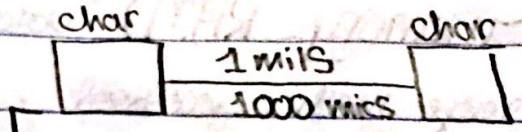
Direct Memory Access (DMA):

- Slow I/O devices such as keyboard, can send one character every one millisecond. 1mils = 1000 mics.
- The CPU can process the character in 2micro seconds through an I

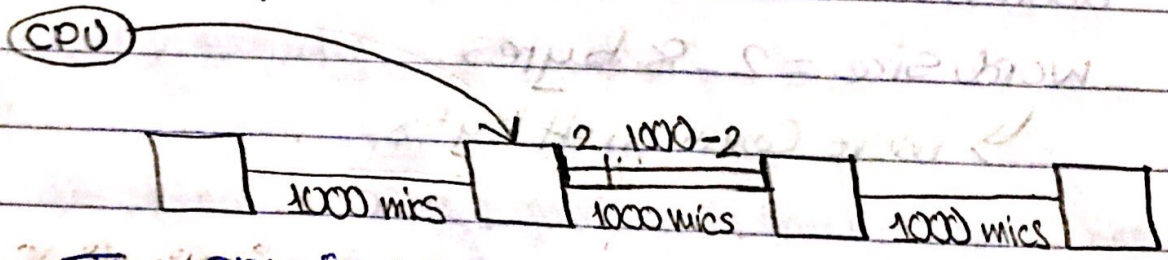
Lecture #5

Feb. 19. 2018
Monday.



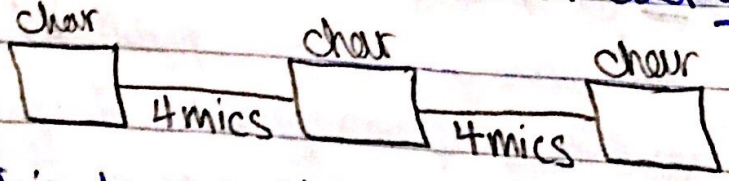
↳ Sends 1 char every 1000 mics.

→ The CPU needs 2 mics for the service routine to handle the interruption.



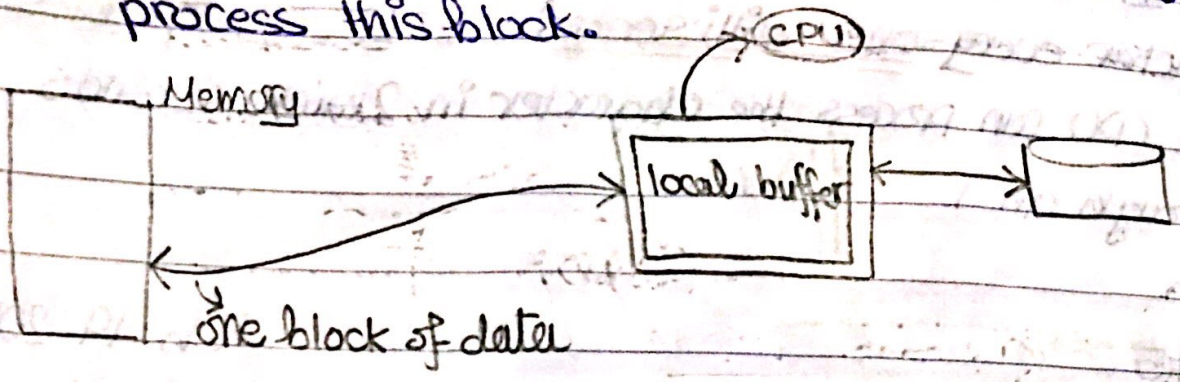
- The CPU is left 1000 - 2 = 998 mics for Asynchronous I/O processing.

- But, In high speed I/O devices (i.e Hard Disk) The HD can send or recieve char every 4 mics.



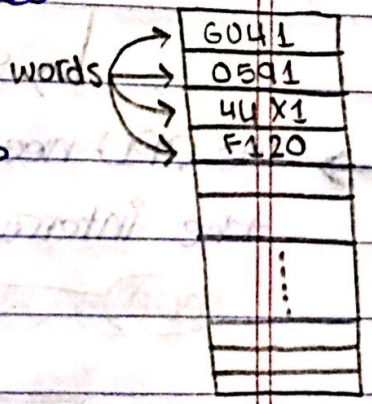
- This leaves the CPU with 4 - 2 = 2 mics for Asynchronous multiprocessing.

* In this case the OS sends one block of data each time & sends an interrupt to the CPU to process this block.



