Fast campus

CS SCHOOL

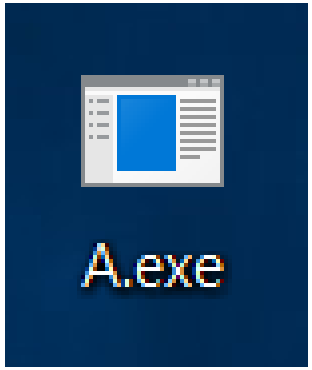
Instruction에 대해

C 코드

```c
#include <stdio.h>
int main(void)
{
    int a = 10, b = 20;
    int c = a + b;
    printf("%d + %d = %d \n",a, b, c);
    getchar();
    return 0;
}
```

Fast campus

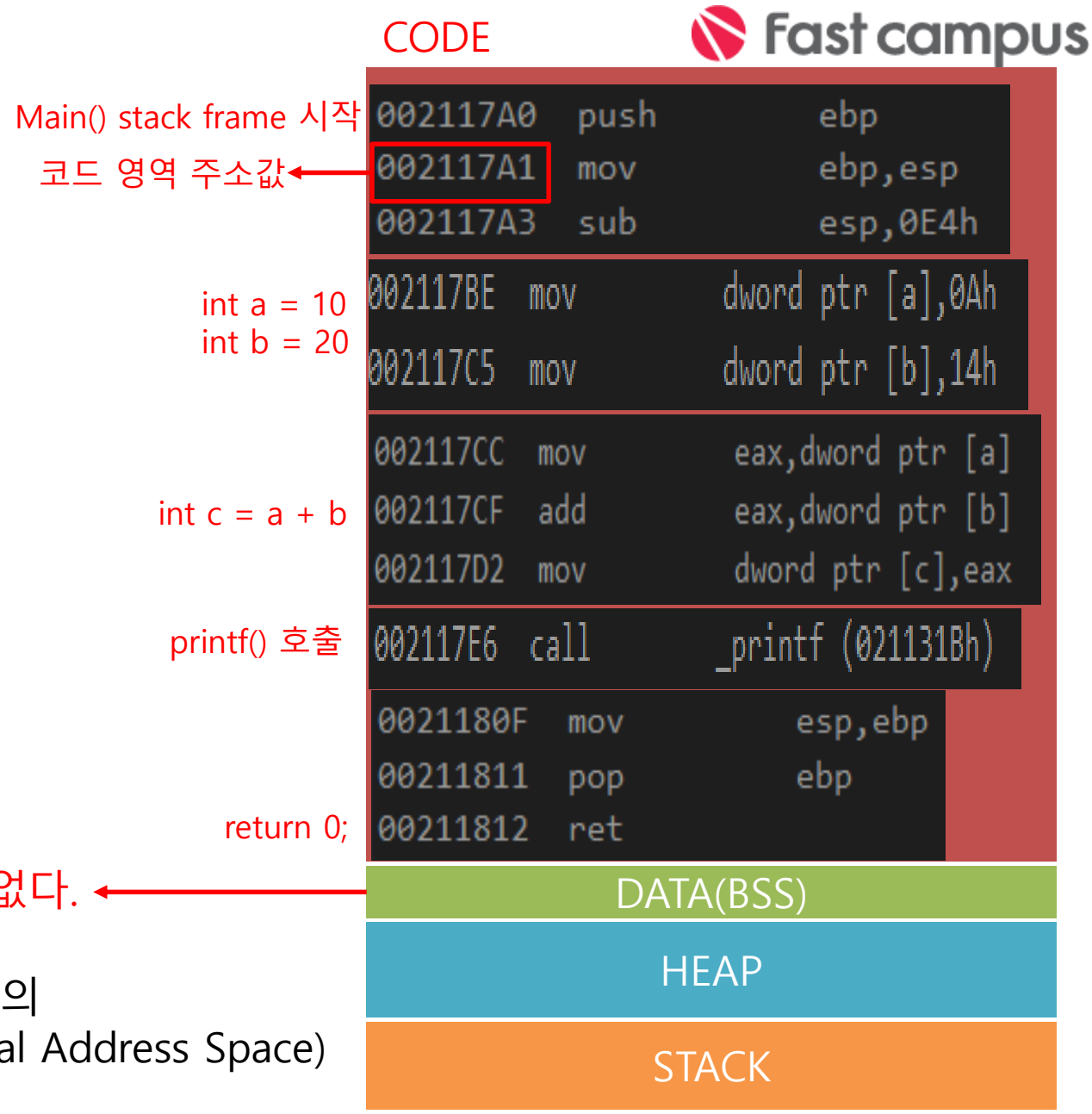
Compile 및 linking을 거쳐 실행 파일(.exe) 생성



이제 이 파일을 더블 클릭해 실행할 때
레지스터와 메인 메모리에서 어떤 일이 일어나는지 보겠습니다.

Instruction에 대해

Instruction들이
모두 코드 영역으로!!



프로세스 실행 시 4G의
가상 주소 공간(Virtual Address Space)

register

- 프로그램 카운터(PC)
- 인스트럭션 레지스터(IR)
- 스택 포인터(SP)
- eax
- ebx
- ecx
- edx
- ebp

Instruction을 한 줄 한 줄 실행해보자!

CODE

Fast campus

```
002117A0  push        ebp
002117A1  mov         ebp,esp
002117A3  sub         esp,0E4h
002117BE  mov         dword ptr [a],0Ah
002117C5  mov         dword ptr [b],14h
002117CC  mov         eax,dword ptr [a]
002117CF  add         eax,dword ptr [b]
002117D2  mov         dword ptr [c],eax
002117E6  call        _printf (021131Bh)
0021180F  mov         esp,ebp
00211811  pop         ebp
00211812  ret
```

DATA(BSS)

HEAP

STACK

register

002117A0

인스트럭션 레지스터(IR)

스택 포인터(SP)

eax

ebx

ecx

edx

ebp

프로그램 카운터는 다음에 실행할 Instruction의 주소값

CODE

Fast campus

```
002117A0  push        ebp
002117A1  mov         ebp,esp
002117A3  sub         esp,0E4h
002117BE  mov         dword ptr [a],0Ah
002117C5  mov         dword ptr [b],14h
