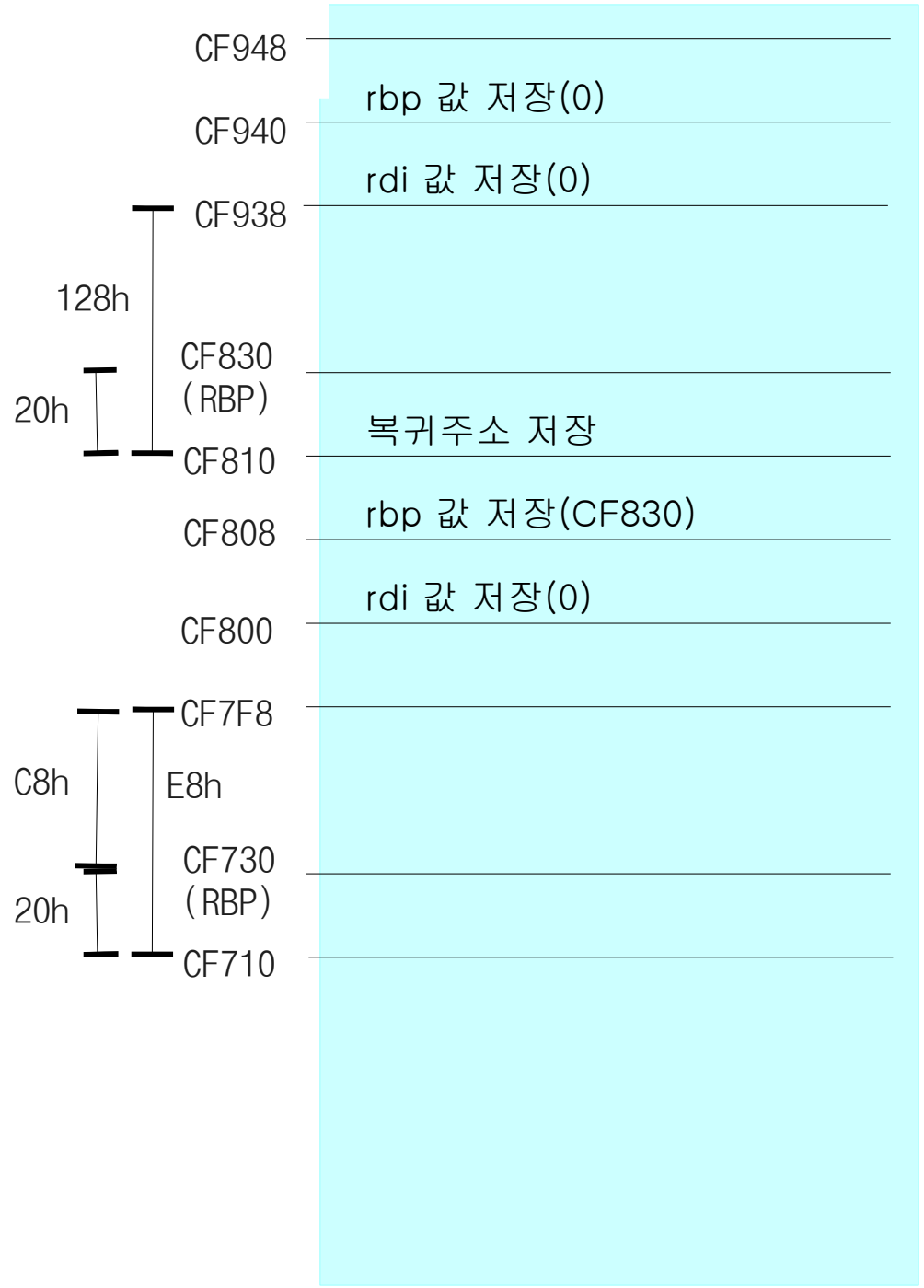
0. RSP = 00000069685CF948
1. RSP = 00000069685CF940
   rbp(00000)값을 rsp 메모리에 저장
2. RSP = 00000069685CF938
   Rdi(000-) 값을 rsp 메모리에 저장
3. RSP = 00000069685CF810
   16진수 ~CF938 에서 128h 를 뺀다.
   (Stack 공간을 확보,지역변수공간생성)
4. RSP = 00000069685CF810
   RBP 에 RSP(~CF810)+20 값을 넣어준다.
   RBP = 00000069685CF830
6. RSP = 00000069685CF810
  Call로 인해서 점프하고
  RSP ~CF808의 메모리에 7번 주소가 저장된다.
