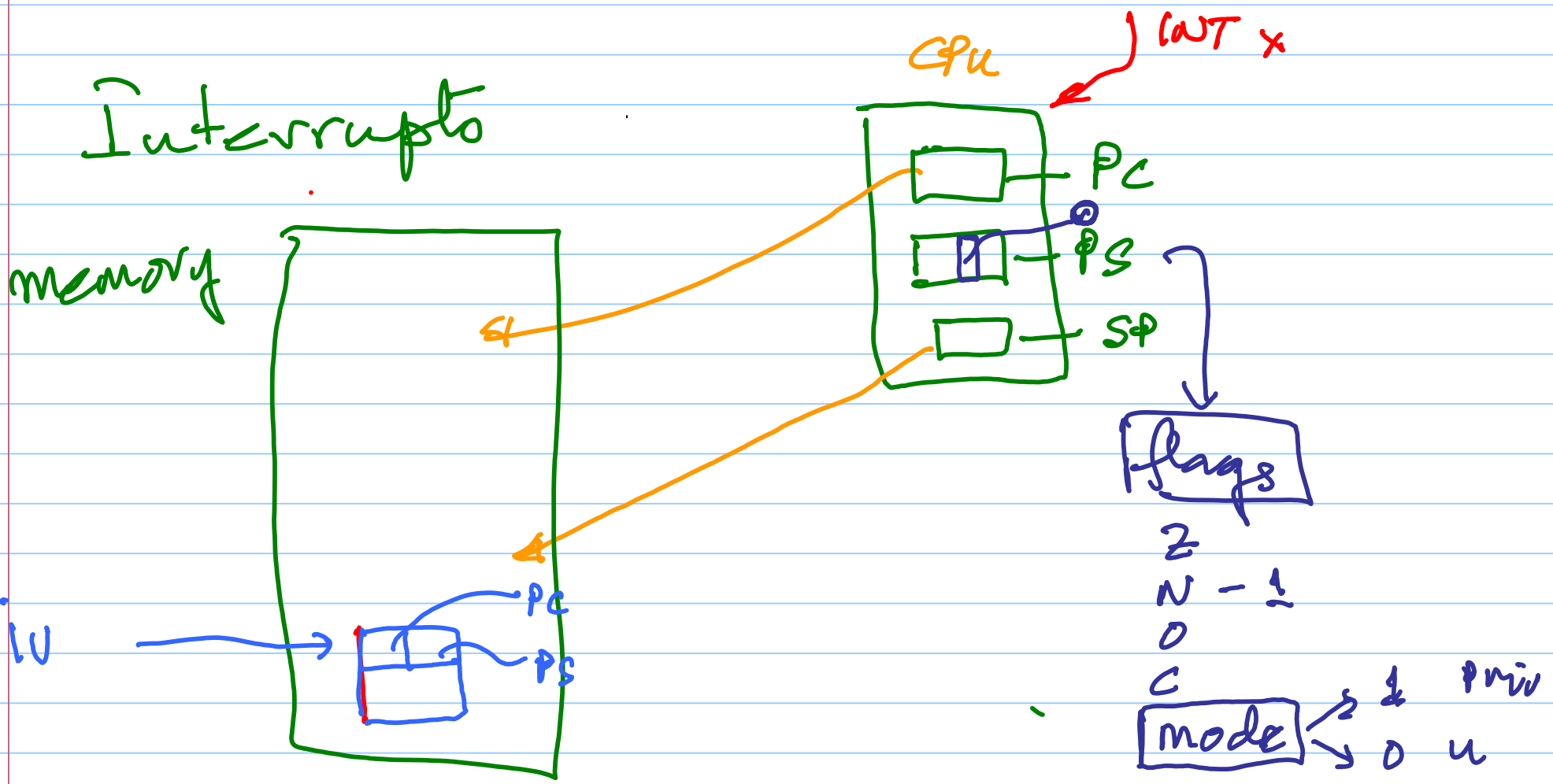
CSE 531: Distributed and Multiprocessor Operating Systems

Fall 2020

Class: 06 Date: 9/14

Note Title



PS → program status

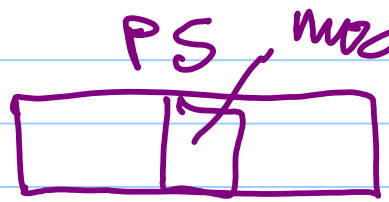
after int ... (x)

