



COMPUTER SKILLS

LESSON 2

Valeria Cardellini
cardellini@ing.uniroma2.it
A.Y. 2015/16

A computer is ...

- “A device that computes, especially a **programmable electronic machine** that performs high-speed mathematical or logical operations or that assembles, stores, correlates, or otherwise processes information”
-- *The American Heritage Dictionary of the English Language*, 4th Edition, 2000

Computer Science/Informatics is about ...

- Procedures to solve problems
 - *The methodological facet*
 - Problem solving and information management
- “Machines” to process procedures
 - *The technological facet*
 - (presently) electronic computing devices and systems that use them
- Both have a long history
 - methodologies (Euclid ~300 B.C., ..., al-Kuwarizmi ~1000 A.D., ... Hilbert ~1800 A.D., Gödel, Turing, ~1900 A.D., ...)
 - machines (abacus ?B.C., ..., Babbage engine ~1800 A.D., ... ENIAC ~1940 A.D., ...)

Actual “birthdate” of computer science

- When technology advances made it realistic building machines able to process recipes million (and more ...) times faster than humans?
 - Mid of past century ...
- **The machine (computer) does the recipe**
 - How hard, tedious and complex is to:
 - Do number crunching (the large-scale processing of numerical data)?
 - Crank through a million genomes?
 - Find one person in a 30,000 campus?
 - Process a million dots on the screen or a billion sound samples?
 - Search a keyword on Google?
 - No problem!
- A word of caution: we have not (yet?) recipes and machines to solve *any* problem
 - Intractable problems
 - Non computable problems

Computer science is concerned with the study of algorithms

- Computer scientists study...
 - How to write and code algorithms
 - Algorithm engineering, software engineering
 - The “units” used in the algorithms
 - Data structures, databases
 - How well the algorithms work (including human-computer interfaces)

Specialized recipes

- Computer scientists can also specialize on special kinds of recipes
 - Recipes that create pictures, sounds, movies, animations (graphics, computer music)
 - Like other people specialize in crepes or barbeque ...
 - Still others look at *emergent properties* of computer “recipes”
 - What happens when lots of recipes talk to one another (networking, non-linear systems) ...
- Our focus will be on recipes to solve scientific/mathematical problems
 - Despite specialization, they share several core concepts with other computer science fields
 - By playing a particular game, we will learn general rules

Core concept

Information representation

- Algorithms work on abstractions (representations)
 - ... and we know what they mean

- “Machines” execute recipes to manipulate representations
 - Without knowing what they mean

What computers understand

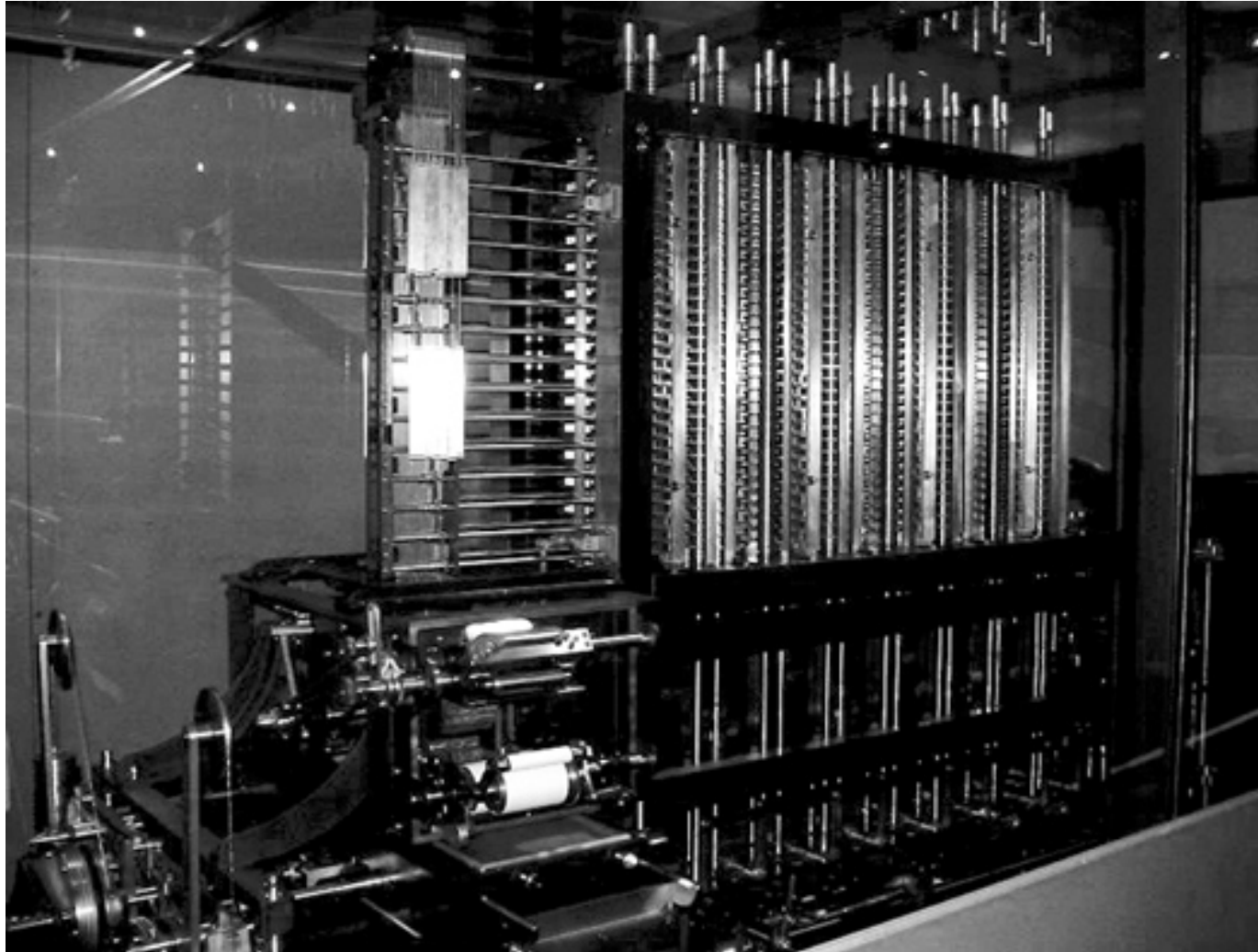
- Quotation from previous slide:
 - “machines” execute recipes to manipulate representations
 - without knowing what do they mean

- It's not really *multimedia* at all
 - It's *unimedia*
 - (said Nicholas Negroponte, founder of MIT Media Lab)
 - *Everything* is 0's and 1's

- Computers are *exceedingly* stupid
 - The only *data* they understand is 0's and 1's (**bit** = binary digit)
 - They can only do the most simple things with those 0's and 1's
 - Move this value here
 - Add, multiply, subtract, divide these values
 - Compare these values, and if one is less than the other, follow this step rather than that one
 - Done fast enough, those simple things can be amazing

