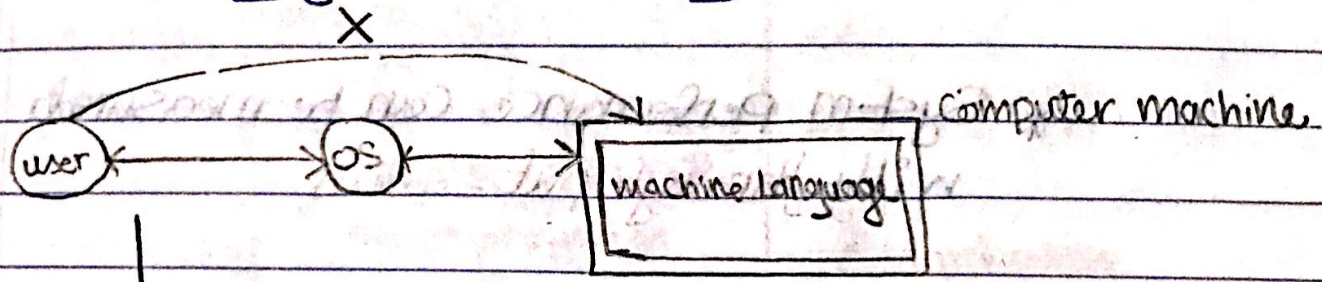
Chapter#1
Introduction

lecture#1
Feb.10.2018

Operating Systems Goals:

(1) overall goal: Execute user programs.
(2) primary goal: Conveniency



↳ it's easier for the user to interact with the OS.

(3) Secondary goal: efficiency.

Computer Resources:

(1) CPU.
(2) Memory.
(3) I/O devices.

} OS manages Resourses.

Operating System (OS): A Set of Algorithms that run the Computer machine.

→ OS manages the computer resources.
→ OS must manage the computer Resources efficiently.

Other goals of OS:

استغلال Utilization of Computer resources.

- Utilization of CPU: make the CPU as busy as possible.
- Utilization of Memory:- to benefit or use memory as much as possible.
- Utilization of I/O devices.

* System performance can be measured with throughput. إنتاجية

* Throughput: number of jobs (programs) that finish execution per unit of time.

Computer

1 HARDWARE:-

- physical devices: chips, wires, Power supplies...
- micro program: A primitive software that communicates with physical devices which is an Interpretter that fetches

→ Fetch: execute machine language instruction.

— machine Language (Assembly Language)

2 OPERATING SYSTEM:

u_1	u_2	----	u_n	Software
Application Packages				Software
Compiler	DB	---etc		Software
O ⇅ S				
Machine Language				Hardware
Microprogram				Hardware
physical devices				Hardware

3 APPLICATION PACKAGES:

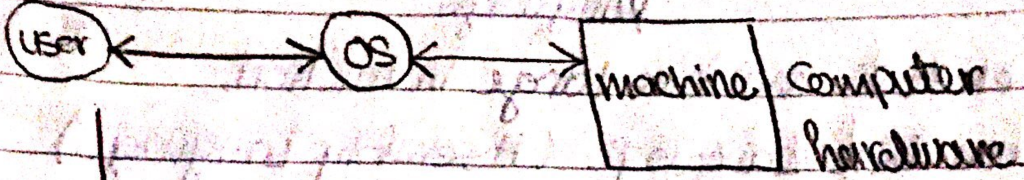
4 User Programs.

Operating System Views: OS Goals

(1) It's a control Program: It controls the execution of all programs.
↳ overall goal.

(2) Extended machine: Extension of the physical machine.
↳ pimary goal.

That is it hides all the Complexity of system programming and provide the user with a simple Clean machine to deal with

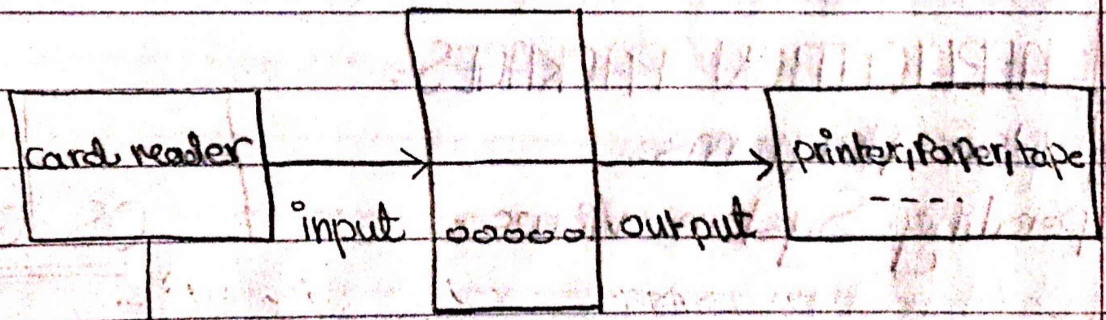


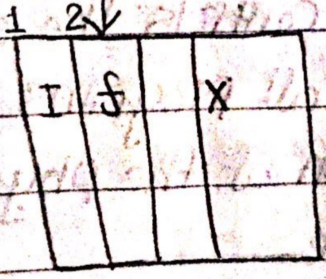
→ The user doesn't have to deal with machine or use assembly / machine Language.

