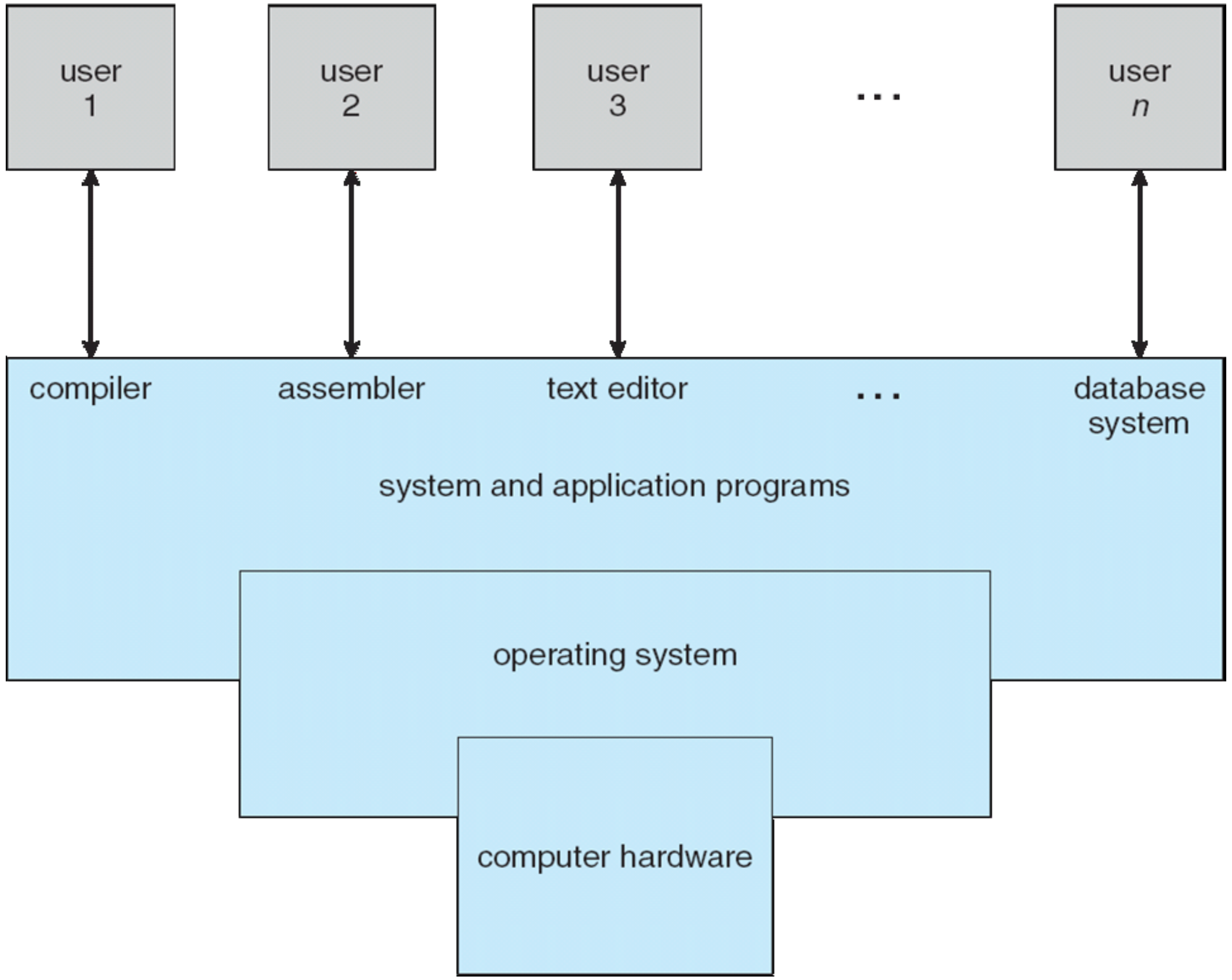
Introduction to Operating Systems

01/17/2023

Professor Amanda Bienz

Computer System

- **Hardware :** basic computing resources
  - CPU, memory, I/O devices
- **Application Programs :** define how resources are used
  - Compilers, web browsers, applications you have written
- **Operating System :** controls hardware and coordinates it’s use among application programs
- **User :** You



Operating System

- **The glue that makes everything work**
- Provides the environment in which *hardware*, *software*, and *data* all work together

Laptop / Desktop

- What are some operating systems for personal computers?
- Designed for one user to monopolize resources (monitor, keyboard, mouse)
- Goals of OS:

Make computer easy to use

Resource utilization

Performance and security

Mobile Device OS

- What are some smartphone / tablet operating systems?
- User interacts with these devices differently than personal computer:
    - Touch screen
    - Voice recognition

User View

- User view : how user interacts with computer
- Think of some devices around your house that have computers
- Do these have a user view, and if so, what is it?

System View

- Can think of the operating system as a **resource allocator**

- Think of an application you have written. What hardware resources were required for your application to run?

