CSC 4320/6320: Operating Systems

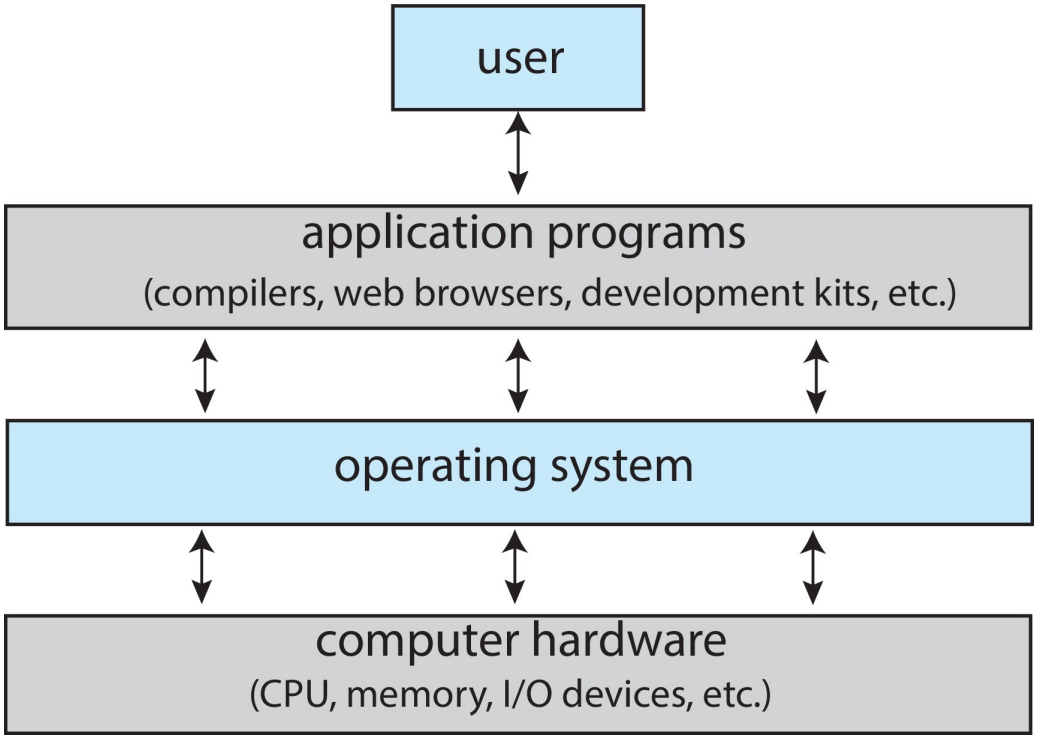
Georgia State University

Chapter 01: Introduction (Part 2)

Instructor: Dr. Md Mahfuzur Rahman
Department of Computer Science, GSU

Review: What is an OS?

An operating system is “fill in the blanks” between … and …?



MS Word I/O Interrupt Example (W)

Georgia State University

Imagine a user presses "Print" to send a document to the printer.

1. MS Word sends the print request to the printer's device controller.
- The device controller is responsible for the job by printer hardware.

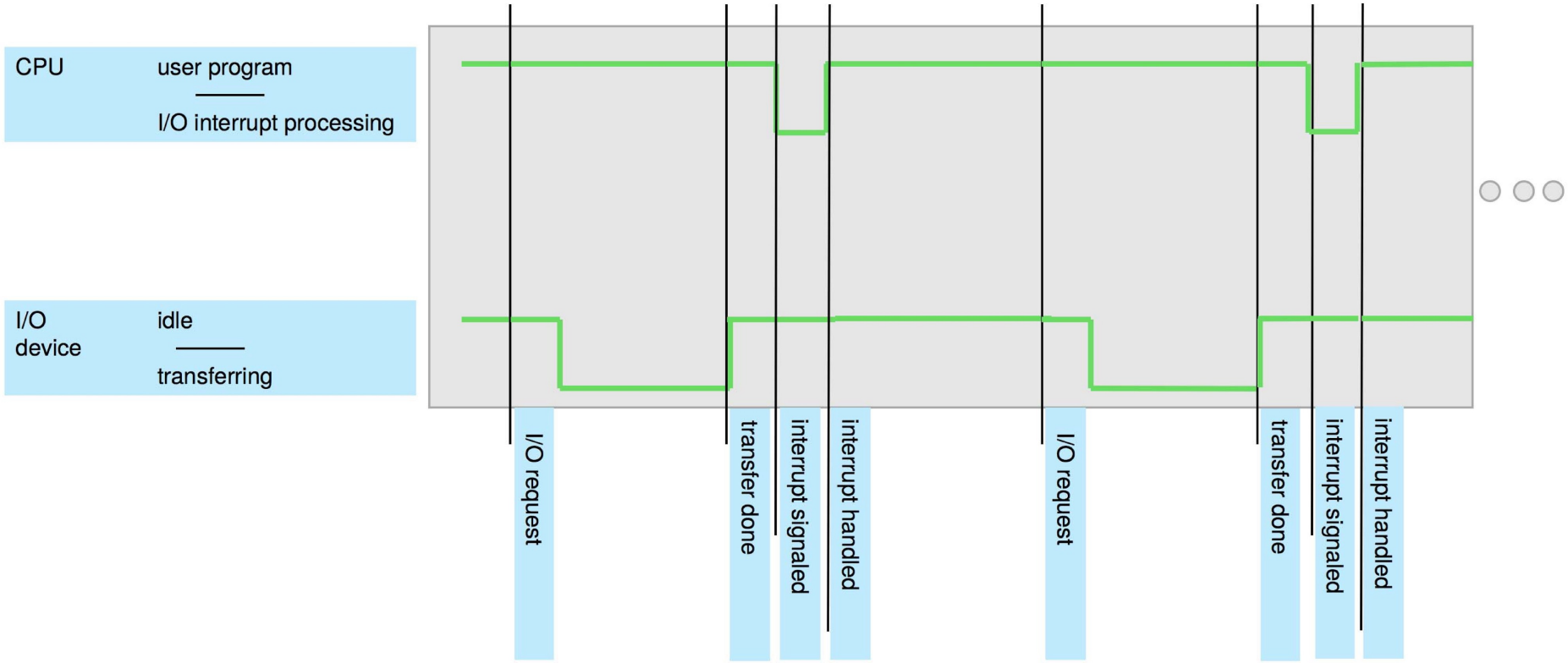
2. MS Word comes back for other tasks (continue typing, editing, etc.)
- Asynchronous I/O. Printer does its job independently.

3. Once the printer has finished its job:
- the printer's device controller sends an **interrupt** signal to the CPU. (i.e., DONE)

4. OS detects the interrupt and executes a specific routine (Interrupt Service Routine)
- update the printer status (e.g., "Printing complete") and/or
- sending a notification back to MS Word.

