Lecture 34 — 15th Oct. 2018

- interrupt / system call (xv6)
- context s/w
- first user process
- fork (child) process creation

---

① int —— system calls.

struct proc

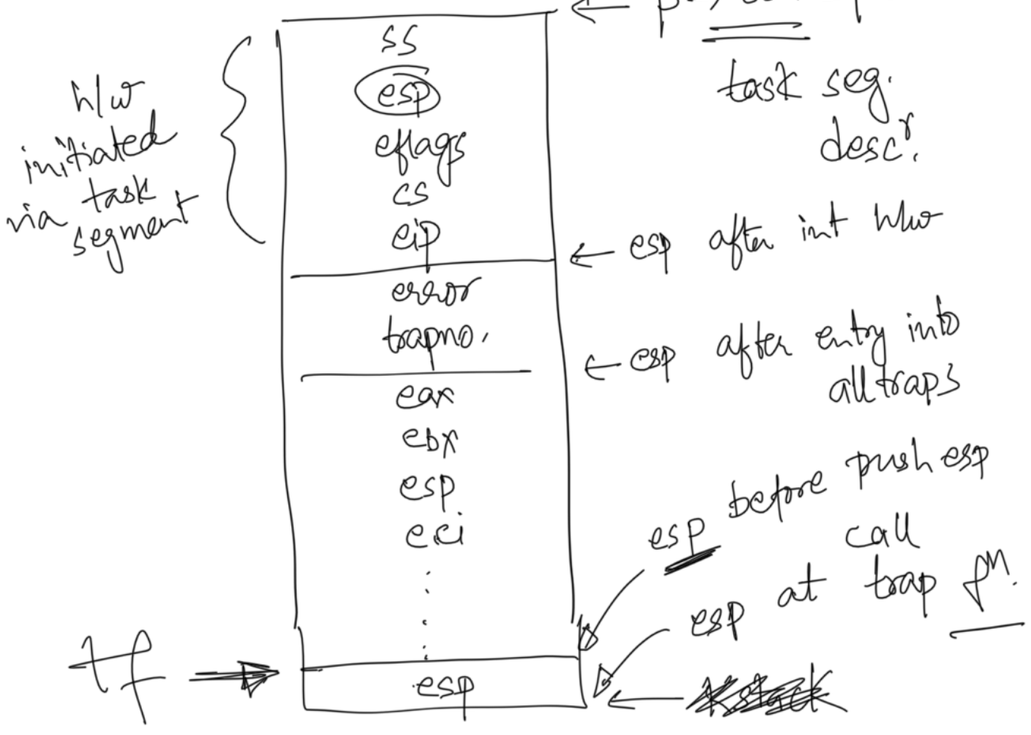
p → tf

p → ts → esp0 ← task seg. desc.

p → context

p → ts

p → Kstack



~~call~~ push esp
call trap || which moves to trap handler.