ARCHITECTURE OF A COMPUTER SYSTEM

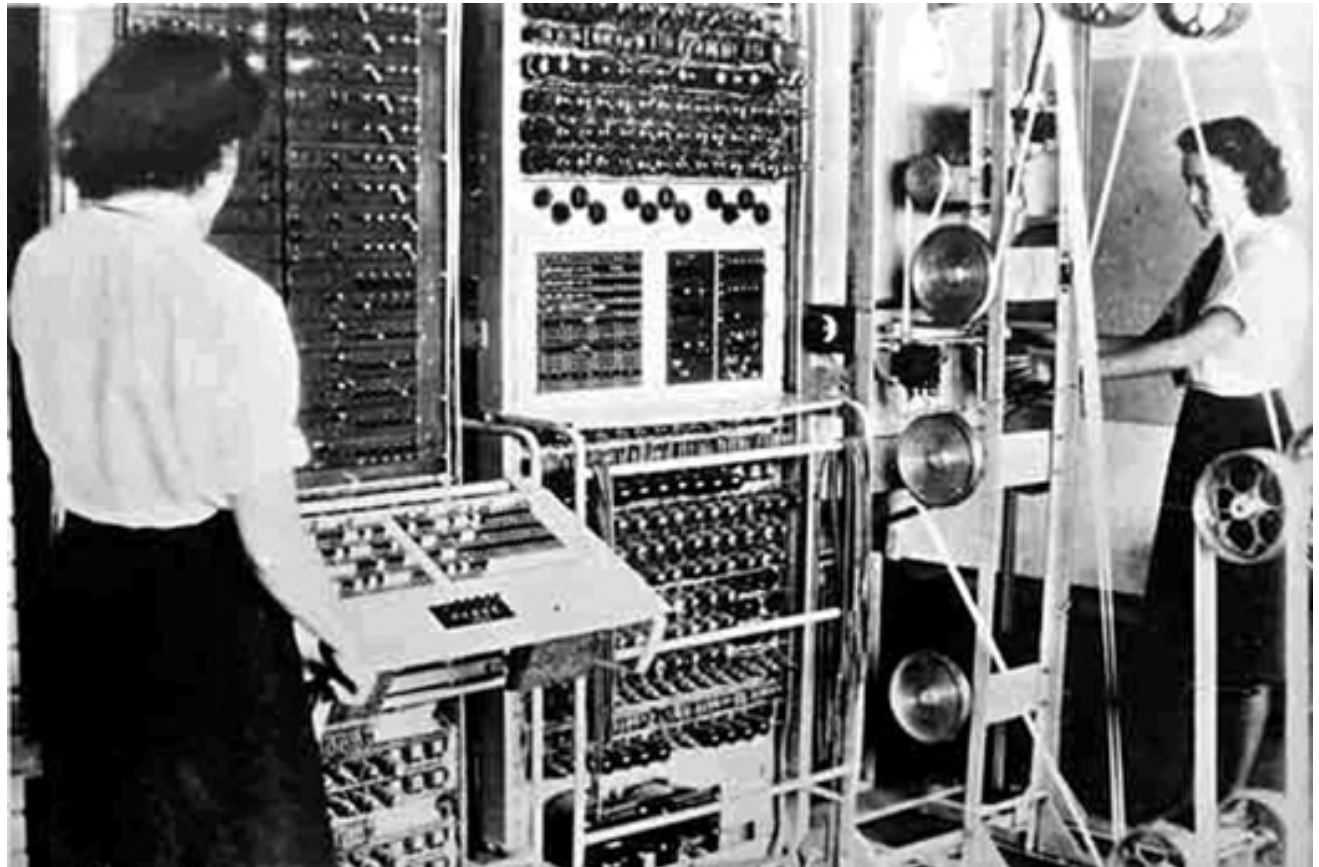
Let's make a step back in 19th century



- The Babbage's difference engine
- Conceived in 1854
- Realized in 1991 (Science Museum in London)

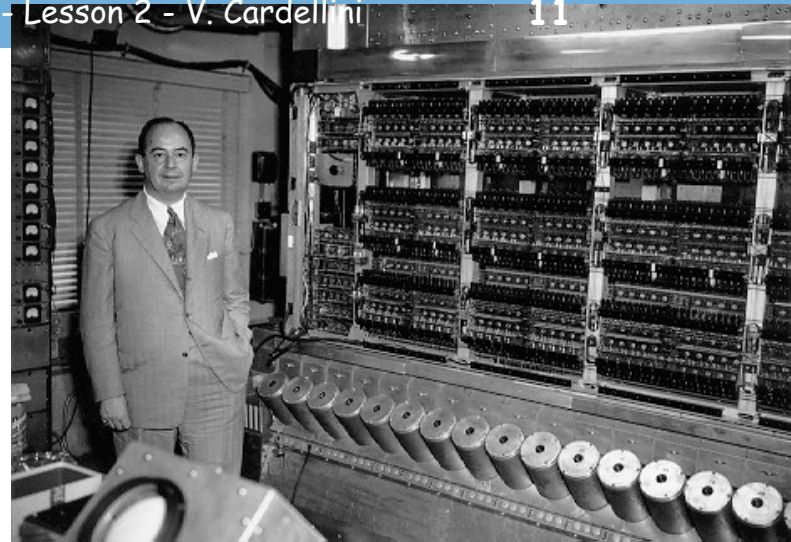
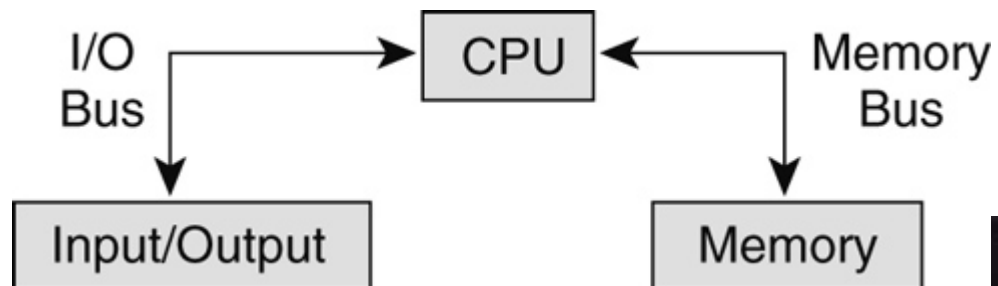
Colossus: built during the II World War

- The world's first operational, programmable electronic compute
- By Max Newman to crack the Enigma code



The Von Neumann architecture

- Conceived around 1940



Suggested reading

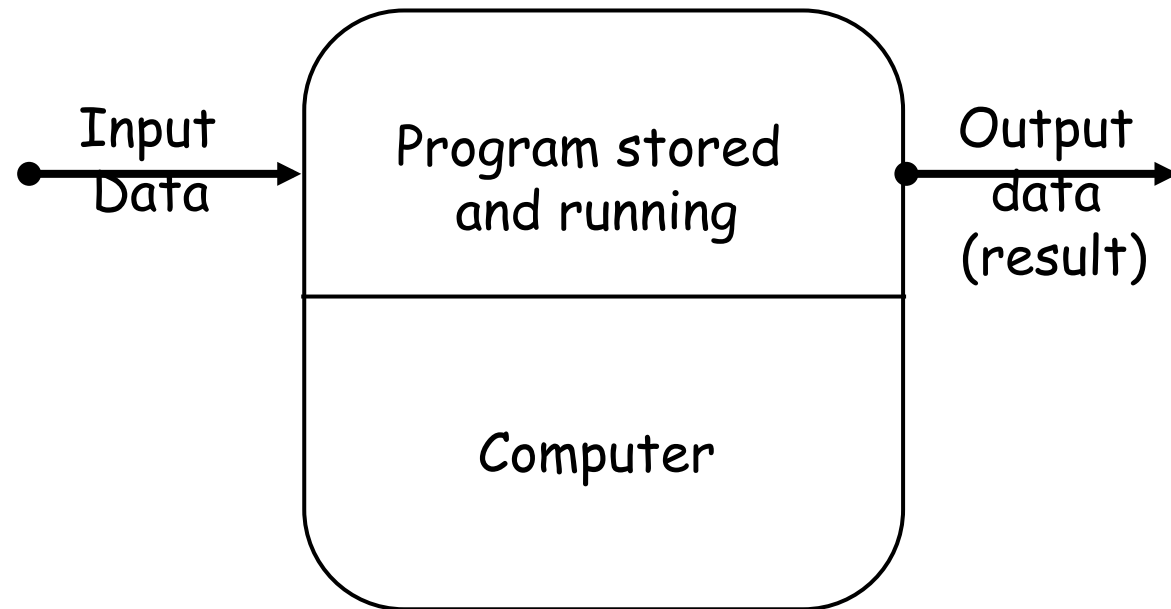
http://en.wikipedia.org/wiki/John_von_Neumann