5. Application may generate further feedback:
- display a message like "Printing completed successfully"

Interrupt Timeline



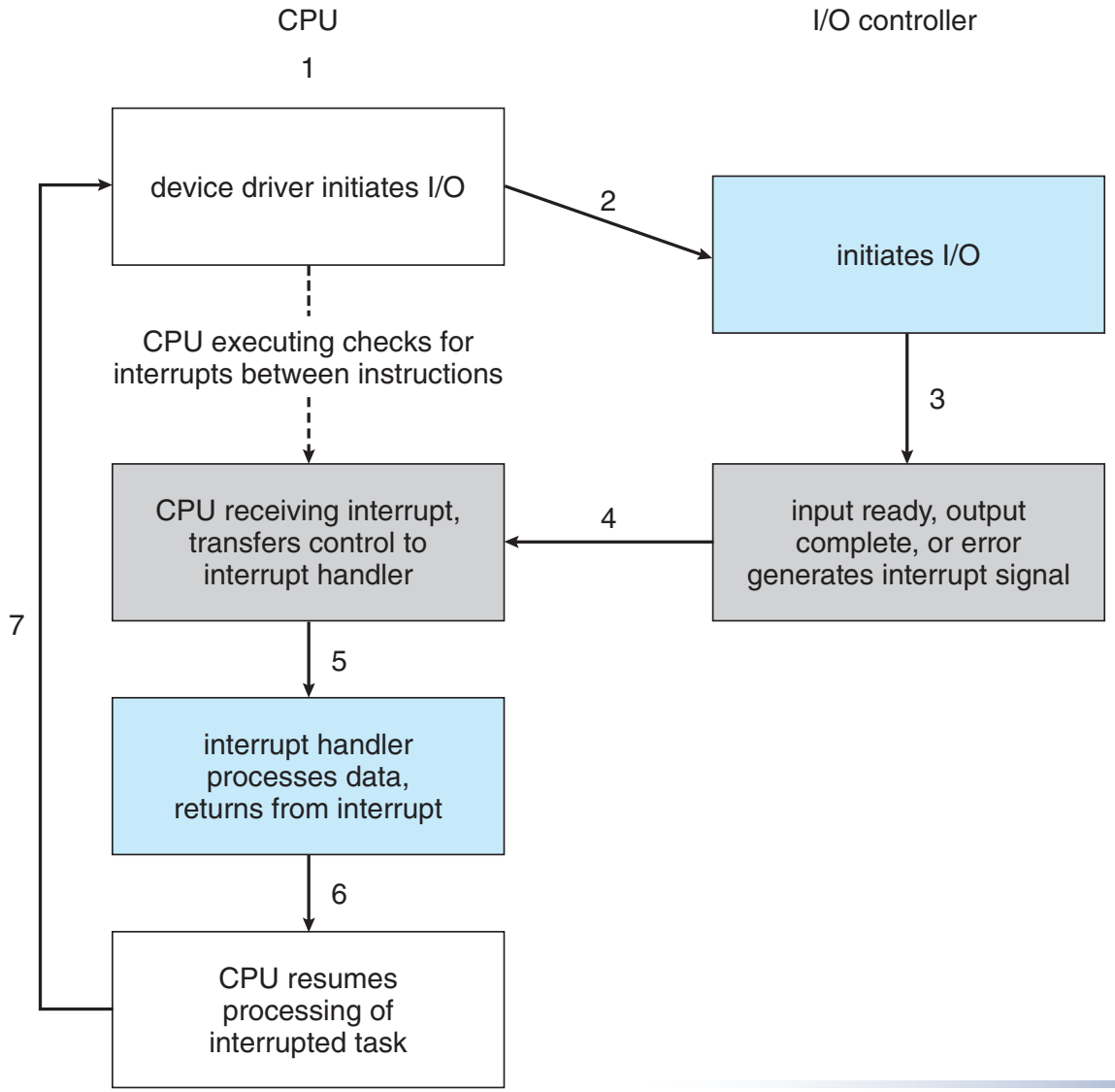
MS Word I/O Interrupt Example – (R)

1. When you press a key, the keyboard's device controller detects the keystroke.
2. Key is temporarily stored in the **keyboard controller's local buffer**.
3. The keyboard's device controller generates an **interrupt** to inform the CPU that a key has been pressed. (Ready to process)
4. The CPU pauses its current task and executes an **Interrupt Service Routine (ISR)** specific to the keyboard input.
5. The ISR processes the keycode (moves the data to a designated location)
6. OS Passes Data to the Application
7. Microsoft Word Updates the Document (visibility)
8. The CPU resumes its tasks until the next interrupt is received (e.g., another key press).

Interrupt Handling

- The operating system preserves the state of the CPU by storing the registers and the program counter
- Determines which type of interrupt has occurred:
- Separate segments of code determine what action should be taken for each type of interrupt

Interrupt-driven I/O Cycle



Nonmaskable and maskable interrupt

Georgia State University

Most CPUs have two interrupt request lines:

- **Nonmaskable interrupt**: Reserved for events such as unrecoverable memory errors.
- **Maskable interrupt:** it can be turned off by the CPU before the execution of critical instruction sequences that must not be interrupted. The maskable interrupt is used by device controllers to request service.

Some Important Terminologies

Interrupt - A hardware mechanism that enables a device to notify the CPU that it needs attention.

interrupt-request line - The hardware connection to the CPU on which interrupts are signaled.

interrupt address / interrupt number - A variable provided along with an interrupt signal to indicate the source of the interrupt.

interrupt service routine (ISR) / interrupt-handler routine - An operating system routine that is called when an interrupt signal is received.

interrupt vector - An operating-system data structure indexed by interrupt address and pointing to the interrupt handlers.

