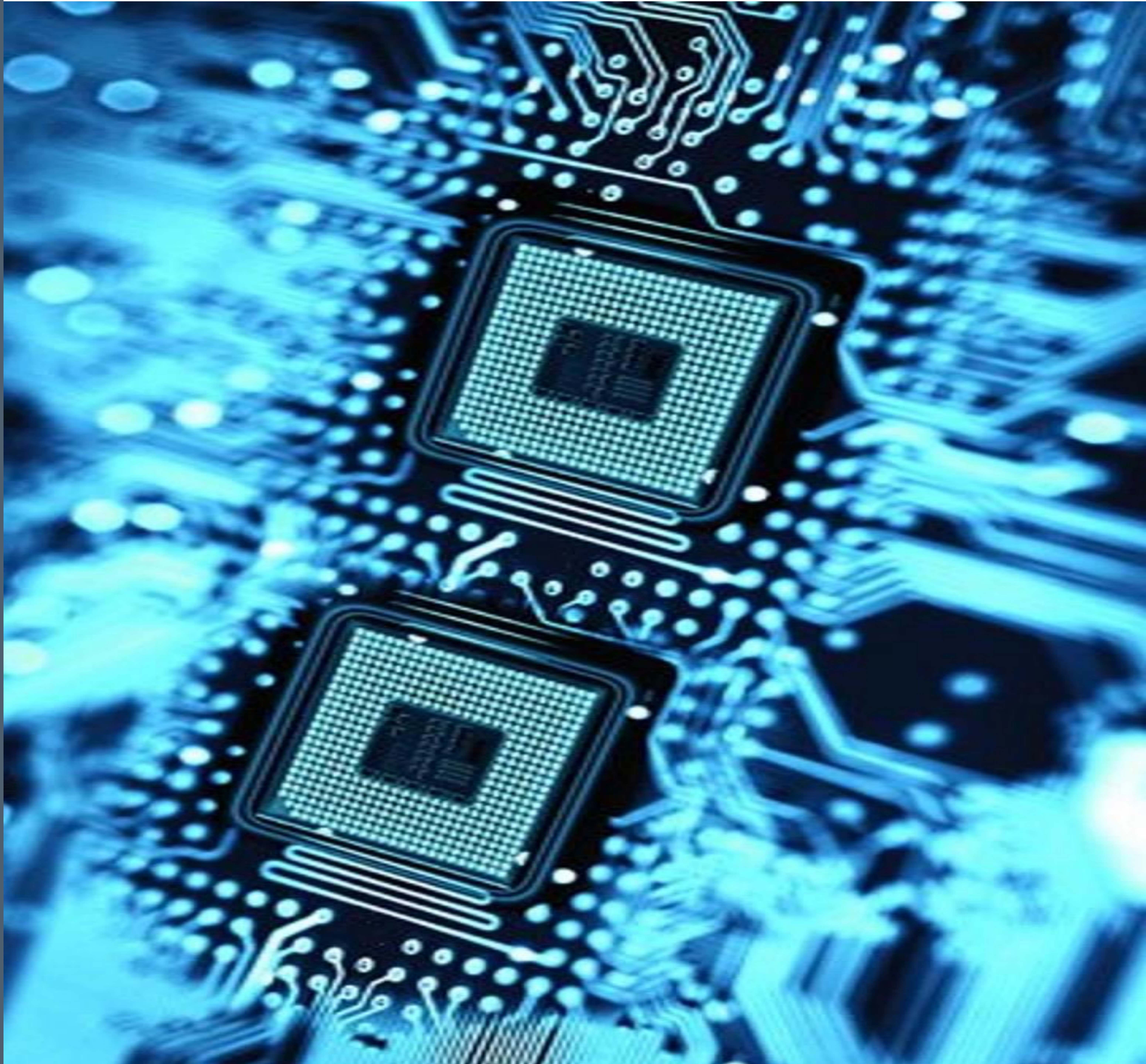
Computer organization & architecture

Course by: Dr. Ahmed Sadek

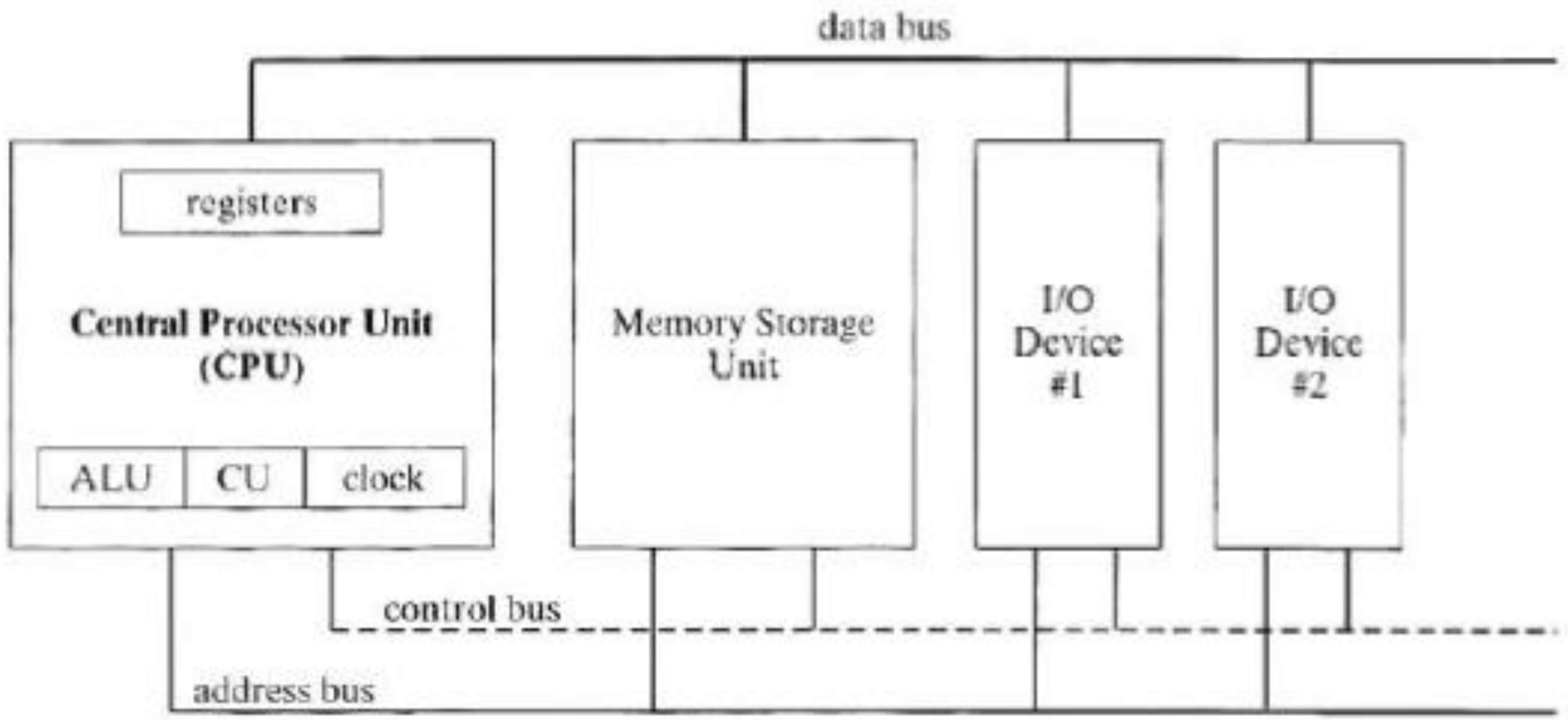
Lab By: Mahmoud Badry

CPU and Memory

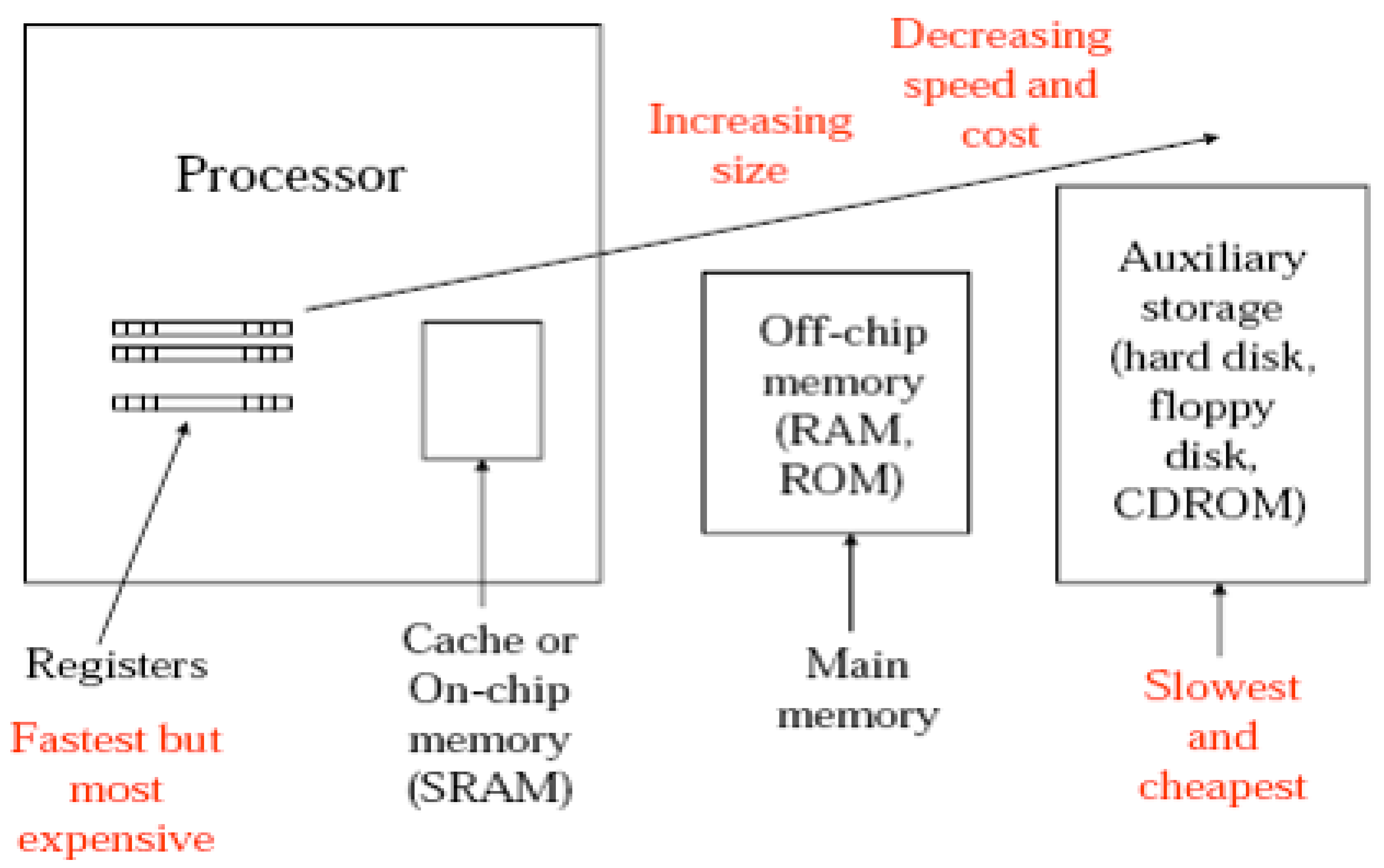
Chapter 2



Microcomputer block diagram



Memory hierarchy

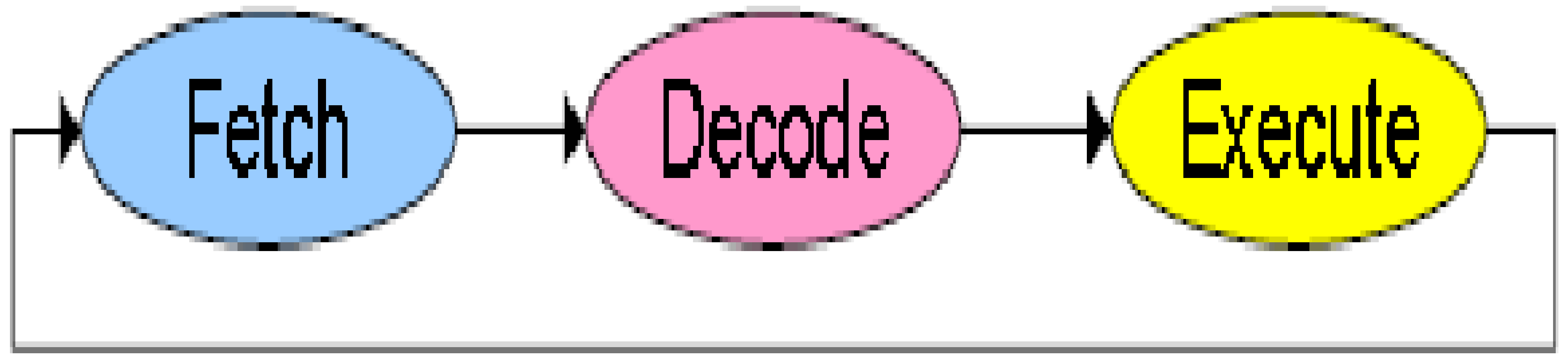

Processor
Increasing size
Decreasing speed and cost
Off-chip memory (RAM, ROM)
Auxiliary storage (hard disk, floppy disk, CDROM)