Computer system

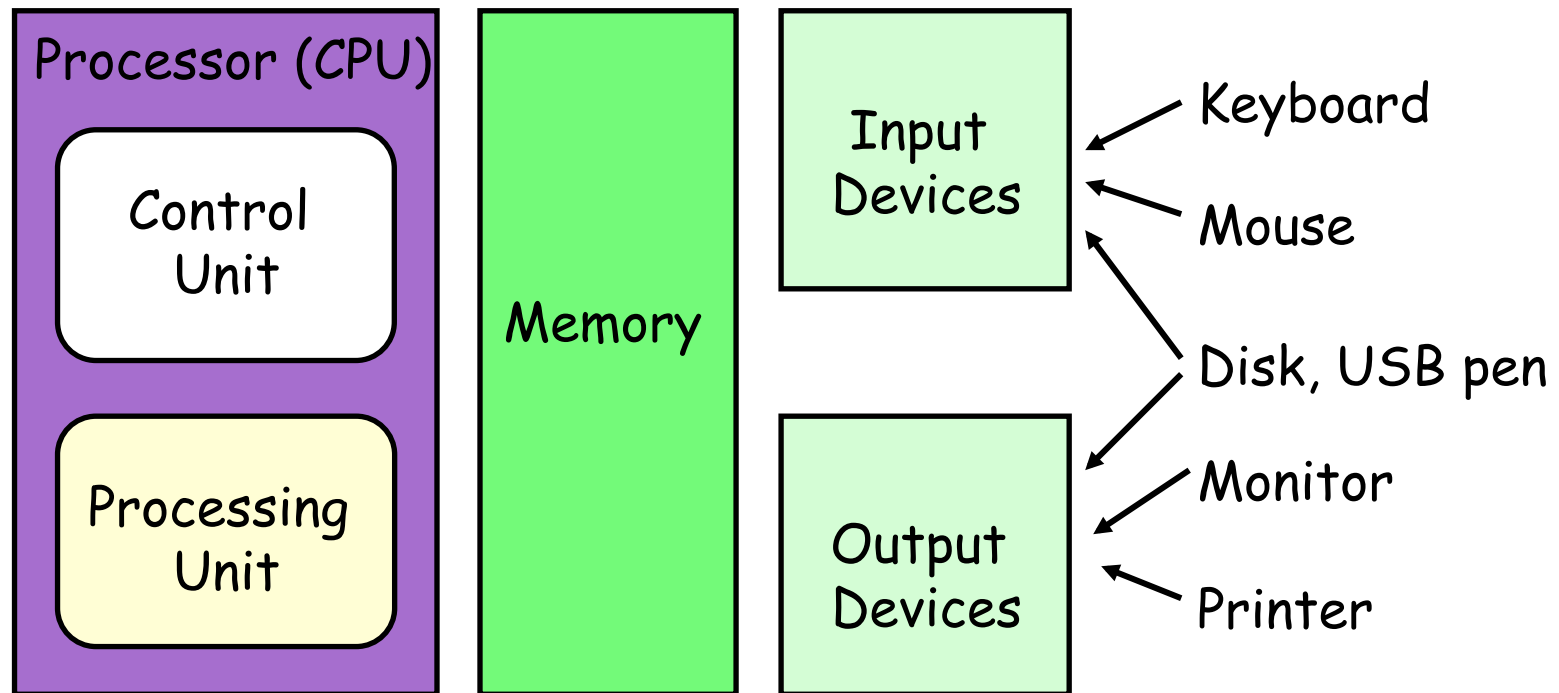
- Machine for the automatic execution of algorithms
 - General purpose
- A **program** is an algorithm coded in some programming language
- **Binary machine language**: a programming language that can be executed directly by the processor (CPU)
- Machine instructions
 - Logic and arithmetic operations on data
 - Data transfer operations (from/to memory)
 - Binary code
 - e.g., 10000110010100000 tells the computer system to perform a sum operation
- **Executable program**: an algorithm coded in machine language

Computer system

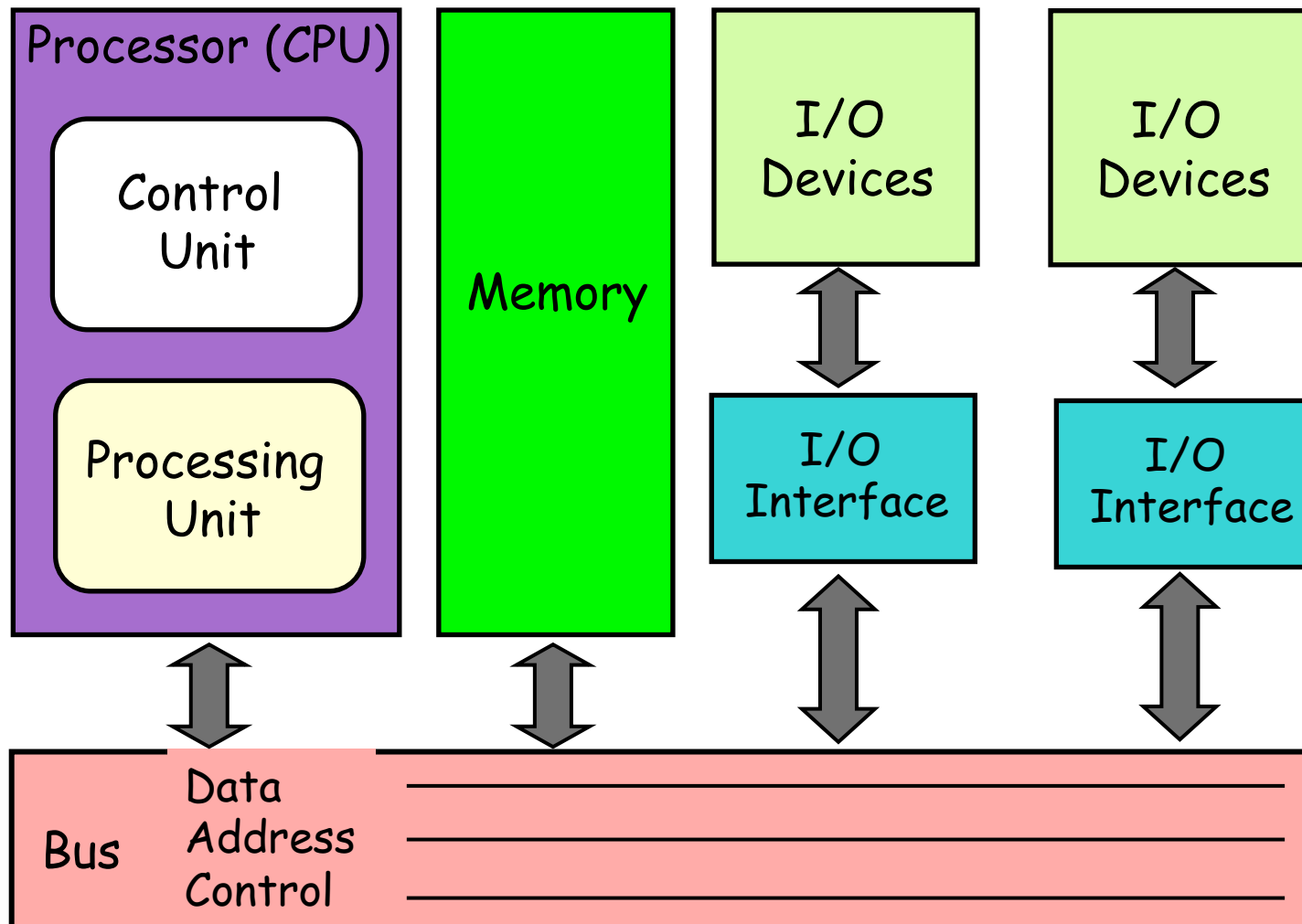
1. Accept input data codified in a digital format
2. Process input data by means of an executable program stored in the memory and running on the computer
3. Produce output data



Main components of a computer system

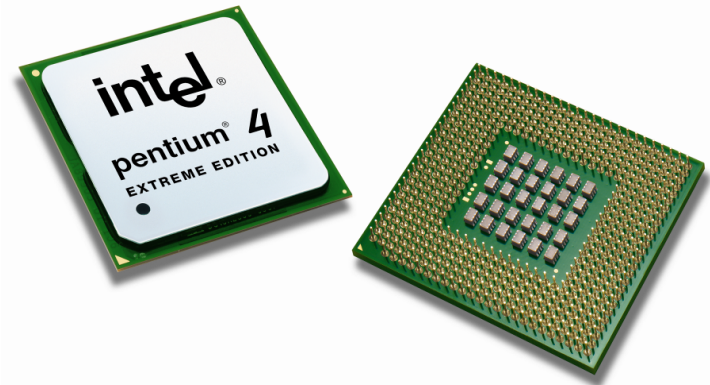


Components interconnection



Central Processing Unit (CPU)

