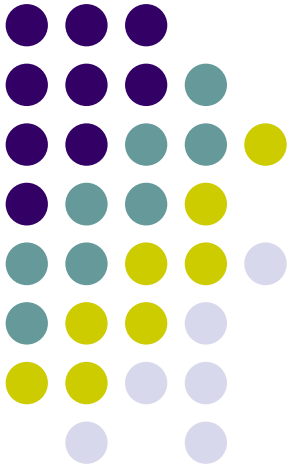
Chapter 1: Introduction

WHAT'S AHEAD:

- What Operating Systems Do
- Computer System Organization
- Computer System Architecture
- Operating System Structure
- Operating System Operations
- Major functions of OS
- Kernel Data Structures
- Computing Environments

WE AIM:

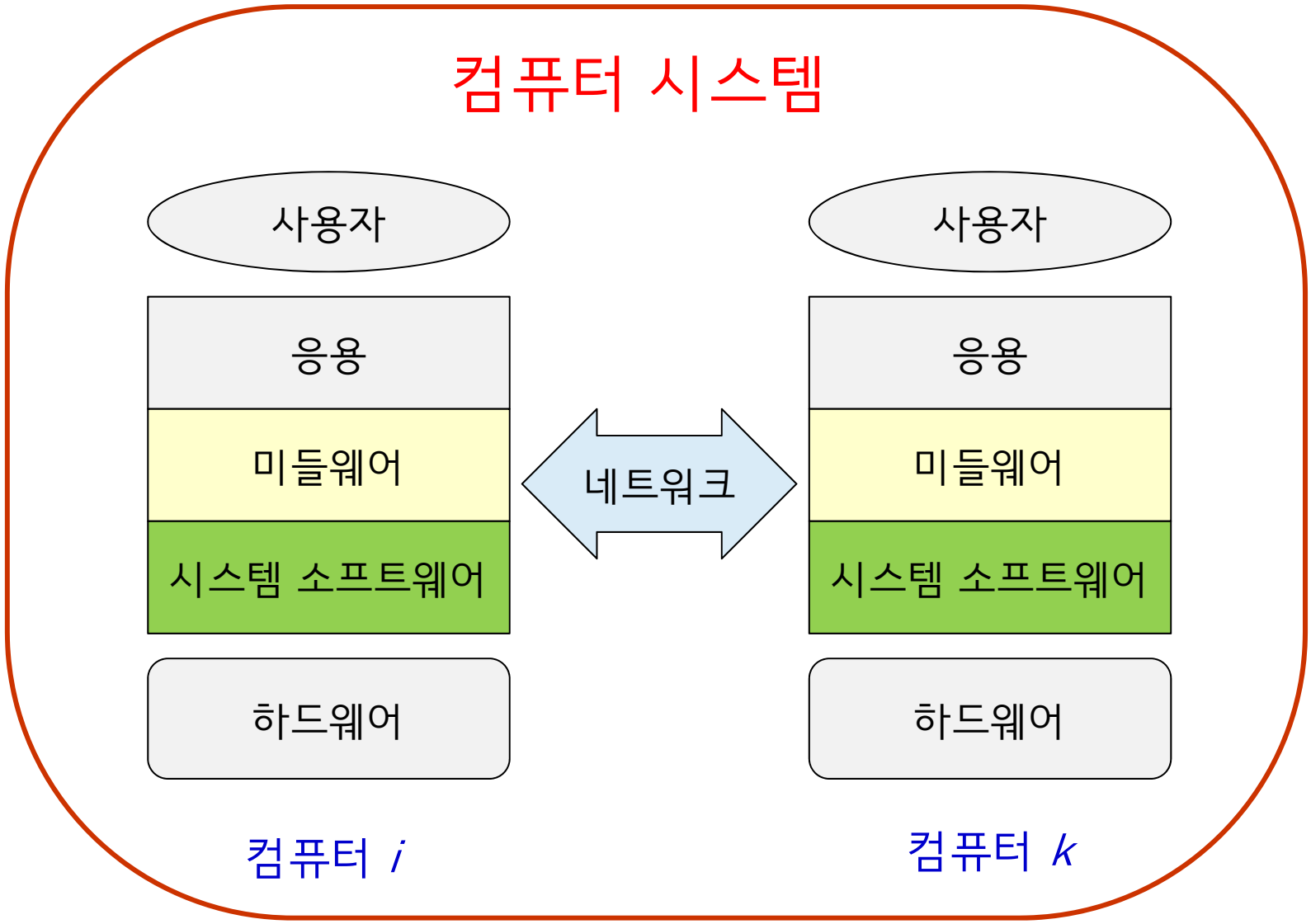
- To describe the structural aspect of computer systems
- To preview the major components of OSs
- To overview various types of computing environments



Note: These lecture materials are based on the lecture notes prepared by the authors of the book titled *Operating System Concepts*, 9e (Wiley)