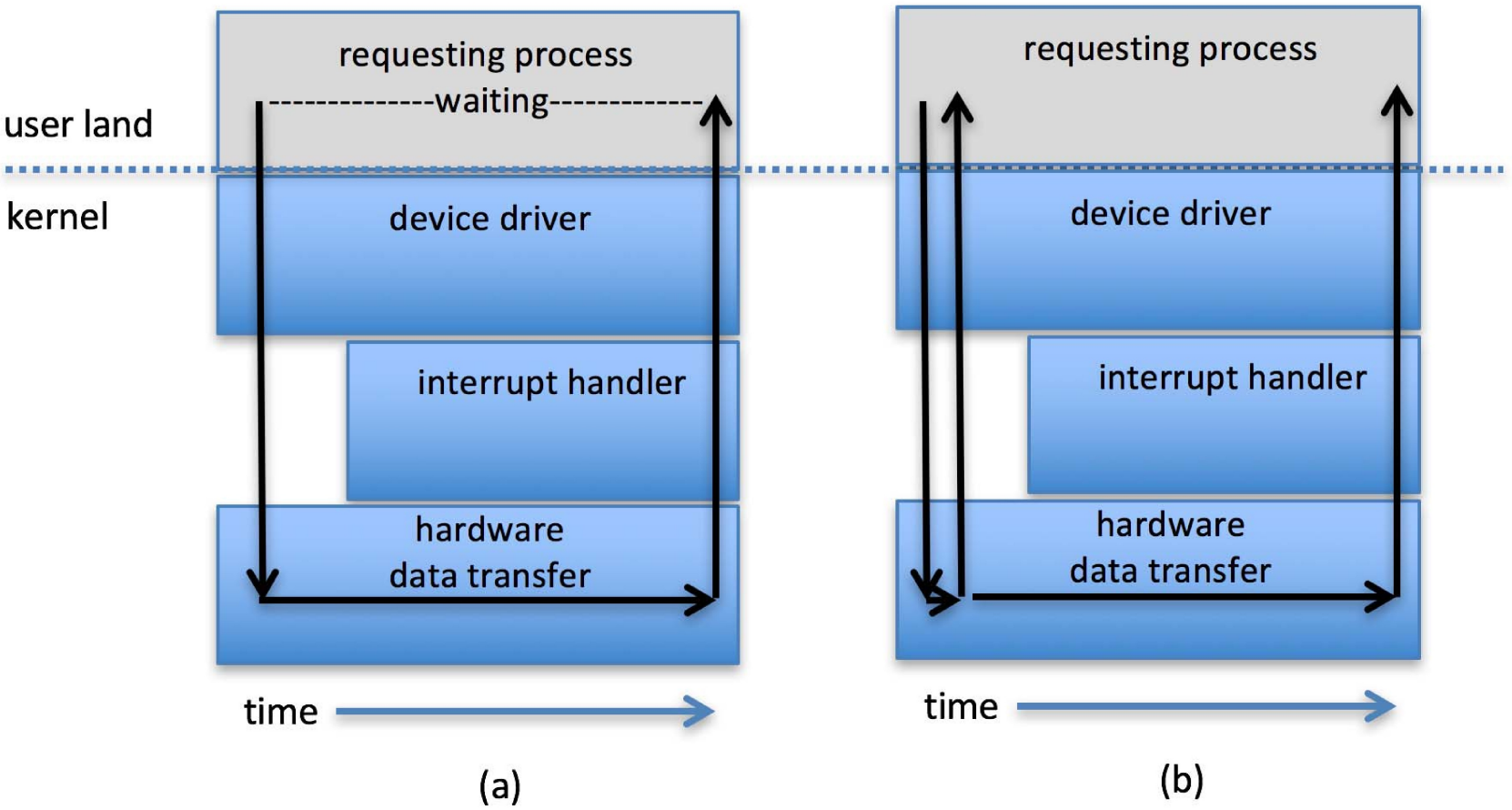
Intel processor event-vector table

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

I/O Structure

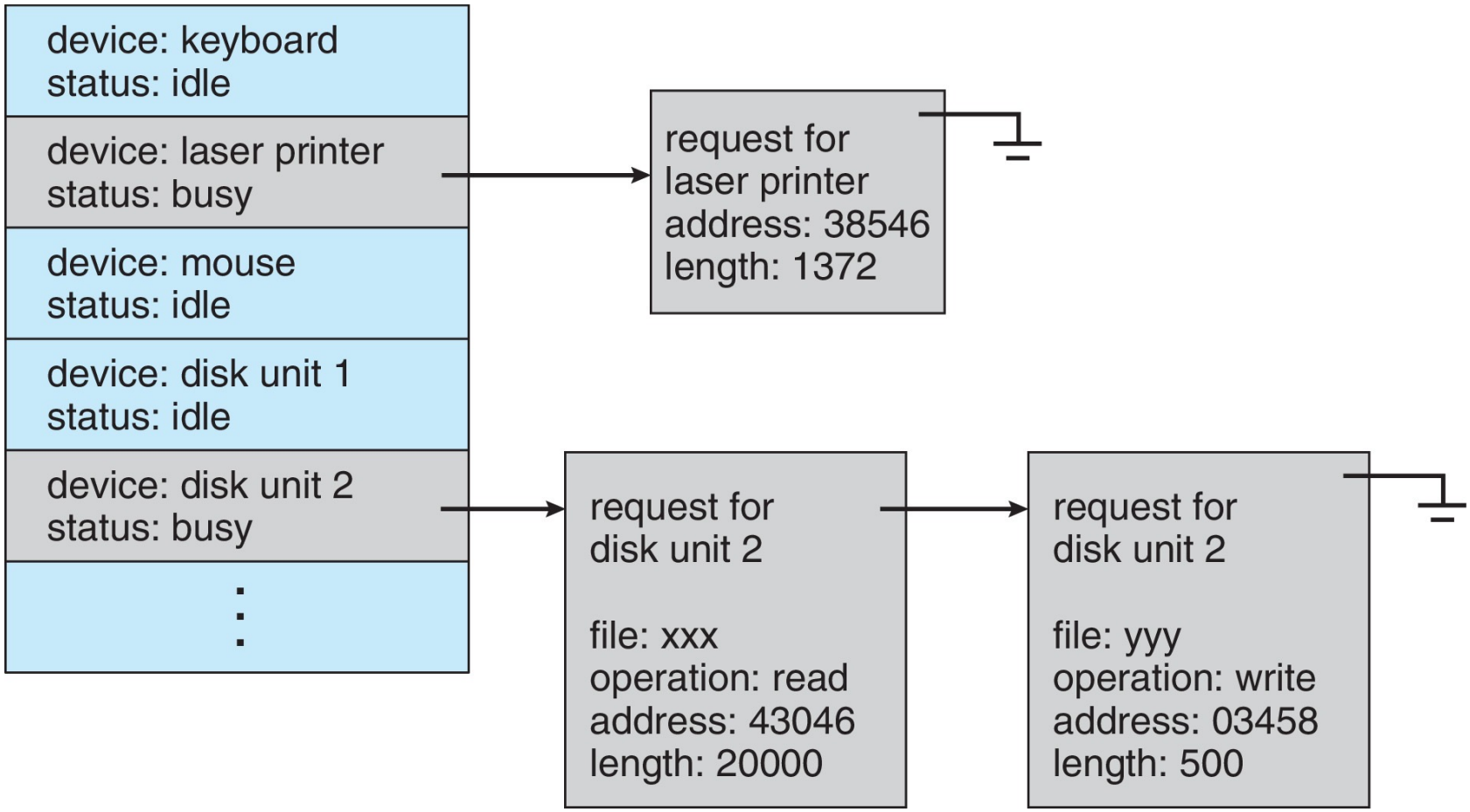
- Two methods for handling I/O
  - After I/O starts, control returns to user program only upon I/O completion
  - After I/O starts, control returns to user program without waiting for I/O completion