Registers
Fastest but most expensive
Cache or On-chip memory (SRAM)
Main memory
Slowest and cheapest

Instruction execution cycle

- To **perform** a given **task**, an appropriate **program** consisting of a list of **instructions** is stored in the **memory**.
- Individual **instructions** are **brought** from the **memory** into the **processor**, which executes the specified operations.
- A typical instruction may be like:

  ```
  Add LOCA,RO
  ```
- The **execution** of a single machine **instruction** can be **divided** into a **sequence** of individual **operations** called the instruction execution cycle.

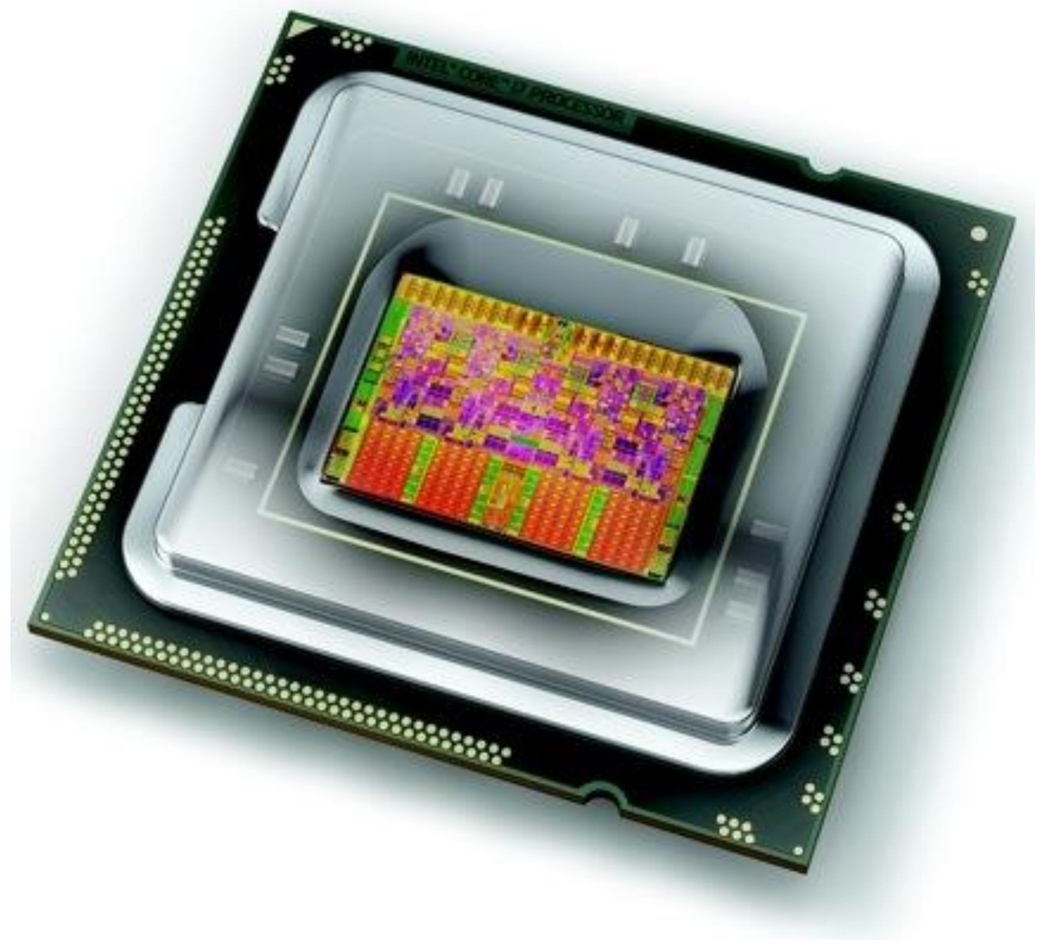
- Here's the instruction sequence:
  - **Fetch**: The **control unit** fetches the instruction , **copying** it from **memory** into the **CPU** and **increments** the program counter (**PC**).
  - **Decode**: The **control unit** determines the **type** of **instruction** to be executed . It **passes** zero or more **operands** to the arithmetic logic unit (**ALU**) and sends signals to the ALU that indicate the type of operation to be performed.
  - **Execute**: The **arithmetic logic unit executes** the instruction, **sends** its data to the **output** operand, and **updates** status **flags** providing information about the output.



