Operating Systems
Chapter 1: Introduction

General Info

- Course : Operating Systems (3 credit hours)
- Instructor : Assoc. Prof. Dr. Marenglen Biba
- Office : Faculty building 2nd floor
- Office Hours : Wednesday 11-13 PM or by appointment
- Phone : 42273056 / ext. 112
- E-mail : marenglenbiba@unyt.edu.al
- Course page : http://www.marenglenbiba.net/opsys/

- Use of E-mail: Always put “Operating Systems” in the subject of your e-mail.

Where, when and why?

- **Course Location and Time**
- Laboratory Room 4B, Tuesday 16-19.
- **Catalog Description**
- This module covers the core concepts of modern operating systems, and provides contextual application of theory, using examples of currently used operating system environments.
- **Course Purpose**

  This course will provide an introduction to operating system design and implementation. The operating system provides an efficient interface between user programs and the hardware of the computer on which they run. The operating system is responsible for allowing resources (such as processors, disks or networks) to be shared, providing common services needed by many different programs (e.g., file service, the ability to start or stop processes, and access to the printer), and protecting individual programs from one another.

What does the OS course contain?

- The course will start with an historical perspective of the evolution of operating systems since their birth. Then it will cover the major components of most operating systems and the tradeoffs that can be made between performance and functionality during the design and implementation of an operating system. Particular emphasis will be given to three major OS subsystems:
  - **process management** (processes, threads, CPU scheduling, synchronization, and deadlock),
  - **memory management** (segmentation, paging, swapping)
  - **storage management** (file systems, disk management, I/O operations).

Why bother with OS?

- Understand the design and implementation issues that have led to the current modern operating systems.
- Understand and apply key concepts for process management in modern operating systems.
- Understand and apply essential concepts for memory management in modern operating systems.
- Understand and apply important concepts of storage management in modern operating systems.
- Understand and compare different operating systems in order to be able to select them in different use scenarios.
- Understand and apply essential concepts for increasing the performance of modern operating systems.

Requisites and Readings

- **Course Prerequisites**
  - Data Structures.
- **Required Readings**
  - Silberschatz, Abraham, Galvin, Peter and Gagne, Greg, (2012). *Operating System Concepts, Ninth edition*, New York, NY: John Wiley & Sons. (**required**).
  - Andrew Tanenbaum, Modern Operating Systems, Prentice Hall. *Second Edition.* (**only specific sections** of the book will be required for special topics).

Contents

- Introduction to Operating Systems
- Operating System Structure
- Processes
- Threads
- CPU Scheduling
- Process Synchronization
- Deadlocks
- Main Memory
- Virtual Memory
- File System Interface
- File System Implementation
- Mass-Storage Systems
- I/O Systems

Grading

Project	40%
Midterm	30%
Final	30%

- Internet use is necessary since students should regularly check the course home page. Material can be downloaded from course website!
- Continued and regular use of e-mail is expected
- Students must keep copies of all assignments and projects sent by e-mail.

Before we start: why failure happens

- Reasons
  - Lack of concentration?
  - Lack of continuity?
  - Lack of determination?
  - Lack of target?
  - Lack of work?
  - …
- Response
  - Hard work will help!!!

Recommendations

- Start studying now
- Do not be shy! Ask any questions that you might have. Every questions makes you a good candidate.
- The professor is a container of knowledge and the goal is to get most of him, thus come and talk.
- Respect the deadlines
- Respect the appointments
- Try to study from more than one source, Internet is great!
- If you have any problems come and talk with me in advance so that we can find an appropriate solution

GOOD LUCK!

Chapter 1: Introduction

- **What Operating Systems Do**
- History of Operating Systems
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management
- Protection and Security
- Kernel Data Structures
- 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

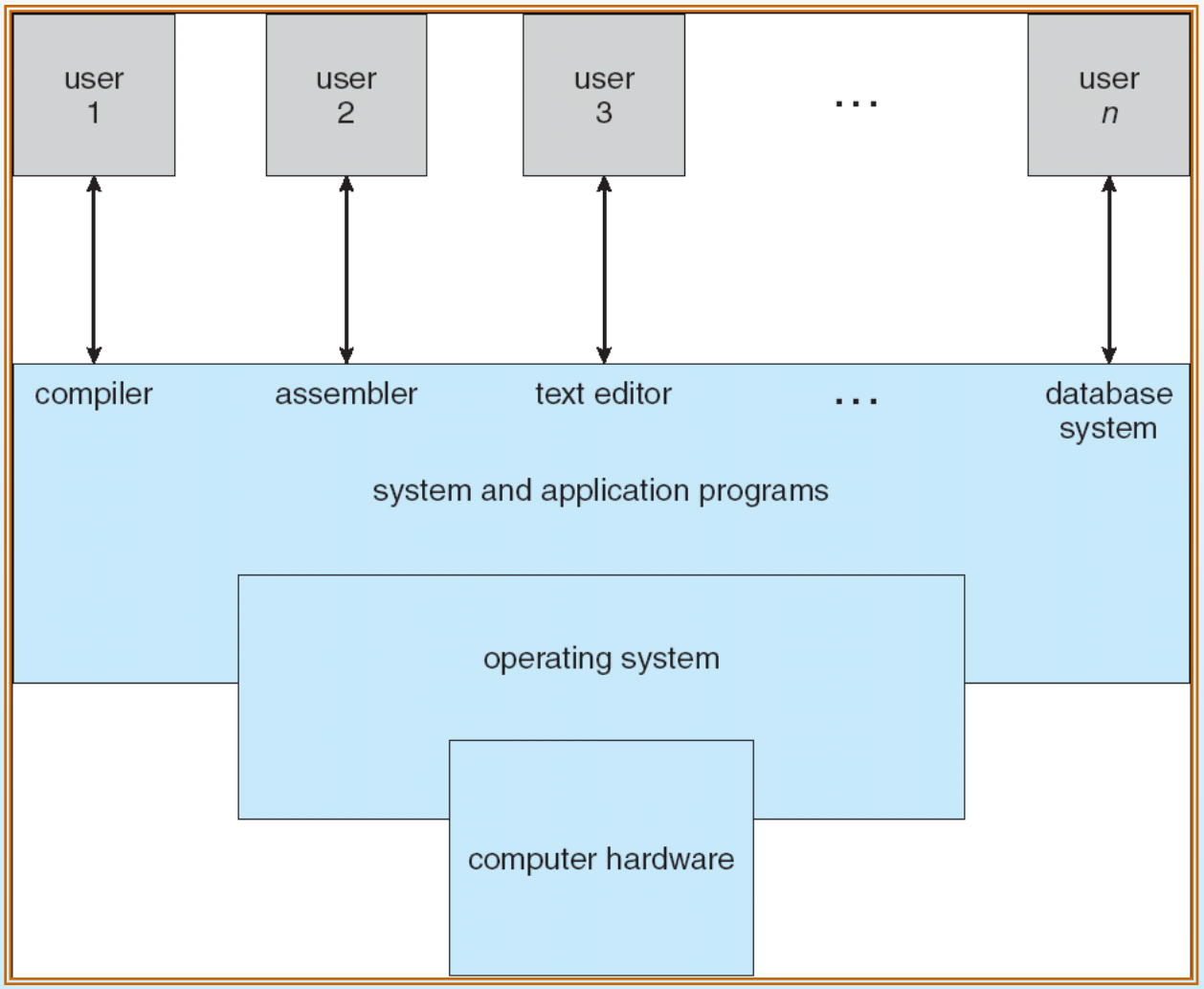
What is an Operating System?

- 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.