Two I/O Methods



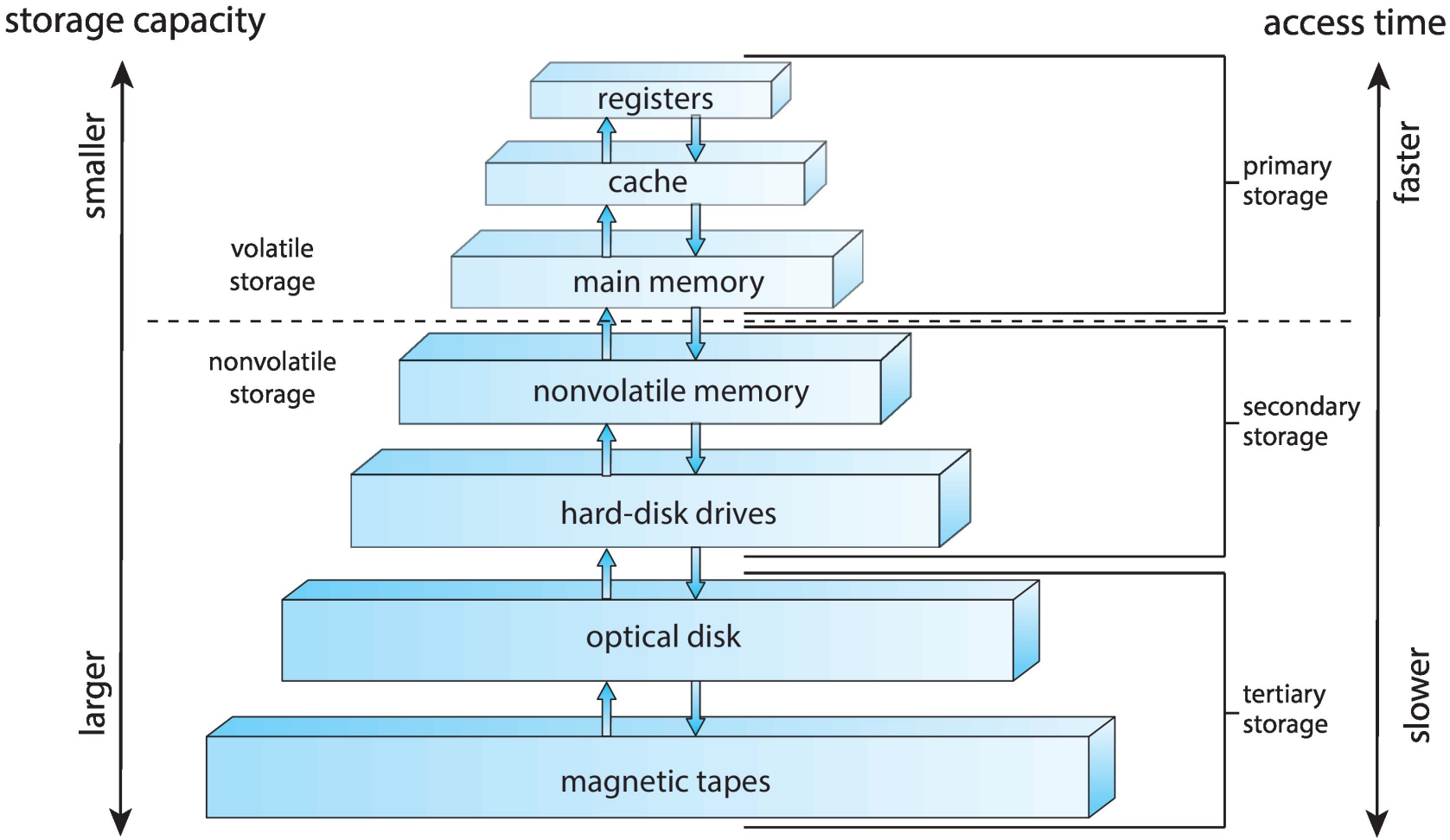
(a) synchronous and (b) asynchronous.

Device-status Table



For asynchronous I/O, the kernel manages this table, which contains an entry for each I/O device, showing the status and related parameters

Storage-Device Hierarchy



Questions

1. The most common secondary storage device is ____.

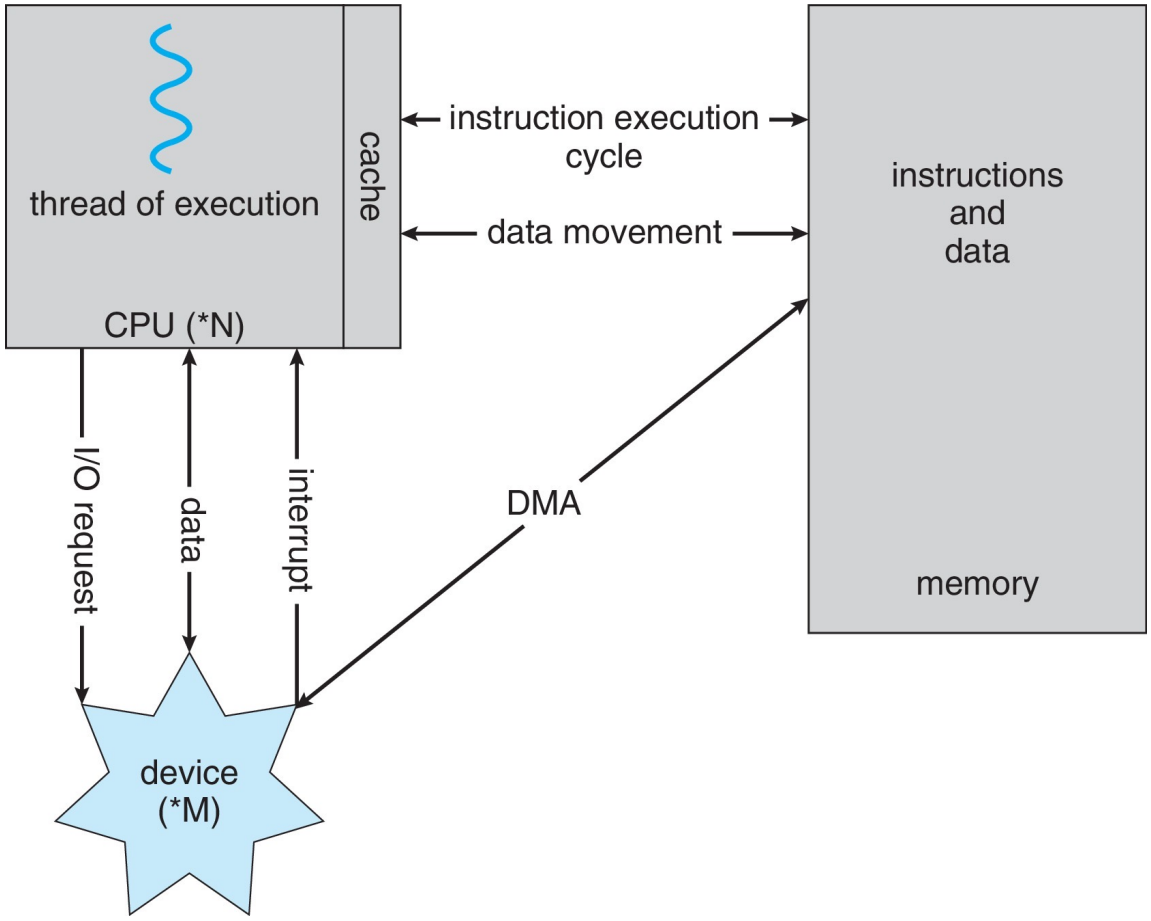
A) random access memory
B) solid state disks
C) tape drives
D) magnetic disk ✓

