Operating Systems: Introduction

Neerja Mhaskar

Department of Computing and Software, McMaster University, Canada

Acknowledgements: Material based on the textbook Operating Systems Concepts (Chapter 1)

Course Introduction

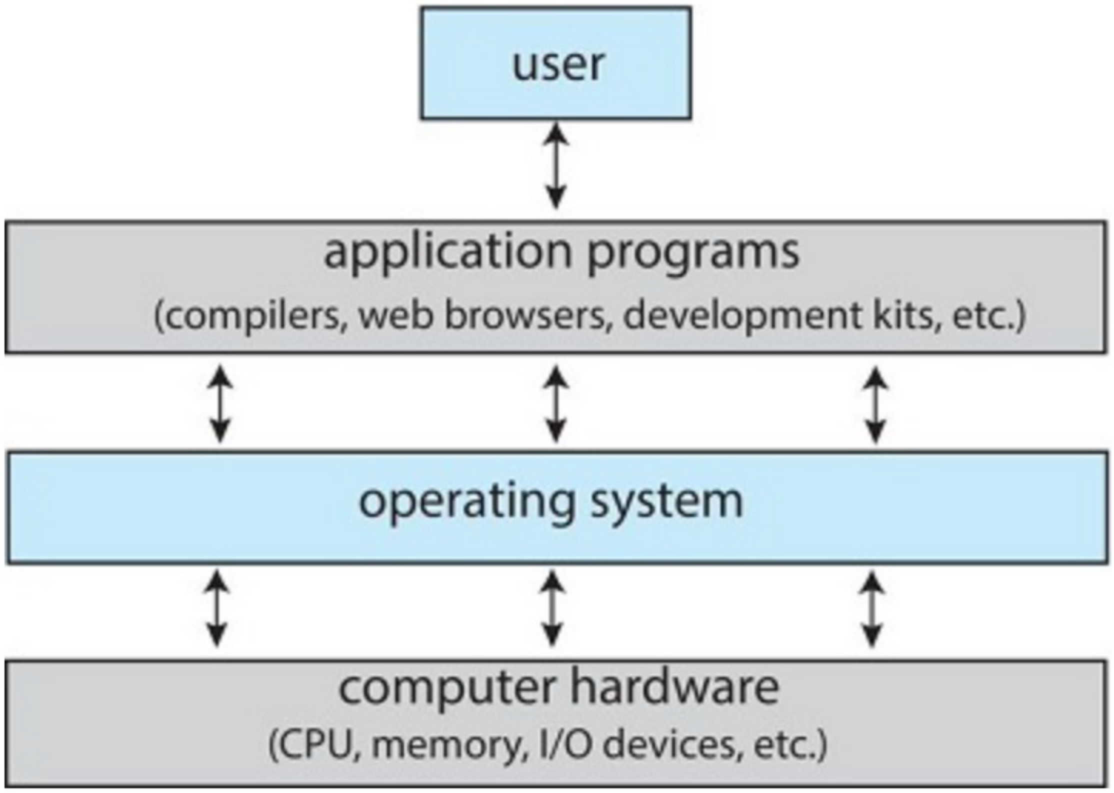
- Take home Lab
  - Installing Virtual Machine
  - Dev Containers
- Groups
  - You will not be able to view or submit assignments if you don't enroll in a group!

What does an Operating System do?

- An Operating system is a
  - Resource allocator: Manages a computer's resources
  - Control program: Controls execution of programs.
  - Acts as an intermediary between a user of a computer and the computer hardware.
- Operating system goals are
  - *Efficient use*
  - *User convenience*
  - *Non-interference*