002117CC  mov         eax,dword ptr [a]
002117CF  add         eax,dword ptr [b]
002117D2  mov         dword ptr [c],eax
002117E6  call        _printf (021131Bh)
0021180F  mov         esp,ebp
00211811  pop         ebp
00211812  ret
```

DATA(BSS)

HEAP

STACK

register

002117A1

PUSH ebp

스택 포인터(SP)

eax

ebx

ecx

edx

ebp

CODE

```
002117A0  push        ebp
002117A1  mov         ebp,esp
002117A3  sub         esp,0E4h
002117BE  mov         dword ptr [a],0Ah
002117C5  mov         dword ptr [b],14h
002117CC  mov         eax,dword ptr [a]
002117CF  add         eax,dword ptr [b]
002117D2  mov         dword ptr [c],eax
002117E6  call        _printf (021131Bh)
0021180F  mov         esp,ebp
00211811  pop         ebp
00211812  ret
```

DATA(BSS)

HEAP

STACK

인스트럭션 레지스터는 이번에 실행할 인스트럭션 정보를 담는다.

register

002117A1

PUSH ebp

스택 포인터(SP)

eax

ebx

ecx

edx

ebp

CODE

```
002117A0  push        ebp
002117A1  mov         ebp,esp
002117A3  sub         esp,0E4h
002117BE  mov         dword ptr [a],0Ah
002117C5  mov         dword ptr [b],14h
002117CC  mov         eax,dword ptr [a]
002117CF  add         eax,dword ptr [b]
002117D2  mov         dword ptr [c],eax
002117E6  call        _printf (021131Bh)
0021180F  mov         esp,ebp
00211811  pop         ebp
00211812  ret
```

DATA(BSS)

HEAP

STACK

스택 포인터는 스택을 가리키고 있다.

