UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

DCISM
DEPARTMENT OF COMPUTER, INFORMATION SCIENCES AND MATHEMATICS

1

CS 3104
OPERATING SYSTEMS

CHAPTER 1

WHAT IS AN OPERATING SYSTEM?

- A software that manages a computer's hardware
- Provides an environment for application programs to run
- Acts as an intermediary between the computer user and the computer hardware
- Varies in accomplishing its tasks in a wide variety of computing environments

WHAT IS AN OPERATING SYSTEM?

- A **software** that includes the following:
  - the always running **kernel**
  - **middleware** frameworks that ease application development and provide features
  - **system programs** that aid in managing the system while it is running

- **Kernel:** the **one program running at all times on the computer**

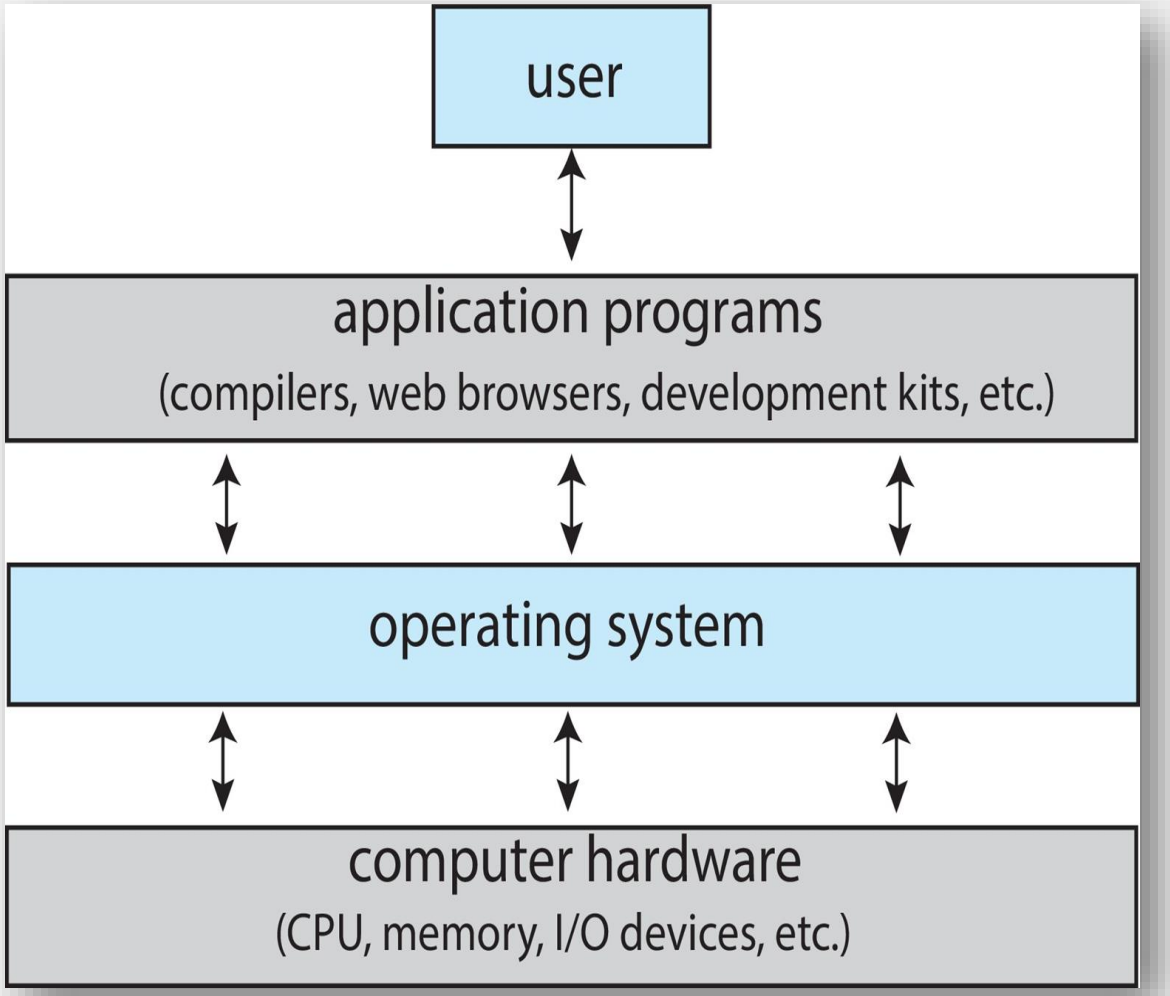
- **Middleware:**
  - a **set of software frameworks** that **provide additional services to application developers**

COMPUTER SYSTEM STRUCTURE

Computer System can be divided into four components:

- **Hardware:**
  - Provides basic computing resources: CPU, Memory, I/O devices
- **Operating System**
  - Controls and coordinates the use of hardware among various applications and users
- **Application Programs**
  - Define the ways in which the system resources are used to solve the computing problems of the users
  - Word processors, compilers, web browsers, database systems, video games
- **Users**
  - People, machines, other computers

ABSTRACT VIEW OF COMPONENTS OF COMPUTER



WHAT OPERATING SYSTEMS DO?

Users View

- Users want convenience
- Ease of use
- Good performance and security
- Don’t care about resource utilization

WHAT OPERATING SYSTEMS DO?