Computer System Structure

- Computer system can be divided into four component

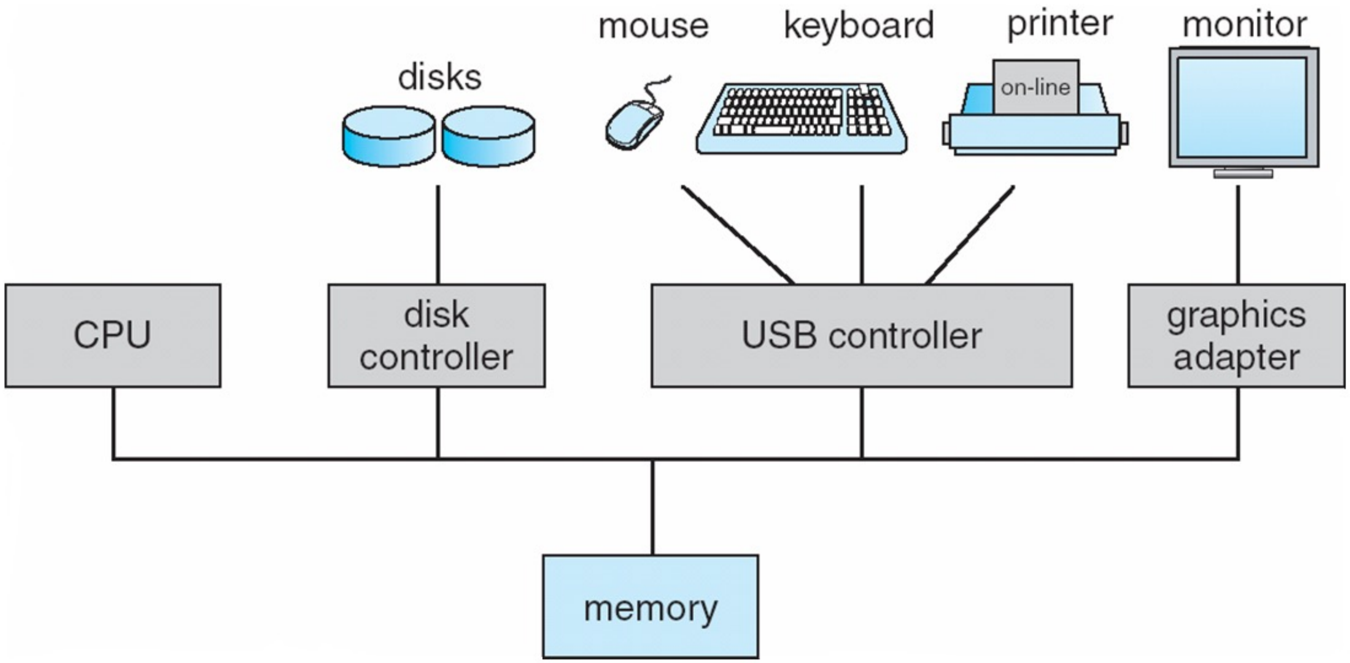

user
application programs
(compilers, web browsers, development kits, etc.)
operating system
computer hardware
(CPU, memory, I/O devices, etc.)

What is an OS?

- Primarily an OS consists of the **Kernel**
  - Kernel stays in main memory (and is the one program that is always running on the computer)
    - Supports process and memory management, some also support file management.
  - Kernel controls the execution of all other programs
  - Other programs (system or user) interact with the kernel through *system calls* (are routines mostly written in a high-level language (C or C++). However, lower-level task written in assembly.)

Computer System Organization

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



Computer Startup

- When a computer system is switched on or rebooted,
  - it automatically loads a program (**bootstrap program**) stored on a reserved part of an I/O device typically a disk (ROM or EEPROM, generally known as **firmware**) and starts executing the program.
- The bootstrap program then
  - Loads the kernel
  - and system programs (also known as system daemons)
- The kernel and system programs run all the time on the computer to provide services.
- After the system is fully booted it waits for an **event** to occur.

Computer System Operation - Interrupts

- An operating system is **event driven** and events occur by interrupts. Therefore, OS is **interrupt driven**.

- **Interrupt:** is a mechanism that enables a device/software to notify the CPU that it needs attention.

- An interrupt is caused by a
  - signal to CPU from a device attached to a computer via system bus (hardware), or
  - from an executing program within the computer through system calls.

- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request.

Computer System Operation: Interrupt Handling

- Interrupt requires the operating system to stop and figure out what to do next.

- When an interrupt occurs:
  - CPU stops executing current task
  - The operating system preserves the state of the CPU by storing registers and the program counter
  - Transfers execution to a fixed memory location, which has the starting address of **Interrupt Service Routine (ISR)**
  - ISR handles the interrupt, after which the interrupted process resumes its execution.