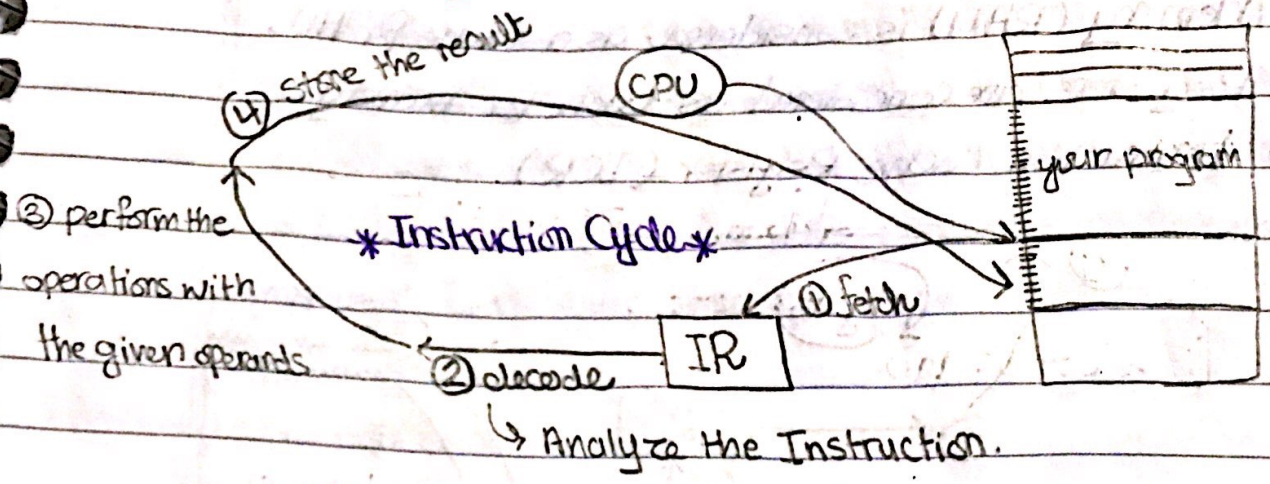
Primary Storage (memory_RAM) "volatile"

- Memory is largest storage media accessed directly by the CPU.
- Memory is array of words, each has address.
- word size = 2_8 bytes
  - ↳ more common 4 bytes



- The CPU performs the following instructions:-
  - (a) Load instruction: Fetch (get) instruction from memory to the IR (Instruction Register)
  - (b) Store Instruction: storing a Register in to memory location.

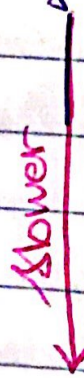
Instruction Cycle:



Secondary Storage: "Hard Disks, tapes, CDs, Flash memories ..."

- Factors that affect the secondary storage:
  (1) speed.
  (2) Cost.
  (3) Volatility. "موقت/دائم"

- The fastest storage medium in the system:



- Register
- Caches
- RAM
- HD
- Tapes
- etc.

* Cache Memory:

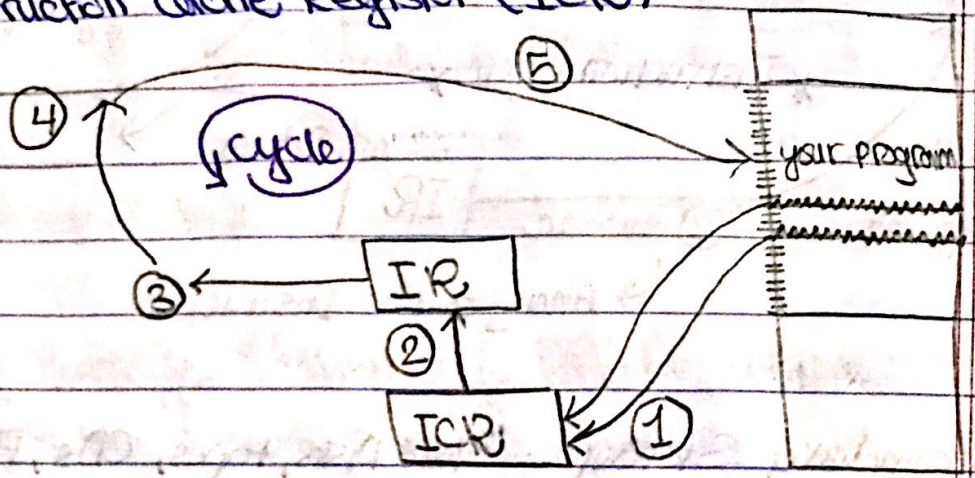
Chashing: is simply copying data to a faster storage medium to speed up execution & ensure good performance.

Examples:

(1) Memory (RAM) is considered as a Cache for HD.

(2) Registers are considered as Cache for memory.

(3) Instruction Cache Register (ICR)



⚠ ① & ② are concurrent [Parallel].

- from ICR to IR is much faster than RAM to IR

- L_1 Cache is the fastest Cache type because it's build within CPU.
  - L_2 bigger but slower.
    - etc.

Lecture #6

Feb. 21. 2018
Wednesday

Hardware Protection:

How the OS protects the Computer Resources?

- I/O devices.
- Memory.
- CPU.

