CSE 330: Operating Systems **Class:** 03 **Date:** 8/25

Fall 2016

Note Title

Protection

$$\begin{tikzcd} \arrow[r] & \text{memory protection} \arrow[d] \arrow[dr] & \\ & & \text{base/bound register} \\ & \text{CPU protection} \arrow[uu, bend left] \arrow[r] & \text{privileged mode} \end{tikzcd}$$

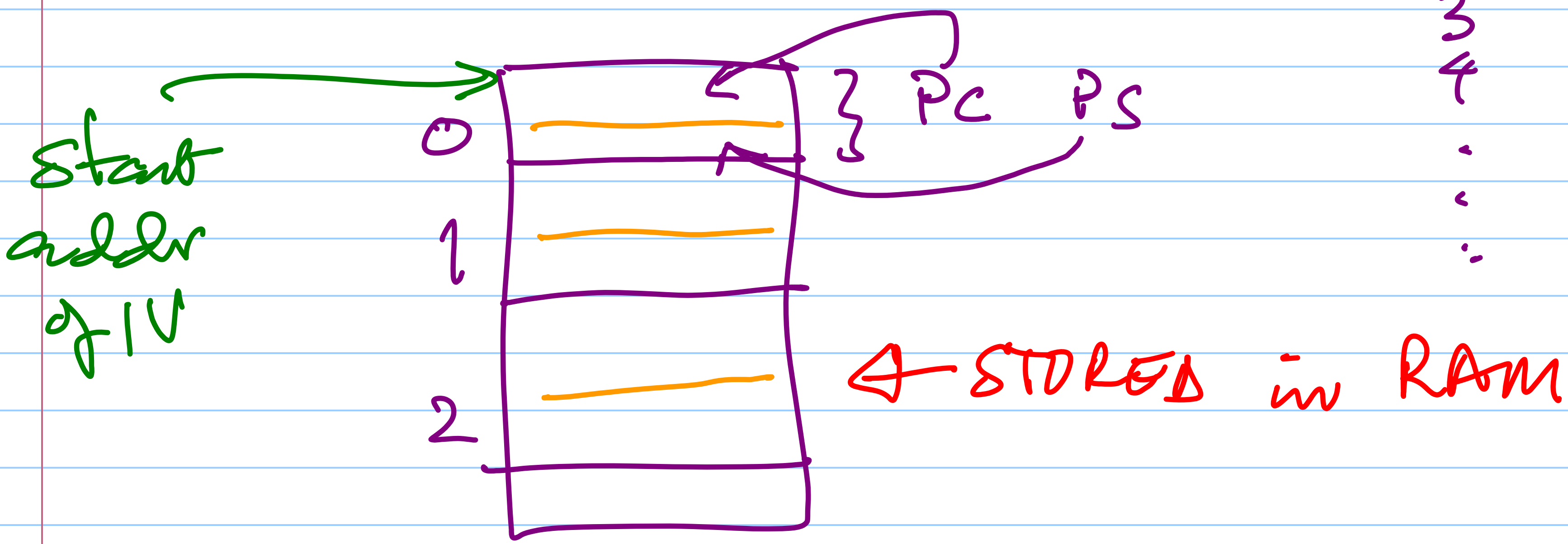
Interrupts

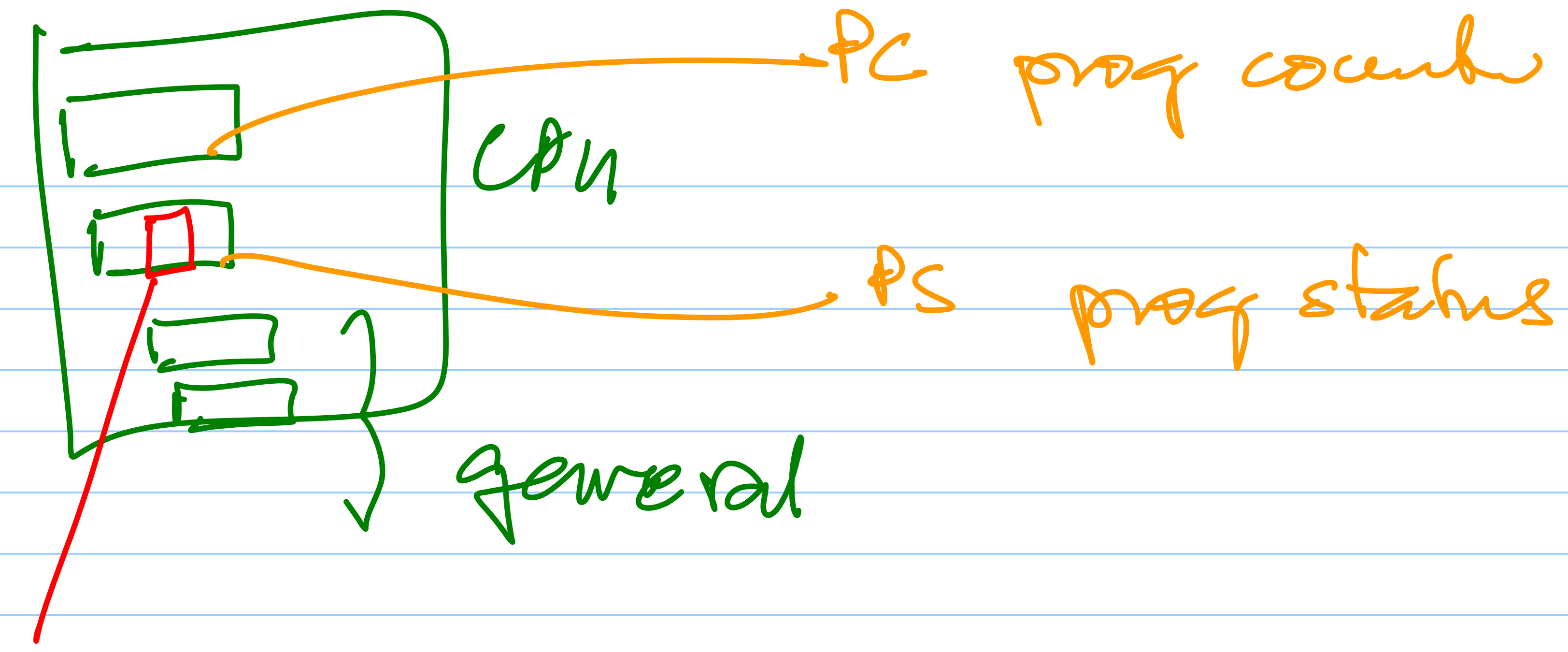
→ "external" → generated by hardware

→ "internal" → generated by software

Interrupts are numbered → 0, 1, 2, 3, 4, ...

