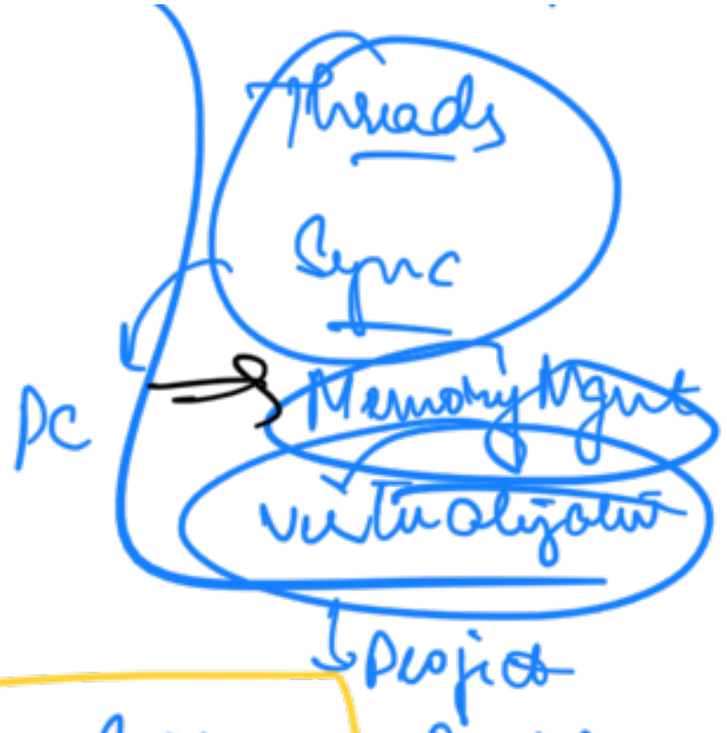
OPERATING SYSTEMS : Processes



Agenda

→ What are OS

→ Uni VS Multiprogramming Systems

→ Processes

→ PCB

→ Interrupts

→ CPU VS I/O Bound Processes

→ CPU Scheduling

→ Scheduling Algos

→ FCFS

→ SRTF

→ RR

How the applications that we develop work under the hood

What is an OS

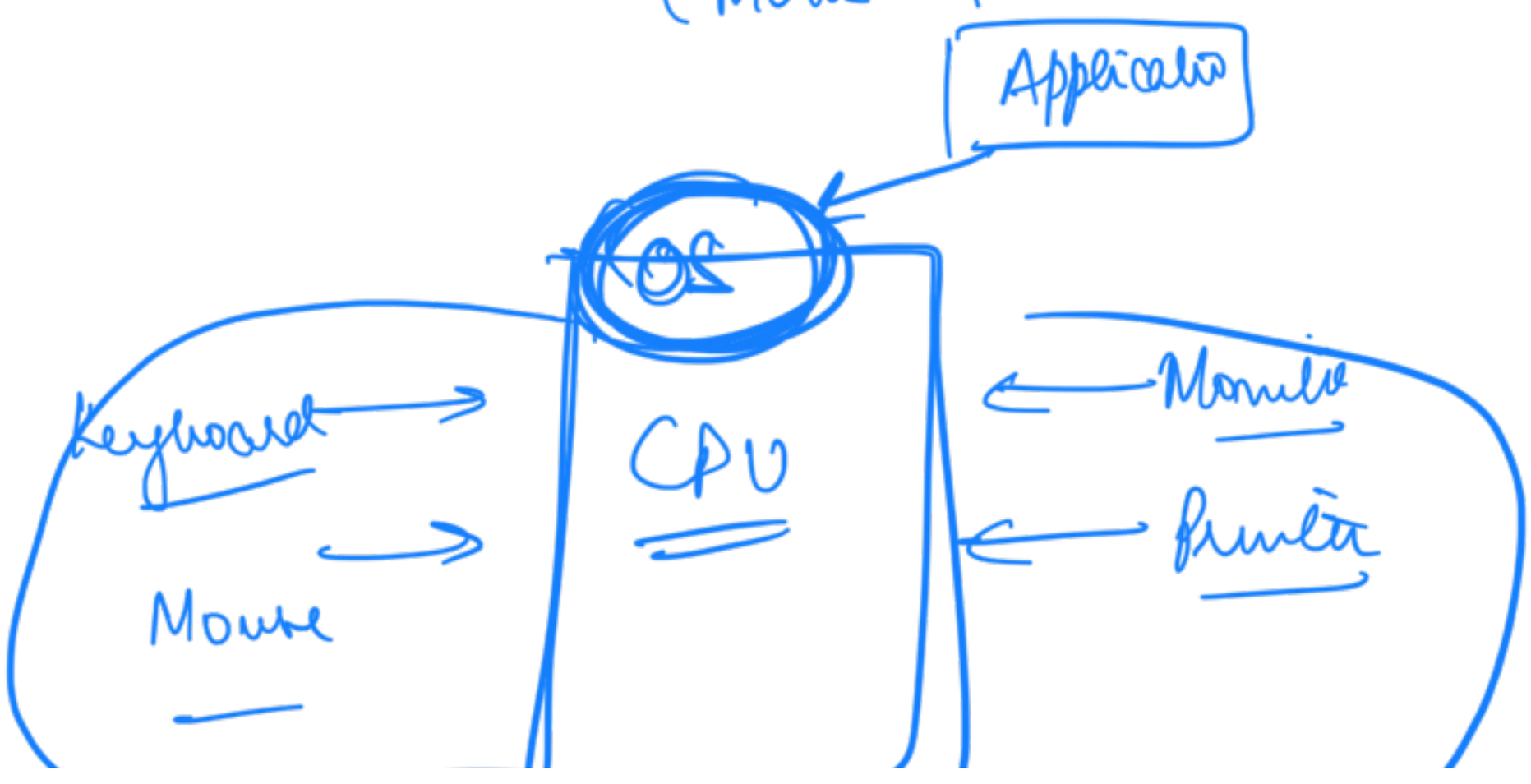
① Resource Manager ⇐

② API for App^n Developers =

incoming client req (network)

takes input from CL (Keyboard)

(Mouse)



Memory (RAM)

OS - Manager

Application

API

OS

CPU

Mou

Pri

RAM

Disk

Resource Manager
To ensure smooth
working of a com

OS → set of methods /API/ system calls
that can be made by Applications
to get work done from the resources