2. Interrupts may be triggered by either hardware or software

True ✓
False

How a Modern Computer Works

1. CPU reads and executes instructions – writes/reads data to / from main memory
2. An instruction may trigger an I/O request
3. Data transfer happens and interrupt is signaled.
4. The I/O request may also trigger a DMA transfer request between a device and the main memory.



A von Neumann architecture

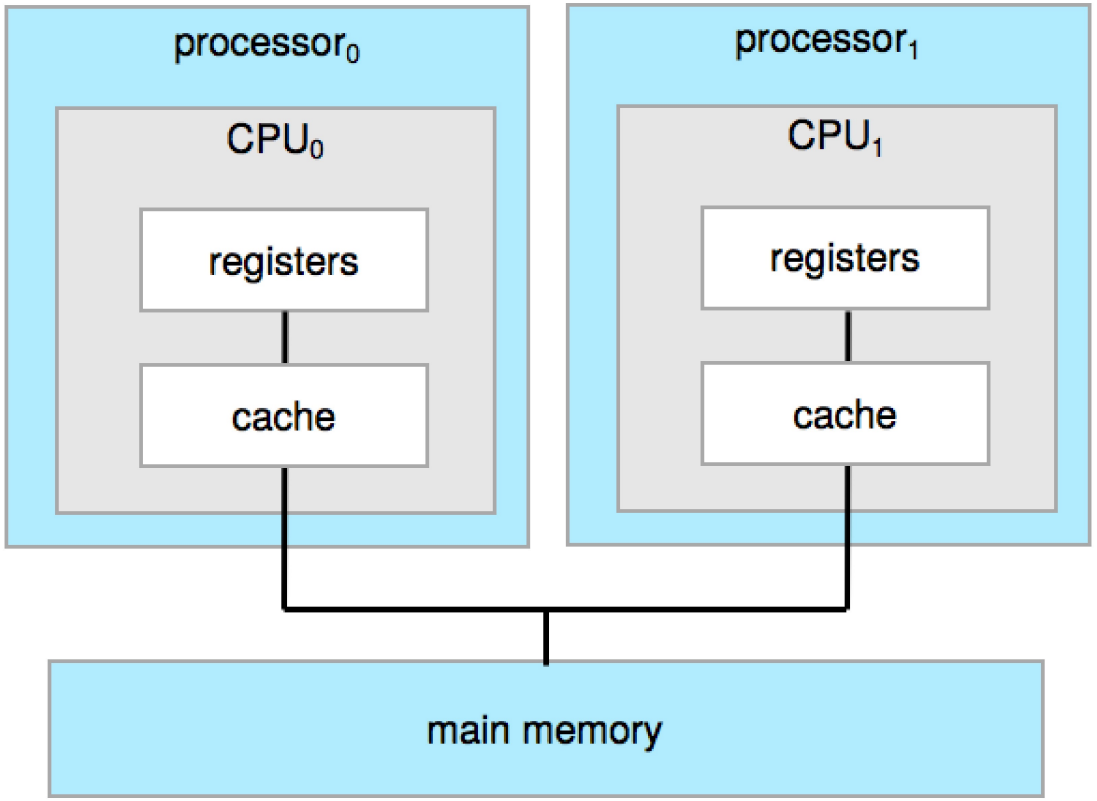
Direct Memory Access Structure

- Interrupt-driven I/O is good for small transfer. (suitable for user interactive processes as in MS word)
- DMA is 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 one interrupt is generated per block, rather than the one interrupt per byte
- CPU is available to accomplish other work, while device controller doing the I/O operations.
- DMA Applications: multimedia players, games, and backup tools, ML application, databases, etc.

Multiprocessor Systems

- two or more hardware processors (CPU cores) in close communication,
- sharing the computer bus and sometimes the clock, memory, and peripheral devices.

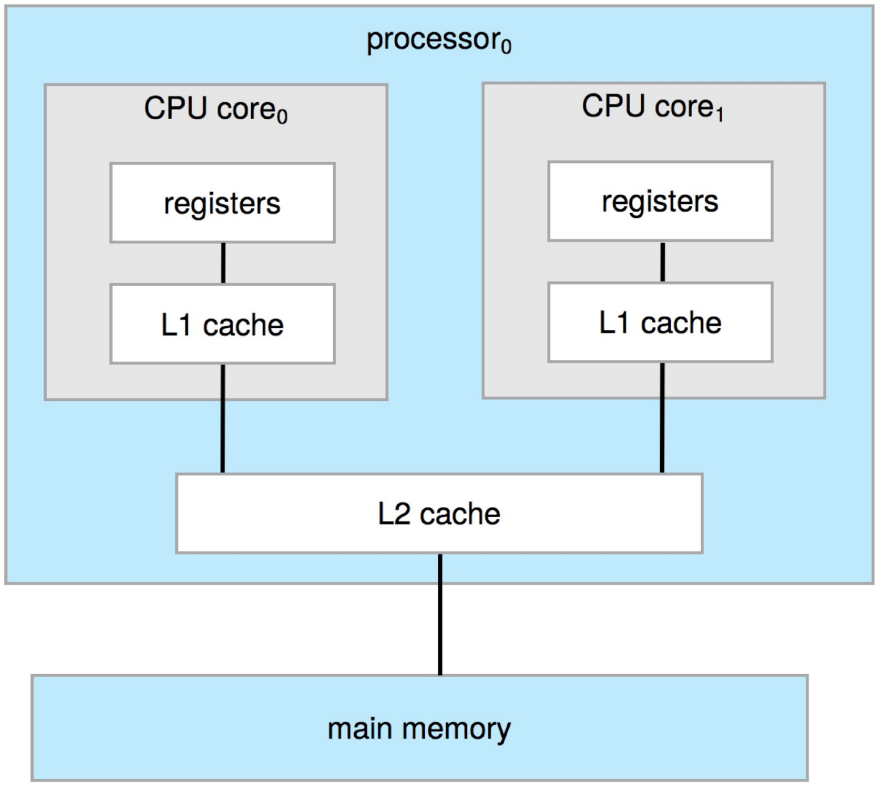
Symmetric Multiprocessing Architecture



Symmetric Multiprocessing – each processor performs all tasks (operating system functions and user processes)

Dual-Core Design

- CPU—The hardware that executes instructions.
- Processor—A physical chip that contains one or more CPUs.
- Core—The basic computation unit of the CPU.
- Multicore—Including multiple computing cores on the same CPU (processor).
- Multiprocessor—Including multiple processors.



