Introduction to Operating Systems and Concurrent Programming

EECS 338, Spring 2016

Eamon Johnson

Chapter 1: Introduction

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:
  - Provide abstractions and services:
    - Facilitate the execution of application programs.
    - Make the computer system easy/convenient to use.
  - Manage the underlying hardware resources in an efficient manner.

What Operating Systems Do

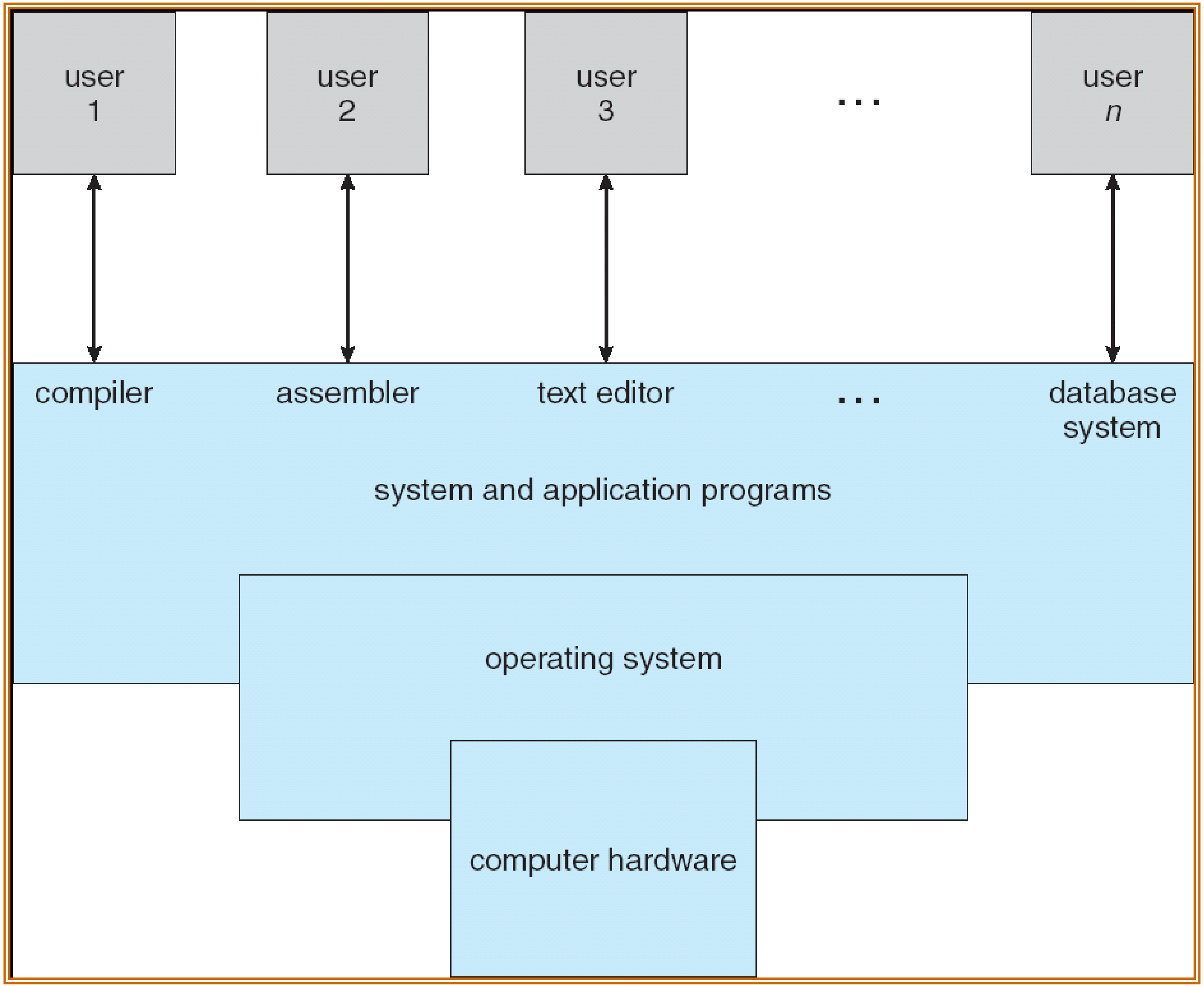
- Depends on the point of view.
- Users want convenience, **ease of use.**
  - Don’t care about **resource utilization.**
- Shared computers such as **servers** must keep all users happy.
- Users of dedicated systems such as **workstations** have dedicated resources, but frequently use shared resources from **servers.**
- Handheld computers are resource poor, optimized for usability and battery life.
- Some computers have little or no user interface, such as embedded computers in devices and automobiles.

Operating System: Different Definitions

- OS is
  - A resource allocator – manages and allocates resources.
  - A control program – controls the execution of user programs and operations of I/O devices
    - Often said to be the kernel – the one program running at all times (all else being application programs).

But, first, back to Computer Systems

Four Components of a Computer System



Computer Startup

- **Bootstrap program** is loaded at power-up or reboot.
  - Typically stored in ROM (read-only memory) or EPROM (erasable programmable read-only memory), generally known as **firmware.**
  - Initializes all aspects of system.
  - Loads operating system kernel and starts execution.

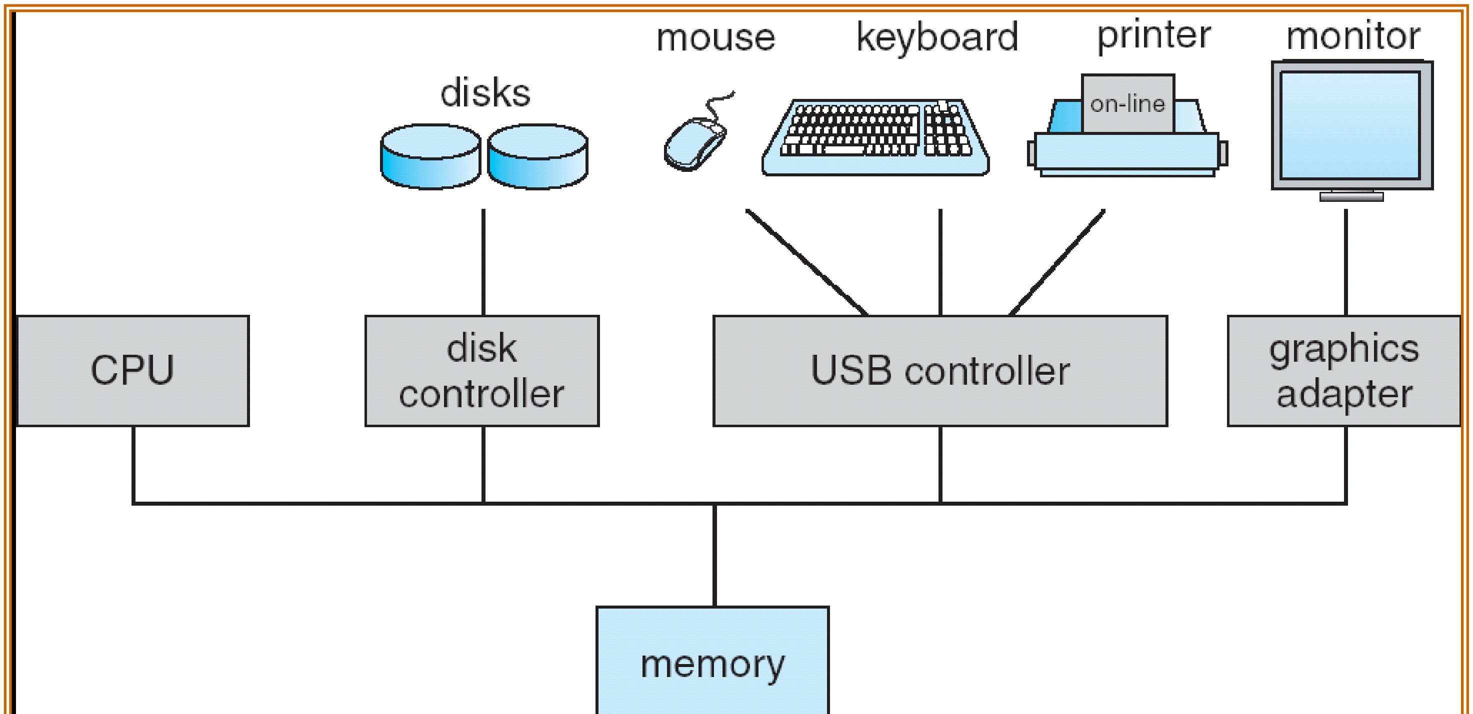
Computer System Components

- Users
- Applications programs – define the ways system resources used to solve users' problems :
  - Compilers
  - database systems
  - business programs
  - video games, etc

  Some of the above are "systems programs".
- Operating system – controls and coordinates the use of the hardware among various application programs for various users.
- Hardware – provides basic computing resources.
  - CPU, memory, I/O devices (disks, CDs, sound cards, network).

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.



Why do we need to know the computer-system structures?

Computer System Operation

- I/O devices and the CPU 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/to the device to/from local buffer of controller.
- Device controller informs CPU that it has finished its operation by causing an **interrupt.**

From Jeff Dean’s Slides, 2007

Numbers Everyone Should Know

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	100 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	10,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from network	10,000,000 ns
Read 1 MB sequentially from disk	30,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

How are interrupts serviced?

- 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.
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*.
- A ***trap*** or ***exception*** is a software-generated interrupt caused either by an error or a user request.
- Most commercial OSs device interrupts are interrupt-driven, not polling-driven.
- Real-time OSs with hard deadlines are polling-driven.

