CHAPTER 1 - INTRODUCTION

Jacob Aae Mikkelsen

OBJECTIVES

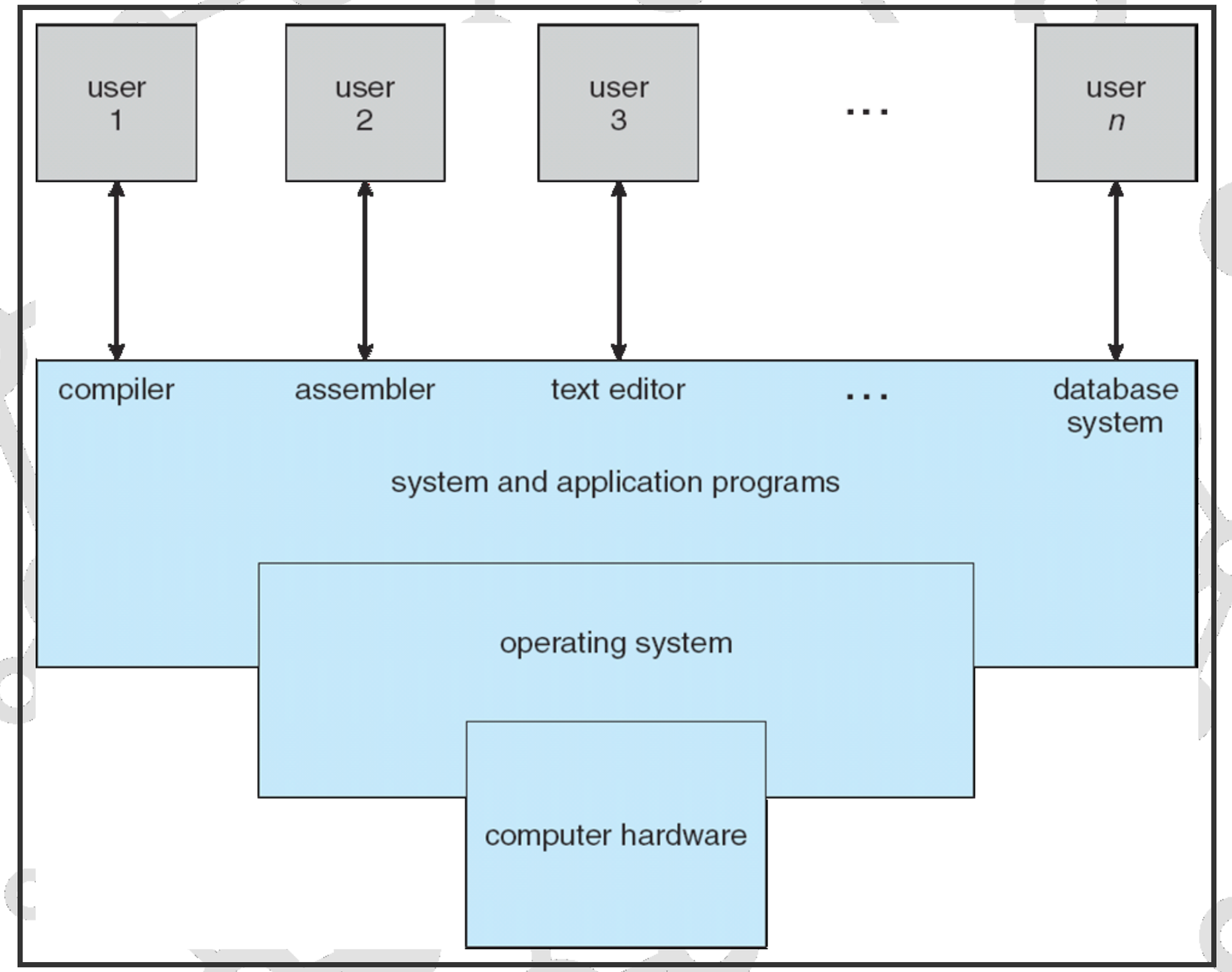
- Provide grand tour of major operating systems components
- Provide coverage of basic computer system organization

WHAT OPERATING SYSTEMS DO

COMPUTER SYSTEM STRUCTURE

1. Hardware
2. Operating system
3. Application programs
4. Users

ABSTRACT VIEW


user 1
user 2
user 3
...
user n
compiler
assembler
text editor
...
database system
system and application programs
operating system
computer hardware

WHAT IS AN OPERATING SYSTEM?

- A program
  - intermediary between a user and the hardware
- Goals:
  - Execute user programs
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

USER VIEW

Role of OS

- PC/Laptop → Monitor, keyboard, mouse, system unit
  - Role: Ease of use and performance
- Mainframe
  - Role: Maximize resource utilization
- Workstations on network
  - Role: Compromise between individual usability and resource utilization

USER VIEW

Role of OS

- Mobile devices → Touch screen
  - Role: Power management, ease of use
- Embedded computers
  - Role: Designed to run without user intervention

SYSTEM VIEW

OS is a **resource allocator**

- CPU Time
- Memory space
- File storage space
- I/O Devices

SYSTEM VIEW

OS is a **control program**

- Prevents
  - errors
  - improper use

WHAT IS AN OPERATING SYSTEM?

⚠ No universally accepted definition

"Everything a vendor ships when you order an operating system" is good approximation

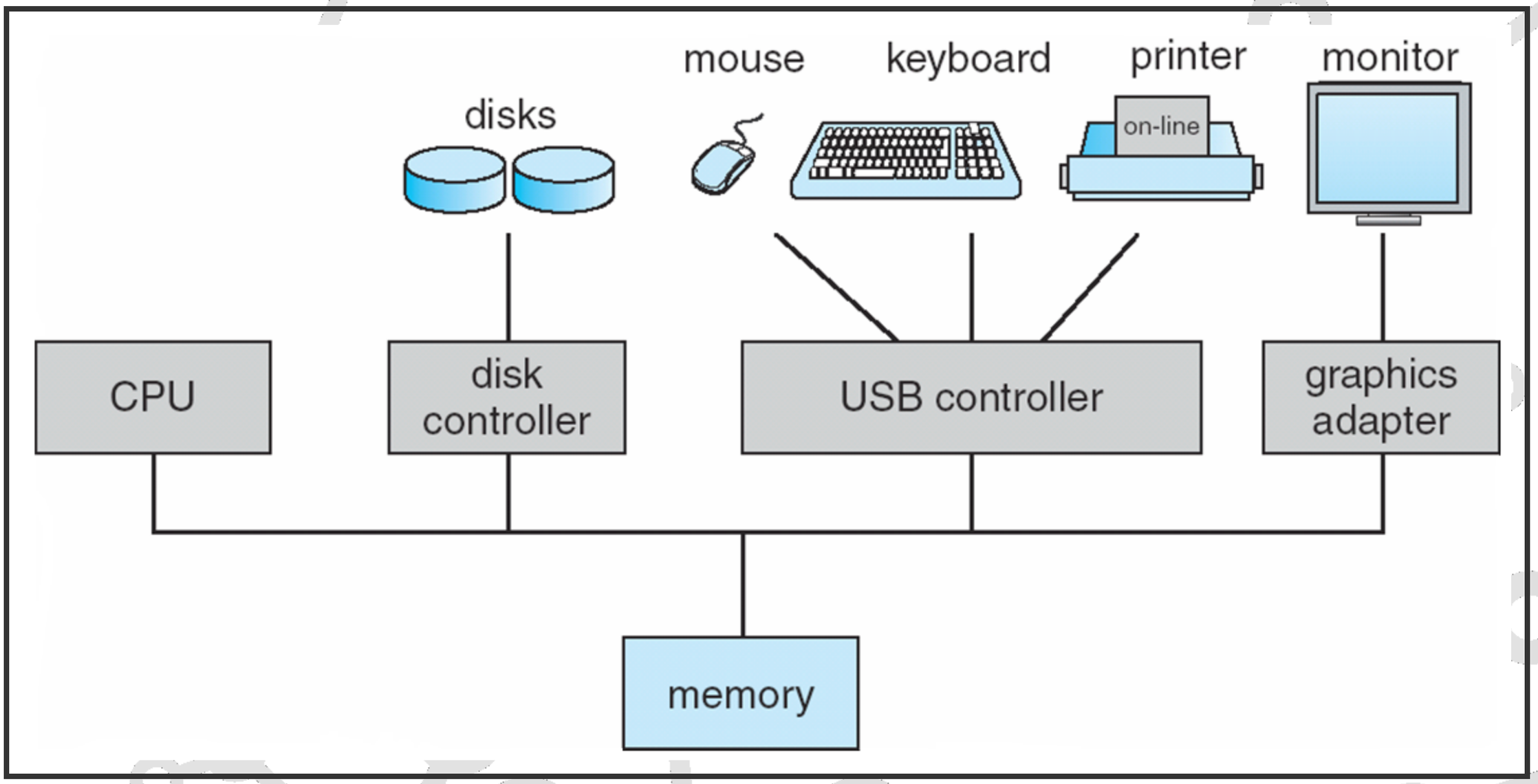
"The one program running at all times on the computer" is the kernel.

Everything else is either

- A system program (ships with the operating system)
- An application program

COMPUTER-SYSTEM ORGANIZATION

MODERN SYSTEM



BOOTING

Use **bootstrap program**

- Initializes all aspects of system
- Loads operating system kernel and starts execution

RUNNING

Providing services to the system and its users.