CPU time

I/O devices

Memory space

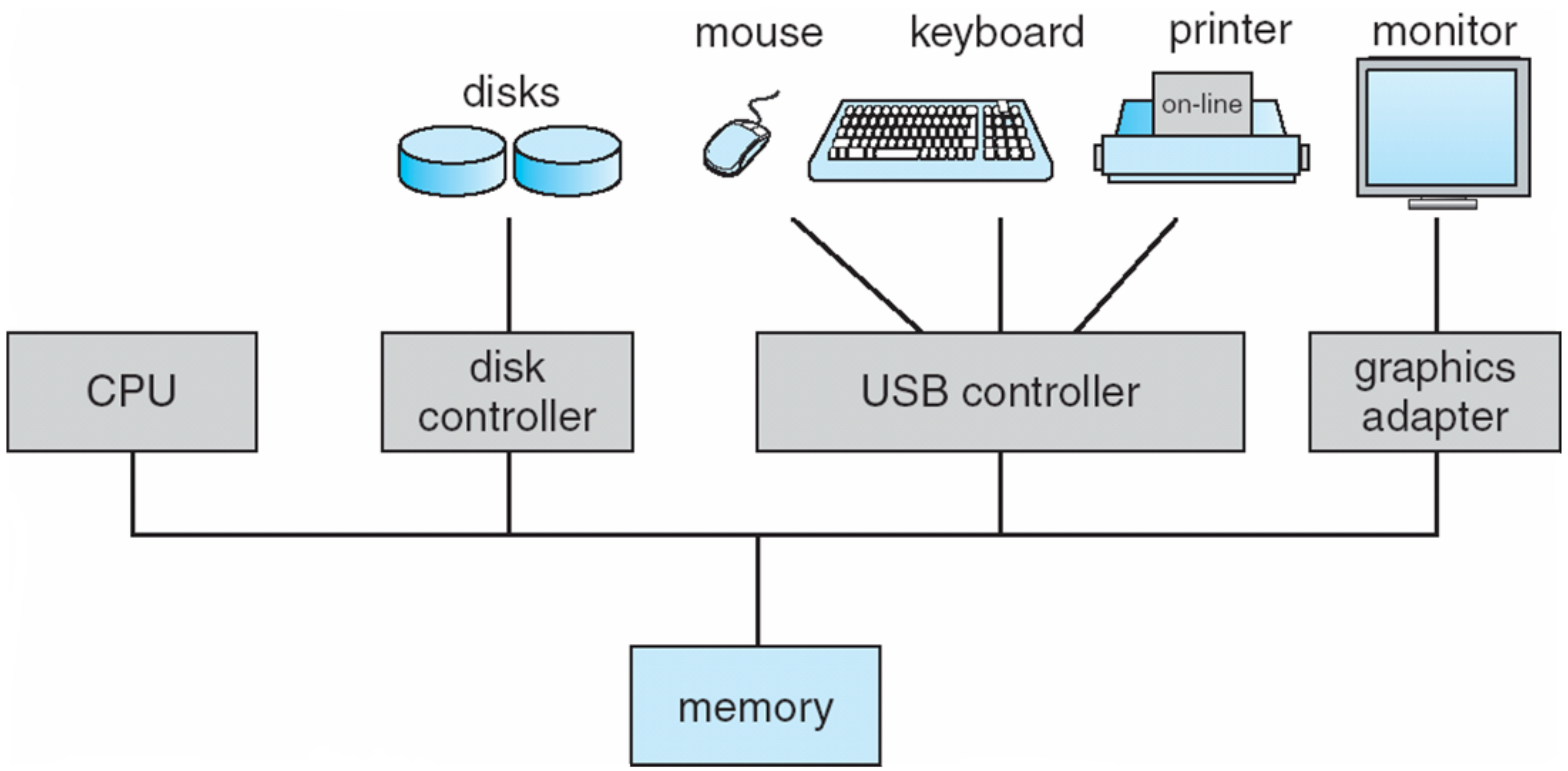
Storage Space

Other resources

- The operating system manages these resources, deciding how to allocate them to different programs and users to operate computer efficiently (and fairly)

Computer-System Overview

- Device controllers and CPU(s) connected through bus
- Device controller has local buffer space and special purpose registers
- Each device controller is responsible for moving data between devices that it controls and its local buffer storage
- Device driver understands the device controller and provides rest of OS with uniform interface to the device
- CPU and device controllers execute in parallel (competing for memory cycles)



I/O Operation

- Suppose I type the ‘w’ key on my keyboard… how does the computer system handle this?
- Device driver loads appropriate registers in device controller
- Device controller reads the content of the registers and decides what action to take (i.e. read from keyboard)
- Controller transfers data to local buffer and then tells the driver transfer has finished
- Driver gives control to other parts of operating system
  - Read : returns data to OS
  - Write : tells OS that write completed successfully
  - Other operations : tells OS that device is busy