(3) Resource Manager: It manages efficiently the computer resources.
↳ Secondary goal.

(4) Kernel: part of the OS that's always running and executing instructions

History & Evolution of OS :-





→ each line needs a card
i.e if a program has 200 lines the user needs 200 cards.

* Hexadecimal was used.

[4] Early Software:
- machine Language.
- Assembly Language. (Assemblers)
- Loaders
- Linkers: the addition of software to program.
- Compilors.

* Performance is poor
  - a great deal of time is wasted in set up time.
  - No overlap between I/O & CPU execution
  - Low CPU utilization (due the big difference between I/O speed & CPU speed)

  Example:

  A fast card reader can read 1200 cards/min
  1200/60 = 20 card/sec.
  The CPU can process 300 cards/sec.

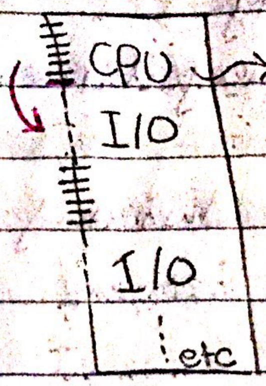
Card reader	CPU	Card reader	CPU
60 Sec	4 Sec	60 Sec	4 Sec

  ∴ percentage of CPU utilization
  $= 4/64 \simeq 1/16 \simeq 6\%$

Lecture #2

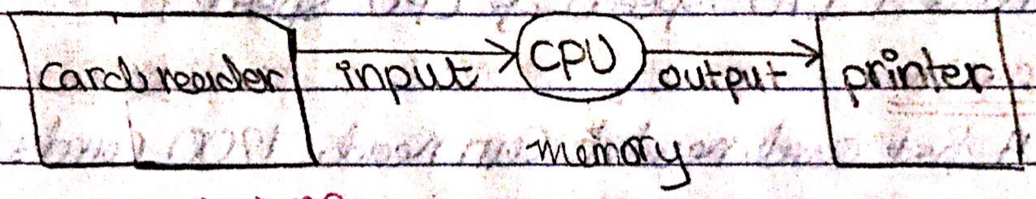
Feb. 12. 2018
Monday

⚠ CPU instruction executes until reaching I/O
↳ printing or reading input.

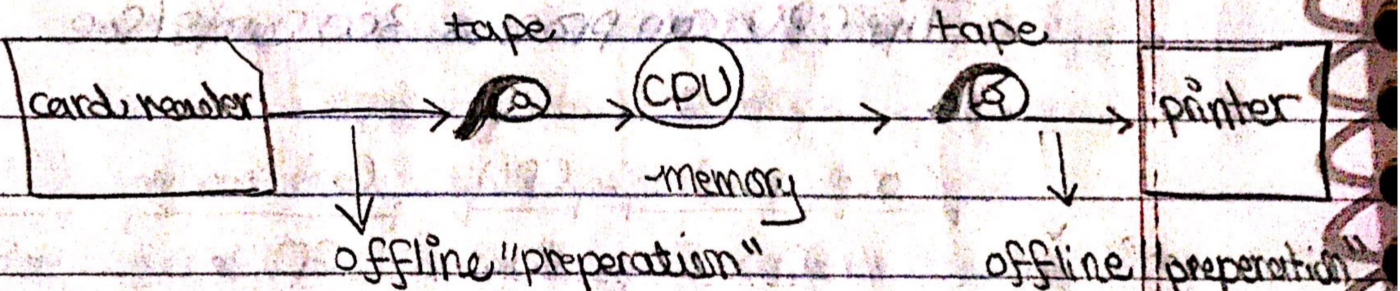


[A] offline Operation:

(1) Before

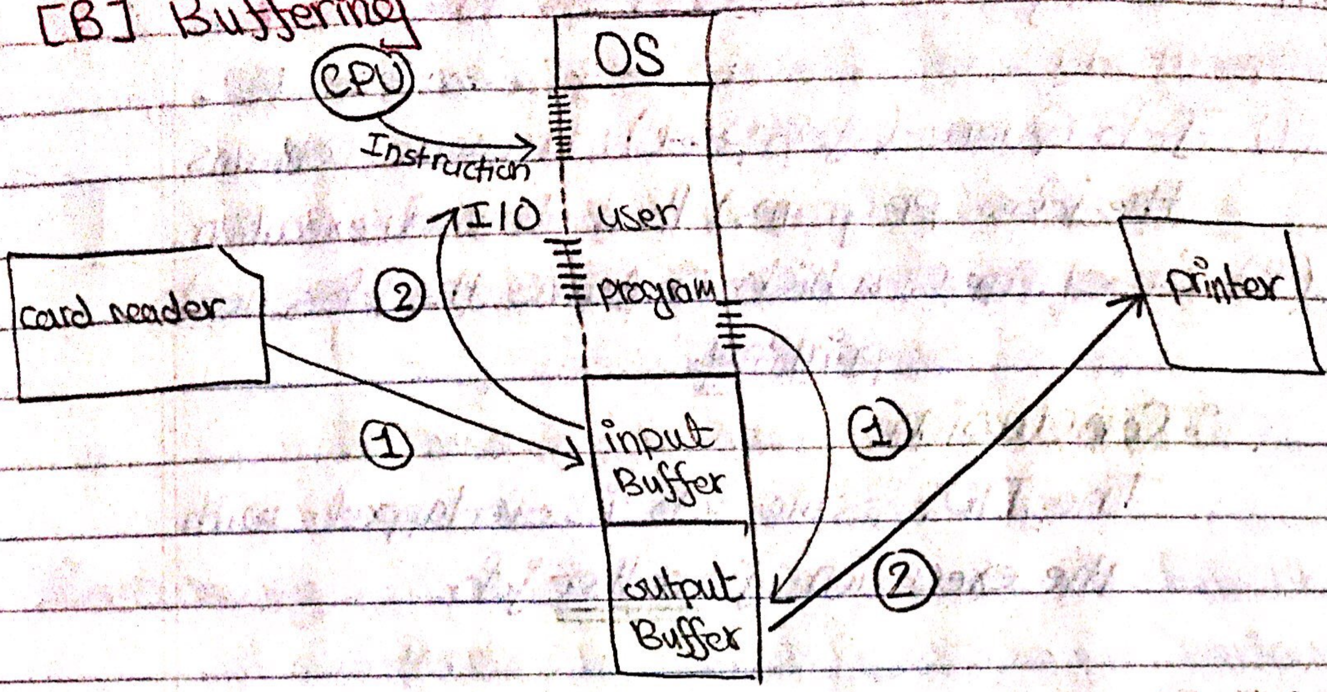


(2) After



⚠ Tape to memory is much more faster than Card reader to memory.
↳ improve execution.

[B] Buffering

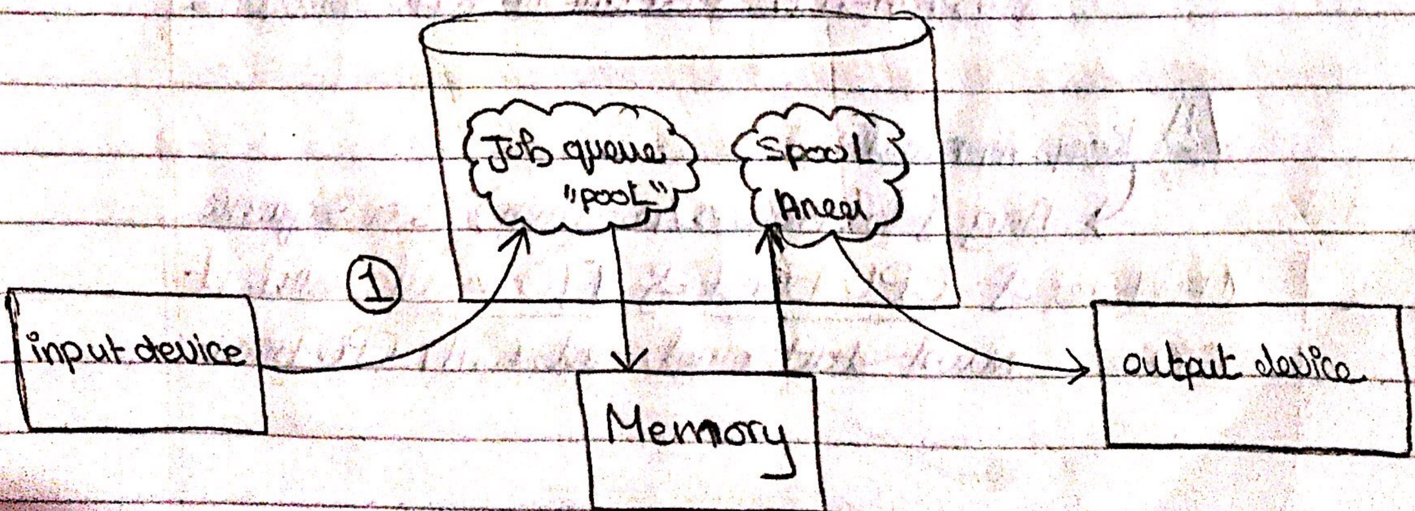


* when CPU reaches I/O it brings data from buffer not from the card reader

∴ Conclusion:-

the I/O of one job is overlapped with the execution of same job.

[C] Spooling



In Spooling 2 kinds of data structures were introduced:

(1) Job Queue (Job Pool): A queue contains the jobs (programs) that demand execution.

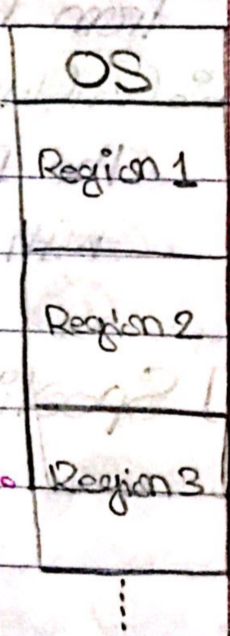
(2) Spool Areas which contains the jobs need printing.