- System processes / system daemons
- Avait events

INTERRUPTS

Event → signalled by interrupt

- Hardware: Trigger interrupt by sending a signal to CPU
- Software: By triggering a system call.

💡 An operating system is interrupt driven

INTERRUPT VECTOR

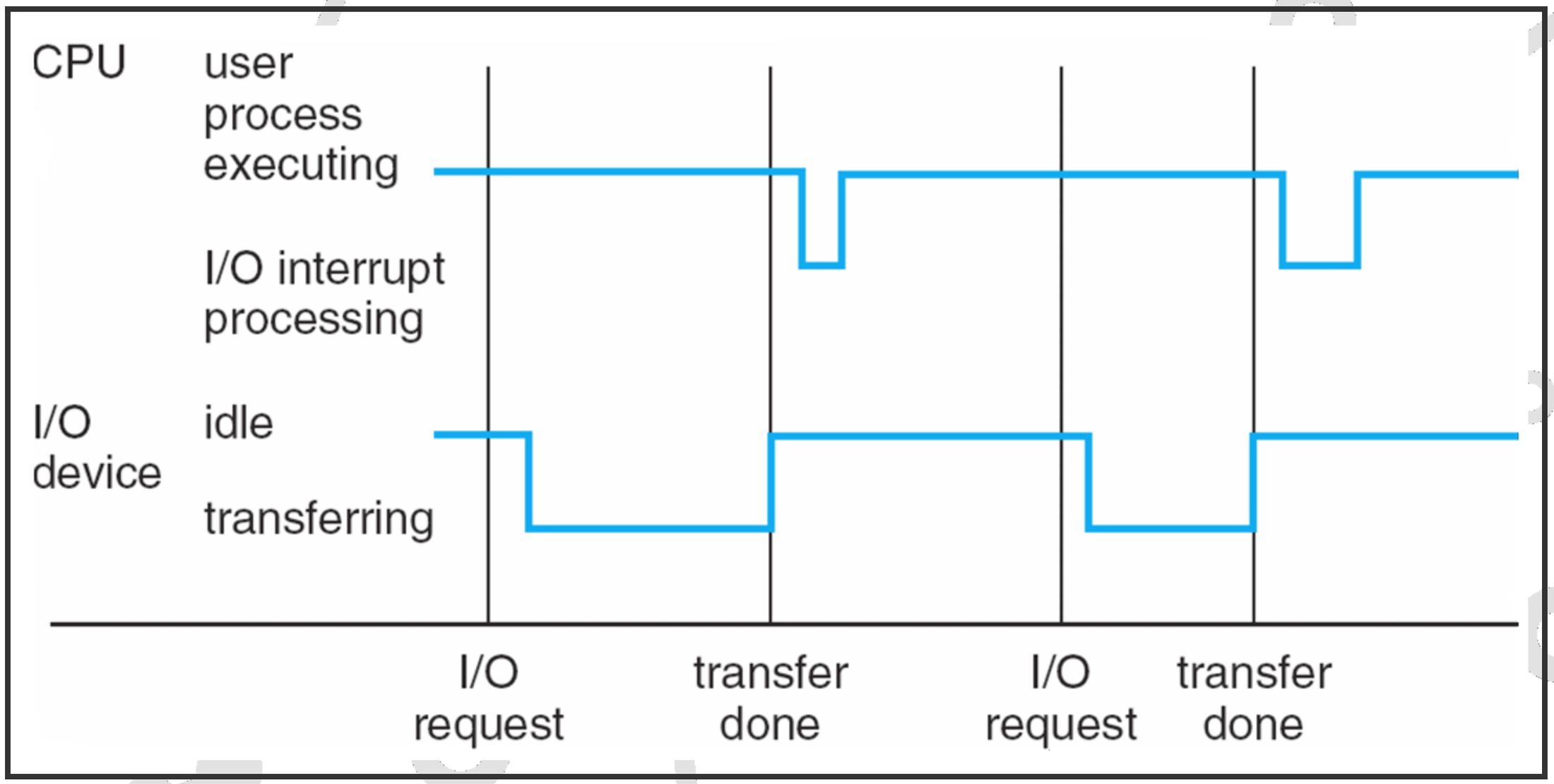
When system is interruptet, it immediately transfers execution to a fixed location

The operating system preserves the state of the CPU by storing registers and the program counter

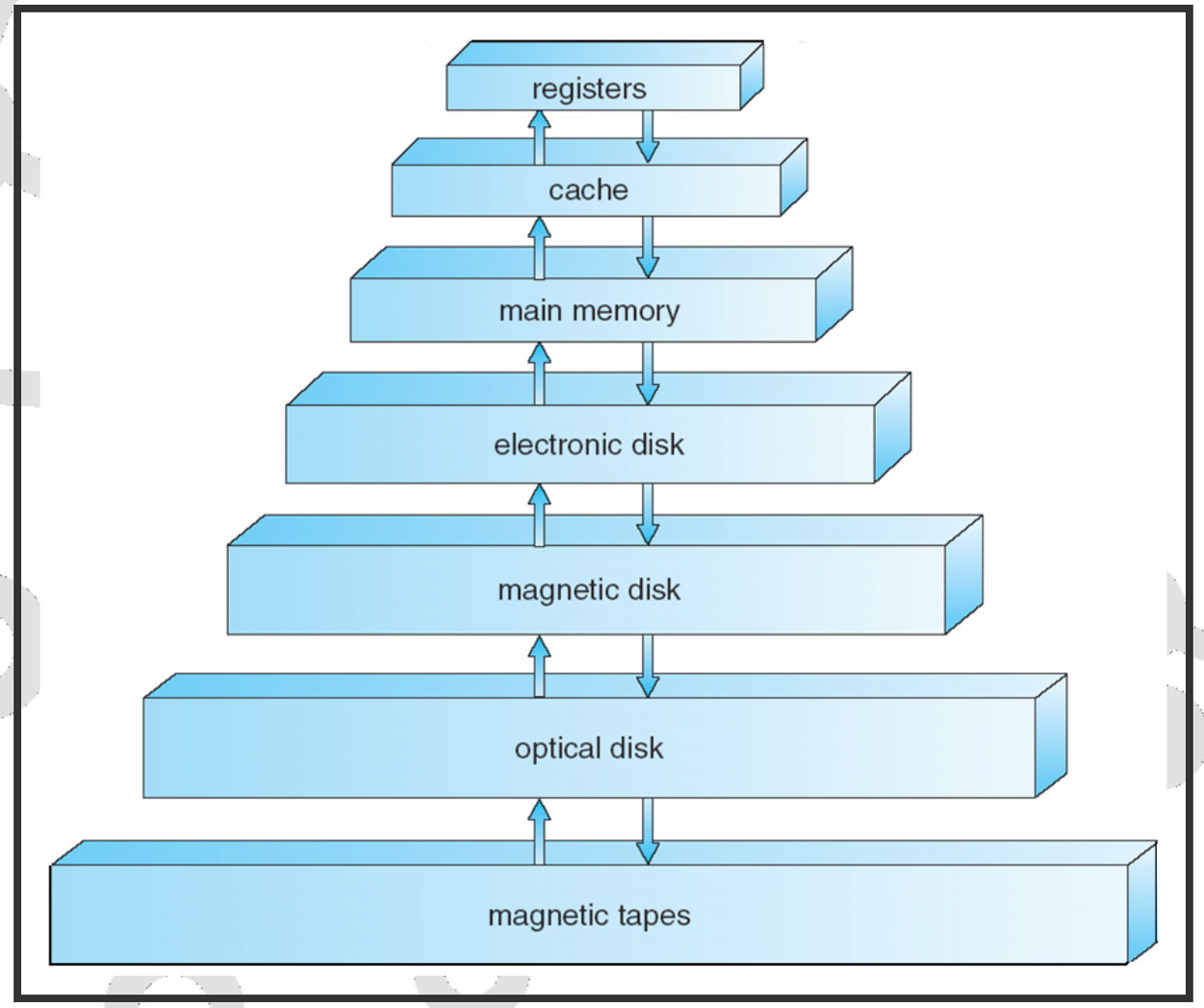
Interrupt Vector is an array with addresses of the interrupt service routines

A **trap** is a software-generated interrupt caused either by an error or a user request

INTERRUPT TIMELINE



STORAGE STRUCTURE



I/O STRUCTURE

General-purpose computer system → CPUs and device controllers connected through a common bus.

Device controller maintains some local buffer storage and a set of special-purpose registers.

Device driver for each device controller.

- Understands the device controller
- provides the rest of the OS with uniform interface to device.

