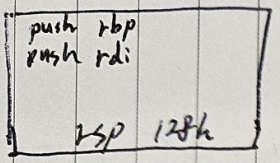
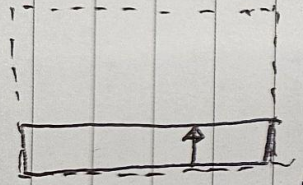
```
[1]
00007FF651CC18C0                        push    rbp                     // Stack에 rbp 저장
00007FF651CC18C2    0000005B84F5F860    push    rdi                     // Stack에 rdi 저장
00007FF651CC18C3    0000005B84F5F858    sub     rsp,128h                // rsp 128h Stack 공간 확보

[2]
00007FF651CC18CA    0000005B84F5F730    rbp,[rsp+20h]
00007FF651CC18E5    0000005B84F5F730    call    for_assembly_function_test  // jmp 이후 다음 mov 열로 복귀

[3]
for_assembly_function_test
00007FF651CC1884    000000A488BBF628    push    rbp                     // Stack에 rbp 저장
00007FF651CC1885    000000A488BBF620    push    rdi                     // Stack에 rdi 저장
00007FF651CC1886    000000A488BBF618    sub     rsp,0E8h                // rsp를 0E8h(232) stack 공간 확보

[4]
00007FF651CC188D    000000A488BBF530    lea     rbp,[rsp+20h]           // rsp에 20h(32)을 더한 주소를 rbp에 저장

[5]
00007FF651CC18A6    000000A488BBF530    lea     rsp,[rbp+0C8h]          // rbp에 0C8h(200)를 더한 주소를 rsp에 저장
                                                                            (0E8h 확보 하기 이전 주소)

[6]
00007FF651CC18AD    000000A488BBF618    pop     rdi                     // rdi를 sp 에서 빼냄
00007FF651CC18AE    000000A488BBF620    pop     rbp                     // rbp를 sp 에서 빼냄
00007FF651CC1907    000000A488BBF628    ret                             // 복귀
```

① 

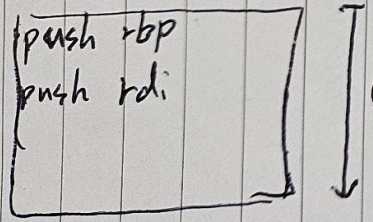
rsp, 128h
// rsp 128h stack 공간확보

② 

rbp, [rsp+20h]
// rsp에 20h를 더한 주소를 rbp에 저장

※ 복귀주소

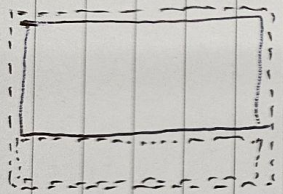
call ... 함수로 jmp 이후 다음 mov 명로 복귀

③ 

0E8h(232)확보

rsp, 0E8h
// rsp를 0E8h(232) stack 공간확보

⑤



lea rsp, [rbp+0C8h]

// rbp에 0C8h(200)을 더한 주소를 rsp에 저장

// ③의 0C8h 확보하기 전 주소

⑥ pop rdi // rdi와 rbp의 값을 빼냄.

pop rbp

* popip ret

sp