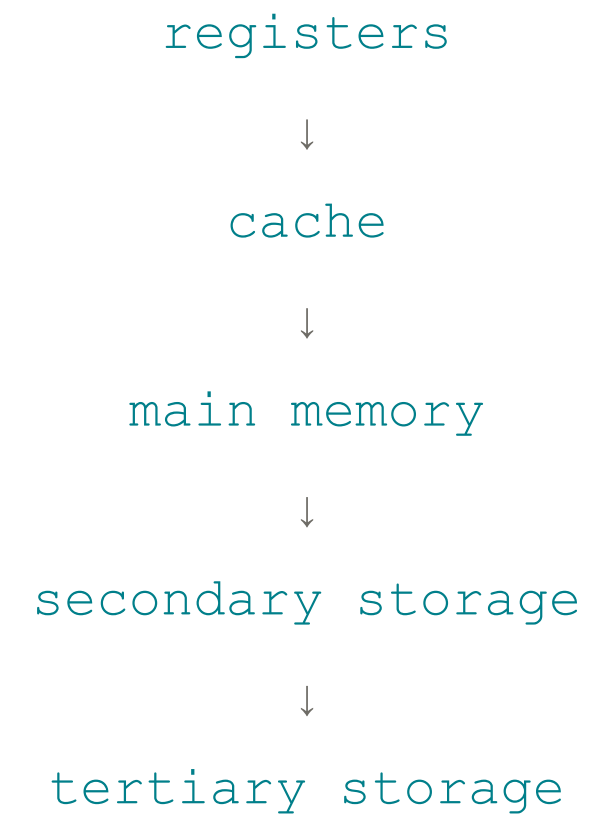


Storage Structure

Where instructions and data live?

youtube.com/ElhosseiniAcademy



Prof. Mostafa Elhosseini

Professor of Smart Systems Engineering

Where We Left Off

LECTURE 2 · §1.2.1

Interrupts

A controller signals the CPU; the vector table finds the handler; the ISR runs and control returns.



TODAY · §1.2.2–1.2.3

Storage

Agenda

- 01 **Primary memory** — RAM, and why it is volatile
- 02 **Firmware** — bootstrap, ROM, EEPROM, flash
- 03 **Bits, bytes, words** — and storage units
- 04 **Memory as addressed bytes** — load and store
- 05 **Von Neumann architecture** , registers, fetch–decode–execute
- 06 **Secondary storage** — HDD and nonvolatile memory
- 07 **The memory hierarchy**

01

Storage Structure

The CPU can load instructions only from memory — so every program must first be **loaded into memory** to run.

Main Memory (RAM)

Working space

Where programs and data are kept while the CPU is **actively using them**.

Volatile

Loses its contents the moment **power is turned off** or otherwise lost.

The bridge

Sits between the CPU and storage devices; **nothing runs without passing through it**.

- **Random-access:** the CPU can access any memory location directly, in roughly the same amount of time, regardless of where that location is .
- **Random Access** = direct access to any address.
- A good contrast is **sequential access**.

What RAM Holds — and Which Kind

STORES

- The operating system
- Running applications
program code and runtime state currently executing.
Chrome – Word
- User data in active use
open documents, browser pages, images, variables,
and temporary results

Far faster than any storage device — HDD or SSD.

TWO FAMILIES

DRAM

Dynamic RAM — dense and inexpensive; **must be refreshed** . Used for **main memory**. — Capacitor

SRAM

Static RAM — faster, costlier, **no refresh needed** . Used for **cache**. — Transistor

user data in active use: In a program — variables, arrays, objects, stack data, heap data, and intermediate results

The Bootstrap Program

The very first program executed when the computer is powered on — it **loads the operating system and starts it**.

01

Power on

02

Run **POST** — power-on self test

Before starting the OS, make sure the machine can actually run.

03

Locate the **OS loader**

Windows Boot Manager
GRUB on many Linux systems

04

Load the OS into RAM and **hand over control**

Since RAM is **volatile**, we cannot trust it to hold the bootstrap program. It is stored instead in **firmware** — ROM, EEPROM, or flash.

