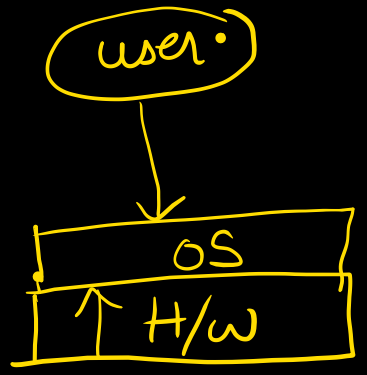
OS : operating Systems:

Syllabus:

- → Process management
- → Process synchronization
- → Deadlocks
- → memory management
- → File system, I/O, & protection
- → Threads and System calls.

Simpler than CN.
less time than CN.
easy than CN.

Why OS is required:



directly interaction with H/W is difficult. more programming is required by user.

→ OS is an intuface b/w user and Hardware.

→ OS manages all the resources of a Computer.

CPU mem files devices.

what is primary goal of OS

Convinience or efficiency.

→ Depends on type of Computer:

Personal:
1) Convienience
2) effi

→ In this course we study PC.

mainframe or Super Computers: → not much in use now.
1) effi
2) Convienience.

Types of OS:

1. Batch OS.

many at once

Batch of Jobs → all submitted at once.

↓

Programs

After a long time user will collect the O/P.

These were used in 1980's.

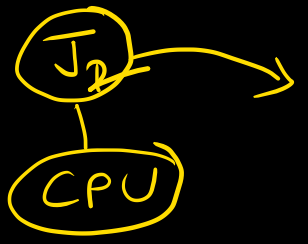
multiprogramming & multitasking: → Same in many textbooks.

Batch. | J1 | J2 | J3 | J4 → only one job will finish at a time.

multipro | J1 | J2 | J1 | J2 | J3 | J1

all the jobs share the CPU. ✓

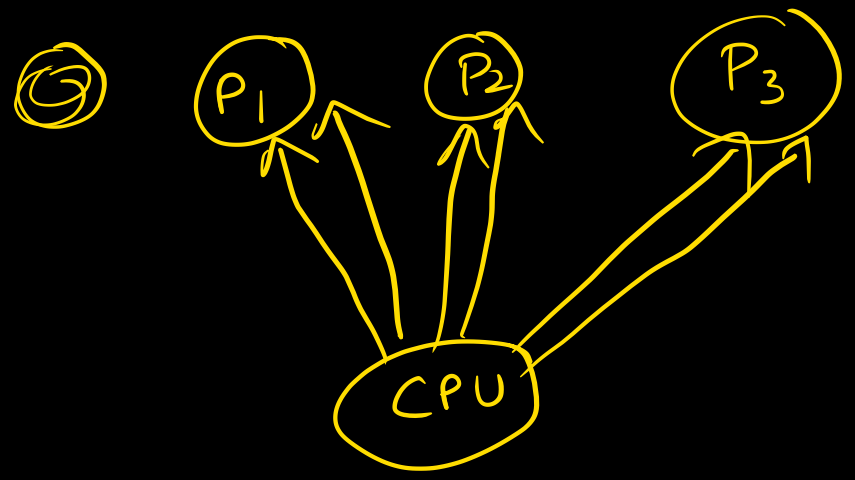
So every job will get a chance frequently.



J1 J2 J3 J1 J2 J3.

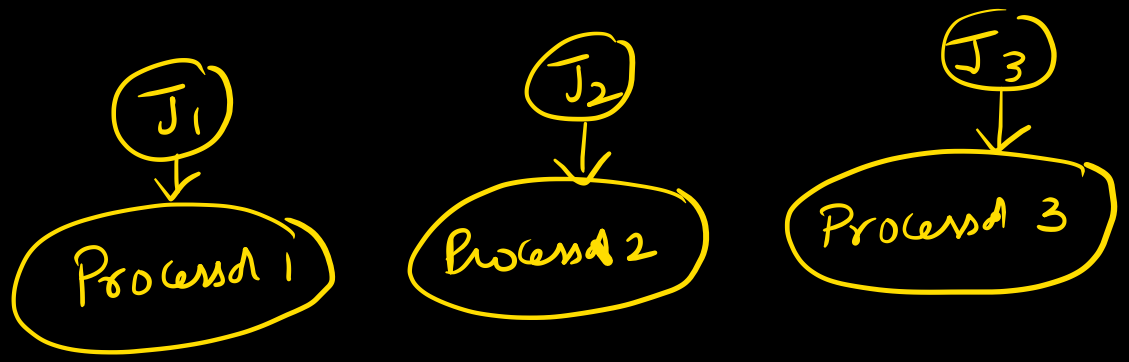
CPU Interactivity in mode

multi. prog:



CPU will switch b/w the processes.

multiprocessing: There will be many processors.



For multiprocessing
→ at least
two processors
are required

So many Jobs can execute in parallel.

