Chapter 1: Introduction

Zhi Wang
Florida State University

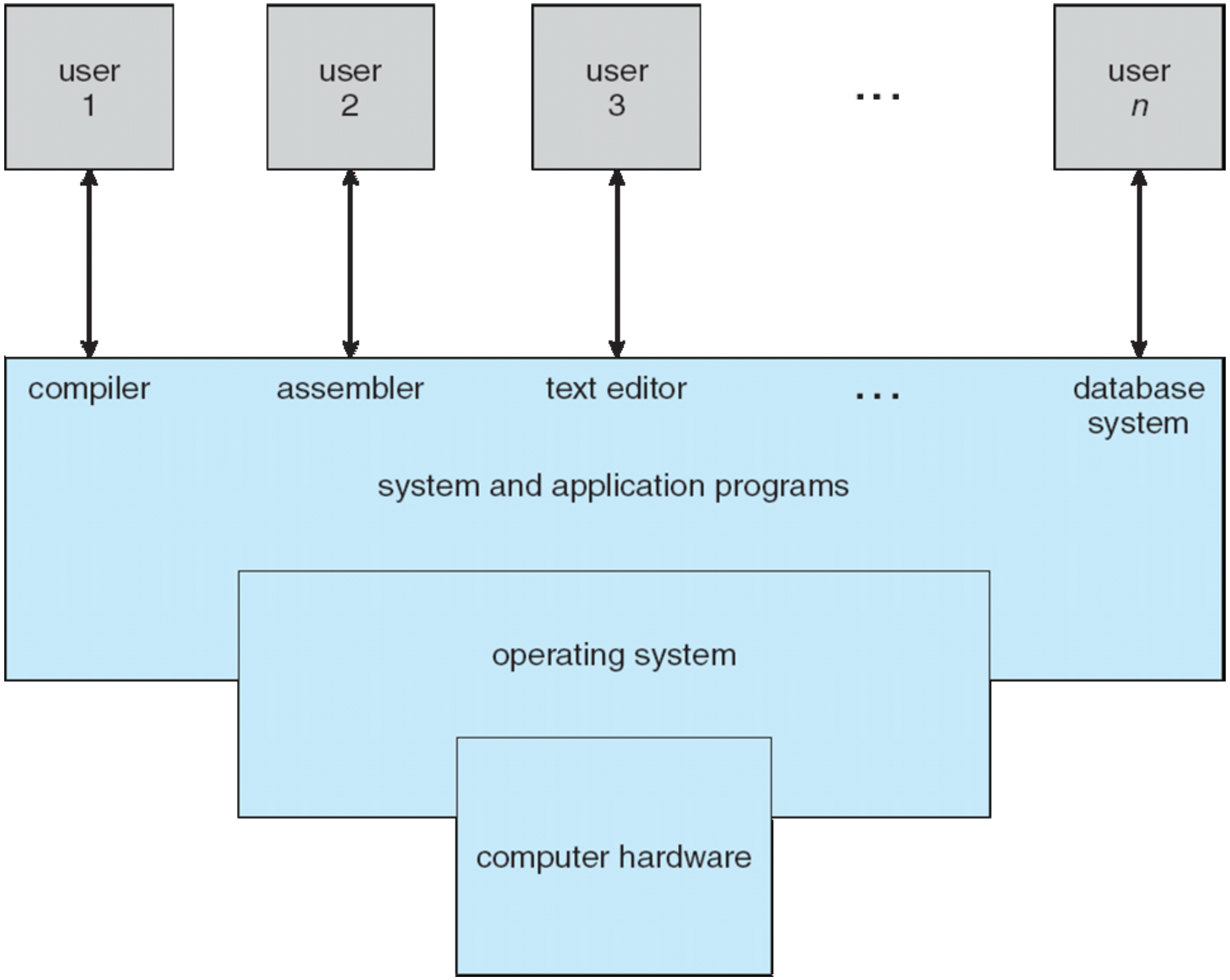
Content

- Computer systems
- Operating system operations
  - process management
  - memory management
  - storage management
  - protection and security

Four Components of a Computer System

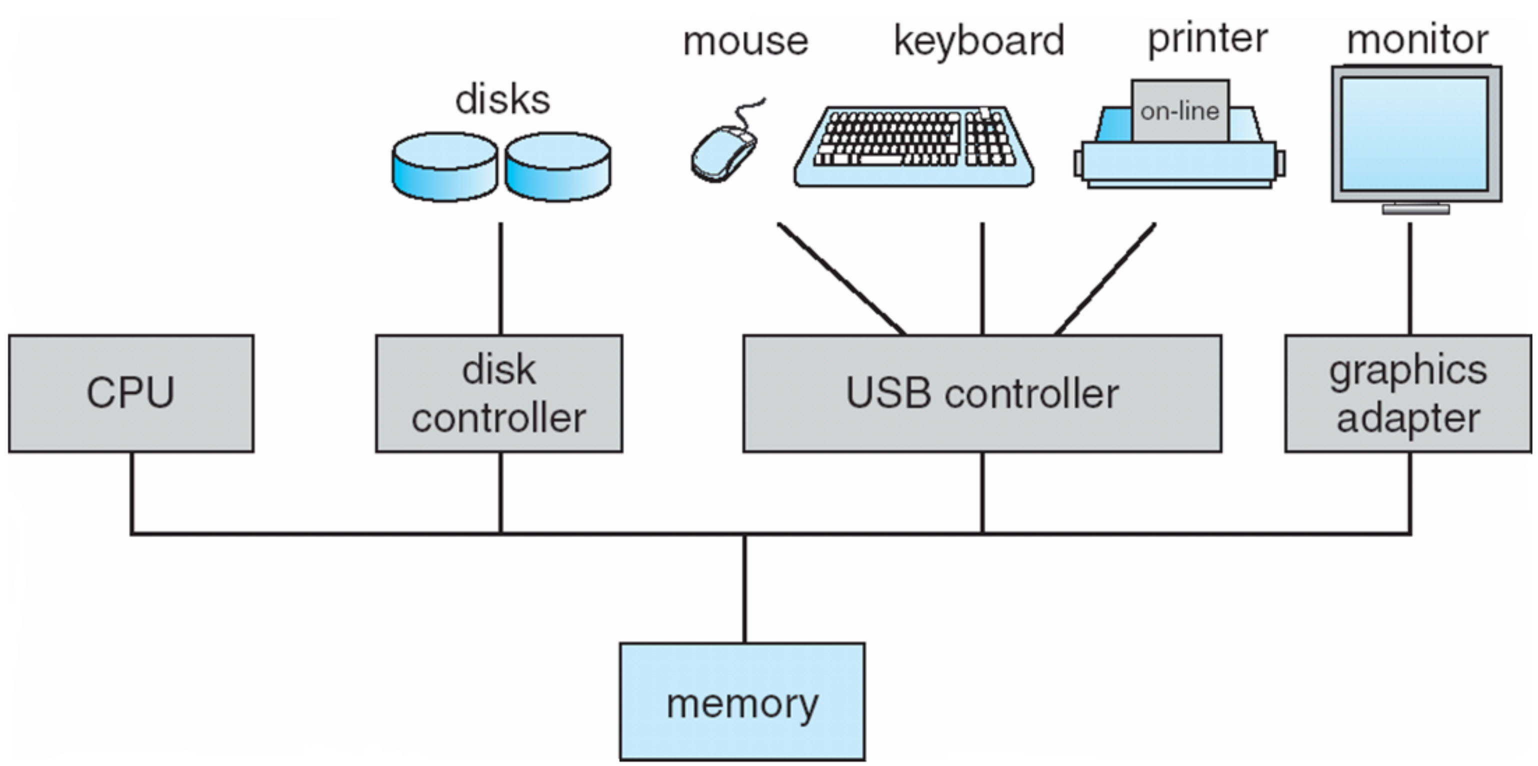
- Computer system has four components:
  - **hardware** provides basic computing resources
    - e.g., CPU, memory, I/O devices
  - **operating system** controls and coordinates use of hardware among users
  - **application** programs use system resources to solve computing problems
    - e.g., word processors, compilers, web browsers….
  - **users**
    - e.g., people, machines, other computers