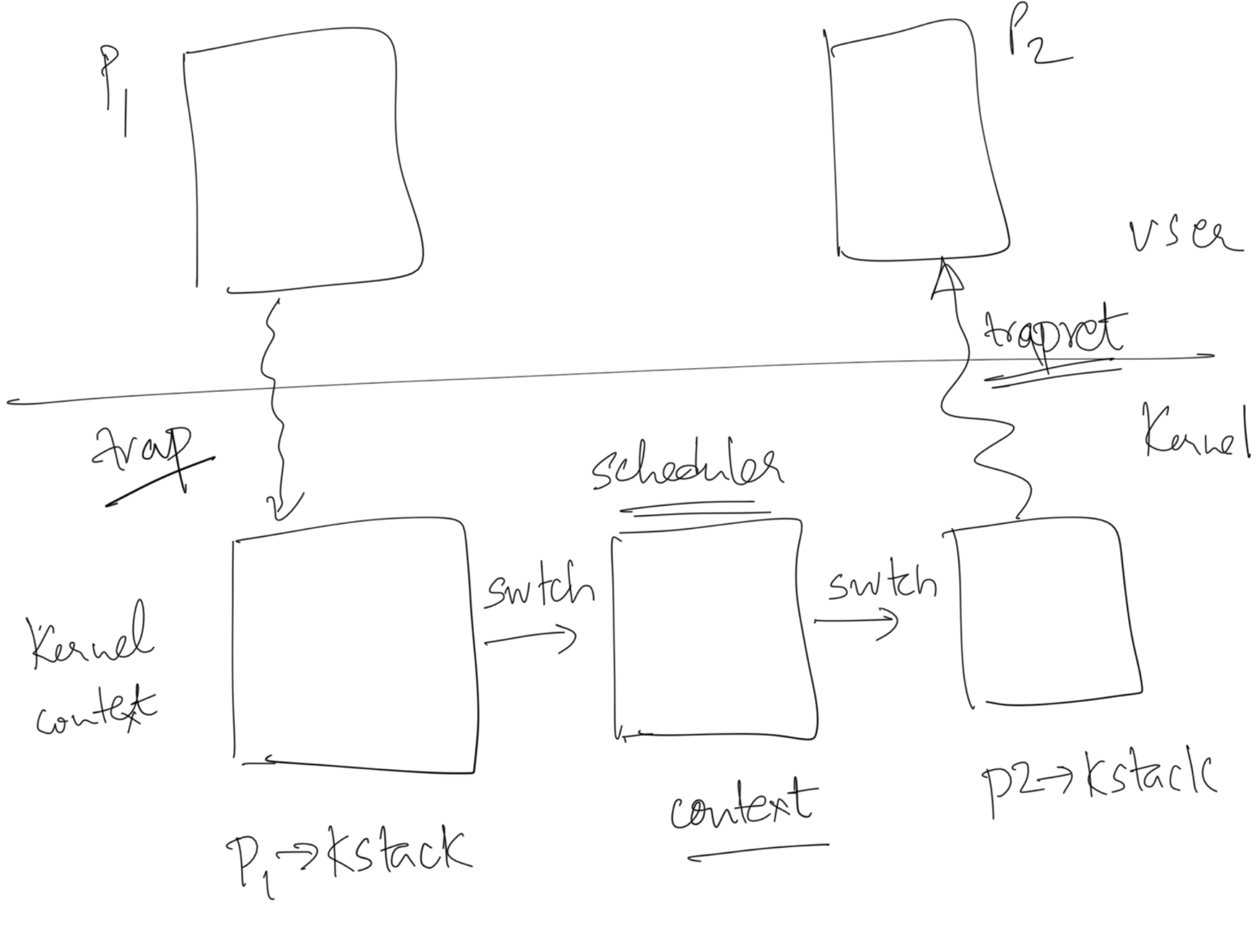
---

(2) context s/w

sched ~~~~ sleep
wakeup
scheduler

wakeup in
a different "context"

