Chapter 1: Introduction

Silberschatz et al, 8th or higher edition

Regarding Attendance

- I will call names, about 10 to 20, in a random manner
- You should confirm by saying “PRESENT”.

Outline

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

What is an Operating System?

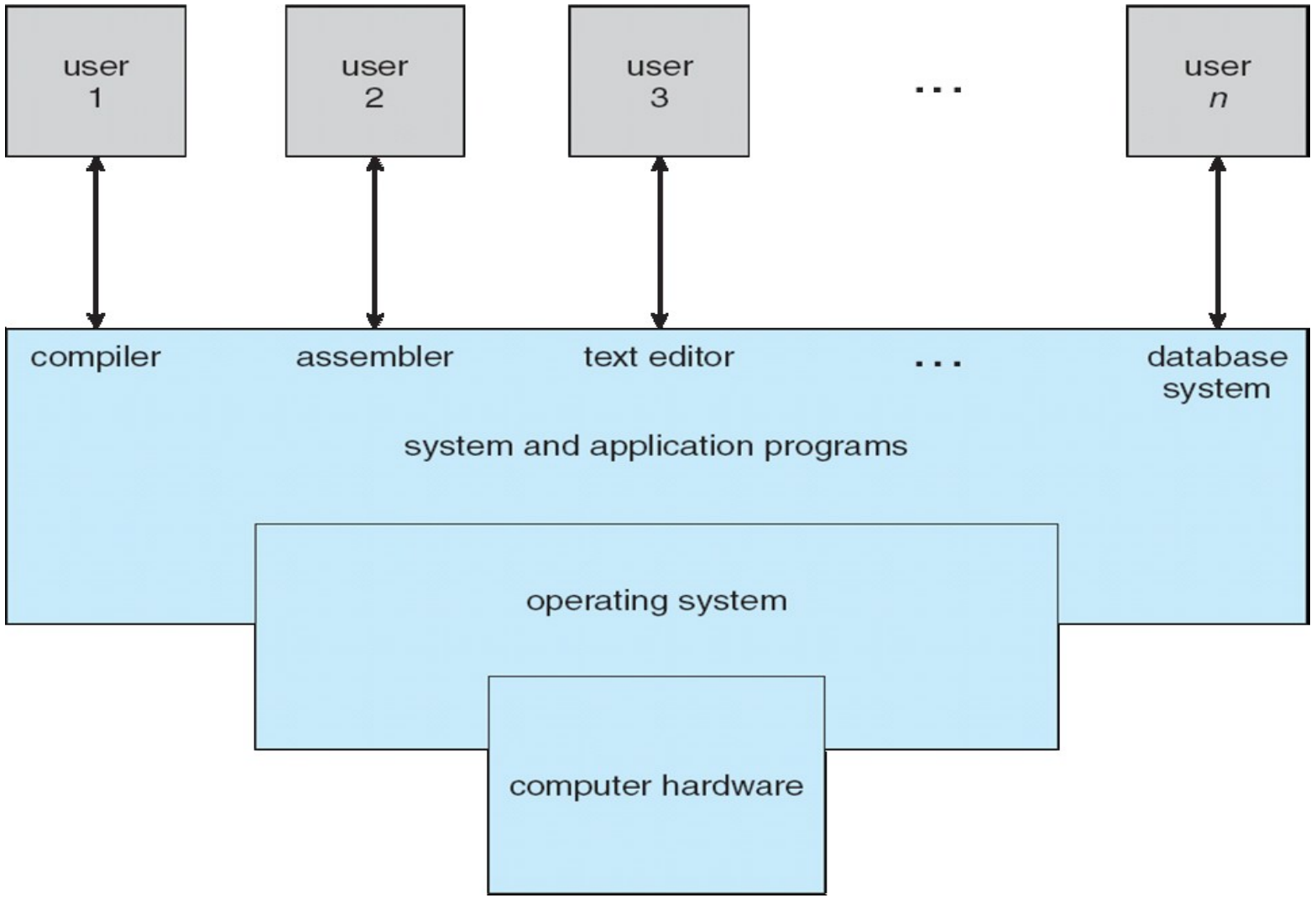
- A program that acts as an intermediary between a user of a computer and the computer hardware

- 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
    - Examples: Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, other computers

Four Components of a Computer System

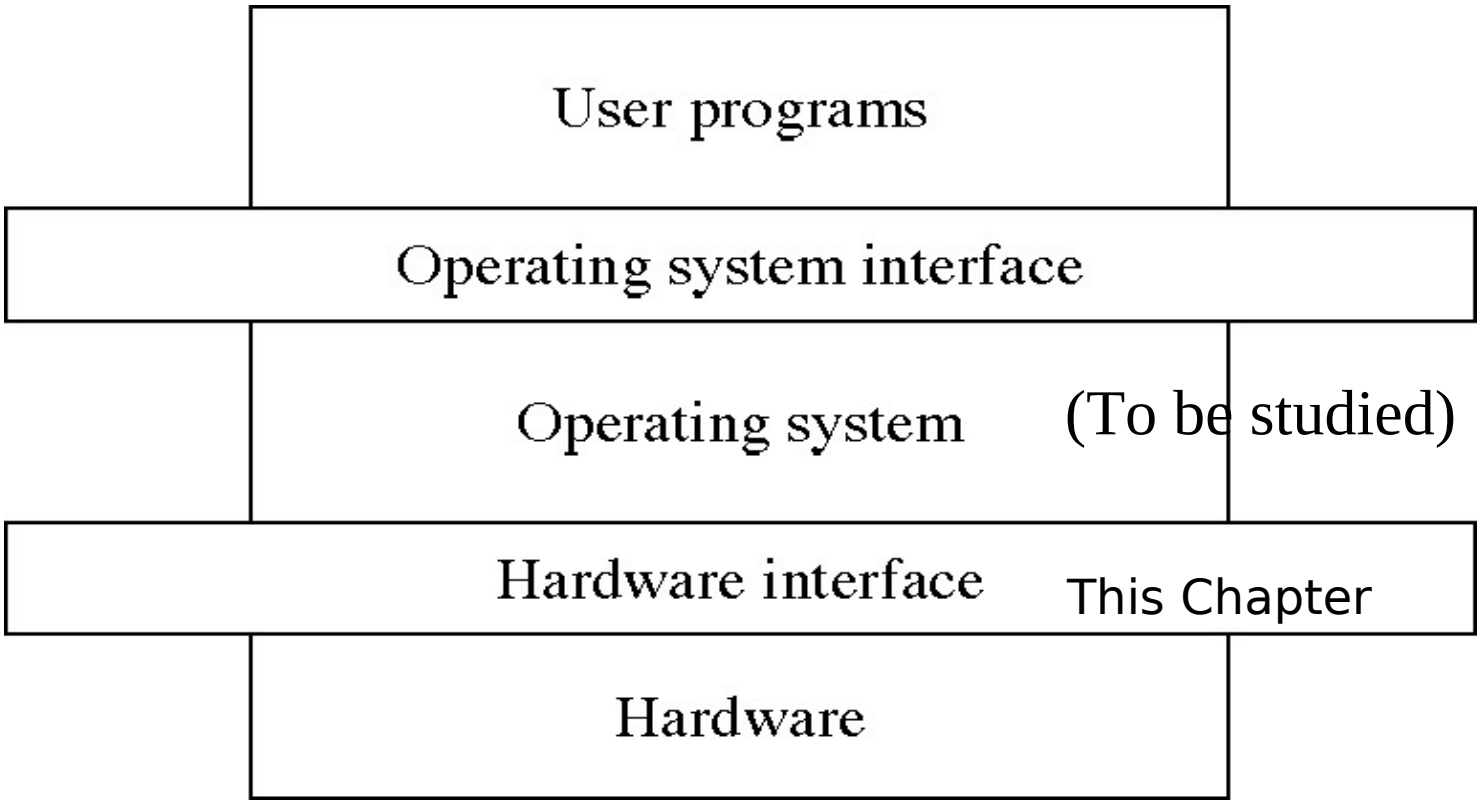


What Operating Systems Do

- Depends on the point of view
- Users view
  - Users want convenience, **ease of use**
  - Don’t care about **resource utilization**
  - But shared computer such as **mainframe** or **minicomputer** must keep all users happy
- Users of dedicate 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

Objectives of this chapter

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



Operating System Definition

- Systems view
  - 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.

Computer Startup

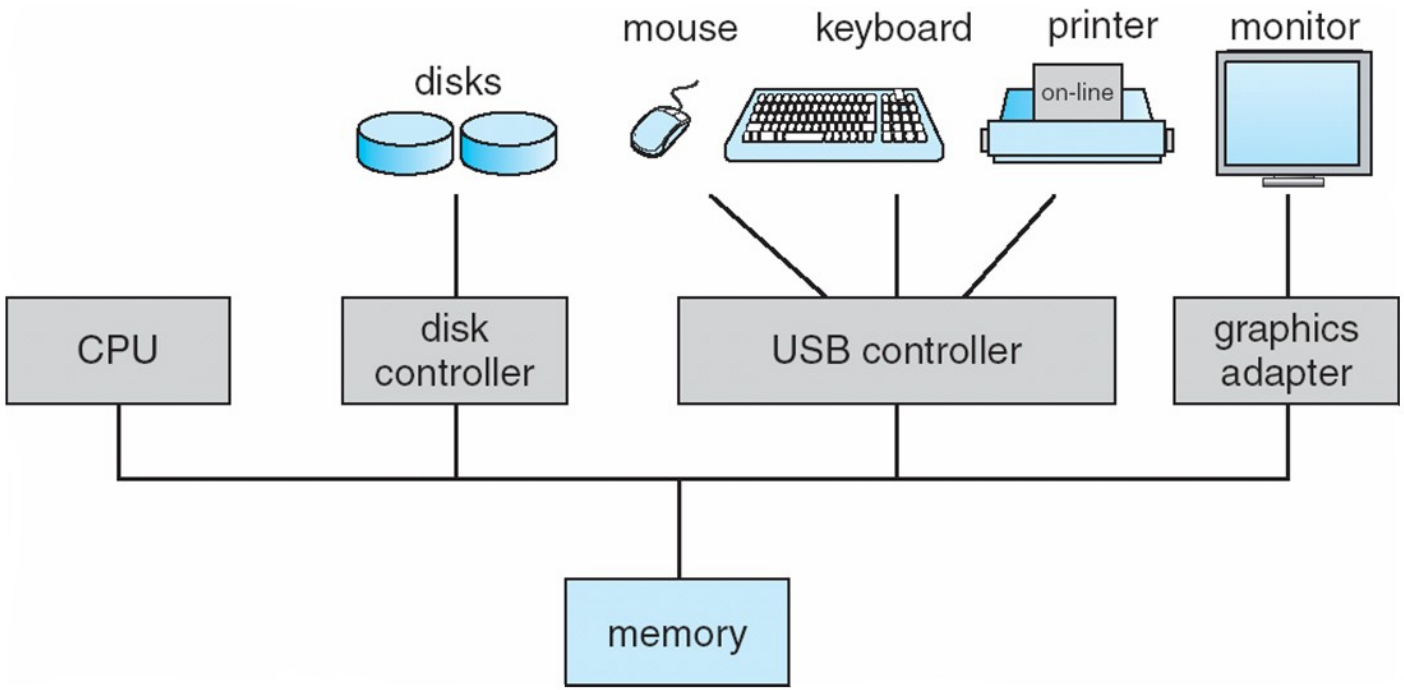
- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