Ex: Dual core – 2 processors
quad core – 4 processors
octa core – 8 processors
:
16
32
64
:

we can do multiprocessing

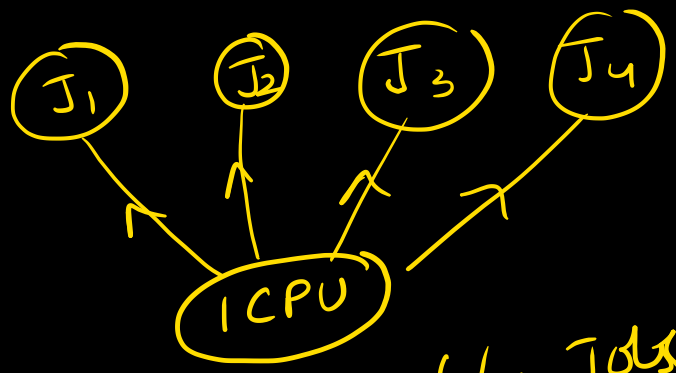
Real time OS:

Jobs will have dealines. ✓

Job & process → same

Job	J_1	J_2	J_3	J_4
Deadline	10:00	10:10	10:20	10:30

J_1 → should be complete by 10:00 otherwise it is not useful.



switches b/w Jobs

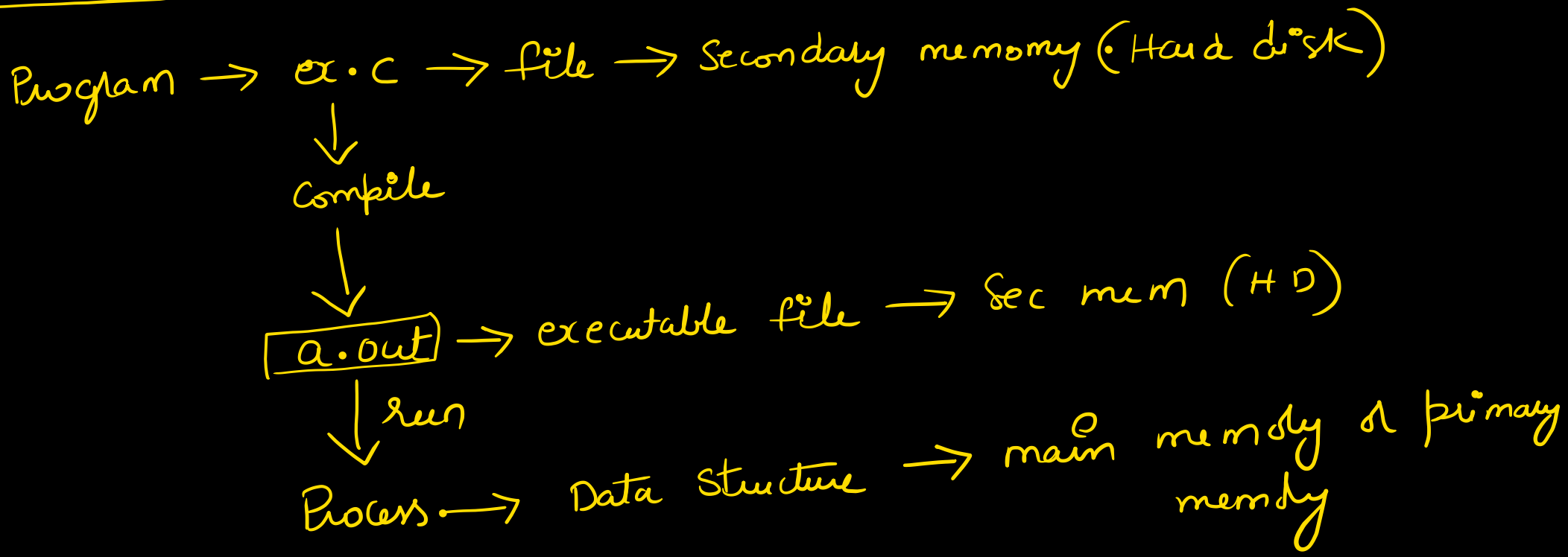
→ multiprogramming

↓

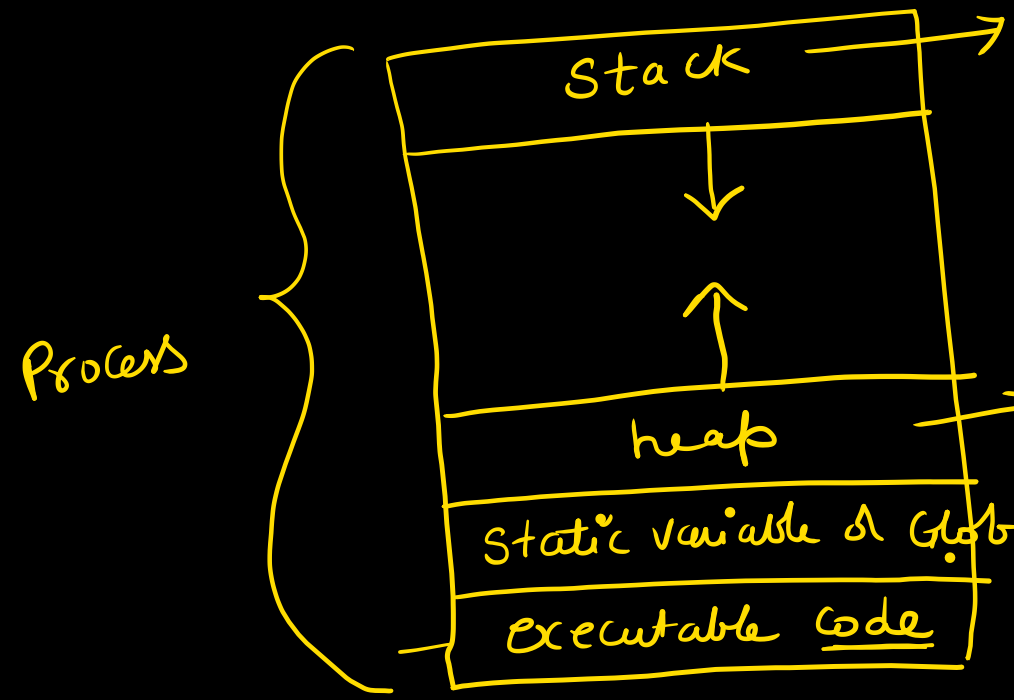
multitasking.

multiprocessing.

what is a process:



Data structure of a process:



→ function calling and local variable

→ dynamic memory allocation

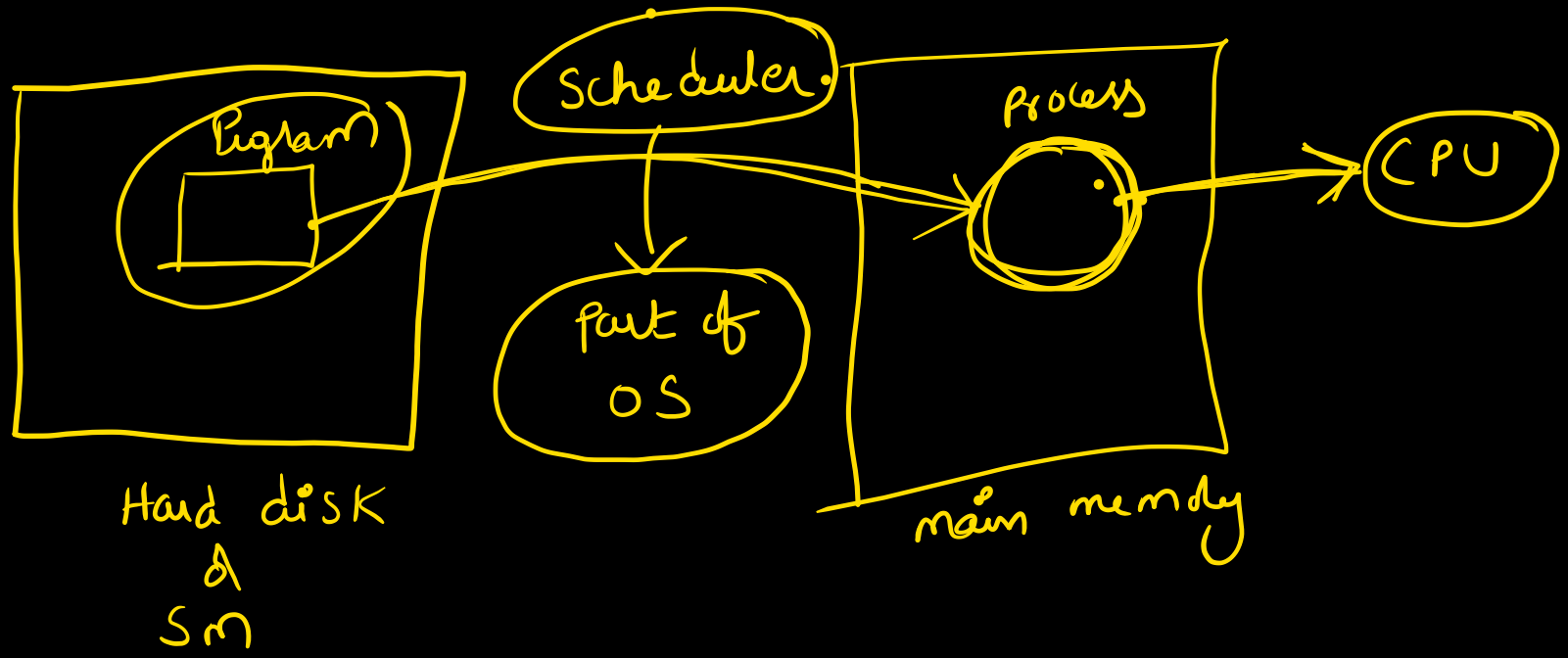
→ we will see in 'C' programming.

main memory & RAM & primary memory → same.

Program → body

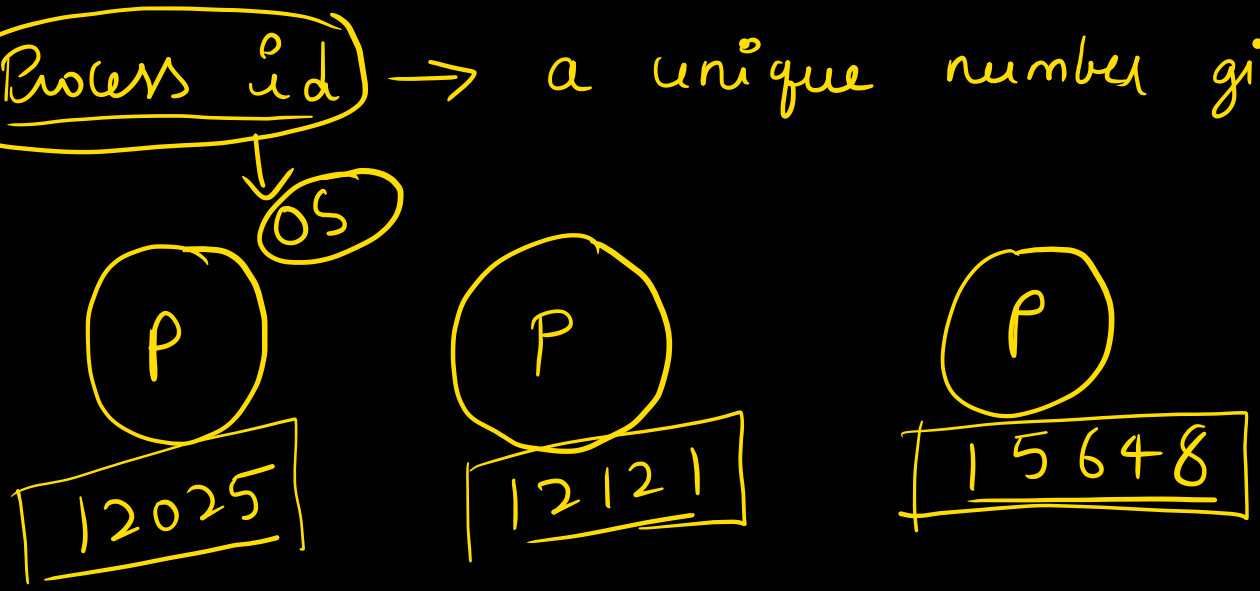
Process → Soul → life

without process we cannot execute a program.