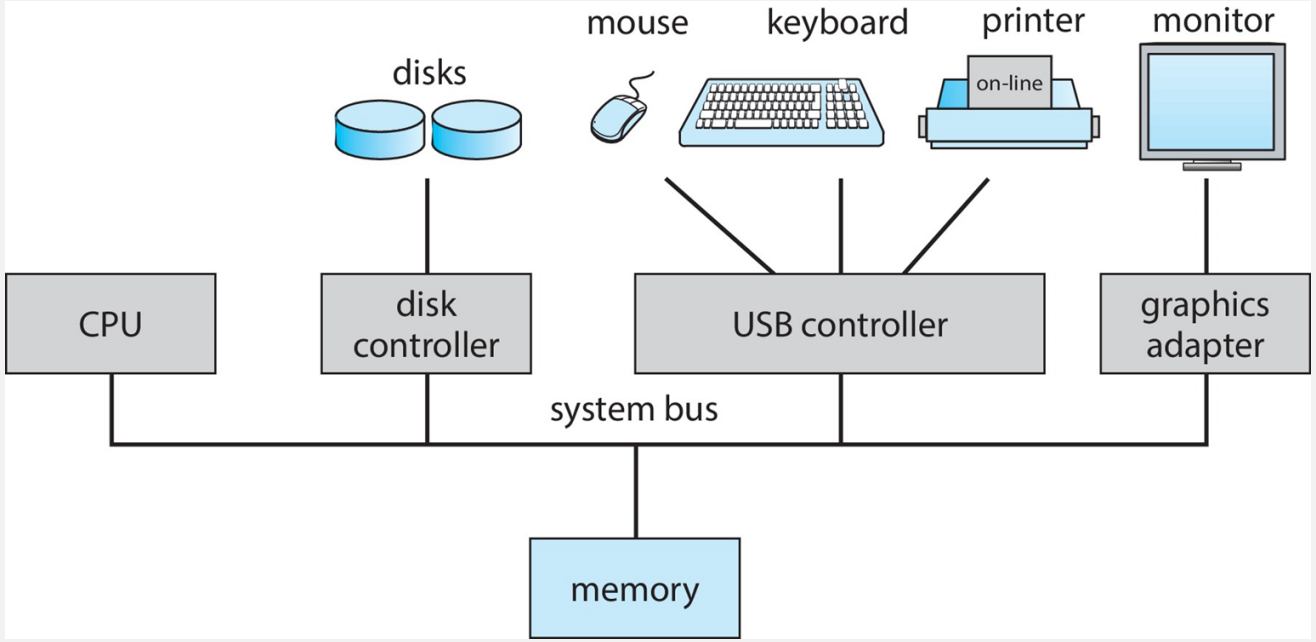
System View

- **OS as a Resource Allocator:**
  - The program most intimately involved with the hardware
  - Acts as the manager of the computer system's resources: CPU, Memory, I/O
  - Decides how to allocate the resources to specific programs and users so that it can operate the computer system efficiently and fairly.

- **OS as a Control Program:**
  - Manages the execution of user programs to prevent errors and improper use of the computer
  - Concerned with the operation and control of I/O devices

COMPUTER SYSTEM ORGANIZATION

A Typical PC Computer System



- **Computer-System Operation**
  - **One or more CPUs** and **device controllers** connect through the **common bus providing access to shared memory**
  - **Concurrent** execution of CPUs and devices are **competing for memory cycles**

COMPUTER-SYSTEM OPERATION

- **I/O** devices and the **CPU** can execute **concurrently**
- Each **device controller** is in charge of a specific device type
- Each **device controller** has a local buffer storage
- Each **device controller** type has an operating system **device driver** to manage it
- **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**

COMPUTER-SYSTEM OPERATION

- **Device Controller**
  - a hardware unit attached to the I/O bus of the computer and works like an interface between a device and a device driver
  - responsible for moving the data between the peripheral devices that it controls and its local buffer storage

- **Device Driver**
  - a specialized software program running as part of the OS that interacts with a device attached to a computer.
  - driver understands the device controller and provides the rest of the operating system with a uniform interface to the device

- **Interrupt** (sometimes referred to as a **trap** or **exception**)
  - a hardware signal from a device to the CPU which tells the CPU that the device needs attention and that the CPU should stop performing what it is doing and respond to the device

COMMON FUNCTIONS OF INTERRUPTS

- Interrupt transfers control to the **interrupt service routine** (**ISR**), generally, through the use of the **interrupt vector table**, which contains the addresses of all the service routines.
- Interrupt architecture must save the address of the interrupted instruction and the contents of the processor status register, so that it can restore them after servicing the interrupt.
- Incoming interrupts are disabled while another interrupt is being processed to prevent a lost interrupt.
- A software-generated interrupt may be raised either by an error or a user request (sometimes called a **trap** or **exception**).
- An OS is interrupt driven.

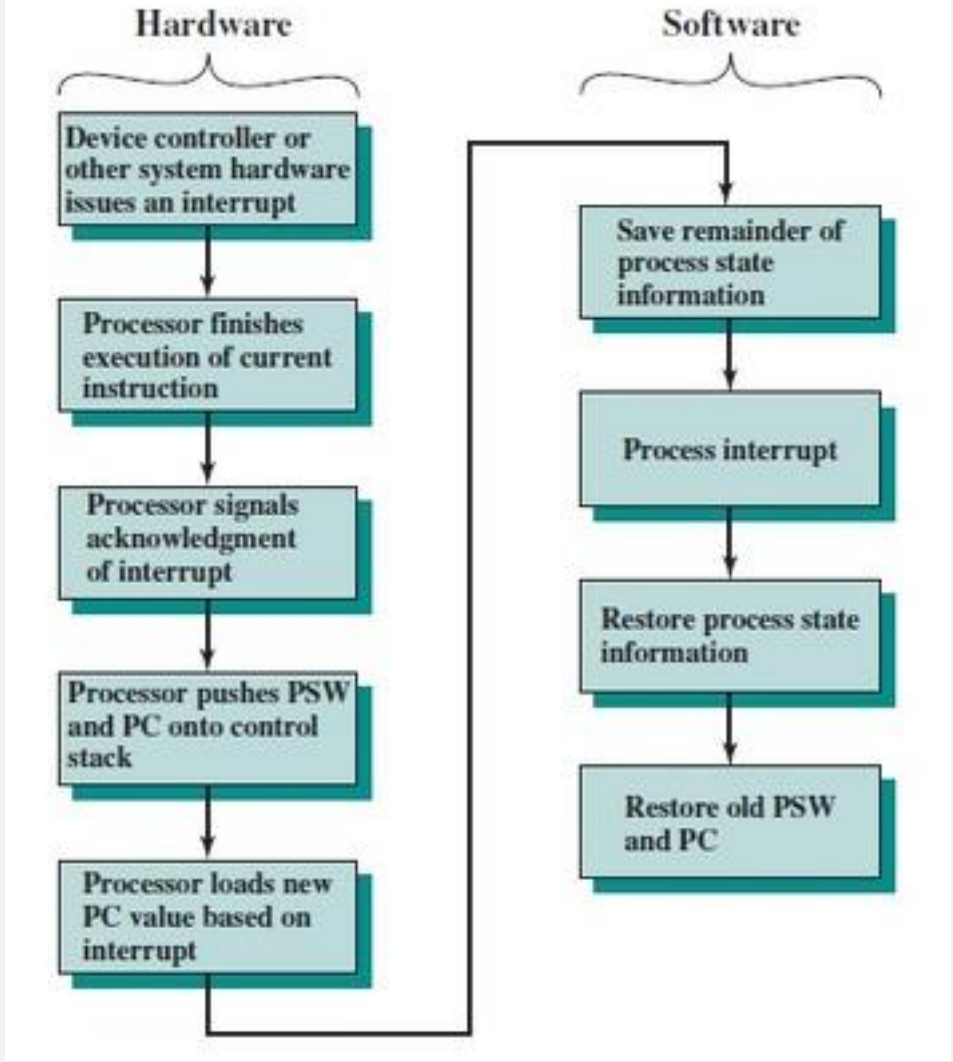
INTERRUPT

- **Interrupt Vector**
  - It is an address, stored as data, located at a particular memory location, which points to where the interrupt service routine can be found.
  - It is an address that informs the interrupt handler as to where to find the ISR.

