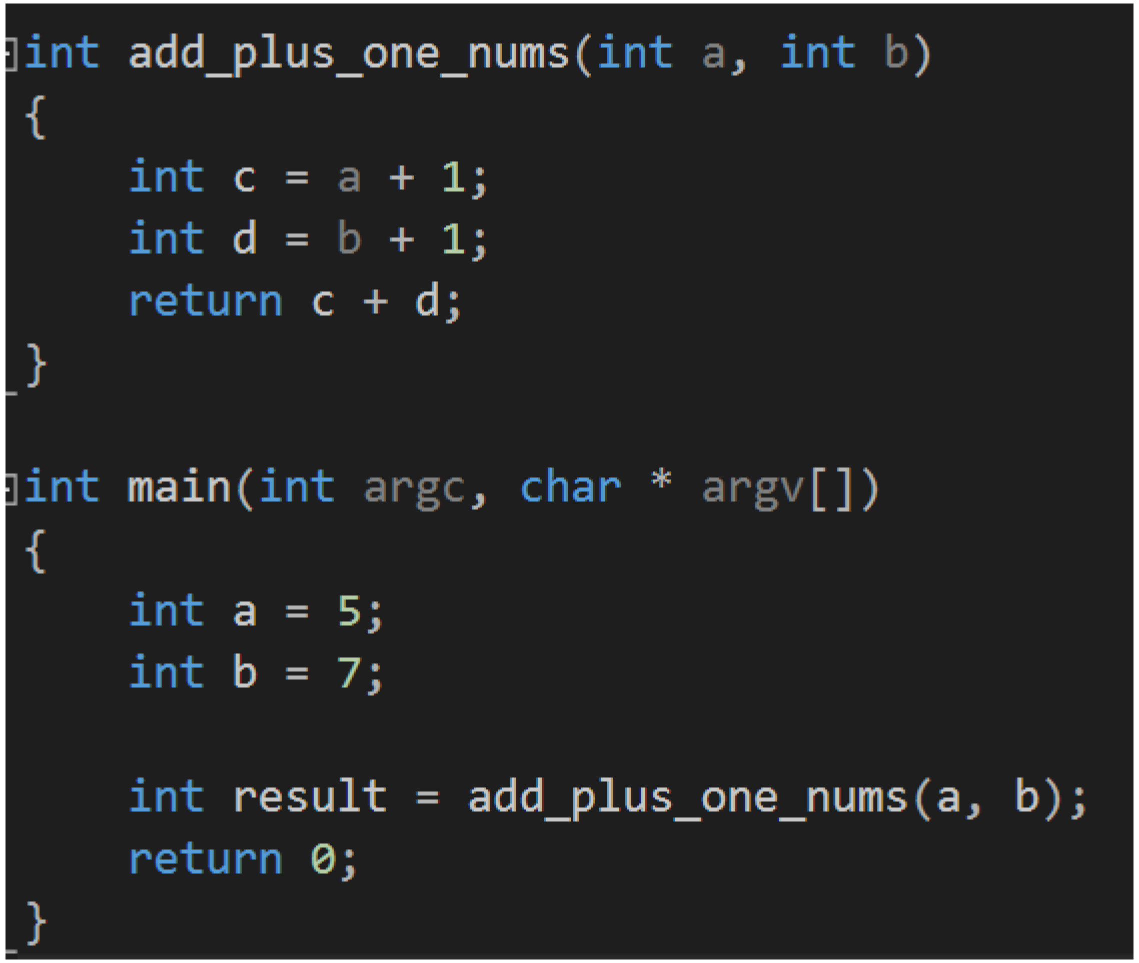
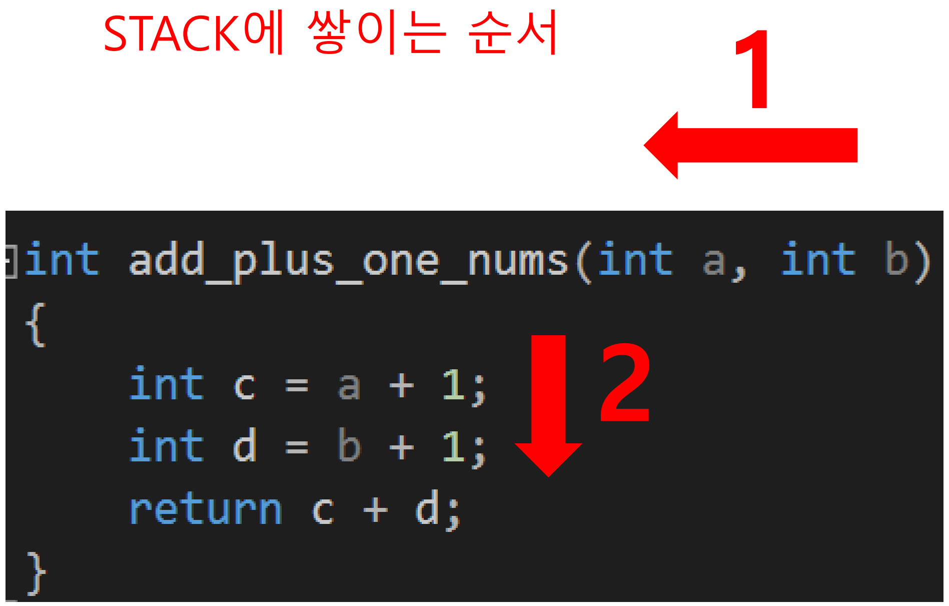
Stack frame



```
int add_plus_one_nums(int a, int b)
{
    int c = a + 1;
    int d = b + 1;
    return c + d;
}

int main(int argc, char * argv[])
{
    int a = 5;
    int b = 7;

    int result = add_plus_one_nums(a, b);
    return 0;
}
```

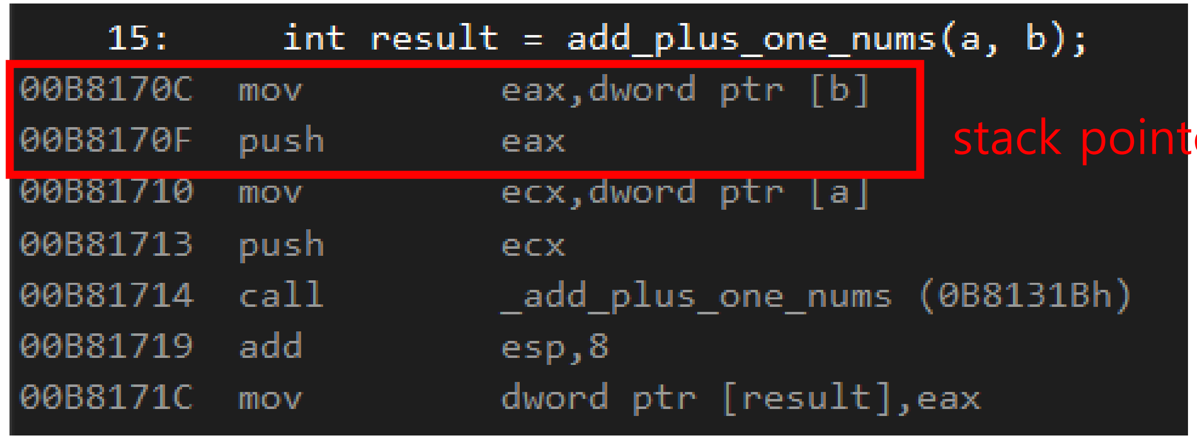
Stack frame



```
int add_plus_one_nums(int a, int b)
{
    int c = a + 1;
    int d = b + 1;
    return c + d;
}
```

Stack frame

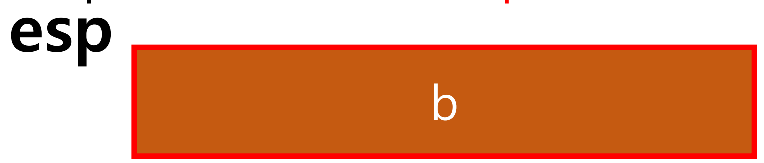
STACK에 쌓이는 순서



```
    15:      int result = add_plus_one_nums(a, b);
00B8170C  mov         eax,dword ptr [b]
00B8170F  push        eax
00B81710  mov         ecx,dword ptr [a]
00B81713  push        ecx
00B81714  call        _add_plus_one_nums (0B8131Bh)
00B81719  add         esp,8
00B8171C  mov         dword ptr [result],eax
```

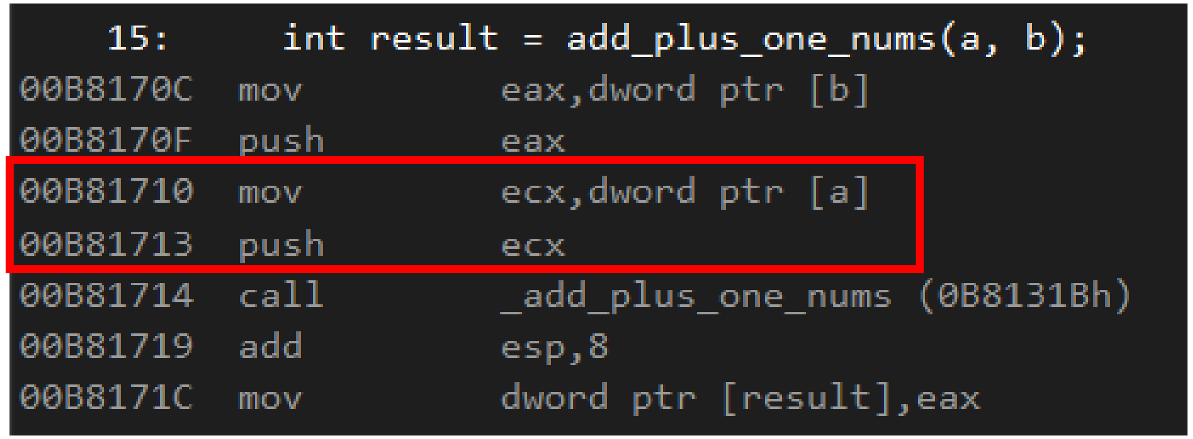
stack pointer가 가리키는 곳에 push

esp : extended stack pointer

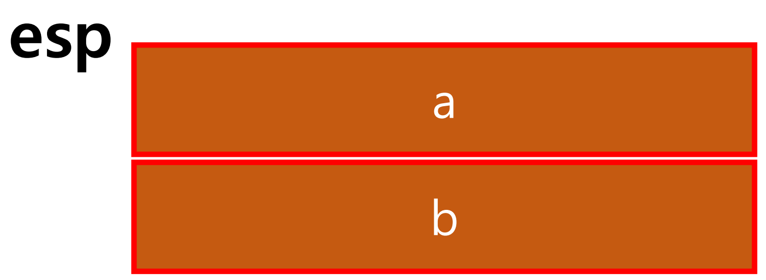


Stack frame

STACK에 쌓이는 순서

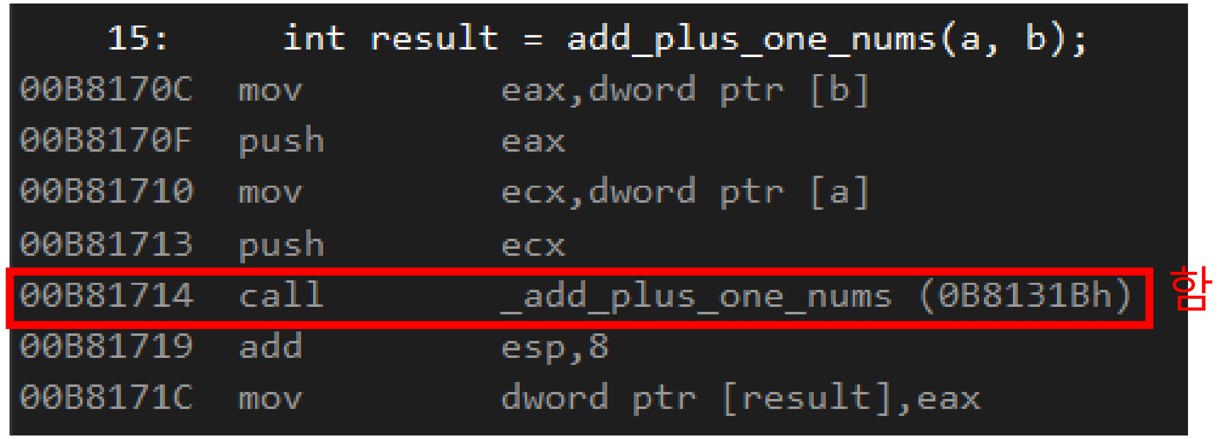


```
    15:      int result = add_plus_one_nums(a, b);
00B8170C  mov          eax,dword ptr [b]
00B8170F  push         eax
00B81710  mov          ecx,dword ptr [a]
00B81713  push         ecx
00B81714  call         _add_plus_one_nums (0B8131Bh)
00B81719  add          esp,8
00B8171C  mov          dword ptr [result],eax
```



