COMP3511 Operating Systems

Topic 1: Introduction

Dr. Desmond Tsoi

Department of Computer Science & Engineering
The Hong Kong University of Science and Technology
Hong Kong SAR, China

Acknowledgment: The lecture notes are based on various sources on the Internet

Computing Devices are Everywhere

Laptops, Tablet PCs, Smart Phones

Servers

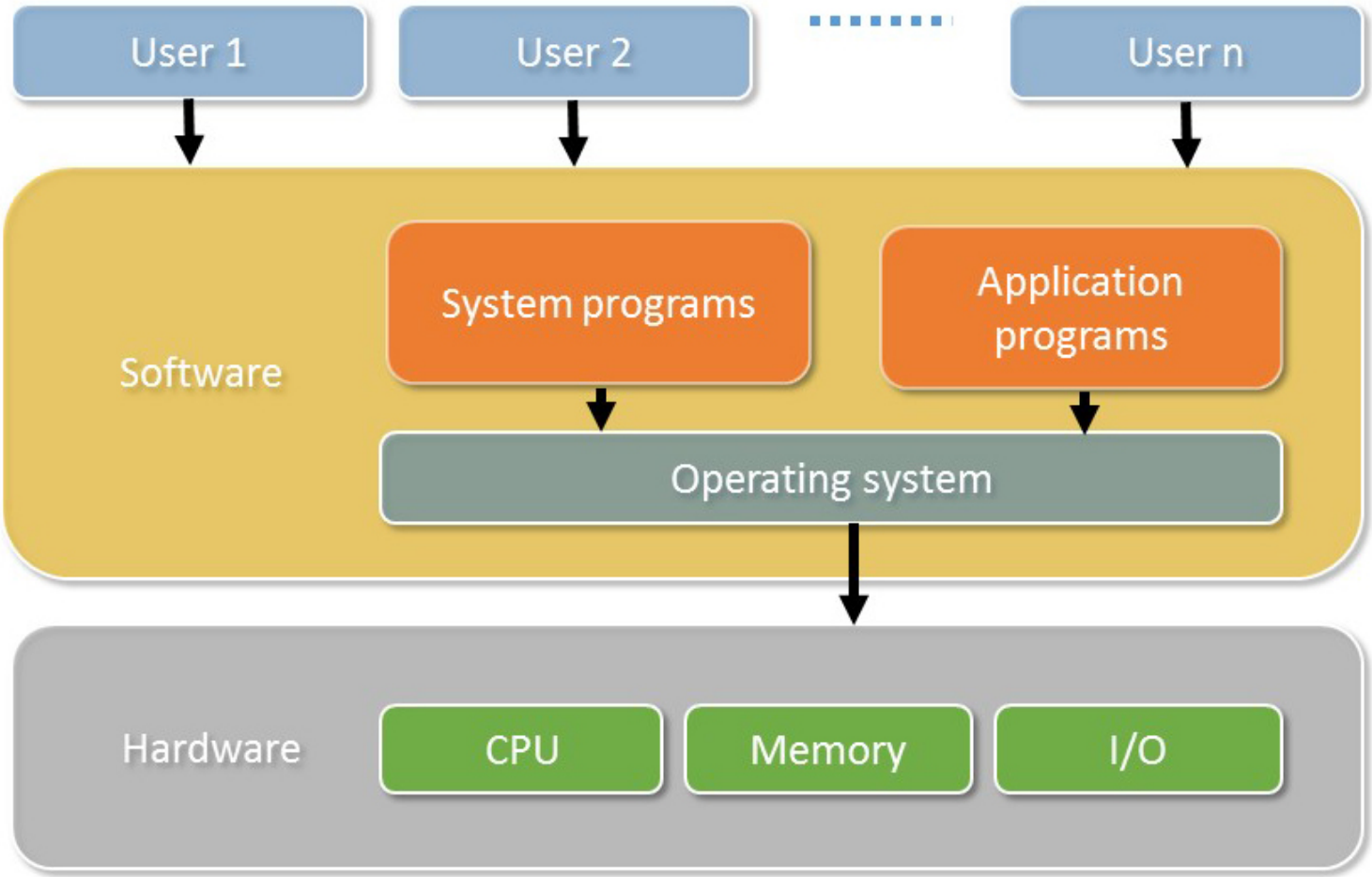
Wearable Devices

Car Navigation Systems

Basic Components of a Computer System

- Hardware: It provides basic computing resources
  - Central Processing Unit (CPU)
  - Memory
  - I/O devices
- Software
  - Operating System: It controls and coordinates use of hardware among various applications and users
  - System and application programs: The define 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

Pictorial View of Basic Computer System Components



- System programs: Compiler, assembler, text editor, ...
- Application programs: Word processors, spreadsheet application, games, ...

Why Study Operating Systems?

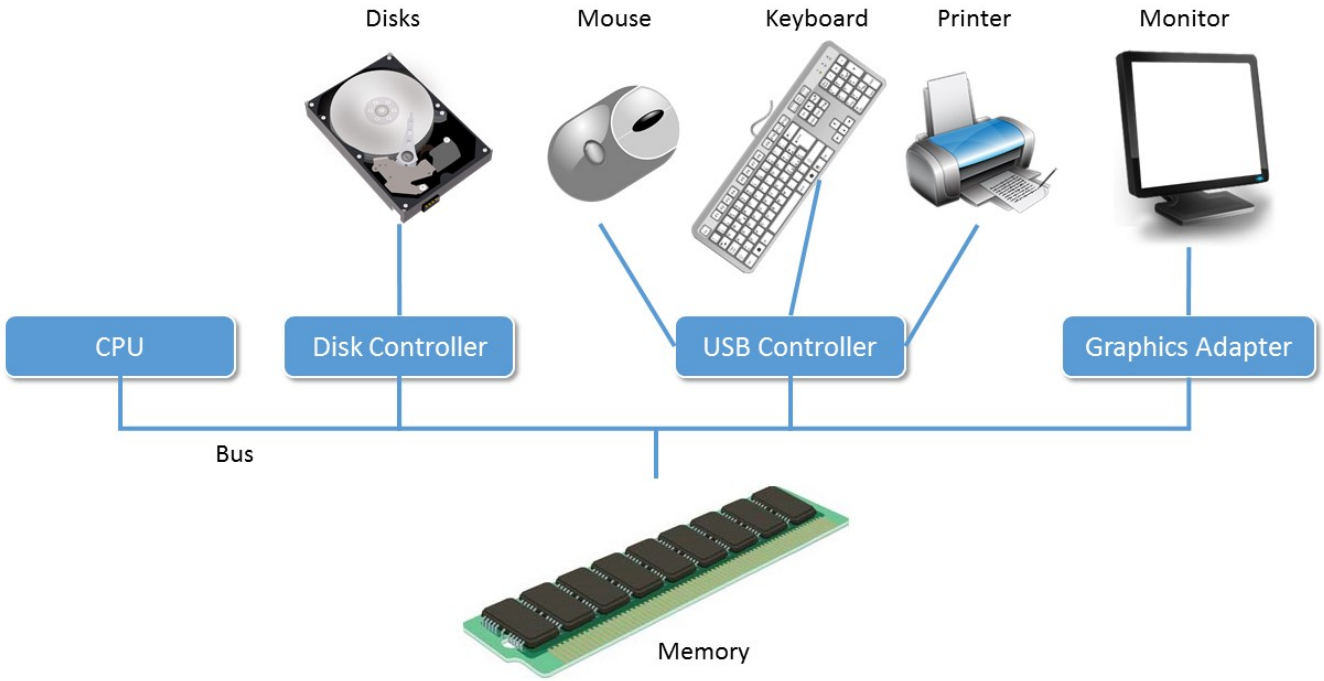
- Operating System (OS) is one of the most fundamental software systems
- By studying OS, you will gain a good understanding of the big picture on how do hardware, programming language, compiler, algorithms, and OS work together
- The knowledge is a vital basis for understanding further about programming, networks, parallel and distributed systems organization
- Possibly, you may
  - get a job at Google / Microsoft / VMware
  - get started in OS research
  - apply OS concepts to your area
  - ...

