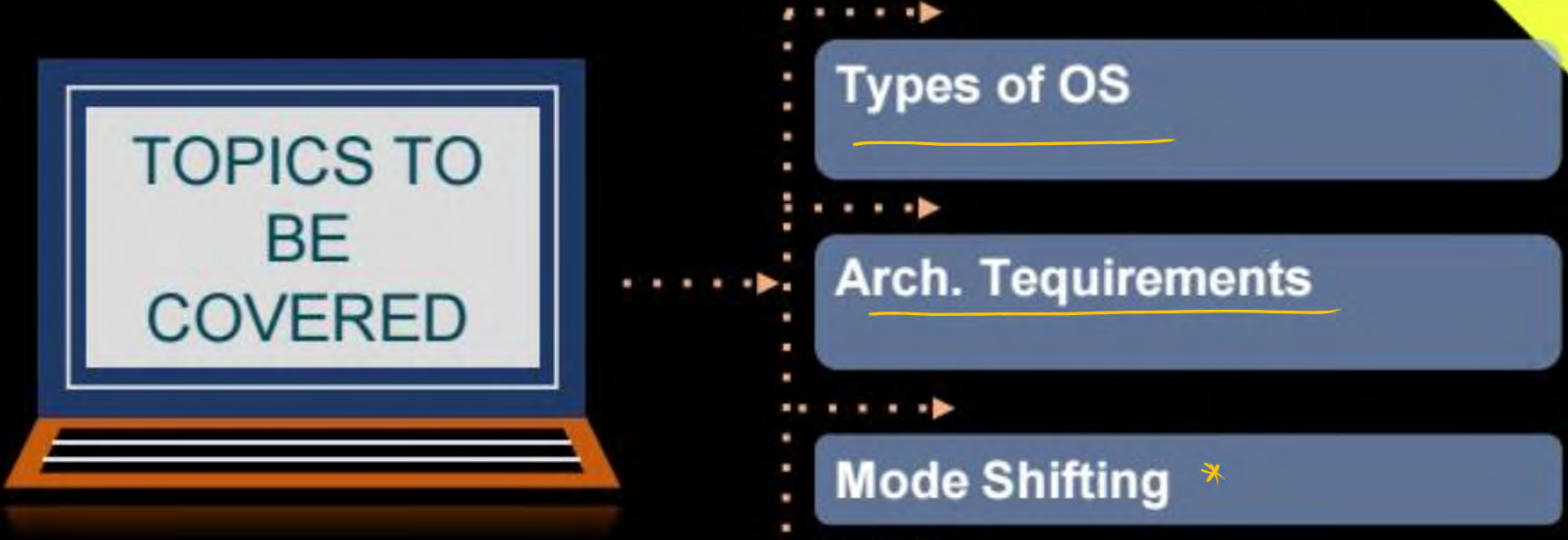
CS & IT ENGINEERING

OPERATING SYSTEM

Intro and Background

Lecture No. 3

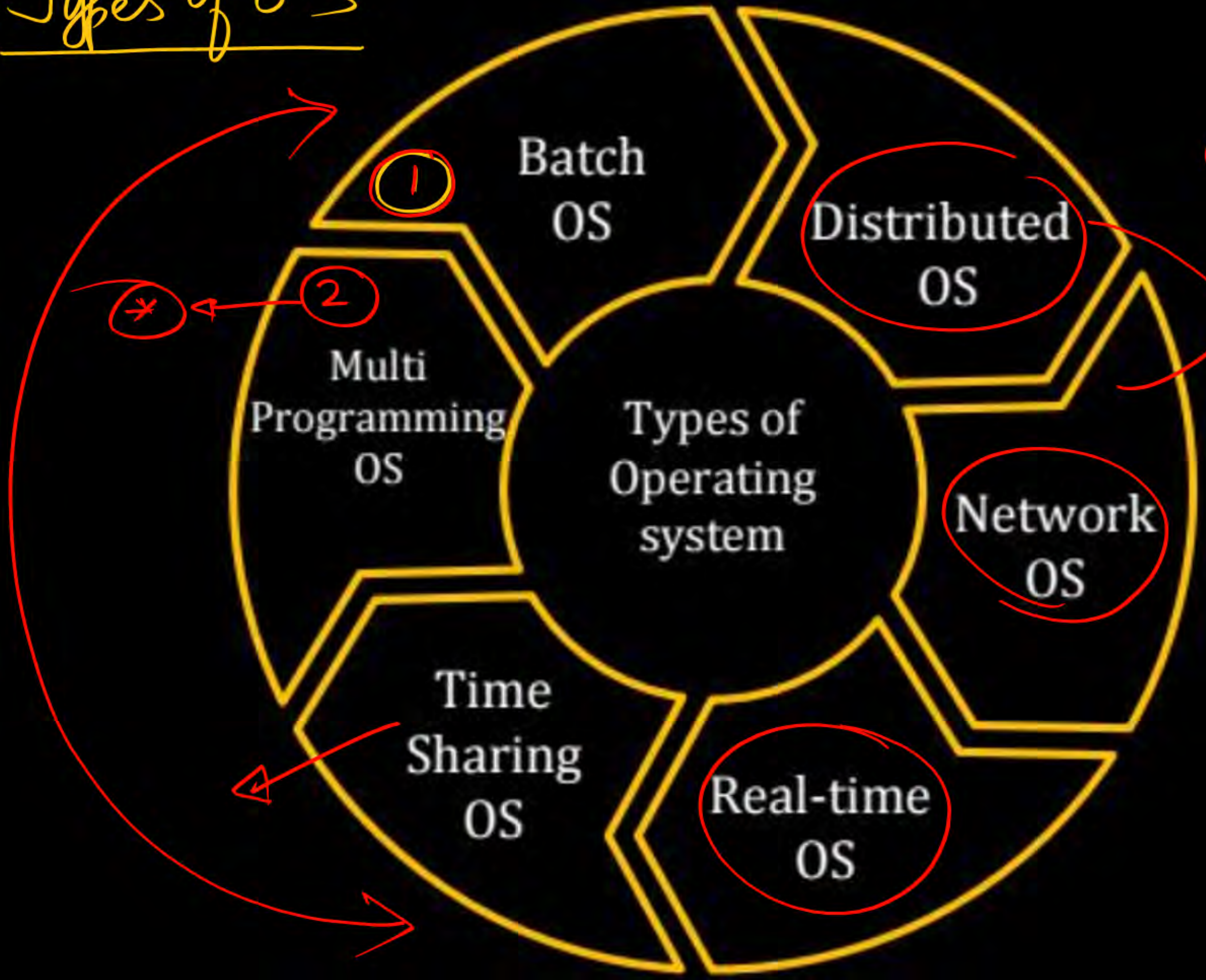
By- Dr. Khaleel Khan Sir



TOPICS TO BE COVERED

- Types of OS
- Arch. Tequirements
- Mode Shifting *

Types of O.S



Types of Operating system

Batch OS

Multi Programming OS

Time Sharing OS

Real-time OS

Network OS

Distributed OS

Basic OS