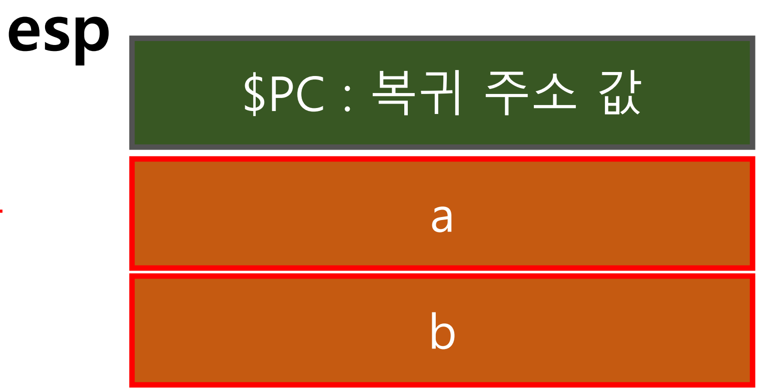
Stack frame

STACK에 쌓이는 순서



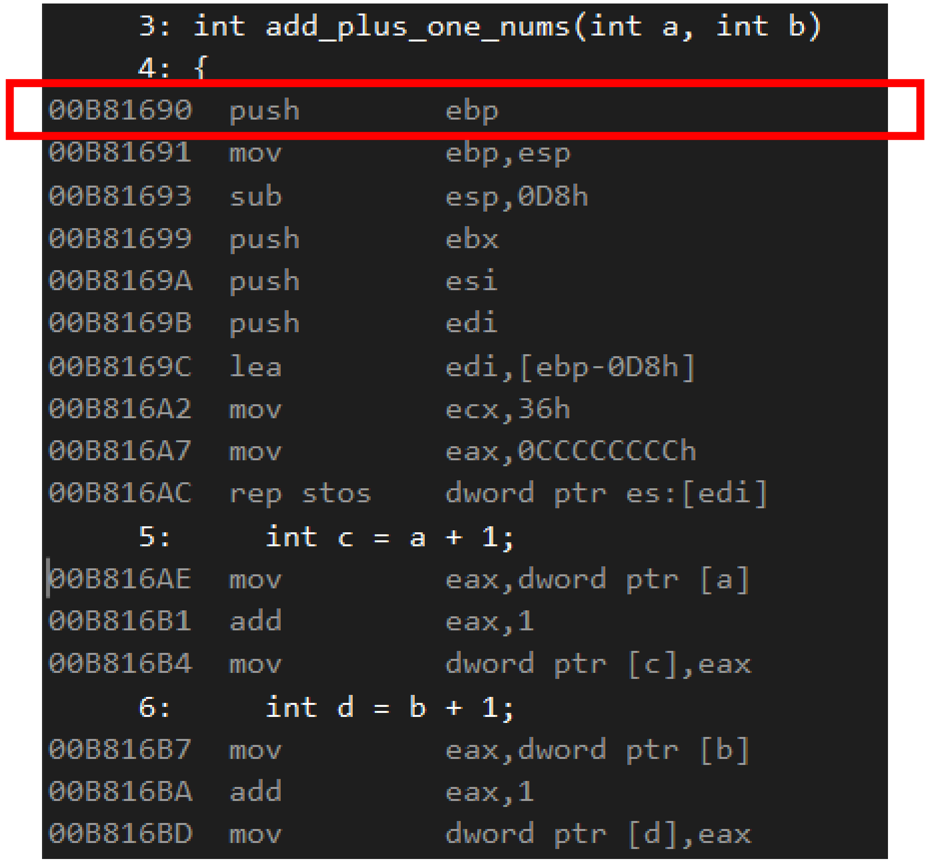
```
    15:      int result = add_plus_one_nums(a, b);
00B8170C   mov          eax,dword ptr [b]
00B8170F   push         eax
00B81710   mov          ecx,dword ptr [a]
00B81713   push         ecx
00B81714   call         _add_plus_one_nums (0B8131Bh)
00B81719   add          esp,8
00B8171C   mov          dword ptr [result],eax
```

함수 호출

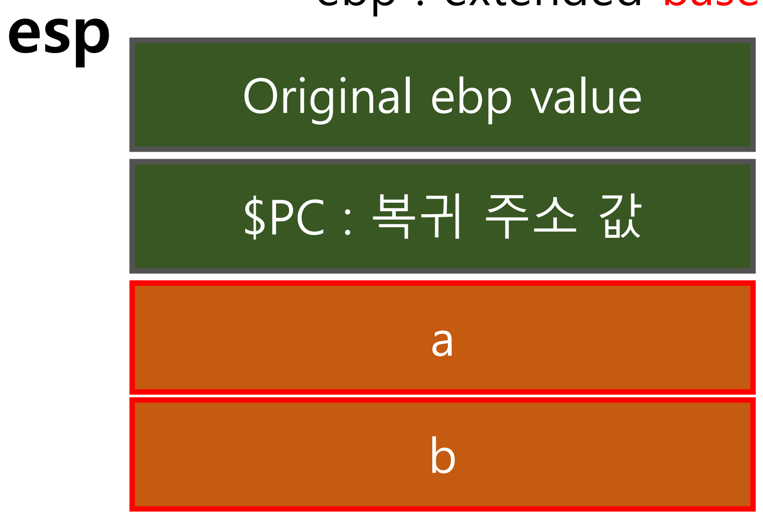


Stack frame

STACK에 쌓이는 순서

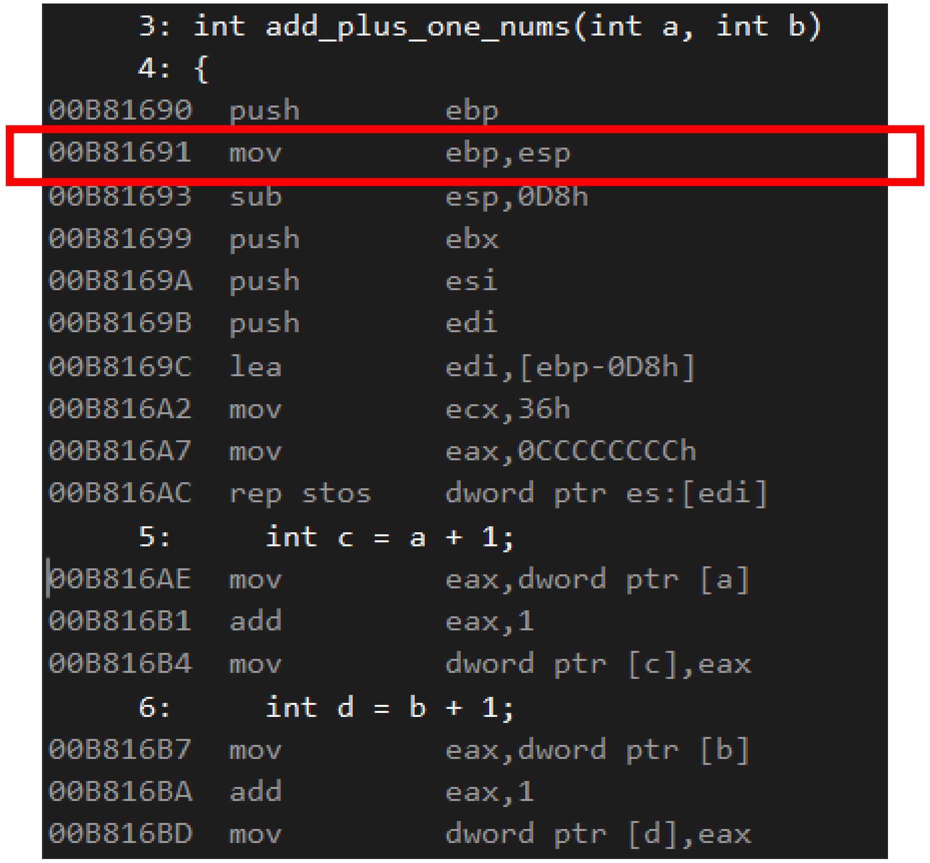

```
     3: int add_plus_one_nums(int a, int b)
     4: {
00B81690  push        ebp
00B81691  mov         ebp,esp
00B81693  sub         esp,0D8h
00B81699  push        ebx
00B8169A  push        esi
00B8169B  push        edi
00B8169C  lea         edi,[ebp-0D8h]
00B816A2  mov         ecx,36h
00B816A7  mov         eax,0CCCCCCCCh
00B816AC  rep stos    dword ptr es:[edi]
     5:     int c = a + 1;
00B816AE  mov         eax,dword ptr [a]
00B816B1  add         eax,1
00B816B4  mov         dword ptr [c],eax
     6:     int d = b + 1;
00B816B7  mov         eax,dword ptr [b]
00B816BA  add         eax,1
00B816BD  mov         dword ptr [d],eax
```

ebp : extended base pointer



Stack frame

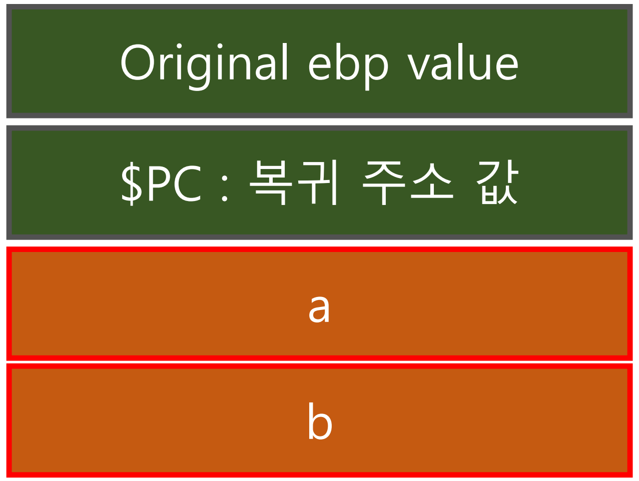
STACK에 쌓이는 순서



```
     3: int add_plus_one_nums(int a, int b)
     4: {
00B81690  push        ebp
00B81691  mov         ebp,esp
00B81693  sub         esp,0D8h
00B81699  push        ebx
00B8169A  push        esi
00B8169B  push        edi
00B8169C  lea         edi,[ebp-0D8h]
00B816A2  mov         ecx,36h
00B816A7  mov         eax,0CCCCCCCCh
00B816AC  rep stos    dword ptr es:[edi]
     5:     int c = a + 1;
00B816AE  mov         eax,dword ptr [a]
00B816B1  add         eax,1
00B816B4  mov         dword ptr [c],eax
     6:     int d = b + 1;
00B816B7  mov         eax,dword ptr [b]
00B816BA  add         eax,1
00B816BD  mov         dword ptr [d],eax
```