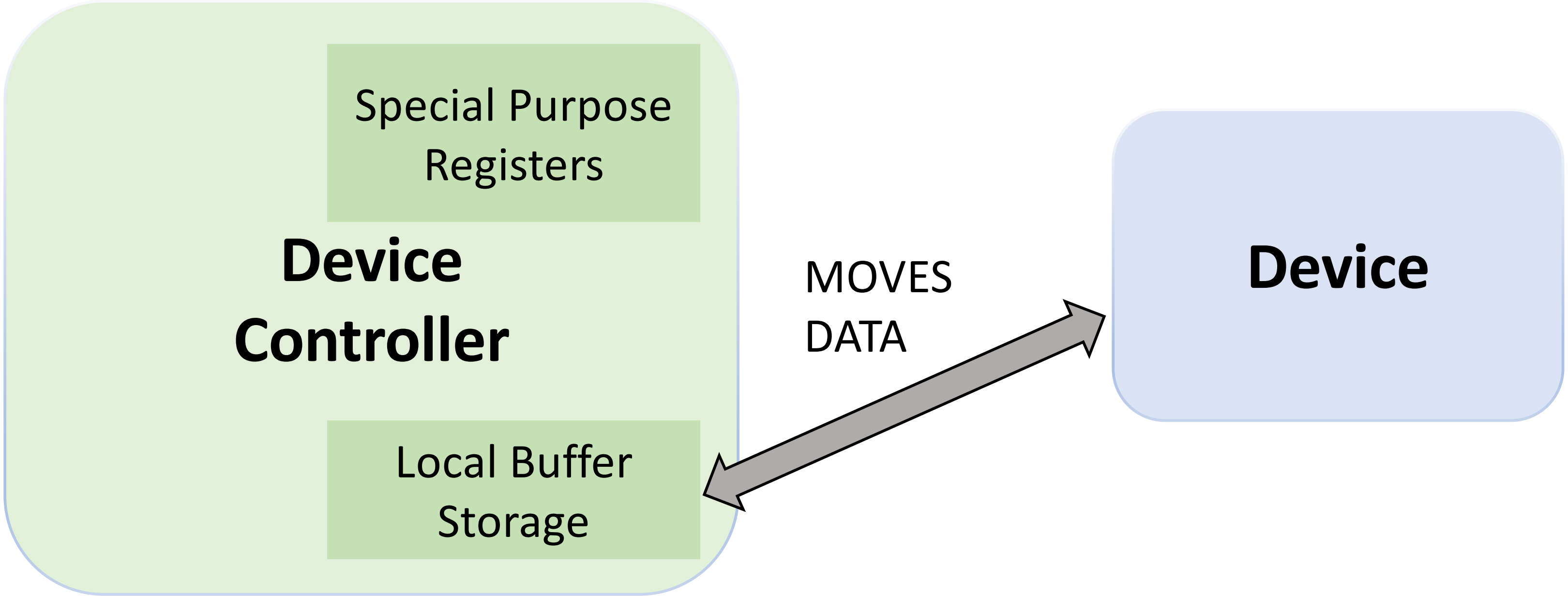
- Note: Separate segments of code determine what action should be taken for each type of interrupt

Computer System Operation: Interrupt Handling Implementation

- Implementing ISR as a routine is slow and therefore inefficient.
- Only predefined interrupts exist
  - Use a table of pointers to the various interrupt routines instead!
  - This table/array of addresses is called **Interrupt vector**
- Interrupt vector is usually stored in low memory.
- Windows and Linux dispatch interrupts in this manner.

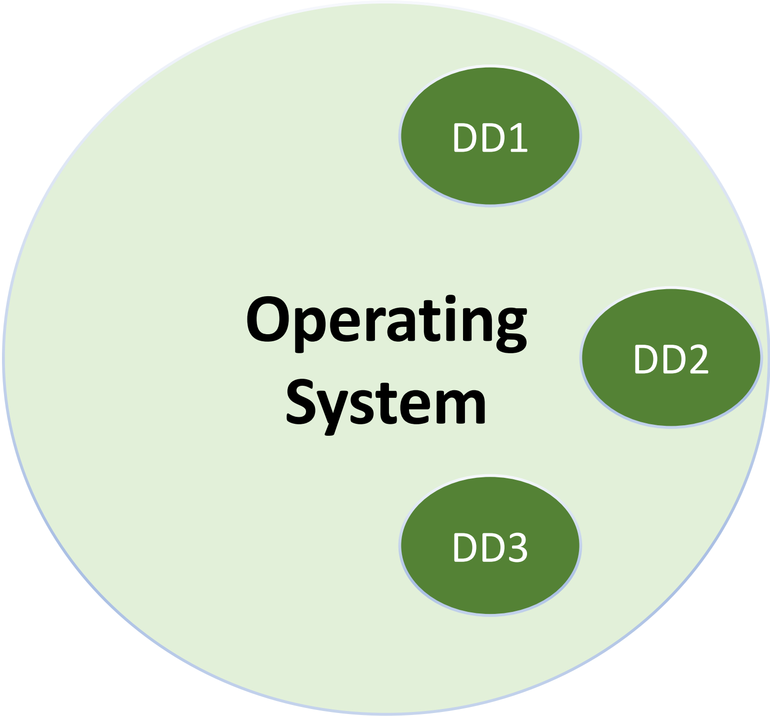
Computer System Operation: I/O Structure

