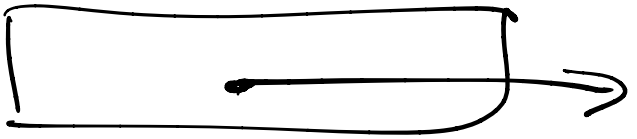
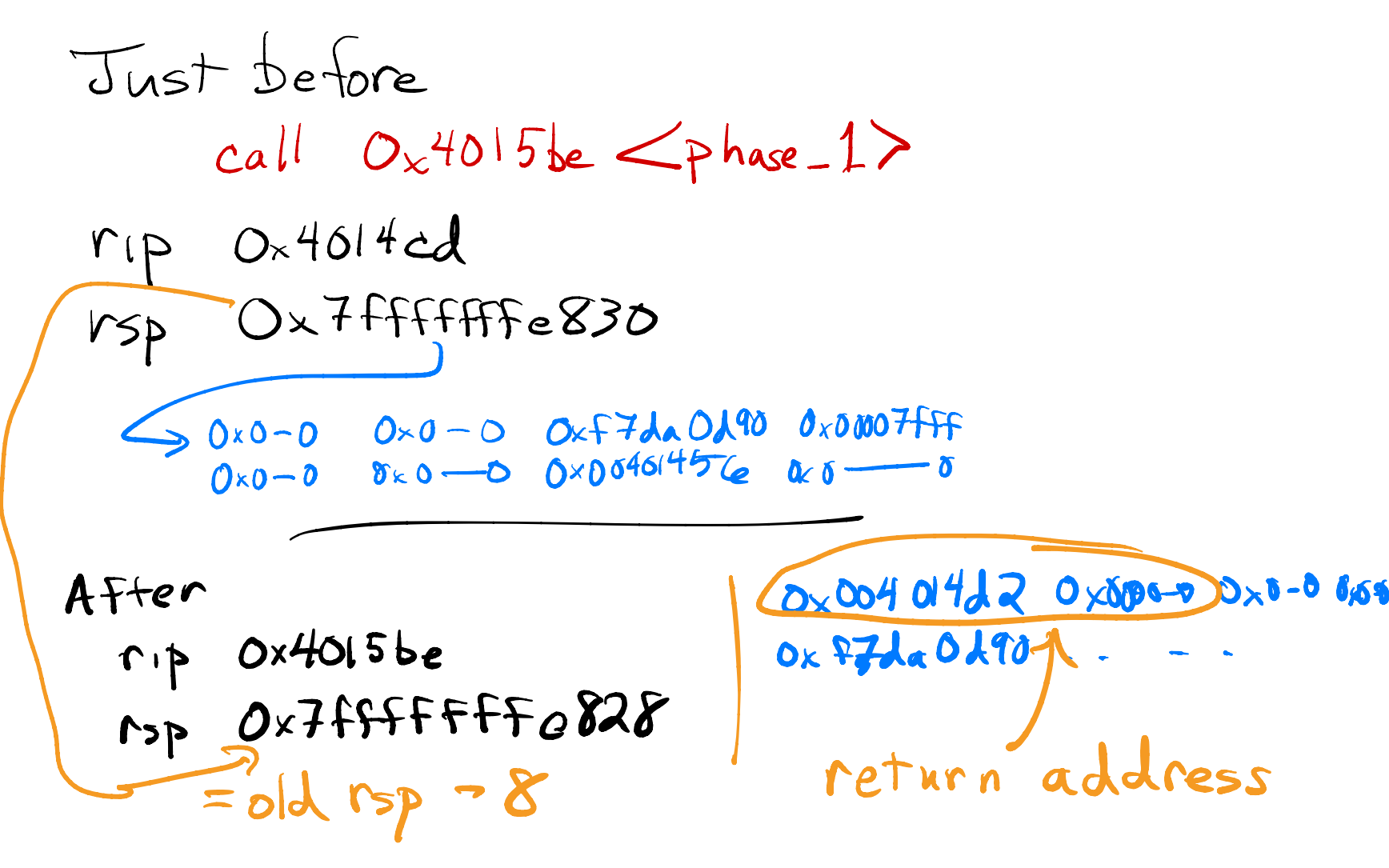
CS208

M, 16 Feb 2026

rip  current instruction

rsp  top of the stack


Just before
call 0x4015be <phase_1>
rip 0x4014cd
rsp 0x7fffffffe830
0x0-0 0x0-0 0xf7da0190 0x00007fff
0x0-0 0x0-0 0x00461456 0x0-0
After
rip 0x4015be
rsp 0x7fffffffe828
= old rsp - 8
0x004014d2 0x0-0 0x0-0
0xf7da0190
return address

