

Chapter 1: Introduction

Grand tour of the major operating systems components



Thanks to the author of the textbook [**SGG**] for providing the base slides. I made several changes/additions. These slides may incorporate materials kindly provided by Prof. Dakai Zhu. So I would like to thank him, too.

Turgay Korkmaz


Chapter 1: Introduction

- What Operating Systems Do **
- Computer-System Organization **
- Computer-System Architecture, Structure, Operations **
- Process Management *****
- Memory Management *****
- Storage Management *
- Protection and Security ***
- Distributed Systems (more later)
- 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

What Operating Systems do?

	\$300
+	
OS (Windows)	\$200
+	
Applications	\$\$\$\$



Do I have to?!

Yes!
Otherwise, a set of silicon circuits
doing nothing good
for you !

Operating system (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 is an Operating System (OS)?

User View

User interface, ease of use vs. performance (*windows vs. command line; terminal vs. minicomputer; networked workstations; handheld; embedded*)

System View

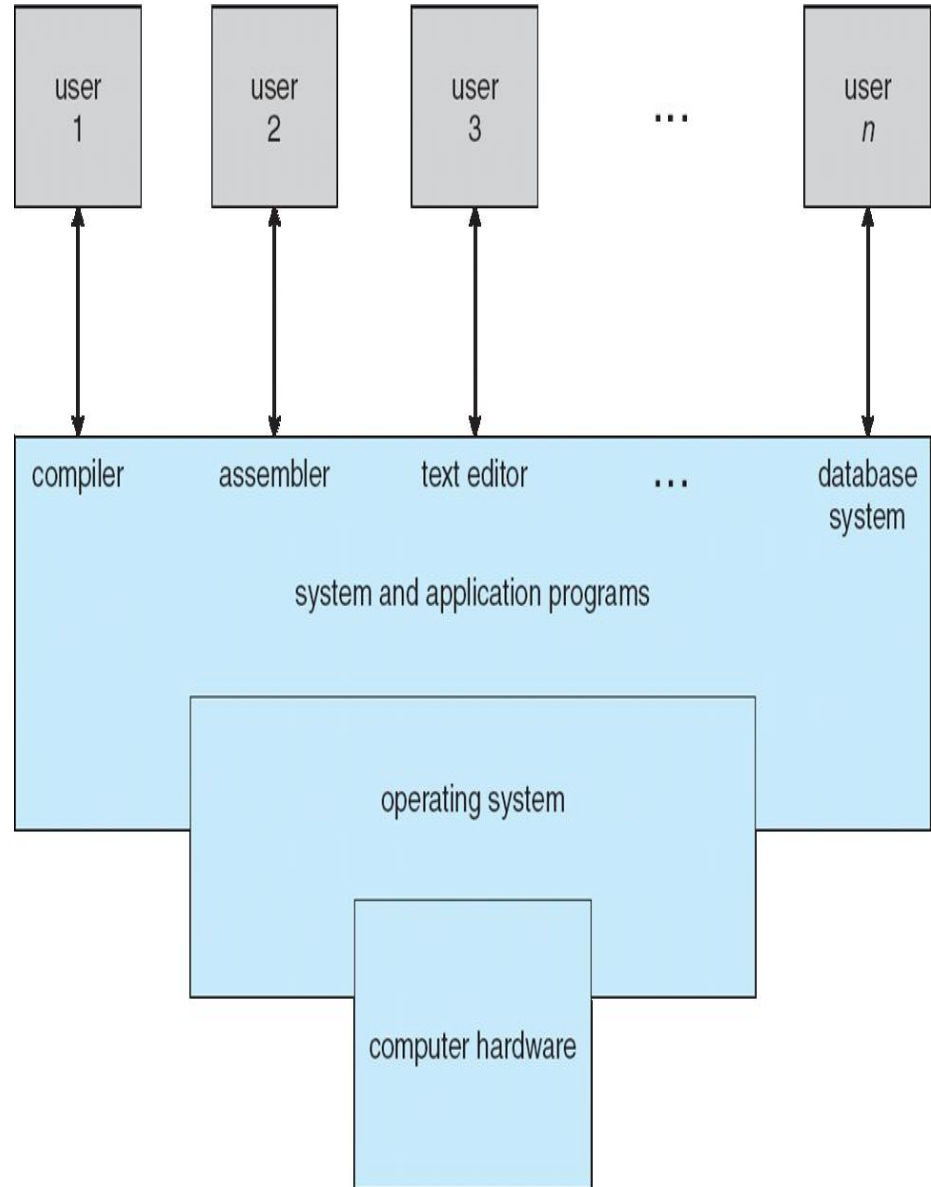
OS is a **resource allocator**

Manages all resources (*CPU, memory, file-storage, I/O devices etc.*)

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



Operating System Definition(s)

Based on the discussion so far, can you define Operating System?

- Here is a good approximation

“Everything a vendor ships when you order an operating system”

- “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
- So we may say: OS is a program (or a set of programs) that
 - **manages** computer hardware and resources,
 - **provides** a basis and user friendly **interface** for applications and users
- No universally accepted or completely adequate definition
- Instead of a definition, we should focus on the tasks that are necessary to accomplish the OS goals we mentioned
 - Execute user programs and make solving user problems easier (user/programmer view)
 - Make the computer system convenient to use... (user view)
 - Use the computer hardware in an efficient manner... (system view)

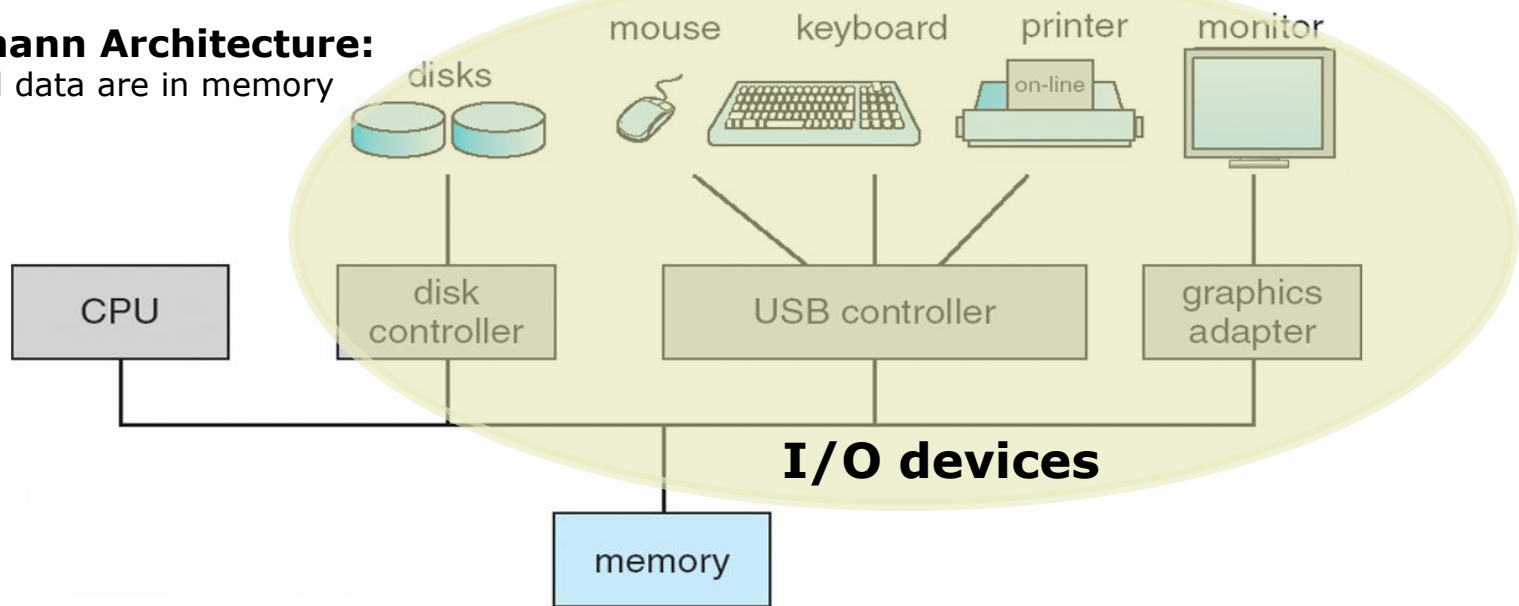
HOW?