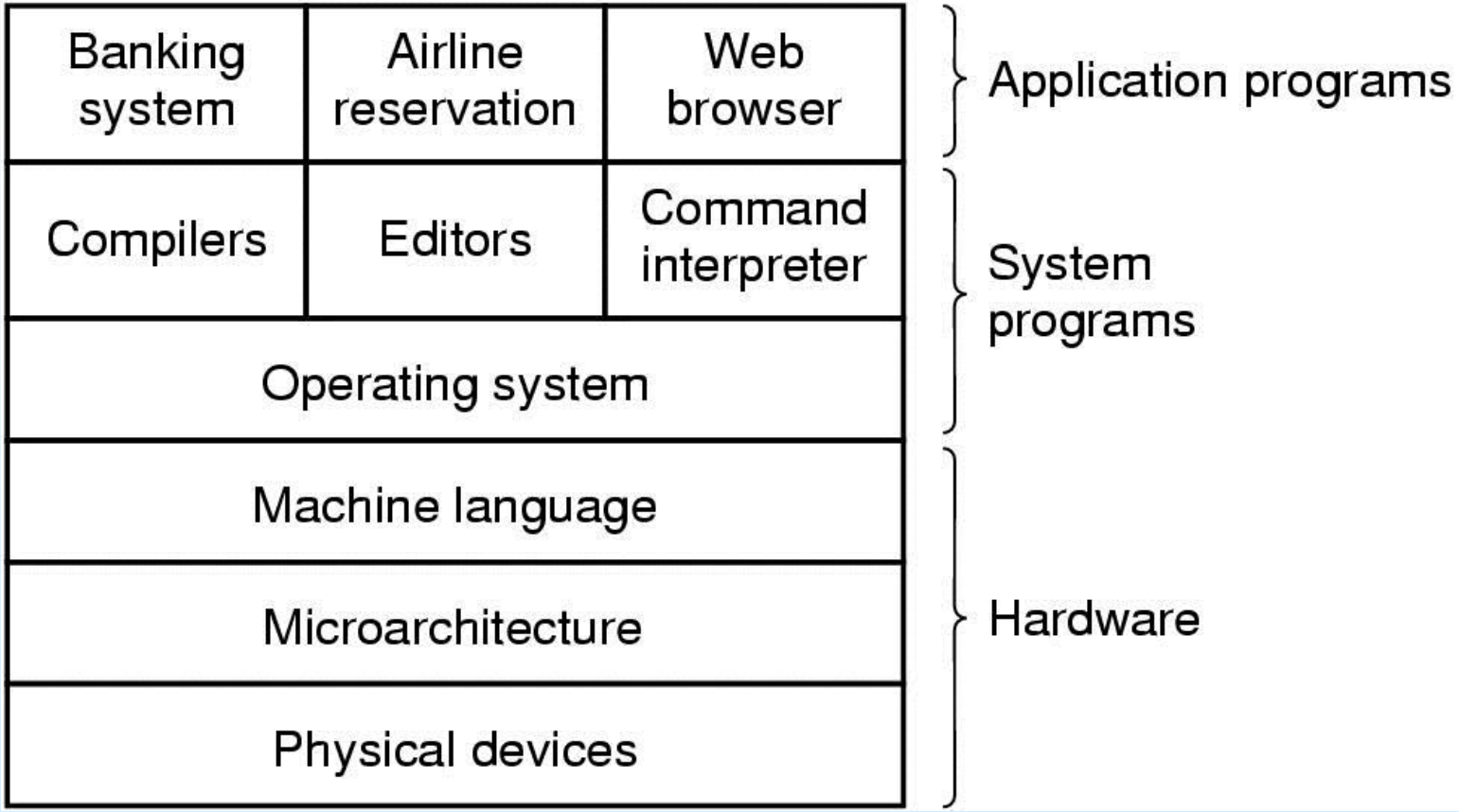
Computer System Structure

- 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

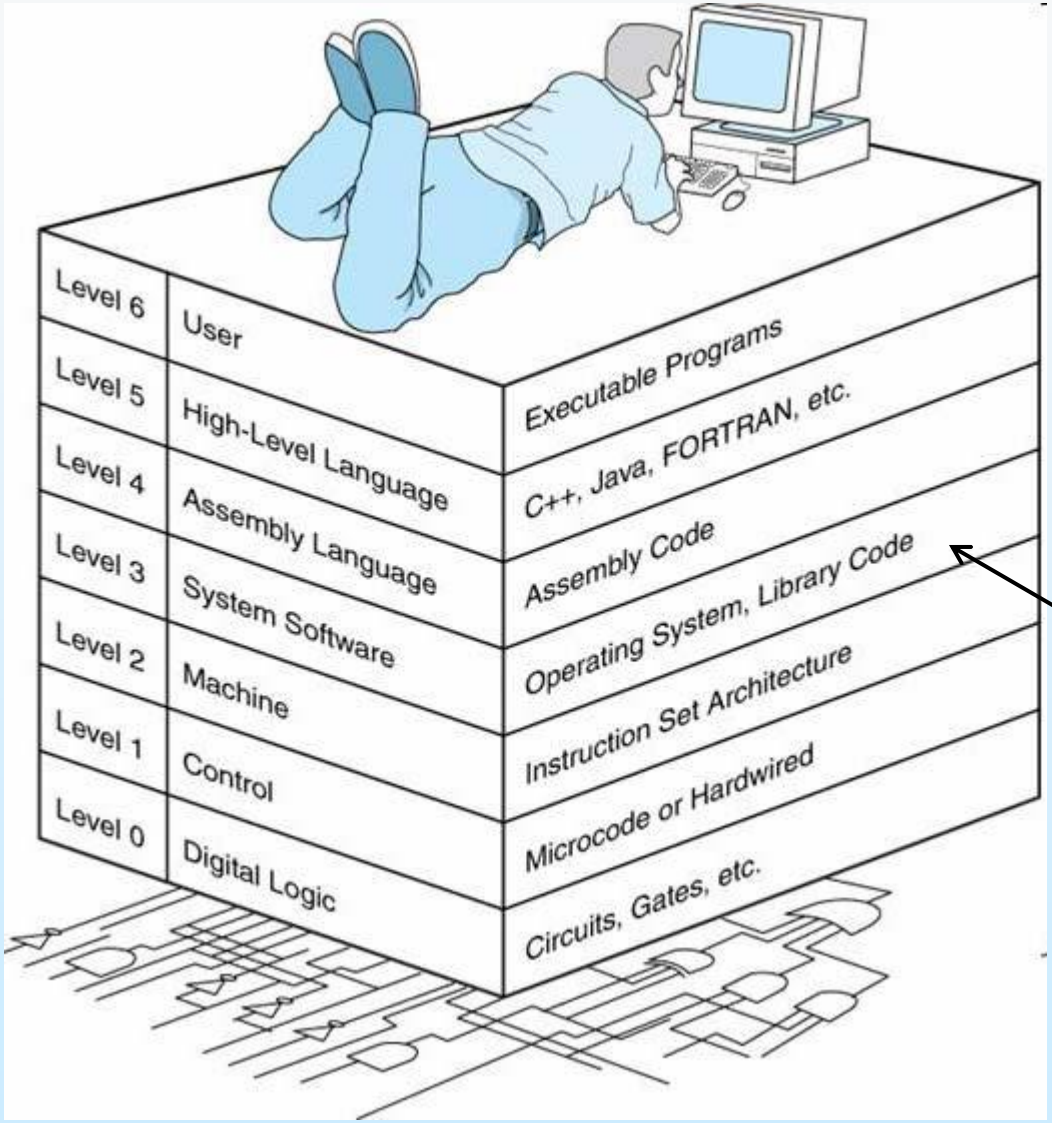
Four Components of a Computer System



Computer components hierarchy



Wish you were here! ☺



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

Operating System Definition (Cont.)

- 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

What is an Operating System

- It is an **extended machine**
  - Hides the messy details which must be performed
  - Presents user with a virtual machine, easier to use
- It is a **resource manager**
  - Each program gets time with the resource
  - Each program gets space on the resource

History of Operating Systems

- First generation 1945 - 1955
  - vacuum tubes, plug boards
- Second generation 1955 - 1965
  - transistors, batch systems
- Third generation 1965 – 1980
  - ICs and multiprogramming
- Fourth generation 1980 – present
  - personal computers

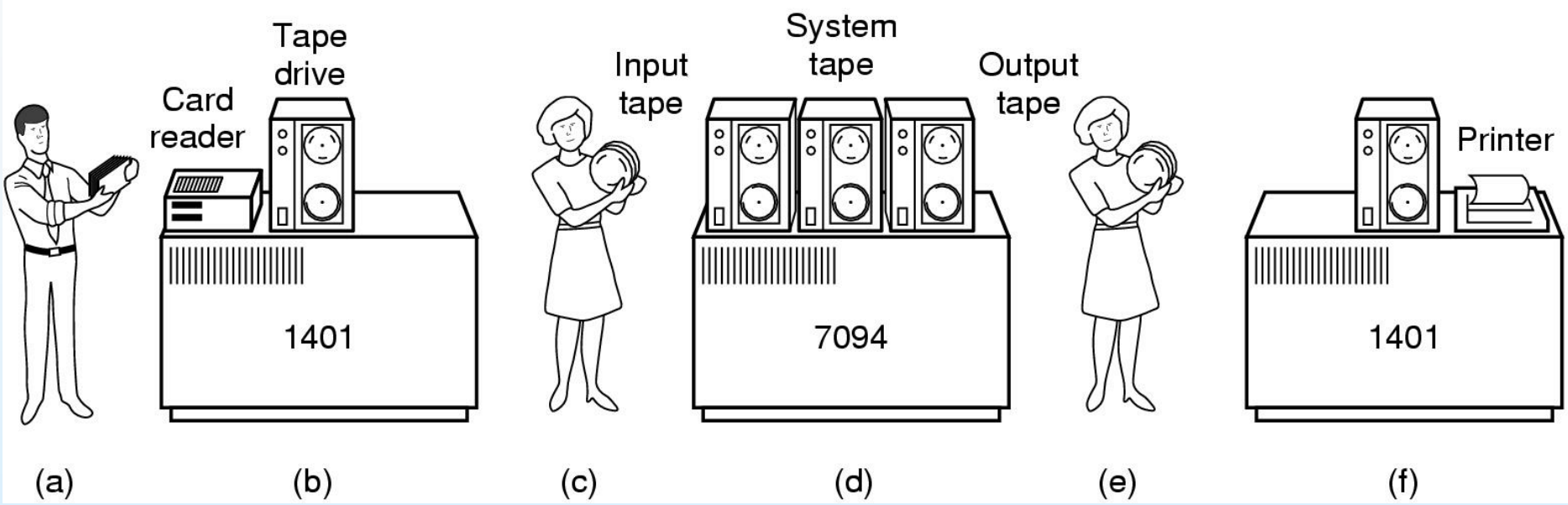
First generation 1945 - 1955

- Not really Operating Systems
- Howard Aiken and John Von Neumann at Institute for Advanced Study Princeton
- J. Eckert and William Mauchley at University of Pennsylvania
- Vacuum Tubes, plug boards
  - Computers were used for calculations and all programming was done in **MACHINE LANGUAGE.**
  - Machine basic functions were controlled through **plugboards**.

Second generation 1955 - 1965

- Introduction of **transistors**
- Programs were first written on paper in the FORTRAN language then they were translated into **punched cards**.
- After the program had finished, a **human operator** would take the result and take it into the output room.
- **Batch system**
  - A collection of jobs given in input
  - IBM 1401: read cards, copy tapes, print output
- Large 2nd generation computers with operating systems
  - Programmed in Assembly and Fortran
  - FMS: Fortran Monitor System
  - IBSYS: IBM operating system for 7094.

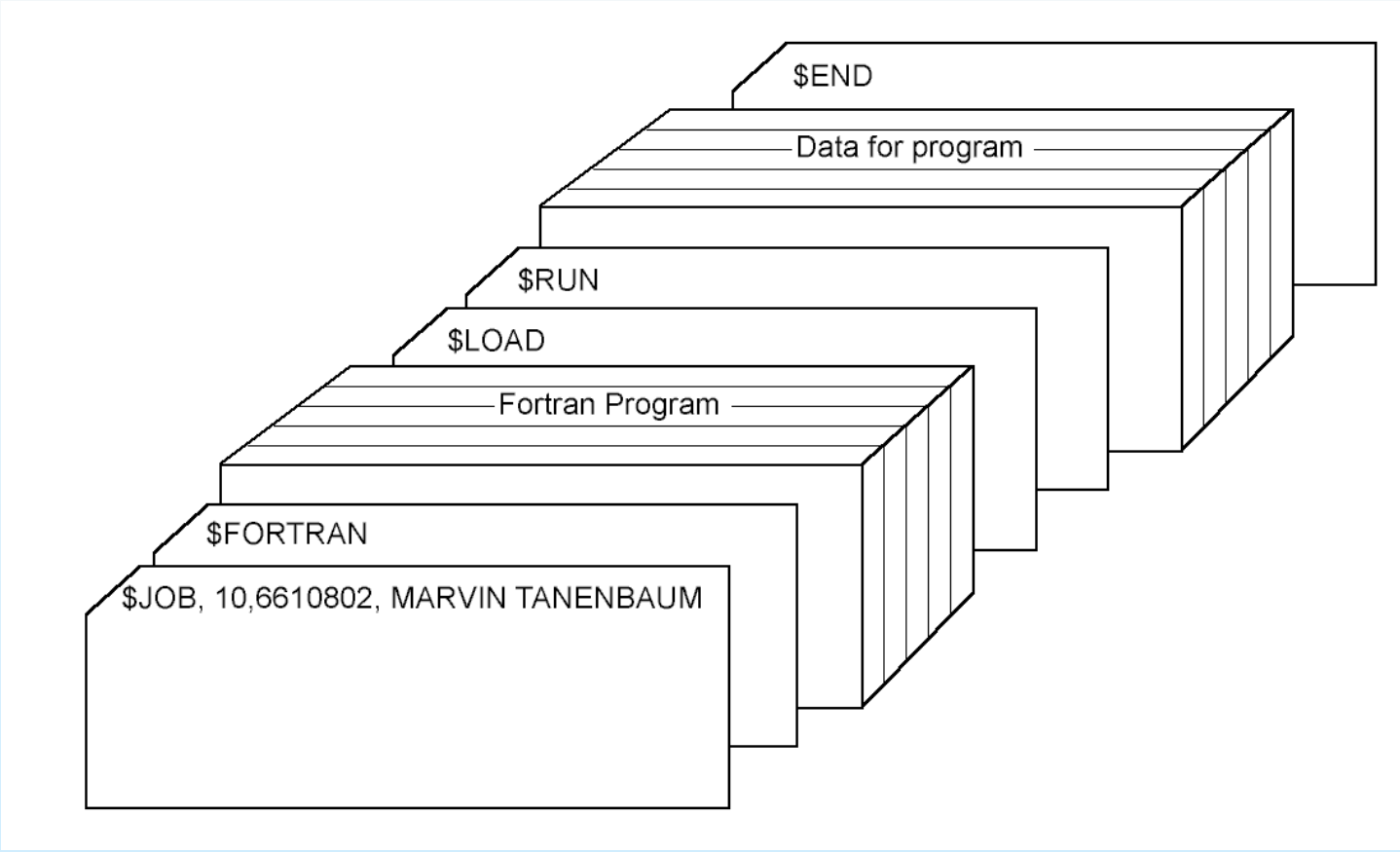
History of Operating Systems (1)