OS
Driver
Panasonic Printer

CPU
OS

Uniprogramming vs Multiprogramming System

→ Only one program can run at one time

↳ ATM Machine

↳ Embedded System

→ Multiprogramming (Unix)

→ More than one prog at a time

→ Multitasking (Windows)

Multiprog Sys v/s Multitasking Systems

Unix → Mac OS, Linux (Program)

Windows → Task

2 types of multiprog categorization

(1) Single v/s Multi User

v1
OS
v2
independent
independ

AWS
BigMachine

2 Preemptive v/s Non Preemptive Multiprog

Non Preemptive

→ Multiprogrammed

→ Only one program will be progressing at a time

→ You cannot switch to another program till the first prog finishes

Task 1

Task 2

T2

Task 1

T0

Task 2

Task 1

Task 2

t1

t2

t1

T0

Timeline

Mind | Brain

CPU

1 core can only
task at a

Quad Core Octa Core 16 core
32 core

Core 1

LP HP MP

[ ] [01] [01, 02] [01, 02, 03]

01 02 02 03 TS 01

Preemptive: where a task can be ~~stopped~~ paused in middle.

Boss HP 

Task

Report tomorrow

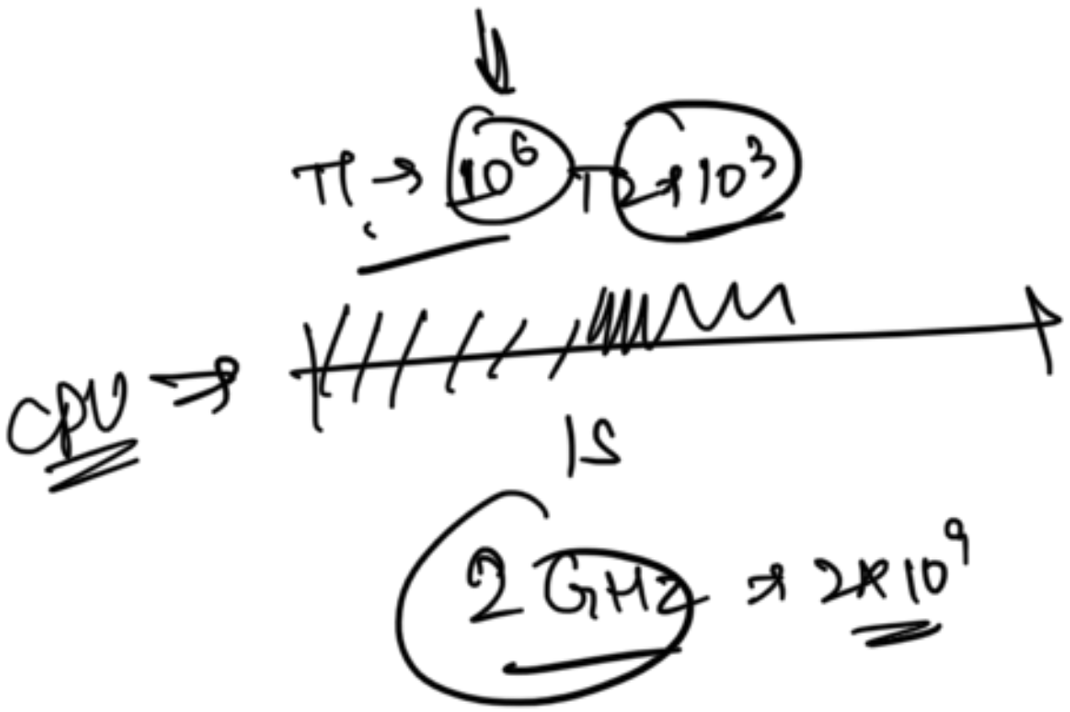
Clean Another room

Clean a room

Non-Preemptive

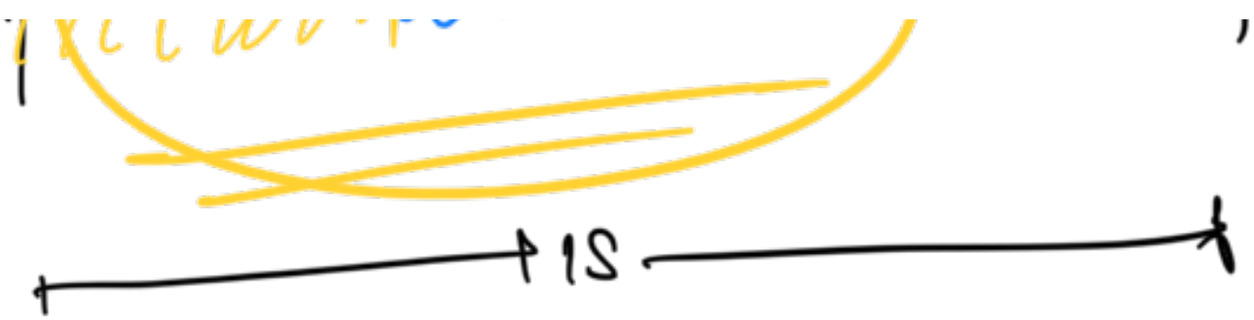
Pre emptive

Non Pre emptive



$\frac{10^6+10^3}{2*10^9}$

$\frac{10^6}{2*10^9}$



CPU [1 intelligent person doing] 4 core 2GHz

GPU [1000 not so intelligent workers] 1000 cores 800MHz

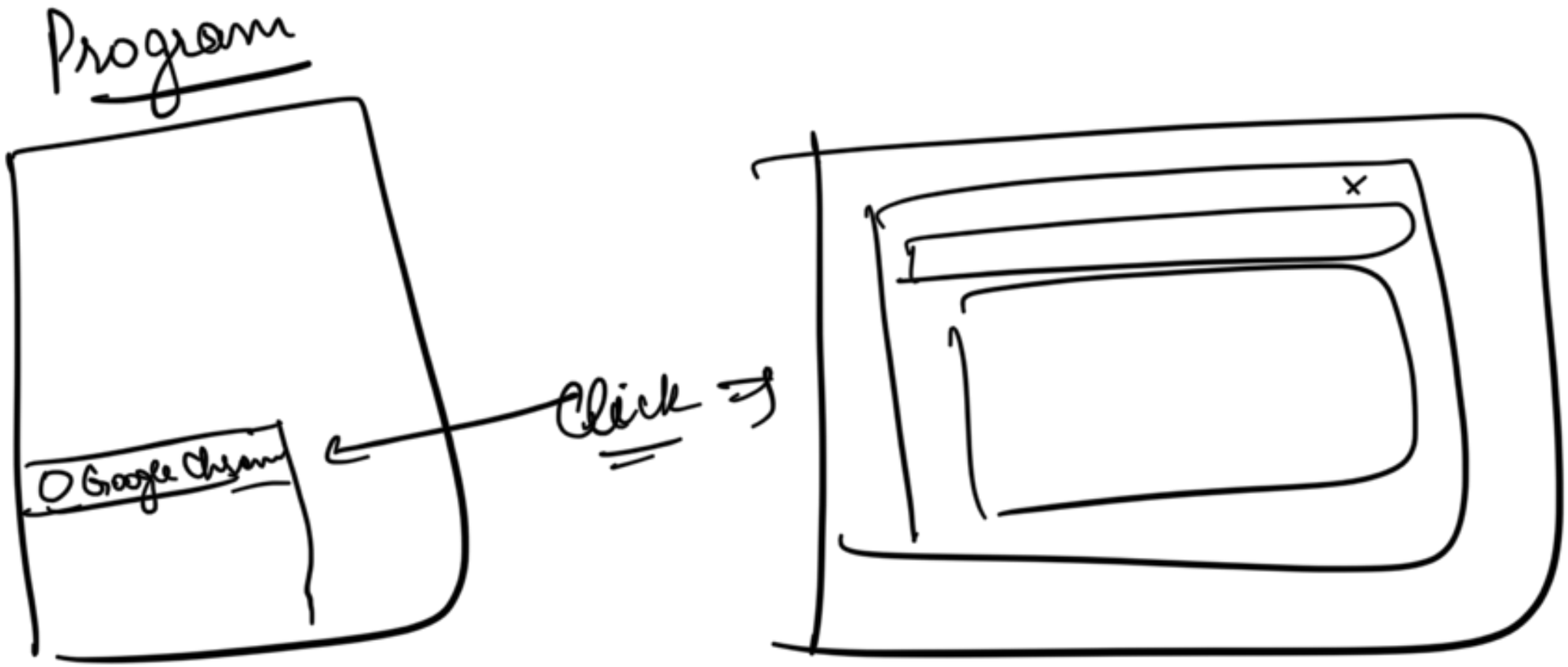
Process

Program ⇒ Set of instructions / Code
that we want to run

⇒ Something that can run

Process ⇒ Program that is running

Multiprog == Multitask == Multiprocess

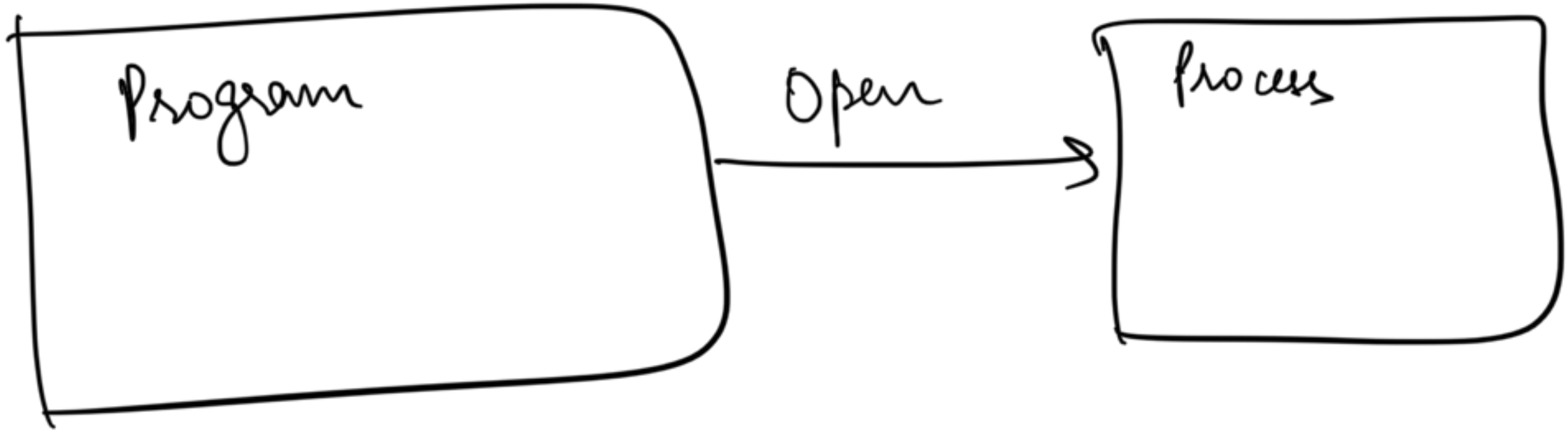


① Program is stored in the disk

② Process is stored in the RAM

(2) [illegible]

(3) Process runs on the CPU



Process Control Block (PCB)

→ Set of data info about every process

→ id

→ resources used (Cpu memory, pranter)

→ cpu related info (priority)

