044 │. 50            PUSH EAX                     ┌<%d>
00401045 │. 68 A0994000   PUSH StackFra.004099A0       │format = "%d□"
0040104A │. E8 18000000   CALL StackFra.printf         └printf
0040104F │. 83C4 08       ADD ESP,8
00401052 │. 33C0          XOR EAX,EAX
00401054 │. 8BE5          MOV ESP,EBP
00401056 │. 5D            POP EBP
00401057 └. C3            RETN
```



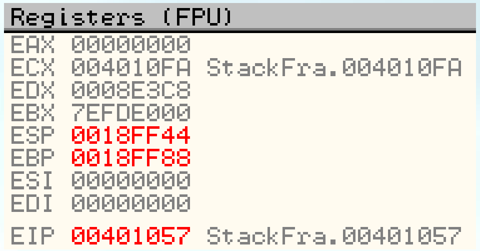
```
00401000 ┌$ 55            PUSH EBP
00401001 │. 8BEC          MOV EBP,ESP
00401003 │. 83EC 08       SUB ESP,8
00401006 │. 8B45 08       MOV EAX,[ARG.1]
00401009 │. 8945 F8       MOV [LOCAL.2],EAX
0040100C │. 8B4D 0C       MOV ECX,[ARG.2]
0040100F │. 894D FC       MOV [LOCAL.1],ECX
00401012 │. 8B45 F8       MOV EAX,[LOCAL.2]
00401015 │. 0345 FC       ADD EAX,[LOCAL.1]
00401018 │. 8BE5          MOV ESP,EBP
0040101A │. 5D            POP EBP
0040101B └. C3            RETN
```



```
Registers (FPU)
EAX 005C1C50
ECX 00000001
EDX 0008E3C8
EBX 7EFDE000
ESP 0018FF40
EBP 0018FF88
ESI 00000000
EDI 00000000
EIP 00401021 StackFra.00401021
```

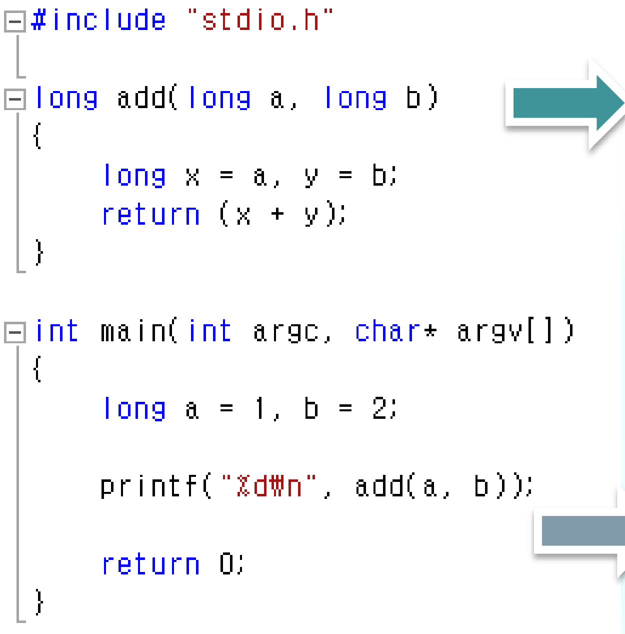


```
Registers (FPU)
EAX 005C1C50
ECX 00000001
EDX 0008E3C8
EBX 7EFDE000
ESP 0018FF38
EBP 0018FF40
ESI 00000000
EDI 00000000
EIP 00401026 StackFra.00401026
```

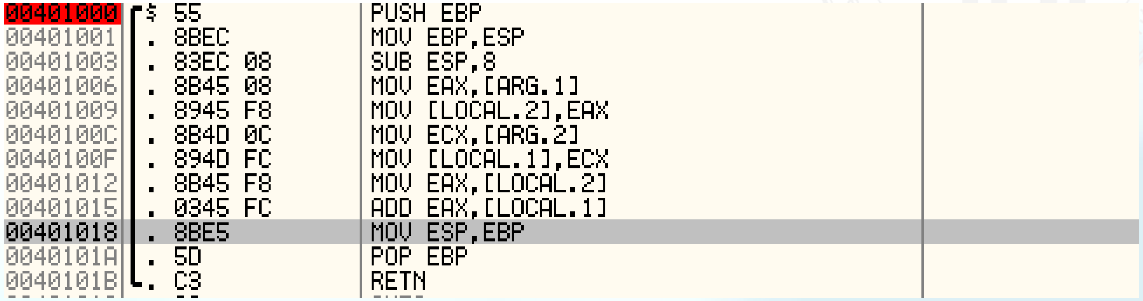


