Introduction

Chapter 1

Scope

- **What Operating Systems Do**
- **Computer-System Organization**
- **Operating-System Structure**
- **Operating-System Operations**

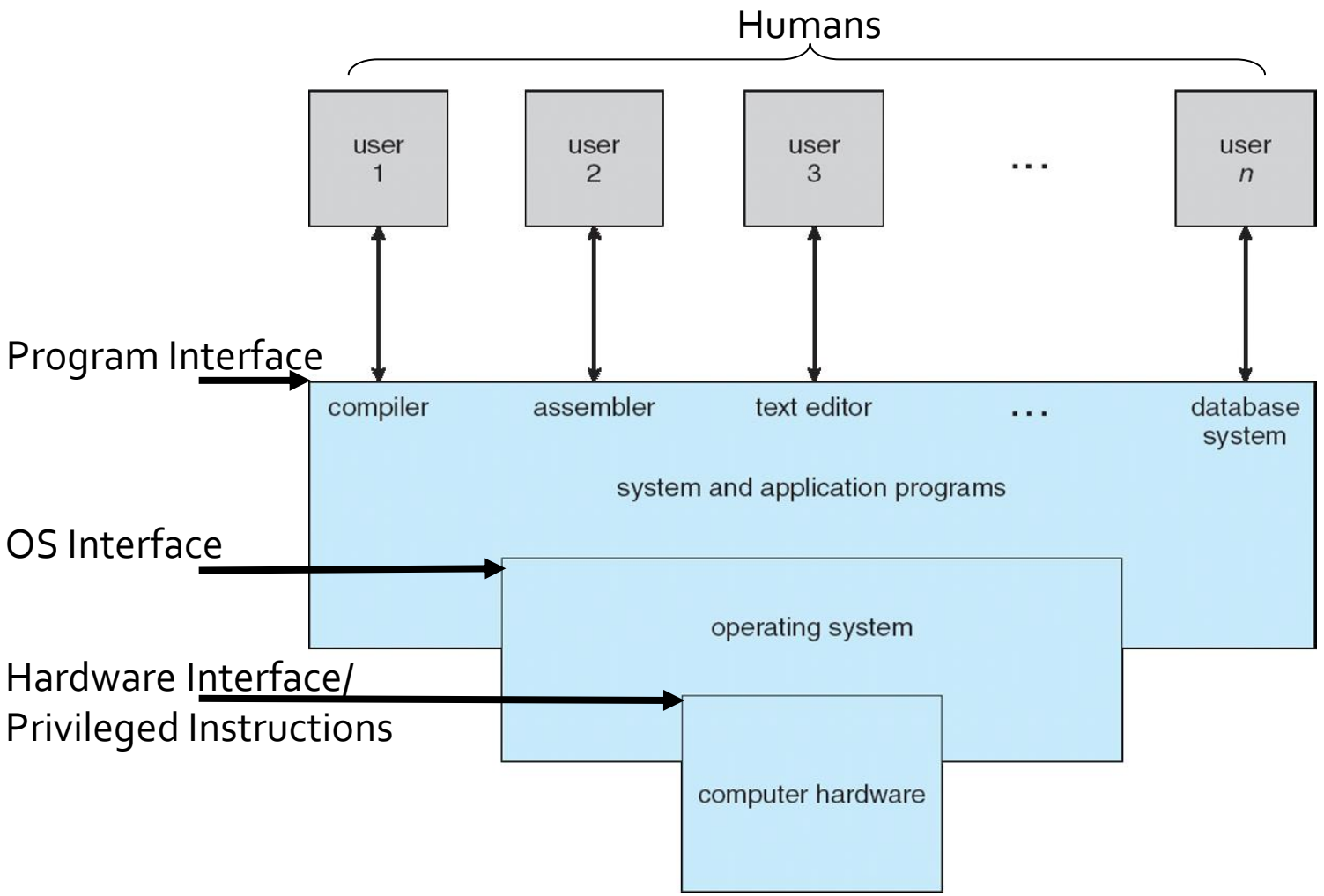
Objectives

- To provide a grand tour of the major operating systems components
- To provide coverage of basic computer system organization

Computer System Structure

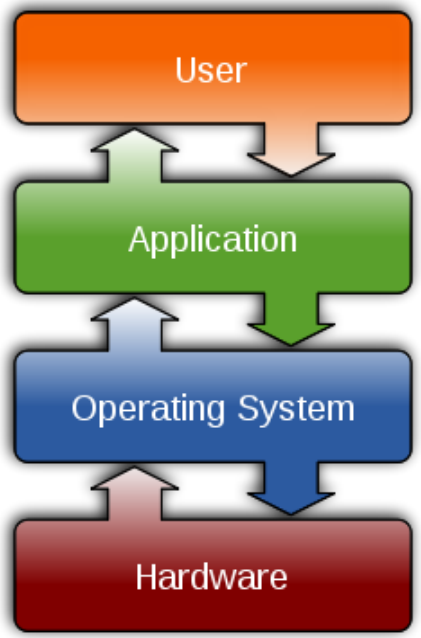
- Computer system can be divided into four components:
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - Application programs – define the ways in which the system 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