- The amazing aspect of Operating Systems is how **varied** they are in performing their tasks
 - Different needs, capabilities, etc
 - Ease of use vs. performance vs. cost

Computer System Organization

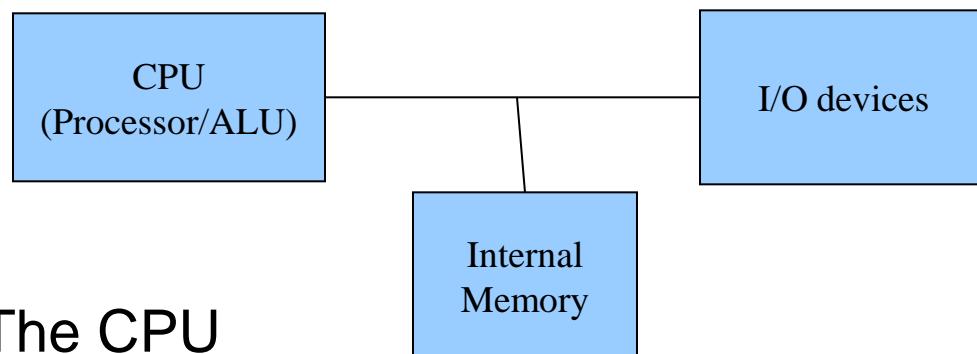
von Neumann Architecture:

Program and data are in memory



One or more CPUs, main memory, and many I/O devices connected through a common bus to perform various tasks


Computer-System Operation



Computer Startup:
bootstrap program is loaded at power-up or reboot: Typically stored in ROM or EPROM, generally known as **firmware**. It initializes all aspects of system and loads operating system **kernel** and starts execution

■ The CPU

- load instructions from main memory,
- load/store data from/to memory system
- controls I/O devices to perform certain tasks

