Operating Systems Concepts (CS-351)

Course Syllabus

- You are required to read the syllabus!
- A copy of the syllabus is available online.
- If something is not clear, ask the instructor.

Introduction to Operating Systems

Silberschatz Chapter 1

Agenda

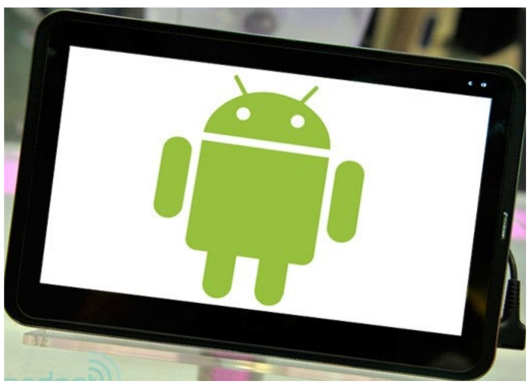
- Operating Systems, what are they? What do they do?
- Computer organization fundamentals
- Operating System Functions
- Process Management
- Memory Management
- Storage Management
- I/O Management
- Protection and Security
- Distributed Systems
- Special Purpose Operating Systems
- Computing Models

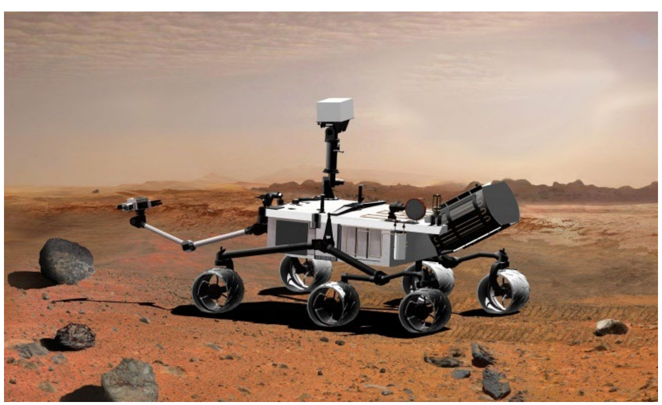
What is an Operating System?

- A program that:
  - Runs at all times (a.k.a. a resident monitor, a.k.a. kernel)
  - Manages computer's hardware resources.
  - Provides basis for application programs.
  - Acts as an intermediary between user and hardware.
  - No completely adequate definition.

- Why do we need operating systems?
  - Increase the usability of computers.
  - Simplify problem solving.