Four Components of a Computer System



Hardware Components

- CPUs & device controllers connect through **buses** to share memory
- Concurrent execution of CPUs & devices compete for memory cycles



Devices

- Each **device controller** is in charge of a particular device type
  - disk controller, USB controller…
- Each device controller has a local buffer
  - I/O: between the device and local buffer of the controller
  - CPU moves data between main memory and controller buffers
- I/O devices and the CPU can execute concurrently
  - DMA (direct memory access)
  - device controller informs CPU that it has finished its operation by causing an interrupt

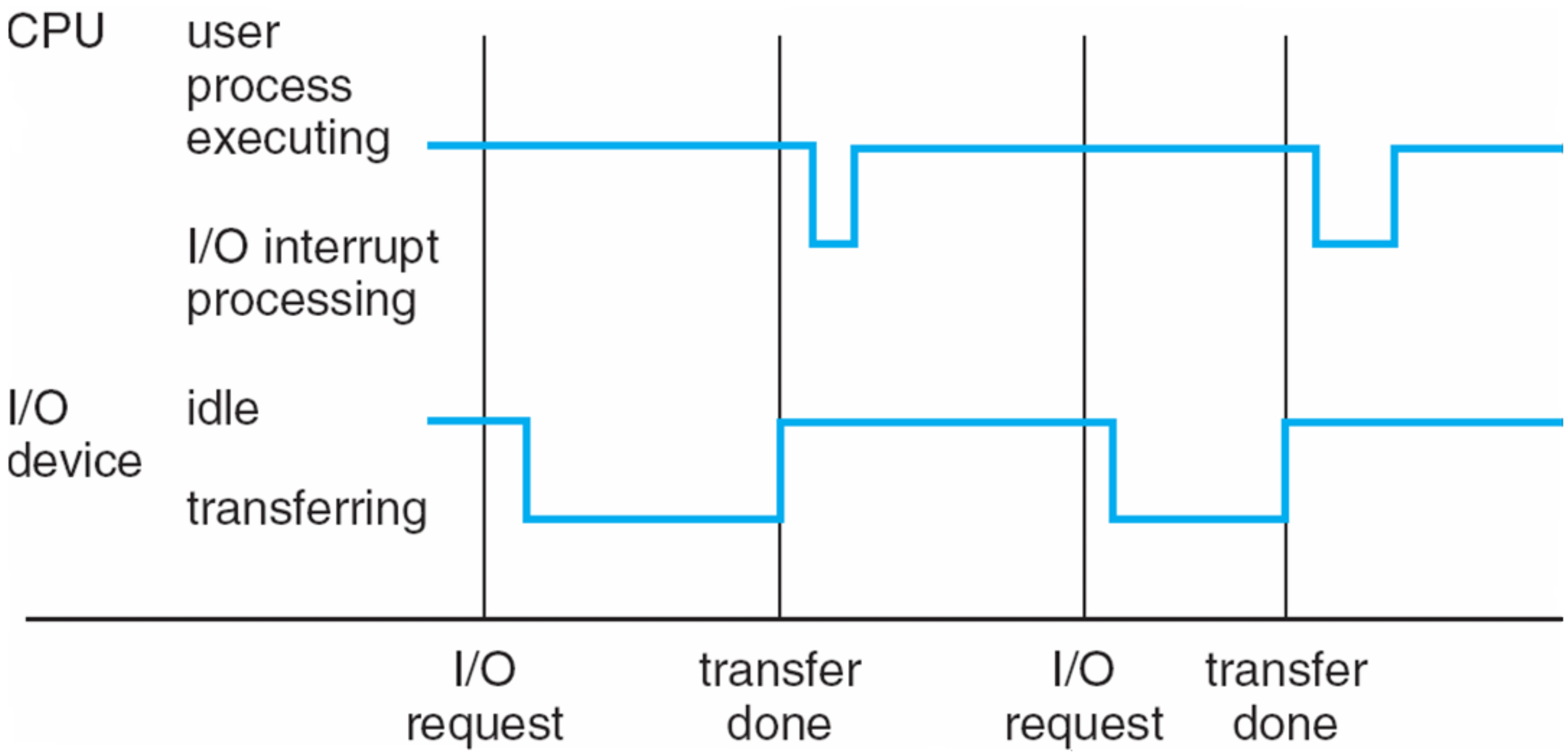
Interrupts and Traps

- Interrupt transfers control to the interrupt service routine
  - **interrupt vector:** a table containing addresses of all the service routines
  - incoming interrupts are disabled while serving another interrupt to prevent a lost interrupt
  - **interrupt handler** must save the (interrupted) execution states
- A **trap** is a software-generated interrupt, caused either by an error or a user request
  - an **interrupt** is asynchronous; a **trap** is synchronous
  - e.g., system call, divided-by-zero exception, general protection exception…
- Operating systems are usually **interrupt-driven**