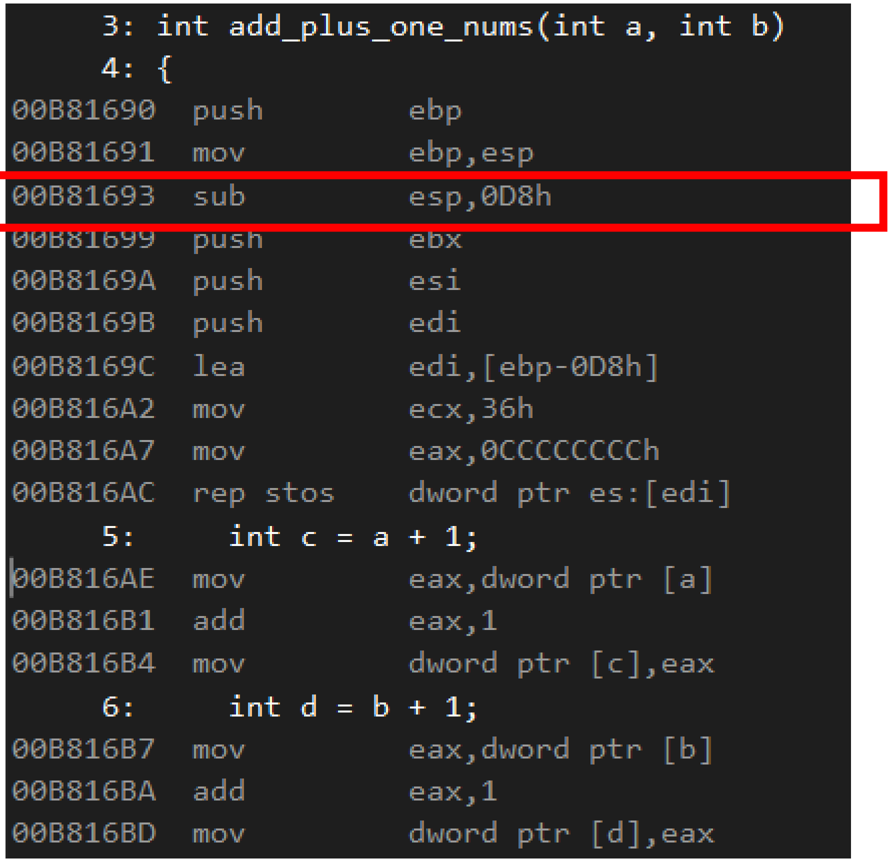
ebp = esp

ebp : extended base pointer



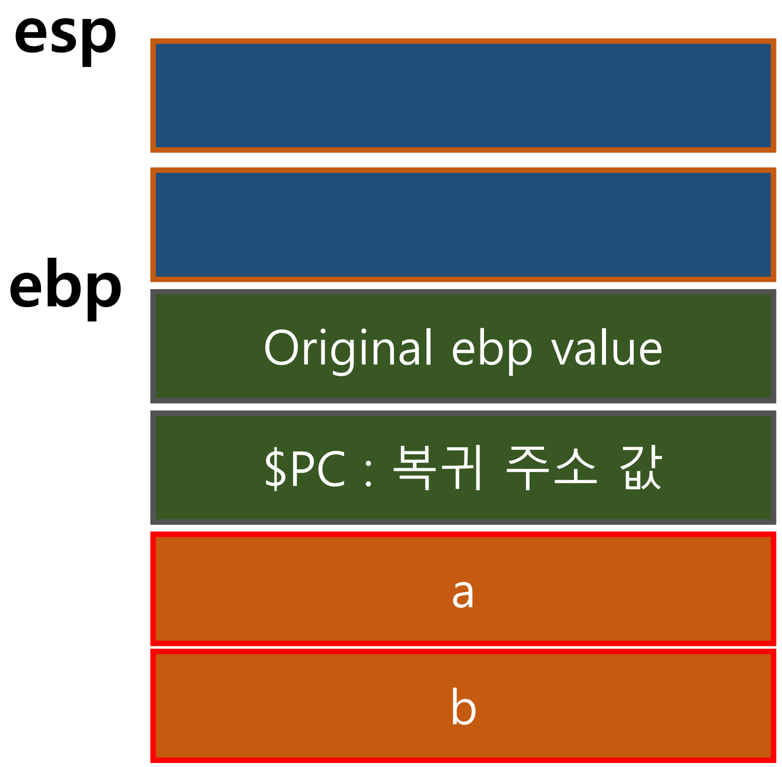
Stack frame

STACK에 쌓이는 순서



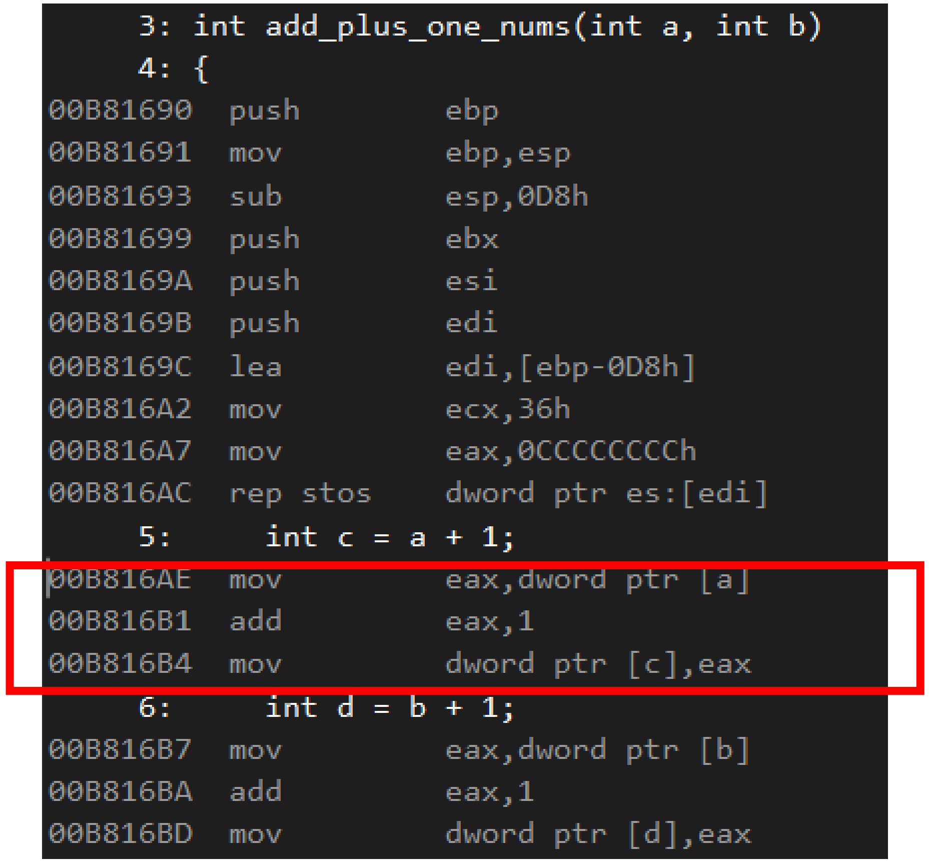
```
     3: int add_plus_one_nums(int a, int b)
     4: {
00B81690  push        ebp
00B81691  mov         ebp,esp
00B81693  sub         esp,0D8h
00B81699  push        ebx
00B8169A  push        esi
00B8169B  push        edi
00B8169C  lea         edi,[ebp-0D8h]
00B816A2  mov         ecx,36h
00B816A7  mov         eax,0CCCCCCCCh
00B816AC  rep stos    dword ptr es:[edi]
     5:     int c = a + 1;
00B816AE  mov         eax,dword ptr [a]
00B816B1  add         eax,1
00B816B4  mov         dword ptr [c],eax
     6:     int d = b + 1;
00B816B7  mov         eax,dword ptr [b]
00B816BA  add         eax,1
00B816BD  mov         dword ptr [d],eax
```

지역 변수 공간 미리 확보



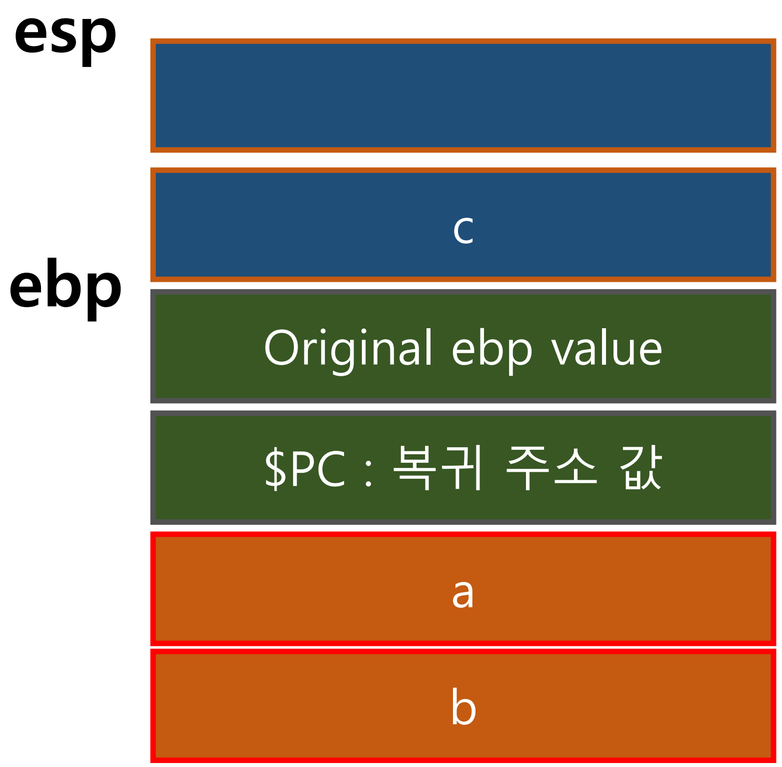
Stack frame

STACK에 쌓이는 순서



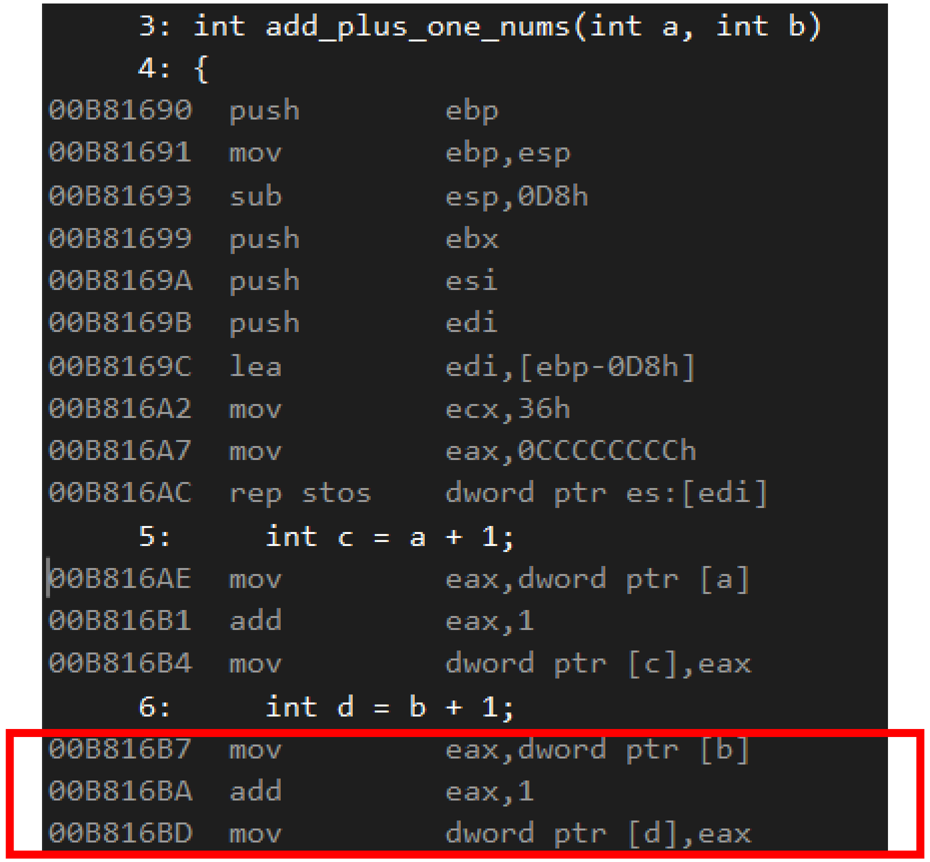
```
     3: int add_plus_one_nums(int a, int b)
     4: {
00B81690  push        ebp
00B81691  mov         ebp,esp
00B81693  sub         esp,0D8h
00B81699  push        ebx
00B8169A  push        esi
00B8169B  push        edi
00B8169C  lea         edi,[ebp-0D8h]
00B816A2  mov         ecx,36h
00B816A7  mov         eax,0CCCCCCCCh
00B816AC  rep stos    dword ptr es:[edi]
     5:     int c = a + 1;
00B816AE  mov         eax,dword ptr [a]
00B816B1  add         eax,1
00B816B4  mov         dword ptr [c],eax
     6:     int d = b + 1;
00B816B7  mov         eax,dword ptr [b]
00B816BA  add         eax,1
00B816BD  mov         dword ptr [d],eax
```