- **Interrupt Vector Table**
  - It is a table of memory addresses of interrupt/exception handler routines.

- **Interrupt Service Routine (ISR)**
  - It is also known as **Interrupt Service Procedure** or **Interrupt Handler**
  - It is the software that attends to the cause of the interrupt.
  - It is a software routine that hardware invokes in response to an interrupt.

SIMPLE INTERRUPT PROCESSING

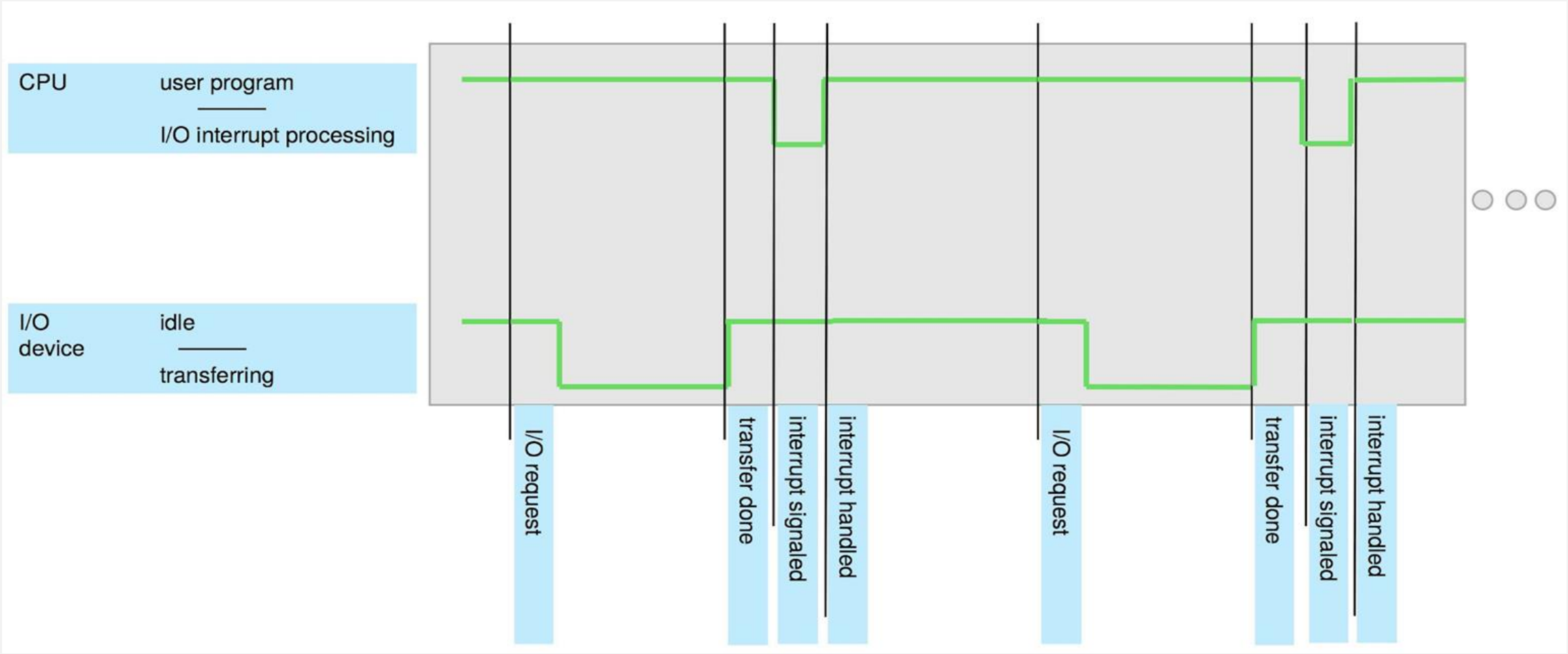


- **Program Status Word (PSW):**
  - A CPU register that contains **status bits** that **reflect the current CPU state**.

- **Program Counter (PC):**
  - A **special-purpose register** that is **used by the processor to hold the address of the next instruction to be executed**.