OS X

Computer System Organization and Concepts

- Computer Organization
- Interrupts and Interrupt Service Routines (ISR)
- Direct Memory Access (DMA)
- Storage Hierarchy Structure
- Caching

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 cycles
- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device and 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 is has finished its operation by causing an interrupt

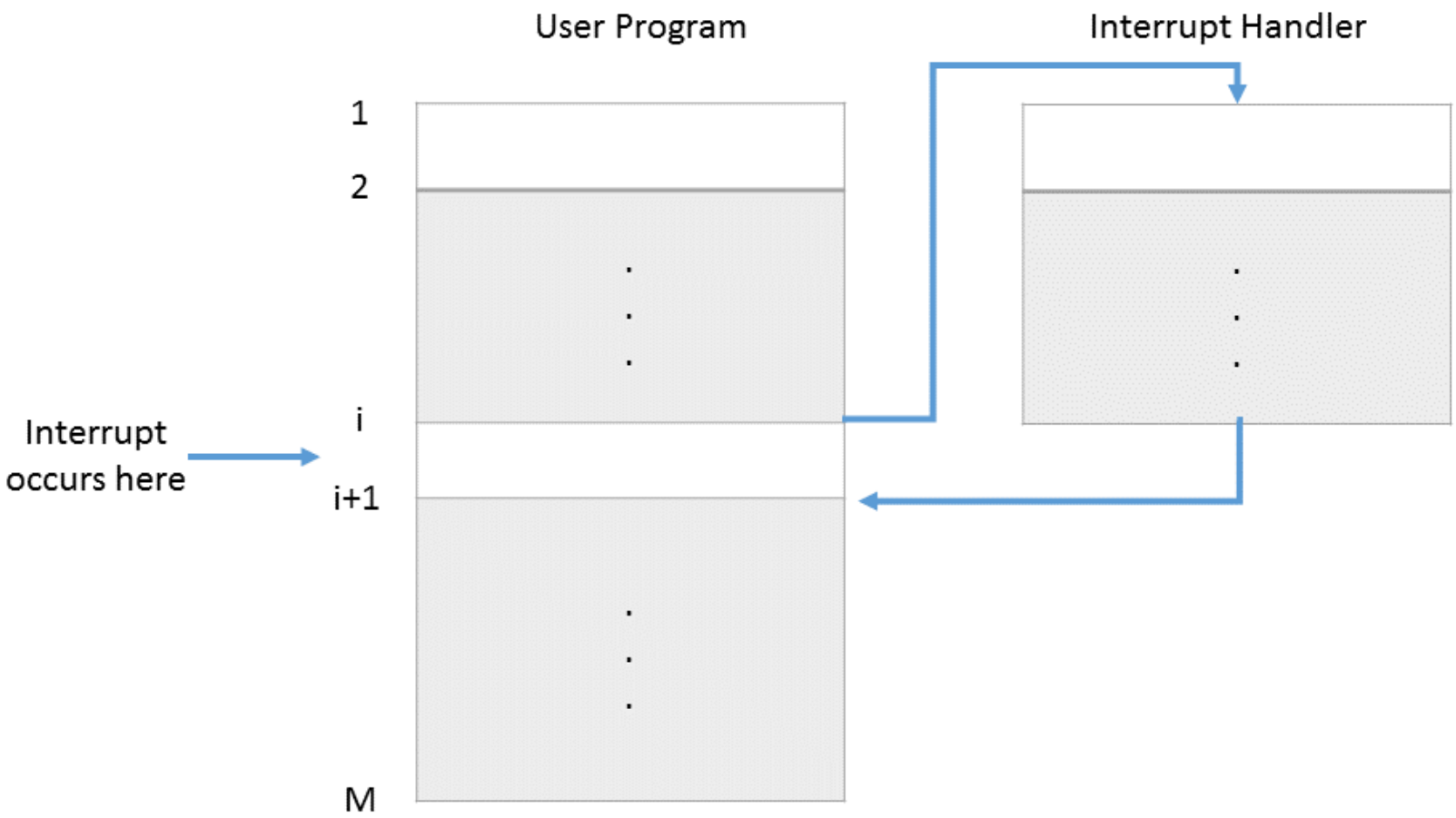
Interrupts

- Interrupts are signals sent to CPU that a running program can be stopped to allow the operating system to do something immediately
- Some activities require the CPU to respond quickly. A very short program Interrupt Service Routine (ISR) (also called interrupt handler) may be all that is necessary to handle a situation, but that program has to be run very shortly after the situation occurs
  - The addresses of all ISRs are stored in an interrupt vector
  - To determine which type of interrupt has occurred, polling (polls and tests every device in turn) and vectored interrupt system (interrupt controller tells the CPU where the interrupt came from) is used
  - Incoming interrupted are disabled while another interrupt is being processed to prevent a lost interrupt
- Interrupts alter a program's flow of control. When an interrupt occurs, the hardware execute the instructions (ISR) at a specified address instead of following the normal program flow
- User programs are interrupted all the time

Types of Interrupts

- External: Generated by an I/O device
  - I/O devices tell the CPU that an I/O request has completed by sending an interrupt signal to the processor
  - I/O errors may also generate an interrupt
  - Most computers have a timer which interrupts the CPU every so many milliseconds
- Internal: Exception / Traps within a program
  - When the hardware detects that the program is doing something wrong, it will usually generate an interrupt
  - Examples: Arithmetic error (e.g., division by zero), addressing error (e.g., bad pointer), invalid instruction, hardware malfunction
  - Internal interrupts are sometimes called exceptions
- Program Generated: Used to transfer control to the operating system
  - Most computers have an instruction that generates an internal interrupt
  - Program generated interrupts are a mean for user programs to call a function of the operating system
  - Some systems refer these interrupts as a SuperVisor Call (SVC)