Firmware / UEFI → flash memory on motherboard

Operating system → SSD/HDD

After boot: Kernel + running programs + active data → RAM

Bootloader → SSD/HDD/boot device

RAM → temporary working memory

Firmware

Storage that is **infrequently written to and is nonvolatile** — a **special kind of software** embedded inside hardware.

NATURE

A set of instructions — program code — stored in chips (**ROM, EEPROM, flash**) on the device itself.

PURPOSE

Low-level control of the hardware , before any operating system or application runs.

BRIDGE

Connects **raw hardware** with **higher-level software**.

EXAMPLES

- Motherboard firmware → UEFI/BIOS
- Router and printer control programs
- Embedded controllers in cars and appliances
- SSD firmware → controls how the SSD operates internally

In a nutshell: firmware is **software written for hardware**.

EEPROM

Electrically erasable programmable read-only memory.

Can be changed — rarely

Writable, but **not designed to be written frequently** .

Low speed

Slow compared with RAM and flash.

Holds the bootstrap

Plus mostly **static programs and data** that are not frequently used.

Device identity

The **iPhone** uses EEPROM to store its **serial number and hardware information** .

In short: EEPROM is nonvolatile memory for **permanent but occasionally updatable** data.

Bit, Byte, Word

0 1

Bit (**B**inary **D**igit)

The basic unit of computer storage; holds **one of two values** . All other storage is built from collections of bits.

01000001

Byte

Eight bits — the smallest convenient chunk of storage, and one character such as “A”. Most machines **move a byte, not a bit** .

8 bytes

Word

The architecture’s **native unit of data** . A machine with 64-bit registers and 64-bit addressing has **64-bit (8-byte) words** .

A **word** means the **amount of data** that the processor naturally handles as **one unit**. It is closely related to the **processor architecture** and **register size**.

Storage Units

UNIT	EXACT	ROUNDED OFF
Kilobyte (KB)	1,024 bytes	≈ 1 thousand bytes
Megabyte (MB)	1,024 ² bytes	≈ 1 million bytes
Gigabyte (GB)	1,024 ³ bytes	≈ 1 billion bytes
Terabyte (TB)	1,024 ⁴ bytes	≈ 1 trillion bytes
Petabyte (PB)	1,024 ⁵ bytes	≈ 1 quadrillion bytes

Networking is the exception: measurements are given in **bits**, because networks move data a bit at a time.

Memory = Addressed Bytes

0x1000 01001000

0x1001 01100101

0x1002 01101100

0x1003 01101100

0x1004 01101111

Every byte has its own **unique address**.

load

Moves a byte or word **from main memory into an internal CPU register** .

store

Moves the content of a register **back into main memory** .

Aside from explicit loads and stores, the CPU **automatically loads instructions** from the location held in the **program counter**. The memory unit itself sees only a **stream of addresses** — it does not know how they were generated or what they are for.

Von Neumann Architecture

Proposed by **John von Neumann, 1945** — program and data stored in the **same memory**.

CPU

Control unit + ALU

Memory

Instructions and data

Input / Output

Keyboard, printer, disk

OPERATION

The CPU fetches instructions and data
sequentially from memory over a shared bus.

THE BOTTLENECK

A **single path** between CPU and memory limits throughput, no matter how fast the processor becomes. **Caches exist largely to soften this.**

CPU = fast factory

Memory = warehouse

Bus / memory path = narrow road

The Instruction–Execution Cycle

01 · FETCH

An instruction is **fetch**ed from **memory** and stored in the **instruction register**.

02 · DECODE

The instruction is **decoded** , and may cause **operands to be fetched** into internal registers.

03 · EXECUTE

The operation **runs on the operands** ; the result may be **stored back into memory** .