Attributes of a process : → properties:

1) Process id → a unique number given to a process

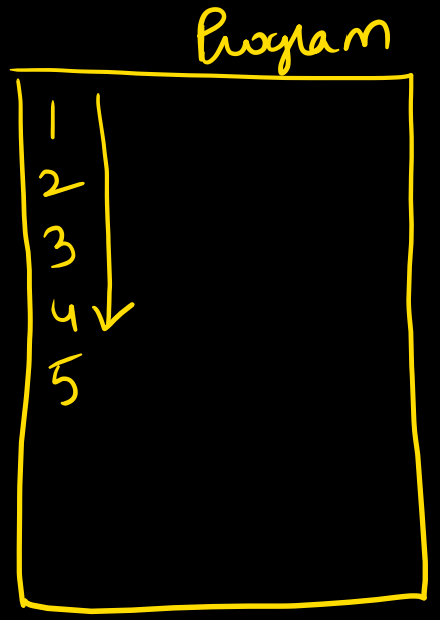


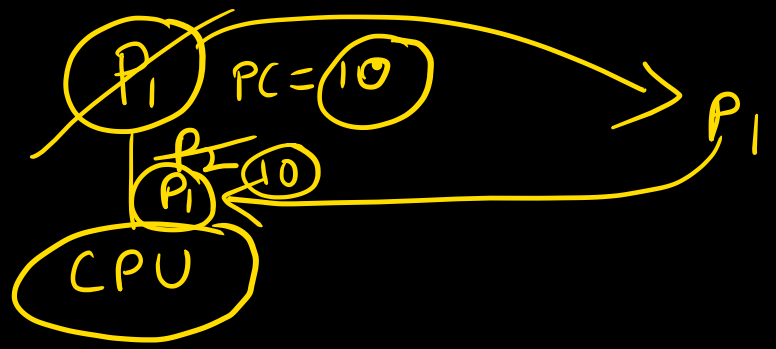
Similar to host number

↓

networking.

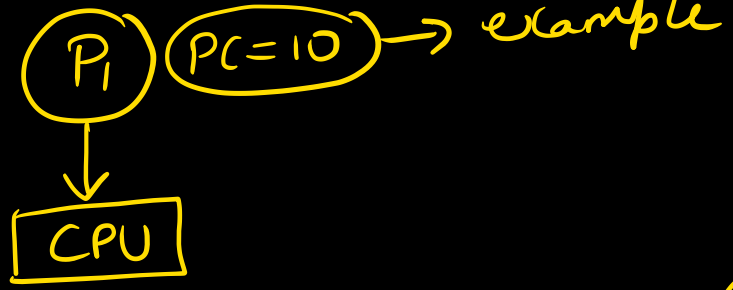
2) Program counter :





PC = 5 ⇒ The next instruction to be executed.

Let P_1 is executing and current PC = 10 → next instruction to be executed.



At this point P_1 got preempted (removed from CPU). After sometime again P_1 started. Here PC will help in resuming from where it left off. i.e instuction 10.

like a bookmark.

3) Process state:

Every process will have some state in its lifetime.

like:

etc.

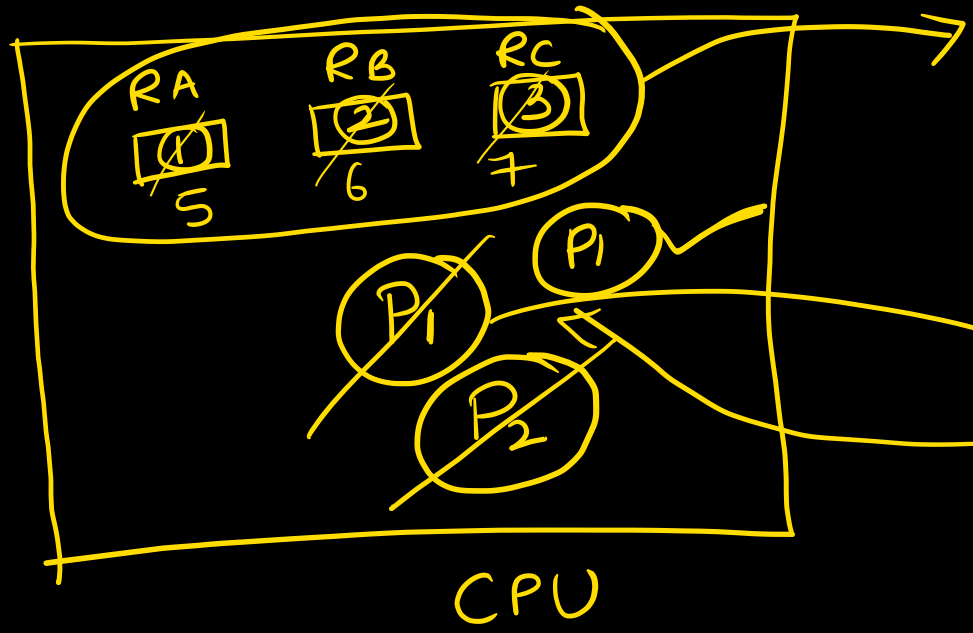
latu.