지역 변수 공간 미리 확보



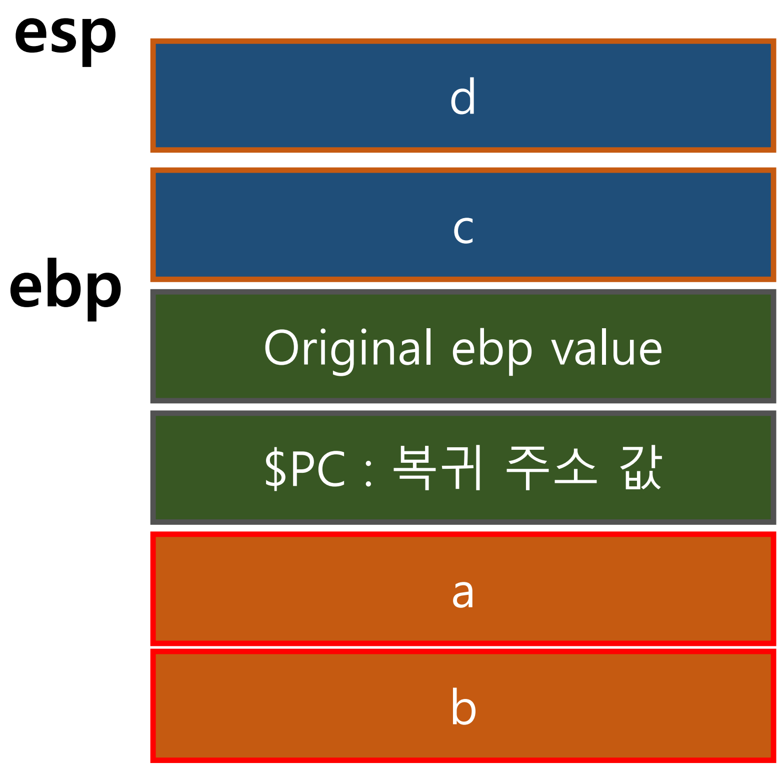
Stack frame

STACK에 쌓이는 순서



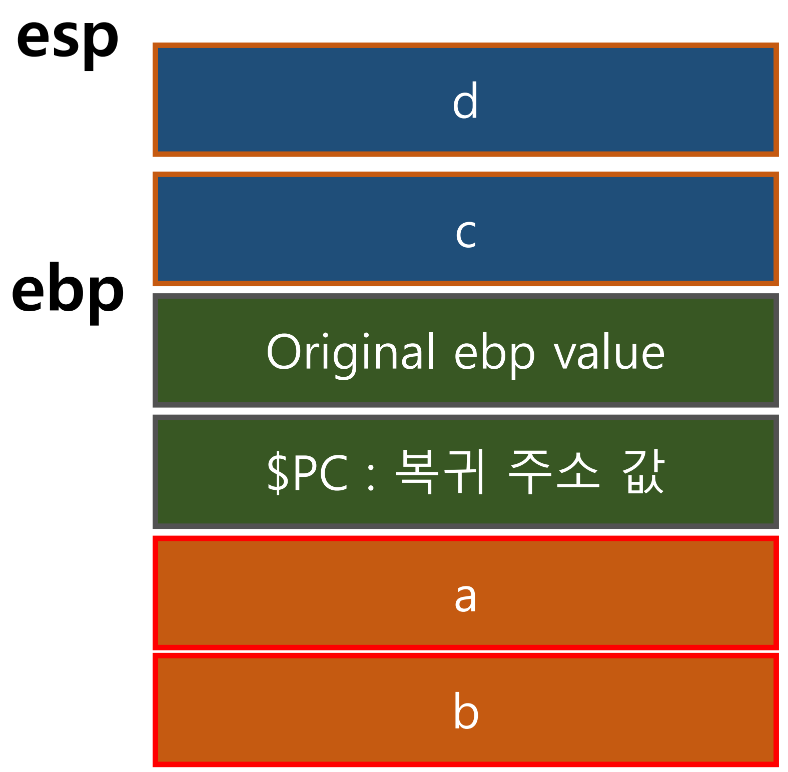
```
     3: int add_plus_one_nums(int a, int b)
     4: {
00B81690  push        ebp
00B81691  mov         ebp,esp
00B81693  sub         esp,0D8h
00B81699  push        ebx
00B8169A  push        esi
00B8169B  push        edi
00B8169C  lea         edi,[ebp-0D8h]
00B816A2  mov         ecx,36h
00B816A7  mov         eax,0CCCCCCCCh
00B816AC  rep stos    dword ptr es:[edi]
     5:     int c = a + 1;
00B816AE  mov         eax,dword ptr [a]
00B816B1  add         eax,1
00B816B4  mov         dword ptr [c],eax
     6:     int d = b + 1;
00B816B7  mov         eax,dword ptr [b]
00B816BA  add         eax,1
00B816BD  mov         dword ptr [d],eax
```

지역 변수 공간 미리 확보



Stack frame

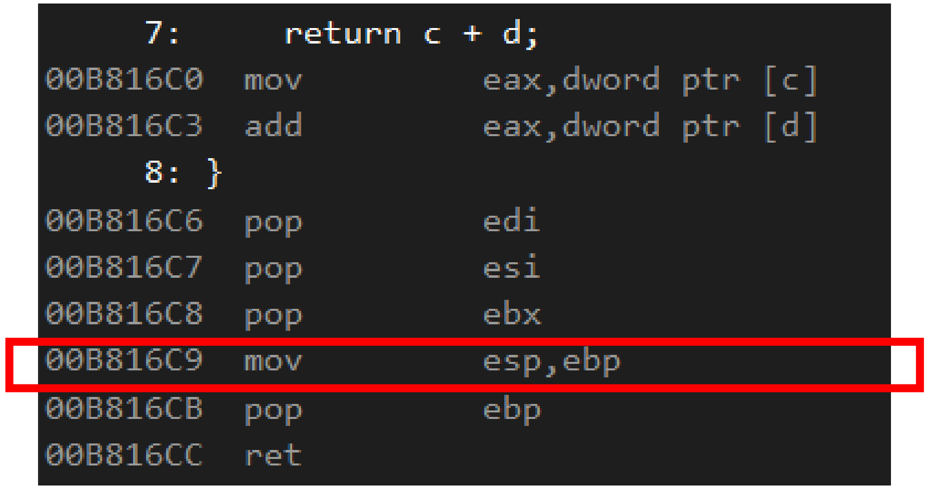
STACK에 쌓이는 순서



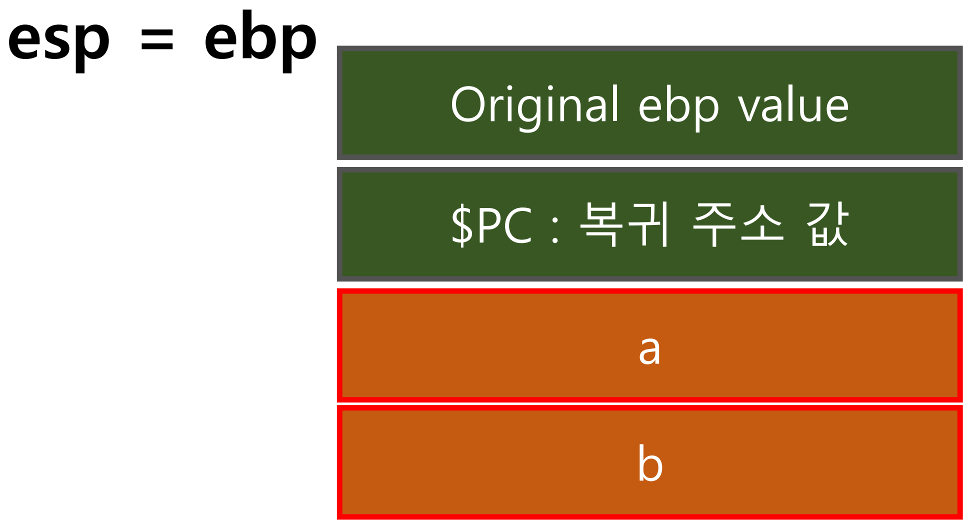
Stack frame

함수 호출이 끝나고 STACK에서 해지되는 순서

실제 지우지는 않지만 stack pointer가 움직인 것이 결국엔 해지

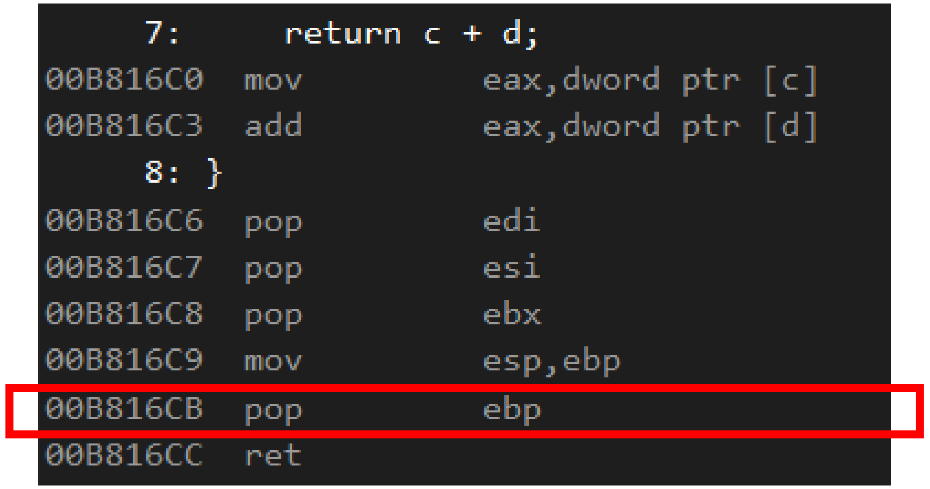


```
      7:      return c + d;
00B816C0   mov          eax,dword ptr [c]
00B816C3   add          eax,dword ptr [d]
      8: }
00B816C6   pop          edi
00B816C7   pop          esi
00B816C8   pop          ebx
00B816C9   mov          esp,ebp
00B816CB   pop          ebp
00B816CC   ret
```