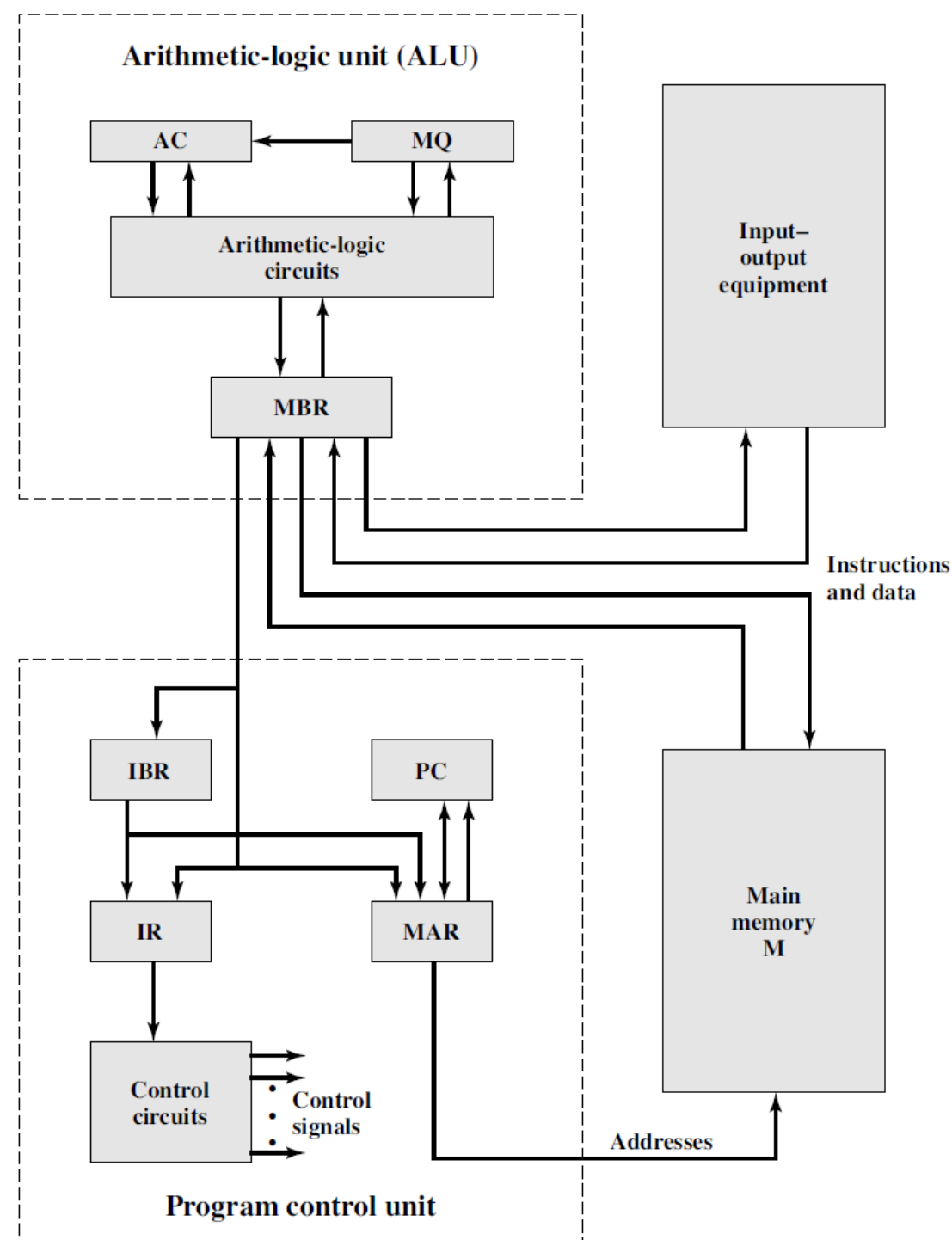
In one line: the cycle repeats forever, and the **program counter** decides where it goes next.