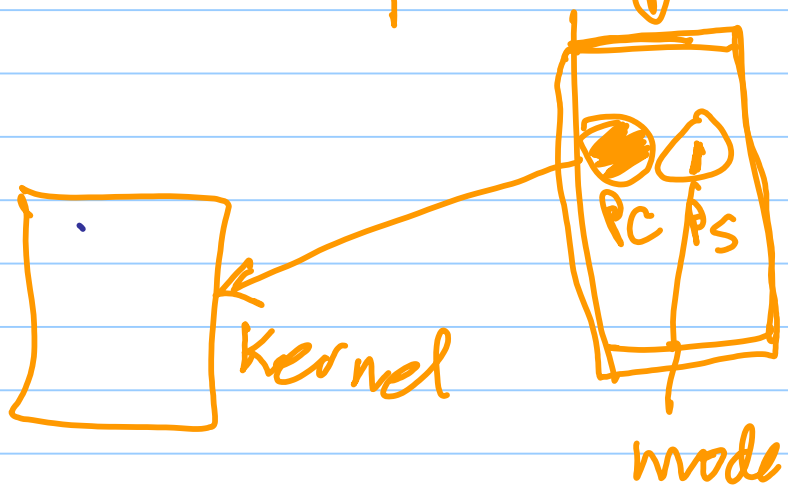
- push pc → store PC → top of Stack + Decrement SP
- push PS
- load PC & PS ← (new)
  from IV → slot x

