Operating Systems

Quiz

Q. MMU is a _____ that converts logical address into the physical address.

a. system program
b. application program
c. firmware
d. all of the above
e. none of the above

Q. what is/are necessary and sufficient condition/s to occur deadlock.

a. resource can be allocated for any one process at a time
b. control of any resource cannot be taken away forcefully from a process
c. each process is holding one resource and requesting for a resource which is held by another process.
d. circular wait
e. all of the above

Quiz

Q. To recover system from deadlock, process which gets terminated is referred as a

a. terminated process
b. target process
c. victim process
d. all of the above

Q During Machine Boot, process to check peripherals connectivity is called as

A. BIOS
B. POST
C. Authentication
D. None of the above

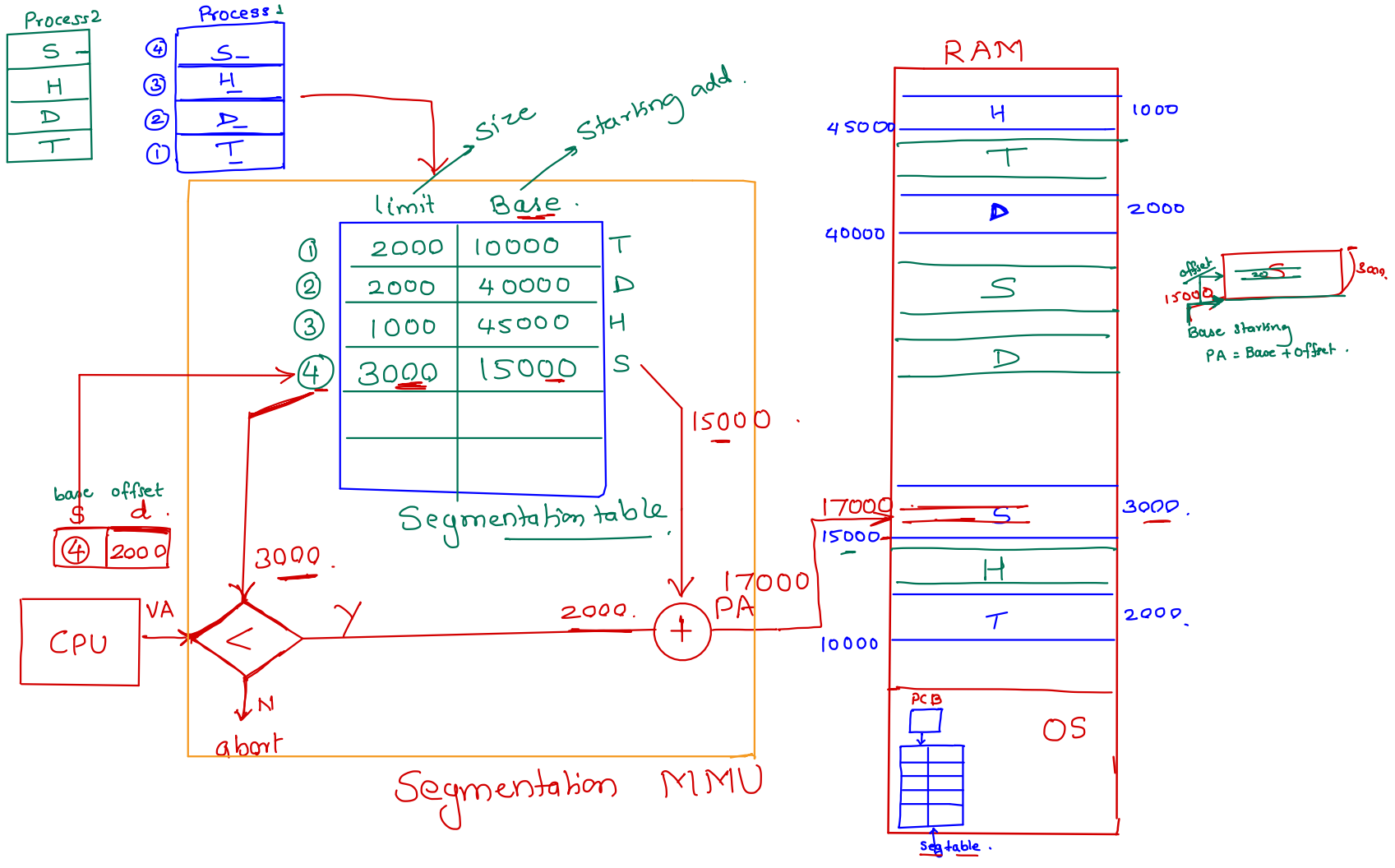
QUIZ

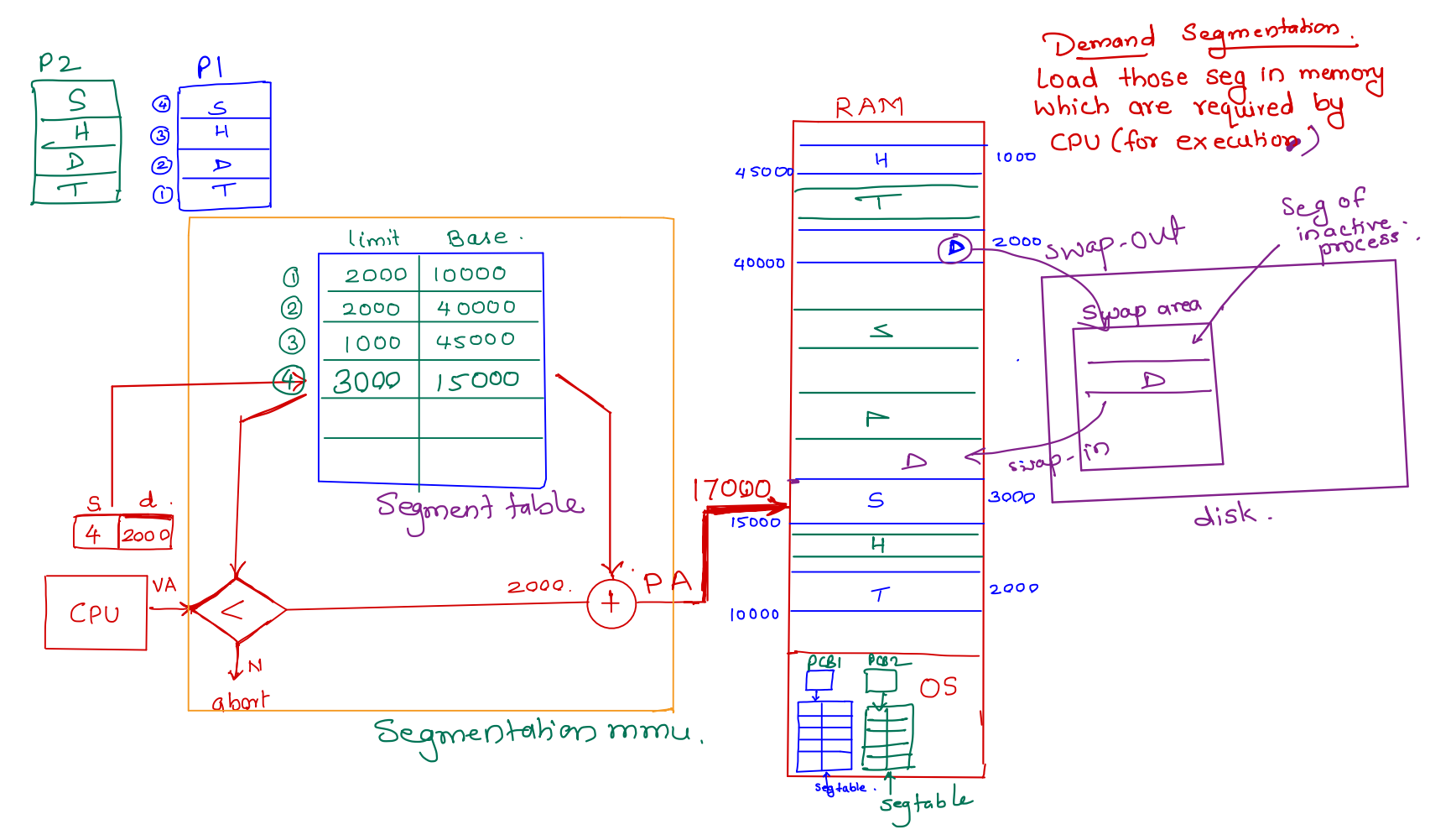
Q. _____ is used to implement virtual memory organisation.

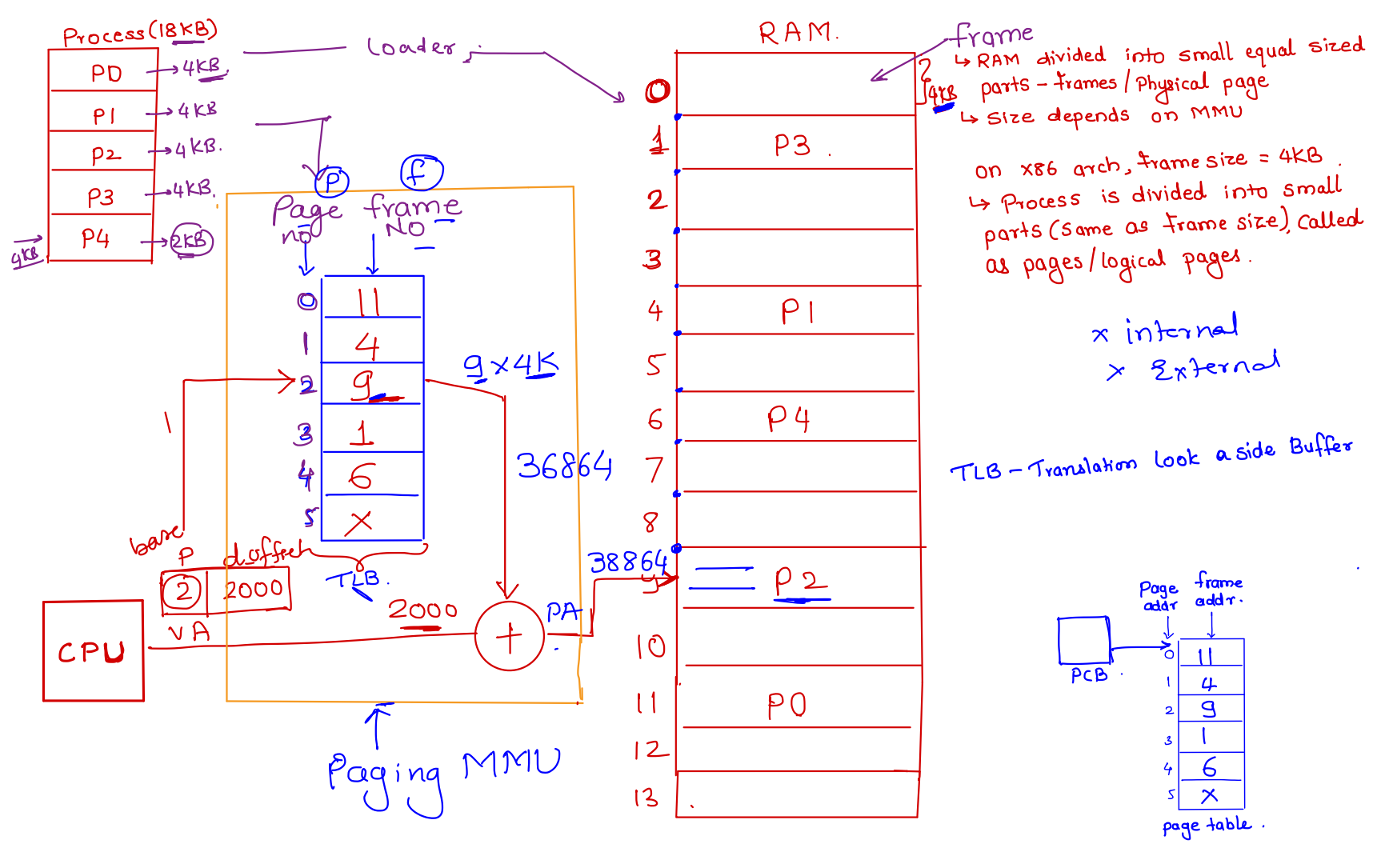
A. Page table
B. Frame table
C. MMU
D. None of the mentioned

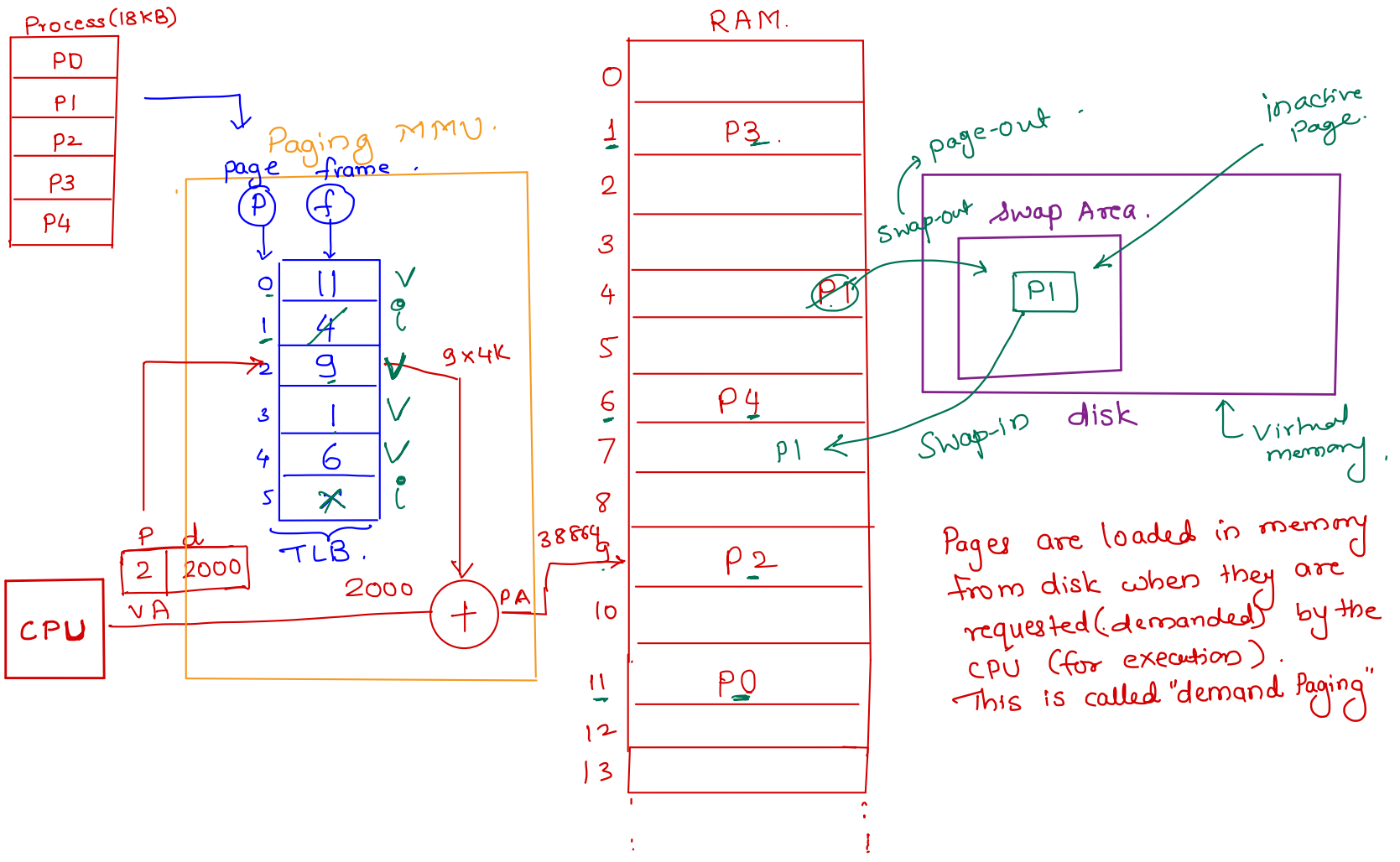
Q. Which of the following section contains magic number in an executable file?