Interrupt Vector → IV





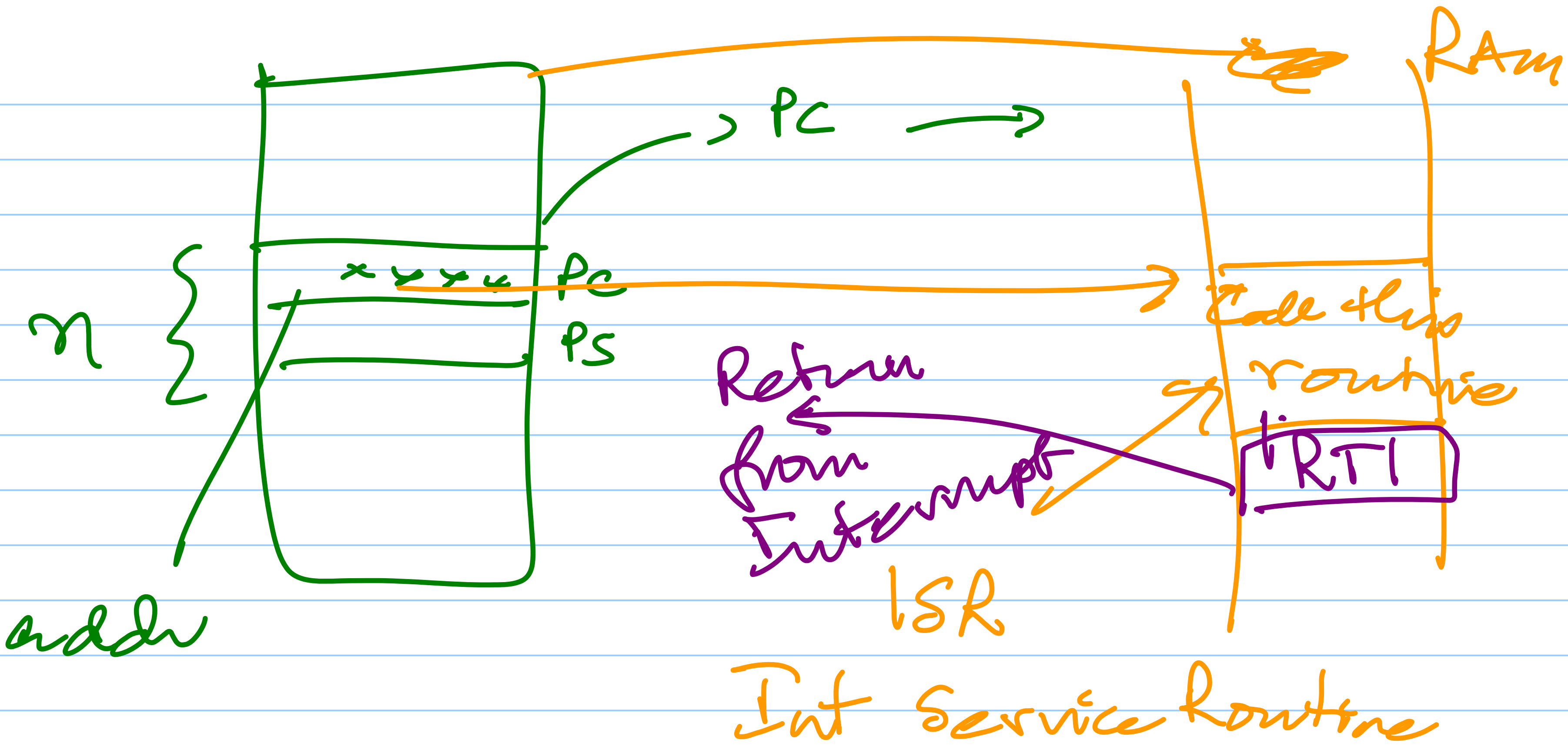
MODE BIT – 0 user

– 1 privileged

→ Interrupt no 'n'

CPU → locate IV[n]

$$\left\{\begin{array}{l}\text{push PC on the stack} \\ \text{" PS " " "} \\ \text{move