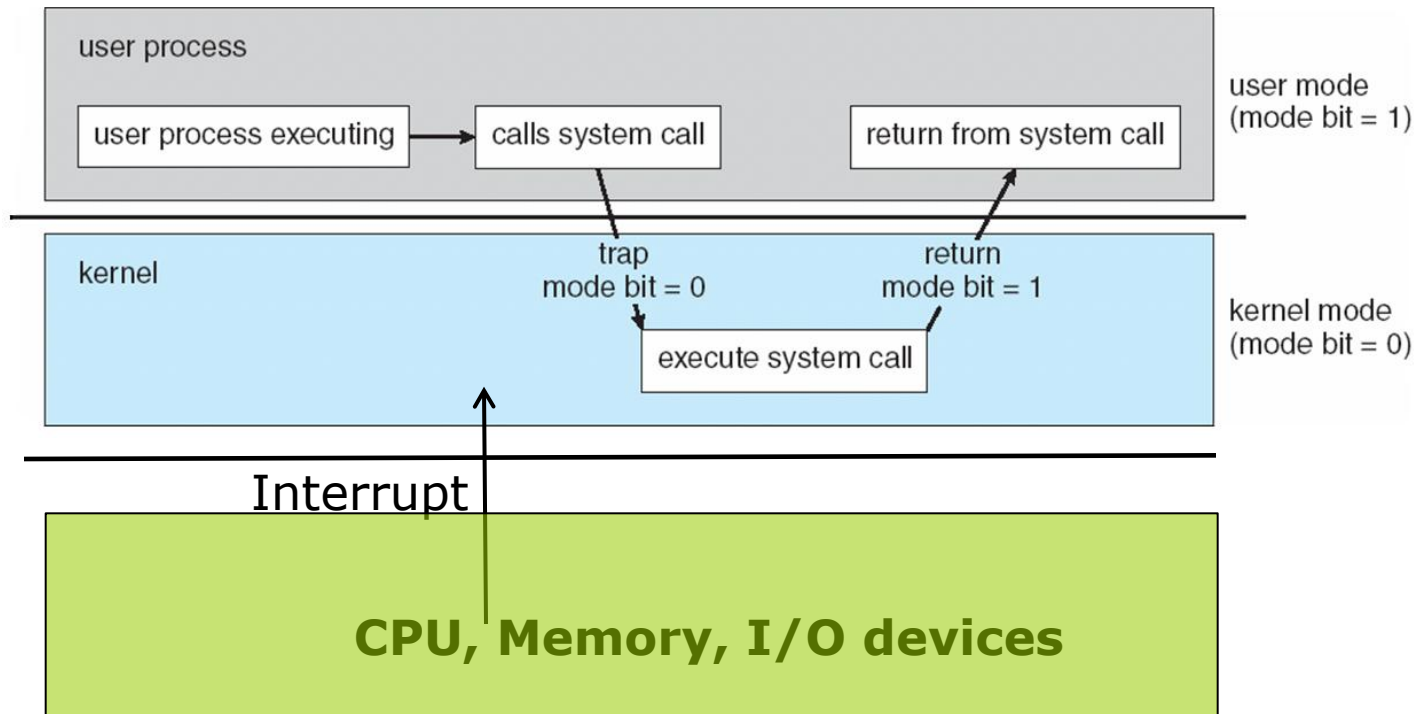
 User process or kernel process

■ I/O devices and the CPU can execute concurrently

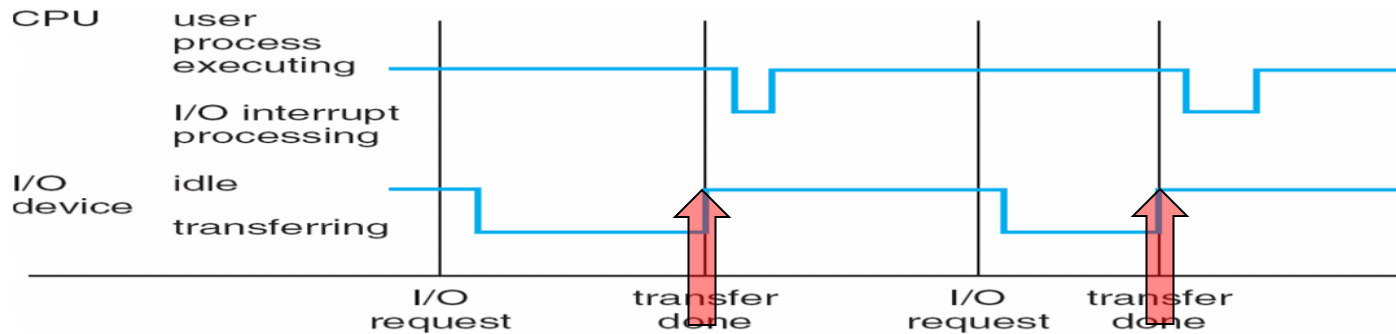
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- CPU moves data from/to main memory to/from local buffers

■ Device controller informs CPU that it has finished its operation by causing an ***interrupt***

Computer-System Operation (cont'd)



Interrupt Handling



1. The interrupt is issued
2. Processor finishes execution of current instruction
3. Processor signals acknowledgement of interrupt
4. Processor pushes PSW and PC onto control stack
5. Processor loads new PC value through the interrupt vector
6. ISR saves remainder of the process state information
7. ISR executes
8. ISR restores process state information
9. Old PSW and PC values are restored from the control stack

Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine (ISR) generally, through the **interrupt vector**, which contains the addresses of all the ISRs
- ISRs: Separate segments of code determine what action should be taken for each type of interrupt
- Interrupt architecture must save the address of the interrupted instruction (*fixed address vs. control stack*)
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*
- Once the interrupt has been serviced by the ISR, the control is returned to the interrupted program.

- Modern operating systems are **interrupt driven**

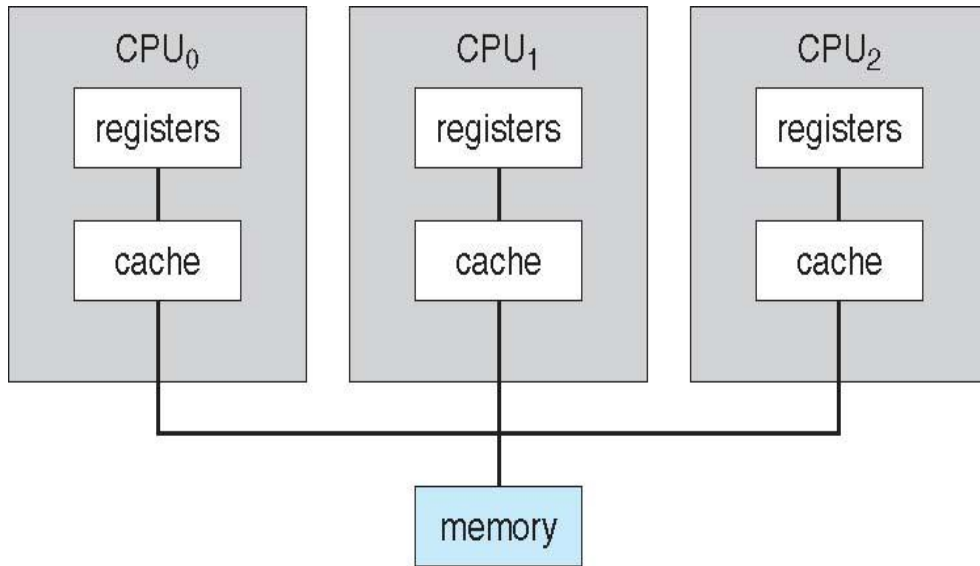
Classes of Interrupts

- **I/O Interrupts**: Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions
- **Timer Interrupts**: Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis
- **Hardware Failure Interrupts**: Generated by a failure (e.g. power failure or memory parity error).
- **Traps (Software Interrupts)**: Generated by some condition that occurs as a result of an instruction execution
 - User request for an operating system service (e.g., system calls)
 - Runtime errors

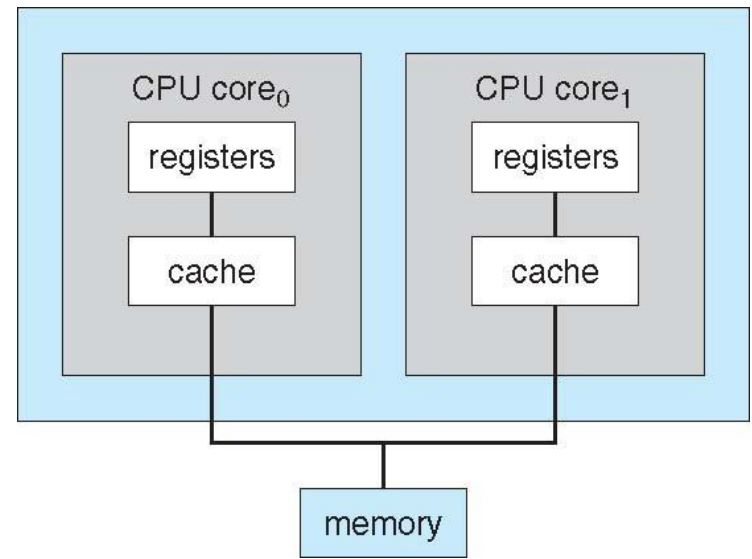
Computer-System Architecture

- Most systems use a **single** general-purpose processor
(PDAs through mainframes)
 - Most systems have special-purpose processors as well
(disk-controller, graphic. Sound processors etc.)
- **Multiprocessors** systems growing in use and importance
 - Also known as **parallel systems**, **tightly-coupled systems**
 - Advantages include
 1. Increased throughput
 2. Economy of scale
 3. Increased reliability – graceful degradation or fault tolerance
 - Two types
 1. **Asymmetric Multiprocessing** *(a master processor manages the others master-slave relation)*
 2. **Symmetric Multiprocessing** *(all processors are peers, each processor performs any task in OS)*

