ARCOS Group

uc3m | Universidad Carlos III de Madrid

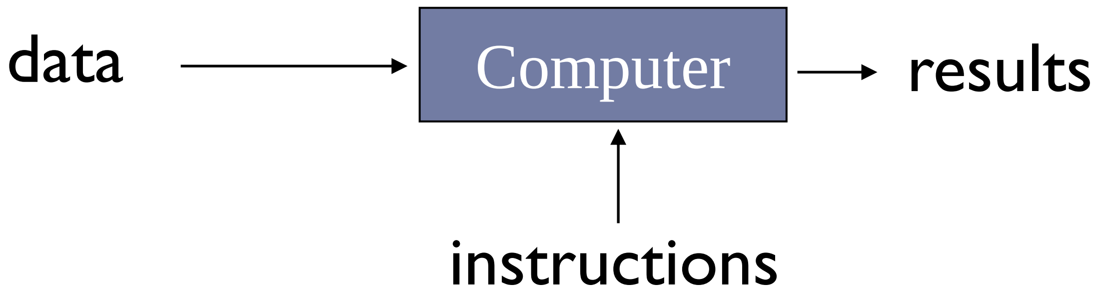
Lesson 1
Introduction to computers

Computer Structure

Bachelor in Computer Science and Engineering

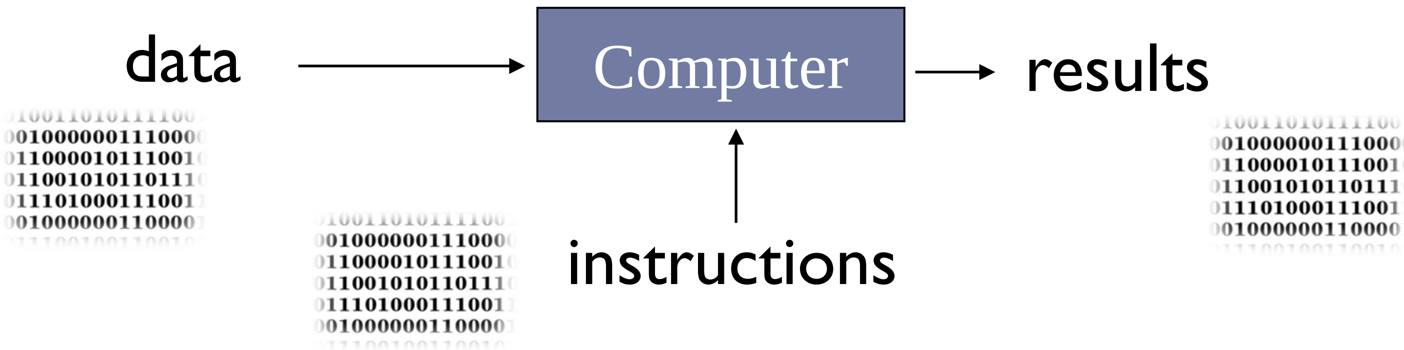
Content

What is a computer?



- **Computer: machine designed to process data.**
  - Instructions are applied to data and then results (data/information) are obtained.

What is a computer?



- **Computer: machine designed to process data.**
  - Digital computer: data and instructions in binary format.

What does a computer look like?