I/O Operation

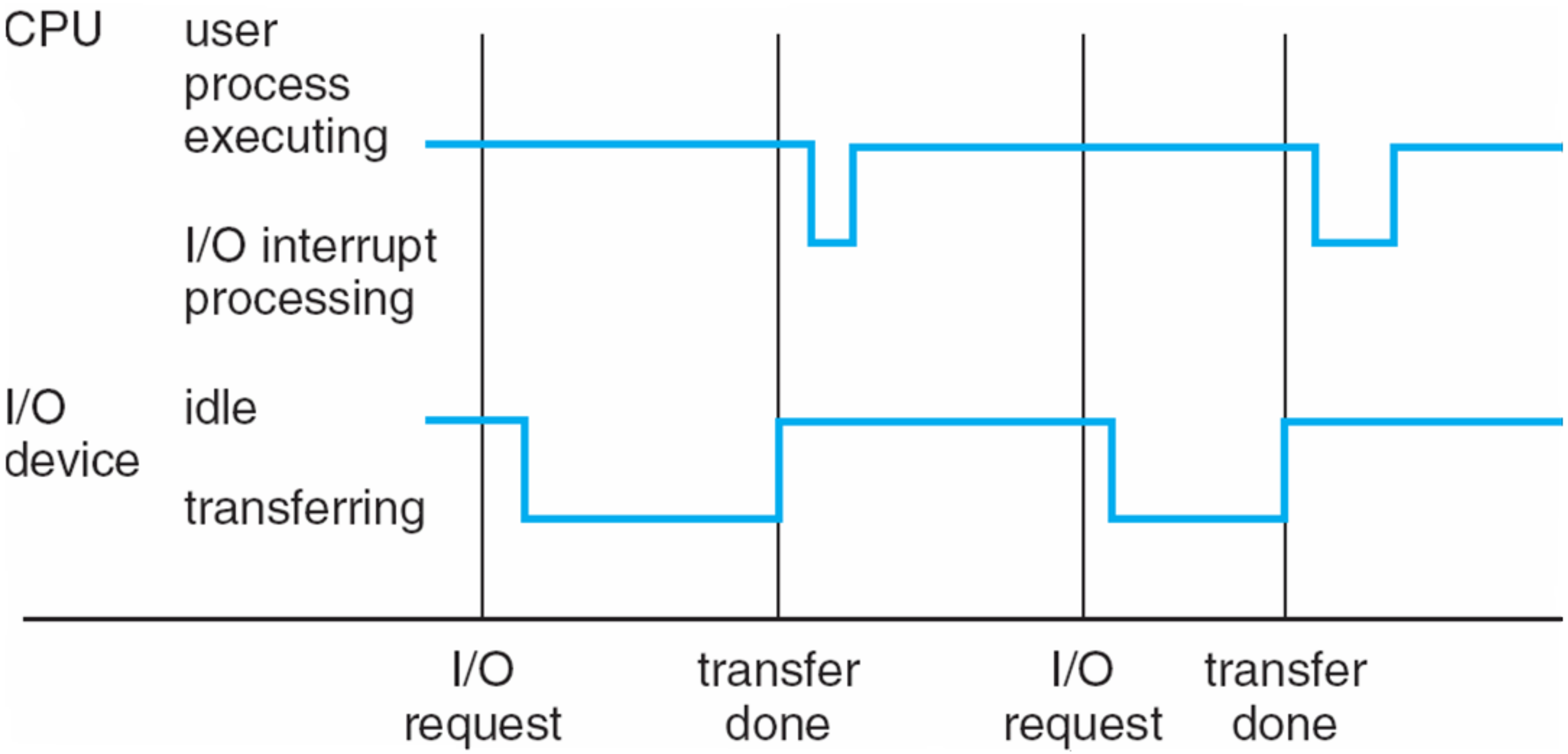
- Suppose I type the ‘w’ key on my keyboard… how does the computer system handle this?
- Device driver loads appropriate registers in device controller
- Device controller reads the content of the registers and decides what action to take (i.e. read from keyboard)
- Controller transfers data to local buffer and then **tells the driver transfer has finished**
- Driver gives control to other parts of operating system
    - Read : returns data to OS
    - Write : tells OS that write completed successfully
    - Other operations : tells OS that device is busy

How does this happen?

Interrupt

- Hardware triggers an interrupt by sending signal to CPU (across bus)
- When CPU is interrupted, it immediately stops what it is doing and executes the interrupt service routine
- The CPU then resumes what it was doing before the interrupt