INTERRUPT TIMELINE

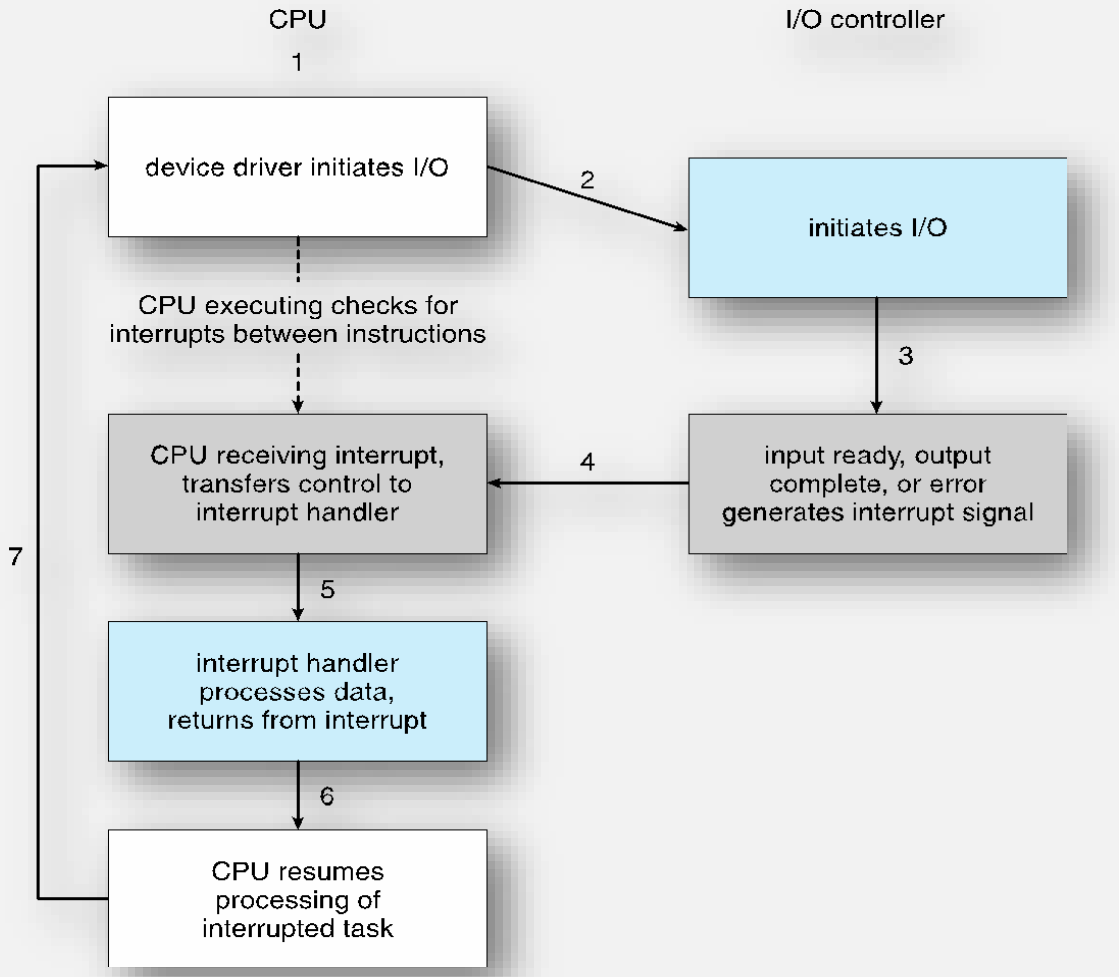
FOR A SINGLE PROGRAM DOING OUTPUT



BASIC INTERRUPT MECHANISM

- The **CPU hardware** has a **wire** called the **interrupt-request line** that the CPU senses after executing every instruction.
- When the **CPU detects** that a **controller has asserted a signal** on the **interrupt-request line**, it **reads the interrupt number** and **jumps** to the **interrupt-handler routine** by using that interrupt number as an **index into the interrupt vector**.
- It then **starts execution at the address associated with that index**.
- The **interrupt handler saves any state it will be changing during its operation**, **determines the cause of the interrupt**, **performs the necessary processing**, **performs a state restore**, and **executes a return from interrupt instruction to return the CPU to the execution state prior to the interrupt**.
- The **device controller** ***raises*** **an interrupt by asserting a signal on the interrupt request line**, the **CPU** ***catches*** **the interrupt and** ***dispatches*** **it to the interrupt handler**, and the **handler** ***clears*** **the interrupt by servicing the device**.

INTERRUPT-DRIVEN I/O CYCLE

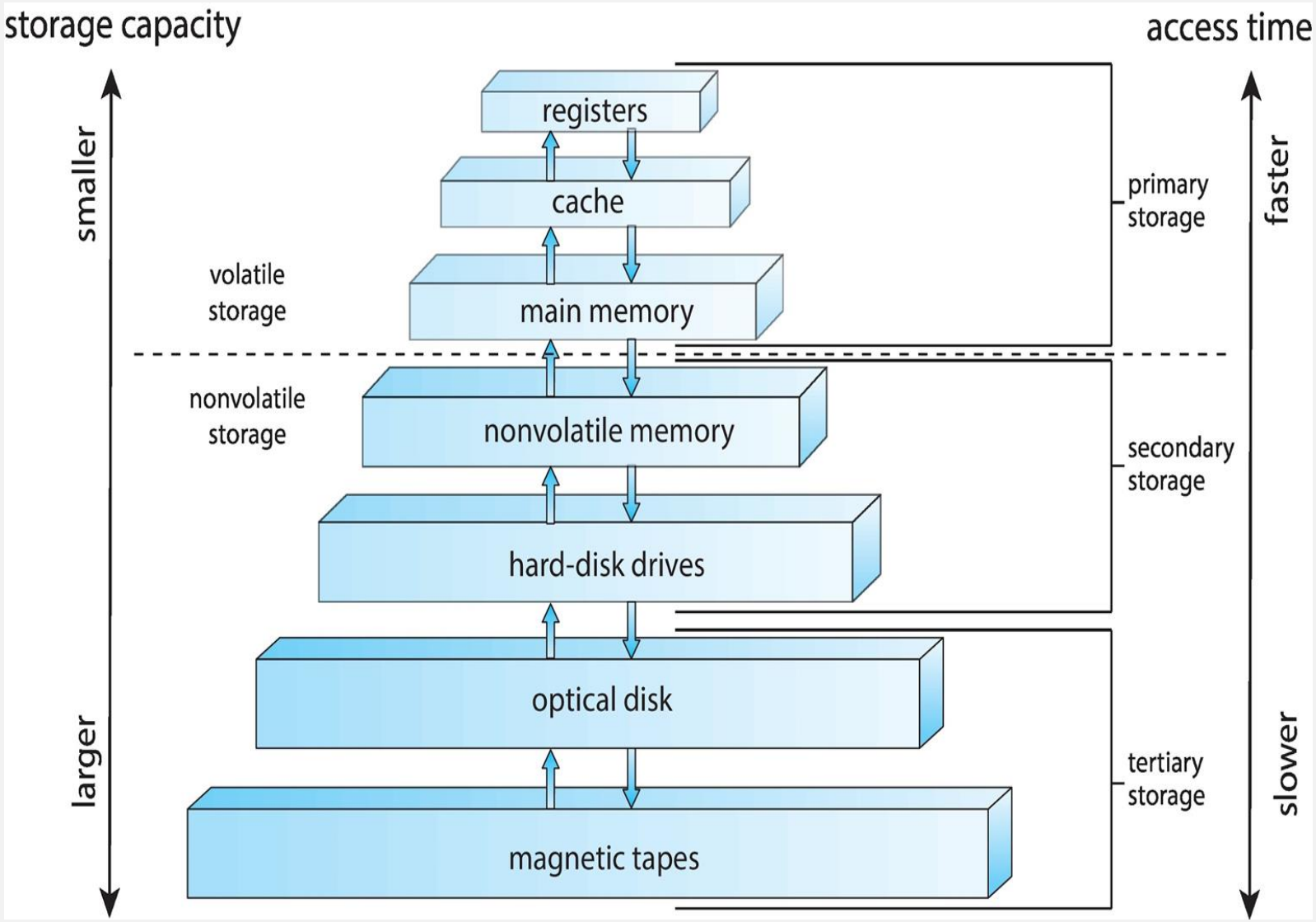


1. A device driver initiates an I/O request on behalf of a process.
2. The device driver signals the I/O controller for the proper device, which initiates the requested I/O.
3. The device signals the I/O controller that is ready to retrieve input, the output is complete or that an error has been generated.
4. The CPU receives the interrupt signal on the interrupt-request line and transfer control over the interrupt handler routine.
5. The interrupt handler determines the cause of the interrupt, performs the necessary processing and executes a **"*return from*"** interrupt instruction.
6. The CPU returns to the execution state prior to the interrupt being signaled.
7. The CPU continues processing until the cycle begins again.