→ Memory limits

→ State

→ Last time executed

GC

B

Macbook

128

128

M1

256

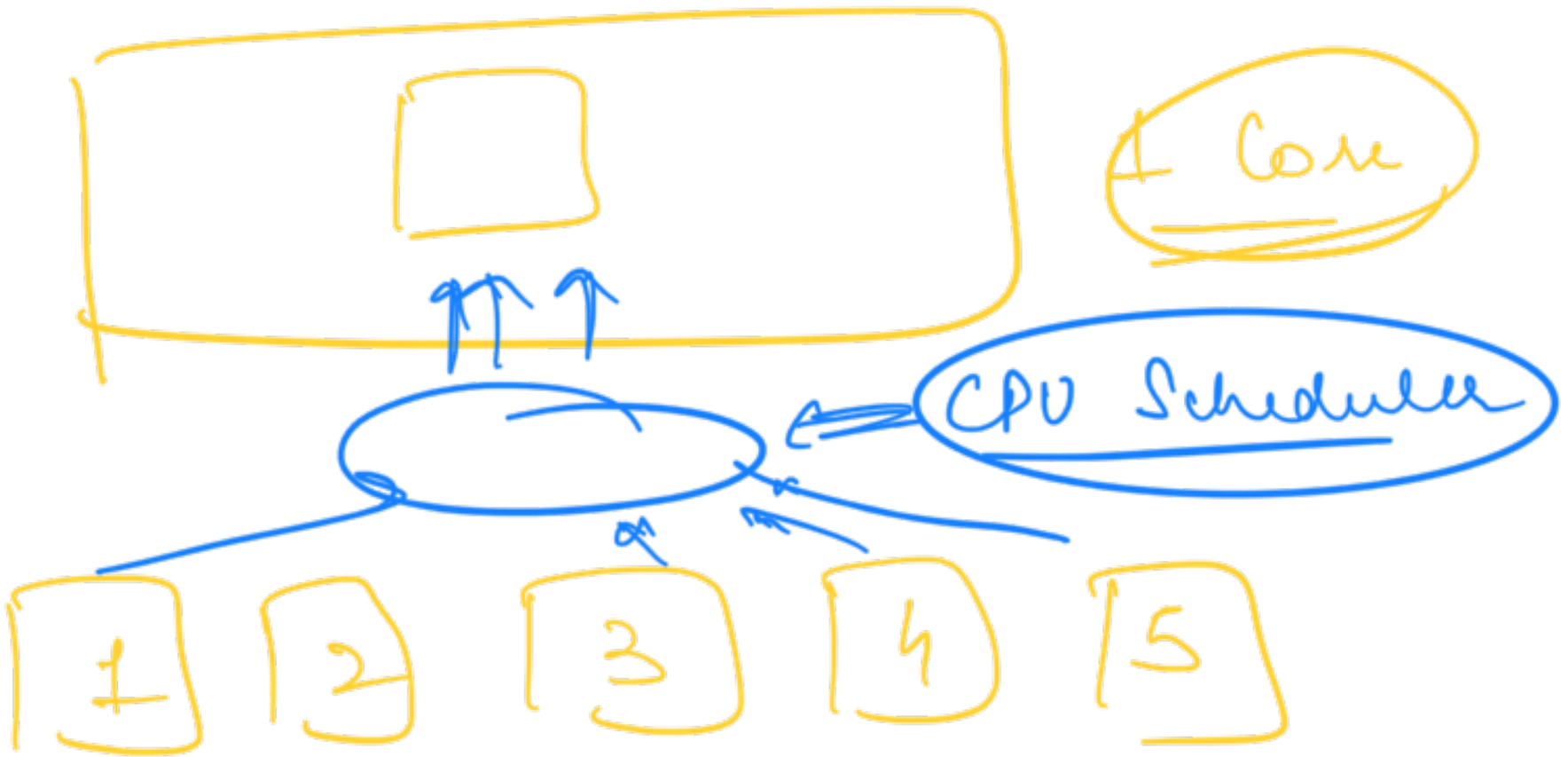
Buy

M2

I/O BP

CPO BP

Multiprocessing / Multiprogramming / Multitasking

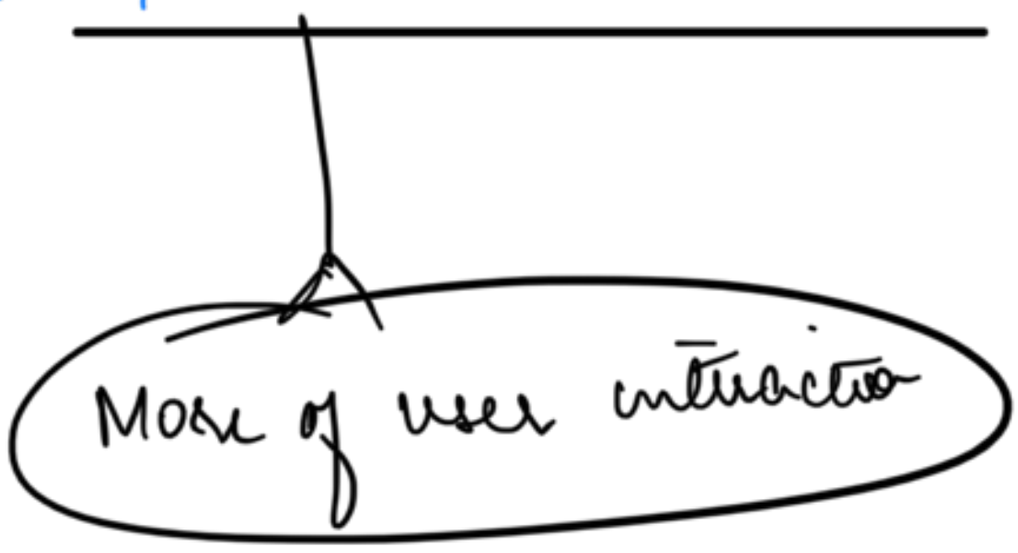


→ 2 types of processes based on what they do

Types of I

① CPU Bound Process

② I/O Bound Process



CPU Intensive

I/O Intensive

More of Calculation work

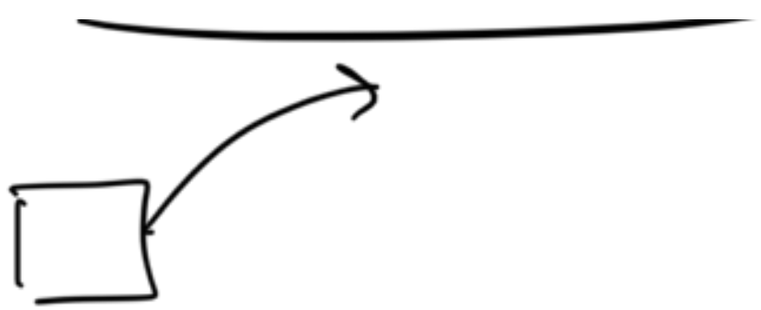
⇒ (Video Processing)

