OS Notes:

1) **What is OS?**
    1) Manages the processes and the memory
    2) It helps in managing the s/w and the h/w
    3) It helps us to interact with the computer without knowing the assembly language(Language of 0’s and 1’s)
2) **Functions of OS?**
    1) Memory Management
    2) Device Management
    3) Processor Management
    4) Security
    5) Error Detection
    6) Coordination between s/w and the user
    7) Job Accounting
    8) File Management
3) **Kernel:**
    1) Contains the minimum requirements for the OS to run eg: all the system call like wait() etc are contained inside the kernel.
4) **BSP(Bootstrap Program):**
    1) First program that runs when we start the OS
    2) Initialises the OS
5) **Types of OS:**
    1) Real Time OS(RTOS)
    2) Single User OS-> Single Task(eg: MS-DoS), Multi-Tasking(MacOS, Windows)
    3) Multi-User OS
6) **Processor:**
    1) It is a chip or a logical circuit that responds and processes the basic instructions
    2) Or, in layman terms we can say: processor executes the processes, inside the CPU
    3) Lifecycle of Processor:
        1) Fetching -> Decode -> Execute -> write-back
    4) Interacts with the input/output devices and also the memory based on the type of tasks it needs to perform.
    5) H/W component capable of executing instructions
7) **Process:**
    1) **Definition:**
        1) Instance of a Program under execution
        2) Process is an active-entity whereas program is a passive-entity

    3) Processor is one of the resource that is required for the process to execute

2) **Management:**
    1) Functions:
        1) To keep track of the status of each process
        2) To suspend and choose some processes based on certain criteria
        3) To coordinate inter-process communication
3) **Structure:**
    1) Code Region:
        1) Contains the executable instructions of the process
    2) Data Region:
        1) Hold the data areas used by the process
    3) Stack Region:
        1) Holds the dynamic data: eg: local variables, arguments to a function
    4) Program Counter(PC):
        1) Address of the next instruction to be executed.
4) **Lifecycle/States:**
    1) New -> Ready -> Running -> Waiting -> Ready -> Running -> Terminated
5) **Context-Switching:**
    1) The task of switching the processor/CPU to another process by saving the state of the old-process and loading the saved-state of the new-process
    2) Features:
        1) Keeps the processor/CPU busy all the time
    3) Occurs:
        1) An I/O operation is initiated
        2) And interrupt occurs
        3) An I/O operation is completed
        4) A process terminates
6) **Scheduling:**
    1) Ready-State -> 50 processes -> how to know which process to run first? ->
       Scheduler(to pass the process from Ready->Running state)
    2) **Policies:**
        1) Non-Preemptive:
            1) CPU is allocated to the process unless the job/execution is over
            2) Eg: FCFS(First Come First Serve)

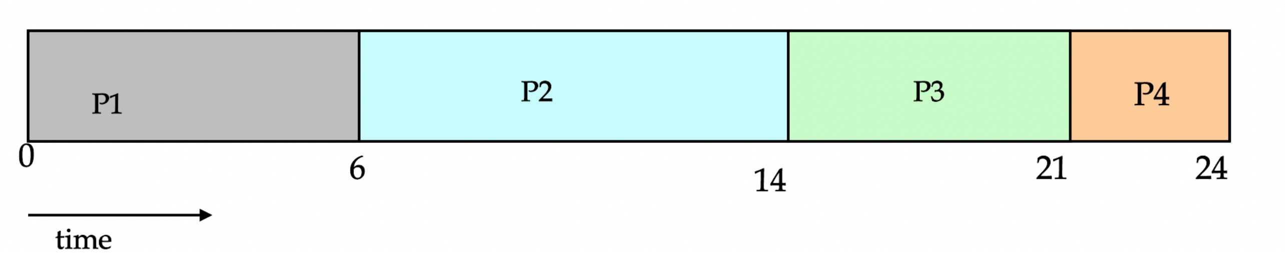
    2) Preemptive:
        1) The processes will switch based on various factors like priority, time-quantum etc.
        2) Eg: Round Robin
3) **Important Formulas:**
    1) **Burst Time/CPU Time:** The time the process will take to execute when it is assigned to the processor
    2) **Turn Around Time(TAT) =** Completion Time - Arrival Time
    3) **Waiting Time(WT) =** Turn Around Time - Burst Time(CPU Time)
    4) **Response Time(RT) =** The difference between the arrival time and the time at which the process first gets the processor/CPU
4) **Arrival Time:**

