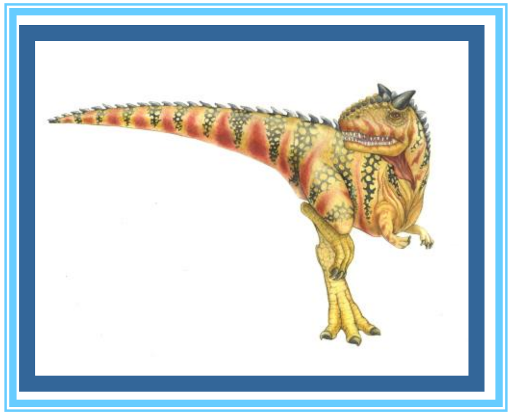
Lecture 1-2
Chapter 1: Introduction
O.S.

Dr. Zahraa Elsayed Mohamed



9/03/2014

Chapter 1: Introduction

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management
- Protection and Security
- Distributed Systems
- Special-Purpose Systems
- Computing Environments
- Open-Source Operating Systems

Objectives

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

Why study Operating Systems?

- To learn how computers work
- To learn how to manage complexity through appropriate abstractions
  - Infinite CPU, Infinite memory, files, semaphores, etc.
- To learn about system design
- performance vs. simplicity, HW vs. SW, etc.
- Because OSs are everywhere!

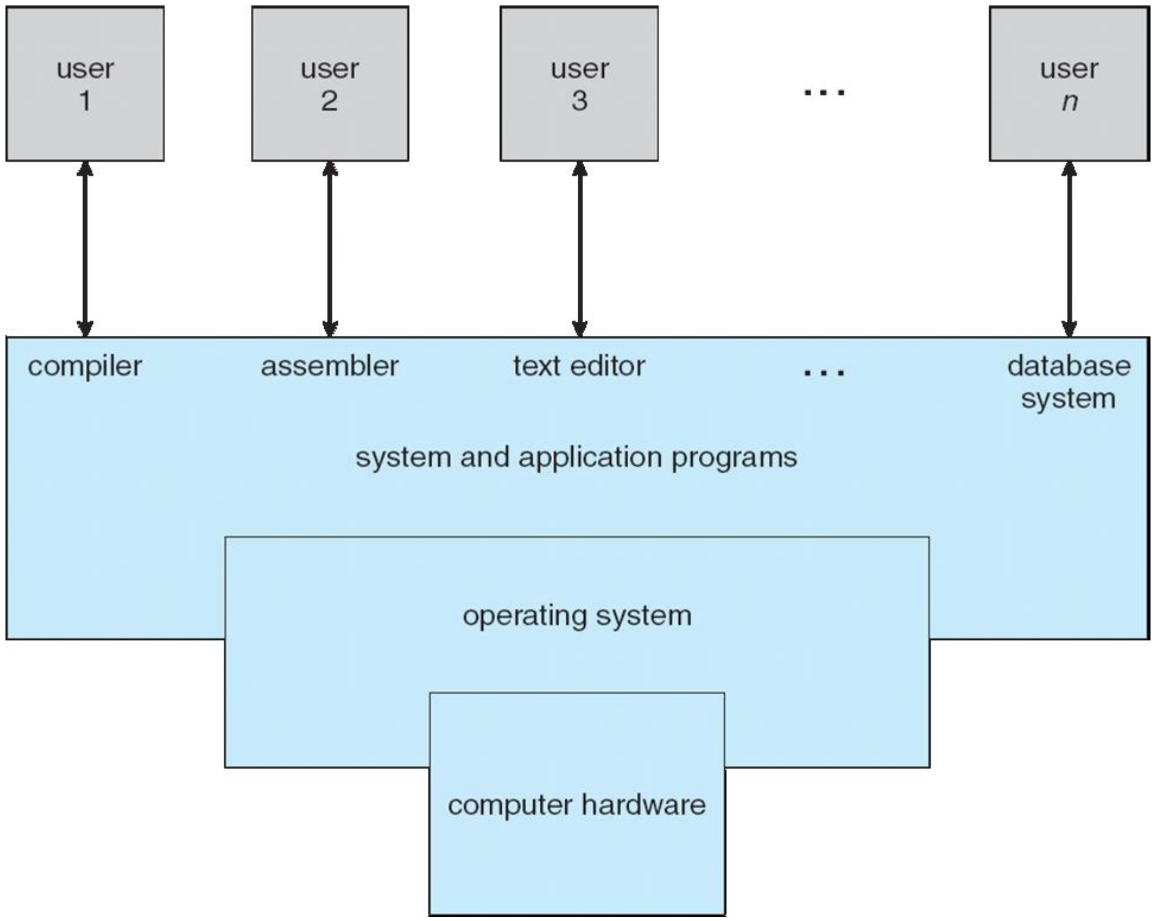
What is an Operating System?

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

Computer System Structure

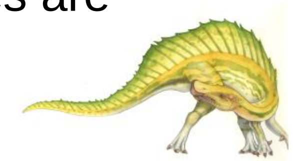
- Computer system can be divided into four components
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - Application programs – define the ways in which the system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



Operating System's Role

- To understand more fully the operating system's role, we next explore operating systems from two viewpoints: that of the user and that of the system.
- **User View**
  - **Single-User** (ease of use, some performance)
    - Most computer users sit in front of a PC, consisting of a monitor, keyboard, mouse, and system unit.
    - System is designed for one user to monopolize يحتكر its resources.
    - The goal is to maximize the work (or play) that the user is performing.
    - the operating system is designed mostly for ease of use, with some attention paid to performance and none paid to resource utilization—how various hardware and software resources are shared.



Operating System's Role

- To understand more fully the operating system's role, we next explore operating systems from two viewpoints: that of the user and that of the system.
- **User View**
  - **User sits at a terminal connected to a mainframe or minicomputer** (compromise between usability and resource utilization)
    - Other users are accessing the same computer through other terminals.
    - These users share resources and may exchange information.
    - The operating system in such cases is designed to maximize resource utilization— to assure that all available CPU time, memory, and I/O are used efficiently and that no individual user takes more than her fair share.

Operating System's Role

- To understand more fully the operating system's role, we next explore operating systems from two viewpoints: that of the user and that of the system.
- **User View**
  - **Users sit at workstations connected to networks of other workstations and servers** (compromise between usability and resource utilization)
    - These users have dedicated resources at their disposal وقد كرس المستخدمين الموارد الموجودة تحت تصرفها, but they also share resources such as networking and servers—file, compute, and print servers.
    - Operating system is designed to compromise ترضية between individual usability and resource utilization.

Operating System's Role

- To understand more fully the operating system's role, we next explore operating systems from two viewpoints: that of the user and that of the system.
- **System View**

  the operating system is the program most intimately involved ارتباطا وثيقاwith the hardware

  - **OS is a 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.
    - manages all resources
    - decides between conflicting requests for efficient and fair resource use

Operating System's Role

- To understand more fully the operating system's role, we next explore operating systems from two viewpoints: that of the user and that of the system.
- **System View**

  the operating system is the program most intimately involved ارتباطا وثيقاwith the hardware

  - **OS is a control program**
    - Controls execution of programs to prevent errors and improper use of the computer