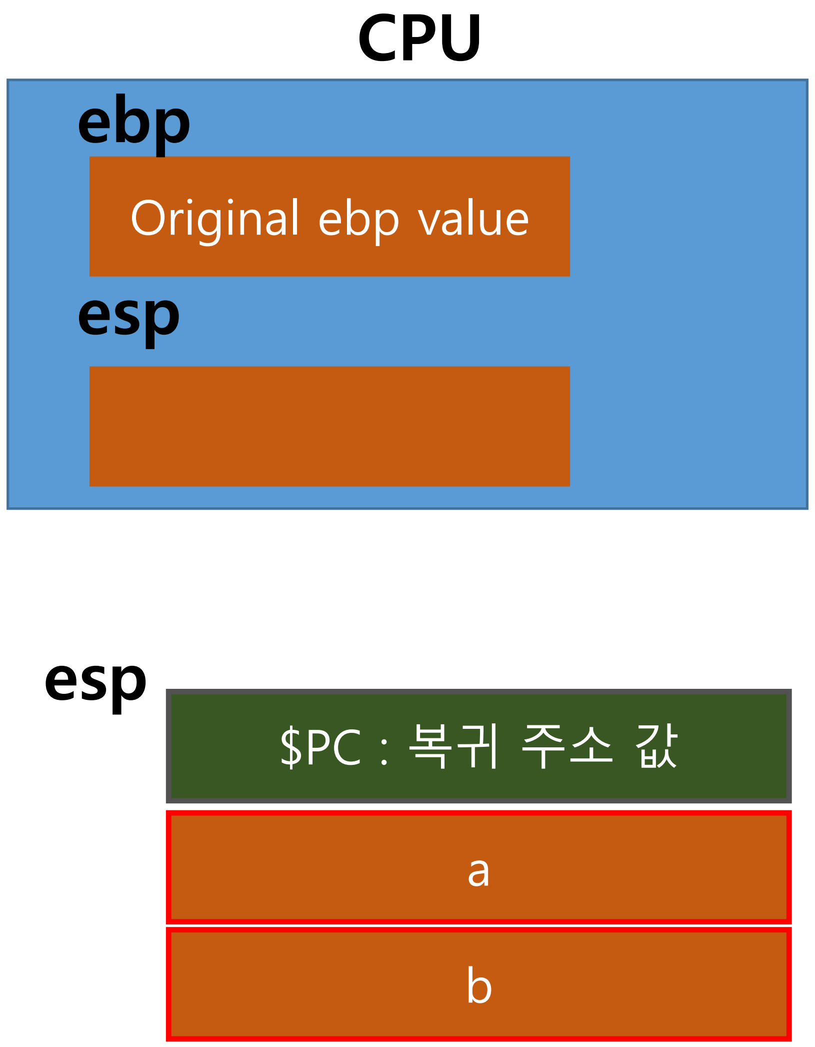


Stack frame

함수 호출이 끝나고 STACK에서 해지되는 순서

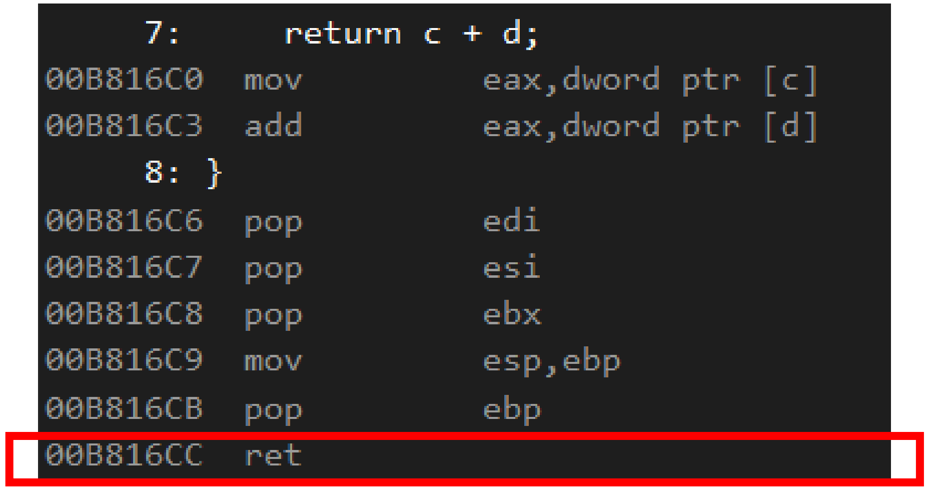


```
      7:      return c + d;
00B816C0   mov          eax,dword ptr [c]
00B816C3   add          eax,dword ptr [d]
      8: }
00B816C6   pop          edi
00B816C7   pop          esi
00B816C8   pop          ebx
00B816C9   mov          esp,ebp
00B816CB   pop          ebp
00B816CC   ret
```



Stack frame

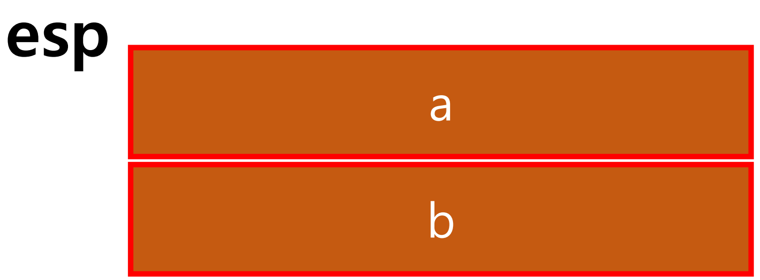
함수 호출이 끝나고 STACK에서 해지되는 순서



```
      7:       return c + d;
00B816C0   mov          eax,dword ptr [c]
00B816C3   add          eax,dword ptr [d]
      8: }
00B816C6   pop          edi
00B816C7   pop          esi
00B816C8   pop          ebx
00B816C9   mov          esp,ebp
00B816CB   pop          ebp
00B816CC   ret
```

