Raghuveer Diya
2301010268

DELTA
Date / /
Pg No.

Operating System Assignment - 1

Part A

1. Despite evolution of hardware, why do modern systems still rely heavily on Operating system ?

Modern systems still rely heavily on OS because they manage and coordinate hardware resources like CPU, memory, storage, I/O devices. OS provides a convenient interface between hardware and users/application, enabling portability, multitasking and security.

2. Which type of OS ~~not~~ would be most suitable for a wearable health device and why ?

Real Time O.S. ensures timely, predictable and reliable response to inputs like heart rate signals processes data with low latency, provides efficient resource management on small, low-power hardware

3. Which structure would you avoid (monolithic, layered, Microkernel) to build a new OS kernel for a performance critical environment and why ?

Avoid a monolithic kernel, while it gives fast system calls they lack modularity and are harder to maintain / debug. A bug in one service can crash whole system, making them unreliable for critical systems.

4.

Refute the claim, because OS structure directly impacts performance, reliability, scalibility and security.
for eg.- microkernel isolates servides for fault tolerance, while a layered structure improves maintainability.
Just "running processes" isn't enough if the system is low, insecure or unstable.

5. Process Switching Error

(i)

The PCB stores CPU registers, program counter state, and memory info. By examining it, we can detect misinitialized registers, wrong state flags, incorrect program counter values that cause faulty switching.

(ii)

When a task unexpectedly moves from running to waiting, content switching saves the current process state (registers, program counter, PCB updates) and loads the state of the next process. It ensures execution resumes correctly later.

(iii)

Use an asynchronous, non-blocking system call because this allows the process to continue execution while the I/O is allocated in the background, preventing CPU from idling.

DELTA
Date / /
Pg No.

Part-B

6. Context switching Time Calculation:
save state = 2 ms
load state = 3 ms
shedulad overhead = 1 ms

(a) Total time = 2+3+1 = 6 ms

(b) Impact on multitasking performance:
- content switching is pure overhead (no useful work is done during this time).
- Higher switching time reduces CPU efficiency, as more time is spent switching than executing process.
- frequent content switches with high overhead can slow down throughput and increase response time.

7. Thread Efficiency check:
Total time in single-threaded = 40 sec
threads per process (ideal conditions | ts/perfect parallelism, no overhead)

Execution time estimate:
$T_{multi} = \frac{T_{single}}{n} = \frac{40}{n}$ seconds.

eg. if n = 2 → 20 sec
n = 8 → 5 sec

How multithreading improves performance
- It improves performance by running tasks in parallel, reducing execution time.

- It keeps CPU busy even during I/O waits, avoiding idle time.
- Threads share resources making execution faster and more efficient.

8.