Computer Startup

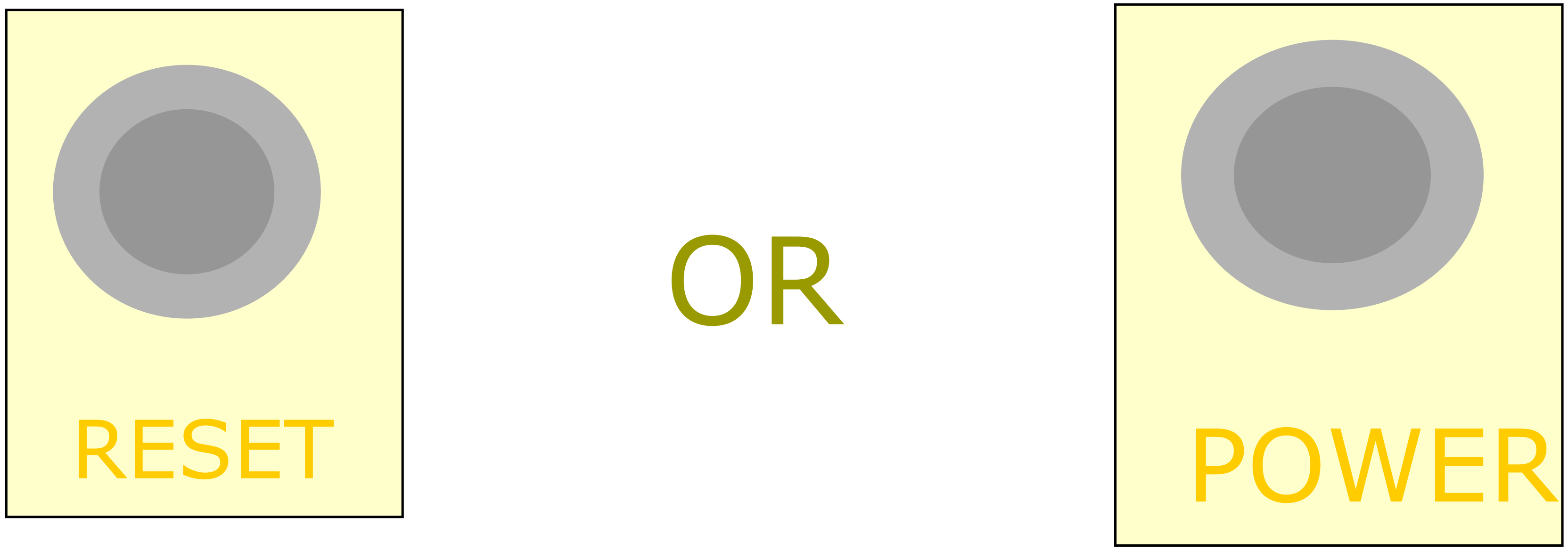
- Bootstrap program is loaded at power-up or reboot
  - Initial program to run
  - Typically stored in ROM or EEPROM, generally known as firmware
  - initializes all aspects of system: system diagnostics
  - loads operating system kernel into memory and starts execution
    - Sometimes, execute boot block code, the boot block loads OS and execute.
  - From now on, computer is under control of OS kernel.
- Operating system
  - executes the first process, such as “init”
  - waits for some event to occur

The boot process

A program cannot be loaded into memory unless a program has already been loaded into memory.

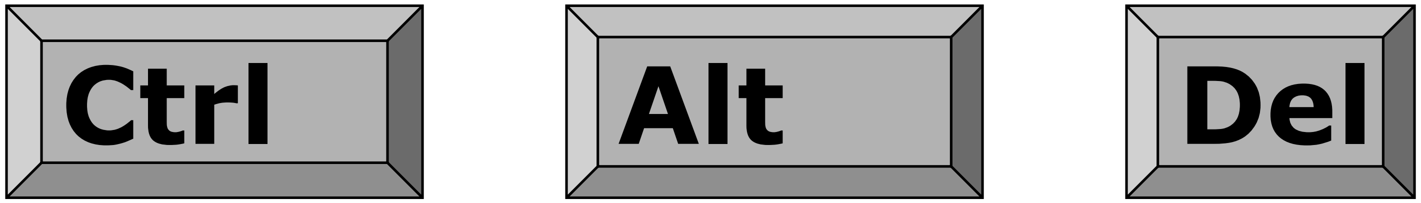
Cold Boot vs. Warm Boot.

Cold Boot



- The computer performs the Power On Self Test (POST) …
- And then loads the Disk Operating System.

Warm Boot.

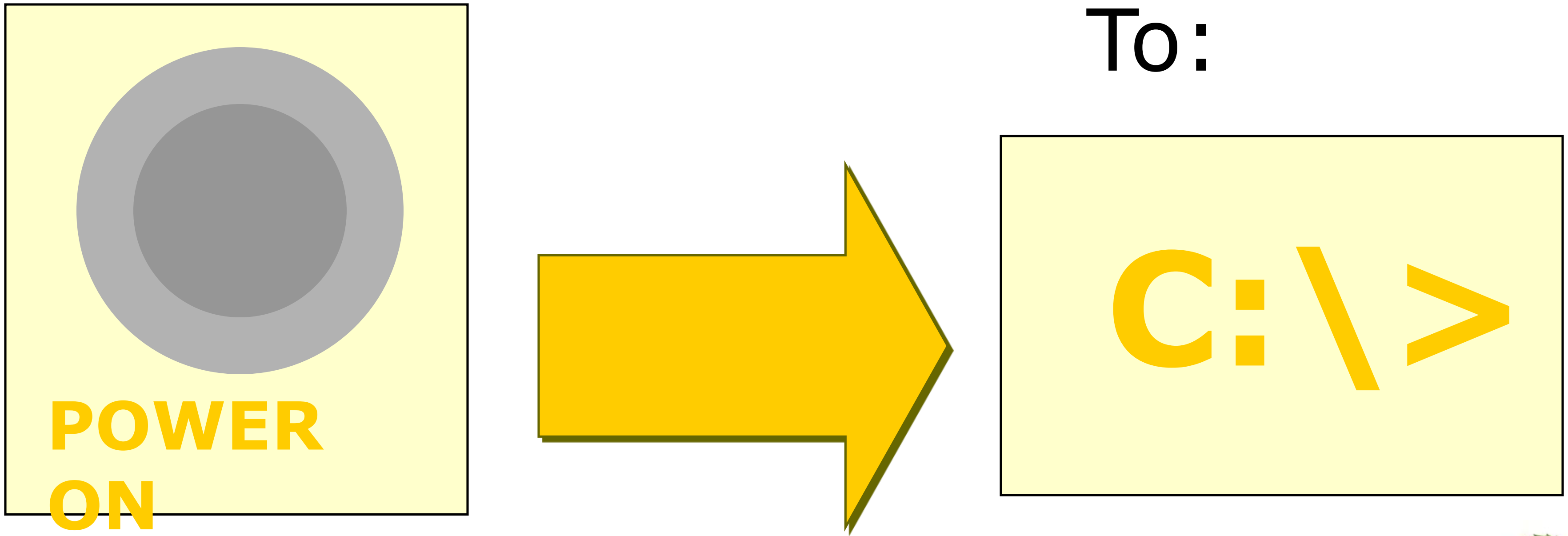


- The computer skips the Power On Self Test (POST) …
- And immediately reloads the Disk Operating System.

The boot process.

From:

To:



The Boot Process.

- Activate the Power On Self Test (POST)
- Load the Disk Operating System (DOS)

The Power On Self Test (POST)

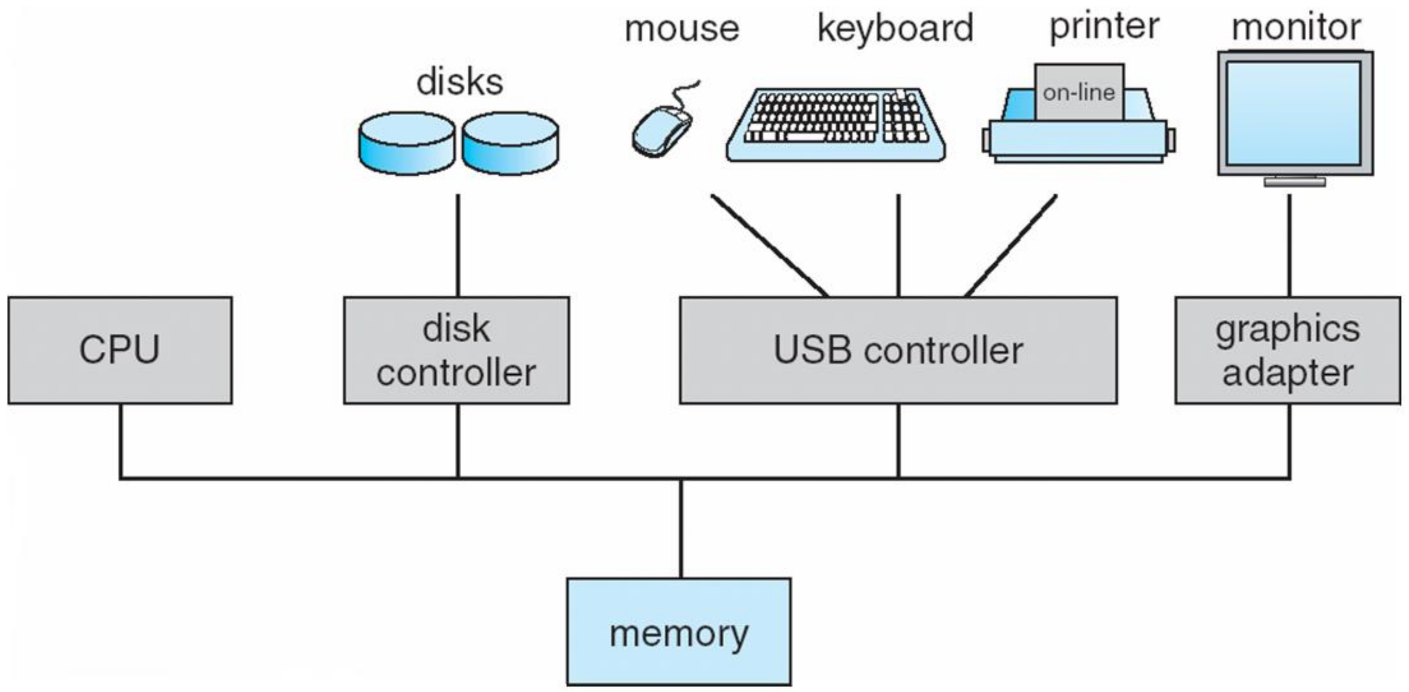
- Checks the CPU.
- Checks the system bus.
- Tests video memory.
- Tests system RAM.
- Tests the keyboard.
- Checks disk drives.
- Verifies configuration.
- Incorporates expansion BIOS.
- Configures Plug and Play Devices.