Time-> -1, 0, , 1 ,2
Ready Queue -> [], [P1,P3,P4], [P1,P3,P4],[P1,P2,P3,P4]
t=0 -> P1,P3,P4
t=1 -> P1,P3,P4
t=2 -> P1,P2,P3,P4

Processes	Arrival Time	Burst Time	TAT
P1	0		
P2	2		
P3	0		
P4	0		

5) FCFS(Non-Preemptive):

Processes	Arrival Time	Burst Time	TAT	Waiting Time	Response Time
P1	0	6	6	0	0
P2	0	8	14	6	6
P3	0	7	21	14	14
P4	0	3	24	21	21
			65	41	41



Avg TAT: 65/4 = 16.25
Avg WT: 41/4 = 10.25
Avg RT: 41/4 = 10.25

6) Round Robin:

Time-Quantum(The time for which the CPU will execute the process)

Eg:

1)
Time-Quantum: 5ms

P1 -> 10ms -> (10-5) -> 5ms -> (5-5) -> 0ms
P2 -> 15ms -> (15-5) -> 10ms -> (10-5) -> 5ms -> (5-5) -> 0ms

CPU(5ms) -> P1(5ms) -> P2(5ms) -> P1(5ms) -> P2(5ms) -> P2(5ms)

2)
Time-Quantum: 5ms

P1 -> 10ms -> (10-5) -> 5ms -> (5-5) -> 0ms
P2 -> 6ms -> (6-5) -> 1ms -> (1-1) -> 0ms

CPU(5ms) -> P1(5ms) -> P2(5ms) -> P1(5ms) -> P2(1ms)

3)

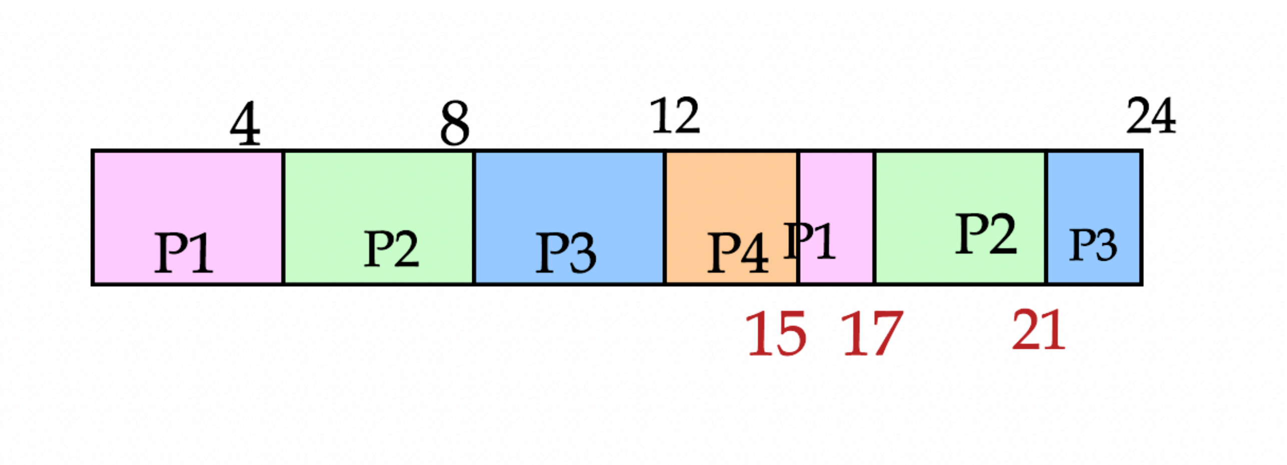
Time Quantum = 4ms

Process	Arrival Time	Burst Time	TAT	Waiting Time	Response Time
P1	0	6	17	11	0
P2	0	8	21	13	4
P3	0	7	24	17	8
P4	0	3	15	12	12
			77	53	24

t=0 -> [P1,P2,P3,P4]