∴ Conclusion:

The I/O of one job is overlapped with the execution of another job.

[D] Multiprogramming Batch systems

"Multi-programming"

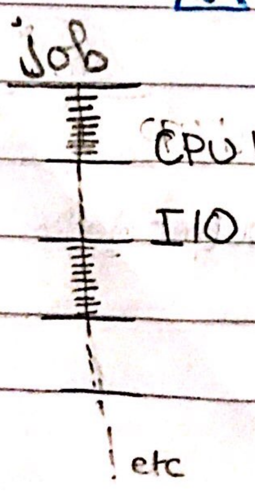


- Memory is divided into several Regions (Partitions).
- Regions sizes normally different.
- Each region contains only one job.

* The CPU switches to another job when the first one needs I/O.

⚠ Keep in mind:

↳ Any job (process/program) is a sequence of CPU burst & I/O wait and it must start and end with CPU burst.



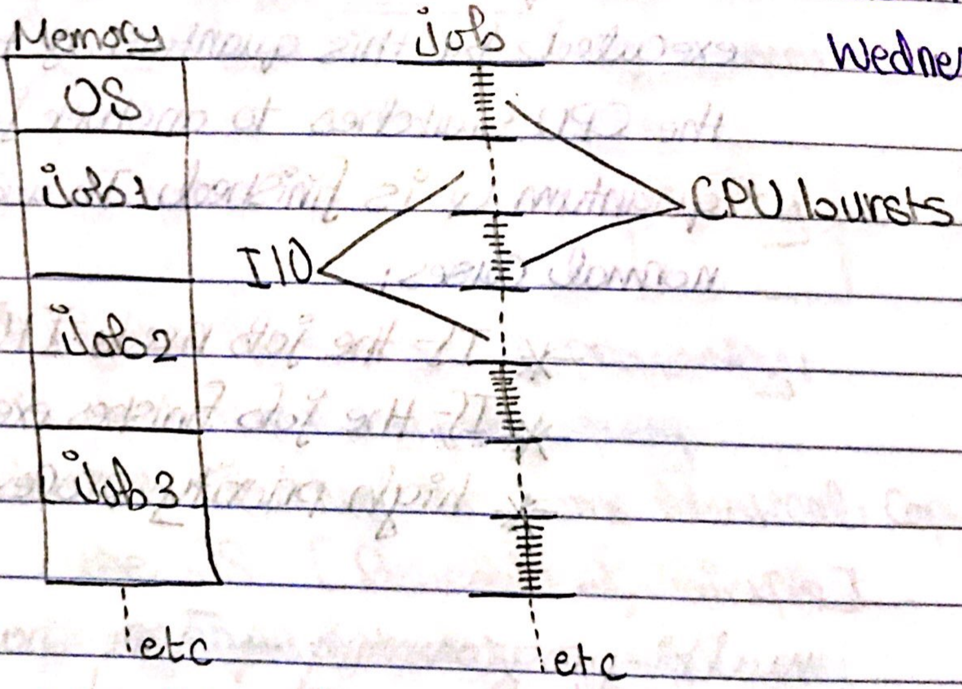
Two kinds of jobs:

(1) CPU-bound job: Contains few very long CPU bursts.
→ most of the time, job needs CPU.

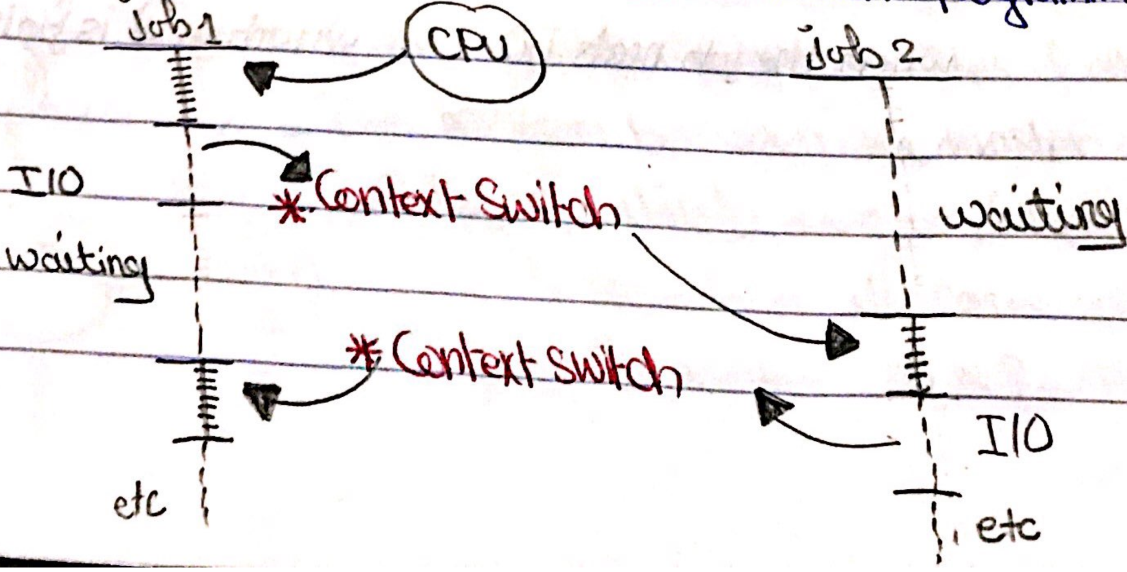
(2) I/O-bound job: Contains many very short CPU bursts.
→ most of the time, the job needs I/O.