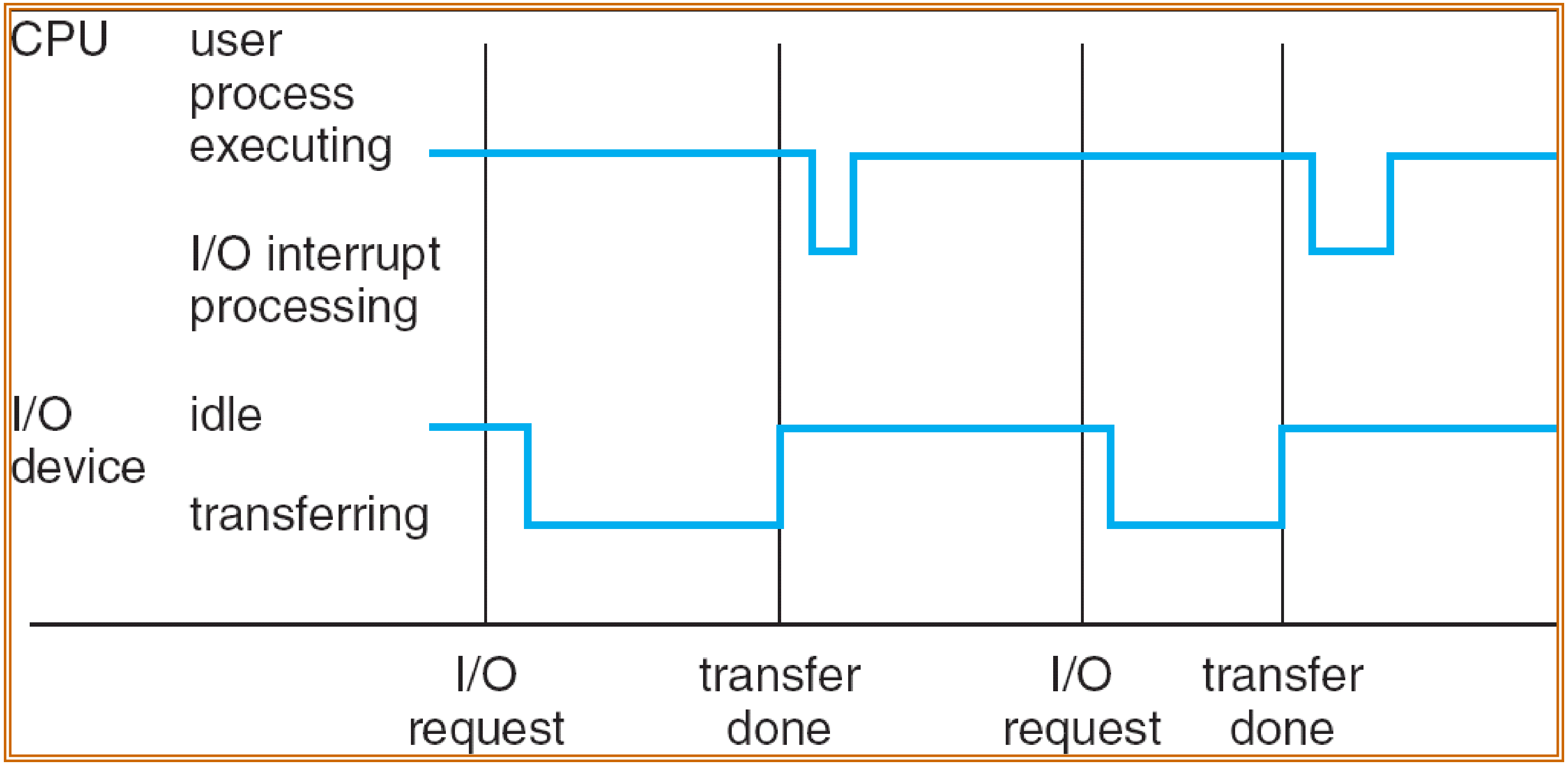
Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter.
- Determines which type of interrupt has occurred:
  - ***vectored interrupt system***
- Separate segments of code determine what action should be taken for each type of interrupt.

The first computer virus in MS-DOS took over an interrupt!

- Two different types of "event" handling within OSs:
  - Interrupt-driven
  - Polling-driven

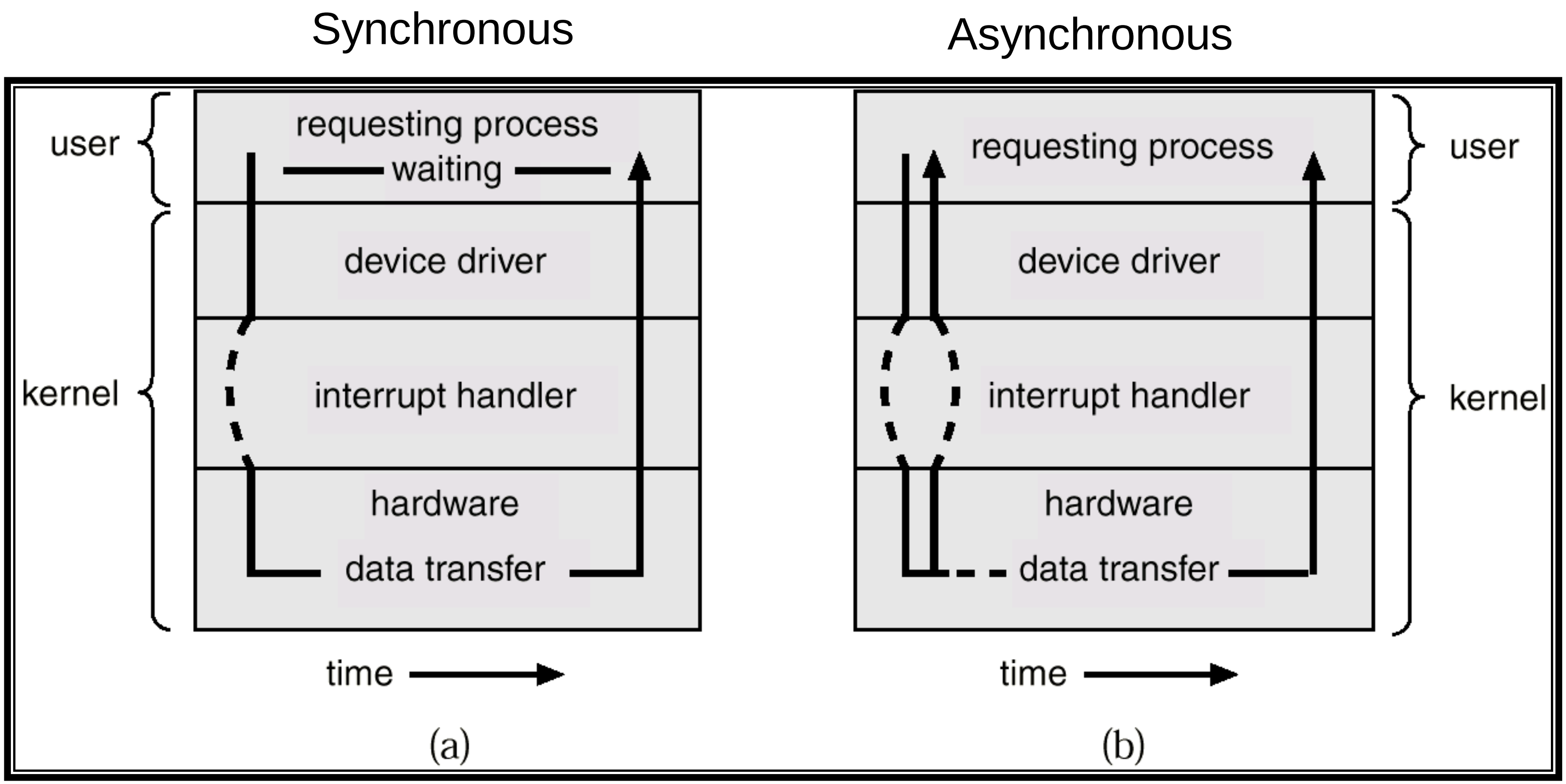
Interrupt Timeline



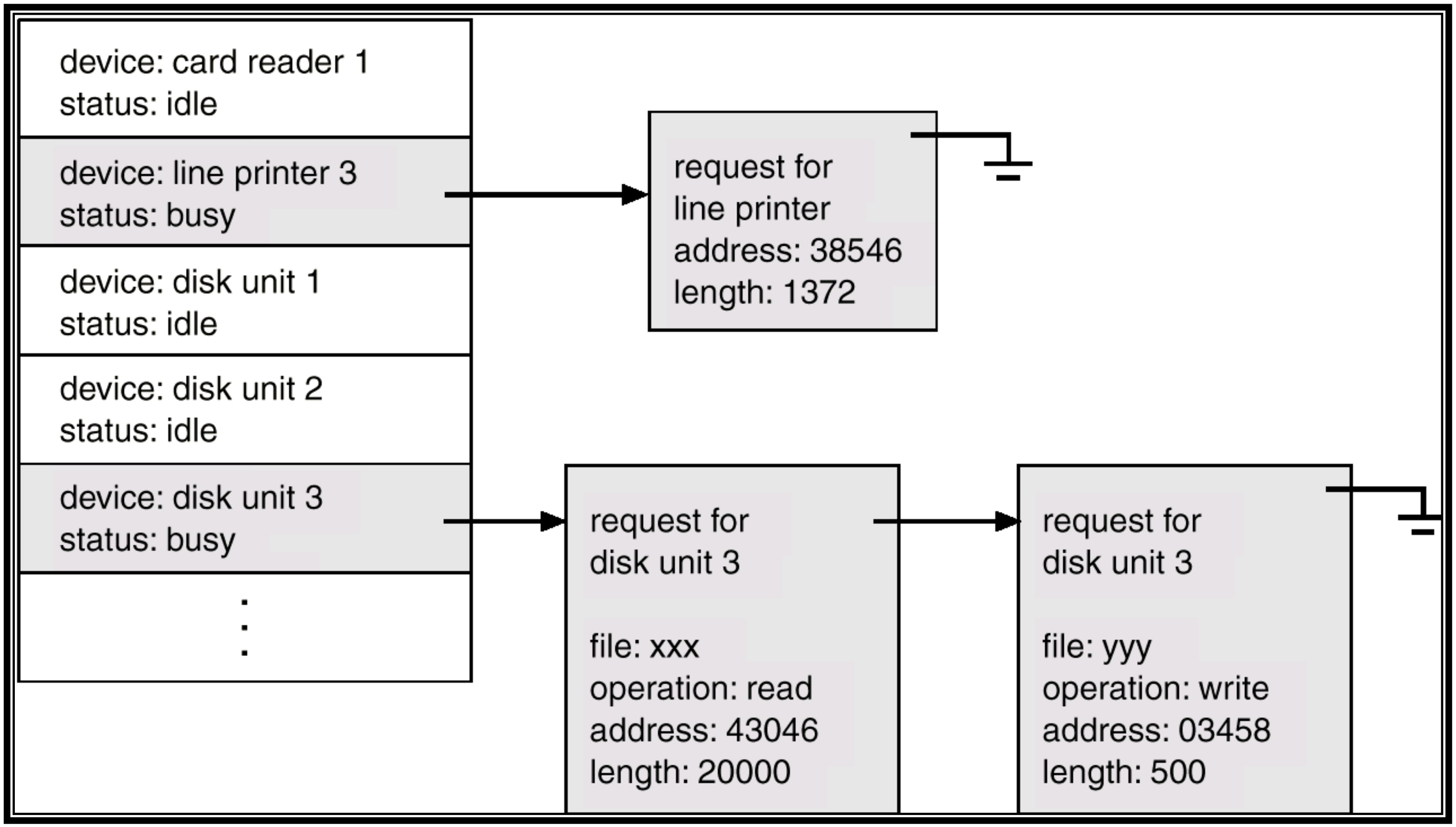
I/O Structure

- **Alternative #1:** 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.
- **Alternative #2:** 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.