P1-> 6 -> 2 -> 0
P2-> 8 -> 4 -> 0
P3-> 7 -> 3 -> 0
P4-> 3 -> 0

CPU(4ms) ->
P1(4ms) -> P2(4ms) -> P3(4ms) -> P4(3ms) -> P1(2ms) -> P2(4ms) -> P3(3ms)



Avg TAT: 77/4 = 19.25

Avg WT : 53/4 = 13.25
Avg RT: 24/4 = 6

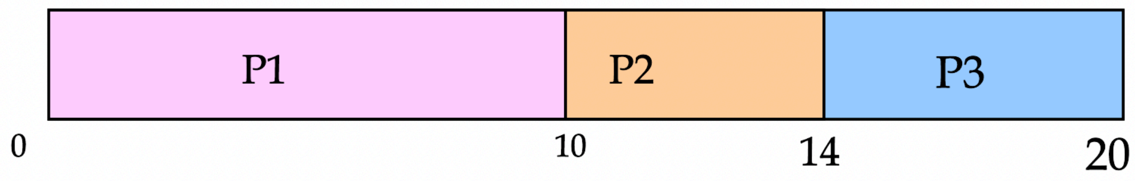
7) Priority Based Scheduling:

Non-Preemptive:

Processes	Arrival Time	Burst Time	Priority	TAT	WT	Response Time
P1	0	10	2	10	0	0
P2	2	4	1	12	8	8
P3	0	6	3	20	14	14
				42	22	22

Ready Queue: [P1,P3],[P1,P3],[P1,P2,P3]

t=0 -> P1, P3
t=1 -> P1, P3
t=2 -> P1,P2,P3
t=10 -> P2,P3 {P1 has completed its execution}
t=14 -> P3



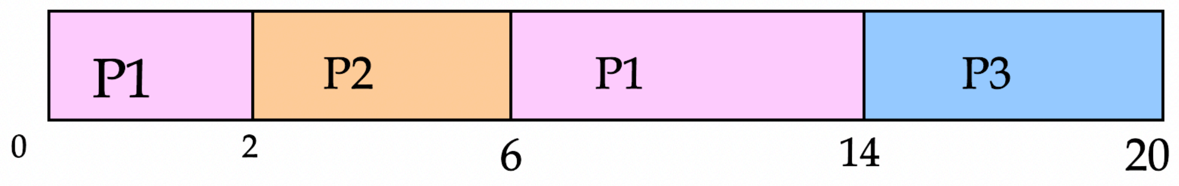
Avg TAT: 42/3 = 14
Avg WT: 22/3 = 7.33
Avg RT: 22/3 = 7.33

Preemptive:

Process	Arrival Time	Burst Time	Priority	TAT	Waiting Time	Response Time
P1	0	10	2	14	4	0
P2	2	4	1	4	0	0
P3	0	6	3	20	14	14
				38	18	14

Ready Queue: [P1,P3],[P1,P3],[P1,P2,P3]

t=0 -> P1, P3
t=1 -> P1, P3
t=2 -> P1,P2,P3



Avg TAT: 38/3 = 12.67
Avg WT: 18/3 = 6
Avg RT: 14/3 = 4.67

Disadvantage:

Some lower priority process have to wait for a longer time because of higher priority processes

Solution: Aging: we increase the priority of the process after sometime.