Transfer of Controls via Interrupts and ISR

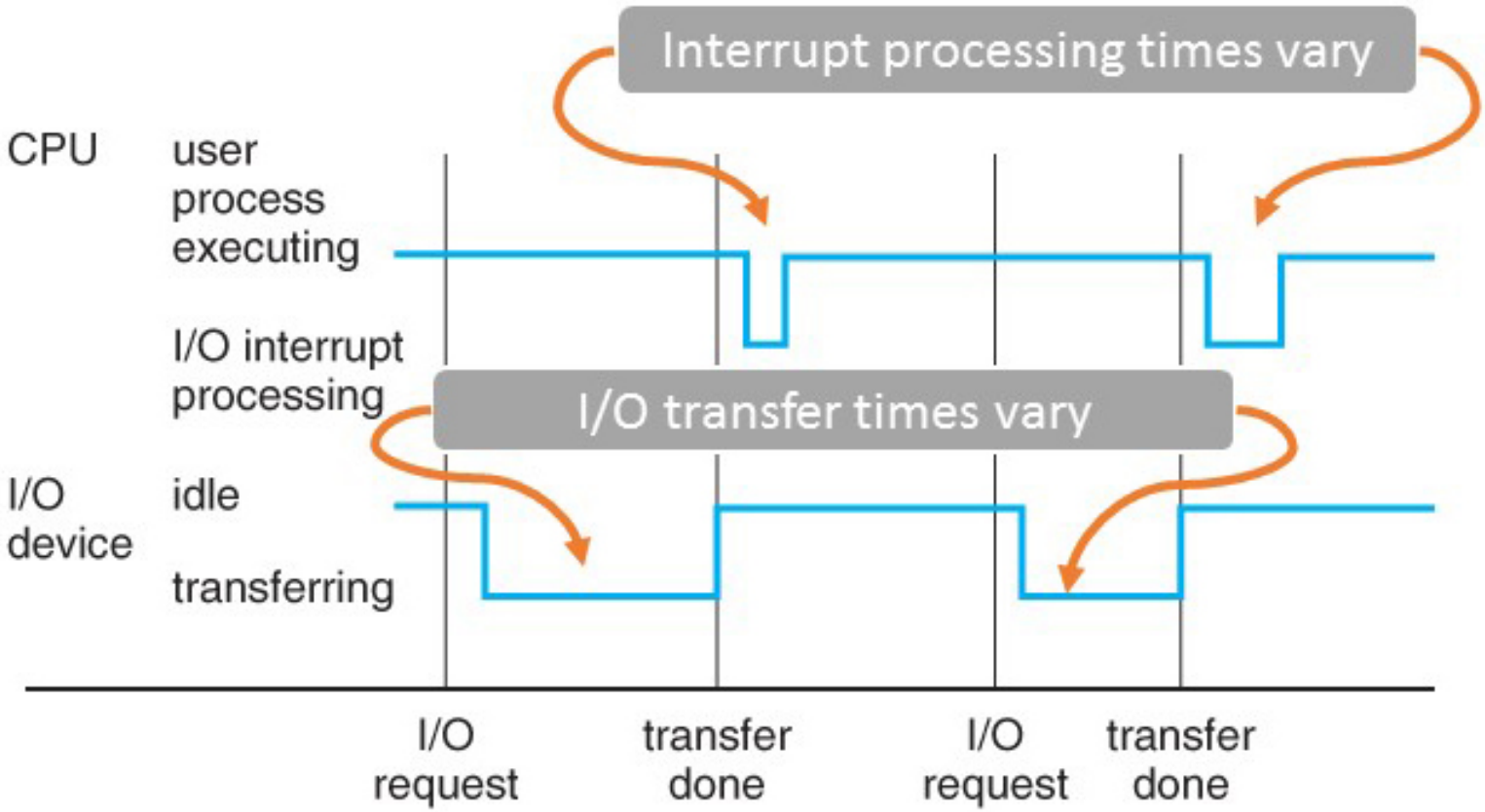


- When an interrupt occurs, execution starts in an ISR
- The ISR processes the event or queues a program to process the event
- After an external interrupt, the service routine will return to the program

Details of Interrupt Processing

1. I/O device issues the interrupt
2. The processor finishes the execution of an instruction
3. The processor checks for an interrupt. If there is one, it then sends an acknowledgment signal to the I/O device that issued the interrupt. This signal allows the device to remove the signal
4. To switch to run ISR, information about the current program is stored, so that its execution may be resumed later
5. The processor loads the entry location of interrupt handler
6. The handler performs the interrupt processing
7. When the handler finishes, the saved register values are stored into the registers that originally hold them when the interrupt handler returns
8. Finally, the interrupt program is restored, thus the program may continue to execute

Interrupt Timeline



Multiple Interrupts

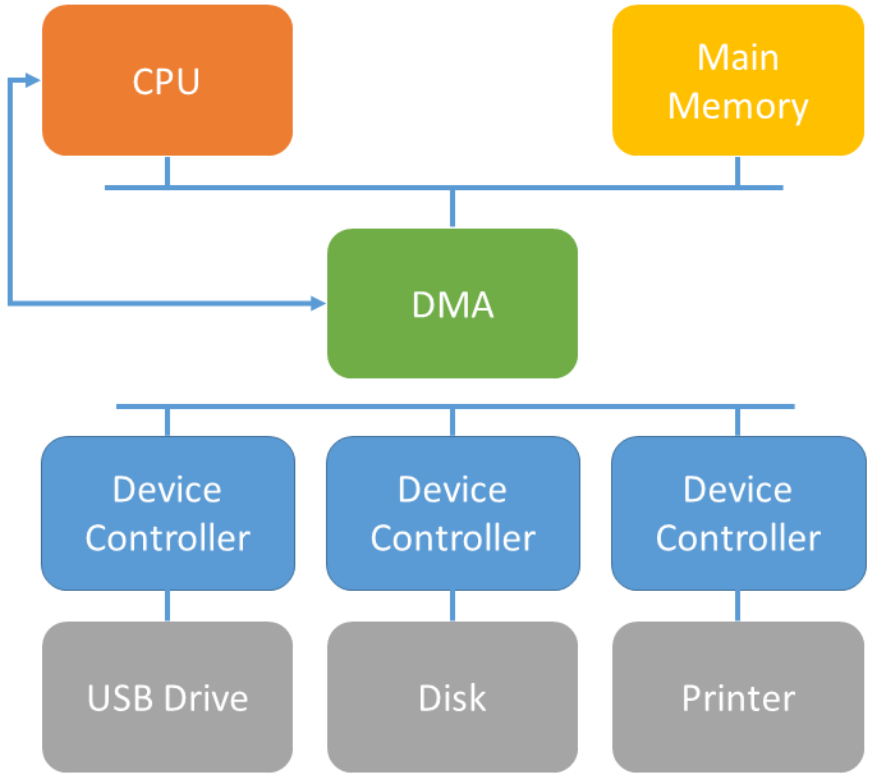
It is possible to have more than one interrupt requests happened at the same time. Two different approaches to handle this

- Sequential interrupt processing
  - Disable interrupt request while an interrupt is being processed
  - All interrupts will be processed sequentially
- Nested interrupt processing
  - All the interrupts are assigned different priorities, so that whenever an interrupt occurs while an interrupt handler is running, their priorities will be compared first, and the further action will be determined according to the result

Direct Memory Access (DMA)

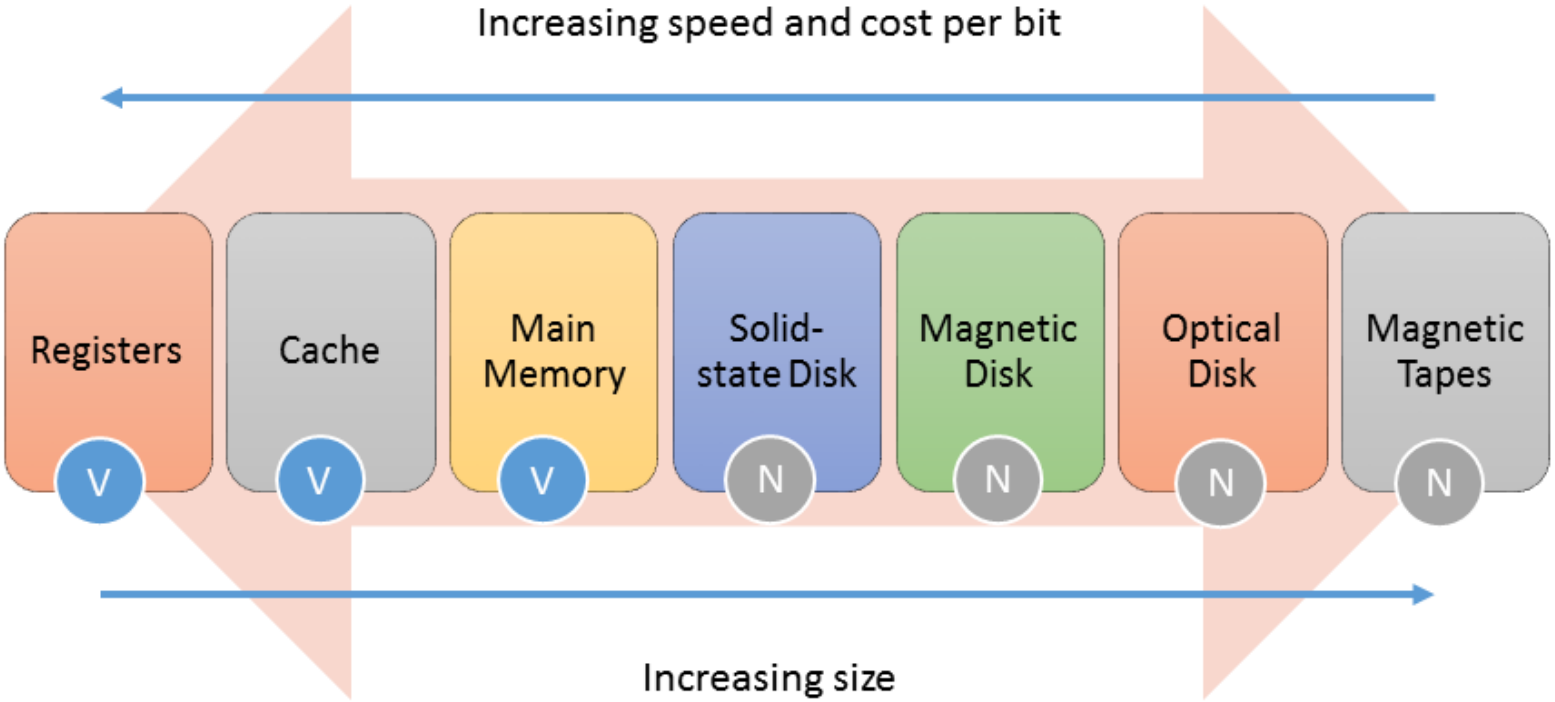
The problem with interrupted-based I/O operations is that the processor has to be involved all the way through the I/O processing