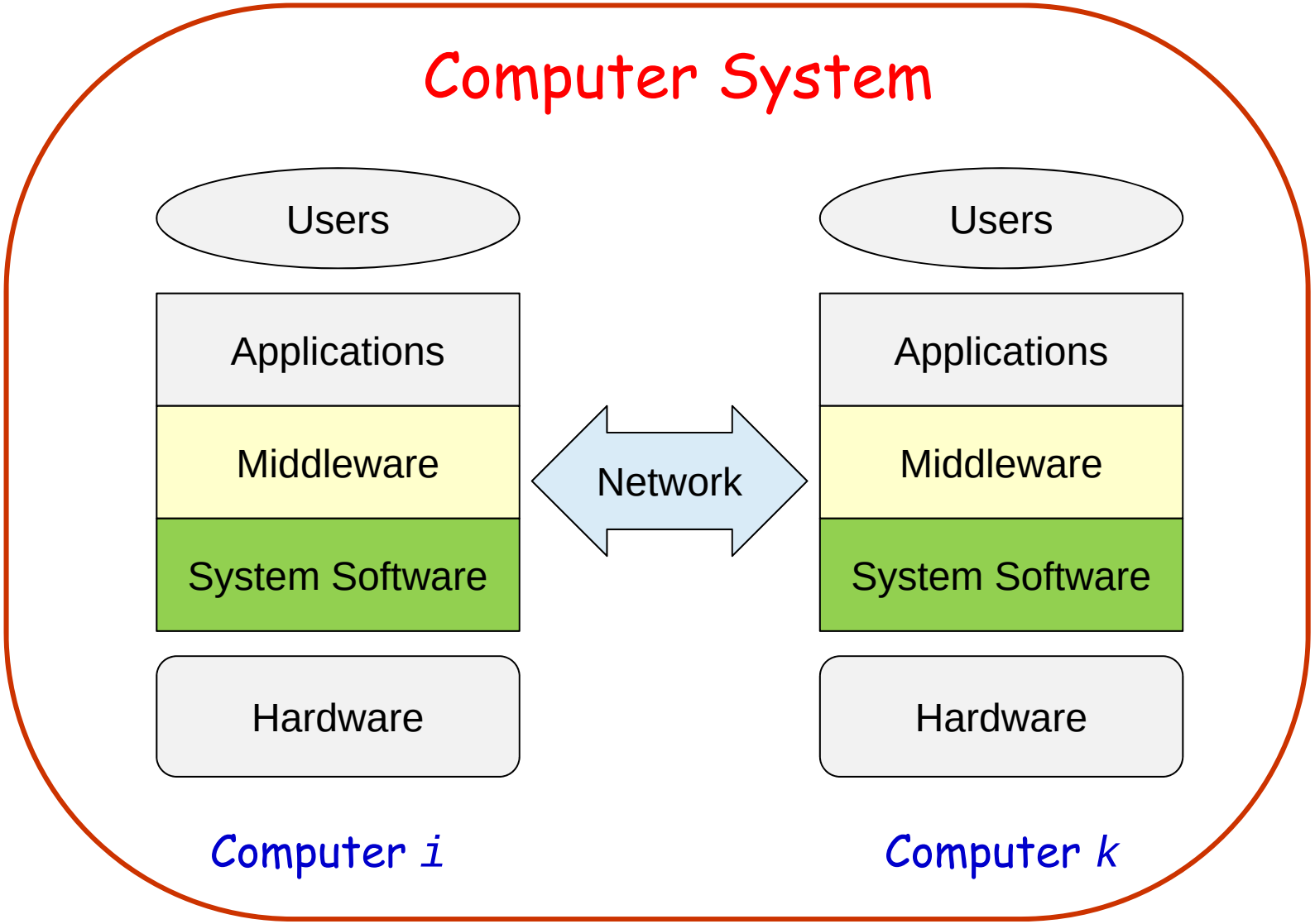
핵심•요점

컴퓨터 시스템의 구성



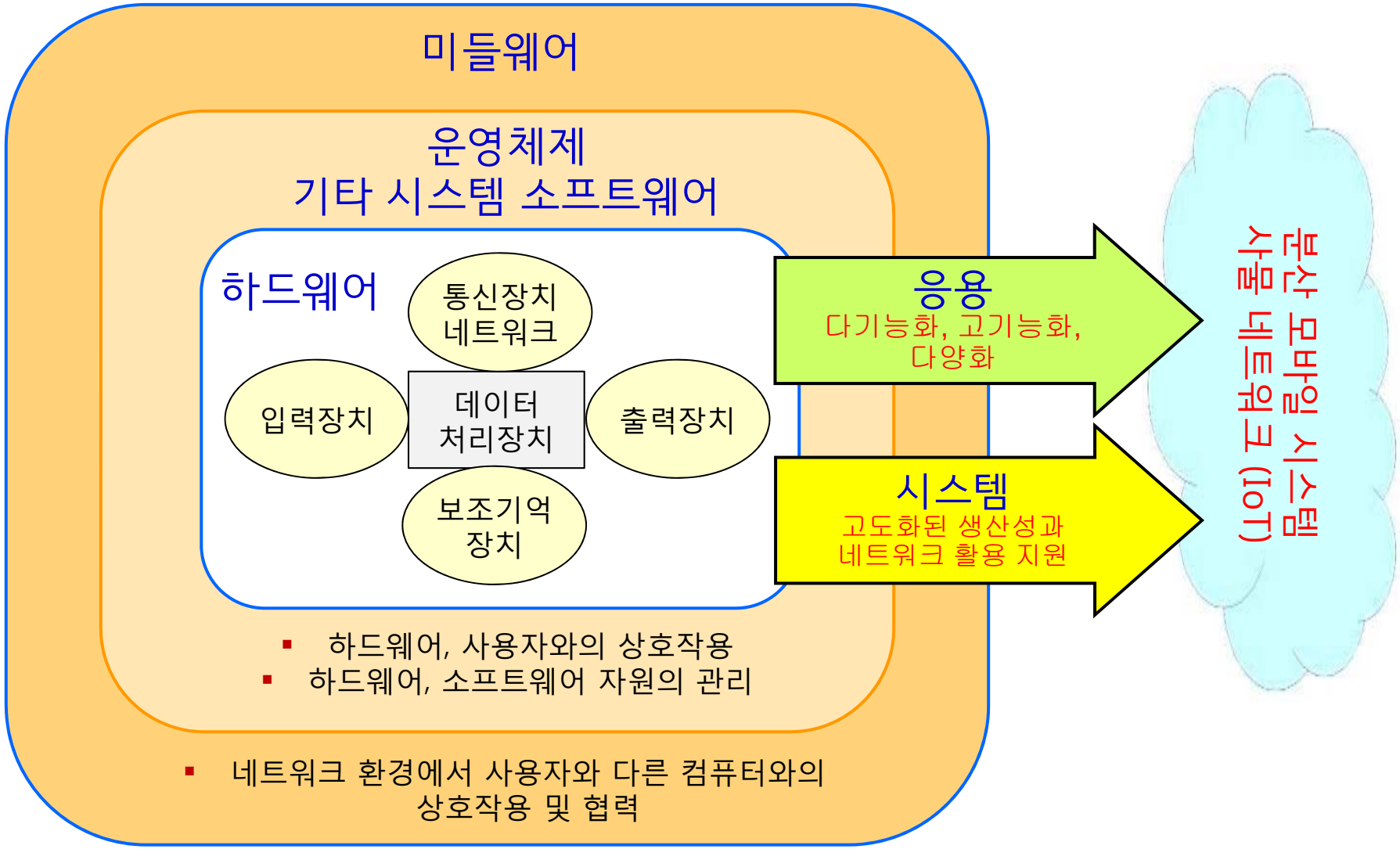
Core Ideas

How Computer Systems Look



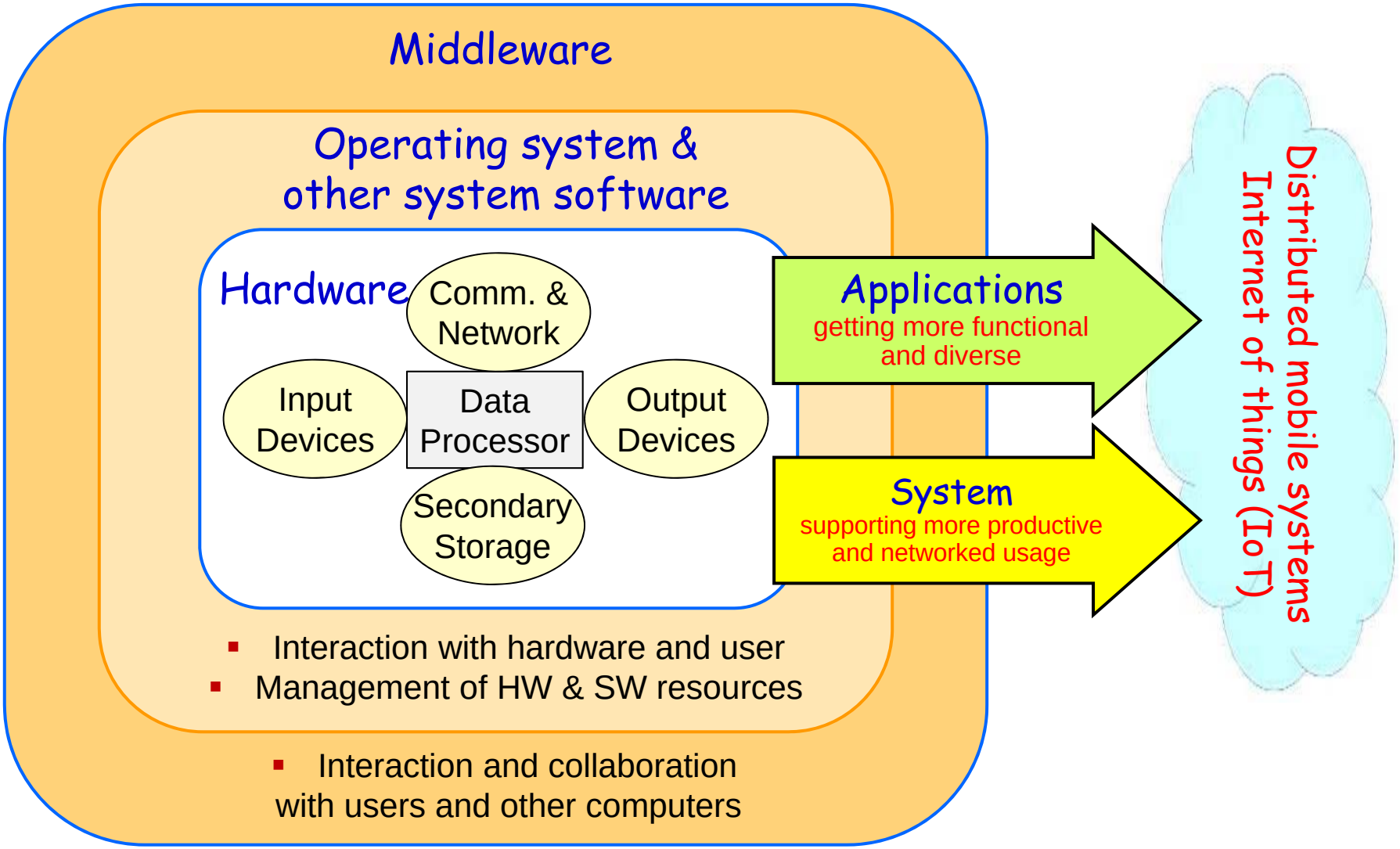
핵심•요점

컴퓨터 시스템의 진화



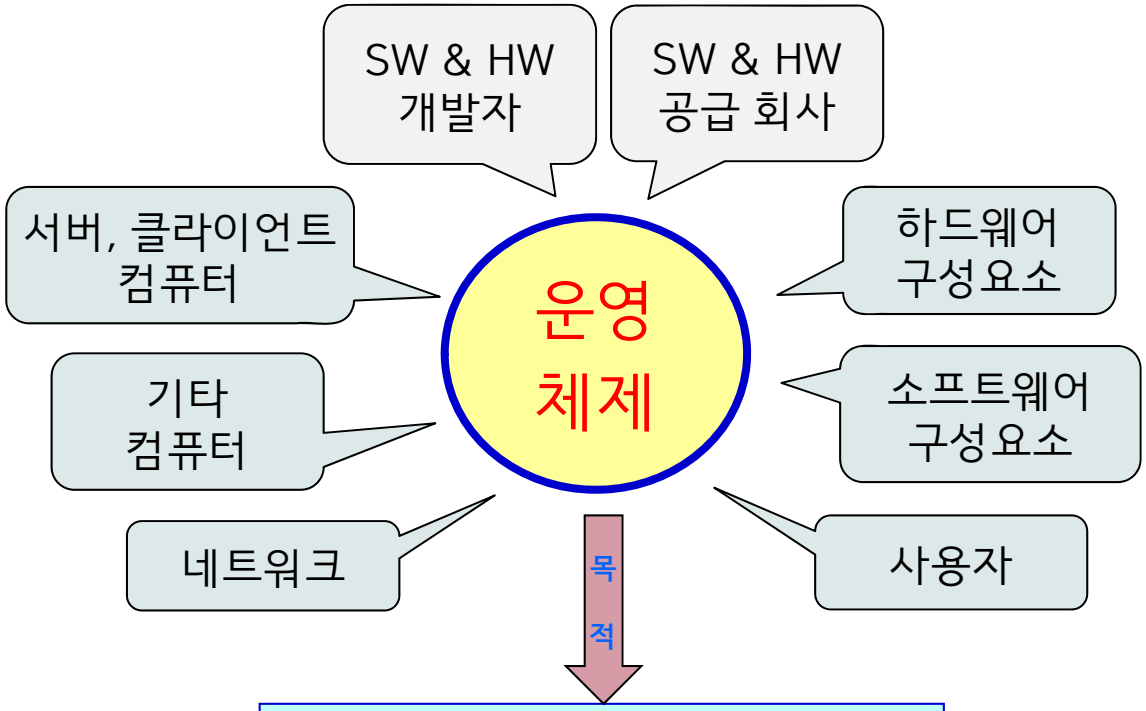
Core Ideas

Evolution of Computer Systems



핵심•요점

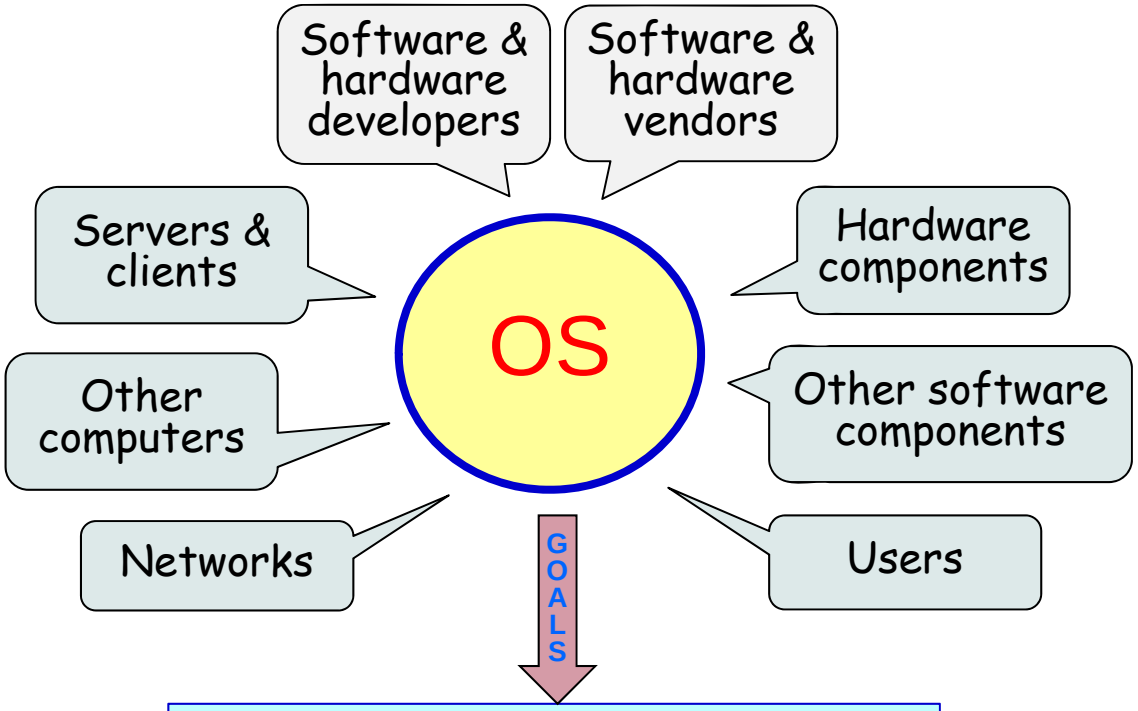
운영체제 목적 및 생태환경



- 하드웨어의 효율적인 **제어**
- 하드웨어/소프트웨어 자원의 효율적인 **관리**
- 다양한 사용자와 외부 세계와의 적절한 **상호작용**

Core Ideas

OS Goals & Its Ecosystem



- To effectively ***control*** hardware components
- To efficiently ***manage*** hardware and software resources
- To adequately ***interact*** with a broad range of users and external world

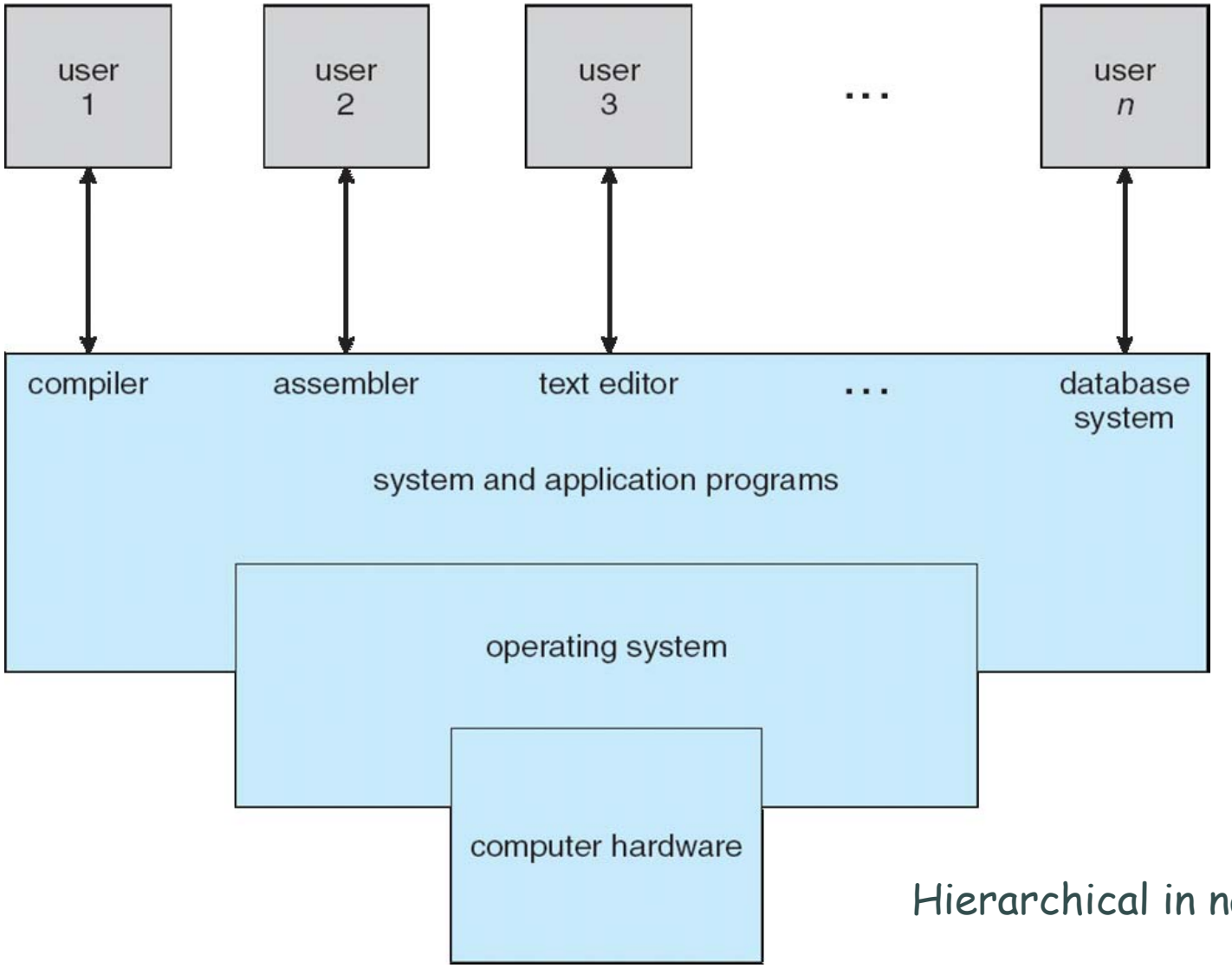
What is an Operating System?

- 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
- Also, OS aims to
  - Please end users, as many as possible