7. input_parameter 에 10을 대입(0A)
  &input_parameter = 0x00000069685CF854
  0x00000069685CF854 = 0000000A
8. input_paramter 값을 ecx 에 저장
   RCX = 000000000000000A
9. RSP = 00000069685CF808 에 복귀주소를 저장하고 점프한다.

```
int main(void)
{
1 00007FF680AE46B0  push        rbp
2 00007FF680AE46B2  push        rdi
3 00007FF680AE46B3  sub         rsp,128h
4 00007FF680AE46BA  lea         rbp,[rsp+20h]
5 00007FF680AE46BF  lea         rcx,[__44B22296_main@c (07FF680AF1008h)]
6 00007FF680AE46C6  call        __CheckForDebuggerJustMyCode (07FF680AE1357h)
    int return_value;

    int input_parameter = 10;
7 00007FF680AE46CB  mov         dword ptr [input_parameter],0A
    return_value = for_assembly_function_test(input_parameter
8 00007FF680AE46D2  mov         ecx,dword ptr [input_parameter
9 00007FF680AE46D5  call        for_assembly_function_test (07FF680AE11C2h)
00007FF680AE46DA  mov         dword ptr [return_value],eax
    printf("return_value = %d₩n", return_value);
00007FF680AE46DD  mov         edx,dword ptr [return_value]
00007FF680AE46E0  lea         rcx,[string "return_value = %d₩n" (07FF680AE9D78
00007FF680AE46E7  call        printf (07FF680AE13BBh)

    return 0;
00007FF680AE46EC  xor         eax,eax
}
```

0. RSP = 00000069685CF808
   RSP = 00000069685CF808 에는 복귀주소가 들어있음.
1. ecx 의 값을 ~CF810에 저장
2. rbp 값을 RSP = 00000069685CF808 메모리에 저장
3. rdi 값을 RSP = 00000069685CF800 메모리에 저장
4. RSP = 00000069685CF7F8 에서 E8h 를 뺀다.
5. RBP 에 RSP = 00000069685CF710 에서 20h 를 더한 값을 넣어준다.

8. number 의 값을 eax에 대입
9. eax 값을 2진수기준 왼쪽으로 한칸 이동(x2)
10. RSP에 C8h 만큼 더해줌으로 E8h sub 했던 곳으로 복귀
11. rdi pop (rsp + 8)
12. rbp pop (rsp + 8)
13. ret

```
int for_assembly_function_test(int number)
{
1 00007FF680AE17C0  mov         dword ptr [rsp+8],ecx
2 00007FF680AE17C4  push        rbp
3 00007FF680AE17C5  push        rdi
4 00007FF680AE17C6  sub         rsp,0E8h
5 00007FF680AE17CD  lea         rbp,[rsp+20h]
6 00007FF680AE17D2  lea         rcx,[__44B22296_main@c (07FF680AF1008h)]
7 00007FF680AE17D9  call        __CheckForDebuggerJustMyCode (07FF680AE1357h)
    return number * 2;
8 00007FF680AE17DE  mov         eax,dword ptr [number]
9 00007FF680AE17E4  shl         eax,1
}
10 00007FF680AE17E6  lea         rsp,[rbp+0C8h]
11 00007FF680AE17ED  pop         rdi
12 00007FF680AE17EE  pop         rbp
13 00007FF680AE17EF  ret
```




```
00007FF680AE46B0  push    rbp
00007FF680AE46B2  push    rdi
00007FF680AE46B3  sub     rsp,128h
00007FF680AE46BA  lea     rbp,[rsp+20h]
00007FF680AE46BF  lea     rcx,[__44B22296_main@c (07FF680AF1008h)]
00007FF680AE46C6  call    __CheckForDebuggerJustMyCode (07FF680AE1357h)
00007FF680AE46CB  mov     dword ptr [input_parameter],0Ah
00007FF680AE46D2  mov     ecx,dword ptr [input_parameter]
00007FF680AE46D5  call    for_assembly_function_test (07FF680AE11C2h)

00007FF680AE17C0  mov     dword ptr [rsp+8],ecx
00007FF680AE17C4  push    rbp
00007FF680AE17C5  push    rdi
00007FF680AE17C6  sub     rsp,0E8h
00007FF680AE17CD  lea     rbp,[rsp+20h]
00007FF680AE17D2  lea     rcx,[__44B22296_main@c (07FF680AF1008h)]
00007FF680AE17D9  call    __CheckForDebuggerJustMyCode (07FF680AE1357h)
00007FF680AE17DE  mov     eax,dword ptr [number]
00007FF680AE17E4  shl     eax,1
00007FF680AE17E6  lea     rsp,[rbp+0C8h]
00007FF680AE17ED  pop     rdi
00007FF680AE17EE  pop     rbp
00007FF680AE17EF  ret

00007FF680AE46DA  mov     dword ptr [return_value],eax
00007FF680AE46DD  mov     edx,dword ptr [return_value]
00007FF680AE46E0  lea     rcx,[string "return_value = %d₩n" (07FF680AE9D7
00007FF680AE46E7  call    printf (07FF680AE13BBh)
00007FF680AE46EC  xor     eax,eax
```