Two I/O Methods



Device-Status Table



Direct Memory Access (DMA) Structure

- **Used for high-speed I/O devices** to 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 one interrupt per byte or word.

Why do we need DMA?

Storage Size Definitions

- Bit: basic unit of storage; 0 or 1.
- Byte: 8 bits.
- Word: Native unit of data of computers:
  - 32-bit-, 64-bit-, or 128-bit words
- Sizes
  - KB (kilobyte; 1,024 bytes)
  - MB (megabyte; 1,024 KB)
  - GB (gigabyte; 1,024 MB)
  - TB (terrabyte; 1,024 GB)
  - Petabyte (1,024 TB).
  - And, so on: Exabyte; Zettabyte; Yottabyte

Storage Structures

- Main memory – only large storage media that the CPU can access directly: random access; volatile.
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity.
- Magnetic disks as secondary storage (electromechanical) --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.
- Solid state disks.
  Faster than magnetic disks; nonvolatile; no moving parts.
  Nevertheless, exhibit push/pull, and read/write time differences.

  Rui Zhang, et. al., "The HVtree: a Memory Hierarchy Aware Version Index", VLDB 2010.

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.

Storage-Device Hierarchy

netic disk

optical disk

magnetic tapes

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 is smaller than storage being cached.
  - **Cache management**: important design problem.
  - Decisions to make: **Cache size** and **replacement policy.**

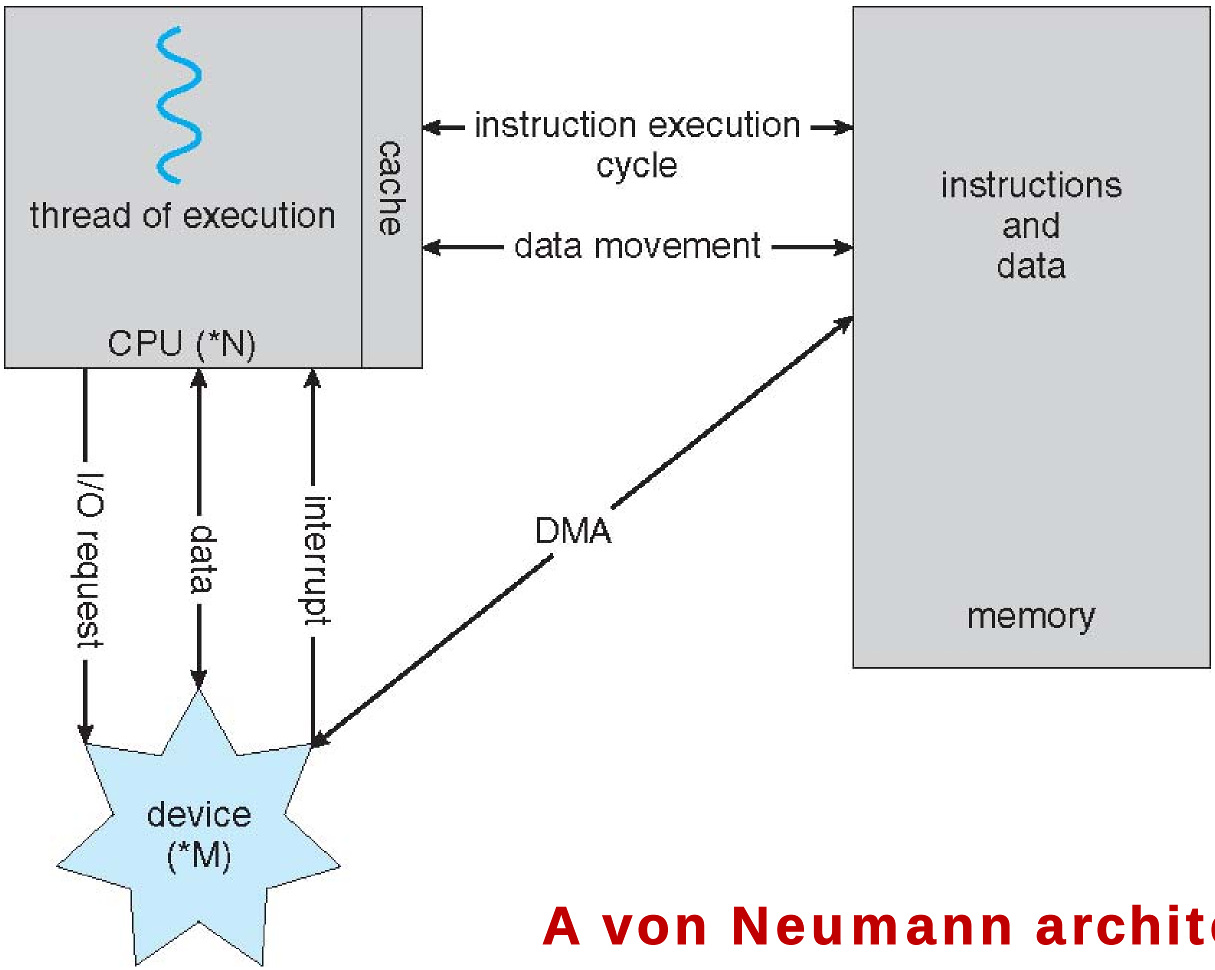
Computer-System Architecture

- **Most systems used to have a single general-purpose processor** (PDAs through mainframes)! **Not true anymore for new computers!**

- **Multiprocessors** systems growing in use and importance.
  - Also known as **parallel systems**, **tightly-coupled systems.**
    - Characterized by **memory and/or clock sharing**.
      Are multicore systems "parallel systems"?
  - Advantages include:
    - **Increased throughput**
    - **Economy of scale**
    - **Increased reliability – graceful degradation** or **fault tolerance.**

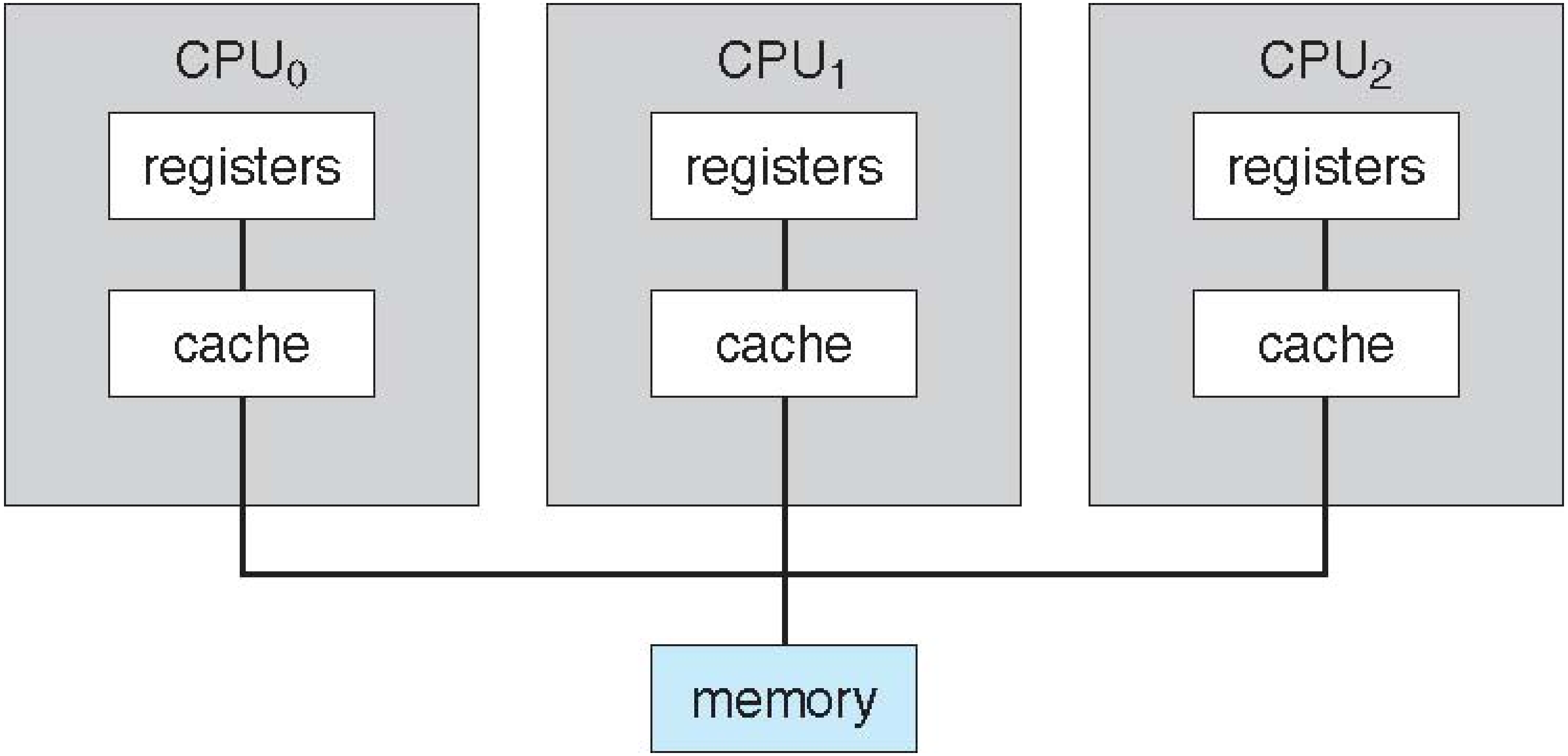
  - Two types:
    1. **Asymmetric Multiprocessing**
    2. **Symmetric Multiprocessing**

