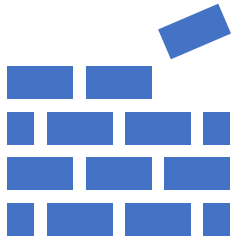
리버싱 이론 : 스택 프레임

이지훈

목차

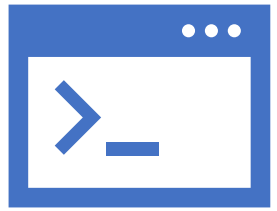


Stack Frame

Last Week

ESP & EBP

Stack Frame



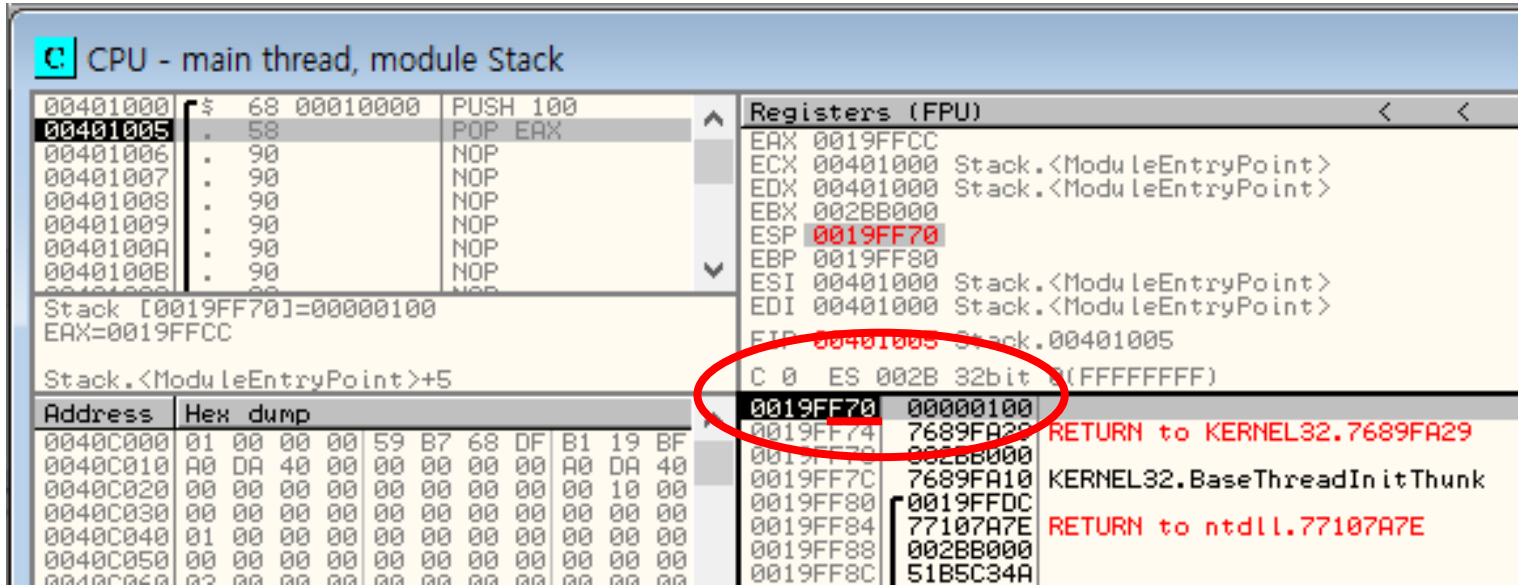
Stack Frame.exe

Stack Frame.cpp

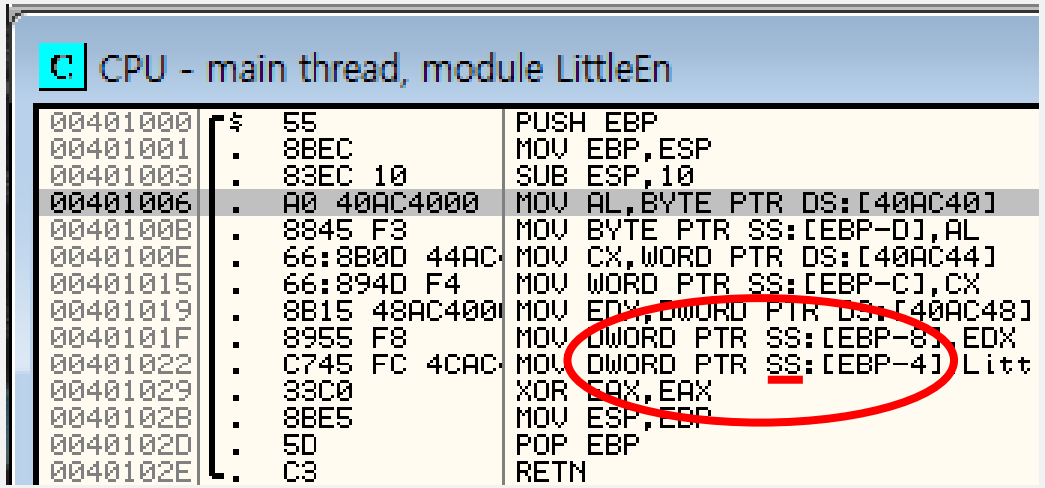
main()

add()