Operating System Definition

- OS is a resource allocator
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a control program
  - Controls execution of programs to prevent errors and improper use of the computer
- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the kernel. Everything else is either a system program (ships with the operating system) or an application program

Computer System Organization

- Computer-system operation
  - 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



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*

- Three techniques are possible for I/O operations:
  - Programmed I/O (Polling)
  - Interrupt-driven I/O
  - Direct memory access (DMA)

I/O Structure

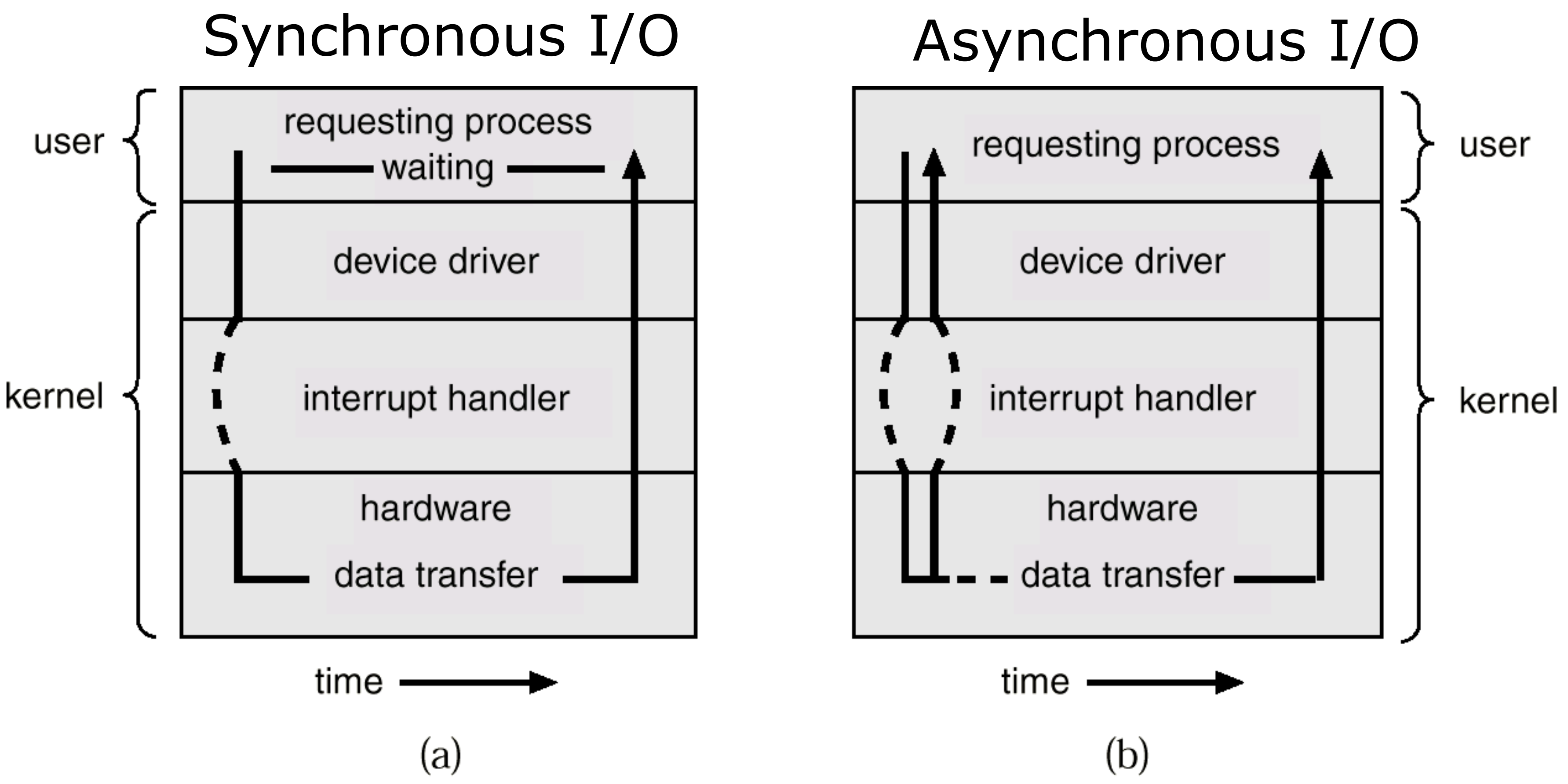
- What is I/O
  - Input and Output
  - Data transfer in between devices and memory/local buffer
  - Devices include keyboard, monitor, mouse, disk, etc.
- Device Controller
  - A hardware which is connected to the common bus
  - One or more devices are connected to a device controller
  - maintains some local buffer storage and a set of special-purpose registers
  - is responsible for moving the data between the peripheral devices and its local buffer storage
- Device Driver
  - A operating system has a device driver for each device controller
  - understands the device controller
  - provides a uniform interface for the device to the rest of the OS

- Three techniques are possible for I/O operations:
  - Programmed I/O (Polling)
  - Interrupt-driven I/O
  - Direct memory access (DMA)

I/O Structure

- To start an I/O operation
  - The device driver sets the appropriate registers within the device controller
  - The device controller examines the contents
  - The device controller starts the transfer of data from the device to its local buffer

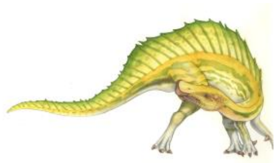
Two I/O Methods



- User Process use Synchronous I/O
- Operating System use Asynchronous I/O

Programmed I/O (Polling)

- I/O instruction is invoked, the processor must take some active role in determining when the I/O instruction is completed.
- CPU periodically checks the status of the I/O module until it finds that the operation is complete.
- CPU is responsible for extracting data from main memory for output and storing data in main memory for input.
- I/O software is written in such a way that the processor executes instructions that give it direct control of the I/O operation, including sensing device status, sending a read or write command, and transferring the data.