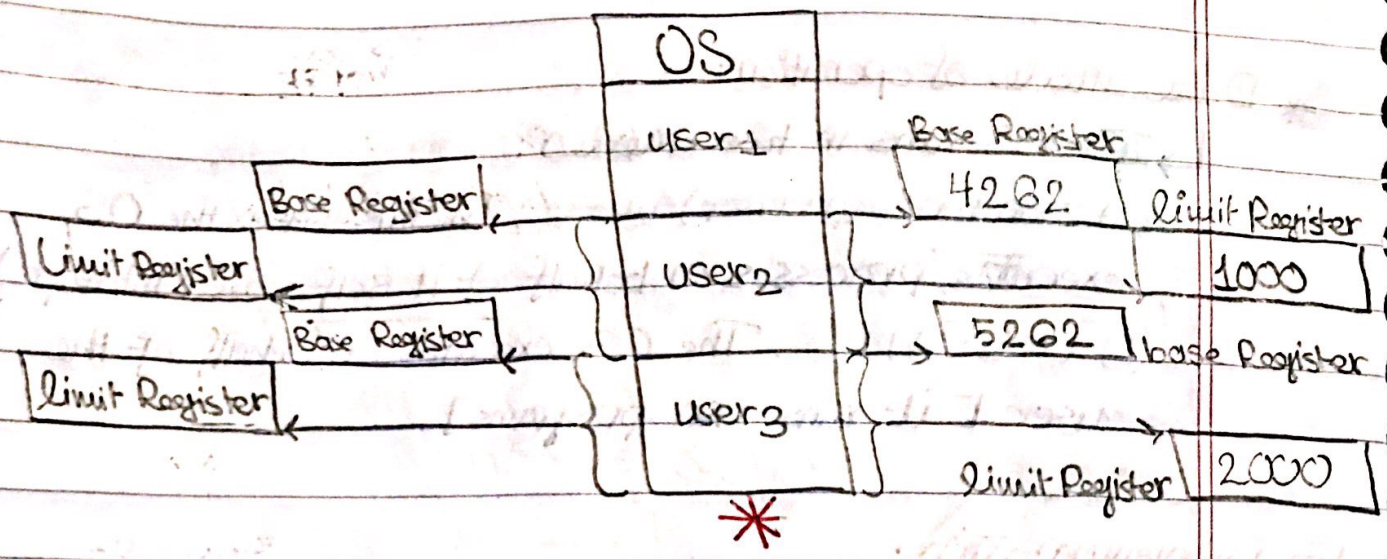
* Dual mode of operation:
  ↳ The OS runs in two modes:
  (a) Monitor (Supervisor) mode: In this case the OS executes process on behalf of it self. (i.e interrupts)
  (b) User mode: The OS executes on behalf of the user [it runs user programs].

Implementation:
↳ one bit is assigned called "mode bit"
0: monitor mode.
1: user mode.

Definition:
A Privileged Instruction is an instruction that's executed only in monitor mode. "OS"

I/O protection:
All I/O instructions are made Privileged Instructions.

Memory Protection:
To protect the memory Allocation space of user program & the OS it self.

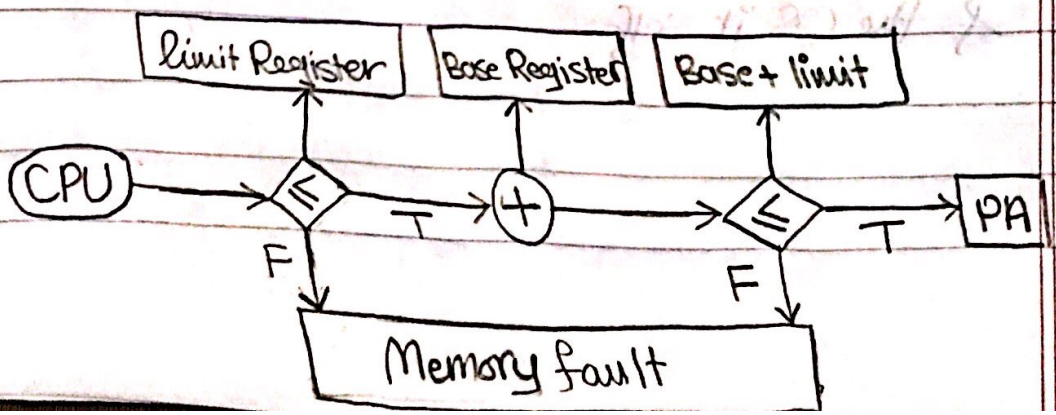


(LA) ← User 2 program → recorded to the memory * where 738 address starts from 4262 & has a limit of 1000

* Logical Address (LA): is the offset of the instruction in your program.
& LA = the address seen in your program.

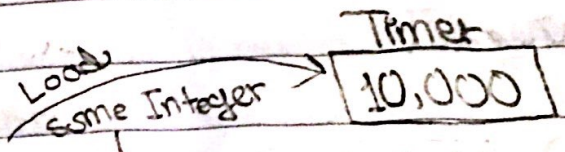
* Physical Address: is the actual address in memory
& PA = Logical Address + Base Register.
(i.e) PA = 738 + 4262 = 4998

➡ The most Important Concept; is How the OS Compute the PA?

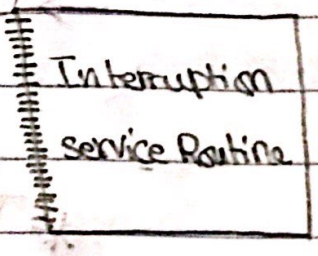


CPU protection:

- Timer



- With every clock tick the timer is decremented by 1
- When the timer reaches zero it sends an interrupt to the CPU.
- When the CPU recieves interruption it executes the interruption service routine which is responsible for checking the CPU.



- Timer can be used in Computer time Calculation.

Operating System Structure:

(1) process management.
(2) Memory management.
(3) File System (I/O) management.

→ The OS services can be provided in two methods:
- System Call: it's an assembly language Instruction.
- System programs.