Instruction execution cycle

- If processor **fetches operands** from **memory** two additional steps are required:
    - **Fetch operands**: If a memory operand is used , the control unit initiates a read operation to **retrieve** the input operand from memory.
    - **Store output operand**: If the output operand is in memory, the control unit initiates a **write** operation to store the data.

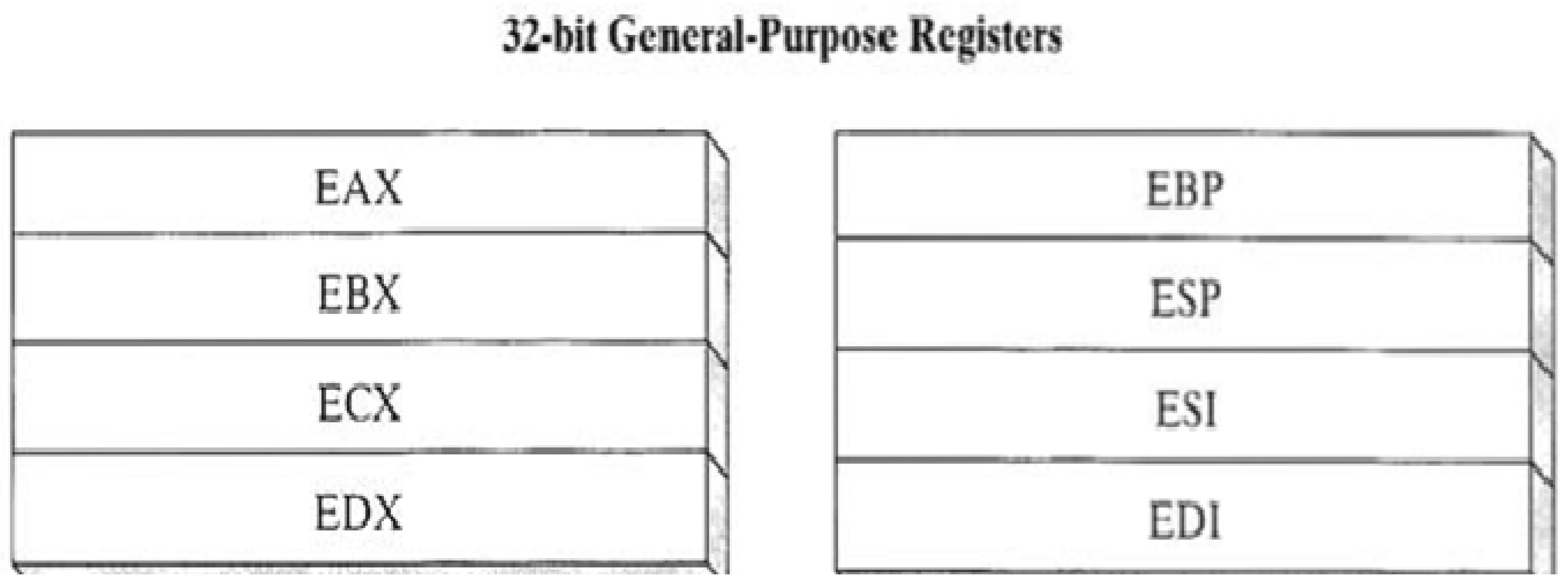
Basic Program Execution Registers



- Registers are high-speed storage locations directly inside the CPU, designed to be accessed at much **higher speed** than conventional memory.
- There are **eight general-purpose registers**, **six segment registers**, a register that holds processor status flags (**EFLAGS**). and an instruction pointer (**EIP**).