  - Please system managers
  - Please software developers

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
    - In many cases, includes middleware functionalities
  - System and application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, video games, compilers, web browsers, database systems
  - Users
    - People, *machines*, other computers

Four Components of a Computer System



Hierarchical in nature

What Operating Systems Do

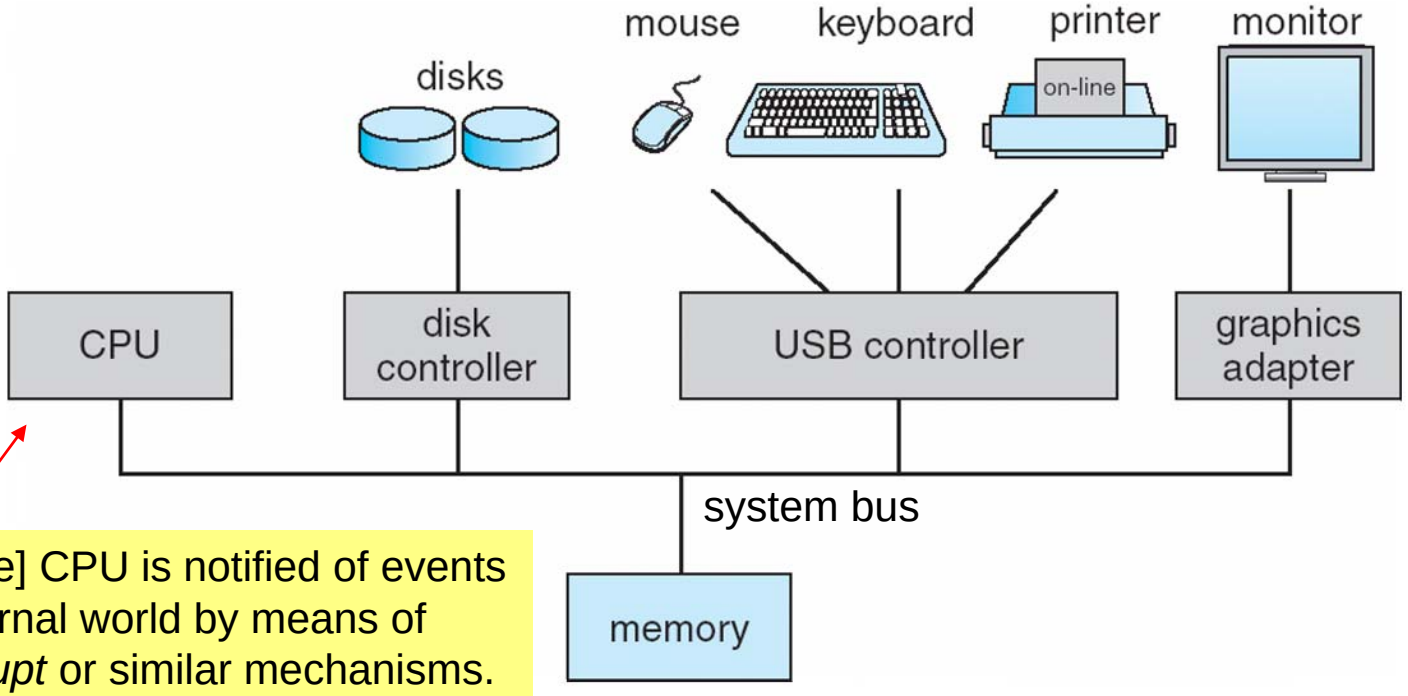
- Depends on the point of view
- Users want convenience, ease of use
  - Don't care about resource utilization
- But shared computers such as mainframes or minicomputers must keep all users happy
- Users of dedicated systems such as workstations have dedicated resources but frequently use shared resources 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 Definition