Early batch system

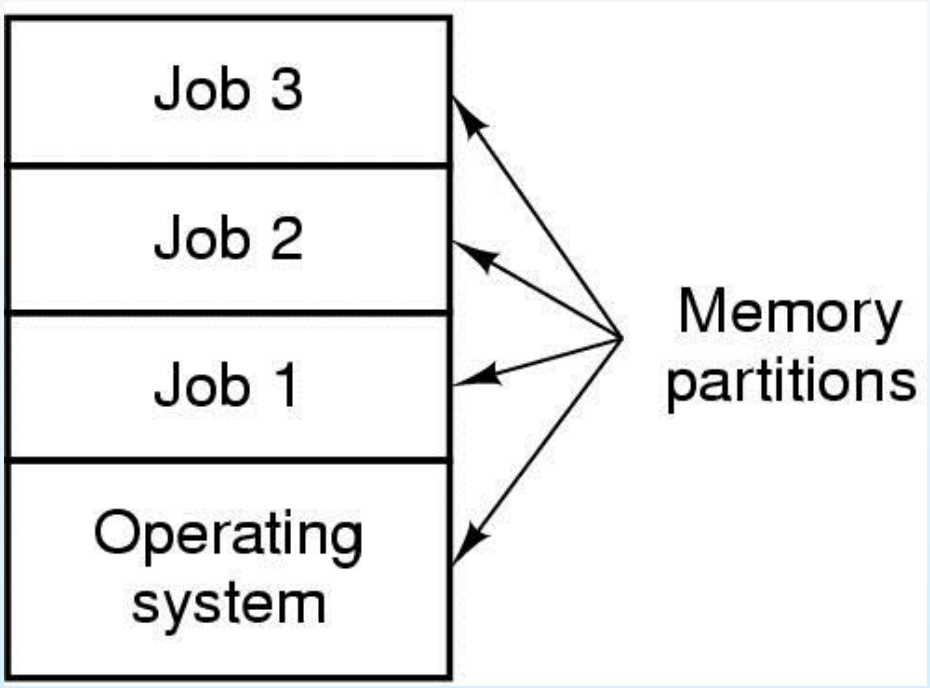
- bring cards to 1401
- read cards to tape
- put tape on 7094 which does computing
- put tape on 1401 which prints output

History of Operating Systems (3)



- Structure of a typical FMS job – 2nd generation

History of OS: 3rd generation 1965 – 1980



- Multiprogramming system
  - three jobs in memory – 3rd generation

3rd generation 1965 – 1980

- Integrated Circuits
- IBM: OS/360
  - Weakness: all software including the OS would run on all models
  - Millions of lines of code written by hundreds of programmers
- **Spooling**
  - Copy jobs from cards onto disk
  - Whenever a running job finishes, load a new one
- **Time-sharing systems**
  - CPU allocation in turn to different jobs
  - CTSS: Compatible Time Sharing System
  - Developed at M.I.T on a specially modified 7094.

3rd generation 1965 – 1980

- Multics: MULTIplexed Information and Computing Service
  - Written in PL/I
  - Introduced seminal idea into the computer literature
- DEC PDP-1
  - Only 4k of 18-bit words
  - 120.000$
  - Culminating in PDP-11
- **Unix**
  - Ken Thompson, wrote from PDP-7 a one-user version of MULTICS.
  - BSD: Berkeley Software Distribution
  - System V: AT&T.
  - Posix, Minix, Linux.

Fourth generation 1980 – present

- CP/M (Control Program for Microcomputers) 1974
  - **Disk-based** operating system
  - To run on 8-bit Intel 8080
  - Digital Research rewrote CP/M and for 5 years it was the most used system in the world
- 1980s
  - IBM released IBM Personal Computer
  - DOS: Disk Operating System
    - Bill Gates bought it from Seattle Computer Products ($50.000)
    - Package DOS/Basic was offered by Gates to IBM
    - IBM wanted some modifications on the system
    - Microsoft's hired programmer Tim Paterson who wrote DOS
    - MS-DOS

Fourth generation 1980 – present

- Apple Macintosh
  - GUI: Graphical User Interface
- Microsoft Windows: 90s
  - Initially run over DOS
  - Not really a different OS
- Windows 95
  - Underlying DOS: only for booting and running old DOS programs.
  - Windows 98
  - Both W95 and Win98 retain large portions of 16-bit assembly language.
- Windows NT (New Technology)
  - Full 32-bit system
  - Would kill off DOS: Win NT 4.0
  - Win NT 4.0 was renamed to Windows 2000.

Fourth generation 1980 – present

- UNIX
  - Best for workstations, high-end computers, network servers
  - Popular on machines with high-performance RISC chips
  - Linux is also going strong on Intel machines
- X Windows
  - Graphical User Interface for UNIX developed at M.I.T.
- Distributed Operating Systems
- Network Operating Systems

The Operating System Zoo

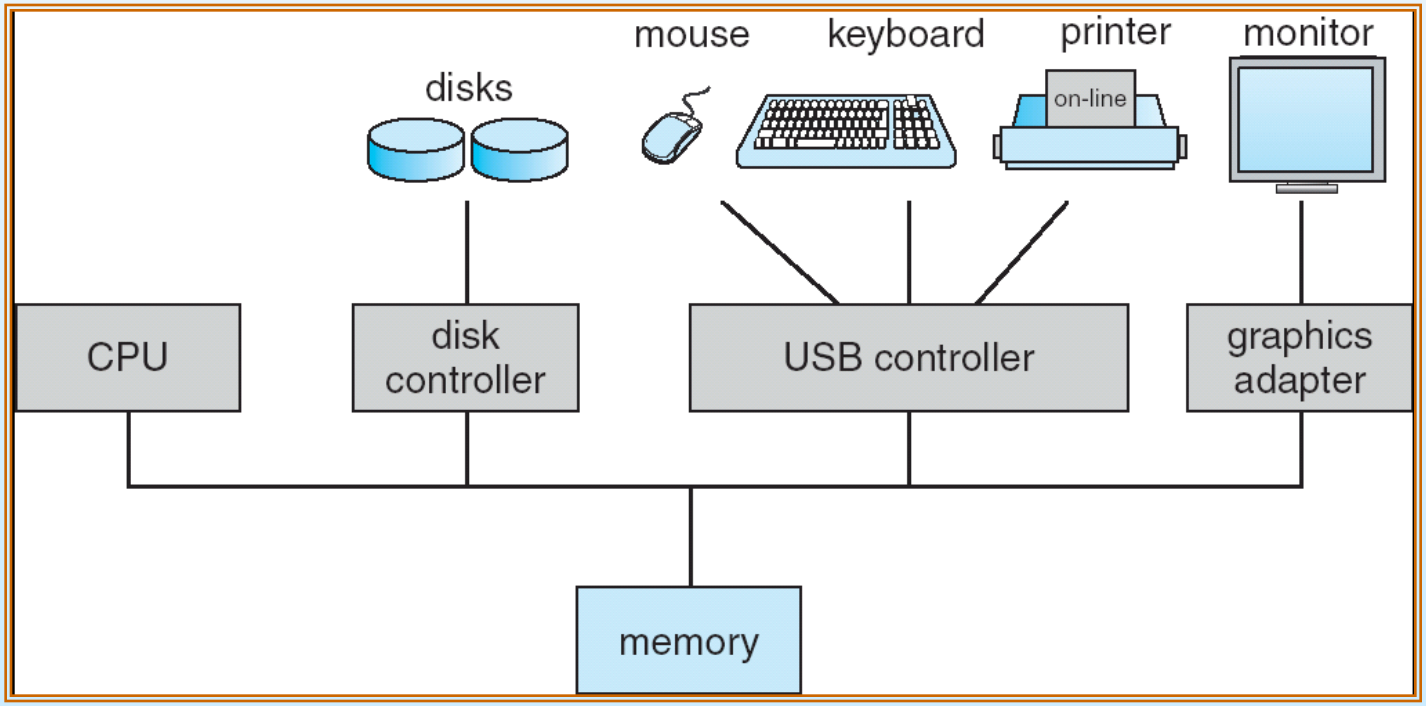
- Mainframe operating systems
- Server operating systems
- Multiprocessor operating systems
- Personal computer operating systems
- Real-time operating systems
- Embedded operating systems
- Smart card operating systems

Computer Startup

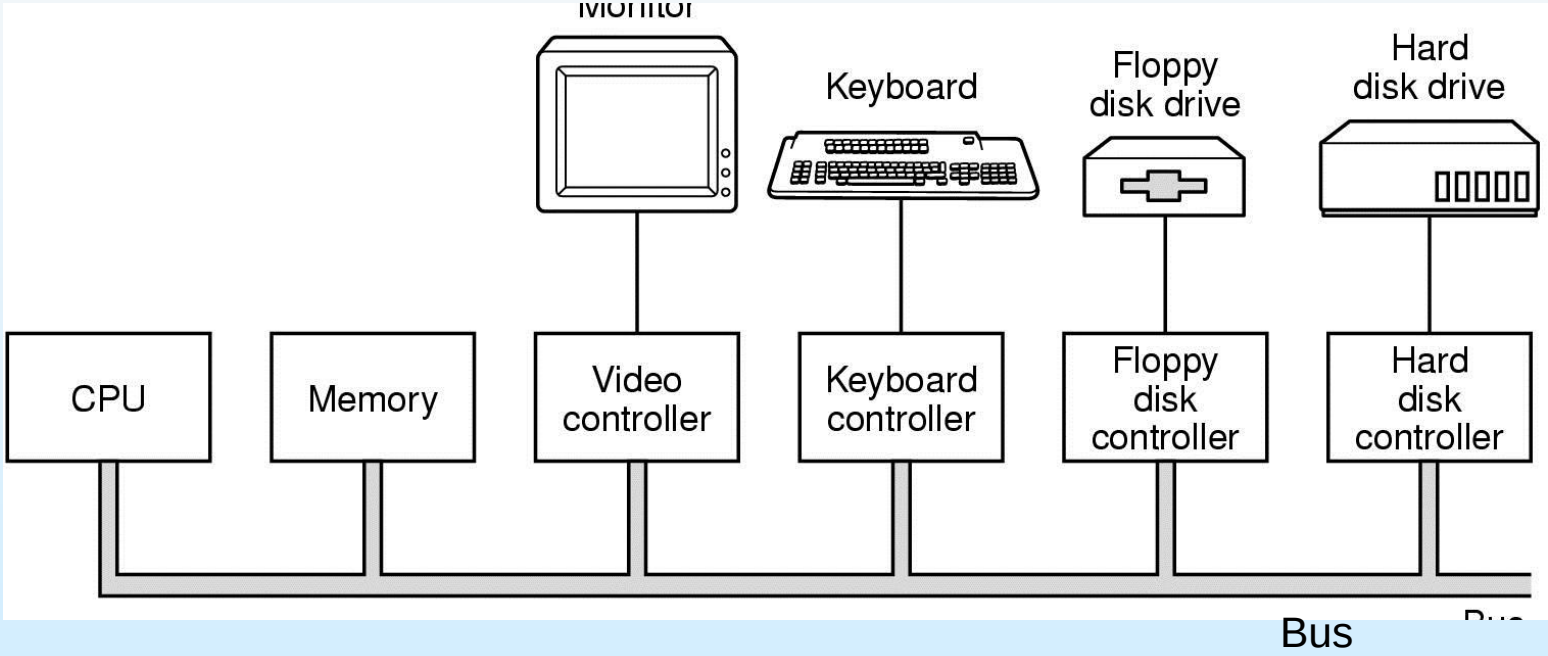
- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

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



Bus


Monitor
Keyboard
Floppy disk drive
Hard disk drive
CPU
Memory
Video controller
Keyboard controller
Floppy disk controller
Hard disk controller
Bus

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***.