register

002117A1

PUSH ebp

스택 포인터(SP)

eax

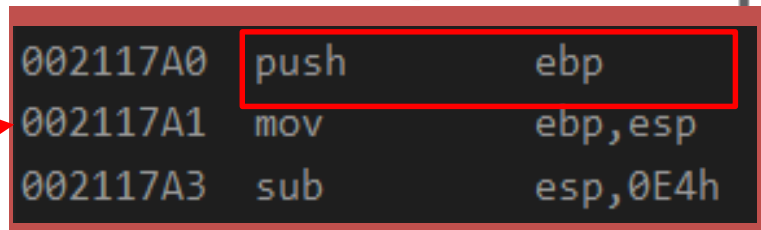
ebx

ecx

edx

ebp

CODE



DATA(BSS)

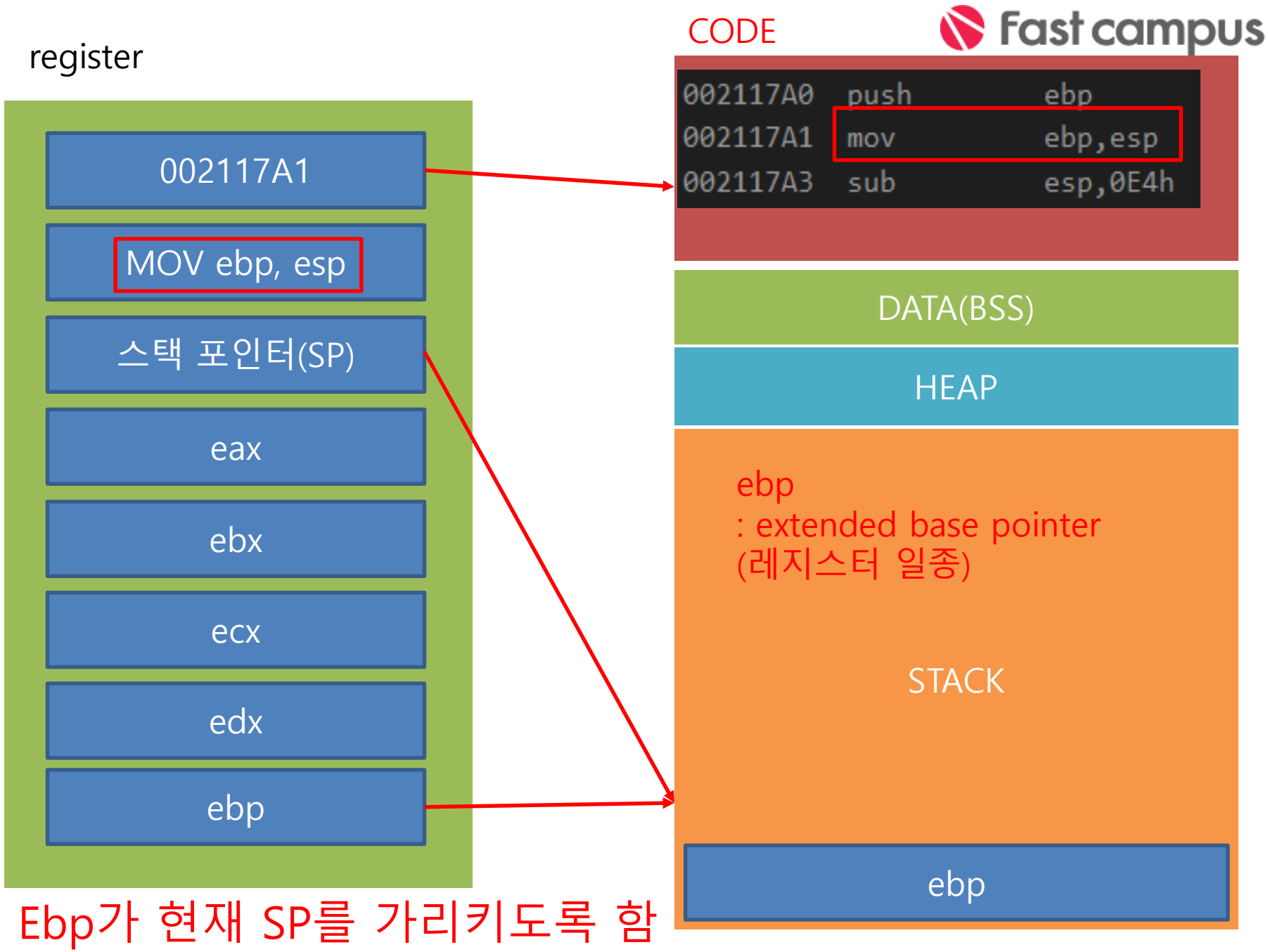
HEAP

ebp
: extended base pointer
(레지스터 일종)

STACK

ebp

stack에 ebp 넣음


register
002117A1
MOV ebp, esp
스택 포인터(SP)
eax
ebx
ecx
edx
ebp
CODE
002117A0 push ebp
002117A1 mov ebp,esp
002117A3 sub esp,0E4h
DATA(BSS)
HEAP
ebp
: extended base pointer
(레지스터 일종)
STACK
ebp
Ebp가 현재 SP를 가리키도록 함

register

002117A5

SUB esp, 12

스택 포인터(SP)

eax

ebx

ecx

edx

ebp

CODE

```