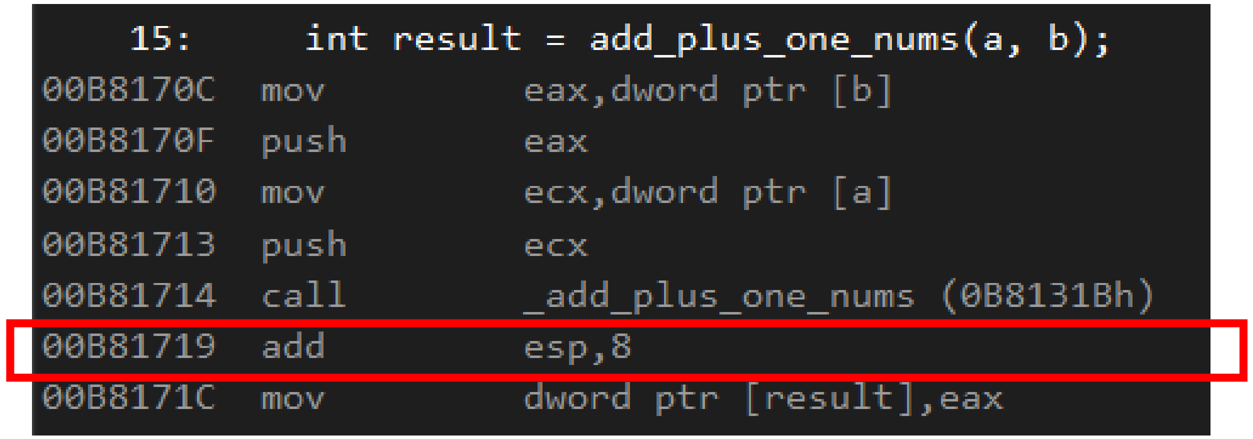
$PC : 복귀 주소 값

ret instruction은 복귀 주소값으로 다시 돌아가 함수 호출 이후부터 실행

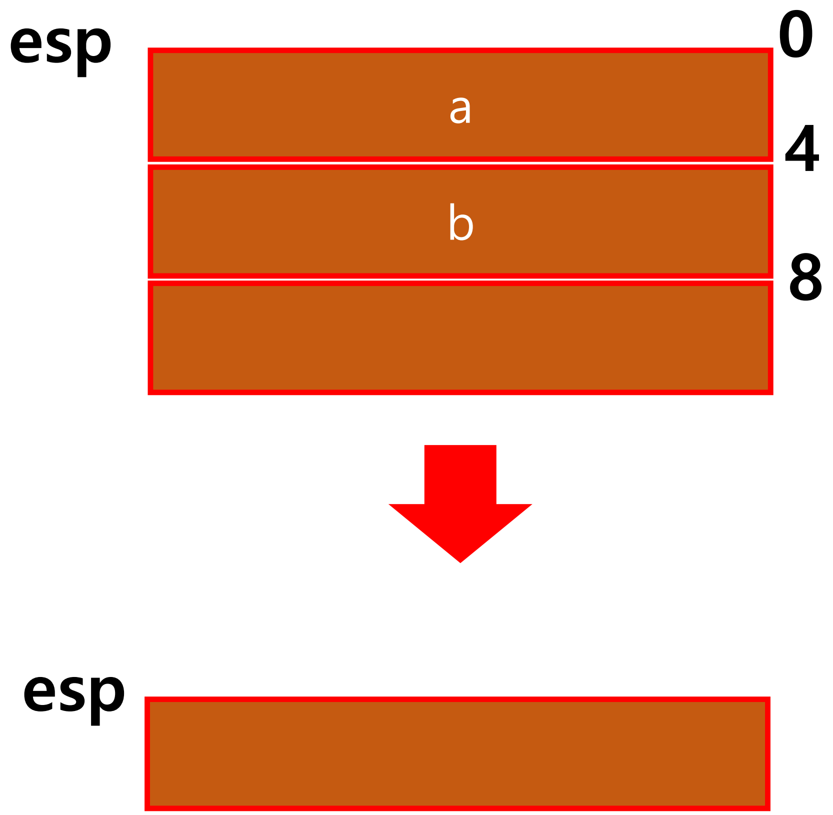


Stack frame

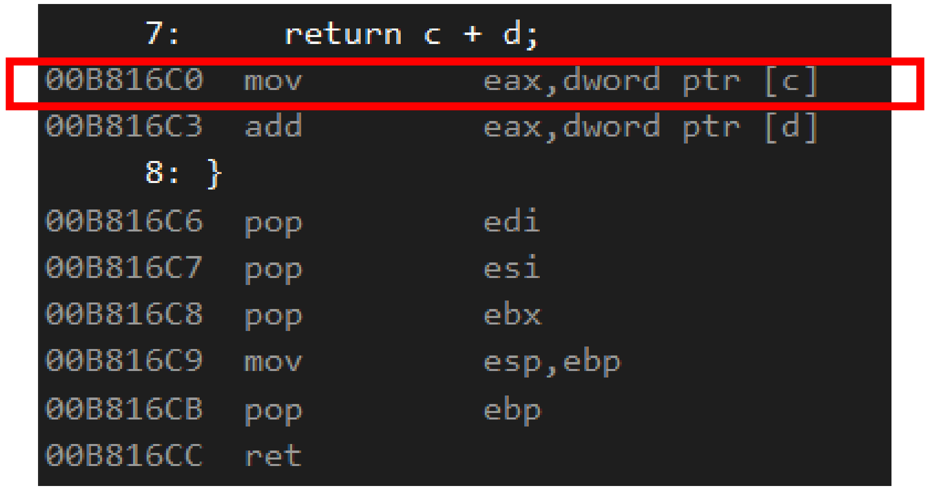
함수 호출이 끝나고 STACK에서 해지되는 순서



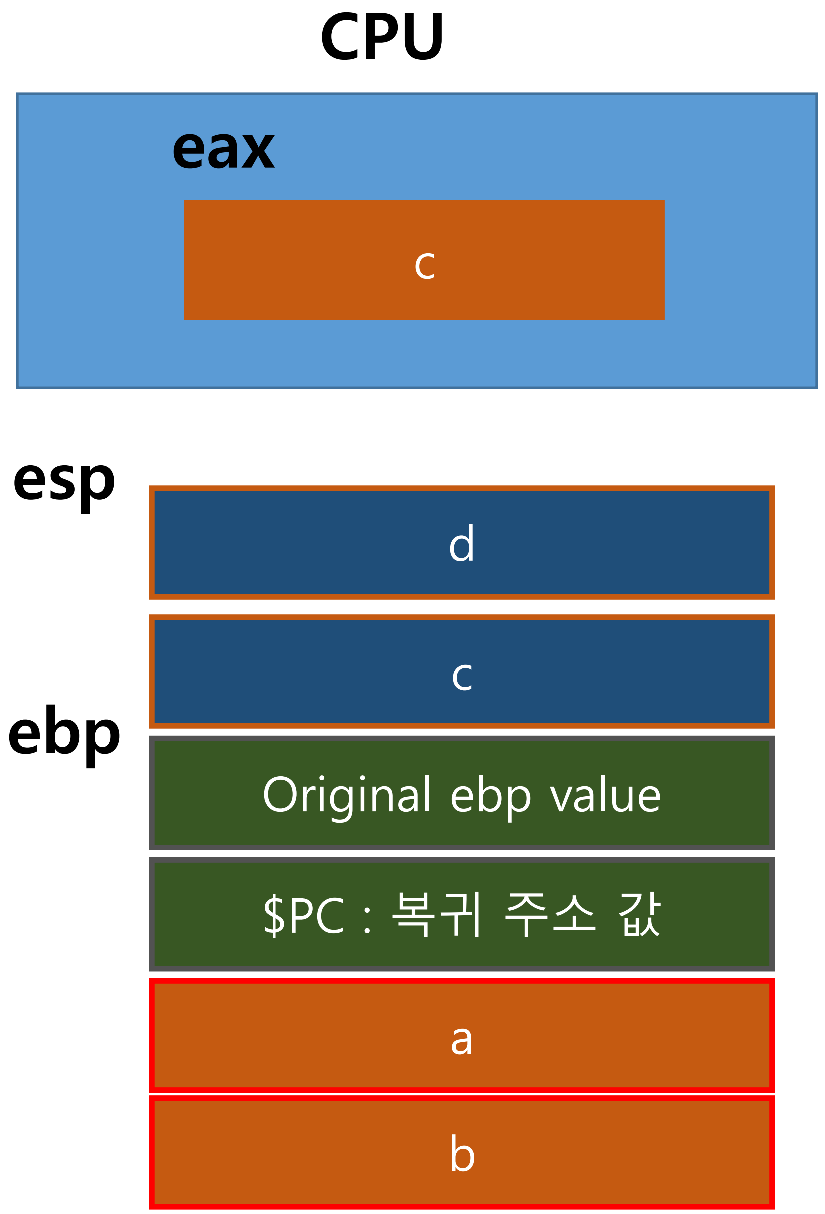
```
   15:      int result = add_plus_one_nums(a, b);
00B8170C  mov          eax,dword ptr [b]
00B8170F  push         eax
00B81710  mov          ecx,dword ptr [a]
00B81713  push         ecx
00B81714  call         _add_plus_one_nums (0B8131Bh)
00B81719  add          esp,8
00B8171C  mov          dword ptr [result],eax
```



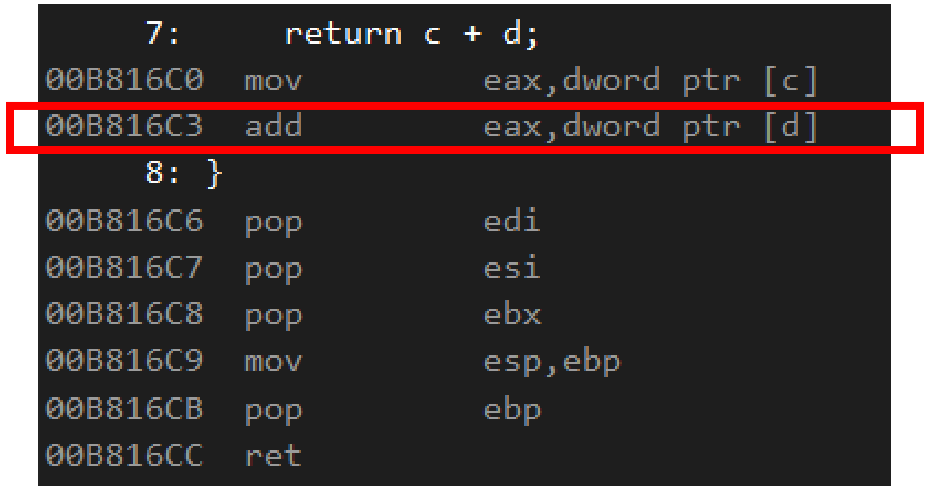
return의 의미



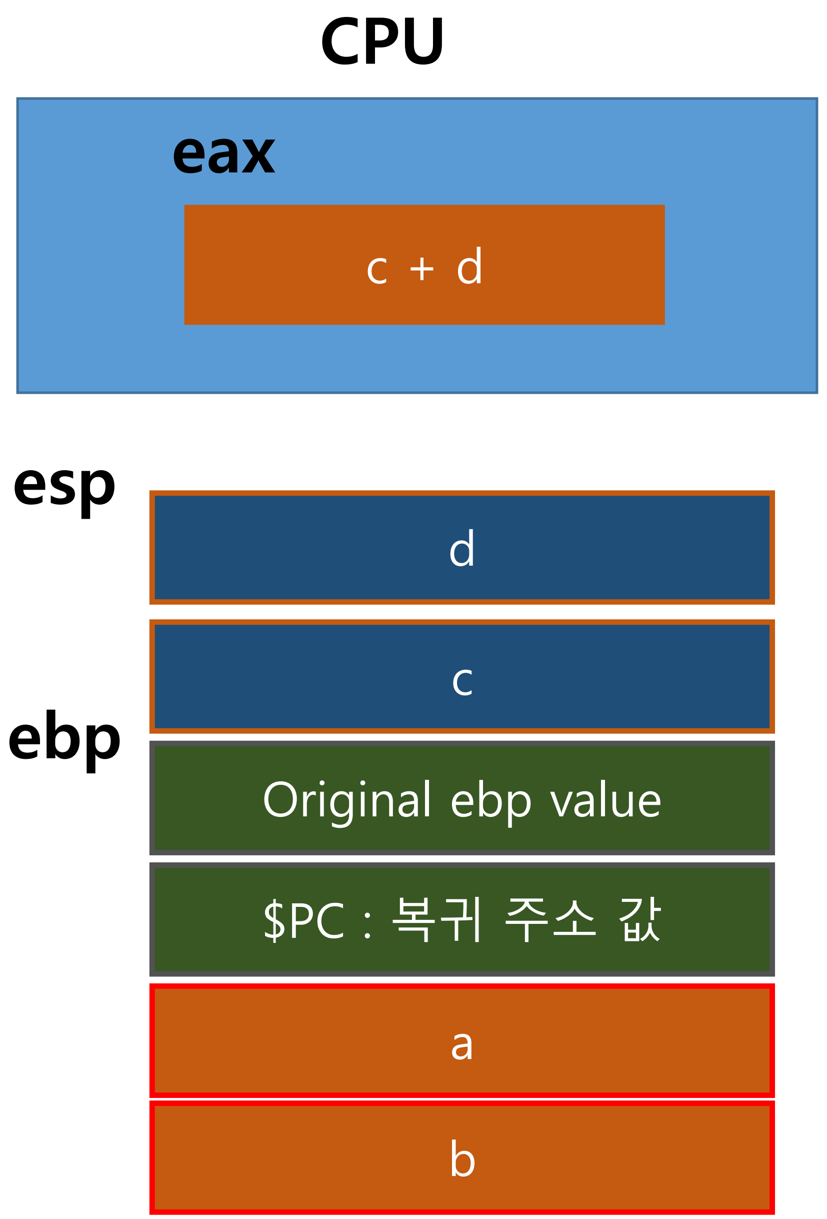
```
     7:      return c + d;
00B816C0  mov          eax,dword ptr [c]
00B816C3  add          eax,dword ptr [d]
     8: }
00B816C6  pop          edi
00B816C7  pop          esi
00B816C8  pop          ebx
00B816C9  mov          esp,ebp
00B816CB  pop          ebp
00B816CC  ret
```



return의 의미

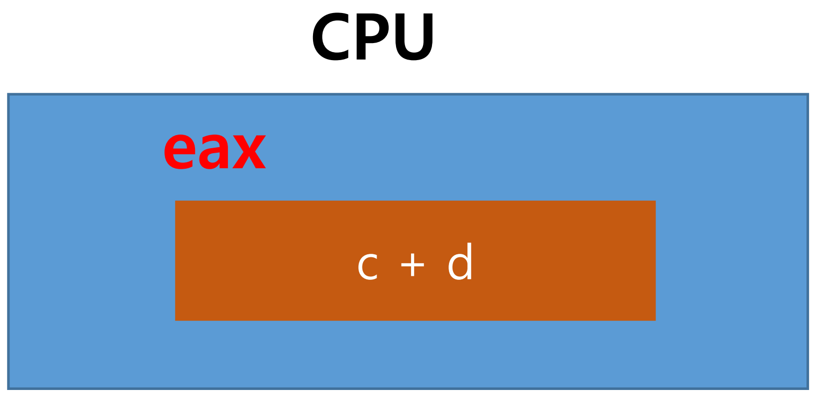


```
     7:       return c + d;
00B816C0  mov          eax,dword ptr [c]
00B816C3  add          eax,dword ptr [d]
     8: }
00B816C6  pop          edi
00B816C7  pop          esi
00B816C8  pop          ebx
00B816C9  mov          esp,ebp
00B816CB  pop          ebp
00B816CC  ret
```



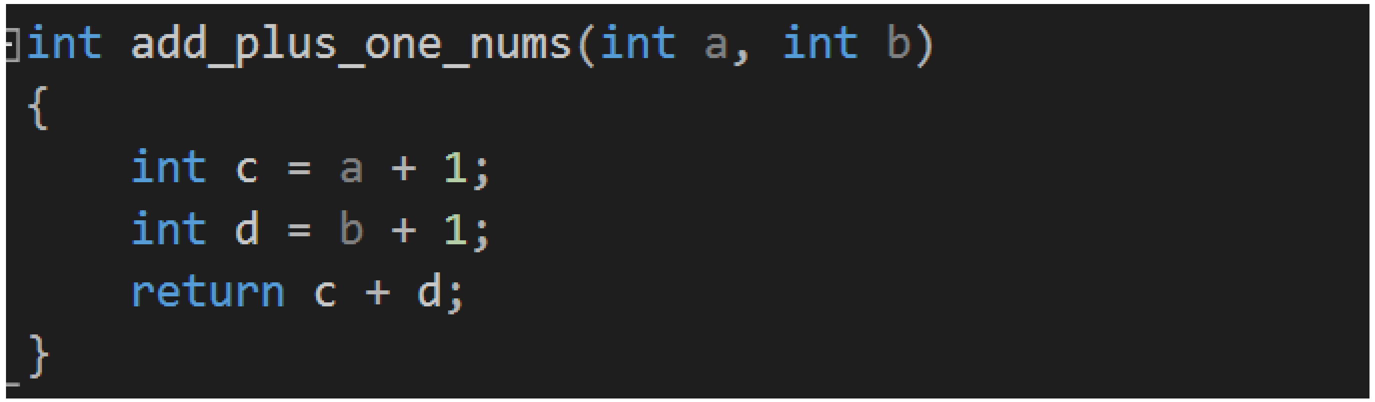
return의 의미

eax가 return 값이 저장되어 있는 레지스터
리턴은 반드시 eax를 거쳐서 한다.