```
Registers (FPU)
EAX 00000000
ECX 004010FA StackFra.004010FA
EDX 0008E3C8
EBX 7EFDE000
ESP 0018FF44
EBP 0018FF88
ESI 00000000
EDI 00000000
EIP 00401057 StackFra.00401057
```

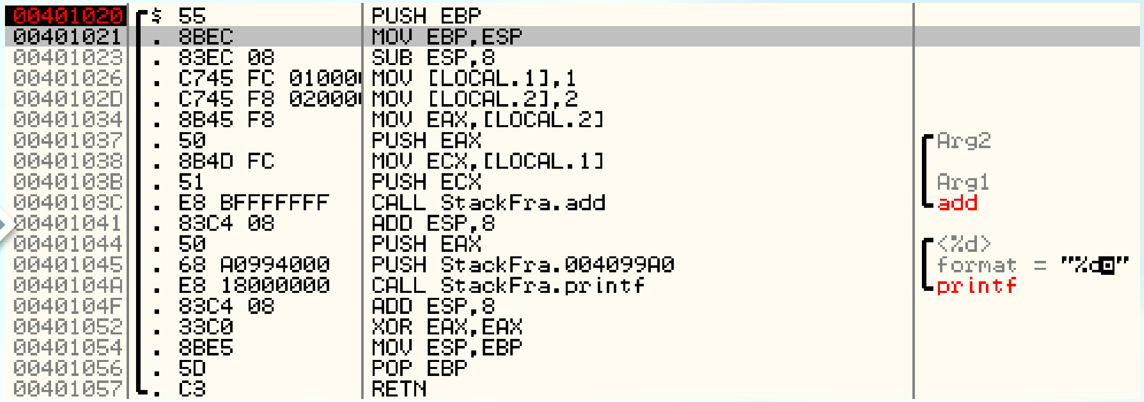
Stack Frame Code



```
#include "stdio.h"

long add( long a, long b)
{
	long x = a, y = b;
	return (x + y);
}

int main( int argc, char* argv[])
{
	long a = 1, b = 2;

	printf("%d₩n", add(a, b));

	return 0;
}
```



```
00401000 ┌$ 55          PUSH EBP
00401001 │. 8BEC        MOV EBP,ESP
00401003 │. 83EC 08     SUB ESP,8
00401006 │. 8B45 08     MOV EAX,[ARG.1]
00401009 │. 8945 F8     MOV [LOCAL.2],EAX
0040100C │. 8B4D 0C     MOV ECX,[ARG.2]
0040100F │. 894D FC     MOV [LOCAL.1],ECX
00401012 │. 8B45 F8     MOV EAX,[LOCAL.2]
00401015 │. 0345 FC     ADD EAX,[LOCAL.1]
00401018 │. 8BE5        MOV ESP,EBP
0040101A │. 5D          POP EBP
0040101B └. C3          RETN
```



```
00401020 ┌$ 55             PUSH EBP
00401021 │. 8BEC           MOV EBP,ESP
00401023 │. 83EC 08        SUB ESP,8
00401026 │. C745 FC 01000  MOV [LOCAL.1],1
0040102D │. C745 F8 02000  MOV [LOCAL.2],2
00401034 │. 8B45 F8        MOV EAX,[LOCAL.2]
00401037 │. 50             PUSH EAX                      ┌Arg2
00401038 │. 8B4D FC        MOV ECX,[LOCAL.1]             │
0040103B │. 51             PUSH ECX                      │Arg1
0040103C │. E8 BFFFFFFF    CALL StackFra.add             └add
00401041 │. 83C4 08        ADD ESP,8
00401044 │. 50             PUSH EAX                      ┌<%d>
00401045 │. 68 A0994000    PUSH StackFra.004099A0        │format = "%d□"
0040104A │. E8 18000000    CALL StackFra.printf          └printf
0040104F │. 83C4 08        ADD ESP,8
00401052 │. 33C0           XOR EAX,EAX
00401054 │. 8BE5           MOV ESP,EBP
00401056 │. 5D             POP EBP
00401057 └. C3             RETN
```