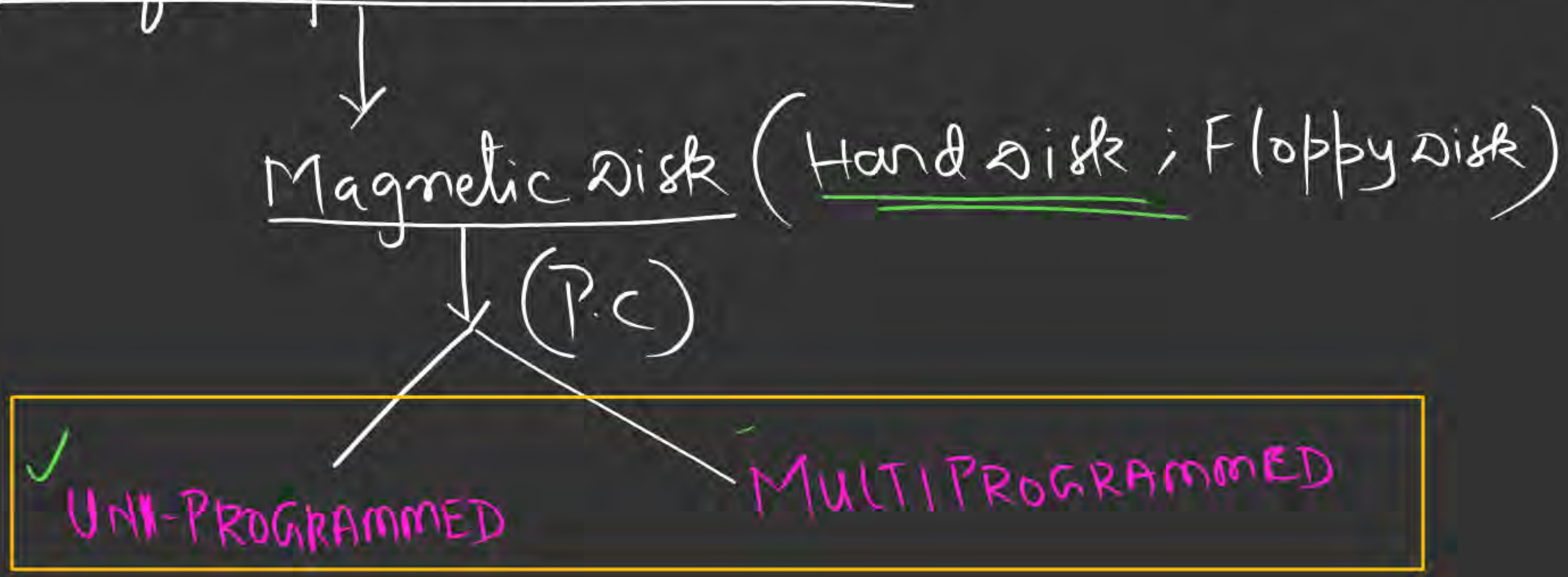
Multi-processor

Advanced OS

1st Gen of Computers (1930-40's) : No O.S

2nd Gen " " (1940-50's) : Magnetic Tapes : → No OS

3rd Gen of Computers (1950's-60's)



Magnetic Disk (Hard Disk ; Floppy Disk)

(P.C)

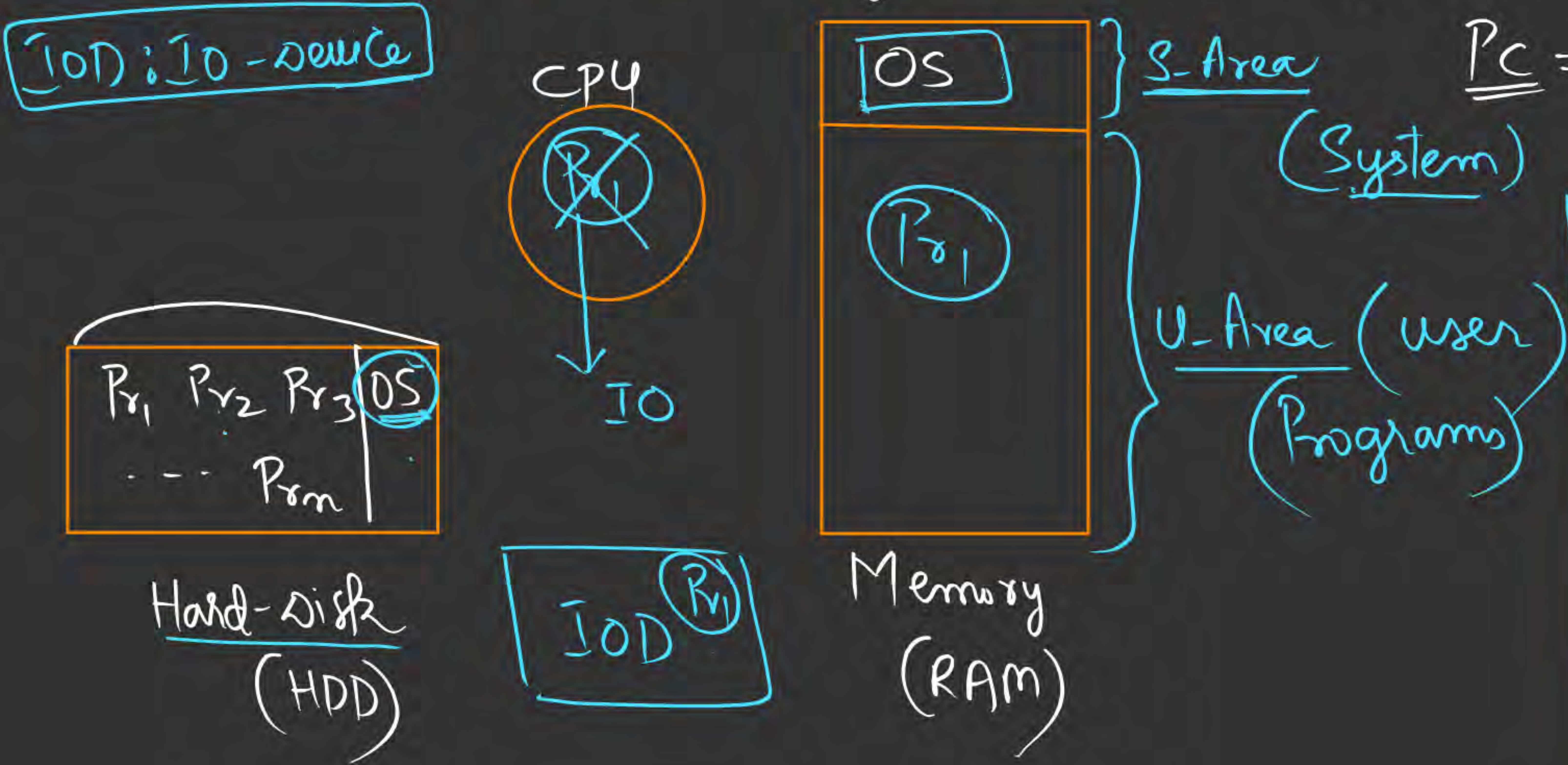
✓ UNI-PROGRAMMED

MULTIPROGRAMMED

Uniprogramming: Ability of the O.S to hold a single Program in Memory.