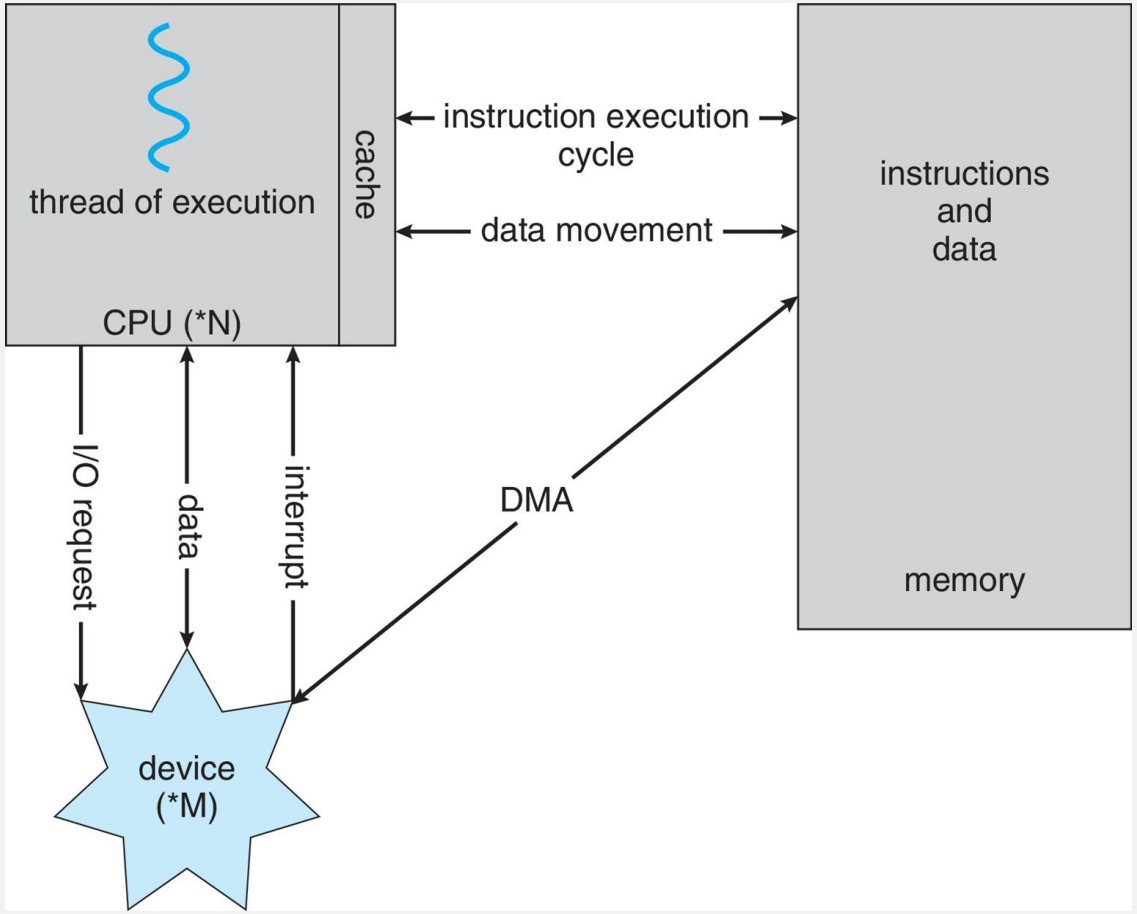
STORAGE STRUCTURE

- **Main Memory**
  - physical memory that is internal to the computer that is a computer's short-term storage.
  - the only large storage media that the CPU can access directly
  - random access and volatile
  - Random-Access Memory in the form of Dynamic Random-Access Memory (**DRAM**)
- **Secondary storage**
  - extension of main memory that provides large **nonvolatile** storage capacity
- **Hard Disk Drives** (**HDD**)
  - rigid metal or glass platters covered with magnetic recording material
  - disk surface is logically divided into **tracks**, which are subdivided into **sectors**
  - **disk controller** determines the logical interaction between the device and the computer
- **Non-Volatile Memory (NVM)**
  - storage devices faster than hard disks and nonvolatile
  - various technologies
  - becoming more popular as capacity and performance increases, price drops

STORAGE HIERARCHY



HOW A MODERN COMPUTER SYSTEM WORKS?



A von Neumann Architecture

DIRECT MEMORY ACCESS STRUCTURE

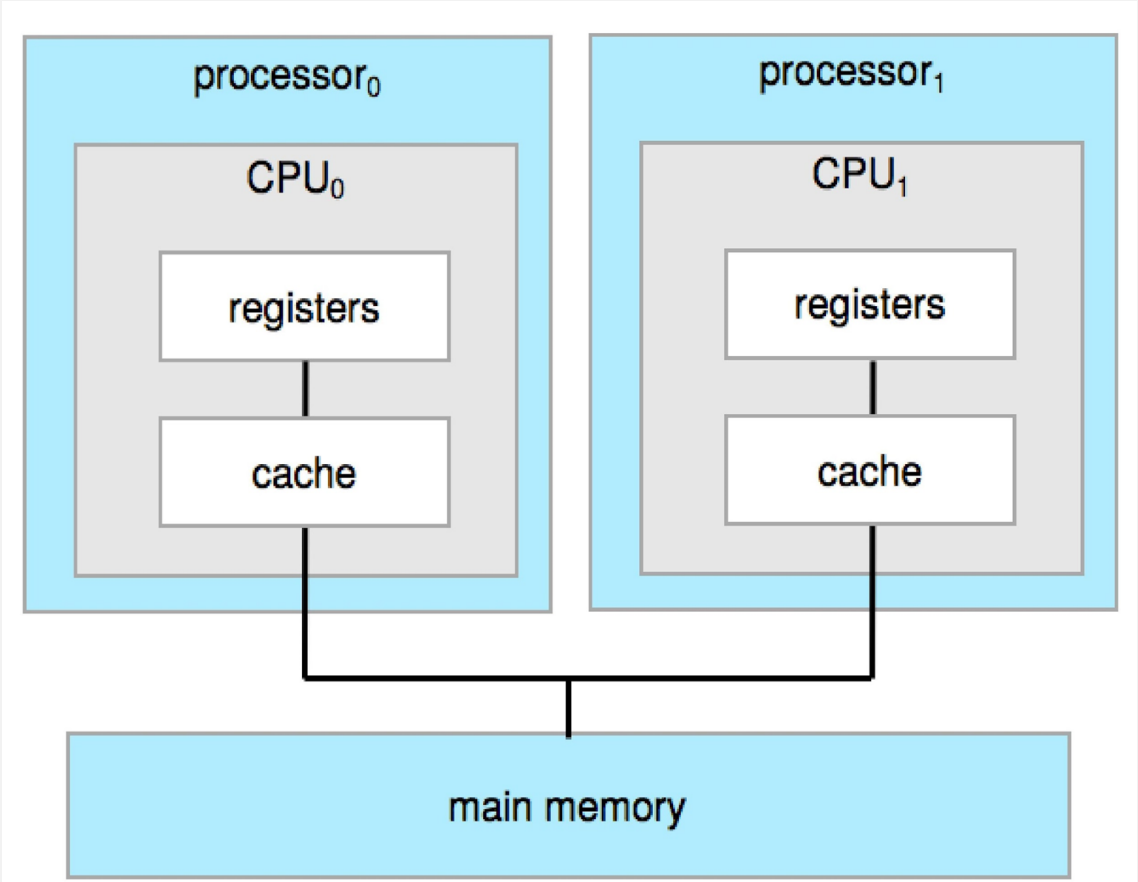
- Used for **high-speed I/O devices** able **to transmit information at close to memory speeds**
- **Device controller** transfers **blocks of data from buffer storage directly to main memory without CPU intervention**
- **Only one interrupt is generated per block**, rather than the one interrupt per byte