- A general-purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus.
- **Device Controller:** The I/O managing processor within a device.



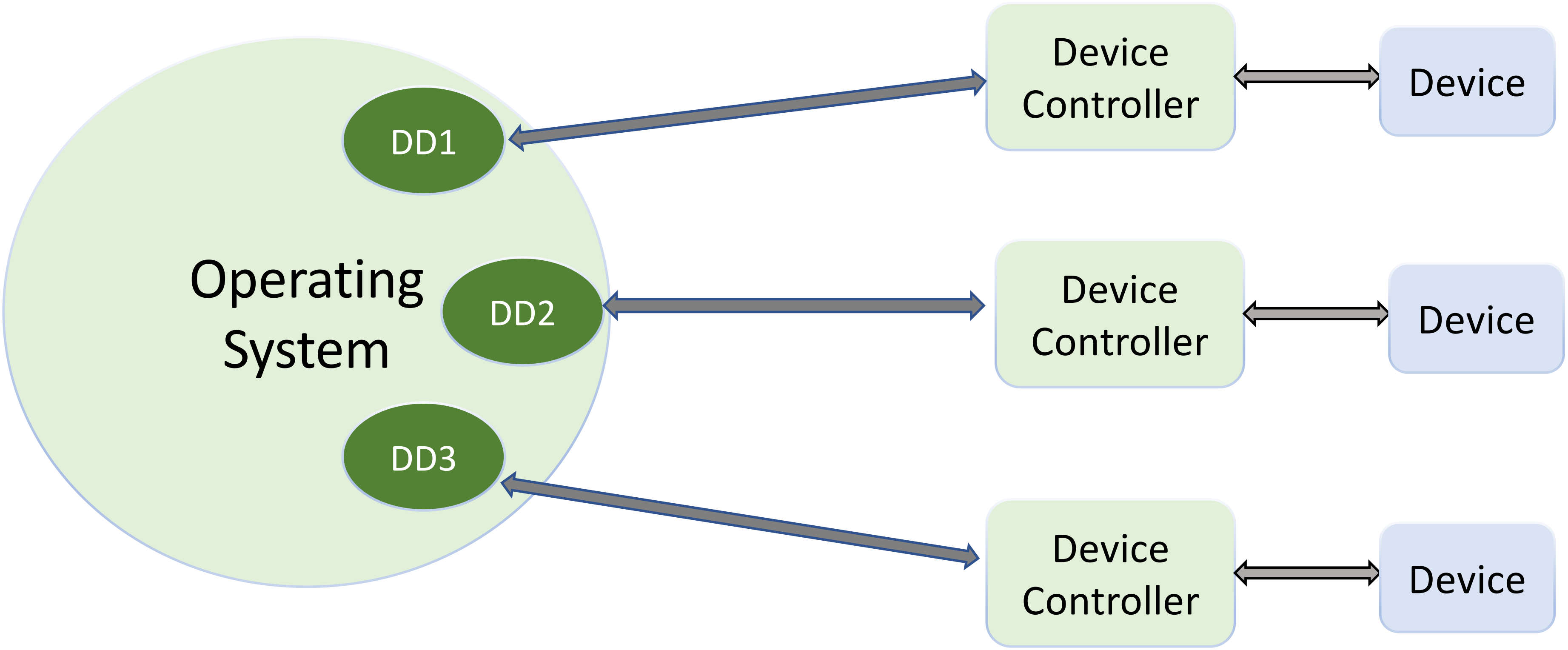
Computer System Operation: I/O Structure

- **Device driver:** An *operating system component* that provides uniform access to various devices and manages I/O to those devices.
- In an operating System there is a *device driver (DD)* for each device controller.