IV[n].PC} \rightarrow \text{PC} \\ \text{move IV[n].PS} \rightarrow \text{PS}\end{array}\right.$$



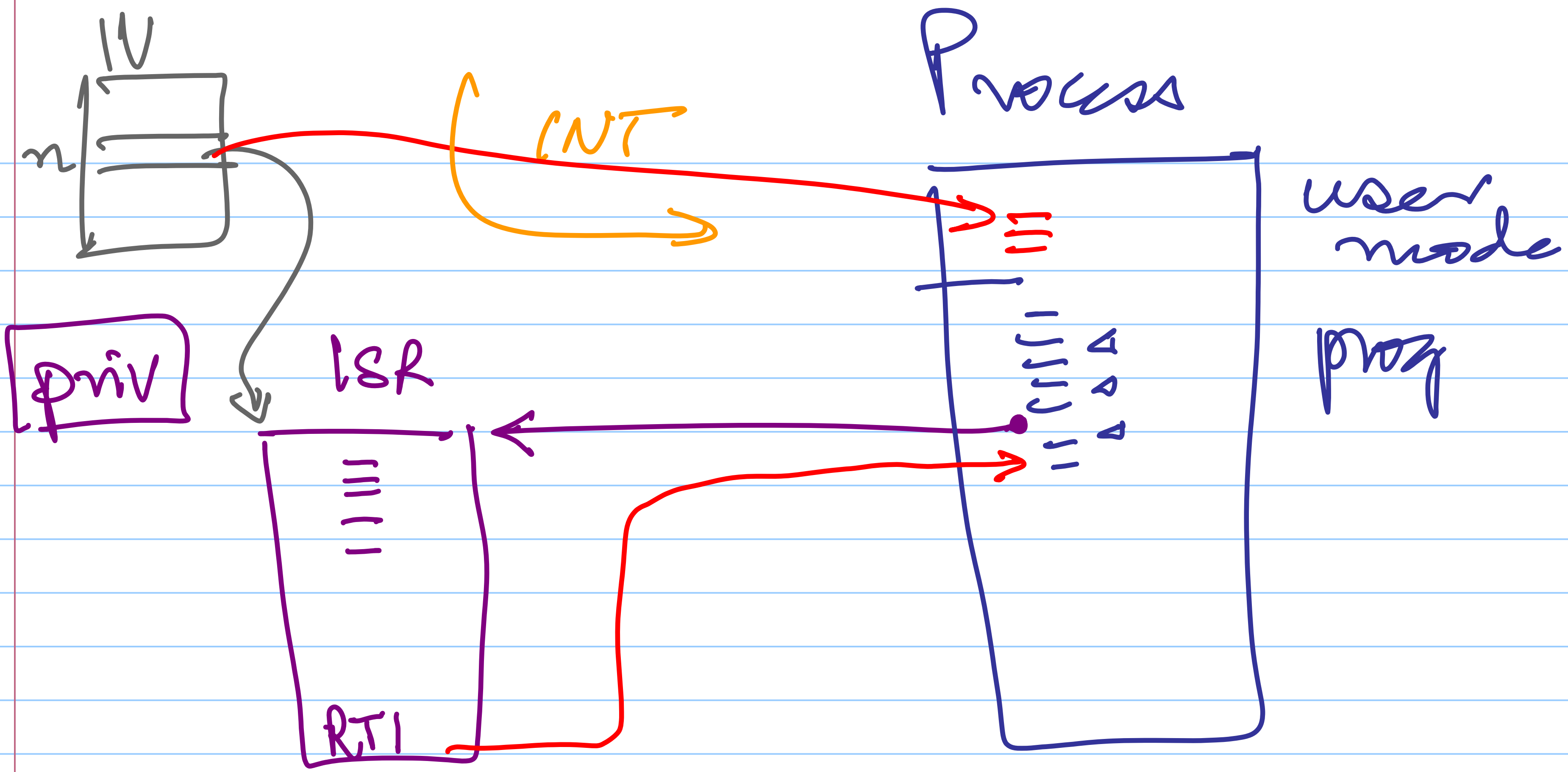
n
addr
x y y y
PC
PS
PC
RAM
Code this routine
RTI
Return from Interrupt
ISR
Int Service Routine