OPERATION

To start an I/O operation

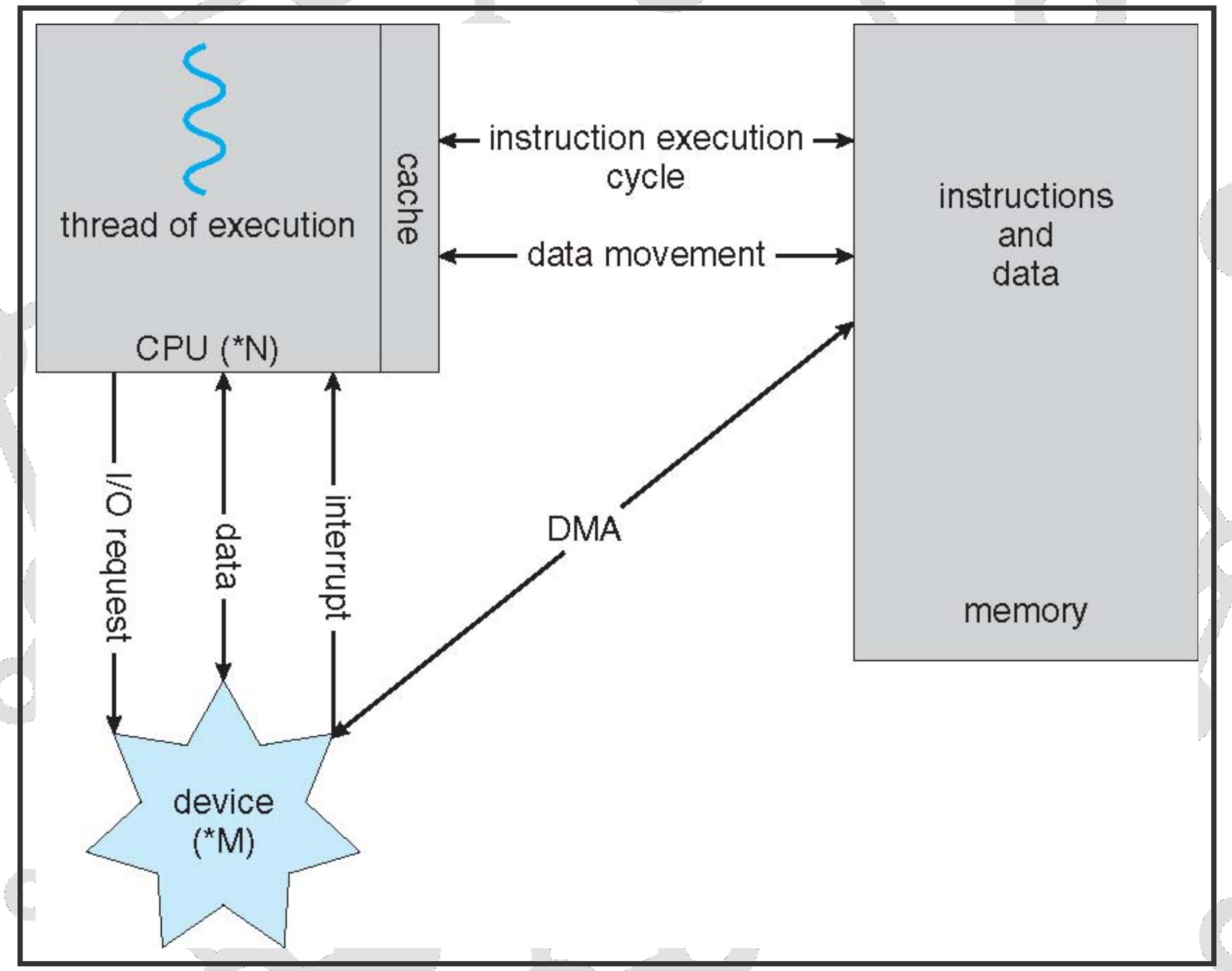
OPERATION

- ⚠ Interrupt-driven I/O is fine for moving small amounts of data → can produce high overhead when used for bulk data movement such as disk I/O .

DIRECT MEMORY ACCESS (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
- One interrupt per block, not one per byte

HOW IT WORKS


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

COMPUTER-SYSTEM ARCHITECTURE

SINGLE-PROCESSOR SYSTEMS

Previously, systems use a single general-purpose processor (PDAs through mainframes)

Most systems have special-purpose processors as well

- Graphics processor
- Disk processor
- Keybord processor

MULTIPROCESSOR SYSTEMS

Multiprocessors/Multicore systems growing in use and importance

- Also known as parallel systems, tightly-coupled systems

MULTIPROCESSOR ADVANTAGES

Advantages include

- Increased throughput
- Economy of scale
- Increased reliability – graceful degradation or fault tolerance

ASYMMETRIC MULTIPROCESSING

- Each processor is assigned a specific task.
- Boss processor controls the system
  - Other processors either look to the boss for instruction or have predefined tasks.
    - This scheme defines a boss–worker relationship.

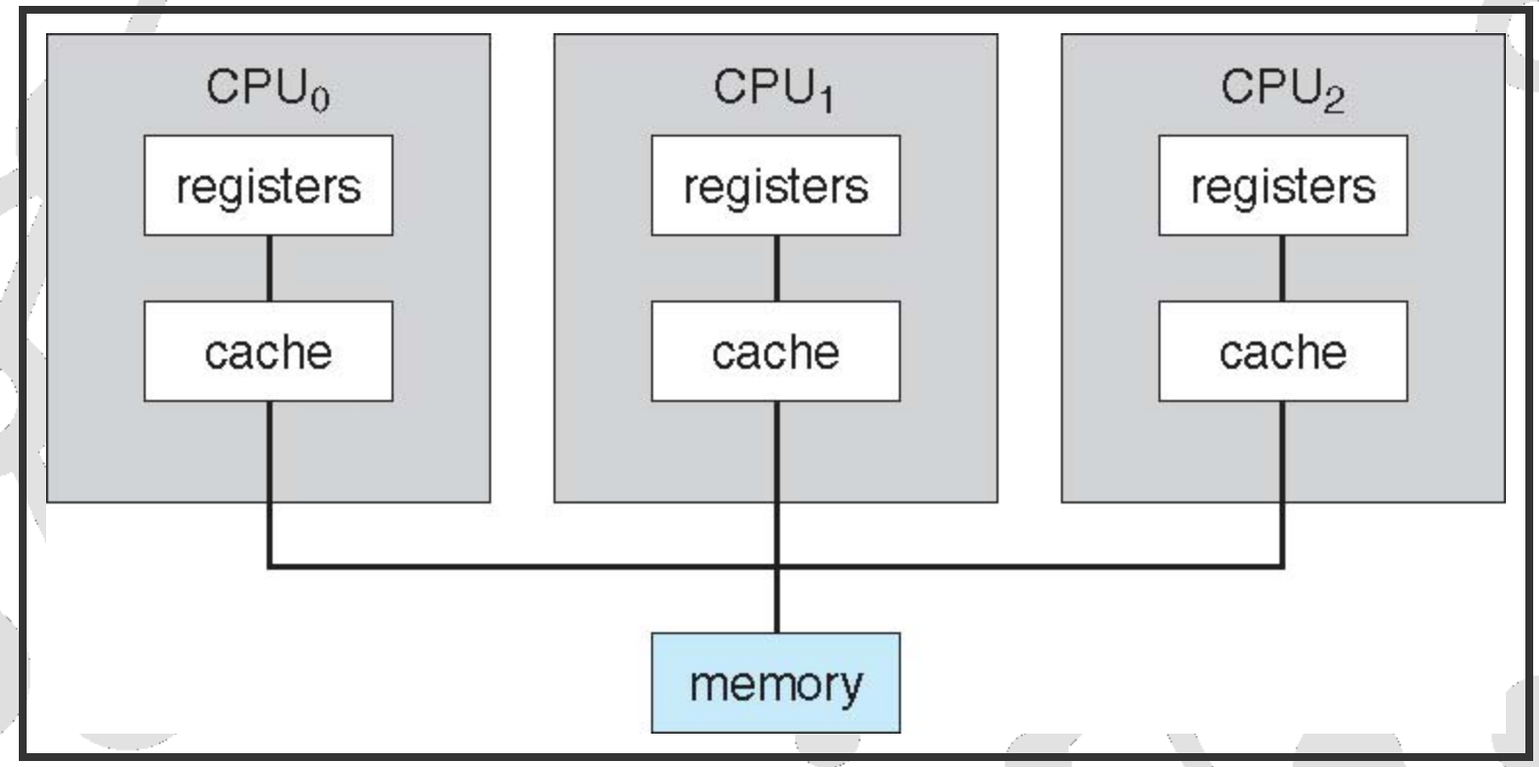
SYMMETRIC MULTIPROCESSING (SMP)

- Most common
- Each processor performs all tasks within the operating system.

Each processor

- **Own** set of registers, and local cache.
- **All** processors share physical memory.

SYMMETRIC MULTIPROCESSING ARCH.