A. bss section
B. code section
C. primary header
D. symbol table

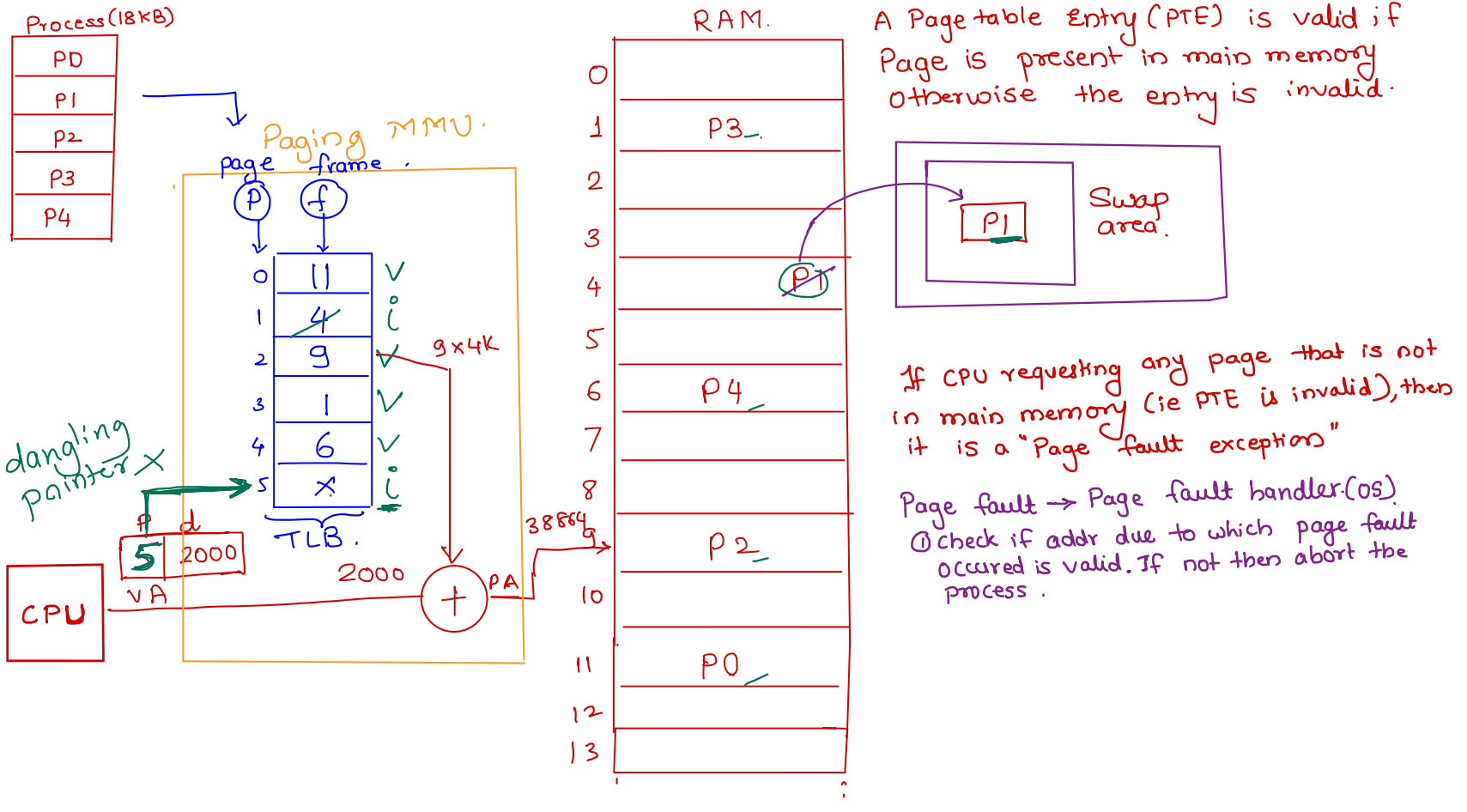

Process2
S
H
D
T
Process 1
4 S
3 H
2 D
1 T
Size
Starting add.
limit
Base.
1 2000 10000 T
2 2000 40000 D
3 1000 45000 H
4 3000 15000 S
Segmentation table.
base
offset
$
d.
4 2000
3000.
CPU
VA
<
Y
N
abort
2000.
+
17000
PA
15000.
Segmentation MMU
RAM
45000
H
1000
T
D
2000
40000
S
D
17000
S
3000.
15000
H
T
2000.
10000
PCB
OS
seg table.
offset
1500
Base starting
PA = Base + offset.


Demand Segmentation.
Load those seg in memory which are required by CPU (for execution)
P2
S
H
D
T
P1
4 S
3 H
2 D
1 T
RAM
45000
40000
1000
2000
3000
15000
10000
2000
H
T
D
S
D
swap-out
Seg of inactive process.
Swap area
D
swap-in
disk.
limit
Base.
1 2000 10000
2 2000 40000
3 1000 45000
4 3000 15000
Segment table
S
d.
4
2000
CPU
VA
<
N
abort
2000.
+
PA
17000
S
H
T
PCB1
PCB2
OS
segtable.
segtable
Segmentation mmu.


Process (18KB)
PD
P1
P2
P3
P4
4KB
4KB
4KB
4KB
2KB
4KB
Loader
RAM.
frame
RAM divided into small equal sized parts - frames / Physical page
Size depends on MMU
4KB
on x86 arch, frame size = 4KB.
Process is divided into small parts (same as frame size), called as pages / logical pages.
x internal
x External
TLB - Translation look a side Buffer
P
f
Page no
frame NO
0 11
1 4
2 9
3 1
4 6
5 X
9x4K
36864
38864
base
P
doffset
2
2000
TLB.
2000
VA
CPU
+
PA
Paging MMU
0
1 P3
2
3
4 P1
5
6 P4
7
8
9 P2
10
11 P0
12
13
PCB
Page addr
frame addr.
0 11
1 4
2 9
3 1
4 6
5 X
page table.



Pages are loaded in memory from disk when they are requested (demanded) by the CPU (for execution).
This is called "demand Paging"

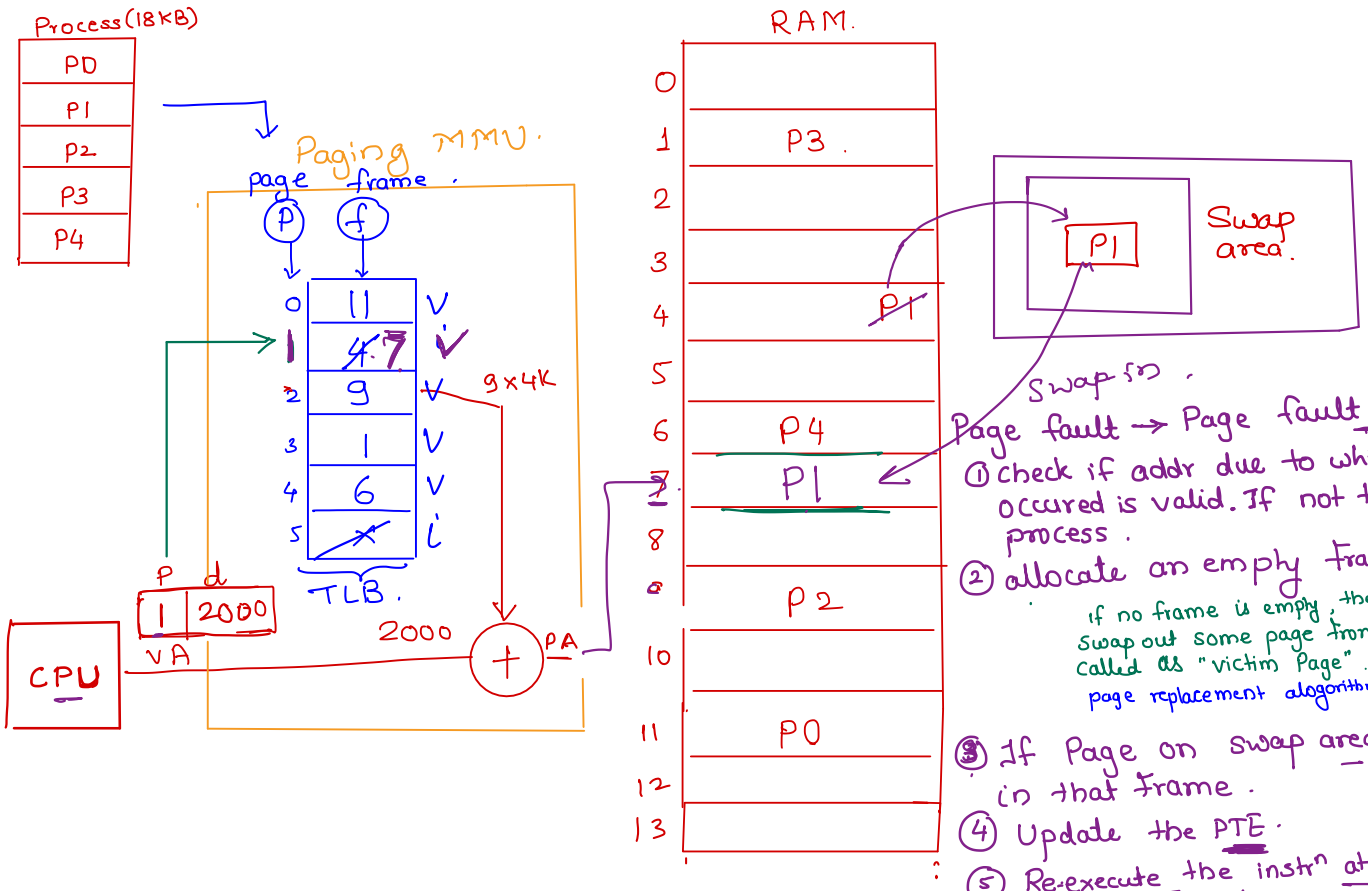


A Page table entry (PTE) is valid if Page is present in main memory otherwise the entry is invalid.

If CPU requesting any page that is not in main memory (ie PTE is invalid), then it is a "Page fault exception"

Page fault → Page fault handler (OS)

① Check if addr due to which page fault occured is valid. If not then abort the process.



Page replacement Algo.

1. FIFO
2. Optimal
3. LRU - Least Recently Use
4. MRU - Most Recently Use
5. LFU - Least frequently Use
6. MFU - Most Frequently Use
7. Second Chance LRU

Page fault → Page fault handler (OS)

1. check if addr due to which page fault occured is valid. If not then abort the process.
2. allocate an emply frame.
   - if no frame is empty, then OS needs to swap out some page from memory - called as "victim Page".
   - page replacement alogorithm to decide.
3. If Page on swap area then swap in page in that frame.
4. Update the PTE.
5. Re-execute the instn at which page fault occured.

Memory Management

Segmentation

* Instead of allocating contiguous memory for the whole process, contiguous memory for each segment can be allocated. This scheme is known as "segmentation".

* Since process does not need contiguous memory for entire process, external fragmentation will be reduced.

* In this scheme, PCB is associated with a segment table which contains base and limit (size) of each segment of the process.

* During context switch these values will be loaded into MMU segment table.

* CPU request virtual address in form of segment address and offset address.

* Based on segment address appropriate base-limit pair from MMU is used to calculate physical address as shown in diag.

* MMU also contains STBR register which contains address of process's segment table in the RAM.

Demand Segmentation

* If virtual memory concept is used along with segmentation scheme, in case low memory, OS may swap out a segment of inactive process.

* When that process again start executing and ask for same segment (swapped out), the segment will be loaded back in the RAM. This is called as "demand segmentation".

* If segment is present in main memory, its entry in seg table is said to be valid. If segment is swapped out, its entry in segment table is said to be invalid.

Memory Management

Paging

* RAM is divided into small equal sized partitions called as "frames" / "physical pages".
* Process is divided into small equal sized parts called as "pages" or "logical/virtual pages".
* page size = frame size.
* One page is allocated to one empty frame.
* OS keep track of free frames in form of a linked list.
* Each PCB is associated with a table storing mapping of page address to frame address. This table is called as "page table".
* During context switch this table contents are loaded into MMU.
* CPU requests a virtual address in form of page address and offset address. It will be converted into physical address as shown in diag.
* MMU also contains a PTBR, which keeps address of page table in RAM.
* If a page is not utilizing entire frame allocated to it (i.e. page contents are less than frame size), then it is called as "internal fragmentation".
* Frame size can be configured in the hardware. It can be 1KB, 2KB or 4KB, ...
* Typical Linux and Windows OS use page size = 4KB.

Memory Management

Demand Paging

* When virtual memory is used with paging memory management, pages can be swapped out in case of low memory.

- The pages will be loaded into main memory, when they are requested by the CPU. This is called as "demand paging".
- A page table entry (PTE) is valid if page is present in main memory , otherwise entry is invalid.
- What are the reason for invalid entry →page is swap out or entry is invalid
- If CPU requesting any page that is not in main memory(i.e. PTE is invalid), then it is a page fault exception.
- **Page fault → Page fault handler(OS generate)**

1. Check if address due to which page fault occurred is valid. If not , then abort the process.
2. Allocate an empty frame.
3. If page is on swap area, swap in page in that frame
4. Update PTE
5. Re-execute instruction at which page fault occurred.

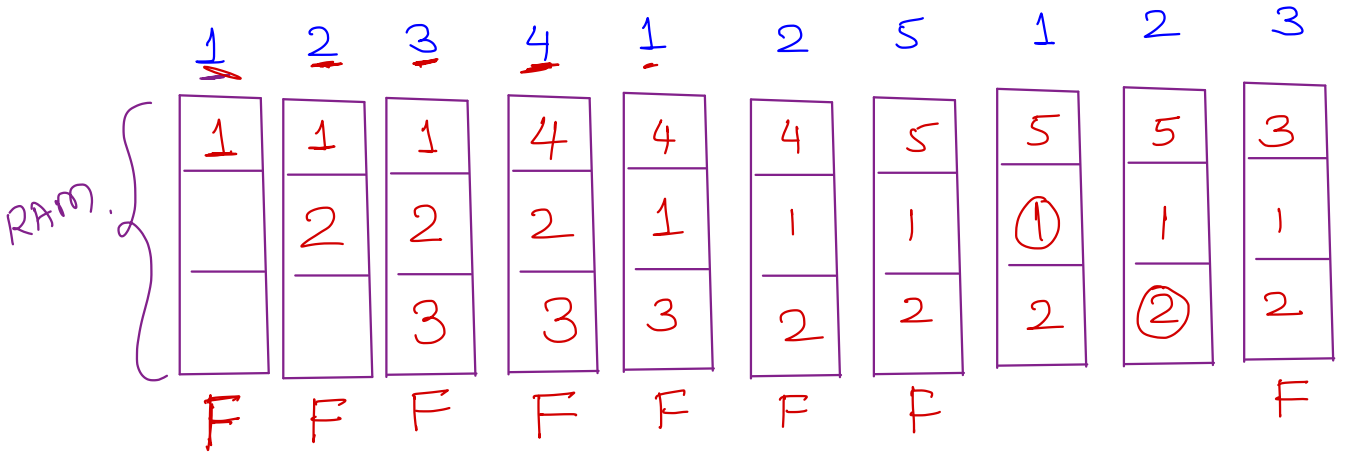
Paging

- If no frame is empty, then OS needs to swap-out some page from memory this is called as “victim page”
- Page replacement algorithm also to decide victim page.