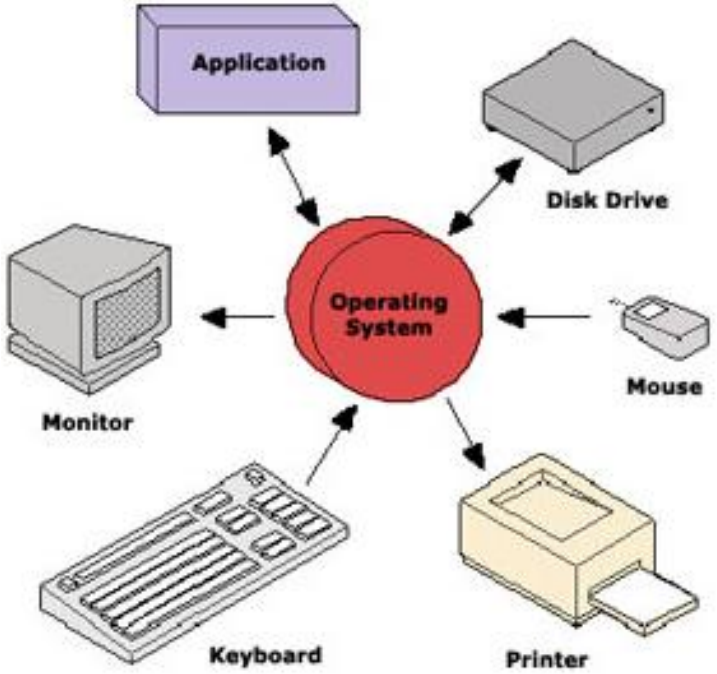
Four Components of a Computer System



What is an Operating System?

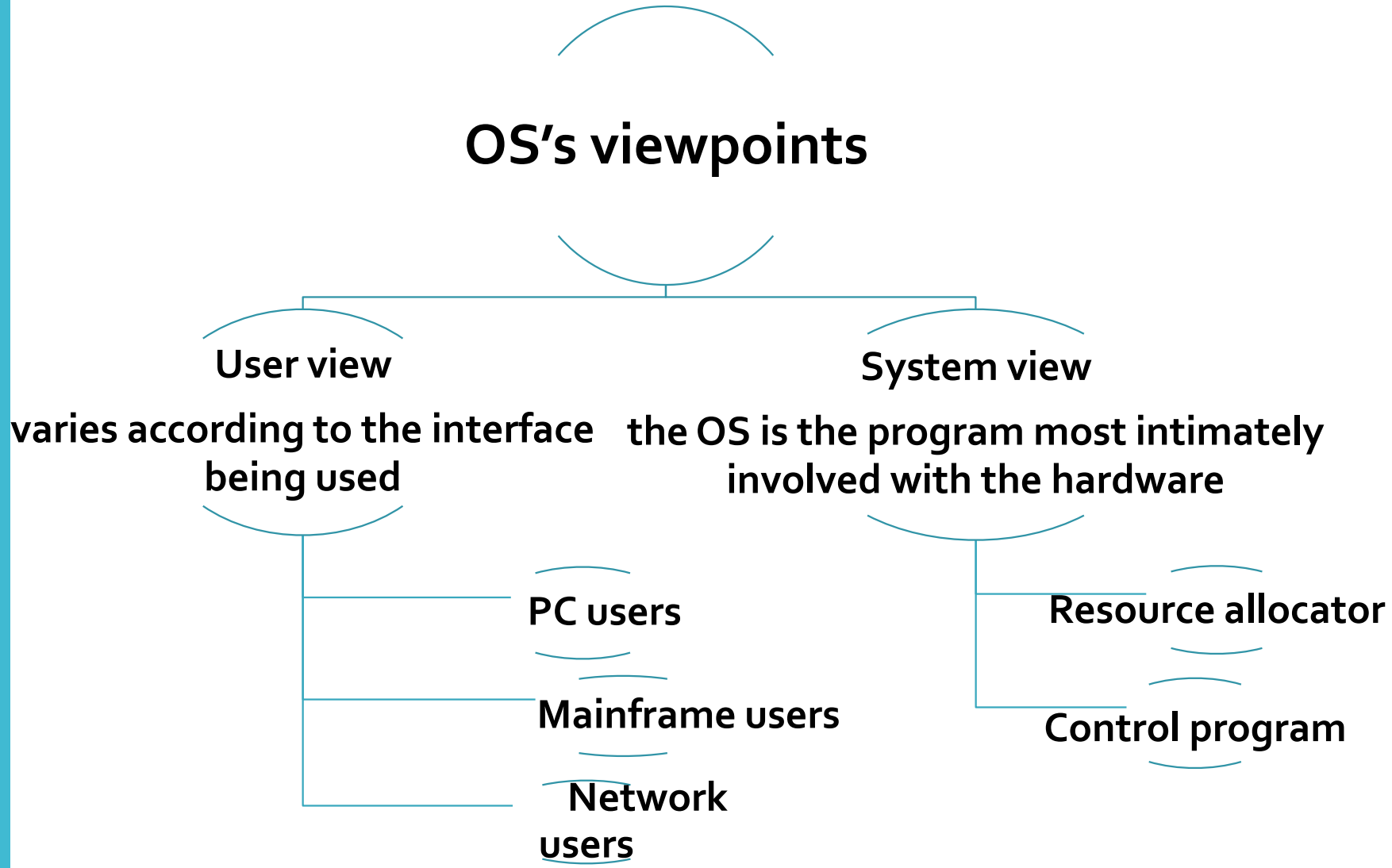
- A program that acts as an intermediary between a user of a computer and the computer hardware