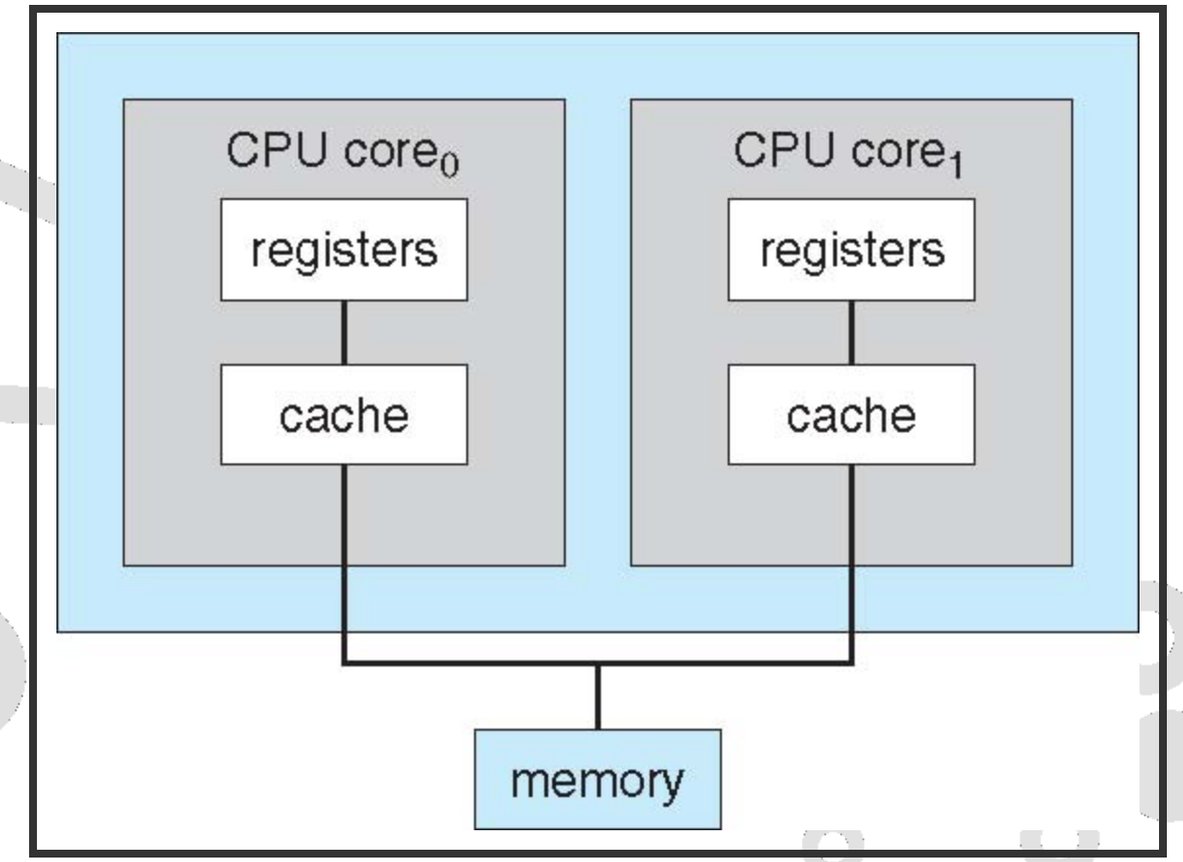


MULTICORE DESIGN

Multiple computing cores on a single chip.

Efficient: on-chip communication is faster than between-chip communication

DUAL-CORE DESIGN



CLUSTERED SYSTEMS

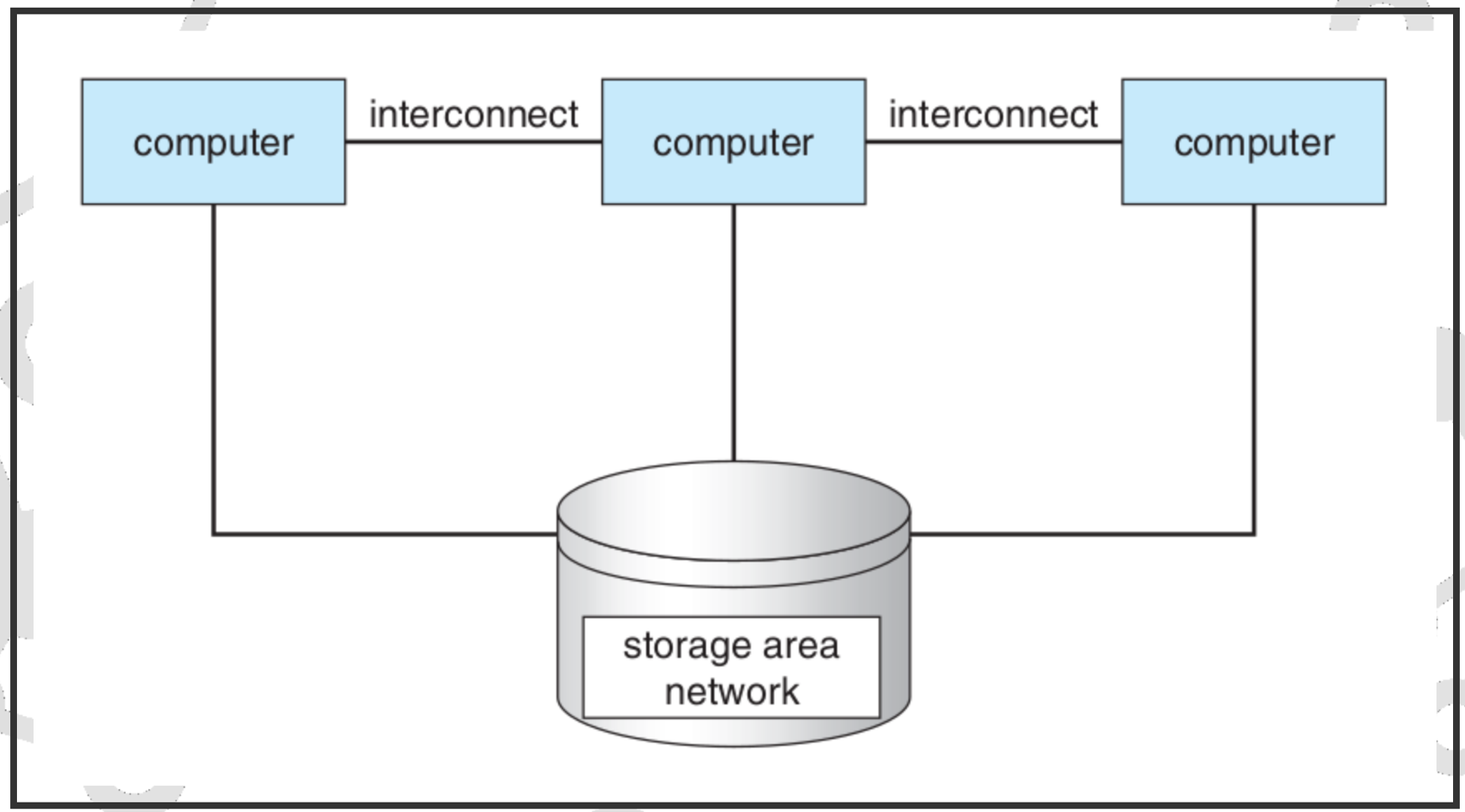
Clustered system → gathers together multiple CPUs.

Differ from the multiprocessor systems: Composed of two or more individual systems — joined together.

Usually used to provide high-availability service

- **Asymmetric clustering** has one machine in hot-standby mode
- **Symmetric clustering** has multiple nodes running applications, monitoring each other

CLUSTERED


computer
interconnect
computer
interconnect
computer
storage area
network

OPERATING-SYSTEM STRUCTURE

MULTIPROGRAMMING

Multiprogramming needed for efficiency

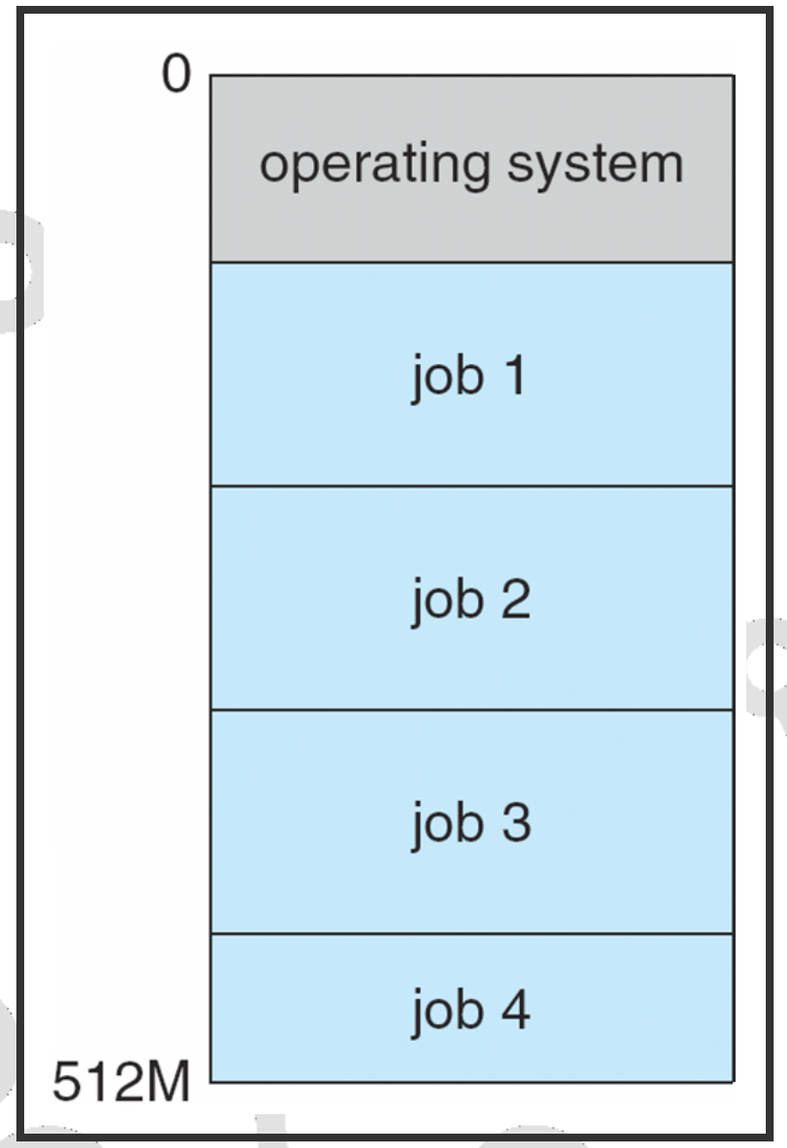
- Single user cannot keep CPU and I/O devices busy at all times
- Multiprogramming organizes jobs (code and data) so CPU always has one to execute
- A subset of total jobs in system is kept in memory
- One job selected and run via job scheduling
- When it has to wait (for I/O for example), OS switches to another job

TIMESHARING

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



OPERATING-SYSTEM OPERATIONS

OPERATING-SYSTEM OPERATIONS

A properly designed operating system must ensure that an incorrect (or malicious) program cannot cause other programs to execute incorrectly.