trap → timer
↓ yield
↳ sched

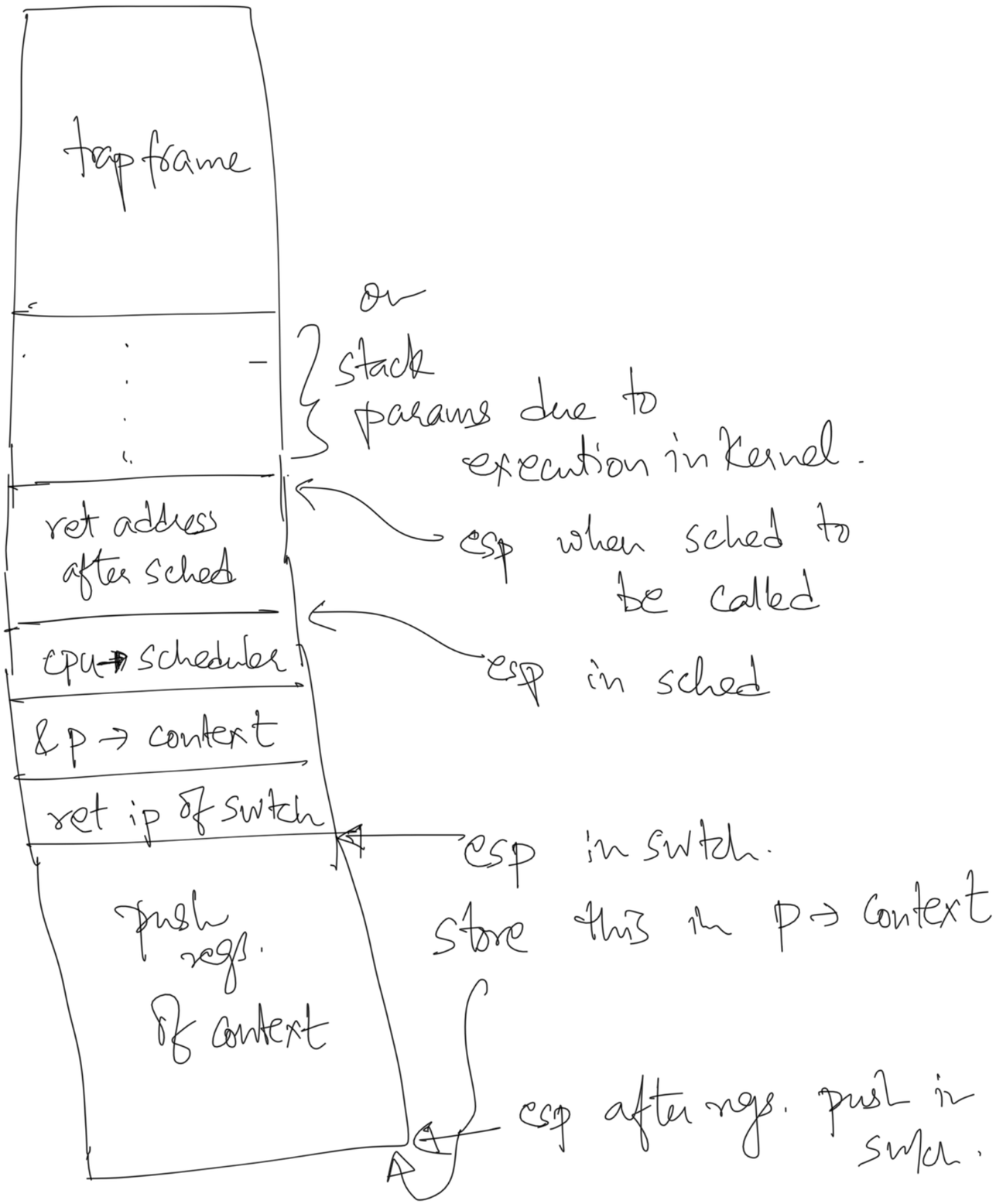


Q. how does swtch return to 'ip' after swtch in another context.

(i) p→context update

- push args. & return address on stack when swtch is called.
- return address used to jump across contexts during swtch.

'p'


trap frame
or
stack
params due to
execution in kernel.
ret address
after sched
esp when sched to
be called
cpu → scheduler
esp in sched
&p → context
ret ip of swtch
esp in swtch.
push
regs.
of context
store this in p → context
esp after regs. pusl in
swtch.

swtch (&p → context, cpu → scheduler)

(*) note: both args. are pointing to stack locations / variables.

— on stack s/w the next action is to pop variables & return from swtch.

⇒ hence, no cip regs is saved!

swtch game plan

(i) get context / stack of new thread.
(ii) push regs. on old stack. & save esp regs. in old → context
(iii) s/w stacks.
(iv) pop from stack
(v) return from swtch.