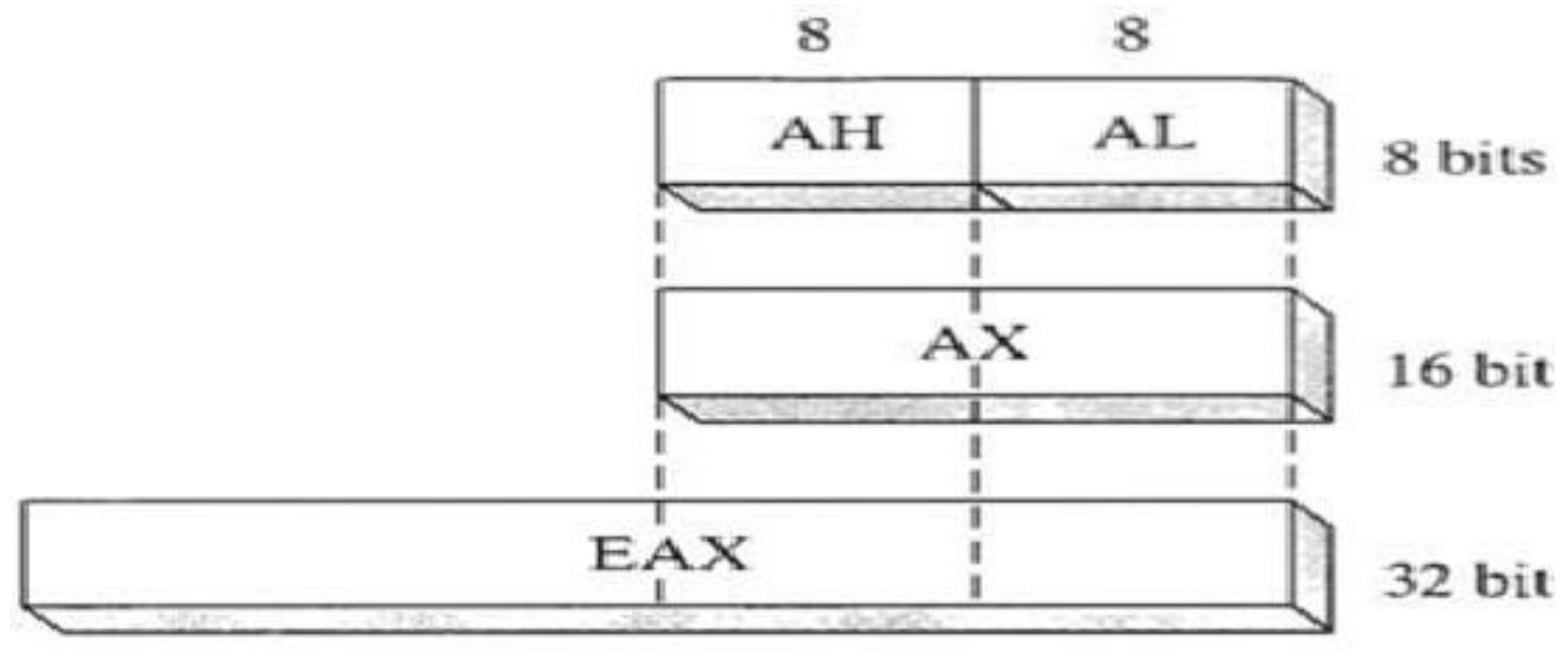
General-purpose registers

- **General-purpose** registers are primarily used for **arithmetic** and **data movement**.