- **Page replacement algorithm**

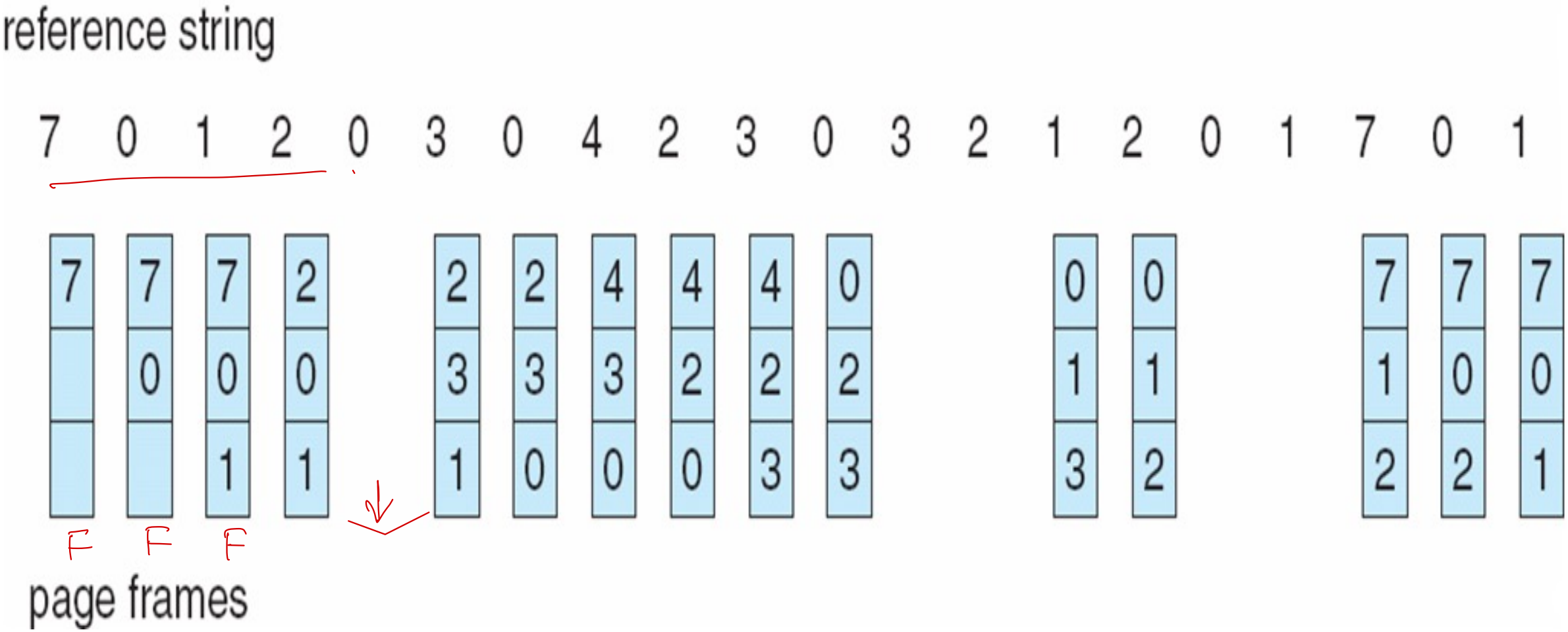
1. FIFO – First in First out
2. Optimal
3. LRU – List Recently used
4. MRU - Most Recently used
5. LFU - List Frequently used
6. MFU - Most Frequently used

① FIFO

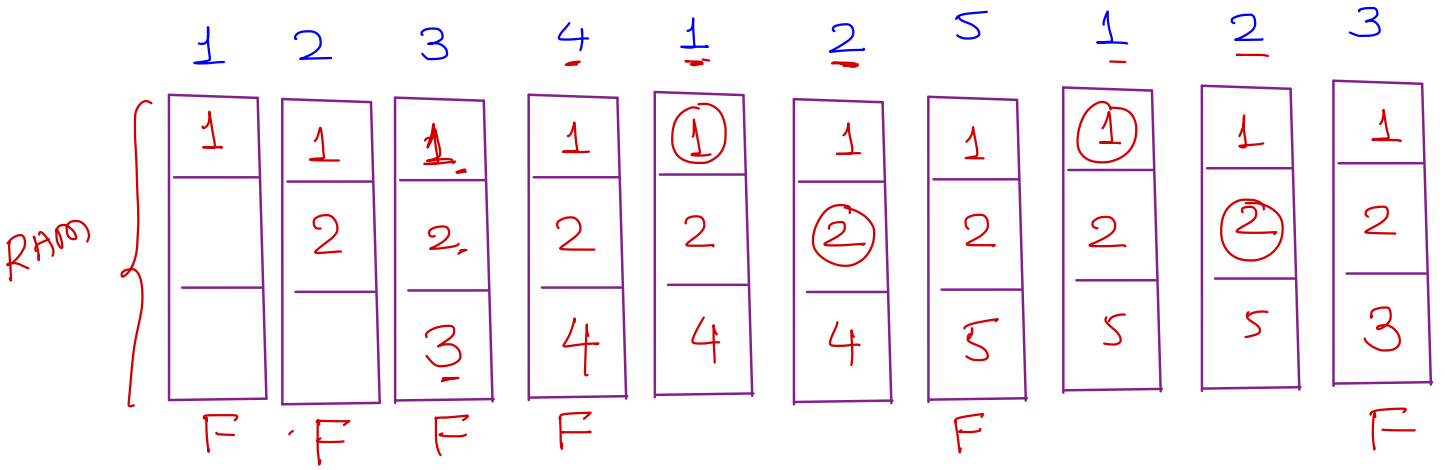


Page fault = 8

FIFO Page Replacement

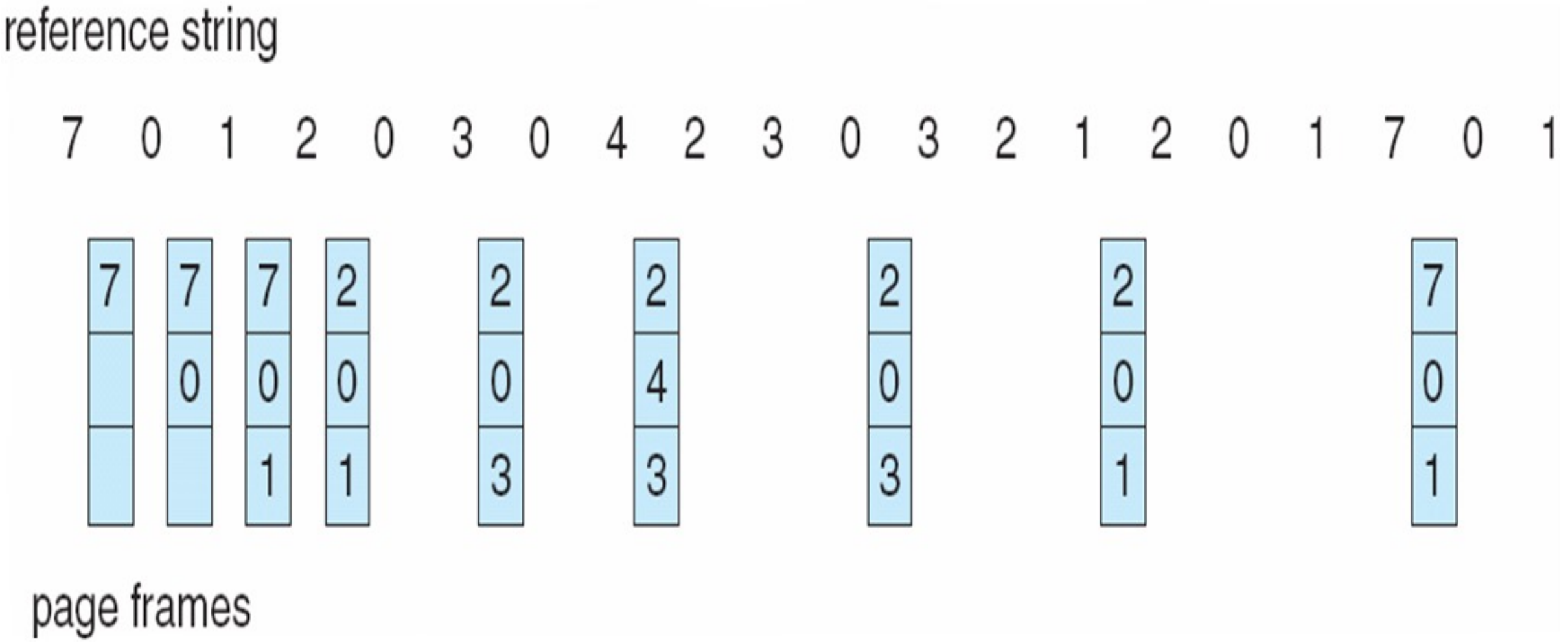


② Optimal Page Replacement :

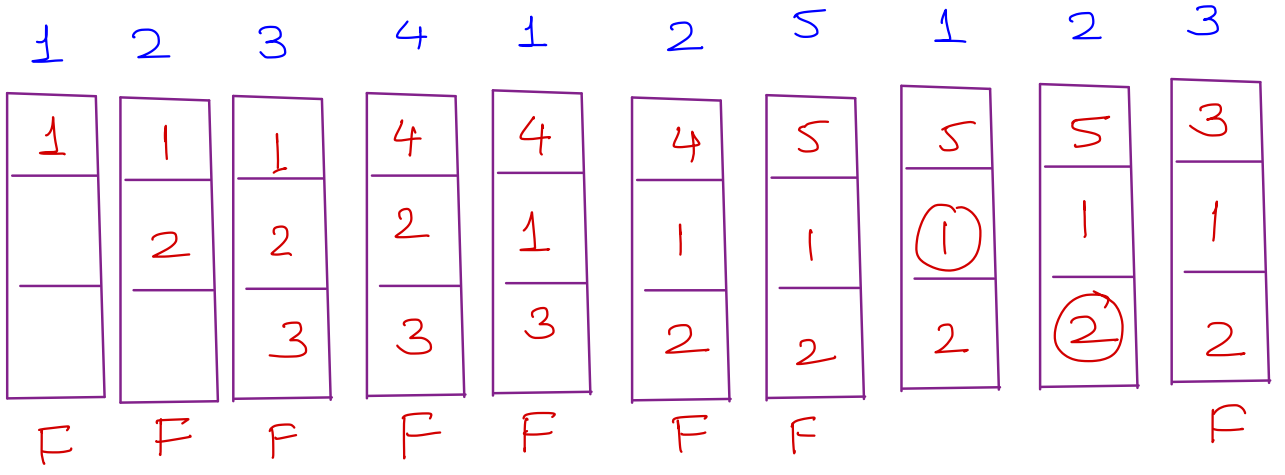


Page fault - 6.

Optimal Page Replacement

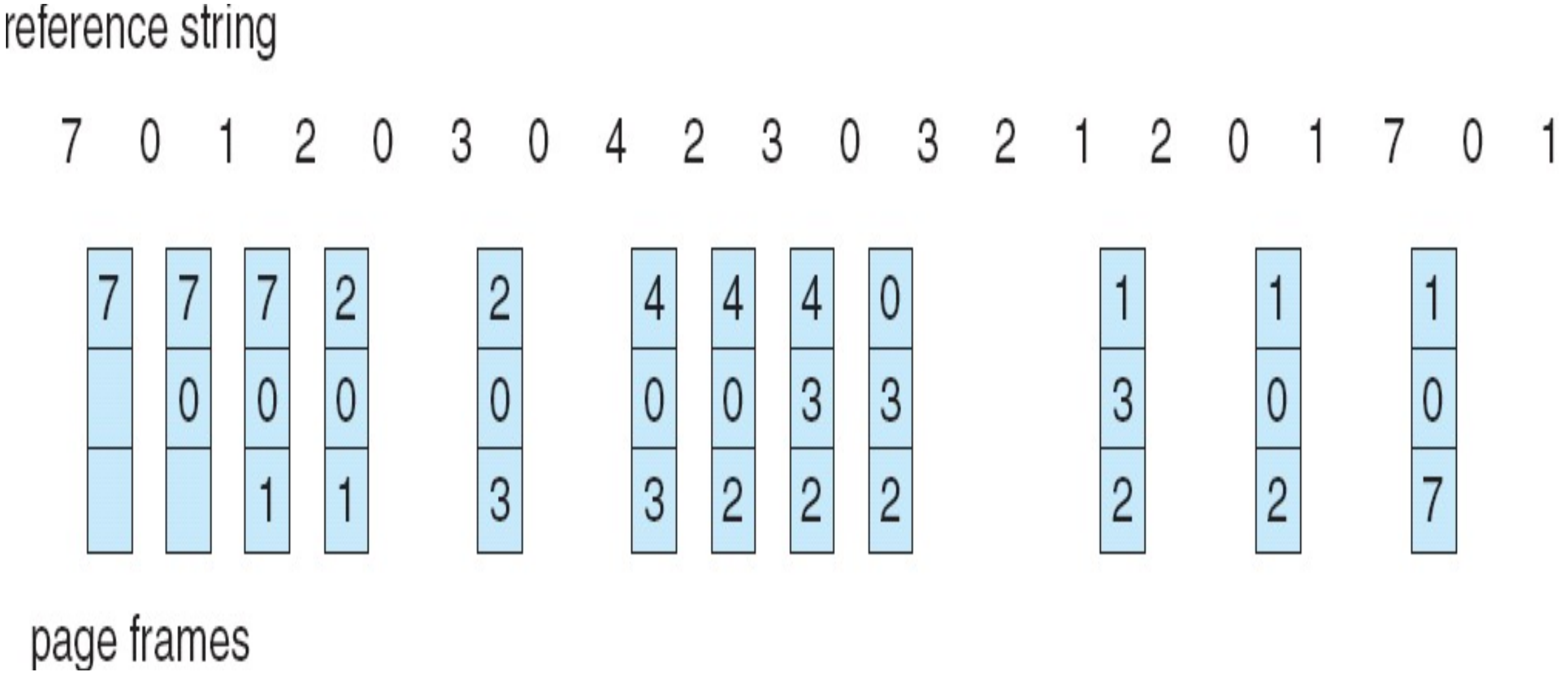


③ LRU Page Replacement:



Page fault - 8

LRU Page Replacement



Thrashing

- If a process does not have “enough” pages, the page-fault rate is very high
  - low CPU utilization
  - OS thinks it needs increased multiprogramming
  - adds another process to system
- *Thrashing* is when a process is busy swapping pages in and out

Q. Which of the following memory allocation method is faster?

A. best fit

B. first fit

C. worst fit

D. none of the above

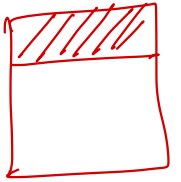
Q. Memory remains unused which is internal to the partition then it is referred as _____.

a. an external fragmentation

b. an internal fragmentation

c. both options A & B

d. none of the above



Q. The associatively mapped virtual memory makes use of _____
a) TLB
b) Page table
c) Frame table
d) None of the mentioned

Q. The operating system maintains a _____ table that keeps track of how many frames have been allocated, how many are there, and how many are available.
a) memory
b) mapping
c) page
d) frame

Operating System Concepts

 # UNIX Operating System:

- UNIX: UNICS – **Uniplexed Information & Computing Services/System.**

- UNIX was developed at **AT&T Bell Labs** in US, in the decade of 1970's by **Ken Thompson, Denies Ritchie** and team.

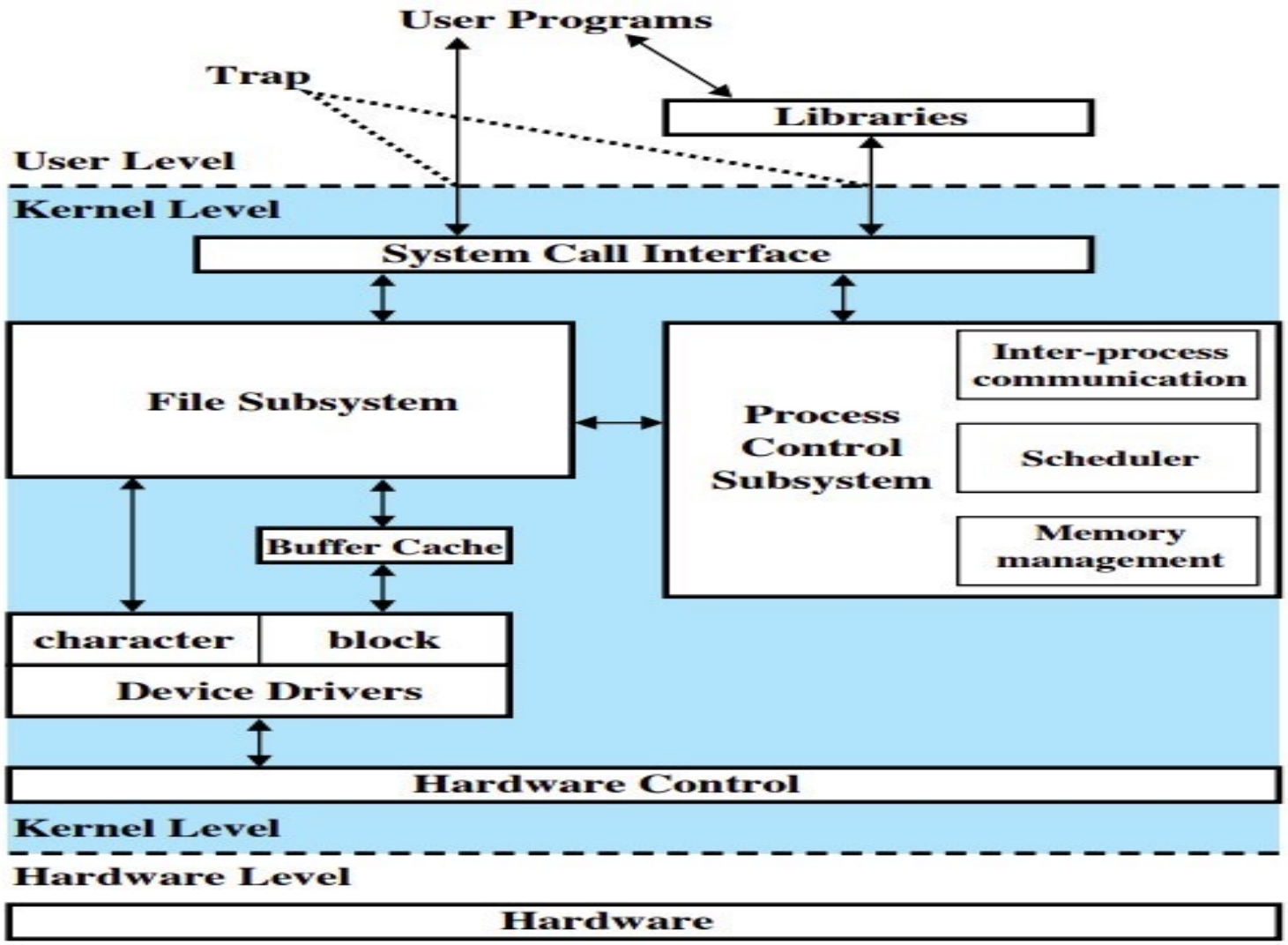
- It was first run on a machine **DEC-PDP-7** (Digital Equipment Corporation – Programmable Data Processing-7).