- OS is a resource manager
  - Allocates and manages all hardware and software resources
  - Decides between conflicting requests for efficient and/or fair resource use
  - Allows users and application programs to make use of resources
- OS is a control program
  - Controls execution of programs to prevent errors and improper use of the computer
- In computer business, no universally accepted definition
  - Usually, include *middleware* functionalities, such as networking, multimedia, GUI, etc.

Computer System Organization

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



[Hardware] CPU is notified of events from external world by means of the *interrupt* or similar mechanisms.

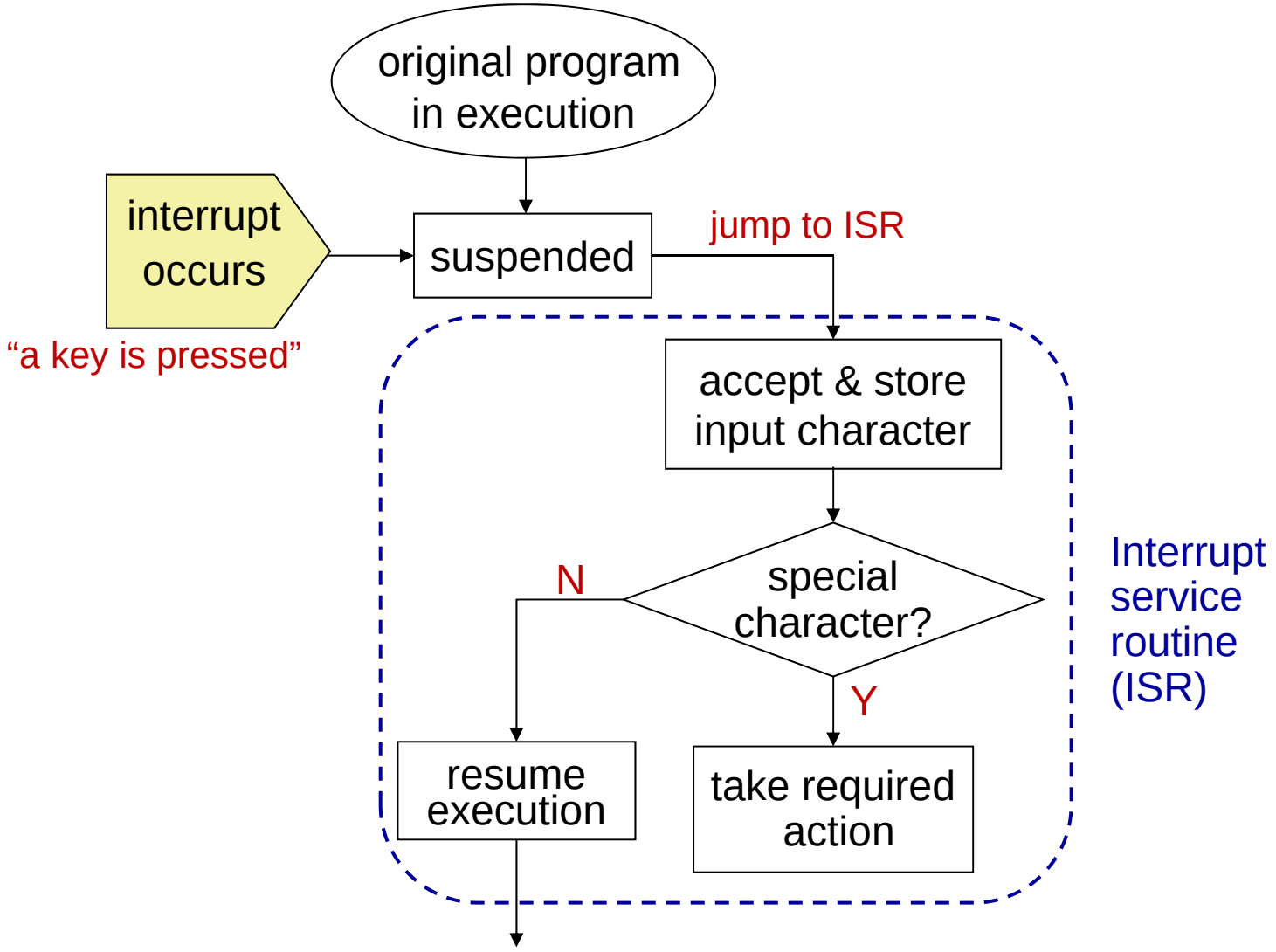
Interrupts

- (Hardware) Interrupt
  - The (hardware) action that an I/O device performs onto CPU to notify an event
  - The primary mechanism by which OS interacts with computer hardware
  - Invisible to users, but drives OS and program executions
  - Most important mechanism for OS implementation
- Interrupt service routine (ISR)
  - Upon occurrence of interrupt, CPU automatically execute ISR to service that interrupt
  - Each interrupt usually has its own ISR
- Software interrupt
  - Invoked by a machine instruction or by the result of execution of an instruction
  - Serviced similarly to the hardware interrupt
  - Caused either by an error or a user request
  - Examples: trap, exception

Interrupt Processing

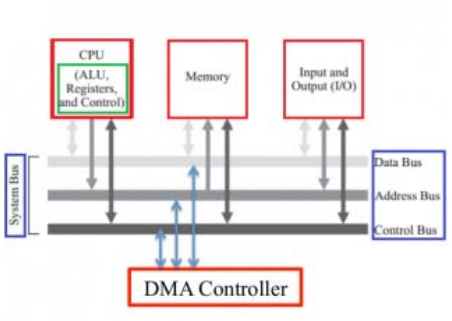
- If an interrupt occurs
  - Saves the program counter (that is, return address) of the interrupted process
  - Transfers control to the ISR, i.e., jump to the ISR
- Upon completion of ISR
  - Resume the execution of interrupted process by restoring the PC
  - Or, return to the location or a program that OS designates
- Examples
  - An event occurs at an I/O device, such as keyboard, mouse, disk, timer, etc.
  - Abnormal, critical, urgent event occurs, such as power outage, component failure, etc.
  - Illegal instructions, such as non-existing instruction, divide-by-zero, etc.

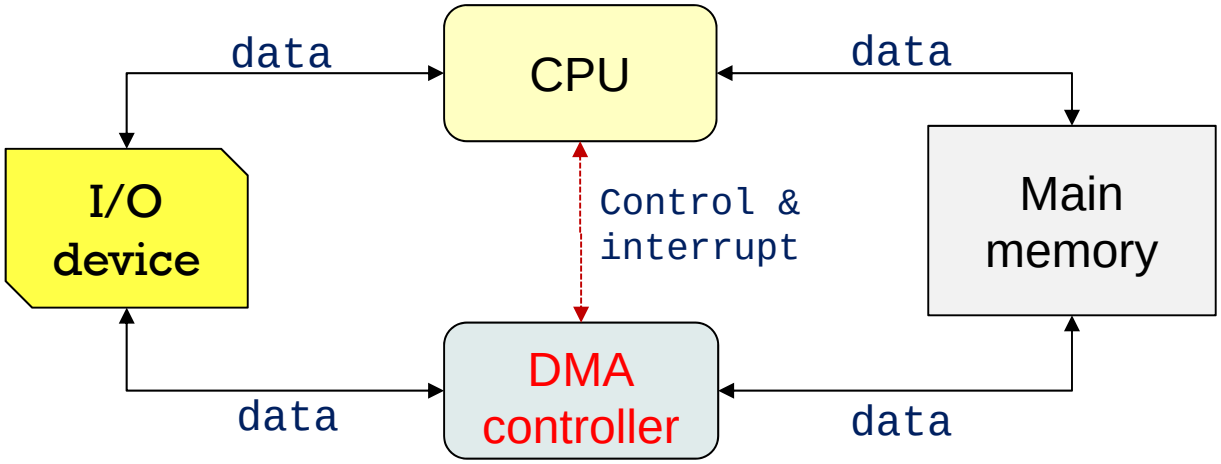
Interrupts – *Keyboard Example*



Direct Memory Access (DMA)