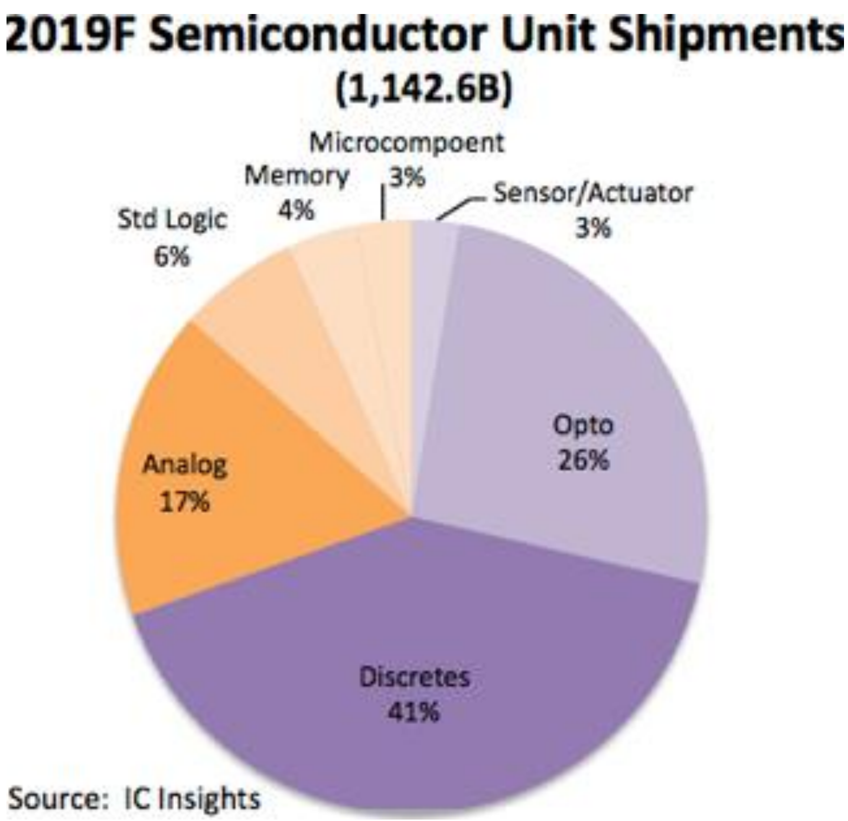




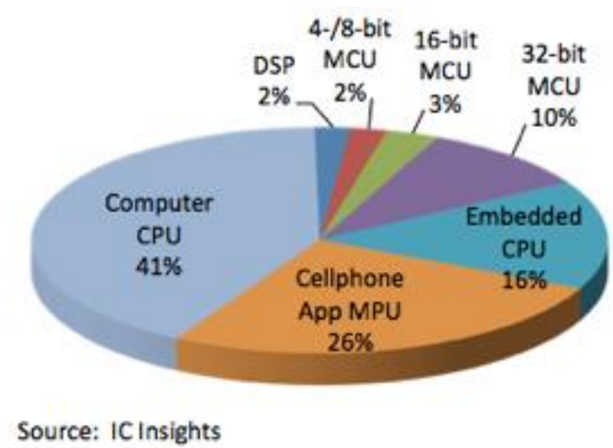




Semiconductor industry



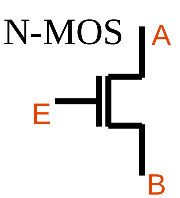
- Processors:
  3% of the industry

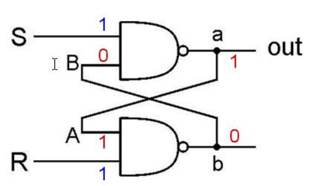


Content

1. What is a computer?
2. **Computer structure and computer architecture**
3. Building blocks for a computer
4. Von Neumann architecture
5. Machine instructions and assembly programming
6. Execution steps of an instruction
7. Main characteristic parameters of a computer
8. Types of computers
9. Historic evolution

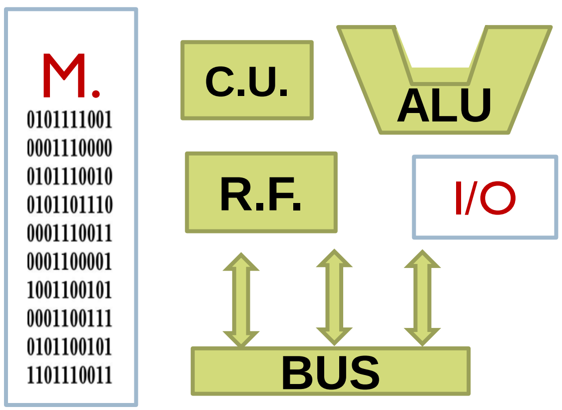
What aspects of a computer do I need to know?