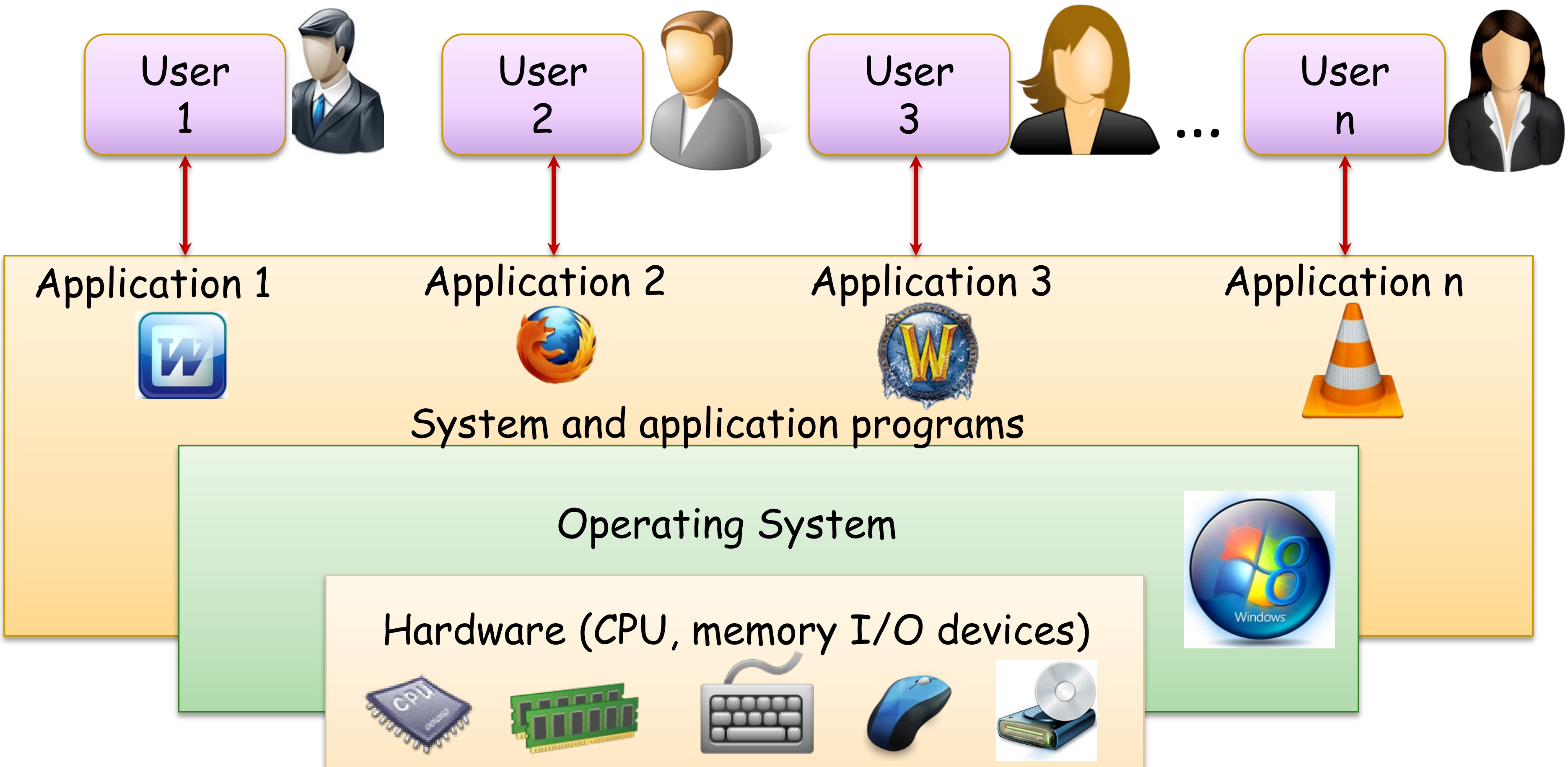
The Diversity of Operating Systems





OS's Place in the Overall System

- Components of a computer system: application programs, operating system, and computer hardware.



Mismatch between hardware capabilities and user needs

Hardware component	Hardware capabilities	User needs
CPU	Machine instructions perform operations on contents of registers and memory locations.	The user thinks in terms of arrays, lists, and other high-level data structures, accessed and manipulated by corresponding **high-level** operations.
Main memory	Physical memory is a **linear** sequence of addressable bytes or words that hold programs and data.	The user must manage a **heterogeneous** collection of entities of various types and sizes, including source and executable programs, library functions, and dynamically allocated data structures, each accessed by different operations.
Secondary storage	Disk and other secondary storage devices are **multi-dimensional** structures, which require **complex** sequences of **low-level** operations to store and access data organized in discrete blocks.	The user needs to access and manipulate programs and data sets of various sizes as individual named entities **without any knowledge** of the disk organization.
I/O devices	I/O devices are operated by reading and writing registers of the device controllers.	The users needs **simple**, **uniform interfaces** to access different devices without detailed knowledge of the access and communication protocols.

What does an operating system do?

- Overall: provides the means for proper use of system resources e.g., hardware, software, and data.
  - Bridging the hardware/user gap
  - Similar to a government, OS serves no useful function when by itself. It only provides an environment in which user programs can do useful work.

What does an operating system do?

- Users and computers perceive the OS differently.
- User perspective of OS: OS is something that makes computers easier to use and simplifies problem solving.
  - Varies according to how users interact with the system:
    - Example: Most users interact with an OS using a keyboard, monitor, and mouse and a GUI.
      - Such systems are designed for ease of use and performance; strive to maximize productivity (or play).
    - Example: Multiple users connect to a mainframe computer using a terminals:
      - Such systems are designed to maximize resource utilization and fair sharing of resources among users.

