c언어 - HW4

임베디드스쿨1기

Lv1과정

2021. 04. 09

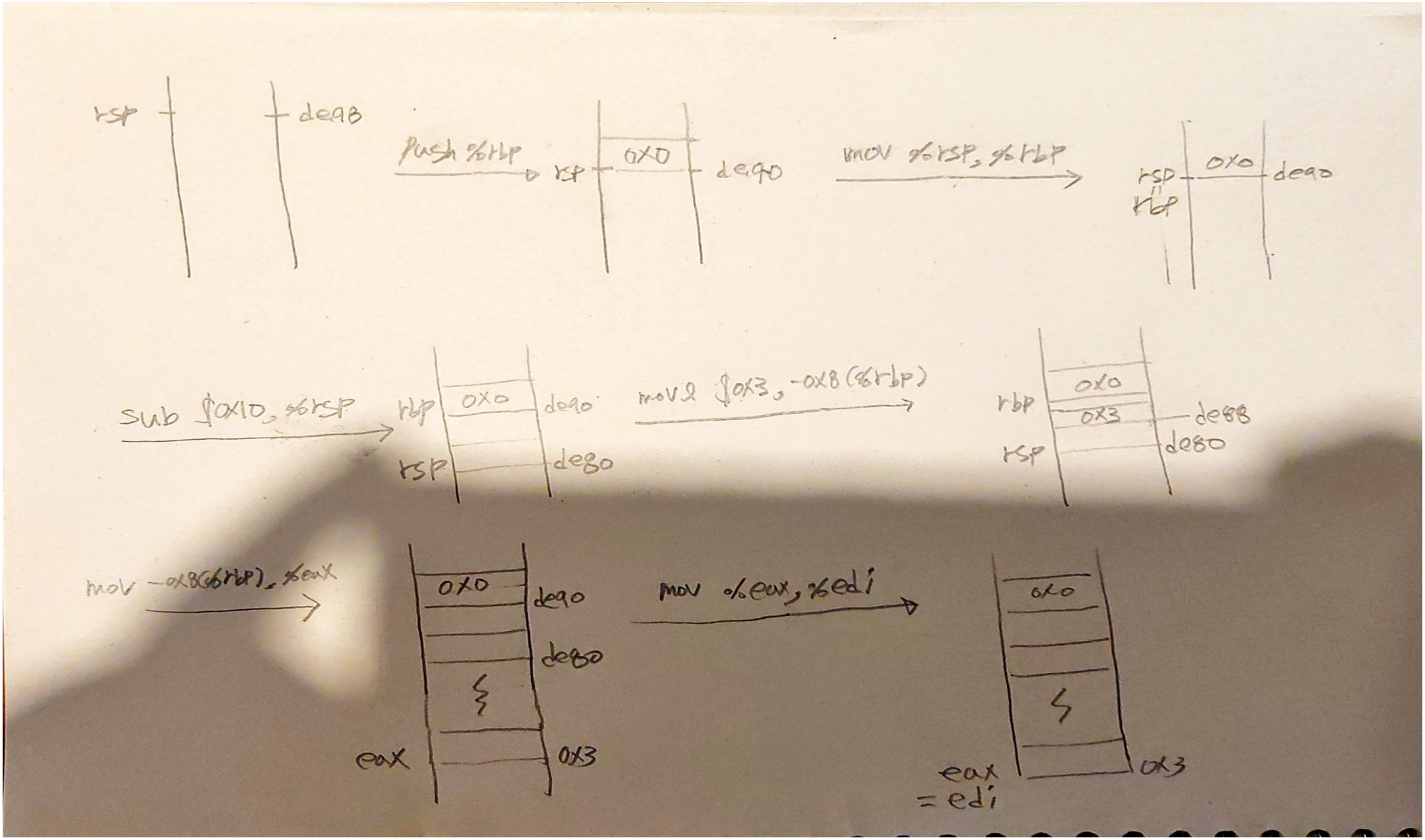
이충

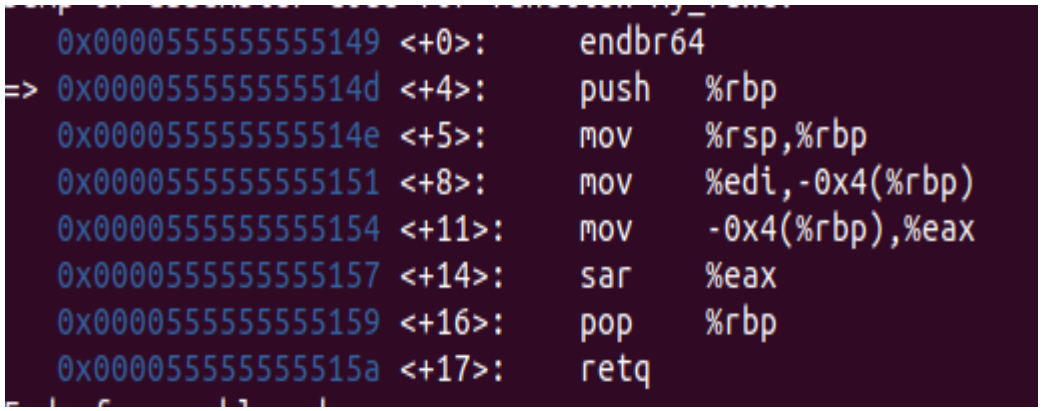
1. 쉬프트연산 함수 기계어분석

```
=> 0x000055555555515b <+0>:     endbr64
   0x000055555555515f <+4>:     push   %rbp
   0x0000555555555160 <+5>:     mov    %rsp,%rbp
   0x0000555555555163 <+8>:     sub    $0x10,%rsp
   0x0000555555555167 <+12>:    movl   $0x3,-0x8(%rbp)
   0x000055555555516e <+19>:    mov    -0x8(%rbp),%eax
   0x0000555555555171 <+22>:    mov    %eax,%edi
   0x0000555555555173 <+24>:    callq  0x555555555149 <my_func>
   0x0000555555555178 <+29>:    mov    %eax,-0x4(%rbp)
   0x000055555555517b <+32>:    mov    -0x4(%rbp),%eax
   0x000055555555517e <+35>:    mov    %eax,%esi
   0x0000555555555180 <+37>:    lea    0xe7d(%rip),%rdi        # 0x555555556004
   0x0000555555555187 <+44>:    mov    $0x0,%eax
   0x000055555555518c <+49>:    callq  0x555555555050 <printf@plt>
   0x0000555555555191 <+54>:    mov    $0x0,%eax
   0x0000555555555196 <+59>:    leaveq
   0x0000555555555197 <+60>:    retq
```

Push %rbp : rsp를 8바이트만큼 내리고 그 위치에 rbp값을 넣는다.
Mov %rsp, %rbp : rsp를 rbp에 복사한다.
Sub $0x10, %rsp: rsp를 16바이트만큼 내린다.
Movl $0x3, -0x8(%rbp): 숫자3을 rbp로부터 8바이트 아래에 저장한다.
Mov -0x8(%rbp), %eax: rbp아래 8바이트에 저장된것을 eax로한다.
Mov %eax, %edi: eax를 edi에 복사한다.
Callq 주소 <함수이름>: push + jump, rsp에 복귀주소 저장하고 함수주소로 이동한다.