- OS goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

What Operating Systems Do



User View

Users use PCs:

- Want convenience, **ease of use**
- Don't care about **resource utilization**

OS is designed mostly for with some attention paid to **performance** and none paid To **resource utilization**

User sits at a terminal connected to a mainframe or minicomputers:

- **Share resources** and may exchange information

OS is designed to maximize **resource utilization,** and keep all users happy

Users set at workstations connected to networks of other workstations and servers:

- Have dedicated **resources** at their disposal
- Share resources such as **networking** and **servers-file**,

OS is designed to compromise between **individual usability** and **resource utilization**

System View

In this context, we can view an OS as a (OS Definition):

- **Resource allocator**:
  - A computer system has many resources that may be required to solve a problem: CPU time, memory space, file-storage space, I/o devices, and so on.
  - The OS acts as the manager of these resources
  - Decides between conflicting requests for efficient and fair resource use

- **Control program**:
  - It manages the execution of user programs to prevent errors and improper use of the computer.
  - It is especially concerned with the operation and control of I/O devices.

OS Definition

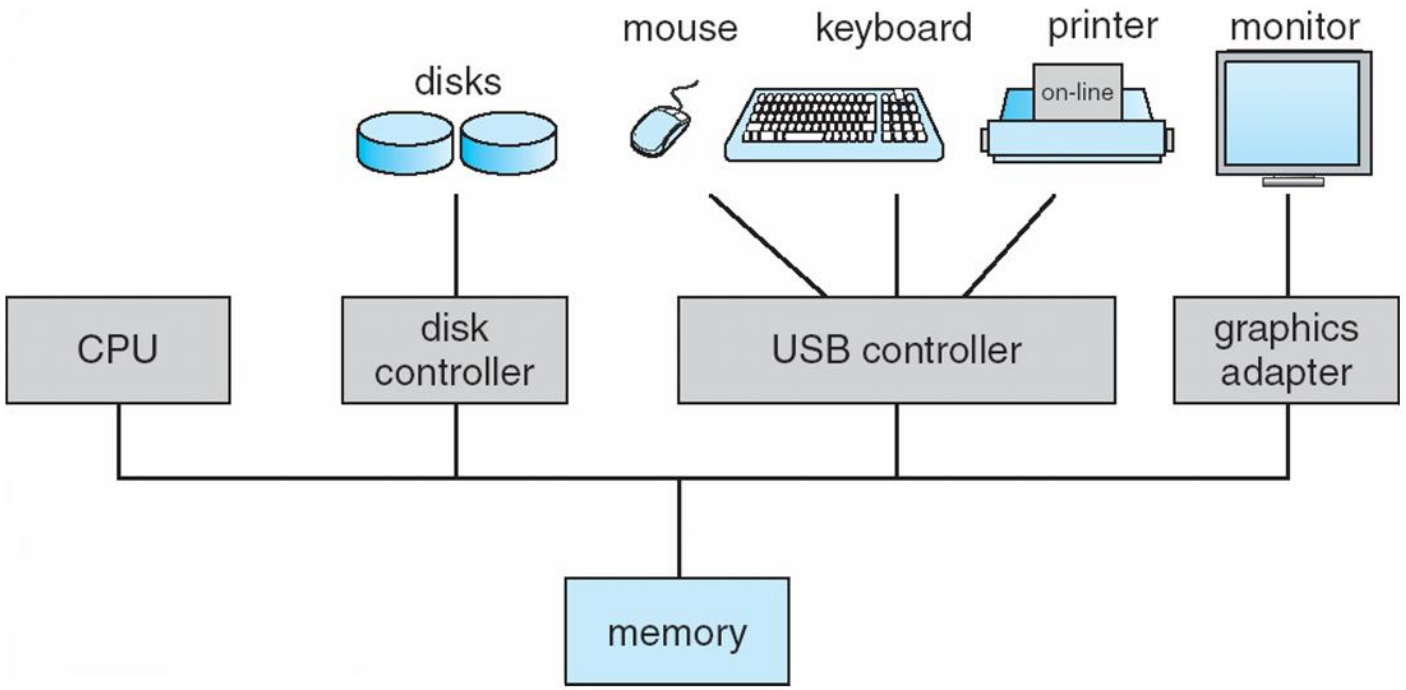
- No universally accepted definition
- "Everything a vendor ships when you order an operating system"
  - Good approximation
  - But varies wildly (vary greatly across systems)
  - Some systems take up less than 1 megabyte of space and lack even a full-screen editor, others may require gigabytes of space and are entirely based on graphical windowing systems
- "The one program running at all times on the computer", that is the kernel.
  - Everything else is either a system program (ships with the operating system) or an application program (all programs not associated with the operation of the system)