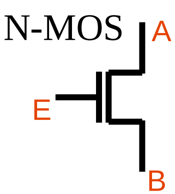


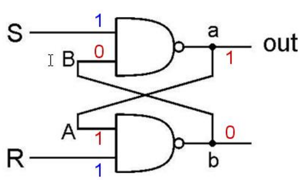


- **Technology:**
  - How components are built

What aspects of a computer do I need to know?

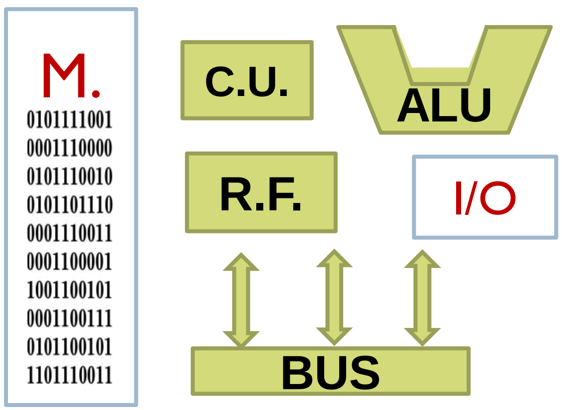


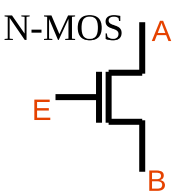


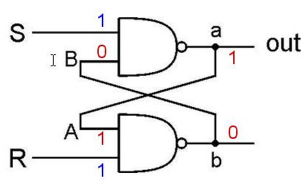


- **Structure:**
  - Components and their organization
- **Technology:**
  - How components are built

What aspects of a computer do I need to know?







- **Architecture:**
  - Attributes visible to a programmer
- **Structure:**
  - Components and their organization
- **Technology:**
  - How components are built

Structure and Architecture

- Structure
  - Components of a computer
  - Organization of the components
- Architecture: visible attributes for programmers
  - Instruction set offered by the computer (ISA, Instruction Set Architecture)
  - Type and format of data that the computer is capable of using.
  - Number and size of registers
  - Input/Output (I/O) techniques and mechanisms
  - Addressing and memory access techniques
- Technology: how the components are built

Content

1. What is a computer?
2. Computer structure and computer architecture
3. **Building blocks for a computer**
4. Von Neumann architecture
5. Machine instructions and assembly programming
6. Execution steps of an instruction
7. Main characteristic parameters of a computer
8. Types of computers
9. Historic evolution

Review

- Binary system based on: 0 y 1

- Building blocks: transistors, logic gates, …:

Transistors

Logic gates

Sequential

Combinational

- Registers
- …

Félix García-Carballeira,   Alejandro Calderón Mateos

Binary system

- Binary

$$X = \begin{array}{cccccccc} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ \ldots 2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \end{array}$$

Binary digit d_i

Weight p_i

- Value = $d_{31} \times 2^{31} + d_{30} \times 2^{30} + \ldots + d_1 \times 2^1 + d_0 \times 2^0$

Félix García-Carballeira, Alejandro Calderón Mateos

Binary system

- Binary

$$X = 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1$$

$$\ldots 2^7 \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0$$

Binary digit d_i

Weight p_i

- Value = $d_{31} \times 2^{31} + d_{30} \times 2^{30} + \ldots + d_1 \times 2^1 + d_0 \times 2^0$

- How many values can be represented with n bits?
- How many bits are necessary to represent m 'values'?

- With n bits, if the values to be represented are numbers and start at 0, what is the maximum representable value?

Félix García-Carballeira, Alejandro Calderón Mateos

Binary system

- Binary

$$X = \begin{array}{cccccccc} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ \ldots 2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \end{array}$$

Binary digit d_i

Weight p_i

- Value = $d_{31} \times 2^{31} + d_{30} \times 2^{30} + \ldots + d_1 \times 2^1 + d_0 \times 2^0$

- How many values can be represented with n bits? **2^n**
- How many bits are necessary to represent m 'values'? **$\text{Log}_2(m)$ rounded up**
- With n bits, if the values to be represented are numbers and start at 0, what is the maximum representable value? **$2^n - 1$**

Question?