- When large volumes of data are to transfer between memory and I/O devices, the processor will be interrupted hundreds or even more times to process interrupts
- To resolve this, Direct Memory Access (DMA) is used
  - 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, so CPU can be released to execute instructions for other process or program



Storage Hierarchy Structure

Storage systems are organized in hierarchy according to their speed, cost, size and volatility



V is denoted as volatile and N is non-volatile

Storage-Device

- Register (Volatile):
  - In CPU, high access speed
- Cache (Volatile):
  - May be located on CPU or module, fast memory allowing access speed approaching register speed
- Main memory (Random Access Memory (RAM)) (Volatile):
  - Only large storage media that the CPU can access directly
- Secondary storage (Solid-state Disk and Magnetic Disk) (Non-volatile):
  - Extension of main memory that provides large non-volatile storage capacity
- Tertiary storage (Optical Disk / Magnetic Tapes) (Non-volatile):
  - CD-ROM and tape used primarily for backup and archival data

Performance of Various Levels of Storage

Level	1	2	3	4
Name	Registers	Cache	Main memory	Disk storage
Typical size	<1 KB	>16 MB	>16 GB	>100 GB
Implementation technology	Custom memory with multiple ports, CMOS	On-chip or off-chip CMOS SRAM	CMOS DRAM	Magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	5,000.000
Bandwidth (MB/sec)	20,000 - 100,000	5000 - 10,000	1000 - 5000	20 - 150
Managed by	Compiler	Hardware	Operating system	Operating system
Backed by	Cache	Main memory	Disk	CD or tape

Note

The performance data is updated constantly

Caching

- The idea of caching is to store copies of data at places that can be accessed more quickly than accessing the original
- Information is copied from slower to faster storage on a temporary basis
- Assumption: Data will be used again soon
- Faster storage (cache) first checks to determine if information is there
  - HIT: If it is, information used directly from the cache (fast)
  - MISS: If not, data copied from slower storage to cache and used there
- Cache smaller than storage being cached
  - Cache management is an important design problem
  - Cache size and replacement policy

Computer System Architecture

- Single-processor & multi-processor systems
- Symmetric multiprocessing (SMP)
- Uniform Memory Access (UMA) & Non-uniform Memory Access (NUMA)
- Clustered Systems

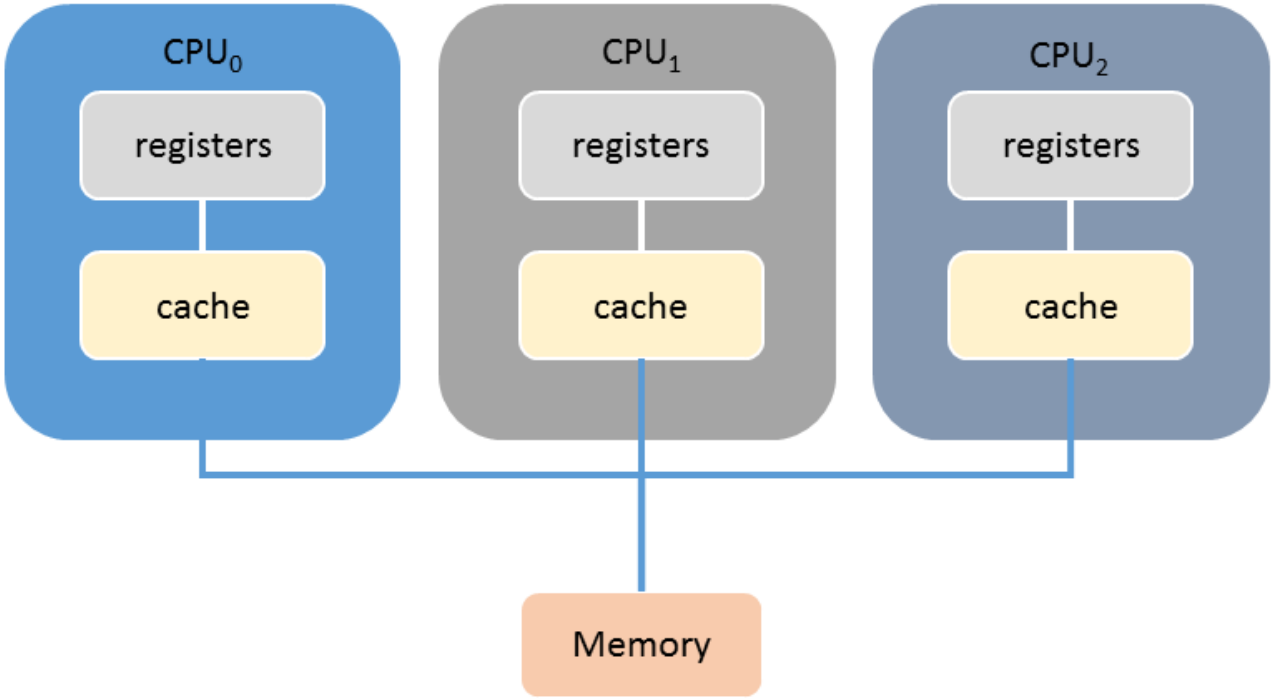
Single-Processor & Multiprocessor Systems

- Most systems used single general-purpose processor
  - Most systems have special-purpose processors as well, which may come in the form of device-specific processors, such as disk and graphics controllers, or on mainframes like I/O processors
  - All of these special-purposes processors run a limited instruction set and do not run user processes
- Multiprocessors systems have begun to dominate the landscape of computing, also known as parallel systems, or multi-core systems
  - Two or more processors in close communication, sharing computer bus and sometime clock, memory, and peripheral devices

Multiprocessor Systems

- Advantages include:
  - Increased throughput: more work can be done in less time
  - Economy of scale: cost less than a number of individual single-processor system
  - Increased reliability: failure of any processor will not affect the functionality of the system
- Two types
  - Asymmetric multiprocessing: One master CPU distributes tasks among multiple slave CPUs, or boss-worker relationship
  - Symmetric multiprocessing (SMP): Most common