What does an operating system do?

- System Perspective:
  - Resource allocator: allocates and manages hardware resources e.g. CPU time, memory space, disk space, and I/O.
  - Control program: manages the execution of user programs.

Process Management

- Process: a unit of work on the system
- Program vs. Process:
  - Program: is a set of instructions (a passive entity).
  - Process: a program in execution (an active entity).
- Processes require resources to run:
  - CPU time
  - Memory
  - Files
  - I/O devices

Process Management: To-do

- Operating system must:
  - Act as an intermediary between the process and the rest of the system.
  - Allocate resources when the process starts, manage them while it runs, and reclaim them when the process terminates.
  - Multiplex resources among multiple processes.
  - Provide means for suspending/resuming a process.
  - Provide means for process synchronization.
  - Provide means for inter-process communications.

Process Management: accessing hardware

- OS acts as an intermediary between a process and the rest of the system:
  - Processes are restricted from accessing hardware directly.
  - If process needs to access hardware (e.g., open a file on the disk), the process must request OS to perform the access on the process's behalf.
    - Shifts the burden of dealing with hardware peculiarities from the application developers to the OS.
    - Allows OS to enforce order e.g., deny invalid/unauthorized accesses.

Process Management: Request Services

- OS exposes a set of system calls: functions which processes can invoke to request services from the operating system such as:
  - Read file from the disk
  - Send data over the network
  - Send message to another process

Process Management: Linux System Calls

- Example: Some Linux system calls:
  - sys_open(): open a file
  - sys_close(): closes the file
  - sys_read(): reads from file
  - sys_write(): writes to file
  - Linux system calls are defined in part of the operating system called the system call table:
  - https://chromium.googlesource.com/chromiumos/docs/+/master/constants/syscalls.md

Process Management: Two Modes

- A typical computer supports two modes of execution (i.e., dual-mode operation): user mode and system mode:
  - User mode (i.e., unprivileged mode): when a process is executing.
    - Allows the process to execute only unprivileged instructions e.g., addition, subtraction, logical operations, etc.
    - CPU ***restricts*** execution of privileged instructions (e.g., instructions for directly accessing hardware, managing OS timers, etc).
  - System mode (i.e., privileged mode): when the OS is executing.
    - CPU ***allows*** execution of privileged instructions.

Process Management: Example

- Example: typical process execution case:
  - 1. Process begins by executing unprivileged instructions and then needs to read a file from the disk (i.e., an I/O resource).
  - 2. Process invokes a system call =>
    - 1. system switches from user mode to system mode
    - 2. OS performs the disk access service the process requested and returns the read data to the process.
    - 3. If the file cannot be read (e.g., file does not exist, process lacks the proper privilege), the OS returns an error to the process.
  - 3. OS completes the requested service => system switches to user mode.

Process Management: Two Modes Benefits

- Why two modes?
  - To protect OS from the errant programs/users.
  - To protect errant users/programs from each other.
  - Example:
    - 1. Process is executing (i.e., the system is in user mode)
    - 2. Process invokes an instruction to update an OS timer—a privileged instruction!
    - 3. The CPU sees an attempt to invoke a privileged instruction in the user mode, and prevents it.

Process Management: Mode bit

- To know what mode the system is currently in, CPU maintains a mode bit
  - Set to 0 when in system mode.
  - Set to 1 when in user mode.
  - Before executing a privileged instruction, the CPU verifies that mode bit == 0.