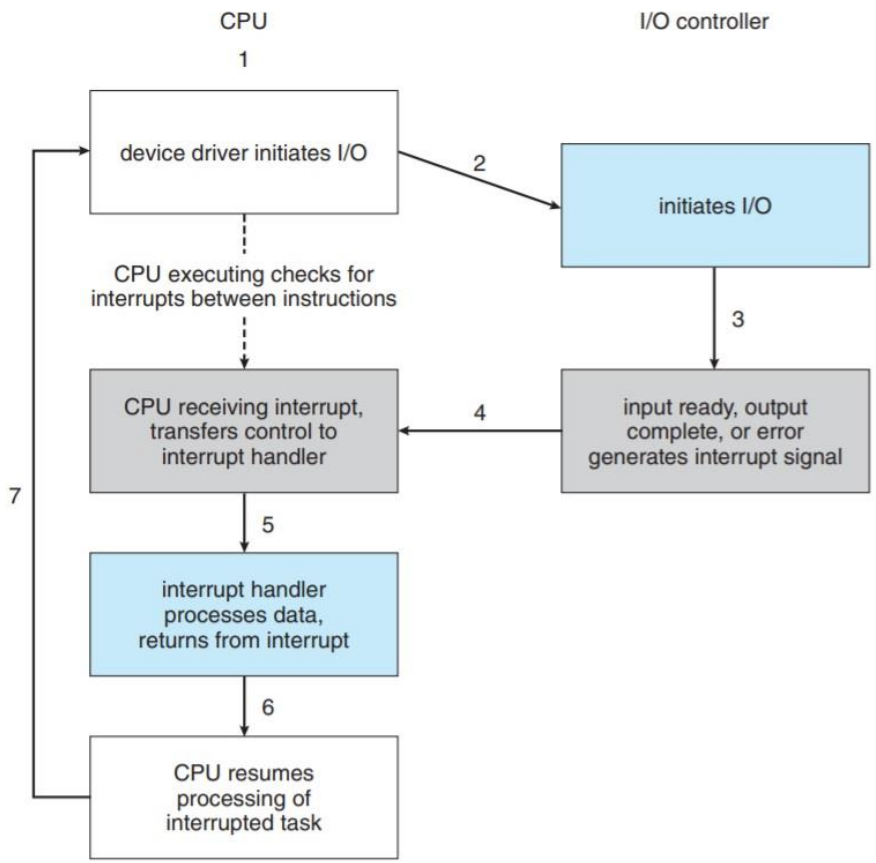
Computer Startup

- When the computer is **powered up** or **rebooted,** it needs to have an initial program (bootstrap program) to run.
- **Bootstrap program** is loaded at power-up or reboot
- Typically stored in **ROM** (read-only memory) or **EEPROM** (electrically erasable programmable read-only memory), generally known as **firmware**
  - Initializes all aspects of system (from CPU registers to device controllers to memory contents)
  - Loads OS kernel and starts execution
- OS then starts executing the first process, and waits for some event to occur
- Event is usually signaled by an **interrupt** from either the hardware or the software
  - Hardware may trigger an interrupt at any time by sending a **signal** to the CPU, usually by way of the system bus.
  - Software may trigger an interrupt by executing a special operation called a **system call** (also, called **monitor call**)

Computer System Organization

- Computer-system operation
  - A modern general-purpose computer system consists of:
    - One or more CPUs, and a number of device controllers connect through common bus providing access to shared memory
    - Each device controller is in charge of a particular device type