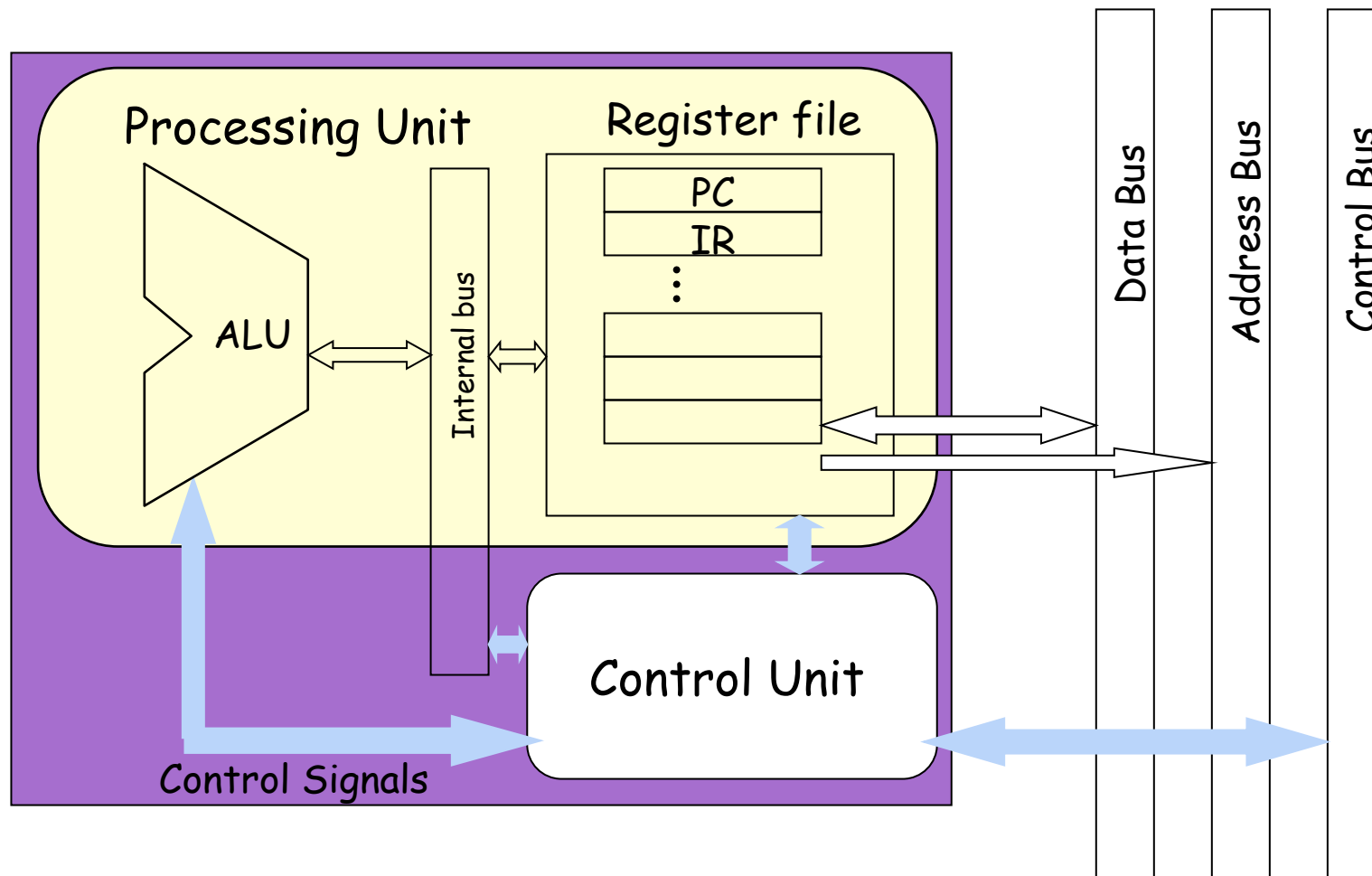
- Execute machine level instructions
- Execution cycle
 1. **Fetch** the instruction from the memory
 2. **Decode** the instruction
 3. **Execute** the instruction
- Each processor is characterized by its own set of instructions (machine level instructions), which is called **ISA** (Instruction Set Architecture)
- Processor speed is measured in gigahertz (GHz)
 - The higher this measurement, the faster the processor
 - One of the many factors that impact on processor performance



Central Processing Unit (CPU)

- The CPU is composed by two subsystems
 1. **Control Unit**
 - Control the sequencing and execution of instructions generating control signals
 2. **Processing Unit** (datapath)
 - Execute the instructions
 - **Arithmetic Logic Unit** (ALU)
 - Executes arithmetic and logic operations on data
 - **Register file**
 - Internal memory for the CPU composed by an array of registers
 - Two fundamental registers are:
 - **Program Counter** (PC), also called Instruction Pointer (IP)
 - Indicate where a computer is in its program sequence, providing the memory address of the next instruction to be executed
 - **Instruction Register** (IR)
 - Code of the instruction to be executed

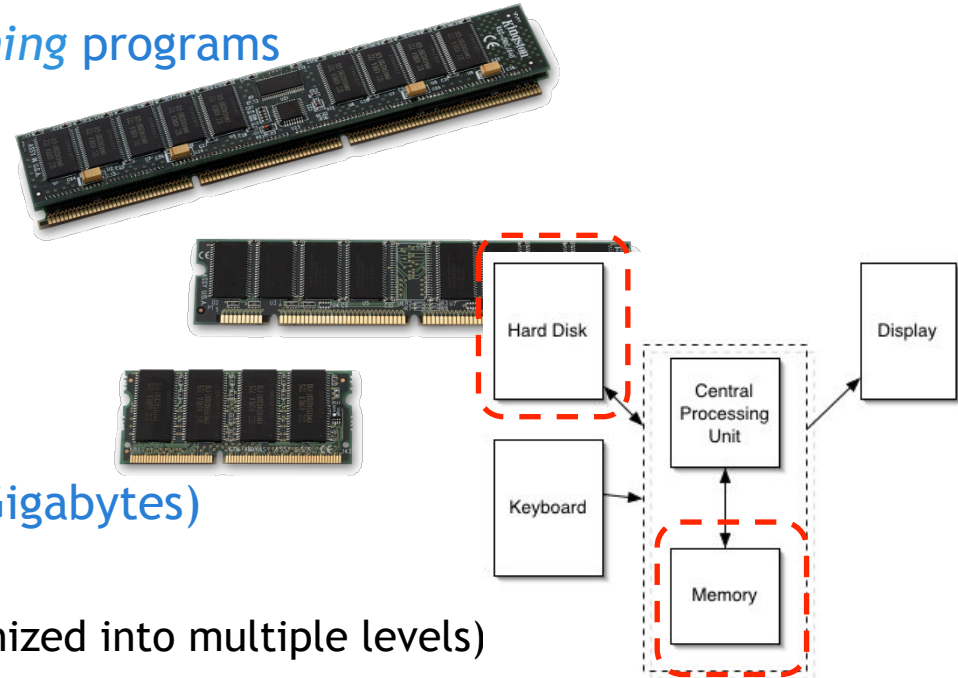
Central Processor Unit (CPU)



Main types of memory

■ Main memory or central memory

- Store data and operations of *running* programs
 - All in binary format
- Volatile (i.e., temporary)
- Random Access Memory (RAM)
 - Constant access time
 - SRAM, DRAM, etc.
- Fast (~10-100 nsecs), expensive
- “Limited” capacity (up to a few Gigabytes)
- Hierarchical structure
 - Cache memory (that can be organized into multiple levels) is faster and more expensive than RAM