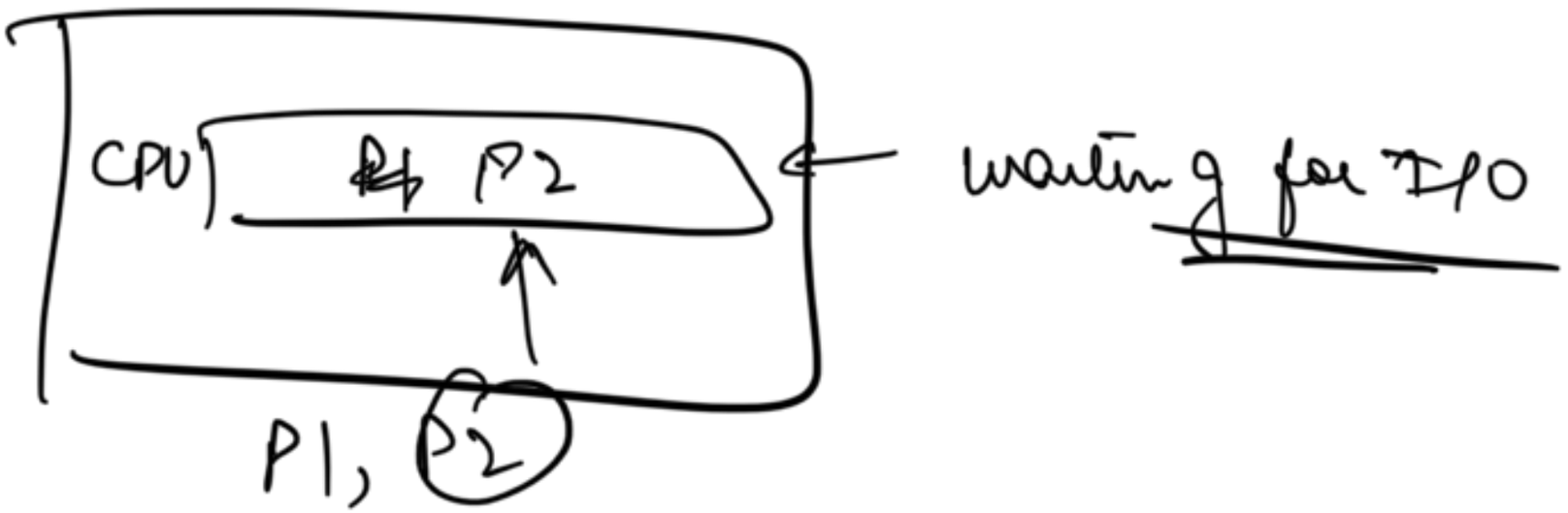
⇒ Most of app these days

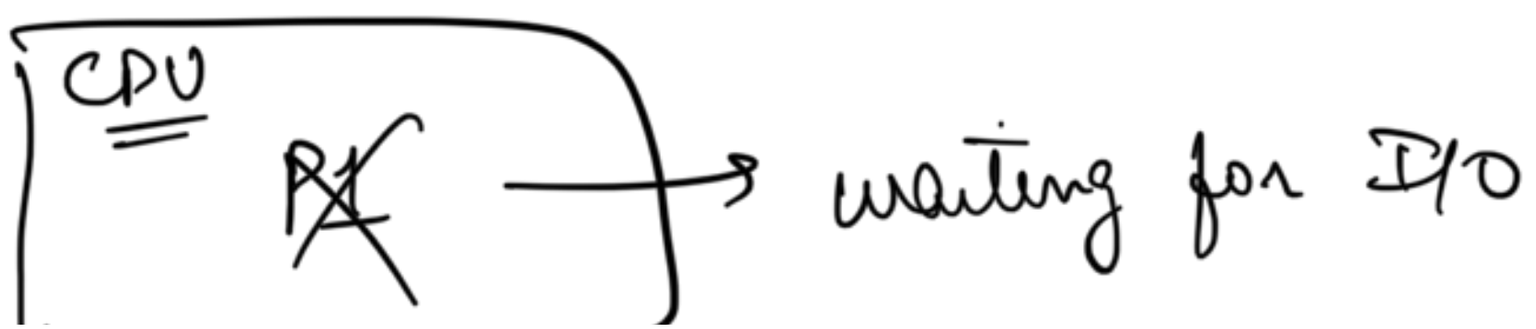
→ When I/O is happening does CPU have any job.

```
cin>> a;
```

→ When I/O is happening CPU is idle.