Last Week


CPU - main thread, module Stack
00401000 $ 68 00010000 PUSH 100
00401005 . 58 POP EAX
00401006 . 90 NOP
00401007 . 90 NOP
00401008 . 90 NOP
00401009 . 90 NOP
0040100A . 90 NOP
0040100B . 90 NOP
Stack [0019FF70]=00000100
EAX=0019FFCC
Stack.<ModuleEntryPoint>+5
Address Hex dump
0040C000 01 00 00 00 59 B7 68 DF B1 19 BF
0040C010 A0 DA 40 00 00 00 00 00 A0 DA 40
0040C020 00 00 00 00 00 00 00 00 00 10 00
0040C030 00 00 00 00 00 00 00 00 00 00 00
0040C040 01 00 00 00 00 00 00 00 00 00 00
0040C050 00 00 00 00 00 00 00 00 00 00 00
Registers (FPU)
EAX 0019FFCC
ECX 00401000 Stack.<ModuleEntryPoint>
EDX 00401000 Stack.<ModuleEntryPoint>
EBX 002BB000
ESP 0019FF70
EBP 0019FF80
ESI 00401000 Stack.<ModuleEntryPoint>
EDI 00401000 Stack.<ModuleEntryPoint>
Stack.00401005
C 0 ES 002B 32bit (FFFFFFFF)
0019FF70 00000100
0019FF74 7689FA29 RETURN to KERNEL32.7689FA29
0019FF78 002BB000
0019FF7C 7689FA10 KERNEL32.BaseThreadInitThunk
0019FF80 0019FFDC
0019FF84 77107A7E RETURN to ntdll.77107A7E
0019FF88 002BB000
0019FF8C 51B5C34A

ESP & EBP



ESP : Extended Stack Pointer, 스택의 최상단 주소값을 가리키는 레지스터

EBP : Extended Base Pointer, 스택 프레임의 베이스 주소를 가리키는 레지스터

SS : Stack Segment, 해당 메모리가 어떤 세그먼트에 소속되어 있는지 표기
(DS : Data Segment, ES : Extra data Segment)

Stack Frame

Stack Frame 기법 : ESP가 아닌 EBP 레지스터를 사용하여 스택 내의 로컬 변수,
파라미터, 복귀 주소에 접근하는 기법.

ESP의 경우 프로그램 안에서 수시로 변경되기에 CPU가 정확한 위치를 참고하기 어려움
-> 기준점, 즉 EBP가 필요하다!

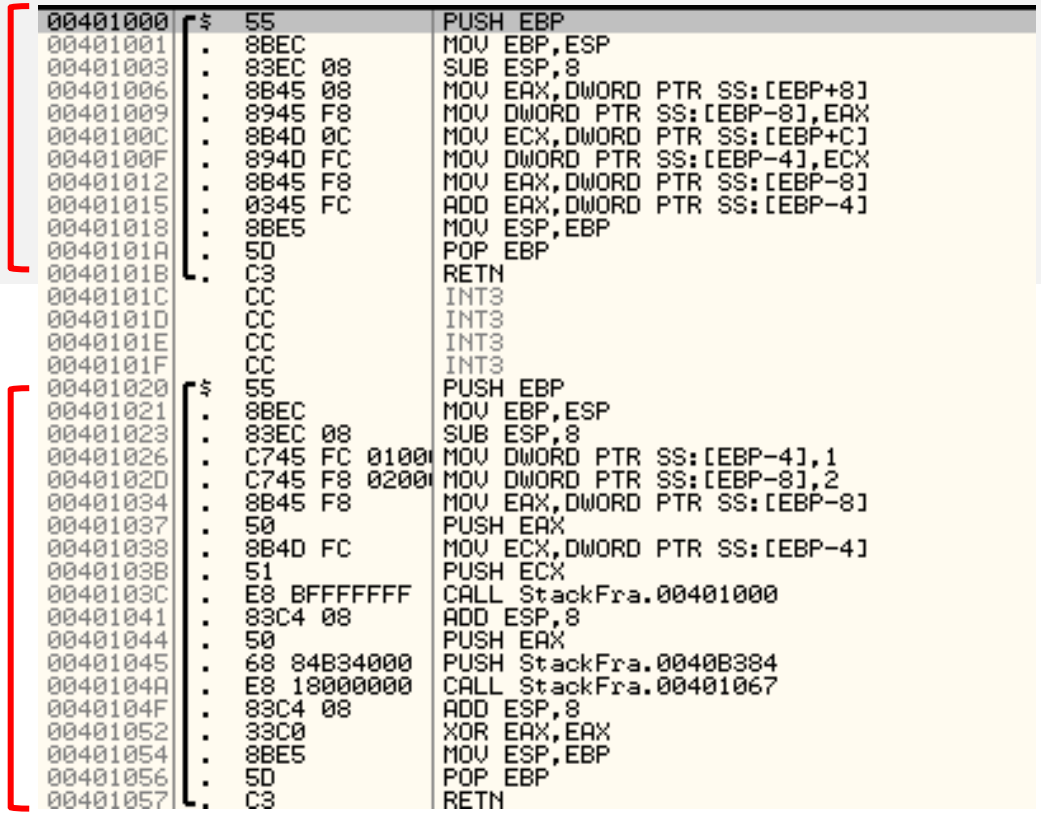
Stack Frame.cpp

```
1
2   #include "stdio.h"
3
4   long add(long a, long b)
5   {
6       long x = a, y = b;
7       return (x + y);
8   }
9
10  int main(int argc, char* argv[])
11  {
12      long a = 1, b = 2;
13
14      printf("%d\n", add(a, b));
15
16      return 0;
17  }
```