General-purpose registers

- **Remember** we are using **16 bits** registers in **EMU8068.** That means we have only **AX**

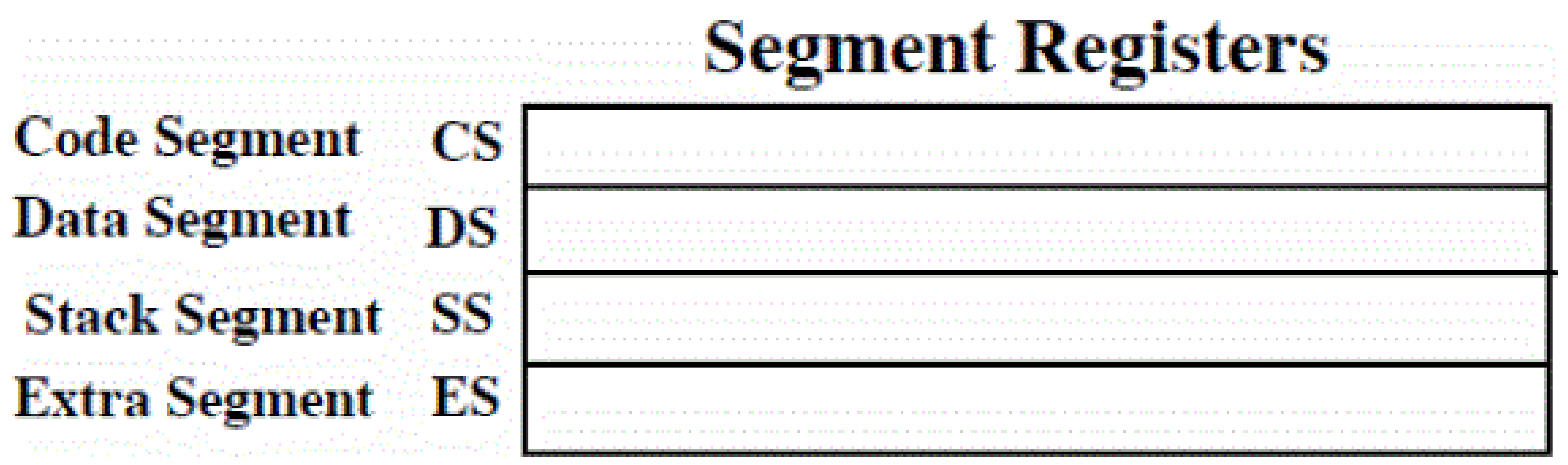


General-purpose registers

- Specialized Uses Some **general-purpose** registers have **specialized** uses:
  - **AX(Accumulator**) is **automatically** used by **multiplication** and **division** instructions.
  - The CPU automatically uses **CX(Count**) as a **loop** counter.
  - **SP**(Stack Pointer) **addresses data** on the **stack** (a system memory structure). It should never be used for ordinary arithmetic or data transfer.
  - **SI** and **DI**(Source Index And Destination Index) are used by high-speed **memory transfer** instructions.

Segment Registers & IP

- The segment registers are used as **base locations** for pre-assigned **memory areas** called segments:
  - **CS** (code segment) - points at the segment containing the **current program**.
  - **DS** (Data Segment) - generally points at segment where **variables** are defined.
  - **ES** (extra segment register) - it's up to a **coder** to define its usage.
  - **SS** (stack Segment) - points at the segment containing the **stack**.
- The **EIP** or instruction pointer register contains the **address** of the next **instruction** to be executed



Status Flags (EFLAGS Register)

- The Status flags reflect the **outcomes** of **arithmetic** and **logical operations** performed by the CPU:
  - The Carry flag (**CF**) is set when the **result** of an **unsigned arithmetic** operation is **too large** to lit into the **destination** .
  - The Overflow flag (**OF**) is set when the result of a **signed arithmetic** operation is either **too large** or **too small** to lit into the **destination**.
  - The Sign flag (**SF**) is set when the result of an **arithmetic** or **logical operation** generates a **negative** result.
  - The Zero flag (**ZF**) is set when the result of an **arithmetic** or **logical** operation generates a **result** of **zero**.
  - The Auxiliary Carry flag (**AC**) is set when an **arithmetic** operation causes a **carry** from for bit with index 3 to bit 4 in an 8-bit operand .
  - The Parity flag (**PC**) **sums** the number of bits that are set in a **number**, and indicates whether the **sum** is **odd** or **even**.

EMU8068

- Now lets play a little with **EMU8068** and try some **examples**.

THANKS