Operating System
Lecture 3

Dr. Ghada Fathy

Table of contents

I/O Structure

- **After I/O starts, control returns to user program only upon I/O completion : synchronous I/O (or blocking I/O)**.

  - Wait instruction idles the CPU — the processor stops doing useful work until the I/O is done.
  - Wait loop (contention for memory access) — sometimes, instead of idling, the CPU repeatedly checks (“polls”) the device to see if it’s done. This wastes CPU cycles.
  - At most one I/O request is outstanding — since the CPU waits for each I/O to finish, it cannot overlap multiple I/O operations.

  Result: Slow overall performance, as CPU and I/O are not working in parallel.

I/O Structure

- **After I/O starts, control returns to user program without waiting for I/O completion: asynchronous I/O (or non-blocking I/O)**.
  - System call –request to the OS to allow user to wait for I/O completion.
  - Device-status table
    - ❑ The OS maintains a table that stores info about each I/O device:
      - Type (disk, printer, keyboard, etc.)
      - Address (location or controller port)
      - State (idle, busy, error, etc.)
    - ❑ This table helps the OS manage multiple devices and know their status.

  Result: CPU can do other work while waiting for I/O to finish → better system performance and multitasking.

- **OS indexes into I/O device table to determine device status and to modify table entry to include interrupt.**

I/O Structure

Mode	When control returns to user	CPU behavior	I/O overlap	Example use
Synchronous I/O	After I/O completes	Waits or loops	No	Simple programs, small systems
Asynchronous I/O	Immediately after I/O starts	Executes other tasks	Yes	Multitasking systems, modern OS

Storage Definitions and Notation Review

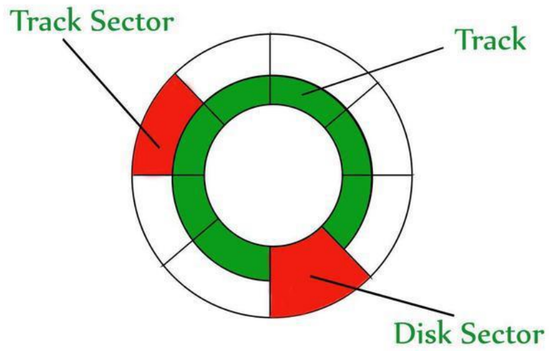
The basic unit of computer storage is the **bit**. A bit can contain one of two values, 0 and 1. All other storage in a computer is based on collections of bits. Given enough bits, it is amazing how many things a computer can represent: numbers, letters, images, movies, sounds, documents, and programs, to name a few. A **byte** is 8 bits, and on most computers it is the smallest convenient chunk of storage. For example, most computers don't have an instruction to move a bit but do have one to move a byte. A less common term is **word**, which is a given computer architecture's native unit of data. A word is made up of one or more bytes. For example, a computer that has 64-bit registers and 64-bit memory addressing typically has 64-bit (8-byte) words. A computer executes many operations in its native word size rather than a byte at a time.

Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes.
A **kilobyte**, or **KB**, is 1,024 bytes
a **megabyte**, or **MB**, is $1,024^2$ bytes
a **gigabyte**, or **GB**, is $1,024^3$ bytes
a **terabyte**, or **TB**, is $1,024^4$ bytes
a **petabyte**, or **PB**, is $1,024^5$ bytes

Computer manufacturers often round off these numbers and say that a megabyte is 1 million bytes and a gigabyte is 1 billion bytes. Networking measurements are an exception to this general rule; they are given in bits (because networks move data a bit at a time).

Storage Structure

- **Main memory** – only large storage media that the CPU can access directly
  - Random access Memory (RAM)
    - ✓ Typically, volatile
  - Read- only Memory (ROM)
    - ✓ Nonvolatile



- **Secondary storage** – an extension of main memory that provides large nonvolatile storage capacity.