Memory Management

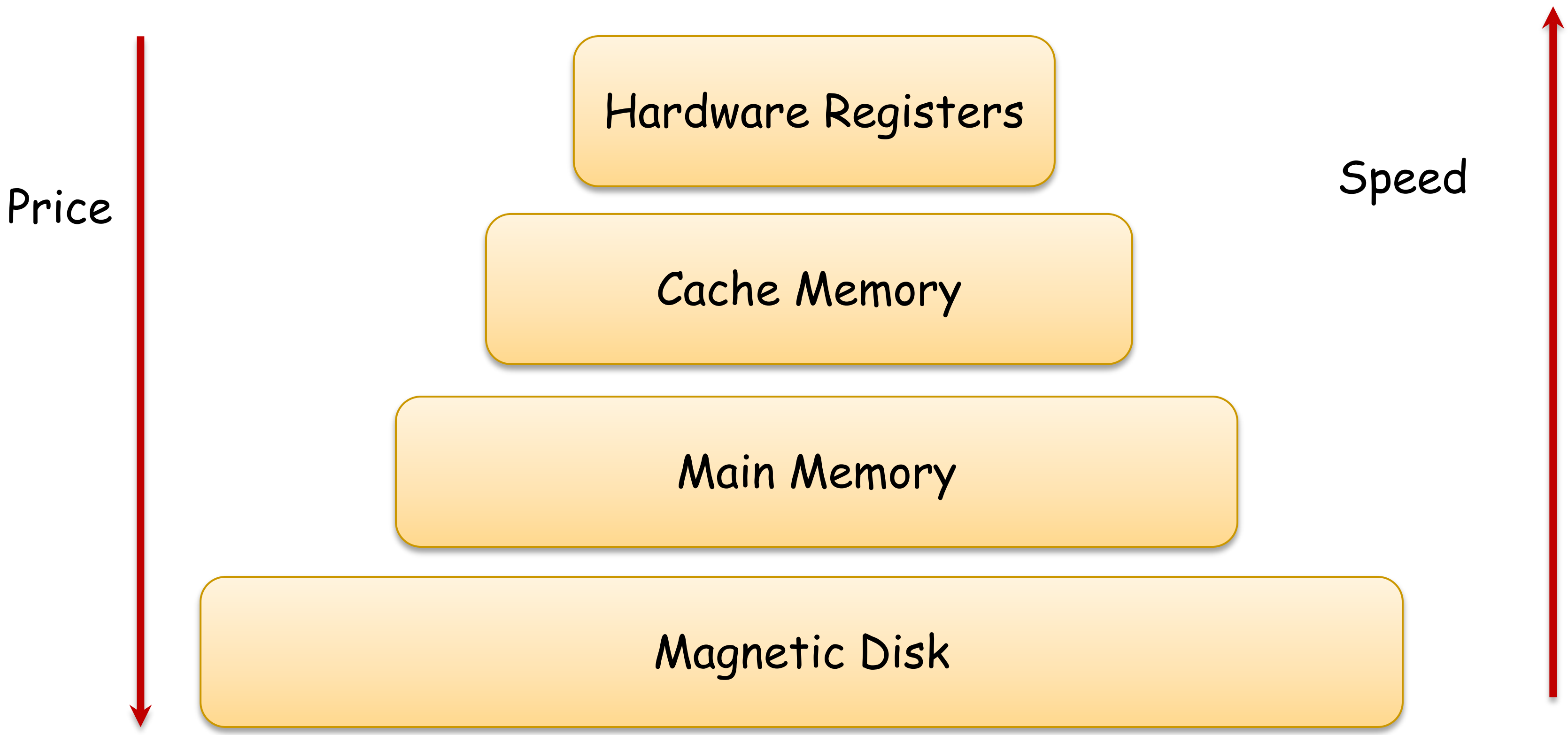
- Main memory:
  - A large array of bytes or words where each byte or word has its own address.
  - The only large storage directly accessible by the CPU.
- For improved resource utilization, several programs are usually kept in memory:
  - Introduces the need for memory management.
- Memory Management:
  - Keep track of what parts of memory are being used by what processes.
  - Decide which processes (or parts of) to move into and out of memory.
  - Allocate and deallocate memory as needed.

Storage Management: Files

- For convenience, the information on the storage device is abstracted into units called files.
- File: collection of related information defined by its creator.
- Operating system file management services:
  - Creating and deleting files.
  - Creating and deleting directories (i.e., collections of files).
  - Mapping files to memory on the storage device.
  - File Backup.
- Mass-storage (e.g., disk) management:
  - Managing free space.
  - Storage allocation.
  - Disk scheduling (i.e., managing multiple operations that read/write to/from the disk).