Stack Frame.exe

add()

main()



```
00401000  ┌$  55            PUSH EBP
00401001  │.  8BEC          MOV EBP,ESP
00401003  │.  83EC 08       SUB ESP,8
00401006  │.  8B45 08       MOV EAX,DWORD PTR SS:[EBP+8]
00401009  │.  8945 F8       MOV DWORD PTR SS:[EBP-8],EAX
0040100C  │.  8B4D 0C       MOV ECX,DWORD PTR SS:[EBP+C]
0040100F  │.  894D FC       MOV DWORD PTR SS:[EBP-4],ECX
00401012  │.  8B45 F8       MOV EAX,DWORD PTR SS:[EBP-8]
00401015  │.  0345 FC       ADD EAX,DWORD PTR SS:[EBP-4]
00401018  │.  8BE5          MOV ESP,EBP
0040101A  │.  5D            POP EBP
0040101B  └.  C3            RETN
0040101C      CC            INT3
0040101D      CC            INT3
0040101E      CC            INT3
0040101F      CC            INT3
00401020  ┌$  55            PUSH EBP
00401021  │.  8BEC          MOV EBP,ESP
00401023  │.  83EC 08       SUB ESP,8
00401026  │.  C745 FC 0100  MOV DWORD PTR SS:[EBP-4],1
0040102D  │.  C745 F8 0200  MOV DWORD PTR SS:[EBP-8],2
00401034  │.  8B45 F8       MOV EAX,DWORD PTR SS:[EBP-8]
00401037  │.  50            PUSH EAX
00401038  │.  8B4D FC       MOV ECX,DWORD PTR SS:[EBP-4]
0040103B  │.  51            PUSH ECX
0040103C  │.  E8 BFFFFFFF   CALL StackFra.00401000
00401041  │.  83C4 08       ADD ESP,8
00401044  │.  50            PUSH EAX
00401045  │.  68 84B34000   PUSH StackFra.0040B384
0040104A  │.  E8 18000000   CALL StackFra.00401067
0040104F  │.  83C4 08       ADD ESP,8
00401052  │.  33C0          XOR EAX,EAX
00401054  │.  8BE5          MOV ESP,EBP
00401056  │.  5D            POP EBP
00401057  └.  C3            RETN
```

main()

```
Registers (FPU)
EAX 00831CD0
ECX 00000001
EDX 00000001
EBX 00357000
ESP 0019FF28
EBP 0019FF70
```

401020 : EBP가 본래 가지고 있는 값을 스택에 넣는다.
-> 스택에 본래 EBP 값을 백업하는 것 (19FF70)

401021 : ESP의 값을 EBP로 옮긴다.
-> EBP의 값은 고정된다. (19FF28)

```
00401020  $  55              PUSH EBP
00401021  .  8BEC            MOV EBP,ESP
00401023  .  83EC 08         SUB ESP,8
00401026  .  C745 FC 0100    MOV DWORD PTR SS:[EBP-4],1
0040102D  .  C745 F8 0200    MOV DWORD PTR SS:[EBP-8],2
00401034  .  8B45 F8         MOV EAX,DWORD PTR SS:[EBP-8]
00401037  .  50              PUSH EAX
00401038  .  8B4D FC         MOV ECX,DWORD PTR SS:[EBP-4]
0040103B  .  51              PUSH ECX
0040103C  .  E8 BFFFFFFF     CALL StackFra.00401000
00401041  .  83C4 08         ADD ESP,8
00401044  .  50              PUSH EAX
00401045  .  68 84B34000     PUSH StackFra.0040B384
0040104A  .  E8 18000000     CALL StackFra.00401067
0040104F  .  83C4 08         ADD ESP,8
00401052  .  33C0            XOR EAX,EAX
00401054  .  8BE5            MOV ESP,EBP
00401056  .  5D              POP EBP
00401057  L. C3              RETN
```

main()

```
Registers (FPU)
EAX 00831CD0
ECX 00000001
EDX 00000001
EBX 00357000
ESP 0019FF28
EBP 0019FF28
```

401020 : EBP가 본래 가지고 있는 값을 스택에 넣는다.
-> 스택에 본래 EBP 값을 백업하는 것 (19FF70)