Interrupt Handling

- Operating system preserves the execution state of the CPU
  - save registers and the program counter (PC)
- OS determines which device caused the interrupt
  - **polling**
  - **vectored** interrupt system
- OS handles the interrupt by calling the device's driver
- OS restores the CPU execution to the saved state

Interrupt Timeline



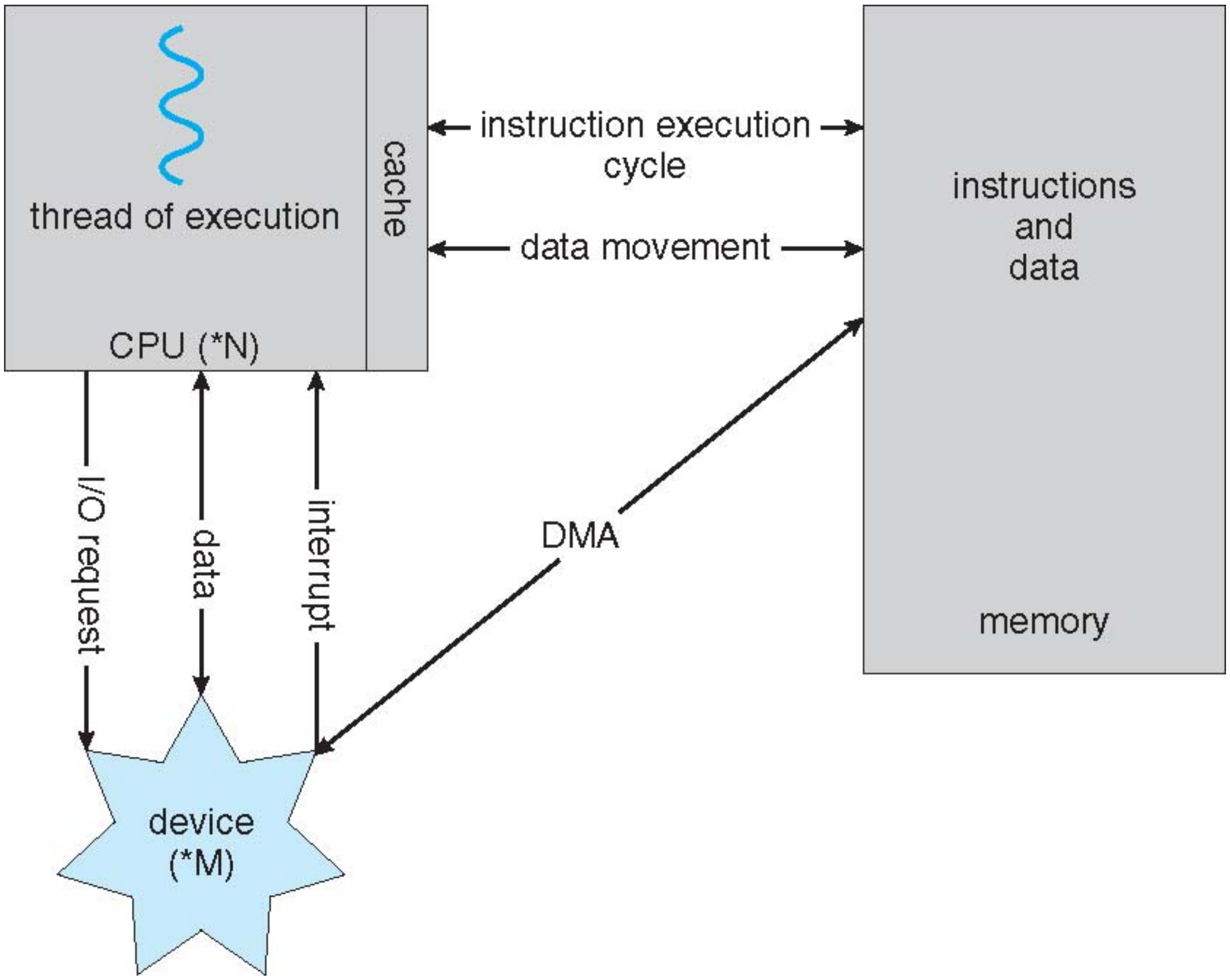
I/O: from System Call to Devices, and Back

- A program uses a **system call** to access system resources
  - e.g., files, network
- Operating system converts it to device access and issues I/O requests
  - I/O requests are sent to the device driver, then to the controller
  - e.g., read disk blocks, send/receive packets…
- OS puts the program to wait (**synchronous I/O**) or returns to it without waiting (**asynchronous I/O**)
  - OS may switches to another program when the requester is waiting
- I/O completes and the controller interrupts the OS
- OS processes the I/O, and then wakes up the program (synchronous I/O) or send its a signal (asynchronous I/O)

Direct Memory Access

- DMA is used for high-speed I/O devices able to transmit information at close to memory speeds
  - e.g., Ethernet, hard disk, cd rom…
- Device driver sends an I/O descriptor the controller
  - **I/O descriptor**: operation type (e.g., send/receive), memory address…
- The controller transfers blocks of data between its local buffer and main memory **without CPU intervention**
  - only one interrupt is generated when whole I/O request completes