CPU
P2
waiting for I/O
P1, P2


CPU
P1
waiting for I/O

P2, P3

CPU

P2

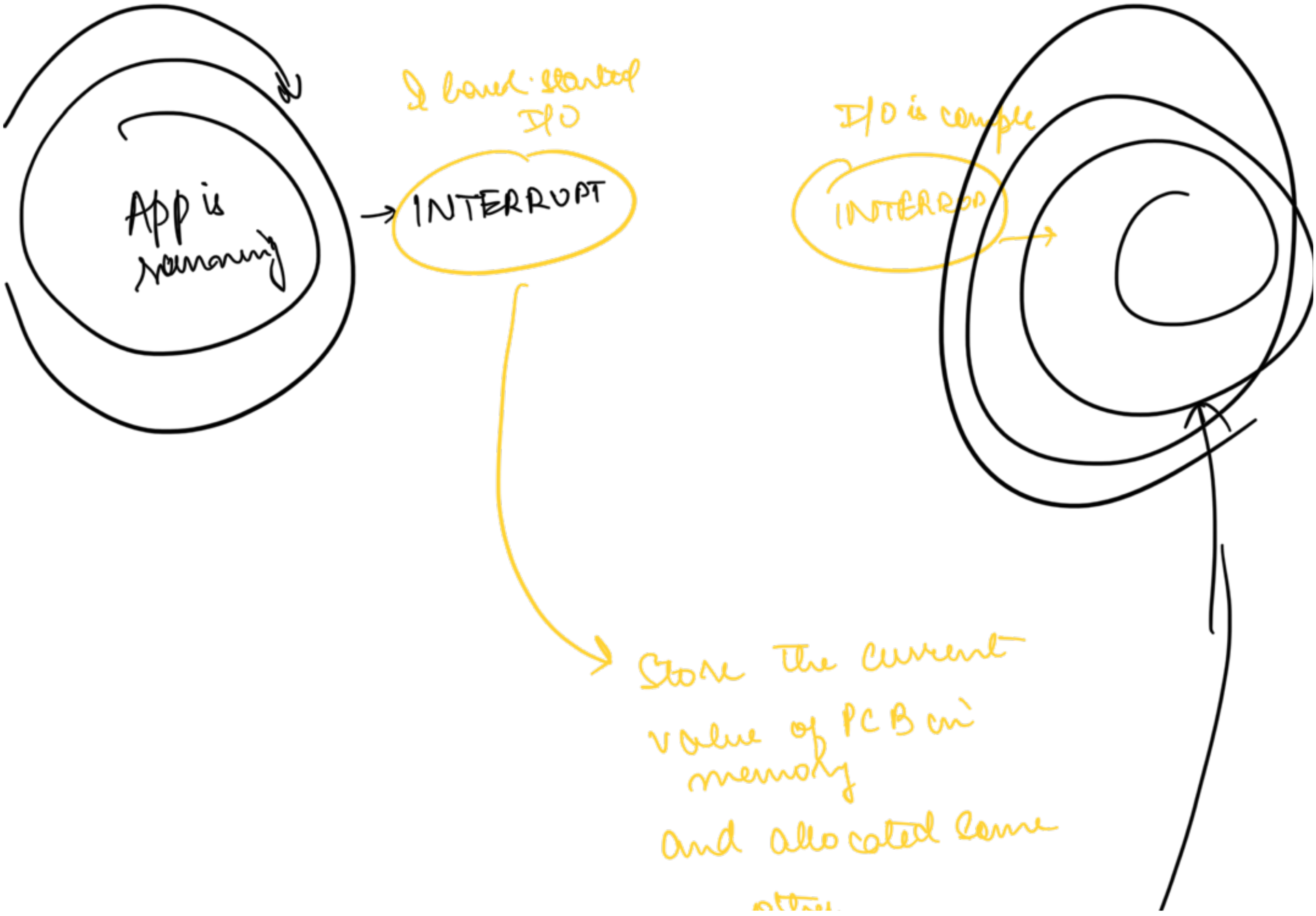
just take some inputs

I/O Bound

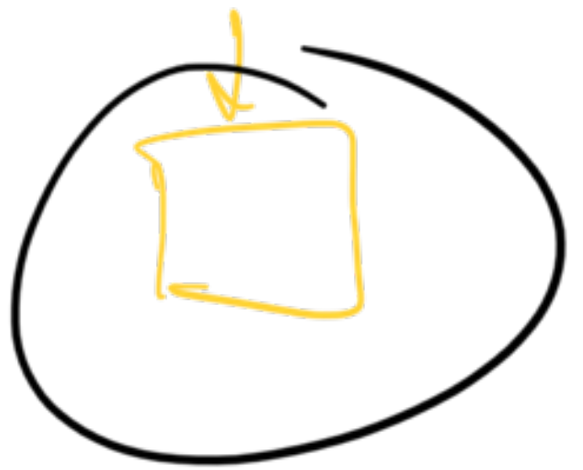
CPU Bound

How I/O happens

interrupt : Signal to the OS that the current program needs to pause to do something else (I/O)


App is running
I have started I/O
INTERRUPT
I/O is comple
INTERRUP
Store the current value of PCB in memory and allocated some

