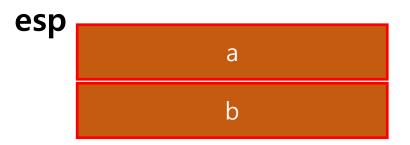
```
jint add_plus_one_nums(int a, int b)
    int c = a + 1;
    int d = b + 1;
    return c + d;
jint main(int argc, char * argv[])
    int a = 5;
    int b = 7;
    int result = add_plus_one_nums(a, b);
    return 0;
```



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

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

15:	int resul	t = 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



15:	int resul	t = add_plus_one_nums(a, b);	
00B8170C	mov	eax, dword ptr [b] esp	
00B8170F	push	eax	\$PC : 복귀 주소 값
00B81710	mov	ecx,dword ptr [a]	\$PC . 흑기 구오 없
00B81713	push	ecx	
00B81714	call	_add_plus_one_nums (0B8131Bh) 함수 호출	a
00B81719	add	esp,8	
00B8171C	mov	dword ptr [result],eax	b

### STACK에 쌓이는 순서

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

esp extended base pointer
Original ebp value
\$PC : 복귀 주소 값

a
b

### STACK에 쌓이는 순서

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

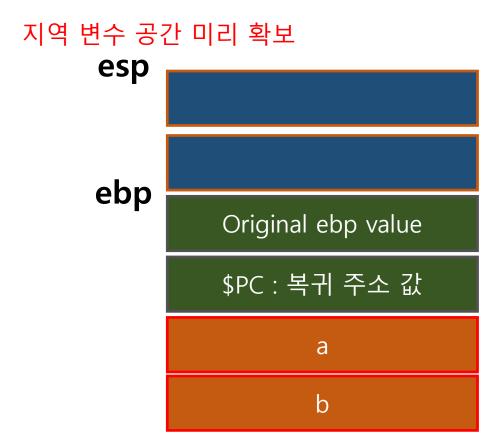
ebp : extended base pointer

ebp = esp
Original ebp value

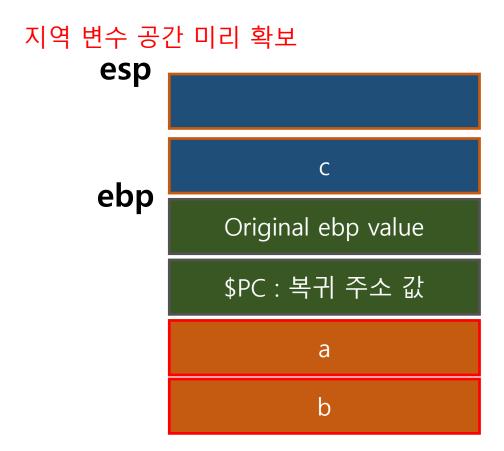
\$PC : 복귀 주소 값

a
b

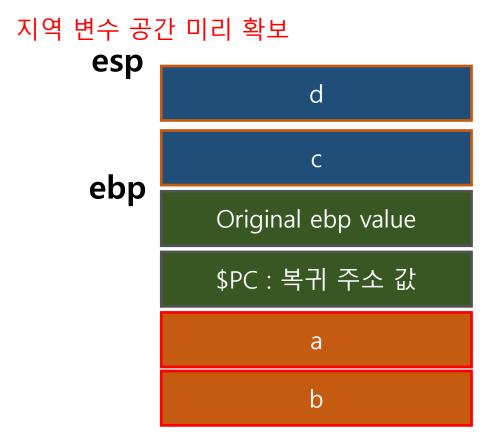
```
3: int add_plus_one_nums(int a, int b)
    4: {
00B81690 push
                     ebp
00B81691
                     ebp,esp
         mov
00B81693 sub
                     esp,0D8h
00B81699 push
                     ebx
00B8169A
         push
                     esi
                     edi
00B8169B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

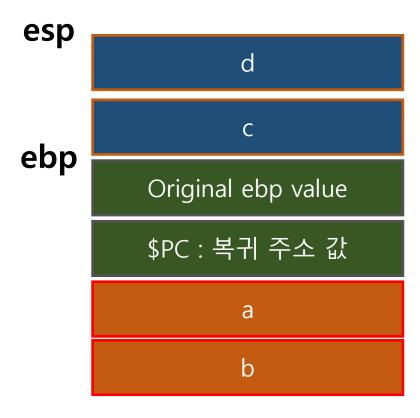


```
3: int add_plus_one_nums(int a, int b)
     4: {
00B81690
          push
                      ebp
00B81691
                      ebp,esp
          mov
00B81693
                      esp,0D8h
         sub
00B81699
          push
00B8169A
          push
                      esi
                 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