RTI $\rightarrow$ load PS from stack
(pop)
load PC from stack
(pop)

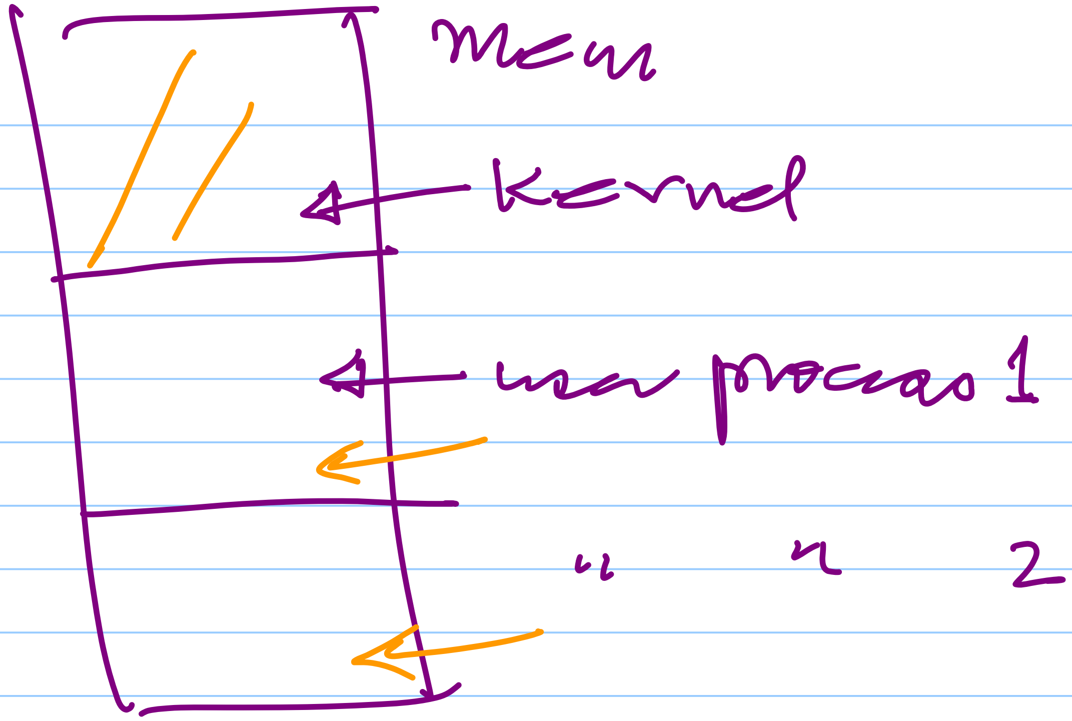
CPU protection

↳ put a 1 on all mode bits of the IV

→ start a user process with CPU mode bit set to 0



IV
n
priv
INT
ISR
RTI
Process
user mode
prog


mem
kernel
user process 1
" " 2

protected using mem prot

<u>Kernel</u> → resident code
→ core part of OS
→ fixed memory location

run in priv mode

Contains : IV, all ISR
code, static data (stack, heap)

