# COMP-346 Midterm Notes

Operating Systems (Concordia University)

# LECTURE 1: INTRO AND OS OVERVIEW

## 1.1 What Is a Operating System?

An operating System a program that manages the computer hardware while acting as **an intermediary between the computer user and computer hardware**. It is also a basis for applications programs. It is necessary to allocate hardware resources, manage memory, security, etc. on a computer as **efficient and convenient** as possible.

“The one program running at all times on the computer” is the kernel.

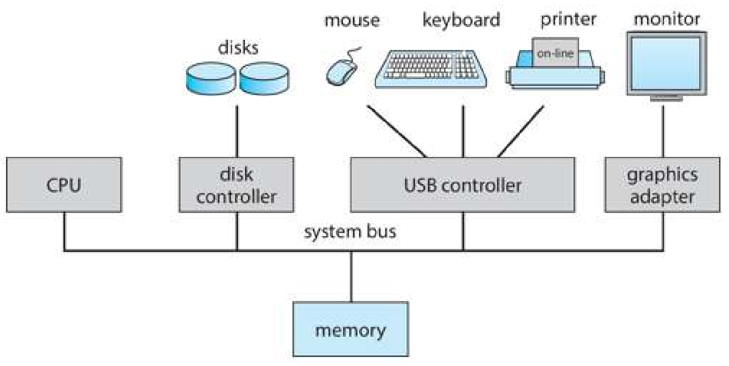
A system program (ships with the operating system, but **not** part of the kernel).

## The Computer System

4 Main Components:

1. Hardware: CPU, Memory, I/O Devices
2. Operating System
3. Application Programs: Word processors, Compilers, Web browsers, etc.
4. Users: People, Machines, Other Computers

## 1.2 Computer System Operation



One or more CPUs, device controllers connect through common bus providing access to shared memory. Concurrent execution of CPUs and devices competing for memory cycles.

Device controllers oversee particular device type. They each have a local buffer. They each have an OS **device driver.**

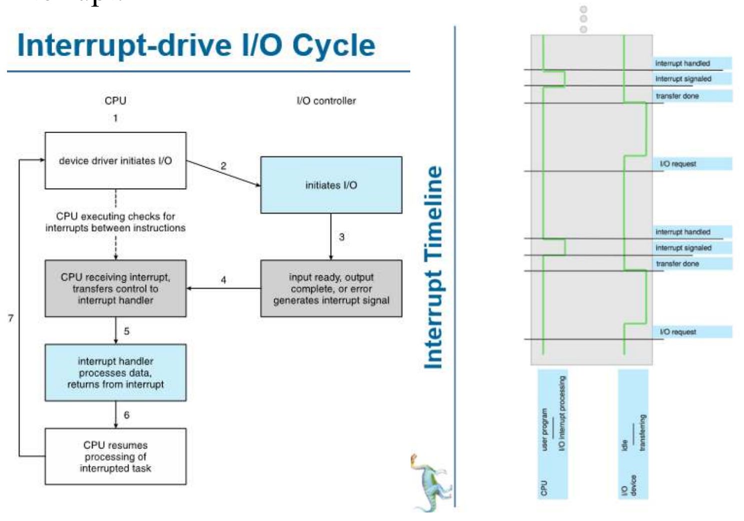
CPU moves data from/to main memory to/from local buffers. I/O is from the device to local buffer of controller. Device controller informs CPU that it has finished its operation by causing an **interrupt**.

### 1.2.1 Interrupts

The occurrence of an event is usually signalled by an **Interrupt** from the hardware of software. **A trap** or **exception** is a software-generated interrupt. Operating system is **interrupt driven.**

When the CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location (**through the interrupt vector**), which contains the starting address where the **Service Routine** of the interrupt is located. The interrupt service routine executes. On completion the CPU resumes its previous (interrupted) computation.

**Handling:** The OS preserves the CPU state by storing the registers and the program counter. Determines which type of interrupt occurred. Separate segments of code determine what action should be taken for each type of interrupt.