Outline

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

Computer System Organization

- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles



Computer-Components

- Computer consists of processor, memory, and I/O components, with one or more modules of each type. These modules are connected through interconnection network.
- 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*.**

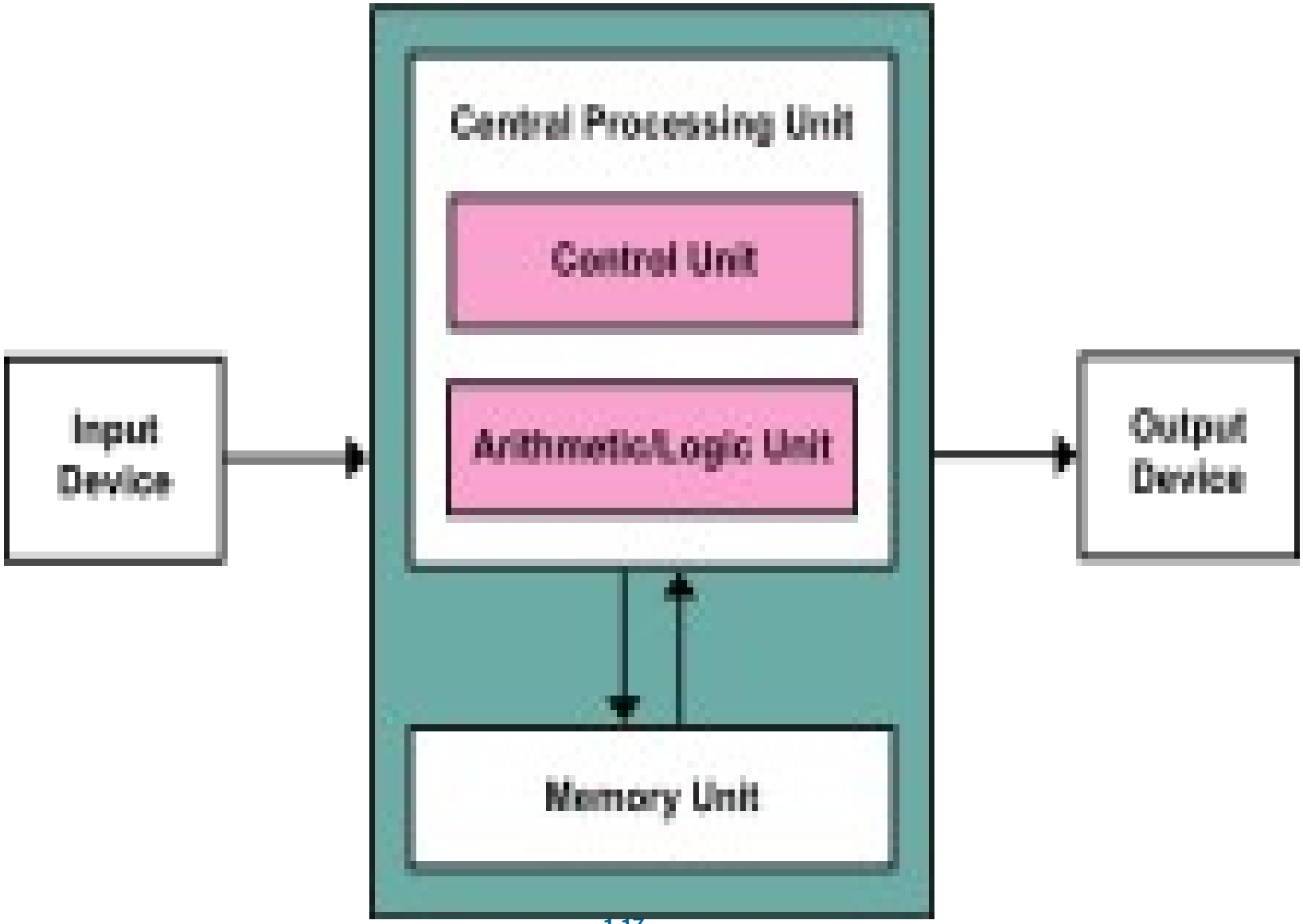
About the Interrupts

- In the next few slides, we will study the basics of interrupts by considering the architecture and operation of a simple computer.

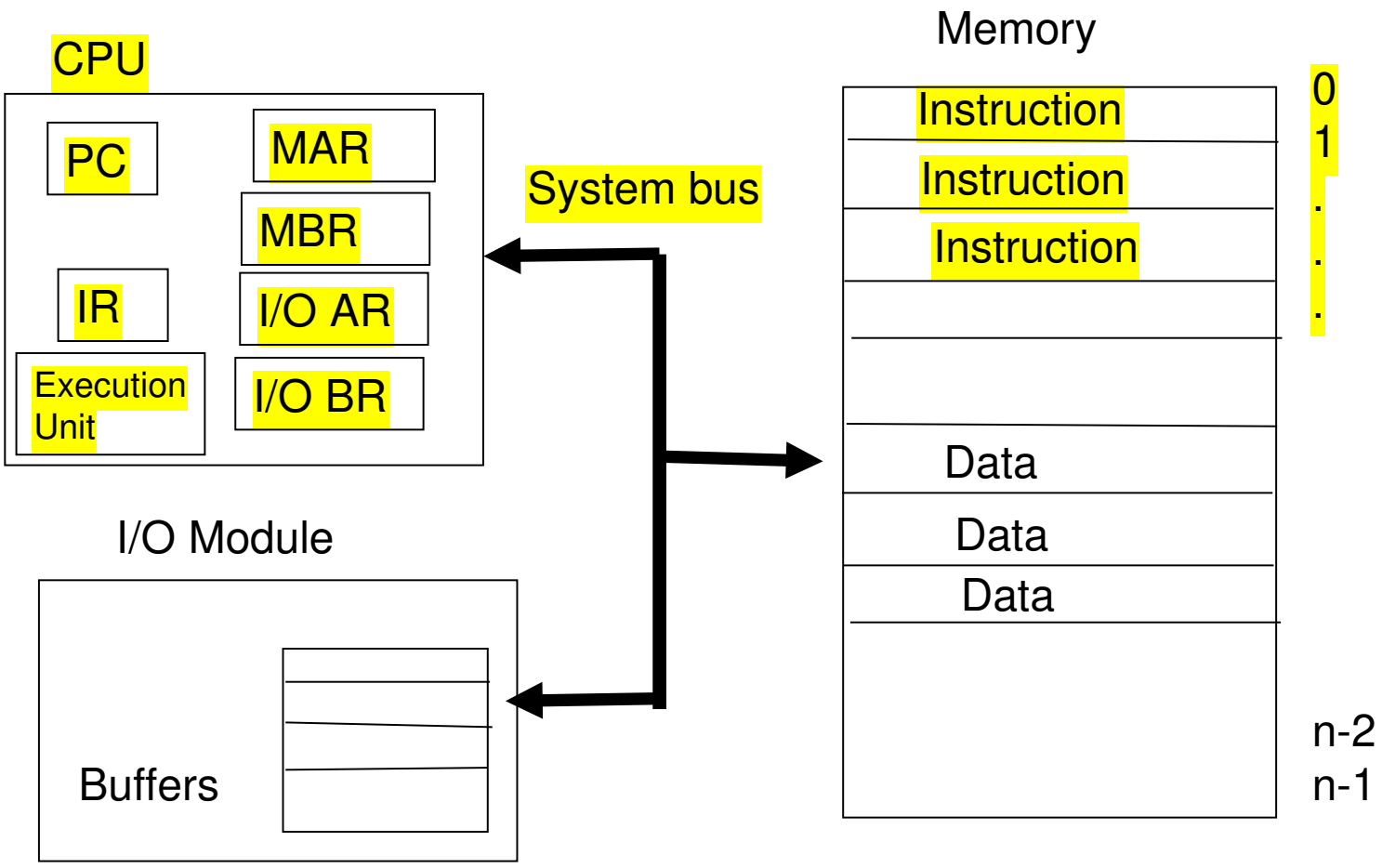
Von Neumann architecture

- The von Neumann architecture—also known as the von Neumann model or Princeton architecture—is a computer architecture based on a 1945 description by the mathematician and physicist John von Neumann and others in the First Draft of a Report on the EDVAC (Electronic Discrete Variable Automatic Computer). That document describes a design architecture for an electronic digital computer with
  - A processing unit that contains an arithmetic logic unit and processor registers
  - A control unit that contains an instruction register and program counter
  - Memory that stores **data and instructions**
  - External mass storage
  - Input and output mechanisms

Von Neumann architecture



Architecture of a simple computer



Architecture of a simple computer

- **Processor:** Controls the operation of the computer and performs data processing functions. It is called CPU.
- **Main memory:** Stores data and programs; it is volatile
- **I/O modules:** Moves data between the computer and external environment.
- **System bus:** mechanism of communication among processors, main memory, and I/O modules.

Simple Computer

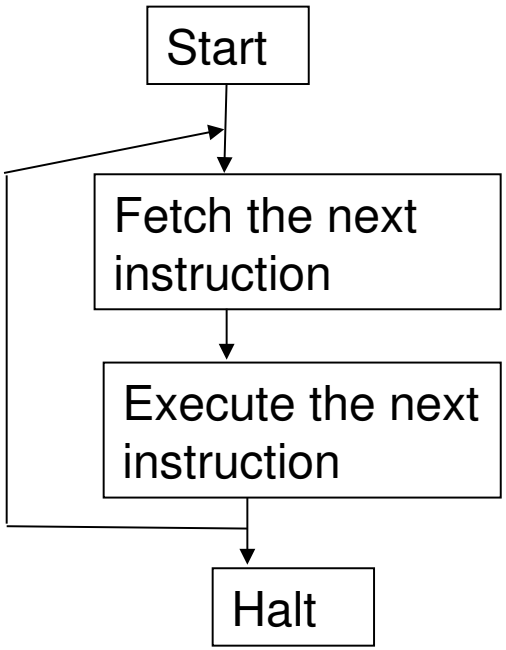
- Operation: Processor controls everything.
  - **MAR**: Memory address register: which specifies the address in memory for the next read or write.
  - **MBR**: Memory buffer register: which contains the data to be written into memory or which receives data read from memory.
  - **I/O AR**: Address of I/O device
  - **I/O BR**: Exchange of data between I/O and computer.

Simple Computer

- Processor Registers: Within the processor there is a set of registers that provide a level of memory that is faster and smaller than main memory.
  - User visible registers: Available to programmer
    - **Data registers**: Can be used by the programmer.
    - **Address Registers**: Contains Main memory address of data and instructions.
      - **Index register**: Index to base value
      - **Segment pointer**: It contains a reference to a particular segment.
      - **Stack pointer**: Points top of the stack.
  - **Control and status registers**: These are employed to control the operation of the processor. Differ from machine to machine.
    - MAR, MBR,I/O AR, and I/O BR
    - **Program Counter:** Contains the address of the instruction to be fetched.
    - **Instruction Register**: Contains the instruction most recently fetched.
    - **PSW:** Program Status word: It is a register or a set of registers.