Multiprocessing Architectures



A single-Core Design

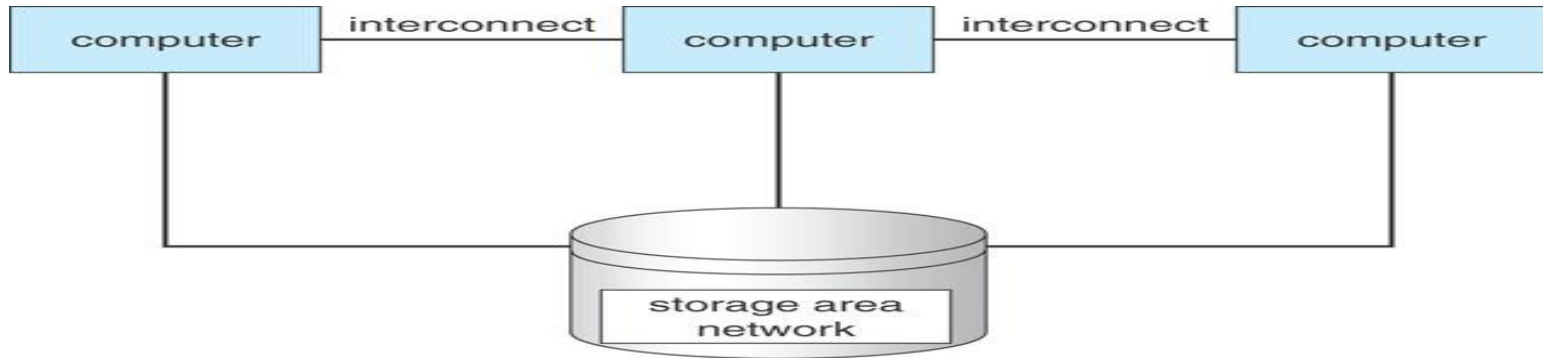


A Dual-Core Design

- Same hardware can be used as symmetric or asymmetric (depends on OS)
- CPUs can have memory, too...
- Uniform Memory Access (UMA) vs. Non-Uniform Memory Access (NUMA)
 - Access time is the same vs. different

Example: Blade servers (multiple multiprocessor systems)

Clustered Systems



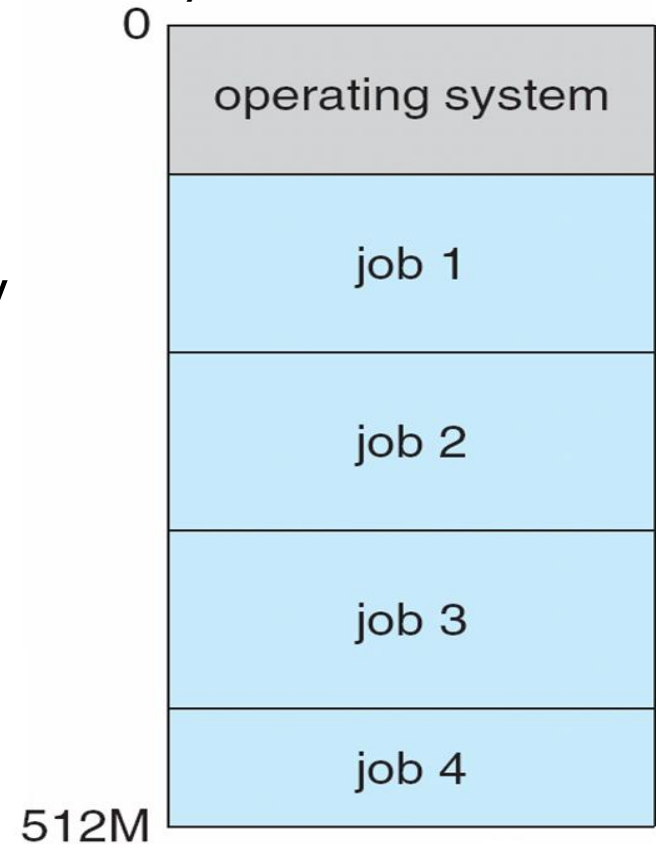
- Like multiprocessor systems, but multiple systems working together
 - Usually sharing storage via a **storage-area network (SAN)**
 - Provides a **high-availability** service which survives failures
 - **Asymmetric clustering** has one machine in hot-standby mode
 - **Symmetric clustering** has multiple nodes running applications, monitoring each other
 - Some clusters are for **high-performance computing (HPC)**
 - Applications must be written to use **parallelization**

Example : Beowulf Cluster (multiple low-cost computer systems)

Operating System Structure

- Single user cannot keep CPU and I/O devices busy at all times,
- How can we better utilize resources then?
 - execute multiple programs
- **Multiprogramming** needed for efficiency
 - 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

Memory Layout for Multiprogrammed System



Operating System Structure (Cont.)

- **Timesharing (multitasking)** 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**
 - CPU is allocated to jobs in **Round-Robin** manner
 - All active users must have a **fair** share of the CPU time: e.g. with 100 ms time quantum
 - 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
- File system – disk management, protection, synchronization, communication, deadlock avoidance are the other important functionalities that the OS must provide...

Operating System Structure (Cont.)

Justifications for Timesharing (multitasking)

Table 1.1. Typical times for components of a computer system. One nanosecond (ns) is 10^{-9} seconds, one microsecond (μs) is 10^{-6} seconds, and one millisecond (ms) is 10^{-3} seconds.

item	time		scaled time in human terms (2 billion times slower)	
processor cycle	0.5 ns	(2 GHz)	1	second
cache access	1 ns	(1 GHz)	2	seconds
memory access	15 ns		30	seconds
context switch	5,000 ns	(5 μs)	167	minutes
disk access	7,000,000 ns	(7 ms)	162	days
quantum	100,000,000 ns	(100 ms)	6.3	years

From USP, Robbins

Process Management

Memory Management

Storage Management

I/O Subsystem

Protection and Security

MAJOR OS TASKS

Process Management

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

!!! MORE
LATER !!!

- 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

!!! MORE
LATER !!!

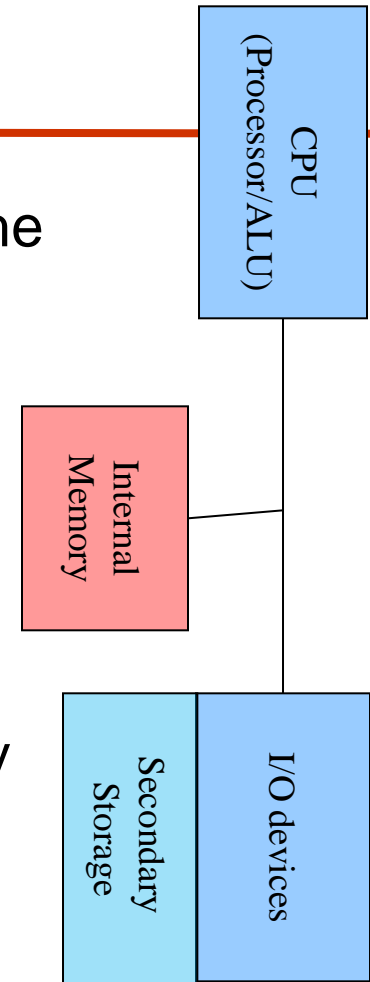
- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
 - Optimizing CPU utilization and computer response to users
- Memory management activities
 - Keeping track of which parts of memory are currently being 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
 - ▶ Mapping files onto secondary storage
 - ▶ Backup files onto stable (non-volatile) storage media