IO: Input-output

IOD: IO-device



PC = Personal Computer

Stored Program Concept

Drawback of uniprogramming: Ex: MS-DOS (Non-GUI) (Disk O.S)

Single Program in Memory is not enough (Sufficient) to keep the cpu always busy;

Idleness of cpu:

→ Thruput will be less (Efficiency)

$$\text{Thruput} = \frac{\text{No. of Programs Completed}}{\text{Unit time}}$$

10 progs — 60's

? — 1 s

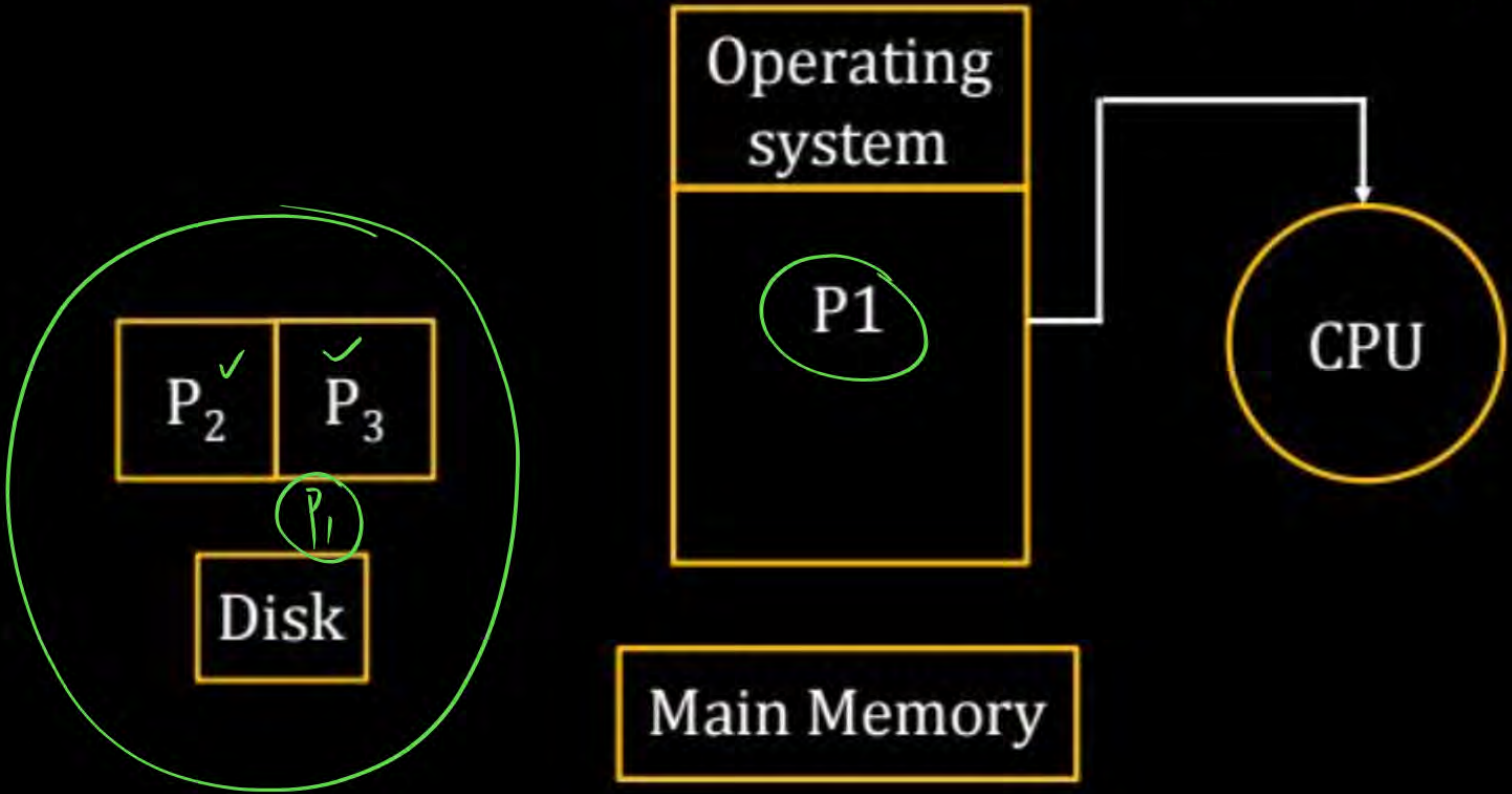
$\frac{10}{60} = \left(1/6\right)$

120 m — 20 topics

1m — ?

$20/120$

Schematic view of Uniprogrammed O.S



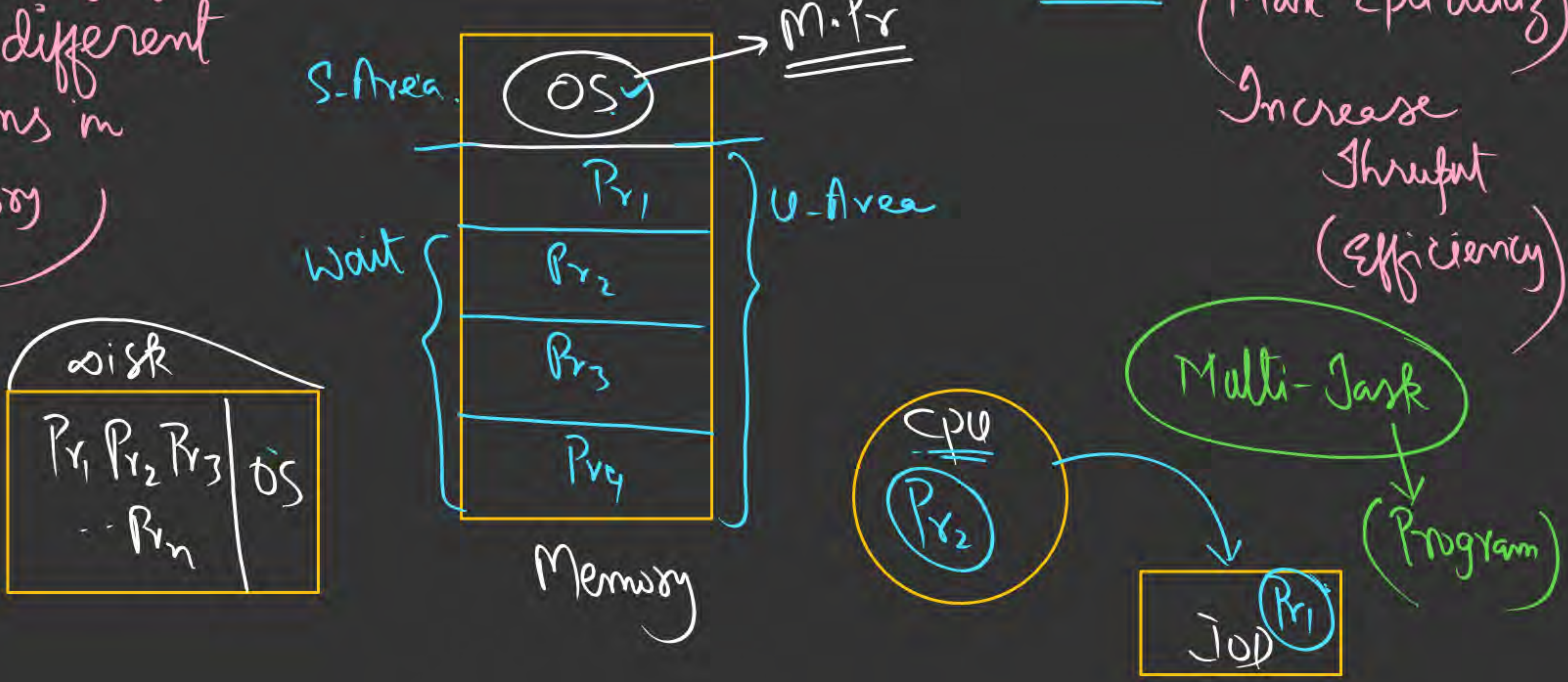
Multiprogramming : Ability of the O.S to hold(manage) multiple Ready to run Programs in Memory,