Simple Computer…

- **PSW:** Program Status word: It is a register or a set of registers.
  - **Sign:** contains sign bit of last arithmetic operation
  - **Zero**: it is set if the result of arithmetic operation is zero
  - **Carry:** It is set if there is a carry or borrow.
  - **Equal:** If the compare result is equality
  - **Overflow:** It is set if the result is overflow.
  - **Interrupt enable/disable:** Used to disable or enable interrupts.
  - **Supervisor: Indicates whether the processor is executing in supervisor or user mode.**



- **Instruction execution:**
  - Program execution is the main function of the computer.
  - Instruction fetch and execute

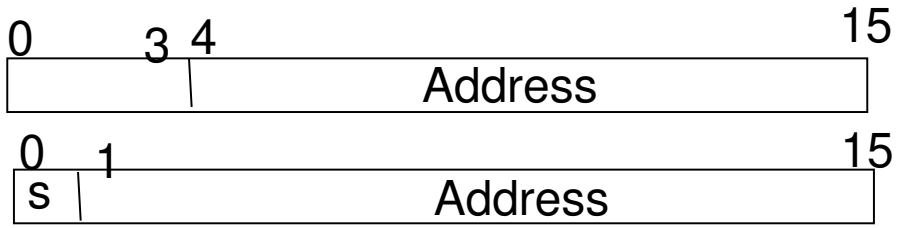
Simple computer…

- **Fetching:** Bringing the instructions from the main memory.
  - PC: Contains the address of the next instruction to be fetched.
    - Incremented unless told otherwise
- **Execute:**
  - The processor interprets the instructions and performs the required action.
    - **Processor-memory:** Transfer data from the memory
    - **Processor-I/O-** transfer data from peripheral device from the memory.
    - **Data processing:** Arithmetic and logic operations
    - **Control:** The instructions may specify the sequence of the next instruction to be fetched.

Simple Computer..

An Example:

Instruction format



Integer format

PC: address of the instruction
IR: Instruction being executed
AC: Accumulator: temporary storage

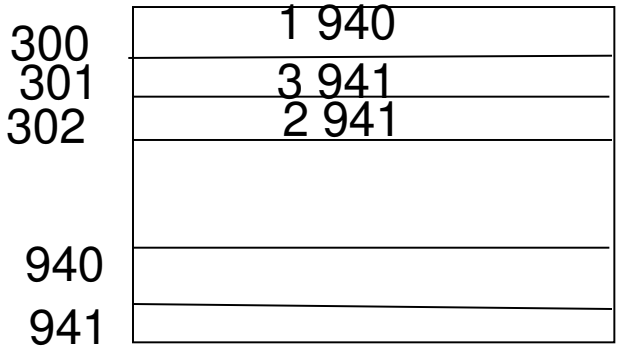
Program

LOAD 940
ADD 941
STORE 941
HLT

1=0001=LOAD ADDRESS: Load AC from memory

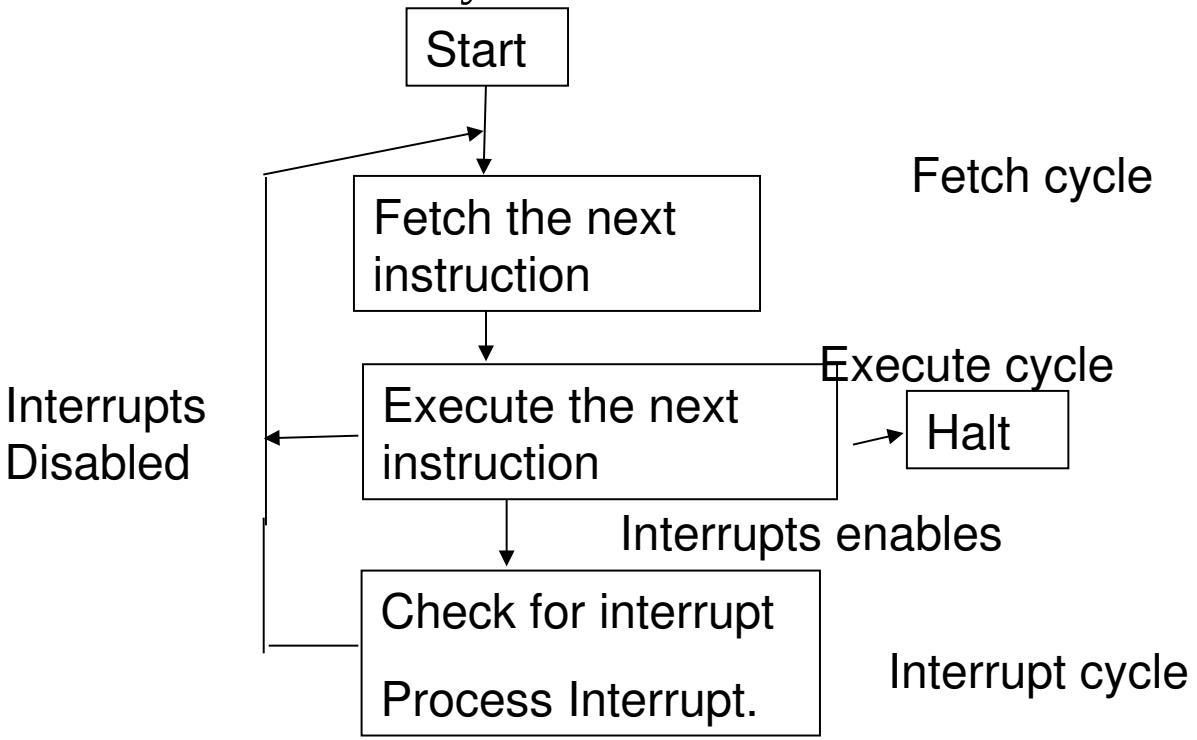
2=0010=STORE ADDRESS: Store AC to memory

3=0101=ADD ADDRESS: Add AC to memory



Interrupt processing

- To improve the performance, interrupts are provided.
- With interrupts, the processor can be engaged in executing other while an I/O operation is in progress.
- Interrupt cycle is added to the instruction cycle.



Common Functions of Interrupts

- Interrupt transfers the 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* is a software-generated interrupt caused either by an error or a user request.
- **An operating system is *interrupt* driven.**

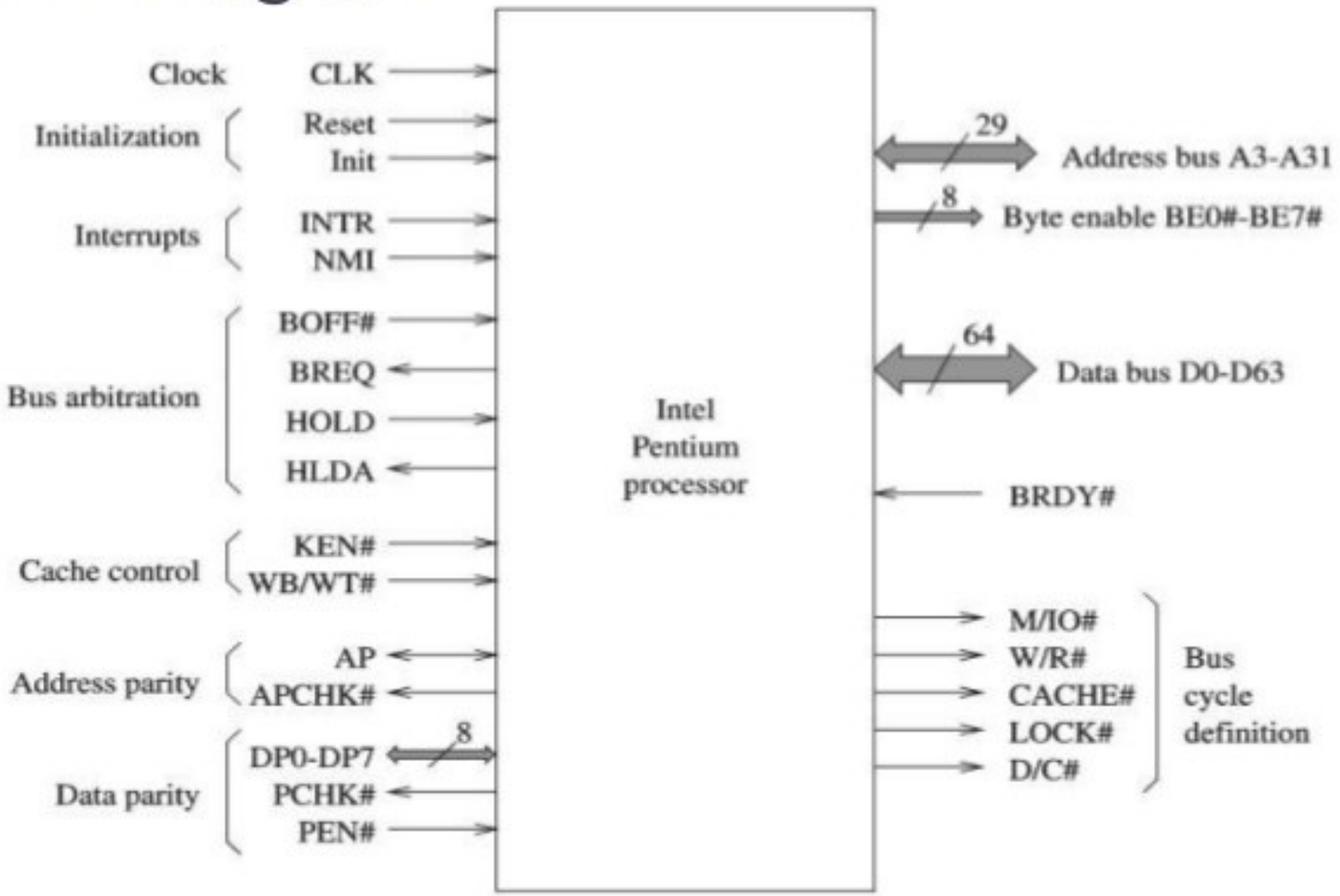
Interrupt Handling

- To start I/O operation, CPU loads the appropriate registers within the device controller
- The device controller examines the contents of these registers to determine what action to take.
- If it s a read operation the controller transfers the data into local buffer.
- Then it informs the CPU through interrupt.
- The operating system preserves the state of the CPU by storing registers and the program counter.

Interrupt Handling…

- **Interrupt handler:** The CPU hardware has a wire called interrupt request line; that CPU senses after executing every instruction.
- When a CPU senses a signal, it saves the PC, and PSW on a stack and jump to the interrupt handler routine at a fixed address in memory.
- **Interrupt vector:** It contains the memory addresses of specialized interrupt handlers.

