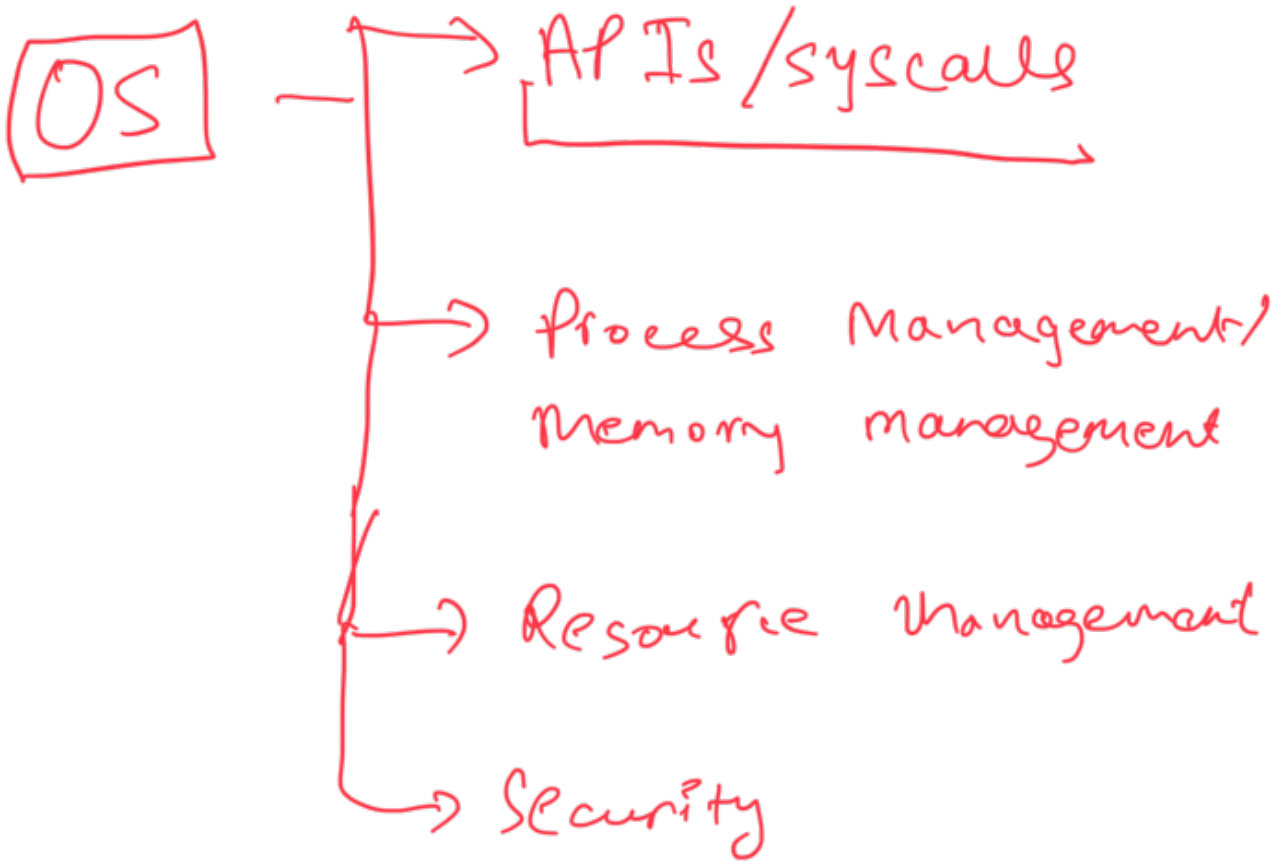
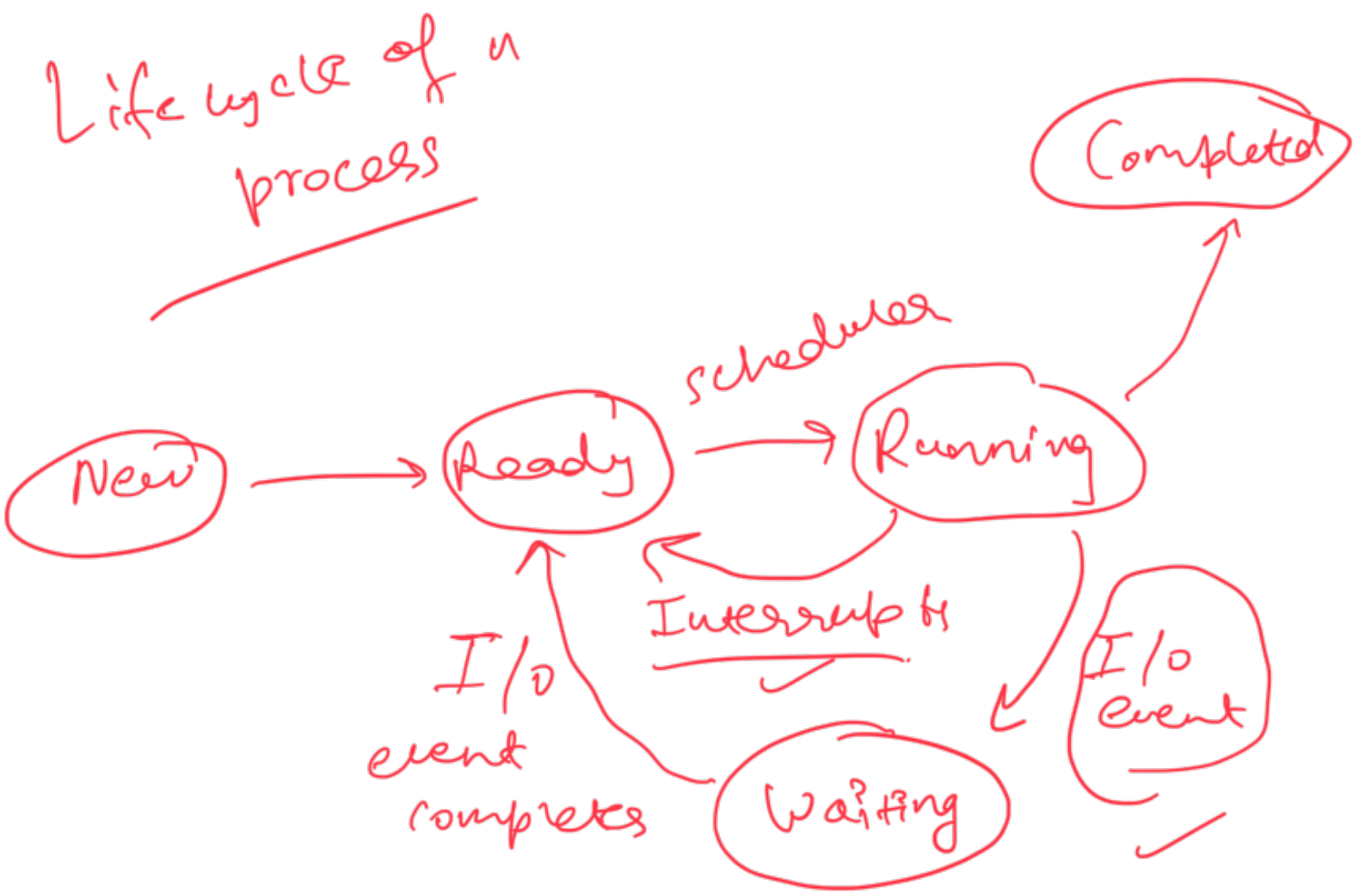
Operating Systems 2