assembly

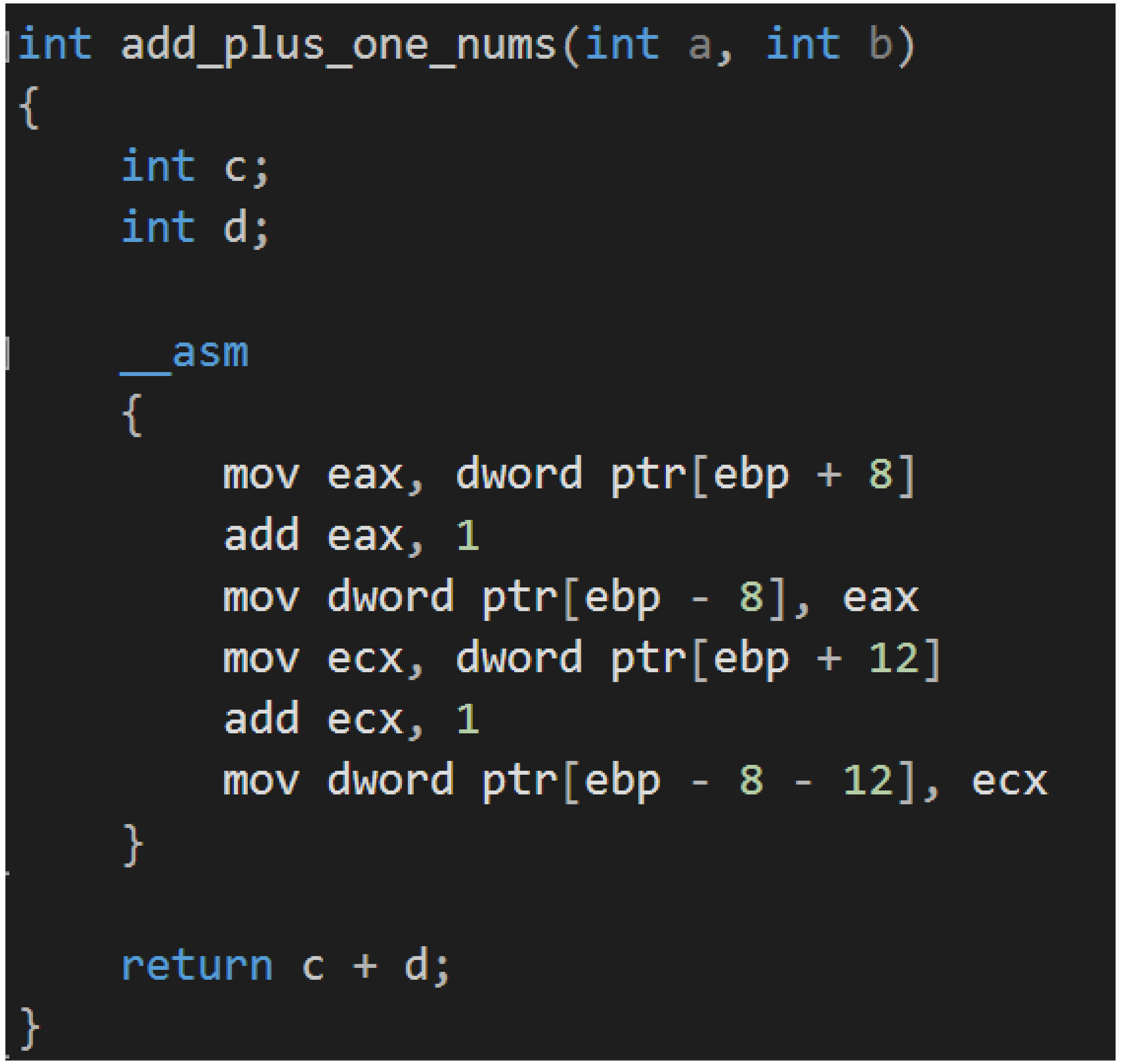
C 버전



```c
int add_plus_one_nums(int a, int b)
{
    int c = a + 1;
    int d = b + 1;
    return c + d;
}
```

assembly

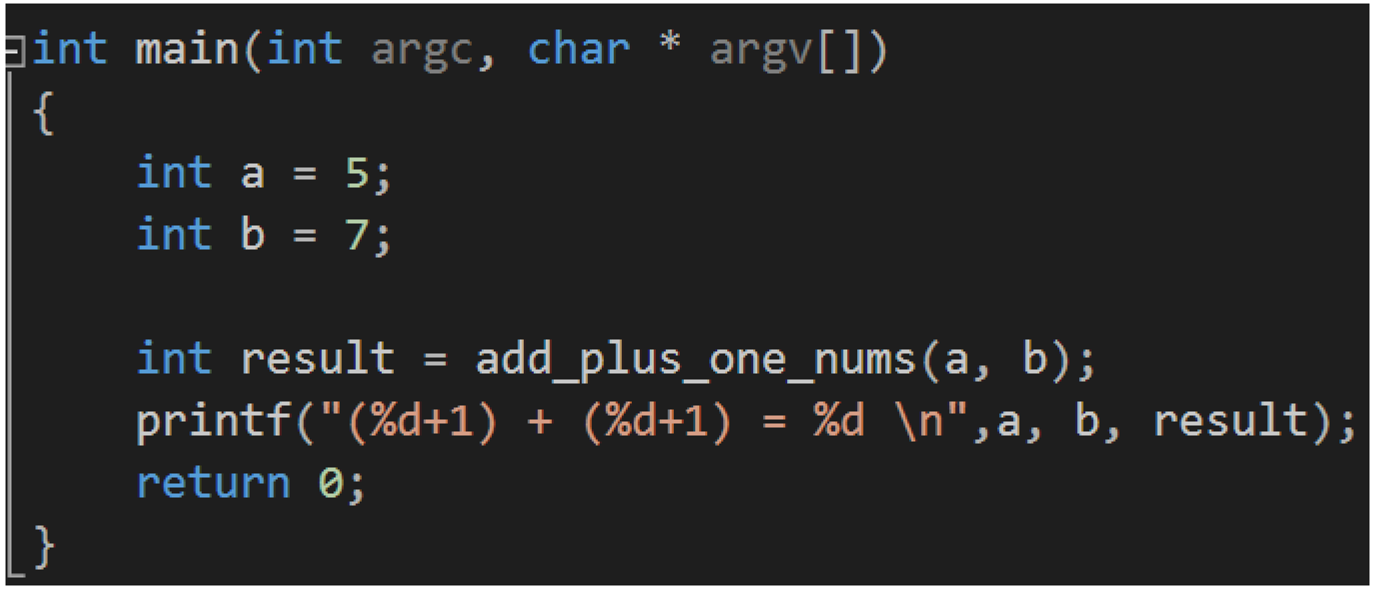
C + assembly 버전



```
int add_plus_one_nums(int a, int b)
{
    int c;
    int d;

    __asm
    {
        mov eax, dword ptr[ebp + 8]
        add eax, 1
        mov dword ptr[ebp - 8], eax
        mov ecx, dword ptr[ebp + 12]
        add ecx, 1
        mov dword ptr[ebp - 8 - 12], ecx
    }

    return c + d;
}
```

assembly

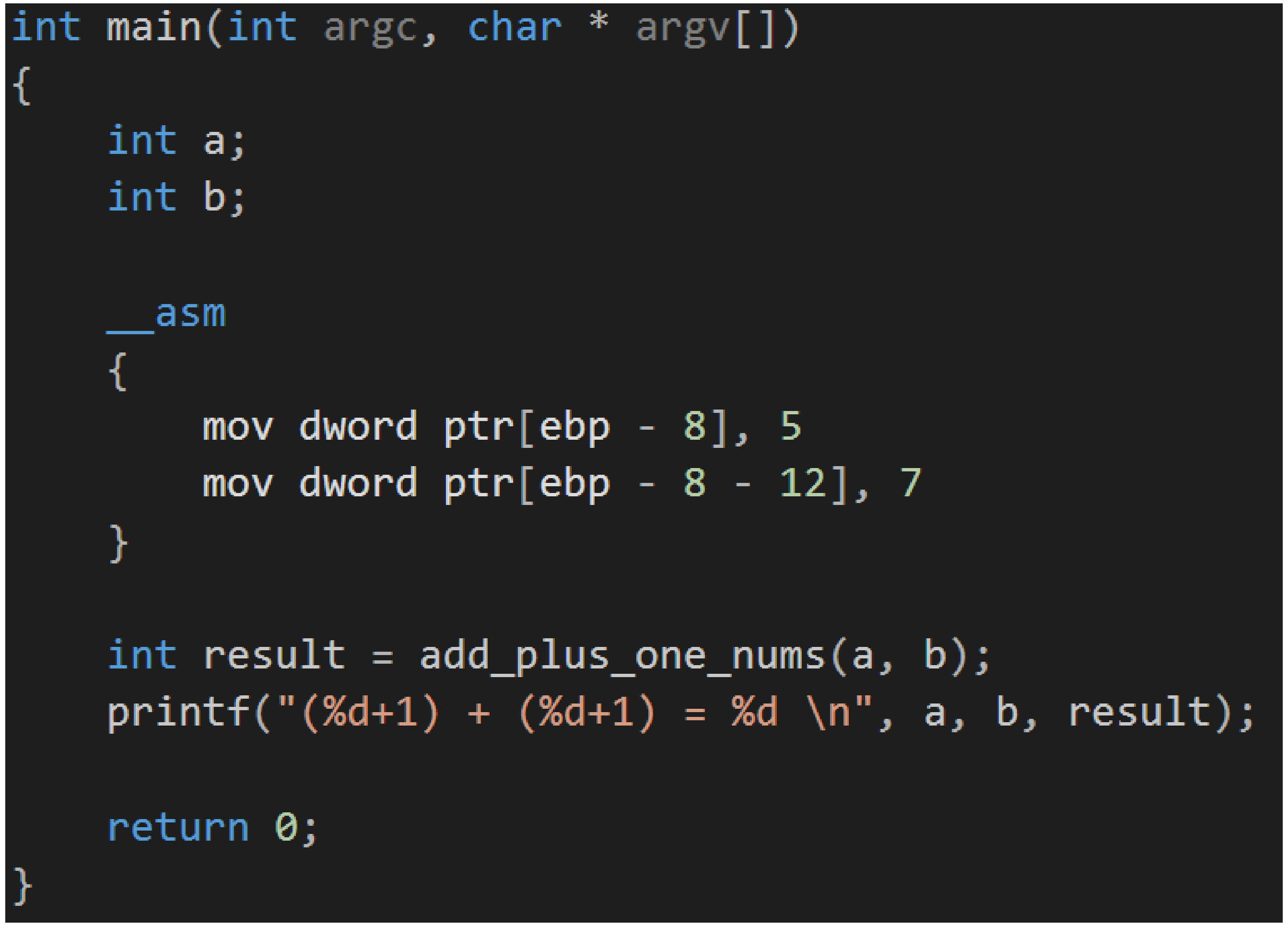
C 버전



```c
int main(int argc, char * argv[])
{
    int a = 5;
    int b = 7;

    int result = add_plus_one_nums(a, b);
    printf("(%d+1) + (%d+1) = %d \n",a, b, result);
    return 0;
}
```