CPU Registers

PC	Program counter — next instruction’s address
IR	Instruction register — instruction being executed
AC	Accumulator — intermediate results
MAR	Memory address register — address to access
MBR	Memory buffer register — data read or written
GPRs	General-purpose registers — working data
FLAGS	Status register — zero, carry, negative

Registers sit at the **top of the hierarchy** — fastest and smallest.

Structure of the IAS Computer



Two units inside the CPU: the **arithmetic-logic unit** that computes, and the **program control unit** that decides what happens next.

Only **two wires** leave for memory — addresses out of **MAR** , instructions and data in and out of **MBR**.

Notice the **input-output equipment** hangs off the same memory path — the subject of **part two**.

أجهزة الإدخال والإخراج متصلة بمسار نقل البيانات نفسه المرتبط بالذاكرة، وليست معزولة في مسار مستقل تمامًا.

Keyboard / Disk → I/O equipment → MBR / Memory

Memory / CPU → MBR → I/O equipment → Printer / Display

Why We Need Secondary Storage

Ideally, programs and data would reside in main memory **permanently** . Two reasons make that impossible:

01 Main memory is usually **too small** to store all needed programs and data permanently.

02 Main memory is **volatile** — it loses its contents when power is turned off or otherwise lost.

So most systems provide **secondary storage** as an **extension of main memory** — its main requirement being to hold **large quantities of data permanently**.

Secondary Storage

Long-term storage that **retains information without power** — nonvolatile memory.

CHARACTERISTICS

- ▲ **Larger capacity** than main memory
- ▼ **Slower access** than RAM
- ▼ **Cheaper cost** per bit

EXAMPLES

- Hard-disk drives (HDD)
- Solid-state drives (SSD)
- Optical discs — CD, DVD, Blu-ray
- Magnetic tape

Most programs live here until they are loaded into memory; many then use it as **both source and destination** of their processing. The most common NVM device is **flash memory**, dominant in phones and tablets.

Beyond the Three Levels

The structure we have described — **registers, main memory, secondary storage** — is only **one of many possible storage system designs** .

Other components

Cache memory, CD-ROM and Blu-ray, **magnetic tape**, and more.

Tertiary storage

Those **slow and large** enough to be used only for special purposes — such as **backup copies** of material stored elsewhere.

Every storage system does the **same job** : store a datum (بيانات) and hold it until it is retrieved. They differ only in **speed, size, and volatility**.

The Memory Hierarchy — the Principle

Storage systems are organised by **capacity and access time** — the time needed to read or write data.

As a general rule there is a **trade-off** (مفاضلة) **between size and speed** , with **smaller and faster memory closer to the CPU**.

A complete design must **balance both** : use only as much expensive memory as necessary, and as much inexpensive nonvolatile storage as possible. **Caches are inserted wherever two components differ sharply in access time.**