Process	Burst time (ms)
P_1	5
P_2	3
P_3	8
P_4	6

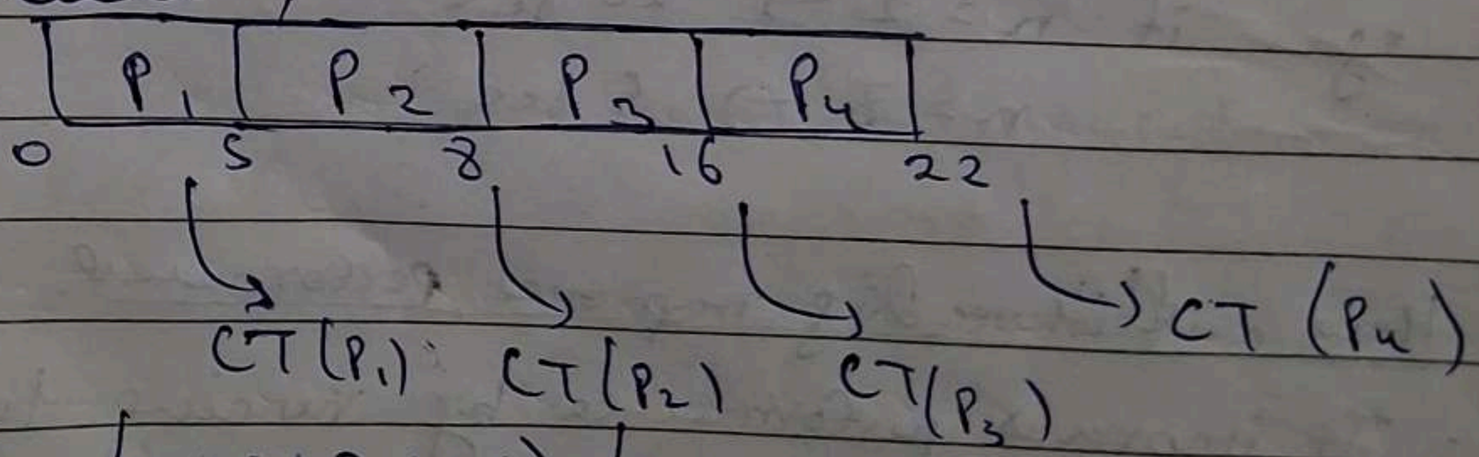
(a) . FCFS

(b)

Process	Arrival time	Burst time	Completion time	Waiting time	TAT
P_1	0	5	5	5-5 = 0	5-0 = 5
P_2	0	3	8	8-3 = 5	8-0 = 8
P_3	0	8	16	16-8 = 8	16-0 = 16
P_4	0	6	22	22-6 = 16	22-0 22

WT = Turnaround - Burst (TAT - BT)
TAT = Completion - Arrival (CT - AT)

Gantt Chart



Avg WT = (0+5+8+16) / 4 = 7.25 ms
avg TAT = (5+8+16+22) / 4 = 12.75 ms.

Date / /
Pg No.

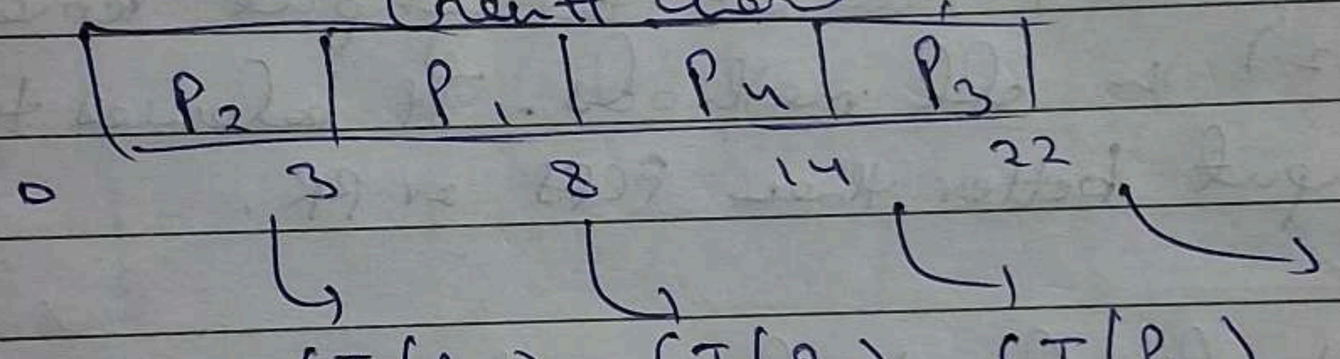
- Non-preemptive SJF

Process	AT	BT	CT	WT	TAT
P_1	0	5	8	8-5=3	8-8=0
P_2	0	3	3	3-3=0	3-3=0
P_3	0	8	22	22-8=14	22-22=0
P_4	0	6	14	14-6=8	14-14=0

TAT = CT - AT

WT = TAT - BT

Gantt chart



avg WT = (3+0+14+8)/4 = 6.25 ms

avg TAT = (8+3+22+14)/4 = 11.75 ms.