401021 : ESP의 값을 EBP로 옮긴다.
-> EBP의 값은 고정된다. (19FF28)

```
00401020  $  55            PUSH EBP
00401021  .  8BEC          MOV EBP,ESP
00401023  .  83EC 08       SUB ESP,8
00401026  .  C745 FC 0100  MOV DWORD PTR SS:[EBP-4],1
0040102D  .  C745 F8 0200  MOV DWORD PTR SS:[EBP-8],2
00401034  .  8B45 F8       MOV EAX,DWORD PTR SS:[EBP-8]
00401037  .  50            PUSH EAX
00401038  .  8B4D FC       MOV ECX,DWORD PTR SS:[EBP-4]
0040103B  .  51            PUSH ECX
0040103C  .  E8 BFFFFFFF   CALL StackFra.00401000
00401041  .  83C4 08       ADD ESP,8
00401044  .  50            PUSH EAX
00401045  .  68 84B34000   PUSH StackFra.0040B384
0040104A  .  E8 18000000   CALL StackFra.00401067
0040104F  .  83C4 08       ADD ESP,8
00401052  .  33C0          XOR EAX,EAX
00401054  .  8BE5          MOV ESP,EBP
00401056  .  5D            POP EBP
00401057  L. C3            RETN
```

main()

EBP ==>	0019FF28	0019FF70
EBP+4	0019FF2C	00401250
EBP+8	0019FF30	00000001
EBP+C	0019FF34	00831C80
EBP+10	0019FF38	00831CD0
EBP+14	0019FF3C	BFC13A62
EBP+18	0019FF40	004012A7
EBP+1C	0019FF44	004012A7
EBP+20	0019FF48	00357000

401020 : EBP가 본래 가지고 있는 값을 스택에 넣는다.
-> 스택에 본래 EBP 값을 백업하는 것 (19FF70)

401021 : ESP의 값을 EBP로 옮긴다.
-> EBP의 값은 고정된다. (19FF28)

```
00401020  $  55            PUSH EBP
00401021  .  8BEC          MOV EBP,ESP
00401023  .  83EC 08       SUB ESP,8
00401026  .  C745 FC 0100  MOV DWORD PTR SS:[EBP-4],1
0040102D  .  C745 F8 0200  MOV DWORD PTR SS:[EBP-8],2
00401034  .  8B45 F8       MOV EAX,DWORD PTR SS:[EBP-8]
00401037  .  50            PUSH EAX
00401038  .  8B4D FC       MOV ECX,DWORD PTR SS:[EBP-4]
0040103B  .  51            PUSH ECX
0040103C  .  E8 BFFFFFFF   CALL StackFra.00401000
00401041  .  83C4 08       ADD ESP,8
00401044  .  50            PUSH EAX
00401045  .  68 84B34000   PUSH StackFra.0040B384
0040104A  .  E8 18000000   CALL StackFra.00401067
0040104F  .  83C4 08       ADD ESP,8
00401052  .  33C0          XOR EAX,EAX
00401054  .  8BE5          MOV ESP,EBP
00401056  .  5D            POP EBP
00401057  L. C3            RETN
```

long a = 1, b = 2 ;

```
Registers (FPU)
EAX 00831CD0
ECX 00000001
EDX 00000001
EBX 00357000
ESP 0019FF28
EBP 0019FF28
```

```
Registers (FPU)
EAX 00831CD0
ECX 00000001
EDX 00000001
EBX 00357000
ESP 0019FF20
EBP 0019FF28
```

401023 : ESP 값에서 8을 뺀다.

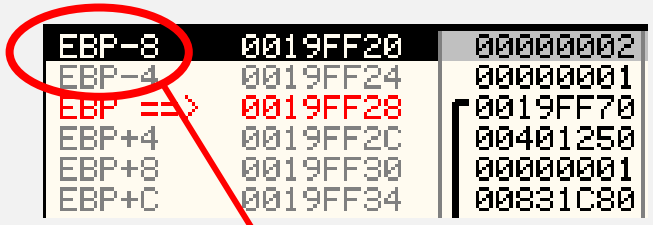
-> main() 함수의 로컬 변수를 스택에 저장하기 위함

-> 4바이트 크기 의 long 타입 변수 2개 이므로
총 8바이트의 공간이 필요하기 때문

401026 ~ 40102D : 각각의 주소에 1 과 2를 넣는다.

```
00401020  $  55            PUSH EBP
00401021  .  8BEC          MOV EBP,ESP
00401023  .  83EC 08       SUB ESP,8
00401026  .  C745 FC 0100  MOV DWORD PTR SS:[EBP-4],1
0040102D  .  C745 F8 0200  MOV DWORD PTR SS:[EBP-8],2
00401034  .  8B45 F8       MOV EAX,DWORD PTR SS:[EBP-8]
00401037  .  50            PUSH EAX
00401038  .  8B4D FC       MOV ECX,DWORD PTR SS:[EBP-4]
0040103B  .  51            PUSH ECX
0040103C  .  E8 BFFFFFFF   CALL StackFra.00401000
00401041  .  83C4 08       ADD ESP,8
00401044  .  50            PUSH EAX
00401045  .  68 84B34000   PUSH StackFra.0040B384
0040104A  .  E8 18000000   CALL StackFra.00401067
0040104F  .  83C4 08       ADD ESP,8
00401052  .  33C0          XOR EAX,EAX
00401054  .  8BE5          MOV ESP,EBP
00401056  .  5D            POP EBP
00401057  L. C3            RETN
```

long a = 1, b = 2 ;



401023 : ESP 값에서 8을 뺀다.