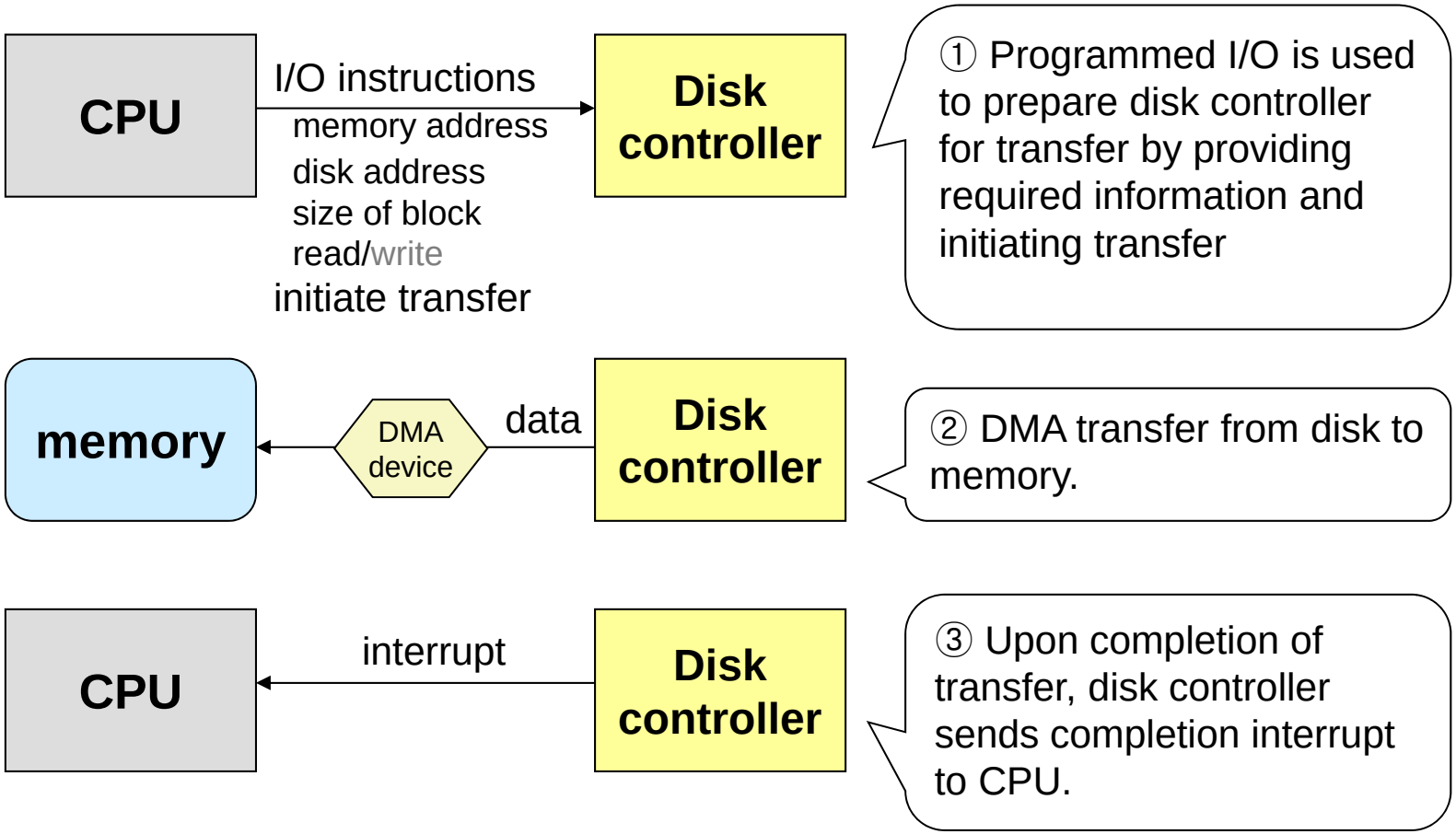
- Used for high-speed I/O devices able to transmit information at close to memory speeds
- DMA controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
  - Only one interrupt is generated per block, rather than the one interrupt per byte





DMA Operations – *An Example*

- Moving data from disk to the main memory



I/O Structure

After I/O starts

1. Control returns to user program only upon I/O completion (Synchronous)
   - Wait instruction idles the CPU until the next interrupt
   - At most one I/O request is outstanding at a time, no simultaneous I/O processing

2. Control returns to user program without waiting for I/O completion (Asynchronous)
   - System call – request to the OS to allow user to wait for I/O completion (The user is forced to wait in the wait queue)
   - Device-status table contains entry for each I/O device indicating its type, address, and state
   - OS indexes into I/O device table to determine device status and to modify table entry to include interrupt

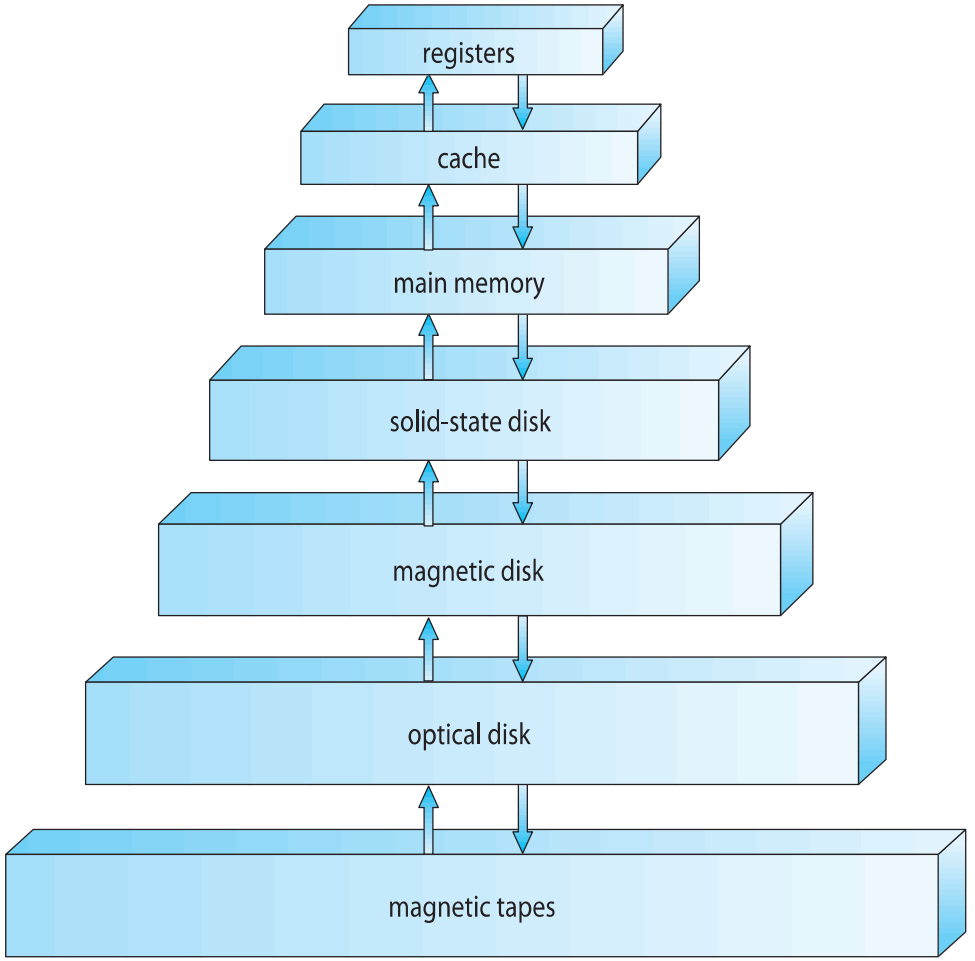
Storage Structure

- 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
  - Typically realized by magnetic disks (or hard disks) and SSDs
- Magnetic disks – large capacity, nonvolatile
  - 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 (SSDs) – faster than magnetic disks, nonvolatile
  - Currently, based on flash memories, especially NAND flash
  - Becoming more popular

Storage Hierarchy

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

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



Caching

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache smaller than storage being cached
  - Cache management important design problem
  - Cache size and replacement policy

➢ *You must understand the difference between* caching *and* buffering.