lecture #3

Feb. 14. 2018
Wednesday



⚠ CPU switches (changes) to another job when the first one needs I/O "multi-programming"



* Context switch: Saves Register for job 1
Reloads Register for job 2.

[E] Time Sharing Systems:

- Same as multi-programming,
- Memory is divided into regions.
- Several jobs are kept in memory.
- Each job is assigned a slice of time called quantum Q. Each job is executed for this quantum of time & the CPU switches to another job when quantum Q is finished. In addition to normal cases:
  * If the job needs I/O.
  * If the job finishes execution.
  * high priority process.

multi-programming ≠ Time Sharing

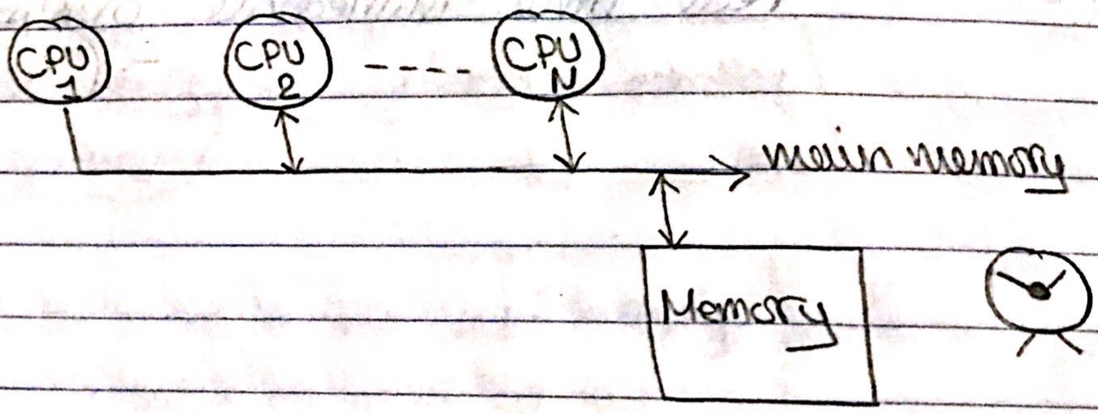
multi-programming	Time Sharing
The CPU switches when the job needs I/O	The CPU switches when Quantum Q is finished.

[F] Parallel Systems:-

Multi-processor systems with more than one CPU in close communication.

(1) Tightly Coupled Systems:

processors share memory, clock, & communication usually takes place in memory



* There are two types of processing:

a_ Symmetric multi-processing

↳ Each CPU has the same identical copy of the OS [Reliable & Simple]

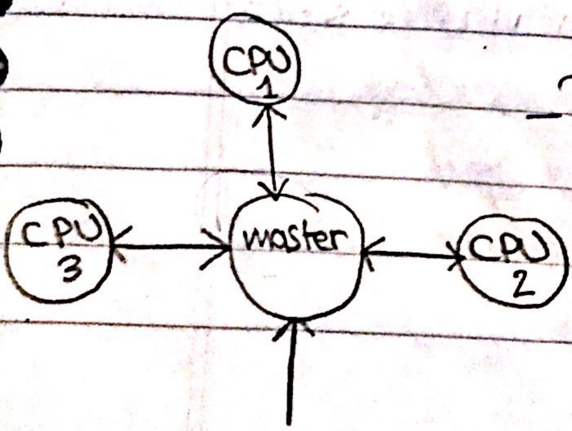
b_ ASymmetric multi-processing

_ There's a huge OS that runs this scheme.

_ There's one CPU called master CPU which controls other activities of other CPUs

_ The relation between the master and other CPUs is called master/slave relationship

↳ Reliable on all cases unless the master CPU is faulty.