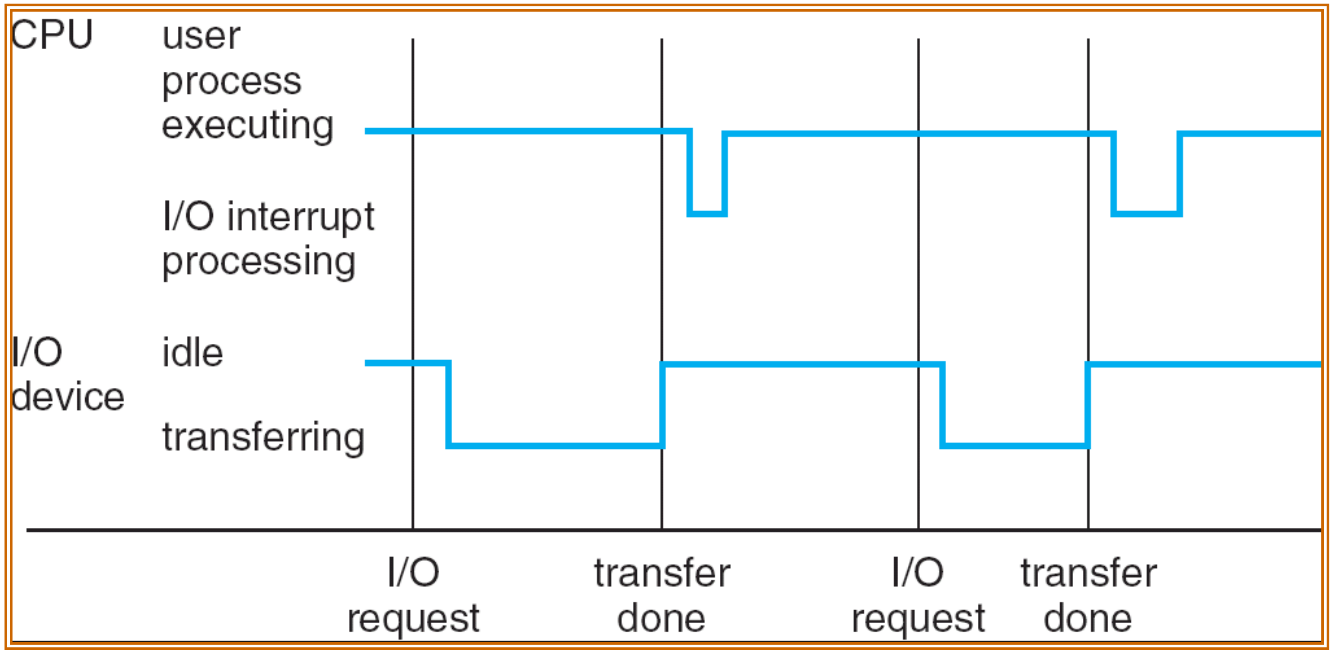


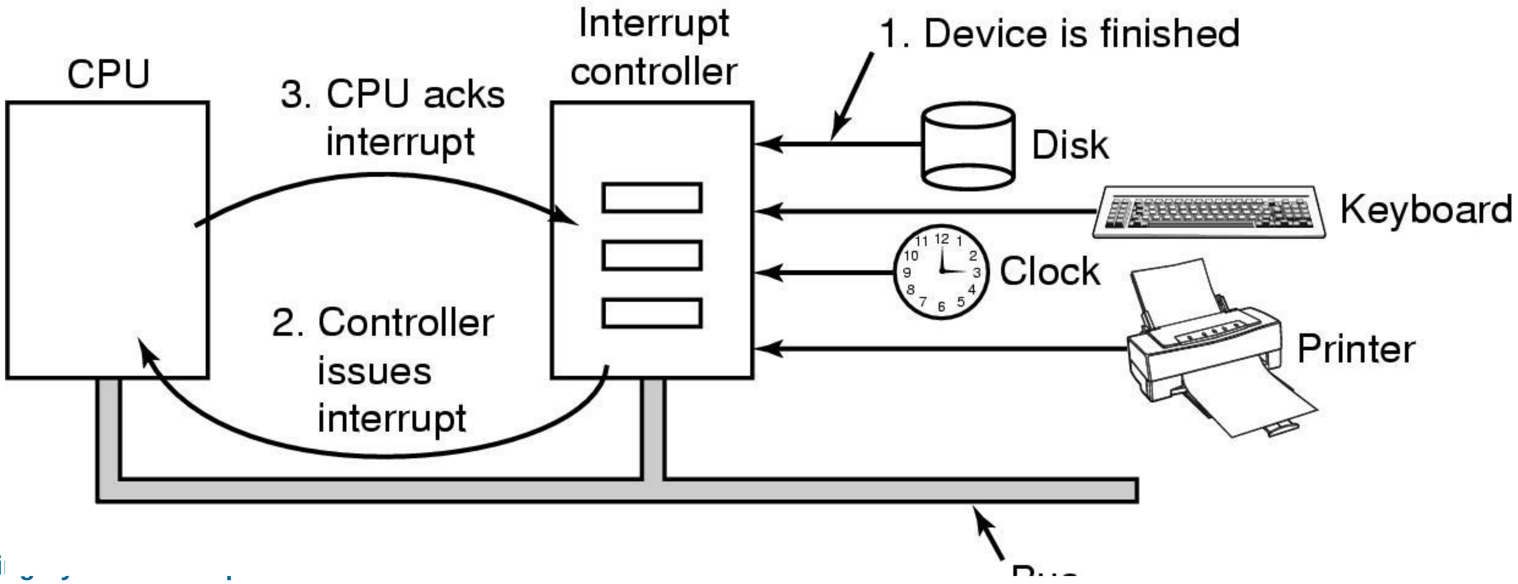
Interrupt

- The occurrence of an event is usually signaled by an interrupt from either the hardware and the software.
  - Hardware may trigger an interrupt at any time by sending signal to the CPU
  - Software may trigger an interrupt by executing a special operation called a system call

- When the CPU is interrupted
  - stops what it is doing
  - immediately transfers execution to a fixed location
  - the fixed location usually contains the starting address where the service routine for the interrupt is located
  - executes the interrupt service routine
  - resumes the interrupted computation on completion.

Interrupt-driven I/O Timeline





Interrupt Service Routine

- Each computer has its own interrupt mechanism, but several functions are common
  - <u>Generic routine</u> and <u>interrupt-specific handler</u>

- invoke a generic routine to examine the interrupt information
- call the interrupt-specific handler.

- Interrupts must be handled quickly.
  - Only predefined number of interrupts are possible
  - A table of pointers to interrupt handler is used
    - Interrupt vector table
  - The table of pointers is sorted in low memory location

Common Functions of Interrupt

Each computer design has its own interrupt mechanism

- Interrupt transfers control to the interrupt service routine generally, through the *interrupt vector*, which contains the addresses of all the service routines.
- Interrupt architecture must save the address of the interrupted instruction.
  - by storing registers and the program counter.
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*.
- An operating system is *interrupt driven*.

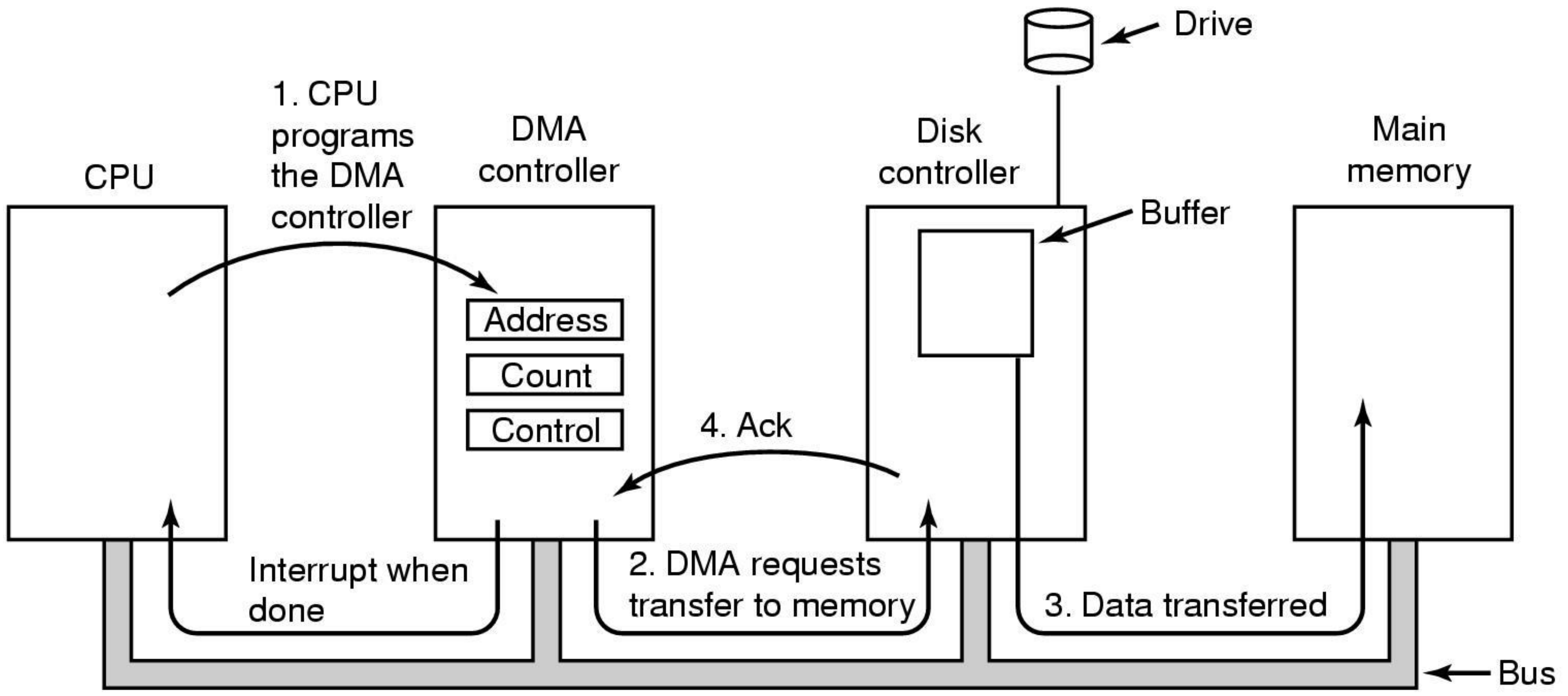
Direct Memory Access (DMA)