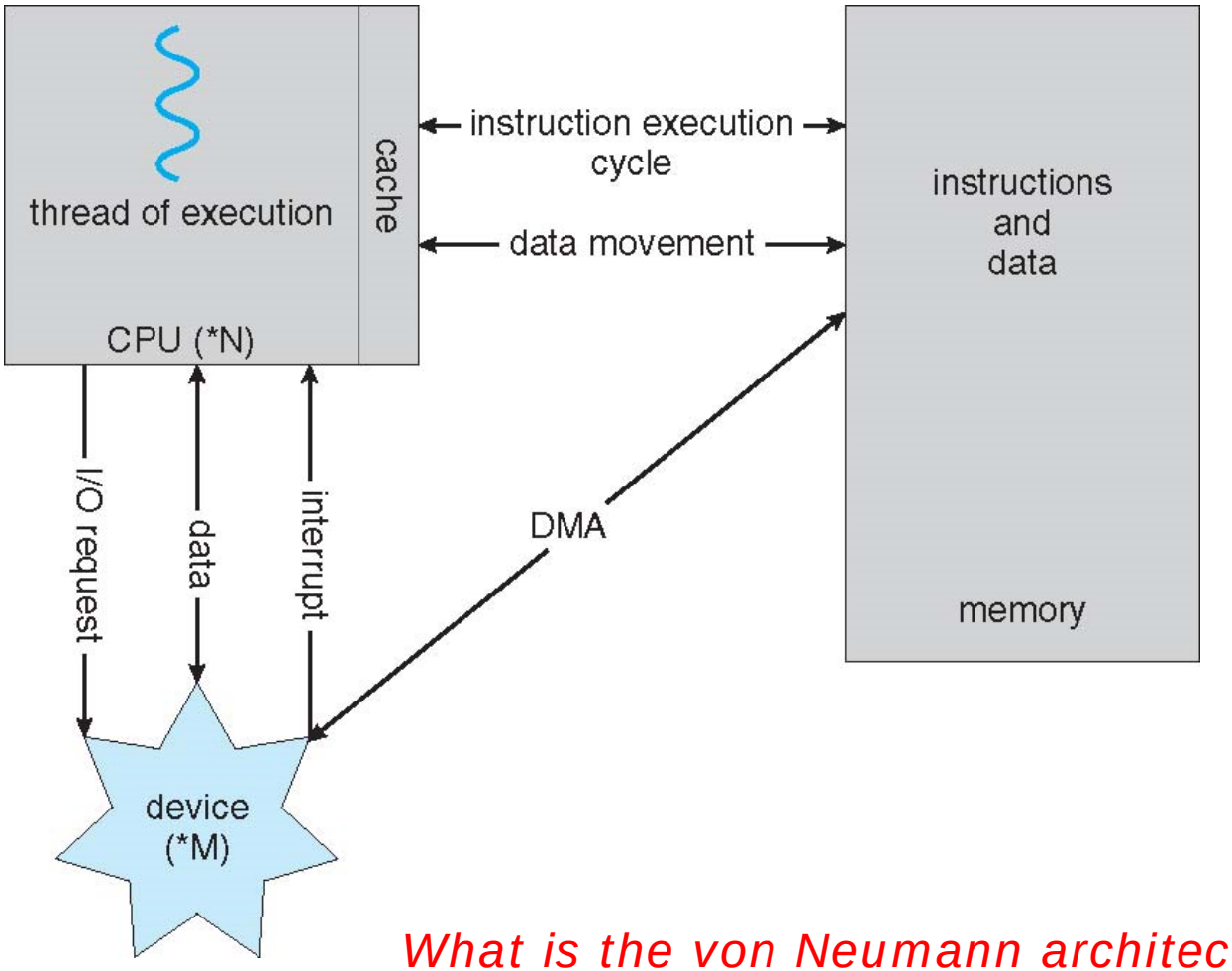
Units of Digital Information

- Basic units
  - bit
  - byte: 8 bits on most computers. may be of different size
  - word: the number of bytes that a computer processes physically or logically at a time, typically 4 bytes
- More units: K, M, G, T, P, E, Z, Y
  - powers of 1000 (SI unit) or powers of 2^{10}=1024 (binary unit)?
  - base 10: kilobyte (KB, 10^3) – megabyte (MB, 10^6) – gigabyte (GB, 10^9) – terabyte (TB, 10^{12}) – petabyte (PB, 10^{15}) – exabyte (EB, 10^{18}) – zettabyte (ZB, 10^{21}) – yottabyte (YB, 10^{24})
  - base 2: (use binary prefix) kibibyte (KiB, 2^{10}), mebibyte (MiB, 2^{20}), gibibyte (GiB, 2^{30}), tebibyte(TiB, 2^{40}), pebibyte (PiB, 2^{50}), exbibyte (EiB, 2^{60}), zebibyte (ZiB, 2^{70}), yobibyte (YiB, 2^{80})

  - http://en.wikipedia.org/wiki/Binary_prefix

- Usage
  - in computer hardware and software: interpret on a case-by-case basis, either base 10 or base 2
  - networking and communication: usually SI prefix (powers of 10^3)

How a Modern Computer Works



What is the von Neumann architecture?

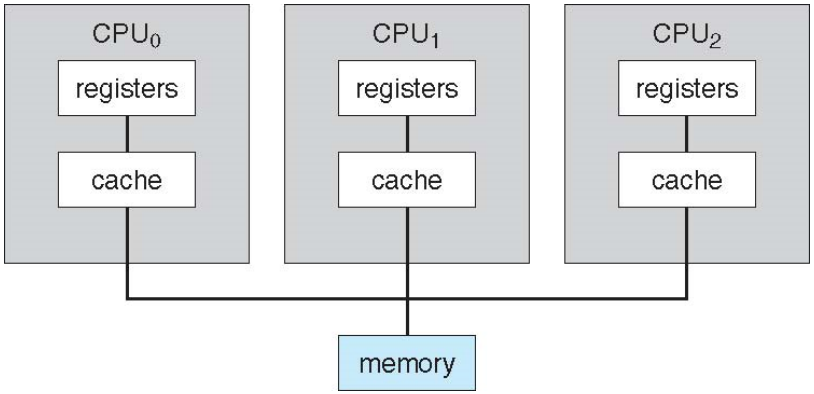
- code + data in memory - mixed
- memory locations – sequential addresses

Computer System Architecture

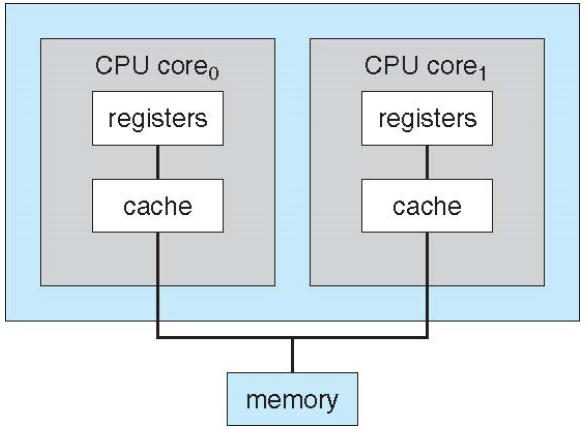
- Most systems use a single general-purpose processor (PDAs through mainframes)
  - Most systems have special-purpose processors as well
- Multiprocessor systems
  - Also known as parallel systems, tightly-coupled systems sharing the main memory
  - Advantages include:
    - Increased throughput
    - Economy of scale
    - Increased reliability – graceful degradation or fault tolerance
  - Two types:
    - Asymmetric Multiprocessing
    - Symmetric Multiprocessing
- Clustered systems
  - Multiple autonomic computers connected via a high-speed network
  - Loosely-coupled system usually sharing the secondary storage

Multiprocessor Systems

- Characteristics
  - CPUs communicate through shared data in the (common) memory
  - OS may run either on a single CPU or all CPUs
- Asymmetric multiprocessing (ASMP, AMP)
  - CPUs may be different in functionality or structure
  - Usually, only a single processor runs the OS and others run as slaves: master-slave relationship
- Symmetric multiprocessing (SMP)
  - All CPUs are treated equally
  - OS may run on any processors
- **Multicore**
  - A microprocessor containing multiple CPUs
  - Either SMP or ASMP



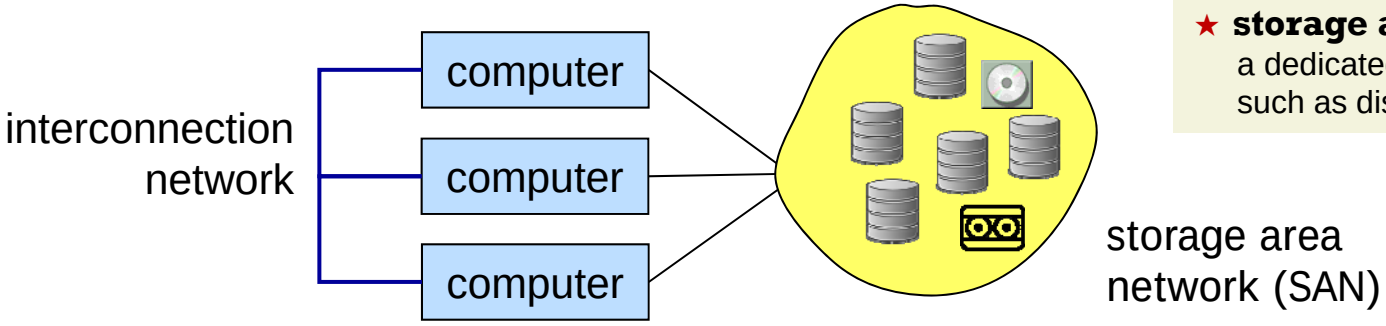
Symmetric multiprocessing architecture



Dual-core microprocessor

Clustered Systems

- Multiple autonomous 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
    - Useful for Internet servers, DB servers, graphics servers, etc.