Lecture #7

Feb. 24. 20[illegible]
Saturday

Process

* Process Concept:-

→ Definition: A process is a program in execution

Process ≡ program ≡ job

* There are different process types in the System:

(1) Batch Process:

generally, Batch process has low priority.

(2) Time Sharing process:

- users.
- program development.
- data entry.
- gamming.

(3) System tasks:

- Interrupt

* Process Contents:

The process contains:

(1) Code Section (Program Counter "PC")

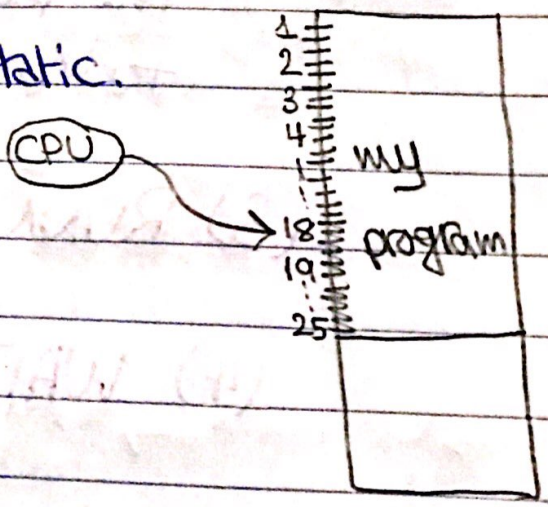
"PC": is the address of the Instruction executed currently. (a)

(2) Data Section:

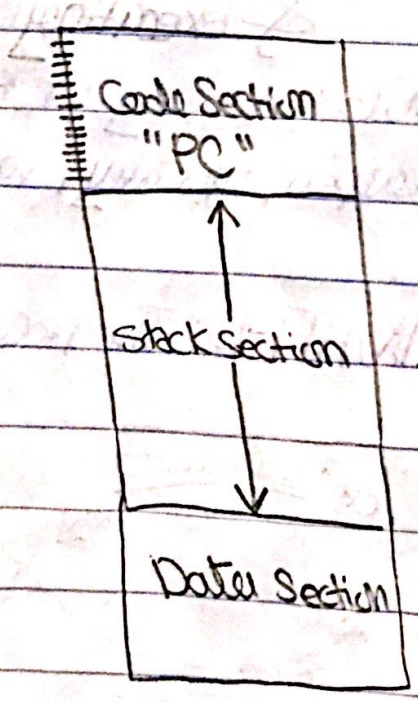
Memory allocation were Input & Output data is stored.

(3) Stack Section:

- Global variables: Static.



(a)



Process.

* Process States:

(1) NEW: The process is newly created.

newly Created Process

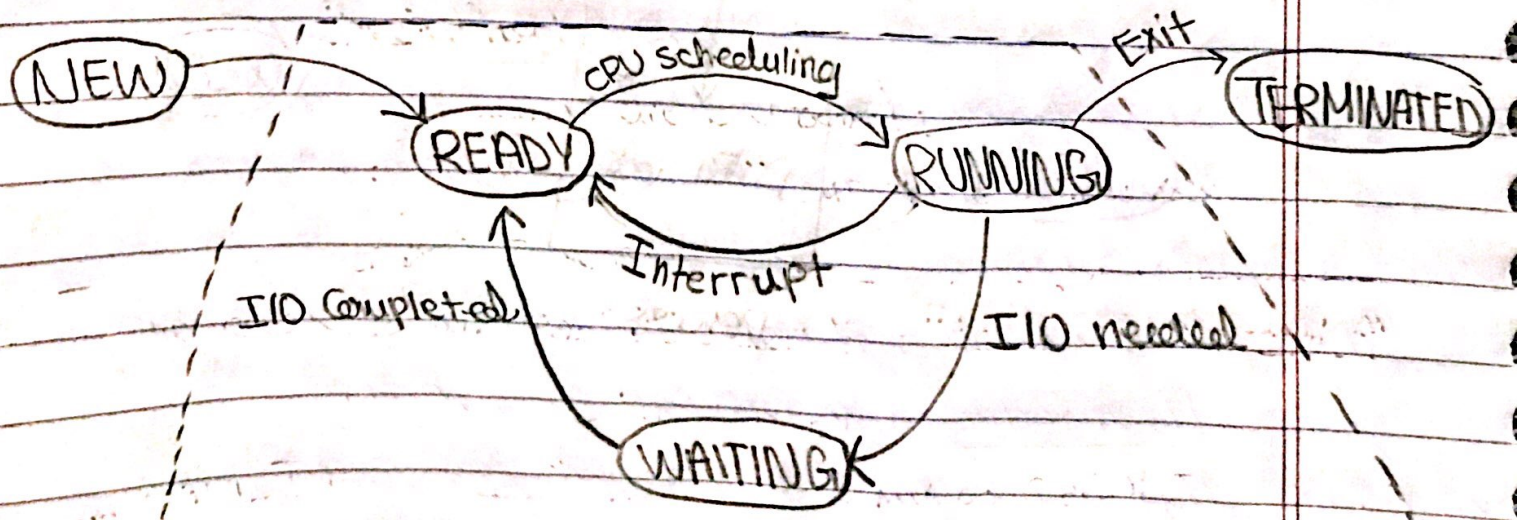
Job Queue "Pool"

(2) READY: The process is Loaded into Memory & ready to run and have the CPU. The process is in READY queue waiting to have the CPU to run.