(2) Loosely Coupled Systems:

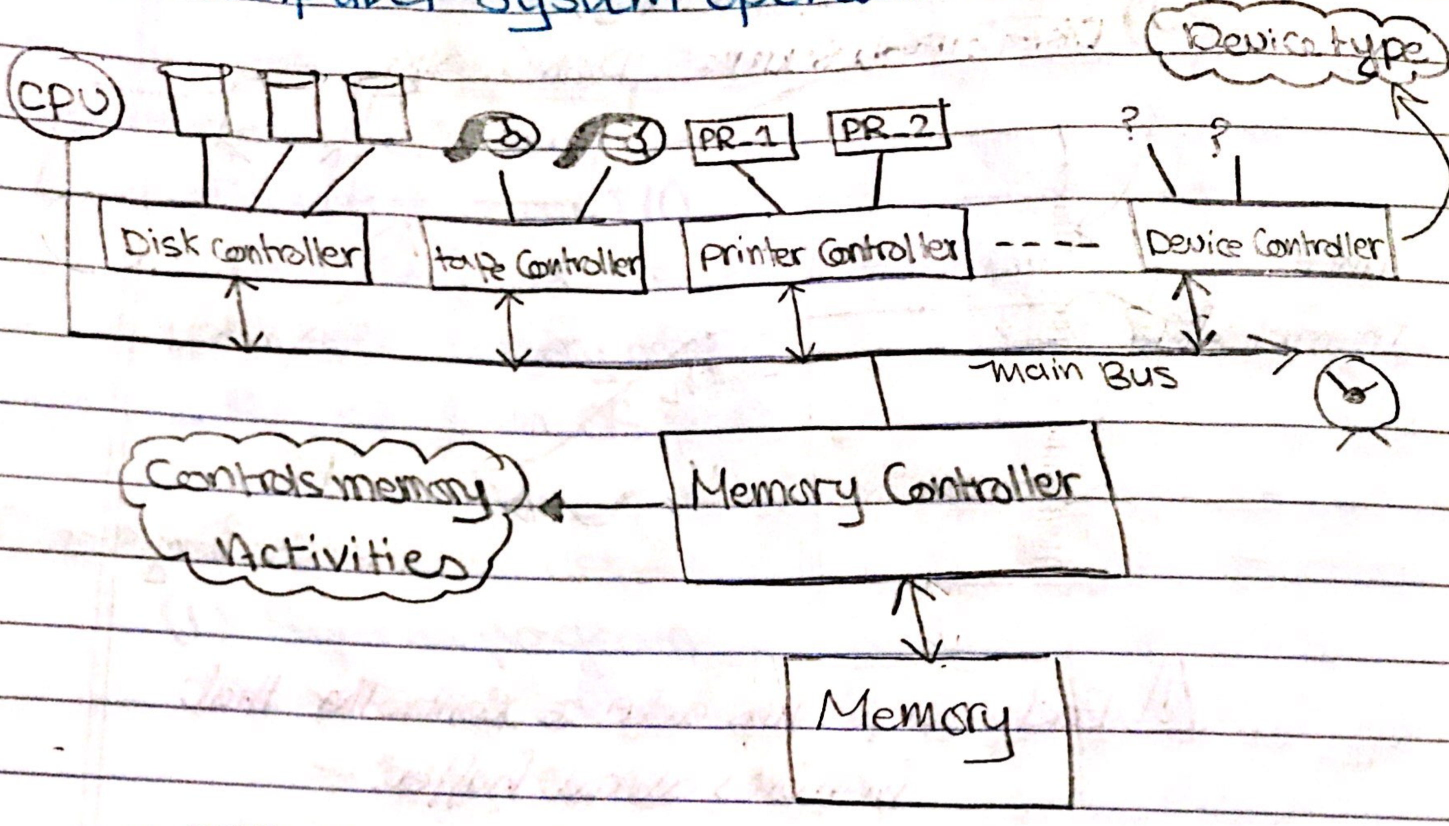
- Networks — كل نظام قائم بذاته
- "Distributed Systems" → Connect via servers.

[G] Real time Systems:

- takes data using sinsors.
- for systems that need Response real time "immediate" استجابة مباشرة