Interrupt-drive I/O Cycle
CPU
I/O controller
1
device driver initiates I/O
2
initiates I/O
CPU executing checks for interrupts between instructions
3
CPU receiving interrupt, transfers control to interrupt handler
4
input ready, output complete, or error generates interrupt signal
5
interrupt handler processes data, returns from interrupt
6
CPU resumes processing of interrupted task
7
Interrupt Timeline
interrupt handled
interrupt signaled
transfer done
I/O request
interrupt handled
interrupt signaled
transfer done
I/O request
CPU
user program
I/O interrupt processing
I/O device
idle
transferring

## 12 I/O Structure

Two ways of handling I/O:

1. After I/O starts, control returns to user program only upon I/O completion:

Wait instruction idles the CPU until next interrupt. Wait loop. At most one I/O request at a time (no simultaneous I/O processing)

2. After I/O starts, control returns to user program without waiting for I/O completion:

**System Call**: Request to the OS to allow user to wait for I/O completion (trigger interrupt).

**Device-status table** contains entry for each I/O device indicating its type, address and state. OS indexes into I/O device table to determine device status and to modify table entry to include interrupt.

### 1.2.2 Storage Structure

**Main Memory:**

Random Access Memory (RAM): **Volatile**, RAM in the **form of dynamic RAM**. Only large storage that the **CPU can access directly**.

**Secondary Storage:** Extension of main memory that **provides large non-volatile storage capacity.**

**Hard Disk Drives:** Rigid metal or glass platters covered with magnetic recording material. Disk surface is logically divided into **tracks**, which are subdivided into **sectors**. The **disk controller** determines the logical interaction between the device and the computer.

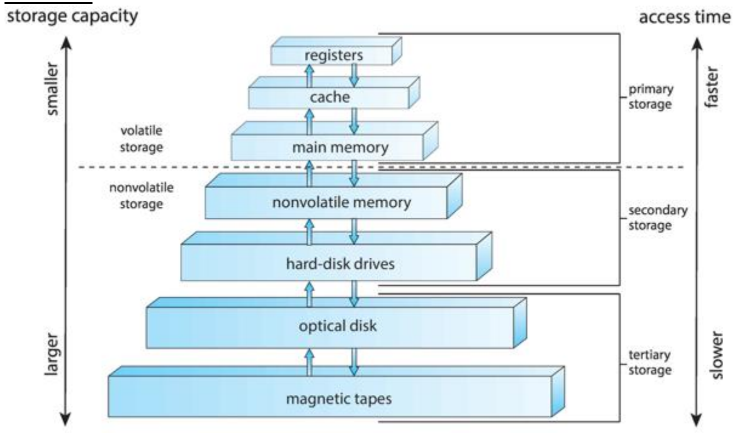
**Non-volatile memory (NVM) devices**– faster than hard disks. Non-volatile. Various technologies. Becoming more popular as capacity and performance increases, price drops.

### Storage Hierarchy

Storage systems organized in hierarchy by Speed, Cost and Volatility

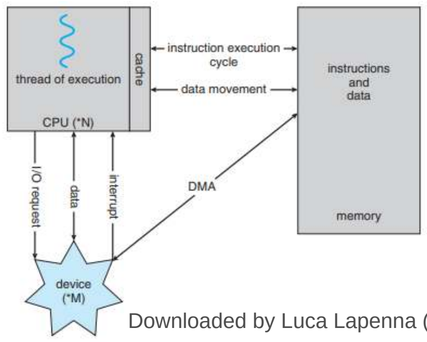
**Caching:** Copying information into faster storage system. Main memory can be viewed **as a cache** for secondary memory.

**Device Driver** for each device controller to manage I/O. Provides uniform interface between controller and kernel.



## 1.2.3 I/O Structure

**Direct Memory Access (DMA):** Used for high-speed I/O devices able to transmit information at close to memory speeds. Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention. Only one interrupt is generated per block, to tell the device driver that the operation has completed.



## 1.3 Computer-System Architecture

Some important definitions: **Processor:** A physical chip that contains one or more CPUs.