4) Priority:

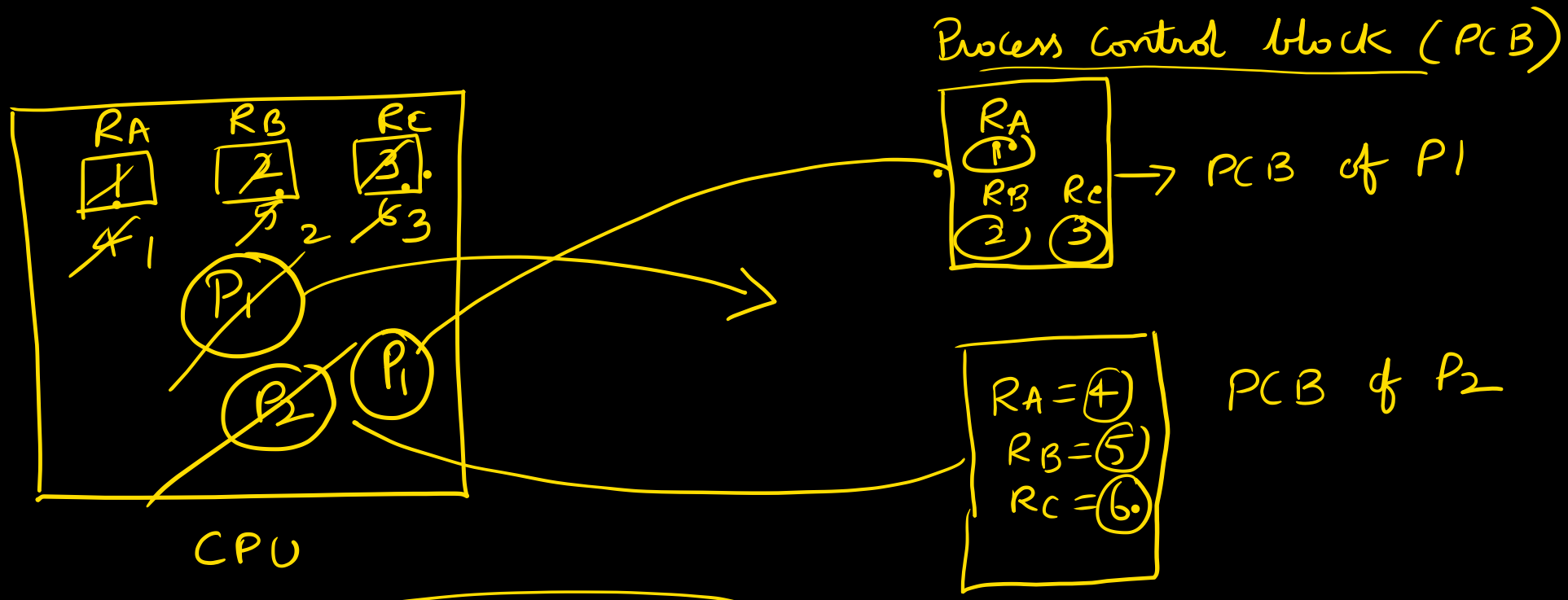
Generally process will have a priority.

But its not compulsary.

5) General purpose registers:

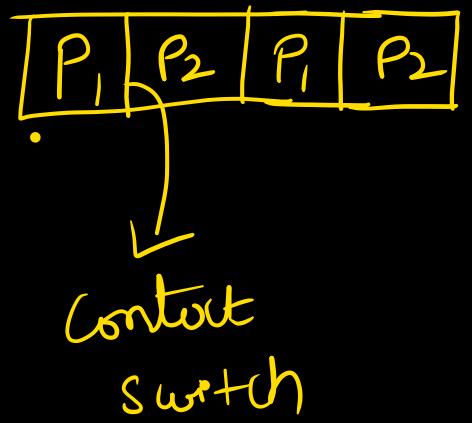


Register values have to be saved so that when a process comes back, those values will be same.



Context switching.

Storing the Register values in PCB and getting new Register value from PCB.


P1
P2
P1
P2
Context
switch

6) List of open files:

Every process opens some files and this information must be maintained.

Ex: (P1) → opened :⇒ F1, F2, F3

(P2) → " ⇒ F6, F8

7) List of open devices:

Every process may open some devices.

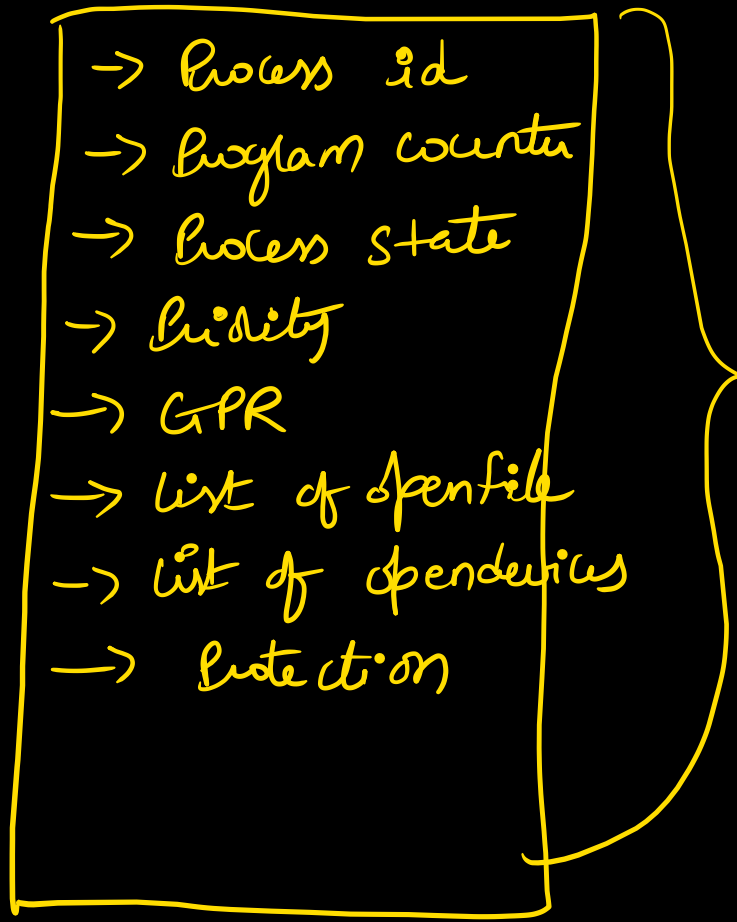
This info must be maintained.