COMPUTER-SYSTEM ARCHITECTURE

- **Single-Processor Systems**
  - Computer systems that have a single processor (one CPU with a single processing **core**)
    - **Core:** the component that executes instructions and registers for storing data locally
  - Most systems use a single general-purpose processor
  - Most systems have special-purpose processors as well

- **Multiprocessor Systems**
  - Systems that have two (or more) processors, each with a single-core CPU
  - Systems that are growing in use and importance
  - Also known as **parallel systems** or **tightly-coupled systems**
  - **Two types:**
    - **Asymmetric Multiprocessing:** each processor is assigned a specific task
    - **Symmetric Multiprocessing:** each processor performs all tasks

SYMMETRIC MULTIPROCESSING (SMP) ARCHITECTURE



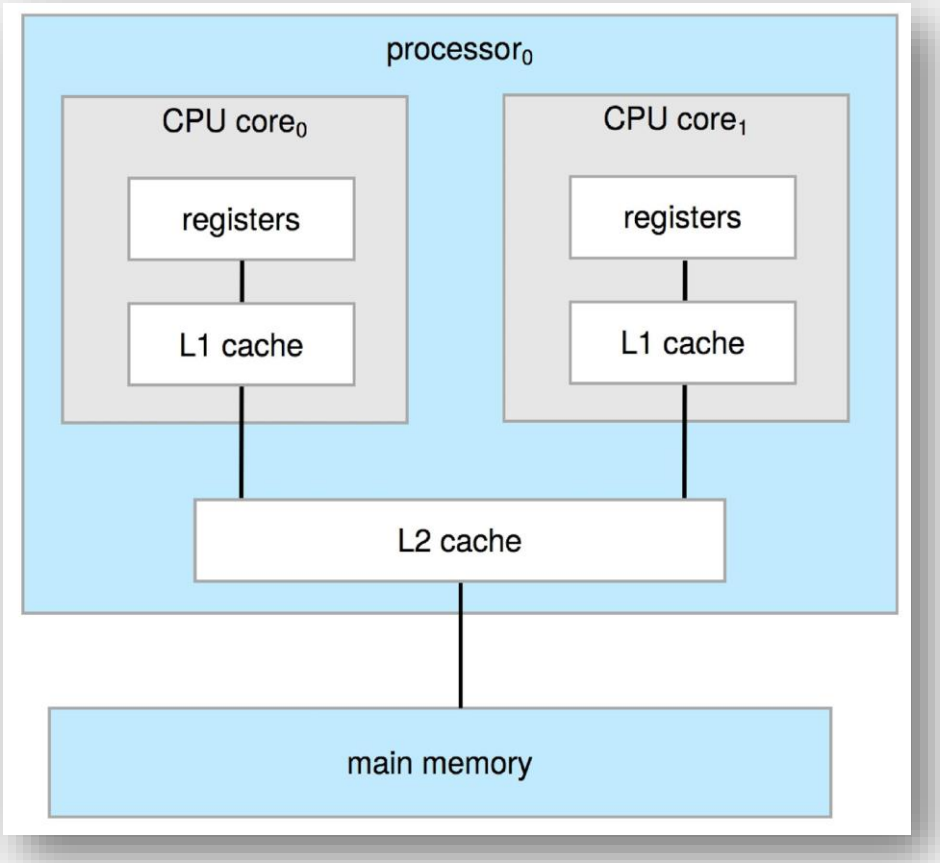
- A **multiprocessor system** in which **each peer CPU processor performs all tasks**, including **operating-system functions** and **user processes**.

A DUAL-CORE DESIGN

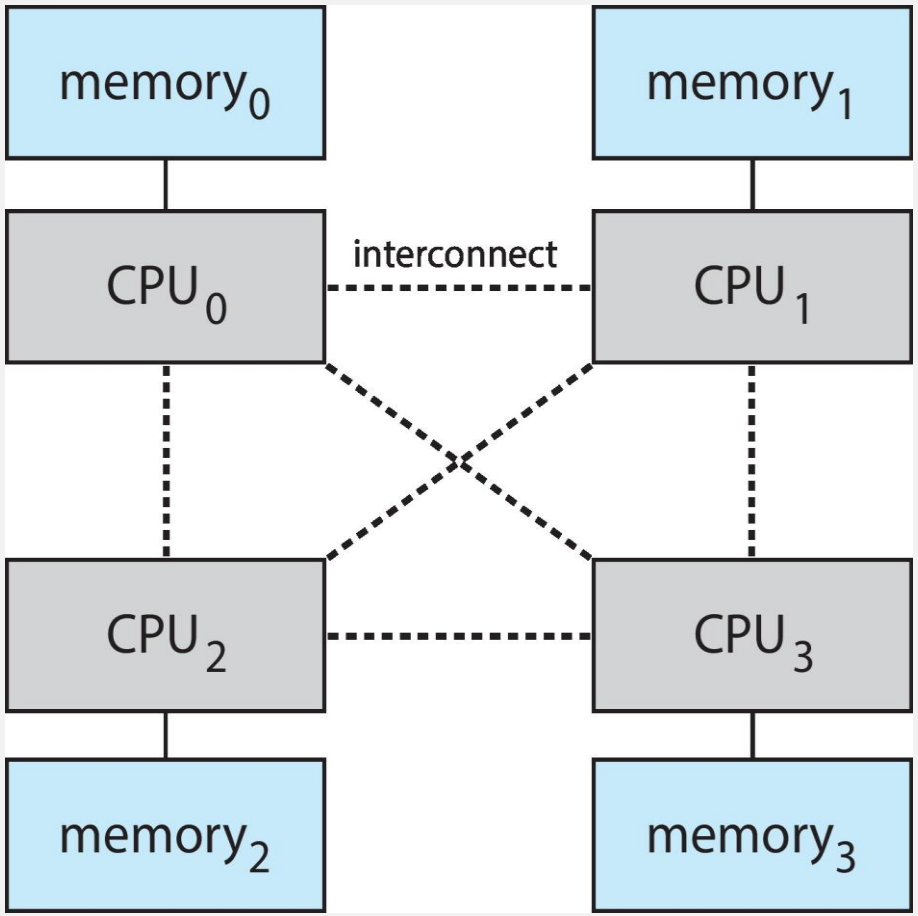
- **Multicore Systems:**
  - **multiple computing cores reside on a single chip**
  - can be **more efficient** than **multiple chips with single cores** because **on-chip communication** is **faster than between-chip communication**