Pin Diagram


Clock
CLK
Initialization
Reset
Init
Interrupts
INTR
NMI
Bus arbitration
BOFF#
BREQ
HOLD
HLDA
Cache control
KEN#
WB/WT#
Address parity
AP
APCHK#
Data parity
8
DP0-DP7
PCHK#
PEN#
Intel
Pentium
processor
29
Address bus A3-A31
8
Byte enable BE0#-BE7#
64
Data bus D0-D63
BRDY#
M/IO#
W/R#
CACHE#
LOCK#
D/C#
Bus
cycle
definition

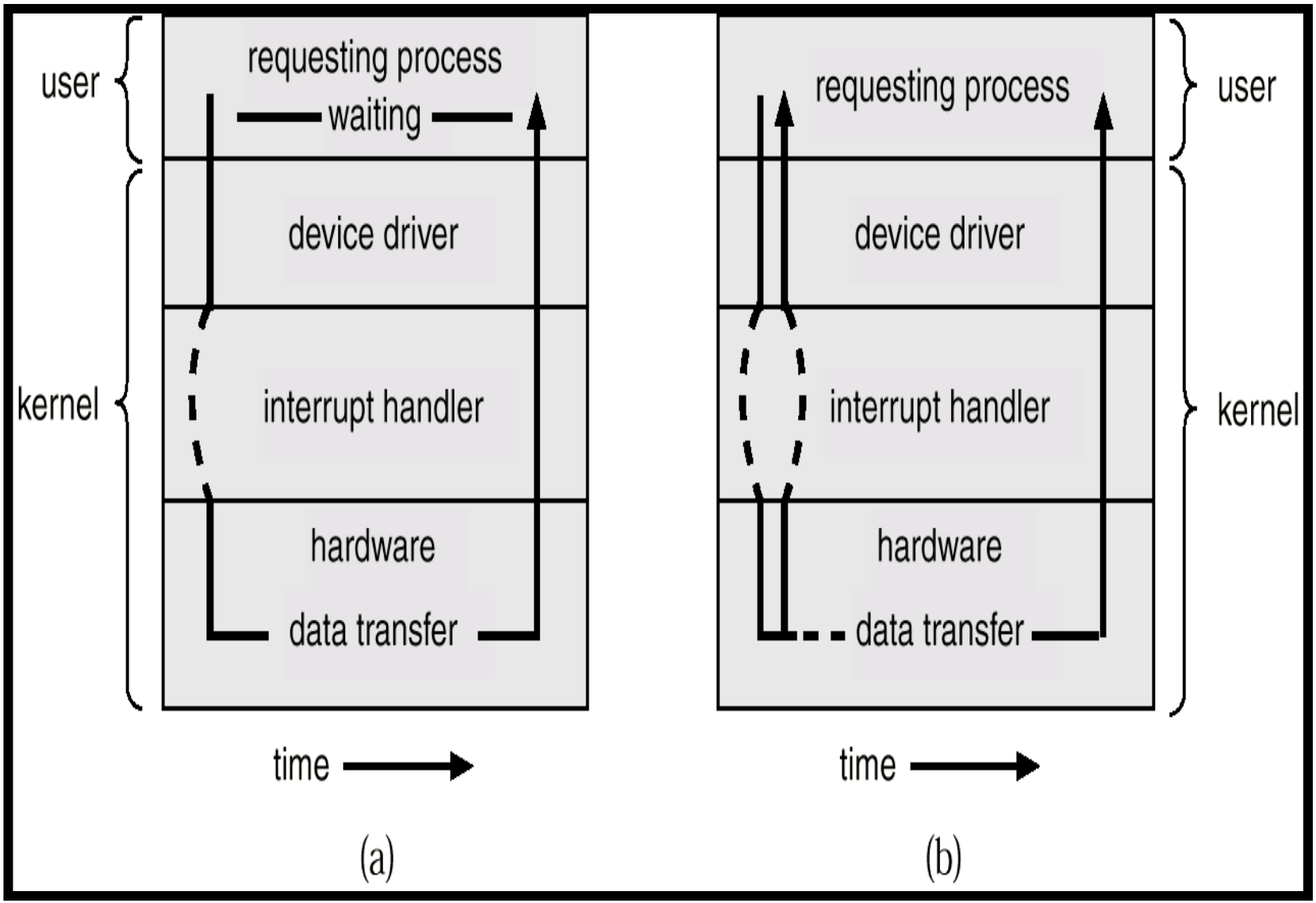
I/O Structure

- Two types of I/O
- **Synchronous I/O**: 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.
- **Asynchronous I/O**: After I/O starts, control returns to user program without waiting 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.

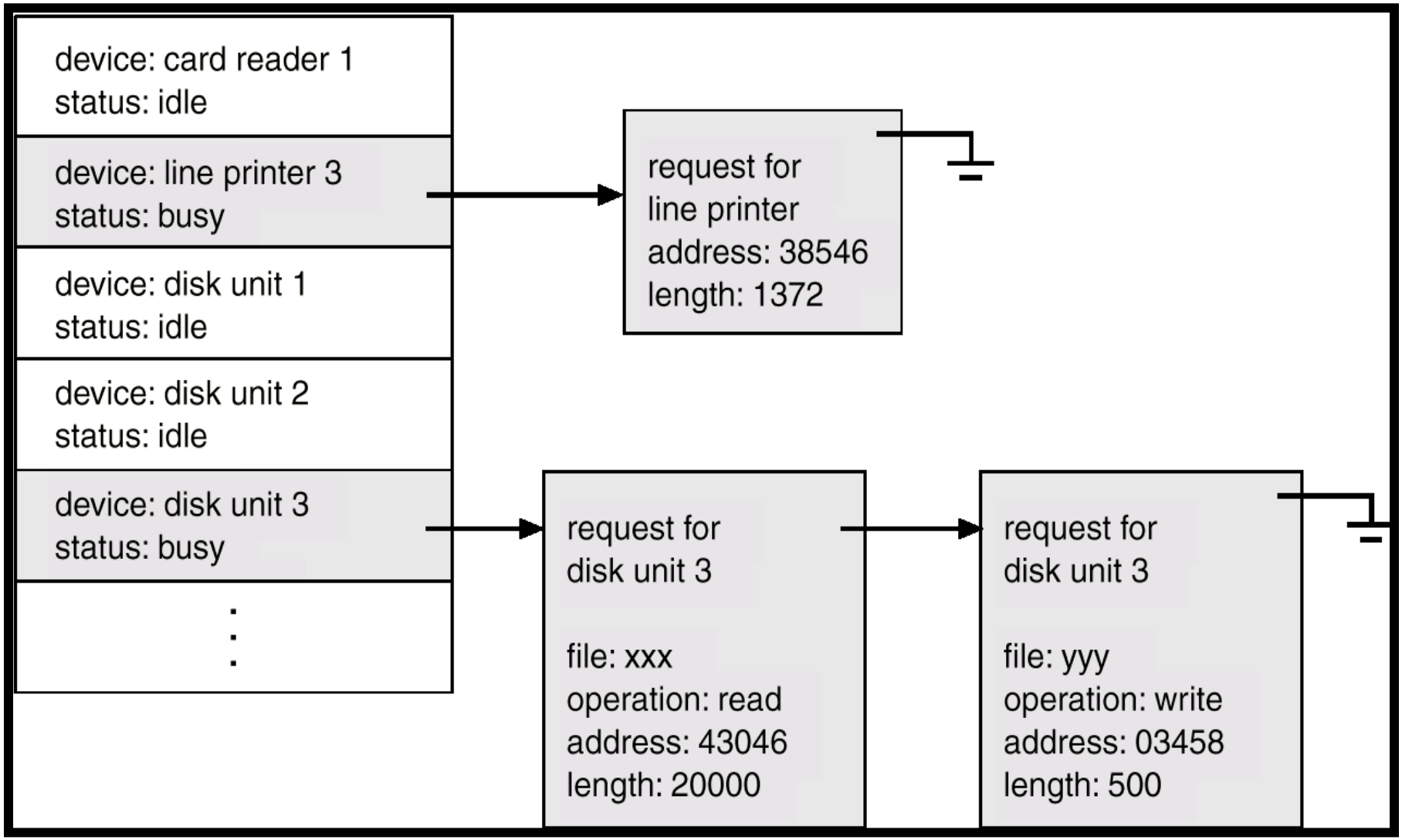
I/O Structure

- Waiting for I/O operation
  - **Loop:** Jump **Loop**
- The above loop continues until interrupt occurs.

Two I/O Methods

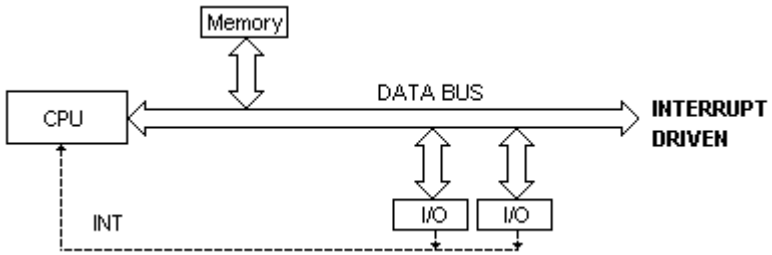


Device-Status Table



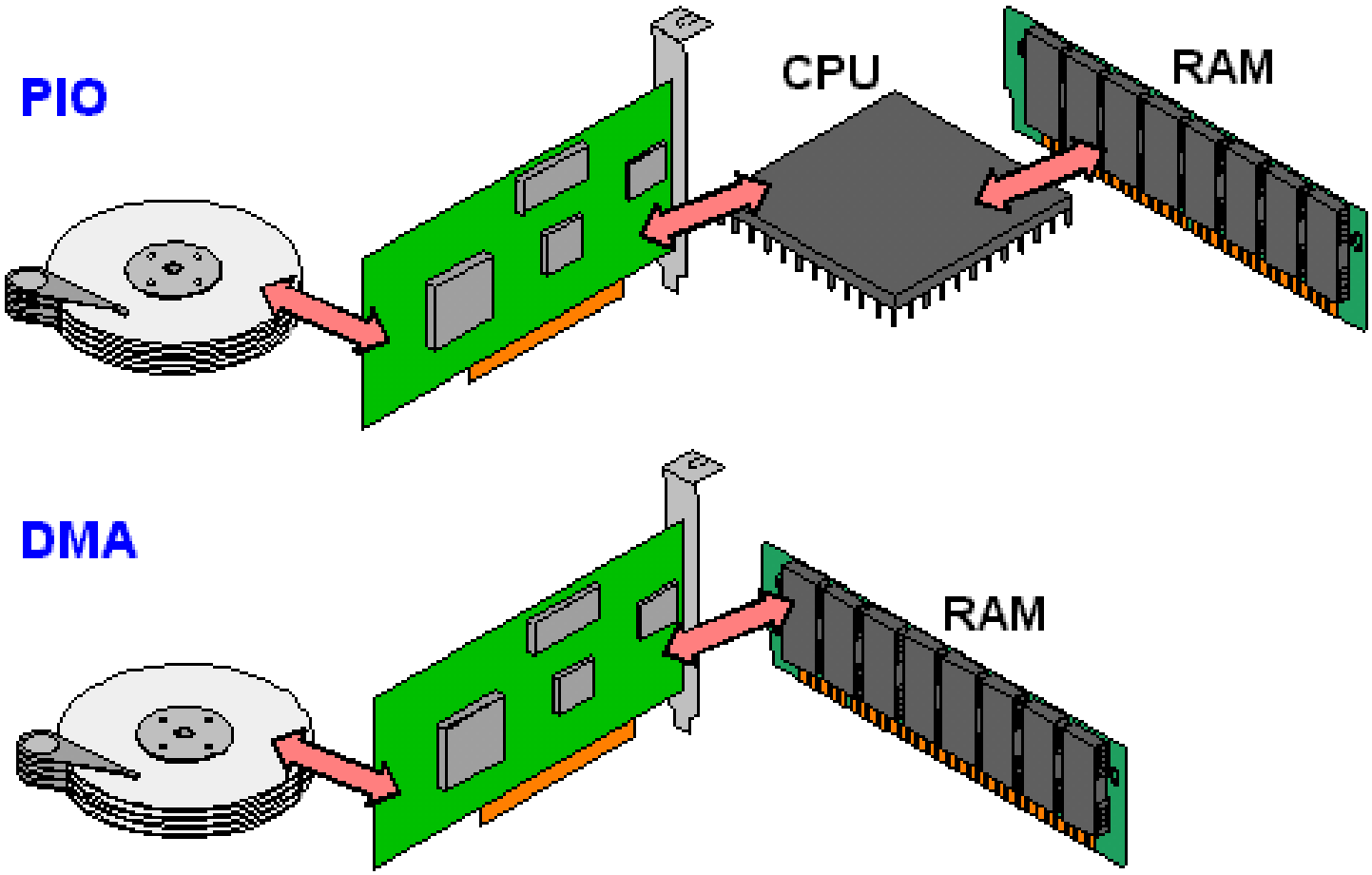
I/O communication techniques

- Three kinds of data transfer
  - **Program data transfer**: CPU checks the I/O status
    - The I/O module does not interrupt the processor.
    - Processor is responsible for extracting the data from main memory and shifting data out of main memory.
    - It is a time-consuming process that keeps the processor busy unnecessarily.
  - **Interrupt driven data transfer:** when I/O is ready it interrupts the CPU
    - In programmed I/O the processor repeatedly interrogate the status of the I/O module.
    - In Interrupt driven data transfer, the I/O module will interrupt the processor to request service when it is ready to exchange data with the processor.