DUAL-MODE AND MULTIMODE OPERATION

Should distinguish between the execution of operating-system code and user- defined code

Need two separate modes of operation:

- **User mode**
- **Kernel mode** (supervisor mode, system mode, or privileged mode)

MODE BIT

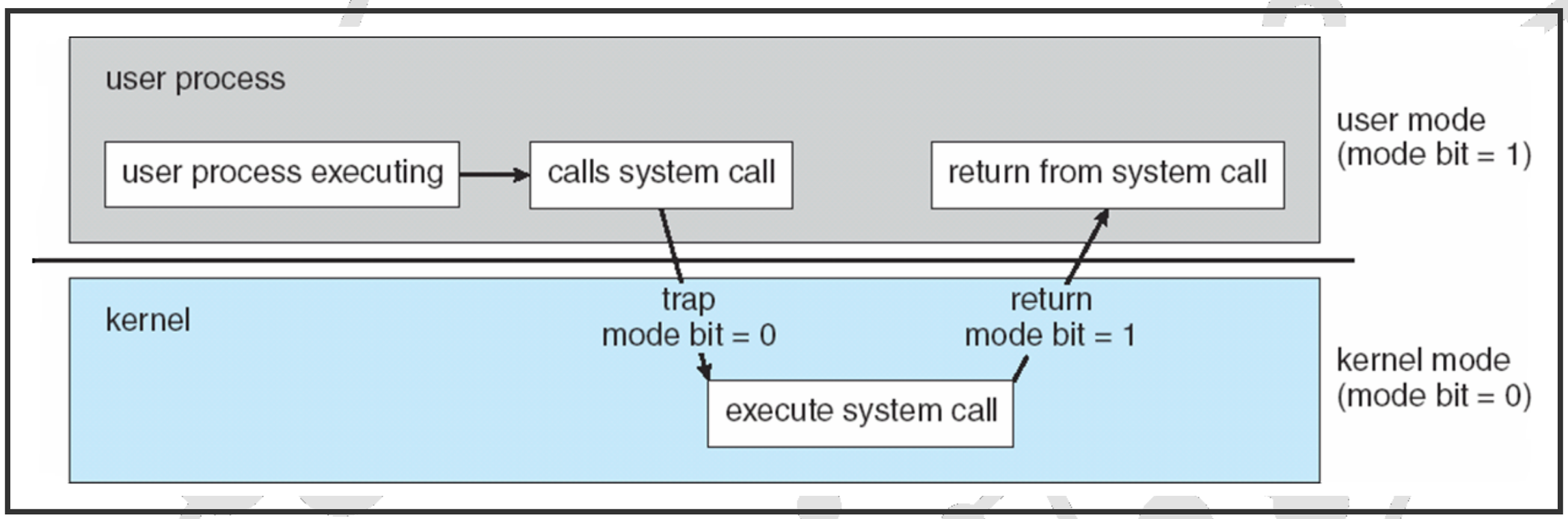
The **mode bit** → hardware supported indicate current mode

- kernel (0)
- user (1).

Some instructions designated as **privileged**, only executable in kernel mode

System call changes mode to kernel, return from call resets it to user

USER AND KERNEL MODE



USER AND KERNEL MODE

MS-DOS was written for the Intel 8088 architecture, which has no mode bit and therefore no dual mode.

Microsoft Windows 7, as well as Unix and Linux—take advantage of this dual-mode feature and provide greater protection for the operating system.

PROCESS MANAGEMENT

PROCESS VS PROGRAM

- A **process** is a **program** in execution.
  - 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

PROCESSES

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

PROCESS MANAGEMENT

OS responsibilities

- Scheduling processes and threads on the CPU s
- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication

MEMORY MANAGEMENT

NECESSARY

All data in memory before and after processing

All instructions in memory in order to execute

MEMORY MANAGEMENT

OS responsibilities

- Keep track of which parts of memory are currently being used
  - Who is using them
- Deciding which processes (or parts of processes) and data to move into and out of memory
- Allocating and deallocating memory space as needed

STORAGE MANAGEMENT

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

FILE-SYSTEM MANAGEMENT

OS activities:

- 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

OS activities

- Free-space management
- Storage allocation
- Disk scheduling

CACHING

Because caches have limited size, **cache management** is an important design problem.

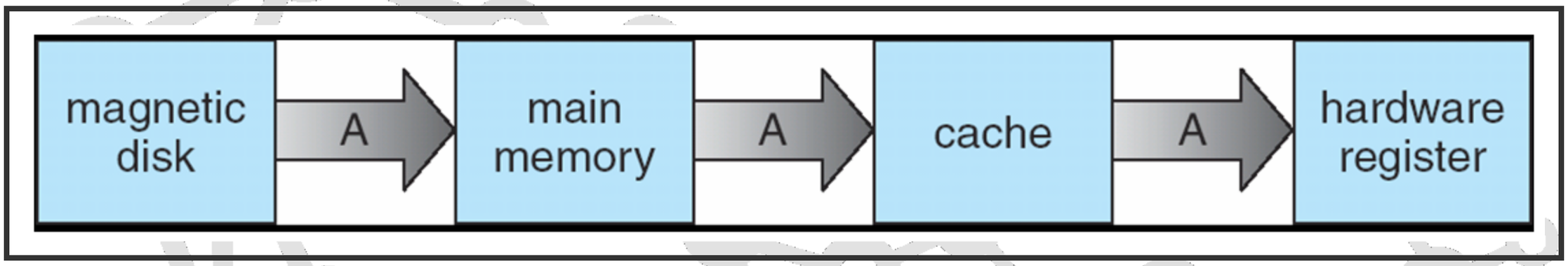
Main memory can be viewed as a fast cache for secondary storage

PERFORMANCE

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

DATA MIGRATION

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



DATA MIGRATION

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

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 VS 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

PROTECTION

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

SECURITY

Huge range of **external attacks**

- denial-of-service
- worms
- viruses
- identity theft
- theft of service

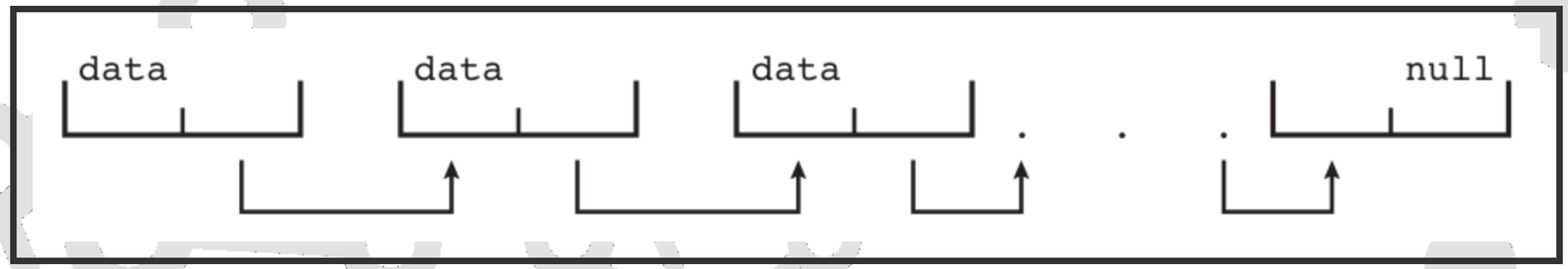
KERNEL DATA STRUCTURES

KERNEL DATA STRUCTURES

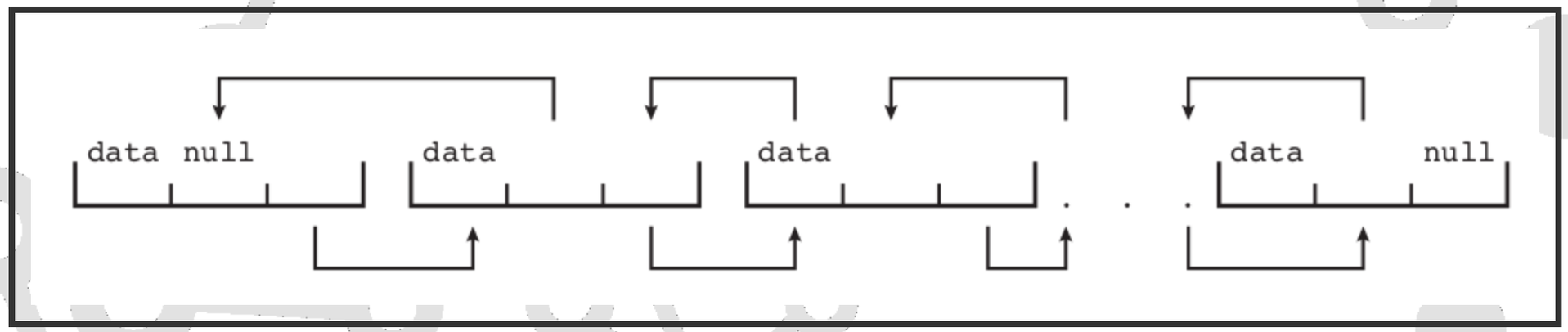
The way data are structured in the system.