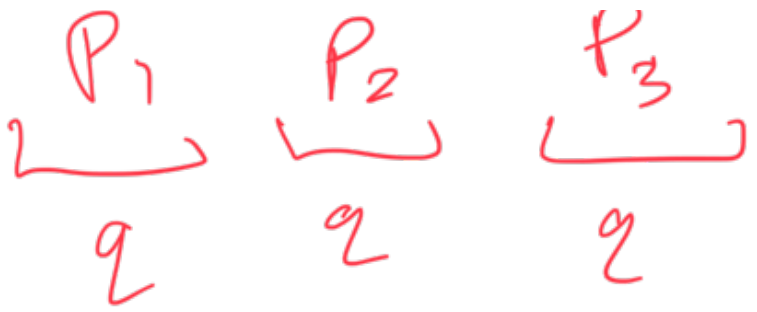


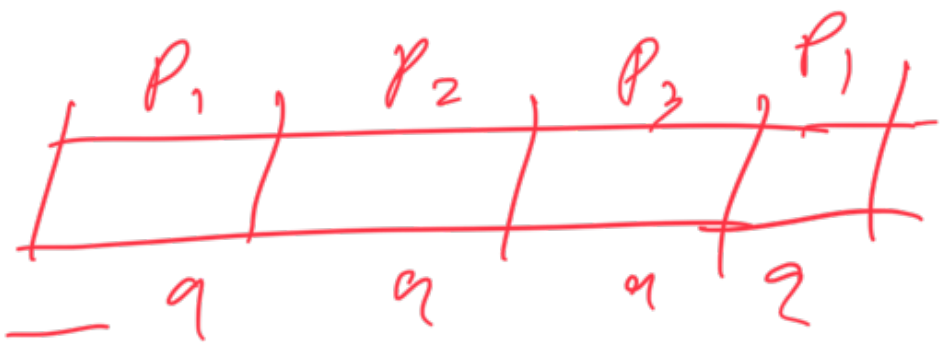
PCB

Process ≡ Program + PCB



Round Robin





Multiprogramming ⊂ (multiple

vs

multiprocessing — multiple programs.
multiple processors

Metrics

- burst time
- wait time
- arrival time

FCFS] Convoy effect

SJF] Starvation

* Preemption

Round Robin

$P_1 = 100$

$P_2 = 2$

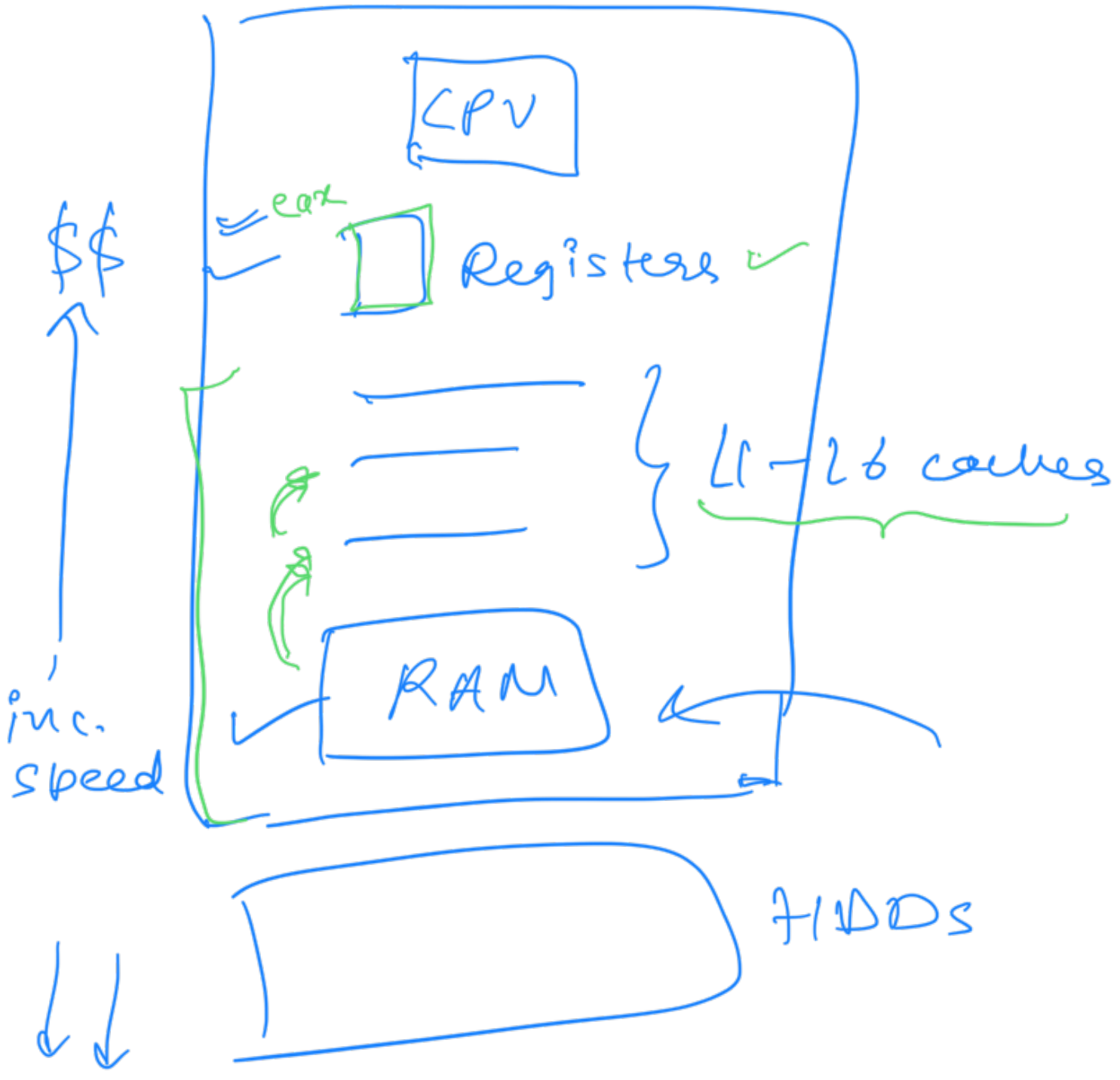
$P_3 = 1$

$P_4 = 1$ →

$P_5 = 1$ ↘

⋮

Memory Management

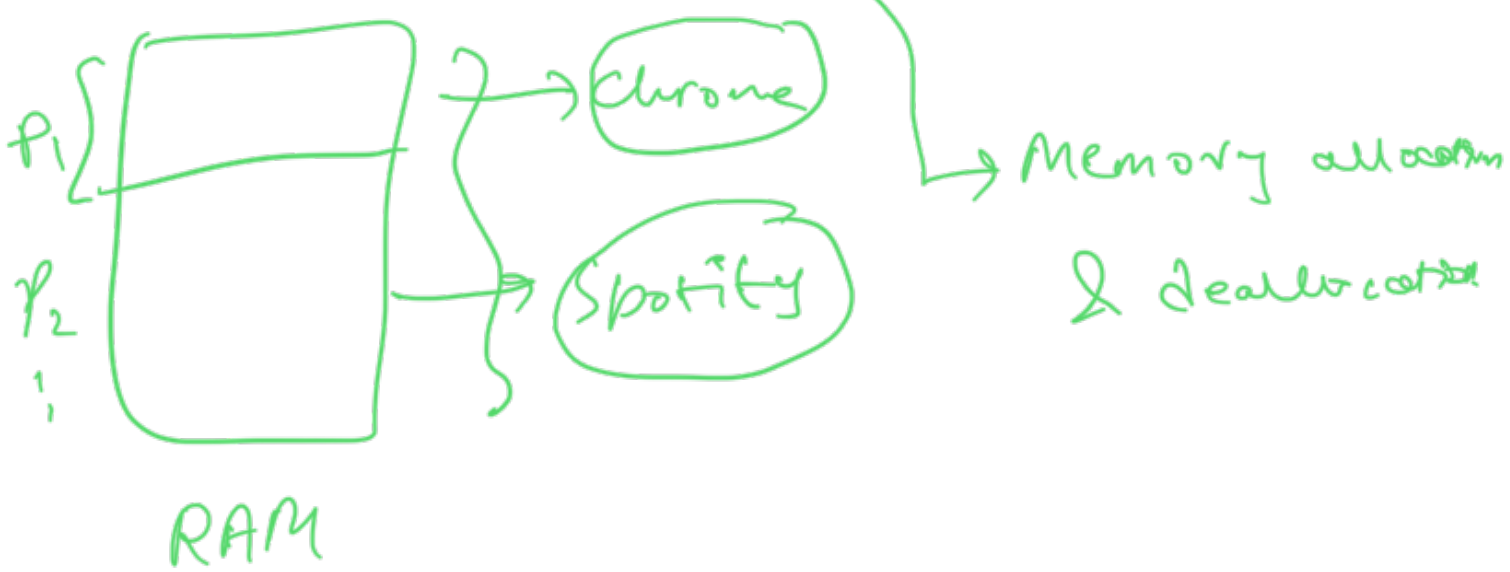


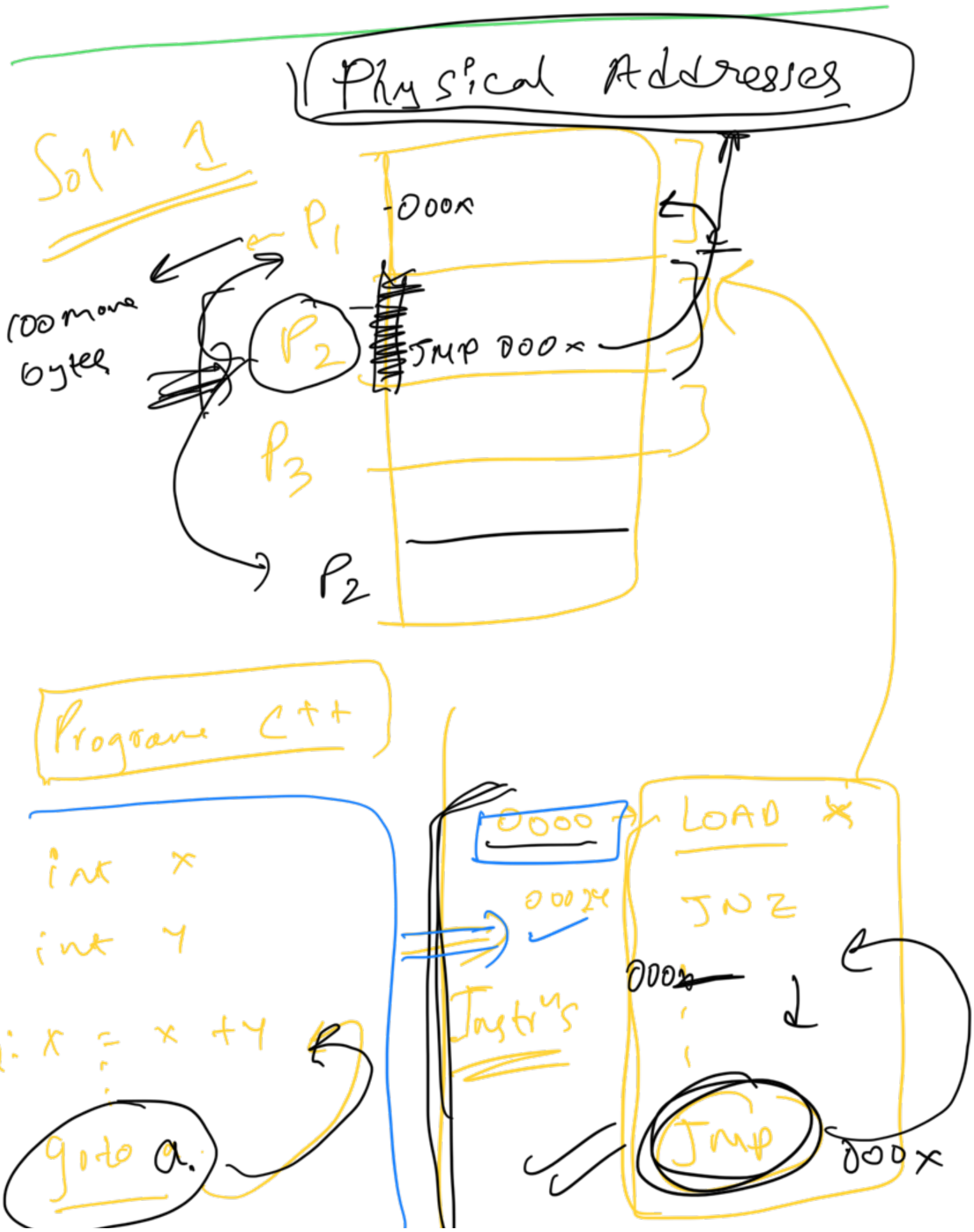
* CPU has to bring programs into RAM to execute them.