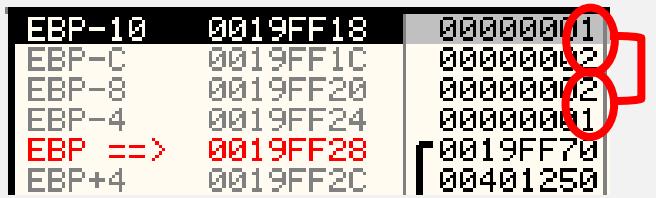
-> main() 함수의 로컬 변수를 스택에 저장하기 위함

-> 4바이트 크기 의 long 타입 변수 2개 이므로
총 8바이트의 공간이 필요하기 때문

401026 ~ 40102D : 각각의 주소에 1 과 2를 넣는다.

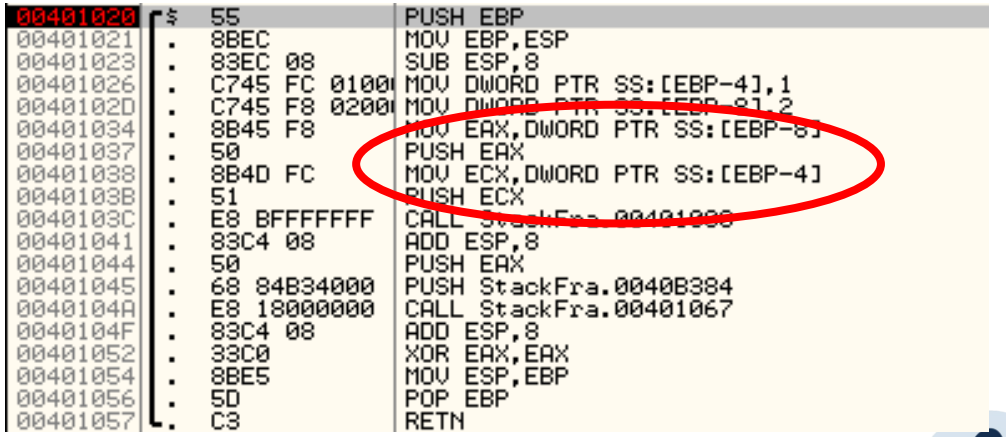
```
00401020  $  55           PUSH EBP
00401021  .  8BEC         MOV EBP,ESP
00401023  .  83EC 08      SUB ESP,8
00401026  .  C745 FC 0100 MOV DWORD PTR SS:[EBP-4],1
0040102D  .  C745 F8 0200 MOV DWORD PTR SS:[EBP-8],2
00401034  .  8B45 F8      MOV EAX,DWORD PTR SS:[EBP-8]
00401037  .  50           PUSH EAX
00401038  .  8B4D FC      MOV ECX,DWORD PTR SS:[EBP-4]
0040103B  .  51           PUSH ECX
0040103C  .  E8 BFFFFFFF  CALL StackFra.00401000
00401041  .  83C4 08      ADD ESP,8
00401044  .  50           PUSH EAX
00401045  .  68 84B34000  PUSH StackFra.0040B384
0040104A  .  E8 18000000  CALL StackFra.00401067
0040104F  .  83C4 08      ADD ESP,8
00401052  .  33C0         XOR EAX,EAX
00401054  .  8BE5         MOV ESP,EBP
00401056  .  5D           POP EBP
00401057  L. C3           RETN
```

... add(a,b));