Revisit Algorithm and Datastructure course for details

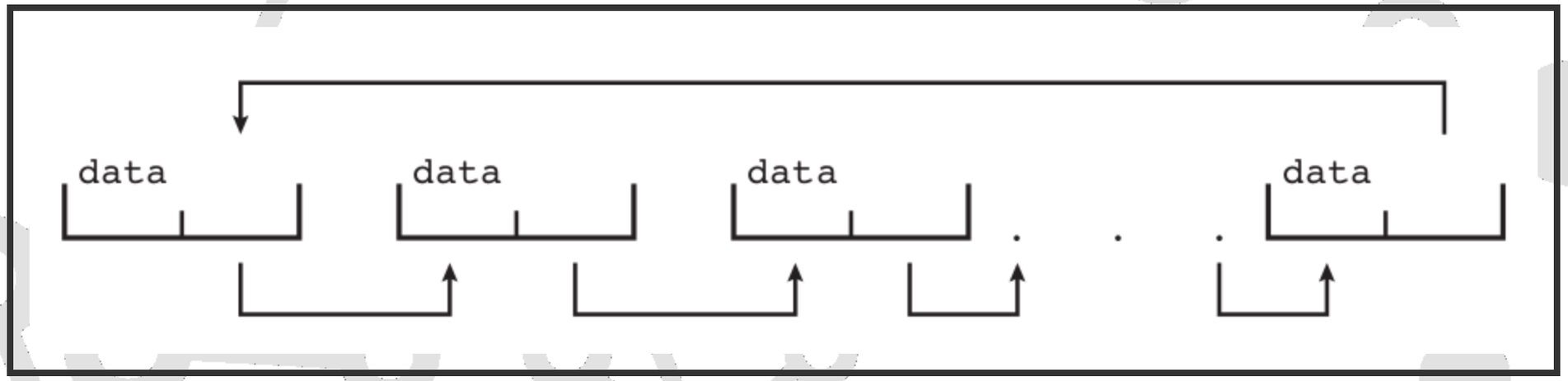
SINGLE LINKED LIST



DOUBLY LINKED LIST



CIRCULAR LINKED LIST



STACK

A **stack** is a sequentially ordered data structure that uses the last in, first out (**LIFO**)

push and **pop** operations

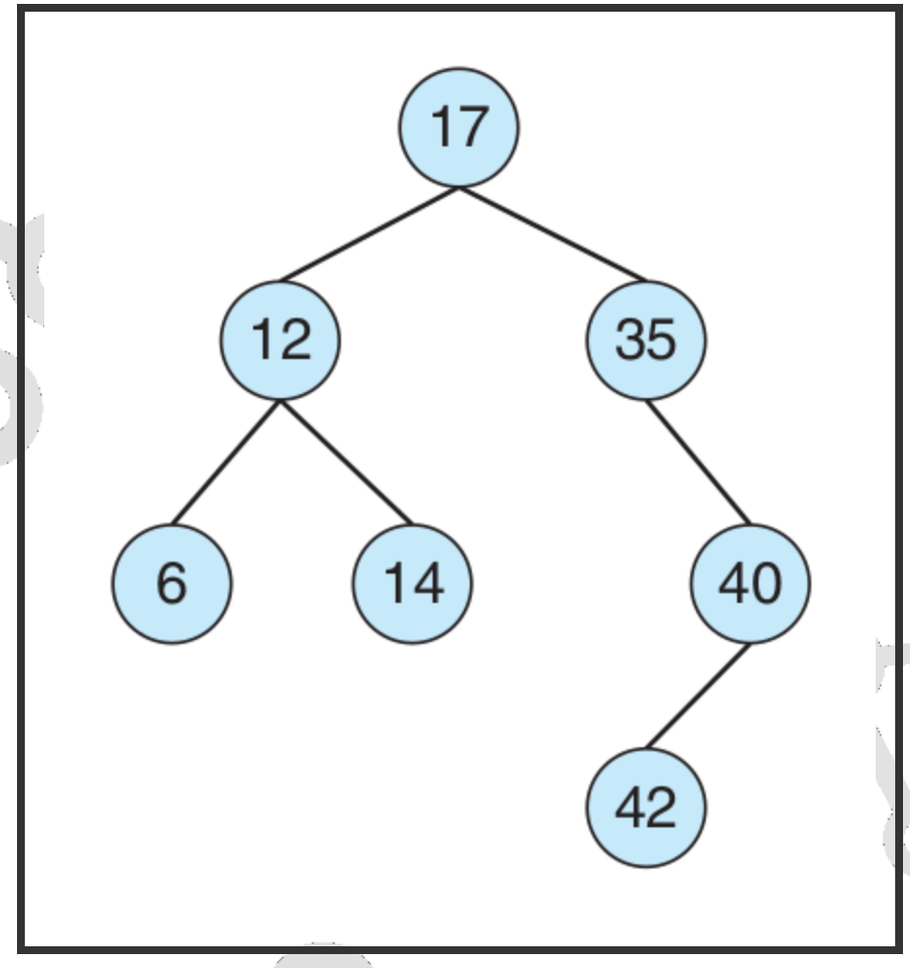
Used when invoking function calls. Parameters, local variables, and the return address are pushed onto the stack when a function is called; returning from the function call pops those items off the stack.

QUEUES

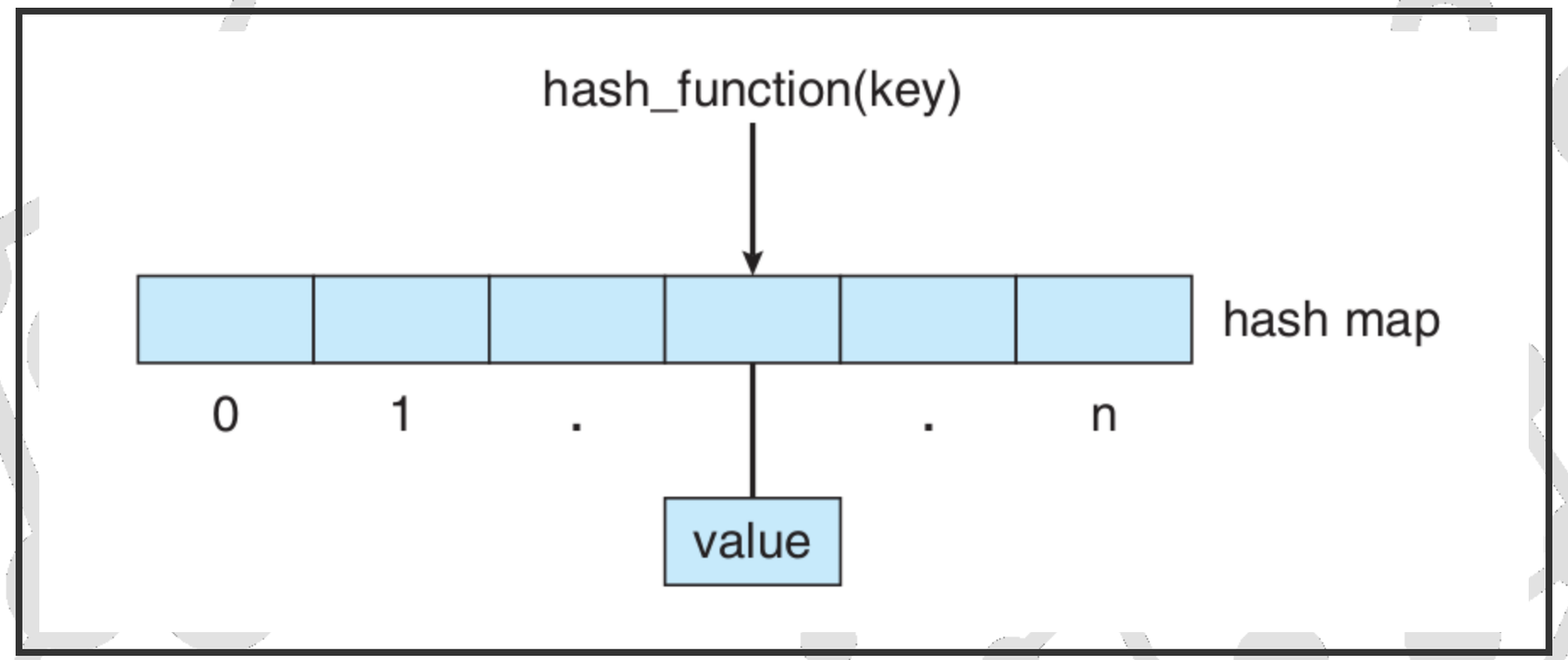
A **queue**, is a sequentially ordered data structure that uses the first in, first out (FIFO) principle

- Jobs that are sent to a printer are typically printed in the order in which they were submitted
- Queue with tasks that are waiting to be run on an available CPU

BINARY SEARCH TREES



HASH MAP



BITMAPS

A **bitmap** is a string of *n* binary digits that can be used to represent the status of *n* items.