process to CPU



List<PCB>

Program

process

Ready Queue

Scheduler

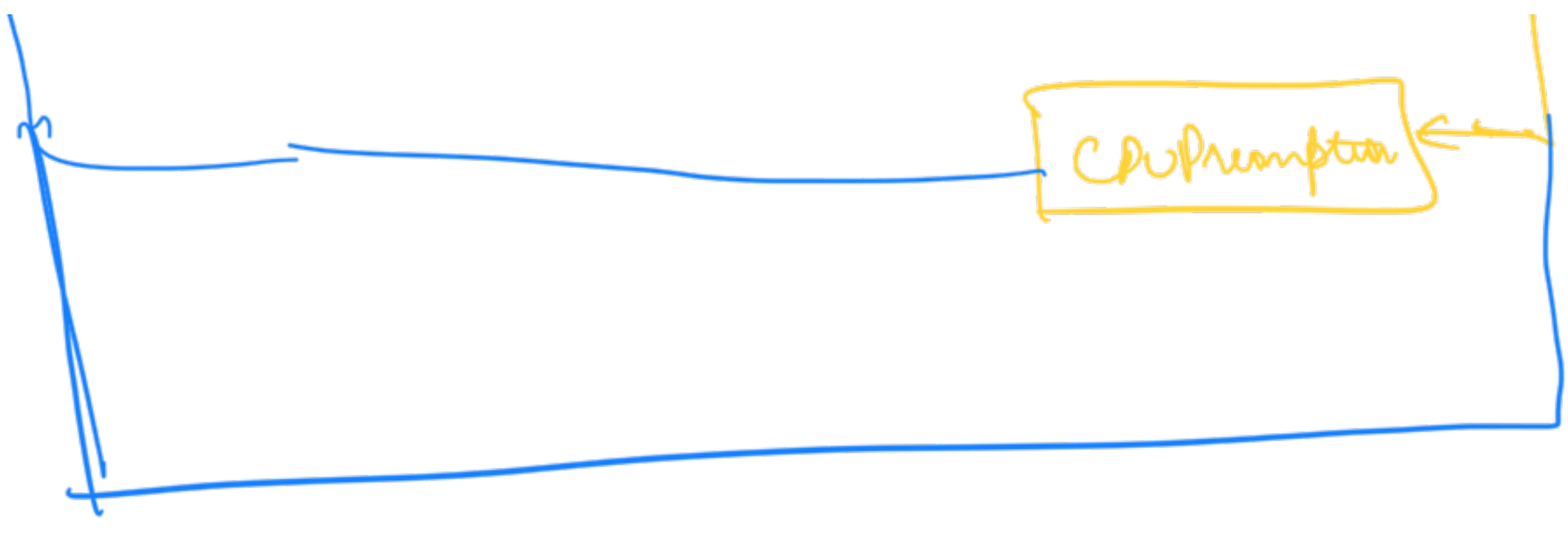
CPU

List<PCB>

I/O Queue

I/O

I/O Request



→ file your taxes

→ Do Scaler assg

→ finish your work

→ Watch Netflix

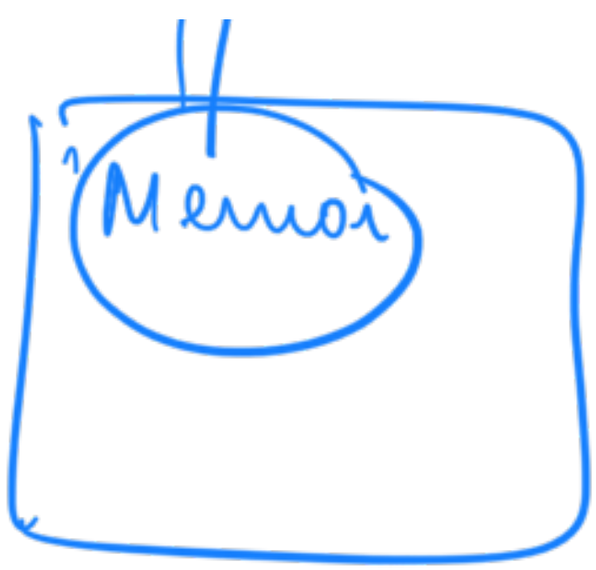

Doing some work
Waiting Mouse
Resumed the work


CPU
P1

P1

P1

Context Switch : Moving process out of a register to memory as well as moving it back onto register takes time



CPU Scheduling Algos

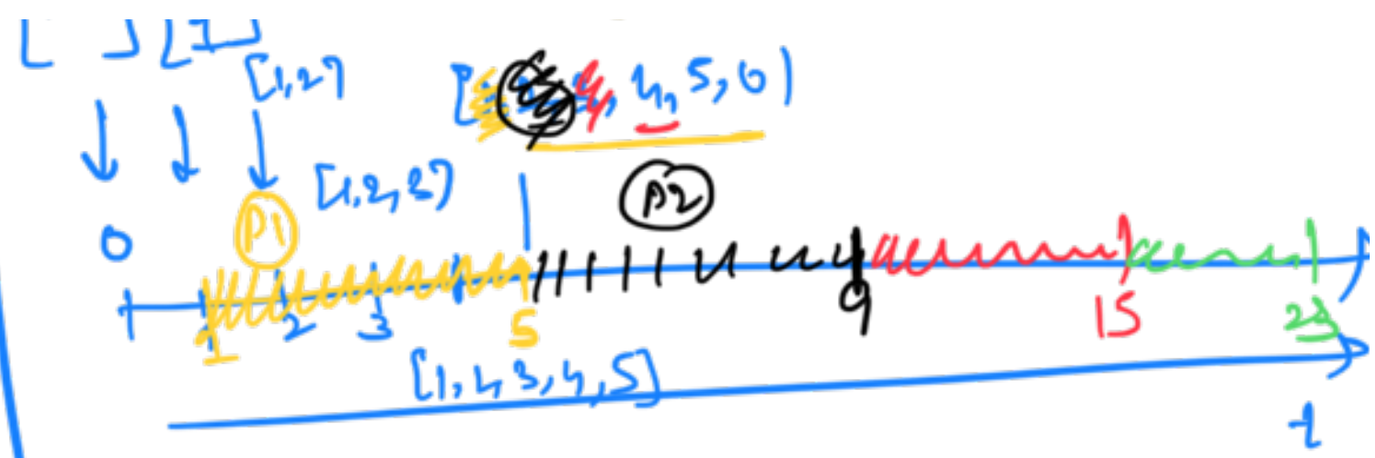
FCFS

→ Non Preemptive

PID	Arrival Time	Time to Complete

Ready Queue

1	1	4	
2	2	5	
3	3	6	
4	4	8	
5	4	2	
	6	5	4



SRTF (Shortest Remaining Time First)

→ Preemptive

→ Decides which process to run whenever:

① A previous process completes ←

② A new process comes = 1/2

(OR) (2) A new [illegible] 1.5

→ It will choose the process which has the least remaining time

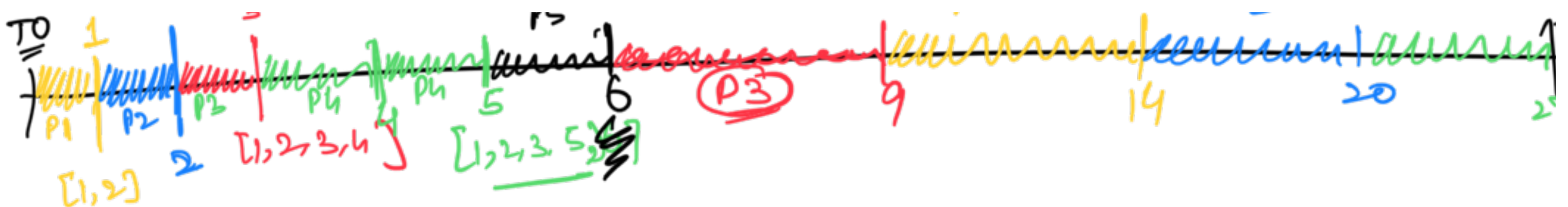
PID	Arrival Time	Burst Time	Remaining Time
1	0.9	8	7 5
2	1	6	4 1 0
3	2	4	2 1 0
4	3	2	1 0
5	4	6	6
6	5	1	0

✓ Burst and arrival time are decimal

T = 2
T = 2.5

[1] [1,2,3] [1,2,3,4,5] [1,2,3,5]

P2 P5 P1



How to determine Burst Time

① Av of previous running time

6 8 5 → 6

② Size of program

To implement SRTF what info do we need :