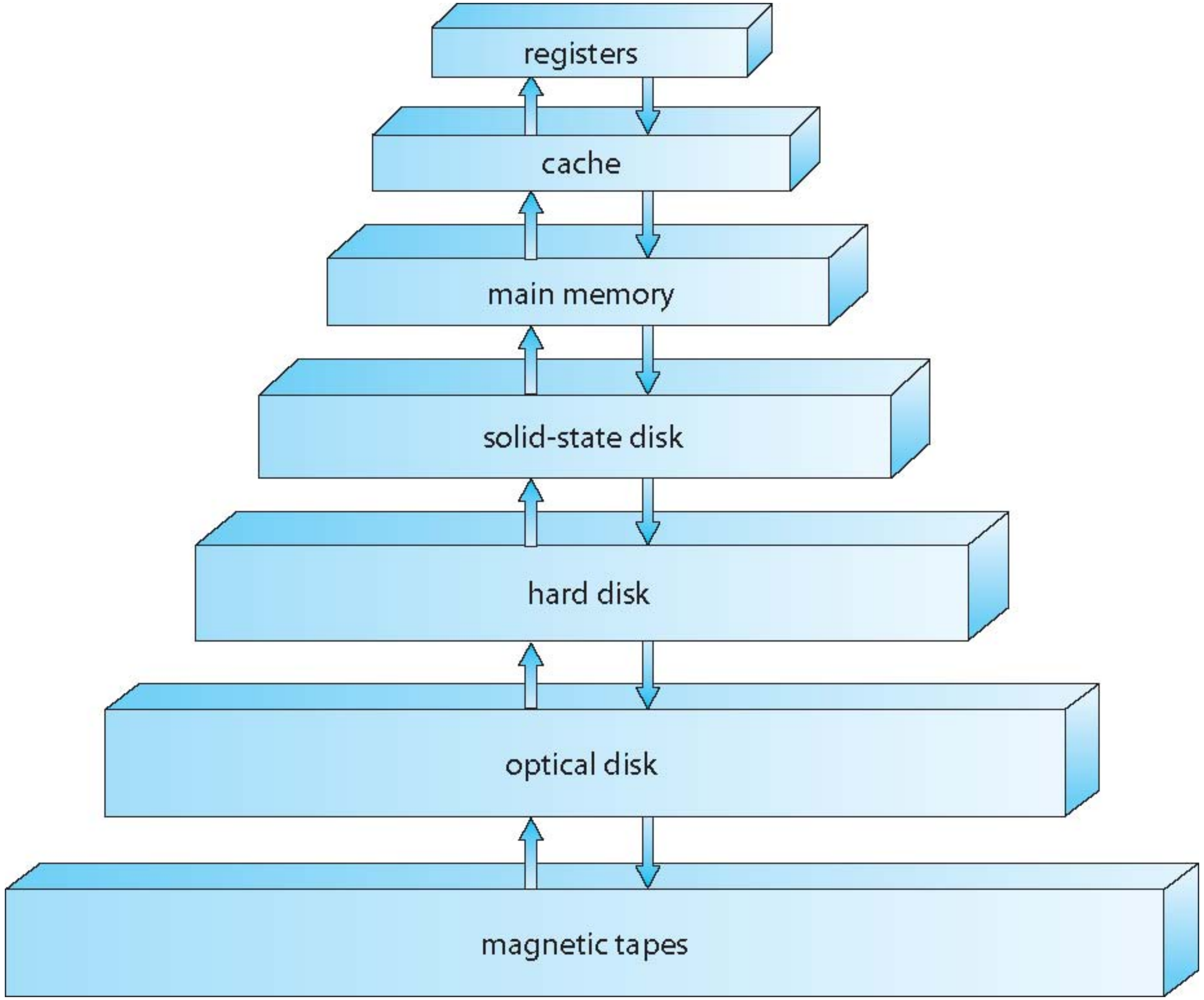
Interrupt


CPU
user process executing
I/O interrupt processing
I/O device
idle
transferring
I/O request
transfer done
I/O request
transfer done

Storage Structure

- CPU can only load instructions from memory
- Programs must be loaded into memory before they can run
- What does memory look like in a general purpose computer?