Computer System Operation: I/O Structure

- Data movement between devices and the computer happens through the interaction between the device drivers and device controllers.



Computer System Operation

- CPU can access only data and instructions only from **main memory (RAM)**
- Programs must be brought into main memory for execution
  - Main memory is volatile
  - Secondary storage needed for permanent storage
- All forms of main memory provide an array of bytes, and each byte has its own address.

Computer System Operation: Storage-Device Hierarchy

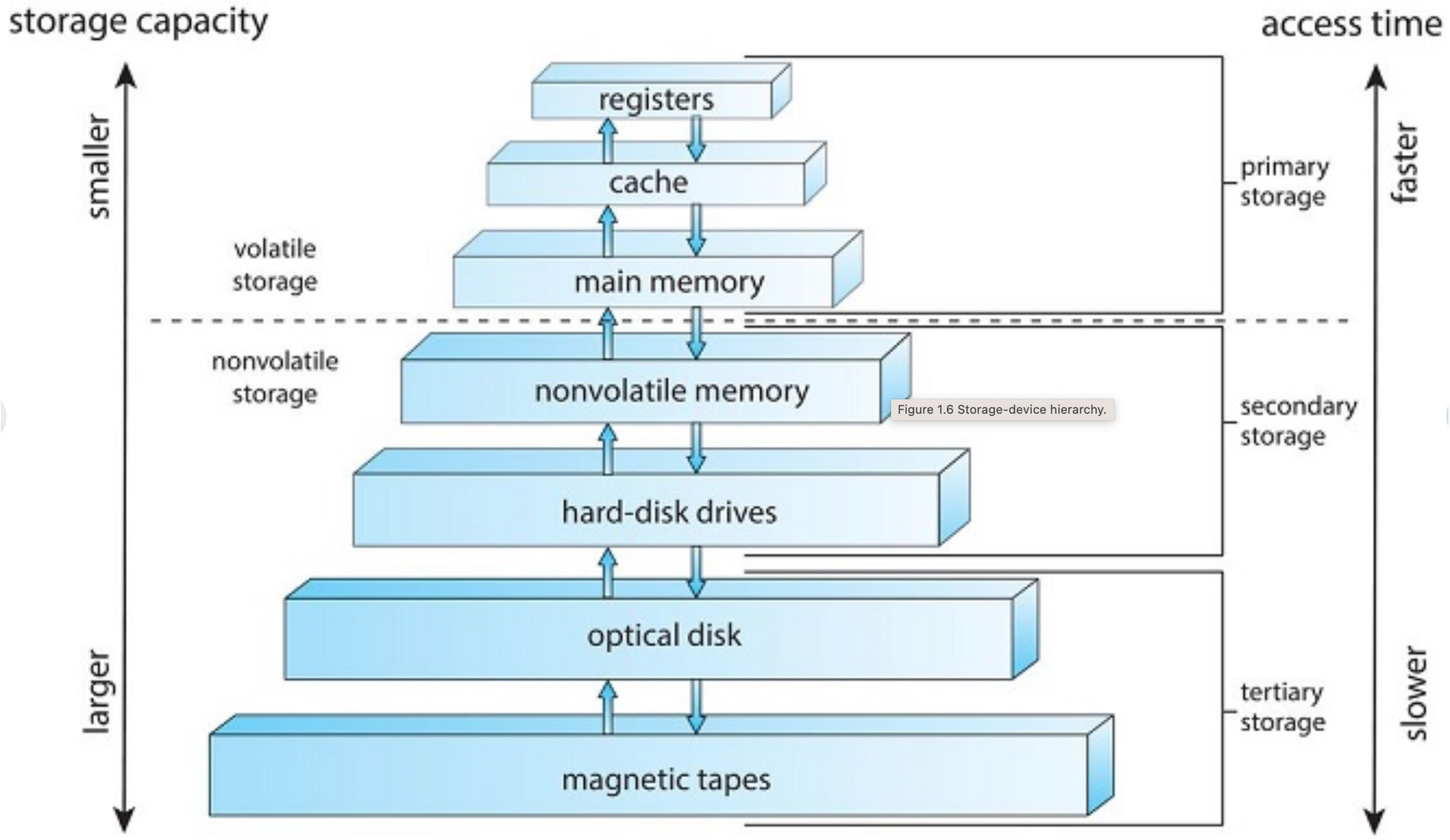


Figure 1.6 Storage-device hierarchy.