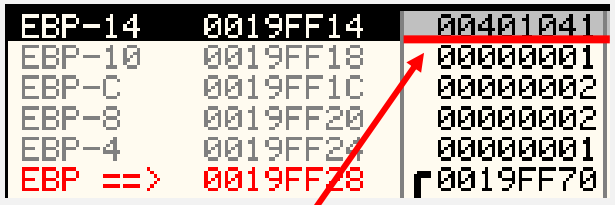


401034~ 40103B : 각각의 변수를 스택에 집어넣는다.
-> 함수 파라미터의 역순 저장

40103C : add() 함수 호출

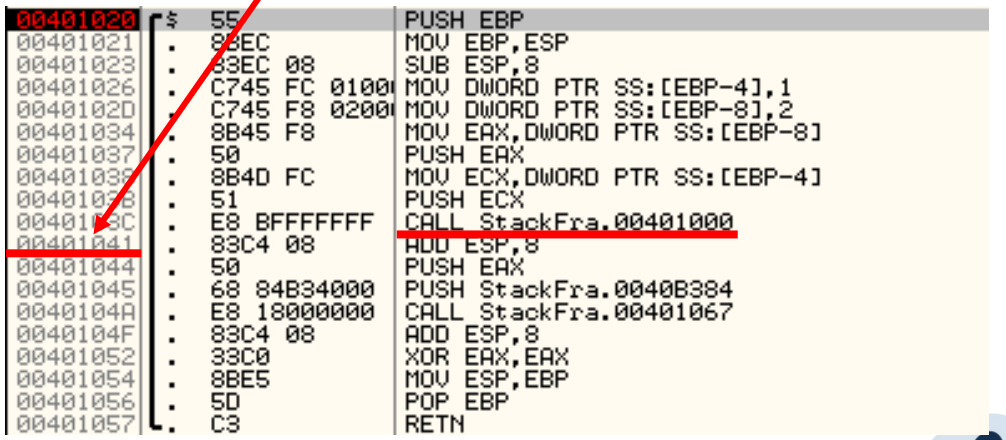


... add(a,b));



401034~ 40103B : 각각의 변수를 스택에 집어넣는다.
-> 함수 파라미터의 역순 저장

40103C : add() 함수 호출



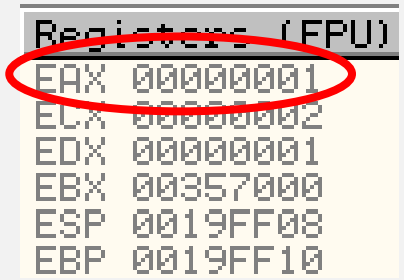
long add(long a, long b)

```
EBP-8     0019FF08   00000001
EBP-4     0019FF0C   00000002
EBP ==>   0019FF10   0019FF28
EBP+4     0019FF14   00401041
EBP+8     0019FF18   00000001
EBP+C     0019FF1C   00000002
```

main() 함수 때와 방식은 동일하지만,
스택의 위치는 다르다.

```
00401000  $  55          PUSH EBP
00401001  .  8BEC        MOV EBP,ESP
00401003  .  83EC 08     SUB ESP,8
00401006  .  8B45 08     MOV EAX,DWORD PTR SS:[EBP+8]
00401009  .  8945 F8     MOV DWORD PTR SS:[EBP-8],EAX
0040100C  .  8B4D 0C     MOV ECX,DWORD PTR SS:[EBP+C]
0040100F  .  894D FC     MOV DWORD PTR SS:[EBP-4],ECX
00401012  .  8B45 F8     MOV EAX,DWORD PTR SS:[EBP-8]
00401015  .  0345 FC     ADD EAX,DWORD PTR SS:[EBP-4]
00401018  .  8BE5        MOV ESP,EBP
0040101A  .  5D          POP EBP
0040101B  .  C3          RETN
```

return (x + y);

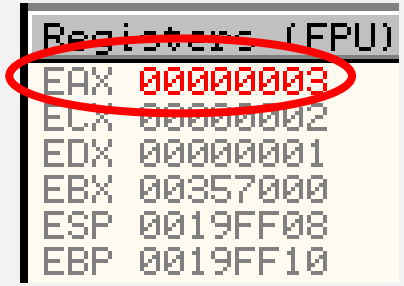


401012 : EAX에 변수 x의 값 ([EBP-8] = 1) 을 넣는다.

401015 : EAX에 변수 y의 값 ([EBP-4] = 2) 를 더한다.

```
00401000  $  55          PUSH EBP
00401001  .  8BEC        MOV EBP,ESP
00401003  .  83EC 08     SUB ESP,8
00401006  .  8B45 08     MOV EAX,DWORD PTR SS:[EBP+8]
00401009  .  8945 F8     MOV DWORD PTR SS:[EBP-8],EAX
0040100C  .  8B4D 0C     MOV ECX,DWORD PTR SS:[EBP+C]
0040100F  .  894D FC     MOV DWORD PTR SS:[EBP-4],ECX
00401012  .  8B45 F8     MOV EAX,DWORD PTR SS:[EBP-8]
00401015  .  0345 FC     ADD EAX,DWORD PTR SS:[EBP-4]
00401018  .  8BE5        MOV ESP,EBP
0040101A  .  5D          POP EBP
0040101B  .  C3          RETN
```

return (x + y);

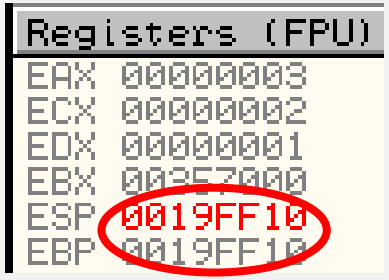


401012 : EAX에 변수 x의 값 ([EBP-8] = 1) 을 넣는다.