Common Functions of Interrupts

- 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*.
- Interrupts can be software or hardware generated.
- A ***trap*** is a software-generated interrupt caused either by an error or a user request.
- Software may trigger an interrupt by executing a special operation called a ***system call***.
- An operating system is ***interrupt*** driven.

Interrupt Handling

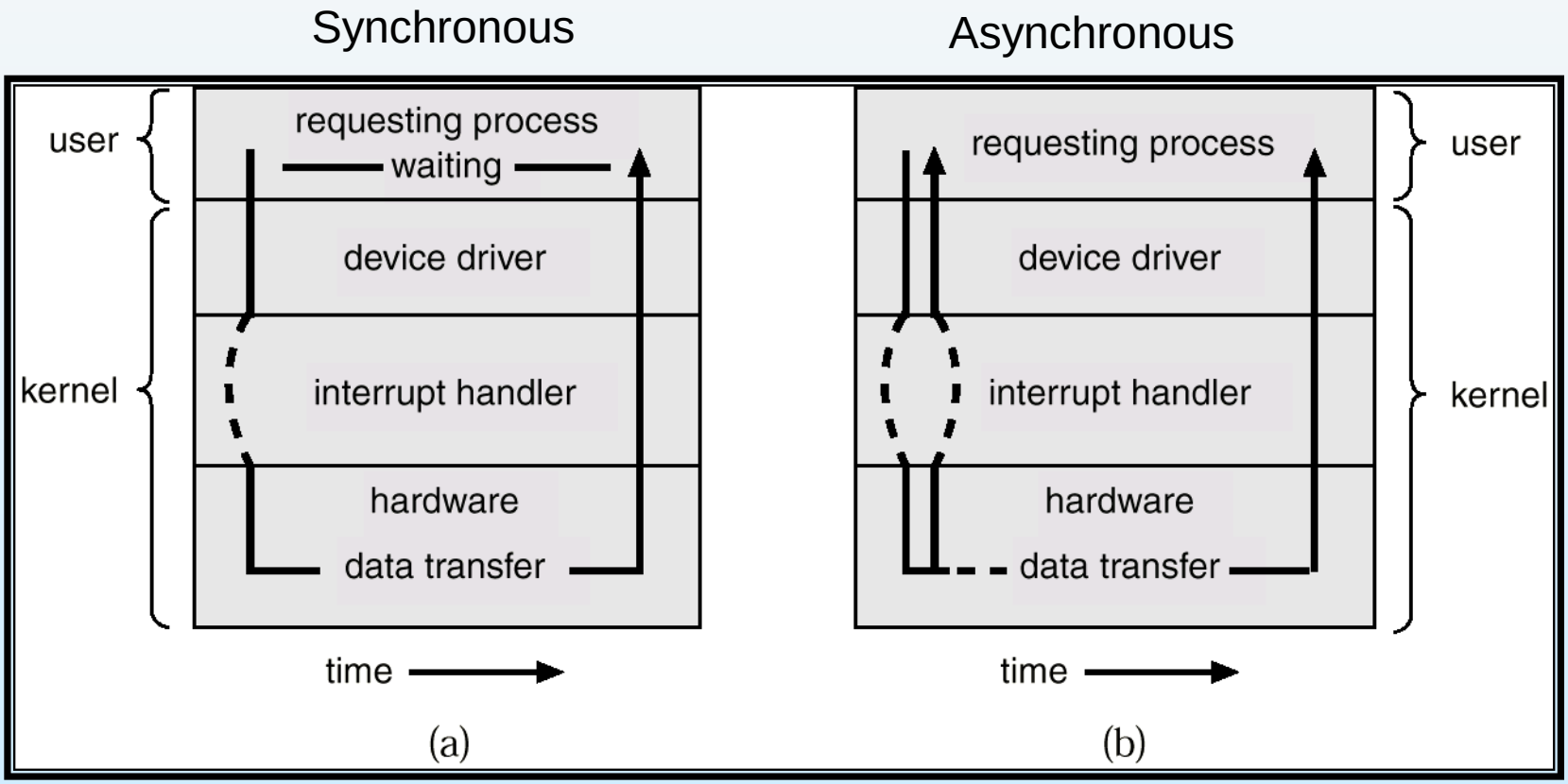
- The operating system **preserves the state** of the CPU by storing registers and the program counter.
- Determines which type of interrupt has occurred:
  - ***Polling****: a polled interrupt is a specific type of I/O interrupt that notifies the part of the computer containing the I/O interface that* ***a device is ready*** *to be read or otherwise handled but* ***does not indicate which device****. The interrupt controller must poll (send a signal out to) each device to determine which one made the request.*
  - **vectored interrupt system**: The alternative to a polled interrupt is a *vectored interrupt*, an interrupt signal that includes the identity of the device sending the interrupt signal.
- **Separate segments of code** determine what action should be taken for each type of interrupt

Two I/O Methods

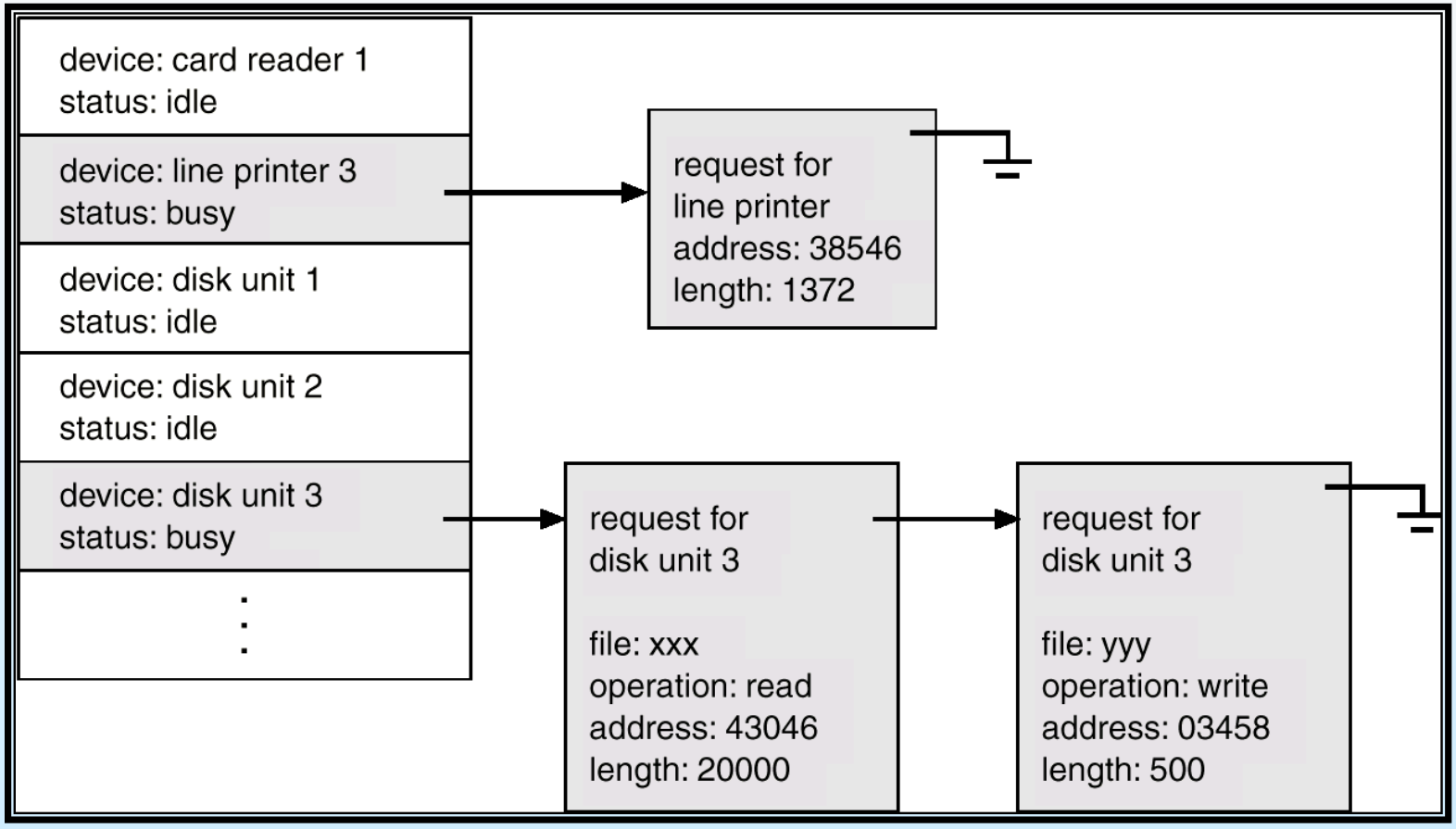
1. After I/O starts, control returns to user program **only upon** I/O completion: **synchronous**