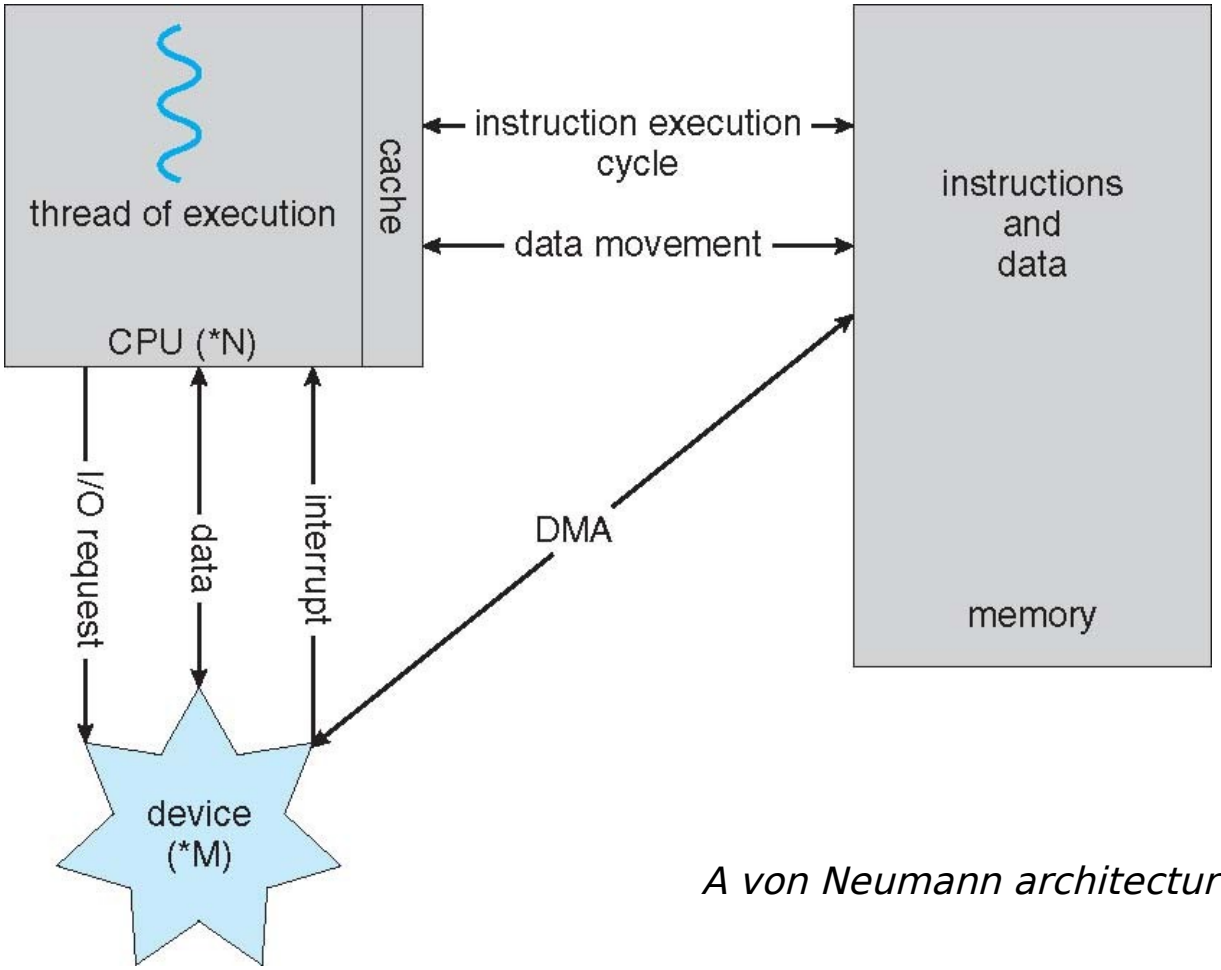


Direct Memory Access (DMA) Structure

- **DMA:**
  - **Used** for high-speed I/O devices able to transmit information at close to memory speeds.
  - Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention.
  - Only on interrupt is generated per block, rather than the one interrupt per byte.
- **DMA:** Processor issues a command to DMA module by sending the following information.
  - Whether read or write is requested.
  - The address of the I/O device involved.
  - The starting location of the memory to read from or write to.
  - The number of words to be read or written.
- The processor continues other work.
- The DMA module transfers the data and interrupts the processor.
- It takes the control of the bus to transfer data from memory.
- The processor becomes slow, or must wait for the bus.



How a Modern Computer (Single Processor System) Works?



A von Neumann architecture

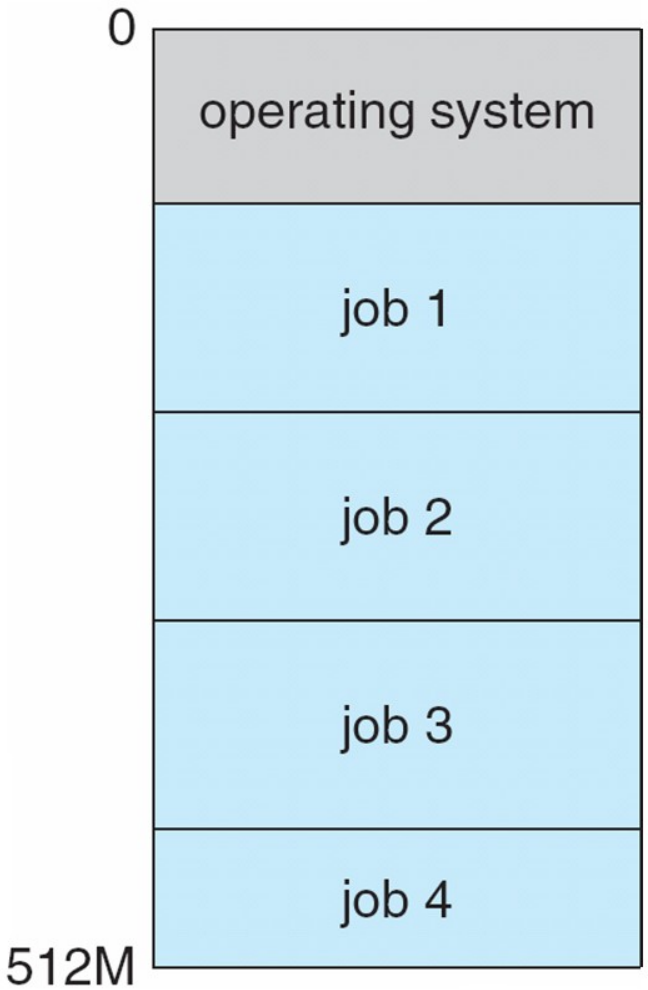
Outline

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

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

Memory Layout for Multiprogrammed System



Operating system operations

- Dual mode operation
- Process Management
- Memory management
- Storage management
- Protection and Security

About Dual-mode operation

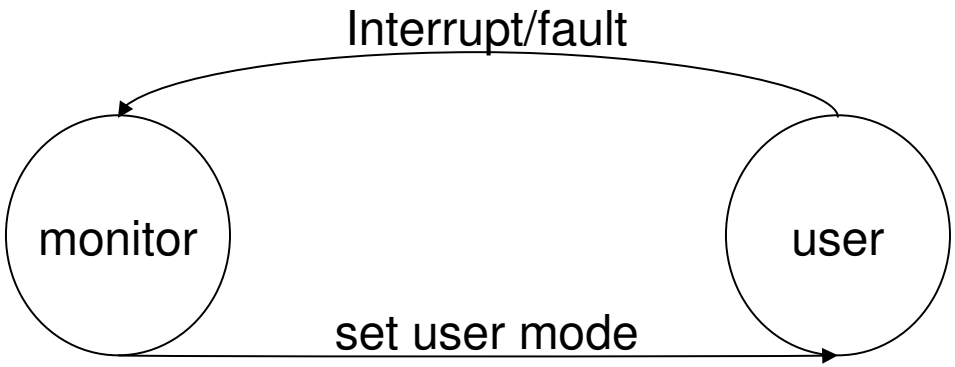
- In Early systems, programmers had complete control over the system.
- A OSs developed the control was given to OS.
- OS started performing many functions such as I/O.
- OS started sharing resources among several programs
- Multiprogramming put several programs at same time.
- Without sharing the error may cause problem to only one program.
- With sharing an error may cause problems to many programs.
  - Sometimes to OS itself.
  - OSs have to be protected from such incorrect programs.
- MS-DOS and MAC-OS allow this kind of error.
- Protection measures.
  - Dual-Mode Operation
  - I/O Protection
  - Memory Protection
  - CPU Protection

Dual-Mode Operation

- Sharing system resources requires operating system to ensure that an incorrect program cannot cause other programs to execute incorrectly.
- Provide hardware support to differentiate between at least two modes of operations.
  1. *User mode* – execution done on behalf of a user.
  2. *Monitor mode* (also *kernel mode* or *system mode*) – execution done on behalf of operating system.
- This architectural enhancement is useful for many aspects of system operation.
- When system starts, hardware starts in monitor mode.
- OS is then loaded and OS starts user processes in user mode.

Dual-Mode Operation (Cont.)