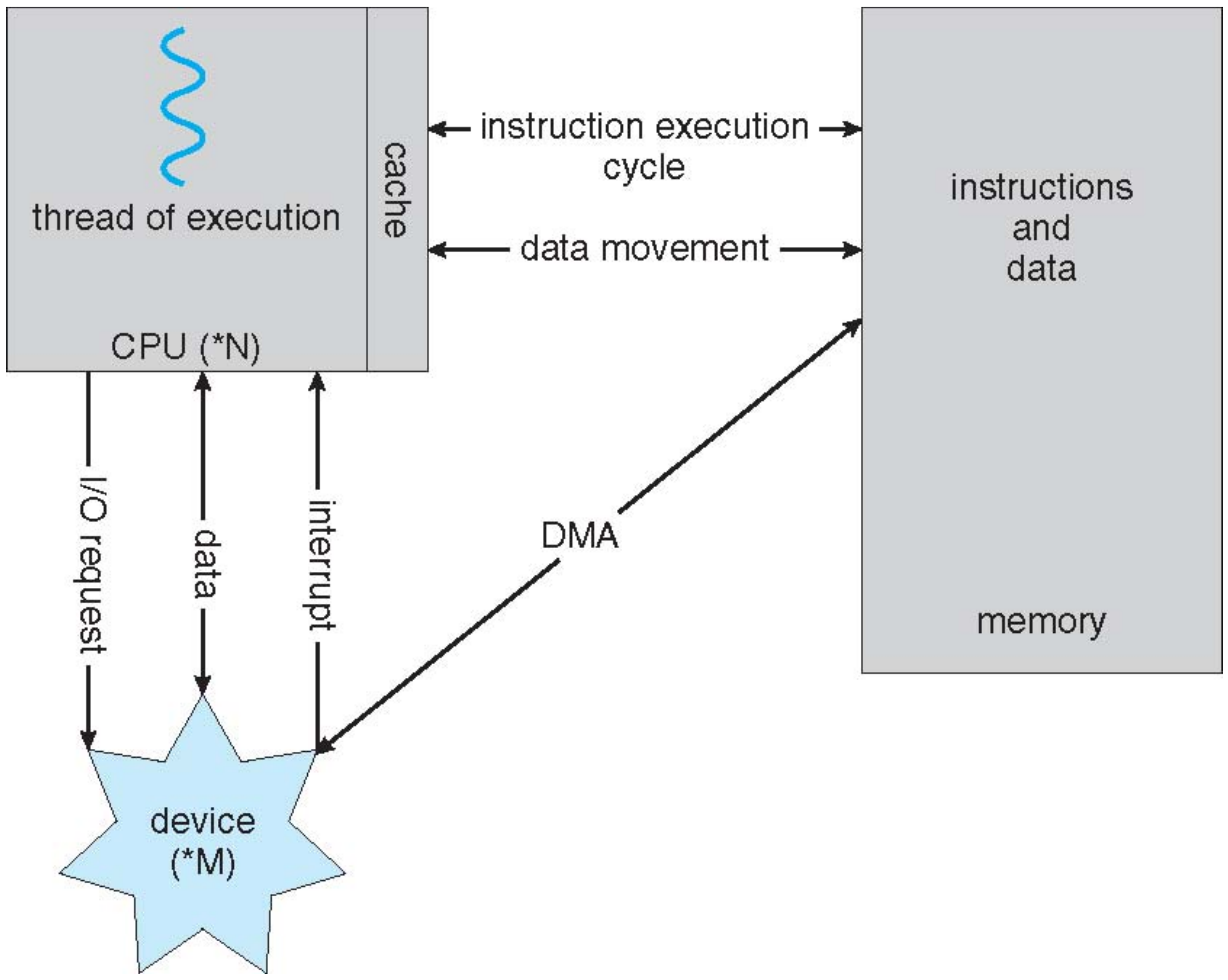
Storage Structure



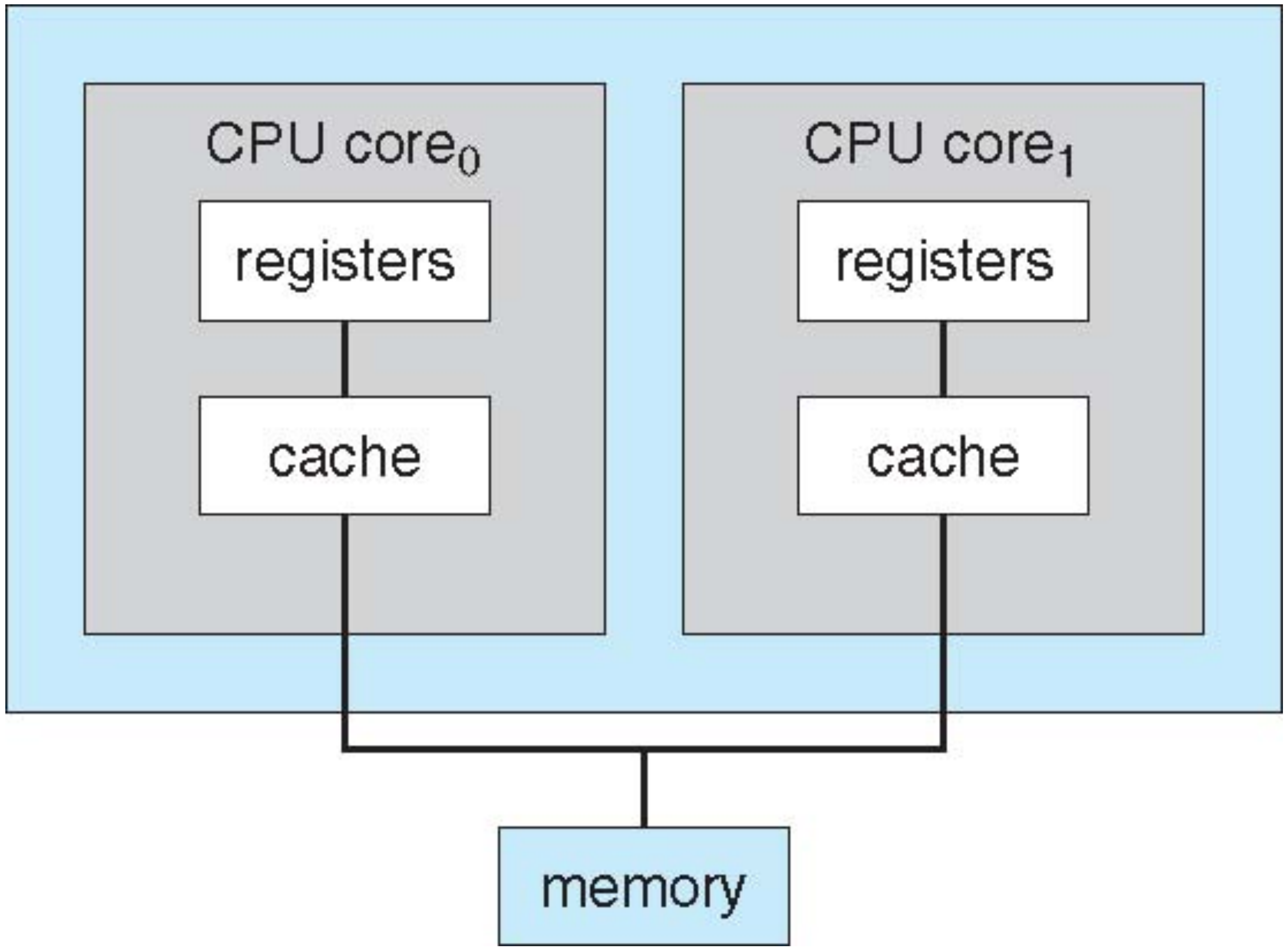
I/O Structure

- Large part of OS is dedicated to managing I/O
  - Important
  - Large variety of devices
- We have discussed interrupt-driven I/O — great for moving small amounts of data
- However, at times I/O will be very very large
- **Direct Memory Access :** device controller transfers entire block of data between device and main memory (CPU is not involved)
  - Only one interrupt - telling driver that transfer has completed