- 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 interrupt is generated per block, rather than the one interrupt per byte.
  - improves the performance of DMA.

Direct Memory Access (DMA)

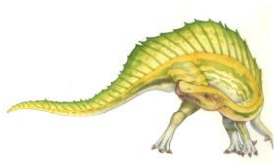


- Operation of a DMA transfer

Storage Structure

- Main memory – only large storage media that the CPU can access directly. (outside CPU)
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity.
  - Magnetic disk are used for the secondary storage

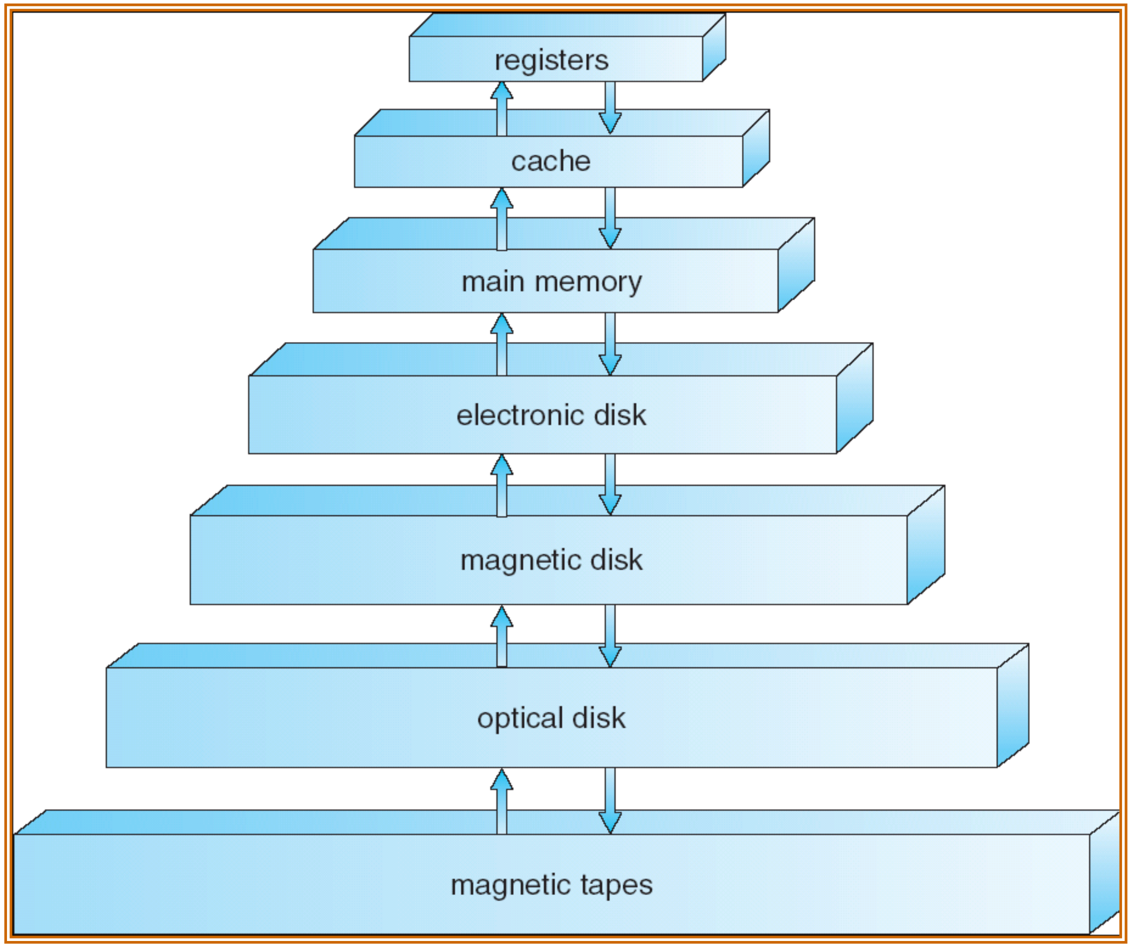
- Magnetic disks – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into *tracks*, which are subdivided into *sectors*.
  - The *disk controller* determines the logical interaction between the device and the computer.

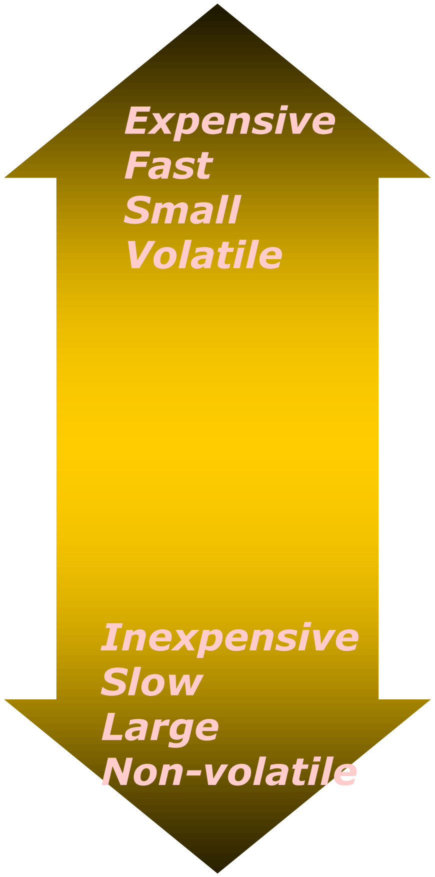


Storage Hierarchy

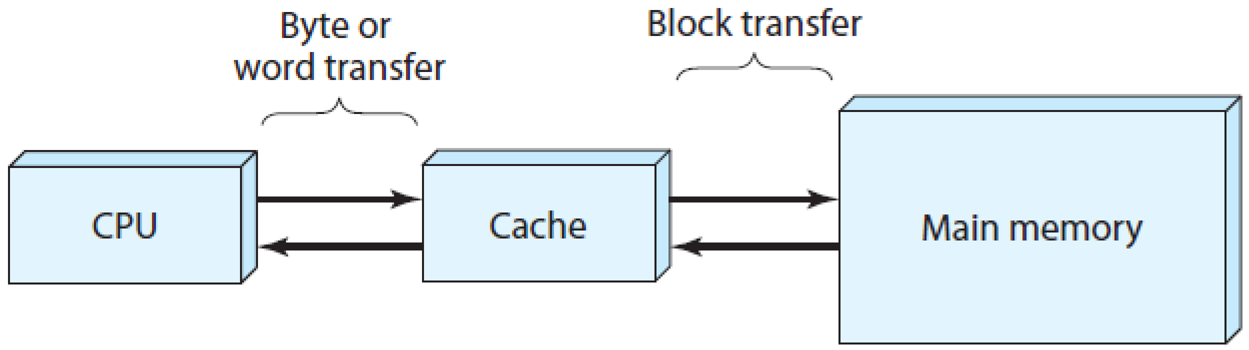
- Storage systems organized in hierarchy.
  - Speed
  - Cost
  - Volatility

Storage-Device Hierarchy





Caching



- Caching – copying information into faster storage system; main memory can be viewed as a last cache for secondary storage
- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there

Computer System Architecture

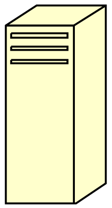
- Single-processor systems
  - One main CPU, most common
  - Variety of single-processor systems: from PDAs (personal digital assistant (PDA) “For reading books or anther things”) to mainframes
- Multiprocessor Systems
  - Two or more CPUs in close communication,
  - shares the computer bus, clock, memory, and peripheral devices
  - three main advantages
    - Increased throughput
      - We can expect to get more work done in less time
    - Economy of scale
      - can cost less than equivalent multiple single-processor systems
    - Increased reliability
      - The failure of one processor will not halt the system, only slow it down.