→ Multiple processes running in ||.

Secure.

Memory Isolation

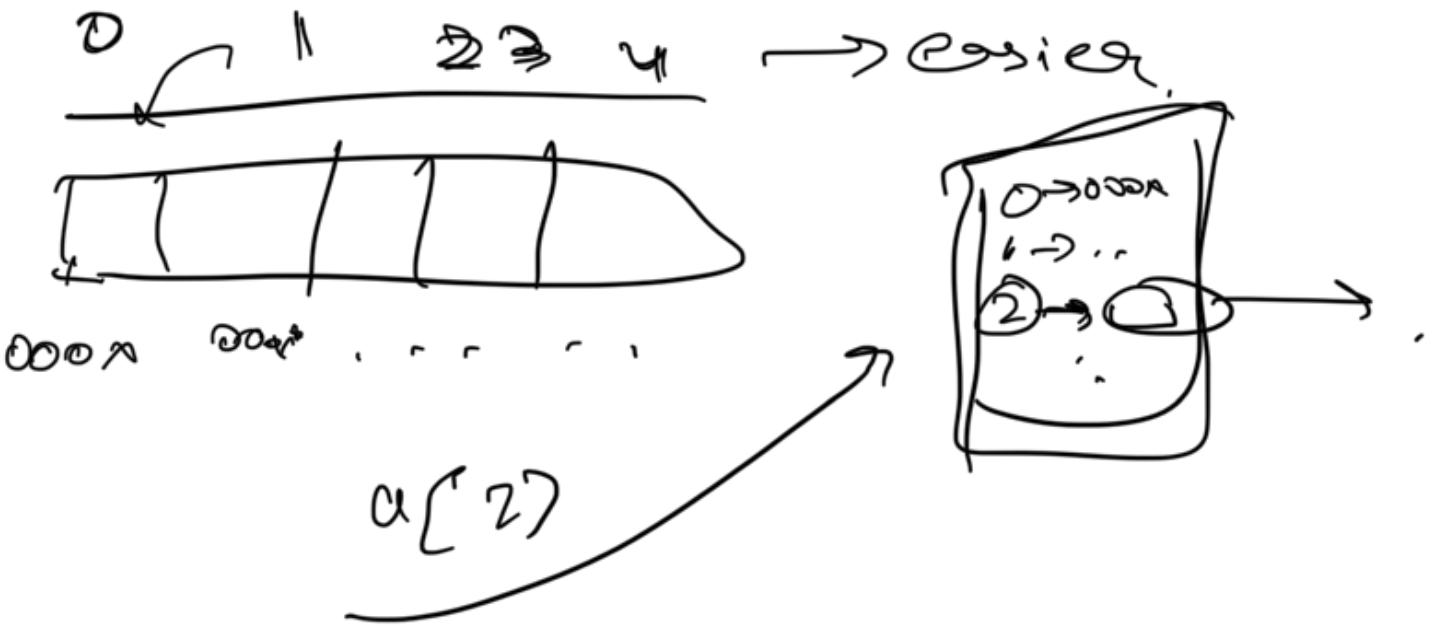


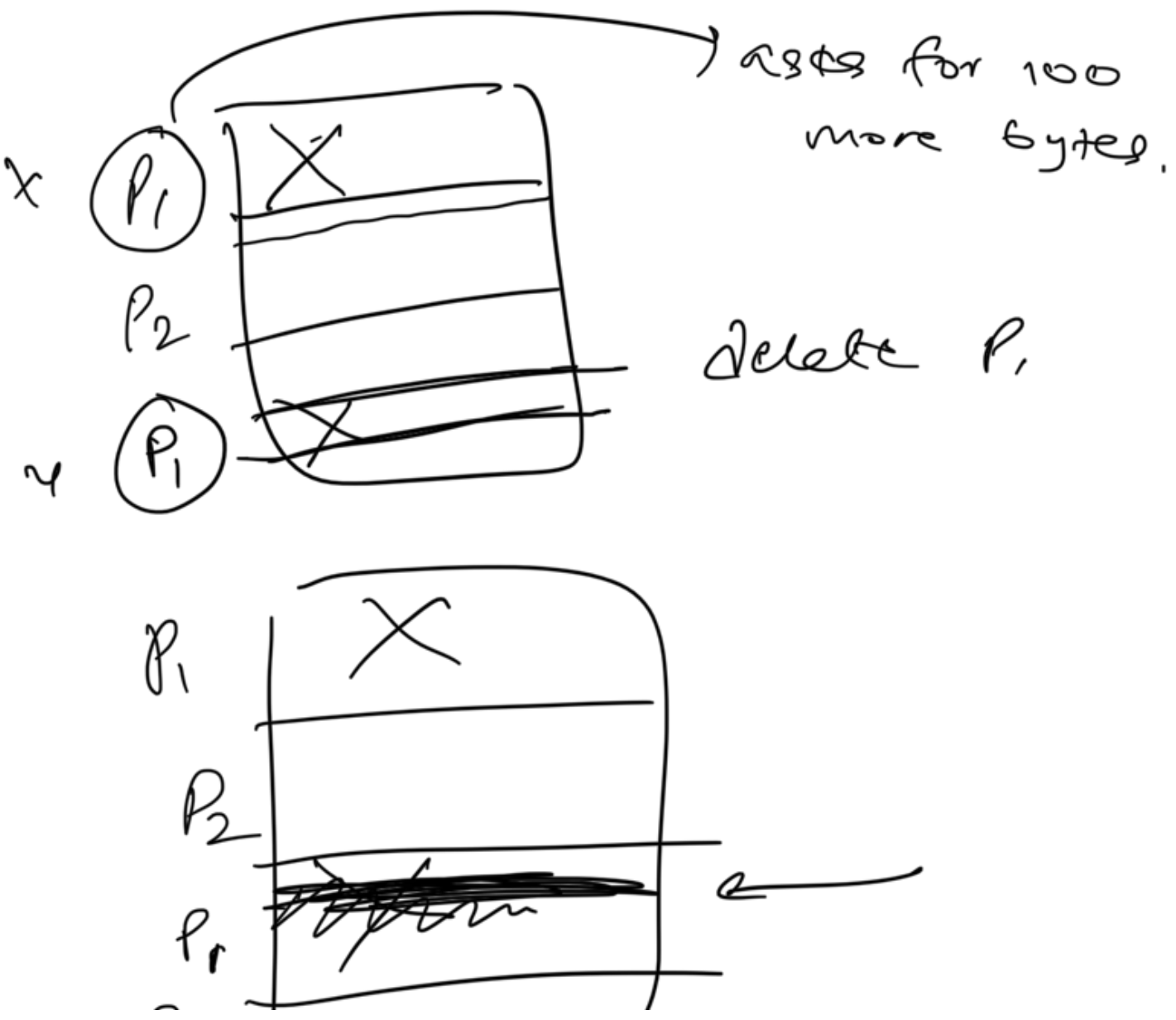


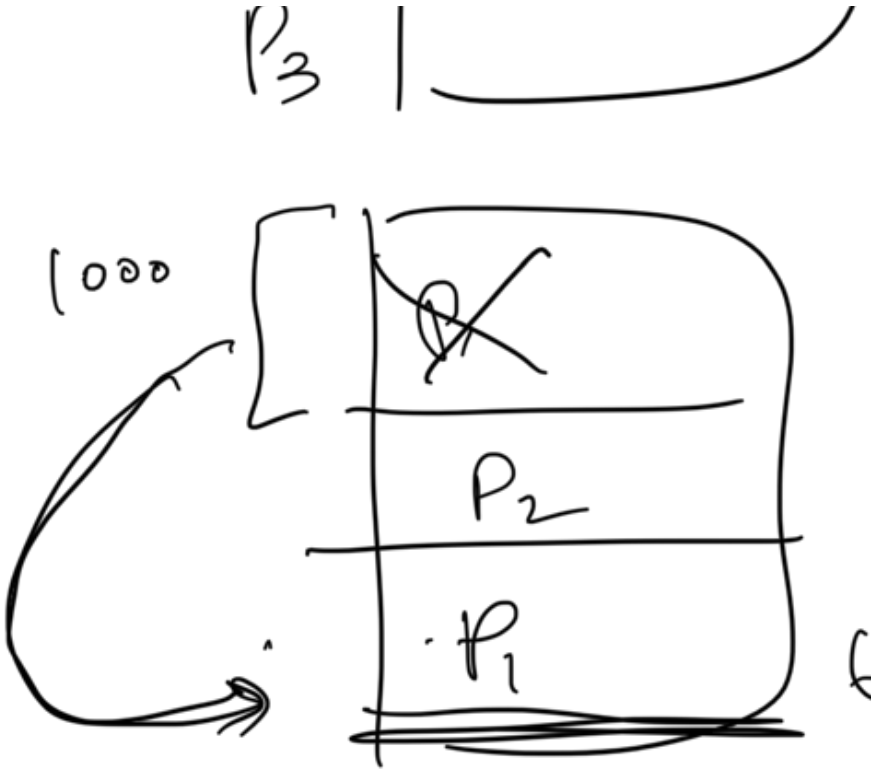
mapping

+ offset.