Chapter#2

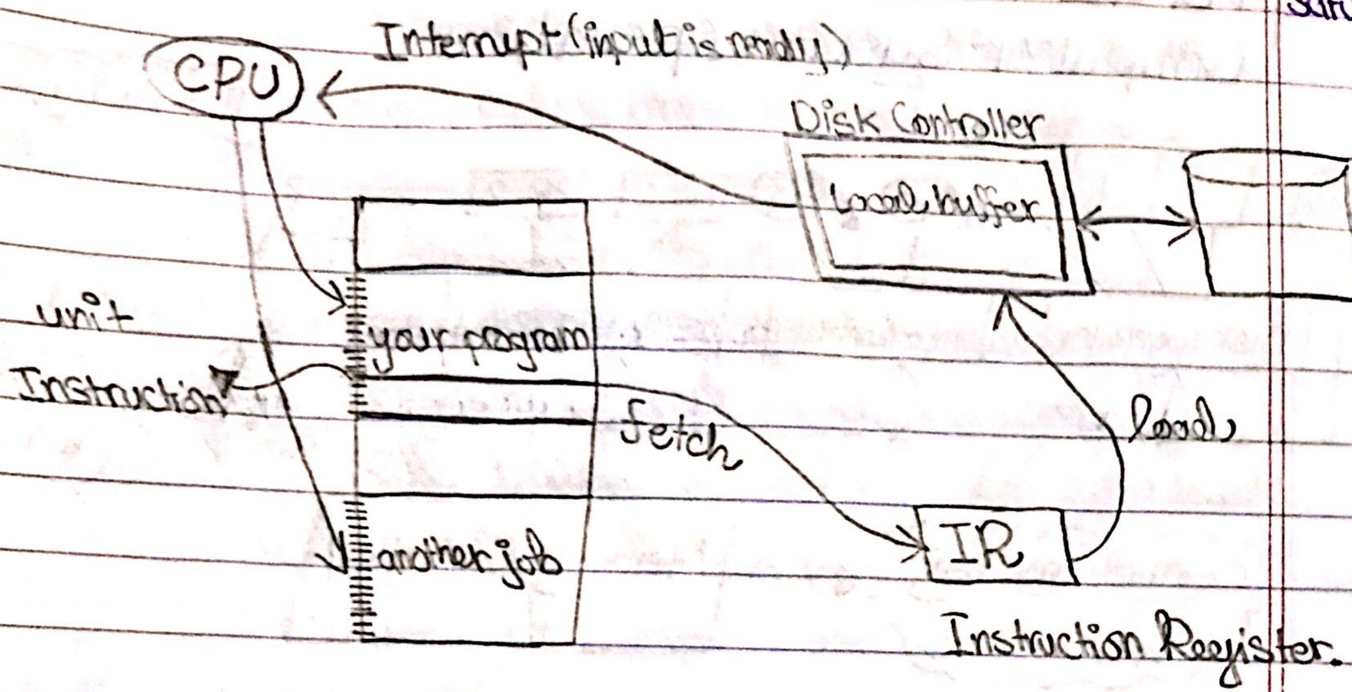
Computer System operations



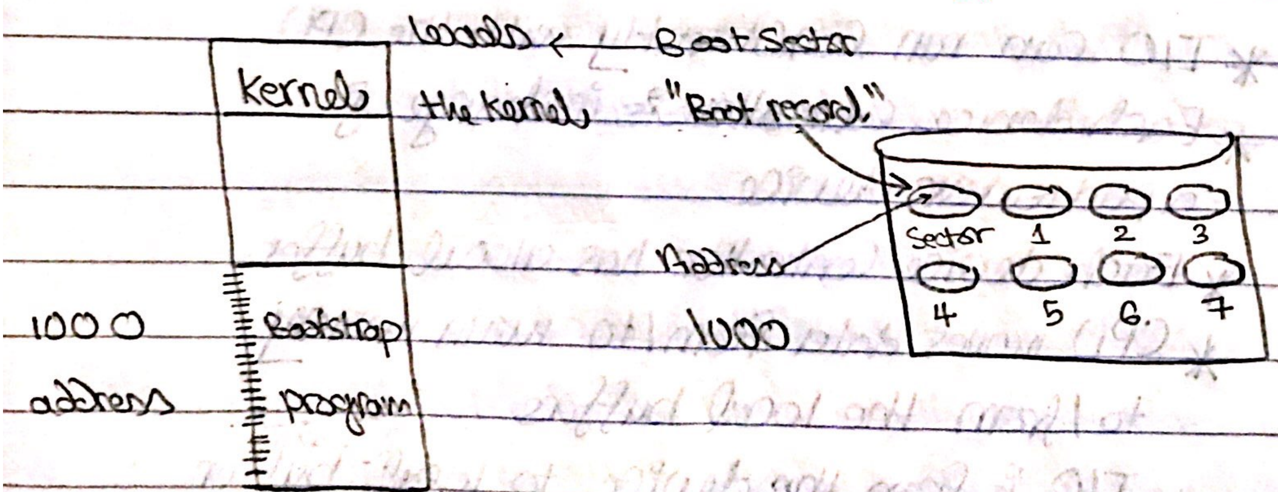
* I/O can run concurrently with the CPU.
* Each device controller is incharge of a particular device.
* Each device controller has a local buffer.
* CPU moves data from/to main memory to/from the local buffers.
* I/O is from the device to local buffer of controller.
* Device controller informs CPU that it has finished its operation by causing an interrupt.

lecture #4

Feb.17.2018
Saturday



⚠ Each device has a device controller that includes a local buffer.



⚠ Operating Systems are Interrupt driven (CPU is interrupted)

Interrupt: A Signal sent to the CPU by: "System Call"

- Hardware
- Software "Trap"

examples:

- Completion of an I/O. "Hardware Interrupt"
- Division by zero. "Software Interrupt"
- Invalid memory access. "hardware Interrupt".
- Request of an OS service.

OS services can be asked:

(1) System program

> Format a:

> Copy A.dat B.dat.

