PAGE NO.:
DATE: / /

ASSIGNMENT-1

(1) An OS provides essential abstraction and resource management (virtual memory, CPU scheduling, device drivers) so applications can be portable and hardware changes are hidden. It also enforces protection, isolation and system services which raw hardware alone cannot supply reliably.

(2) Best OS for a wearable heart monitor
A small Real-Time Embedded OS (RTOS) is most suitable — it offers deterministic task scheduling, low latency for interrupt-driven sensor reads, a tiny memory / CPU footprint and strong power-management features required for battery-operated wearables.

(3) Avoid a microkernel for a performance-critical environment because its heavy reliance on inter-process communication and frequent context switches for basic services adds latency and overhead compared with a monolithic kernel.

(4) Refute — structure matters: it affects performance, reliability, security, maintainability and real-time behaviour. Different structures trade off these properties (eg. monolithic → high performance, microkernel → better isolation), so choice impacts system correctness and non-functional guarantees.

PAGE NO.:
DATE: / /

(8) (i) PCB analysis for process switching errors.
The Process Control Block stores key state (program counter, registers, stack pointer, process state). By inspecting it after a context switch, one can detect uninitialised registers or incorrect state transitions.

(ii) When a task moves unexpectedly from Running → Waiting, the OS performs a context switch, which precisely means:
- Saving the current CPU state (registers, PC, flags, ~~stag~~ stack pointer) into the process's PCB.
- Updating the PCB with the new state (Waiting)
- Selecting the next process and loading its saved context from its PCB into the CPU.

(iii) System call type for mid-execution I/O.
Use a non-blocking asynchronous system call.

- Reason:- The process should continue execution while the OS allocates / initializes I/O resources in the background. Blocking or synchronous calls would stall the real-time execution, which is undesirable mid-process.

(6) (a) Total time = save state + load state + scheduler overhead

= 2 ms + 3 ms + 1 ms = 6 ms.

(b) Impact on multitasking performance:-

PAGE NO.:
DATE: / /

Each context switch cost 6 us of CPU time that cannot do useful work. Frequent switching reduces effective CPU throughput, increases latency for running tasks, and can degrade ~~cost~~ real-time responsiveness — so systems aim to minimize unnecessary context switches or make them cheap.

(7) Given: single-threaded execution = 40 s.
Threads per process = 2 (ideal scaling)

Estimate: with perfect parallelism,
execution time ≈ 40 s / 2 = 20 s.

Threads allow parallel execution (on multiple cores) and overlap of CPU work with I/O or waiting, improving CPU utilisation and reducing overall completion time.

(8) Gantt (arrive at 0):

- FCFS: P1 (0–5) P2 (5–8) P3 (8–16) P4 (16–22)
- SJF (non-preemptive): P2 (0–3) P1 (3–8) P4 (8–14) P3 (14–22)

Average waiting / turnaround:

- FCFS: Avg waiting = 7.25 us, Avg turnaround = 12.75 us.
- SJF: Avg waiting = 6.25 us, Avg turnaround = 11.75 us.
- RR (q = 4): Avg waiting = 10.75 us, Avg turnaround = 16.25 us.

Best Balance :- SJF — lowest avg waiting and turnaround (optimal for these jobs), though it needs burst estimates and may starve long jobs.

PAGE NO.:
DATE : / /

Q) (i) Cloud migration.

(a) A microkernel architecture is best – it keeps only minimal services in kernel space, while other services (drivers, file systems, etc) run in user space. This improves scalability (easy to extend services) and security (fault isolation, less chance of entire system crash).

(b) Virtual machines provides isolation (each VM runs its own OS, faults dont affect others), management (snapshots, migration, provisioning) and resource optimisation (hypervisor allocates CPU/ memory/storage dynamically).

(ii) Smart home system.

(a) The OS can use priority-based scheduling to ensure critical processes like intrusion detection get CPU immediately, while less urgent ones (lighting) are scheduled later.
IPC mechanisms (like message queues, shared memory, or signals) allow devices to communicate efficiently and coordinate responses.

(b) Suitable algorithms:

- Priority Scheduling (Preemptive) → ensures real-time critical tasks run first.
- Earliest Deadline First (EDF) → ideal for real-time IoT systems where tasks have strict timing deadlines.
- Rate Monotonic Scheduling (RMS) → effective for periodic IOT tasks like sensor checks.