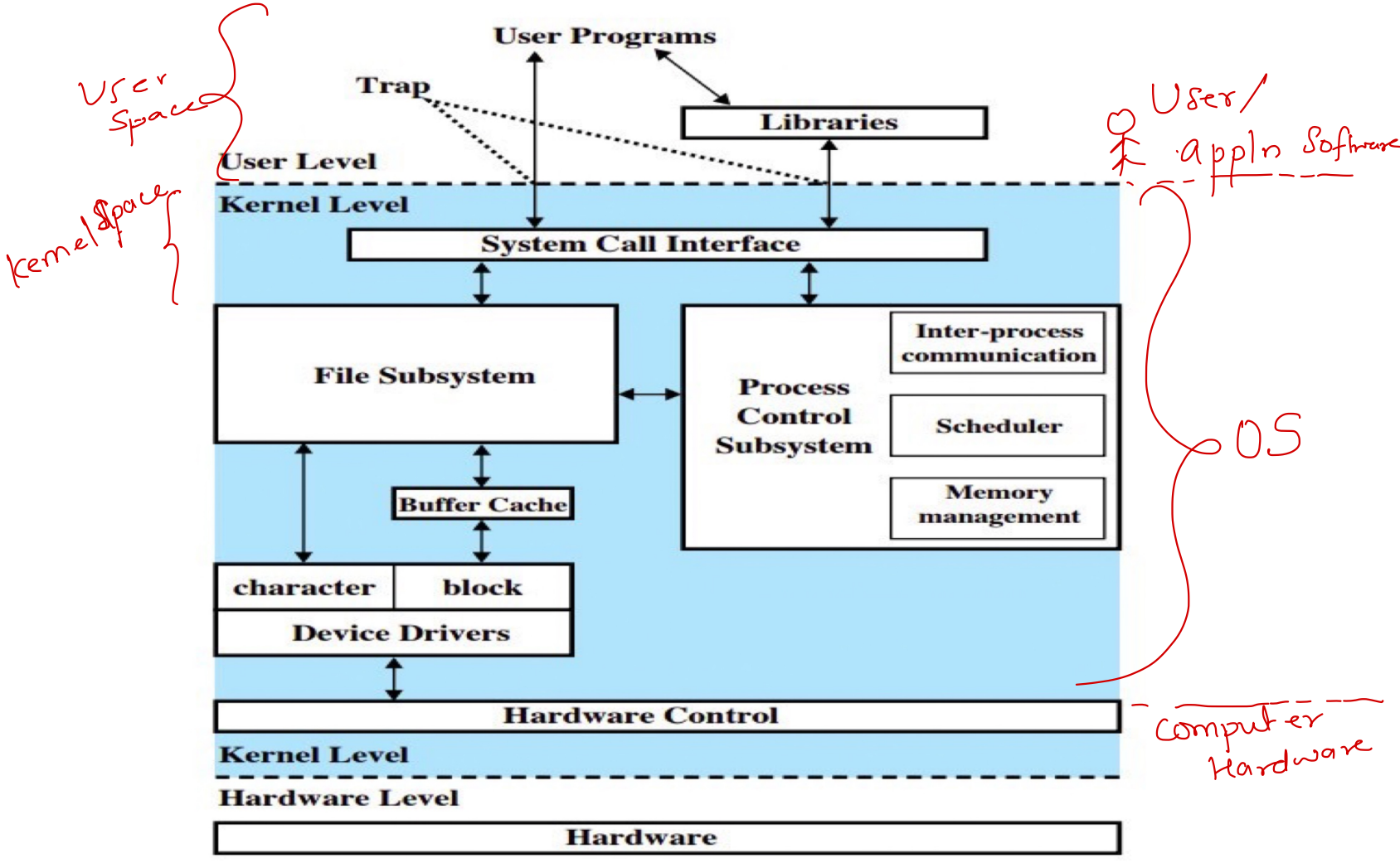
- UNIX is the first **multi-user, multi-programming** & **multi-tasking** operating system.

- UNIX was specially designed for developers by developers

- System architecture design of UNIX is followed by all modern OS's like Windows, Linux, MAC OS X, Android etc..., and hence UNIX is referred as **mother of all modern operating systems.**

Operating System Concepts




User Programs
Trap
Libraries
User Level
Kernel Level
System Call Interface
File Subsystem
Process Control Subsystem
Inter-process communication
Scheduler
Memory management
Buffer Cache
character
block
Device Drivers
Hardware Control
Kernel Level
Hardware Level
Hardware
User Space
Kernel Space
User / appln Software
OS
Computer Hardware

Operating System Concepts

- Kernel acts as an interface between programs and hardware.

- Operating System has subsystems like **System Call Interface Block, File Subsystem Block, Process Control Subsystem Block (which contains IPC, Memory Management & CPU Scheduling), Device Driver, Hardware Control/Hardware Abstraction Layer.**

- There are two major subsystems:

1. Process Control Subsystem
2. File Subsystem

- In UNIX, whatever is that can be stored is considered as a file and whatever is active is reffered as a process.

- **File has space & Process has life.**

Operating System Concepts

- From UNIX point of view all devices are considered as a file

- In UNIX, devices are catagorises into two catagories:

1. Character Devices: Devices from which data gets transfered character by character --> character special device file

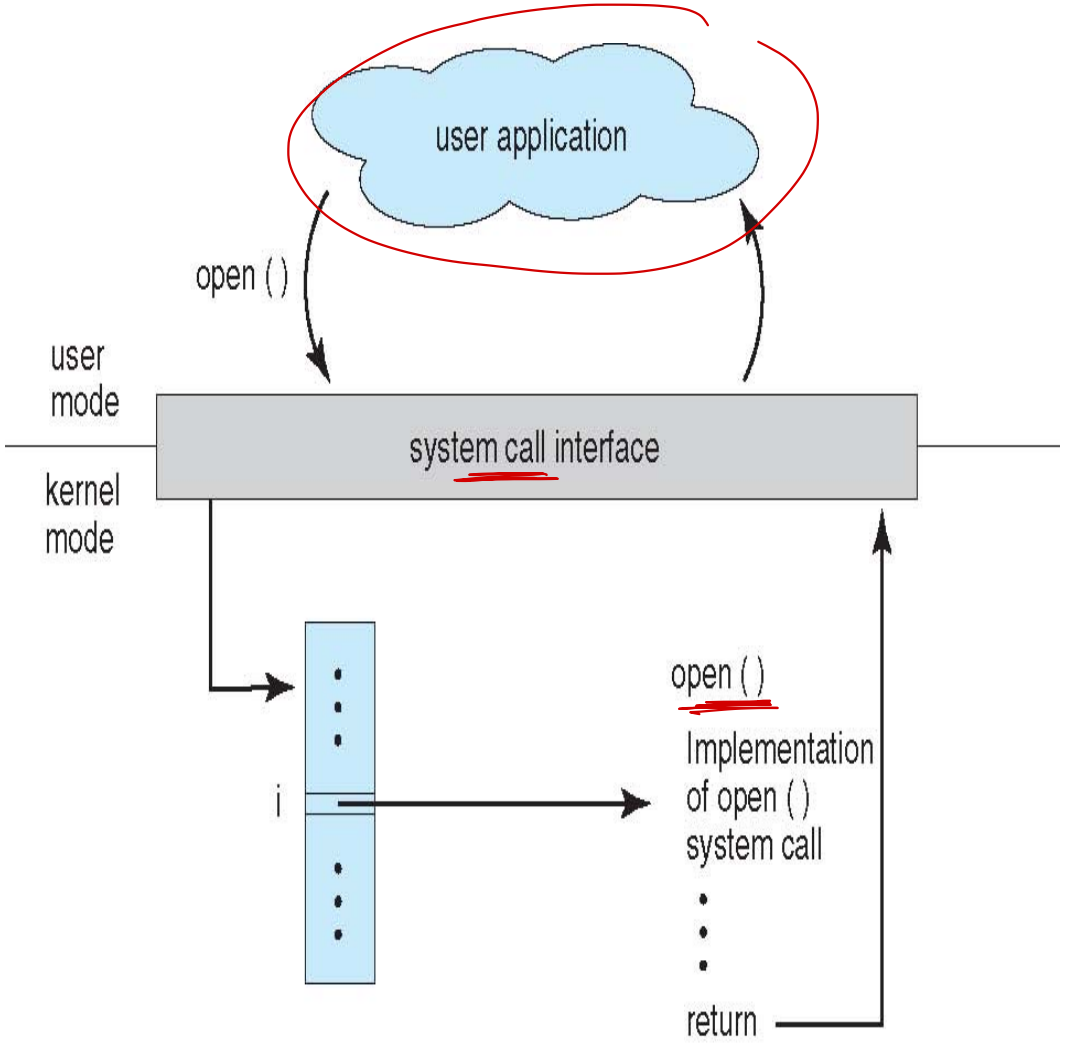
e.g. keyboard, mouse, printer, monitor etc...

2. Block Devices: Devices from which data gets transfered block by block --> block special device file

e.g. all storage devices.

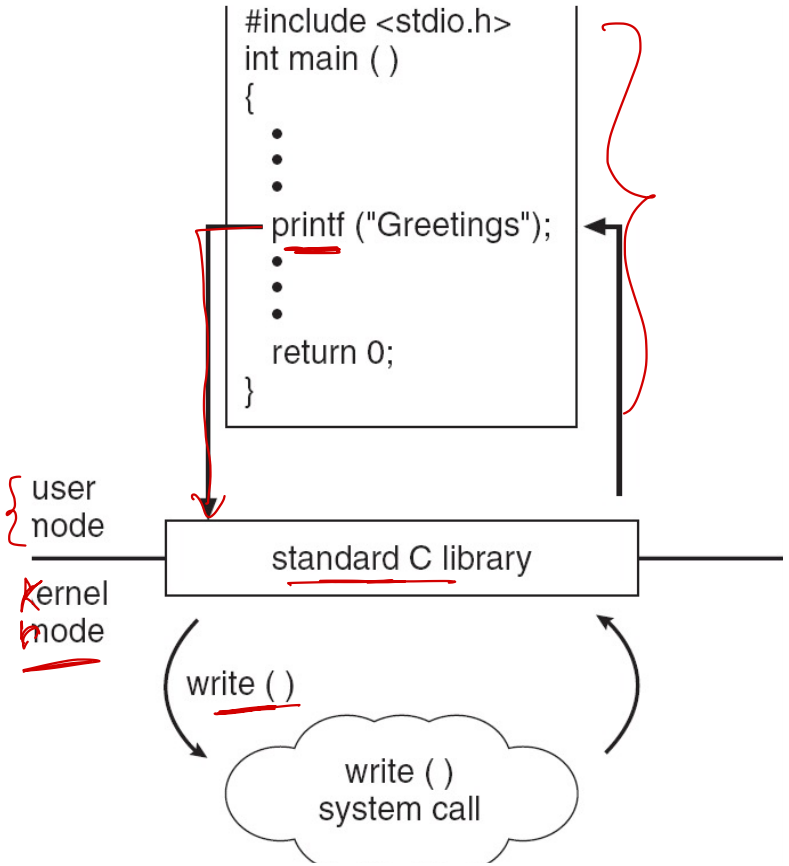
- **Device Driver:**It is a program/set of programs enable one or more hardware devices to communicate with the computer's operating system.

System Call and OS Relationship



C Example

```
#include <stdio.h>
int main ( )
{
   •
   •
   •
   printf ("Greetings");
   •
   •
   •
   return 0;
}
```



System Call Categories

Process control

(fork(),exit(),wait()

- load, execute, end, abort
- create process, terminate process
- get and set process
- allocate and free memory

File management

Read(),write()

- create file, delete file, open, close file
- read, write

Device management

Read(),write()

- request device, release device
- read, write
- get device attributes, set device attributes

Communications

Pipe(),shmget()

- send, receive messages
- transfer status information

Protection and Security

Chmod(),chown()

- Grant permissions
- Change ownership

Information maintenance

Getpid(),sleep()

- get time or date, set time or date
- get system data, set system data

System Calls

- File management
  - Open()→ called by fopen() in C
  - Close()→ called by fclose() in C
  - Read()→ called by fread(), fscanf(), fgets(), fgetc() in C
  - Write()→ called by fwrite(), fprintf(), fputs(), fputs() in C
  - Lseek()→ called by fseek(), ftell()
  - Chmod()→ to change the file permissions(rwx)
  - Chown()→ to change the file owner(rwx)

- Memory management
  - brk() → called by malloc() in C
  - nmap()

 # Dual Mode Operation:

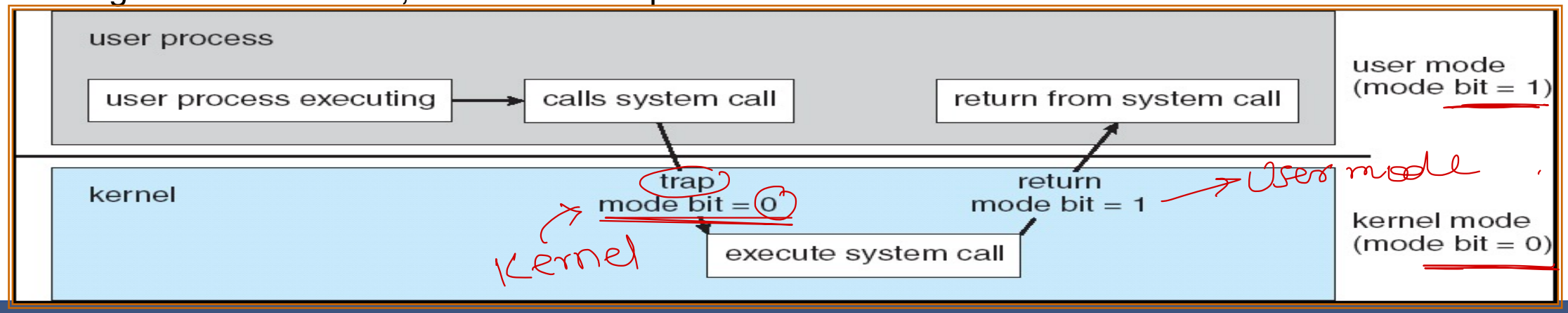
- System runs in two modes: System Mode and User Mode

1. System Mode:

- When the CPU executes system defined code instructions, system runs in a system mode.
- System mode is also reffered as kernel mode/monitor mode/supervisor mode/ privileged mode.