  - **Hard disks** – rigid metal or glass platters covered with magnetic recording material
    - 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

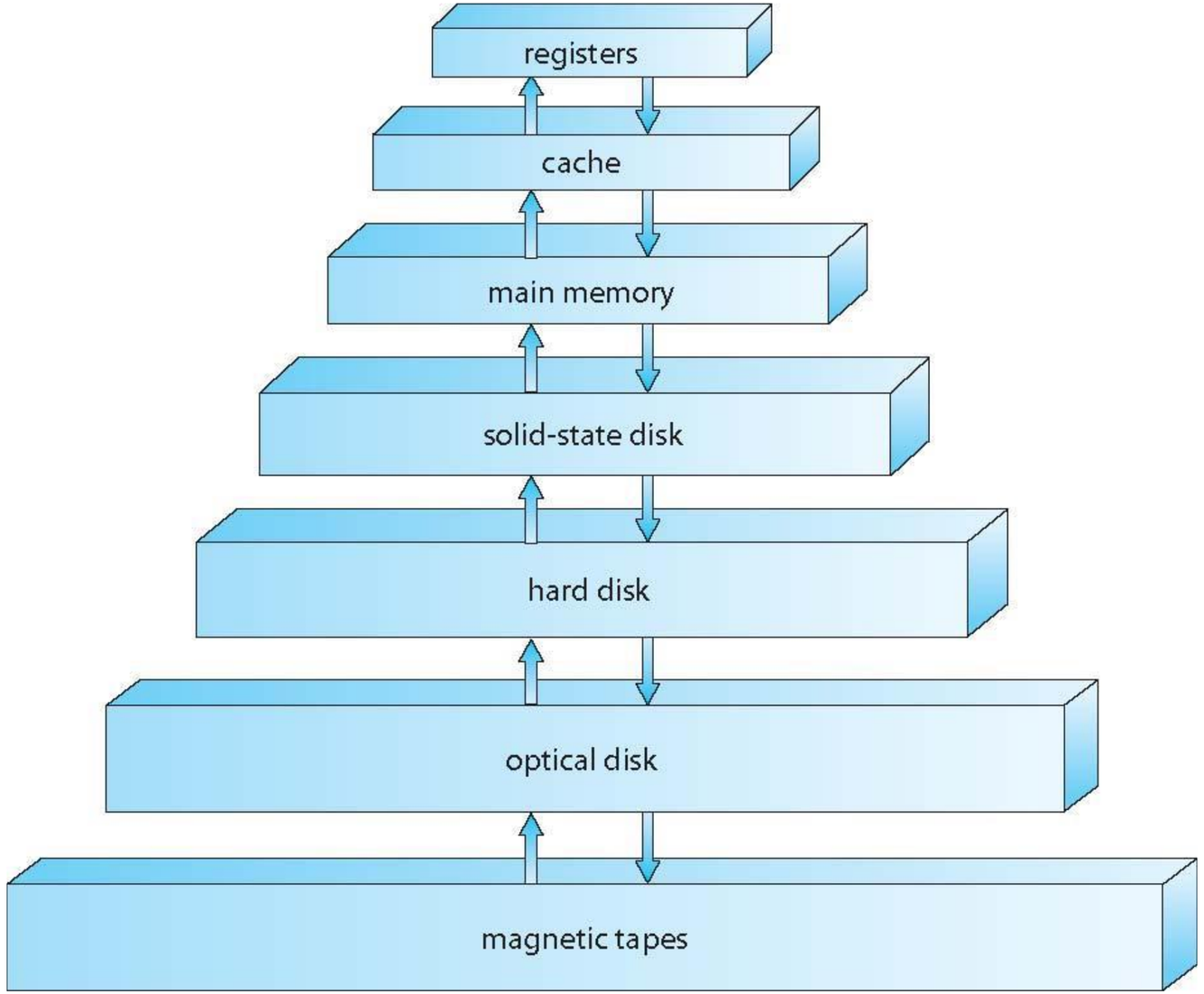
  - **Solid-state disks** – faster than hard disks, nonvolatile Various technologies Becoming more popular

Storage Hierarchy

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

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

Storage-Device Hierarchy



Caching

- **Caching** means **temporarily storing frequently used data in faster storage**, so the system can access it quickly instead of repeatedly fetching it from slower memory or storage.
- Information in use copied from slower to faster storage temporarily.
- Caching happens **at multiple levels**