Operating System Operation: Multiprogramming

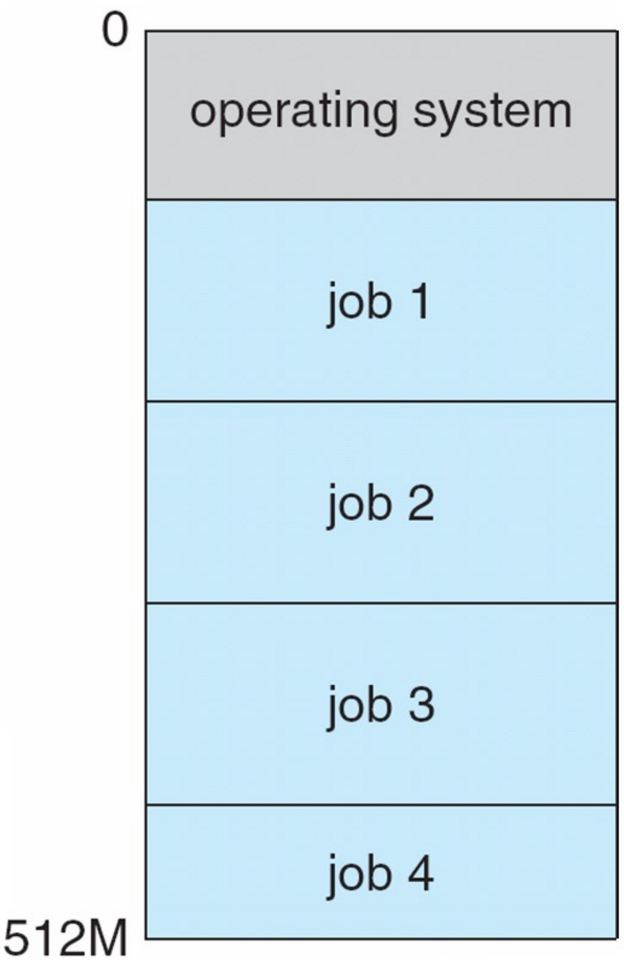
- An operating system provides the environment within which multiple programs are executed

How does multiprogramming work?

- OS organizes jobs so CPU always has one to execute
  - A subset of total jobs in system is kept in main memory
  - One job is 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

Low Memory



High Memory

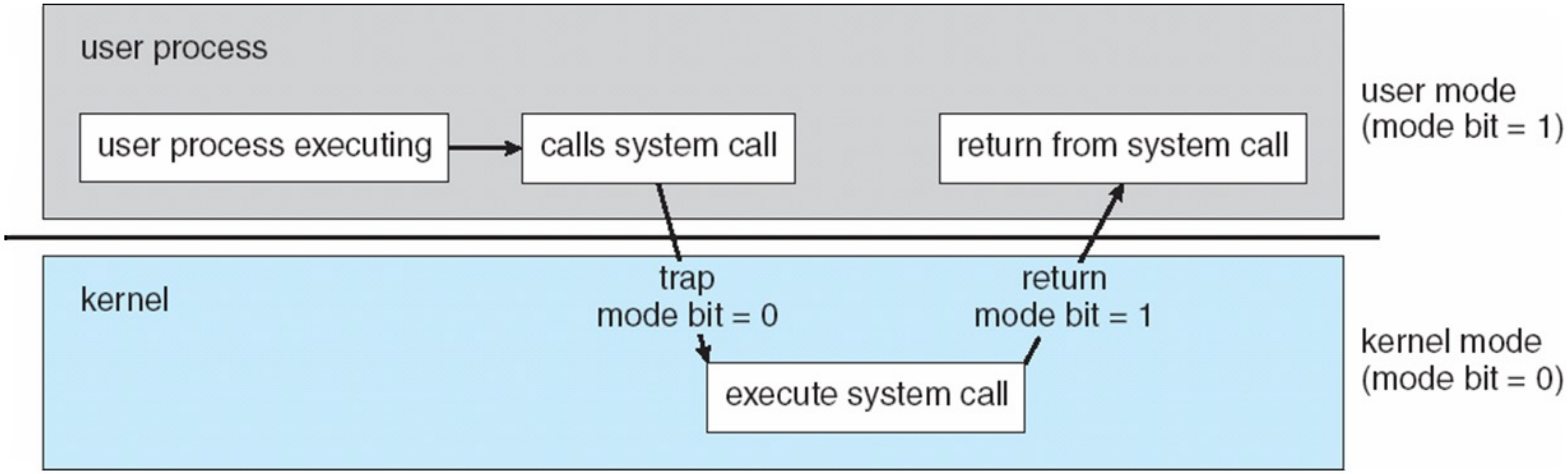
Operating System Operation

- **Timesharing** (**multitasking**) is logical extension of multiprogramming
  - CPU switches jobs so frequently that users can interact with each job while it is running, creating ***interactive*** computing
  - **Response time** is **< 1 second**