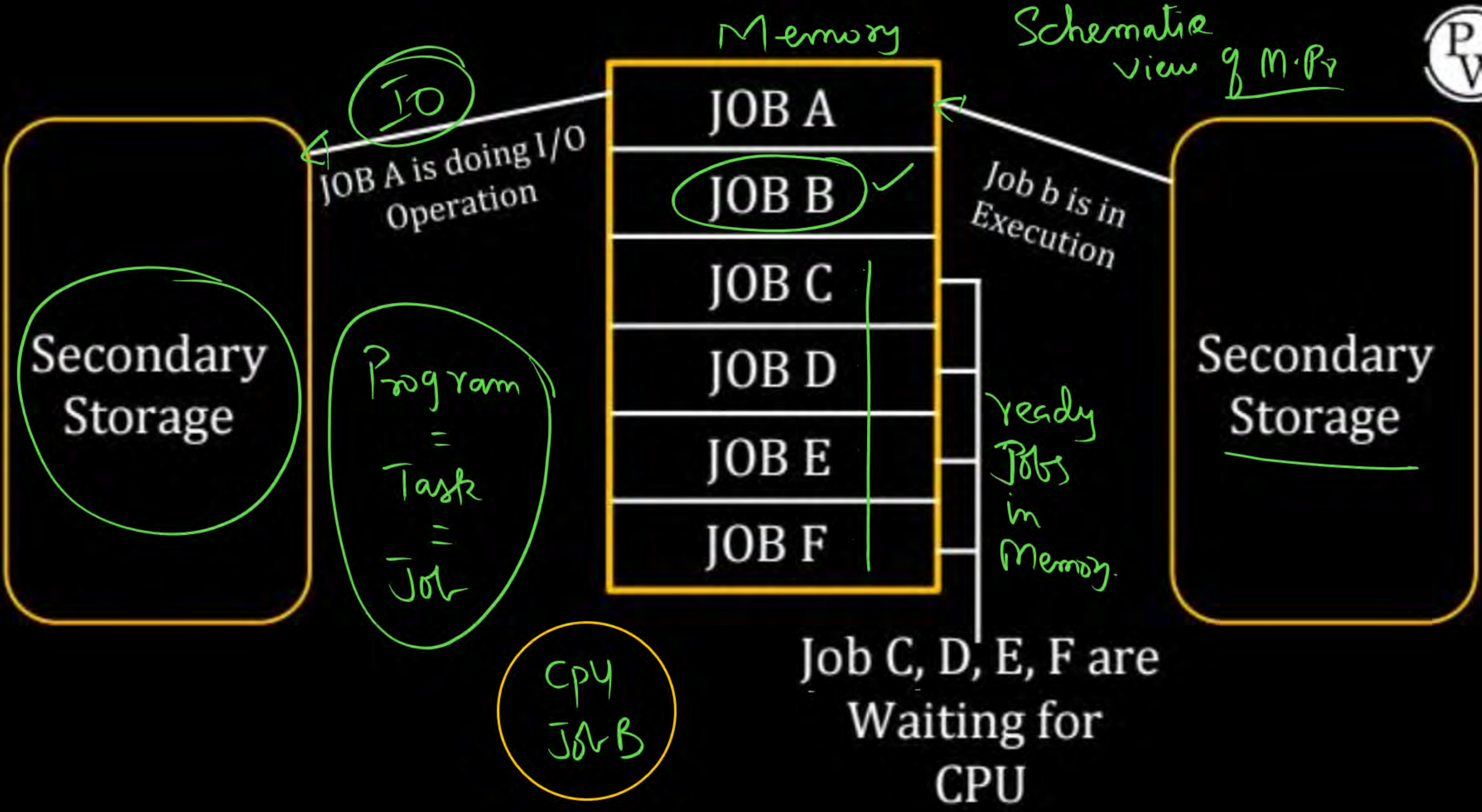
↓

(Multiplexing of cpu among different Programs in memory)

objective : (Max cpu utiliz)

Increase Thruput (Efficiency)





Memory

Schematic view of M.Pr

IO

JOB A is doing I/O Operation

JOB A

JOB B ✓

JOB C

JOB D

JOB E

JOB F

Job b is in Execution

Secondary Storage

Program = Task = Job

CPU
Job B

ready Jobs in Memory

Job C, D, E, F are Waiting for CPU

Secondary Storage

* Types of M. Pr.

1) Non-PreEmptive (N.Pr) (WIN 3.0)

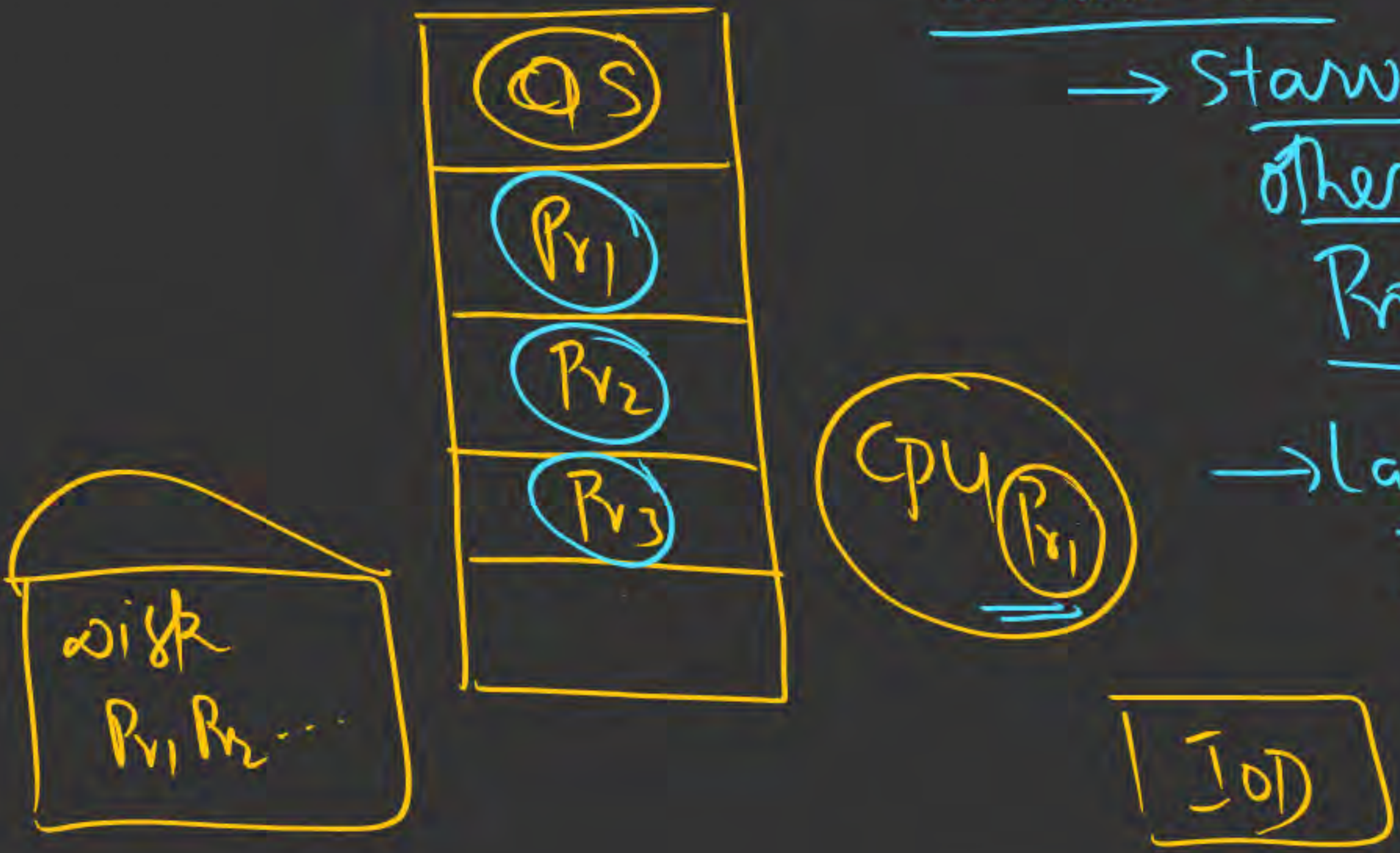
No forceful dealloc.