(3) RUNNING: The process is having the CPU, & execution.

(4) WAITING: waiting for some event to happen, typically, waiting for I/O.

(5) TERMINATED: The process finishes execution.



Major process state.

Lecture #8

Feb. 26. 2018
Monday

Where data about the process is stored?!

→ Answer → Process Control Block (PCB):

It's a kind of a data structure, generally it's a table.

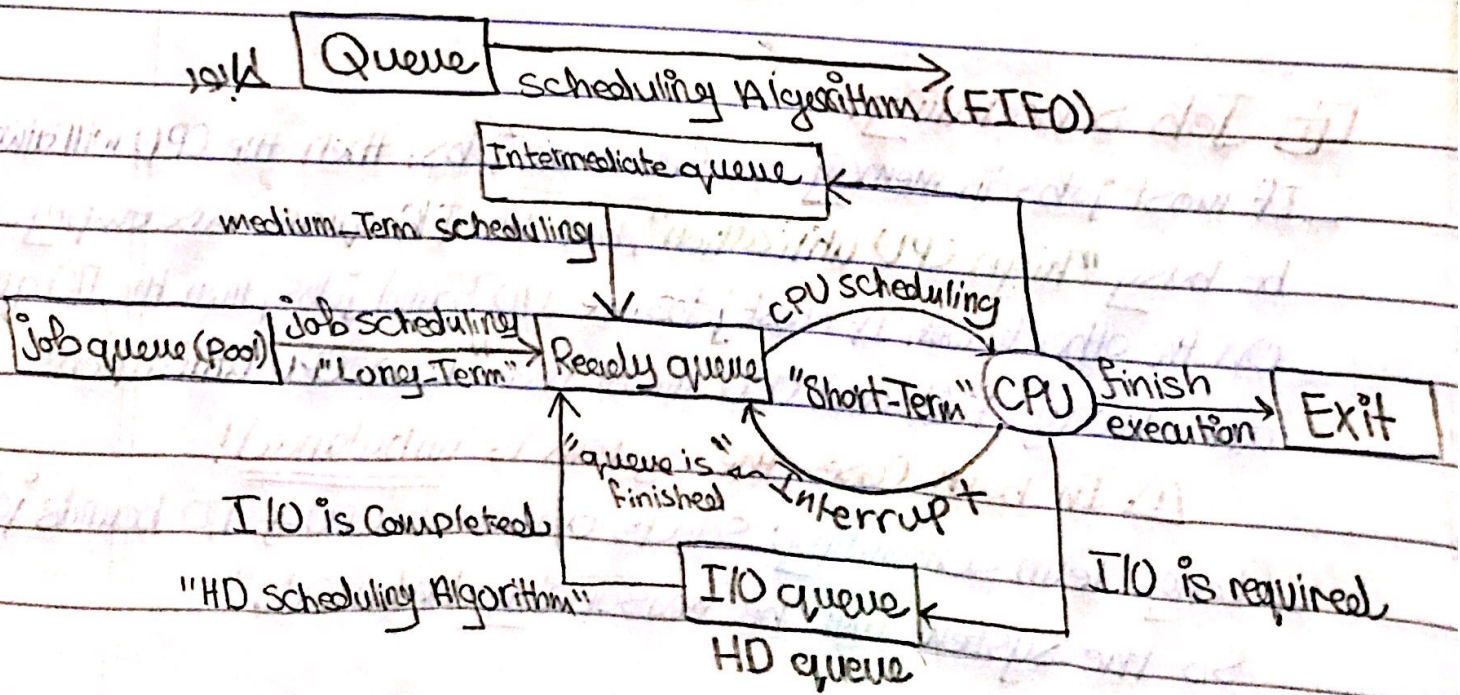
Data?!

(1) process ID.
(2) Process State.
(3) program Counter (PC).
(4) Registers.
(5) Scheduling Info.
(6) Memory Info (Base & limit Registers).
(7) Accounting Info.

PCB

Process ID
Process state
program Counter
⋮

Scheduling & System Queues.



Long-Term "job" scheduling:

It's selecting a job from the job queue to be admitted to memory and in term it's added to the READY queue. This selection "Interupt" is Invoked "Called", this invoked (Seconds, Minutes). That is, the OS has enough time to decide carefully which job to fetch. "Long-Term"

Short-Term Scheduling:

It's selecting a process "job" from READY queue to be given to CPU to RUN.
(milliseconds, microseconds, Nanoseconds), Therefore it should be very fast. "short-Term"

→ Definition → Degree of multi-programming: is the # of jobs in the memory "READY queue". Therefore, Long-Term scheduling controls the degree of multi-programming.

Job scheduling:

- If most jobs in memory are CPU bound jobs, then the CPU will always be busy "high CPU utilization", but the I/O queues are empty.
- On the other hand, If most jobs are I/O bound jobs, then the I/O queues are always full and CPU almost free "Low CPU utilization"

⚠ In both cases the system is unbalanced!