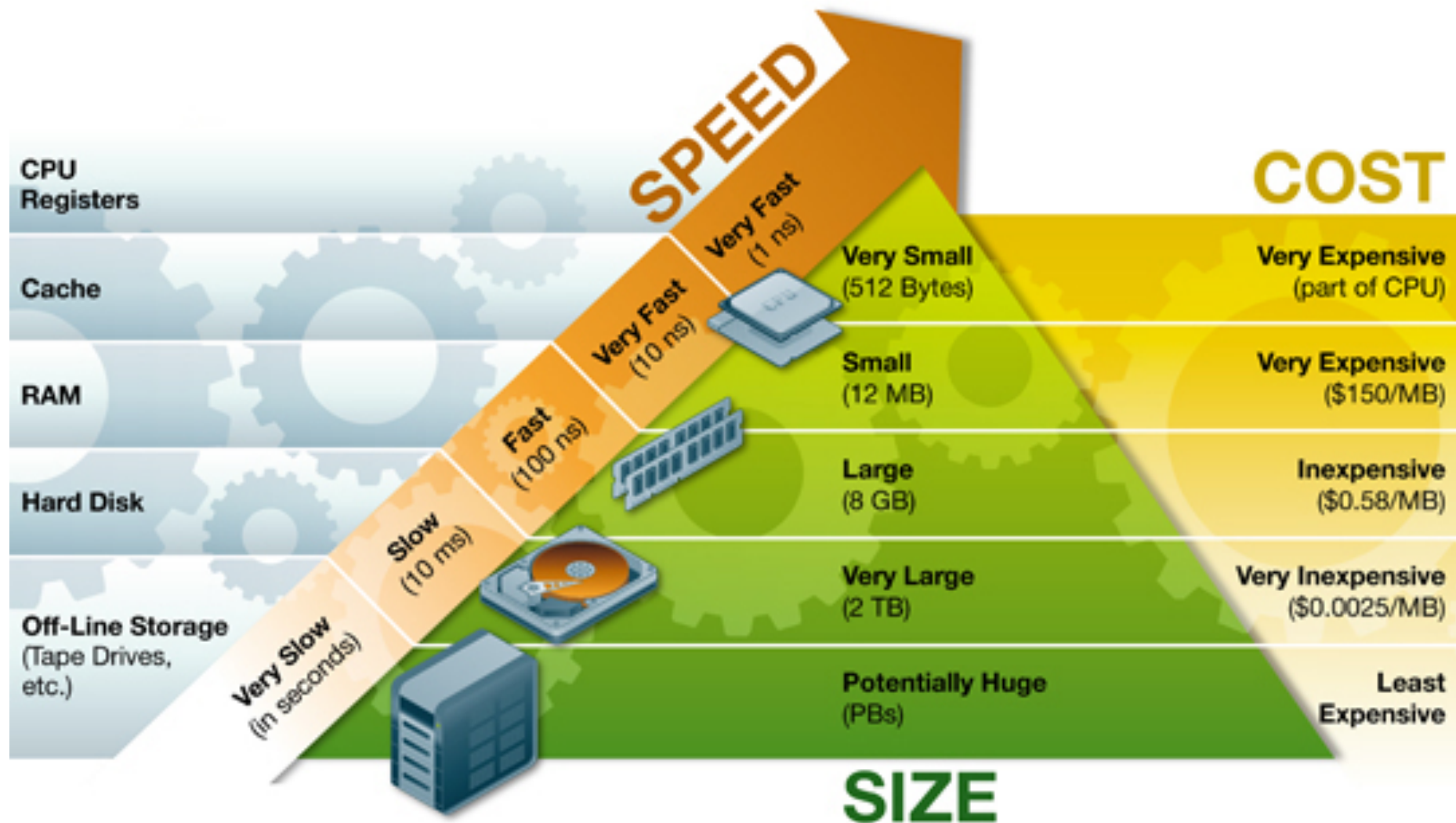


■ Secondary memory: hard disk, solid state disk (SSD), flash drive, USB memory, CD, etc..

- Non-volatile (i.e., permanent)
- Large capacity (several hundred Gigabytes)
- Slow (~msecs), cheap



Memory hierarchy

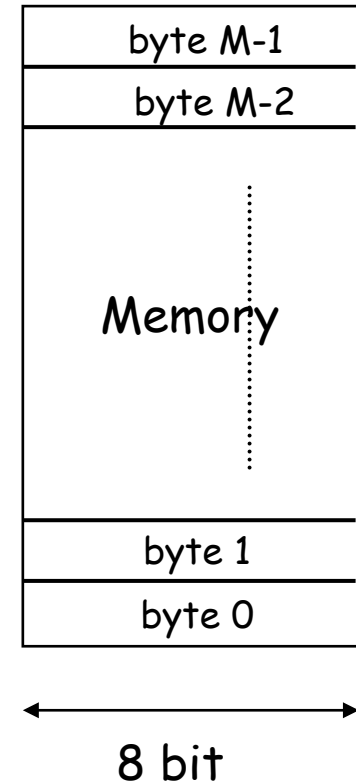


Memory hierarchy

- Top levels of the hierarchy (internal and main memory):
 - Made up of CPU registers, CPU cache and main memory
 - CPU registers and first level(s) of cache are internal to the CPU: very small but very fast and expensive
 - Volatile
 - The only storage components that are directly accessible to the CPU
- Bottom levels of the hierarchy (secondary storage)
 - Made up by solid state disks, hard disks, Flash drives, USB memory, ...
 - Non-volatile: do not lose data when the computer is powered off
- Going down the hierarchy: slower, larger and cheaper memory
- Going up the hierarchy: faster, smaller and more expensive memory

Main memory

- Consisting of *cells (locations)*, with each cell consisting in turn of a fixed number of binary elements
 - Each binary element (**bit**) can store (represent) only two values: 0 or 1
 - Usually, one **cell** = **1 byte** or 1B (8 bit)
 - **Words** are group of bytes: 2, 4, or 8 bytes = 16, 32, or 64 bits
- Each cell is associated with a unique *address* in the range $[0, 1, \dots, M-1]$ (analogous to street addresses for houses)
 - *M*: memory size (e.g., 4 Gbytes, 2 Mbytes, 512 Kbytes)
 - Main memory can be seen as a “vector” of bytes
- CPU reads/writes cell content by specifying the cell address
 - **Read**: to fetch the content of a memory cell
 - **Write**: to modify the content of a memory cell
 - *m* bit address \Rightarrow address space 2^m
 - not necessarily: $M = 2^m$



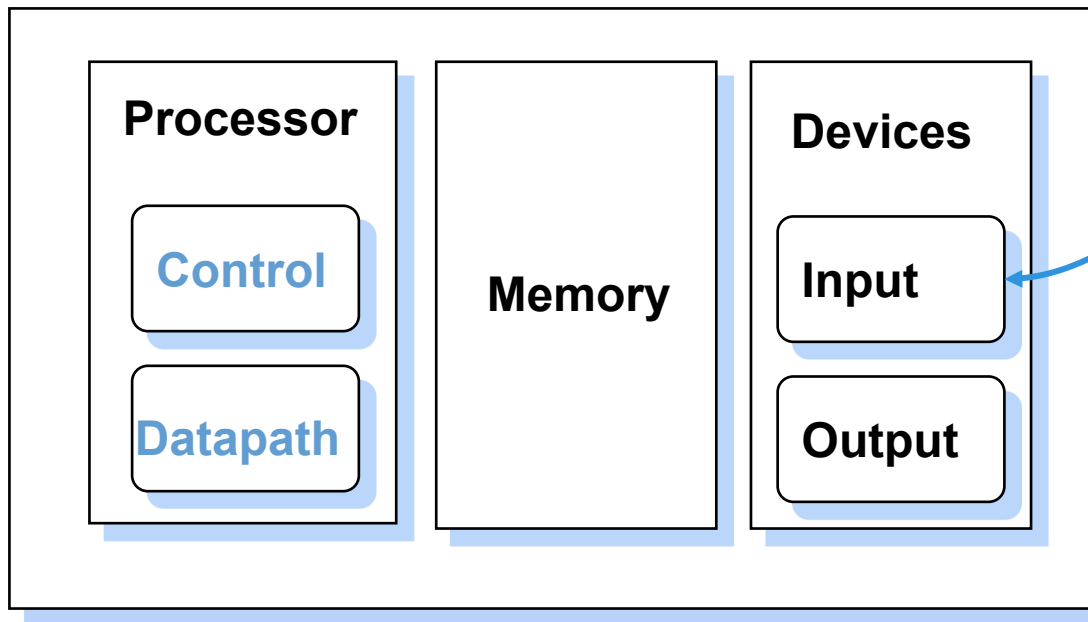
Multiples of bytes

- Unit symbol of byte: B
 - 1 KB = 1 Kilobyte = 1024 B = 2^{10} B $\approx 10^3$ B
 - 1 MB = 1 Megabyte = 1024 KB = 2^{20} B $\approx 10^6$ B
 - 1 GB = 1 Gigabyte = 1024 MB = 2^{30} B $\approx 10^9$ B
 - 1 TB = 1 Terabyte = 1024 GB = 2^{40} B $\approx 10^{12}$ B
 - 1 PB = 1 Petabyte = 1024 TB = 2^{50} B $\approx 10^{15}$ B
 - 1 EB = 1 Exabyte = 1024 PB = 2^{60} B $\approx 10^{18}$ B
 - 1 ZB = 1 Zettabyte = 1024 EB = 2^{70} B $\approx 10^{21}$ B
-
- According to IDC, in 2013 there were 4.4 ZB in the digital universe, that will become 44 ZB in 2020: a huge amount of data (see Big Data)

