SUMMARY

This report explains the fundamental concepts of **Operating Systems**, focusing on process management, memory management, CPU scheduling, and synchronization problems.

Part A: Process Management

Processes are independent programs in execution with their own memory, while threads are smaller units within a process that share resources. The process state model includes five stages—**New, Ready, Running, Waiting, and Terminated**—showing how processes move during execution. Two common **Inter-Process Communication (IPC)** methods are **Shared Memory**, where processes share a memory region for fast communication, and **Message Passing**, where processes exchange data through system-managed channels like pipes or sockets.

Part B: Memory Management

Paging and **Segmentation** are two techniques for memory allocation. Paging divides memory into fixed-size pages, while segmentation divides it into variable-sized segments based on logical divisions like code or data. The **Translation Lookaside Buffer (TLB)** is a small, fast cache that stores recent page table entries to speed up address translation. A higher TLB hit rate results in faster memory access. The **Effective Memory Access Time (EMAT)** was computed as **130 nanoseconds**, showing how TLB improves performance.

Part C: CPU Scheduling

First-Come, First-Served (FCFS) and **Round Robin (RR)** are two CPU scheduling algorithms. FCFS executes processes in the order they arrive but can lead to long waiting times for shorter jobs. Round Robin gives each process equal time slices (quantums), improving fairness and response time for interactive systems. Based on the example given, both algorithms had similar **average waiting times** of about **6.67 ms**, though Round Robin is better suited for multitasking environments.

Part D: Synchronization Problems

Three classic synchronization problems illustrate process coordination issues:

- The **Dining Philosophers Problem** shows how deadlock can occur when multiple processes share limited resources, solved using semaphores or by limiting access.
- The **Producer-Consumer Problem** involves managing a shared buffer between producers and consumers using semaphores to prevent race conditions and ensure proper resource use.
- The **Reader-Writer Problem** deals with shared data access, allowing multiple readers but ensuring writers have exclusive control to maintain data consistency.