Put it Together



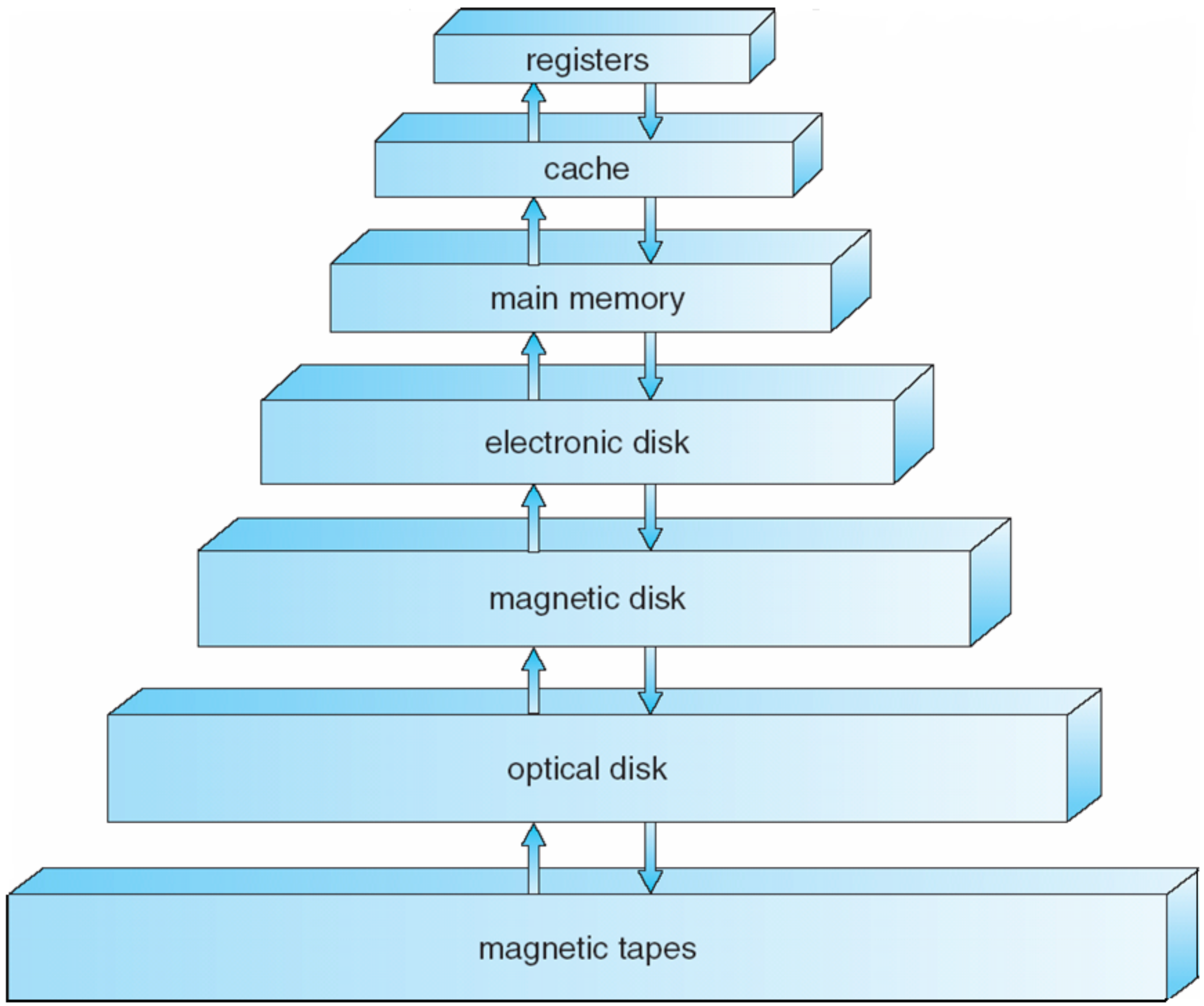
Storage Structure

- Main memory: the only large storage that CPU can directly access
  - **random access**, and typically **volatile**
- Secondary storage: large **nonvolatile** storage capacity
  - Magnetic disks are most common second-storage devices
    - rigid metal or glass platters covered with magnetic recording material
    - disk surface is logically divided into **tracks** and **sectors**
    - disk controller determines the interaction between OS and the device

Storage Hierarchy

- Storage systems can be 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 Hierarchy