2. User Mode:

- When the CPU executes user defined code instructions, system runs in a user mode.
- User mode is also reffered as non-priviledged mode.
- Throughout execution, the CPU keeps switch between kernel mode and user mode



Dual Mode Operation:

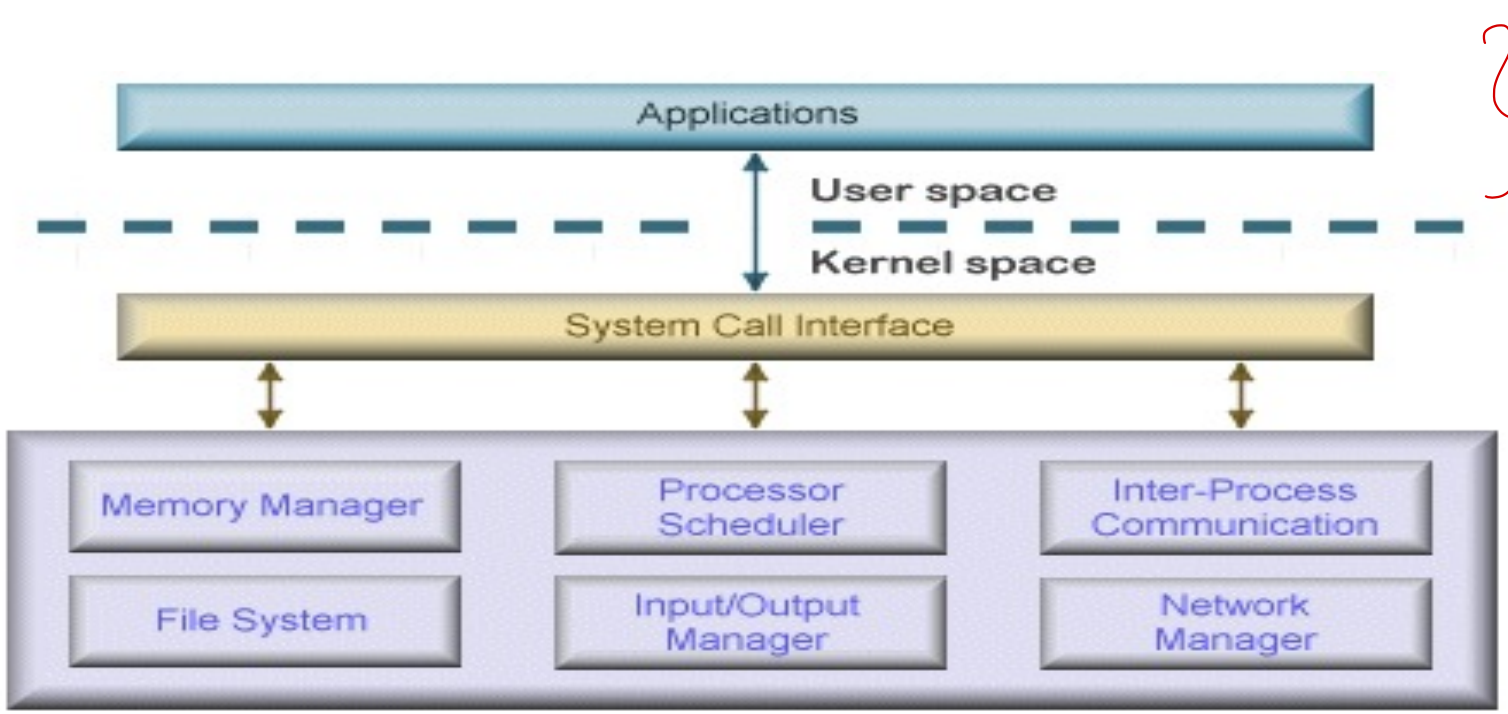
- Throughout an execution of any program, the CPU keeps switcesh in between kernel mode and user mode and hence system runs in two modes, it is reffered as **dual mode operation.**

- Todifferentiate between user mode and kernel mode one bit is there onto the CPU which is maintained by an OS, called as **mode bit**, by which the CPU identifies wheather currently executing instruction is of either system defined code instruction/s or user defined code instruction/s.
- In Kernel mode value of **mode bit = 0**, whereas
- In User mode **mode bit = 1.**

Kernel space vs User space

Part of the OS runs in the kernel model known as the OS kernel

Other parts of the OS run in the user mode, including service programs , user applications, etc.they run as processes they form the user space (or the user land)

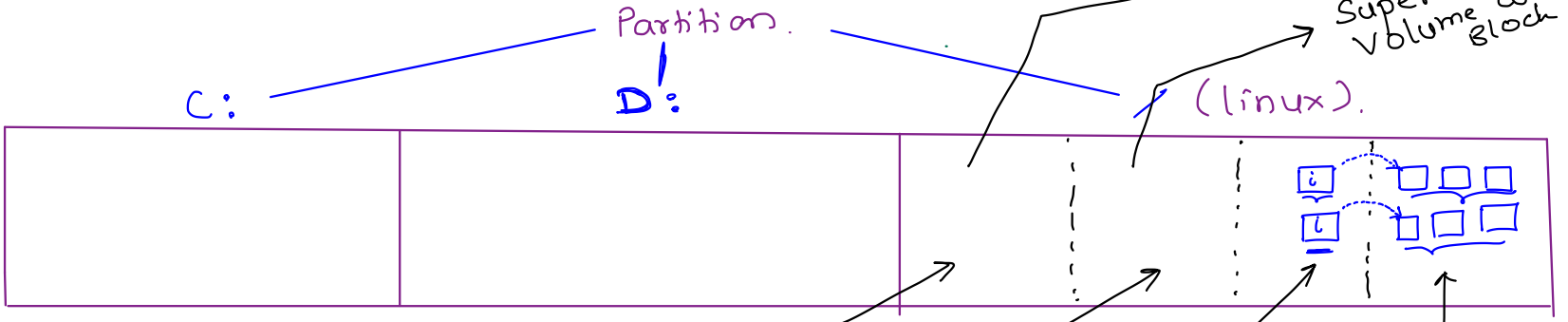


User prog → User space

OS

File = Collection of data & info on storage device.

File = Contents (data) + Information (metadata)



Booting info ✓

① bootstrap.
② bootloader

Partition Info ✓

① label.
② capacity
③ free block.

metadata/info. ✓

File Control Block.
FCB/i-node.

① type
② permission.
③ User
④ size
⑤ name.
⑥ timestamp.

data block ✓

2 or 4KB.
OS

File Organisation.

File

- file is a collection of logically related data or information
- file is a stream of bytes/bits
- file is a basic storage unit
  - File = data + metadata
  - Data : Actual File Contents
  - Metadata : Information about file.
- File Attributes:
  - Name, type, location, size etc..
- File Operations
  - Create, delete, write, read

- **"file system“ is a way to store data onto the disk in an organized manner so that it can accessed efficiently**

- e.g. Each OS has its own filesystem like, UNIX: UFS(UNIX Filesystem), Linux: Extended filesystem ext2, ext3, ext4, Windows: FAT, NTFS etc..., MAC OSX: HFS(Hierarchical Filesystem) etc...

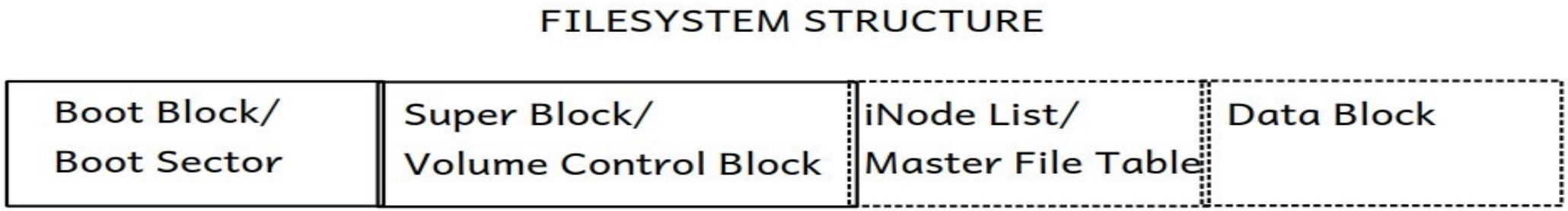
What is an inode / FCB

- An inode (index node) is a control structure that contains key information needed by the OS to access a particular file. Several file names may be associated with a single inode, but each file is controlled by exactly ONE inode.

- On the disk, there is an inode table that contains the inodes of all the files in the filesystem. When a file is opened, its inode is brought into main memory and stored in a memory-resident inode table.
- Information about the file can be kept in one structure referred as "FCB" i.e. File Control Block/iNode
  - inode/FCB contains info about the file like:
    - name of the file
    - type of the file
    - size of the file
    - parent folder location
    - access perms for user/owner, grp member and others etc

File System Structure

File system divides disk/partition logically into sectors/blocks, like **boot sector/boot block, volume control block/super block, master file table/iNode list block and data**



1. Boot Block: It contains information about booting the system like bootstrap program, bootloader etc...
2. Super Block: It contains information about remaining sections, like total no. of data blocks, no. of free data blocks, no. of allocated data blocks etc....
3. iNode List: It contains linked list of iNode's of all files exists on a disk.
4. Data Block: It contains actual data.

File Allocation on Disk

- Low level access methods for a file depend upon the disk allocation scheme used to store file data
  - Contiguous
  - Linked
  - Block or indexed

① Contiguous Allocation

HDD

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

	Start	Size
f1.txt	12	5
f2.txt	38	3
f3.txt	64	10

} Stored in inode or dir entry of the file.

adv.

→ Sequential.

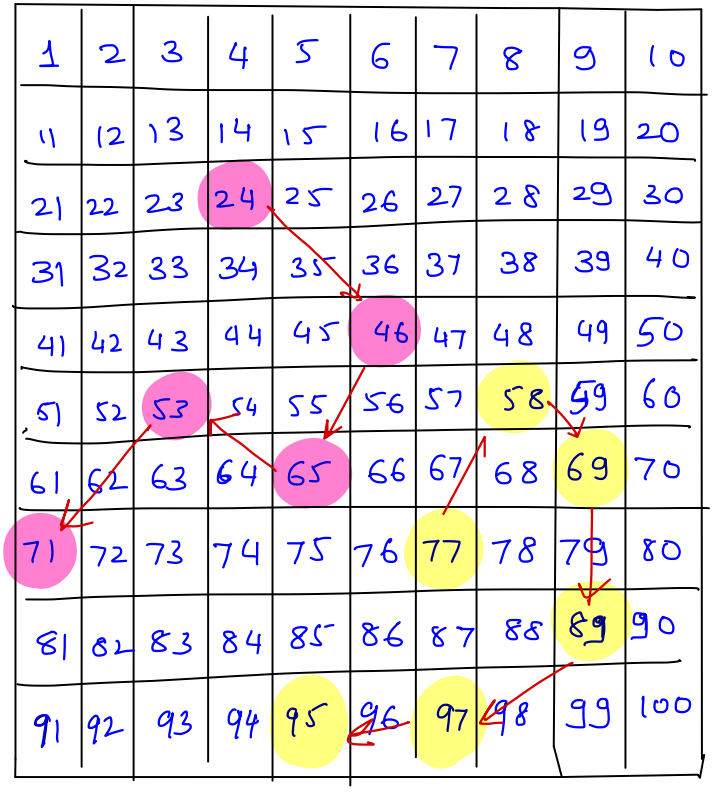
→ Random.

disadv.

→ file grow.

→ External frag.

② Linked Allocation.



	Start	end.
f1.txt	24	71
f2.txt	77	95

} Stored in inode or dir entry

→ Sequential.
→ file grow
→ External frag.

disadv
→ Random

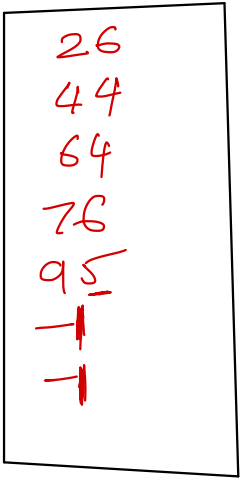
Fat

③ Indexed Allocation.

Index
48

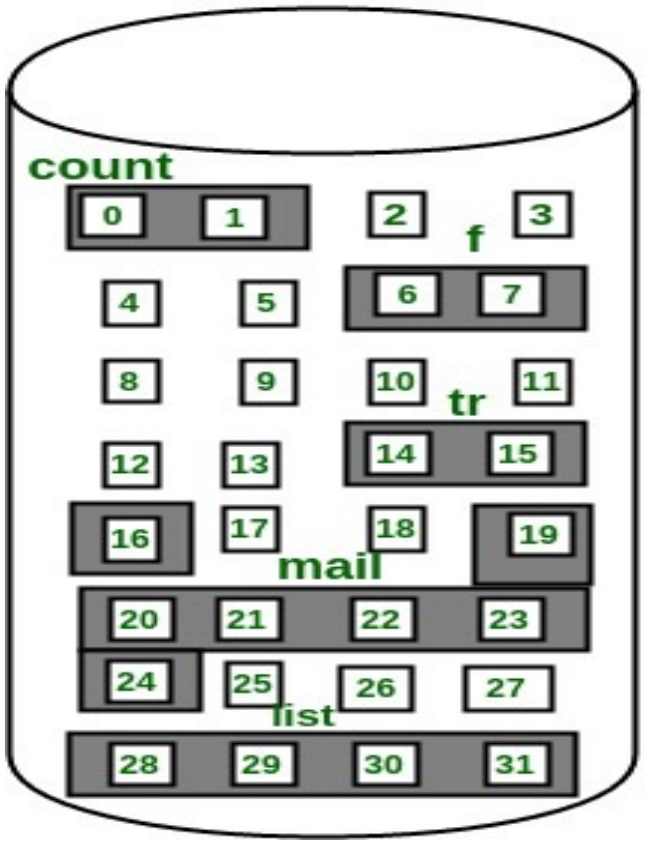
f1.txt → Stored in inode or directory entry.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100



Linux :- Ext2/3.

Contiguous Allocation



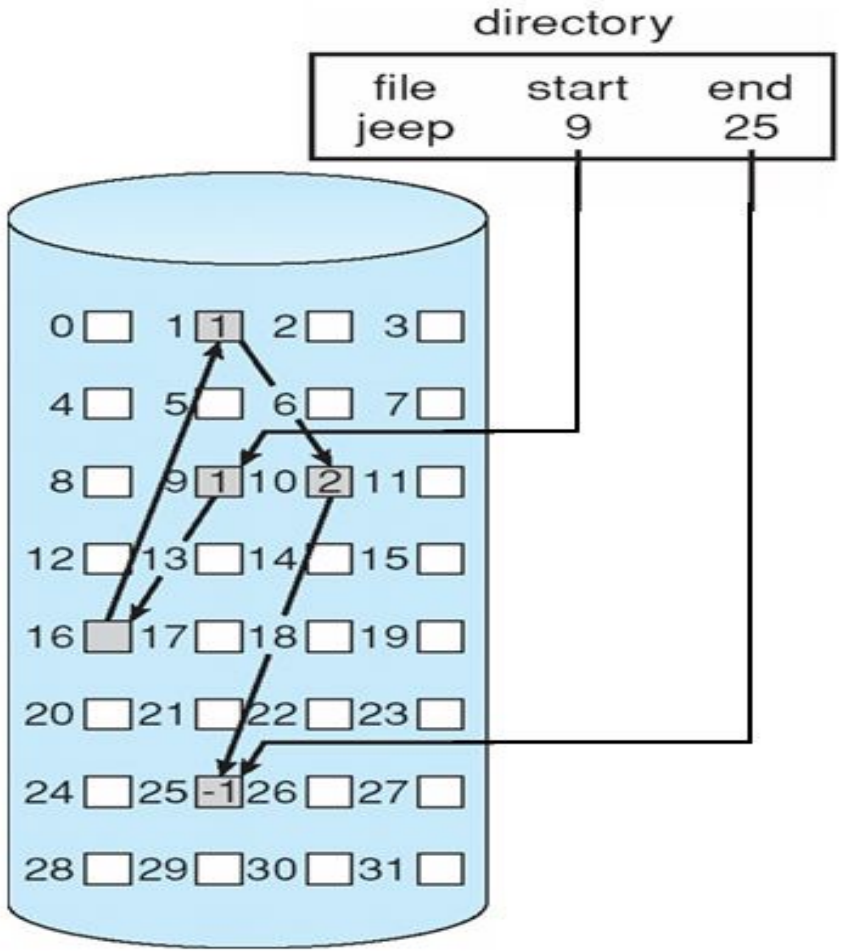
Directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