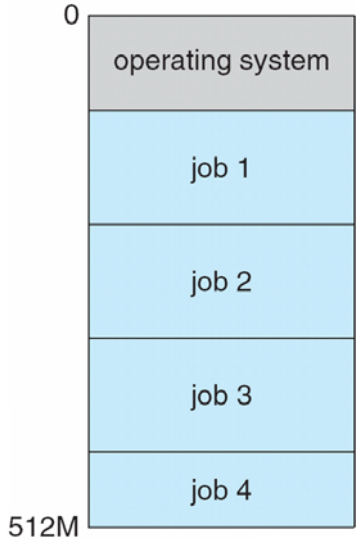


★ **autonomous system**:
a computer system with its own OS

★ **storage area network (SAN)**:
a dedicated network of storage devices such as disk arrays and tape drives

Operating System Structure & Operations

- Multiprogramming needed for efficiency
  - When it has to wait (for I/O for example), OS switches to another job
  - 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
  - Multitasking
    - Multiple tasks running on CPU during the same period of time
    - Based on the multiprogramming technique

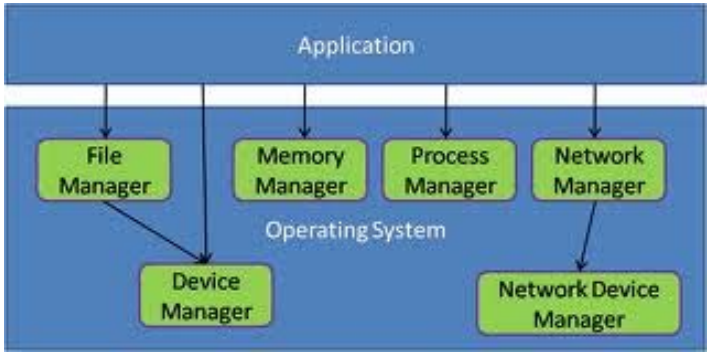


Memory layout for a multiprogrammed system

- Timesharing systems
  - CPU switches jobs regularly while each job is given a fixed time slot (say, 20 msec) in round robin fashion
  - Based on multiprogramming and multitasking methods
  - Provide the basis for interactive computing in multi-user environment
    - 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

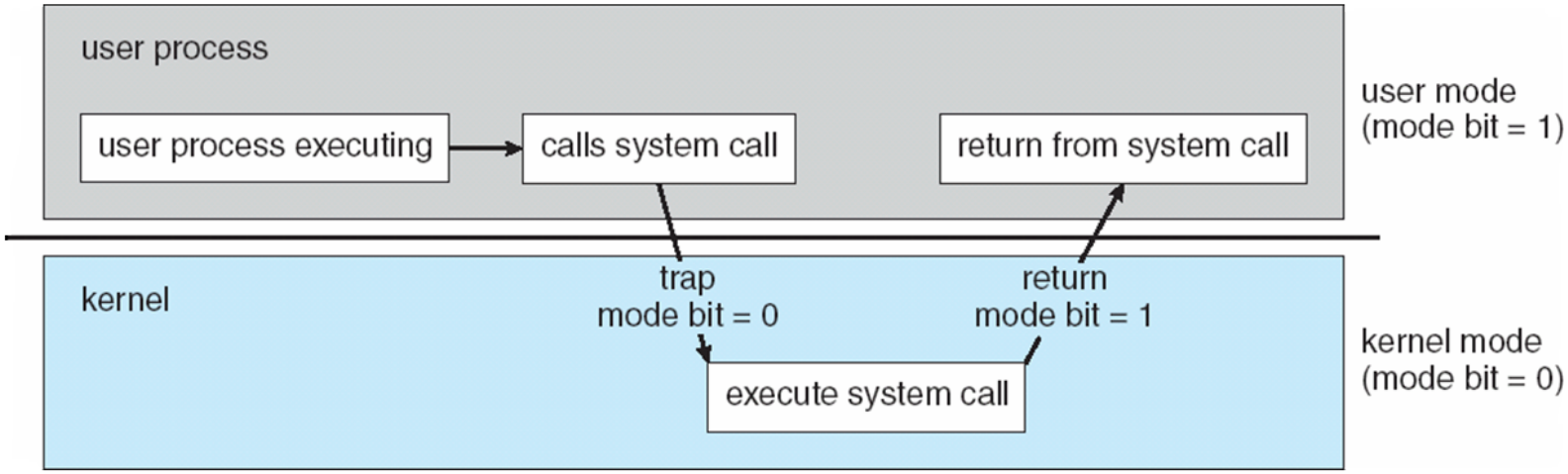
Operating System Operations

- Often driven by hardware interrupt mechanism
- Software error or request creates exception or trap, considered "software interrupt"
  - Division by zero, request for OS service, etc.
  - May be caused by other process problems including an infinite loop, and processes modifying each other or the OS
- Dual-mode operation allows OS to protect itself and other system components
  - Dual modes: 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

- Example: Timer to prevent infinite loop / process hogging resources
  - [User mode] Call a system service for checking infinite loop
    - System service: Set interrupt after specific period
  - [Kernel mode] (OS) sets the timer to a desired value
    - Hardware timer continues decrementing the timer value
    - When the value becomes zero, generate an interrupt
  - [Kernel mode] Set up the calling process to regain control, or terminate program that exceeds allotted time
  - [User mode] (*If allotted time is not exceeded*) Return from system call → resume the execution

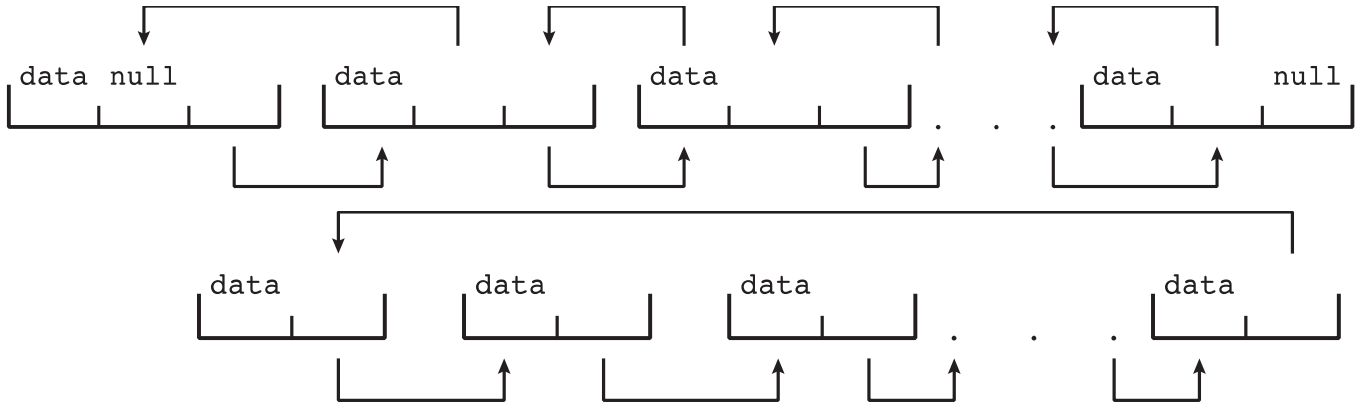


Major Functions of OS

- Process and thread management
  - A process is the object that executes a program
  - OS creates, runs, suspends and terminates a process
- Memory management
  - In von Neumann architecture, all instructions and data are located at main memory
  - Memory space are always short of the space needed for program execution → allocation and deallocation of memory
- Storage and I/O management
  - Abstracts physical storage properties to logical storage unit - file
  - General interface to I/O devices using device drivers
  - I/O memory management through buffering, caching or spooling
- System protection and security
  - Protection - controls access of processes or users to resources defined by the OS
  - Security – defenses the system against internal and external attacks

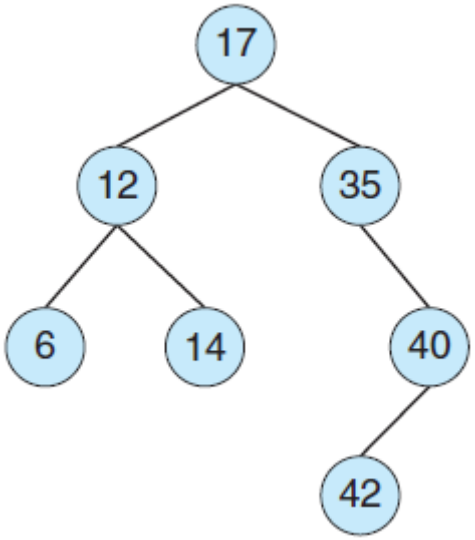
Kernel Data Structures

- Many similar to standard programming data structures
- Kernel characteristics that affect data structures
  - Dynamic – must accommodate current varying status and operations
  - Intensive search and scanning operations
  - Mostly reside in main memory
  - Consists of data and pointers (addresses)
- Stacks, queues, and lists
  - Stacks and queues implement LIFO and FIFO, respectively
  - Lists
    - Singly linked list, doubly linked list, circular linked list

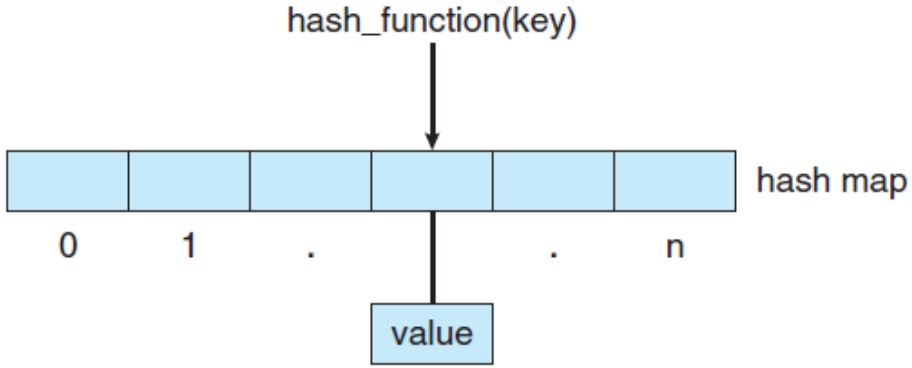


Kernel Data Structures

- Trees
  - General tree
  - Binary search tree
    left <= right
    - Search performance is O(n)
    - Balanced binary search tree is O(log n)