2. After I/O starts, control returns to user program **without waiting** for I/O completion: **asynchronous**

Two I/O Methods



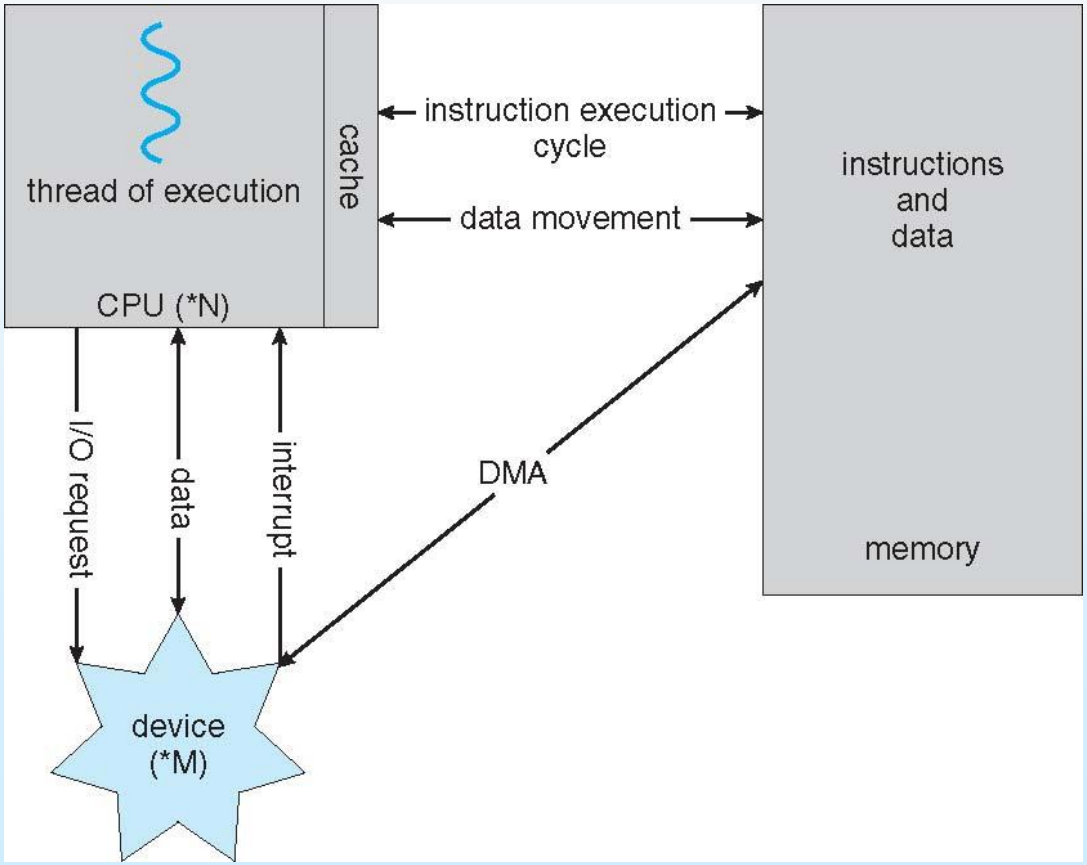
Device-Status Table



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.

DMA


thread of execution
CPU (*N)
cache
instruction execution cycle
data movement
instructions and data
memory
I/O request
data
interrupt
DMA
device (*M)

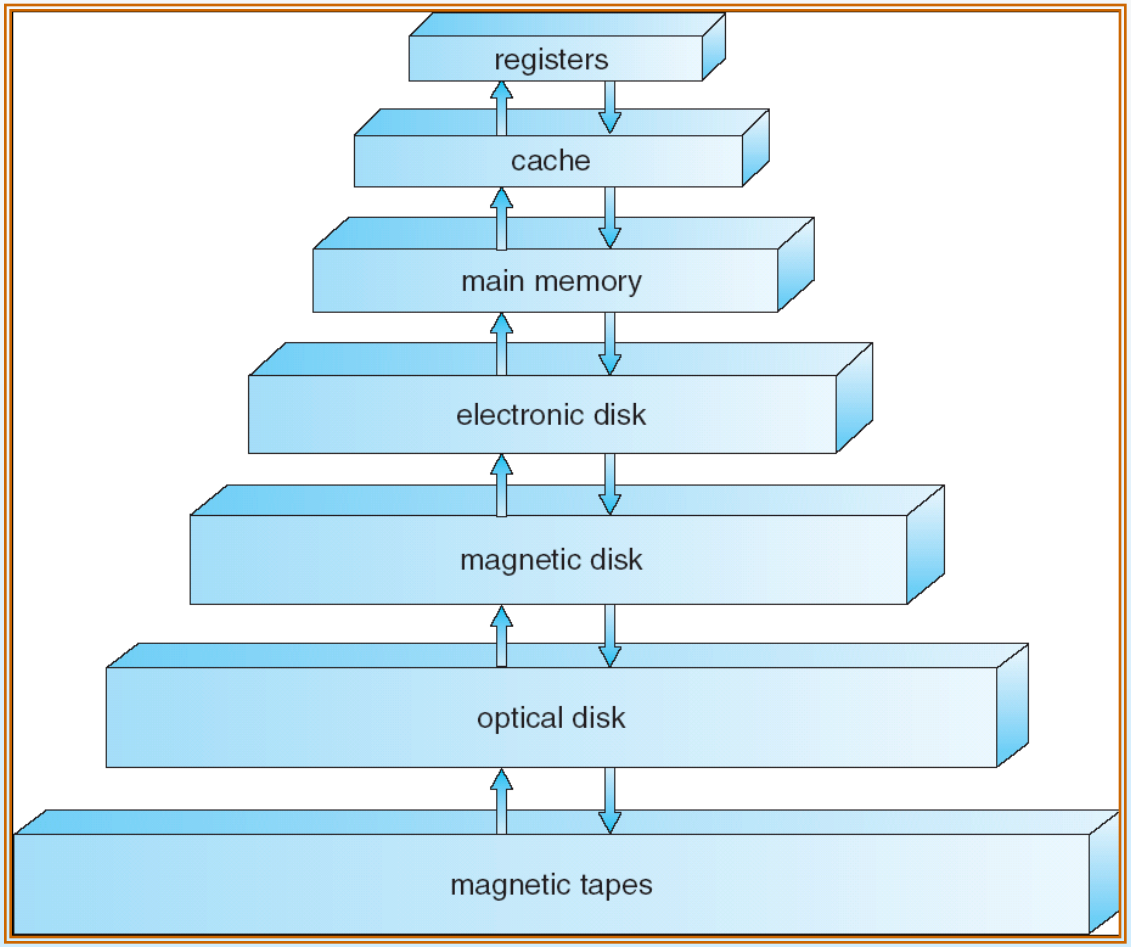
Storage Structure

- **Main memory** – only large storage media that the CPU can access directly.
- **Secondary storage** – 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.

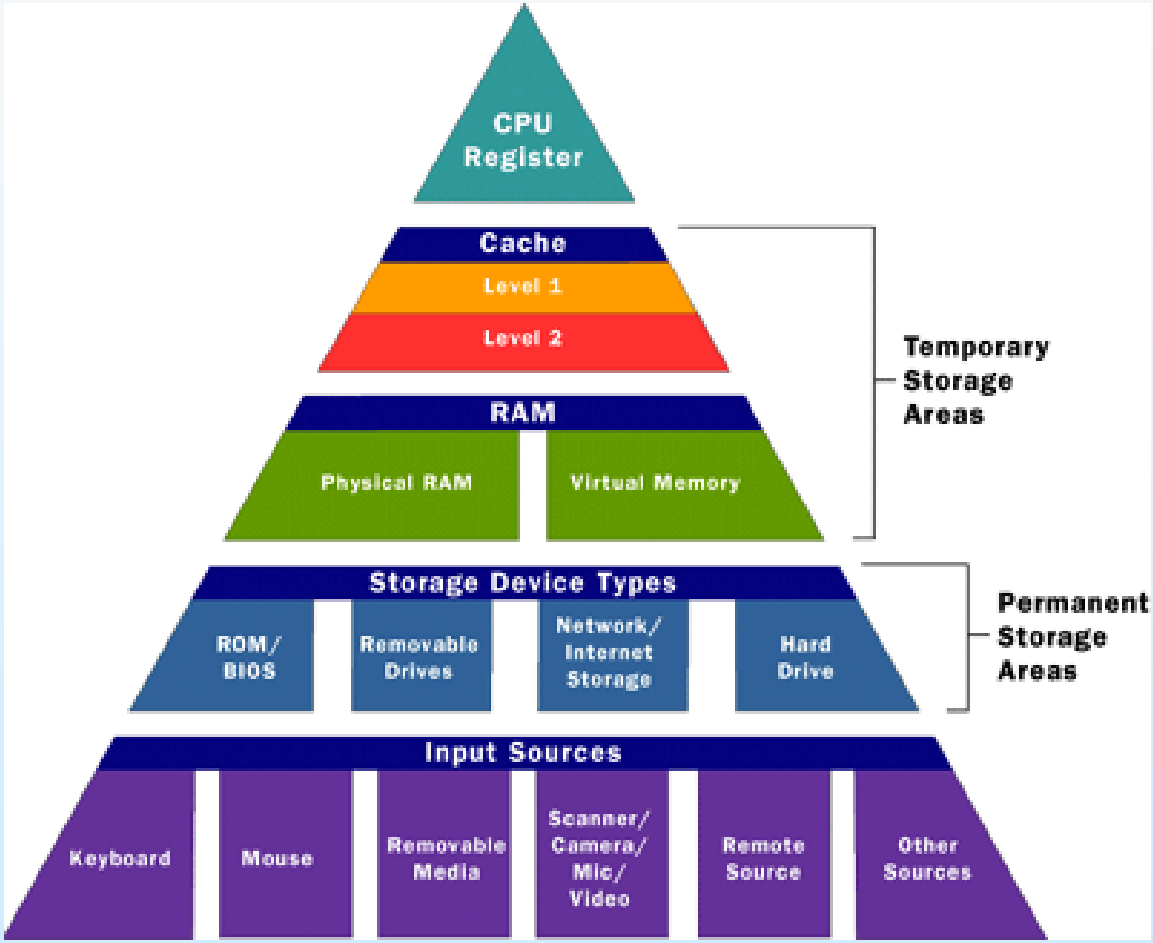
Storage Hierarchy

- 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.