OS → forced to create virtual memory for each process.







a) Takes lot of time

b) Contiguous Stretch of bytes might not be there

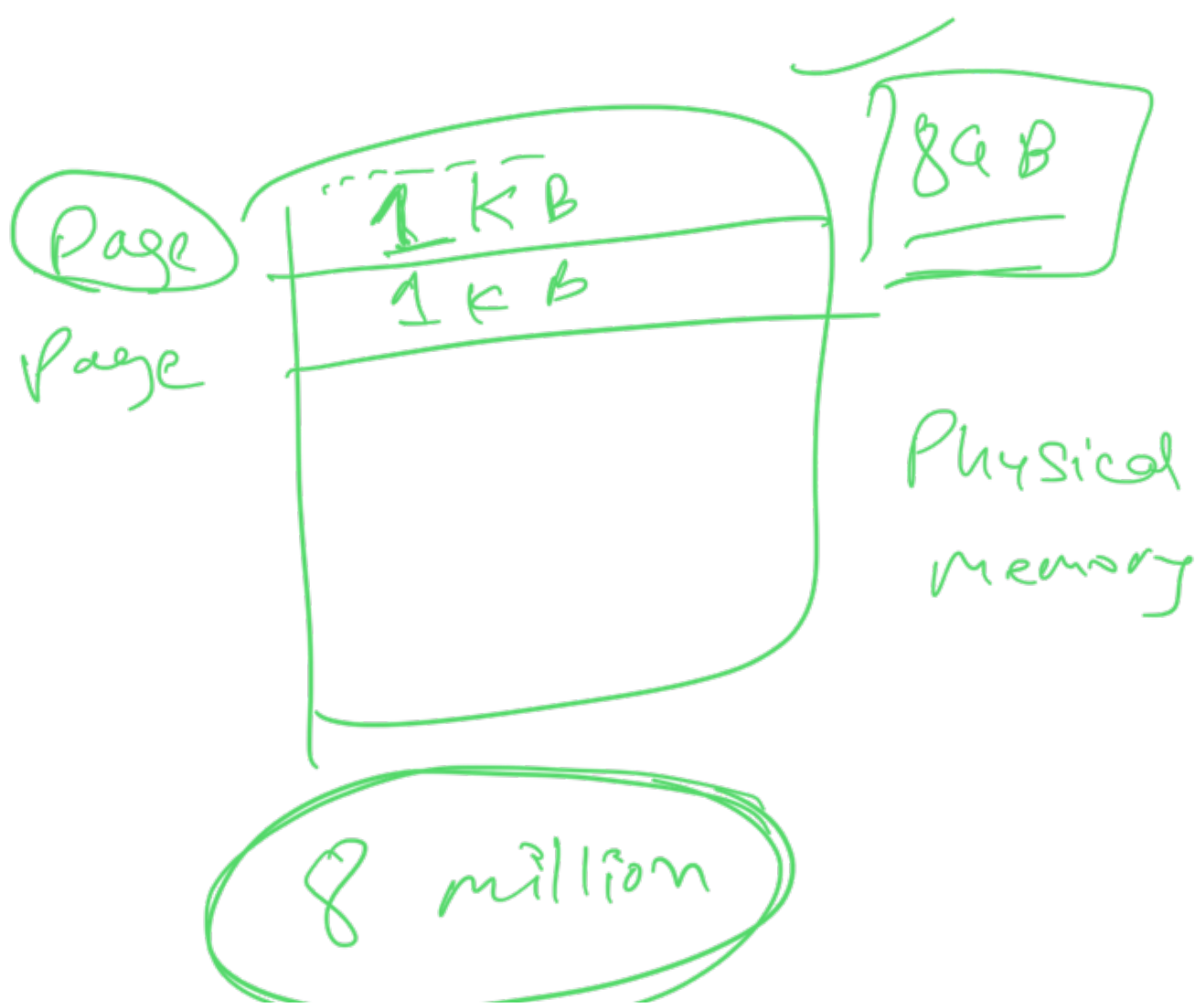
500

500

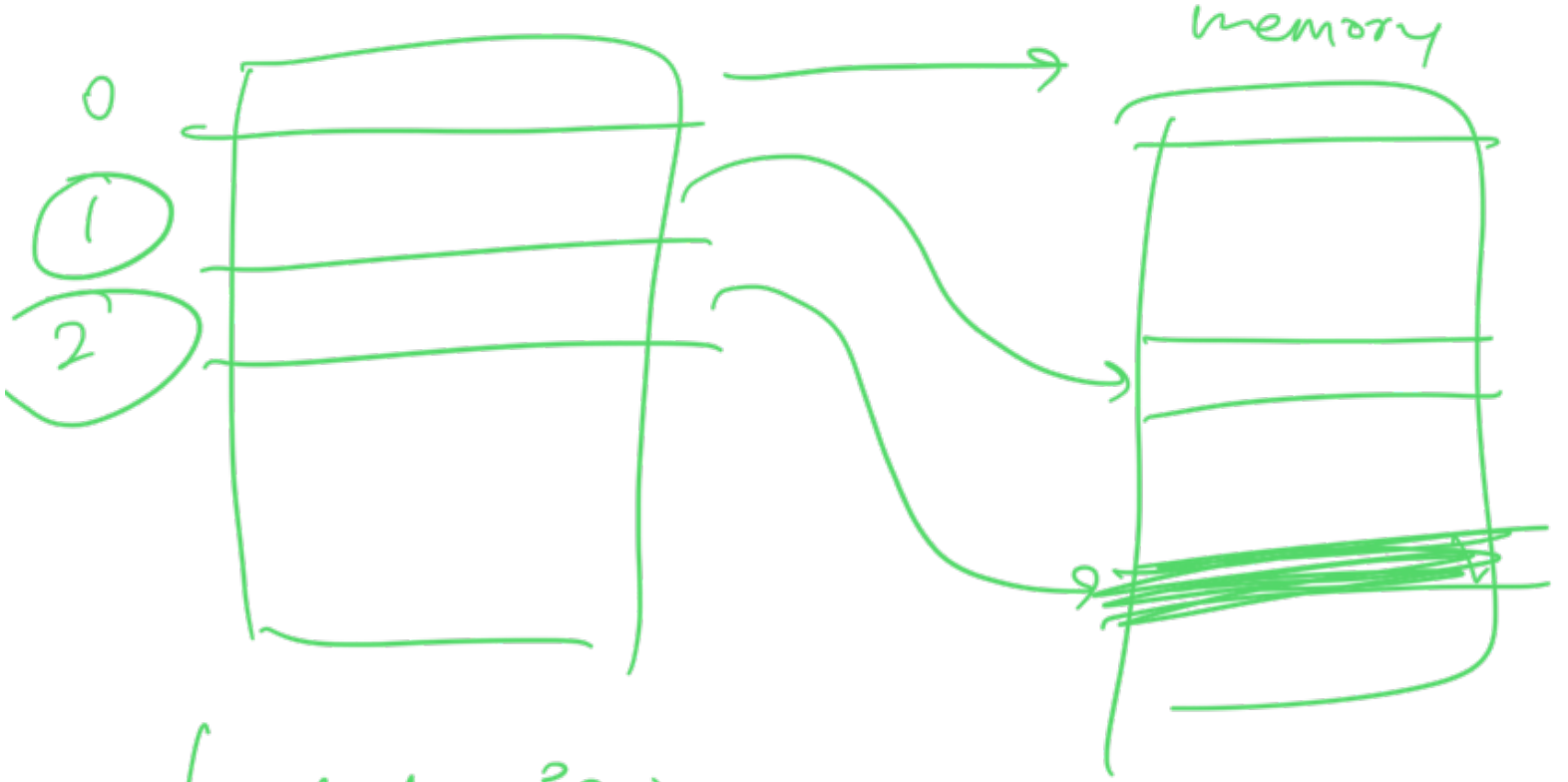
1000 bytes

Logical Memory

↳ create pages.



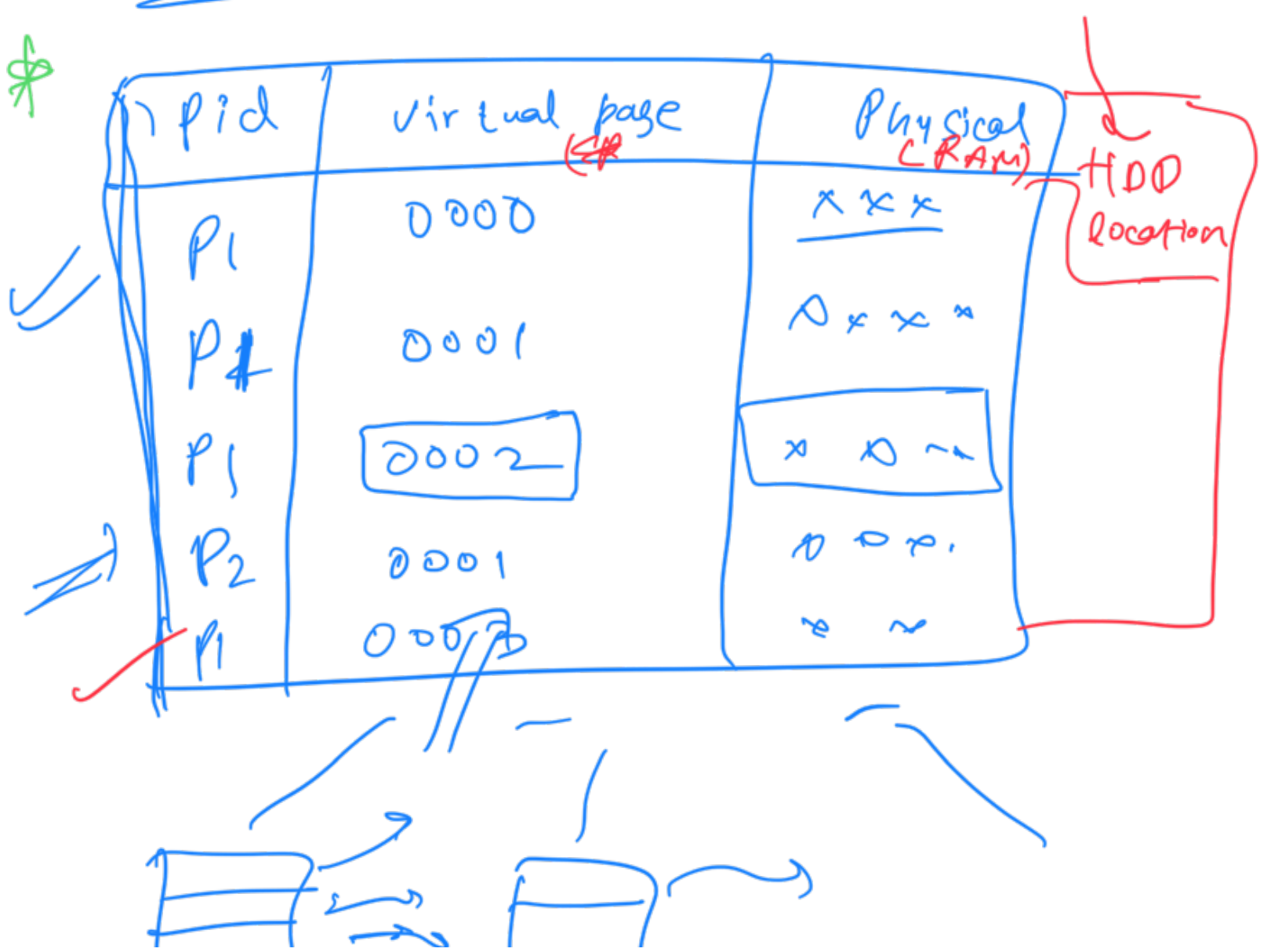
For each process, maintain a virtual memory.