System calls

→ how to write some data on ..... monitor
screen
file etc

printf ("Hello world\n")

$\hookrightarrow$ library routine

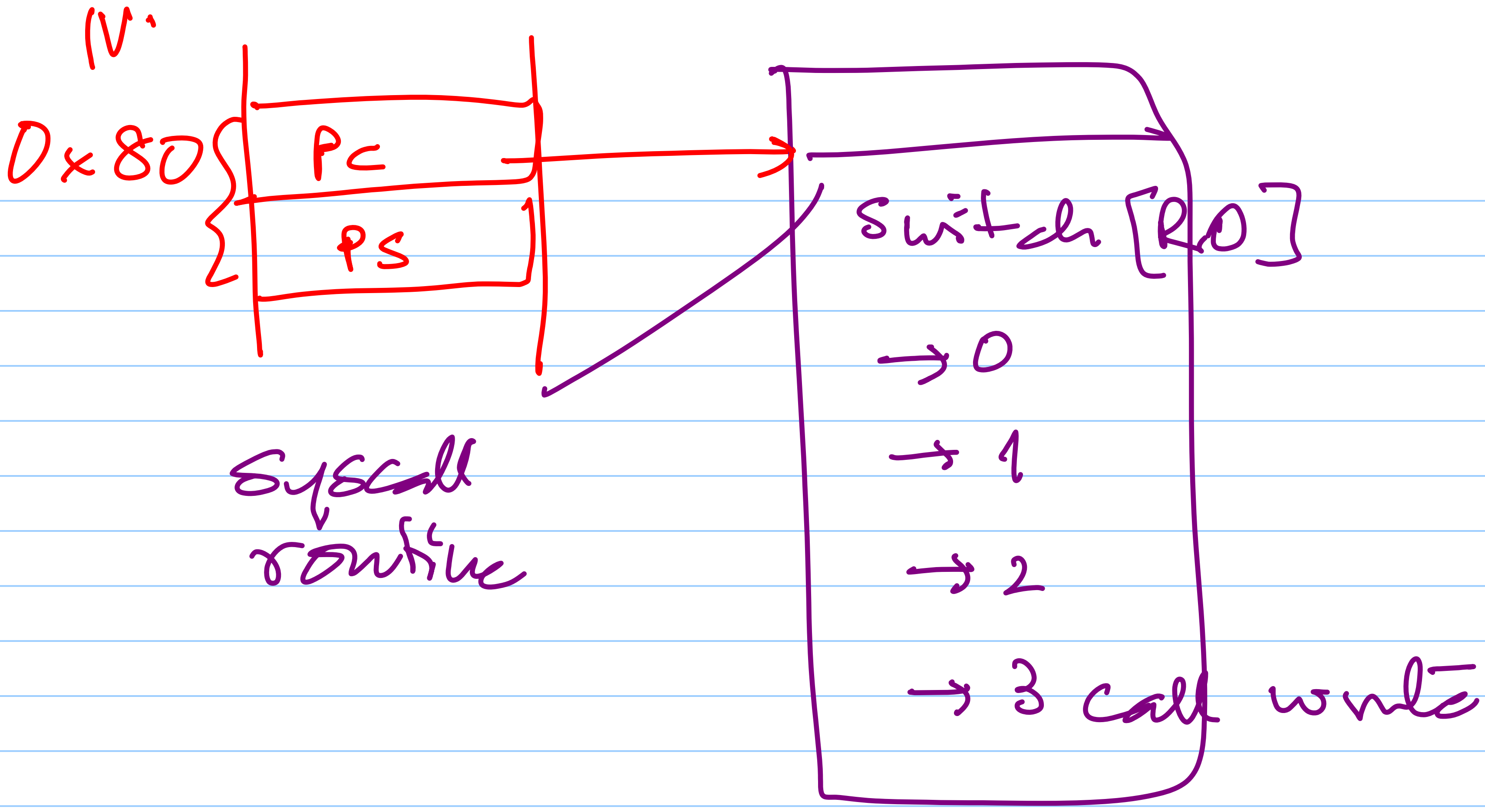
{ puts "hello world" into mem & gets a ptr
& move ptr to R1