002117A0  push        ebp
002117A1  mov         ebp,esp
002117A3  sub         esp,0E4h
```

DATA(BSS)

HEAP

esp
: extended stack pointer
(SP)

STACK

ebp

SP로 미리 지역변수의 메모리 영역을 확보

register

002117C5

MOV a, 0Ah(10)

스택 포인터(SP)

eax

ebx

ecx

edx

ebp

CODE

```
002117BE  mov         dword ptr [a],0Ah
002117C5  mov         dword ptr [b],14h
```

DATA(BSS)

HEAP

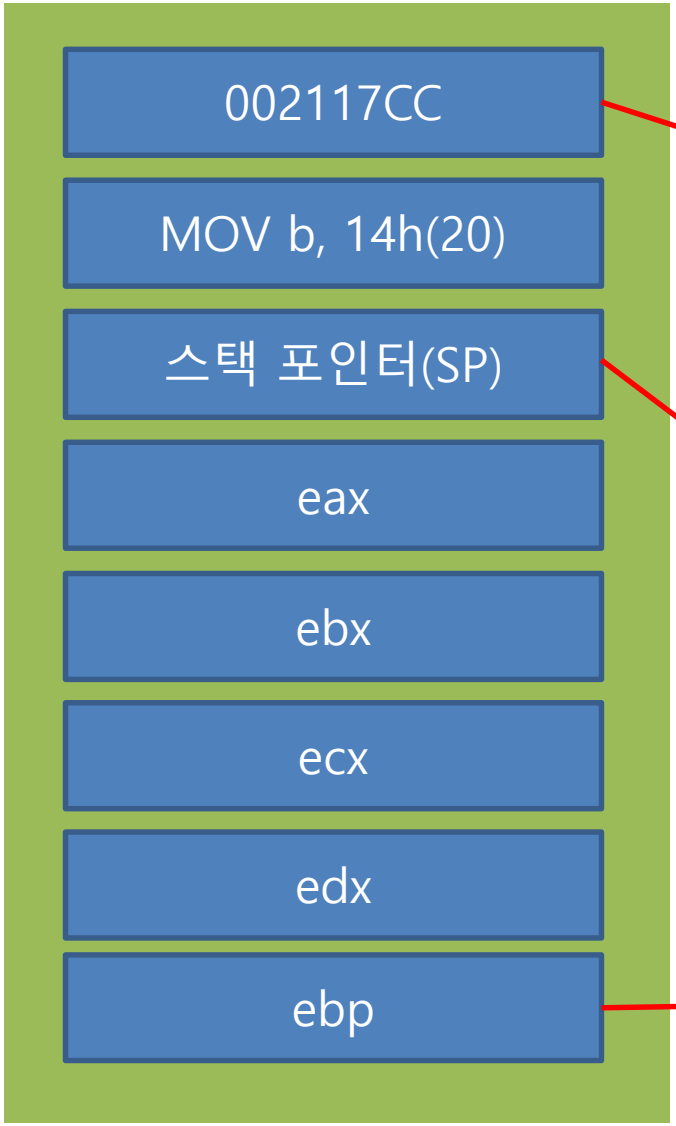
STACK

a(10)

ebp

변수 a에 10 대입

register



CODE

Fast campus

```
002117BE  mov         dword ptr [a],0Ah
002117C5  mov         dword ptr [b],14h
```