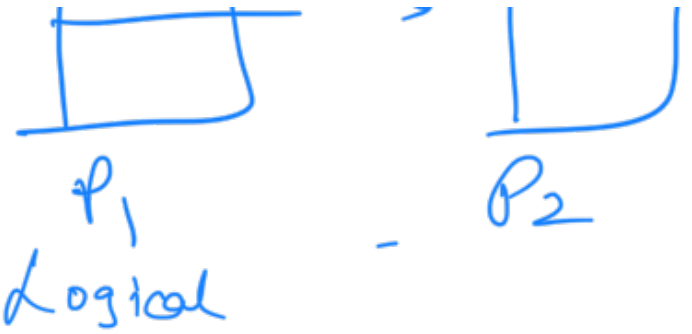


Logical view

Per process.

Process Page Table



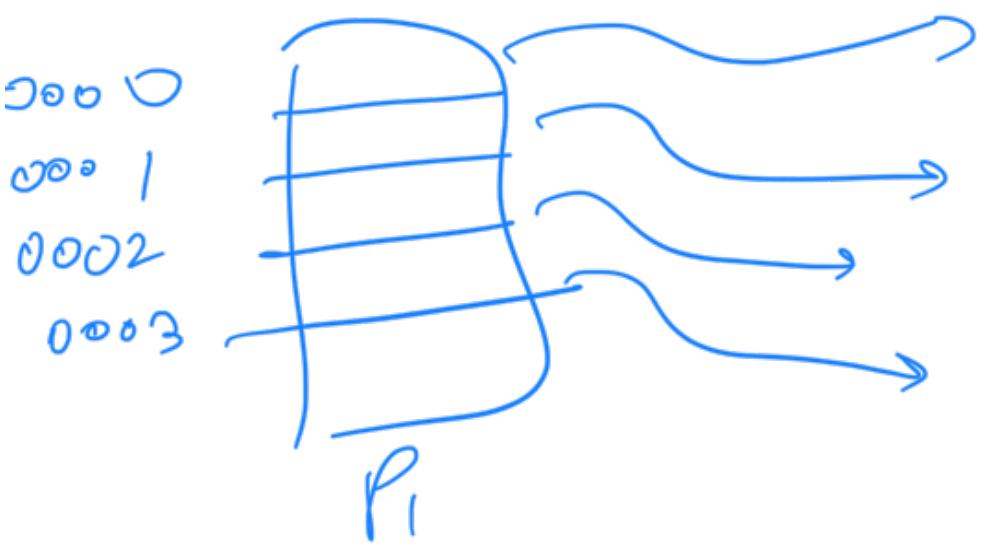


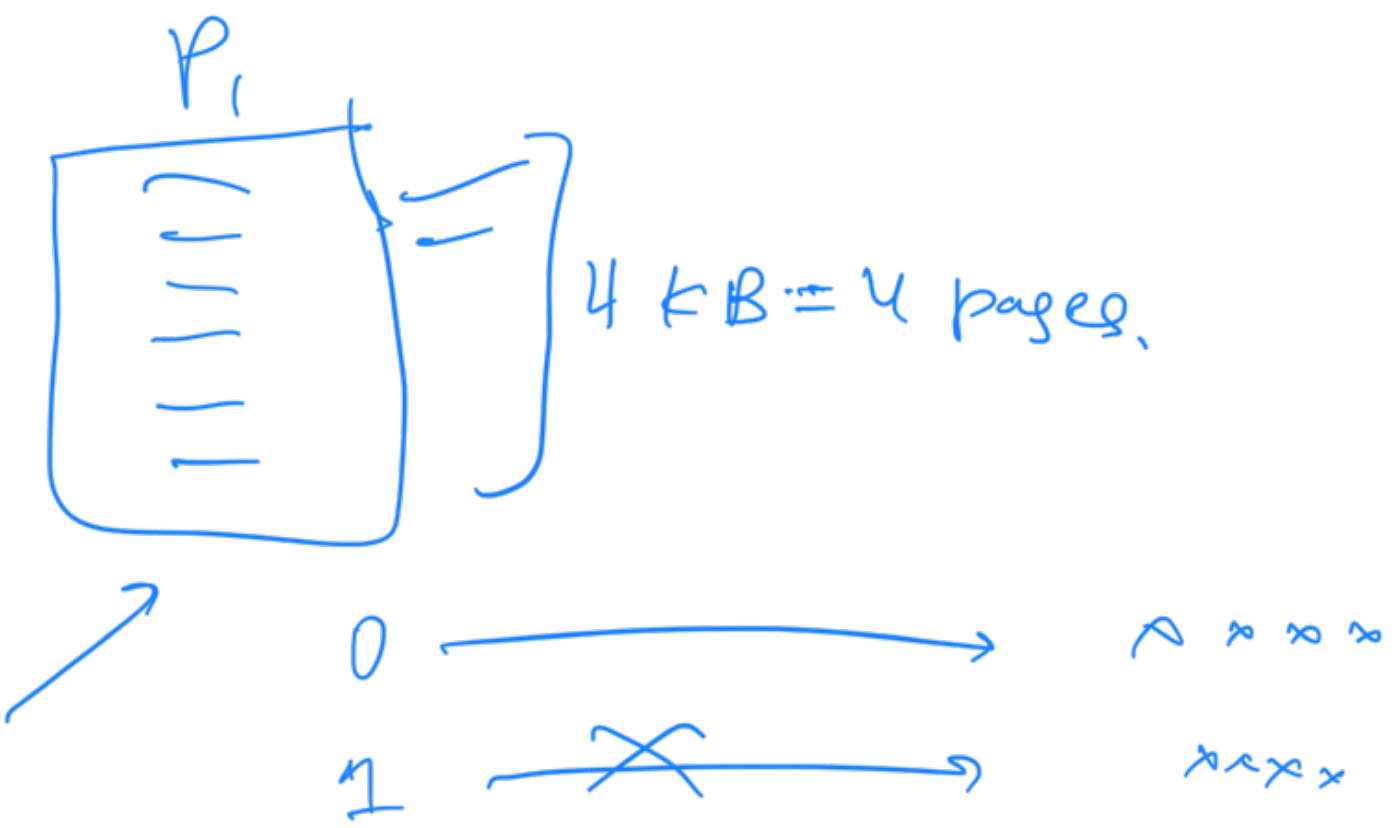
Logical

P1

2 move pages.