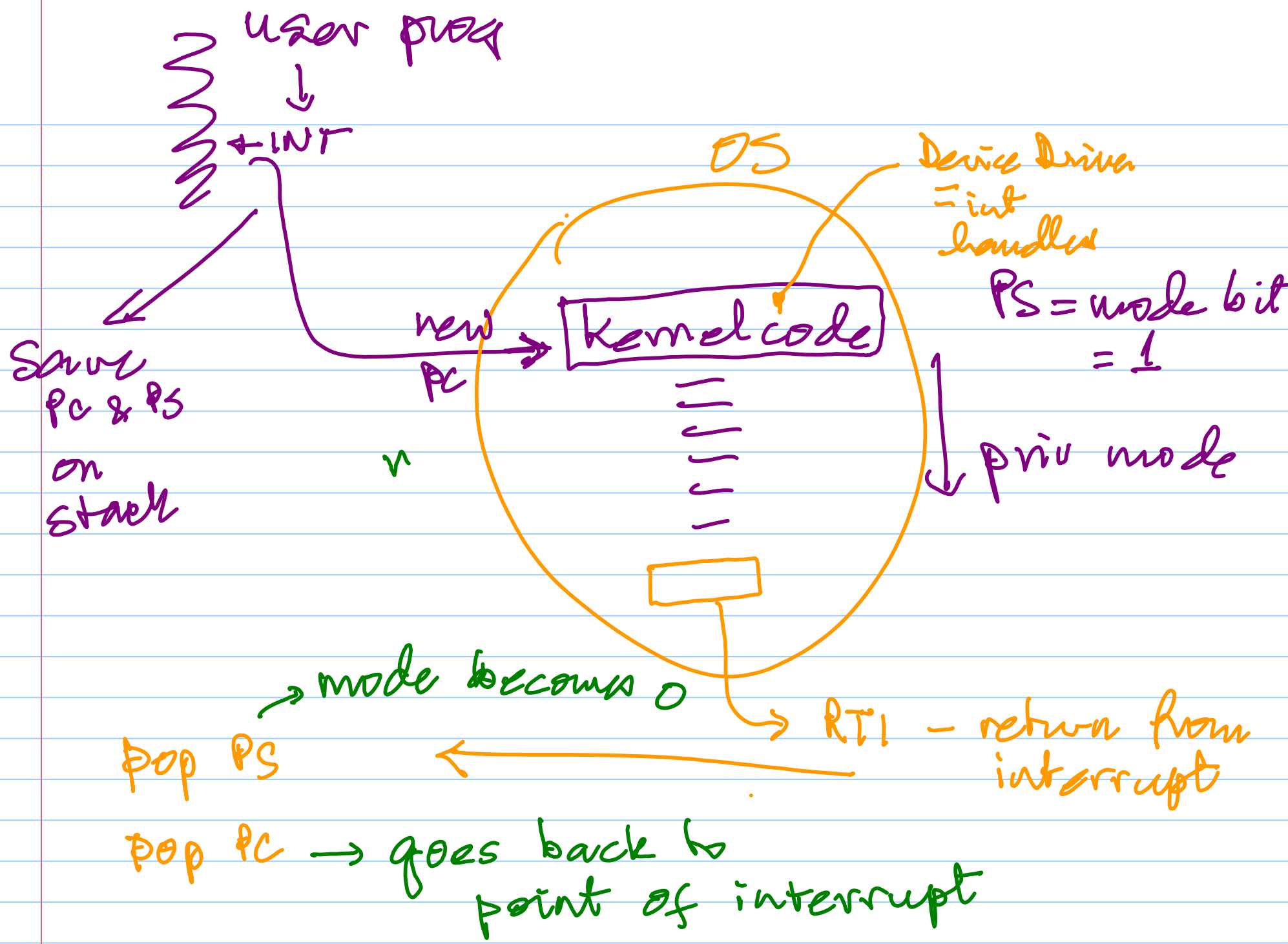
new PC
new PS

IV

hard coded



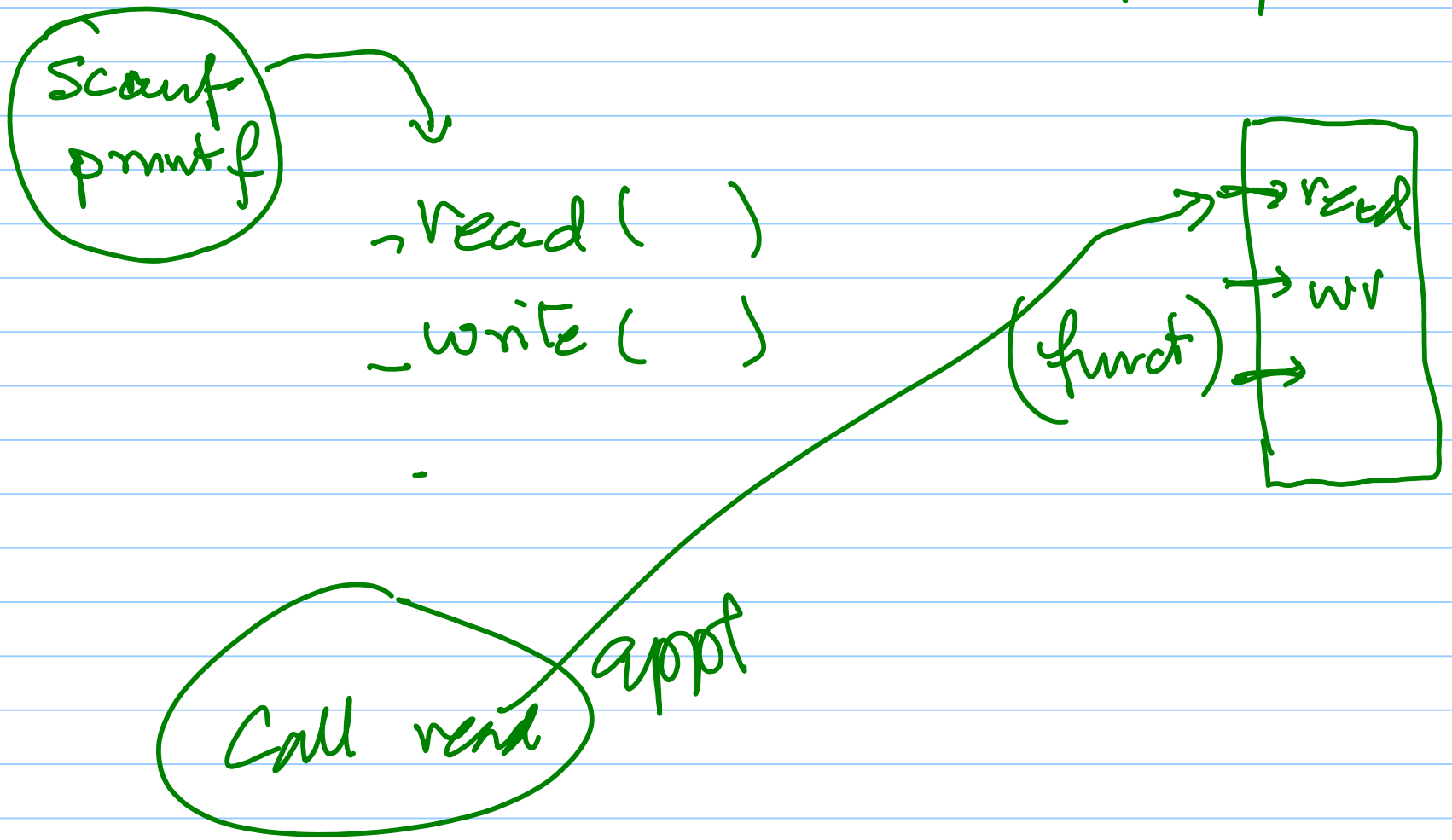



user prog
INT
OS
Device Driver = int handler
PS = mode bit = 1
save PC & PS on stack
new PC
Kernel code
priv mode
mode becomes 0
RTI - return from interrupt
pop PS
pop PC → goes back to point of interrupt