7) Threads:

1) Basics:

1) Lightweight as compared to processes.
2) Process-> Complete Program, Threads-> Segment of a program
3) In case of multi-threading two parts of the same program can execute simultaneously
4) Thread share same **Address Space, Data Section, Resources**
5) Threads have different **Stack Section(local data), PC**
6) Context-Switching is easy in case of threads

2) Lifecycle/States:

New -> Runnable -> Running -> wait/sleep/block -> Runnable-> Running -> Terminated

3) Lifecycle methods:

1) wait(): Running->waiting
2) sleep(): Running->sleeping
3) suspend(): Running->blocking
4) Running->Runnable:
    1) yield(){it can stop the currently executing thread and give chance to threads that are in the waiting state but of the same priority}:

    2) notify()/notifyAll() {waiting state}
    3) sleep time is up {sleeping state}
    4) resume() {blocking state}

8) Memory:

1) Component to store information for immediate usage
2) Types:
    1) Primary(RAM):
        1) Expensive
        2) Fast
        3) Works directly with the processor
        4) Implemented using Stack data-structure
        5) Stores local-data, returns addresses, used for parameter passing
        6) Stack overflow error will occur when too much of the stack is used
        7) Maximum size is already determined when the program starts
    2) Secondary(HDD):
        1) Cheap
        2) Slow
        3) Does not directly work with the processor
    3) Cache:
        1) Very high speed semi-conductor memory
        2) Acts as a buffer between main-memory and CPU
    4) Heap:
        1) Variables must be destroyed manually
        2) On Demand allocation
        3) Can have fragmentations
        4) Used for Dynamic allocations
        5) Responsible for memory leaks
3) Process Register:
    1) Every processor has a local storage -> Register
    2) Holds the data that is being processed by the processor/ CPU
    3) Two Types:
        1) General-Purpose
        2) Special-Purpose

4) **Cache V/S Register**

Cache	Register
Less faster and less efficient	Accessing it is faster and more efficient
Contains instruction code	It cannot
Machine instructions can be stored	Cannot
——-	In multiple-core processor all the processors share common cache

5) Management:

1) Core functionality of OS

2) Decides the location of memory to allocate and de-allocate based on the process requirement.

6) Address Space:

1) Physical Address:

1) Actual locations in the main-memory

2) Since it cannot be accessed directly by the user-program:

——-a logical/virtual address need to mapped to make the respective physical address available

3) The above address mapping -> **MMU(**Memory-Management-Unit**)**

4) MMU: is a h/w component responsible for translating a logical address to a physical address

2) Logical Address:

1) Created by CPU during Program execution

2) Do not exist physically

3) Used as a reference to access the physical address

7) Fragmentation:

1) Defintion:

1) during allocation and de-allocation , the memory is broken into pieces

2) because of the above it happens that we cannot assign that memory and memory space remains free even-though available

2) Types:

1) Internal:

Occurs when allocated>requested

2) External:

Occurs when memory is available but in non-contiguous manner

8) Swapping:

1) We swap the process from the main-memory to the secondary memory.

2) Swapping is done w.r.t secondary memory.

3) Swap-in -> main-memory-> secondary-memory

4) Swap-out-> secondary-memory-> main-memory

8) File Systems:

1) Organising and retrieving files from the Hard-Disk

2) Stored in the form Directories(Folders)-> we can have other files/directories

3) Windows -> NTFS -> New Technology File System

4) **Management:**

1) All the file related operations like moving, locating, opening etc. all are handled by this

9) Security in OS:

1) Providing protection to computer resources like memory, CPU, data etc

Using:

1) Authentication
    1) Using username/password
    2) Using card/key
    3) Using attributes, like bio-metrics
2) Authorisation
3) OTPs, Random Numbers(ICICI cards), Secret Keys, Network Login

10) Program Threats:

1) Trojan horse: Captures the credentials and sends to malicious users
2) Trap Door: Can access certain parts of the program without logging in
3) Logic Bomb: program misbehaves when met with certain conditions, harder to detect
4) Virus: replicate themselves, cause system crash, file changes

11) System Threats:

1) Worm: replicate files, using up the computer resources
2) Port scanning: hackers, can detect system vulnerabilities
3) Denial of Service

NOTE:

1) SJF(Shortest Job First): It can be both non-preemptive as well as preemptive.

Assignment:

1) **Have a brief reading about:**
    1) Micro-controller
    2) Micro-processor
    3) Embedded-processor
    4) DSP and Media-processor