A dual-core design with two cores on the same chip.

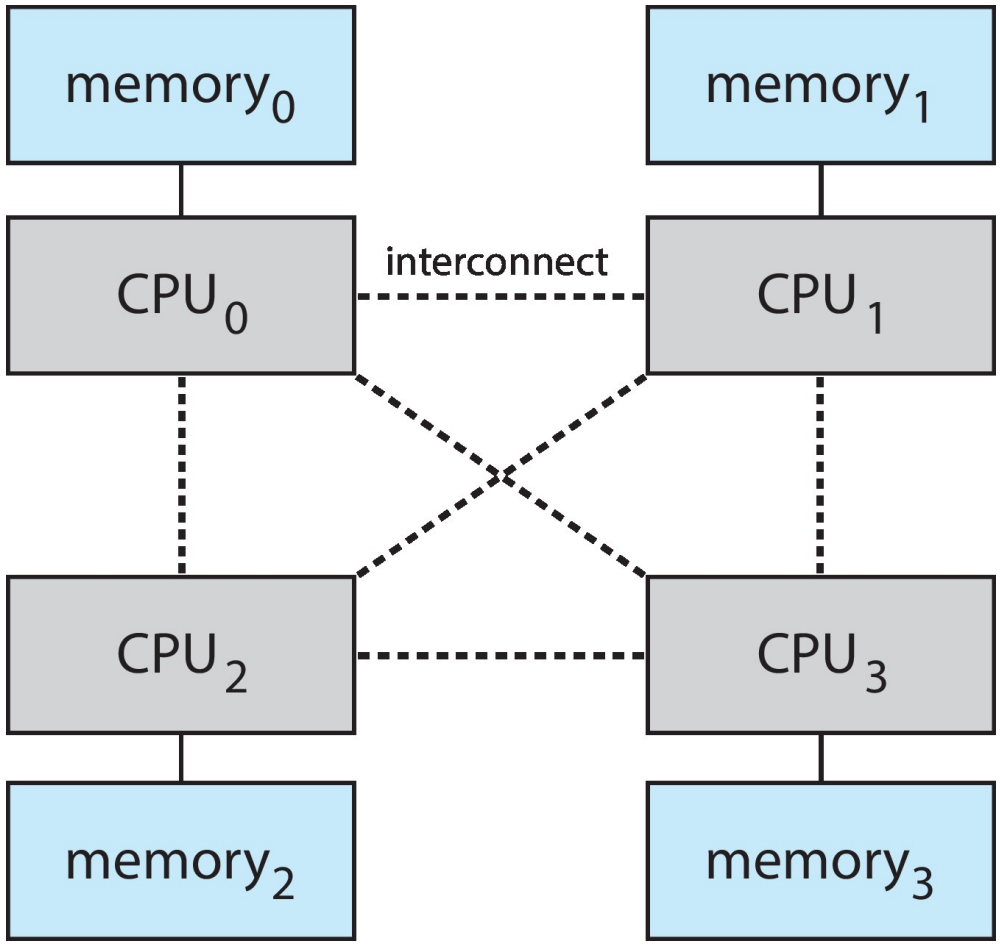
Let's think…

- Multicore systems can be more efficient than multiple chips with single cores. Why?
  - on-chip communication is faster than between-chip communication.
  - uses significantly less power than multiple single-core chips.

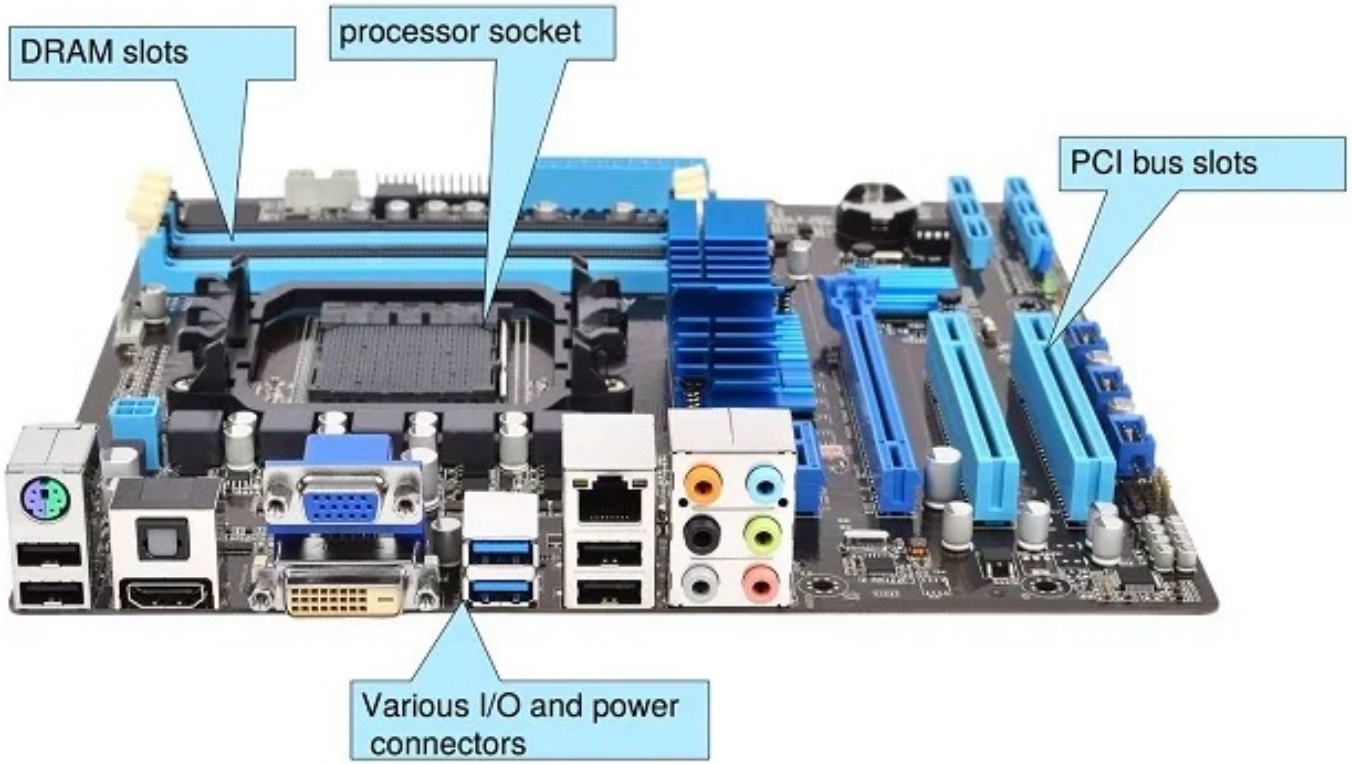
Non-Uniform Memory Access System

Once we add too many CPUs, contention for the system bus becomes a bottleneck and performance begins to degrade.