User 1
User 2
INT
Kernel
Passive

System calls

explicit call to kernel made from appl program



read (arg 1, arg 2)

↳ put args in Registers (R_1, R_2) 4

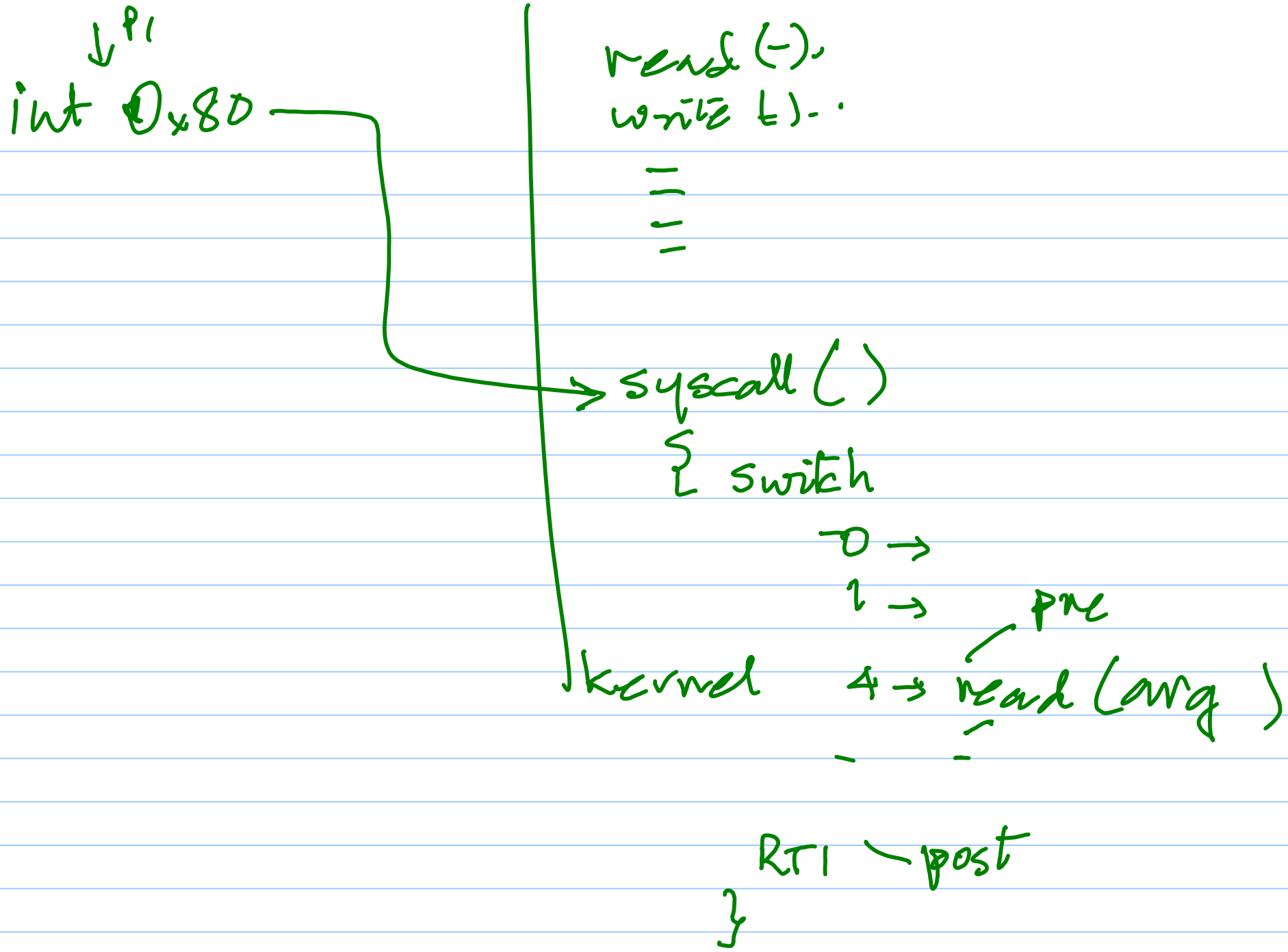
put [syscall ID] in R0

[INT] [0x80] 5

↳ cause / simulate an interrupt of type

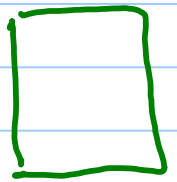
return arg in R_x

0
1
2
3 - read -
4 - write -
5 - open -
:
:
200


int 0x80
read(-).
write(-).
syscall()
{ switch
0 →
1 →
4 → read(arg)
kernel
pre
RTI → post
}