Operating-System Operations

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** is CPU Status bit used to indicate the current mode
    - Provides ability to distinguish when CPU is running in user code or kernel code
    - Kernel mode bit = 0 and User mode bit = 1
    - **Privileged instructions** only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user (see next slide)

Transition from User to Kernel Mode - Illustration



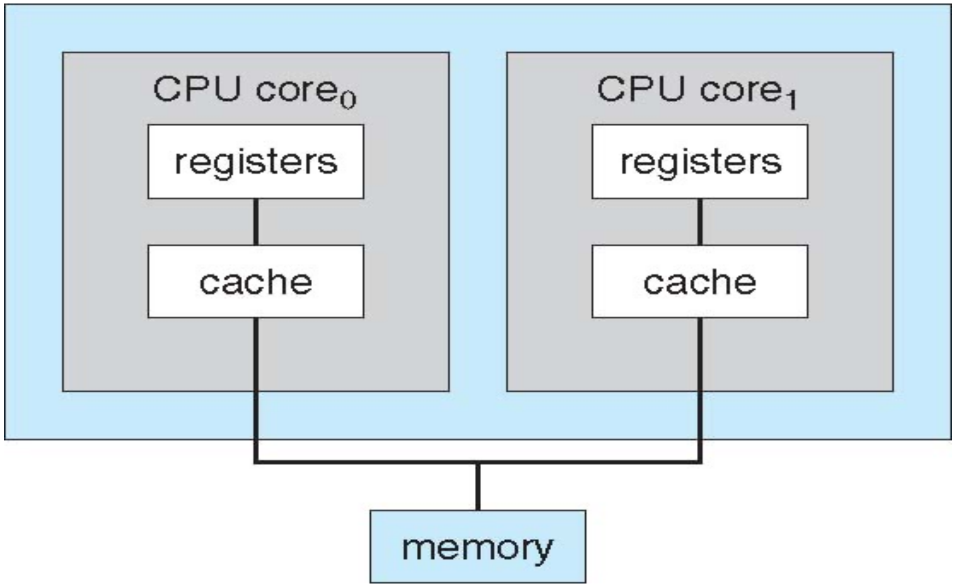
Computer-System Architecture

- A **single processor** system has
  - One general purpose CPU
  - Possibly more than one special purpose CPUs (cannot process user request)
- **Multiprocessors** (parallel systems, multi-core systems)
  - More than one general purpose CPU’s (sometimes share memory, bus and peripherals)
  - Advantages include: Increased throughput, Economy of scale, and Increased reliability

Multi-core Design

- Multiprocessor systems can be Multi-chip and/or **multicore** (More than one core on a single chip)
- Multicore systems are efficient as on chip communication is faster and consumes less power

Example of a dual core design:



Key functionality of an OS as a resource Manager

- Process Management
- Memory Management
- Storage and File management
- Protection and Security

Process

- Process is a program in execution

- Program is ***passive*** entity stored on disk (executable file), process is ***active***

  - Program becomes process when executable file loaded into memory

Process representation in Linux

- Processes in Linux are referred to as **tasks**.
- Represented by the C structure `task_struct`

```
pid_t pid; /* process identifier */

long state; /* state of the process */

unsigned int time_slice /* scheduling information */

struct task_struct *parent; /* this process's parent */

struct list_head children; /* this process's children */

struct files_struct *files; /* list of open files */

struct mm_struct *mm; /* address space of this process */
```