Suppose we have several resources, and the availability of each resource is indicated by the value of a binary digit

- 0 means that the resource is available,
- 1 indicates that it is unavailable

COMPUTING 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

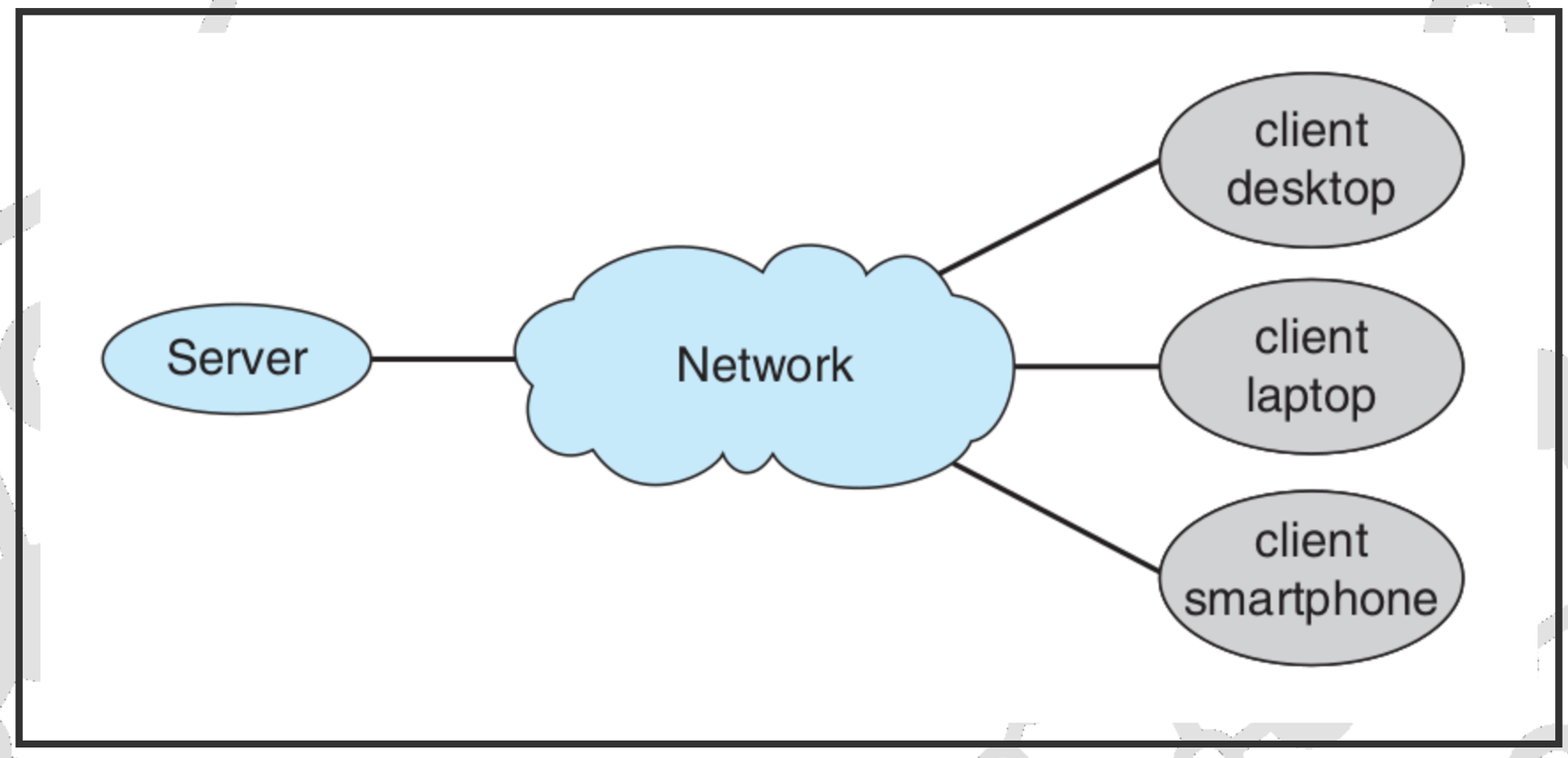
CLIENT-SERVER COMPUTING

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

CLIENT SERVER



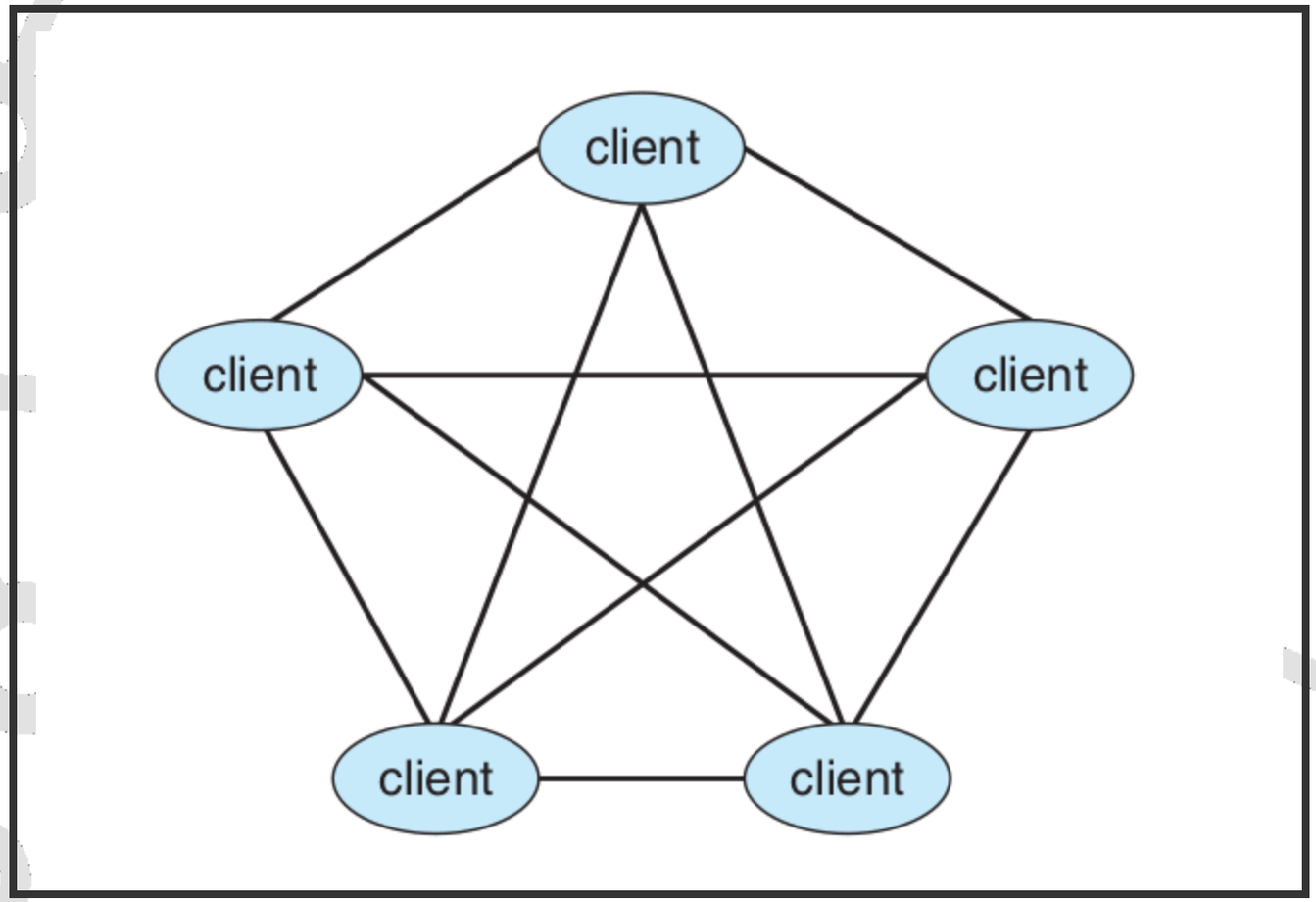
PEER-TO-PEER COMPUTING

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**

PEER-TO-PEER



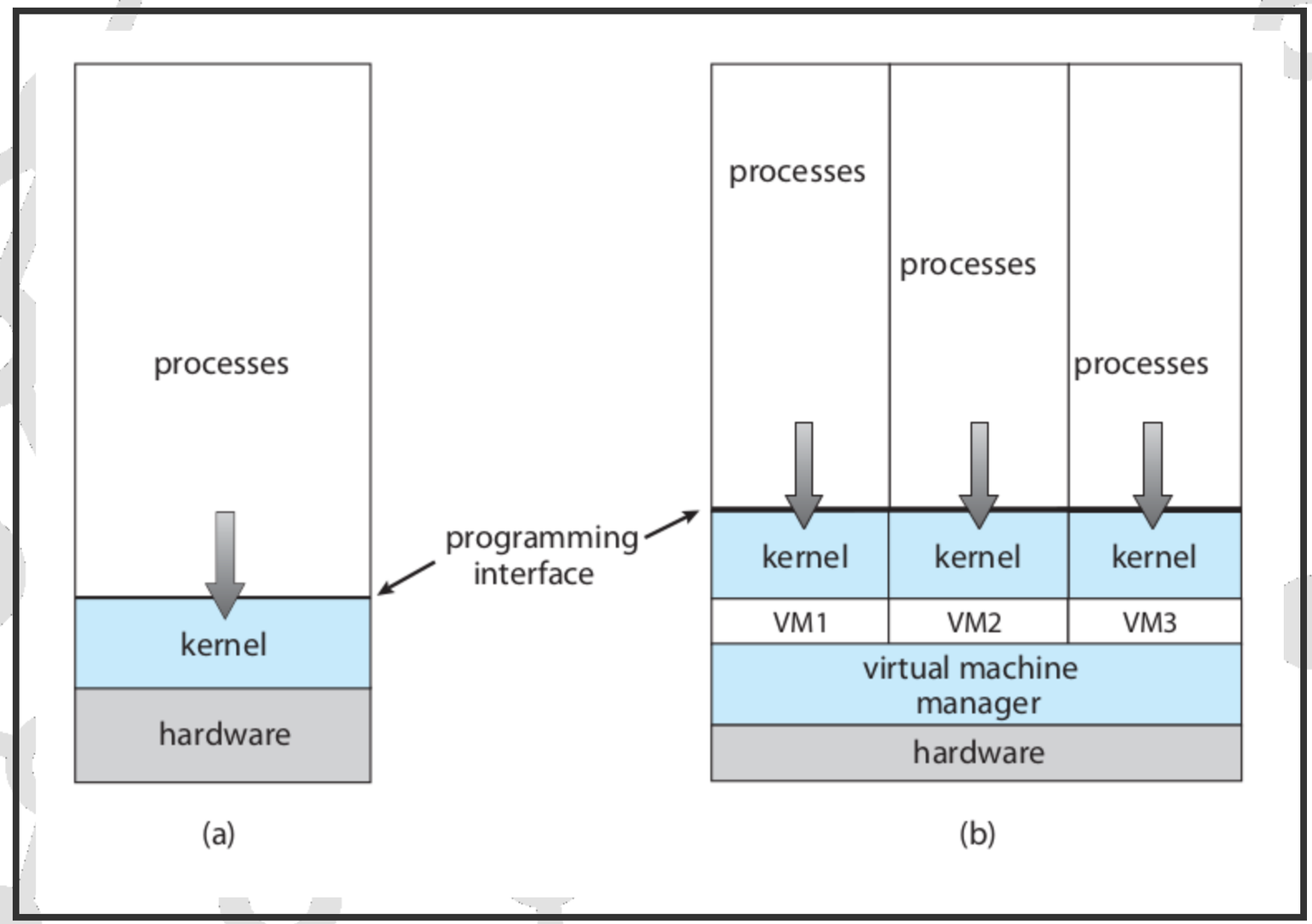
VIRTUALIZATION

Virtualization is a technology that allows operating systems to run as applications within other operating systems.

Emulation is used when the source CPU type is different from the target CPU type.

For example, an Apple laptop running Mac OS X on the x86 CPU can run a Windows guest to allow execution of Windows applications.

VIRTUALIZATION



CLOUD COMPUTING

Cloud computing is a type of computing that delivers computing, storage, and even applications as a service across a network.

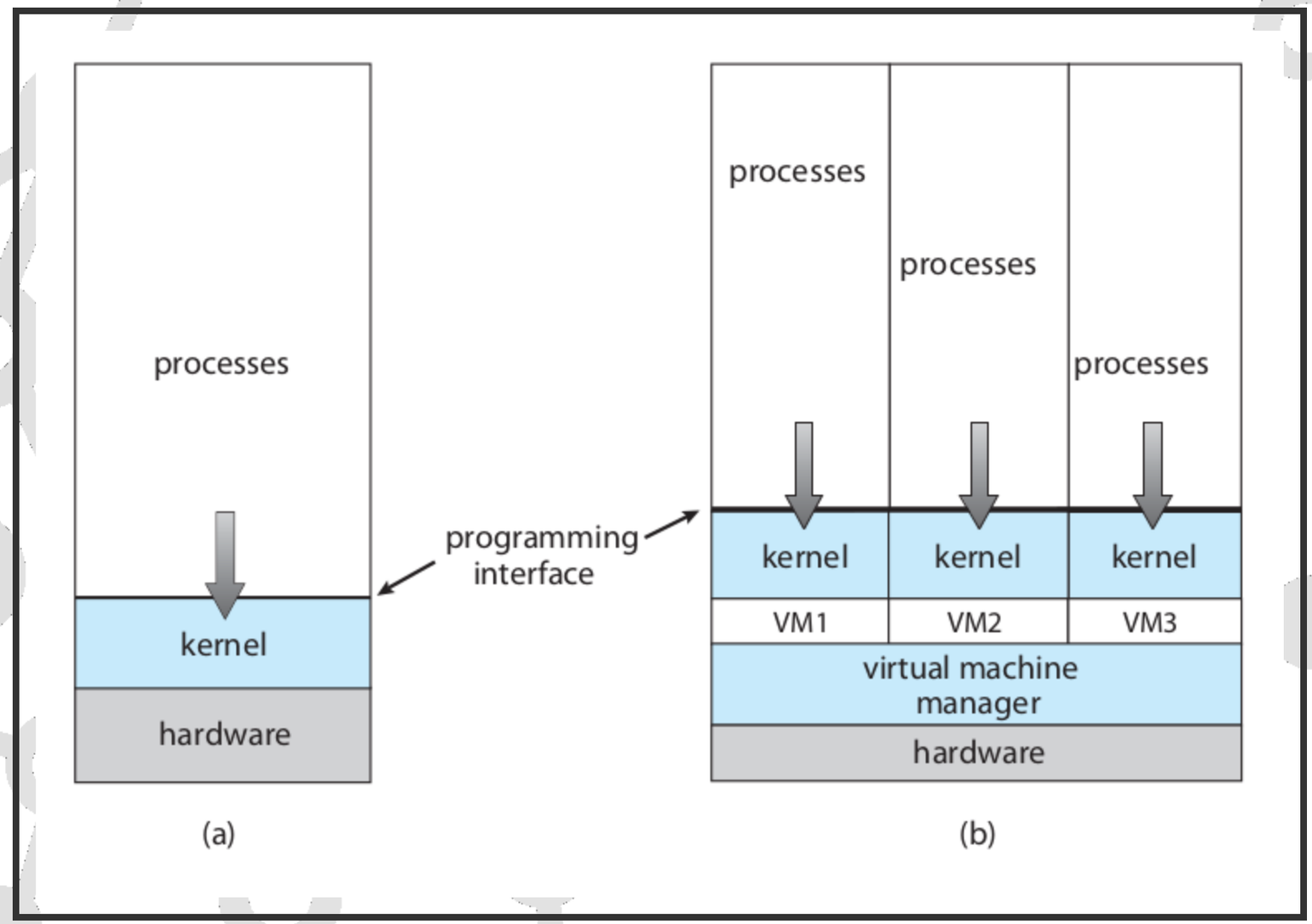
CLOUD COMPUTING TYPES

- Public cloud: available via the Internet to anyone willing to pay for the services
- Private cloud: a cloud run by a company for that company's own use
- Hybrid cloud — includes both public and private cloud components

CLOUD COMPUTING TYPES

- Software as a service (SaaS): one or more applications (such as word processors or spreadsheets) available via the Internet
- Platform as a service (PaaS): a software stack ready for application use via the Internet (for example, a database server)