Storage Structure

- **Main memory** is the only large storage media that the CPU can access directly *(to get/store program instructions + data)*
 - RAM, DRAM vs. ROM, EEPROM
 - Array of words,
 - Each word has an address
 - load/store instructions
 - Too small
 - Volatile
- **Secondary storage** is an extension of main memory that provides large **nonvolatile** storage capacity
 - 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

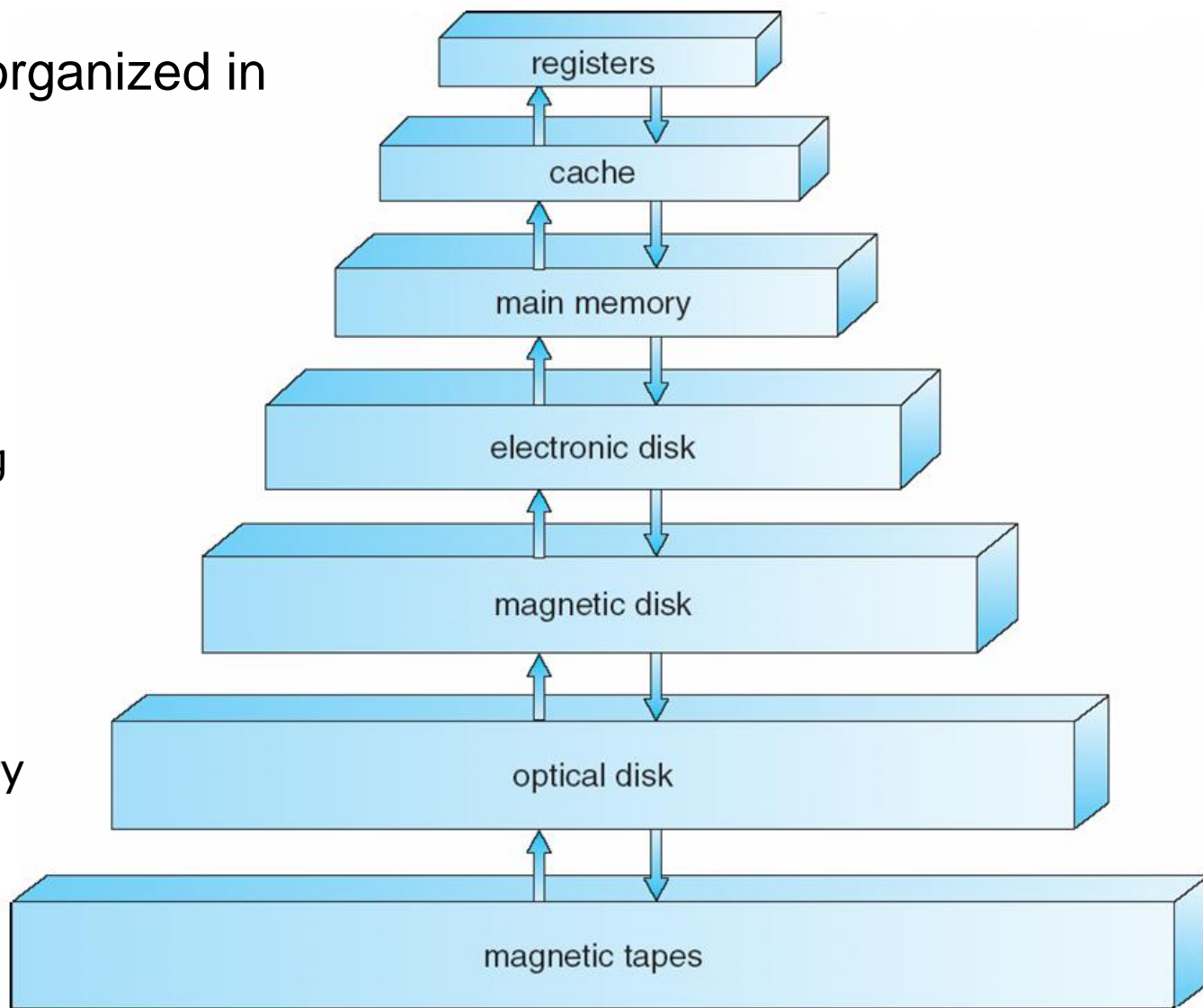


Storage Hierarchy

Storage systems organized in hierarchy

- Speed
- Cost
- Volatility

Caching – copying information into faster storage system; main memory can be viewed as a last *cache* for secondary storage



Caching

- 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
- Cache smaller than storage being cached
 - Cache management important design problem
 - Cache size and replacement policy

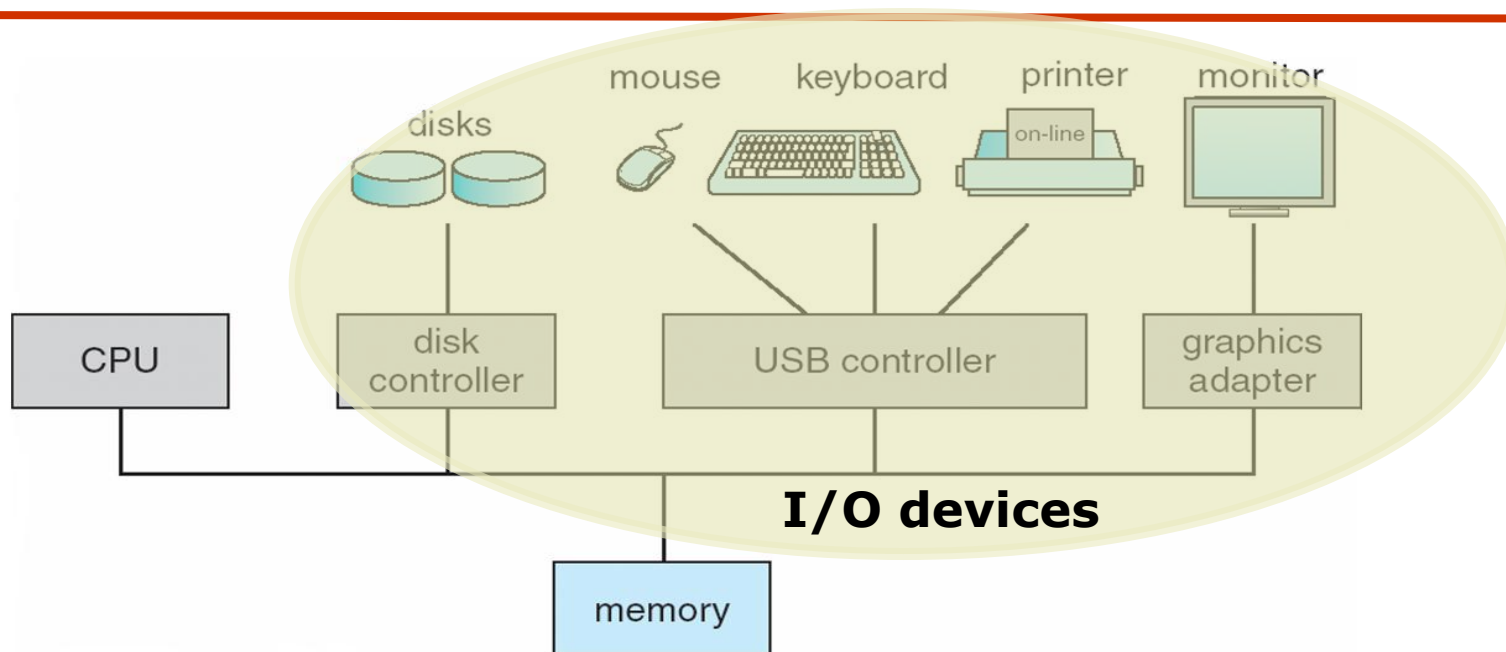
Mass-Storage Management

- 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)