Performance of Storages

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

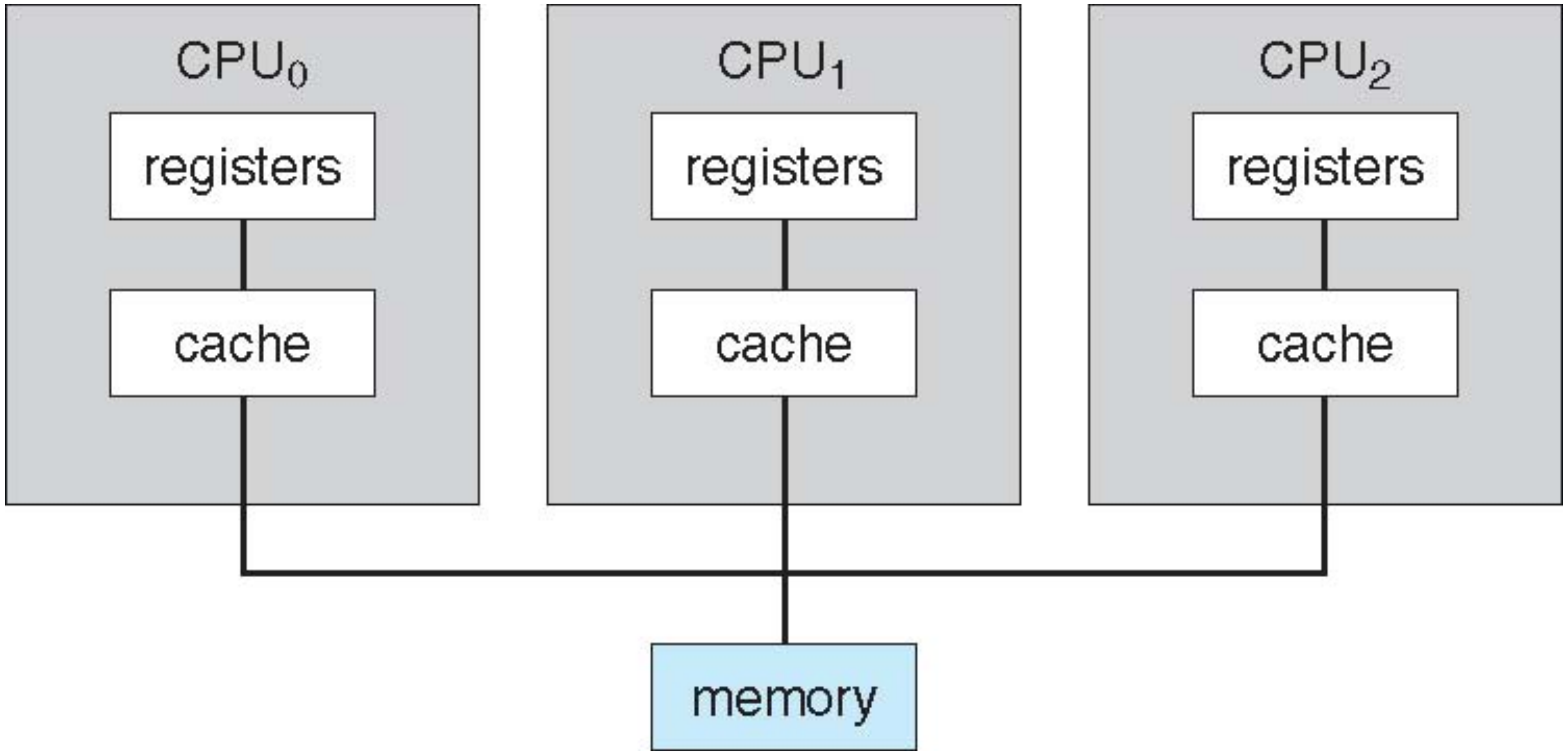
Caching

- Caching is an important principle, performed at many levels
  - e.g., in hardware, operating system, user program…
- **Caching**: data in use copied from slower to faster storage temporarily
  - faster storage (cache) is checked first to determine if data is there
    - if it is, data is used directly from the cache (fast)
    - if not, data is first copied to cache and used there
- Cache is usually smaller than storage being cached
- Cache management is an important design problem
  - e.g., cache size and replacement policy

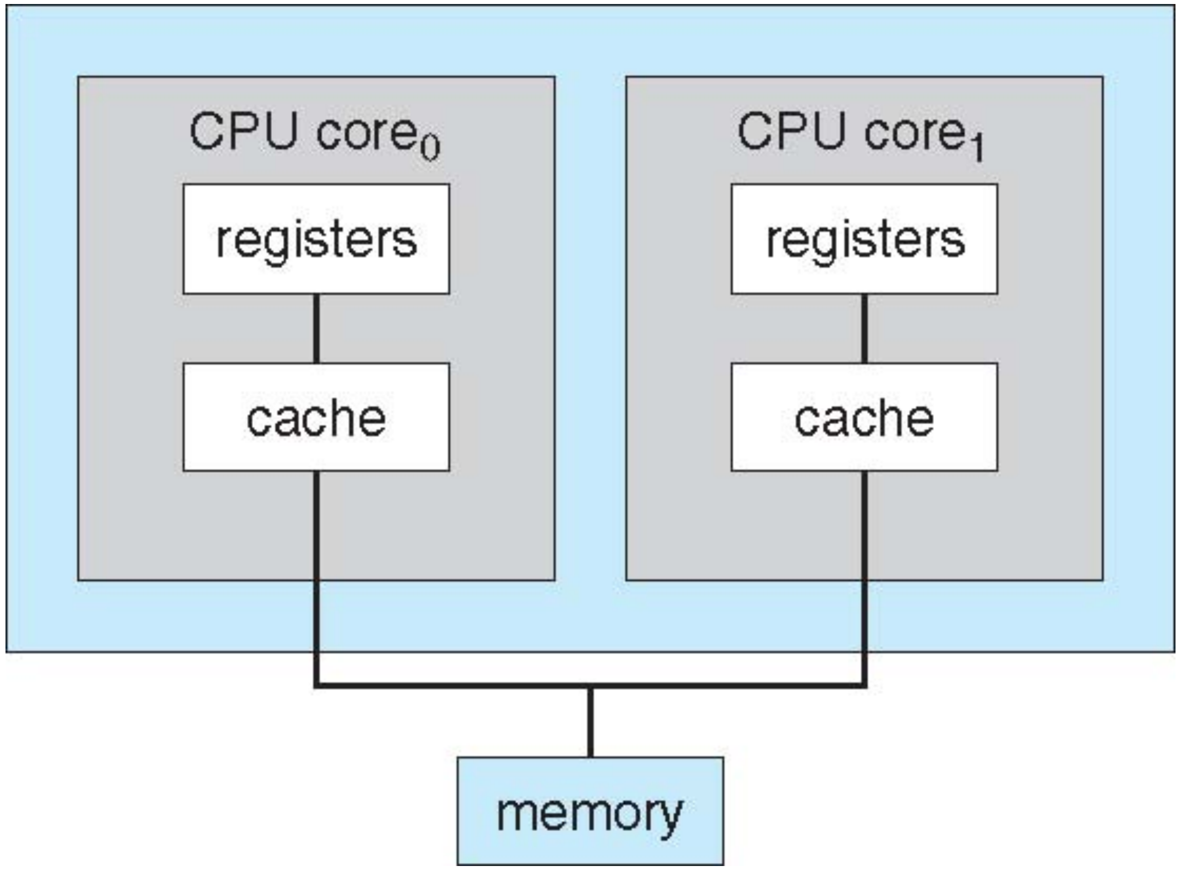
Multiprocessor Systems

- Most old systems have one single general-purpose processor
  - e.g., smartphone, PC, server, mainframe
  - most systems also have special-purpose processors as well
- Multiprocessor systems have grown in use and importance
  - also known as parallel systems, tightly-coupled systems
  - advantages: increased throughput, economy of scale, increased reliability -- graceful degradation or fault tolerance
  - two types: **asymmetric multiprocessing** and **symmetric multiprocessing**