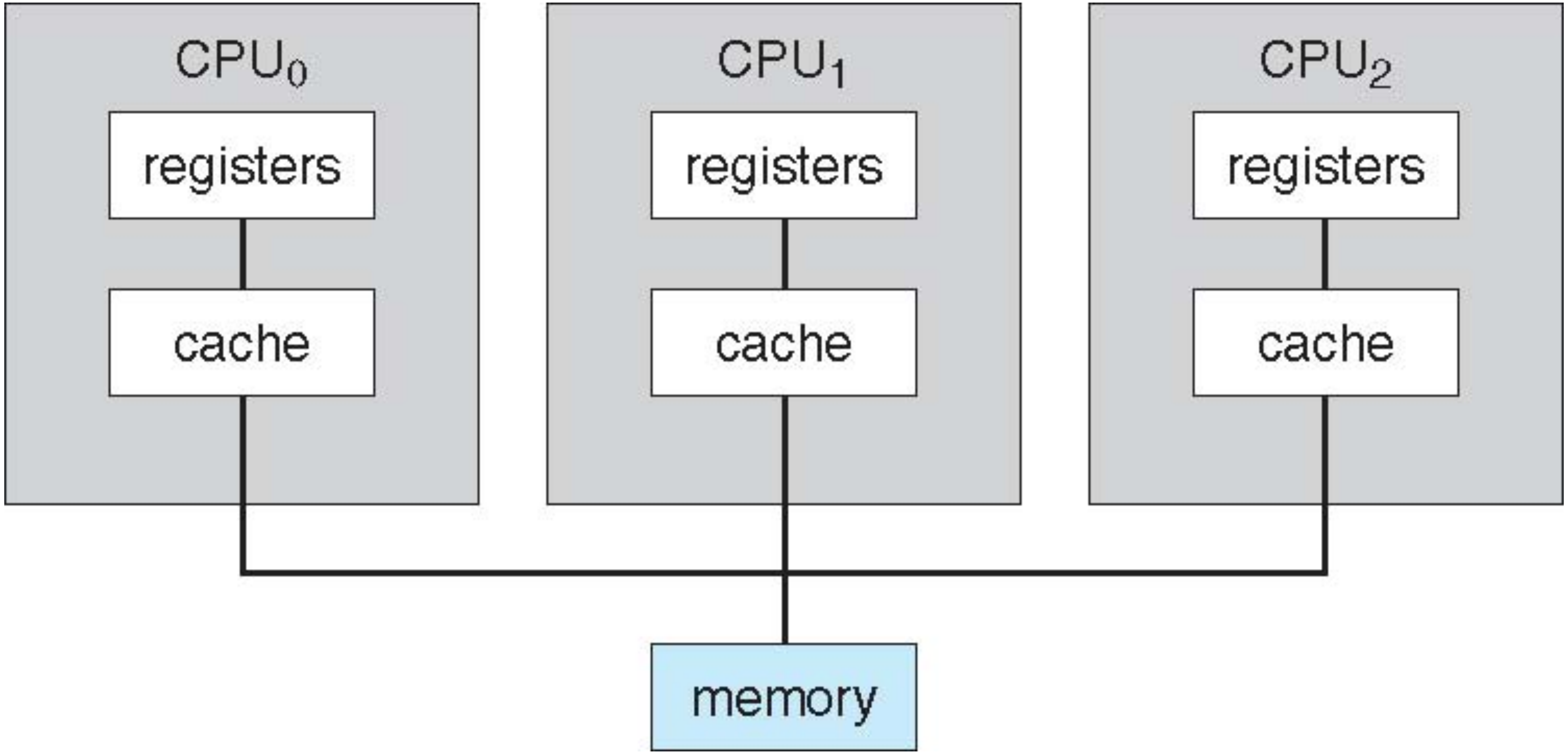
How a Computer Works



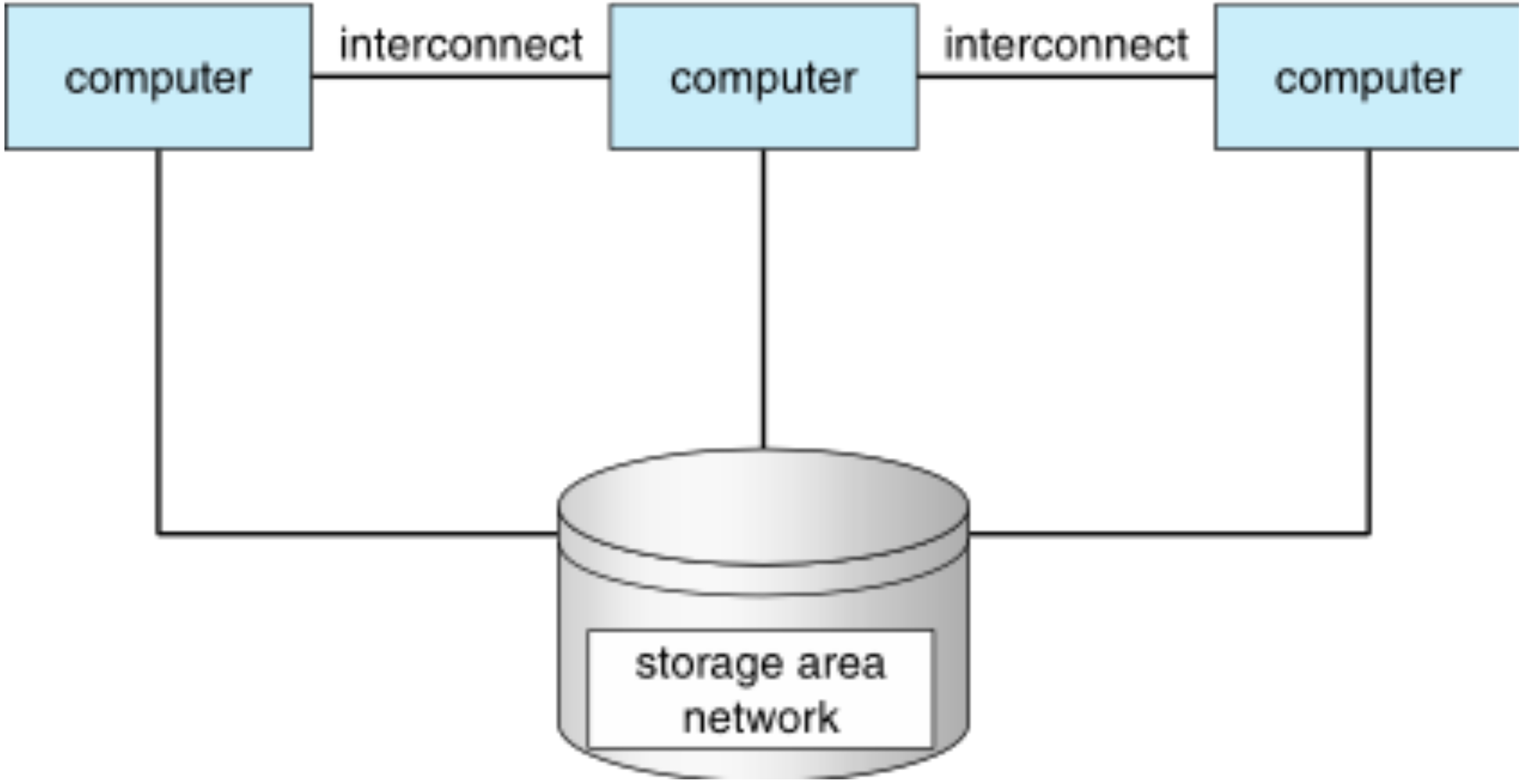
Multicore Processor Architecture



Symmetric Multiprocessing Architecture



Clustered System Architecture

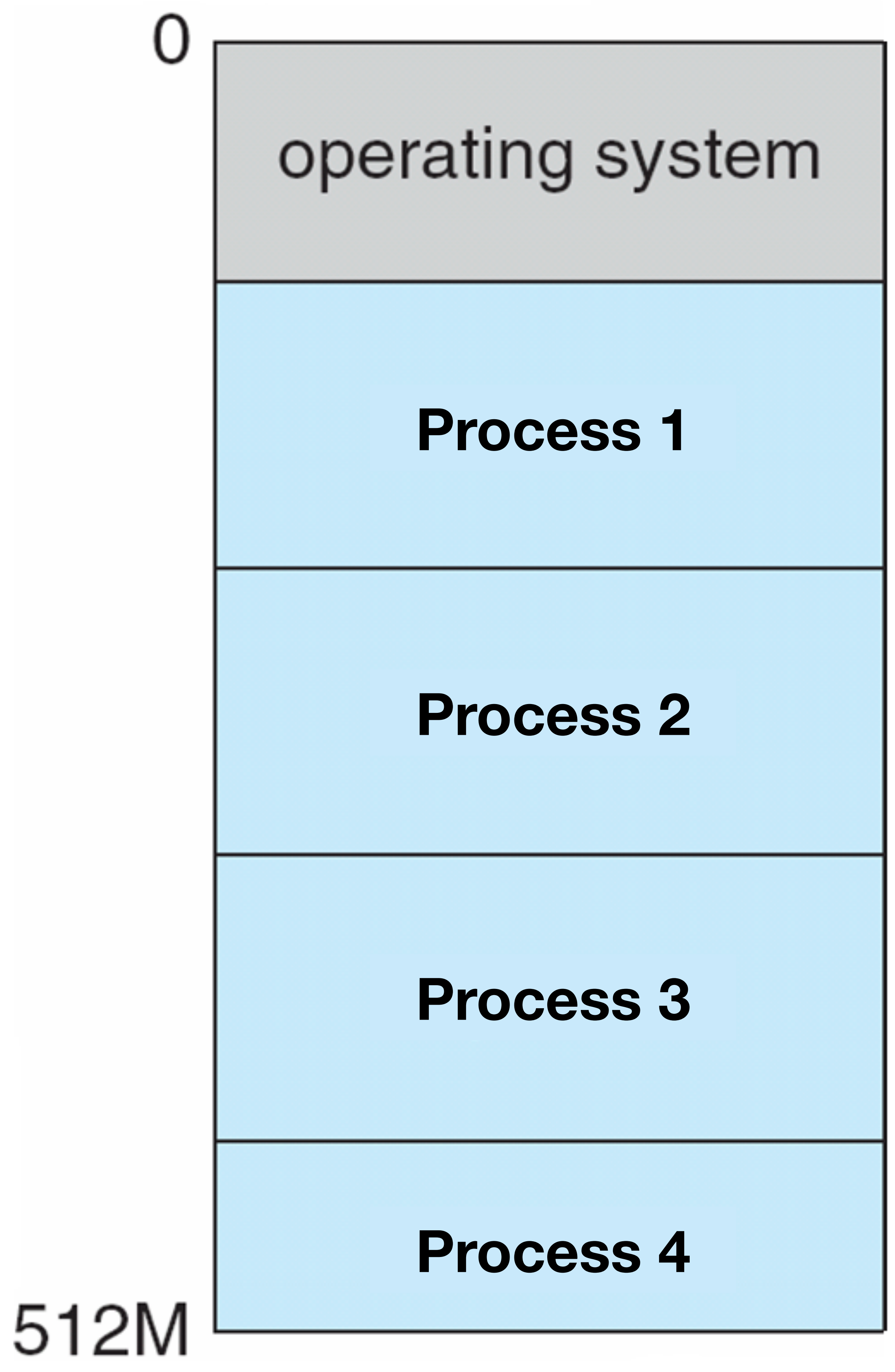


Multiprogramming

- OS has ability to run multiple programs
- A single program cannot keep the CPU or the I/O devices busy at all times
- **Multiprogramming :** organizes programs (code and data) so that CPU always has one to execute
    - Increases CPU utilization
- **Process :** program in execution
- **Job Scheduling :** method of selecting jobs to execute next
- When OS must wait (e.g. I/O), switches to another job