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
- **Interrupt handlers** and **device drivers** are crucial in the design of efficient I/O subsystems

I/O Structure



- Storage is only one of **many types** of I/O devices, but there are several others
- Each device has a device controller (more devices can be attached to one controller – USB, SCSI)
 - Each controller has local buffer/registers
- OS has a **device driver (dd)** for each device (interface for the controller)
- To start an I/O operation, device driver loads the appropriate registers in the controller

I/O Structure

- After I/O starts, control returns to user program 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

- After I/O starts, control returns to user program without waiting for I/O completion
 - **System call** – request to the operating system 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
 - Operating system indexes into I/O device table to determine device status and to modify table entry to include interrupt

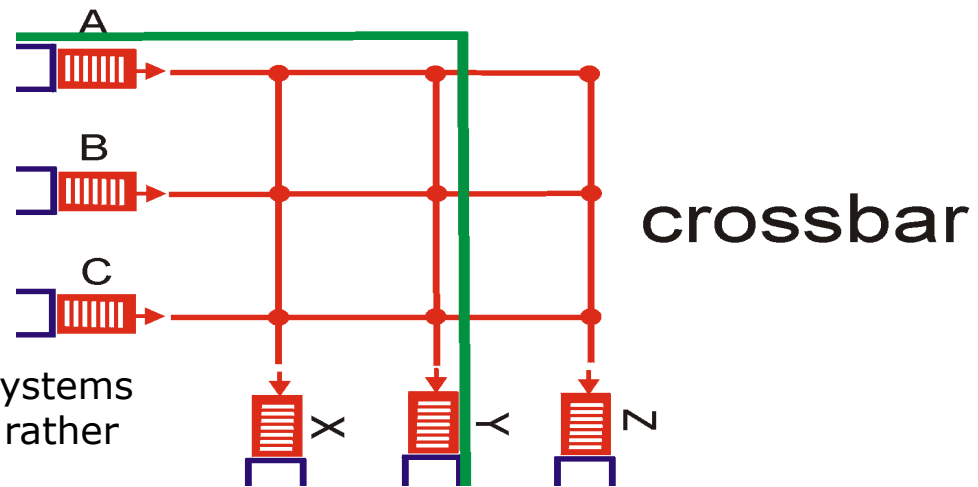
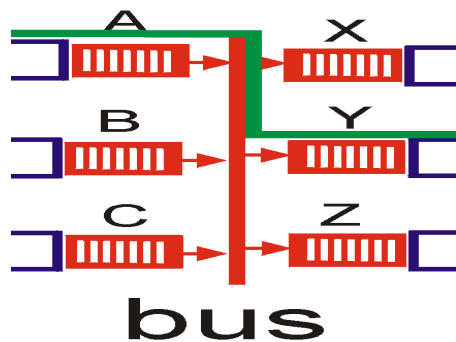
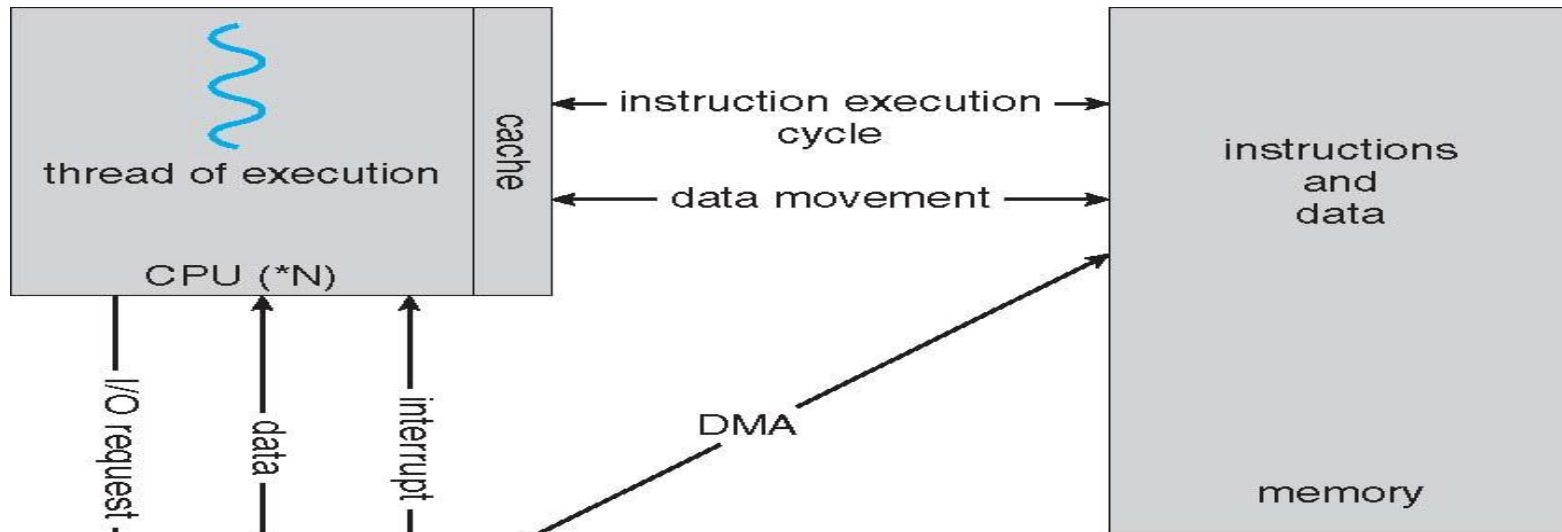
Blocking

Non-blocking

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

How a Modern Computer Works



High-end systems use switch rather than bus architecture. Why?