Symmetric Multiprocessing Architecture



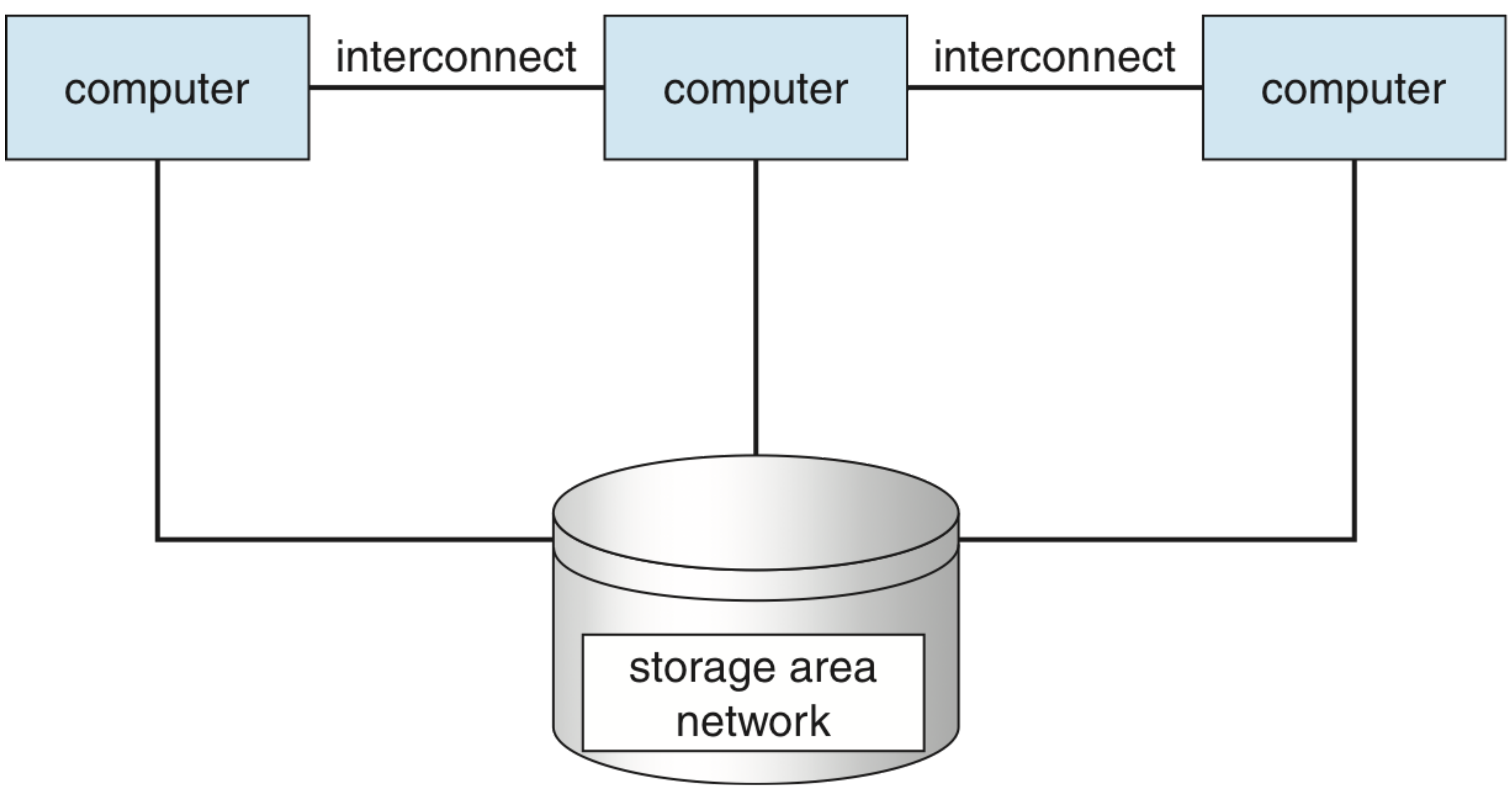
A Dual-Core Design



Clustered Systems

- Multiple systems work together ***through high-speed network***
  - usually sharing storage via a storage-area network (SAN)
- Clusters provide a high-availability service that can survive failures
  - **asymmetric** clustering has one machine in hot-standby mode
  - **symmetric** clustering has multiple nodes running applications, monitoring each other
- Some clusters are designed for high-performance computing (HPC)
  - applications must be written to use parallelization

Clustered Systems



Distributed Systems

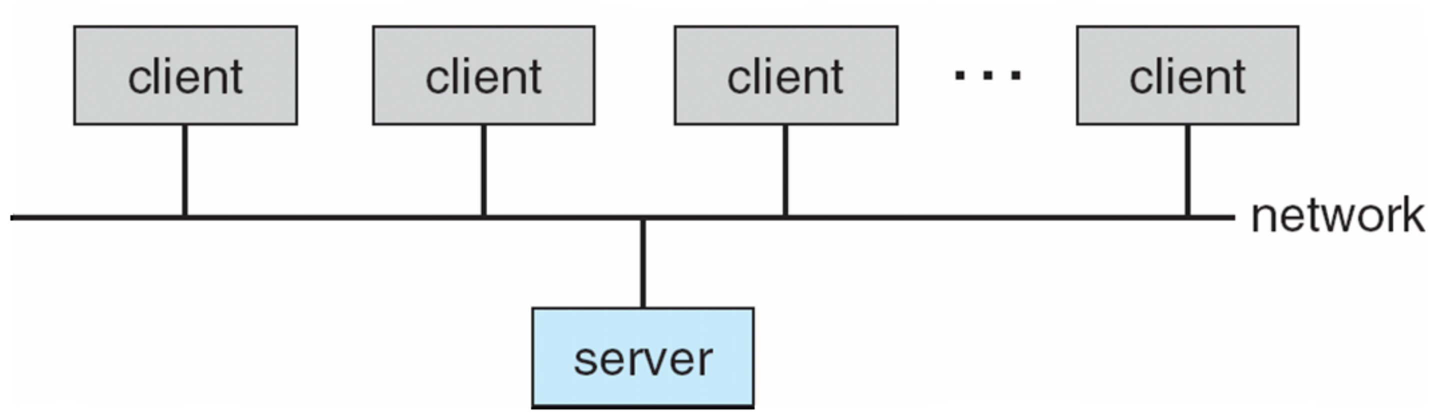
- A collection of separate, possibly heterogeneous, systems inter-connected through networks
- **Network OS** allows systems to exchange messages
- A **distributed system** creates the illusion of a single system

Special-Purpose Systems

- Real-time embedded systems most prevalent form of computers
  - vary considerably
  - use special purpose (limited purpose) real-time OS
- Multimedia systems
  - streams of data must be delivered according to time restrictions
- Handheld systems
  - e.g., PDAs, smart phones
  - limited CPU, memory, and power
  - used to use reduced feature OS

Client-Server Computing

- Dumb terminals were supplanted by smart PCs
- Servers responds to requests generated by clients
  - database server provides an interface for client to access database
  - file server provides an interface for clients to store and retrieve files



Peer-to-Peer Computing

- Another model of distributed system
- P2P does not distinguish clients and servers
    - instead all nodes are considered peers
    - may each act as client, server or both
- A node must join P2P network
    - registers its service with central lookup service, or
    - broadcast request for and respond to service via a discovery protocol
- Examples include BitTorrent, Napster and Gnutella