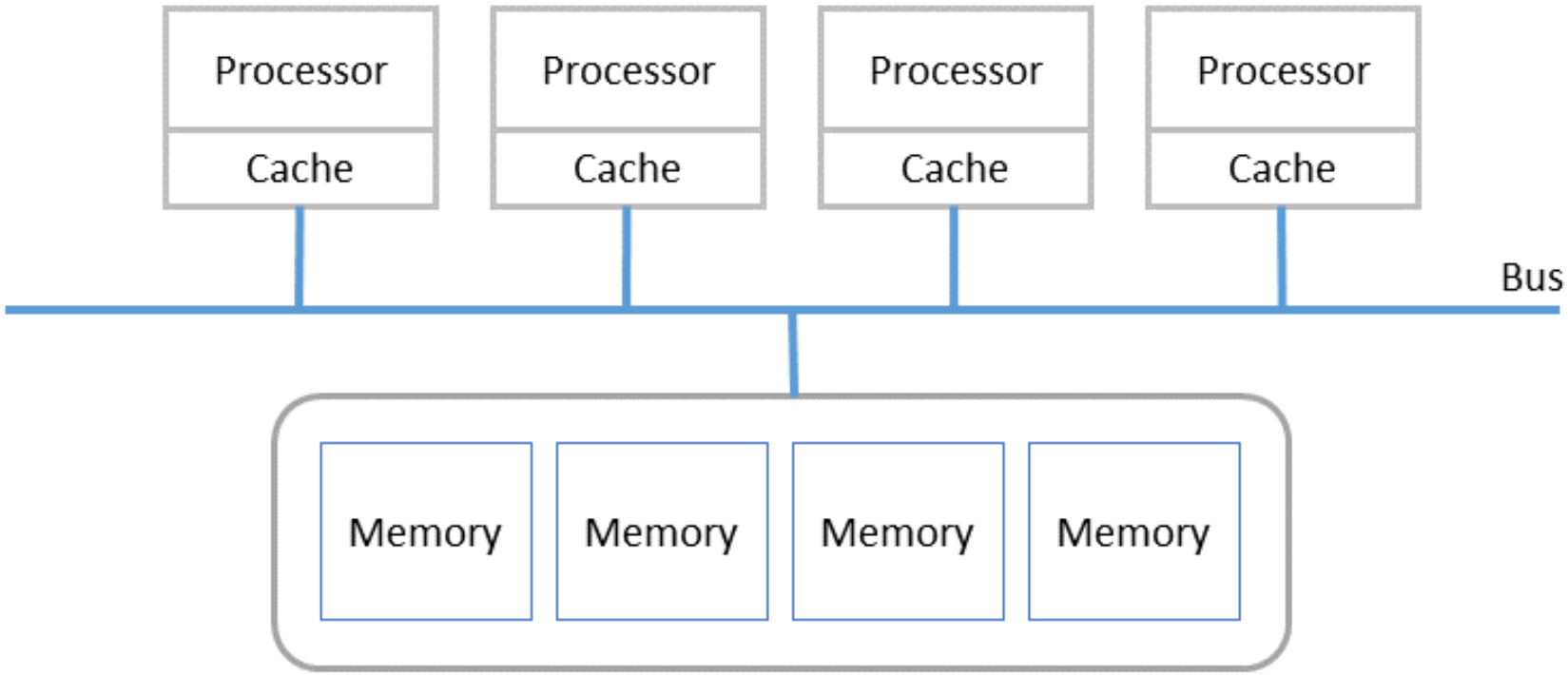
Symmetric Multiprocessing (SMP)



- Each processor has its own set of registers and local cache
- All processors share physical memory
- Multi-core: multiple computing cores on a single chip, faster (on-chip communications) and less power than multiple single-core chips

Uniform Memory Access

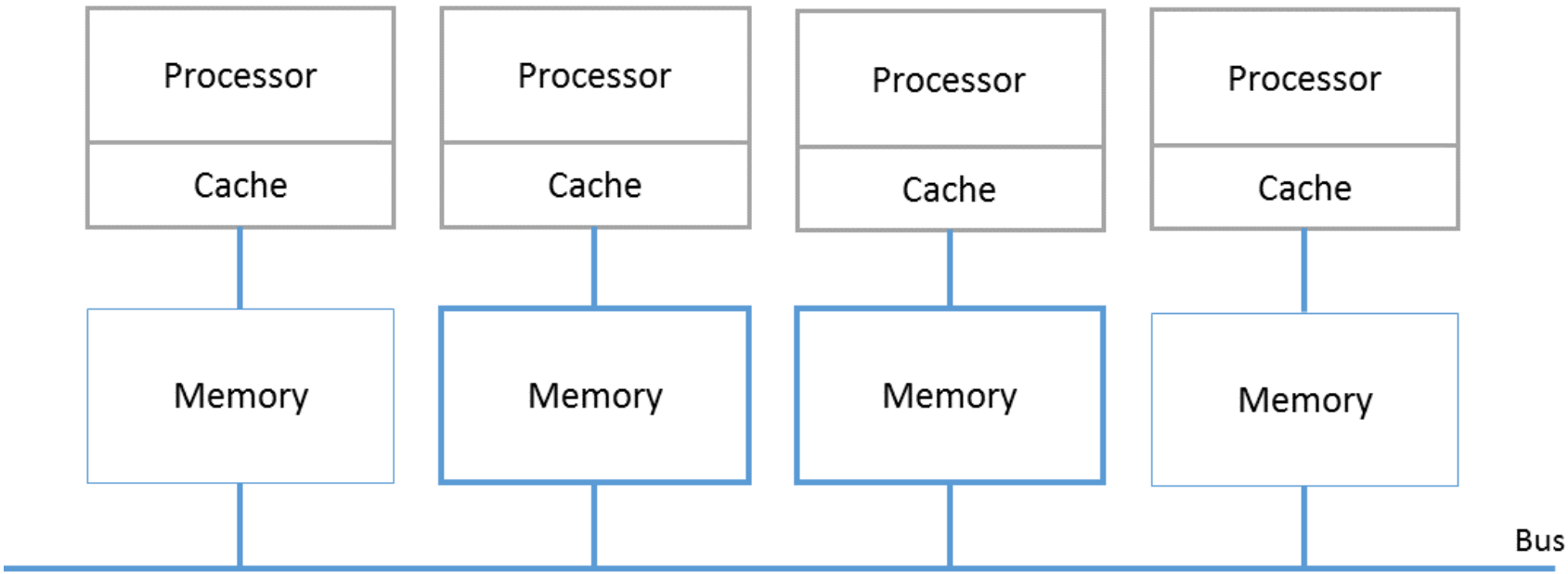
- Multiprocessing can change memory access mode from Uniform Memory Access (UMA) to Non-uniform Memory Access (NUMA)
- In UMA, access to any RAM from any CPU takes the same amount of time regardless of which shared memory module contains the data to be retrieved, since all processors access shared memory through a shared bus



Uniform Memory Access

Non-uniform Memory Access

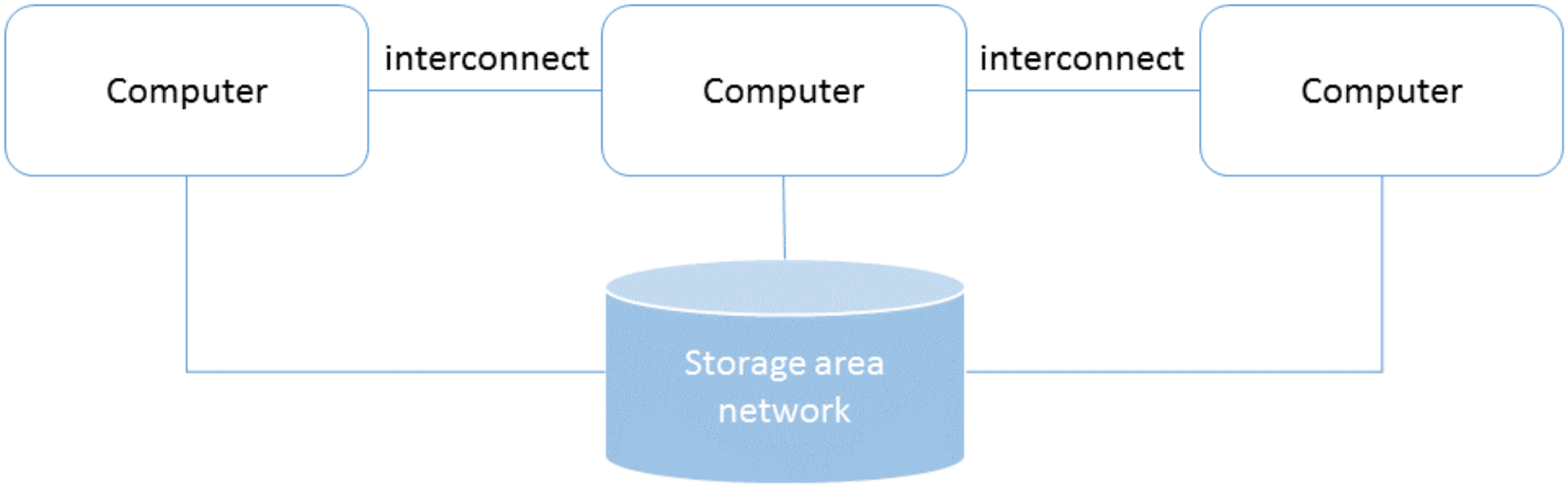
- In NUMA, each processor has its own local memory and it can access any memory module belonging to another processor using a shared bus
- If data resides in local memory, access is fast. If data resides in remote memory, access is slower. This creates a performance penalty, which OS needs to minimize



Non-Uniform Memory Access

Clustered Systems

- Unlike multiprocessor systems, a cluster system is composed of two or more individual systems, and considered loosely coupled
- Each system can be a single-processor system or a multi-core



Clustered Systems (Cont'd)

- A cluster system usually shares storage via a storage-area network (SAN) and provides a high-availability service which survives failures
- Two types
  - Asymmetric clustering has one machine in hot-standby mode (does nothing but monitoring) while other is running applications
  - Symmetric clustering has multiple nodes running applications, and are monitoring each other
- Some clusters are for High-Performance Computing (HPC): Applications must be written to use parallelization to run on all computers in the cluster concurrently
- Some have Distributed Lock Manager (DLM) to avoid conflicting operations when accessing shared data

How do we tame complexity?