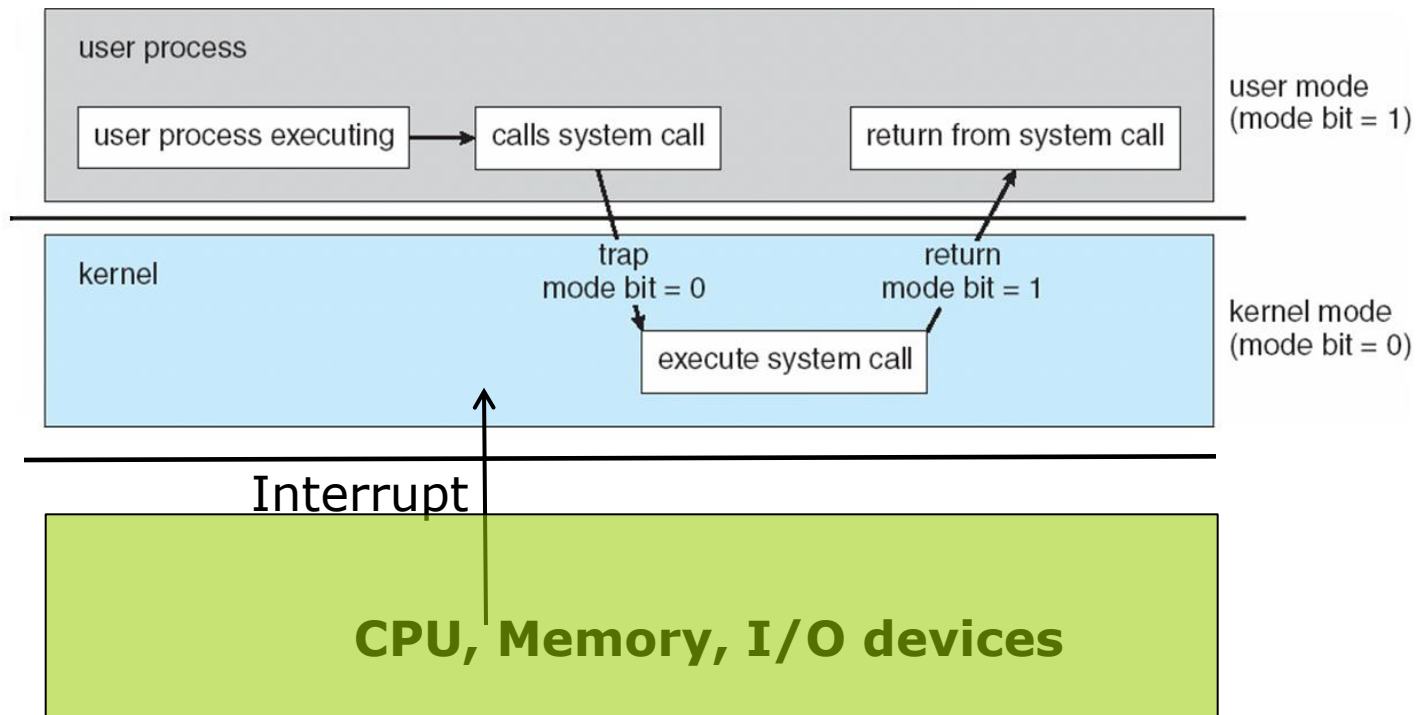
Protection and Security

- **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

Protection and Security (cont'd)

- Modern OSes are **Interrupt driven**
 - HW Interrupt ,
 - SW error or request (**trap**)
 - ▶ Division by zero, request for operating system service
 - For each interrupt there is a **ISR** (interrupt service routine)
- OS and user programs share the same HW and SW resources
 - Error in one program may cause errors in another program or in OS
 - ▶ For example: infinite loop, processes modifying each other or the operating system
 - **Protection** is essential to the proper execution of OS
- To protect OS and other system components, we need HW support to differentiate various modes of execution
 - **Dual-mode** operation: **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

Transition from User to Kernel Mode



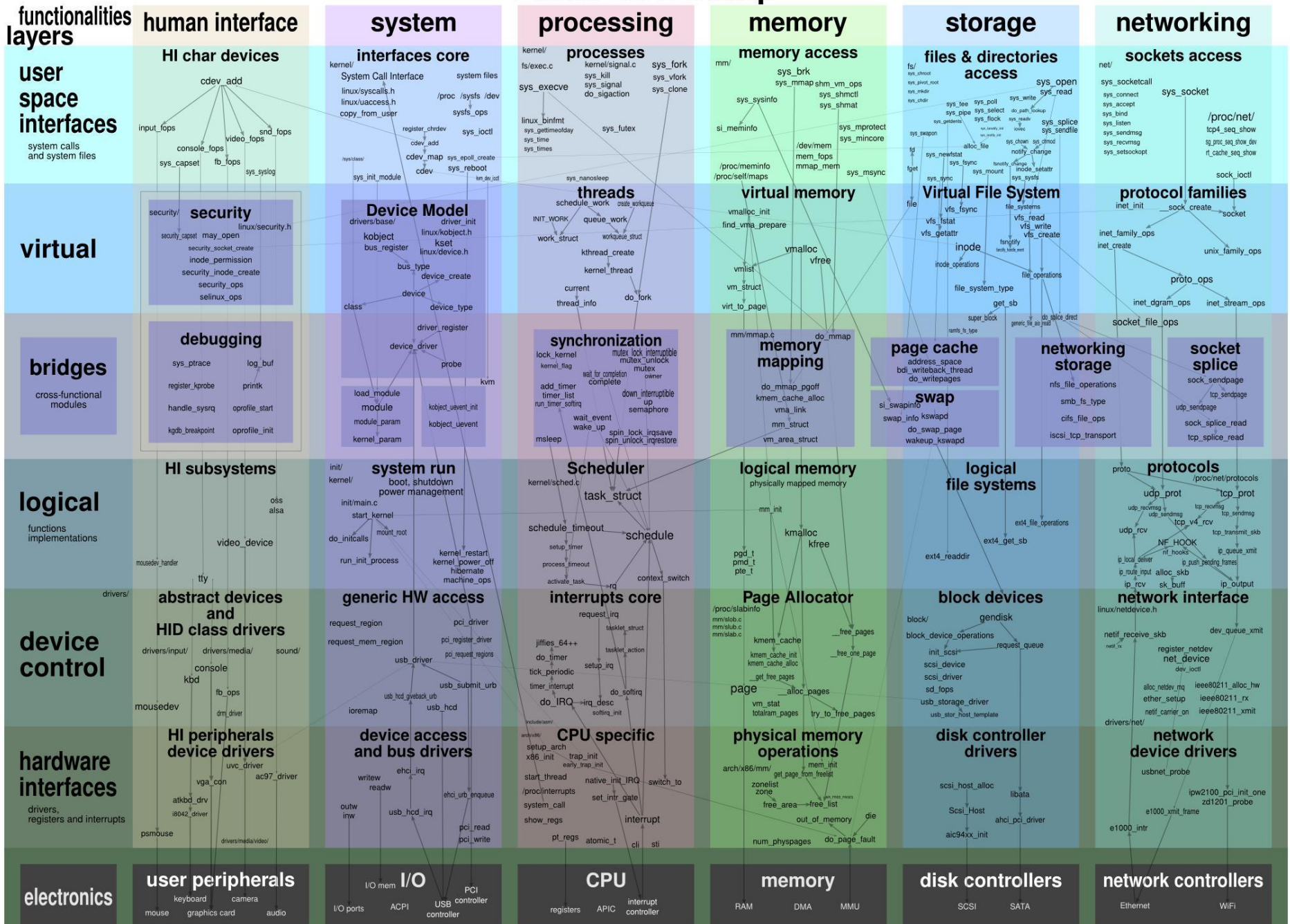
Kernel Data Structures

- List
- Stack,
- Queues
- Trees
- Hash functions
- Bitmaps

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**

Linux kernel map



OS DIFFERENT SYSTEMS AND OS TYPES

Distributed Systems

!!! MORE
LATER !!!