Web-Based Computing

- Web has become ubiquitous
  - more devices become connected to allow web access: PCs, smartphone, tablets, refrigerator…
- Web server farms become highly sophisticated
  - power is most expensive for big data centers

What Operating Systems Do

- Users want convenience, ease of use
  - don’t care much about resource utilization
- Shared computers (e.g., mainframe) must keep all users happy
  - users of dedicate systems frequently use shared resources from servers
  - e.g., gmail, google doc…
- Handhold devices are resource constrained, optimized for usability and battery life
  - e.g., smartphones, tablets
- Some computers have little or no user interface
  - e.g., embedded computers in devices and automobiles

What Operating Systems Do

- OS is a **resource allocator**
  - it manages all resources
  - it decides between conflicting requests for efficient and fair resource sharing
- OS is a **control program**
  - it controls program execution to prevent errors and improper use of system

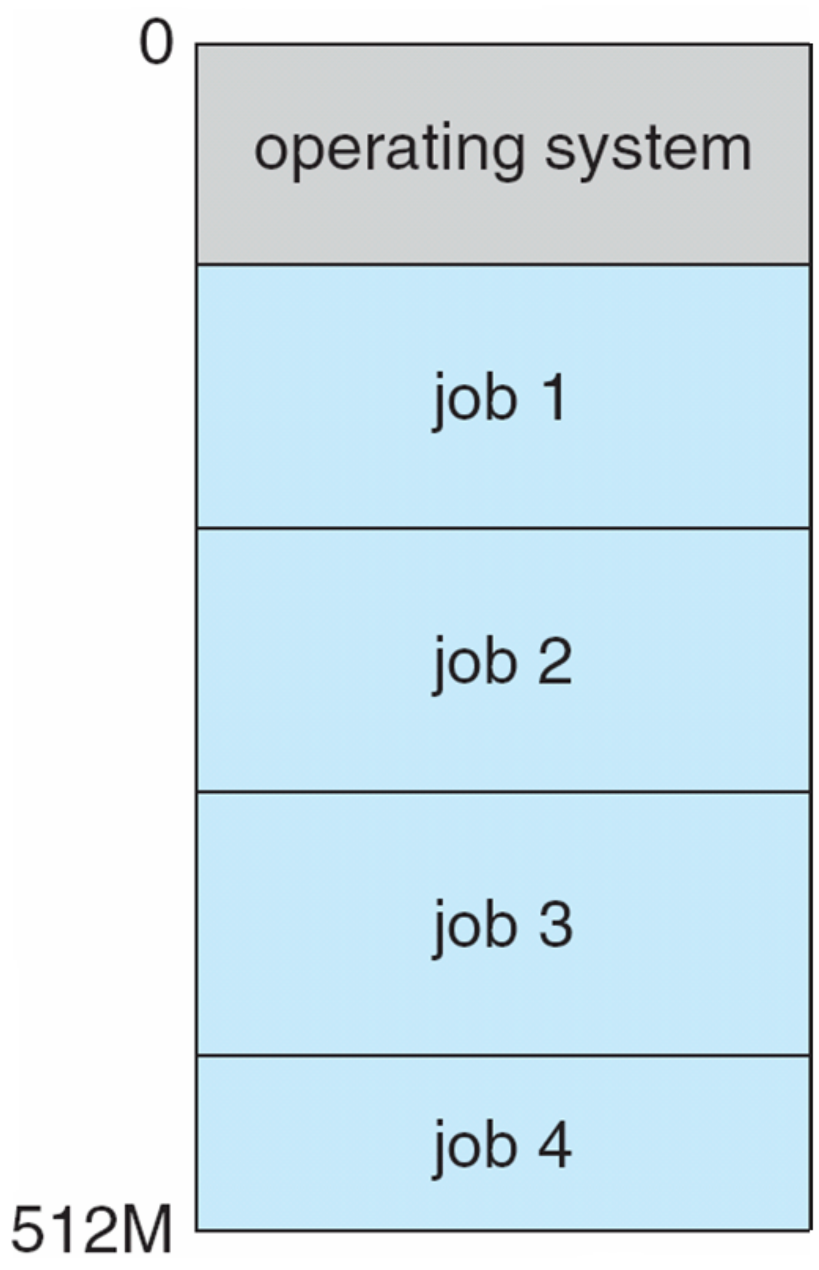
Operating System Definition

- A good approximation is "**everything a vendor ships when you order an operating system**"
  - no universally accepted definition
  - what the vendor ships can vary wildly
- Kernel is "**the one program running at all times on the computer**"
  - what about demon programs that starts with the kernel such as init?
- Everything else is either a system program or an application program
  - system programs are shipped with the OS

Operating System Structure

- **Multiprogramming** is necessary for efficiency
  - single user cannot keep CPU and I/O devices busy at all times
  - user's computing tasks are organized as jobs (code and data)
  - kernel schedules jobs (job scheduling) so CPU always has things to do
    - a subset of total jobs in system is kept in memory
    - when a job has to wait (e.g., for I/O), kernel switches to another job
- **Timesharing** (multitasking) extends the multiprogramming
  - OS switches jobs so frequently that users can interact with each running job
  - response time should be < 1s
  - each user has at least one program executing in memory (**process**)
  - if several jobs ready to run at the same time (**CPU scheduling**)