2) PreEmptive (Pr)

drawback:

→ Starvation to other waiting Programs

→ lack of Interactiveness



PreEmptive : Forceful deallocation

Non-Pr: running Program on cpu will not be forced to leave the cpu. But it will release the cpu voluntarily (on its own)

(i) Complete its execution

(ii) Needs IO/Sys. Call

PreEmptive M. Pr. O.S

(Objective) (Improve Interactiveness, responsiveness)

WIN-95
⋮
(WIN-10)
UNIX
LINUX
MAC

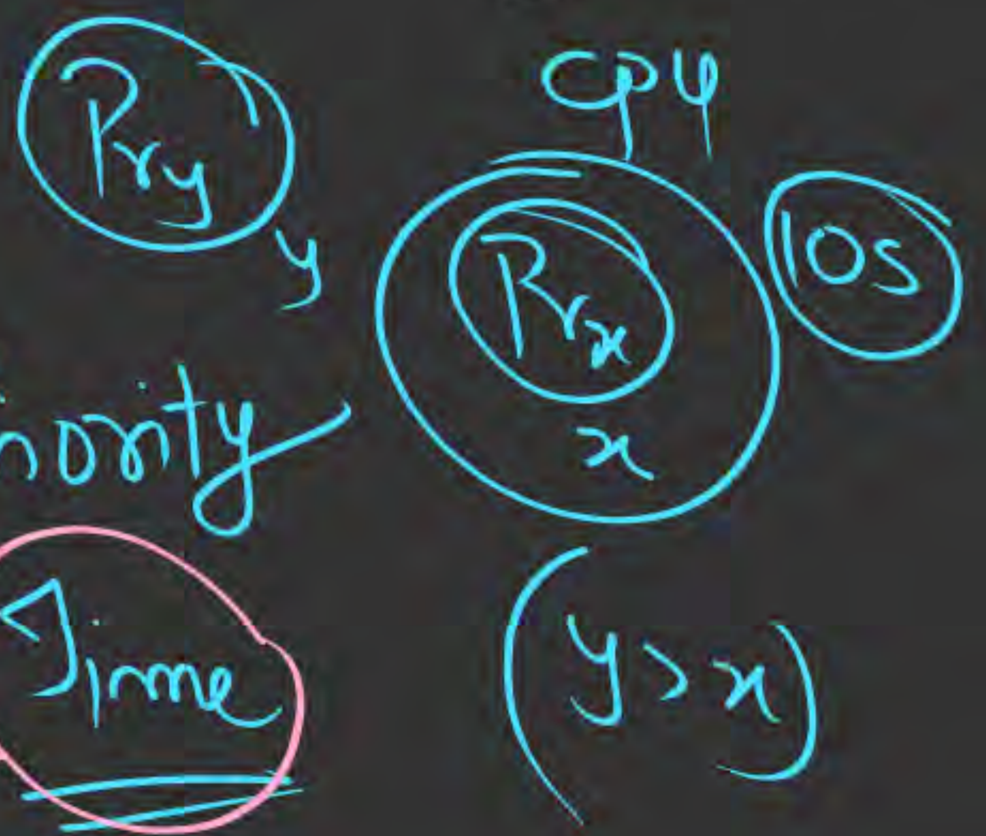
: Program running on cpu, can get (OS) forcefully deallocated from cpu (PreEmption), So that other waiting Programs can get their chance to run on cpu

(How is PreEmption done)

→ Priority

→ Time

(M.Pr. Timeshared OS)