An alternative approach is instead to provide each CPU (or group of CPUs) with its own local memory that is accessed via a small, fast local bus. This approach is called NUMA.



Motherboard



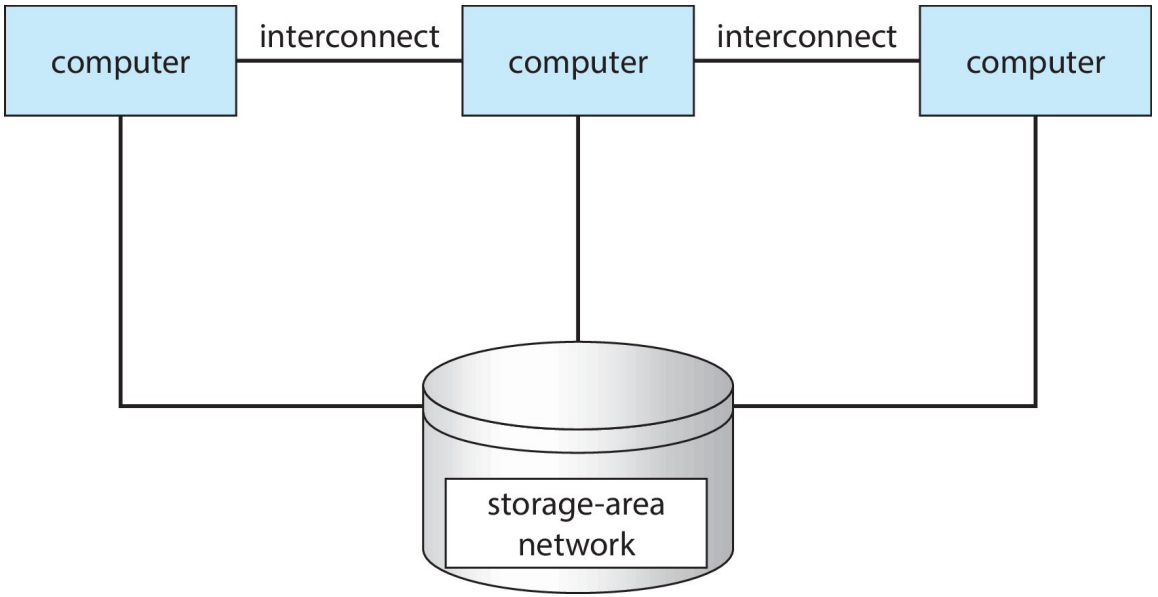
This board is a fully functioning computer, once its slots are populated. Even the lowest-cost general-purpose CPU contains multiple cores. Some motherboards contain multiple processor sockets. More advanced computers allow more than one system board, creating NUMA systems.

Let's brainstorm…

A dual-core system requires each core has its own cache memory. True or False?

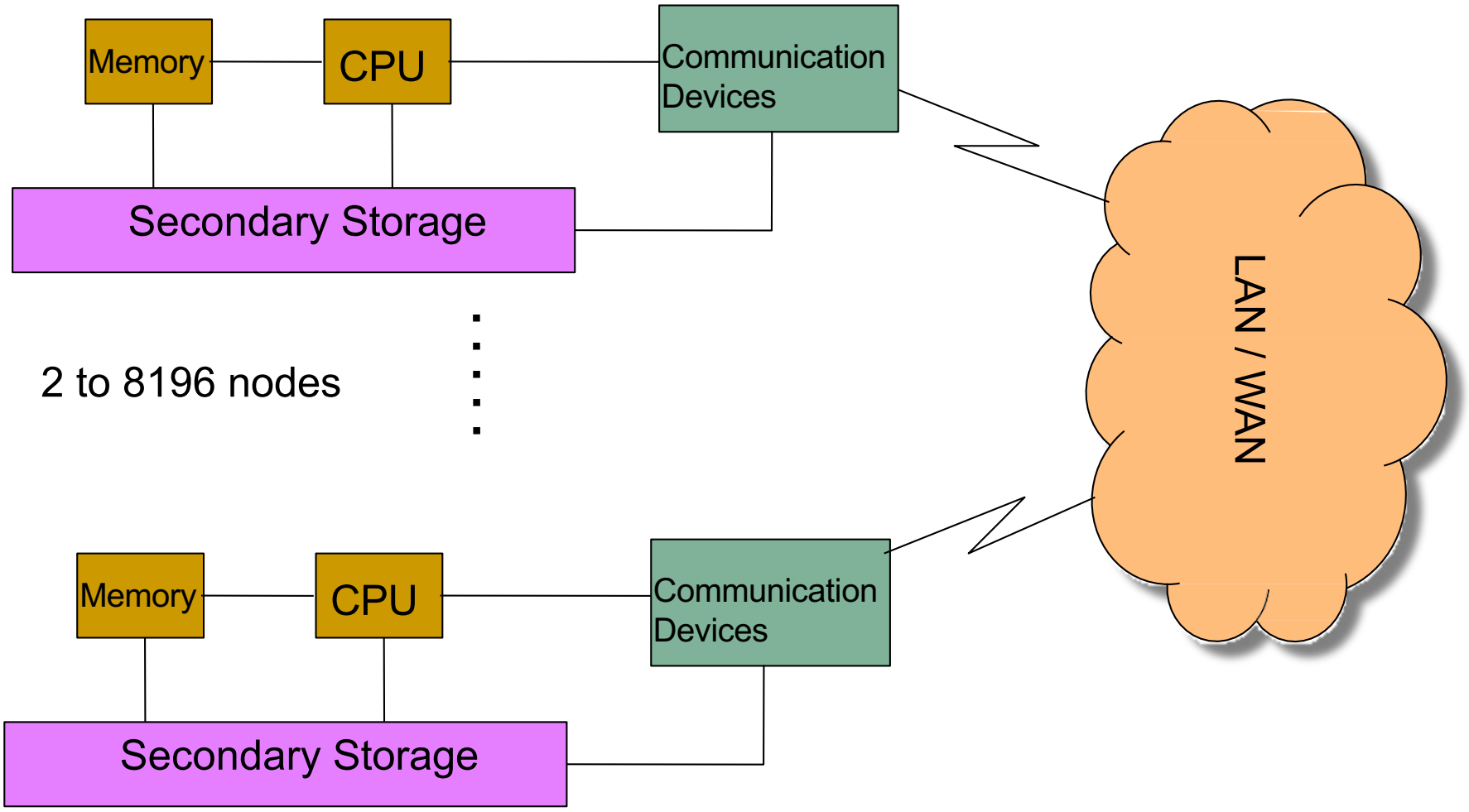
- A dual-core system does not necessarily require each core to have its own cache memory. While some dual-core systems have separate caches for each core (referred to as private caches), others may use a shared cache that both cores can access. The specific cache architecture depends on the design of the processor.