- *Mode bit* added to computer hardware to indicate the current mode: monitor (0) or user (1).
- Whenever OS takes control the mode bit is 0.
- When an interrupt or fault occurs hardware switches to monitor mode.
- This dual-mode operation protects computer from errant users and errant users from other.
- This protection can be achieved by designating some of the instructions as privileged instructions.
- ***Privileged instructions* can be issued only in monitor mode.**
  - **MS-DOS was written without mode bit**
  - **Pentium provides duel mode operation: so it provides greater protection.**



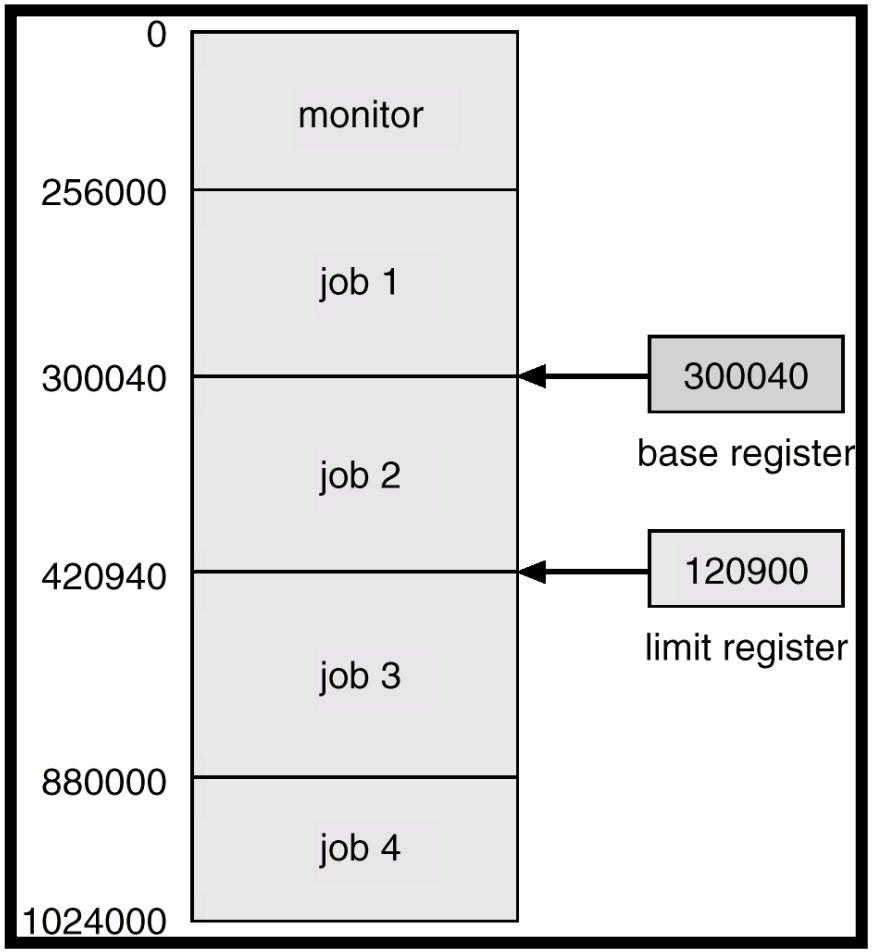
I/O Protection

- A user program may disrupt the normal operation
  - By issuing a illegal I/O operation.
- Solution: All I/O instructions are privileged instructions.
- Must ensure that a user program could never gain control of the computer in monitor mode
- A user program that, as part of its execution, stores a new address in the interrupt vector. How to protect it ?
- **Answer: Consider Interrupt instructions which modify interrupt vector as privileged instructions.**

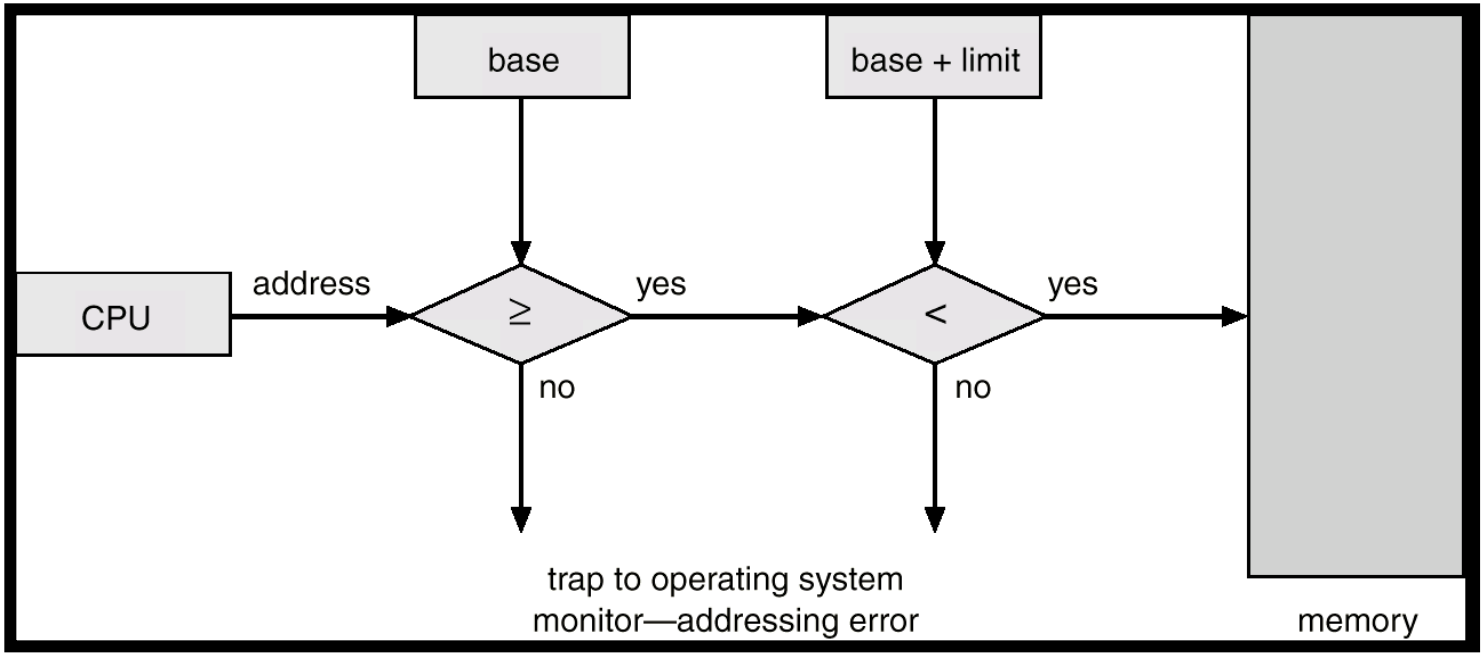
Memory Protection

- Must provide memory protection at least for the interrupt vector and the interrupt service routines.
- In order to have memory protection, add two registers that determine the range of legal addresses a program may access:
  - **Base register** – holds the smallest legal physical memory address.
  - **Limit register** – contains the size of the range
- Memory outside the defined range is protected.

Use of A Base and Limit Register



Hardware Address Protection



Hardware Protection

- When executing in monitor mode, the operating system has unrestricted access to both monitor and user's memory.

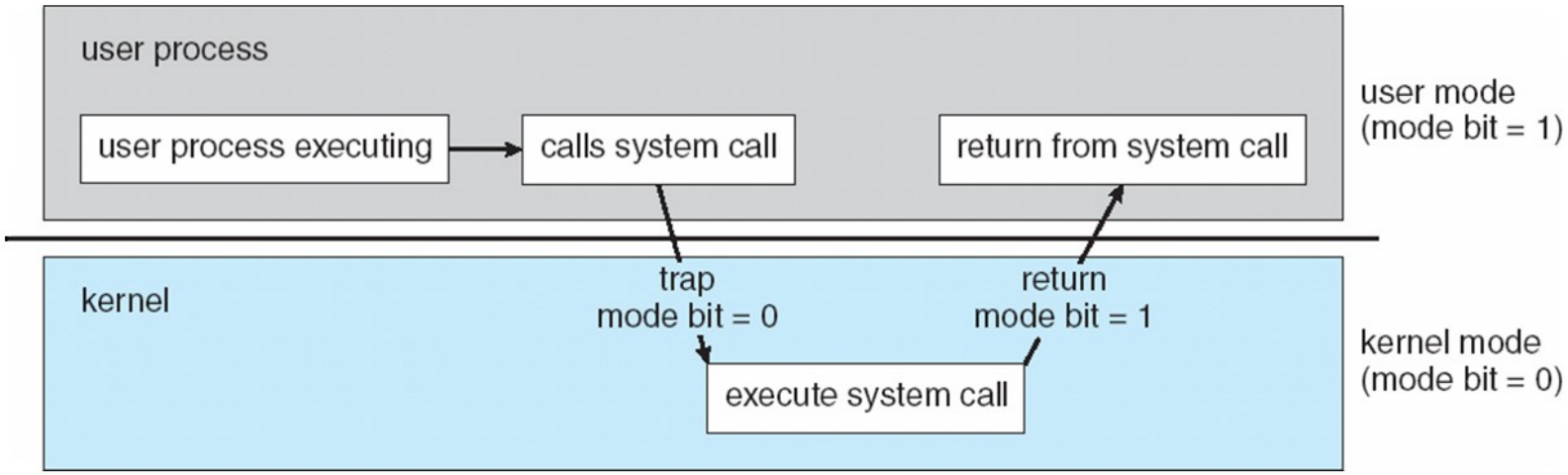
- The load instructions for the *base* and *limit* registers are privileged instructions.

CPU Protection

- Ensures that OS maintains control.
- *Timer* – interrupts computer after a specified period to ensure operating system maintains control.
  - Timer is decremented every clock tick.
  - When timer reaches the value 0, an interrupt occurs.
- Timer commonly used to implement time sharing.
- Timer also used to compute the current time.
- Load-timer is a privileged instruction.

Transition from User to Kernel/Supervisor Mode

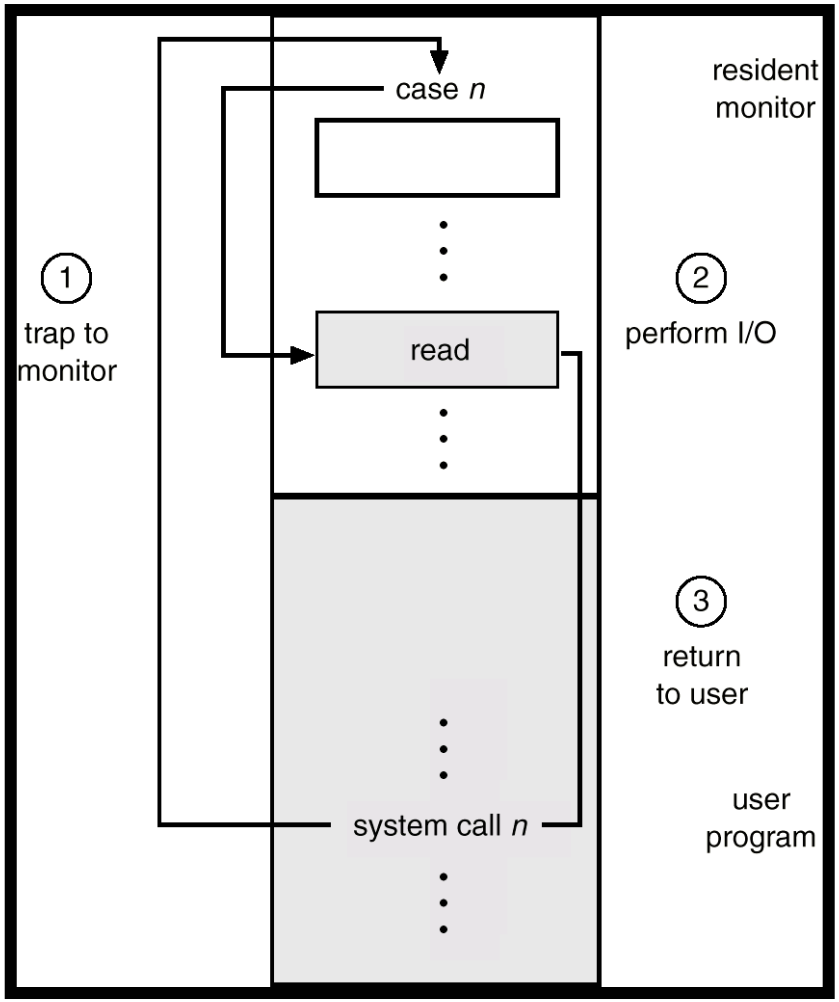
- Timer to prevent infinite loop / process hogging resources
  - Set interrupt after specific period
  - Operating system decrements counter
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time



General system operation

- To improve the utilization led to the development of multiprogramming and timesharing.
  - Resources of the computers are shared among many programs and processes
- Sharing led to allow OS to have control over I/O to provide continuous, consistent and correct operation.
- Dual mode of operation is introduced.
- I/O instructions and instructions to modify memory management are privileged instructions.
  - HLT instruction is privileged
  - The instructions to turn-on and turn-off are privileged.
- The instructions to change user mode to monitor mode are privileged.
- User must ask the monitor to do I/O. Such a request is called **system call**.
- When a system call is executed it is treated by the hardware as a software interrupt.
- Control is passed to interrupt vector to a service routine to the OS by setting the mode bit to monitor mode.
- The OS examines the request , and passes necessary information and checks correctness and executes the request.
- It returns the control to the statement after the system call.