Storage Management: Hierarchy

- Caching: temporarily store data/instructions in faster storage (i.e., cache) and access them from there.



- Must ensure data consistency along all levels of the hierarchy.

I/O Management

- Purpose: hide the peculiarities of specific devices from the user.
- Example: the Unix I/O subsystem provides:
  - Functionality to manage data buffering, caching, and spooling.
  - A general device driver interface.
  - Drivers for specific hardware devices (which know how to control the specific device).

I/O Management: Example

- Example: How Unix/Linux I/O subsystem provides uniform interface for wide range of devices:
  - This code works on many different devices:

```
// Linux abstracts devices as files; application developers can
// read/write data to/from devices just like reading/writing files.

// Open the device (just like opening a file!)
int fd = open("/dev/somedevice");

// Write 10 integers to the device
for(int iCount=0; iCount < 10; ++iCount){
   fprintf(fd, "%d\n", iCount);
}

close(fd)
// Example from: http://www.cs.columbia.edu/~krj/os/lectures/L24-IO.pdf
```

Protection and Security

- Protection: controls the access of users and processes to the system resources (e.g., memory, files, etc).
  - Internal to the OS
  - Example: process A attempts to (illegally) write to the memory of process B. The operating system detects the violation and terminates process A.
- Security: defending the OS against external threats.
  - Example: malware (e.g., viruses, worms, etc).
- Protection vs. Security: definitions vary

Computing Environments: Distributed Systems

- A collection of physically separate, networked systems that share resources.
- Advantages of resource sharing: increases computation speed, functionality, data availability, and reliability.
- Distributed systems can be interconnected via:
  - Local-Area Network (LAN): connects systems within a single floor room or building, or
  - Wide-Area Network (WAN): connects buildings, cities, or countries.

Student Participation

What multiprocessors and multi-computers (distributed systems) have in common is ____

A) the same type of applications
B) the same type of interconnection networks
C) the ability to carry out operations in parallel

Computing Environments: Mobile Computing

- Computing on handheld smartphones and tablet computers.
- Sacrifice screen size, memory capacity, CPU power, in favor of portability:
  - This will change as mobile devices grow more powerful.
- Allows for applications impractical on traditional systems (e.g. laptops, desktops, etc).
  - Navigation
  - Augmented reality
  - Many others!
- Dominant OSs:
  - iOS: designed for Apple's iPhone and iPad platforms.
  - Android: runs on all sorts of devices.