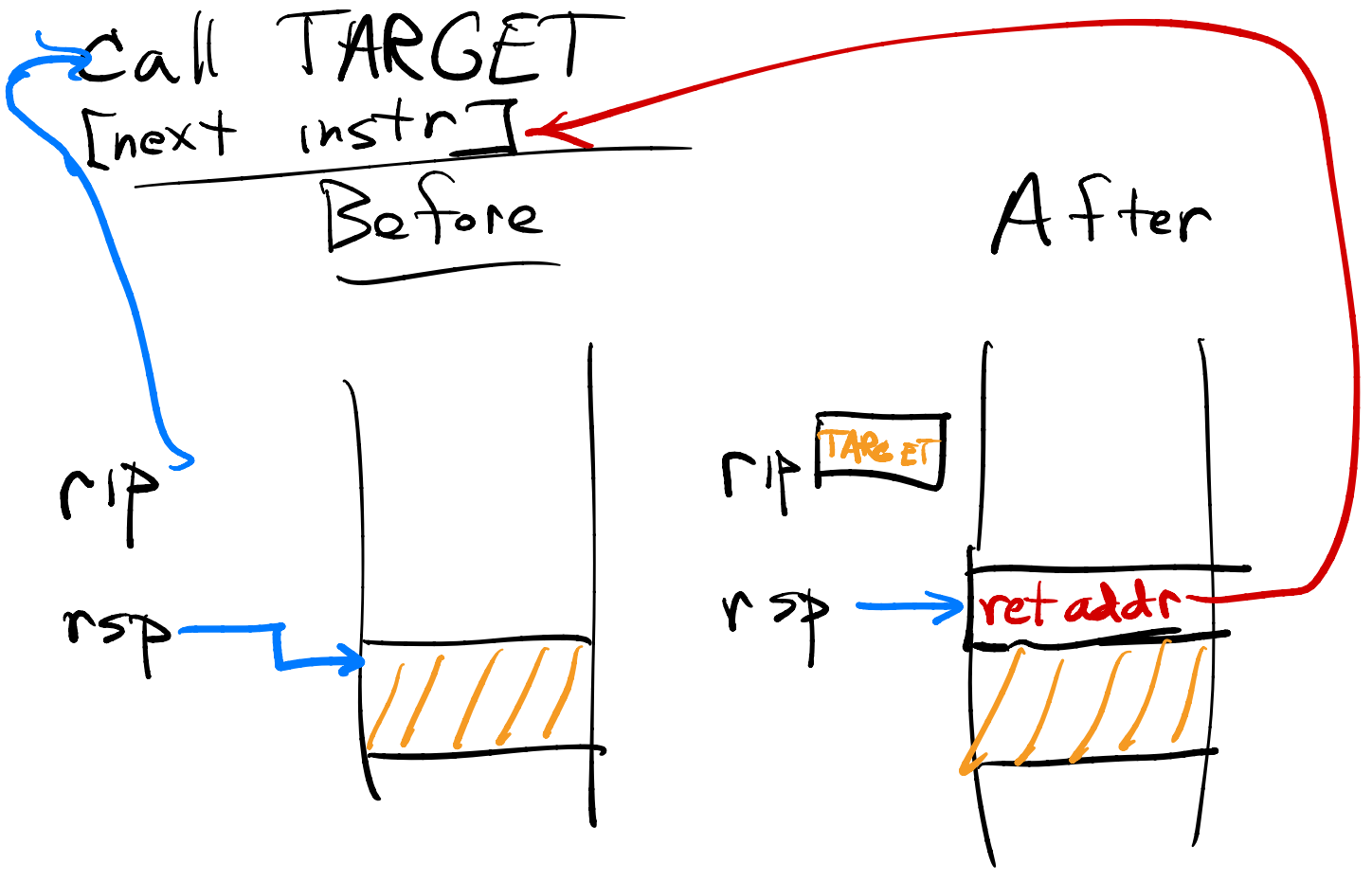
call TARGET

① rsp = rsp − 8

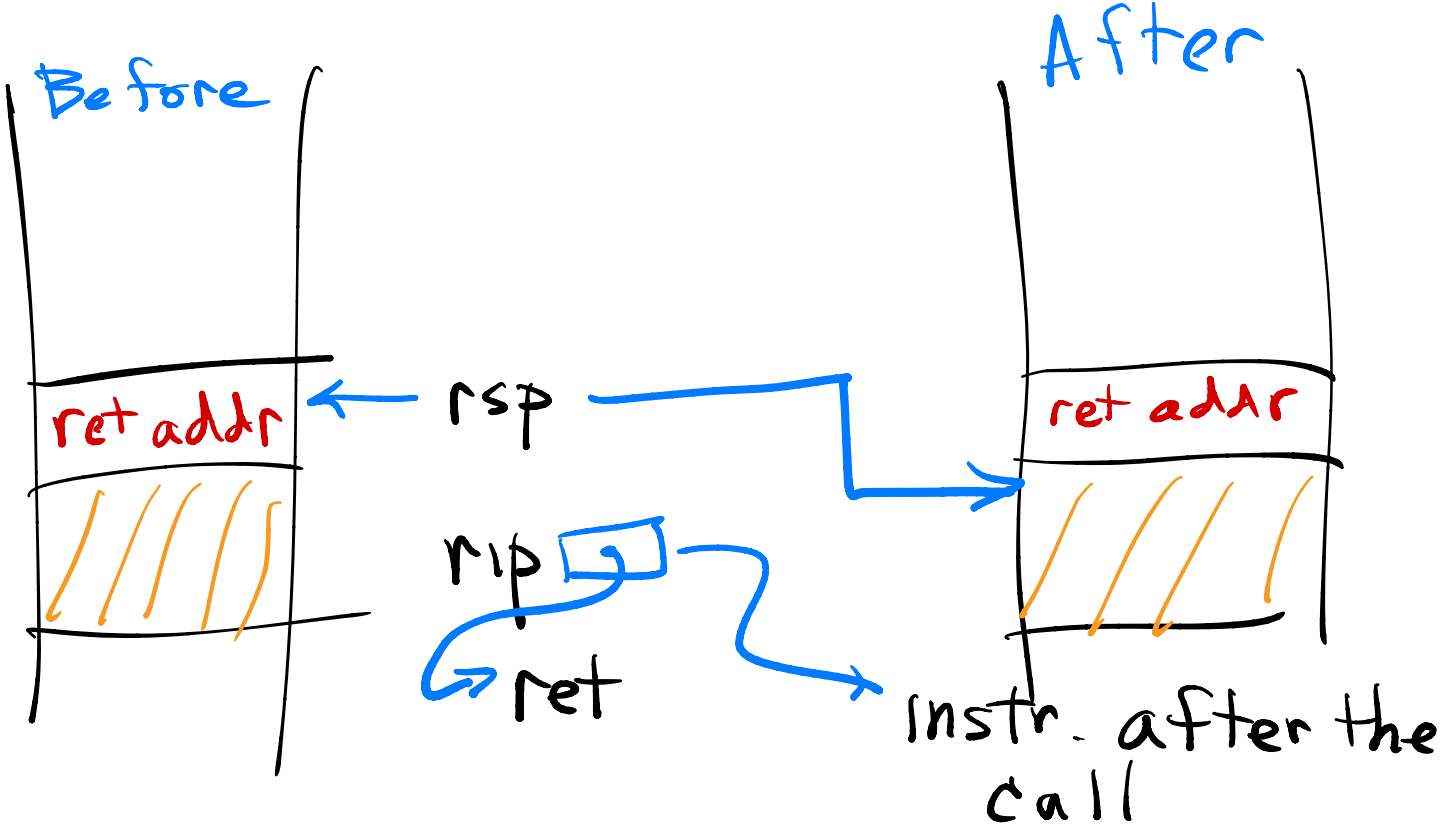
Push return address

(rsp) = addr of instruction immediately after "call TARGET"

② rip = TARGET


call TARGET
[next instr]
Before
After
rip
rsp
rip
TARGET
rsp
ret addr

ret


Before
After
ret addr
rsp
ret addr
rip
ret
instr. after the call

ret

① rip = (rsp)

return address

② rsp = rsp + 8

pops the return address into rip

Jeff's zoo1, calling phase_1

0x4014cd call 0x4015be <phase1>
0x401422 call ——— <phase_unlocked>) in main

rip = 0x4014cd
rsp = 0x7fffffffe830

---

call → rip = 0x4015be
rsp = 0x7fffffffe828

---

ret → Before ret
rip = 0x401528
rsp = 0x7f—fe828

After ret
rip = 0x401422
rsp = 0x7f-f0830

```
int f(int x, int y) {
    ...
    return result;
}

int main() {
    int a = f(6, 14);
    :
}
```

What does main do?

① `movl $6, %edi`
`movl $0xE, %esi`

② call f

pushes r.addr
sets rip = f

After f returns?

Something like

`movl %eax, somewhere`

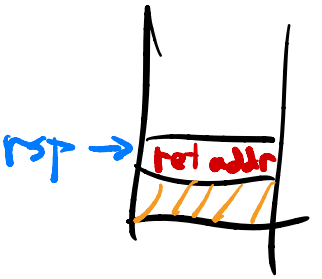
ie. put ret value in variable a

```
int f(int x, int y) {
    ...
    return result;
}

int main() {
    int a = f(6, 14);
    :
}
```

What does f need to do?

As f begins



"caller"

① Save regs that main owns

e.g. "push %rbx" ② make space on the stack for locals

e.g. "sub $0x20, %rsp"

Before f executes ret,

needs to:

① delete space on stack

add $0x20, %rsp

③ pop saved reg

pop %rbx

Things to think about

1. What are the responsibilities of caller & callee?
   (eg. main) (eg. f)
2. What does the stack look like during recursion?