puts a syscall code for write into R0

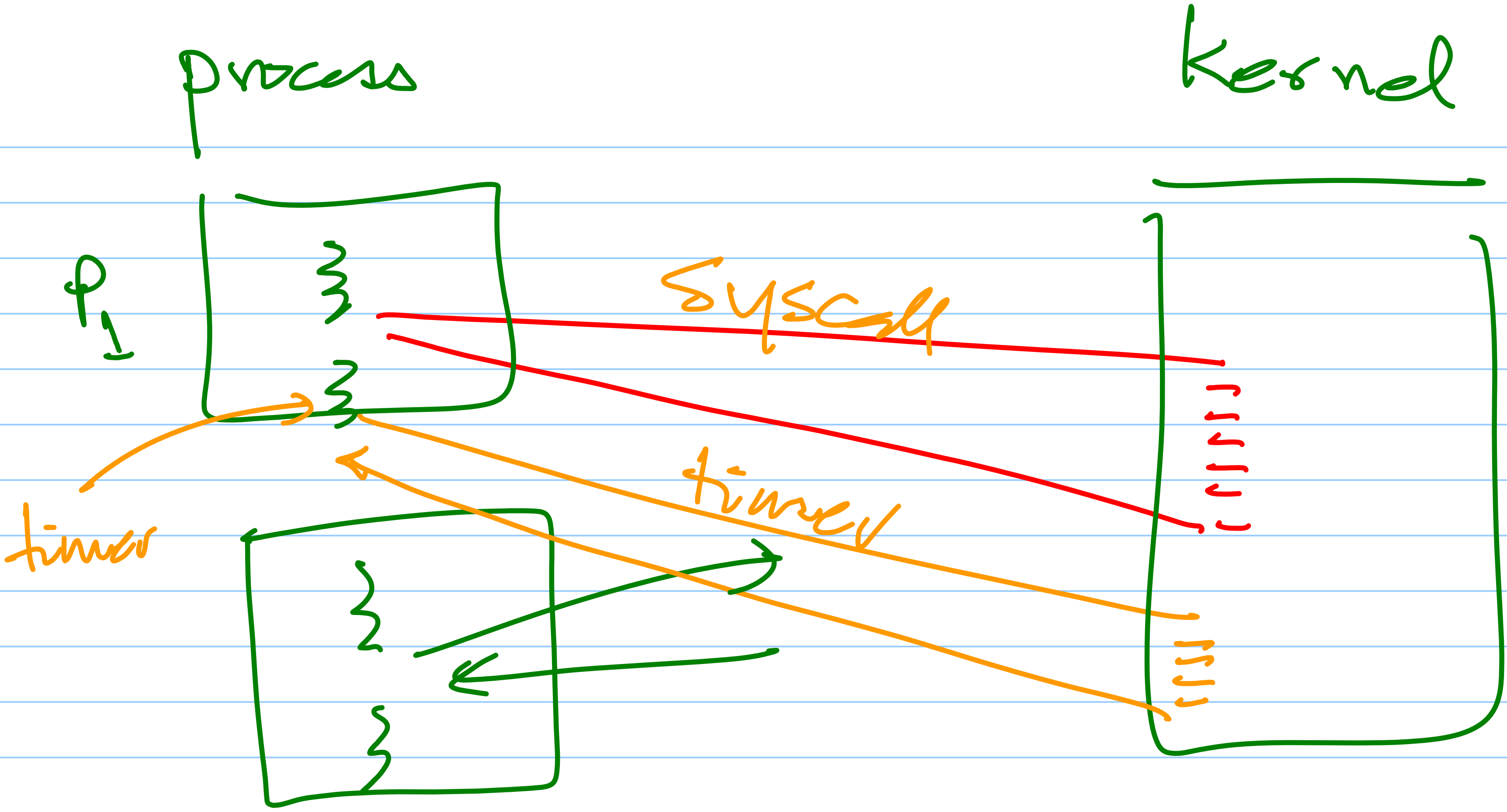
printf { sets up R0, R1

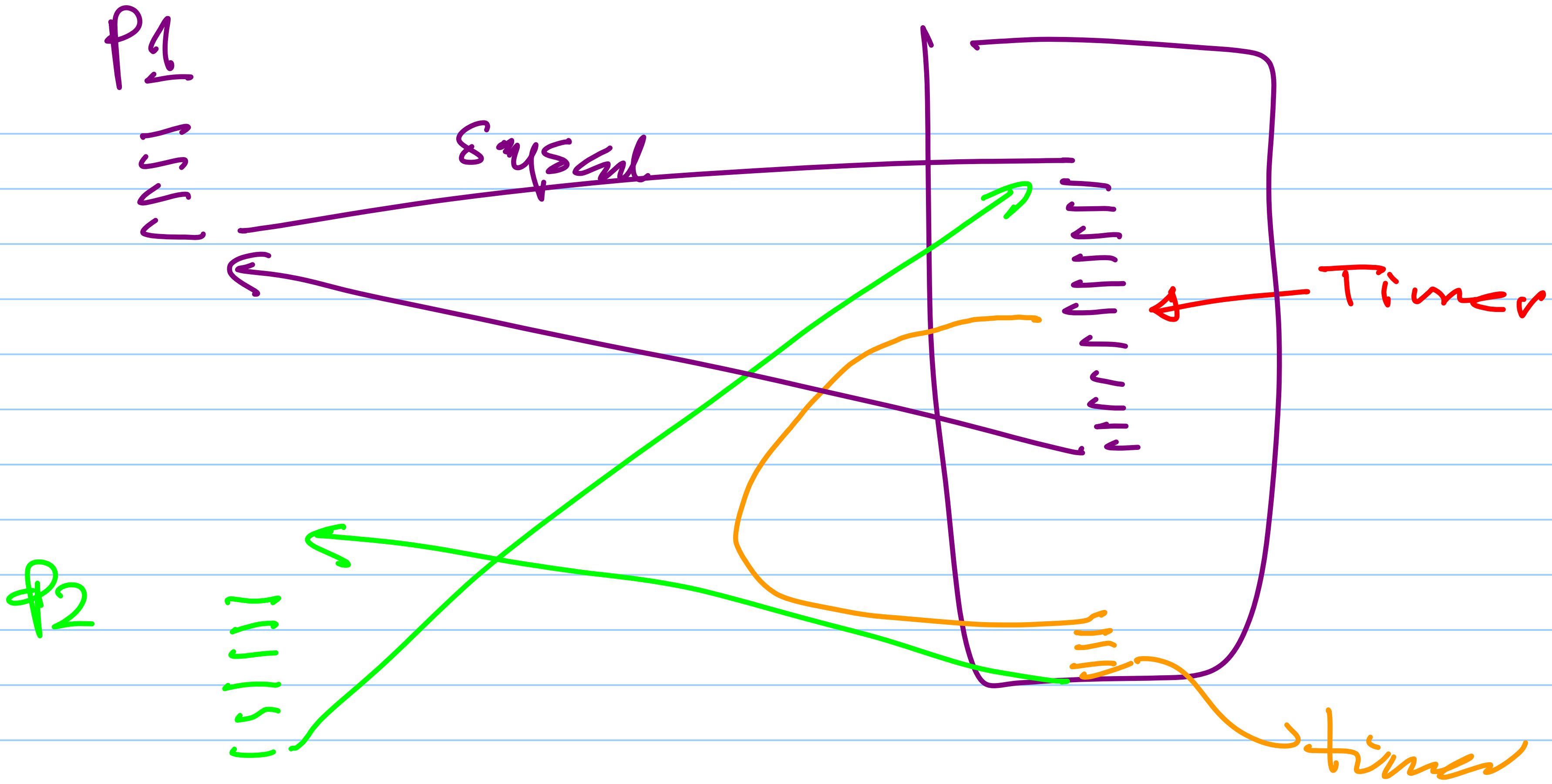
INT 0x80 $\rightarrow$ generates an interrupt numbered 0x80

$\longleftarrow$ Call to the ISR x80


IV:
0x80
PC
PS
syscall routine
switch [R0]
→ 0
→ 1
→ 2
→ 3 call write

```
write() {                    = finds output
                                 device
   ↑
kernel                        = sets up output
routine                         ↳ use device
   ↓                                driver
priv mode                       = // done
                     }
```


process
kernel
P_1
syscall
timer
timer


P1
syscall
Timer
P2
timer