Computer-System Operation

- I/O devices and the CPU can execute concurrently
- Each device controller is in-charge of a particular device type
- Each device controller has a local buffer
- 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

Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines

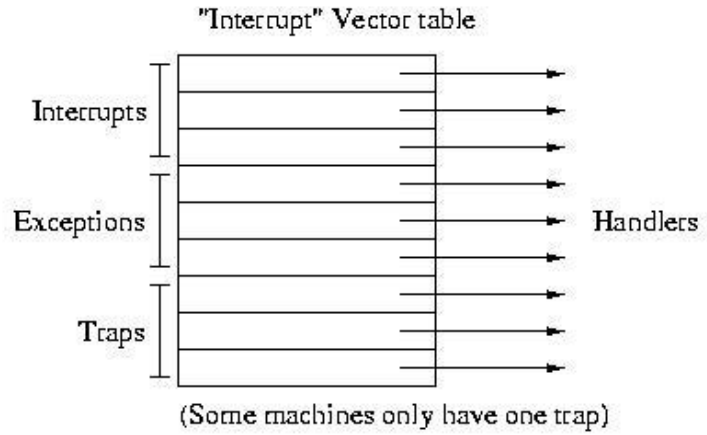


Table 11-1: Interrupt Vector Table for the 8051

Interrupt	ROM Location (Hex)	Pin
Reset	0000	9
External hardware interrupt 0 (INT0)	0003	P3.2 (12)
Timer 0 interrupt (TF0)	000B	
External hardware interrupt 1 (INT1)	0013	P3.3 (13)
Timer 1 interrupt (TF1)	001B	
Serial COM interrupt (RI or TI)	0023	

https://slideplayer.com/slide/6969422/24/images/17/Table+11-1%3A+Interrupt+Vector+Table+for+the+8051.jpg

Common Functions of Interrupts- cont.

- Interrupt architecture must **save the address of the interrupting instruction**
- Incoming interrupts are ***disabled*** while another interrupt is being processed to prevent a *lost interrupt*
- A ***trap*** is a software-generated interrupt caused either by an error or a user request
- An OS is **interrupt driven**
- **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 OS service
    - Other process problems include infinite loop, processes modifying each other or the operating system