<UNIX> <WIN-MS>

Multiprogramming vs Multi-Tasking

↓

Pre Emptive based M. Pr. O.S

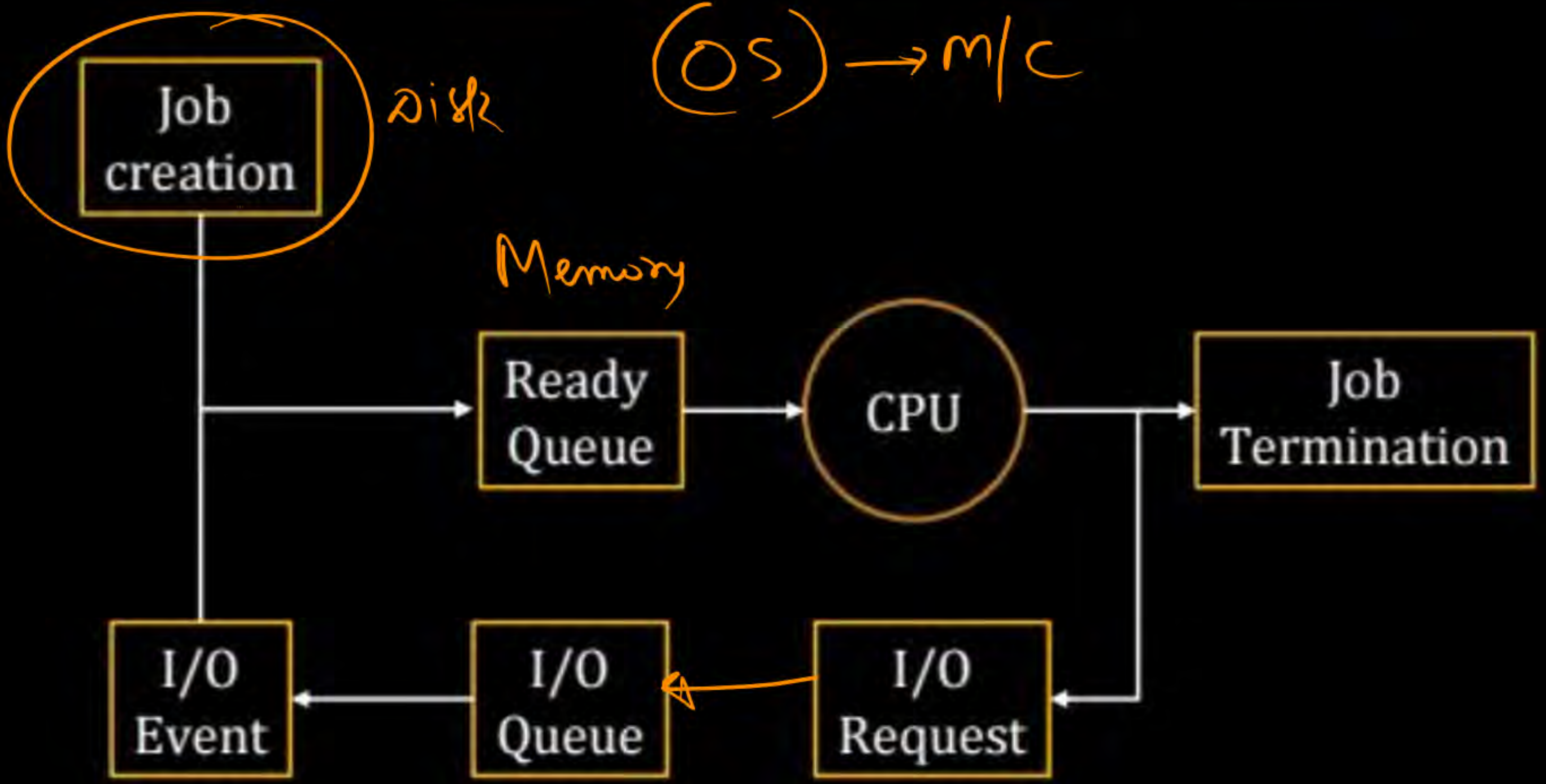
Objective

→ Study the design & Implementation | Policies of a Pre-Emptive based M. Pr. O. S (like UNIX|LINUX|WIN|MAC)

Multiprogramming

A Schematic view of Multiprogramming

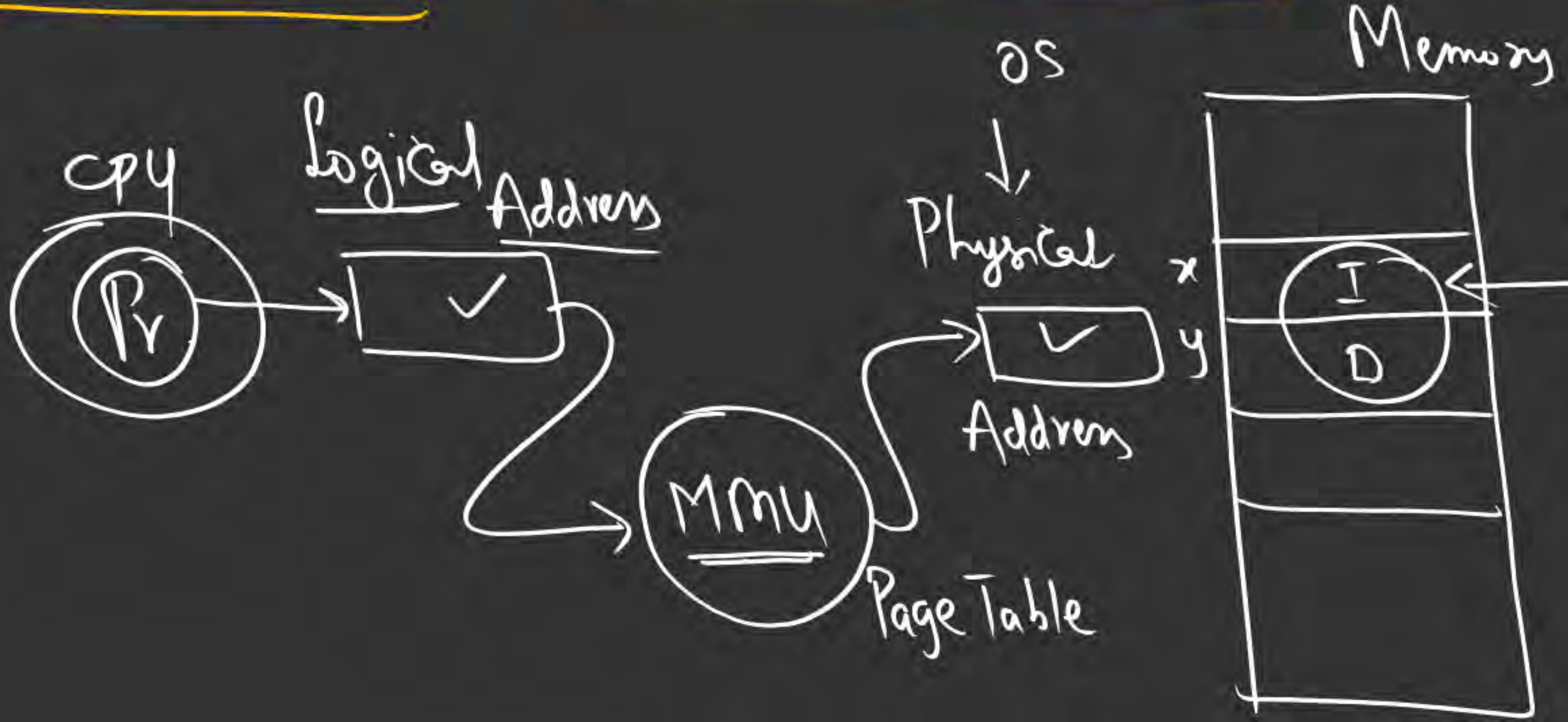
(OS) → m/c



* Architectural (H/w) Requirements for Implementing a M.Pr. O.S

① IO (Sec-Storage devices) (Disk) : DMA [Direct Memory Access] < CPU ; Memory ; IO (Sec. Storage) >