How a Modern Computer Works: A Single or N-Processor System



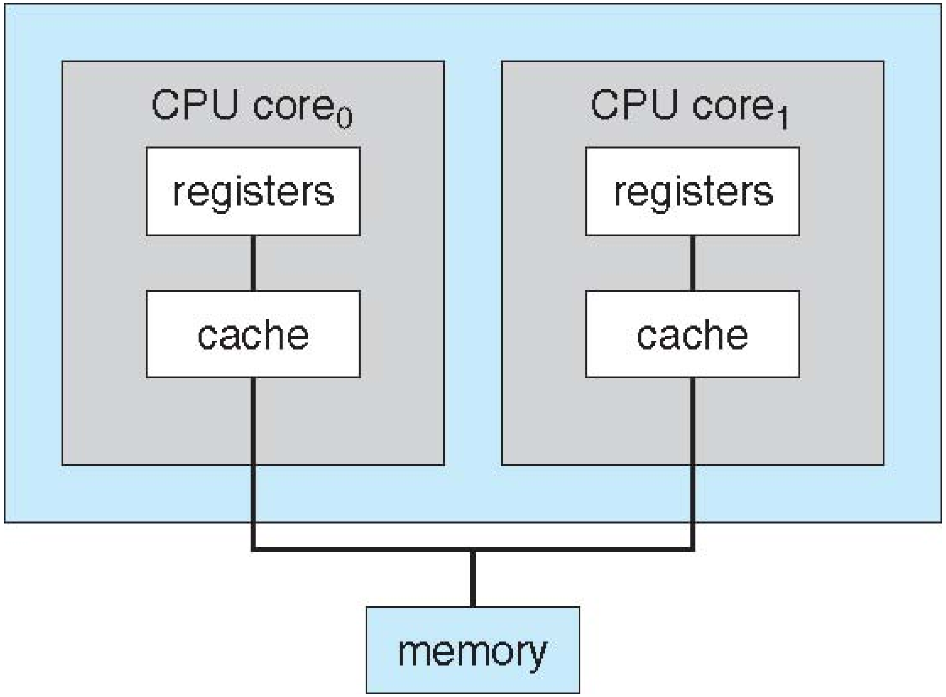
A von Neumann architecture

Symmetric Multiprocessing Architecture: Also a Von Neumann Architecture



A Dual-Core Design

- **UMA (**Uniform memory Access**)** and
  NUMA (Non-uniform memory access)
  architecture variations.
- Multi-chip and **multicore**
- **Each core** now **with 8 or more hardware threads!**



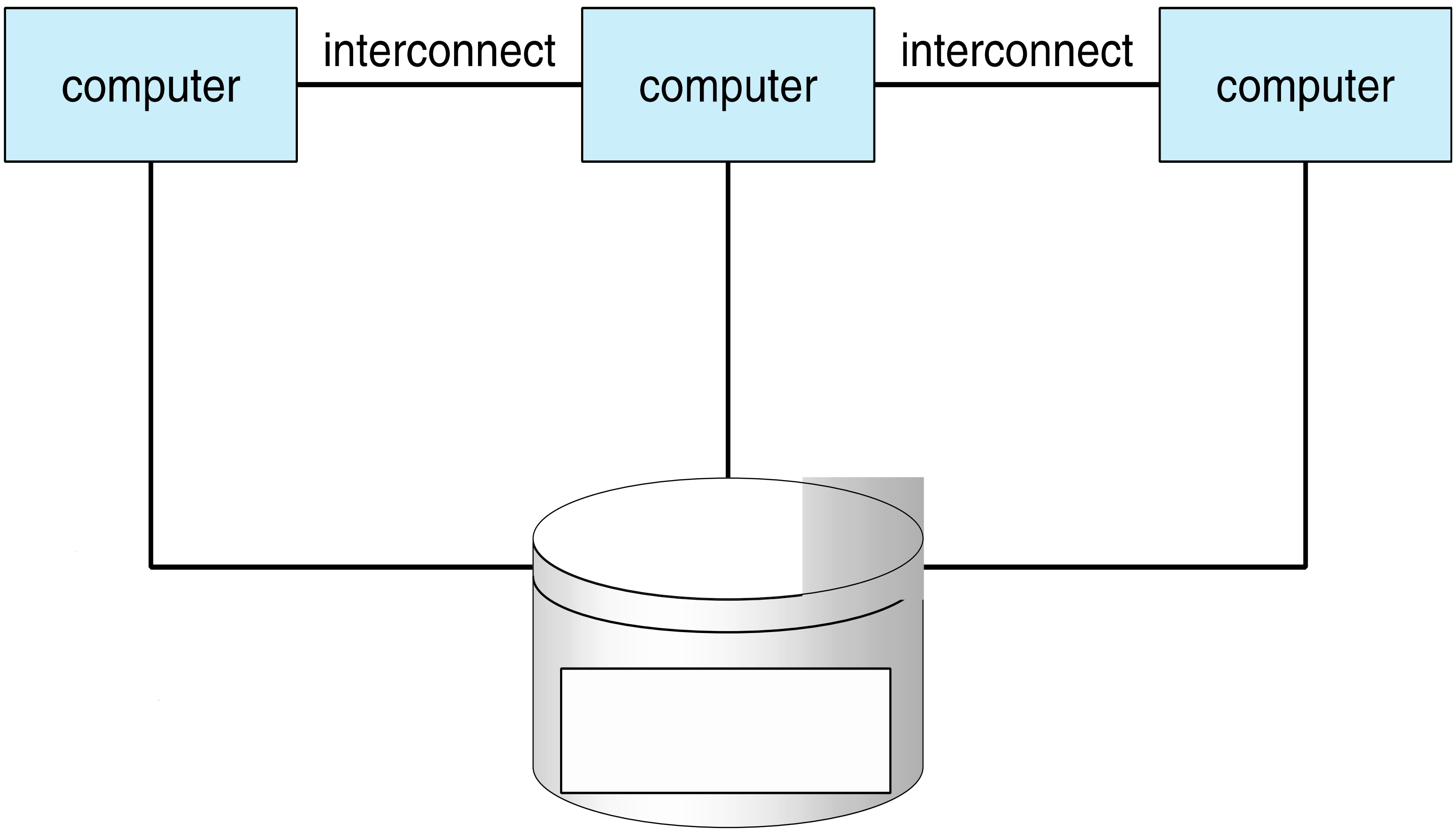
Clustered Systems

- Like multiprocessor systems, but multiple systems working together.
  - Not networked system of computers.
  - 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.**

  - Some have **distributed lock manager (DLM)** to avoid conflicting operations.

Clustered Systems



Operating Systems (OS)

Operating System Services

- Provide functions helpful to user:
  - **User interface** - All OSs have a user interface (UI).
    - Varies between Command-Line Interface (CLI), Graphics User Interface (GUI), Batch, etc.
  - **Program execution** - The system must be able to load a program into memory, run it, and terminate execution--normally or abnormally (indicating error)
  - **I/O operations** - A running program may require I/O, which may involve a file or an I/O device.
  - **File-system manipulation** - of particular interest in organizing information in external storage. Programs need to read and write files/directories, create/delete/search, list file info, and manage permissions.

Operating System Services (Cont.)

- **Communications** – Processes exchange information, on the same computer or between computers over a network.
  - Communications may be via shared memory or through message passing (packets moved by the OS).
- **Error detection** – OS needs to be constantly aware of possible errors
  - May occur in the CPU and memory hardware, in I/O devices, in user program.
  - For each type of error, OS should take the appropriate action to ensure correct and consistent computing.
  - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system.

Operating System Components

- In order to provide these services, an OS contains various functional components
  1. **Process/Thread management**
  2. **Memory management**
  3. **Storage management (file systems, disks)**
  4. **Mass-storage management**
  5. **IO subsystem (device drivers, e.g., communication)**
  6. **Protection and security**

1. Process/Thread Management
Job, Process, Thread: Difference?

- **Job:** A user program in its entirety. Terminology from batch operating systems.
  Mostly used in mainframes.
  But, in time-sharing OSs, sometimes, people refer to the execution of an app as a whole as a “job”.

- **Process:** A program in action.

  process ≠ program
  process ≠ job

- **Thread:** A separate line of control (execution) within a process.

  thread ≠ process

From Jeff Dean's Slides, 2007

Threads

If you're not using threads, you're wasting ever larger fractions of typical machines

Machines have 4 cores now, 8 soon, going to 16.

- think about how to parallelize your application!
- multi-threaded programming really isn't very hard if you think about it upfront

- Threading your application can help both throughput and latency

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 reclaim 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 OS-processes, running concurrently on one or more CPUs.
  - Concurrency by multiplexing the CPUs among the processes / threads.

Process Management Activities

The OS 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.

Operating System Structure

- **Multiprogramming** needed for efficiency.
  - Single user 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**)--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** is< 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.**
  - 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.

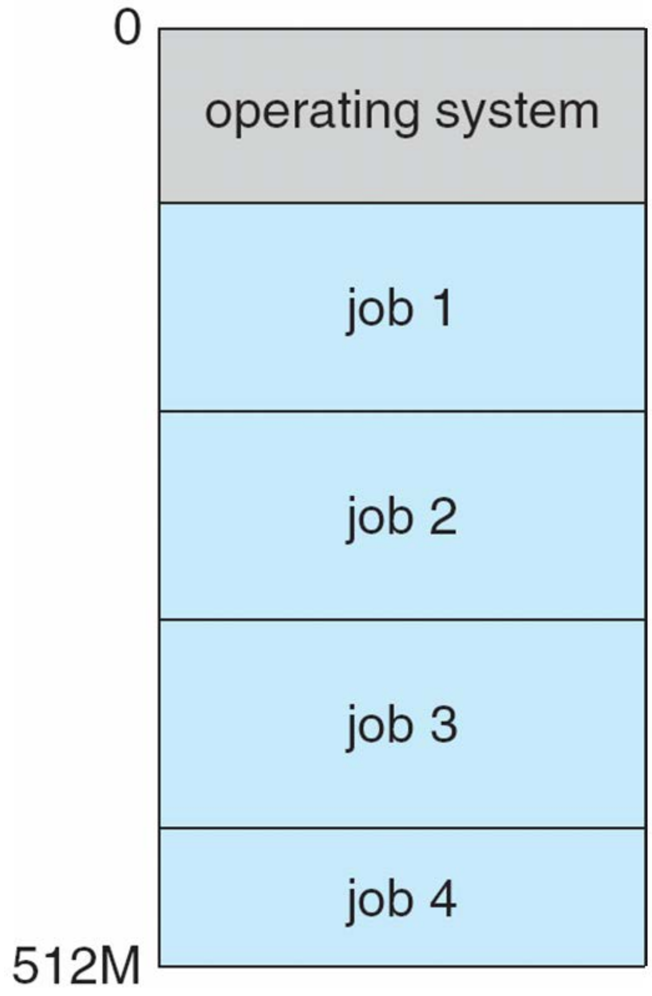
2. Memory Management

- All data of a process in memory--before and after processing.
- All instructions in memory in order to execute.
- Determines what is in memory when
  - Optimizing CPU utilization and computer response time.
- 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 reclaiming memory space as needed.
- Memory Protection: Base register + offset registers

Main memory Sizes

- Main memory sizes keep escalating.
  - 32 TB of main memory in a server with 384 processor cores: See http://www.oracle.com/us/corporate/press/2016809
- Main-Memory-Only databases are becoming common.