401015 : EAX에 변수 y의 값 ([EBP-4] = 2) 를 더한다.

```
00401000  $  55          PUSH EBP
00401001  .  8BEC        MOV EBP,ESP
00401003  .  83EC 08     SUB ESP,8
00401006  .  8B45 08     MOV EAX,DWORD PTR SS:[EBP+8]
00401009  .  8945 F8     MOV DWORD PTR SS:[EBP-8],EAX
0040100C  .  8B4D 0C     MOV ECX,DWORD PTR SS:[EBP+C]
0040100F  .  894D FC     MOV DWORD PTR SS:[EBP-4],ECX
00401012  .  8B45 F8     MOV EAX,DWORD PTR SS:[EBP-8]
00401015  .  0345 FC     ADD EAX,DWORD PTR SS:[EBP-4]
00401018  .  8BE5        MOV ESP,EBP
0040101A  .  5D          POP EBP
0040101B  .  C3          RETN
```

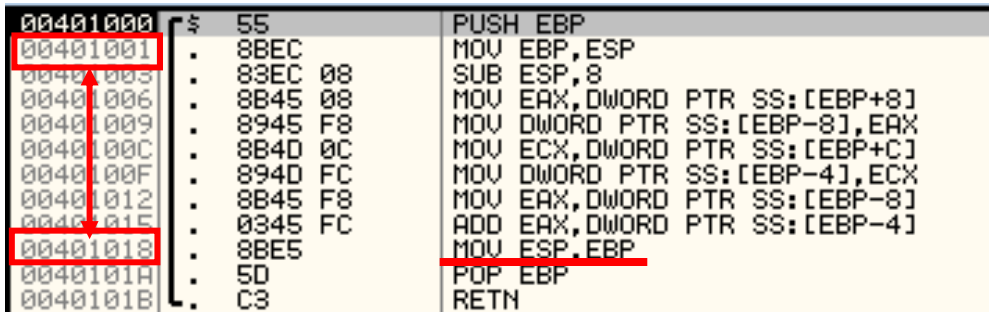
return (x + y);



401018 : EBP 값을 ESP에 대입한다.
-> 401001 주소와 대응하는 코드

40101A : 스택에 백업한 EBP 값을 복원한다.
-> 401000 주소와 대응하는 코드

40101B : 스택에 저장된 복귀 주소로 리턴



return (x + y);

```
Registers (FPU)
EAX 00000003
ECX 00000002
EDX 00000001
EBX 00357000
ESP 0019FF14
EBP 0019FF28
```

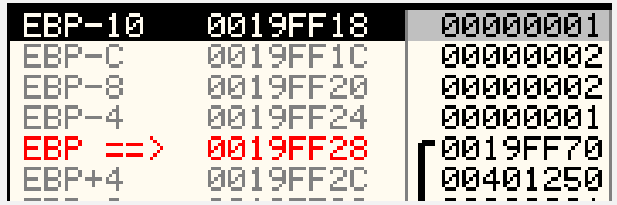
401018 : EBP 값을 ESP에 대입한다.
-> 401001 주소와 대응하는 코드

40101A : 스택에 백업한 EBP 값을 복원한다.
-> 401000 주소와 대응하는 코드

40101B : 스택에 저장된 복귀 주소로 리턴

```
00401000  $  55          PUSH EBP
00401001  .  8BEC        MOV EBP,ESP
00401003  .  83EC 08     SUB ESP,8
00401006  .  8B45 08     MOV EAX,DWORD PTR SS:[EBP+8]
00401009  .  8945 F8     MOV DWORD PTR SS:[EBP-8],EAX
0040100C  .  8B4D 0C     MOV ECX,DWORD PTR SS:[EBP+C]
0040100F  .  894D FC     MOV DWORD PTR SS:[EBP-4],ECX
00401012  .  8B45 F8     MOV EAX,DWORD PTR SS:[EBP-8]
00401015  .  0345 FC     ADD EAX,DWORD PTR SS:[EBP-4]
00401018  .  8BE5        MOV ESP,EBP
0040101A  .  5D          POP EBP
0040101B  L. C3          RETN
```

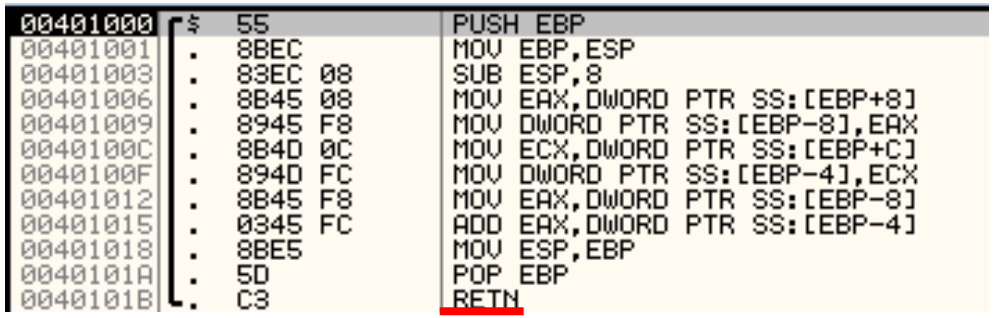
return (x + y);



401018 : EBP 값을 ESP에 대입한다.
-> 401001 주소와 대응하는 코드

40101A : 스택에 백업한 EBP 값을 복원한다.
-> 401000 주소와 대응하는 코드

40101B : 스택에 저장된 복귀 주소로 리턴



main()

```
EBP-8     0019FF20   00000002
EBP-4     0019FF24   00000001
EBP ==>   0019FF28   0019FF70
EBP+4     0019FF2C   00401250
EBP+8     0019FF30   00000001
EBP+C     0019FF34   00831C80
```