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

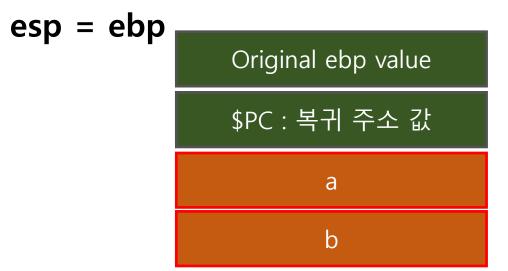




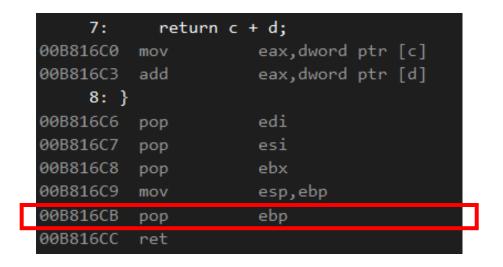
## 함수 호출이 끝나고 STACK에서 해지되는 순서

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

실제 지우지는 않지만 stack pointer가 움직인 것이 결국엔 해지



## 함수 호출이 끝나고 STACK에서 해지되는 순서

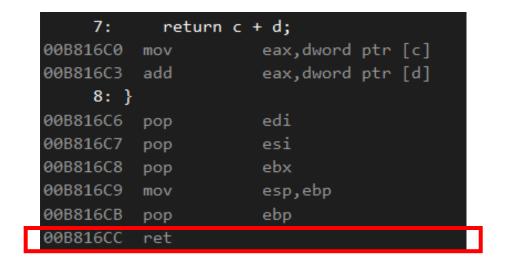


## **CPU**



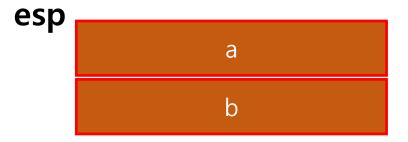


## 함수 호출이 끝나고 STACK에서 해지되는 순서



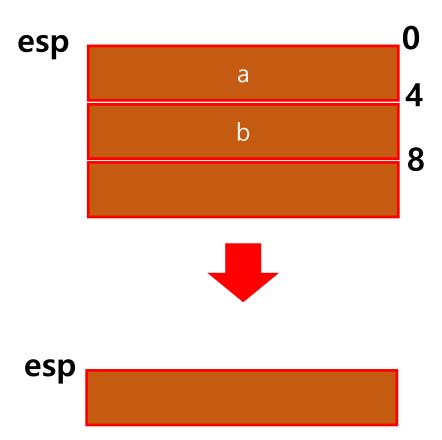
\$PC : 복귀 주소 값

ret instruction은 복귀 주소값으로 다시 돌아가 함수 호출 이후부터 실행

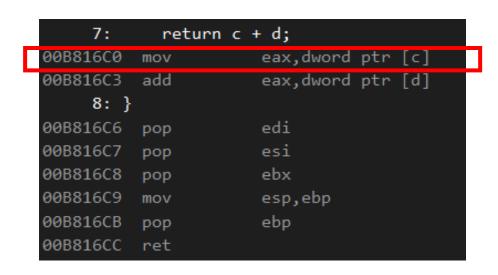


# 함수 호출이 끝나고 STACK에서 해지되는 순서

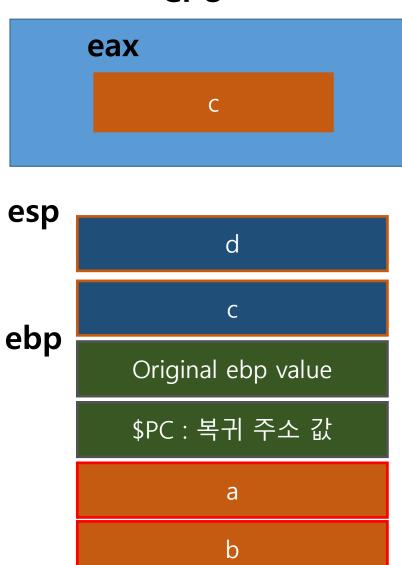
15:	int res	ult = add_plus_one_nums(a, b);
00B81700	mov	eax,dword ptr [b]
00B8170	- push	eax
00B81710	mov mov	ecx,dword ptr [a]
00B8171	3 push	ecx
00B81714	1 call	_add_plus_one_nums (0B8131Bh)
00B81719	add a	esp,8
00B81710	C mov	dword ptr [result],eax



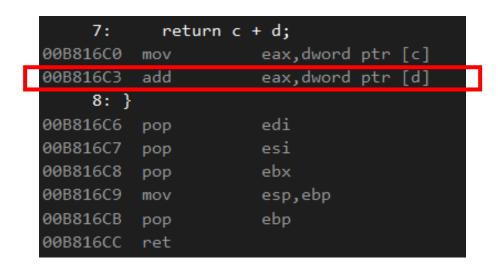
## return의 의미



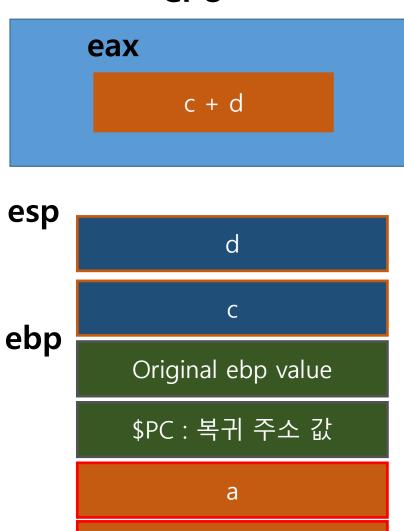
# **CPU**



## return의 의미

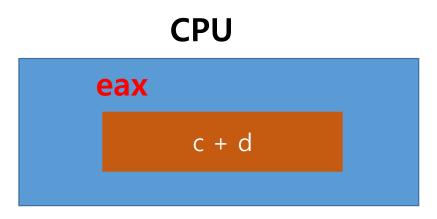


# **CPU**



b

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



assembly

C 버전

```
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

```
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;
}
```

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;
```

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

```
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]
    }
}
```

```
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;
}
```

```
∃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;
```