- Collection of physically separate, possibly heterogeneous, computer systems that are networked to provide the users with access to various resources.
- Network
 - (LAN, MAN, WAN)
 - Network OS
- Advantages of distributed systems.
 - **Resource Sharing**
 - Computation speed up – load sharing
 - Reliability
 - Communications

Special-Purpose Systems

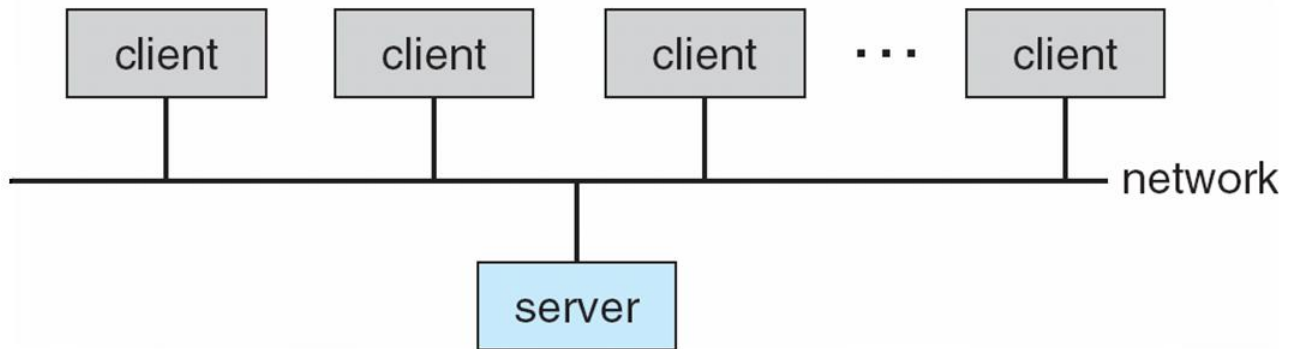
- **Real-time** systems: rigid time requirements on the operations of a processor or the flow of data
 - **Hard** real-time: critical tasks must be completed on time
 - **Soft** real-time: no absolute timing guarantees (e.g. “best-effort scheduling”); multimedia applications;
- An **embedded** system is a component of a more complex system
 - Control of home and car appliances (microwave oven, DVD players, car engines, ...)
 - Control of a nuclear plant or Missile guidance
- Multimedia Systems
- Handheld Systems

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

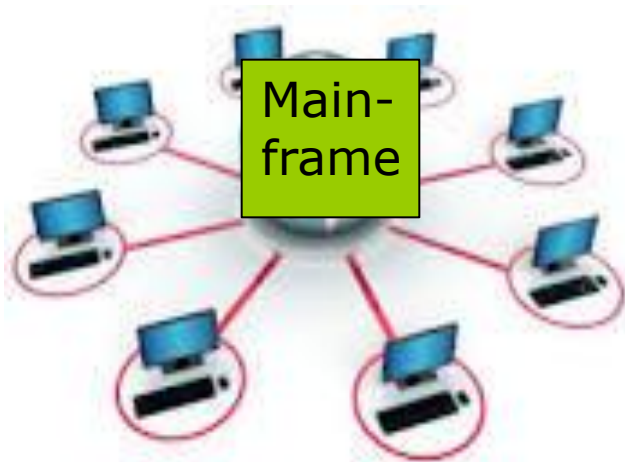
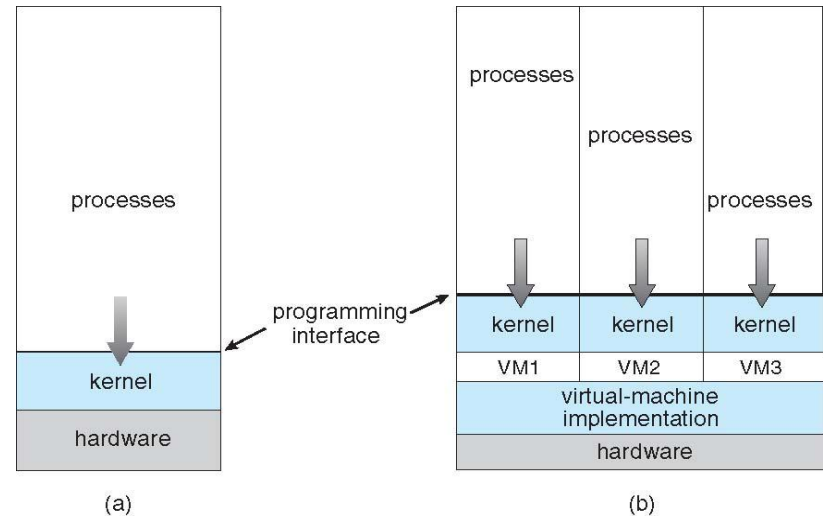
- Another model of distributed system
- P2P does not distinguish clients and servers
 - Instead all nodes are considered peers
 - May each act as client, server or both
 - Node must join P2P network
 - Registers its service with central lookup service on network, or
 - Broadcast request for service and 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

Virtualization and Cloud Computing

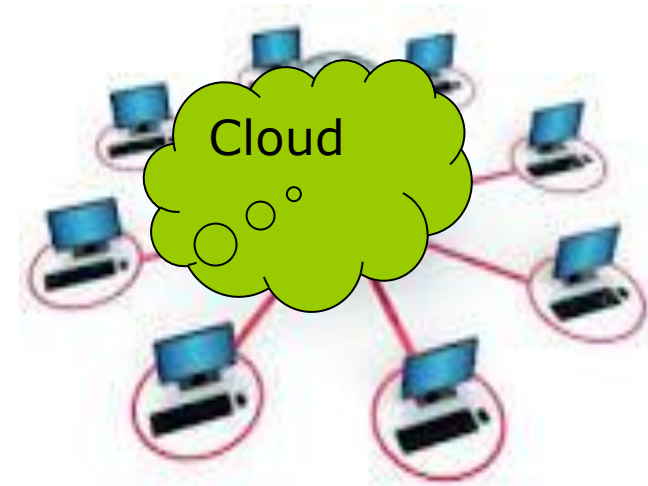
- **Virtualization:** Run an OS as Application in another OS
- Emulation, Interpretation
- **Cloud Computing**
 - SaaS, PaaS, IaaS, XaaS...



Then...



Along the way...



Now...

End of Chapter 1