3 pages

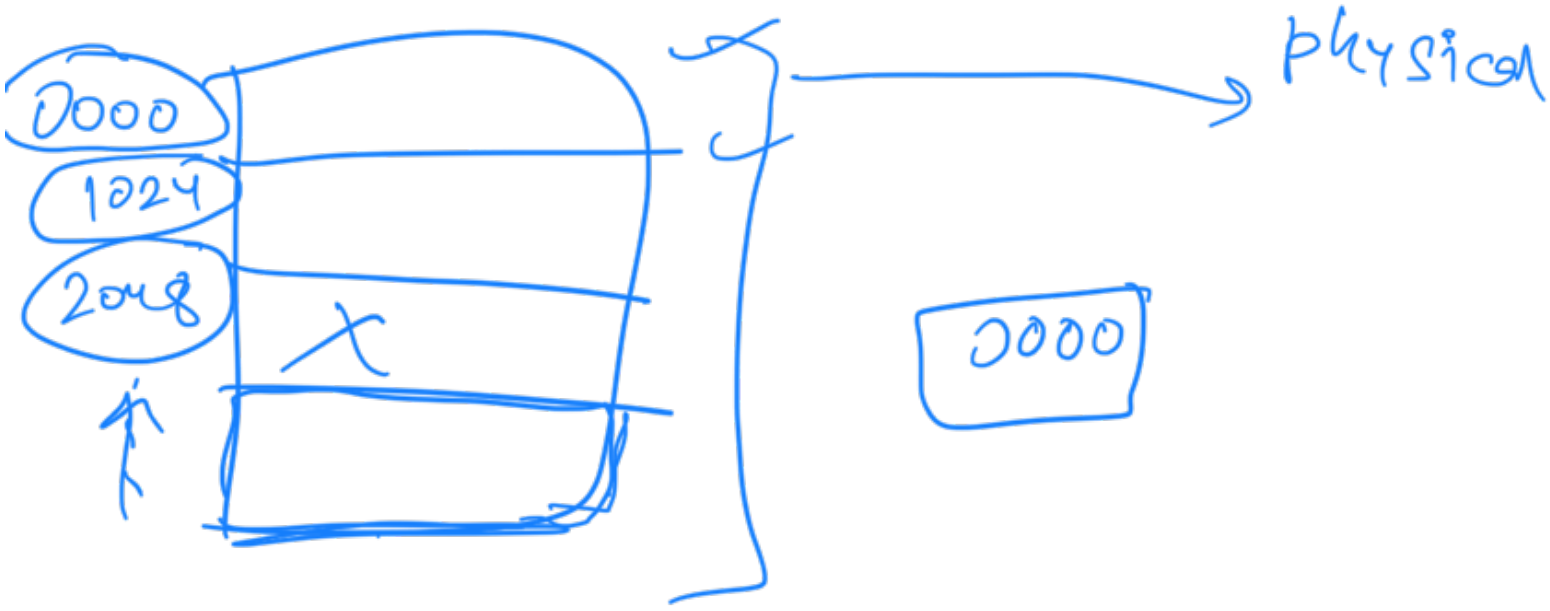


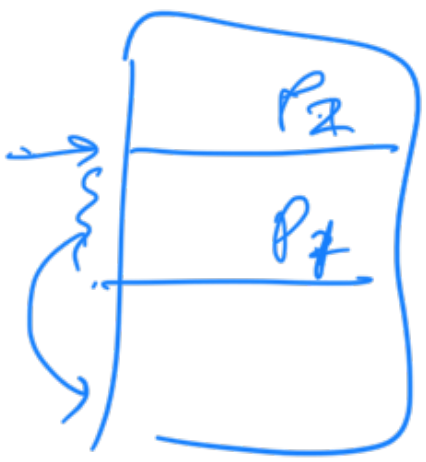


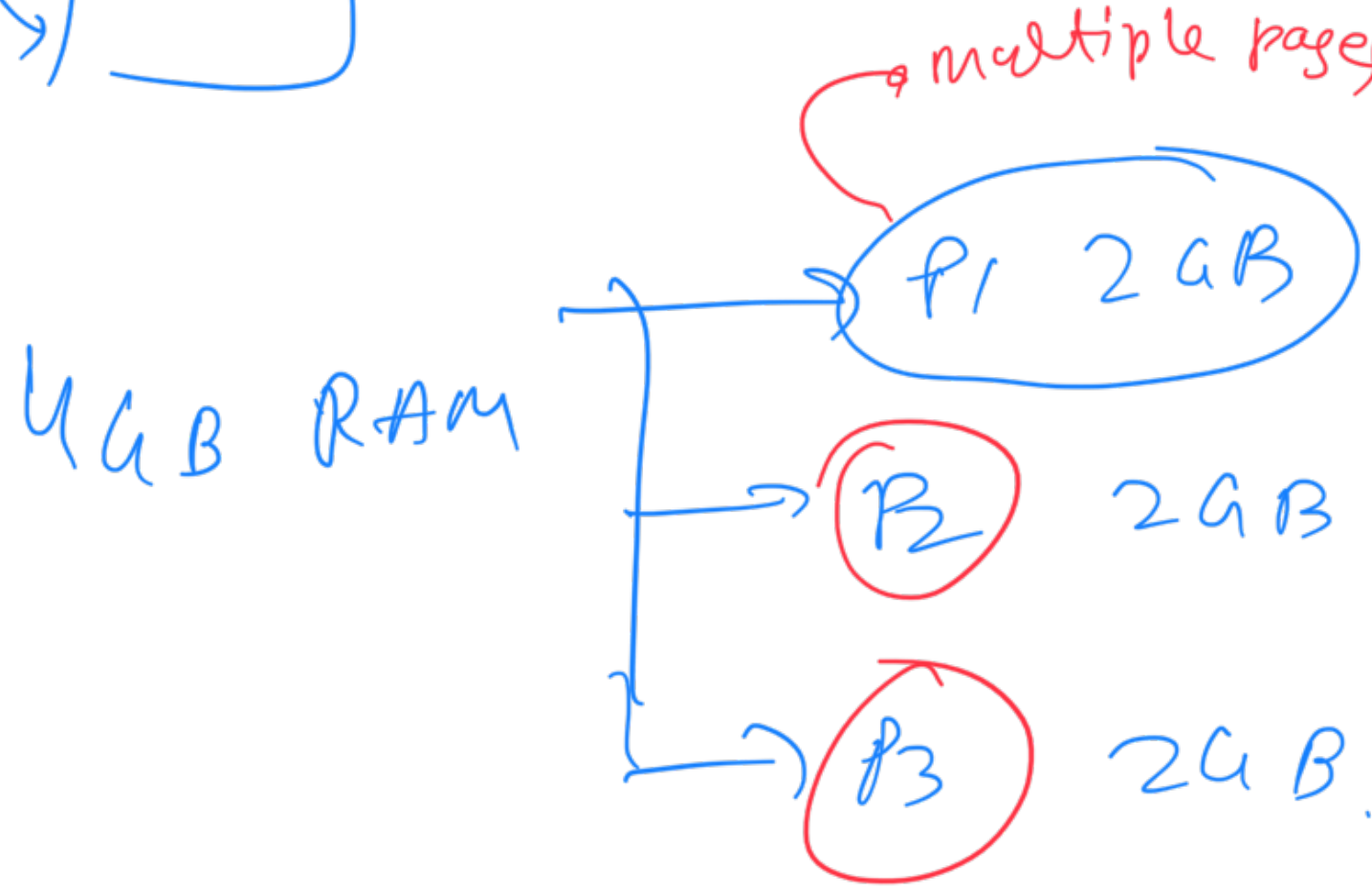
assembly

0
1
2
3
4

virtual addresses







Paging concept — you would only need a few pages at a time.

L16 RAM HDD

Pid	Virtual	Physical addr. in RAM	HDD location
P1	0000	xxxx	—
P1	0001	null	35...
P1			
P2			
P2			

Bring the page from HDD to the RAM.

↳ might also have to move a page out.

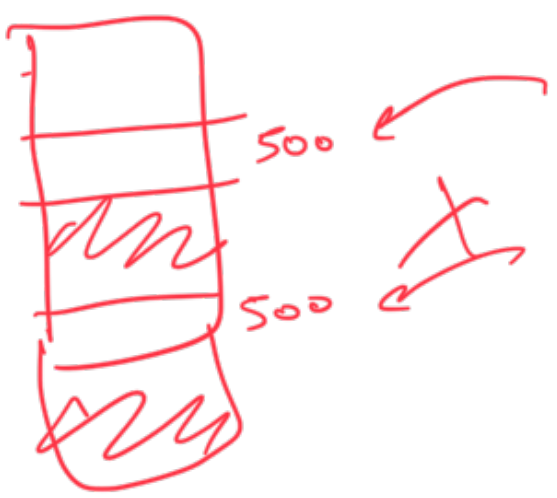
Page fault

Promise: — min (the amt of

of page faults).

→ Physical Memory

→ contiguous
a) time
b) 500 bytes
500 bytes
still cannot
store.



→ complex
Translation.

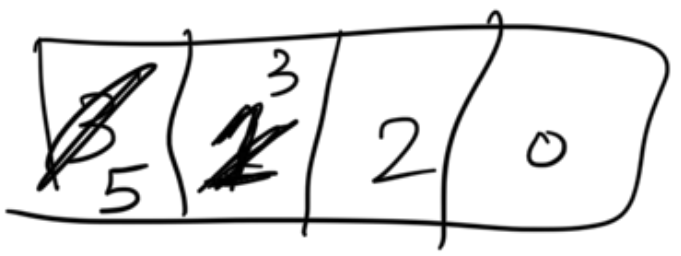
→ if you don't keep it contiguous, FRAGMENTATION

→ a) pages
b) Logical → Physical

Page Fault ≡ HDD → RAM

a) FIFO

Requests for pages → 3 1 2 0 5 0 1 3 2

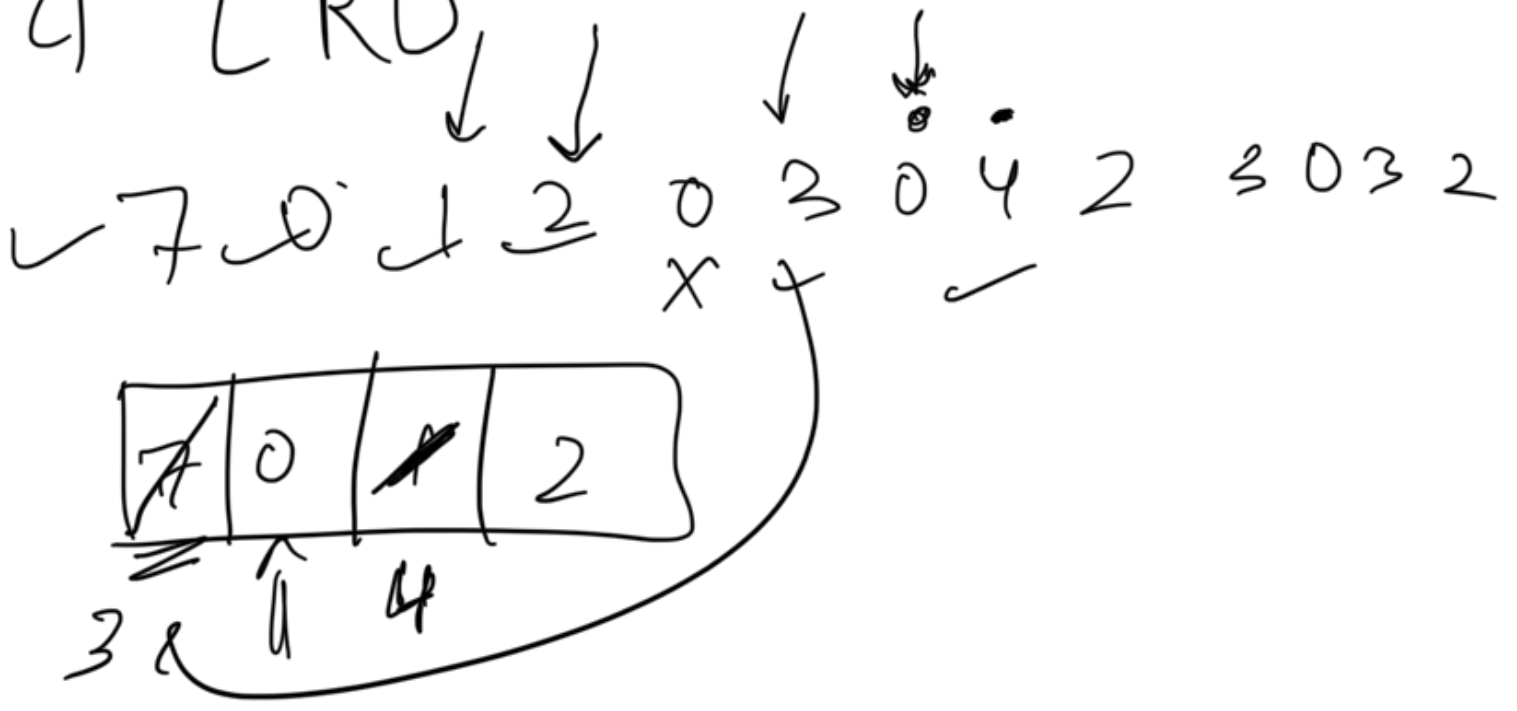


RAM
(4 pages)

6 page faults

b) LIFO | bad memory access follows locality of reference.