Program execution

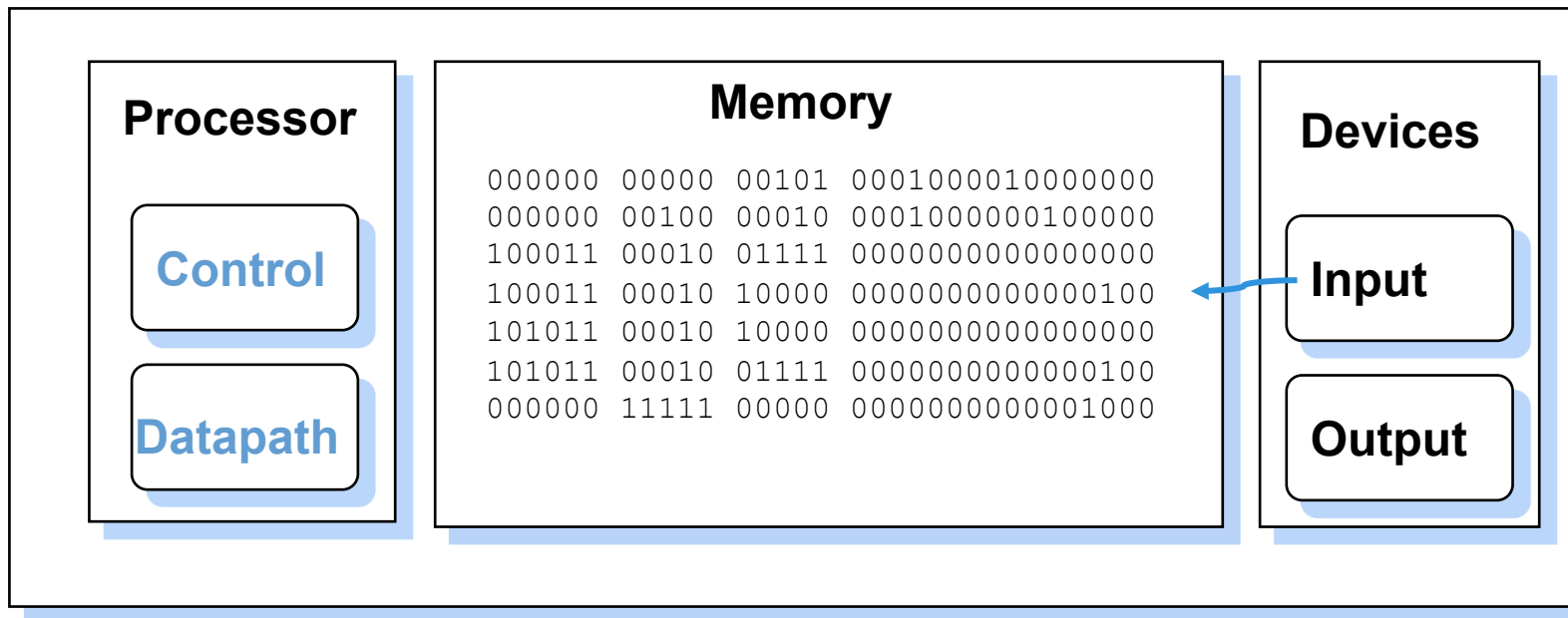
Input: Code/Data

```
000000 00000 00101 00010000010000000
000000 00100 00010 0001000000100000
100011 00010 01111 0000000000000000
100011 00010 10000 00000000000000100
101011 00010 10000 00000000000000000
101011 00010 01111 00000000000000100
000000 11111 00000 00000000000001000
```