- File is allocated large contiguous chunks
- Expanding the file requires copying
- Dynamic storage allocation - first fit, best fit
- **External fragmentation occurs on disk**

Linked Allocation

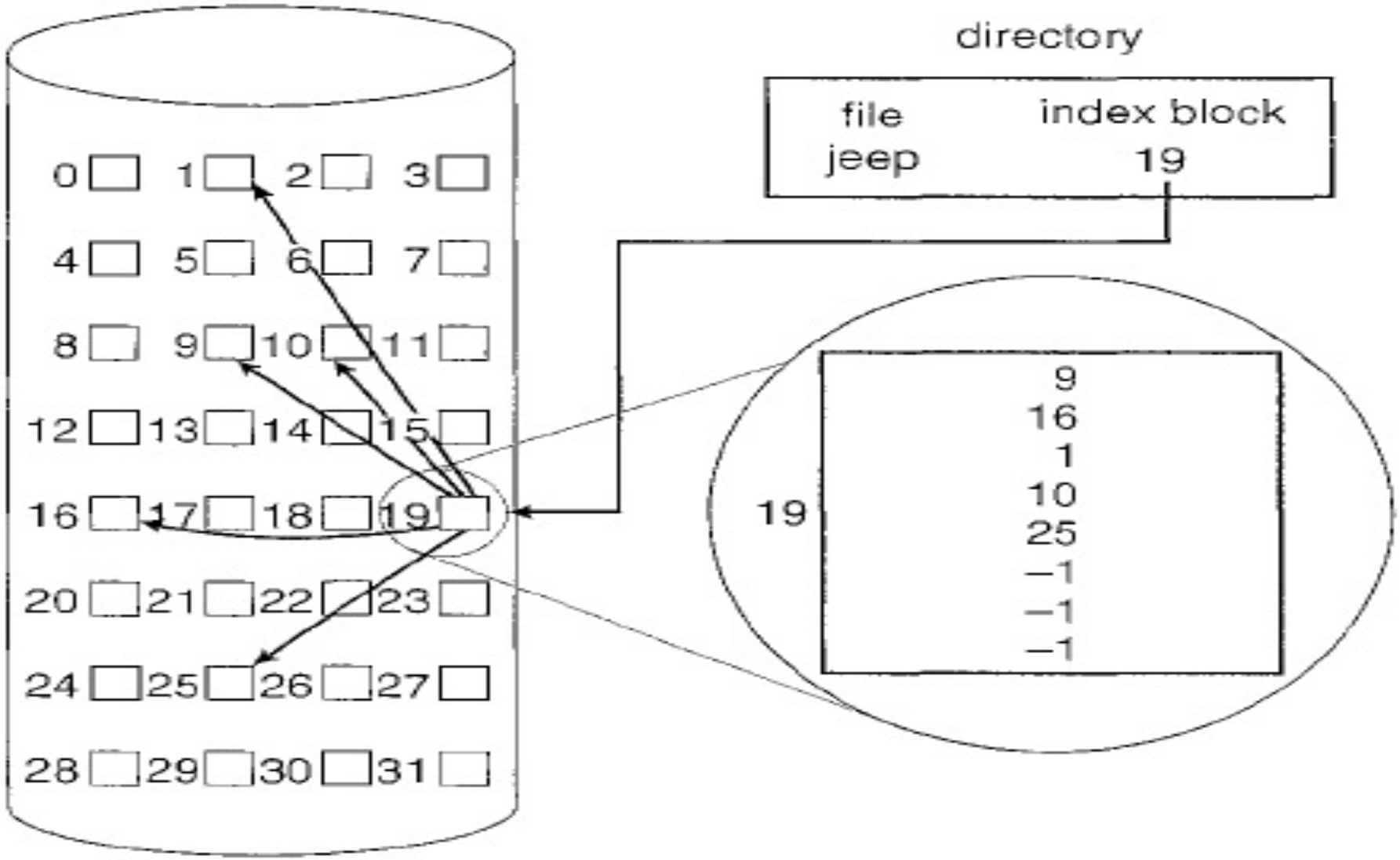
- Each file is a linked list of disk blocks, which may be scattered on the disk
- Directory contains a pointer to the first and last blocks, and each block contains a pointer to the next block
- **Advantages**:
  - No external fragmentation
  - Easy to expand the size of a file
- **Disadvantages**:
  - Not suitable for random access within a file
  - Pointers take up some disk space
  - Difficult to recover a file if a pointer is lost or damaged
- Blocks may be collected into **clusters** of several blocks
  - Fewer pointers are necessary
  - Fewer disk seeks to read an entire file
  - Greater internal fragmentation



Block / Indexed

- A special block known as the **Index block** contains the pointers to all the blocks occupied by a file.
- The ith entry in the index block contains the disk address of the ith file block.
- The directory entry contains the address of the index block.
- When the file is created, all pointers in the index block are set to *nil.*
- *When* the i^{th} block is first written, a block is obtained from the free-space manager and its address is put in the i^{th} index-block entry.

Indexed Allocation



Disk Scheduling

- **Disk scheduling** is done by operating systems to schedule I/O requests arriving for the disk.
- Disk scheduling is important because:
  - Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by the disk controller. Thus other I/O requests need to wait in the waiting queue and need to be scheduled.
  - Two or more request may be far from each other so can result in greater disk arm movement.
  - Hard drives are one of the slowest parts of the computer system and thus need to be accessed in an efficient manner.

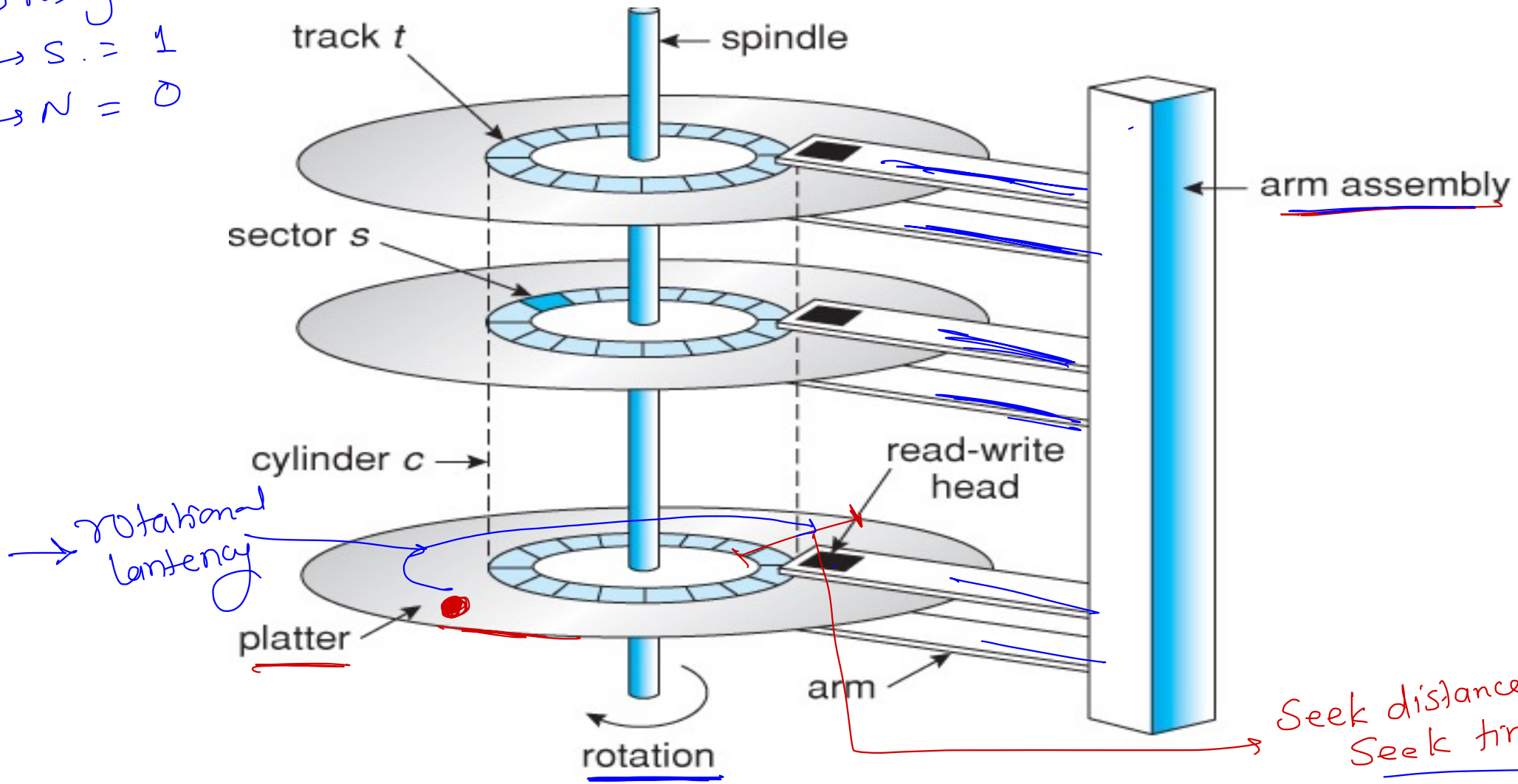
Hard Disk

disk access ⇒ rotational lantency + seek time.

HDD → magnetic Pole

N → S = 1

S → N = 0



Disk Scheduling Criteria

Seek Time

- Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write.
- minimum average seek time is better.

Rotational Latency

- The time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads.
- minimum rotational latency is better.

Transfer Time

- The time to transfer the data.
- It depends on the rotating speed of the disk and number of bytes to be transferred.

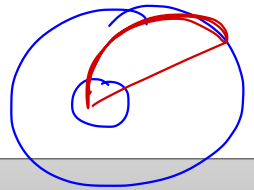
Disk Access Time

- **SeekTime+Rotational Latency +Transfer Time**

Disk Response Time

- The average of time spent by a request waiting to perform its I/O operation.
- minimum variance response time is better.

Disk Scheduling Algorithms



First Come First Serve

- FCFS, the requests are addressed in the order they arrive in the disk queue.

Shortest Seek Time First

- SSTF (Shortest Seek Time First), requests having shortest seek time are executed first.
- it decreases the average response time and increases the throughput of system.

Scan/Elevator

- SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path.

CSCAN

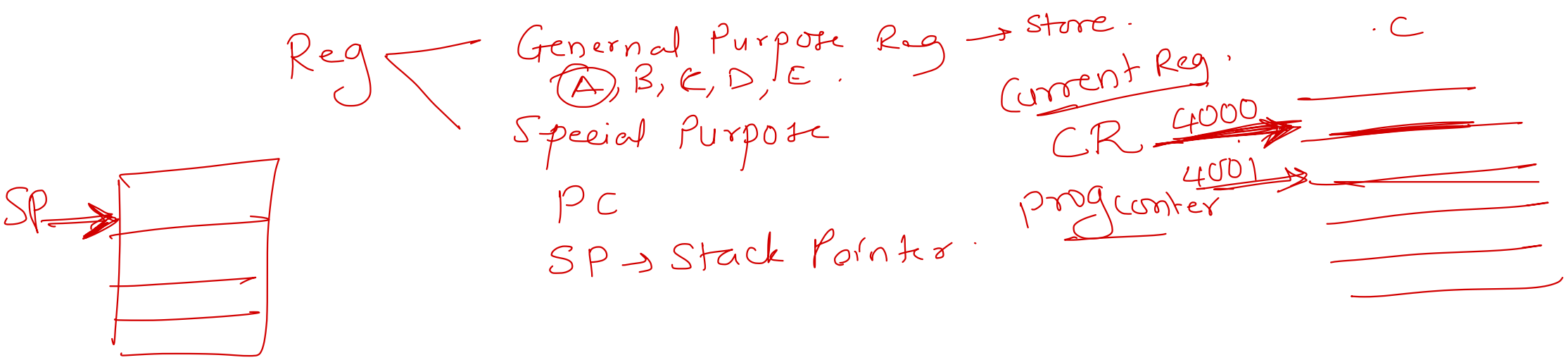
- disk arm moves in a circular fashion and this algorithm is also similar to SCAN algorithm and hence it is known as C-SCAN (Circular SCAN).

Look

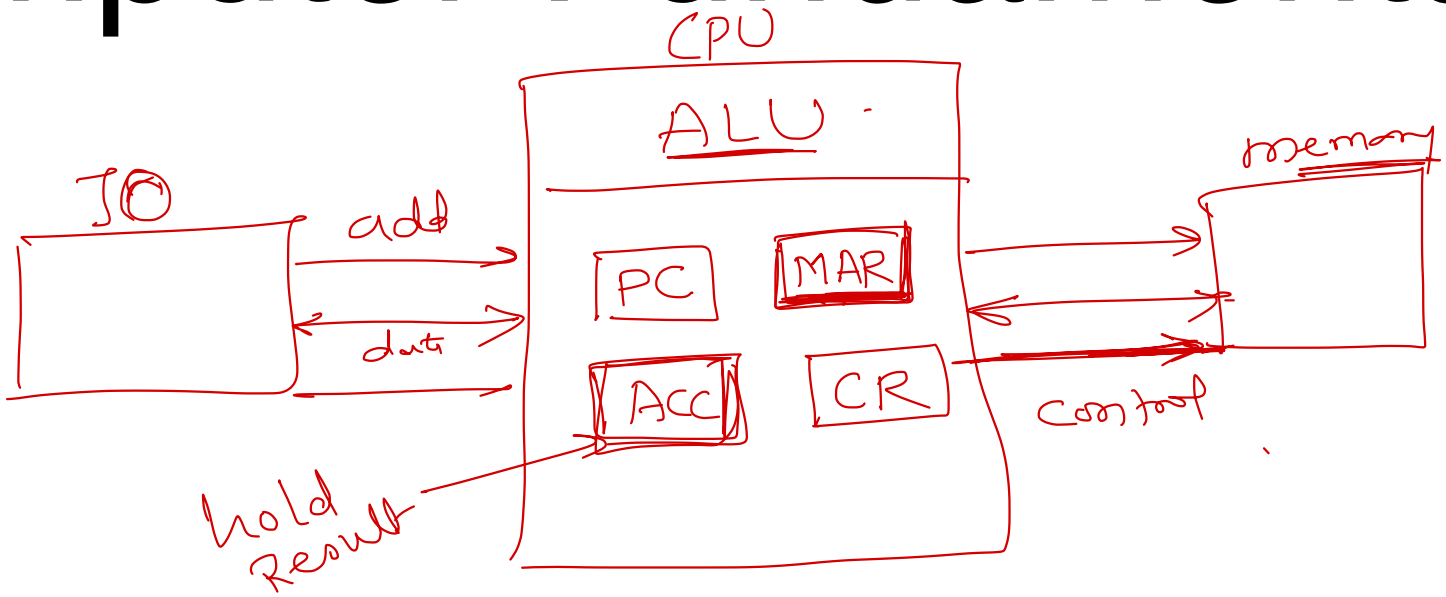
- similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only.