Memory Layout for Multiprogrammed System



OS Operations

- **Interrupt driven** (but, sometimes, **polling-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.

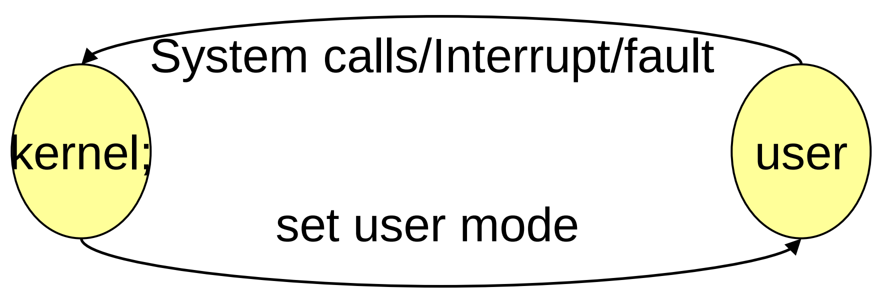
- **Dual-mode** operation allows OS to protect itself and other system components.
  - **User mode** and **kernel mode.**
  - **Mode bit** provided by hardware.
    - Provides ability to distinguish when system is running user code or kernel code.
    - Some instructions designated as **privileged**, only executable in kernel mode.
    - System call changes mode to kernel, return from call resets it to user.

- Increasingly CPUs support multi-mode operations.
  - i.e. **virtual machine manager** (**VMM**) mode for guest **VMs**

User Mode and Kernel Mode

Switch of modes:

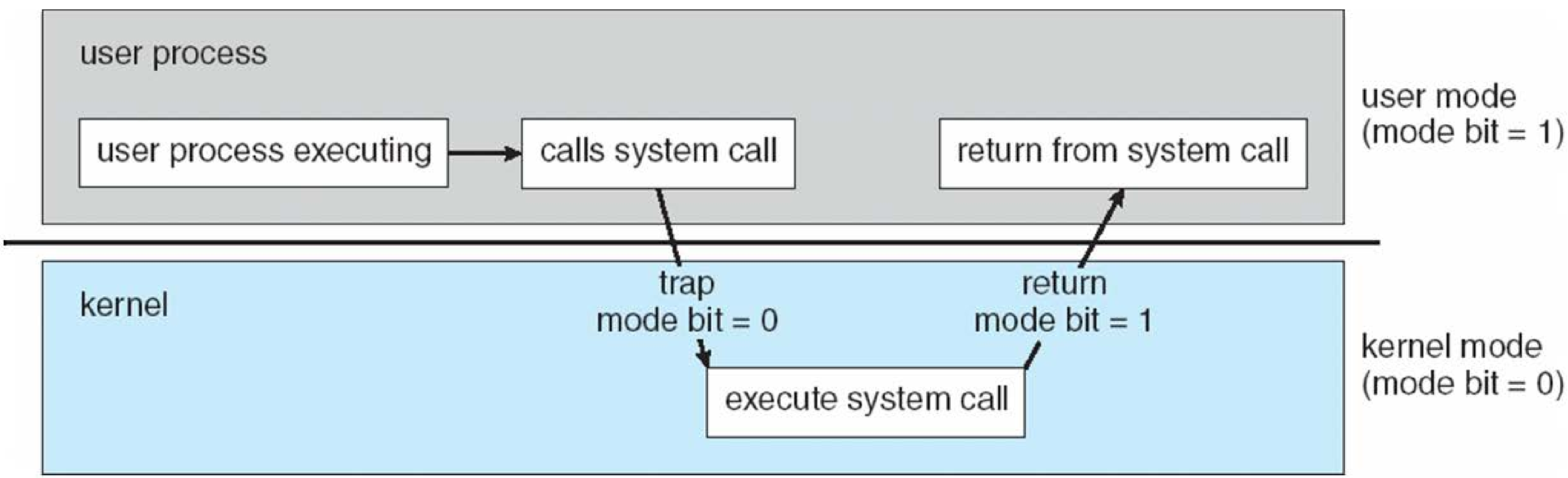
- Interrupt driven by hardware.
- Software error or request creates **exception** or **trap; e.g.,**
  - Division by zero, request for operating system service
  - Illegal access (modification) to the operating system kernel.



Transition from User to Kernel Mode

Example: Timer to prevent infinite loop / process hogging resources.

- Set interrupt after specific period.
- Operating system decrements counter.
- When counter zero generate an interrupt.
- Set up before scheduling process to regain control or terminate program that exceeds allotted time.



3. 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 directories,
    - Mapping files onto secondary storage, and
    - Backup files onto stable (non-volatile) storage media.

4. Mass-Storage Management

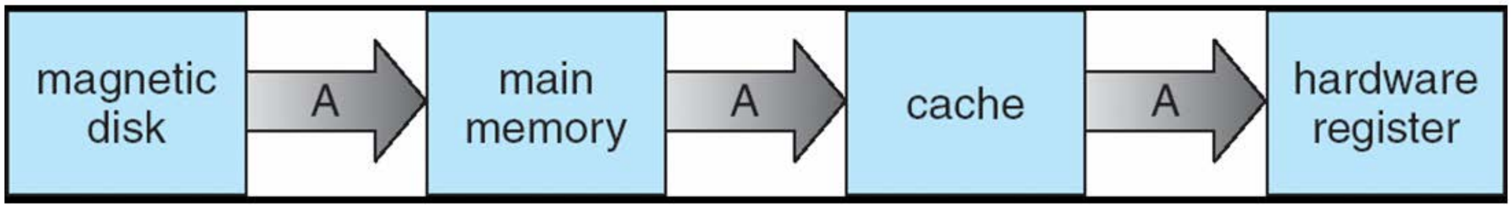
- Disk is used to store data that does not fit in main memory or data that must be kept for a “long” period of time.
- Proper management is of central importance.
- Entire speed of computer operation hinges on disk subsystem and its algorithms.
- OS activities
  - Free-space management,
  - Storage allocation,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

Performance of Different Levels of Storage

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk 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 is even more complex.
  - Several copies of a datum can exist.
  - Various solutions covered in Distributed Systems Chapter.

5. I/O Subsystem Management

- One purpose of OS is to hide peculiarities of hardware devices from the user (**device independence** concept from 1970's)

- 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.

7. 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 and file.
  - **Privilege escalation** allows user to change to **effective ID** with more rights.

How to Protect Resources?

- CPU Protection: Timer interrupts
- OS Protection:

  CPU has user/supervisory (kernel) modes

  "mode" bit

  user mode: OS executes user program on user's behalf.

  All windows XP/NT/7/8 subsystems execute in user mode.

  Instruction set: ordinary and privileged instructions.

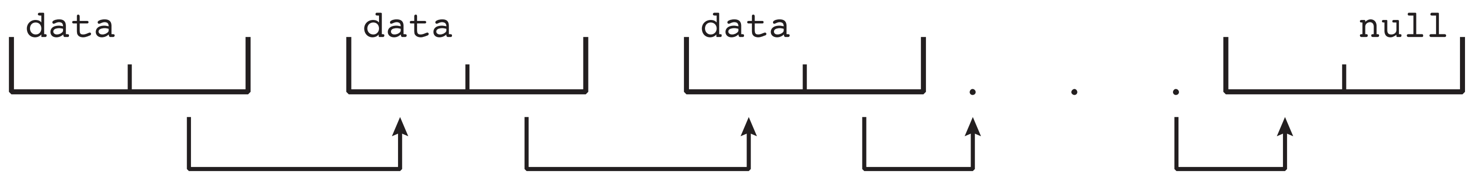
  All I/O instructions are privileged instructions.

  Security breach: user program executing when the mode bit is set to a "supervisory" mode.