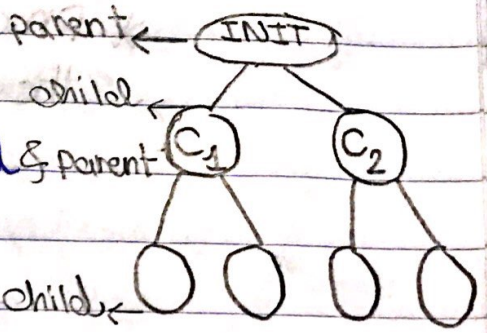
- Long-Term scheduling: selects a mix of CPU & I/O bound jobs So the System will be reasonably balanced.

Lecture #9

Feb. 28. 2018
Wednesday

Process Creation:

* parent process creates children processes, which, in turn create other processes, forming a Tree of processes.



→ **Resource Sharing:-**

(1) parent & children share all resources.

(2) children share subset of parent's resources.

(3) Parent & child share no resources. "parent & children Compete for All resources"

→ **Execution:-**

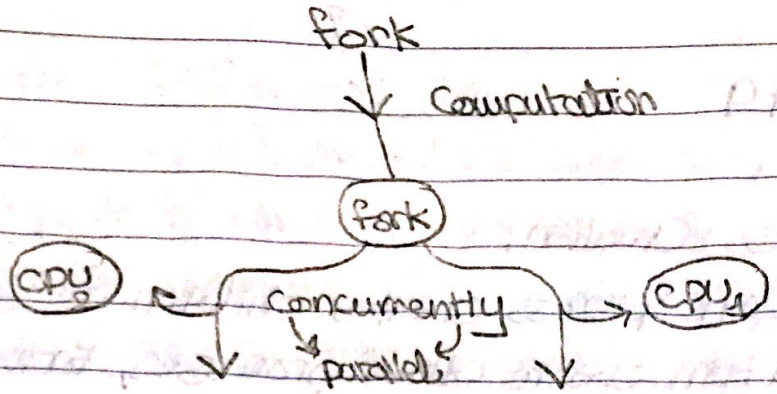
- Parent & children execute Concurrently.
- Parent waits until children terminate.

→ **Address Space:-** "Allocation of process in Memory"

- child duplicate of parent.
- child has a program loaded into it.

→ **Unix examples:**

- Fork System call creates new process.
- execve system call used after a fork to replace the process' memory space.



Process termination :-

* Process executes last statement & asks the operating System to delete it "exit"
  - output data from child to parent 'via fork'
  - process' resources are deallocated by OS.
    ↳ when a process is killed, all of its resources are deallocated.

* parent may terminate execution of children processes 'Abort'
  - task assigned to child is no longer required.
  - child has exceeded allocated resources.
  - Parent is exiting.
    # OS doesn't allow child to continue if its parent terminates.
    # Cascading Termination: If the parent process ends 'exit' all children & subchildren are exited.

Cooperating processes :-

Concurrent processes are either :

(1) Independant process : Cannot affect or be affected by the execution of another process.

(2) Cooperating process : Can affect or be affected by the execution of another process.

* Advantages of process cooperation :
  - Information Sharing.
  - Computation Speed-up.
  - Modularity.
  - Convenience.

→ Producer-Consumer Problem

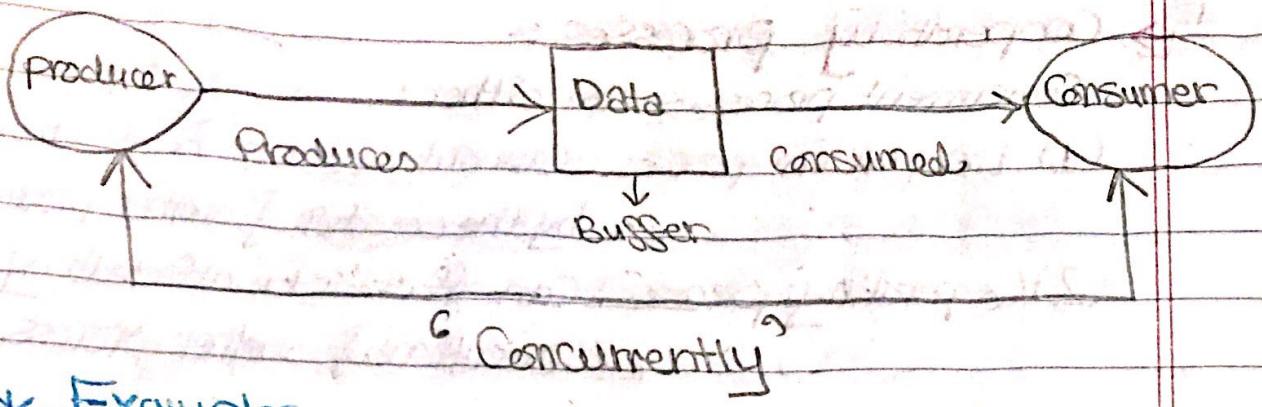
↳ example on Concurrent process?

* talking about Concurrency requires :
  - Cooperating among processes
    - (i.e Communication among processes)
  - Synchronization of processes action.

* To illustrate the idea of cooperating process consider the producer-consumer problem :

Producer process produces information.

Consumer process consumes this information.



"Concurrently"

* Examples:-
  - print program produces characters Consumed by the Printer.
  - Compiler produces assembly Code, Consumed by the assembler.
  - assembler produces machine Language Code Consumed by the Loader.