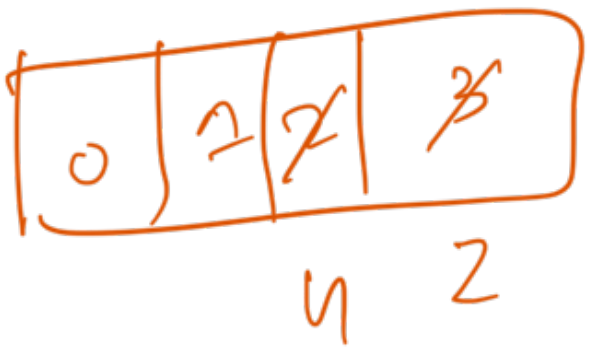
c) LRU



d) LFU

Optimal ?

Requests

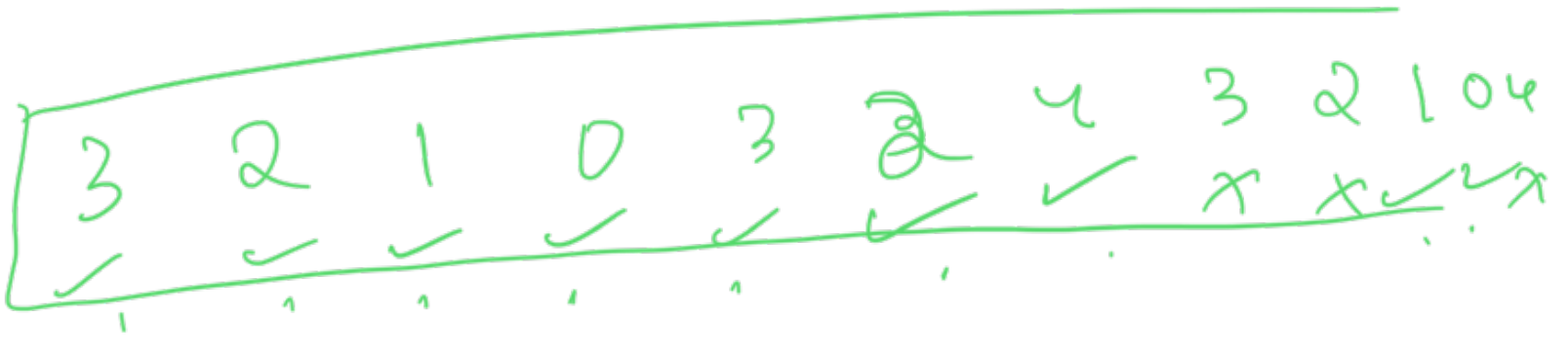


Belady's Anomaly

Inc. the # of pages in RAM.
you would expect lesser #
of page faults.

FIFO

3 →

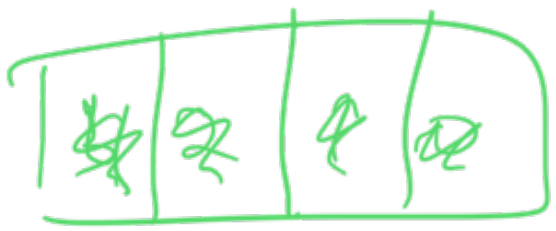


3 pages

9 page faults.

3 2 1 0 3 2 4 3 2 1 0 4

✓ ✓ ✓ ✓ x x ✓ ✓ ✓ ✓ ✓ ✓



4 2 2 1

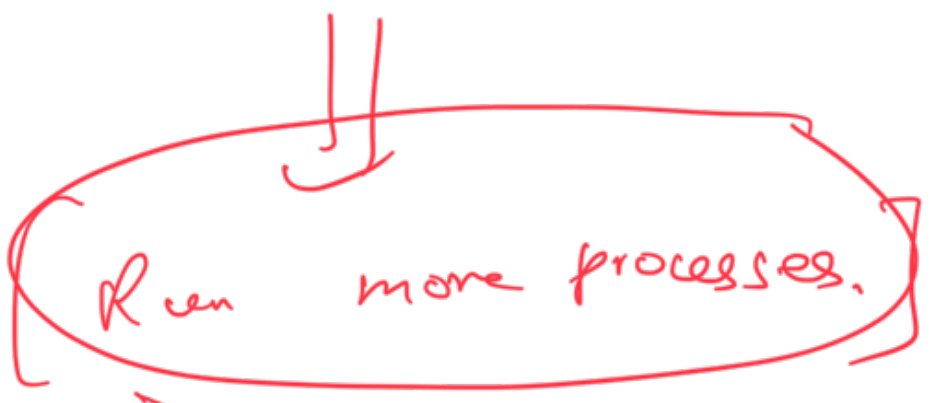
0 4

10 page faults.

↳ (doesn't happen with LRU / Future.)

Old OS

CPU utilization ↓



P1 1 2 3 4

P2 10 11 12 15

P2

CPU → Round robin

P2

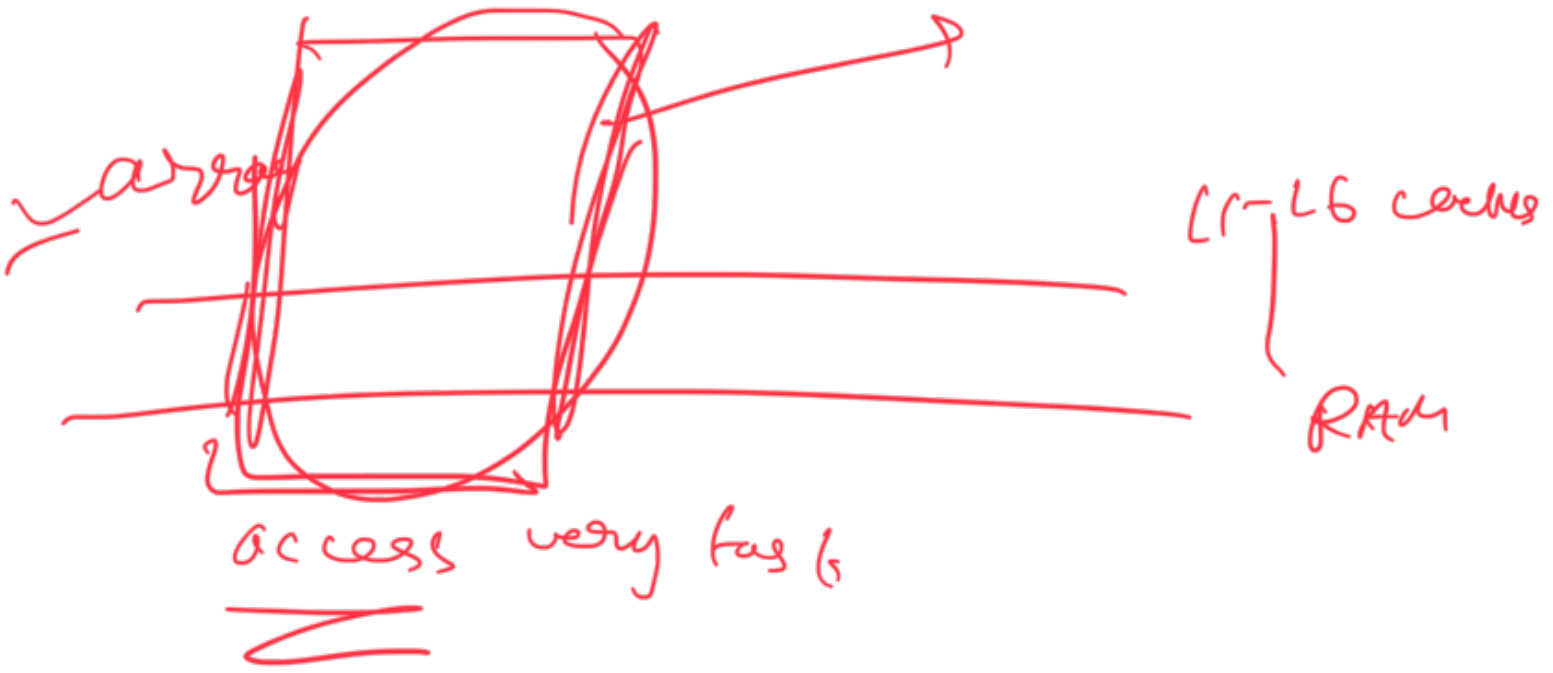
↑ page faults.

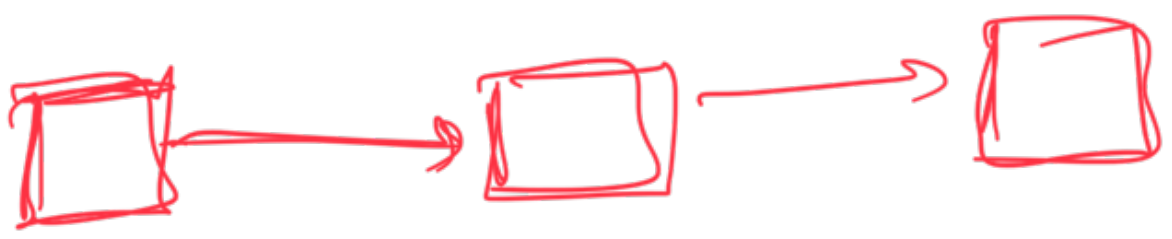
||

↓ CPU utilization.

Thrashing

To memory always comes in blocks.





name

⇒ age

}

[ ] students.