Clook

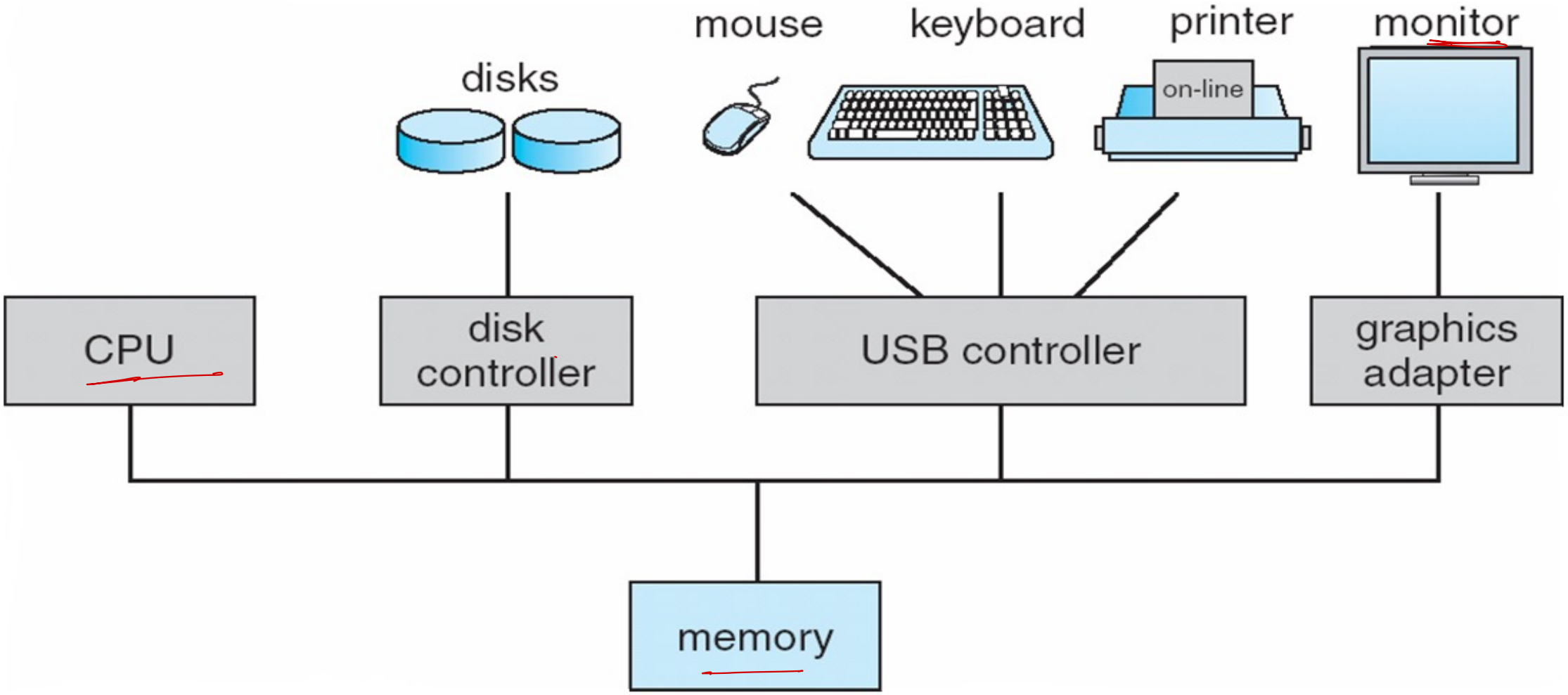
- CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request.



Computer Fundamentals



A Computer System



Computer Fundaments

Computer Structure

* CPU: Central Processing Unit
* RAM: Main memory
* Disk: Secondary storage
* IO Devices: Keyboard, Monitor, ...
* Bus: Set of wires connecting CPU to other peripherals
    * Address bus
    * Data bus
    * Control bus

Computer Fundaments

IO Device

* The Input unit allows programs and data to be entered into the computer.
    * e.g. Keyboard (primary), Mouse, Joystick, Touchpad, Touch pen, Scanner, Microphone, Webcam, Punch card, Bar code scanner, MICR scanner, Fingerprint, ...
* The Output unit allows the results of processing to be exported to the outside world or other devices or saved to be used later.
    - e.g. Monitor (primary), printer, plotter, Speakers, projector, ...

Computer System Components

1. **Hardware** – provides basic computing resources (CPU, Memory, I/O devices, Communication).
2. **Operating System** – controls and coordinates use of the hardware among various application programs for various users.
3. **System & Application Programs** – ways in which the system resources are used to solve computing problems of the users (Word processors, Compilers, Web browsers, Database systems, Video games).
4. **Users** – (People, Machines, other computers).

What happens when we start a computer?? (Booting Process)

- Hardware doesn't know where the operating system resides and how to load it.
- **Bootstrap Program :**
  - Initial program to run a system
  - Locating and Loading OS Kernel in main memory
- Where it is stored ??? ROM

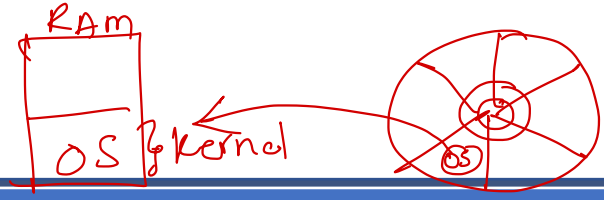
•If any storage device/partition contains one special program called as **"bootstrap program"** in its first sector i.e. in a boot sector then such a device/partition is referred as **bootable device/partition.**

•e.g. hard disk drive, pen drive, CD/DVD

Steps of Booting

1. Machine Boot

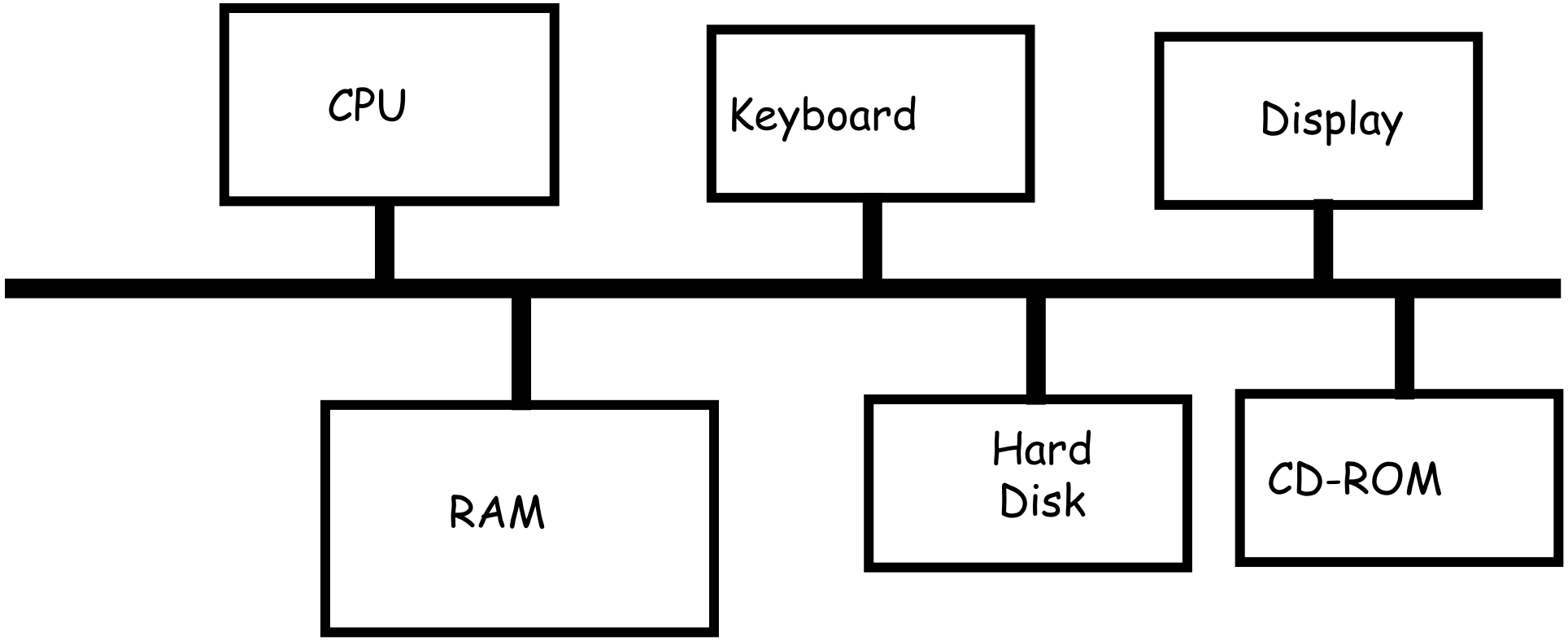
- When we switch on the power current gets passed to the motherboard and one program gets invoked named as **"BIOS"** which is exists in the ROM memory on motherboard.
- BIOS -- Basic Input Output System -- which is **a micro-program.**
- A micro-program is a program which is smaller in size and can be stored into the memory with its all possible set of input values.
- first step of BIOS is **"POST" - Power On Self Test**, under POST BIOS checks whether all peripherals are connected properly or not and their working status.
- **"peripherals or peripheral devices" --** devices which are connected to the motherboard externally are called as peripherals.
- after POST BIOS executes **"bootstrap loader",** bootstrap loader searches for available bootable devices and selects any one out of it as per the defined priorities.



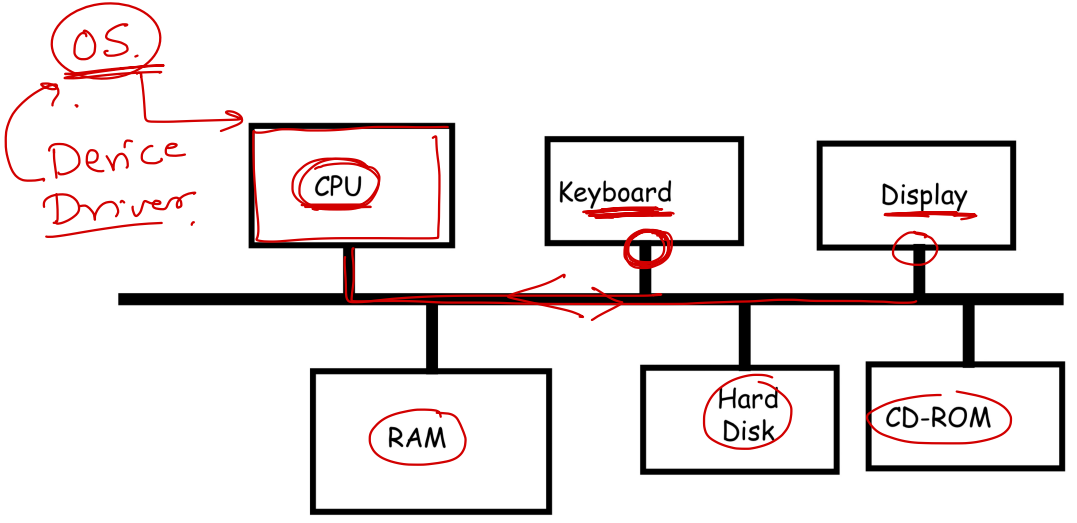
2.System boot:

- if hard disk drive got selected as a bootable device and if it contains multiple OS's have installed on it, then "**bootloader**" program gets executes.
- **Boot loader program** displays list of operating system installed onto the machine, so that user can select any one at a time from and it invokes bootstrap program of selected operating system.
- Bootstrap program locates the kernel and load it into the main memory.

Computer Fundamentals



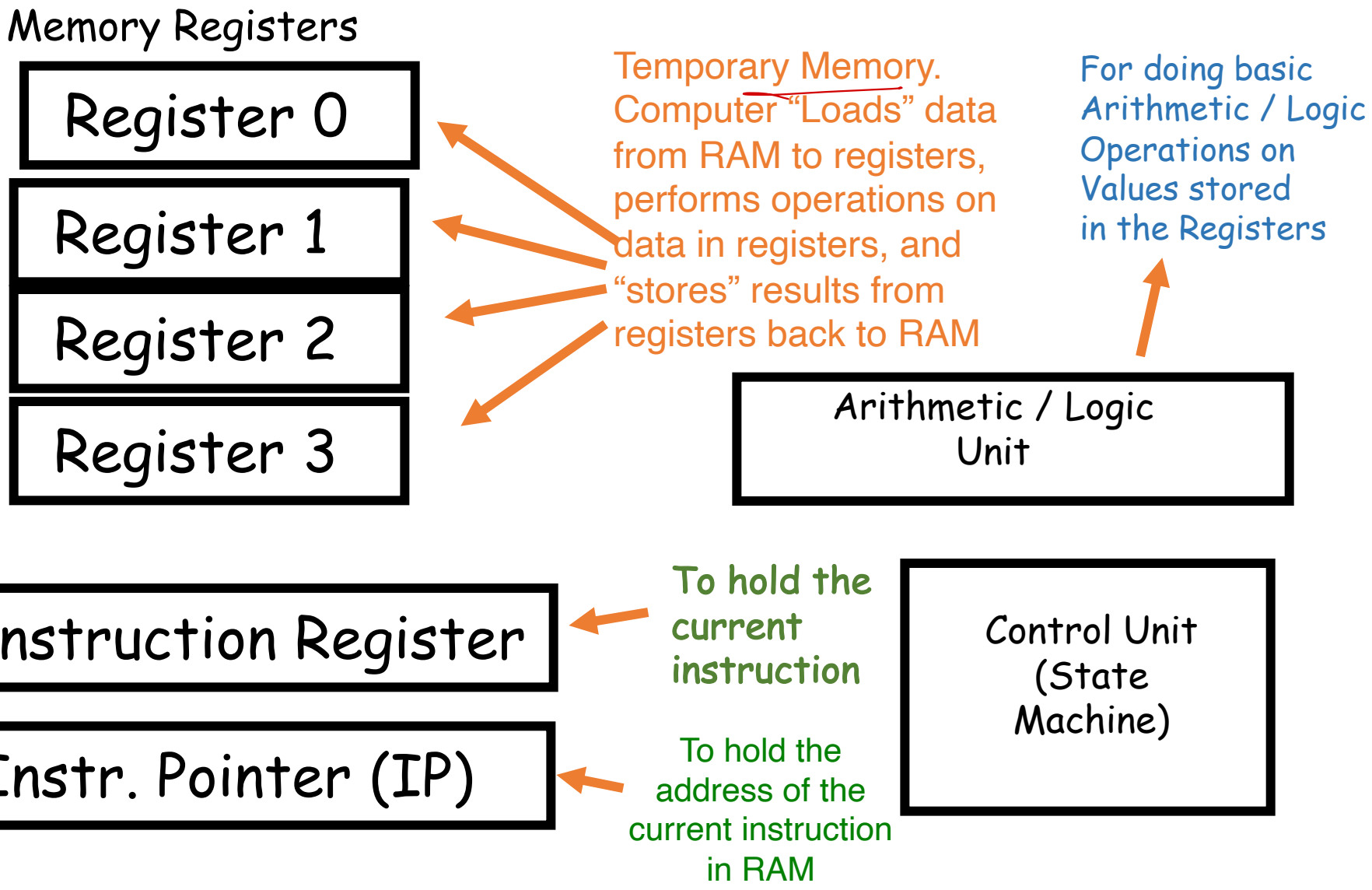
It is a system concept integrating software and hardware to specify the design of computing systems



Device Driver :- is a prog within OS (part of OS) that send or receive data/cmd to/from IO device controller and also handles interrupts send from the device

Each IO device has its own internal dedicated processing unit is called as IO controller / IO module.

CPU


Memory Registers
Register 0
Register 1
Register 2
Register 3
Temporary Memory.
Computer "Loads" data from RAM to registers, performs operations on data in registers, and "stores" results from registers back to RAM
For doing basic Arithmetic / Logic Operations on Values stored in the Registers
Arithmetic / Logic Unit
Instruction Register
To hold the current instruction
Instr. Pointer (IP)
To hold the address of the current instruction in RAM
Control Unit (State Machine)

Bus, CU, ALU, Memory

Bus

- It is a simplified way for many devices to communicate to each other.
- It is internal arrangement of computer system which includes design of the processor , memory and input/output units.