DATA(BSS)

HEAP

STACK

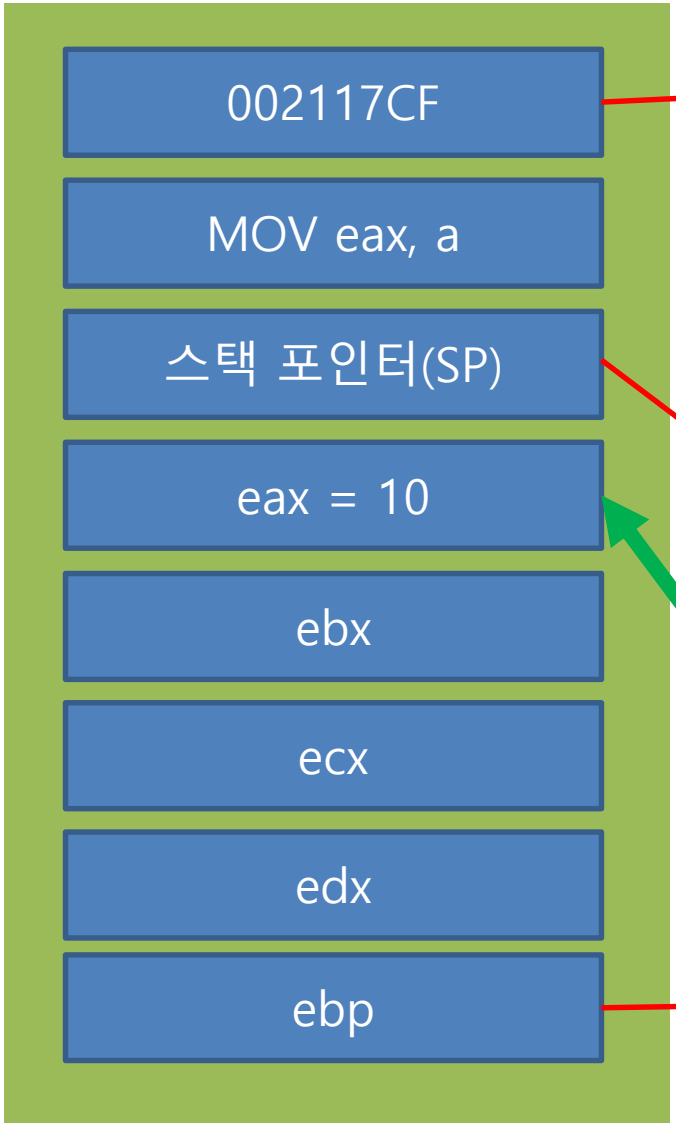
b(20)

a(10)

ebp

변수 b에 20 대입

register



CODE

```
002117CC  mov         eax,dword ptr [a]
002117CF  add         eax,dword ptr [b]
002117D2  mov         dword ptr [c],eax
```

DATA(BSS)

HEAP

STACK

b(20)

a(10)

ebp

eax에 변수 a의 값 LOAD

register

002117D2

ADD eax, b

스택 포인터(SP)

eax = 10 + 20

ebx

ecx

edx

ebp

CODE

```
002117CC  mov         eax,dword ptr [a]
002117CF  add         eax,dword ptr [b]
002117D2  mov         dword ptr [c],eax
```

DATA(BSS)