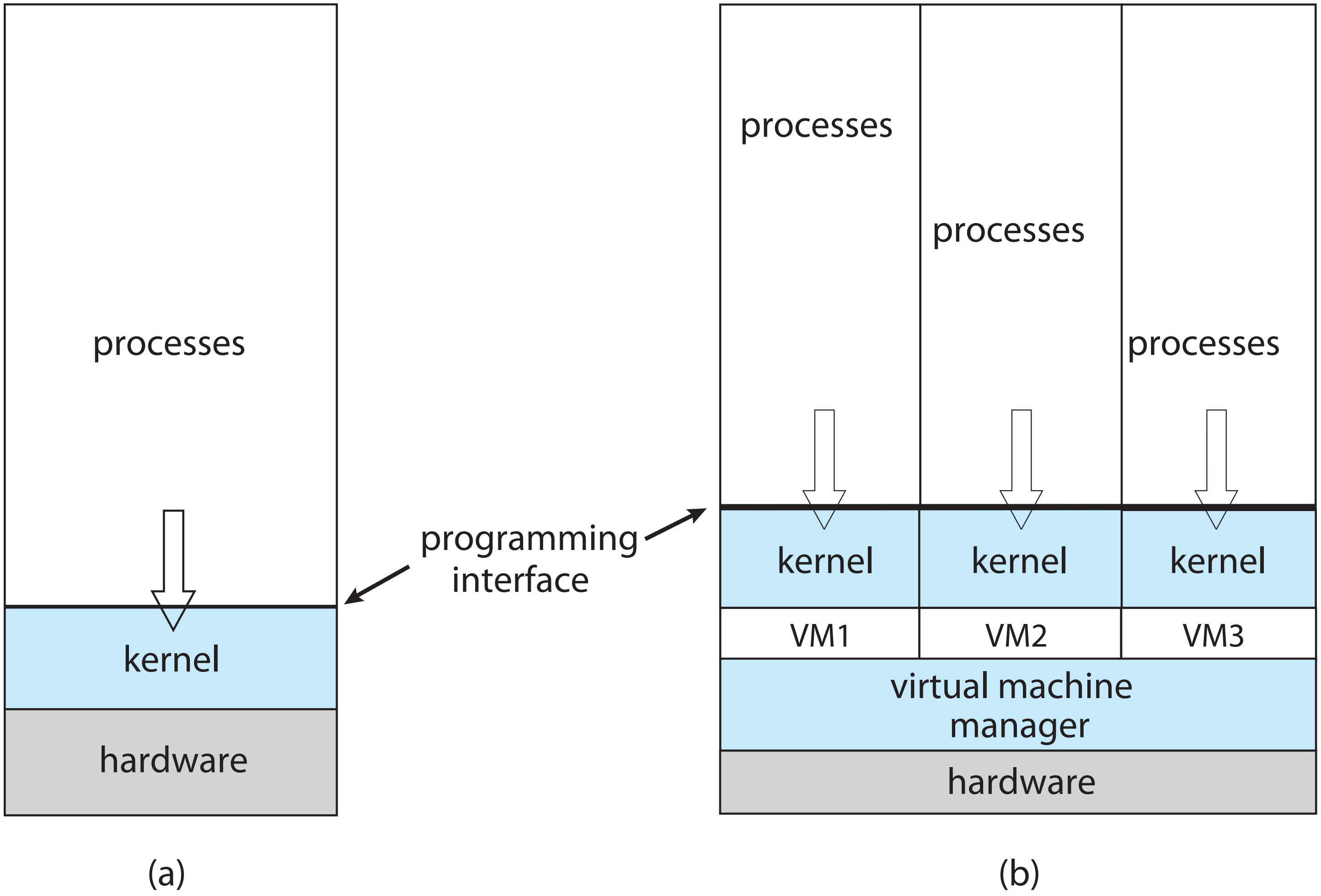
Kernel Data Structures

- Many similar to standard programming data structures
  - Linked lists (single, double and circular)
  - Binary search tree
  - Hash Tables
- Bitmap - string of *n* binary digits representing the status of *n* items
- Linux data structures defined in ***include*** files `<linux/list.h>`, `<linux/kfifo.h>`, `<linux/rbtree.h>`

Computing Environments - Virtualization

- **Virtualization** is a *technology* that creates abstraction of the computer hardware.
- Thereby creating *an illusion* that all its operating systems are running on it own private computer.
- **VMM** (virtual machine Manager) provides virtualization services

Computing Environments - Virtualization


processes
kernel
hardware
(a)
programming
interface
processes
processes
processes
kernel
kernel
kernel
VM1
VM2
VM3
virtual machine
manager
hardware
(b)