- Infrastructure as a service (IaaS): servers or storage available over the Internet (for example, storage available for making backup copies of production data)

CLOUD COMPUTING



OPEN-SOURCE OPERATING SYSTEMS

OPEN-SOURCE OPERATING SYSTEMS

Open-source operating systems are those available in source-code format rather than as compiled binary code.

- **Linux** is the most famous open-source operating system
- **Microsoft Windows** is a well-known example of the opposite closed-source approach
- **Apple's Mac OS X and iOS** comprise a hybrid approach.

OPEN SOURCE HISTORY

- 1950s: a great deal of software was available in open-source format.
- Computer and software companies eventually sought to limit the use of their software to authorized computers and paying customers.
- 1978: BSD UNIX started as a derivative of AT&T's UNIX .
- 1994: Due to lawsuit by AT&T, eventually a fully functional, open-source version, 4.4BSD-lite, was released.

OPEN SOURCE HISTORY

Copyrighted material

- copy protection/digital rights management (DRM) wouldn't be effective if source code were published.
- 1983: Richard Stallman started the GNU project to create free, open-source, UNIX - compatible OS.
- 1985: GNU Manifesto: All software should be free and open-sourced. Also formed the Free Software Foundation (FSF)

OPEN SOURCE HISTORY

- 1991: Linus Torvalds, released a rudimentary UNIX-like kernel using the GNU compilers and tools and invited contributions worldwide.
- 2005: Sun open-sourced most of the Solaris code as the OpenSolaris project. (Closed since 1991)

OPEN SOURCE HISTORY