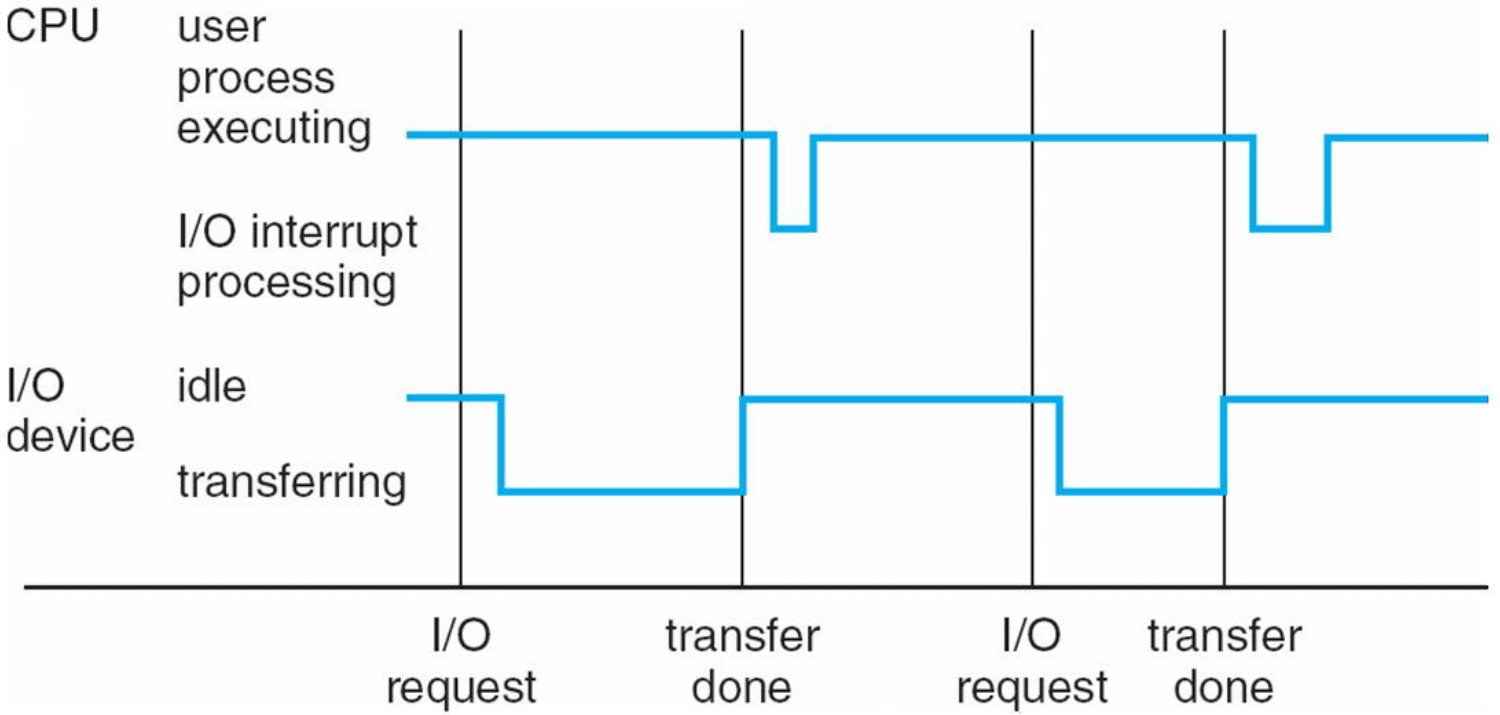
Interrupt Handling

- The **OS preserves the state of the CPU** by storing registers and the program counter
- Determines which type of interrupt has occurred:
  - **Polling interrupt (Who interrupted me?)**
    - A specific type of I/O interrupt that notifies the part of the computer containing the I/O interface that a device is ready to be read or otherwise handled but does not indicate which device.
    - The interrupt controller must poll (send a signal out to) each device to determine which one made the request.
  - **Vectored interrupt system**
    - An interrupt signal that includes the identity of the device sending the interrupt signal.
- Separate segments of code determine what action should be taken for each type of interrupt

Interrupt Timeline

1- I/O device informs CPU that it has finished its operation by causing an interrupt

2- When the CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location.

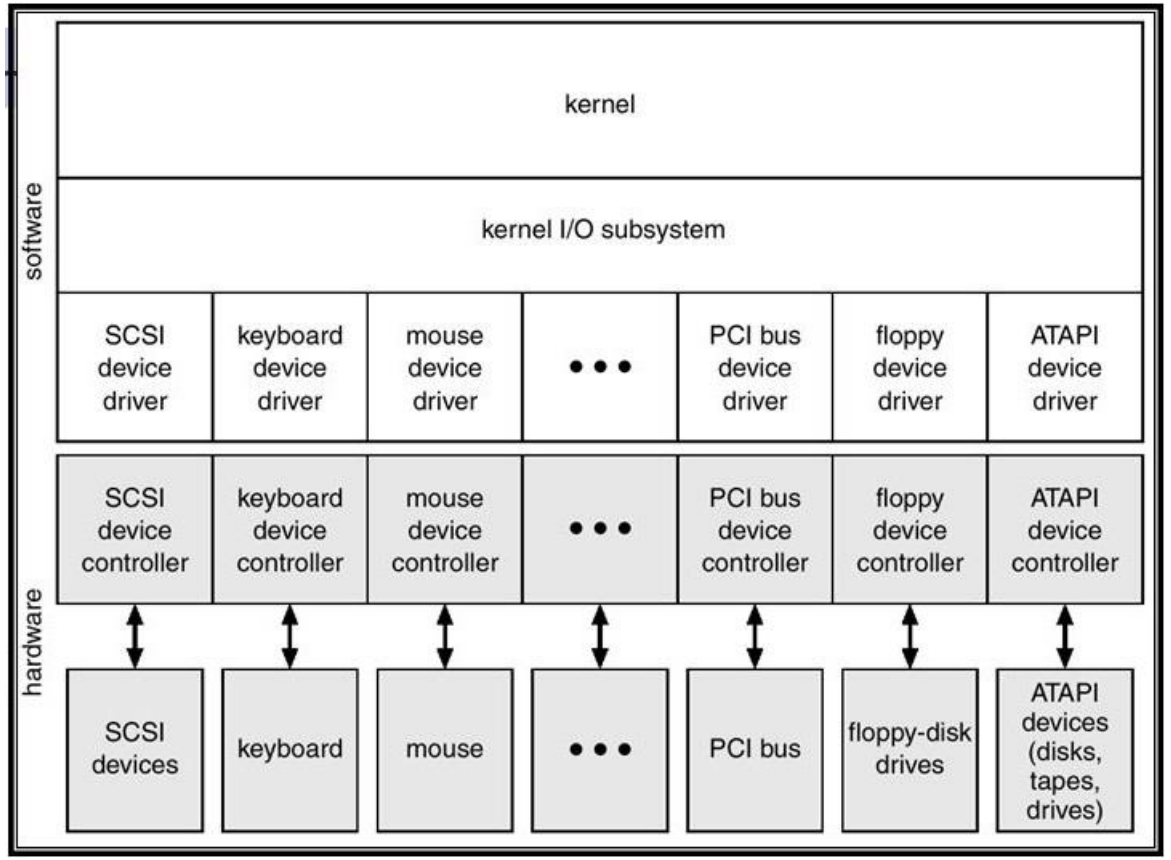


3- The fixed location usually contains the starting address where the service routine for the interrupt is located.