8) Protection:

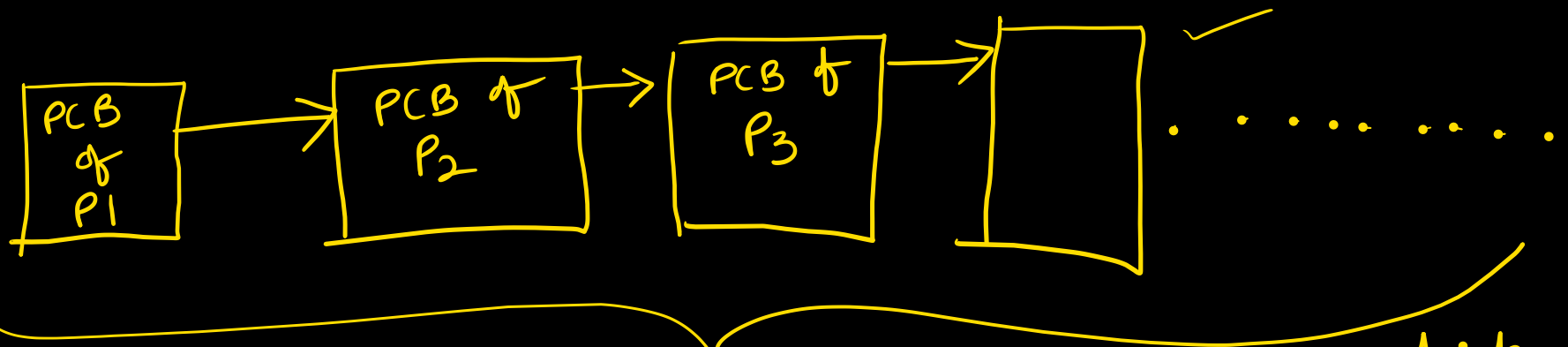
This is required for security → (later)

Every process will be manage by storing all information in Process Control block (PCB). By looking at PCB, we will know everything about a process.
Every process will have a PCB.

PCB



OS will maintain a linked list of PCBs of every process.



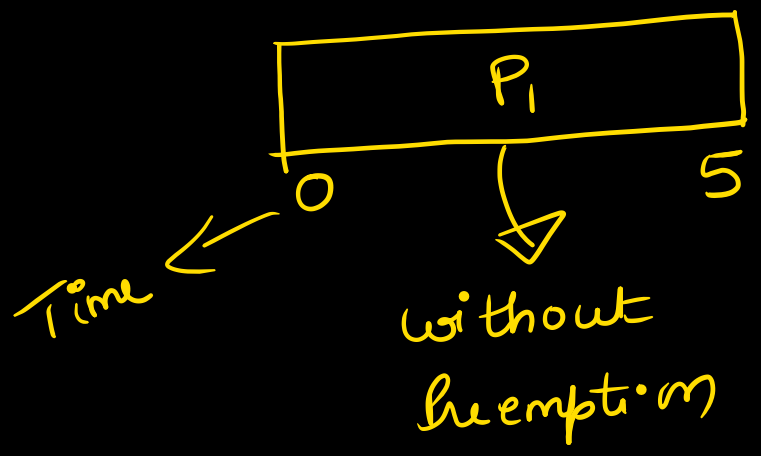
OS will look into PCB and decide which process to give CPU.

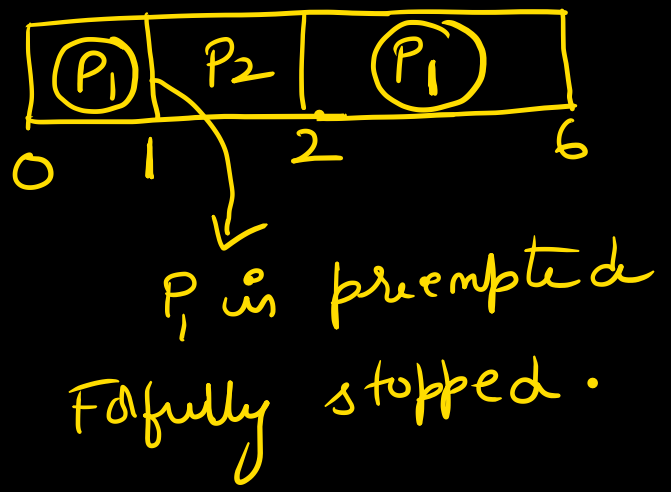
PCB is also called Context. It will be helpful in context switching

PCB is ~~maintained~~ maintained by OS:

Preemption: Forefully taking off CPU is preemption.