Clustered Systems



Multiple CPUs. Clustered systems differ from the multiprocessor systems in that they are composed of two or more individual systems—or nodes—joined together; each node is typically a multicore system. Computers/systems are usually connected via LAN.

Multi-Computer/Cluster



Questions

1. Which of the following would lead you to believe that a given system is an SMP-type system?

A) Each processor is assigned a specific task.
B) There is a boss–worker relationship between the processors.
C) Each processor performs all tasks within the operating system. ✓
D) None of the above

2. A clustered system _____.

A) gathers together multiple CPUs to accomplish computational work ✓
B) is an operating system that provides file sharing across a network
C) is used when rigid time requirements are present
D) can only operate one application at a time

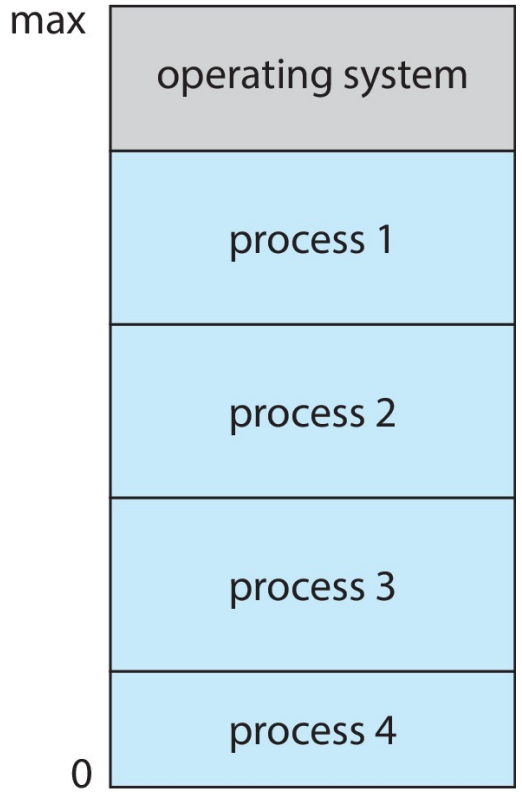
Multiprogramming (Batch system)

- Single user cannot always keep CPU and I/O devices busy
- 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 job has to wait (for I/O for example), OS switches to another job

Multiprogramming is a technique that increases CPU utilization by organizing jobs (code and data) so that the CPU always has a job to execute.

Memory Layout for Multiprogrammed System

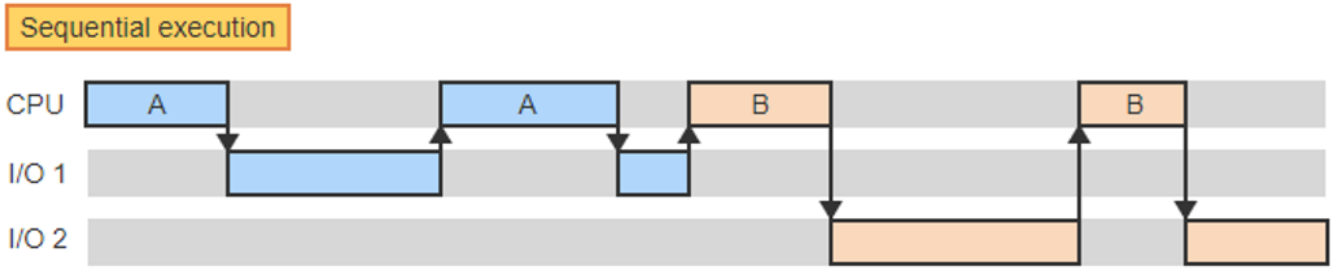
The operating system keeps several processes in memory simultaneously. The operating system picks and begins to execute one of these processes.



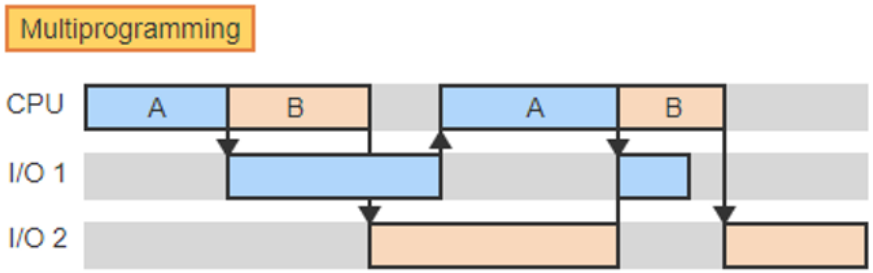
Multitasking (Timesharing)

- The concurrent performance of multiple jobs.
- A logical extension of Batch systems– the 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

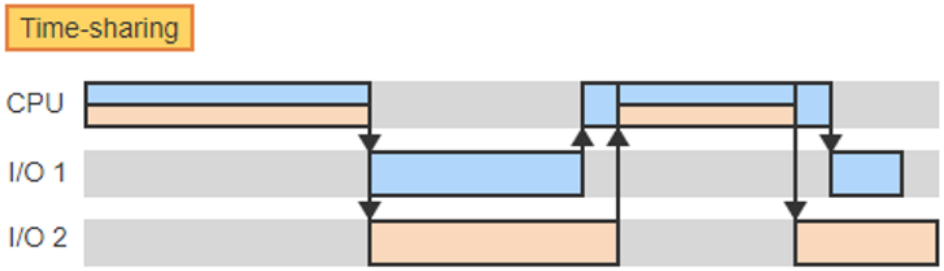
Multiprogramming vs. Multitasking



- Process A finishes, then process B starts



- Under multiprogramming, both A and B are active. When A needs I/O operation, B uses the CPU rather than making it idle.



- When one processor is waiting for I/O, the other can use CPU at 100%. When both processes are running, the CPU is time-shared at 50% each