401041 : ESP에 8을 더한다.
-> add() 함수에게 넘겨준 파라미터를 정리하는 것

401044 : EAX 값에는 add()함수에서 저장된 리턴 값이 있음

401045 ~ 40104F : printf() 함수 내용. (C 표준 라이브러리)

```
00401020  $  55              PUSH EBP
00401021  .  8BEC            MOV EBP,ESP
00401023  .  83EC 08         SUB ESP,8
00401026  .  C745 FC 0100(   MOV DWORD PTR SS:[EBP-4],1
0040102D  .  C745 F8 0200(   MOV DWORD PTR SS:[EBP-8],2
00401034  .  8B45 F8         MOV EAX,DWORD PTR SS:[EBP-8]
00401037  .  50              PUSH EAX
00401038  .  8B4D FC         MOV ECX,DWORD PTR SS:[EBP-4]
0040103B  .  51              PUSH ECX
0040103C  .  E8 BFFFFFFF     CALL StackFra.00401000
00401041  .  83C4 08         ADD ESP,8
00401044  .  50              PUSH EAX
00401045  .  68 84B34000     PUSH StackFra.0040B384
0040104A  .  E8 18000000     CALL StackFra.00401067
0040104F  .  83C4 08         ADD ESP,8
00401052  .  33C0            XOR EAX,EAX
00401054  .  8BE5            MOV ESP,EBP
00401056  .  5D              POP EBP
00401057  L. C3              RETN
```

main()

```
Registers (FPU)
EAX 00000003
ECX 00000002
EDX 00000001
EBX 00357000
ESP 0019FF1C
EBP 0019FF28
```

```
EBP-C     0019FF1C   00000003
EBP-8     0019FF20   00000002
EBP-4     0019FF24   00000001
EBP ==>   0019FF28   0019FF70
EBP+4     0019FF2C   00401250
EBP+8     0019FF30   00000001
```

401041 : ESP에 8을 더한다.
-> add() 함수에게 넘겨준 파라미터를 정리하는 것

401044 : EAX 값에는 add()함수에서 저장된 리턴 값이 있음

401045 ~ 40104F : printf() 함수 내용. (C 표준 라이브러리)

```
00401020  $  55              PUSH EBP
00401021  .  8BEC            MOV EBP,ESP
00401023  .  83EC 08         SUB ESP,8
00401026  .  C745 FC 0100(   MOV DWORD PTR SS:[EBP-4],1
0040102D  .  C745 F8 0200(   MOV DWORD PTR SS:[EBP-8],2
00401034  .  8B45 F8         MOV EAX,DWORD PTR SS:[EBP-8]
00401037  .  50              PUSH EAX
00401038  .  8B4D FC         MOV ECX,DWORD PTR SS:[EBP-4]
0040103B  .  51              PUSH ECX
0040103C  .  E8 BFFFFFFF     CALL StackFra.00401000
00401041  .  83C4 08         ADD ESP,8
00401044  .  50              PUSH EAX
00401045  .  68 84B34000     PUSH StackFra.0040B384
0040104A  .  E8 18000000     CALL StackFra.00401067
0040104F  .  83C4 08         ADD ESP,8
00401052  .  33C0            XOR EAX,EAX
00401054  .  8BE5            MOV ESP,EBP
00401056  .  5D              POP EBP
00401057  L. C3              RETN
```

return 0;

```
Registers (FPU)
EAX 00000000
ECX 00401102 Stac
EDX 0019E324
EBX 003F7000
ESP 0019FF2C
EBP 0019FF70
```

401052 : main()함수의 리턴 값 세팅 (0)

-> XOR 명령어의 경우 실행 속도가 빨라

레지스터를 초기화할 때 자주 사용된다.

401054 ~ 401057 : add() 함수 때와 방식은 동일하다.

```
00401020  $  55            PUSH EBP
00401021  .  8BEC          MOV EBP,ESP
00401023  .  83EC 08       SUB ESP,8
00401026  .  C745 FC 0100  MOV DWORD PTR SS:[EBP-4],1
0040102D  .  C745 F8 0200  MOV DWORD PTR SS:[EBP-8],2
00401034  .  8B45 F8       MOV EAX,DWORD PTR SS:[EBP-8]
00401037  .  50            PUSH EAX
00401038  .  8B4D FC       MOV ECX,DWORD PTR SS:[EBP-4]
0040103B  .  51            PUSH ECX
0040103C  .  E8 BFFFFFFF   CALL StackFra.00401000
00401041  .  83C4 08       ADD ESP,8
00401044  .  50            PUSH EAX
00401045  .  68 84B34000   PUSH StackFra.0040B384
0040104A  .  E8 18000000   CALL StackFra.00401067
0040104F  .  83C4 08       ADD ESP,8
00401052  .  33C0          XOR EAX,EAX
00401054  .  8BE5          MOV ESP,EBP
00401056  .  5D            POP EBP
00401057  L. C3            RETN
```

Q & A