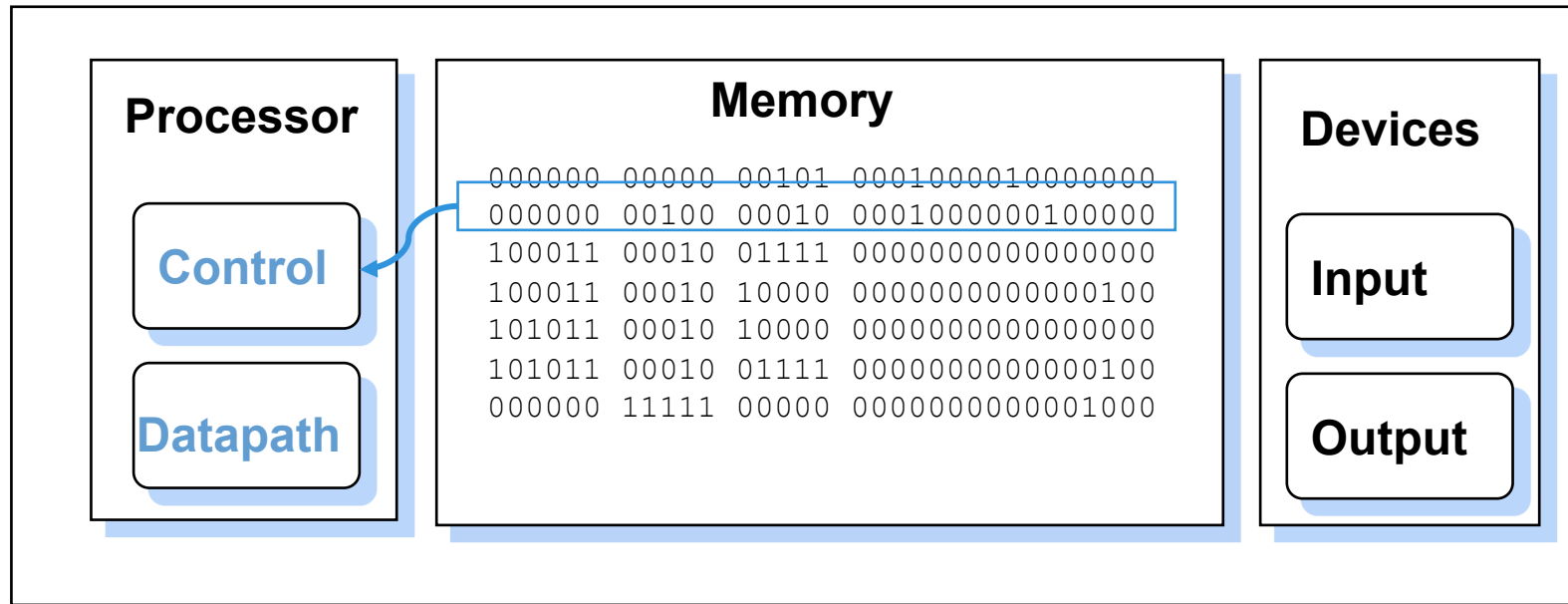
Program execution

Code/Data stored in memory



Program execution: fetch

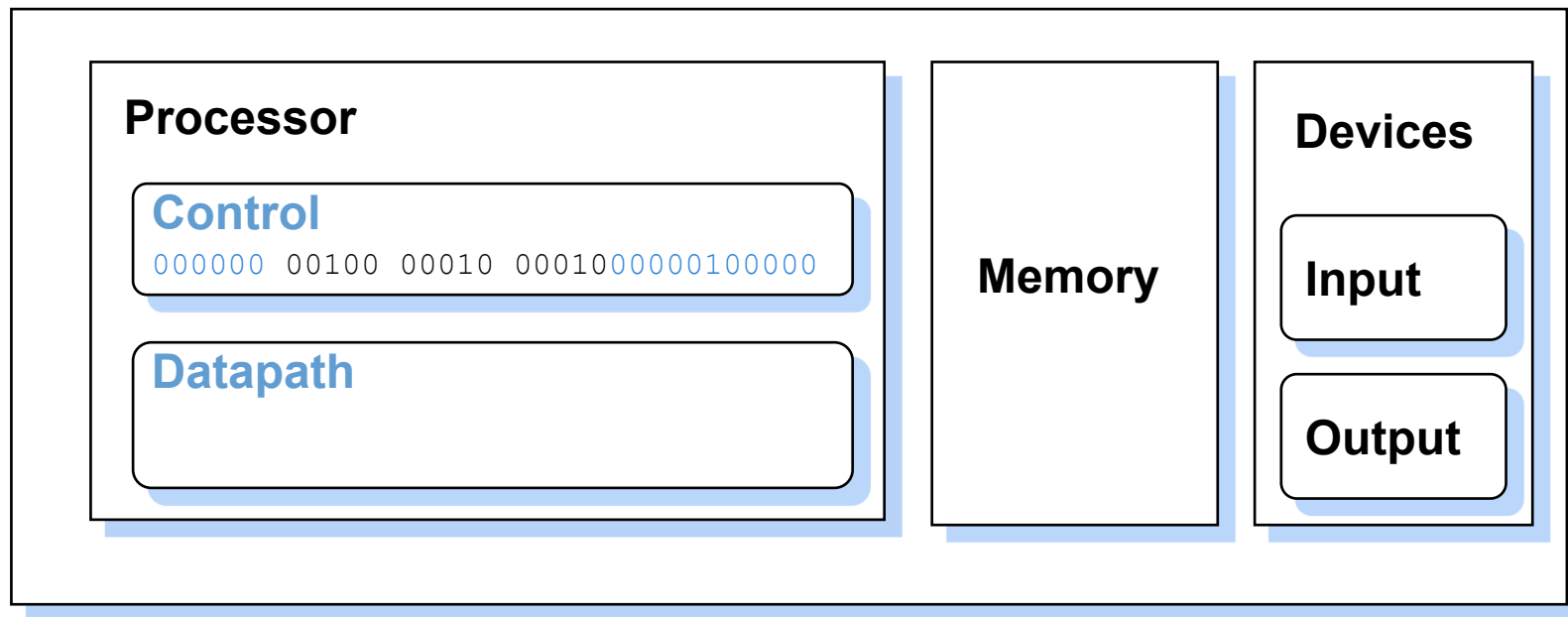
The processor **fetches from the memory** an instruction.
Which instruction? That pointed by the program counter



Program execution: decode

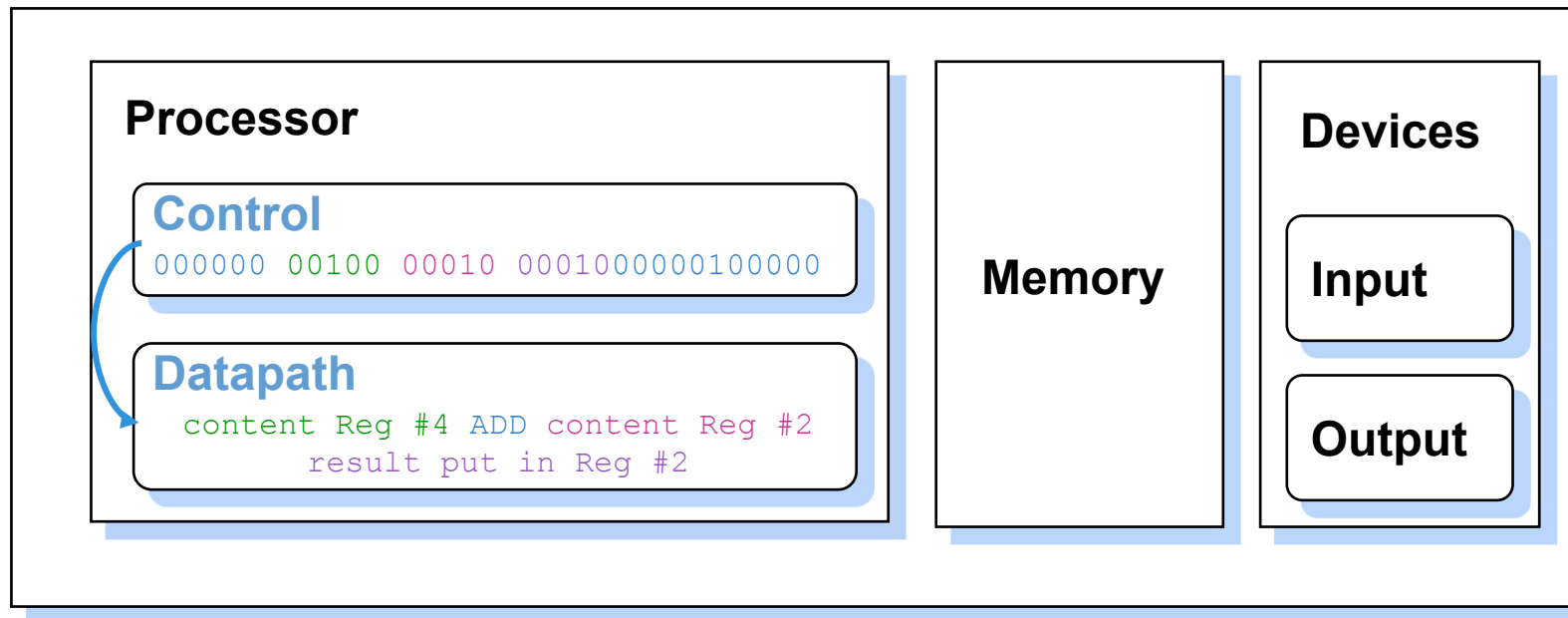
The control unit in the processor
decodes the instruction, that is
determines what actions the instruction
requires

What do I have to do?

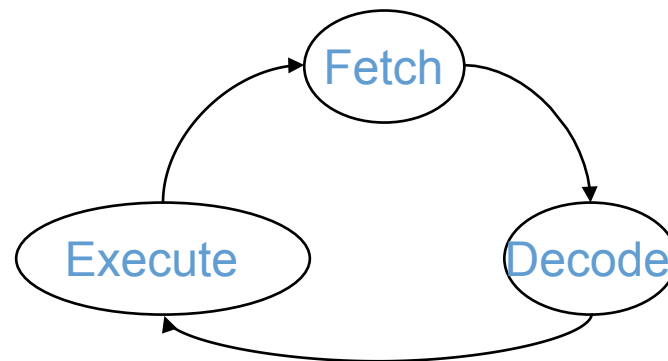
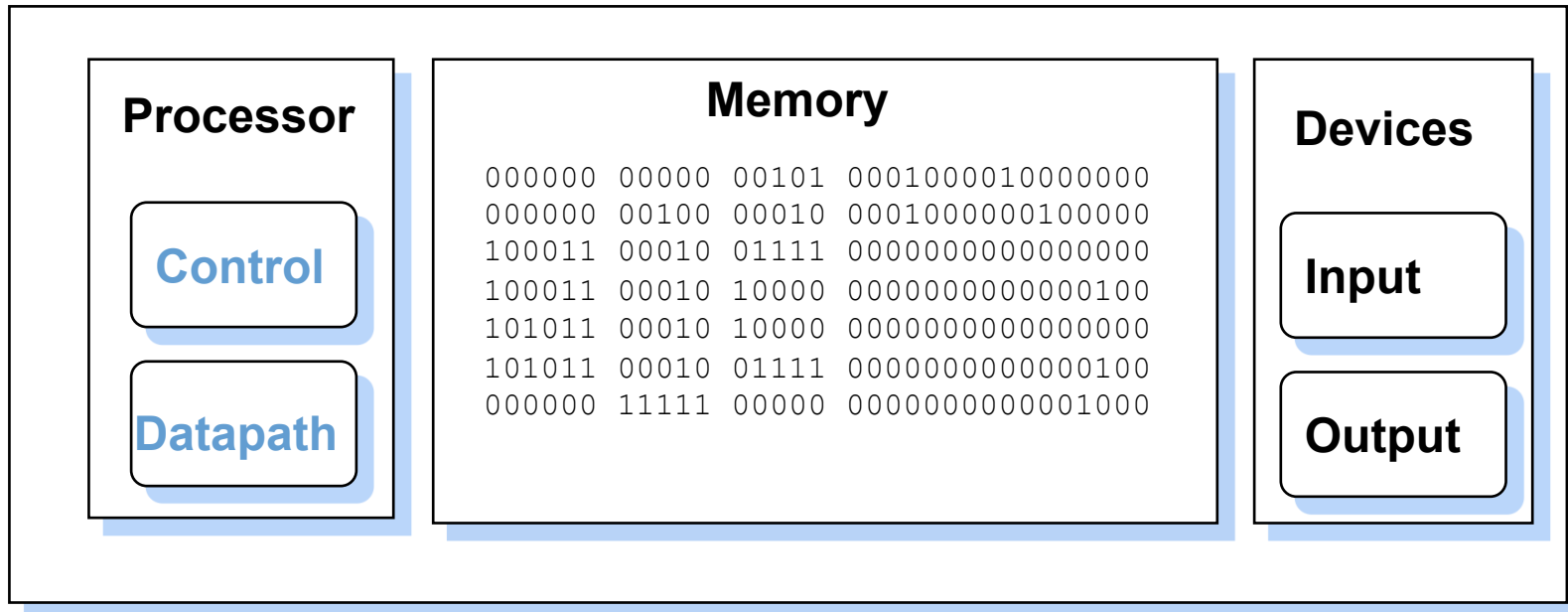