- Every piece of computer hardware is different
  - Different CPU
    - ★ Pentium, PowerPC, ColdFire, ARM, MIPS
  - Different amounts of memory disk
  - Different types of devices
    - ★ Mice, Keyboards, Sensors, Cameras, Fingerprint readers
  - Different networking environment
    - ★ Cable, DSL, Wireless, Firewalls...

Questions

- Does every program have to be altered for every piece of hardware?
- Does a faulty program crash everything?
- Does every program have access to all hardware?
- ...

Fundamentals of Operating Systems

- What is Operating System (OS)?
- Overview of how OS works?
- What does OS do?
- OS implementation
- Multiprogramming and multitasking
- Dual-mode operation
- Timer
- Open-Source Operating Systems

What is an Operating System?

Definition: Operating System

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

Operating System goals:

- Control and coordinate the use of system resources (hardware and software)
- Use the computer hardware in an efficient and protected manner
- Make the computer system convenient to use for users (services)

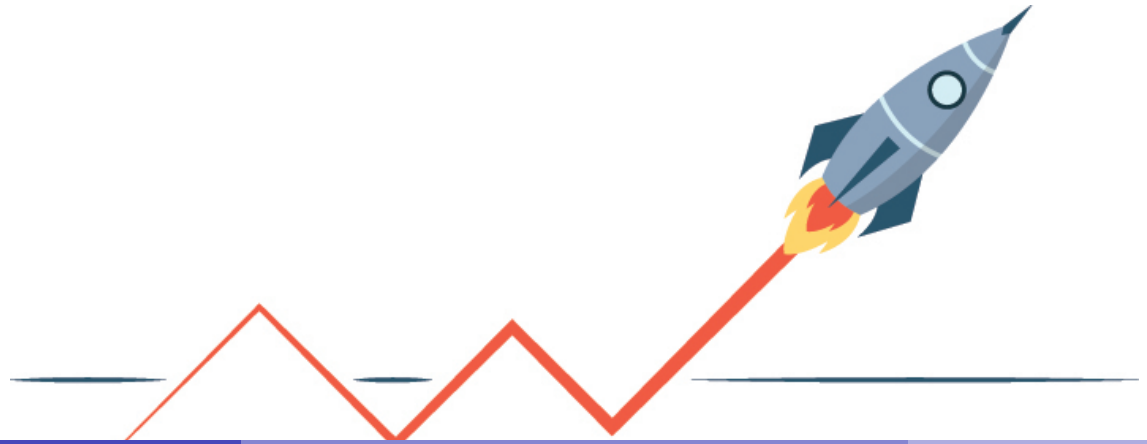
The one program running at all times on the computer is the kernel. Everything else is either a system program (ships with operating system) or an application program

Overview of How OS Works

- Computer startup and boot up the OS
  - CPU jumps to bootstrapping code, typically stored in Read-Only Memory (ROM) or Erasable Programmable ROM (EPROM), known by general term firmware
  - Bootstrapping code initializes all aspects of the system, from CPU registers to device controllers to memory contents
  - It loads OS kernel and starts execution, which can start to provide services to the system and users
  - Some services are provided outside the kernel, by system programs are also loaded into the memory at boot time to become system processes, or system daemons that run the entire time when the kernel is running
  - It also creates first process (e.g. init in Linux)
- Then, OS waits for events
  - OS is not involved if there is no event
  - Otherwise, OS is involved in handling events
    - ★ Interrupts from hardware
    - ★ System calls or exceptions caused by applications (cause interrupts)

What does Operating System do?

- OS is a resource allocator
  - Manages all resources (e.g., processors, memory, disk, etc.)
  - Decides between conflicting requests for efficient and fair resource use
  - Prevent errors and improper use of the computer
  - Topics to be discussed
    - ★ Processor management
    - ★ Memory management
    - ★ Device management
    - ★ File management
- OS is a facilitator
  - Provides facilities that everyone needs
  - Provides standard libraries, windowing systems
  - Make application programming easier, faster, less error-prone

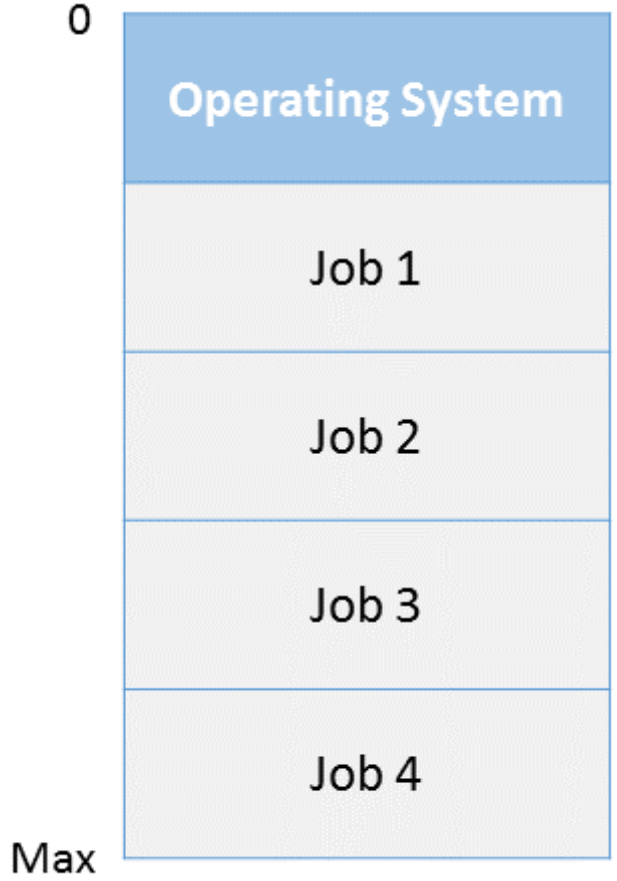


OS Implementation

- OS is a concurrent, real-time program
- It provides multiprogramming and multitasking / timesharing
  - Multiprogramming needed for efficiency
  - Multitasking/Timesharing is a logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing platform - user input vs. computer response

Multiprogramming

- A single program 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, while all jobs are initially kept on the disk in the job pool
- One job selected and run via job / CPU scheduling. When it has to wait (for I/O for example), CPU switches to another job