디버그 과정

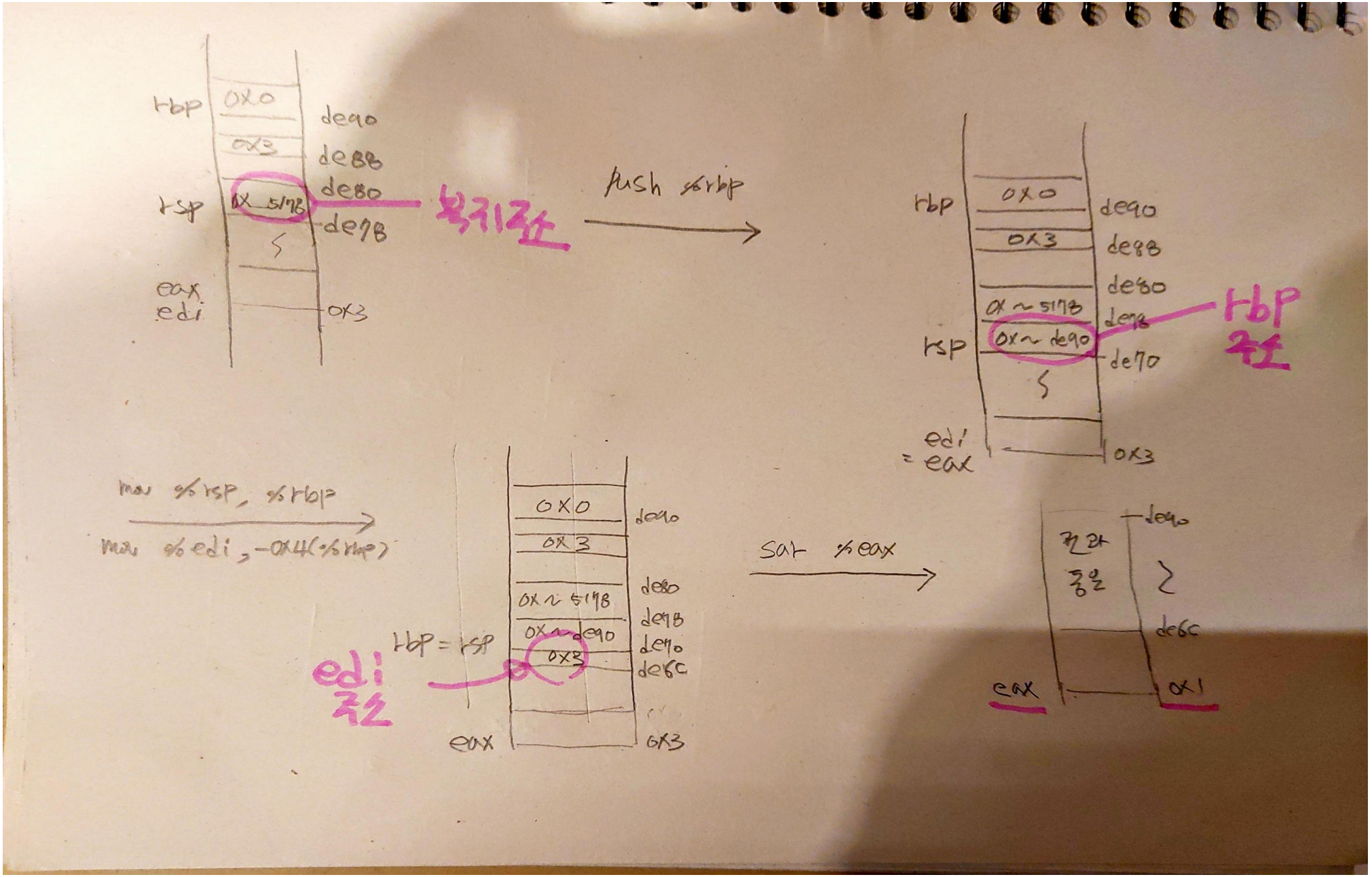

rsp
deq8
Push %rbp
rsp
0x0
deq0
mov %rsp, %rbp
rsp
rbp
0x0
deq0
sub $0x10, %rsp
rbp
0x0
deq0
rsp
deq0
movl $0x3, -0x8(%rbp)
rbp
0x0
0x3
deq8
rsp
deq0
mov -0x8(%rbp), %eax
0x0
deq0
deq8
eax
0x3
mov %eax, %edi
0x0
eax
=edi
0x3