General System Architecture…



Outline

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

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 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)

- **1. 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

2.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 – by OS or applications
  - 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

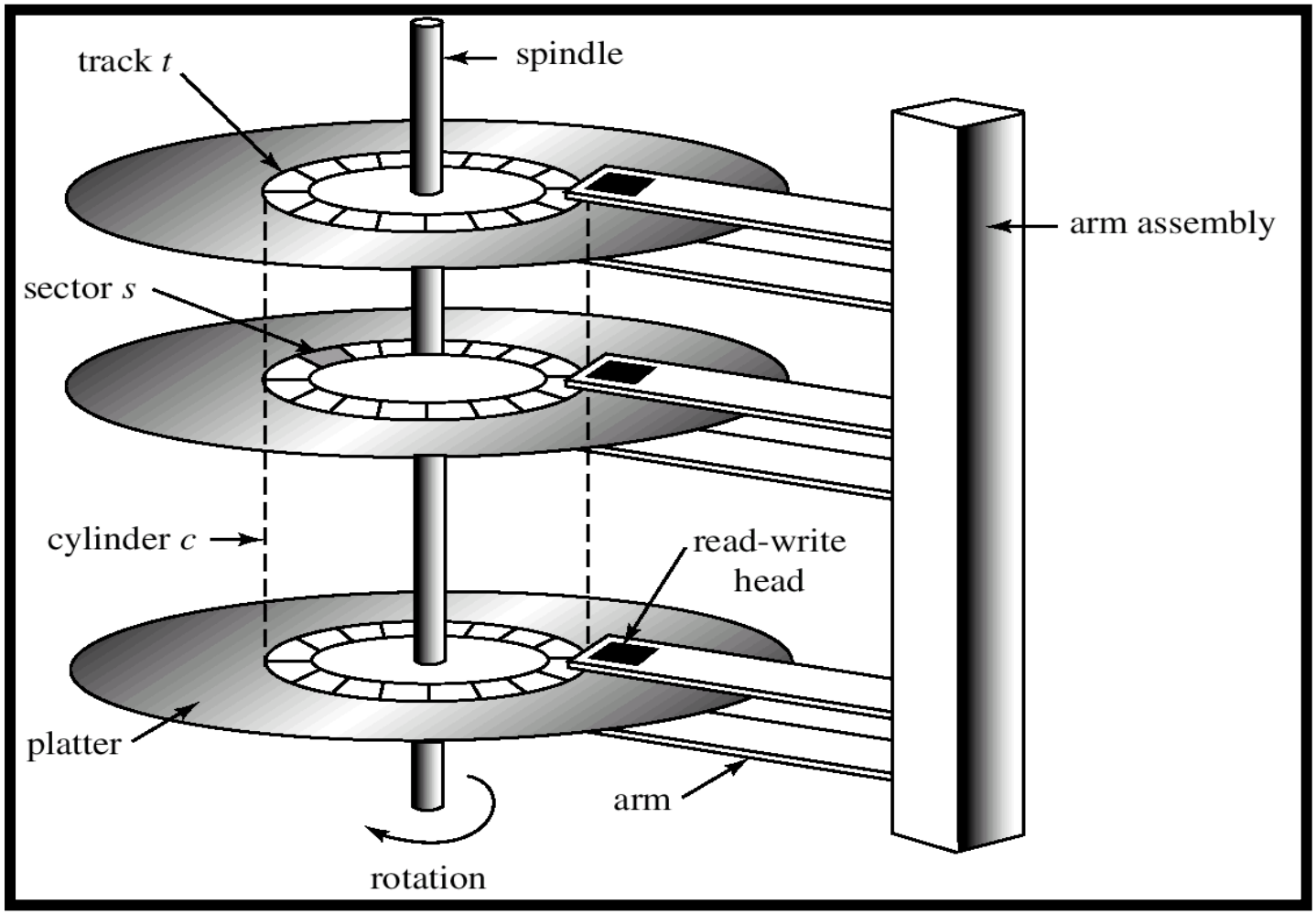
- I/O subsystem responsible for
  - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
  - General device-driver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

Storage Structure

- Main memory – only large storage media that the CPU can access directly
  - **Random access**
  - Typically **volatile**
- 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

Moving-Head Disk Mechanism

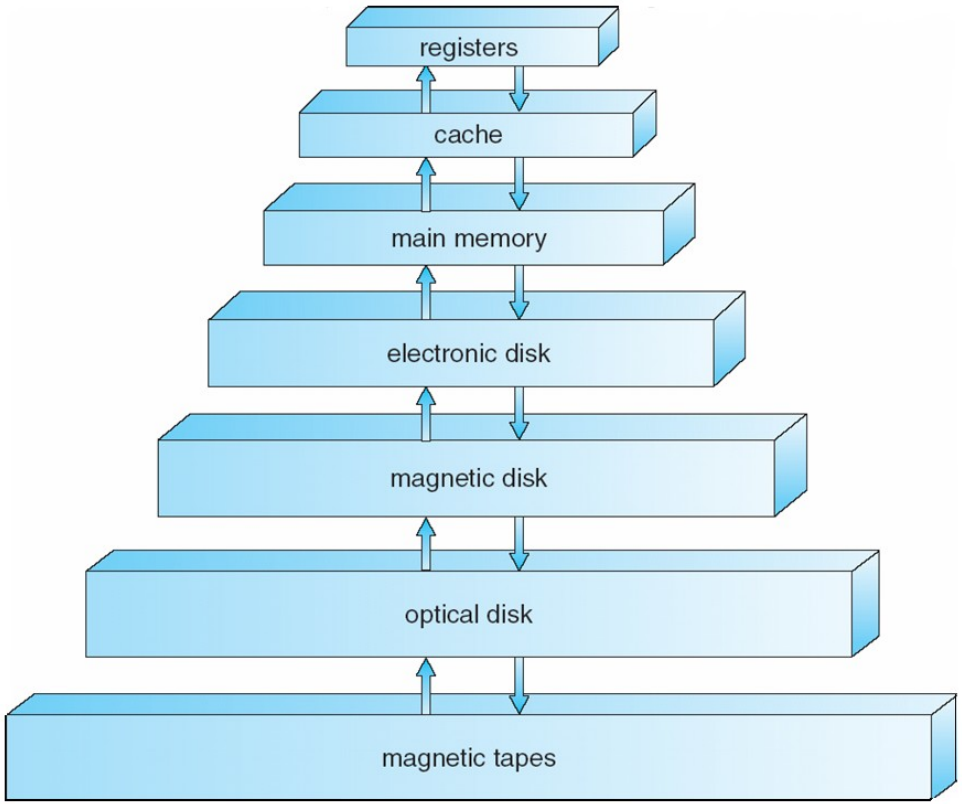


Storage Hierarchy

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

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

Storage-Device Hierarchy



- Increase in Latency
- Decrease in cost

- Decrease in Latency
- Increase in cost

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 is an 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

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

- Lists
  - Single, double, circular
- Stacks
  - Last in first out
- Queues
  - First in first out
- Trees
  - Binary search tree, balanced tree (B-tree)
- Hash functions and maps
  - Input to hash function is data and output is numeric value, which can be used as an index.
- Bitmaps
  - Status of n items.
  - Example: Consider 1000 disk blocks. We use 1000 bits to know the status. Each of 1000 bits represents the status of one block (occupied (1) or unoccupied (0)).

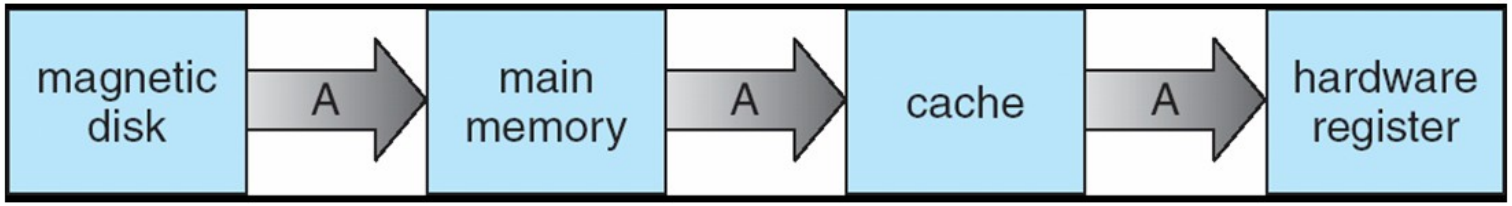
Computer Architectures

Multi-processor systems

- Also known as parallel systems or tightly coupled systems
- Advantages
  - Increased throughput:
    - Number of tasks finished per unit time.
  - Economy of scale
    - Total cost is less due to sharing of resources, like disk, mass storage and power supplies.
  - Increased reliability
    - One processor will not halt the system
    - Graceful degradation
      - The service depends on the available hardware
    - Fault tolerant
      - The service level does not decrease. Duplicate components.
        - HP nonstop