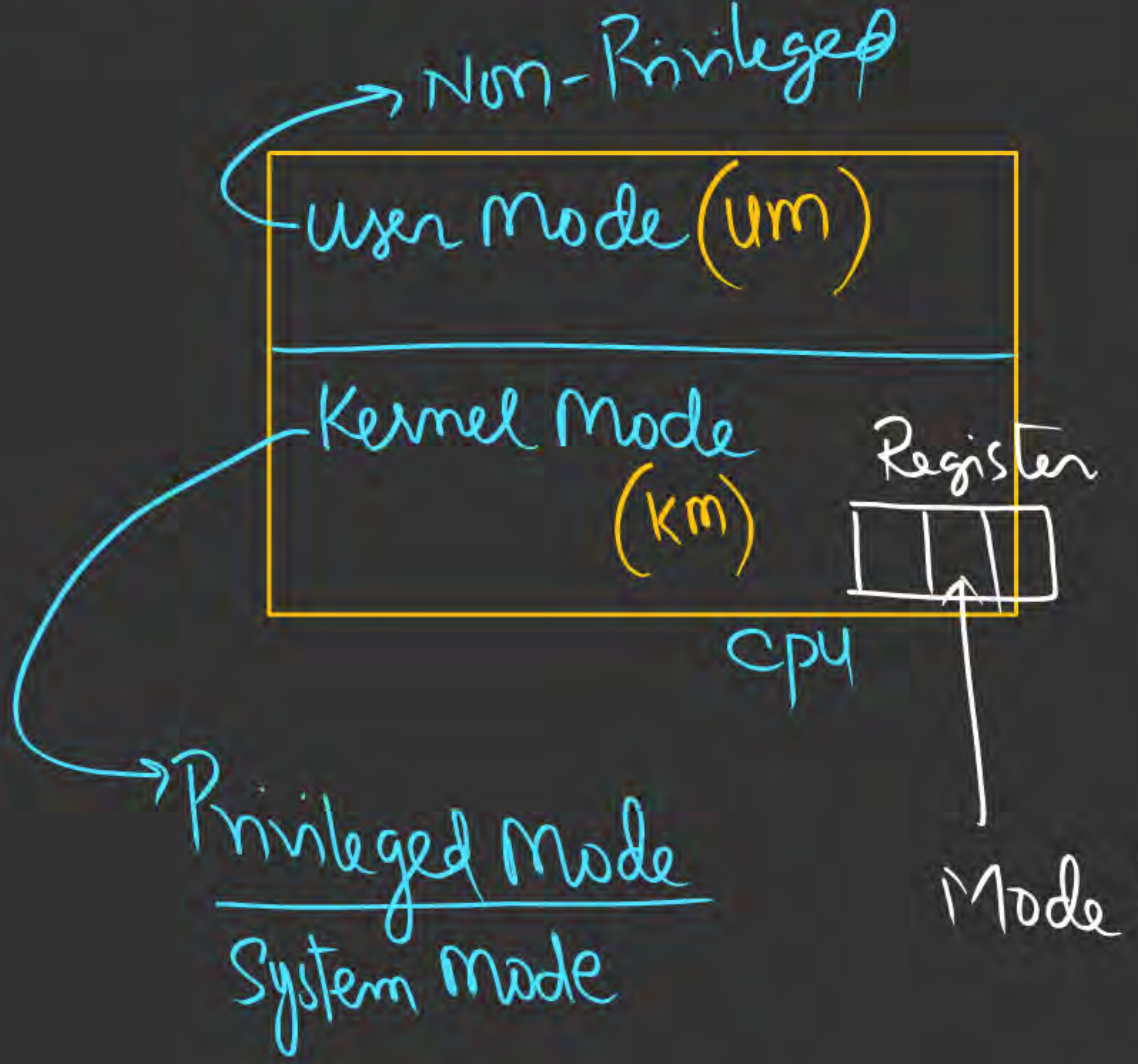
② Memory (RAM) : Address Translation.



MMU: Mem Mgmnt. unit

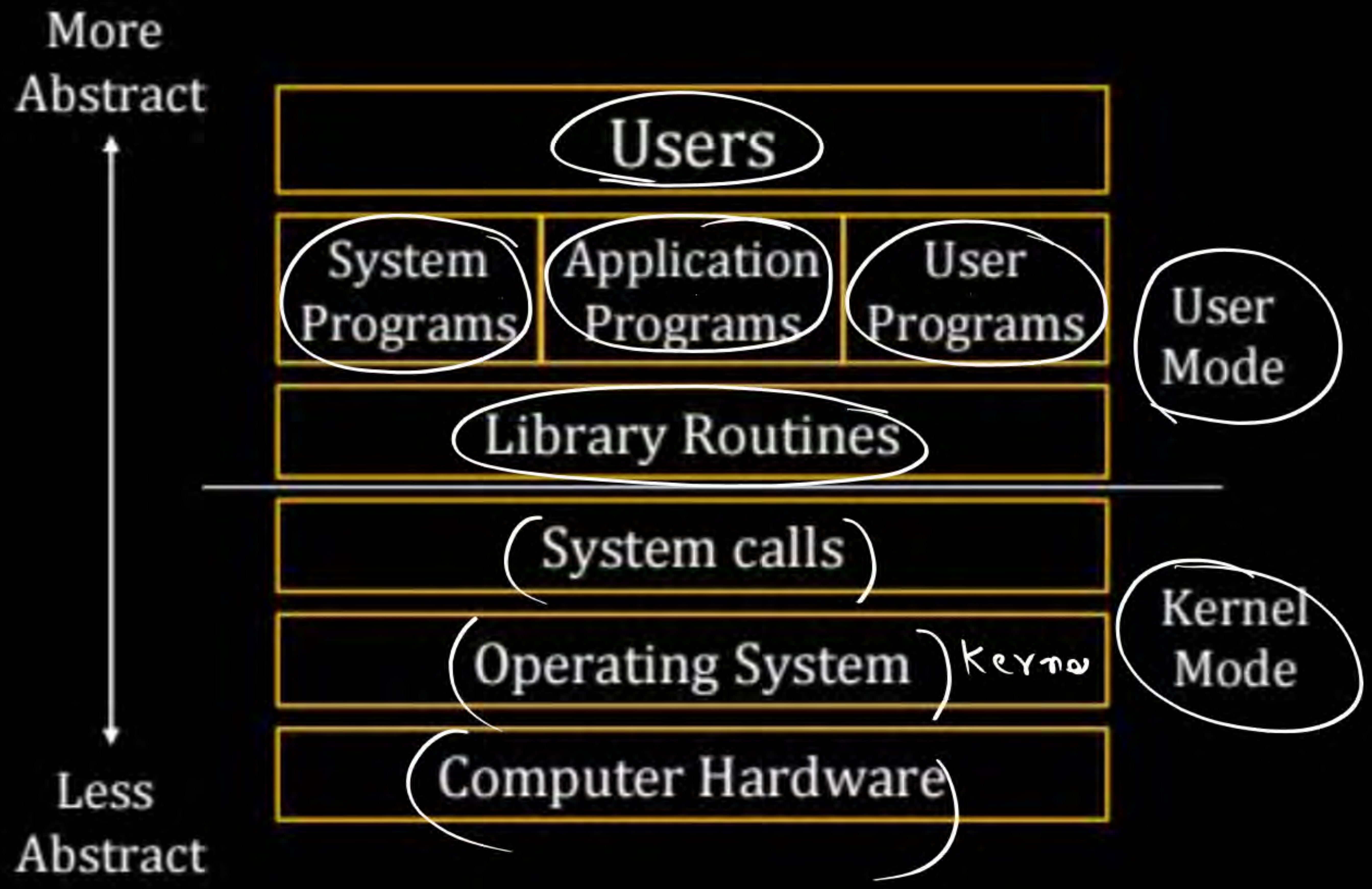
(*) 3) CPU : Dual mode operation :

Processor   Two



Mode bit < 0 → KM
1 → UM

UM	KM
1) All user appl. run in UM	1) All OS routines Programs run in K.M
2) UM is PreEmptive (Non-Atomic)	2) KM is non-PreEmptive(Atomic)
3)	