- Hash function and maps
  - can create a hash map



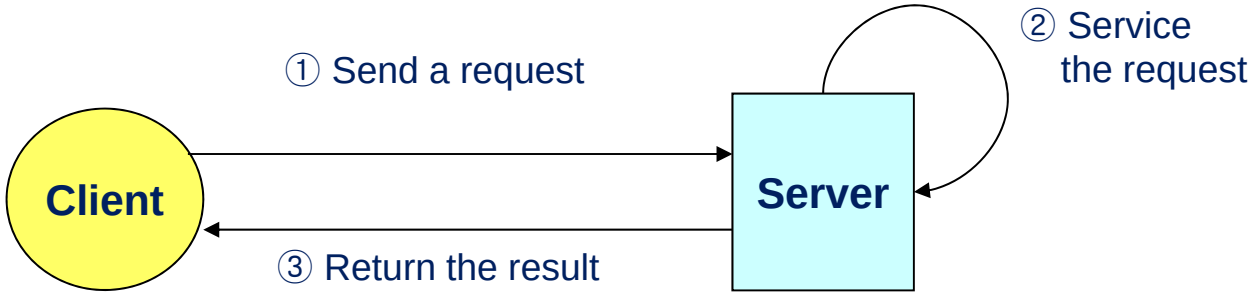
- Bitmap
  - String of n binary digits representing the status of n items
- Linux data structures
  - Defined in `include` files <linux/list.h>, <linux/kfifo.h>, <linux/rbtree.h>

Computing Environments

- Traditional: Stand-alone general purpose machines
  - "Centralized computing"
  - But blurred as most systems interconnect with others (i.e. the Internet)
- Distributed computing paradigm
  - Based physically on a collection of separate, possibly heterogeneous, autonomous systems networked together
  - Based logically on a client-server model
  - Computers communicate with each other using messages
  - Diverse types of approaches and implementations, such as Internet/Web computing, mobile computing, cloud computing, etc.

Computing Env. – Distributed System

- Client-server model
  - Basic model for distributed systems

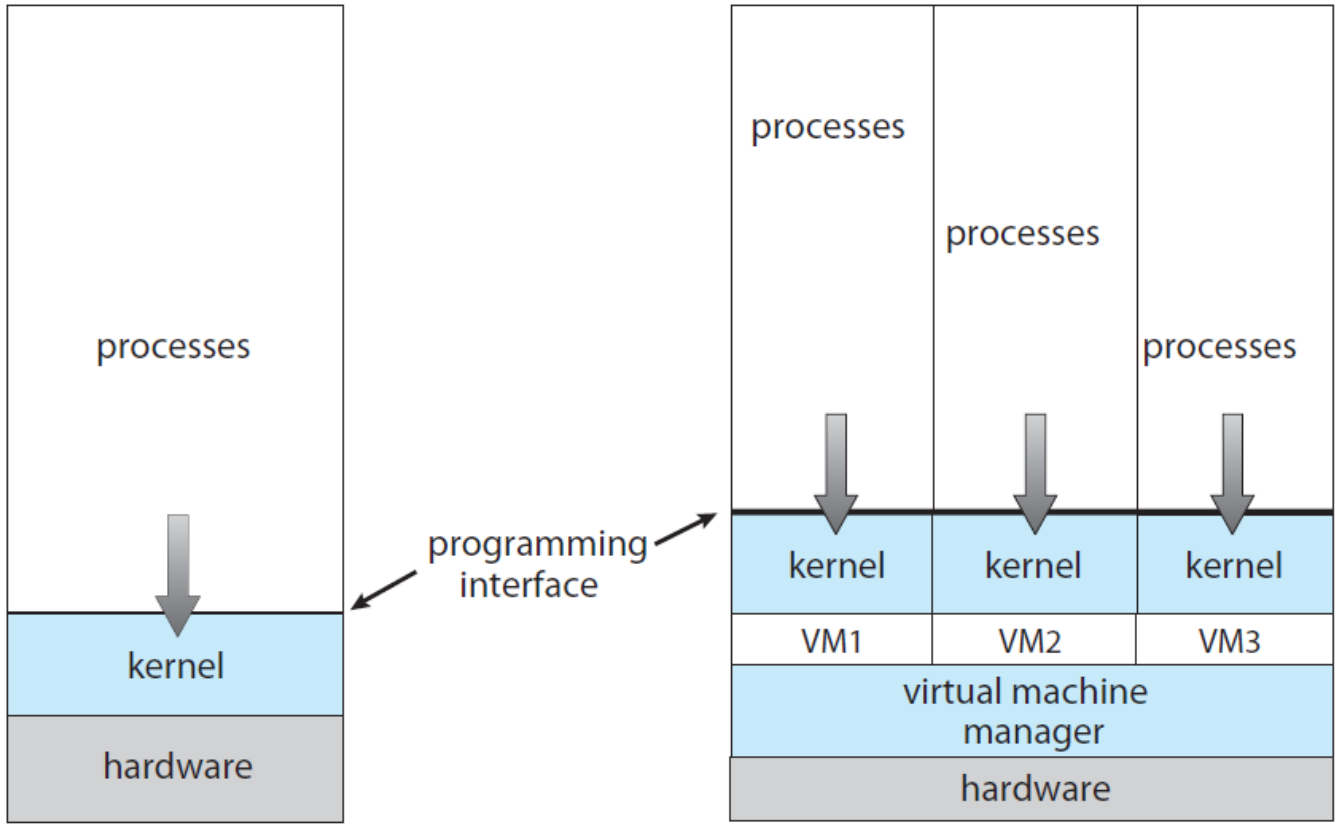


  - Implementations
    - Client: PCs, laptop computers, smartphones, tablets, etc.
    - Server: compute server, file server, Web server, etc.
    - Network: LAN(e.g., Ethernet, WiFi), WAN(e.g., Internet), etc.
- Services involving multiple servers
  - e.g. sets of data that are distributed across many servers

Computing Env. - Virtualization

- Allows OSs to run on another OS platforms
  - Vast and growing industry
  - Emulation used when source CPU type different from target type (i.e. PowerPC to Intel x86) – very slow, though
- [Hardware] Virtualization
  - Creation of a virtual machine that operates like a real computer with an OS
  - OS natively compiled for CPU, running guest OSs also natively compiled
  - Virtual machine manager [or monitor] (VMM) provides virtualization services
- Use cases involve laptops and desktops running multiple OSs for exploration or compatibility
  - Apple laptop running Mac OS X host, Windows as a guest
  - Developing apps for multiple OSs without having to have multiple systems
  - Executing and managing compute environments within data centers or cloud servers

Computing Env. – Virtualization Model



(a)

Without virtualization

(b)

VMware: Virtual machine manager(VMM) may run with or without the host OS

Computing Env. – Real-Time Embedded Systems

- Real-time embedded systems most prevalent form of computers
  - Applications: diverse, special purpose, limited purpose
  - Examples: robots, mobile phones, guided missiles, spacecrafts, all sorts of machines, and many other special computing environments as well
  - Some have OSs called real-time OS or RTOS, some perform tasks without a full-fledged OS
- Real-time OS must be designed to meet time constraints, such as deadline and period
  - Temporal correctness in addition to logical correctness
  - Correct operation only if time constraints are met

Summary

- 운영체제(OS)
  - 정의: 자원의 관리, 프로그램 실행의 제어
  - 기능: 편리한 사용자 인터페이스, 효율적인 자원의 관리
- 컴퓨터시스템 구조
  - 인터럽트(interrupt)
  - DMA
  - 저장장치의 계층적 구성
  - 저장장치의 계층 구조에서 캐싱(caching)의 역할
- 컴퓨터시스템 구성– CPU 개수
  - 프로세서 개수: =1 or >1 ? (multiprocessors)
  - 컴퓨터 개수: =1 or >1 ? (클러스터형, 분산형)
  - 마이크로프로세서 칩 속의 CPU 개수: =1 or >1 (멀티코어)
- 운영체제(OS) 구조
  - 멀티프로그래밍: why?, how?, 운영체제에서의 의의
  - 시분할시스템

- Operating systems (OSs)
  - Definition: resource manager, control program execution
  - Functions: user-friendly interface, efficient resource management
- Computer system organization
  - Interrupt
  - DMA
  - Storage hierarchy
  - Role of caching in the storage hierarchy
- Computer system - # CPU
  - # processors: =1 or >1? (multiprocessors)
  - # computers: =1 or >1? (clustered, distributed)
  - # CPU in a microprocessor chip: =1 or >1 (multicore)
- OS structure
  - multiprogramming: why?, how?, significance in OS
  - timesharing system

Summary (Cont.)

- 운영체제의 동작
  - 소프트웨어 실행에 따른 예외적인 상황에 대한 처리
  - 사용자 모드 vs. 시스템 모드 (혹은 커널 모드)
- 운영체제의 기능적/관리적 구성 요소
  - 프로세스, 메모리, 저장장치, 입출력(I/O), utility, 등
- 시스템 보호 vs. 시스템 보안
- 커널 자료구조
- 컴퓨팅 환경: 분산 컴퓨팅 시스템
  - 클라이언트-서버 컴퓨팅 모델
  - peer-to-peer (P2P) 모델
  - 클라우드 컴퓨팅, 모바일 컴퓨팅, 인터넷/웹 컴퓨팅, 등
- 컴퓨팅 환경: 가상화
  - 가상기계(virtual machine)
- 컴퓨팅 환경: 실시간 내장 시스템

- OS operations
  - Processing exceptions caused by software
  - User mode vs system mode (or kernel mode)
- Functional/managemental components of OS
  - Process, memory, storage, I/O, utility, etc.
- System protection vs. system security
- Kernel data structures

- Computing environments: distributed computing systems
  - Client/server computing model
  - Peer-to-peer computing (P2P) model
  - Cloud computing, mobile computing, Internet/Web computing, etc.
- Computing environment: virtualization
  - Virtual machines
- Computing environment: real-time embedded systems