Migration of Integer A from Disk to Register

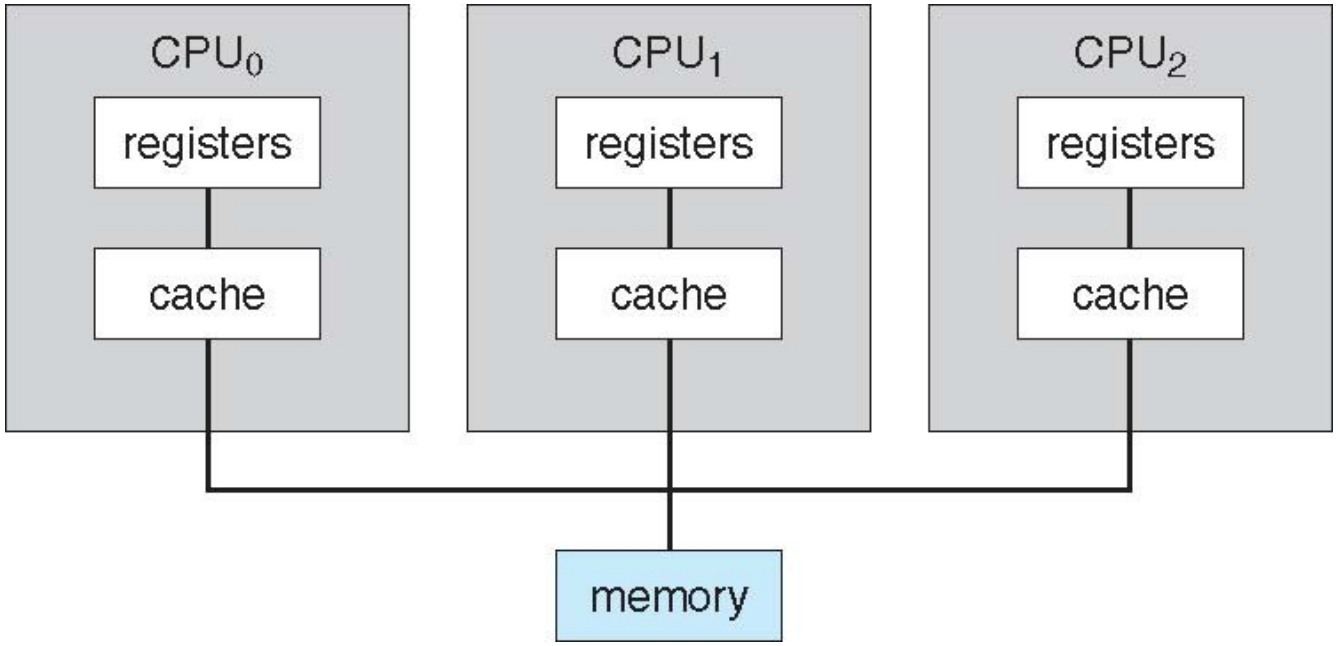
- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy



- Multiprocessor environment must provide cache coherency in hardware such that all CPUs have the most recent value in their cache

- Distributed environment situation even more complex
  - Several copies of a datum can exist
  - Various solutions covered in Chapter 17

Symmetric Multiprocessing Architecture

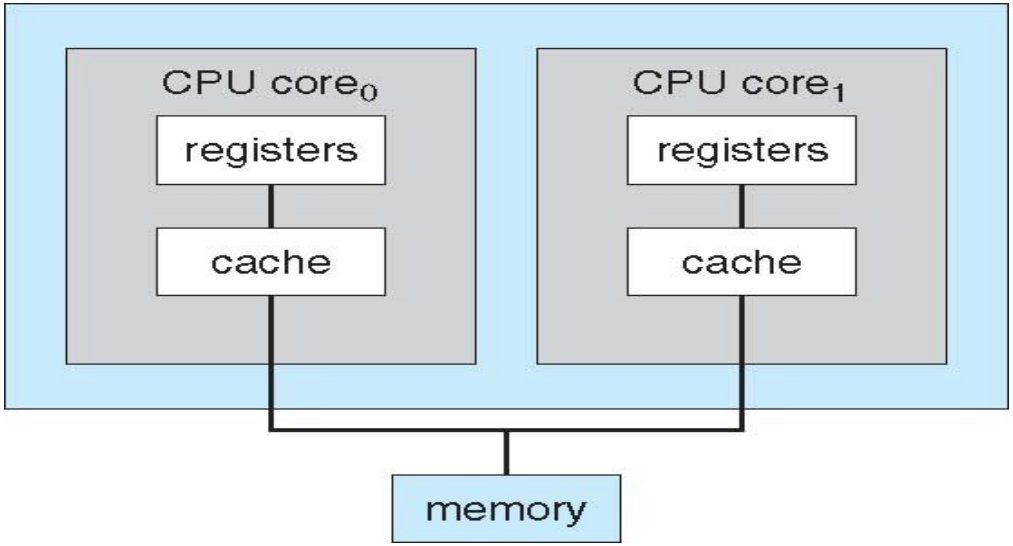


- Symmetric multiprocessing: each processor performs all tasks
  - Solaris

- Asymmetric multiprocessing: each processor is assigned a specific task

- Memory access model is changed from Uniformed memory access (UMA) to Non uniformed memory access (NUMA):

Recent Trend: A Dual-Core Design



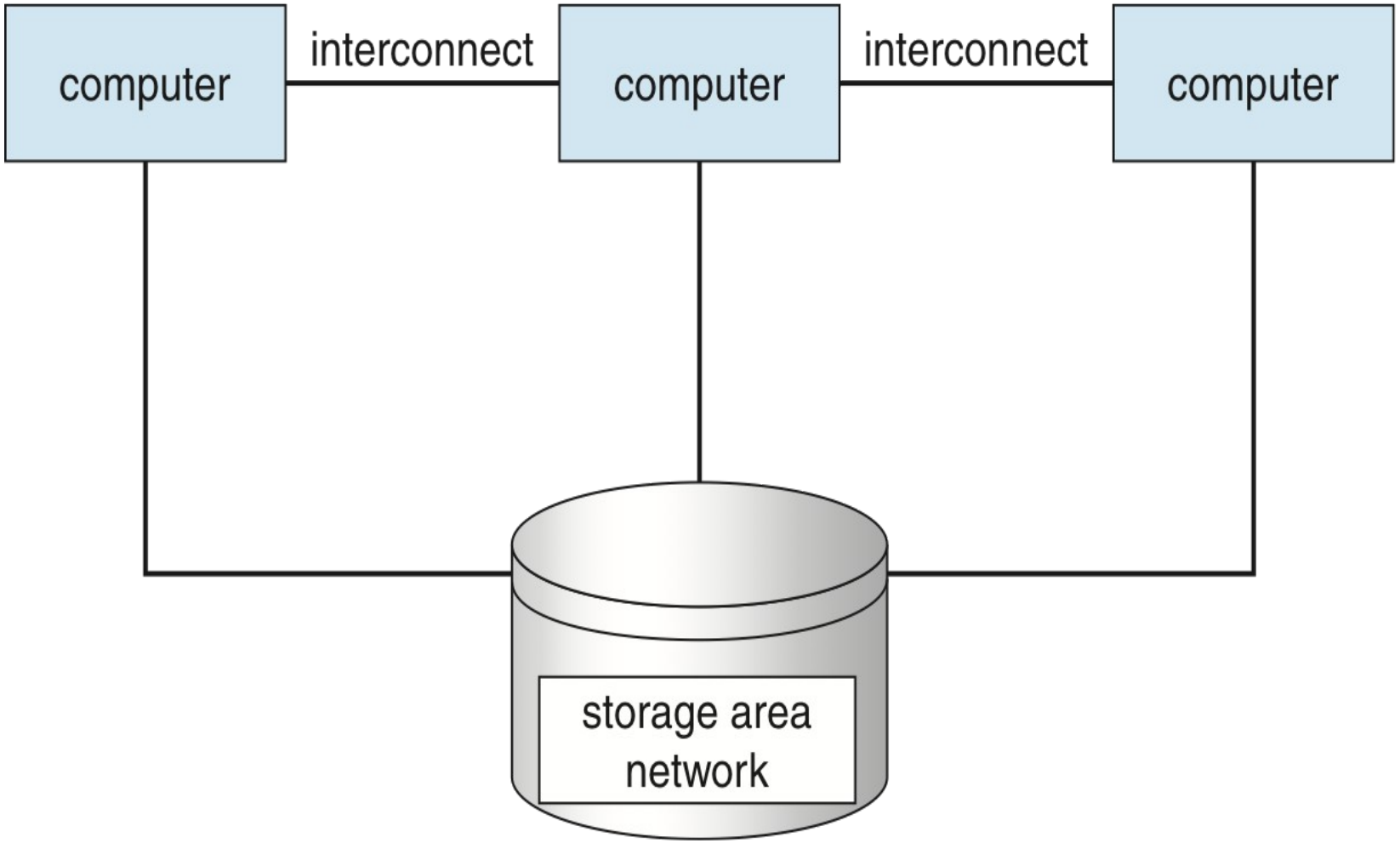
- Including multiple computing cores on a single chip.
  - Less power, communication faster
- However, it puts a pressure on OS designers and application programmers
- Blade servers:
  - Each blade server runs the OS independently
  - Multiple processor boards are placed in the same chasis.

Blade Servers

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**

Clustered Systems



Outline

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

Computing Environments

Distributed Computing

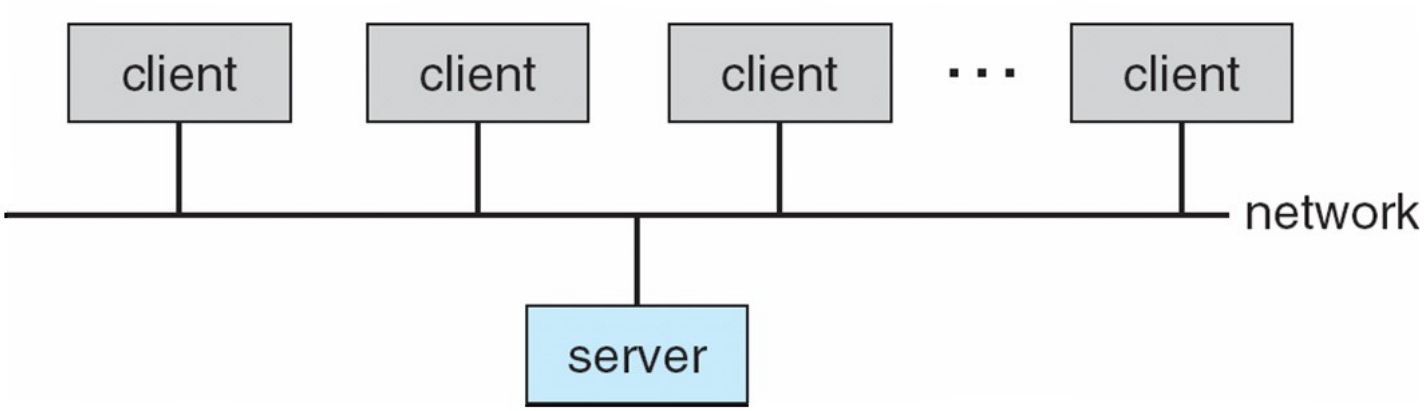
- Collection of separate, possibly heterogeneous, systems networked together
  - Network is a communications path
    - Local Area Network (**LAN**)
    - Wide Area Network (**WAN**)
    - Metropolitan Area Network (**MAN**)
- **Network Operating System** provides features between systems across network
  - Communication scheme allows systems to exchange messages
  - Illusion of a single system

Network 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

Client-server computing

- 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



Real-time embedded systems

- Real-time embedded systems most prevalent form of computers
  - Vary considerable, special purpose, limited purpose OS, **real-time OS**
- Multimedia systems
  - Streams of data must be delivered according to time restrictions
  - Quality of service
- Handheld systems
  - PDAs, smart phones, limited CPU, memory, power
  - Reduced feature set OS, limited I/O

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*

Virtualization

- It allows one operating system to run as an application of other operating system
- Emulation

Cloud Computing

- Delivers computing, storage, and applications as a service across network
- Employs thousands of servers and network

Web-Based Computing

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

Open-Source Operating Systems

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

End of Chapter 1