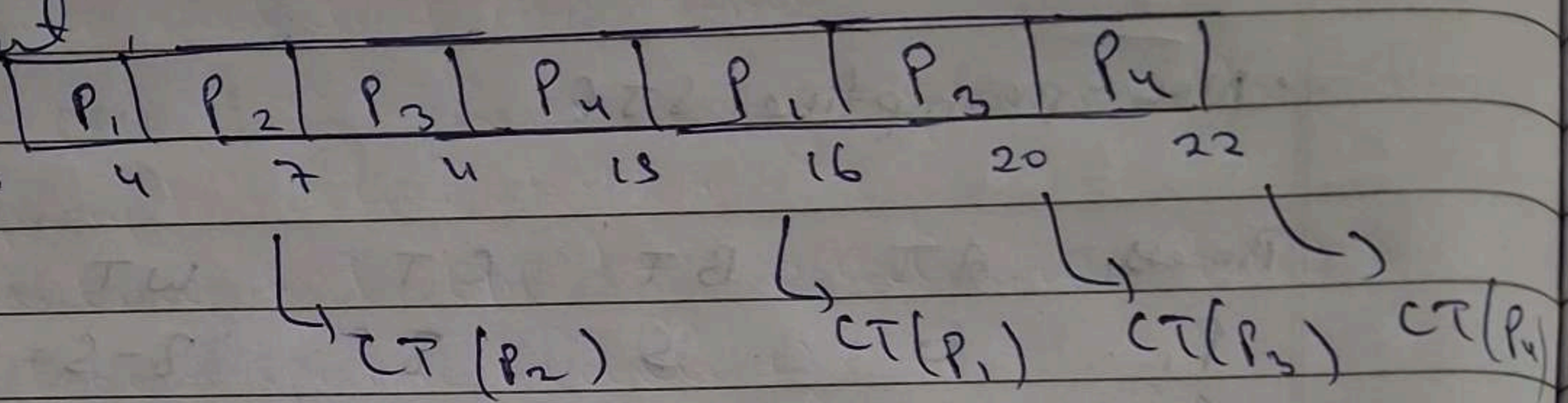
- Round Robin (quantum = 4 ms)

Process	AT	BT	CT	WT	TAT
P_1	0	5	16	16-5=11	16-0=16
P_2	0	3	7	7-3=4	7-0=7
P_3	0	8	20	20-8=12	20-0=20
P_4	0	6	22	22-6=16	22-0=22

WT = TAT - BT

TAT = CT - AT

Gantt chart



avg WT = (11+4+12+16)/4 = 10.75 ms
avg TAT = (16+7+20+22)/4 = 16.25 ms

(c) Non-preemptive SJF is best because it gives the lowest avg waiting time (6.25 ms) and turnaround time (11.75 ms), while throughput remains the same (4/22) in all methods. It balances turnaround and throughput better than FCFS or RR.