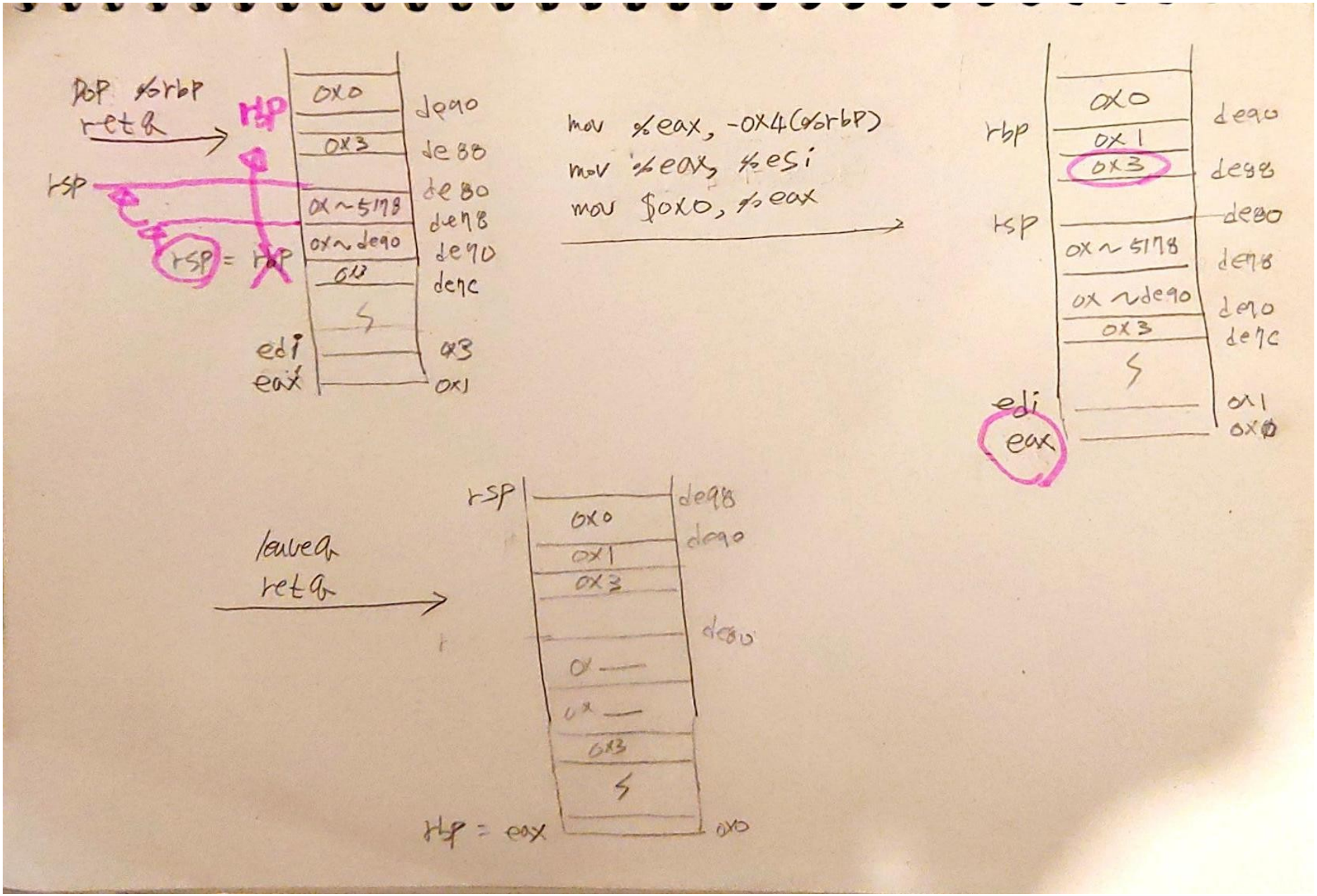


Sar %eax: eax를 오른쪽으로 1비트 쉬프트연산
Pop %rbp: rsp에 저장된값을 rbp에 전달한다.
Retq: pop %rip, rsp에 저장된값을 rip에 전달한다.

```
=> 0x000055555555515b <+0>:     endbr64
   0x000055555555515f <+4>:     push   %rbp
   0x0000555555555160 <+5>:     mov    %rsp,%rbp
   0x0000555555555163 <+8>:     sub    $0x10,%rsp
   0x0000555555555167 <+12>:    movl   $0x3,-0x8(%rbp)
   0x000055555555516e <+19>:    mov    -0x8(%rbp),%eax
   0x0000555555555171 <+22>:    mov    %eax,%edi
   0x0000555555555173 <+24>:    callq  0x555555555149 <my_func>
   0x0000555555555178 <+29>:    mov    %eax,-0x4(%rbp)
   0x000055555555517b <+32>:    mov    -0x4(%rbp),%eax
   0x000055555555517e <+35>:    mov    %eax,%esi
   0x0000555555555180 <+37>:    lea    0xe7d(%rip),%rdi        # 0x555555556004
   0x0000555555555187 <+44>:    mov    $0x0,%eax
   0x000055555555518c <+49>:    callq  0x555555555050 <printf@plt>
   0x0000555555555191 <+54>:    mov    $0x0,%eax
   0x0000555555555196 <+59>:    leaveq
   0x0000555555555197 <+60>:    retq
```

Mov %eax, -0x4(%rbp): eax를 rbp 4바이트 아래에 저장.
Mov -0x4(%rbp), %eax: rbp 4바이트 아래 저장된거을 eax에전달
Lea 0xe7d(%rip), %rdi: rip에 0xe7d를 더한값을 rdi에 복사
Mov $0x0, %eax: eax를 0x0으로한다.
Leaveq: mov %rbp %rsp + Pop %rbp
==> rbp의 주소를 rsp로 복사하고 rsp에 저장된값을 rbp에 전달한다.


rbp
0x0
0x3
rsp
0x ~ 5178
eax
edi
0x3
복귀주소
push %rbp
rbp
0x0
0x3
0x ~ 5178
0x ~ de90
rsp
edi
= eax
0x3
rbp
주소
mov %rsp, %rbp
mov %edi, -0x4(%rbp)
rbp = rsp
edi
값
0x0
0x3
0x ~ 5178
0x ~ de90
0x3
eax
0x3
sar %eax
값과
동일
eax
0x1


POP %rbp
retq
rbp
rsp
rsp = rbp
0x0
0x3
0x~5178
0x~de90
0x3
edi
eax
0x3
0x1
mov %eax, -0x4(%rbp)
mov %eax, %esi
mov $0x0, %eax
rbp
rsp
0x0
0x1
0x3
0x~5178
0x~de90
0x3
edi
eax
0x1
0x0
leaveq
retq
rsp
0x0
0x1
0x3
0x3
rbp = eax
0x0

숙제를 다 하지 못하였습니다.
못한부분은 다음주에 함께 올리겠습니다.