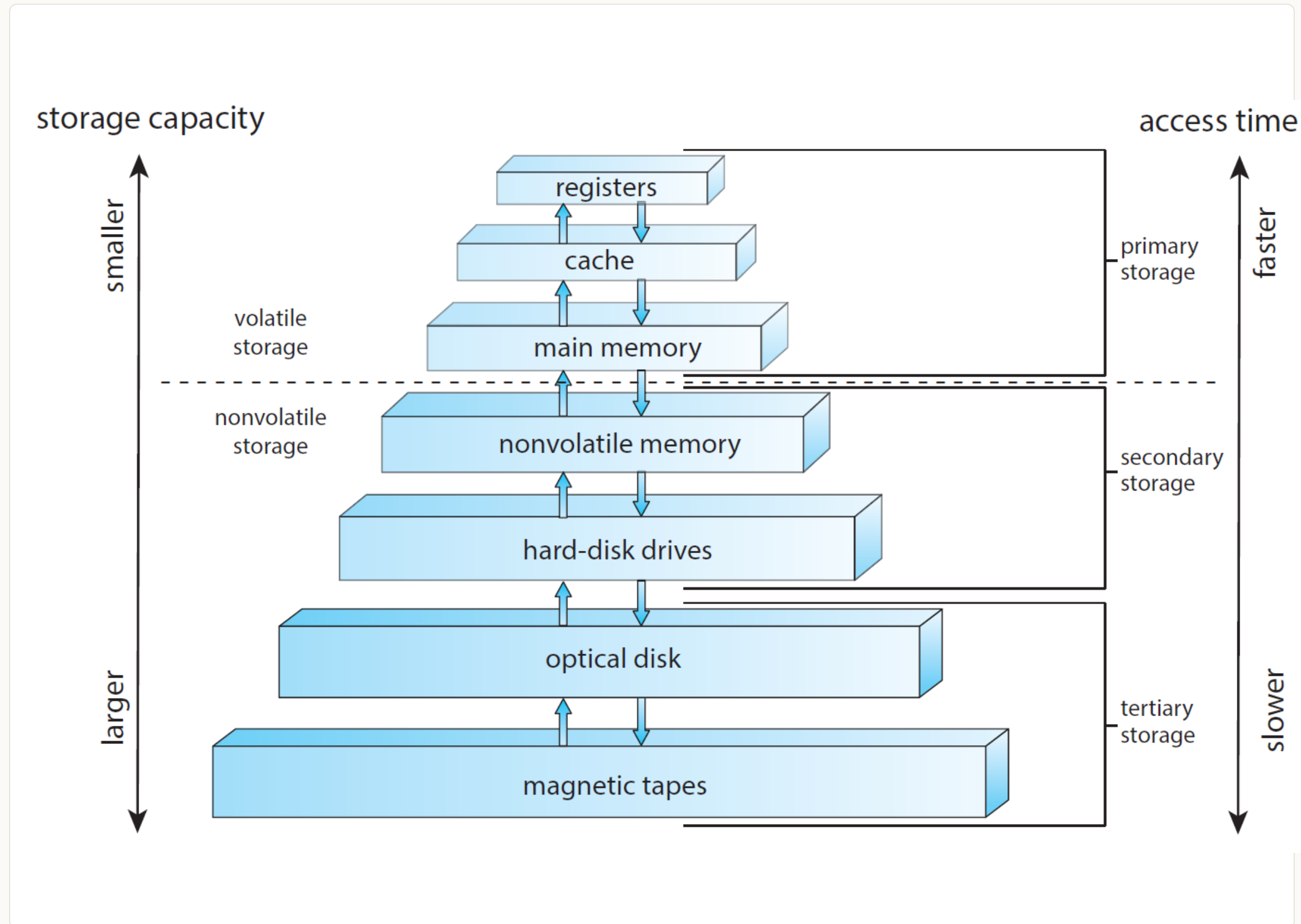
Faster

More expensive per byte, and **smaller**.

Slower

Cheaper per byte, and **larger**.

Figure 1.6 — Storage-Device Hierarchy



The **dashed line** divides **volatile storage** above from **nonvolatile storage** below .

The top four levels are built from **semiconductor memory** . NVM devices at the fourth level are generally **faster than hard disks** ; the most common form is **flash memory**.

Data must reach nonvolatile storage for safekeeping — volatile storage loses everything when power is removed.

Levels of Memory

- | | | |
|----|---|-------------------------|
| 01 | Registers | fastest, smallest |
| 02 | Cache — L1, L2, L3 | very fast, limited |
| 03 | Main memory (RAM) | moderate speed and size |
| 04 | Secondary storage — HDD, SSD | large and slower |
| 05 | Tertiary storage — tapes, backup, cloud | very large, very slow |

Terminology for the Rest of the Course

Memory

Volatile storage. Where a particular device matters — a register, say — we will **name it explicitly** .

NVS

Nonvolatile storage. Retains its contents when power is lost. Two distinct types:

Mechanical

HDDs, optical disks, holographic storage, magnetic tape. **Larger and cheaper per byte.**

Electrical — NVM

Flash memory, FRAM, NRAM, SSD. **Costlier, smaller, faster.**