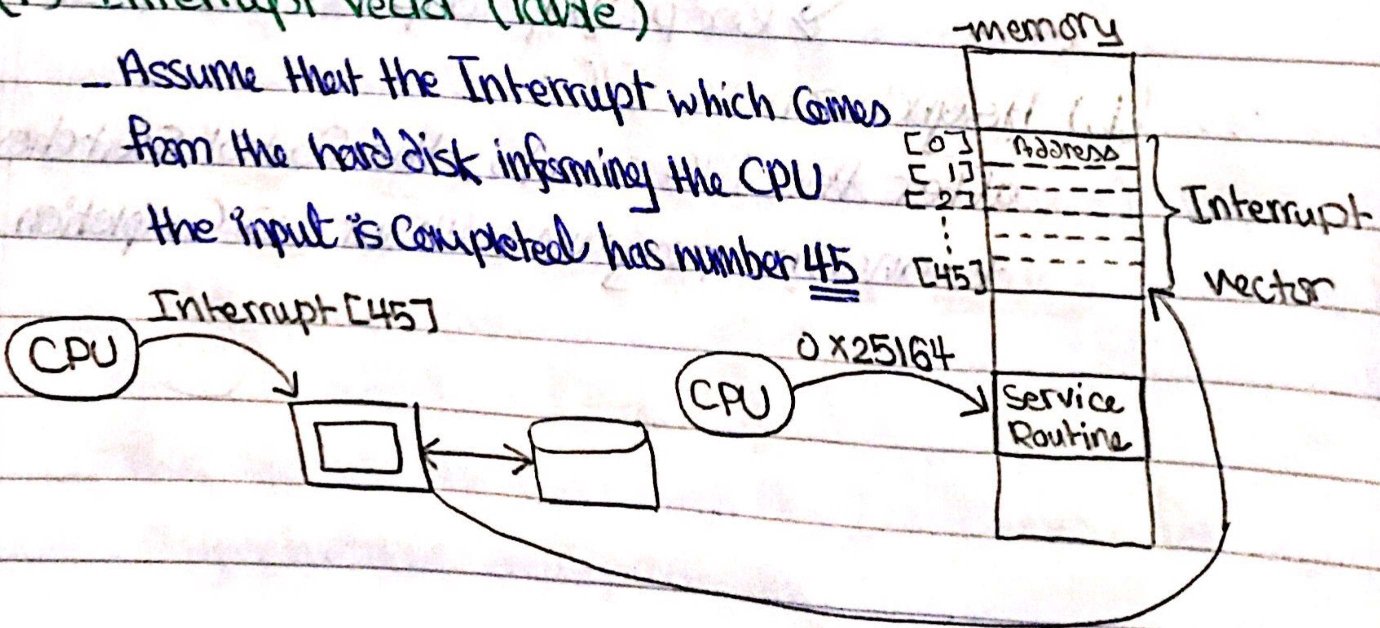
(2) System Call

It's an assembly language Instruction.

Interrupt Handling:

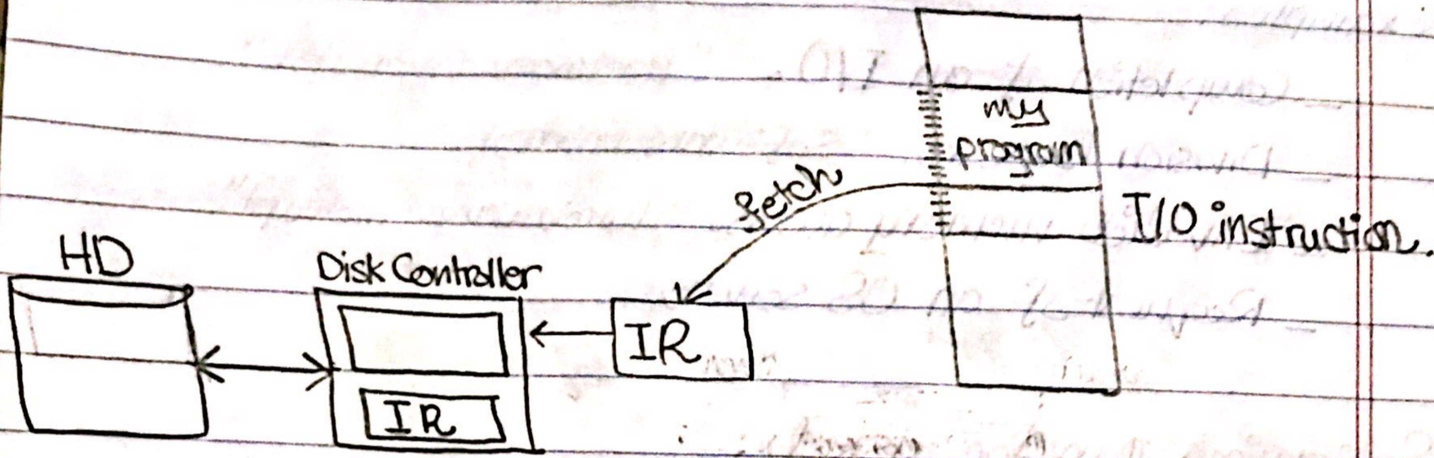
(1) Interrupt vector (Table)

- Assume that the Interrupt which comes from the hard disk informing the CPU the input is completed has number 45



(2) By polling:

I/O interrupt structure:



There are two types Of I/O:

(a) Synchronous I/O:

after the I/O Starts, the Control returns to program only upon I/O Completion.

→ The CPU waits until the I/O is Completed.

↳ wait Instruction "CPU is idle"

↳ loop: jmp loop

(b) Asynchronous I/O:

after the I/O starts, the Control switches to another program without I/O Completion.