HEAP

STACK

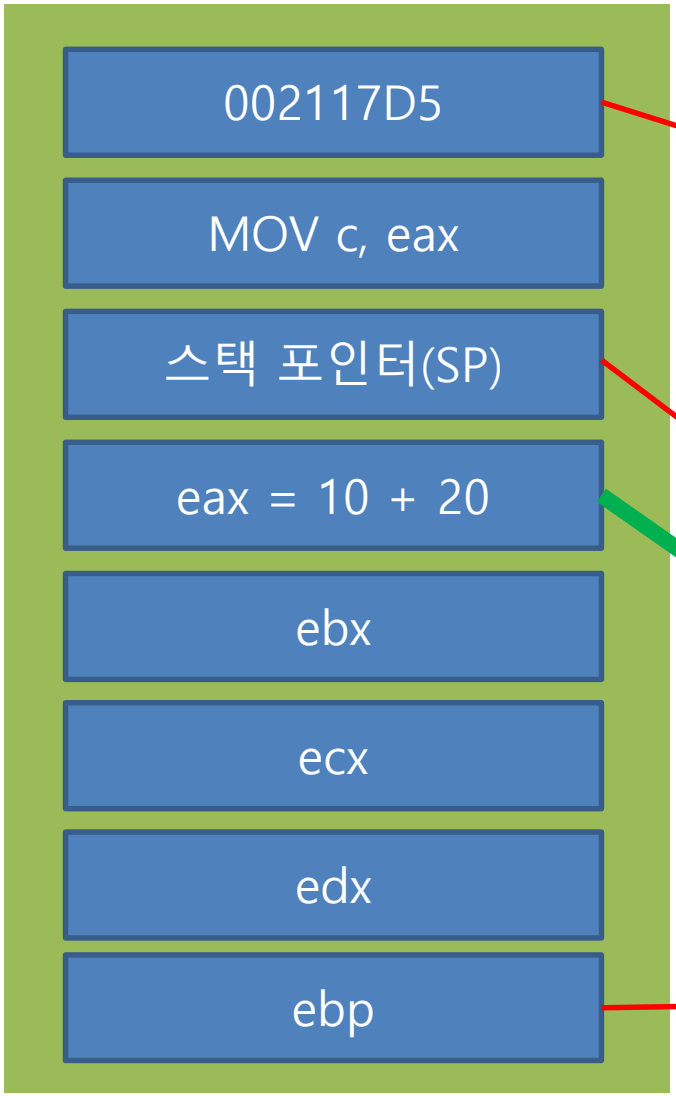
b(20)

a(10)

ebp

eax에 저장된 값과 b의 값을 LOAD 후
연산, Eax에 저장

register



CODE

```
002117CC  mov         eax,dword ptr [a]
002117CF  add         eax,dword ptr [b]
002117D2  mov         dword ptr [c],eax
```

DATA(BSS)

HEAP

STACK

c(30)

b(20)

a(10)

ebp

eax에 저장된 값을 변수 c에 STORE

register

002117EA

CALL _printf()

스택 포인터(SP)

eax

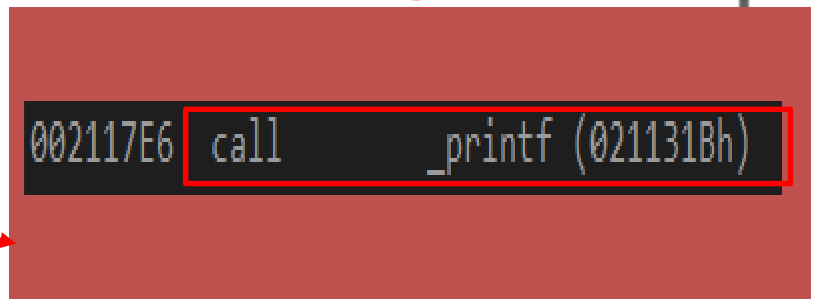
ebx

ecx

edx

ebp

CODE



DATA(BSS)

HEAP

STACK

c(30)

b(20)

a(10)

ebp

printf() 함수 호출

register

00211811

MOV esp, ebp

스택 포인터(SP)

eax

ebx

ecx

edx

ebp

CODE

```
0021180F   mov          esp,ebp
00211811   pop          ebp
00211812   ret
```

DATA(BSS)