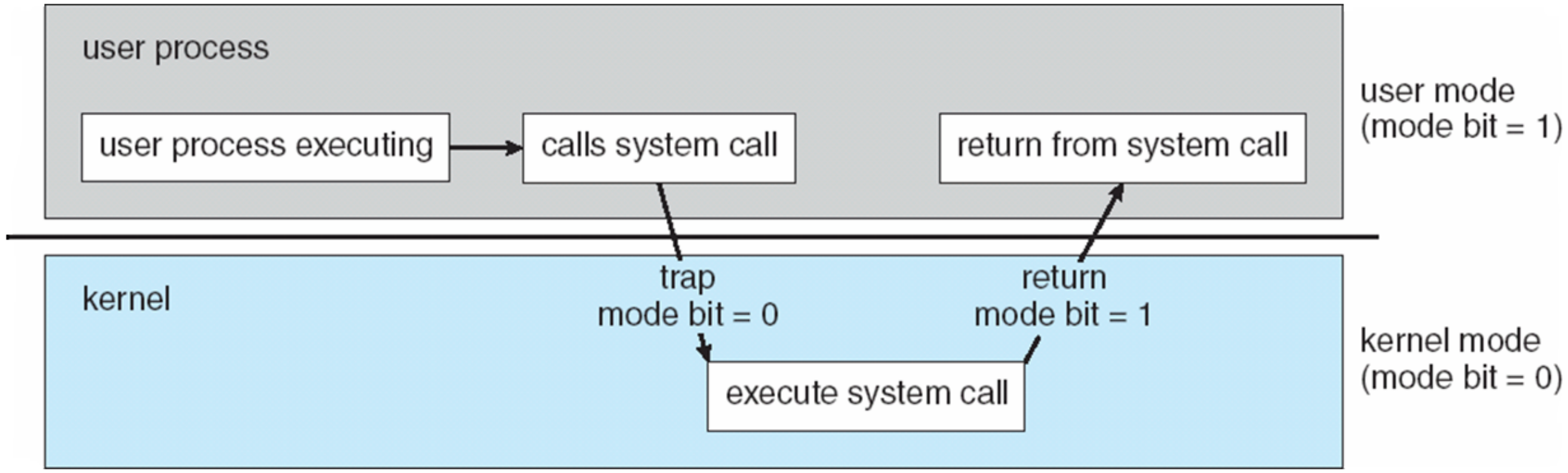
Timesharing (multitasking)

- CPU switches processes so frequently that users can interact with each process while it is running
  - Loads multiple processes into memory (to switch between)
  - I/O waits on user — instead of CPU waiting idly, OS rapidly switches to another process
- **CPU Scheduling :** If multiple jobs are ready to run at the same time, need to schedule when each job should run
- **Swapping :** moving processes in and out of memory
  - Necessary when all processes don't fit in memory
- **Virtual Memory :** allows execution of processes that are not completely in memory



OS Operations

- OS must insure that incorrect or malicious program cannot alter programs (or the OS itself)
- Need to distinguish between execution of OS code and user-defined code
- *User mode* vs *kernel mode*
- Computer hardware has a **mode bit :** kernel (0) and user (1)
- Whenever the operating system gains control of the computer, it is in kernel mode
- **Privileged operations :** only executed in kernel mode



Timer Management

- Operating system must maintain control over the CPU
- Some things that we cannot allow to happen:
  - Program getting stuck in infinite loop
  - Fail to call system services
  - Never return control to operating system
- Timer set to interrupt computer after specified amount of time
- If timer interrupts, control transfers to OS which may treat interrupt as fatal error or might give program more time

Process Management

- Process needs resources:

  I/O Devices CPU Time Memory Files

- Single threaded process has one program counter that specifies which instruction to execute next
- Execution of a process is sequential
- CPU executes one instruction after another until process is complete
- OS is responsible for:
  - Creating and deleting user and system processes
  - Scheduling processes / threads on CPU
  - Suspending and resuming processes
  - Process synchronization and communication

Memory Management

- Main memory : large array of bytes
- **Instruction-fetch cycle :** CPU reads instructions from memory
- **Data-fetch cycle :** CPU reads and writes data from/to memory
- To execute a program, all (or a part) of instructions must be in memory
  - All (or a part) of data needed by program must be in memory
- Memory management : what is in memory, and when?
  - Keep track of which parts of memory are currently being used, and by whom
  - Decide which processes and data to move into and out of memory
  - Allocate and deallocate memory space as needed

File-System Management

- OS provides uniform, logical view of information storage
- Files organized into directory
- OS controls which user can access each file and how they may access it
- OS is responsible for:
  - Create and delete files / directories
  - Primitives to manipulate files / directories
  - Map files onto secondary storage
  - Backup files onto stable (non-volatile) storage

Mass-Storage Management

- Disks used to store :
  - Data that doesn’t fit in main memory
  - Data that must be kept for a long time
- Speed of computer relies on algorithms to access disks
- OS responsibilities:
  - Manage free space
  - Allocate storage
  - Disk scheduling

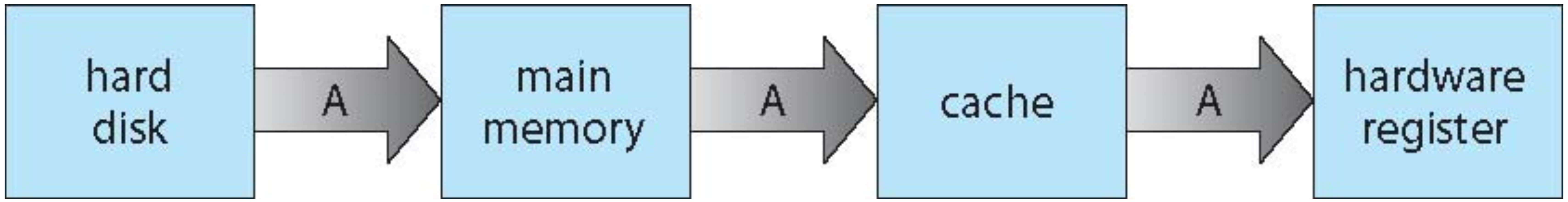
Storage Hierarchy

Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

Cache Management

- Data that was recently accessed from main memory is stored in cache, where it can be accessed more quickly

- Loaded into register to be used immediately



- Multiprocessors need **cache coherency** (all CPUs see same value in cache)

I/O Subsystem

- Operating system hides peculiarities of hardware devices from user
- OS responsibilities:
  - Memory management of I/O
    - Buffering : storing data temporarily while being transferred
    - Caching : storing parts of data in faster storage for performance
    - Spooling : overlapping output of one job with input of other jobs
  - General device-driver interface
  - Drivers for specific hardware devices