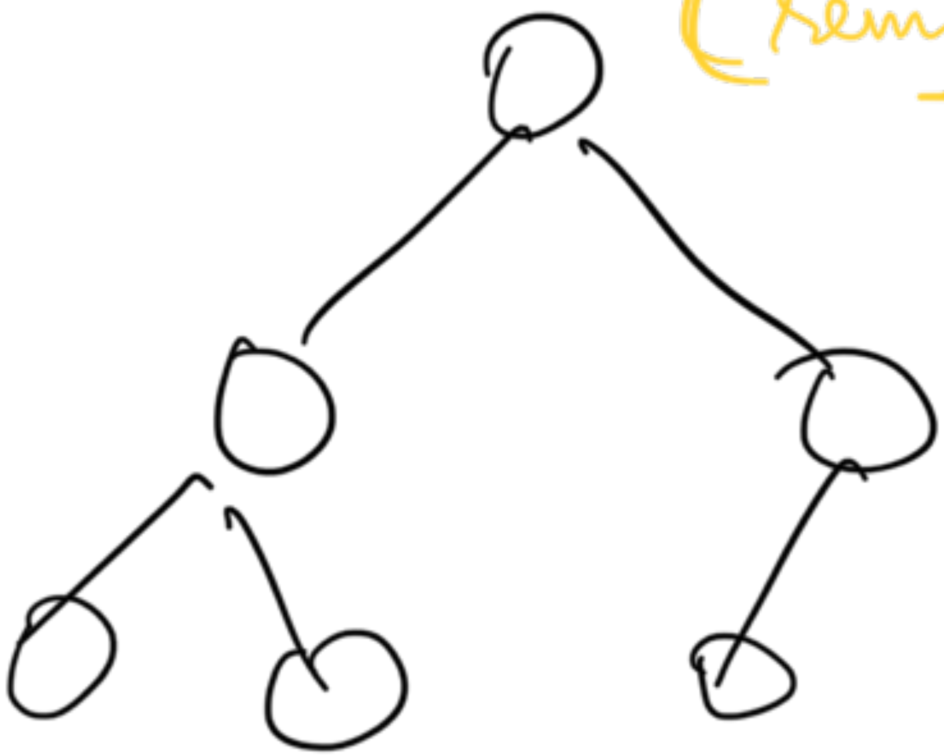
(arrival of new process

at every step (OR
comp of a process)

needs to know the process with shortest remaining time

→ Min^n in O(1) ≈ O(logN)   Priority Queue (Min Heap)

(remaining time, p-id)



⇓ … null;

Current process = null

heap = {}

```
class OS {
    processEntered(pid, burstTime) {
        → heap.put((burstTime, pId))
        → Scheduler()
    }
}

Scheduler() {
    nextProcess = heap.extractMin()
    prevProcess = currentProcess
    currentProcess = nextProcess
    if prevProcess.remainingTime != 0 {
        heap.put((pp.id, pp.rt))
    }
}
```

}

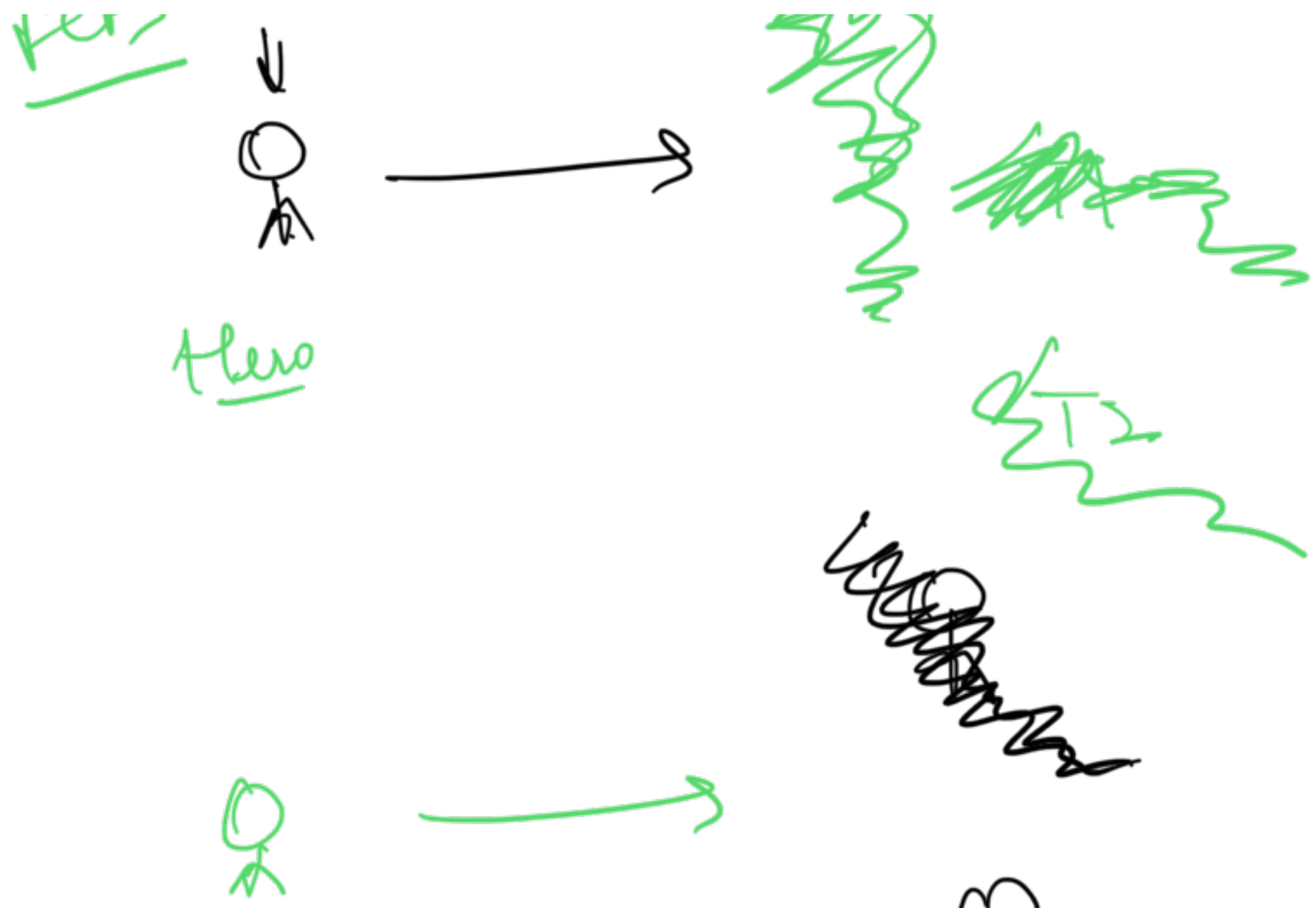
FCFS, RR, SJF

SJF (Shortest Job first)

⇒ Non Preemptive Algo

Whenever a new process comp, out of the process currently in ready Queue

⇒ One which has shortest burst time



- Throughput
  - RR → (LB)