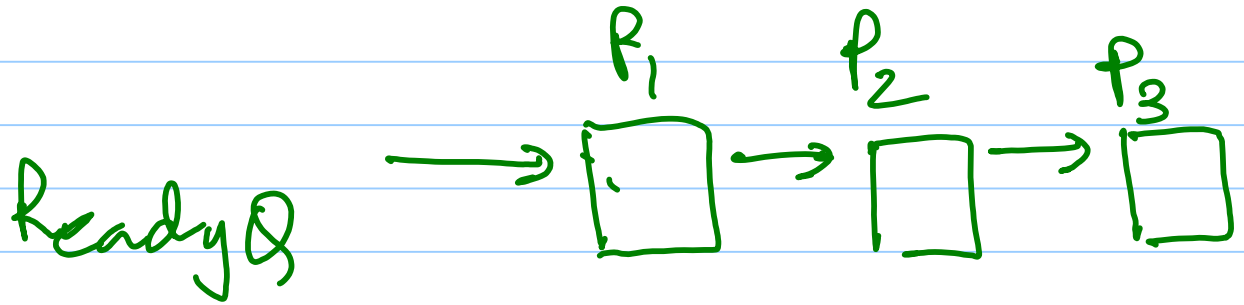
Context switching

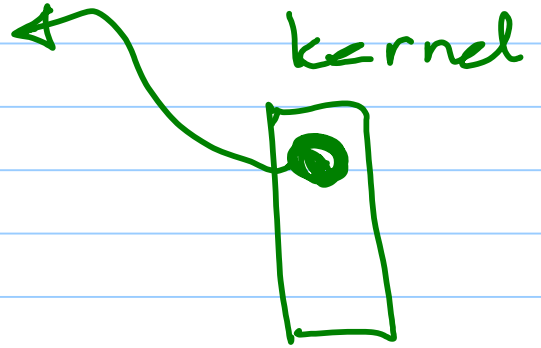
processes (thread)



PCB [process control block]

TCB

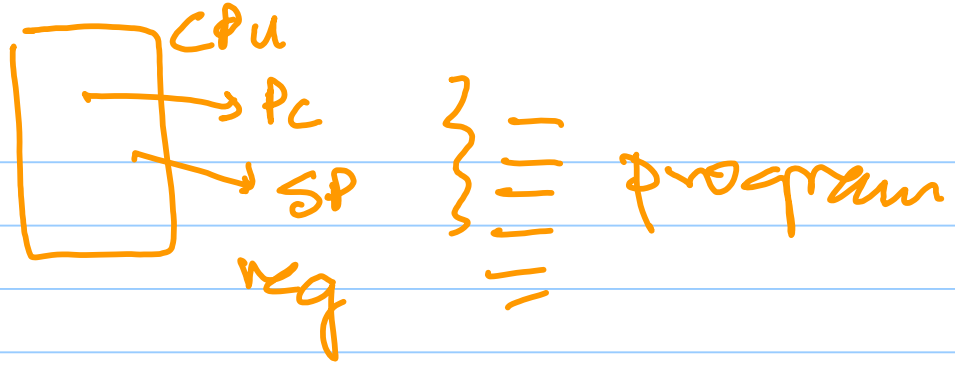




PCB
prev
state
PID
SP
next

CPU
PC
SP
R1

P1



→ timer →

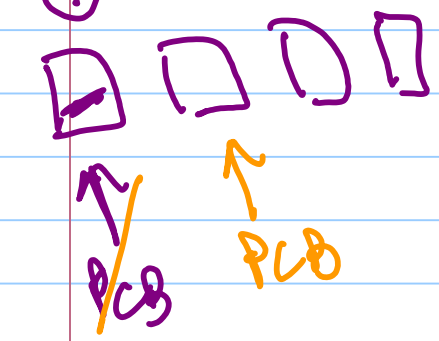
```
timer int handler
{ Save context
  load next context
} ← return
```

timer → ~~push PC~~ PS

{ push R0 ... Rx (to current stack)

move SP to PCB.SP

goto ReadyQ & find new PCB



(fix ReadyQ)

move PCB.SP to SP

pop R0 ... Rx from stack

RTI → executes the new process

→ does not go back to old Proc

Context switch.

x
P1
x
Stash
PC = x
SP
load PC from stack
PC = y
RTI
PC
y
x