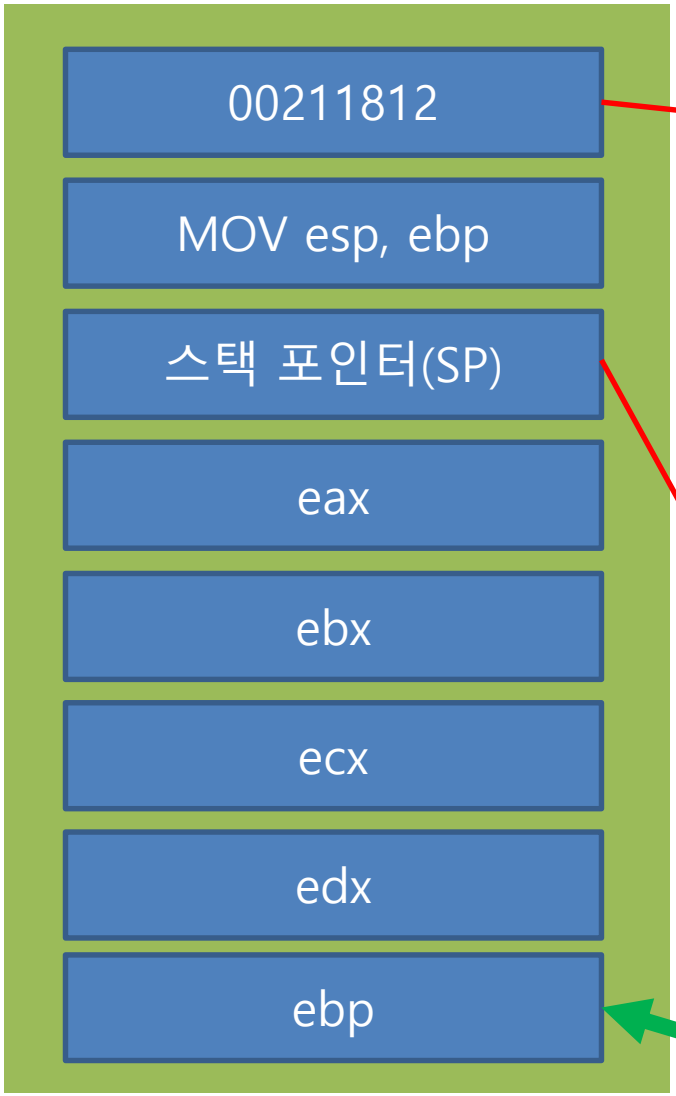
HEAP

STACK

ebp

SP에 ebp 값을 넣음으로써
스택을 비우는 역할을 수행함

register



CODE

```
0021180F   mov          esp,ebp
00211811   pop          ebp
00211812   ret
```

DATA(BSS)