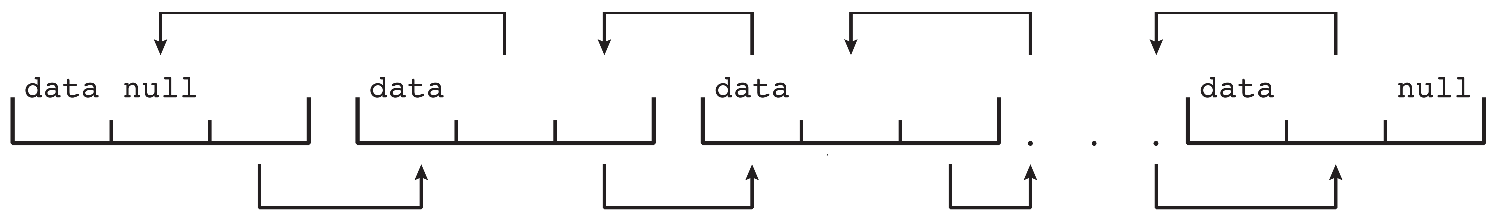
Memory protection: memory bound registers and out-of-bound access interrupts (traps)

More on OS: Kernel Data Structures

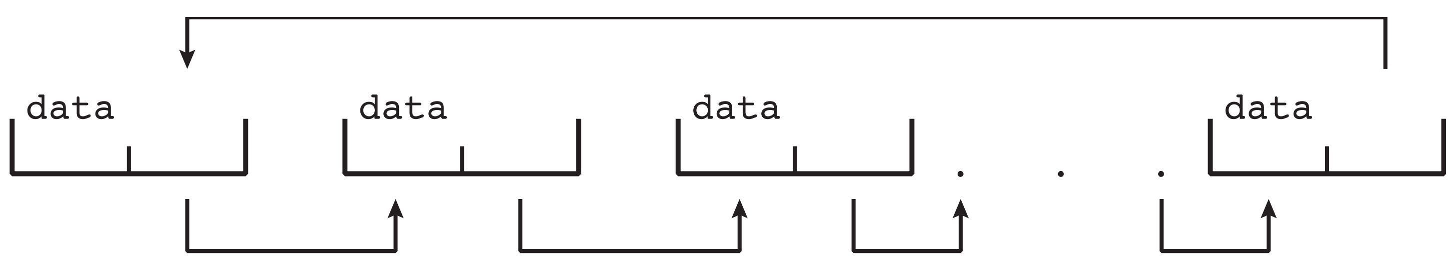
- Use standard programming data structures.
- ***Singly linked list***



- ***Doubly linked list***

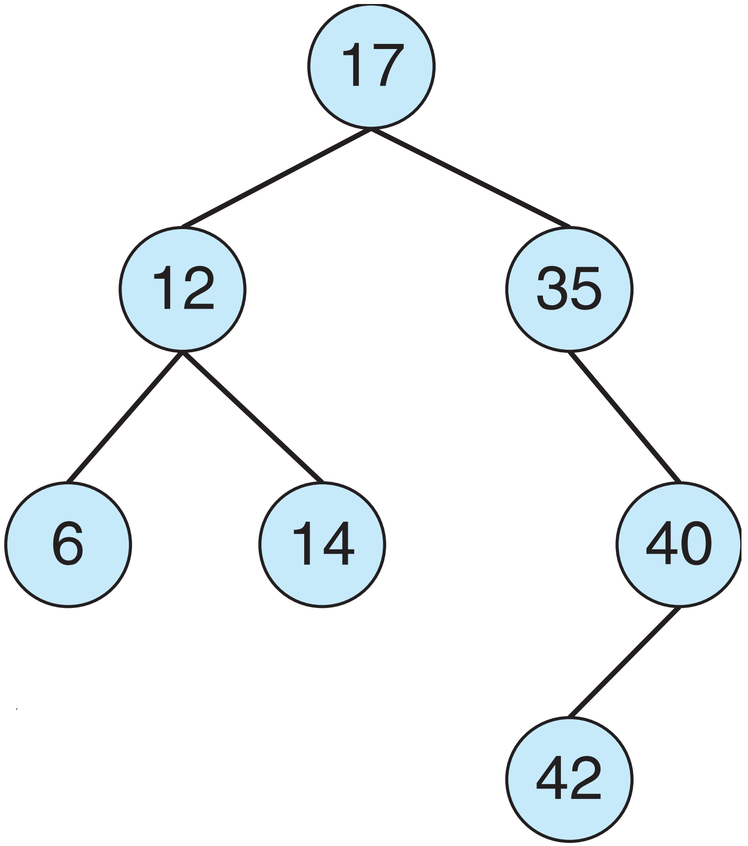


- ***Circular linked list***



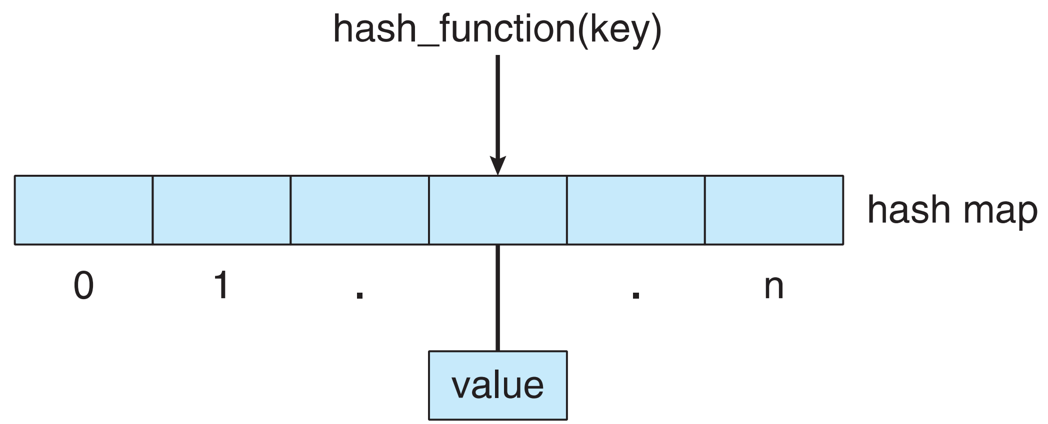
Kernel Data Structures

- **Binary search tree**
  left <= right
  - Search performance is *O(n)*
  - **Balanced binary search tree** is *O(lg n)*



Kernel Data Structures

- **Hash function** can create a **hash map.**



- **Bitmap** – string of *n* binary digits representing the status of *n* items.

- Linux data structures defined in ***include*** files `<linux/list.h>`, `<linux/kfifo.h>`, `<linux/rbtree.h>`

Computing Environments

- Traditional
- Client-Server
- Mobile
- Distributed
- Peer-to-Peer
- Virtualization
- Cloud Computing
- Real-time environments
  - Embedded (e.g., in cars)
  - Stand-alone (e.g., in factories).

Computing Environments - Traditional

- Stand-alone general purpose machines.
- But blurred as most systems interconnect with others (i.e., the Internet).
- **Portals** provide web access to internal systems.
- **Network computers** (**thin clients**) have less power.
- Mobile computers interconnect via **wireless networks.**
- Networking is now ubiquitous – even home systems use **firewalls** to protect computers in the home network from Internet attacks.

Computing Environments - Mobile

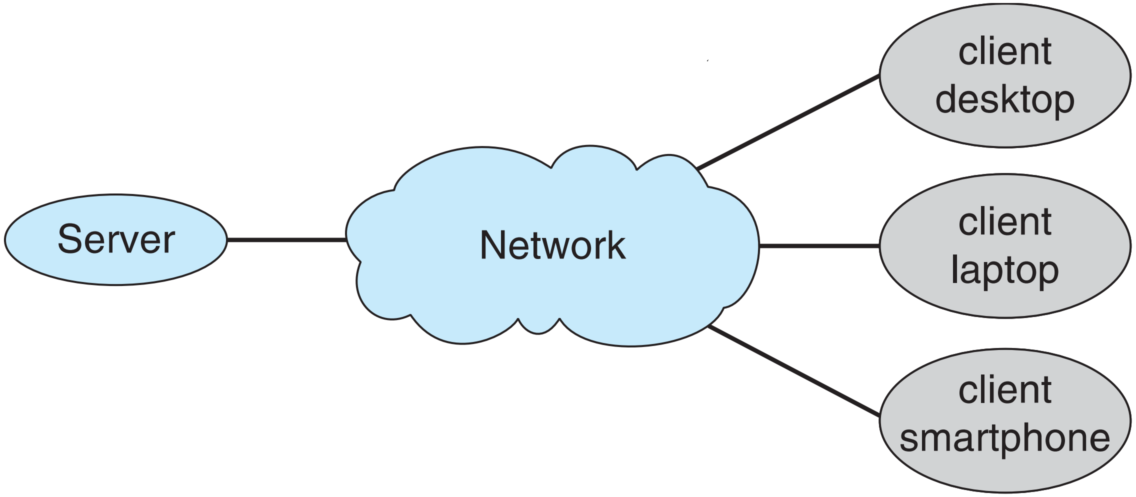
- Handheld smartphones and tablets.
- Functional difference between smartphones/tablets and "traditional" laptops keeps decreasing.
- Extra OS features--e.g., GPS, gyroscope, sensors galore: barometric/fingerprint identity/accelerometer/ibeacon (universally unique identifier)/proximity/ambient light sensors, etc.
- Allow new and innovative types of apps (e.g., ***augmented reality***).
- Use IEEE 802.11 wireless, or cellular data networks for connectivity.
- Only three mobile OS's as of 2015:
  - **Apple iOS**,
  - **Google Android**, and
  - **Microsoft 10** (for Microsoft Surface).

Computing Environments – Distributed

- Distributed
  - Collection of separate, possibly heterogeneous, systems networked together.
    - Network is a communications path, TCP/IP most common.
      - Local Area Network (LAN)
      - Wide Area Network (WAN)
      - Metropolitan Area Network (MAN)-connectivity in a geographic area.
      - Personal Area Network (PAN)
  - Network Operating System (NOS) provides features between systems across network.
    - Communication scheme allows systems to exchange messages.
    - Illusion of a single system.

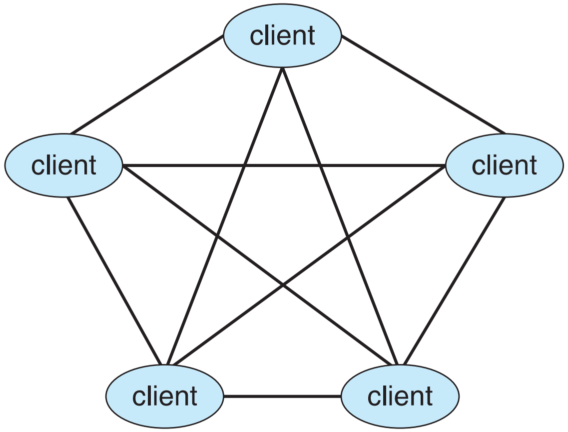
Computing Environments – Client-Server

- Client-Server Computing
  - Clients are PCs, tablets, or smart phones.
  - Many systems now have servers, responding to requests generated by clients
    - Compute-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.



Computing Environments - Peer-to-Peer (P2P)

- Another model of distributed system.



- P2P does not distinguish clients and servers.
  - Instead all nodes are considered peers.
  - Each may 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, **Voice over IP** *(***VoIP***)* such as Skype.

Computing Environments - Virtualization

- Allows operating systems to run applications within other OS's.
  - Vast and growing industry.

- **Emulation** is used when source CPU type differs from target CPU type (i.e., PowerPC to Intel x86).
  - Generally, the slowest method.
  - When the original code is not (or, cannot be) compiled to native code, it is **interpreted.**
  - How does emulation differ from simulation?
    Simulation: Execution (imitation) of an environment or equipment by software modeling.