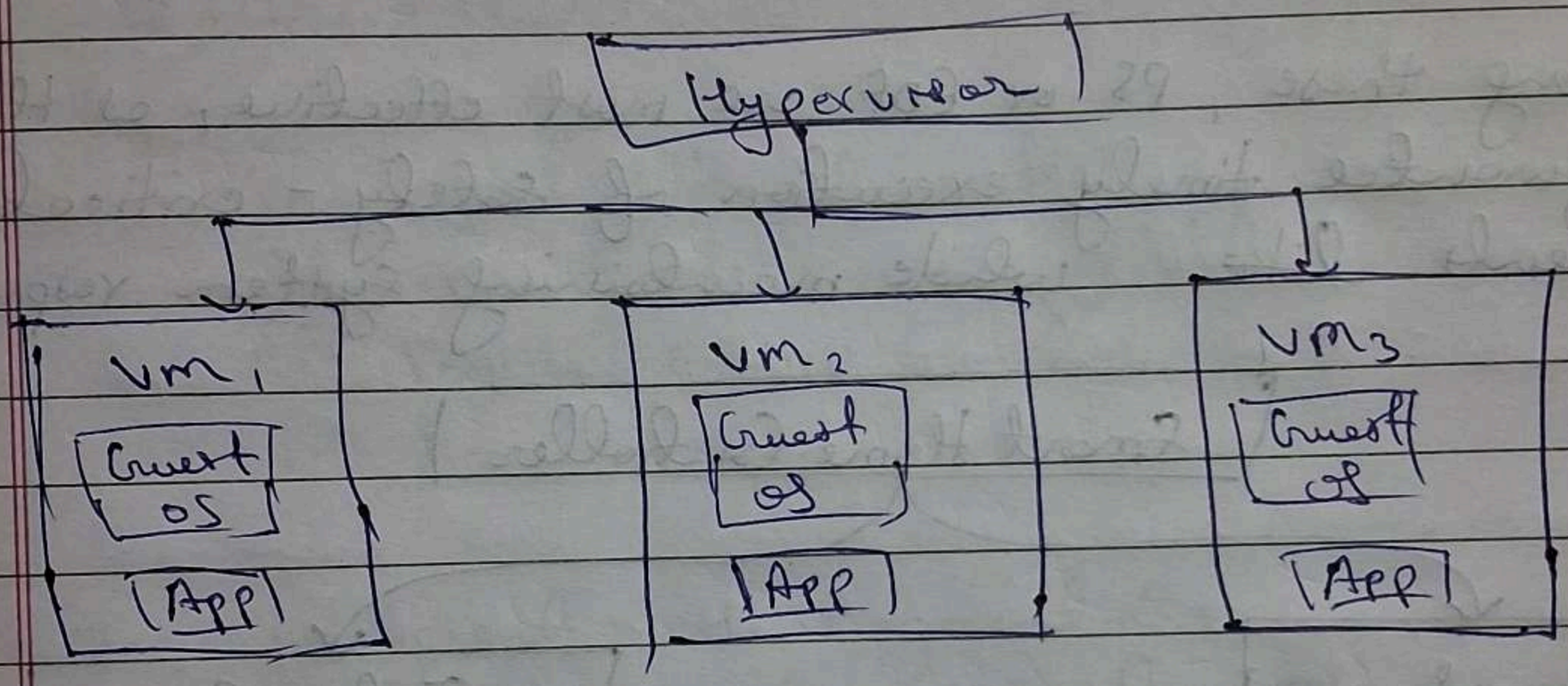
9. (i) Virtualized cloud migration

(a) Microkernel Architecture would be the best choice because
High scalibility → services (like drivers, file system, networking) run in user space and can be extended/updated easily.
High security and fault isolation → failure in one service doesn't crash the entire system.

(b) Role of virtual machines in Migration

- Isolation → Each VM run its own OS, preventing one service failure or attack from affecting others.

- Management → VMs can be created, paused, or migrated dynamically across servers for load balancing.
- Resource optimization → Hypervisor allocates CPU, memory, and I/O efficiently among VMs, ensuring better utilization and reducing wastage.



(ii) Smart Home System (IOT devices)

(a) OS role with scheduling + IPC

- process scheduling → The OS assigns higher priority to critical tasks (eg. intrusion detection) so they preempt less urgent ones (like lighting).
- Inter-process Communication (IPC) → Enables devices and controller processes to exchange data quickly (eg - camera sends motion alert → controller process reacts immediately).

(b) Suitable Scheduling Algorithms

- priority scheduling → ensures urgent processes like security alerts prempt routine tasks.
- Earliest deadline first (EDF) → useful when tasks must complete before specific deadlines.
- Rate-monotonic scheduling (RMS) → suitable for periodic tasks, like sensor checks.

Among these, PS or EDF are most effective, as they guarantee timely execution of safety-critical events like while maintaining system responsiveness.