**CPU:** The hardware that executes instructions.

**Core:** The basic computation unit of the CPU.

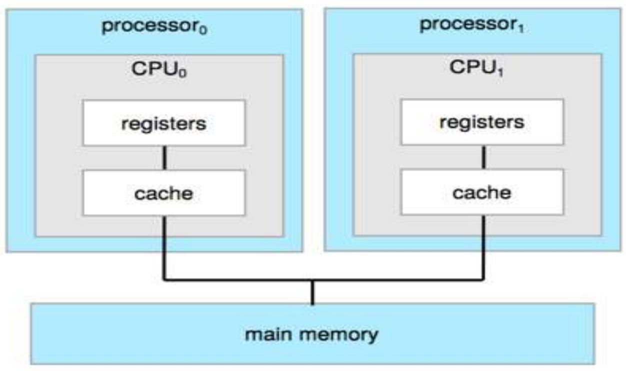
**1.3.1 Single-Processor Systems:** Back in the day, most systems use a single general-purpose processor, with the addition of **special-purpose processors.** These special purpose processors run a limited instruction set and **do not** run processes.

**1.3.2 Multiprocessor Systems:** AKA parallel system or tightly-coupled systems. Advantages:

1. Increased throughput
2. Economy of scale
3. Increased reliability

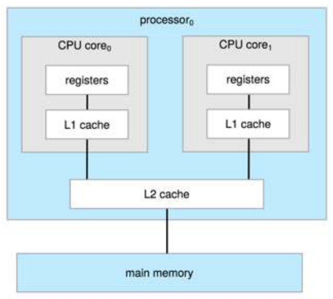
Their a two types of multiprocessor systems:

1. **Symmetric multiprocessing (SMP)**: Each processor performs all tasks, including operating-system functions and user processes.

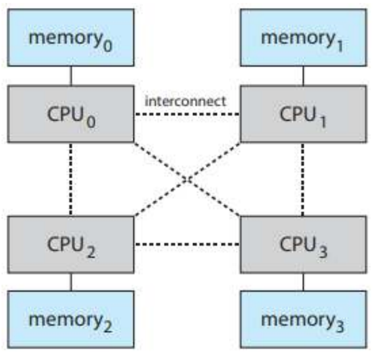


2. **Asymmetric multiprocessing:** Each processor is assigned a specified task.

**Multicore:** Including multiple computing cores on the same CPU. Example of a Dual-Core Design.

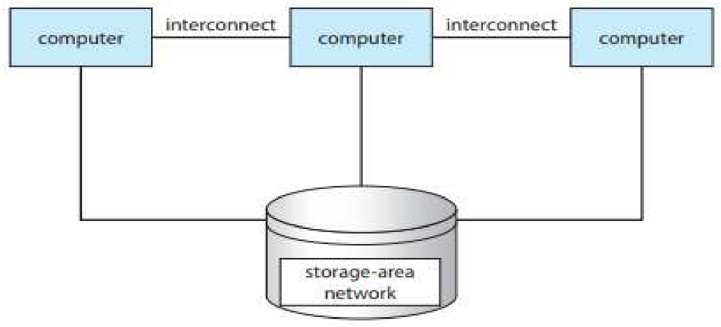


**Non-Uniform Memory Access System (NUMA):** each CPU (or group of CPUs) with its own local memory that is accessed via a small, fast local bus. The CPUs are connected by a shared system interconnect, so that all CPUs share one physical address space. Advantage: Scale more effectively with more processors added.



### 1.3.3 Clustered Systems:

Like multiprocessor systems, but multiple systems (or nodes) are working together. It provides a **high-availability service** which survives even if one system fails. Share storage via a **storage-area network (SAN)**. Some can provide **high-performance computing (HPC)** for applications**,** but these applications must be written to use **parallelization (**divides a program into separate components that run in parallel on individual cores in a computer or computers in a cluster**).** **Distributed lock manager (DLM):** To provide shared access, the system must also supply access control and locking to ensure that no conflicting operations occur.