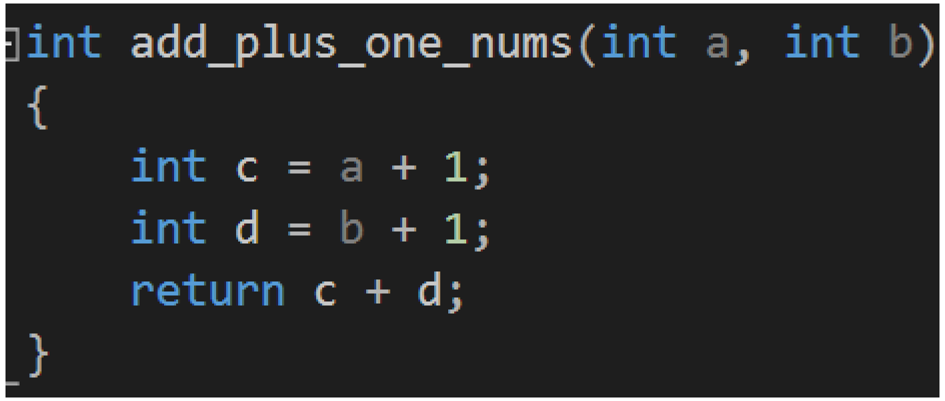
assembly

C + assembly 버전



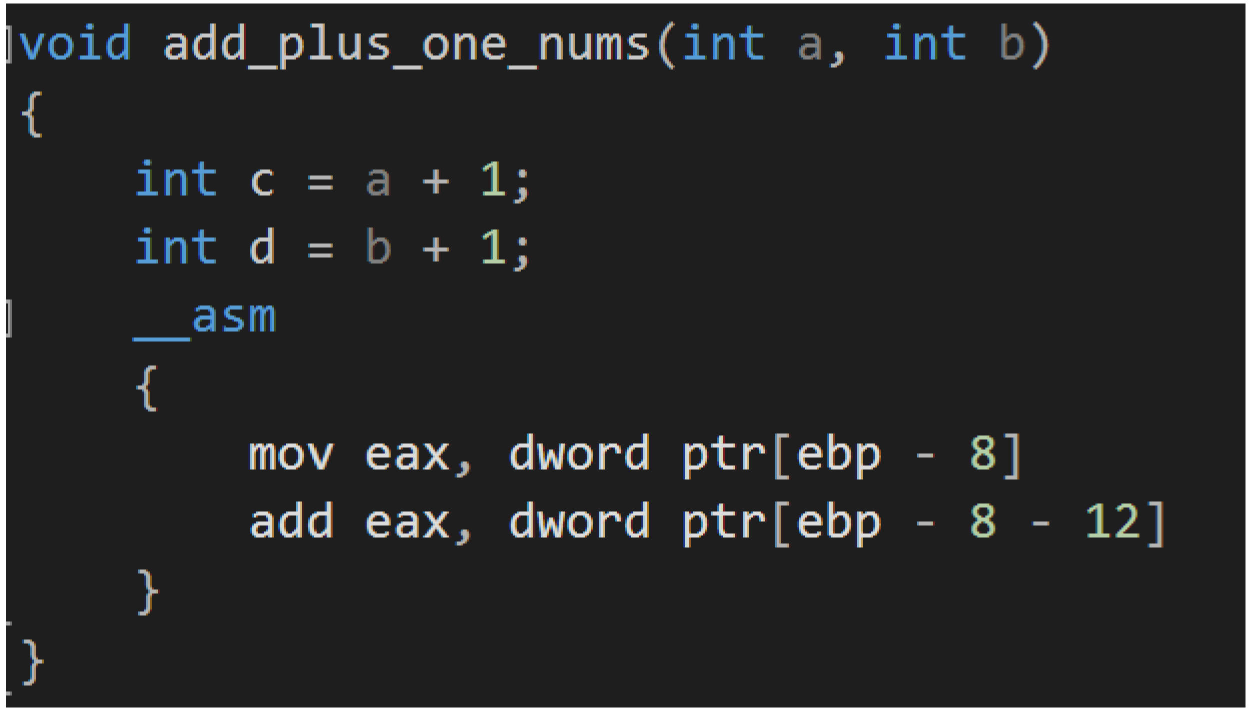
```
int main(int argc, char * argv[])
{
    int a;
    int b;

    __asm
    {
        mov dword ptr[ebp - 8], 5
        mov dword ptr[ebp - 8 - 12], 7
    }

    int result = add_plus_one_nums(a, b);
    printf("(%d+1) + (%d+1) = %d \n", a, b, result);

    return 0;
}
```

리턴 없이 리턴 값 가져다 쓰기



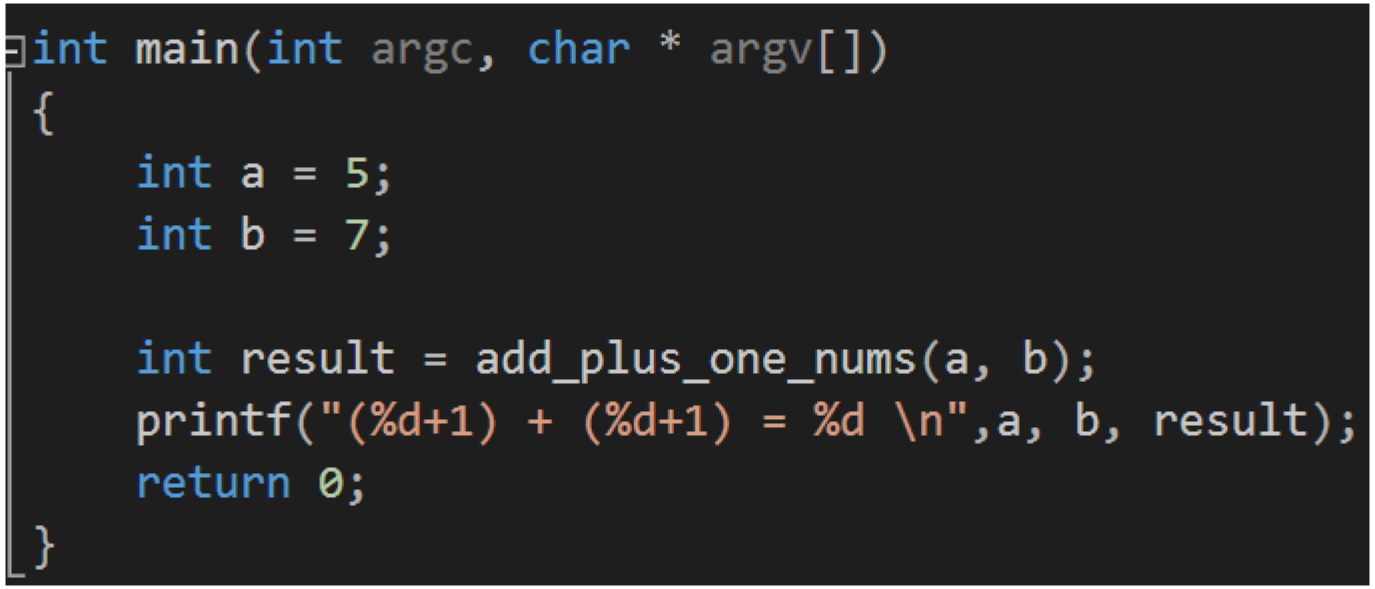
```
int add_plus_one_nums(int a, int b)
{
    int c = a + 1;
    int d = b + 1;
    return c + d;
}
```

리턴 없이 리턴 값 가져다 쓰기



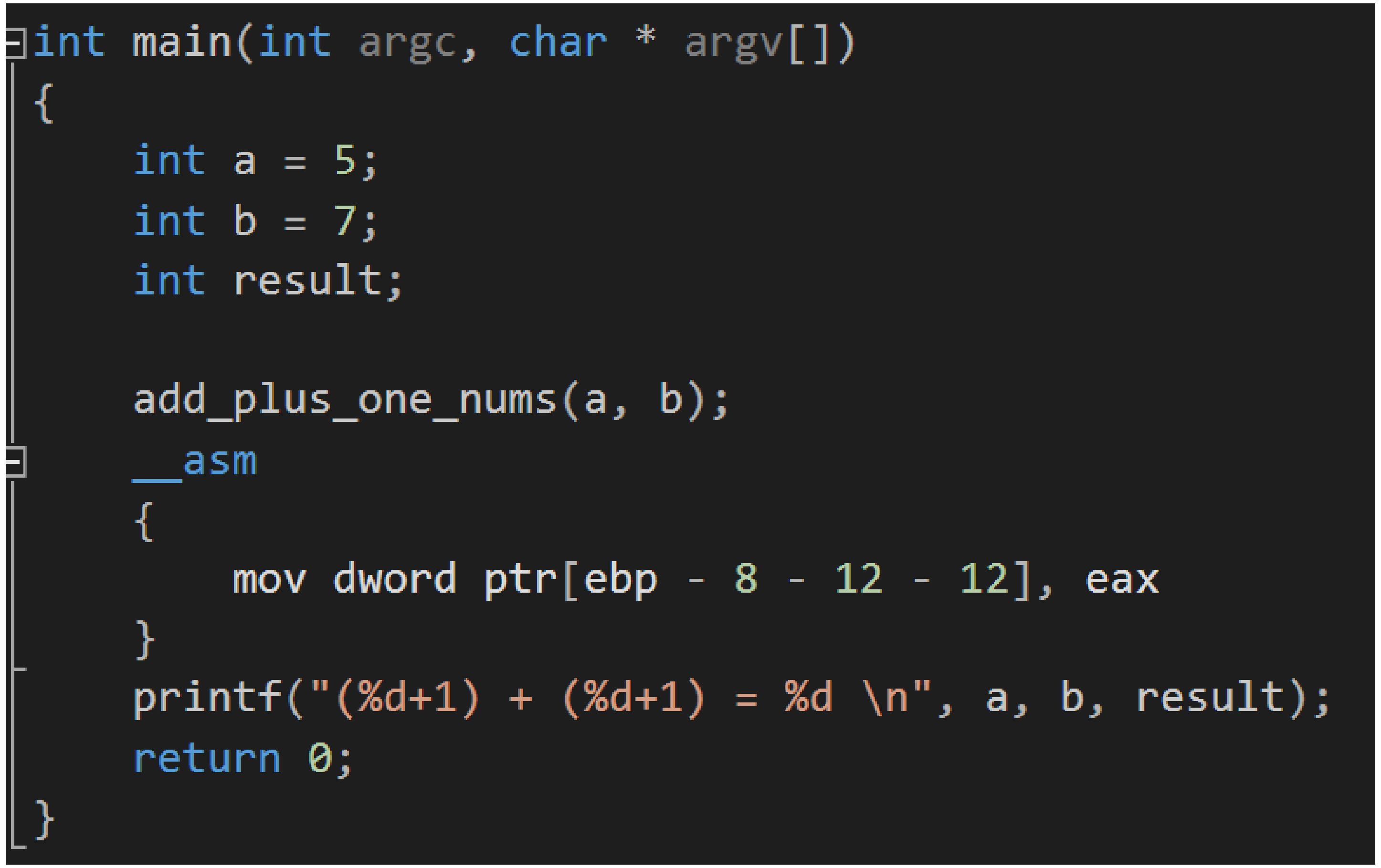
```
void add_plus_one_nums(int a, int b)
{
    int c = a + 1;
    int d = b + 1;
    __asm
    {
        mov eax, dword ptr[ebp - 8]
        add eax, dword ptr[ebp - 8 - 12]
    }
}
```

리턴 없이 리턴 값 가져다 쓰기



```c
int main(int argc, char * argv[])
{
    int a = 5;
    int b = 7;

    int result = add_plus_one_nums(a, b);
    printf("(%d+1) + (%d+1) = %d \n",a, b, result);
    return 0;
}
```

리턴 없이 리턴 값 가져다 쓰기



```
int main(int argc, char * argv[])
{
    int a = 5;
    int b = 7;
    int result;

    add_plus_one_nums(a, b);
    __asm
    {
        mov dword ptr[ebp - 8 - 12 - 12], eax
    }
    printf("(%d+1) + (%d+1) = %d \n", a, b, result);
    return 0;
}
```