Let P_1 → need 5 units





Very important topic

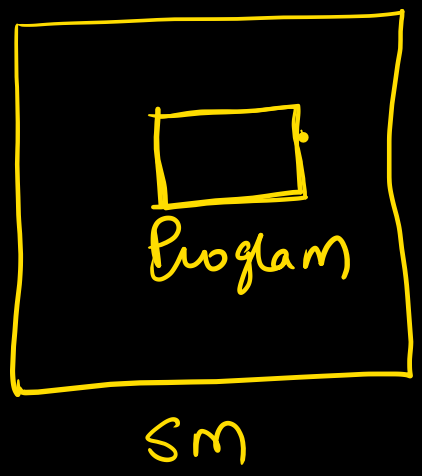
Pay attention

States of a process:

Break - 5 min

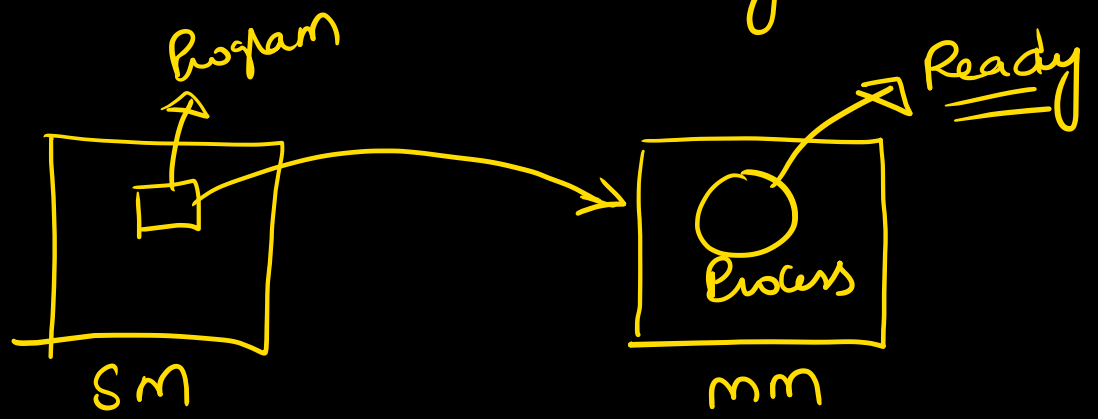
States of a process:

1) new state → Process is about to be created → Program is still in secondary memory.



Technically Process is not yet Created.
This is in little bit ambiguous.
Because, there is no process yet but we are saying process is in new state.

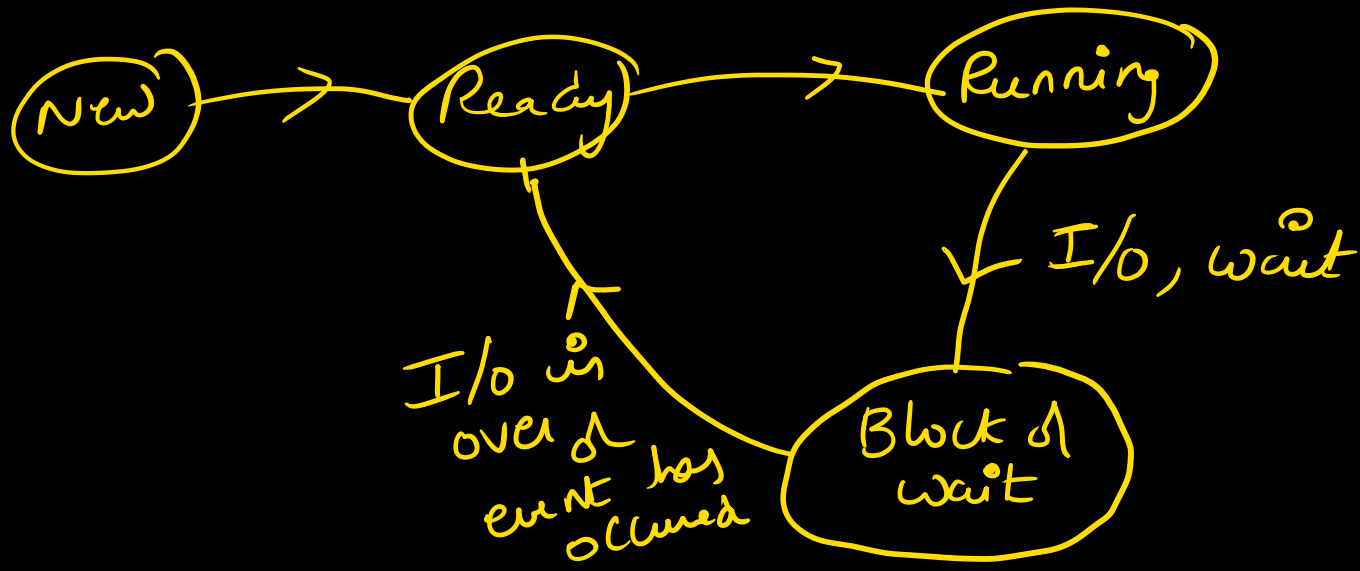
2) Ready State: The process is created and it is in main memory and ready to be executed.



There can be many processes in new & ready state.

3) Running: If a process is executing in the CPU, then it is in Running state. If we have 1 CPU, then only a max of one process can be in Running state.

4) Block & wait: A process may need I/O & gets blocked because of some events, then it we go to block & wait state.