Memory Layout for Multiprogrammed System

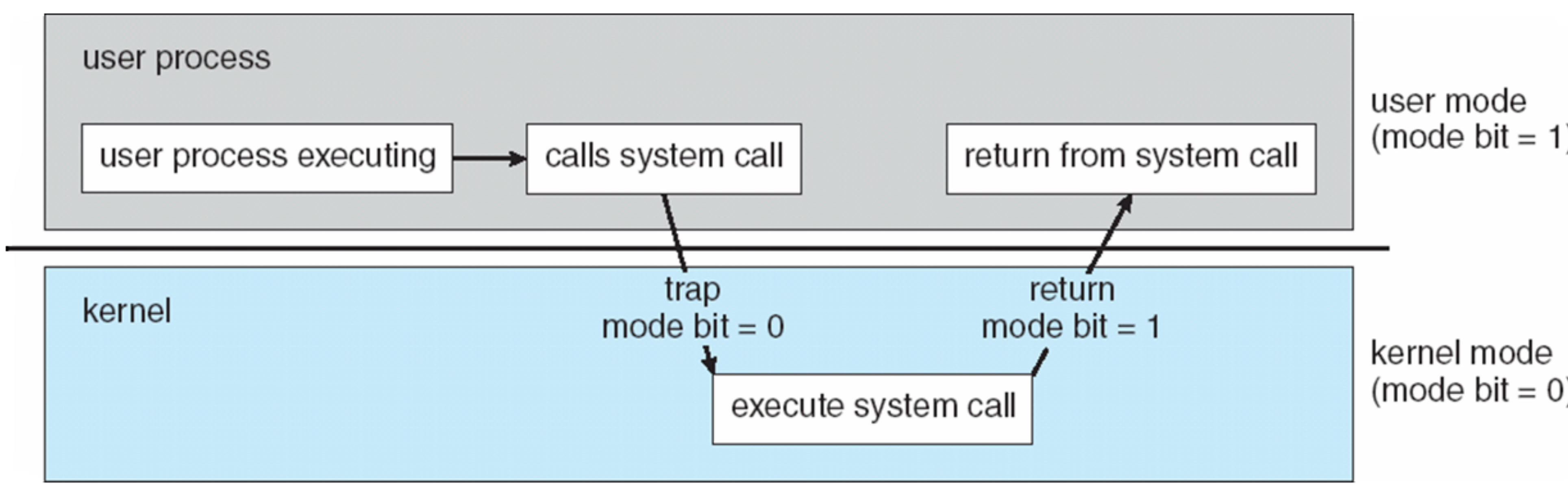


Dual-mode operation

- Operating system is usually interrupt-driven
- **Dual-mode operation** allows OS to protect itself and other system components
  - **user mode** and **kernel mode**
  - a **mode** bit distinguishes when CPU is running user code or kernel code
  - some instructions designated as **privileged**, only executable in kernel
  - **system call** changes mode to kernel, return from call resets it to user

Transition between Modes

- System calls, exceptions, interrupts cause transitions between kernel/user modes
- Timer used to prevent infinite loop or process hogging resources
  - to enable a timer, set the hardware to interrupt after some period
  - OS sets up a timer before scheduling process to regain control
    - the timer for scheduling is usually **periodical** (e.g., 250HZ)
    - **tickless kernel**: on-demand timer interrupts



Process Management

- A process is **a program in execution**
  - program is a *passive* entity, process is an *active* entity
  - a system has many processes running concurrently
- Process needs resources to accomplish its task
  - OS reclaims all reusable resources upon process termination
  - e.g., CPU, memory, I/O, files, initialization data
- Single-threaded process has one program counter
  - **program counter** specifies *location of next instruction to execute*
  - processor executes instructions sequentially, one at a time, until completion
- Multi-threaded process has **one program counter per thread**

Process Management Activities

- Process creation and termination
- Processes suspension and resumption
- Process synchronization primitives
- Process communication primitives
- Deadlock handling

Memory Management

- Memory is the main storage directly accessible to CPU
  - data needs to be kept in memory before and after processing
  - all instructions should be in memory in order to execute
- Memory management determines what is in memory to optimize CPU utilization and response time
- Memory management activities:
  - keeping track of which parts of memory are being used and by whom
  - deciding which processes and data to move into and out of memory
  - allocating and deallocating memory space as needed

Storage Management (File Systems)

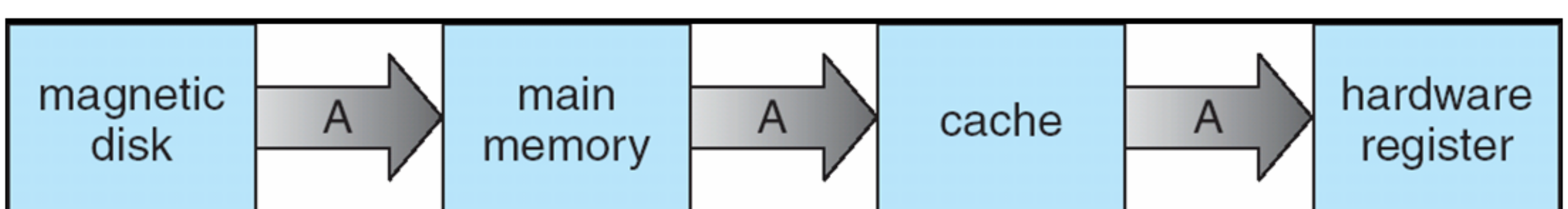
- OS provides a uniform, logical view of data storage
  - **file** is a logical storage unit that abstracts physical properties
    - files are usually organized into **directories**
    - access control determines who can access the file
- File system management activities:
  - creating and deleting files and directories
  - primitives to manipulate files and directories
  - mapping files onto secondary storage
  - backup files onto stable (non-volatile) storage media

Mass-Storage Management

- Disk subsystem manages mass storages
  - disks are used to store:
    - data that does not fit in main memory
    - data that must be kept for a “long” period of time
  - entire speed of the system hinges on disk subsystem and its algorithms
    - some storage needs not be fast (e.g., optical storage or magnetic tape)
- Mass-storage management activities:
  - free-space management
  - storage allocation
  - disk scheduling

Migration of Data Through Storage Layers

- System must use most recent value, no matter where it is stored
- Many levels of **data coherency**
  - cache coherency for multiprocessors (cache snooping)
    - all CPUs have the most recent value in their cache
  - synchronization for multi-processes or multiple threads
  - distributed environment situation even more complex
    - several copies of a datum can exist



I/O Subsystem

- I/O subsystem hides peculiarities of hardware devices from the user
- I/O subsystem is responsible for:
  - manage I/O memory
    - **buffering**: to store data temporarily while it is being transferred
    - **caching**: to store parts of data in faster storage for performance
    - **spooling**: the overlapping of output of one job with input of other jobs
  - define the general device-driver interfaces
    - object-oriented design pattern
  - manage device drivers for specific hardware devices

Protection and Security

- **Protection**: *mechanism* for controlling access to resources
  - User access control determines who it is and who can do what
    - each user has a user id including the name and an associated number
    - files and processes are associated with a user ID to determine access right
    - group id allows set of users to be defined and managed
    - privilege escalation is an attack that allows user to change to effective ID with more rights
- **Security**: defense of the system against internal and external attacks (*policy*)
  - e.g., denial-of-service, worms, viruses, identity theft, theft of service

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, BSDs, MINUX…

End of Chapter 1