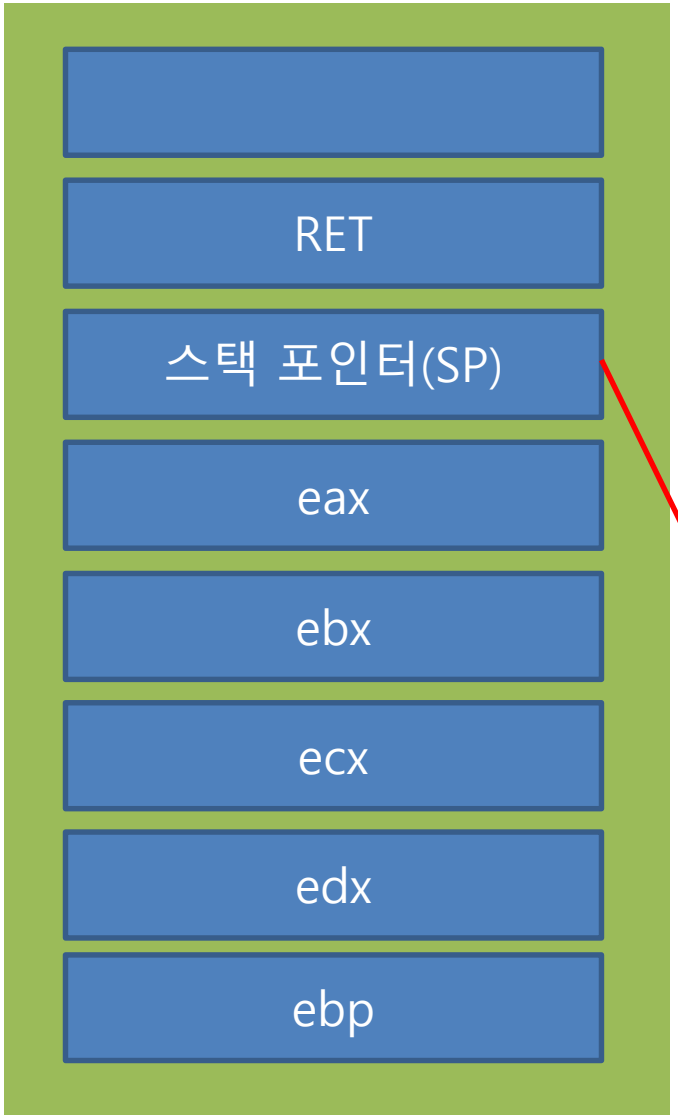
HEAP

STACK

ebp

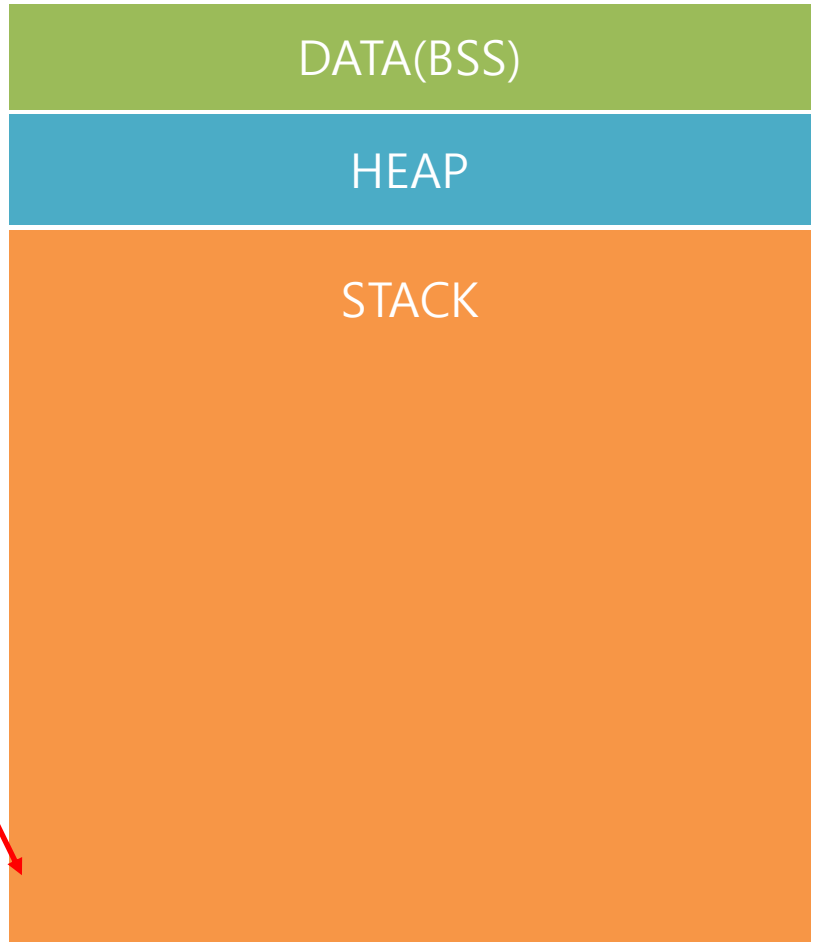
스택에 저장되어 있던 이전 ebp 값을
ebp 레지스터에 다시 저장

register



CODE

```
0021180F   mov          esp,ebp
00211811   pop          ebp
00211812   ret
```



RET 실행하면서 프로세스 종료 및 메모리 해제