Two types of clustering:

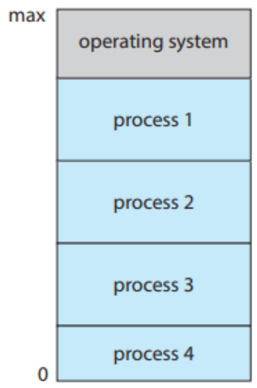
1. **Asymmetric clustering**: has one machine in a **hot-standby mode** while the other is running the applications. The hot-standby machine monitors the active server and jumps in place in case the active server fails.
2. **Symmetric clustering:** Two or more hosts are running applications and are monitoring each other. **More efficient** as it uses all available hardware.

## 1.4 Operating-System Operations

**Bootstrap:** Initial program that runs when computer is powered up or rebooted. Stored in the ROM or EPROM, generally known as firmware. It locates the OS kernel and loads it into memory. Starts **system daemons** (services provided outside of the kernel).**,** which runs the entire time the kernel is running. At this point, the OS awaits for an event to trigger an interrupt. An interrupt can be a software error (division by 0), a request for operating system service (**system call**) or other process problems include infinite loop, processes modifying each other or the operating system.

### 1.4.1 Multiprogramming and Multitasking

Single program cannot, in general, keep either the CPU or the I/O devices busy at all times. **Multiprogramming (Batch System)** organizes jobs (code and data) so CPU always has one to execute. A subset of total jobs in system is kept in memory. One job selected and run via **job scheduling**. When job has to wait (for I/O for example), OS switches to another job. Here is a memory layout of multiprogramming system ->



**Multitasking** is a logical extension of batch systems. The CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing. Here are some characteristics of multitasking:

- **Response time** should be < 1 second
- Each user has at least one program executing in memory ⇨ **process**
- If several jobs ready to run at the same time ⇨ **CPU scheduling**
- If processes don' t fit in memory, **swapping** moves them in and out to run
- **Virtual memory** allows execution of processes not completely in memory

### 1.4.2 Dual-Mode Operation

We need to separate modes of operation: user mode and kernel mode. **Dual-mode** operation allows OS to protect itself and other system components. **Mode bit**, provided by hardware, provides ability to distinguish when system is running kernel mode (0) or user mode (1). How do we guarantee that user does not explicitly set the mode bit to kernel? System call changes mode to kernel, return from call resets it to user. Some instructions designated as **privileged**, only executable in kernel mode.

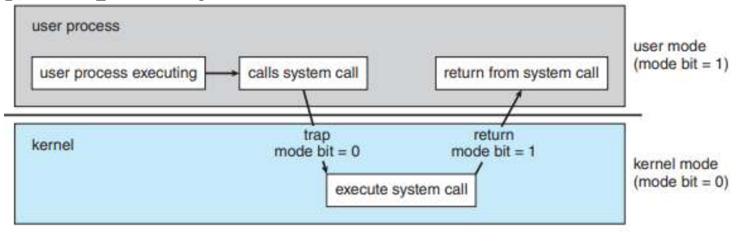


**Figure 1.13** Transition from user to kernel mode.

### 1.4.3 Timer

**Timer** to prevent infinite loops or other process hogging resources. It is set to interrupt the computer after some time period. Period can be fixed or variable. Keeps a counter that is decremented by the physical clock. Operating System set the counter (privileged instruction). When counter zero, generates an interrupt. Set up before scheduling process to regain control or terminate program that exceeds allotted time.

## 1.5 Resource Management

### 1.5.1 Process Management

A **process** is a program in execution. It is a unit of work within the system. Program is a **passive entity**; Process is an **active entity**.
Process needs **resources** to accomplish its task: CPU, memory, I/O, files. In addition to resources, various **initialization data (input)** may be passed along. Once process terminates, the OS will reclaim any reusable resources.
A single-threaded process has one **program counter** specifying location the next instruction to execute, process executes instruction sequentially, one at a time, until completion. A multi-threaded process has one program counter per thread.
Typically system has many processes, some user, some operating system running concurrently on one or more CPUs. Concurrency by multiplexing the CPUs among the processes/threads.
The OS is responsible for the following process management activities:
1. Creating and deleting both user and system processes
2. Suspending and resuming processes
3. Providing mechanisms for process synchronization
4. Providing mechanisms for process communication
5. Providing mechanisms for deadlock handling