Memory layout for multiprogrammed System

Multitasking / Timesharing

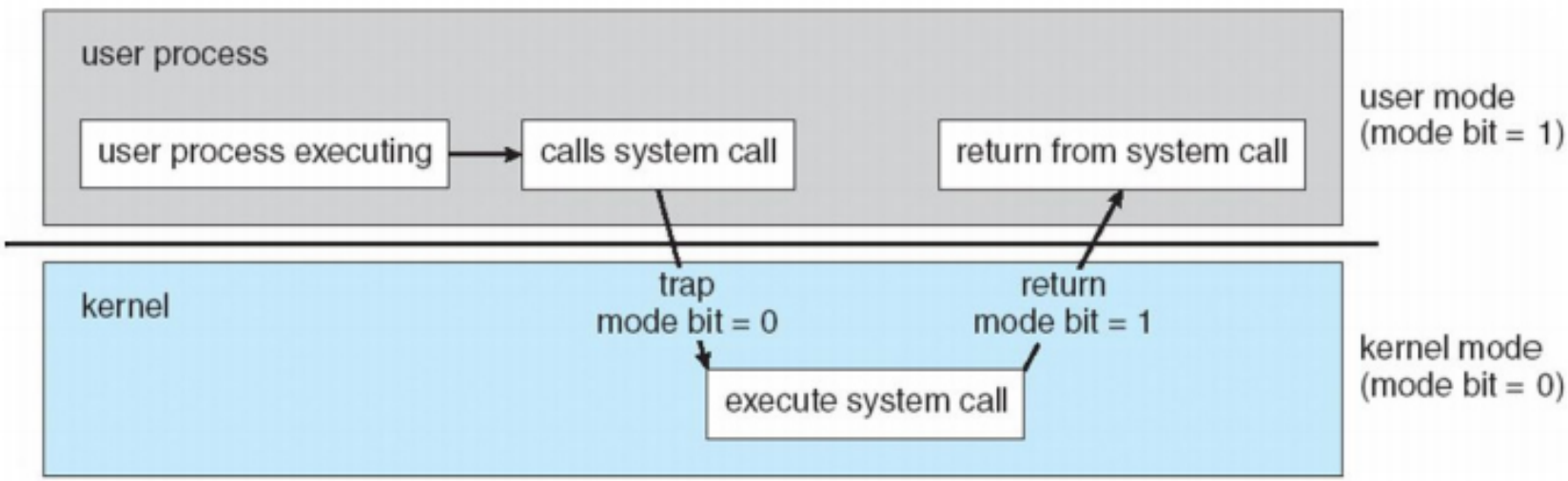
- Response time typically should be < 1 second
- Each user has at least one program executing in memory (Process)
- If several jobs are ready to run at the same time (CPU scheduling)
- A time-shared OS allows many users to share the computer simultaneously, giving each user the impression that the entire computer is dedicated to it

Dual-mode Operation

- There are two modes (dual-mode) of CPU execution, known as user mode and kernel mode (or supervisor mode)
  - User mode: Certain machine instructions cannot be executed, certain registers / memory locations cannot be accessed, and I/O cannot be accessed
  - Kernel mode: No restriction at all, i.e. can execute all machine instructions, can reference all memory locations
- 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 user code or kernel code
- Some instructions designated as privileged, which can only be executed in kernel mode (for example: I/O control, timer and interrupt management)
- System call changes the mode to kernel mode, return from the system call and resets the mode to user mode

Dual-mode Operation (Cont'd)

- Transitions from user mode to kernel mode:
  - System calls, interrupts, other exceptions



- The dual mode of operation provides a rudimentary mean for protecting the operating system from errant users and errant users from one another

Timer

- The primary duty of OS is to prevent user programs from getting stuck into an infinite loop or hogging resources and never returning control back to OS
- To accomplish this, OS uses a device known as Timer
- The main task of timer is to interrupt the CPU after a specific period of time
  - Set interrupt after specific period
  - OS decrements counter
  - When counter reaches 0, an interrupt occurs
- Timer is setup before scheduling process or terminate program that exceeds allocated time

Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary closed-source
  - Linux is the most common one
  - Microsoft Windows is a well-known closed-source OS
- Apple's Mac OS X and iOS, hybrid approach containing an open source kernel named Darwin yet with other closed-source components
- Benefits: programmers can contribute to the code, arguably more secure, bugs may be easily located or faster
- Counter to the copy protection and Digital Rights Management (DRM) movement, otherwise would not be effective if code are open-source
- Free Software Foundation (FSF): Richard Stallman started GNU project in 1983 to create a free and open-source UNIX compatible OS. Examples include
  - GNU/Linux
  - BSD UNIX (including core of Mac OS X)
  - Sun Solaris

Computing Environments

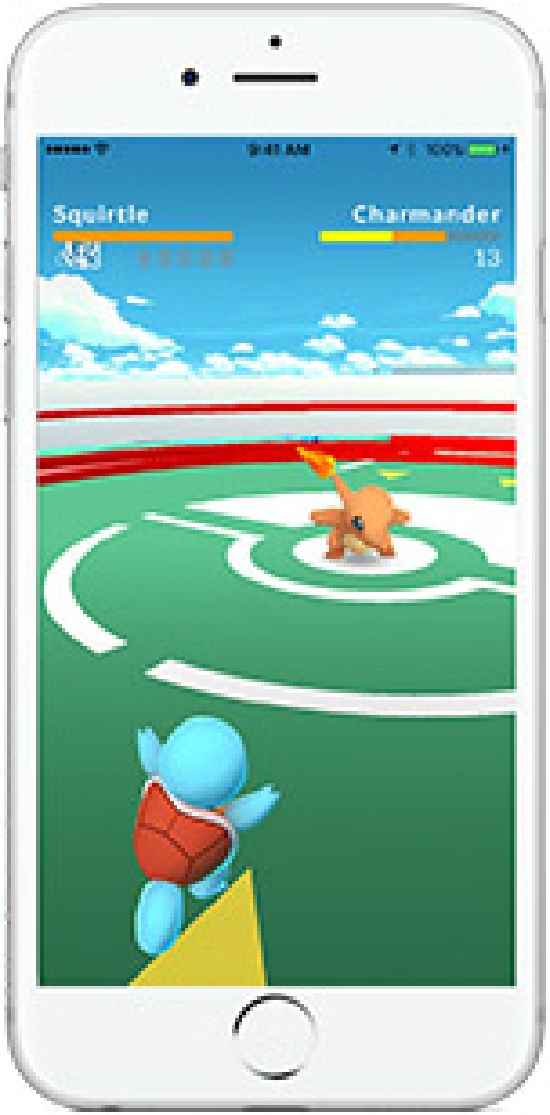
- Traditional computing environments
- Mobile computing environments
- Distributed computing environments
- Client-service computing environments
- Peer-to-peer (P2P) computing environments
- Virtualization computing environments
- Cloud computing environments
- Real-time embedded systems

Traditional Computing Environments

- Stand-alone general purpose machines, yet most systems interconnect with others (i.e. the Internet)
- Portals provide web access to internal systems
- Network computers (thin clients) are like Web terminals

- Mobile computers interconnect via wireless networks
- Networking becoming ubiquitous - even home systems use firewall to protect home computers from Internet breaches

Mobile Computing Environments



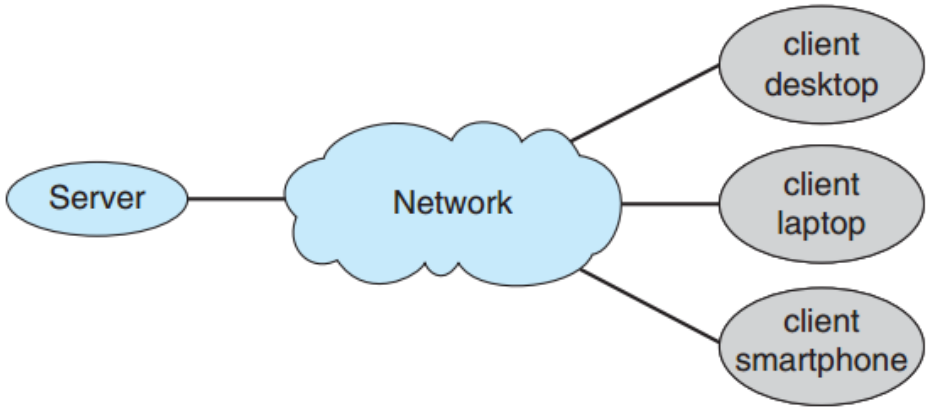
- Handheld smartphones, tablets, etc.
- Different between them and a "traditional" laptop?
  - Extra features: more OS features (GPS, accelerometers, and gyroscope)
  - Allow new types of apps like augmented reality (Pokemon Go)
  - Use IEEE 802.11 wireless, or cellular data networks for connectively
  - Leaders are Apple iOS and Google Android

Distributed Environments

- Collection of separate, possibly heterogeneous, system networked together
  - Network is a communication paths, TCP/IP most common
    - ★ Local Area Network (LAN)
    - ★ Metropolitan Area Network (MAN)
    - ★ Wide Area Network (WAN)
    - ★ Personal Area Network (PAN)
  - Network Operating System provides features between systems across network
    - ★ Communication scheme allows systems to exchange message
    - ★ Illusion of a single system