- **Dual-Core Processor:**
  - A **single chip** that contains **two distinct processors** that **work simultaneously**

- **Core:** the **basic computation unit** of the CPU



NON-UNIFORM MEMORY ACCESS (NUMA) SYSTEM

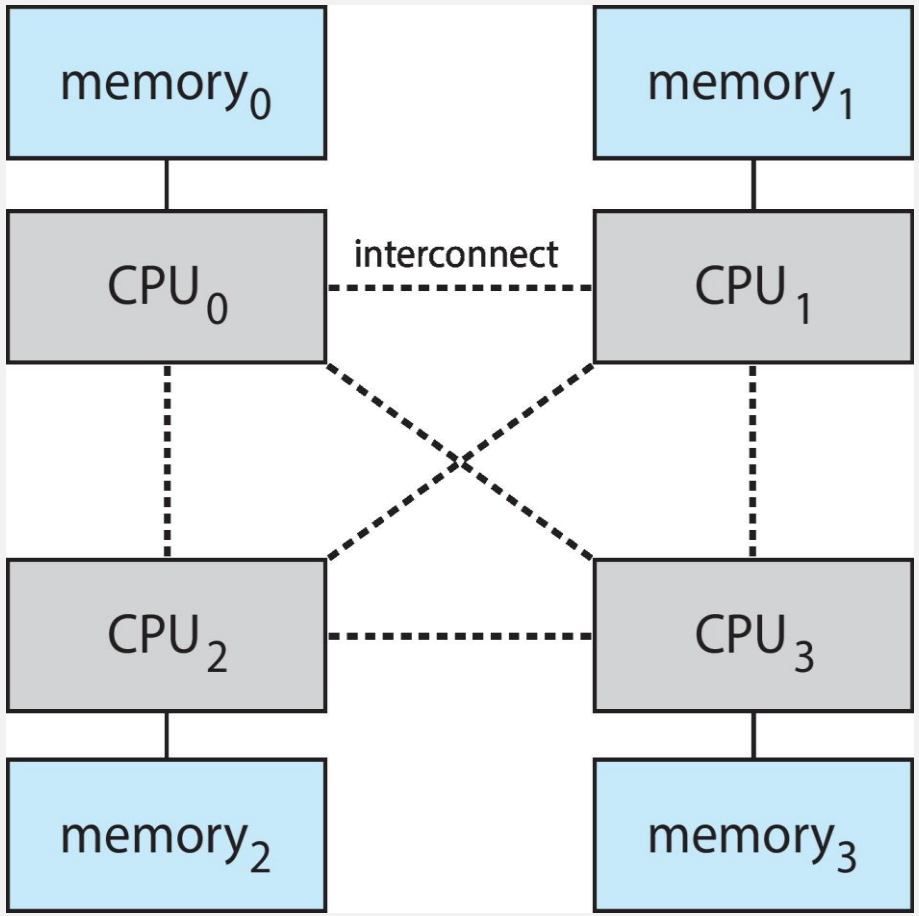


NUMA Multiprocessing Architecture

- **Fact:**
  Adding additional CPUs to a multiprocessor system will **increase computing power**

- **Issue:**
  The **above concept does not scale very well**, and **once too many CPUs are added**, **contention for the system bus** becomes a **bottleneck** and **performance begins to degrade**.

NON-UNIFORM MEMORY ACCESS (NUMA) SYSTEM

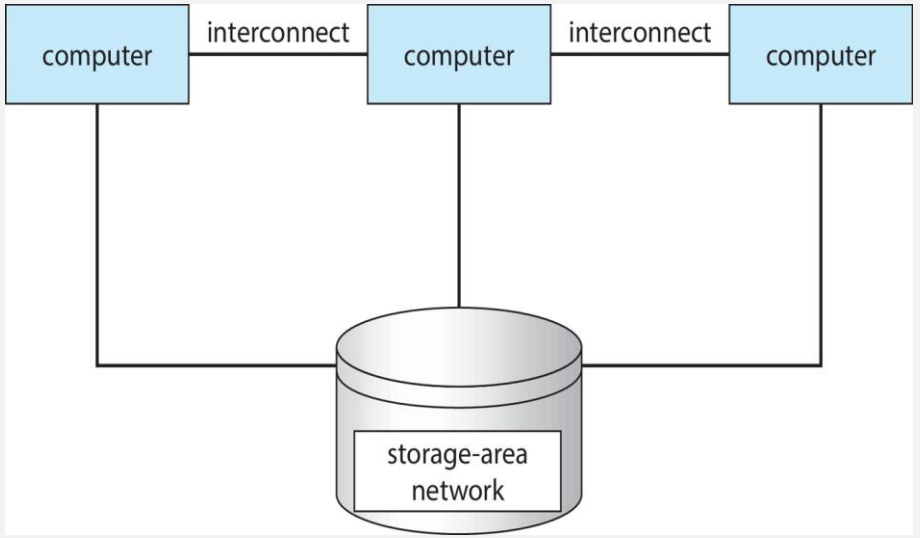


NUMA Multiprocessing Architecture

- **Solution (Alternative Approach): NUMA**
  - **Provide 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**

CLUSTERED SYSTEMS

- Another **type of multiprocessor systems,** which **gathers together multiple CPUs**
- **Composed of two or more individual systems (or nodes) joined together**; **each node is typically a multicore system**
- Considered as **loosely coupled systems**
- Usually **share storage** via a **Storage Area Network (SAN)**
- **Closely linked** via a **Local Area Network** (**LAN**)



CHARACTERISTICS OF CLUSTERED SYSTEMS

- Provides a **high-availability service** which **survives failures**
  - Can be structured as follows:
    - **Asymmetric clustering:**
      - ✓has one machine in hot-standby mode while the other is running the applications
    - **Symmetric clustering**:
      - ✓has multiple nodes running applications, monitoring each other

- Some clusters are for **High-Performance Computing (HPC)**
  - **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)

- Some have **Distributed Lock Manager** (**DLM**) to **avoid conflicting operations**