Computer System Architecture

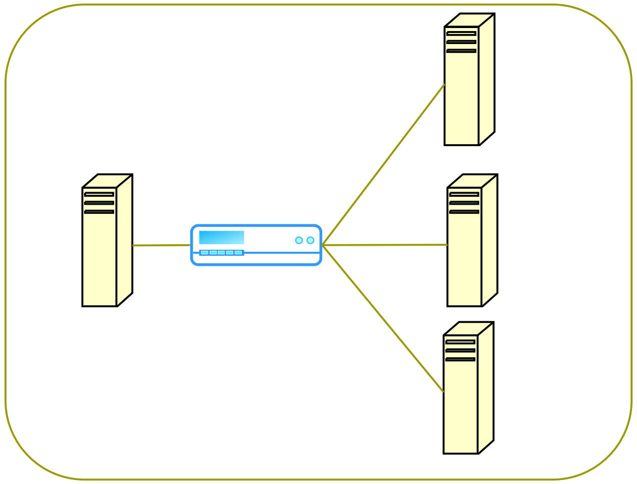
- Multiprocessor Systems
  - One disadvantage
    - Increased complexity
      - They are also more complex in both hardware and software than uni-processor systems.
  - Two types of Multiprocessor systems
    - Asymmetric multiprocessing
      - Master-slave relationship of processor
      - Master processor schedules and allocate work to the slave processors.
      - I/O operations usually are in charge of master processor.
    - Symmetric multiprocessing (SMP)
      - No master-slave relationship between processors
      - All processors are peer to each other
      - Each processor performs it’s own task without interference from others.

Computer System Architecture

- **Clustered System**
  - **gather together multiple systems to accomplish computational work.**
  - **The multiple systems are interconnected to each other via LAN.**
  - **Clustering is usually used to provide high-availability**
  - **provide high performance computing environments**
  - **Examples:**
    - **Web server clusters.**



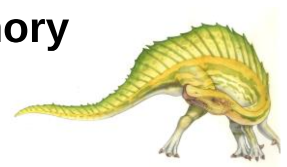
Single web server



Web server cluster

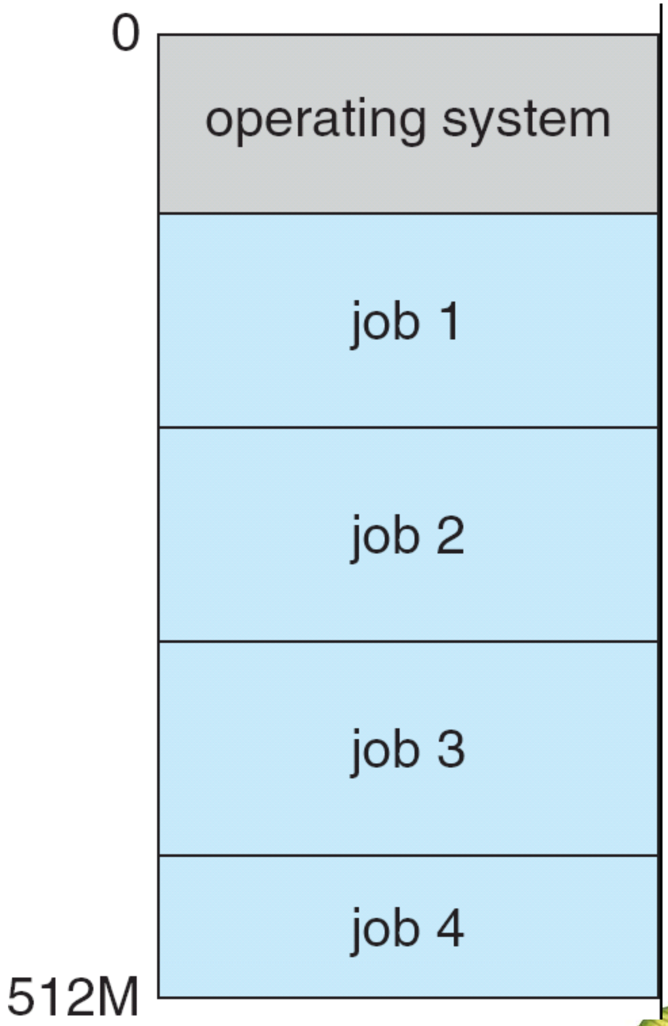
Operating System Structure

- Multiprogramming needed for efficiency
  - 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
  - One job selected and run via job scheduling
  - When it has to wait (for I/O for example), OS switches to another job
- Timesharing (multitasking) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, <u>creating interactive computing</u>
  - Response time should be < 1 second
  - Each user has at least one program executing in memory ⇨process
  - All processes residing on disk awaiting allocation of main memory ⇨ Job pool
  - If several jobs are ready to be brought into memory, and if there is not enough room for all of them ⇨ job scheduling
  - 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



Memory Layout for Multiprogrammed System

- OS keeps several jobs in memory simultaneously.
- Set of jobs in the memory can be a subset of the jobs kept in the job pool.
- OS picks and begins to execute one of the jobs in memory
- In I/O operation of a job, OS switches to another job to execute.
- When the job needs to wait, the CPU is switched to another job, and so on.

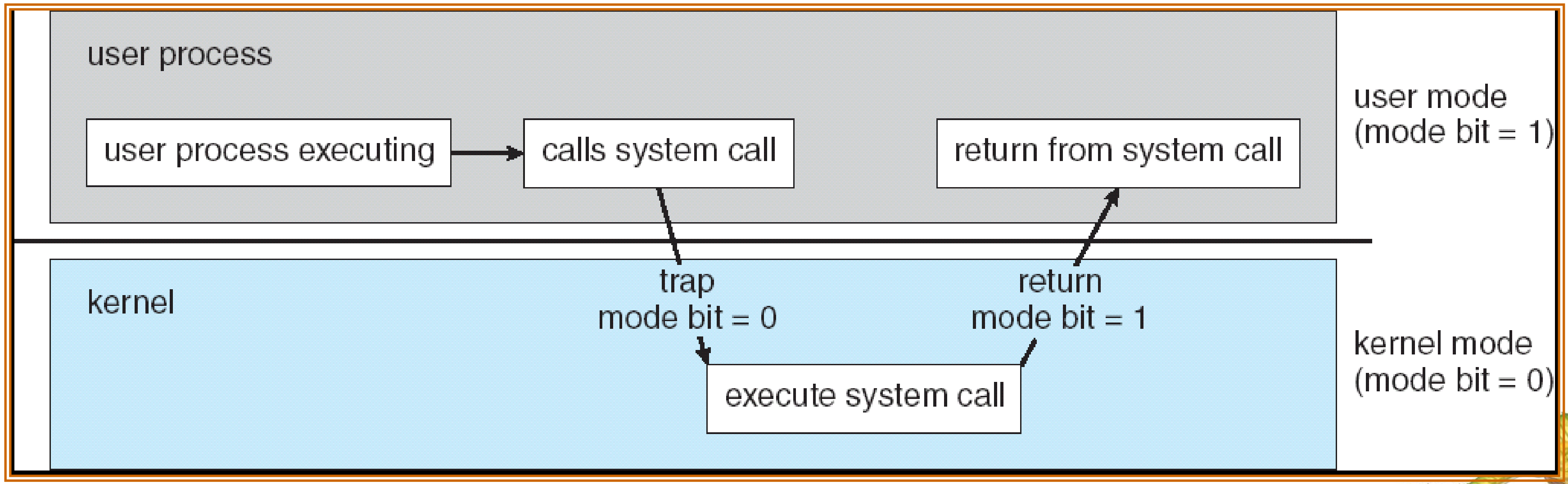


Operating-System Operations

- Modern OS are Interrupt driven by hardware.
- Software error or request creates exception or trap
  - Division by zero, request for operating system service
  - Other process problems include infinite loop, processes modifying each other or the operating system
- Dual-mode operation allows OS to protect itself and other system components
  - User mode
  - Kernel mode (called supervisor mode, system mode, privileged 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