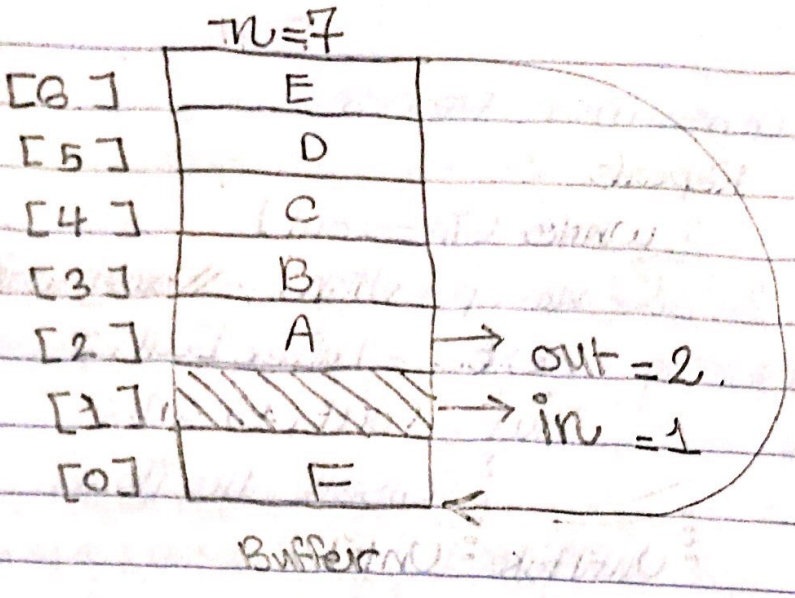
Lecture # 10

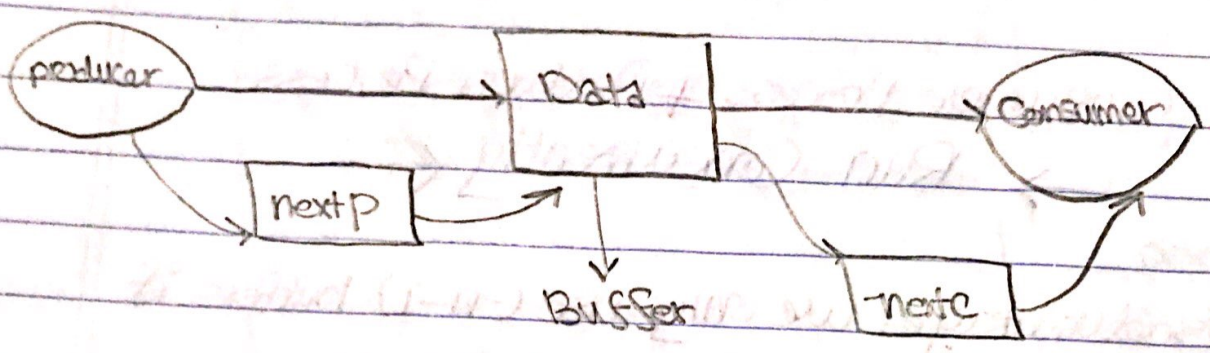
March 3, 2018
Saturday

* Data Structure Required:

```
Const int n ; //Size of Buffer.
type item; // item = char, bit, word.
var int buffer [n];
int in; // index where we add items in the buffer.
int out; // index where we take items from the buffer.
item nextP; // next produced item.
item nextC; // next Consumed item.
```

⚠ we will use the Circular Buffer idea in the implementation.





* Buffer is Full: $(in+1)\%n = out$.

Circular Buffer

* Buffer is Empty: $in = out$.

* Producer Process:

```
Repeat
    produce an item in nextP.
    while (in+1)%n == out
        no-operation;          // busy waiting
    buffer[in] = nextP;
    in = (in+1)%n;
until False
```

* Consumer process:

```
Repeat
    while (in == out)
        no operation;   //busy waiting
    nextc = buffer[out];
    out = (out + 1) % n;
    Consume the item;
Until False
```