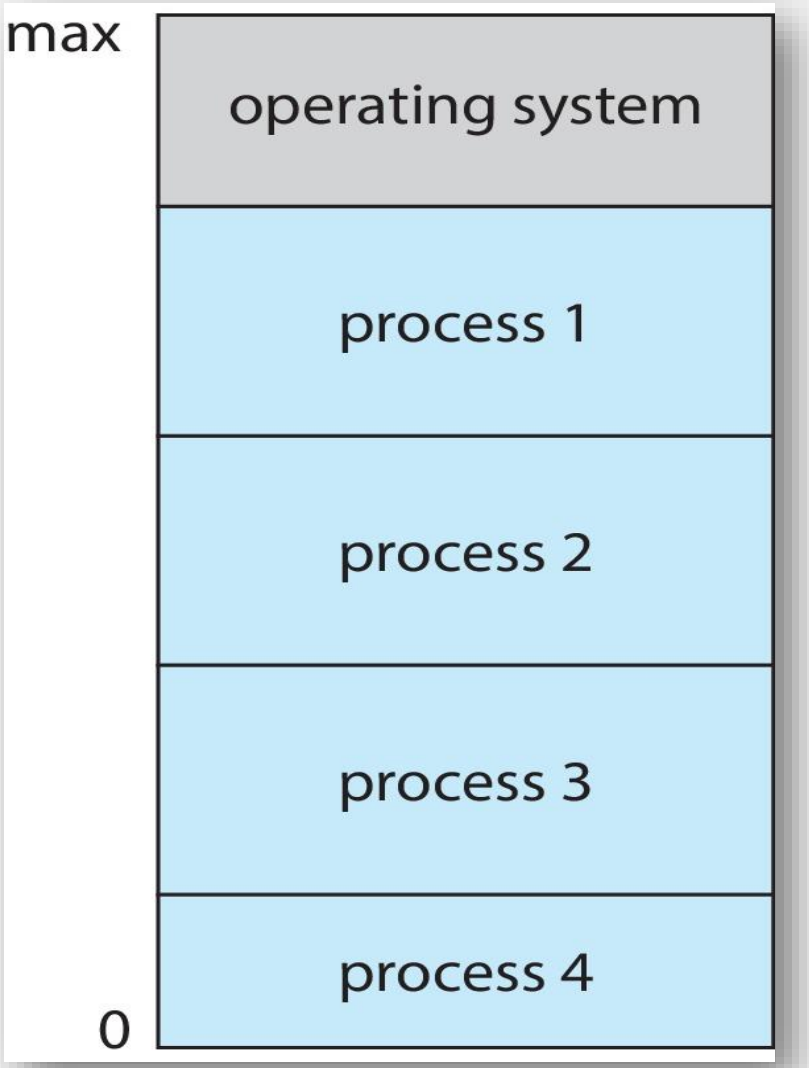
OPERATING-SYSTEM OPERATIONS

- **Bootstrap Program**
  - Needed to power-up or reboot the computer system
  - Typically stored in ROM, EPROM, or another non-volatile memory (in **firmware**)
  - Simple code to initialize the system
  - Loads operating system kernel and starts execution
- Starts **system daemons** (services provided outside of the kernel)
- Kernel is **interrupt driven** (hardware and software)
  - Hardware interrupt by one of the devices
  - Software interrupt (**exception** or **trap):**
    - Software error (e.g., division by zero)
    - Request for operating system service: **system call**
    - Other process problems include infinite loop, processes modifying each other or the operating system

MULTIPROGRAMMING

- **Multiprogramming** (**Batch system**): needed for efficiency
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming 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 it has to wait (for I/O for example), OS switches to another job

MEMORY LAYOUT FOR MULTIPROGRAMMED SYSTEM



MULTITASKING

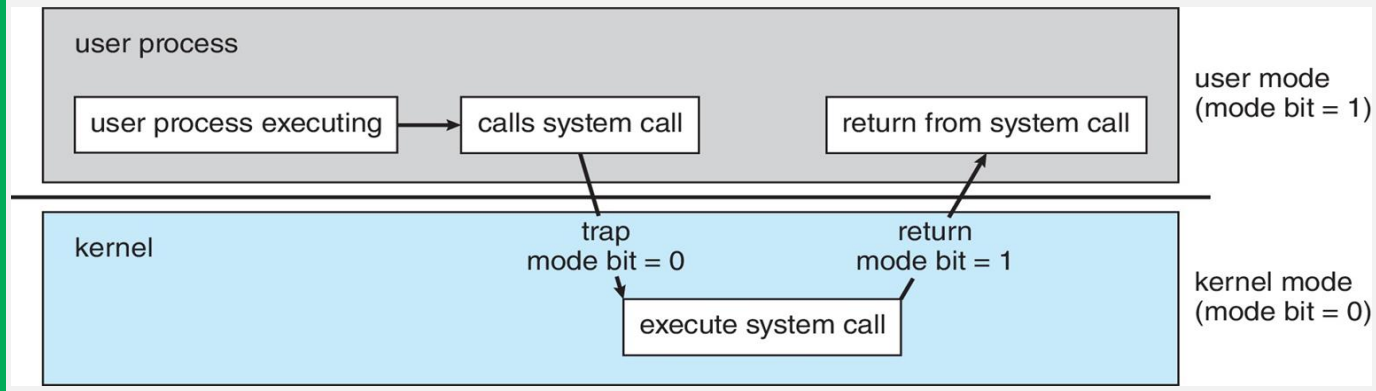
- **Multitasking (Timesharing)**:
  - Logical extension of multiprogramming
  - **CPU switches jobs so frequently** that users can interact with each job while it is running
  - **Response time** should be **< 1 second**
  - Each user has at least one program executing in memory (**process**)
  - If several jobs are 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

DUAL-MODE AND MULTIMODE OPERATION

- **Dual-Mode** operation allows OS to protect itself and other system components
  - **User mode** and **Kernel mode**
  - **Mode bit** is provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions are designated as **privileged** (only executable in kernel mode)
    - **System call changes mode to kernel, return from call resets it to user**

- Many CPUs support multimode operations
  - **Virtual Machine Manager** (**VMM**) can create and manage **VMs**

TRANSITION FROM USER TO KERNEL MODE



- **Life cycle of instruction execution in a computer system:**
  - **Initial control resides in the operating system**, where instructions are executed in **kernel mode**.
  - When **control is given to a user application**, the mode is set to **user mode**.
  - Eventually, **control is switched back to the OS** via an **interrupt, a trap, or a system call**.

- **System call:** a call (function) that is required to invoke the services provided by the operating system

- **How Does a System Call Work?**

1. Initially, a processor executes a user program in the user mode.
2. Then if the program needs the services of the operating system, the processor is interrupted by the system call.
3. A system call is always prioritized over the other executions and the system call is then executed in the kernel mode.
4. Once the system call is executed completely, the control goes back to the user mode.
5. And the execution of the program resumes back in the user mode.

TIMER

- **Timer** is used to **prevent infinite loop / process hogging resources**
  - Timer is set to interrupt the computer after a specified period
  - Keep a counter that is decremented by the physical clock
  - Operating system set the counter (privileged instruction)
  - When the counter reaches 0, an interrupt occurs
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time

- **NOTE:**
  - **Process Hogging Resources:**
    - A process which consumes a large amount of system resources compared to its importance or function.

UNIVERSITY of SAN CARLOS
SCIENTIA • VIRTUS • DEVOTIO

DCISM
DEPARTMENT OF COMPUTER, INFORMATION SCIENCES AND MATHEMATICS

1

CS 3104
OPERATING SYSTEMS

*** END ***

THANK YOU!