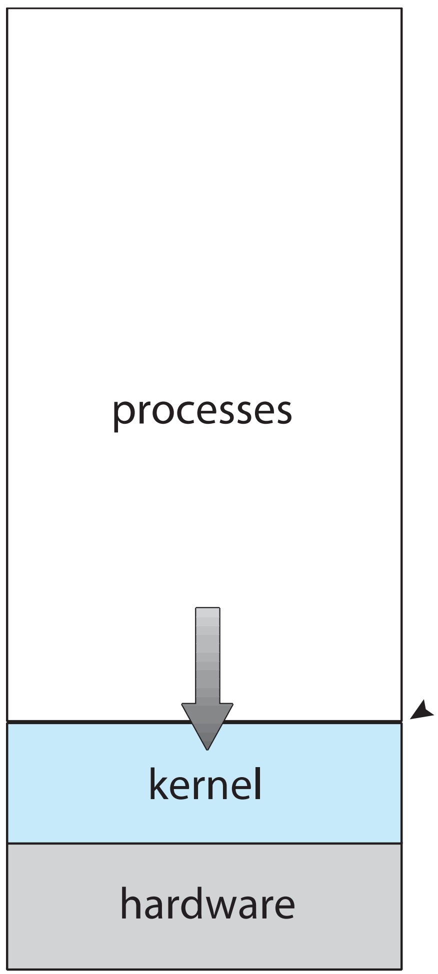
- **Virtualization** – The OS, natively compiled for the given CPU, running **guest** OS's, natively compiled for the same CPU.

Computing Environments - Virtualization

- Laptops and desktops running multiple OS's for exploration or compatibility.
  - Apple laptop running Mac OS X host, Windows as a guest.
  - Developing apps for multiple OS's without having multiple systems.
  - Executing and managing compute environments within data centers.

- VMM (Virtual Machine Manager) can run natively, in which case they are also the host.
  - There may not be a general purpose host.
    - VMware ESX which provides enterprise level computer visualization.
    - Citrix XenServer which provides server virtualization.

Computing Environments - Virtualization



(a)

achine

Computing Environments – Cloud Computing

- Delivers computing, storage, databases and apps as a service across a network.

- Logical extension of virtualization as cloud computing is also based on virtualization.
  - Amazon **EC2** has
    - thousands of servers,
    - millions of VMs,
    - PBs of storage available across the Internet,
    - pay based on usage.

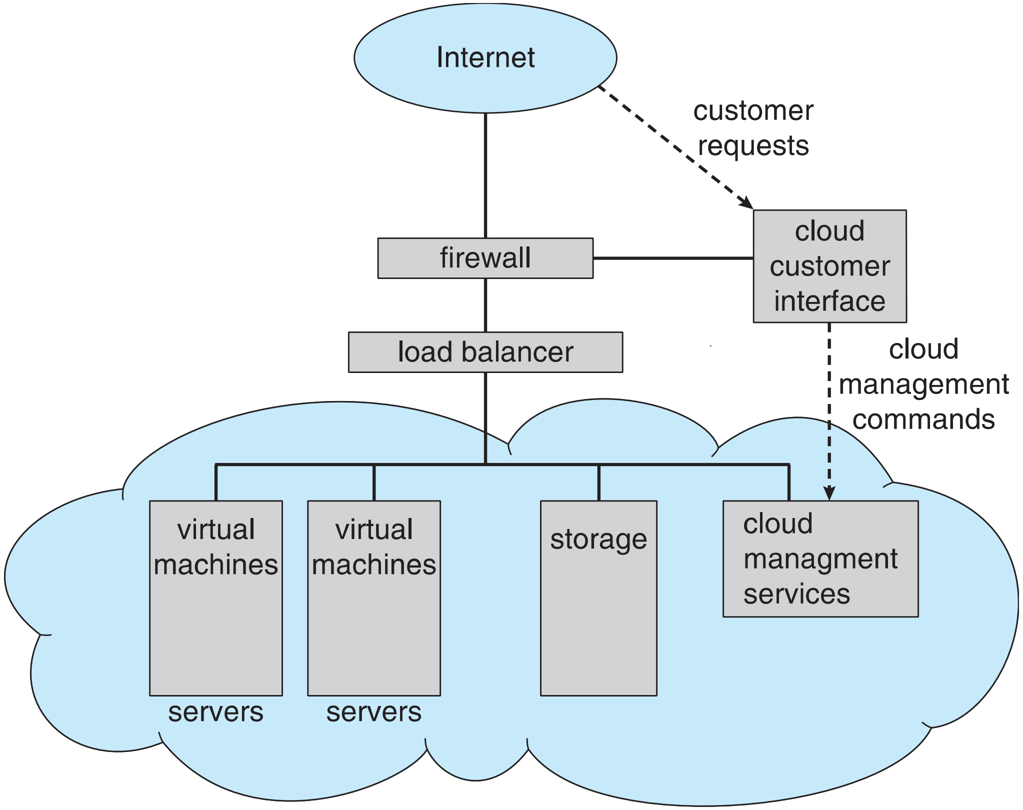
Computing Environments – Cloud Computing

Many types of cloud computing:

- **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 (i.e., a word processor).
- Platform as a Service (**PaaS**) – software stack ready for application use via the Internet (i.e., a database server).
- Infrastructure as a Service (**IaaS**) – servers or storage available over Internet (i.e., storage available for backup use).

Computing Environments – Cloud Computing

- Cloud compute environments are composed of traditional OS's, plus VMMs, plus cloud management tools.
  - Internet connectivity requires security like firewalls.
  - Load balancers spread traffic across multiple applications.



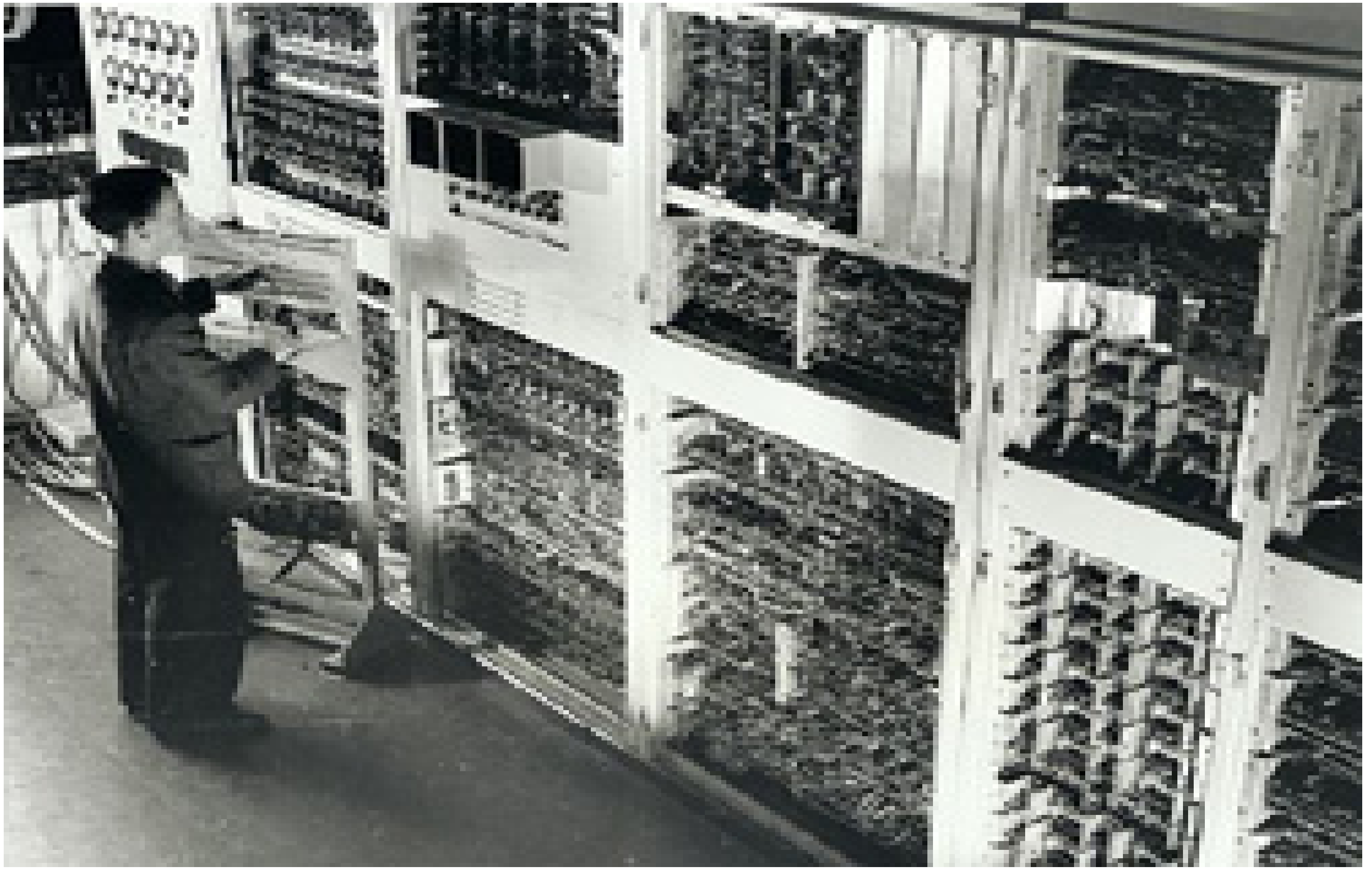
Computing Environments – Real-Time Embedded Systems

- Real-time embedded systems are the most prevalent form of computers.
  - Vary considerably: special purpose, limited purpose OS, **real-time OS.**
  - Specialized use; but their use is expanding: smart homes, smart city, smart cars, driverless (self-driving) cars, …
- Many other special real-time computing environments exist.
  - Some have OS's, some perform tasks without an OS.
- Real-time OS has well-defined fixed (hard/soft) time constraints.
  - Processing ***must*** be done within constraint.
  - Correct operation only if constraints met.

Open-Source Operating Systems

- Some OSs are 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** and **BSD UNIX** (which includes the core of **Mac OS X**), and many more.
- Can use VMM like VMware Player (Free on Windows), Virtualbox (open source and free on many platforms - http://www.virtualbox.com)
  - Used to run guest operating systems for exploration.