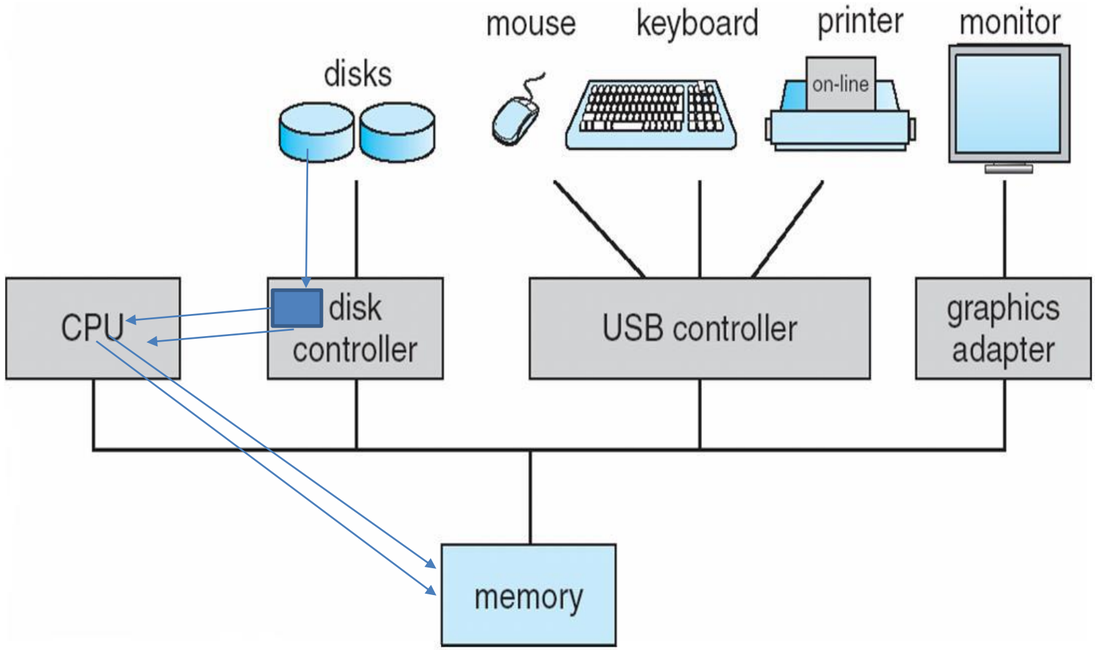
Level	**Cache Type**	**Example**
Hardware	CPU cache	L1, L2, L3 caches storing recently used instructions/data
Operating System	Disk cache / Page cache	OS keeps recently accessed files in RAM
Software / Applications	Web cache, database cache	Browser stores web pages; DB caches queries

Caching

- 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

Direct Memory Access Structure

- **Direct Memory Access (DMA)** is a technique that allows certain **high-speed I/O devices** (like disks, network cards, or sound cards) to **send or receive data directly to/from main memory without involving the CPU for every word**.

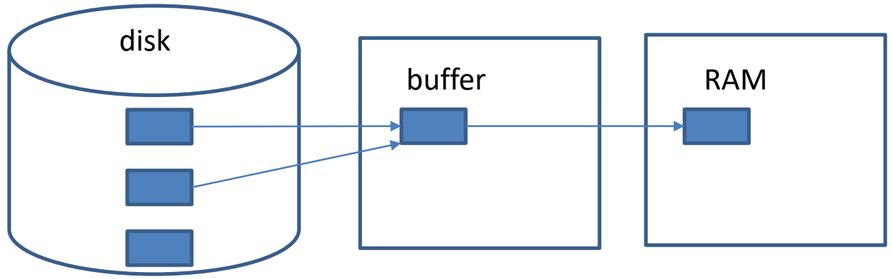


Why DMA Is Needed?