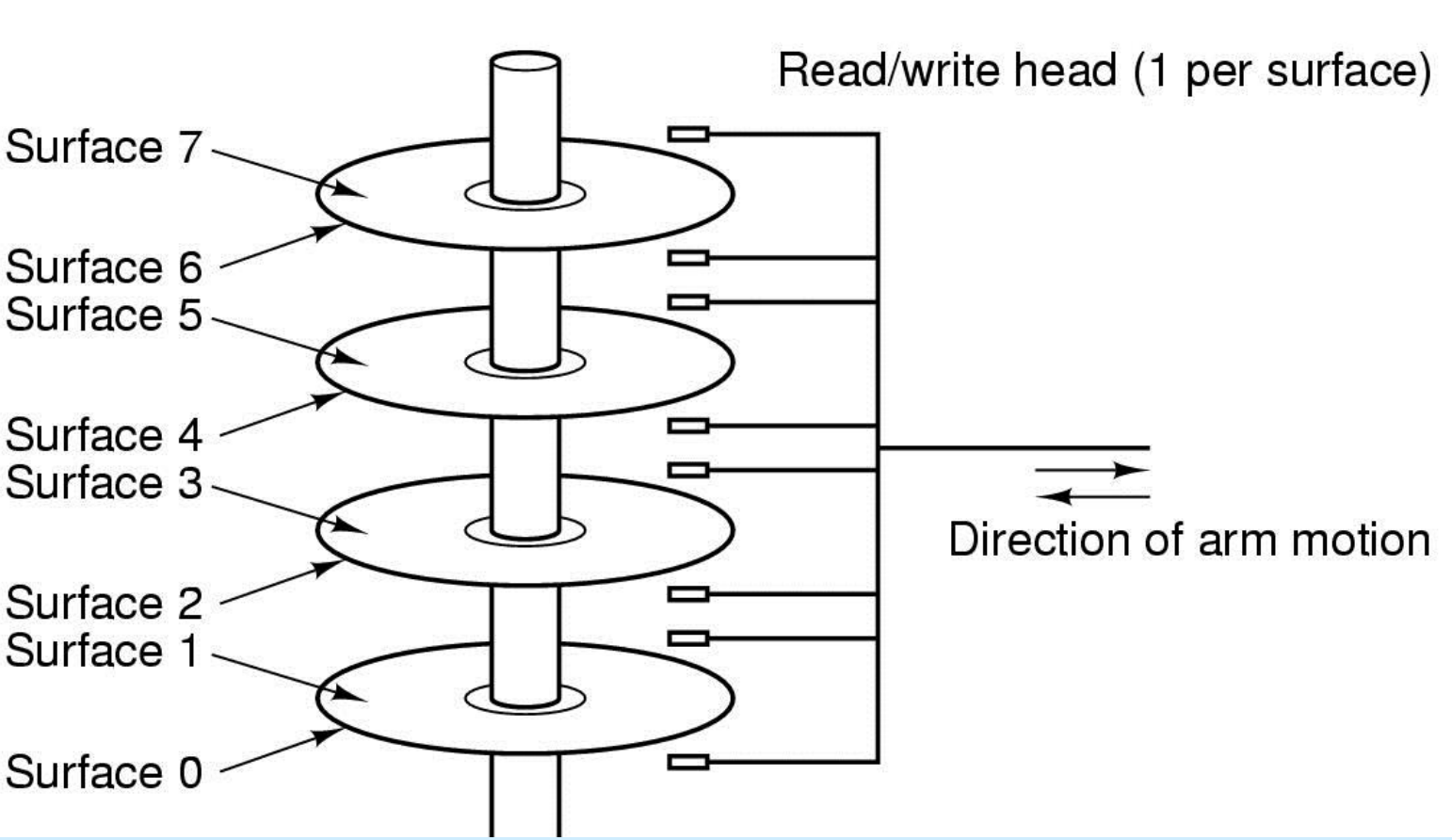
Storage-Device Hierarchy



Sub-levels within each level



Disk is slow


Read/write head (1 per surface)
Surface 7
Surface 6
Surface 5
Surface 4
Surface 3
Surface 2
Surface 1
Surface 0
Direction of arm motion

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

Performance of Various Levels of Storage

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

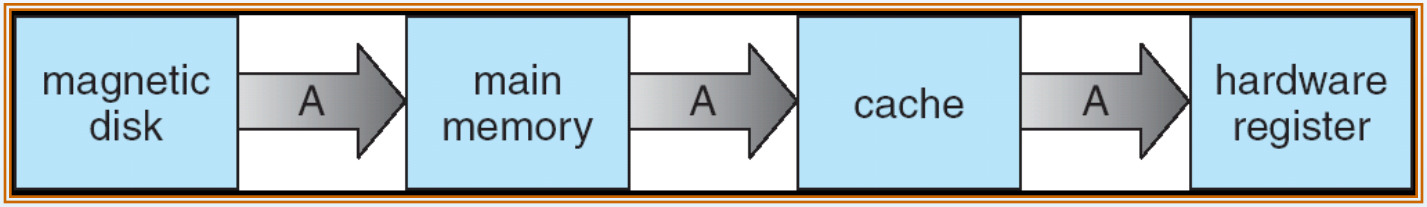
Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 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

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)** 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**
  - 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

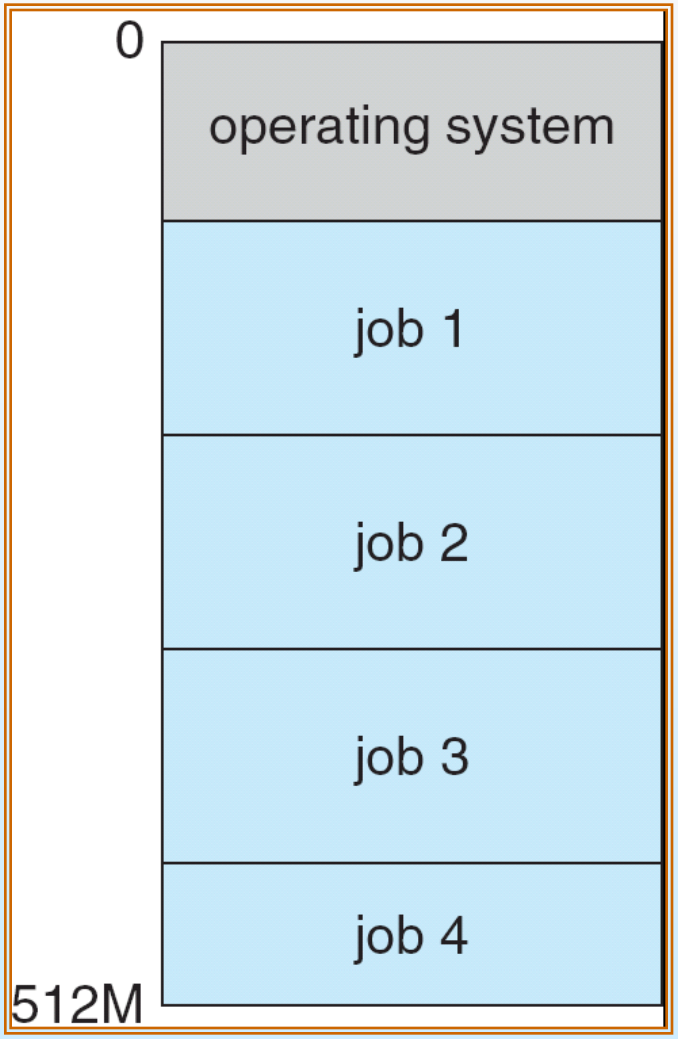
Migration of Integer A from Disk to Register

- 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

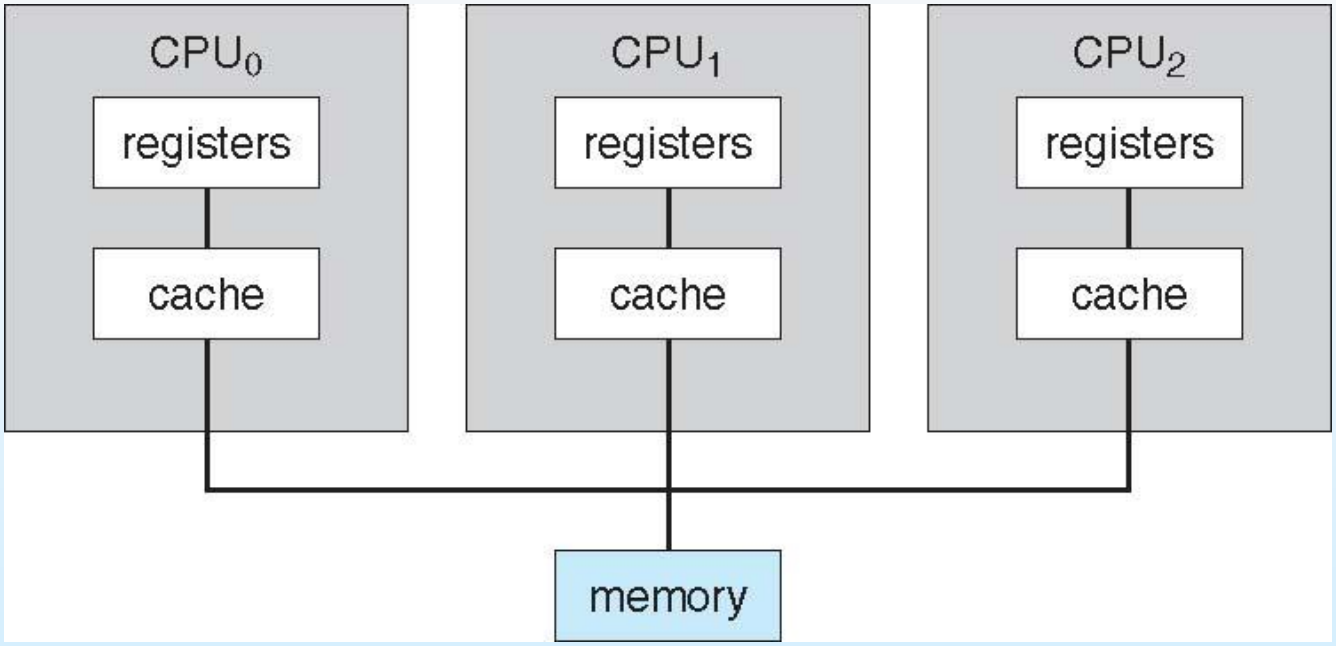
Memory Layout for Multiprogrammed System



Computer-System Architecture

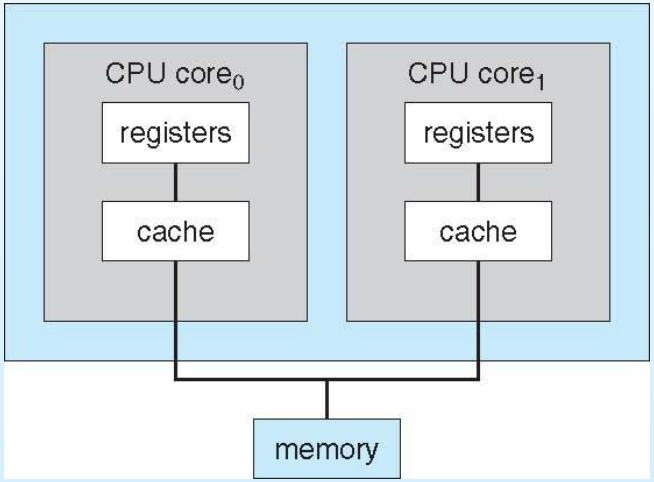
- Most systems use a single general-purpose processor
  - Most systems have special-purpose processors as well
- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Advantages include:
    1. **Increased throughput**
    2. **Economy of scale**
    3. **Increased reliability** – graceful degradation or fault tolerance
  - Two types:
    1. **Asymmetric Multiprocessing** – each processor is assigned a specific task.
    2. **Symmetric Multiprocessing** – each processor performs all tasks