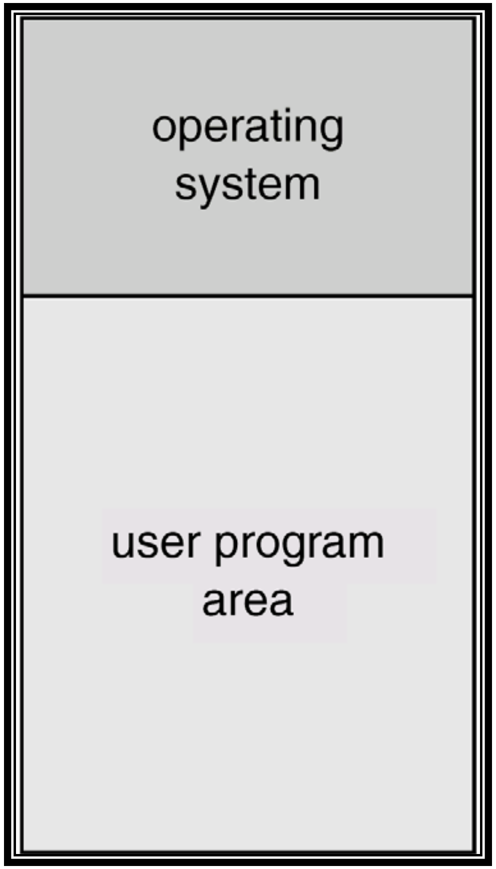
History of Evolution of OSs



Mainframe Systems: Batch Systems 1950's

- First rudimentary operating system.
- Reduce setup time by batching similar jobs
- Automatic job sequencing – automatically transfers control from one job to another.
- Resident monitor – always in memory.
  - initial control in monitor.
  - control transfers to job.
  - when job completes, control transfers back to monitor.

Memory Layout



Mainframe Systems: Time-Sharing Systems 1960’s (Multix, Unix, …)

- The CPU is multiplexed among several jobs that are kept in memory and on disk (the CPU is allocated to a job only if the job is in memory).

- A job is ***swapped in and out of memory*** to the disk – virtual memory management technique is needed.

- On-line communication between the user and the system is provided:
  When the OS finishes the execution of one command, it seeks the next “control statement” from the user’s keyboard.

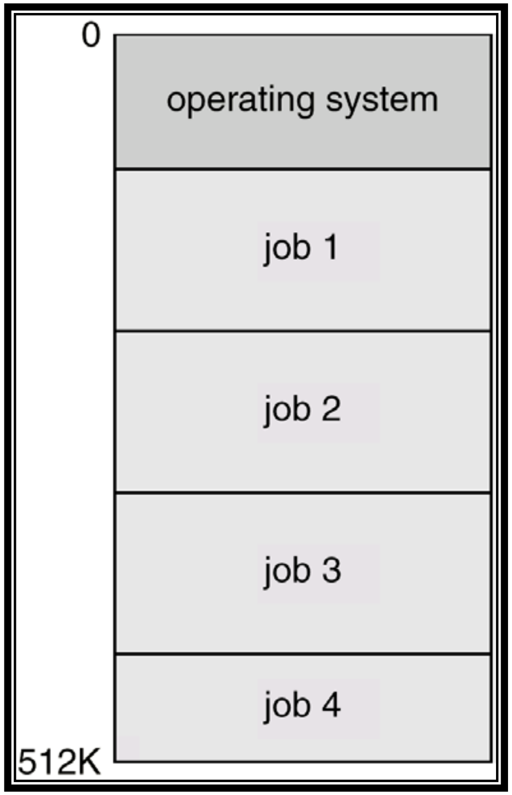
- On-line system must be available for users to access data and code.
  Virtual memory, file system, protection,
  communication and synchronization …

Mainframe Systems: Multiprogrammed Systems-- 1970's

- Several jobs are kept in main memory at the same time, and the CPU is *multiplexed* among them.

- OS features needed for multiprogramming:
  - Memory management – the system must allocate the memory to several jobs concurrently.
  - CPU scheduling – the system must choose among several jobs ready to run.
  - Devices need to be allocated/deallocated to jobs during execution.

Memory Layout



Desktop Systems-- 1980's

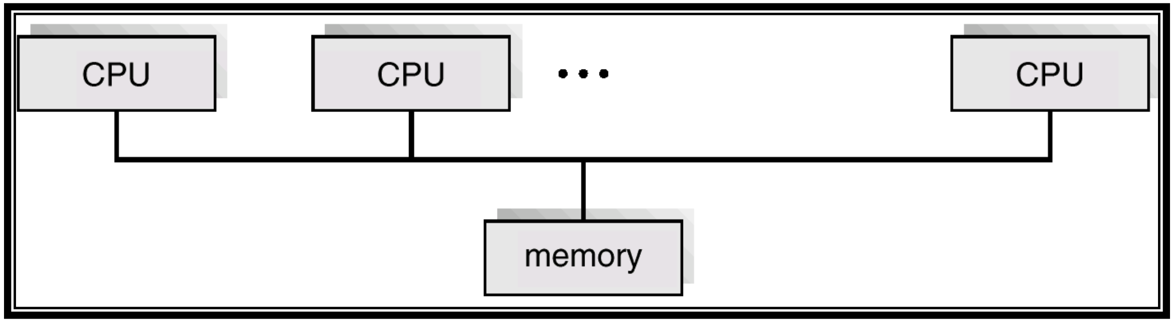
- Personal computers – computer system dedicated to a single user.
  - I/O devices – keyboards, mice, display screens, small printers.
- Goals: user convenience and responsiveness.
  - Adopt technology developed for larger operating systems.
  - Often individuals have sole use of computer and do not need advanced CPU utilization of protection features.
- May run several different types of operating systems (Windows, MacOS, UNIX, Linux).

Multiprocessor Systems (1)-- 1960s and later

- Multiprocessor systems with more than one CPU in close communication.
- ***Tightly coupled system*** – processors share memory and a clock; communication usually takes place through the shared memory.
- Advantages of parallel system:
  - Increased throughput
  - Economical
  - Increased reliability
    - graceful degradation
    - fail-soft systems

Multiprocessor Systems (2)

- ***Symmetric multiprocessing*** *(SMP)*
  - Each processor runs an identical copy of the operating system.
  - Many processes can run at once without performance deterioration.
  - Most modern operating systems support SMP
- ***Asymmetric multiprocessing***
  - Each processor is assigned a specific task; master processor schedules, and allocates work to slave processors.
  - More common in extremely large systems.

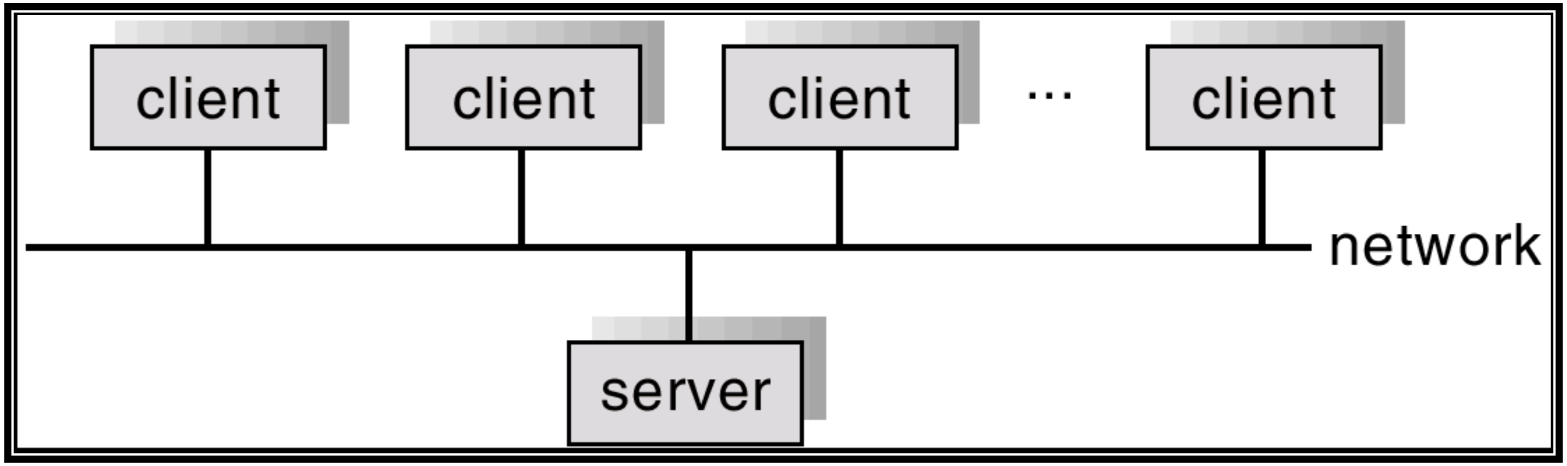


Symmetric Multiprocessing Architecture

Distributed Systems-- 1960's

- ***Loosely coupled system*** – distribute the computation among several computers.
  - Each computer can be autonomous and has its own resources.
  - Communication with network protocol support.
- Advantages of distributed systems:
  - Resource Sharing
  - Computation speed-up – load sharing
  - Reliability
  - Communications
- Requires networking infrastructure.
  - Local Area Networks (LAN) or Wide Area Networks (WAN)
  - May be either client-server or peer-to-peer systems.

Distributed Systems: Client-Server—1970's



General Structure of Client-Server

Clustered Systems—1980’s

- Clustering allows two or more systems to share storage.
- Provides high reliability.
- *Asymmetric clustering*: one server runs the application while other servers standby.
- *Symmetric clustering*: all N hosts are running the application.

Real-Time Systems—1960's

- Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.
- Well-defined **fixed-time constraints**; may be either ***hard*** or ***soft.***
- Hard real-time:
  - Secondary storage limited or absent;
    data stored in short term memory or read-only memory (ROM).
  - Conflicts with time-sharing systems, not supported by general-purpose operating systems.
- Soft real-time
  - Limited utility in industrial control of robotics.
  - Useful in applications (multimedia, virtual reality) requiring advanced operating-system features.

Handheld, Mobile, Integrated Systems—1990s

- Mobile devices:
  - Personal Digital Assistants (PDAs)
  - Cellular Phones
  - Mobile Tablets
- Sensors, TinyOS
- Integration: AppleTV, Google TV, ROKU, …

Migration of Operating-System Concepts and Features