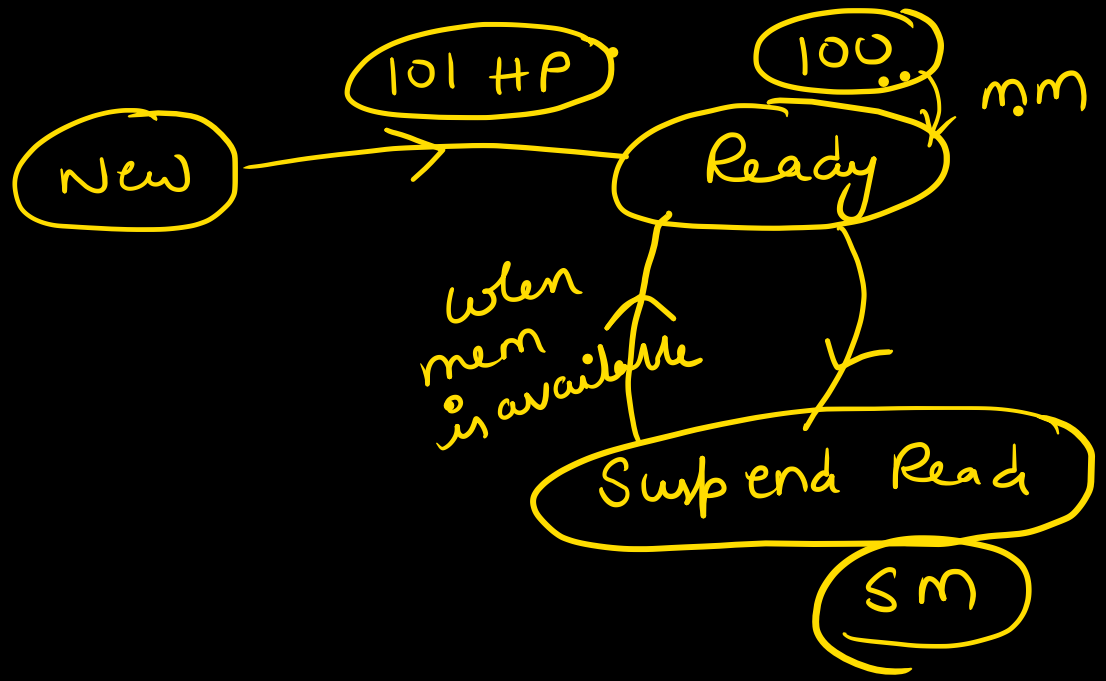


This is not complete

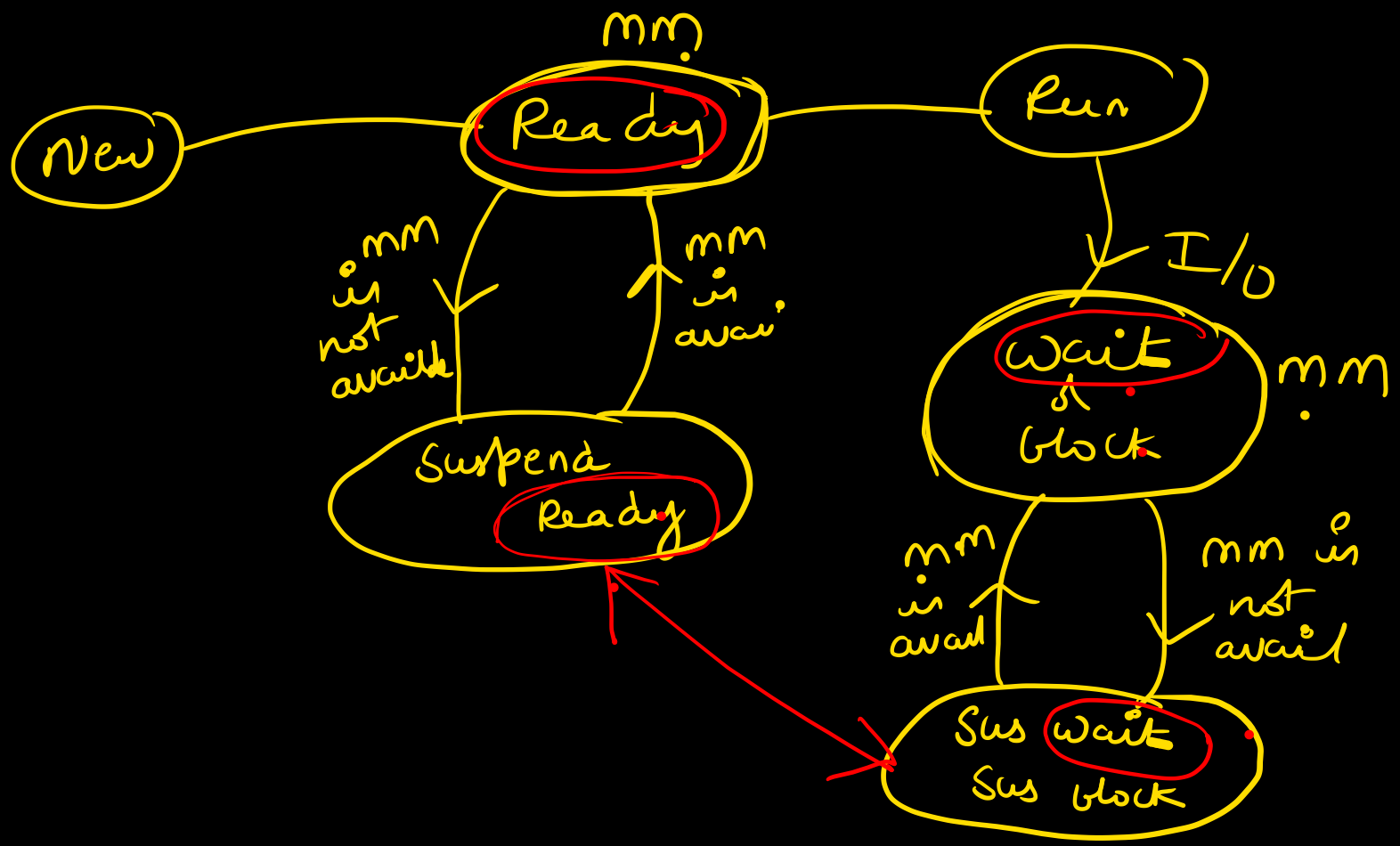
Pls wait till I complete.

5) Termination & Completion: once the process finishes, it is killed and even Context & PCB is deleted

6) Suspend Ready: If there is no space in main memory then one of the processes has to be moved to secondary memory to create more space. Then process will move to Suspend Ready from Ready.



7) Suspend wait & Suspend block:



not yet Complete

Tomorrow complete