Control Unit

- Control is responsible for determining what action is to be performed on what data.
- controls all operations and it controls devices which are connected to the computer system by coordinating with device controllers.
- Fetch-Decode-Execute

ALU (Arithmetic Logic Unit)

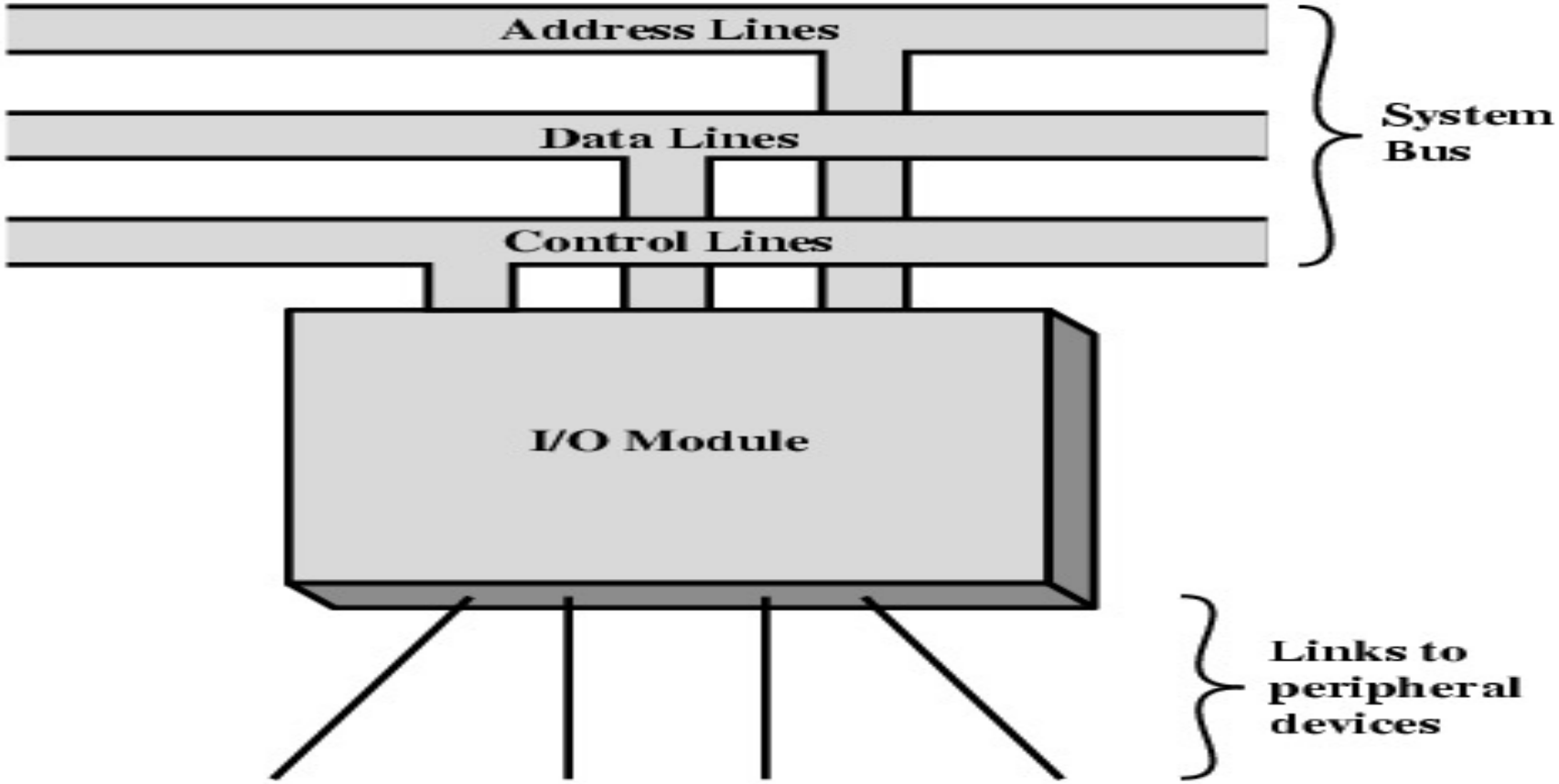
- ALU is mainly comprised of **logic gates**, circuits made from transistors that take inputs.
- ALU performs all arithmetic and logical operations.

Memory

- Memory consists of circuits whose primary purpose is to **hold information**, but only temporarily.
- When you talk about the memory of a computer, most often you're talking about its RAM.

Input/Output

- The Input unit allows programs and data to be entered into the computer.
- The Output unit allows the results of processing to be exported to the outside world or other devices or saved to be used later.



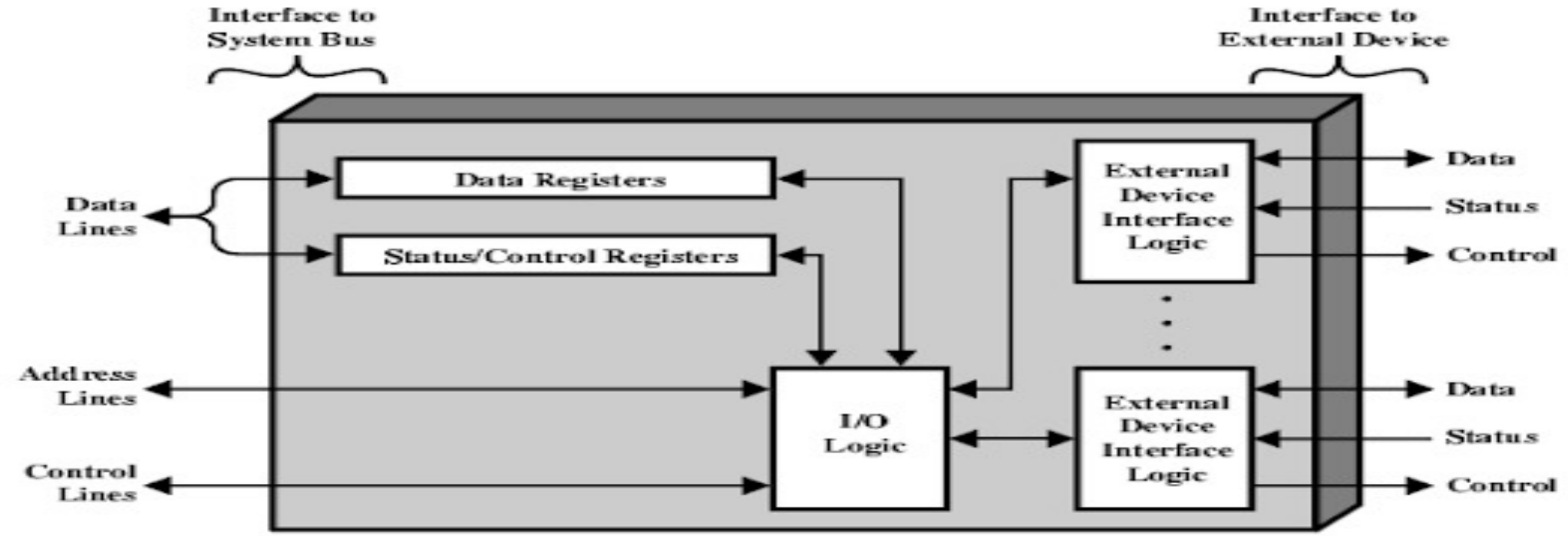


Figure 7.4 Block Diagram of an I/O Module

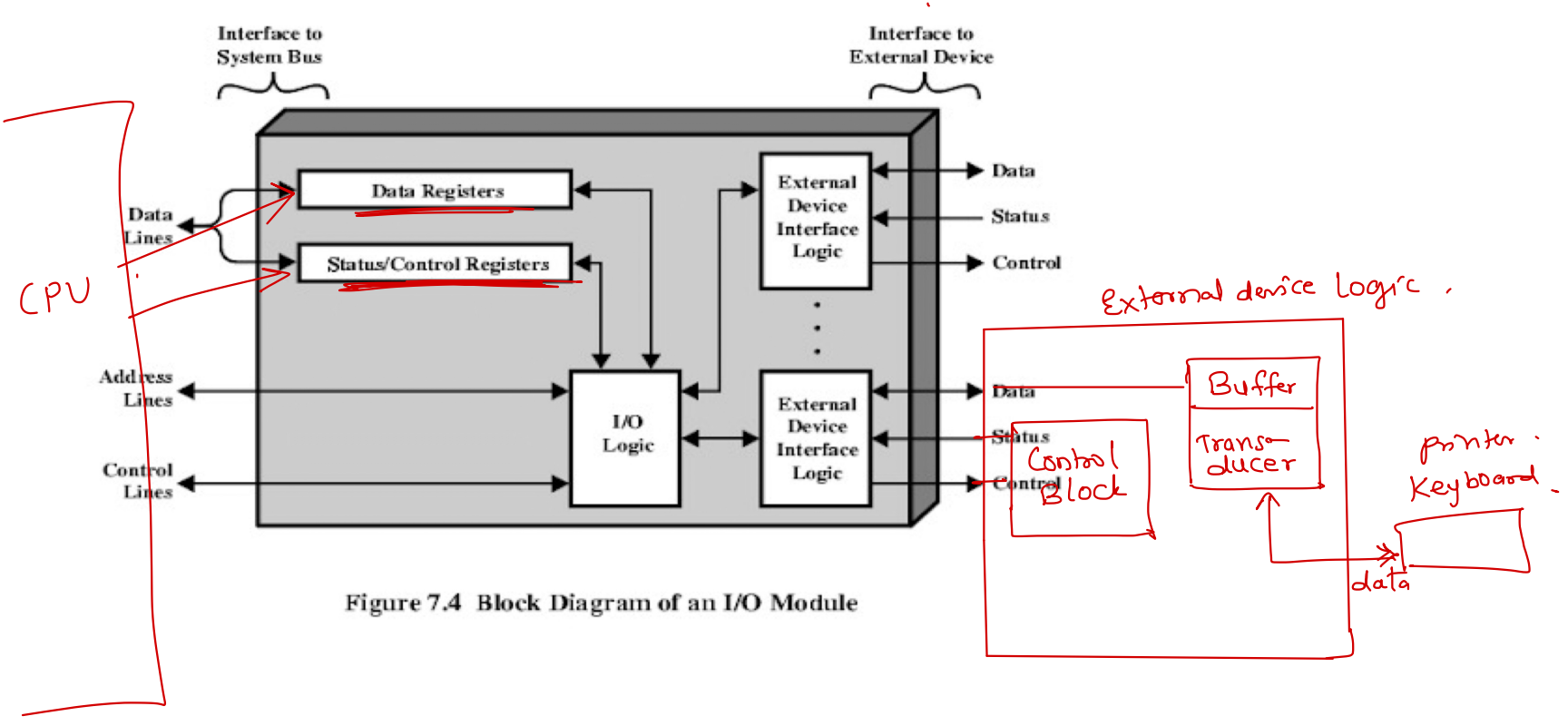


Figure 7.4 Block Diagram of an I/O Module

IO Module/Controller Functions

* Control & Timing
* CPU Communication
* Device Communication
* Data Buffering
* Error Detection

IO Steps

* CPU checks I/O module device status
* I/O module returns status
* If ready, CPU requests data transfer
* I/O module gets data from device
* I/O module transfers data to CPU

Device Interface Components

* The control logic is the I/O module's interface to the device
* The data channel passes the collected data from or the data to be output to the device. On the opposite end is the I/O module, but eventually it is the processor.
* The transducer acts as a converter between the digital data of the I/O module and the signals of the outside world.
* Keyboard converts the motion of the key into data representing the key pressed or released
- Temperature sensor converts the amount of heat into a digital value
* Disk drive converts data to electronic signals for controlling the read/write head

Computer Fundaments

IO Techniques

* Communication between memory and IO devices.

- IO Techniques
  * Programmed IO
  * Interrupt-driven IO
  * Direct Memory Access

Programmed IO

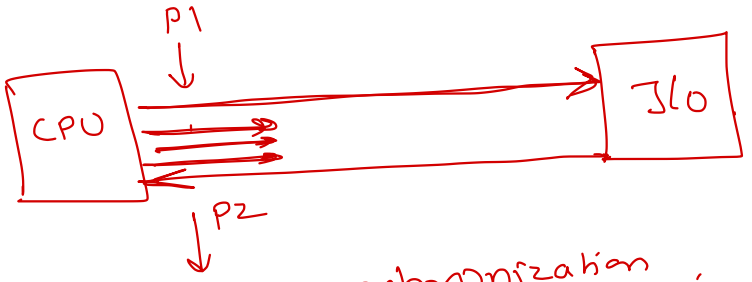
* CPU waits for IO operations to be completed. This is also called "Polling".

- As CPU is faster, so CPU time is wasted.
- When OS/program waits for the IO to be completed, it is called a "Synchronous IO".

Interrupt driven IO

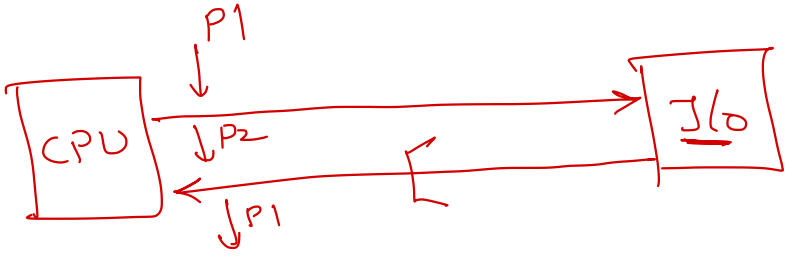
- CPU issues a command, and proceeds for its work until interrupted by IO device.
- Better utilization of CPU time.
- Since OS does not wait for the IO completion, it is called an "Asynchronous IO".

Programmed I/O



Asynchronization.

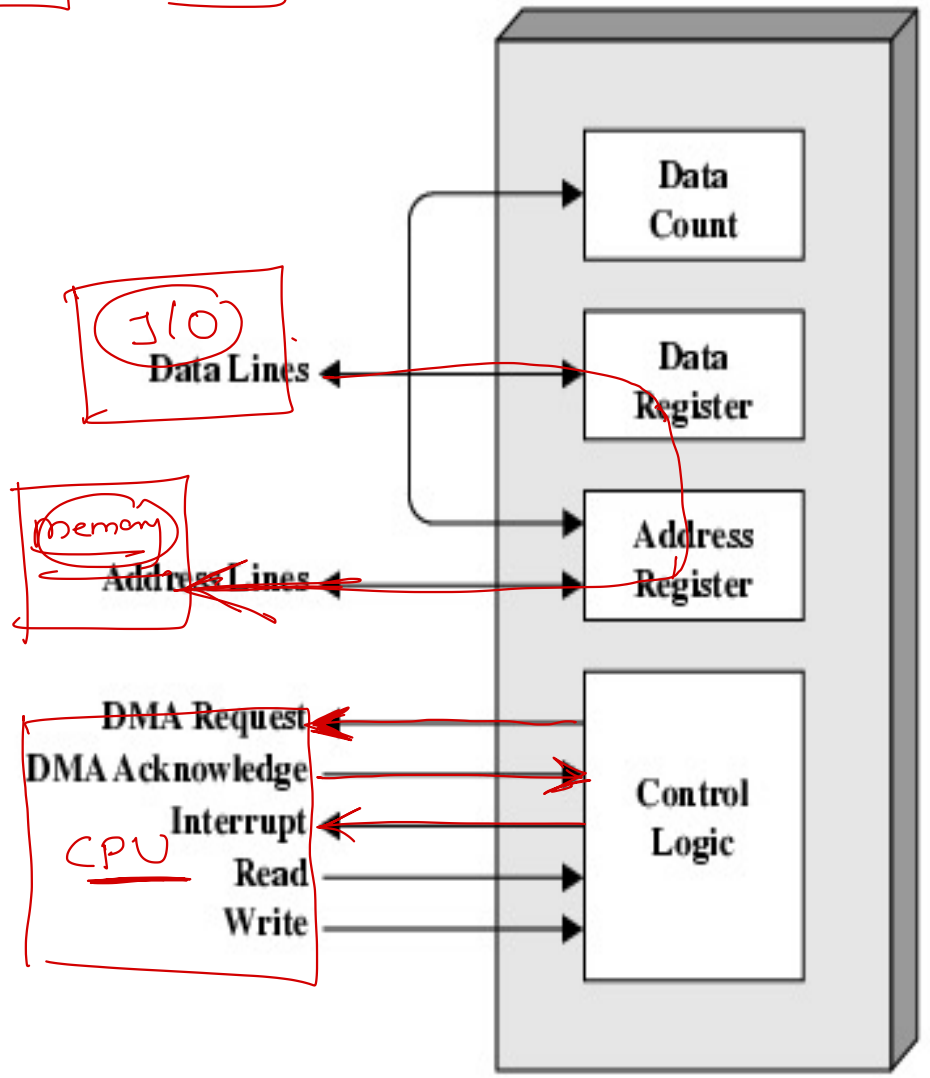
Interrupt I/O



Direct Memory Access

- Interrupt driven and programmed I/O require active CPU intervention
  - Transfer rate is limited
  - CPU is tied up
- DMA Operations:
  - When the processor wishes read or send a block of data, it issues a command to the DMA module by sending some information to DMA module.
  - The information includes:
    - read or write command, sending **through read and write control lines**.
    - number of words to be read or written, communicated on the **data lines** and stored in the **data count register.**
    - starting location in memory to read from or write to, communicated on data lines and stored in the **address register**.
    - **address of the I/O device** involved, communicated on the **data lines**.

  When the transfer is complete, the DMA module sends an interrupt signal to the processor to inform that it has finish using the system bus



Thank You