Transition from User to Kernel Mode

- At system boot time, the hardware starts in kernel mode
- The OS is then loaded and starts user applications in user mode
- Whenever a trap or interrupt occurs, the hardware switches *from user mode to kernel mode* (mode bit = 0)
- Whenever the OS gains control of the computer it is kernel mode.
- The system always *switches to user mode* before *passing control to a user program*. (mode bit =1)



Timer

- The OS maintains control over the CPU.
- The OS must prevent a user program from
  - getting stuck in an infinite loop
  - not calling system services
  - never returning control to the OS

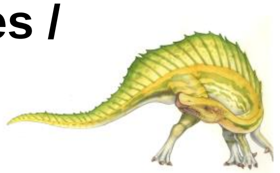
- Timer to prevent ***infinite loop*** / ***process hogging resources***
  - Set interrupt after specific period (fixed timer)
  - Operating system decrements counter (variable timer)
    - When counter zero generate an interrupt

- Before turning over control to the user, the OS ensures that the timer is set to interrupt.
  - If the timer interrupts, control transfers automatically to the OS.

Process Management

- A process is a program in execution. It is a unit of work within the system.
  - Program is a *passive entity*, process is an *active entity*.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires release of any reusable resources
- Single-threaded process has one program counter specifying location of next instruction to execute
  - Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread

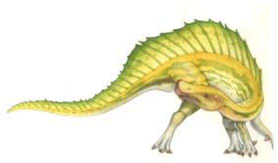
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
  - Concurrency by multiplexing the CPUs among the processes / threads



Process Management Activities

The operating system is responsible for the following activities in connection with process management:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling



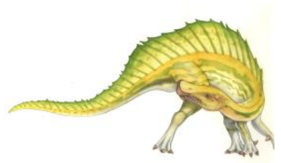
Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
  - Optimizing CPU utilization and computer response to users

- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Storage Management

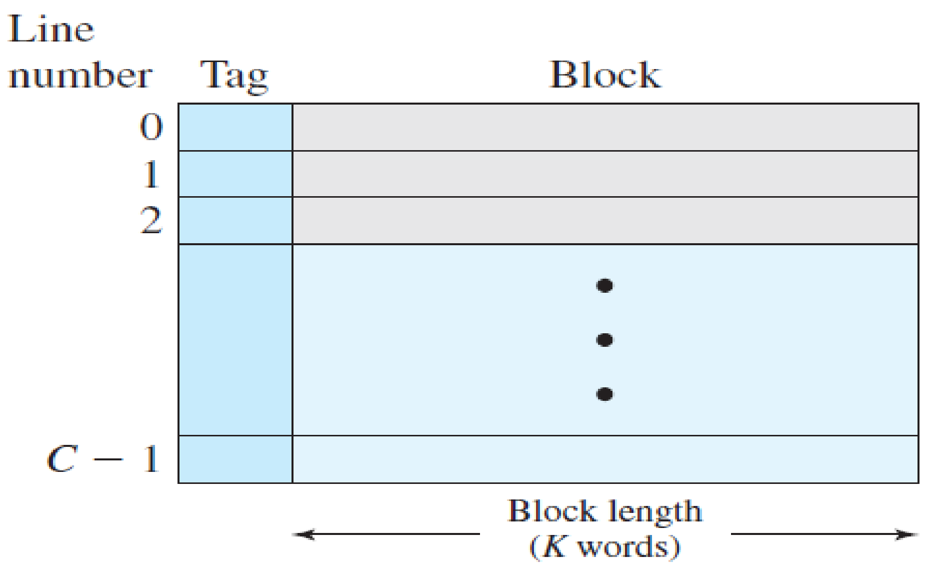
- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - file
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and dirs (dir, type, copy, rename)
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media



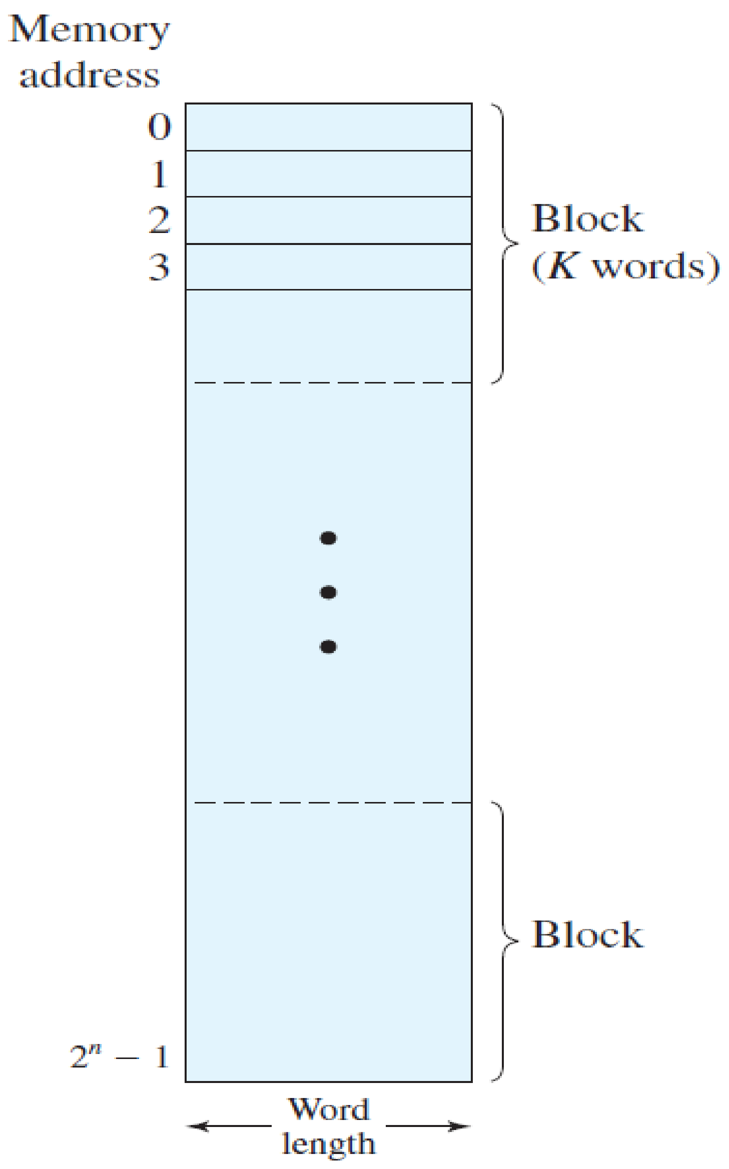
Mass-Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed
  - Varies between WORM (write-once, read-many-times) and RW (read-write)