Symmetric Multiprocessing Architecture



A Dual-Core Design

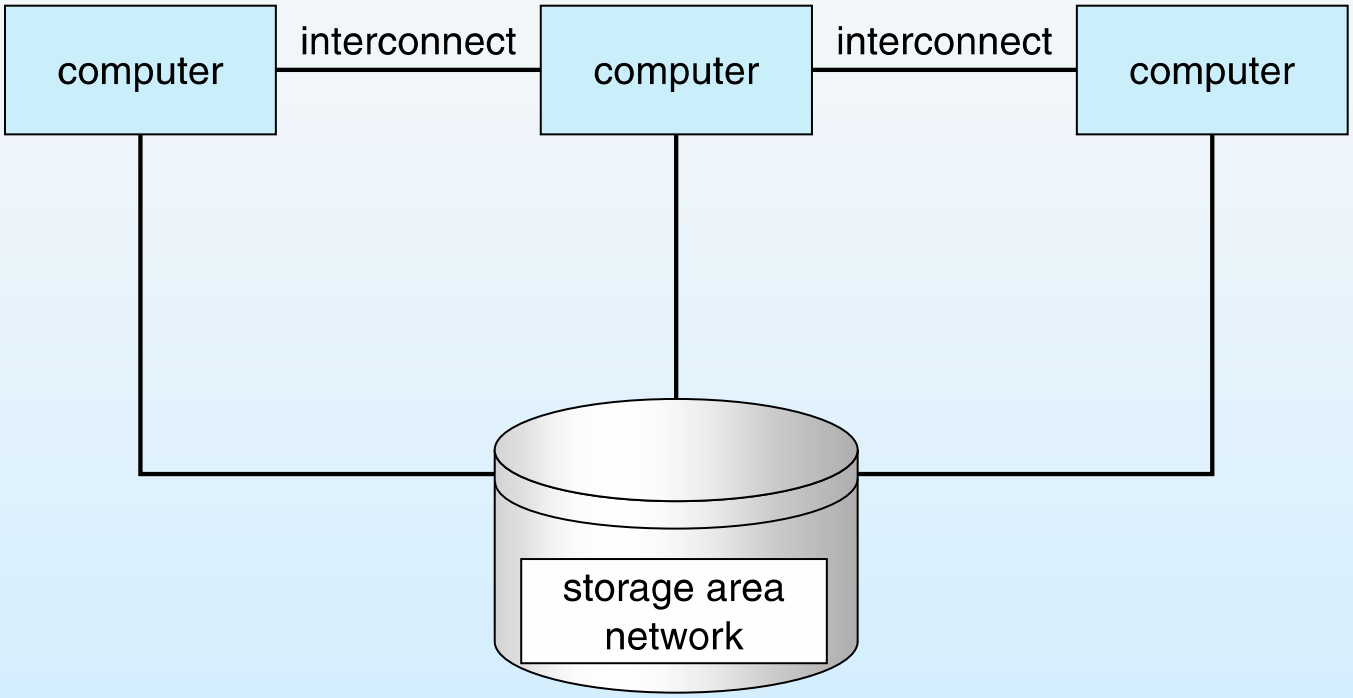
- Multi-chip and **multicore**
- Systems containing all chips
  - Chassis containing multiple separate systems



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 (a machine that just monitors the others)
    - **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

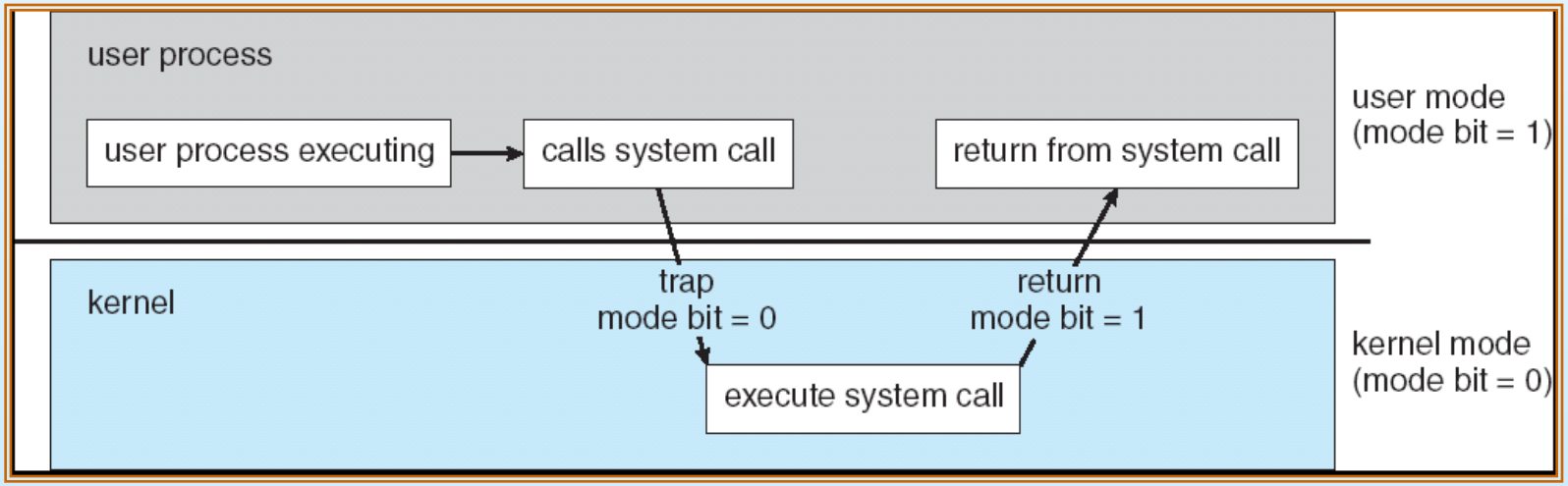


Operating-System Operations

- 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** 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

- 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



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 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
  - 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

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

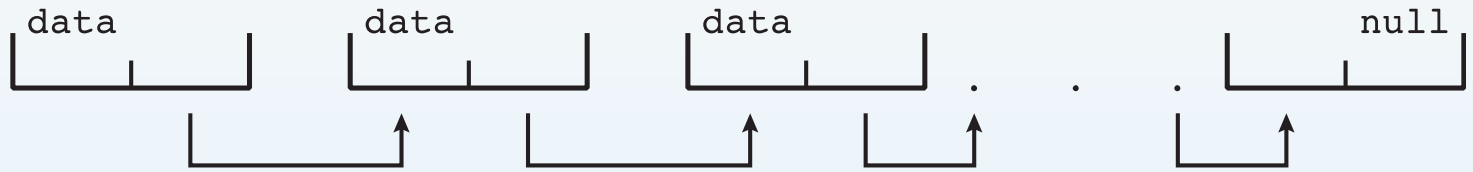
- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including
    - **buffering** (storing data temporarily while it is being transferred)
    - **caching** (storing parts of data in faster storage for performance)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

Protection and Security

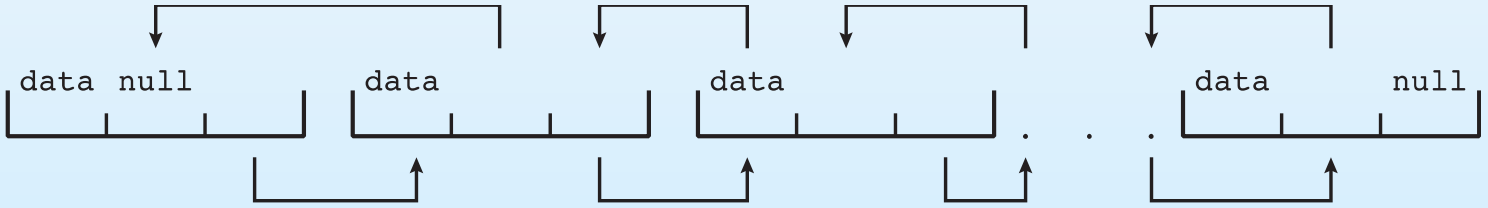
- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first 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

Kernel Data Structures

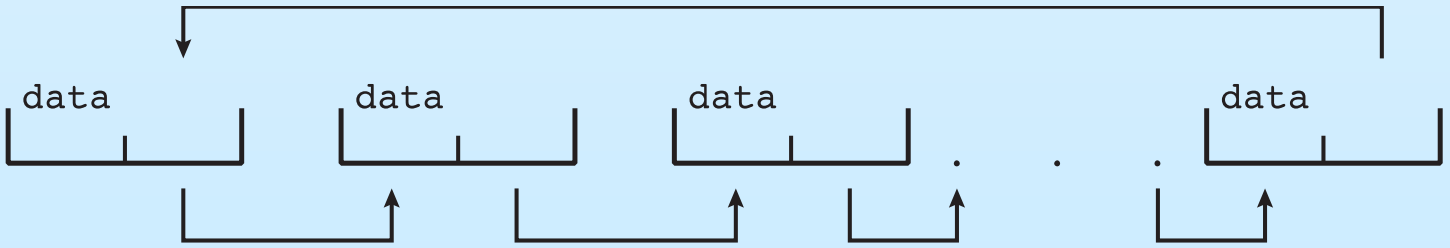
- Many similar to standard programming data structures
- ***Singly linked list***



- ***Doubly linked list***

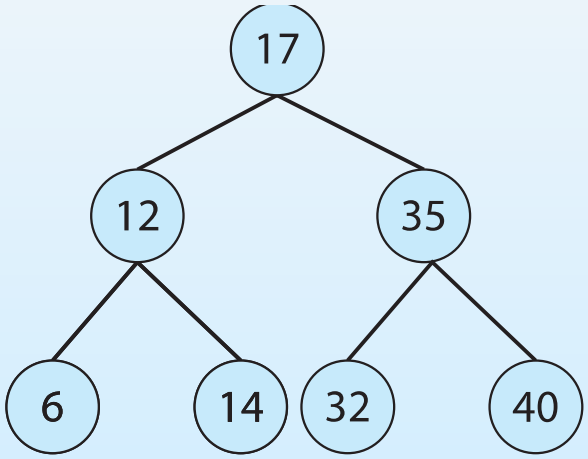


- ***Circular linked list***



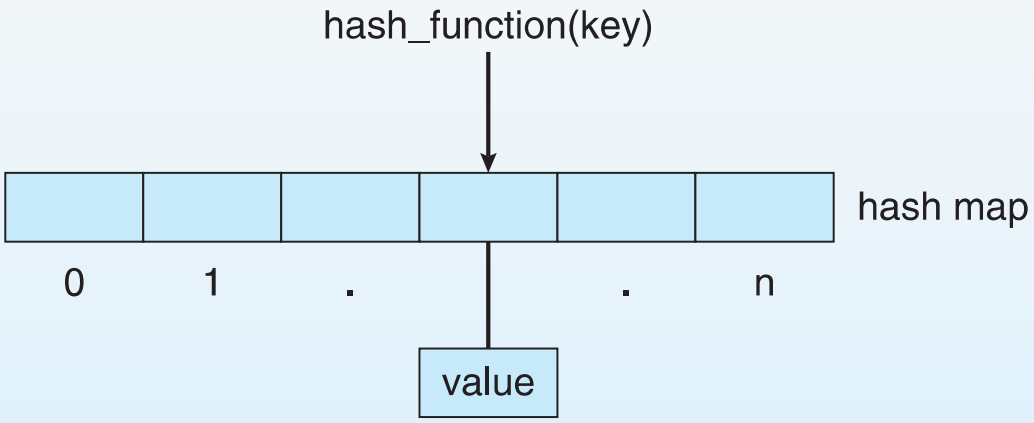
Kernel Data Structures

- Binary search tree



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

- 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**) are like Web terminals
- Mobile computers interconnect via **wireless networks**
- Networking becoming ubiquitous – even home systems use **firewalls** to protect home computers from Internet attacks