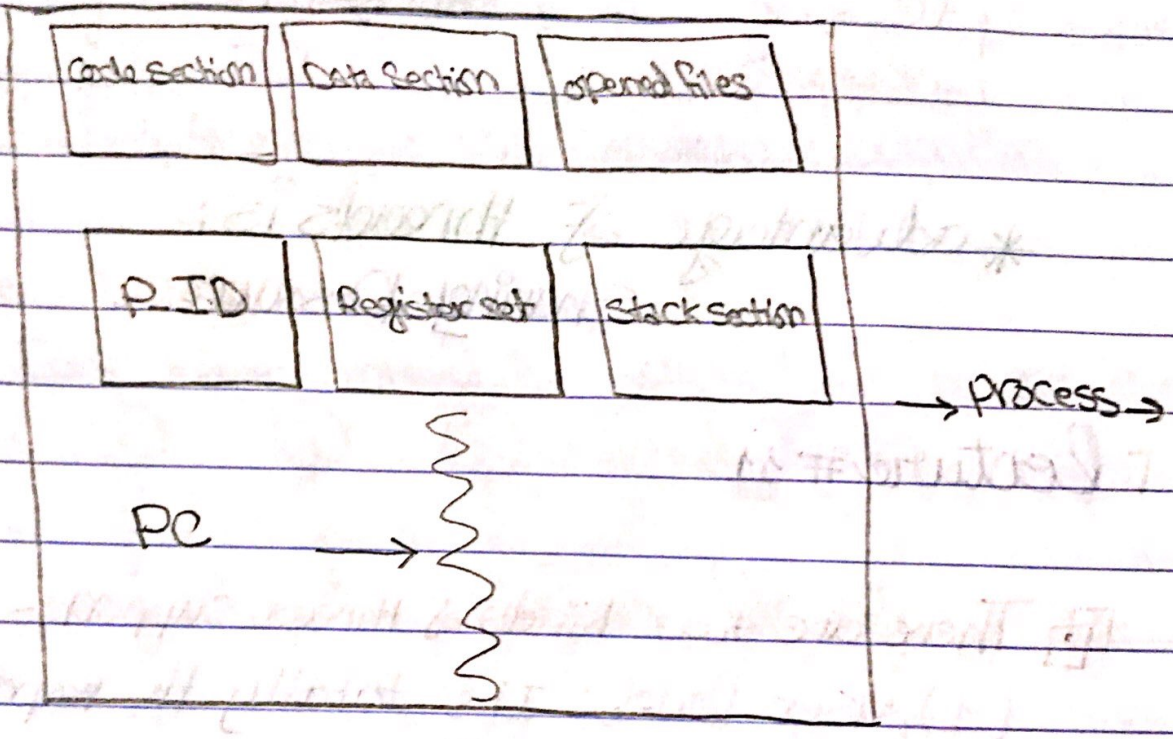
Consumer Process + Producer Process
→ Run Concurrently ←

* Disadvantage: we only use (n-1) buffers if we have n buffers.

Threads:

Process (Heavy Weight Process)

Code section	Process ID
Stack section	Registers set
Data section	Program Counter

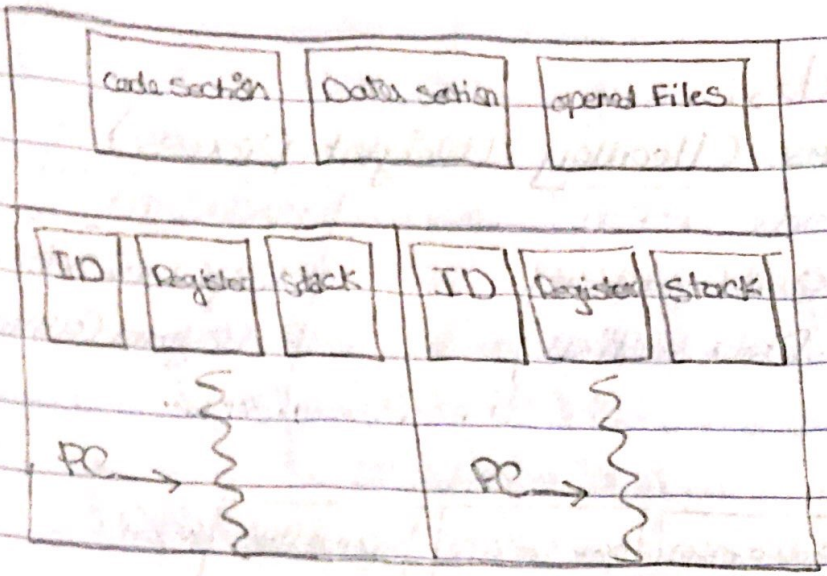


Thread (Light Weight Process) Contains:-

- Program Counter.
- Registers set.
- Stack section.

→ All peer threads; share:

- Code section.
- Data section.
- I/O Resources.



*advantage of threads is:
Sharing Resources.

lecture #11

March 5, 2018
Monday

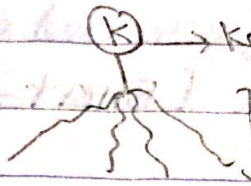
There are two kinds of thread support:-
(1), user level: It's totally the responsibility of the user (very complex & difficult)
at
(2) at kernel level: most OS support this kind of threads.

* Relationship between user threads & kernel threads:
→ Think about user threads as processes (programs)
& → Think about Kernel threads as CPUs

* We have → multi-threads.
→ multi-CPUs.

(1) Many-To-One:

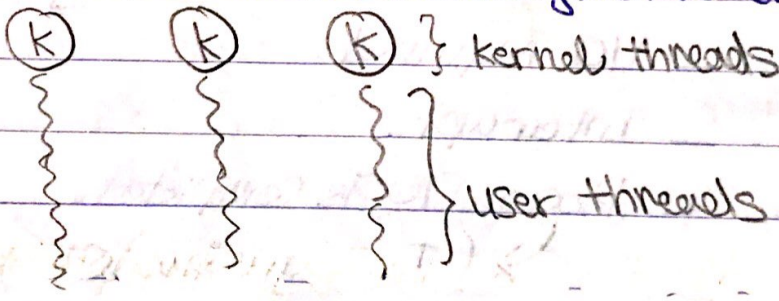
Many user threads are mapped to one kernel thread



⚠ Disadvantage: No concurrent execution.

(2) One-To-One

Each kernel thread is assigned to one user thread.



∴ Main advantage: This allows concurrent execution.

⚠ Disadvantage: we need enough kernel threads

(3) Many-To-Many:

Many user threads are mapped to an equal or less number of kernel threads.