Program execution: execute

The processing unit **executes the instruction** (eventually interacting with the memory loading operands and saving results)

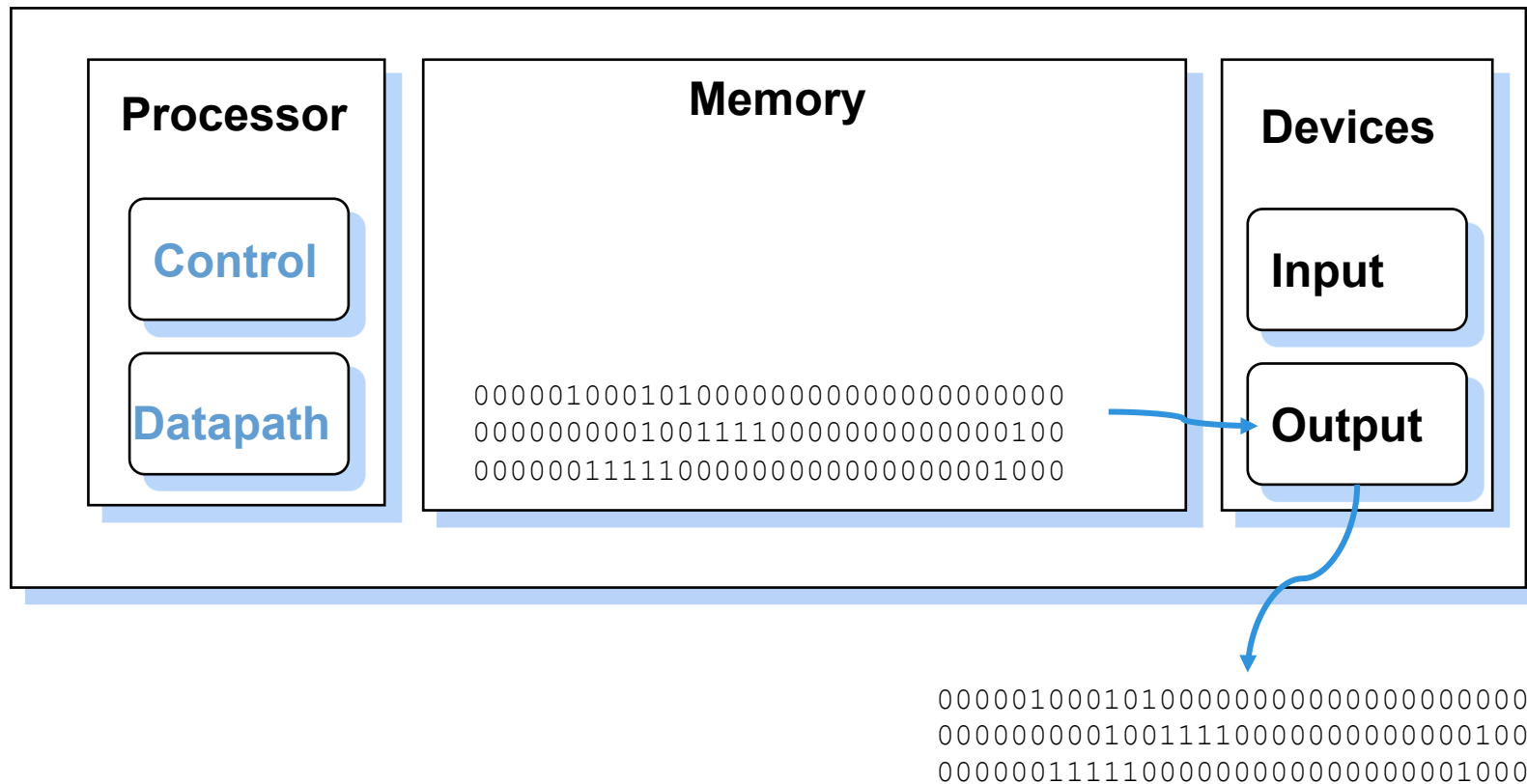


Program execution



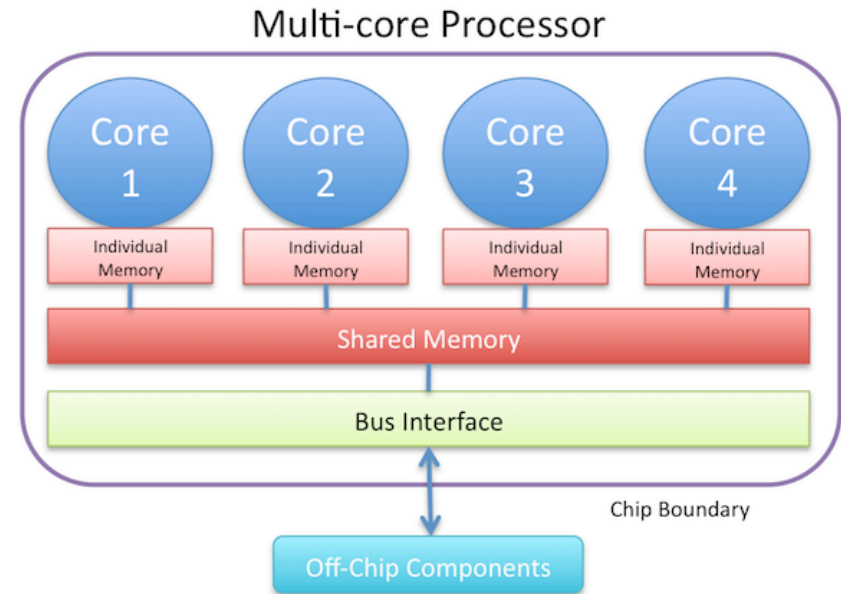
Program execution

When the program terminates, the results are sent to the output device



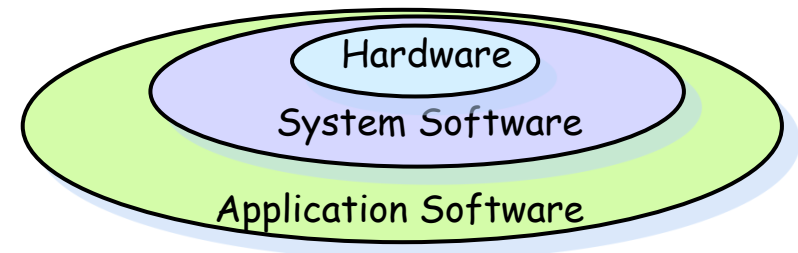
Multi-core processor

- A multi-core processor is a single computing component with two or more independent actual processing units (called **cores**)
 - E.g., quad-core means 4 cores on the same chip
- Each core reads and executes program instructions
- Multiple cores can run multiple instructions at the same time, increasing overall speed



Layers of abstraction

- Basically, we have not this “raw” vision of a computer
- High-level operations
 - Mathematical functions (log, sine, ...)
 - Text processing
 - ...
- High-level memory
 - Set of cells identified by a *name*
 - user defined
 - High-level cell content
 - text, image, set of ..., ...
 - Secondary memory organized as a set of named files
- Achieved through stratified layers of software



Summary

- What computers understand
- How computers work

Suggested material

- Introduction to computer systems architecture and programming
http://www.londoninternational.ac.uk/sites/default/files/programme_resources/lse/lse_pdf/subject_guides/is1168_ch1-4.pdf
- Introduction to computer architecture
<http://people.cis.ksu.edu/~schmidt/300s05/Lectures/ArchNotes/arch.html>
- A short video: Intro to computer architecture
https://www.youtube.com/watch?v=HEjPop-aK_w