4- The interrupt service routine executes; on completion, the CPU resumes the interrupted computation

I/O Structure

- A large portion of OS code is dedicated to managing I/O



I/O Structure-cont.

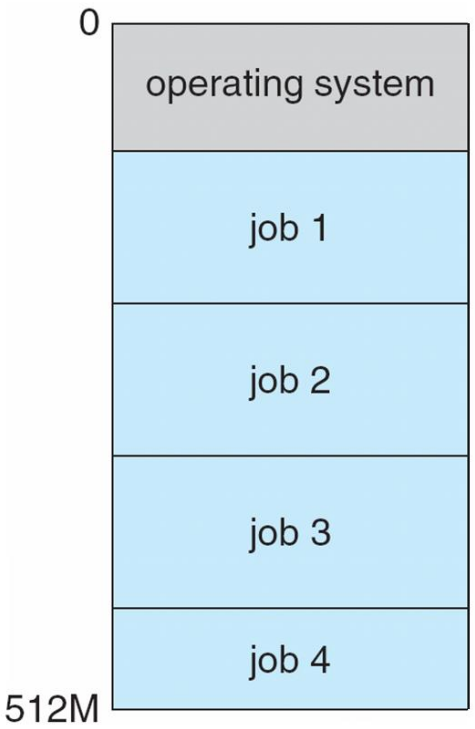
- **After I/O starts, control returns to user program in two ways**:
  - **Synchronous-** Only upon **I/O completion**
    - `wait()` instruction idles the CPU until the next interrupt
    - Wait loop (contention for memory access)
    - At most one I/O request is outstanding at a time, no simultaneous I/O processing
  - **Asynchronous- Without** waiting for **I/O completion**
    - **System call** – request to the OS to allow user to wait for I/O completion
    - **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

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

OS Structure

- **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 the OS has to wait (for I/O for example), it switches to another job
  - Multiprogramming ***increases CPU utilization,*** but it ***doesn't provide for user interaction*** with the computer system.



Memory Layout for Multiprogrammed System

OS Structure-cont.

- **Multitasking(Timesharing)** is logical extension in which CPU

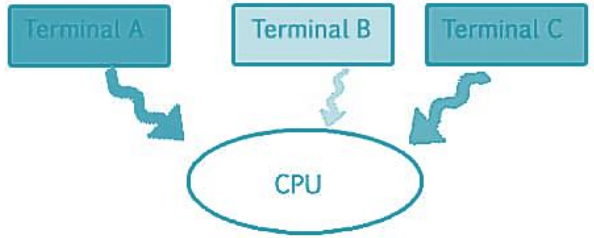
switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing

  - **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

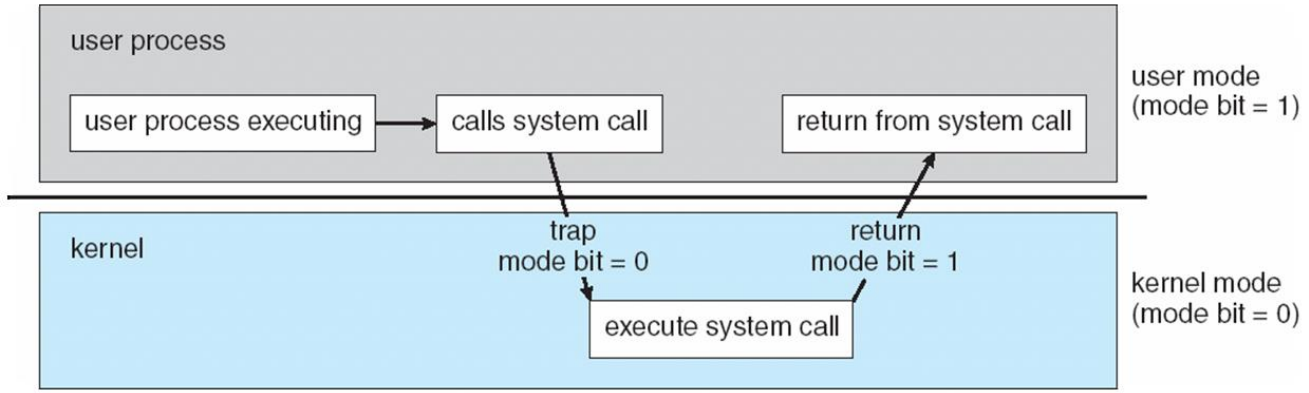
  - **Virtual memory** allows execution of

