## Week 7

### 1. Operating System Overview

* Definition: A program that controls execution of application programs and acts as an interface between applications and hardware.
* Objectives: Convenience, Efficiency, Evolvability.

Functions:

* Exploits hardware resources of processors.
* Provides services to users.
* Manages memory and I/O devices.

### 2. Computer System Components

Basic Elements

* Processor (CPU): Executes instructions, processes data.
* Main Memory: Stores programs and data, volatile in nature.
* I/O Modules: Handles communication between the computer and external devices (e.g., disks, terminals).
* System Bus: Facilitates communication between CPU, memory, and I/O modules.
* Instruction Execution
* Instruction Cycle: Fetch → Decode → Execute
* Interrupts: Allow efficient CPU utilization by handling slow I/O operations asynchronously.

Types of Interrupts:

* Program Interrupts: Errors like division by zero.
* Timer Interrupts: OS functions like scheduling.
* I/O Interrupts: Device completion signals.
* Hardware Interrupts: Power failure, memory errors.

### 3. Memory Architecture

Memory Hierarchy

* Factors affecting design: Capacity, speed, cost.
* Trade-offs:
* Faster access = higher cost per bit.
* Higher capacity = lower cost per bit but slower speed.
* Principle of Locality: Frequently accessed data should be stored in faster memory.

Cache Memory

* Purpose: Provides high-speed access to frequently used data.
* Levels:
* L1 (Fastest, Smallest),
* L2 (Moderate speed and size),
* L3 (Larger but slower than L1/L2).

Virtual Memory & Paging

* Virtual Memory: Expands available memory by using disk storage.
* Paging: Divides memory into fixed-sized pages, dynamically mapping virtual to physical addresses.

### 4. Input/Output (I/O) Techniques

* Programmed I/O: CPU constantly checks for device readiness (inefficient).
* Interrupt-Driven I/O: CPU issues commands and performs other tasks until notified.
* Direct Memory Access (DMA): Data transfers between memory and I/O without CPU intervention, improving efficiency.

### 5. Symmetric & Multicore Processing

* Symmetric Multiprocessing (SMP)
* Multiple CPUs sharing the same memory and I/O.

Advantages:

* Increased performance.
* Higher availability (failure of one CPU doesn’t halt system).
* Incremental growth (add processors as needed).
* Multicore Processors
* Multiple processing cores on a single chip.
* Benefits: Parallel processing, improved efficiency, reduced power consumption.

### 6. Evolution of Operating Systems

* Serial Processing: No OS, manually operated.
* Batch Systems: Job scheduling for efficiency.
* Multiprogramming: Allows multiple programs to run concurrently.
* Time-Sharing Systems: Interactive multi-user environments.
* Key OS Developments
* Process management (multitasking).
* Memory management (virtual memory, paging).
* Information security (user authentication, data protection).

### 7. Process & Resource Management

Process Concept

* Definition: A running program instance with its own state and resources.
* Components:
  + Executable program.
  + Data (variables, workspace).
  + Execution context (registers, priority, I/O status).

Process Scheduling

* Short-Term Scheduler: Allocates CPU time to processes.
* Long-Term Scheduler: Manages system workload.

Memory Management Responsibilities

* Process isolation.
* Automatic allocation & deallocation.
* Protection & access control.
* Support for modular programming.

### 8. OS Architectures & Fault Tolerance

Architectural Approaches

* Microkernel: Minimal OS kernel with modular services.
* Multithreading: Multiple execution threads within a process.
* Distributed OS: Manages resources across multiple machines.

Fault Tolerance

* Definition: Ability to maintain operation despite failures.
* Methods of Redundancy:
  + Physical (hardware backups).
  + Temporal (repeating operations).
  + Information (error correction mechanisms).

### Conclusion

* OS is essential for managing hardware and software efficiently.
* Advancements in OS design continue to improve performance, security, and fault tolerance.
* Future trends include increased parallelism, virtualization, and AI-driven optimizations.