Computing Environments - Mobile

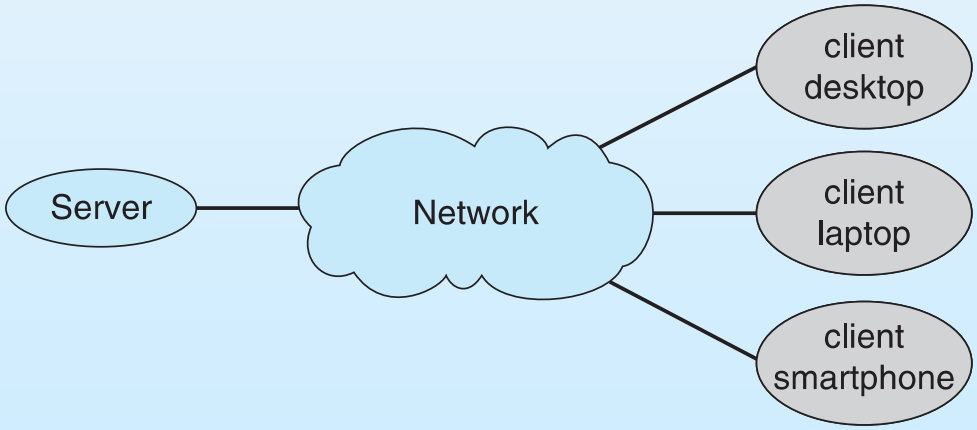
- Handheld smartphones, tablets, etc
- What is the functional difference between them and a “traditional” laptop?
- Extra feature – more OS features (GPS, gyroscope)
- Allows new types of apps like ***augmented reality***
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders are **Apple iOS** and **Google Android**

Computing Environments – Distributed

- Distributed computing
  - 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**)
      - **Personal Area Network** (**PAN**)
  - **Network Operating System** provides features between systems across network
    - Communication scheme allows systems to exchange messages
    - Illusion of a single system

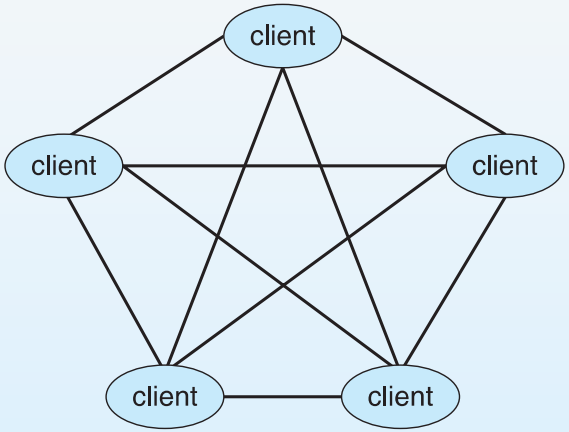
Computing Environments – Client-Server

- Client-Server Computing
  - Dumb terminals supplanted by smart PCs
  - Many systems now **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

- 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, **Voice over IP** (**VoIP**) such as Skype



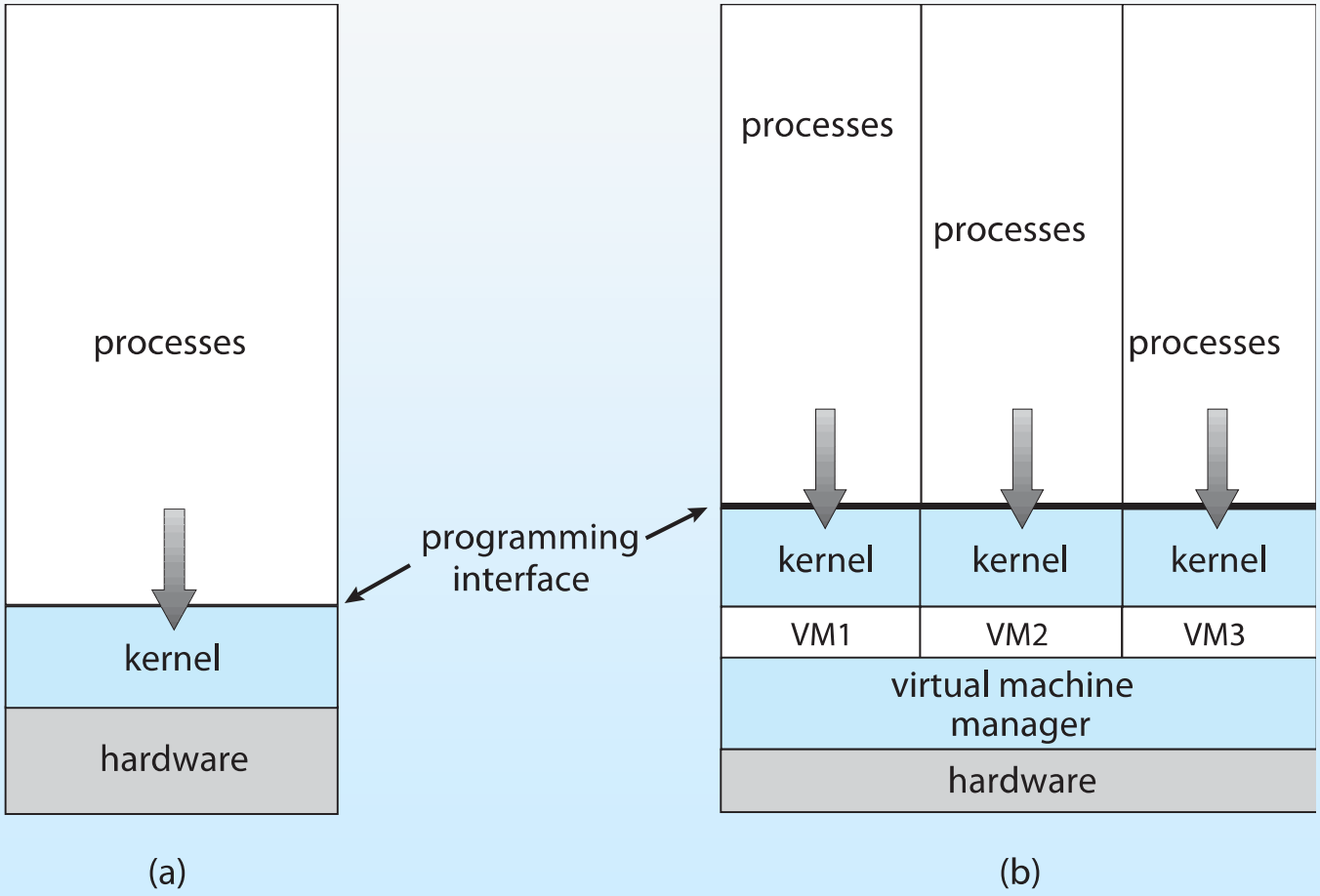
Computing Environments - Virtualization

- Allows operating systems to run applications within other OSes
  - Vast and growing industry
- **Emulation** used when source CPU type different from target type (i.e. PowerPC to Intel x86)
  - Generally slowest method
  - When computer language not compiled to native code – **Interpretation**
- **Virtualization** – OS natively compiled for CPU, running **guest** OSes also natively compiled
  - Consider VMware running WinXP guests, each running applications, all on native WinXP **host** OS
  - **VMM** (virtual machine Manager) provides virtualization services

Computing Environments - Virtualization

- Use cases involve laptops and desktops running multiple OSes for exploration or compatibility
  - Apple laptop running Mac OS X host, Windows as a guest
  - Developing apps for multiple OSes without having multiple systems
  - QA testing applications without having multiple systems
  - Executing and managing compute environments within data centers
- VMM can run natively, in which case they are also the host
  - There is no general purpose host then (VMware ESX and Citrix XenServer)

Computing Environments - Virtualization

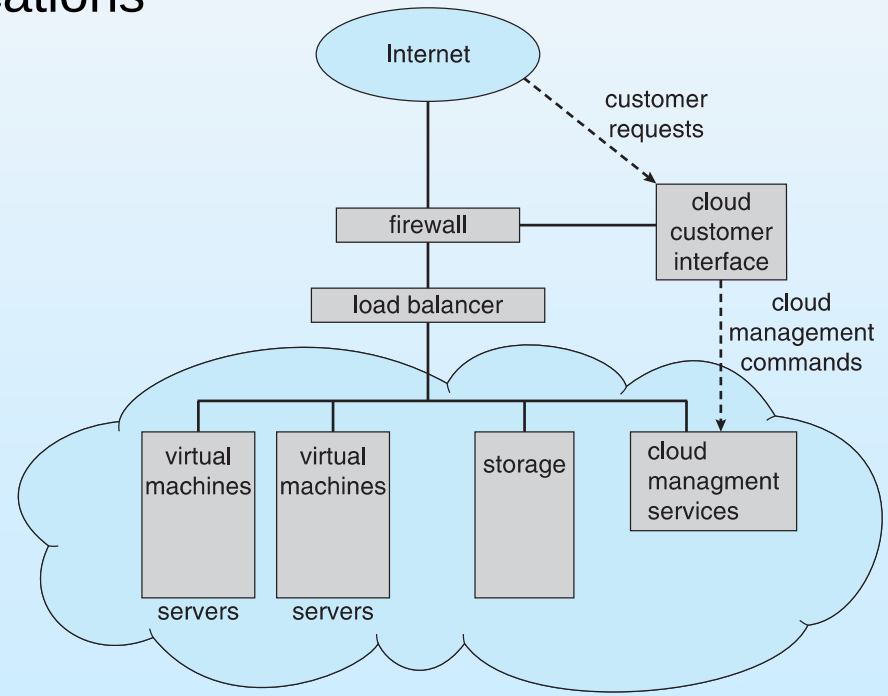


Computing Environments – Cloud Computing

- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization because it uses virtualization as the base for it functionality.
  - Amazon **EC2** has thousands of servers, millions of virtual machines, petabytes of storage available across the Internet, pay based on usage
- Many types
  - **Public cloud** – available via Internet to anyone willing to pay
  - **Private cloud** – run by a company for the company's own use
  - **Hybrid cloud** – includes both public and private cloud components
  - Software as a Service (**SaaS**) – one or more applications available via the Internet (i.e., 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 computing environments composed of traditional OSes, 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 most prevalent form of computers
  - Vary considerable, special purpose, limited purpose OS, **real-time OS**
  - Use expanding
- Many other special computing environments as well
  - Some have OSes, some perform tasks without an OS
- Real-time OS has well-defined fixed time constraints
  - Processing ***must*** be done within constraint
  - Correct operation only if constraints met

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** and **BSD UNIX** (including 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)
  - Use to run guest operating systems for exploration

Readings

- Silberschatz: Chapter 1.
- Tanenbaum: Sections 1.1, 1.2, 1.3
- Get slides from website
  - http://www.marenglenbiba.net/opsys/

Keep in mind ☺

Exp.	Explicit	Prefix	Exp.	Explicit	Prefix
10^{-3}	0.001	milli	10^{3}	1,000	Kilo
10^{-6}	0.000001	micro	10^{6}	1,000,000	Mega
10^{-9}	0.000000001	nano	10^{9}	1,000,000,000	Giga
10^{-12}	0.000000000001	pico	10^{12}	1,000,000,000,000	Tera
10^{-15}	0.000000000000001	femto	10^{15}	1,000,000,000,000,000	Peta
10^{-18}	0.000000000000000001	atto	10^{18}	1,000,000,000,000,000,000	Exa
10^{-21}	0.000000000000000000001	zepto	10^{21}	1,000,000,000,000,000,000,000	Zetta
10^{-24}	0.000000000000000000000001	yocto	10^{24}	1,000,000,000,000,000,000,000,000	Yotta

End of Chapter 1