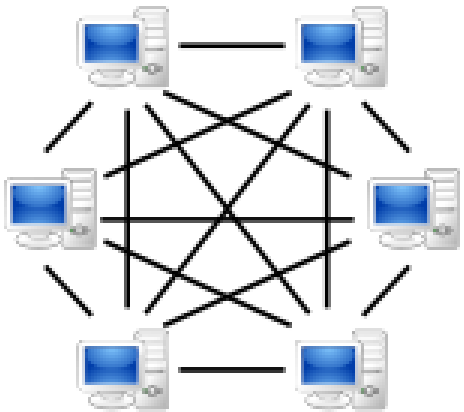
Client-Service Computing Environments

- Dumb terminals replaced by smart PCs
- Many systems now servers, responding to requests generated by clients
  - Server system provides an interface to client to request services (i.e., database)
  - File-server system provides interface for clients to store and retrieve files



Peer-to-Peer (P2P) Computing Environments

- Another model of distributed system
- P2P does not distinguish clients and servers
  - Instead all nodes are considered as peers
  - May each act as client, server or both
  - Node must join a P2P network
    - ★ Registers its service with central lookup service on network, or
    - ★ Broadcast request for service and respond to requests for service via discovery protocol
  - Examples include Napster and Gnutella, Voice over IP (VoIP) such as Skype



Virtualization Computing Environments

- Allows operating systems to run as applications within other OSes
  - Vast and growing industry
- Virtualization: OS natively compiled for CPU, running within another OS, also native to that CPU
  - This allows the user to install multiple operating systems to run applications written for operating systems other than the native host
  - An Apple laptop running Mac OS X on x86 CPU can run a Windows guest to allow execution of Windows applications
  - Examples: VMware, IBM PowerVM, Linux-VServer, etc.

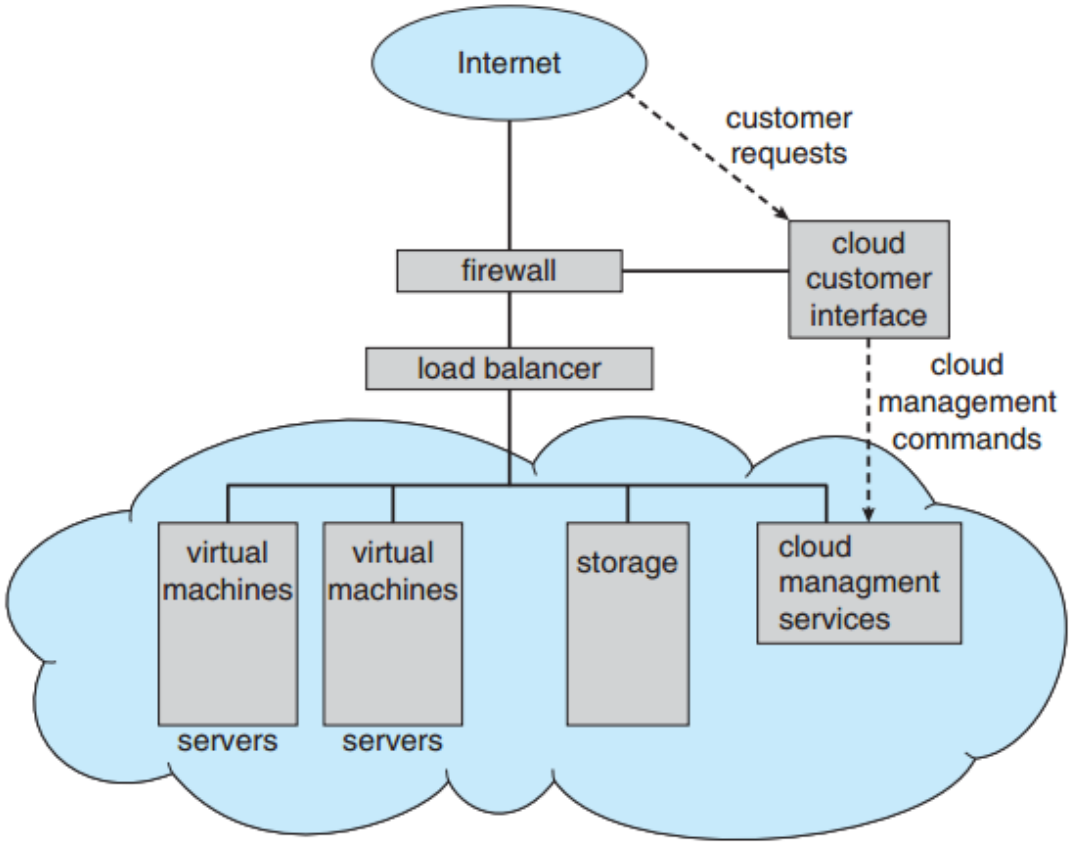
Cloud Computing Environments

- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization as based on virtualization
  - Amazon EC2 has thousands of servers, millions of VMs, petabytes (PBs) or storage available across the Internet, pay based on usage

- Many types
  - Public cloud: available via Internet to anyone willing to pay
  - Private cloud: run by a company for the company's own use
  - Hybrid cloud: includes both public and private cloud components
  - Software as a Service (SaaS): one or more applications available via the Internet (e.g., word processor)
  - Platform as a Service (PaaS): software stack ready for application use via the Internet (e.g., a database server)
  - Infrastructure as a Service (IaaS): servers or storage available over Internet (e.g., storage available for backup use)

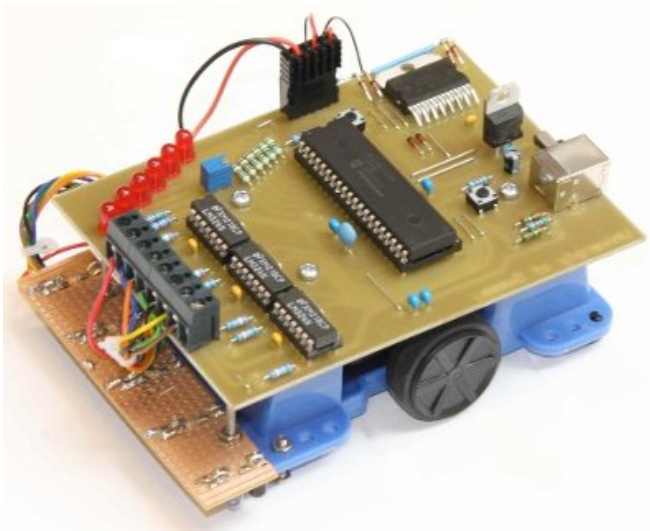
Cloud Computing Environments (Cont'd)

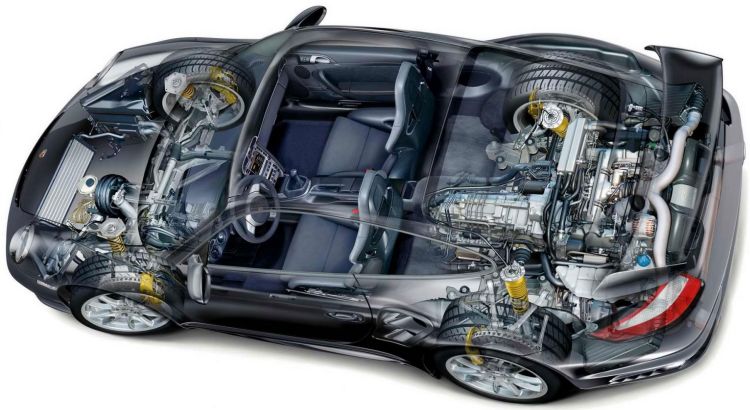
- Cloud computing environments composed of traditional OSes, plus virtual machines, plus cloud management tools
  - Internet connectivity requires security like firewalls
  - Load balancers spread traffic across multiple applications



Real-Time Embedded Systems

- Real-time embedded systems are most prevalent form of computers
  - Car engines, robots, DVDs, microwave ovens, everywhere
  - Vary considerable, special purpose, limited purpose OS, real-time OS
  - Usage expanding rapidly
- Many other special computing environments as well
  - Some have OSes, some perform tasks without an OS
- Embedded systems almost always run real-time operating system
- Real-time OS has well-defined fixed time constraints
  - Processing must be done within constraint
  - Correct operation only if constraints met





That's all!

Any questions?