processes not completely in memory



OS Operations

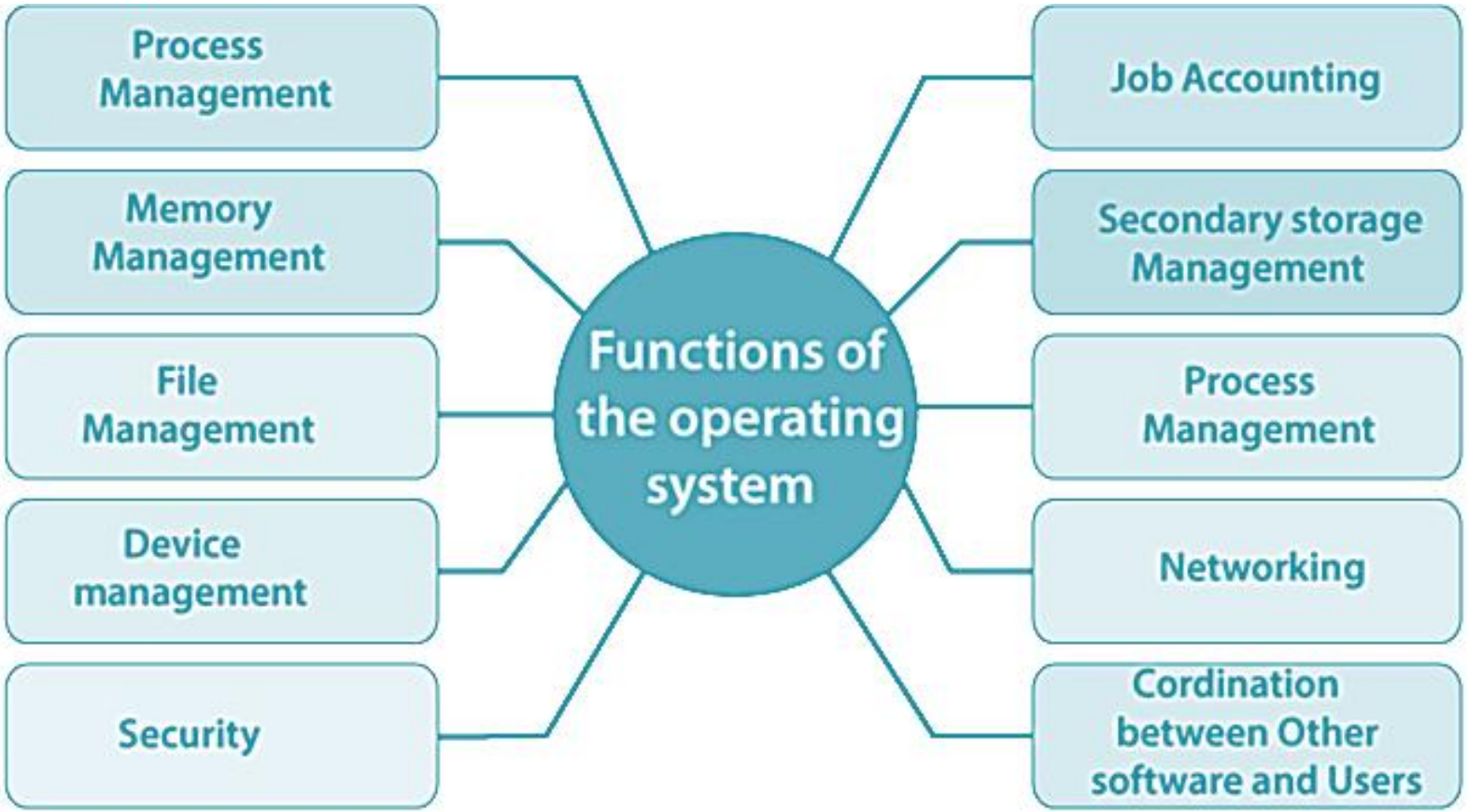
- To ensure proper execution of OS operations, we **must distinguish between the execution of OS code and user-defined code**
- OS has two separate modes:
  - **User mode** and **kernel mode**
    - **Mode bit** provided by hardware
      - Provides ability to distinguish when system is running user code or kernel code
      - Some instructions designated as **privileged,** only executable in kernel mode
      - System call changes mode to kernel, return from call resets it to user
- This **dual-mode of operation** allows OS to protect itself and other system components



Timer

- We must ensure that the OS maintains control over the CPU.
- **Timer** is used to prevent infinite loop / process hogging resources
  - Set interrupt after specific period
  - OS decrements counter
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time

OS Functions



End of Chapter 1

- Class Activity
  - What are the three main purposes of an operating system?
  - We have stressed the need for an OS to make efficient use of the computing hardware. When is it appropriate for the OS to forsake this principle and to "waste" resources? Why is such a system not really wasteful?
  - How does the distinction between kernel mode and user mode function as a rudimentary form of protection (security) system?