Normally, for I/O:

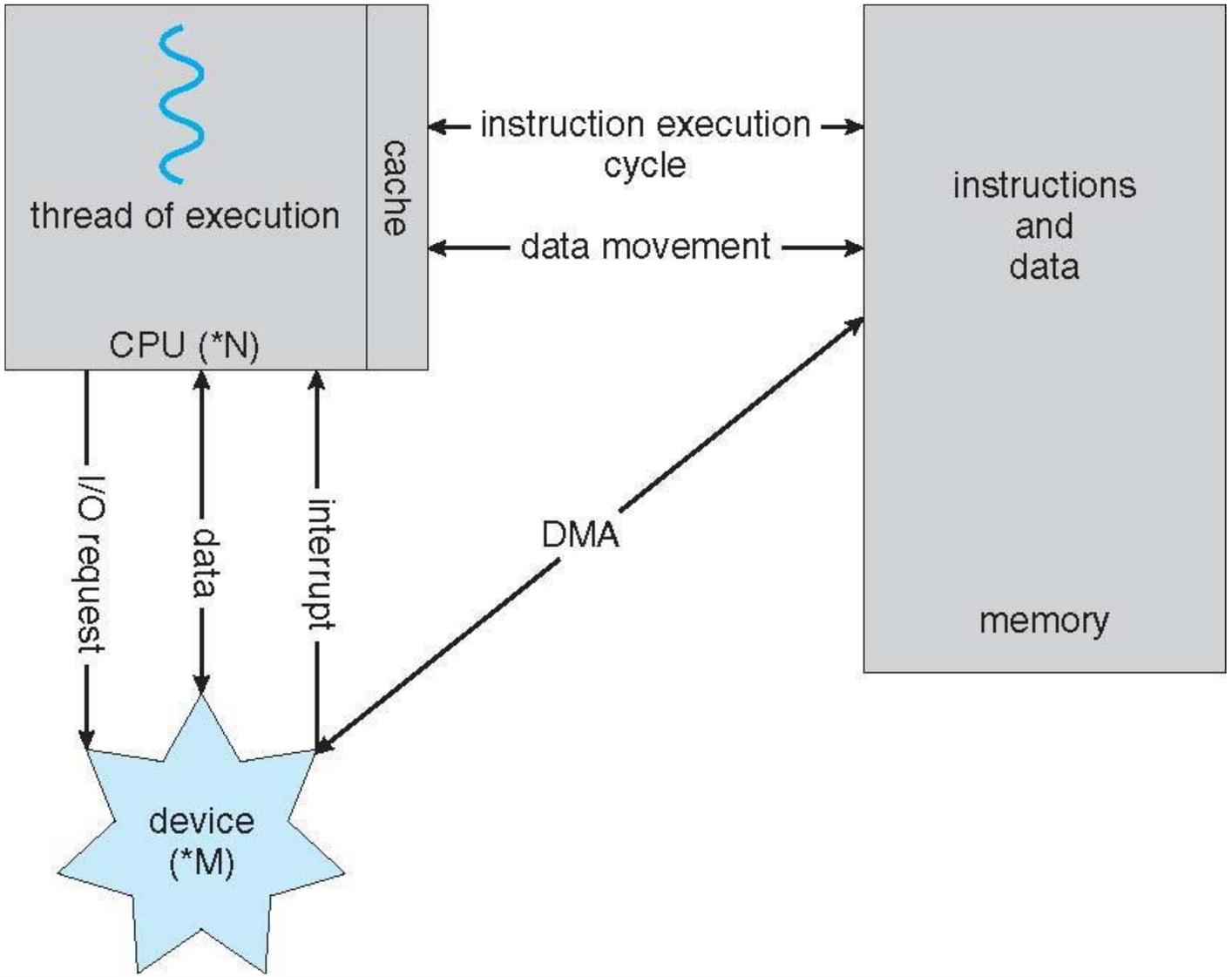
The **CPU** reads data from the I/O device (one word at a time).
Then the CPU **writes** that data into **main memory**.
This repeats for every word — meaning **CPU is busy the whole time**.

- 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.

How a Modern Computer Works




Thank
you