Real-Time Embedded Systems

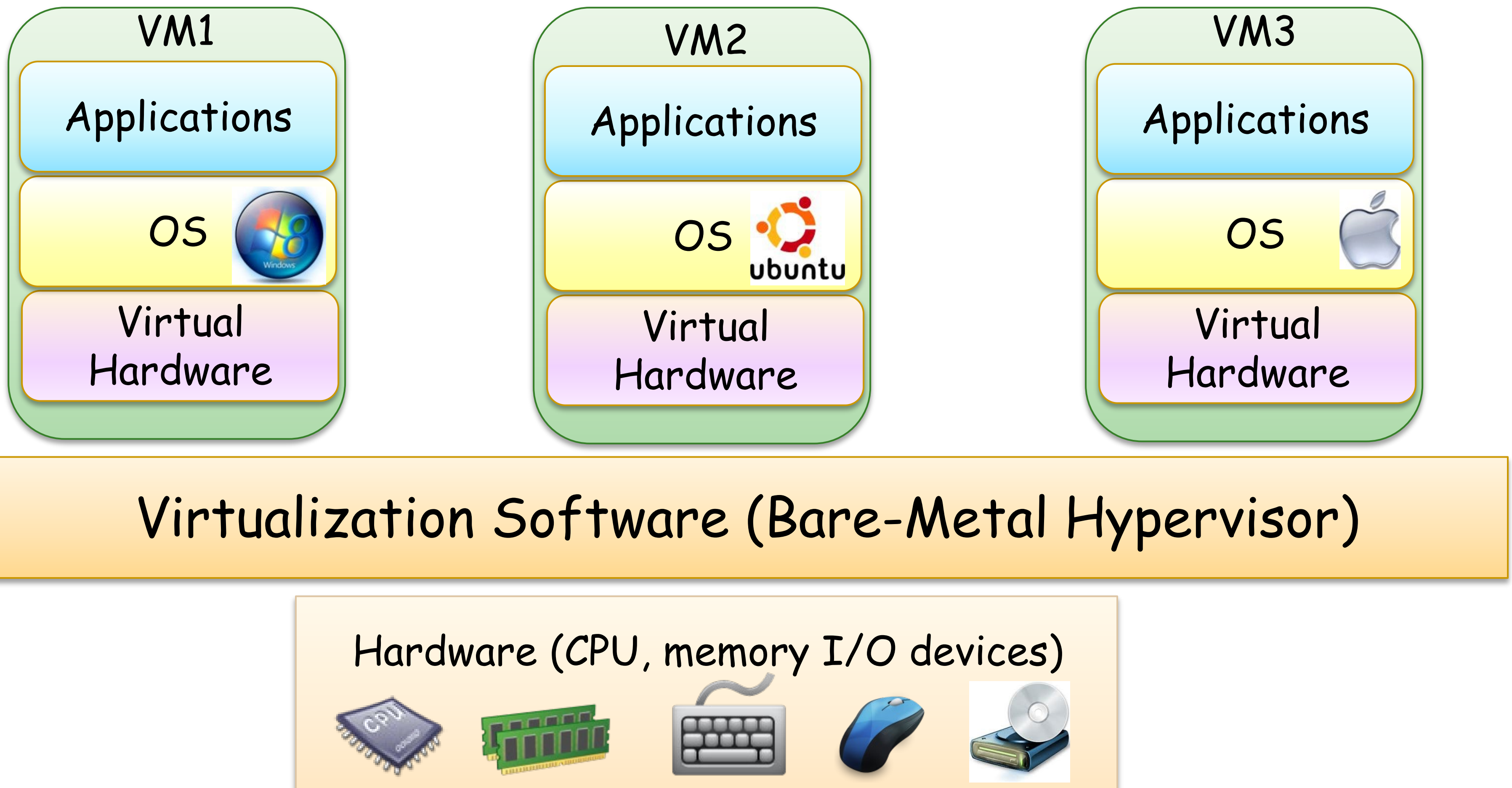
- Embedded Systems: systems dedicated to specific tasks e.g. controlling car engines, robotic arms, etc.
  - Usually have little to no user interface.
  - One of the most prevalent types of computers.
- Real-Time Systems: systems where tasks must complete within the defined timing constraints.
  - Example: the robotic arm must stop moving before it smashes into a car it was building (not after!).
  - Embedded systems usually run real-time operating systems.

Multimedia Systems

- Systems specialized to deliver media content (i.e. a mix of text, video, sound, etc).
- Some media must be delivered within certain timing constraints.