More Abstract

Users

System Programs

Application Programs

User Programs

Library Routines

User Mode

System calls

Operating System

Kerna

Kernel Mode

Computer Hardware

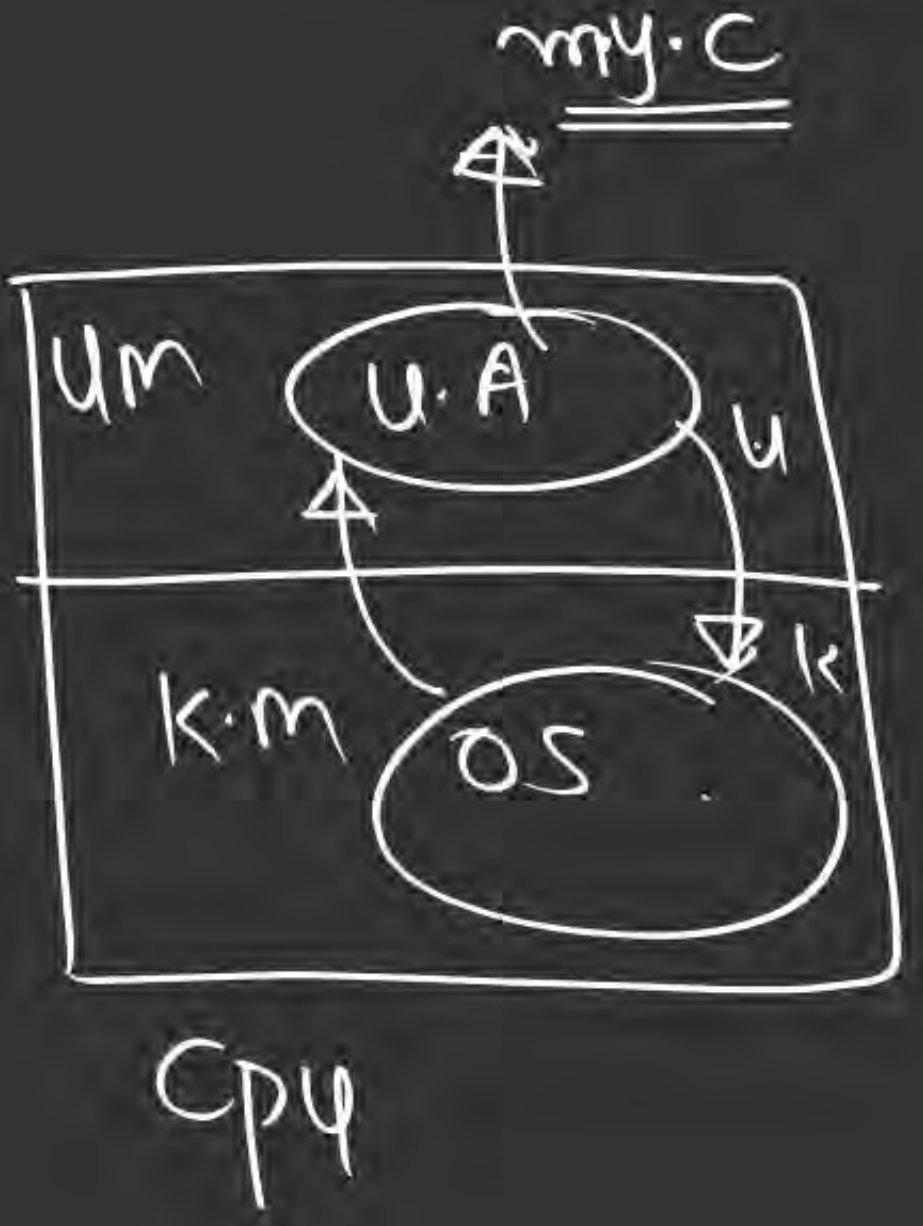
Less Abstract

→ Many a Times, it becomes necessary to shift the Mode from U → K & back from K → U ;

** Mode Shifting

(is needed to avail OS - Services)

(??)

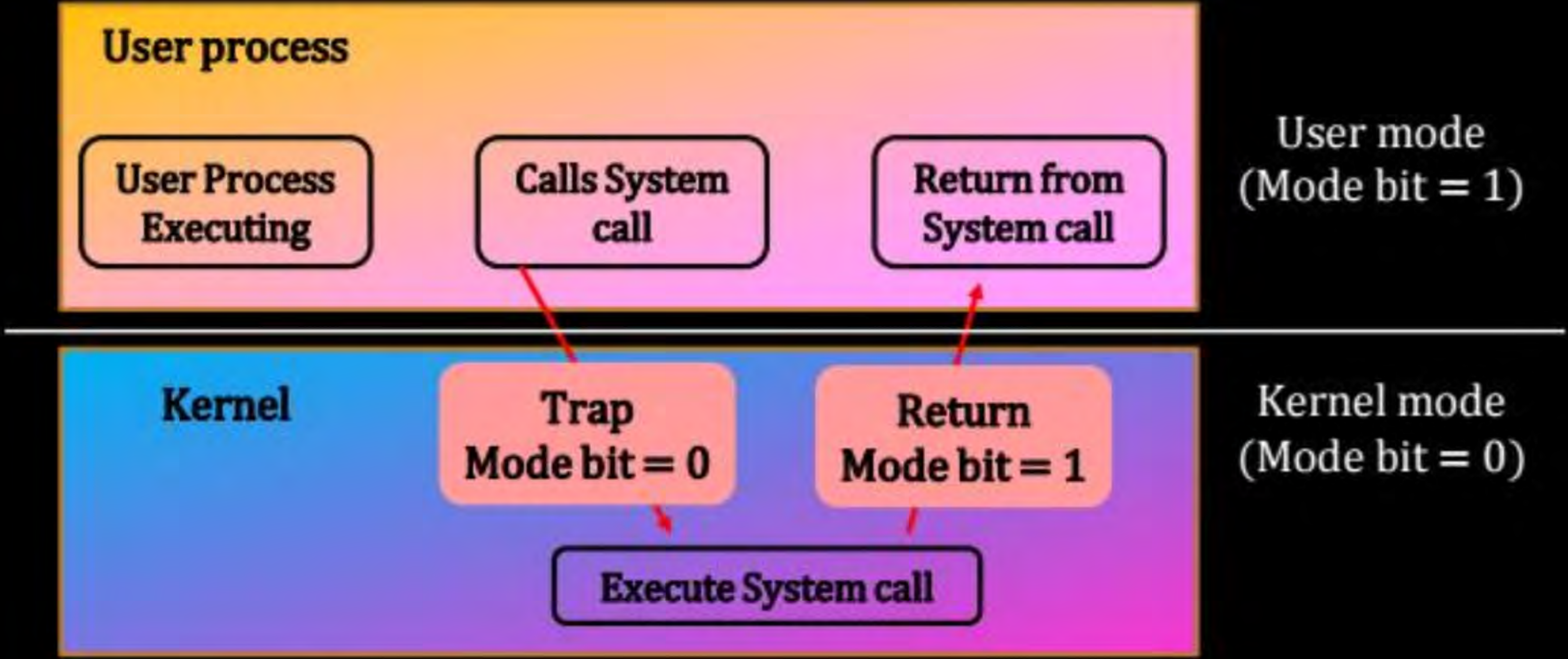


1. **Kernel Mode**

   In Kernel mode, the executing code has complete and unrestricted access to the underlying hardware. It can execute any CPU instruction and reference any memory address. Kernel mode is generally reserved for the lowest-level, most trusted functions of the operating system. Crashes in kernel mode are catastrophic; they will halt the entire PC.

2. **User Mode**

   In User mode, the executing code has no ability to directly access hardware or reference memory. Code running in user mode must delegate to system APIs to access hardware or memory. Due to the protection afforded by this sort of isolation, crashes in user mode are always recoverable. Most of the code running on your computer will execute in user mode.



User process

User Process Executing

Calls System call

Return from System call

User mode
(Mode bit = 1)

Kernel

Trap
Mode bit = 0

Return
Mode bit = 1

Execute System call

Kernel mode
(Mode bit = 0)

THANK YOU GW SOLDIERS !