Caching



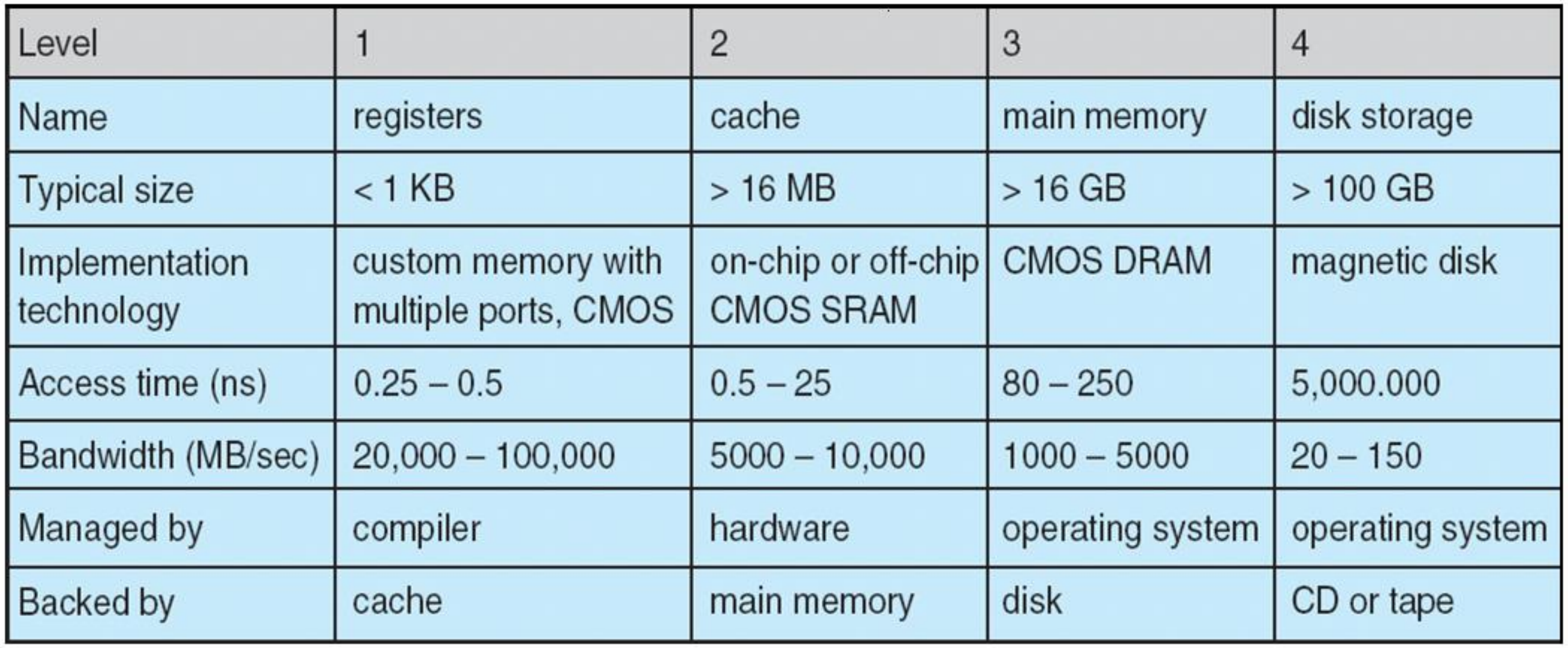
(a) Cache



(b) Main memory

Performance of Various Levels of Storage

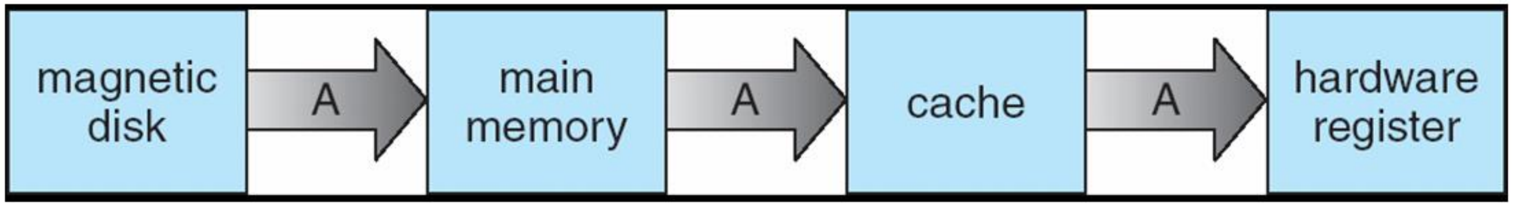
- Movement between levels of storage hierarchy can be explicit or implicit



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

Migration of Integer A from Disk to Register

- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy



- Multiprocessor environment must provide cache coherency in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex
  - Several copies of a datum can exist
  - Various solutions covered in Chapter 17

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
  - General device-driver interface
  - Drivers for specific hardware devices

Protection and Security

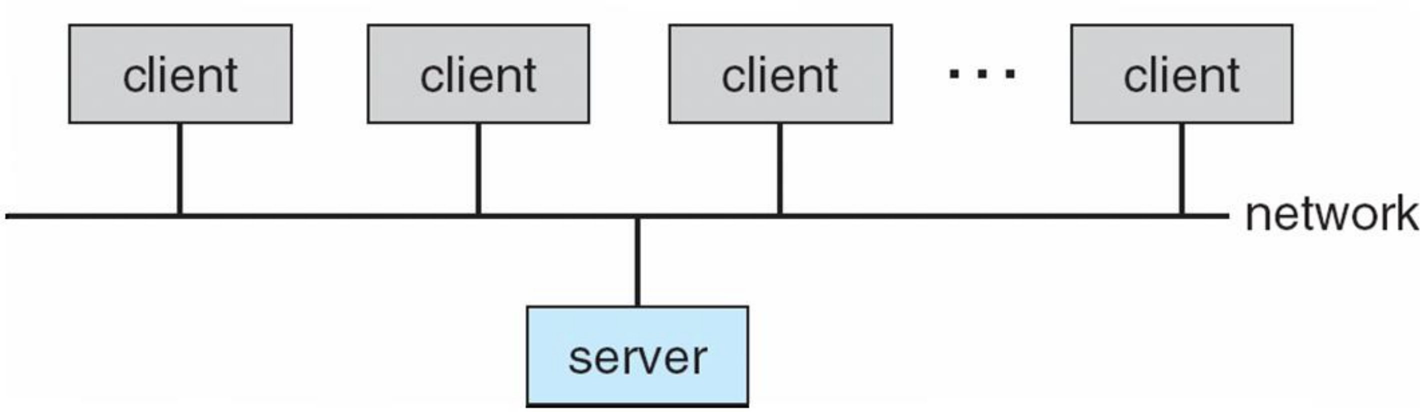
- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
  - User identities (**user IDs**, security IDs) include name and associated number, one per user
  - User ID then associated with all files, processes of that user to determine access control
  - Group identifier (**group ID**) allows set of users to be defined and controls managed, then also associated with each process, file
  - **Privilege escalation** allows user to change to effective ID with more rights

Computing Environments

- Traditional computer
  - Blurring over time
  - Office environment
    - PCs connected to a network, terminals attached to mainframe or minicomputers providing batch and timesharing
    - Now portals allowing networked and remote systems access to same resources
  - Home networks
    - Used to be single system, then modems
    - Now firewalled, networked

Computing Environments (Cont)

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



Peer-to-Peer Computing

- Another model of distributed system
- P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - May each act as client, server or both
  - Node must join 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*

Web-Based Computing

- Web has become ubiquitous
- PCs most prevalent devices
- More devices becoming networked to allow web access
- New category of devices to manage web traffic among similar servers: **load balancers**
- Use of operating systems like Windows 95, client-side, have evolved into Linux and Windows XP, which can be clients and servers

Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary closed-source
- Counter to the copy protection and Digital Rights Management (DRM) movement
- Started by Free Software Foundation (FSF), which has “copyleft” GNU Public License (GPL)
- Examples include GNU/Linux, BSD UNIX (including core of Mac OS X), and Sun Solaris

Storage Hierarchy

- Storage systems organized in hierarchy.
  - Speed
  - Cost
  - Volatility

Computer System Operations

- 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 a data transfer from the device to local buffer of controller.
- Device controller informs CPU that it has finished its operation by causing an *interrupt*.

Summary

- Three main goals of OS
  - Process Management, Environment Management, Resource Management
- Four components of Computer System
  - Hardware, OS, program, user
- OS is interrupt driven.
  - Interrupt is a request signal from hardware and software to OS.
  - Interrupt transfers control to the interrupt service routine
- I/O is data transfer in between devices and local buffer
  - device controller and device driver are involved.
  - Synchronous I/O vs. Asynchronous I/O
- Computer system has a hierarchical storage structure.
  - Register, Cache, Memory, Secondary storage, Tertiary storage
- OS provides multitasking environment which creates interactive computing.
- Dual-mode operation allows OS to protect itself and other system components
  - User mode, Kernel mode
- OS provides protection and security mechanism.

End of Chapter 1