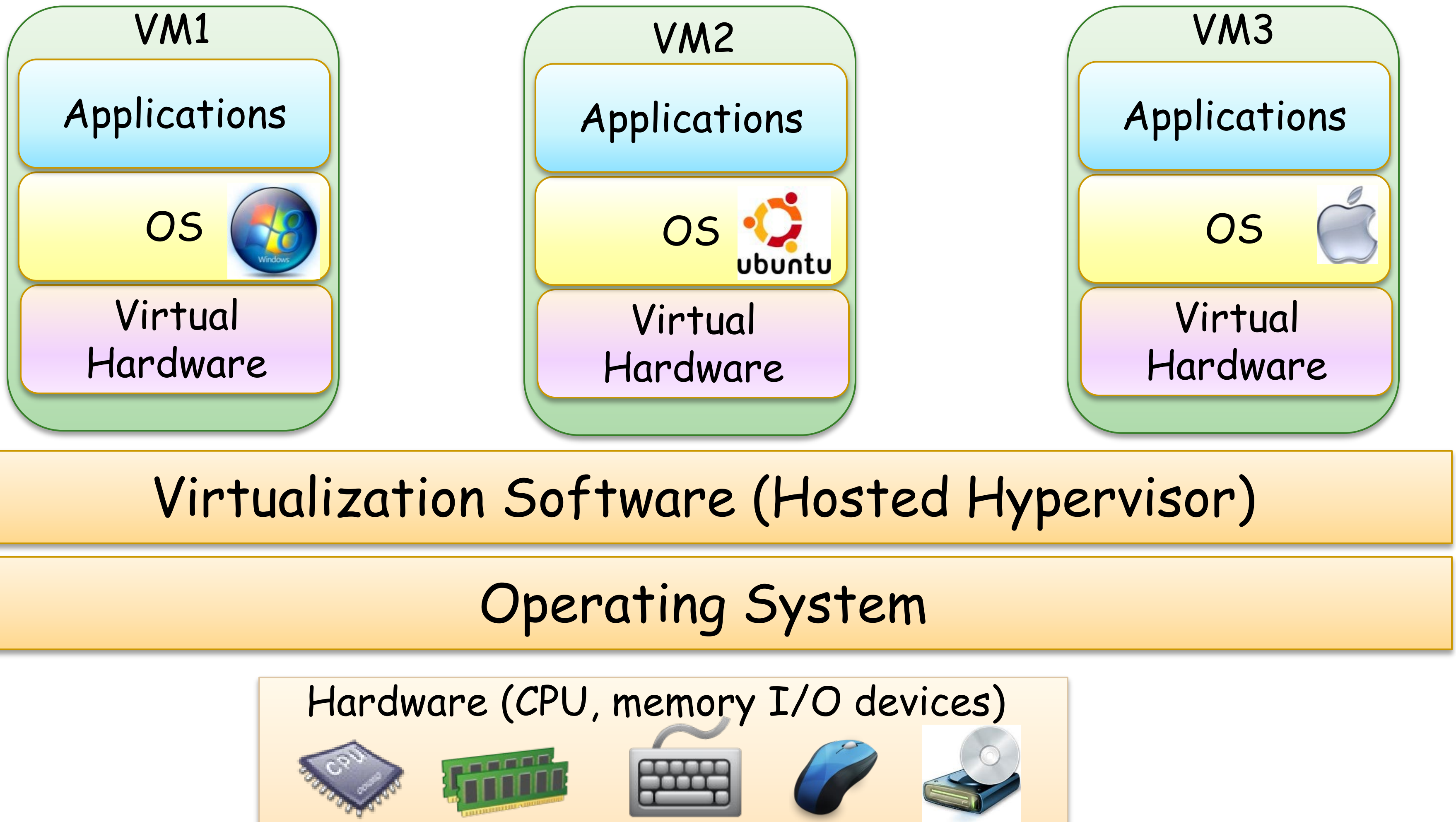
Computing Models: Virtual Machines (1)

- Virtual Machines (VMs): a software that can run its own operating system and applications just like a real physical system.



Computing Models: Virtual Machines (2)

- Virtual Machines (VMs): a software that can run its own operating system and applications just like a real physical system.



Computing Models: Cloud Computing

- Delivers services over the network.
- Three types of cloud computing services:
  - Infrastructure-as-a-Service (IaaS): provides hardware resources e.g., storage, CPU, and networking services.
  - Platform-as-a-Service (PaaS): provides hardware and platform ready for applications (e.g., a database server).
  - Software-as-a-Service (SaaS): provides software applications over the network.
- Helps reduce operating costs.
- Helps to improve resource utilization (by combining computing resources).
- Makes it easier to tackle large scale computing problems.

Computing Models: Cloud Computing Types

- Types of clouds:
  - Public: available to anybody willing to pay for the services.
    - Example: Amazon's Elastic Cloud (EC2)
    - Example: Google's App Engine
  - Private: a cloud run by a company for its own use.
    - Example: IBM SmartCloud Foundation
  - Hybrid: a combination of public and private.