[ ] age. ✓ ✗ ✗

OS → User space

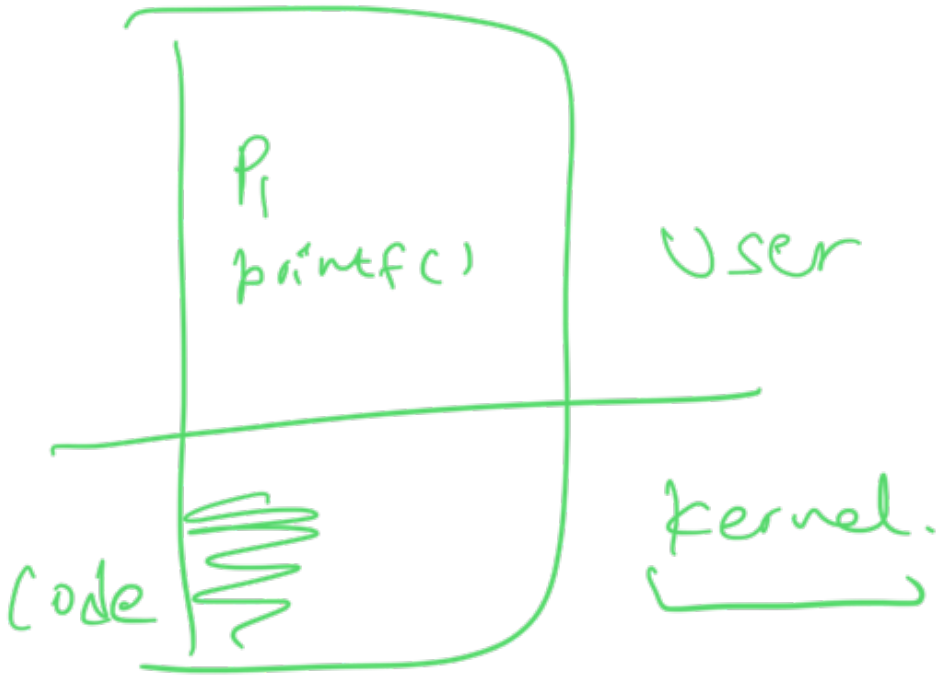
less privileged

Ring 3

Kernel space }

more privileges

Ring 0



→ Syscalls: write( ) | fork( ) | exec( )

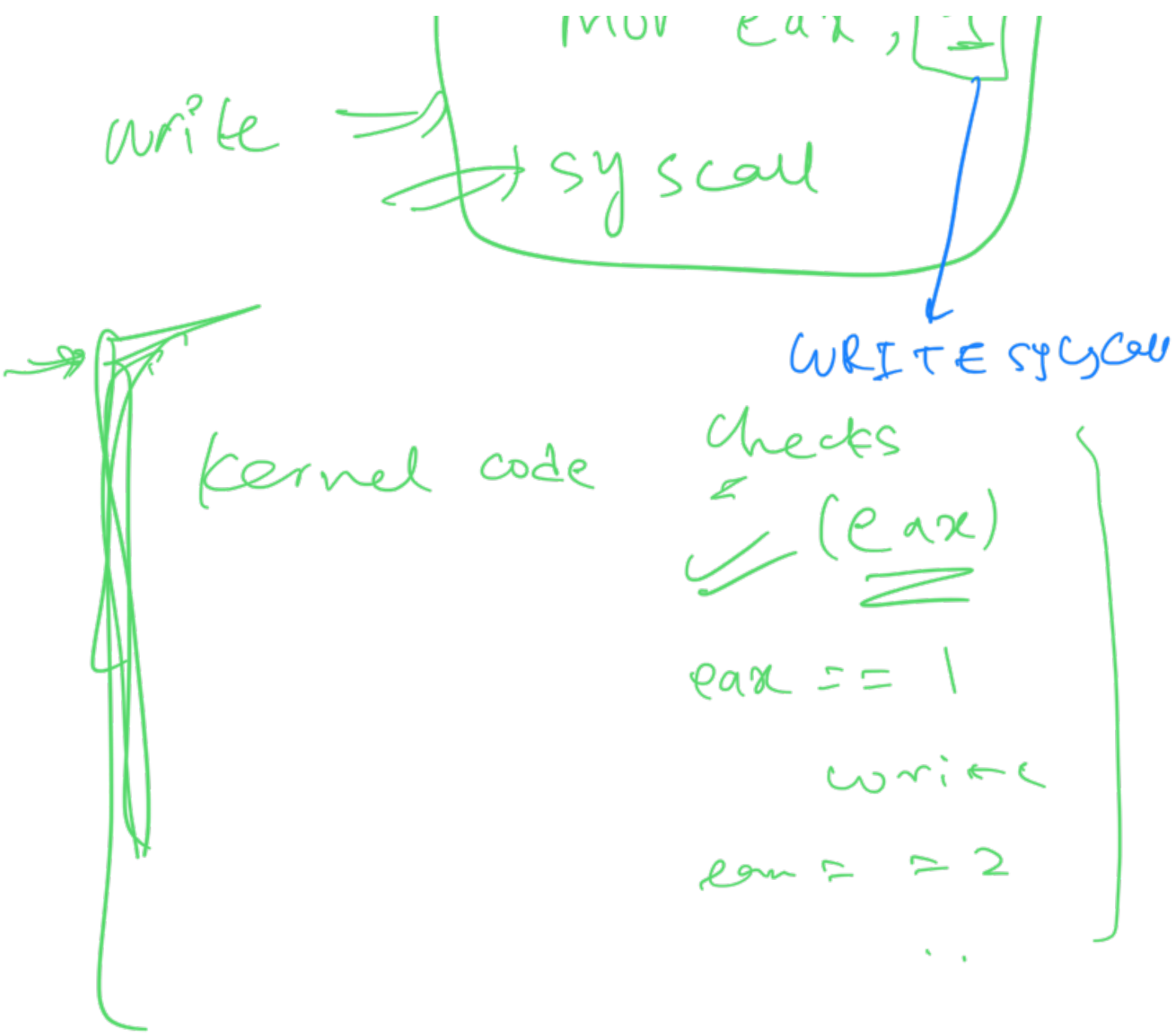
Kernel → on boot up, CPU starts at level 0 (most privileged)

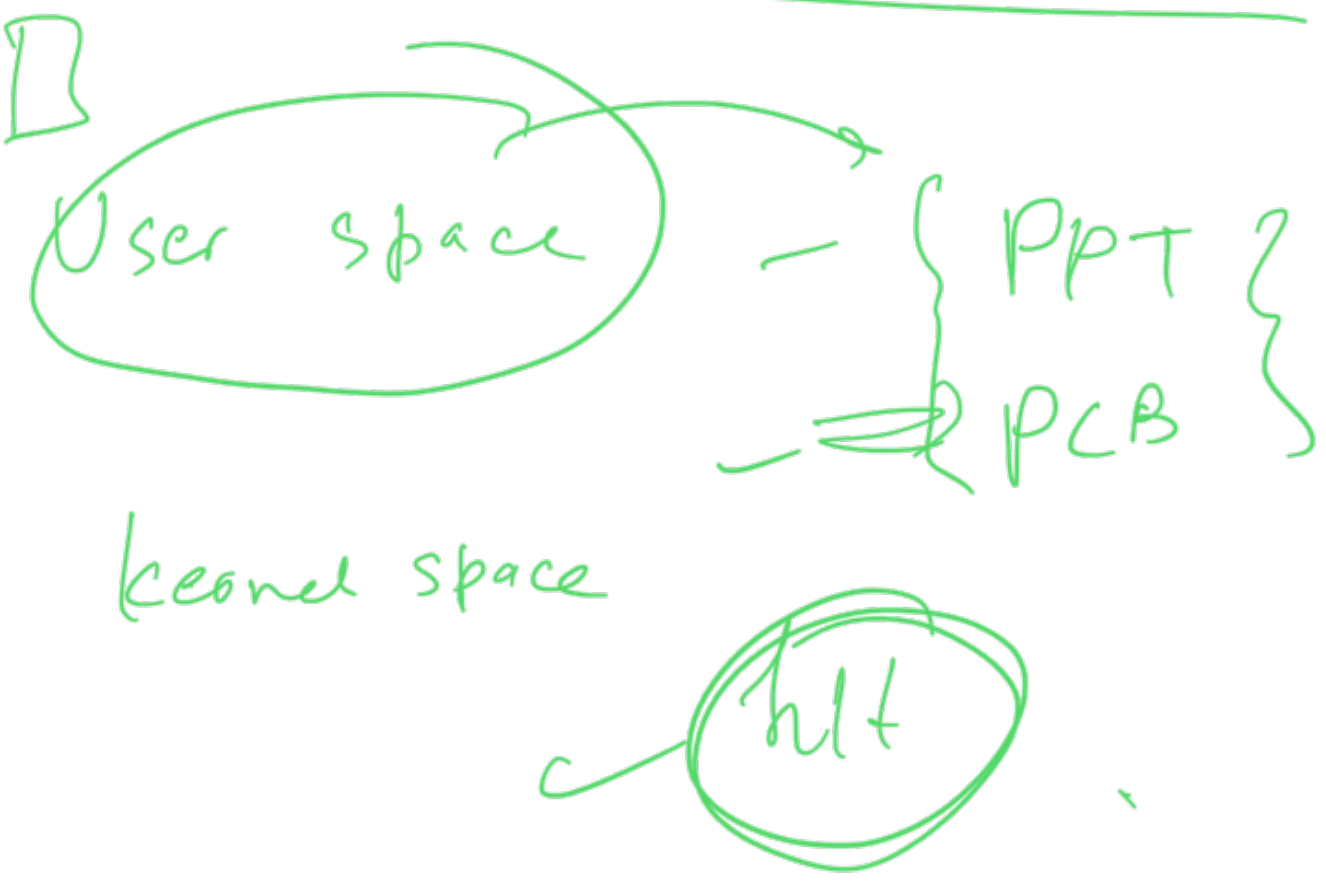
IO ∞ ↳ (S).

IA32_LSTAR → <kernel address>

* HW get locked down.

* Can enter back through Syscall - BUT you can't control what gets executed - since the address is fixed.





Kernel :- Core of a computer's OS.

Manages hw in protected space.

Tables

fork ——— executing

H/w interrupt :- by H/w device to signal that they need attention from OS.

S/w interrupt :- Syscall.

printf()

write(fd, _, _)

[Interrupt [timer, keyboard, I/O, syscall)

CPU stops executing → goes to kernel mode

Interrupt

Restores state.

does ops

handler,

Same state of CPU