- How many different codes can be coded with 8 bits?
- How many bits are needed to represent 512 codes?

Review

- Binary system based on: 0 y 1

- Building blocks: transistors, logic gates, …:

Transistors

Logic gates

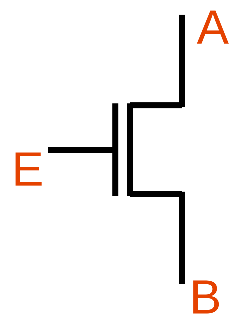
Sequential Combinational

- Registers
- …

Félix García-Carballeira, Alejandro Calderón Mateos

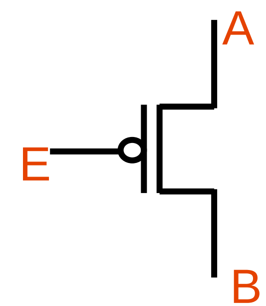
Transistor

N-MOS



E	Behavior
1	Connects A to B (open circuit)
0	Does not connect A to B (closed circuit)

P-MOS



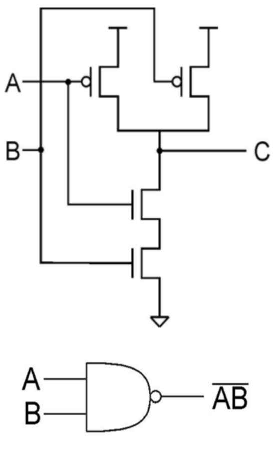
E	Behavior
1	Connects A to B (open circuit)
0	Does not connect A to B (closed circuit)

- A transistor acts as a switch
- The p-type and n-type transistors are MOSFET (Metal-Oxide-Semiconductor-Field-Effect Transistor) transistors.
- Origin of CMOS family in the combination of p-type and n-type transistors.

Félix García-Carballeira, Alejandro Calderón Mateos

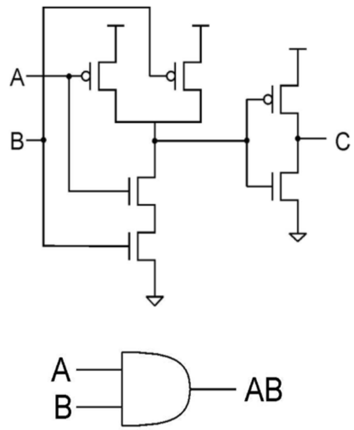
Logic gates

NAND



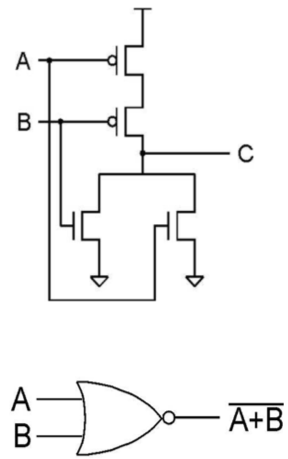
A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

AND



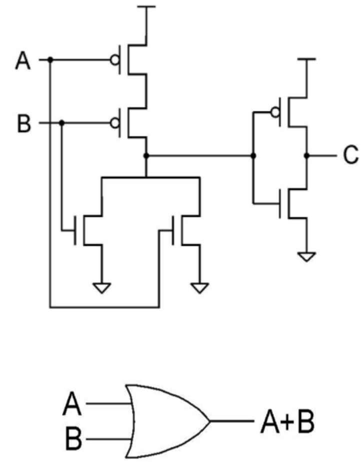
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

NOR



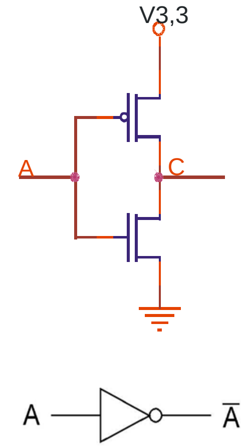
A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

OR



A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

NOT



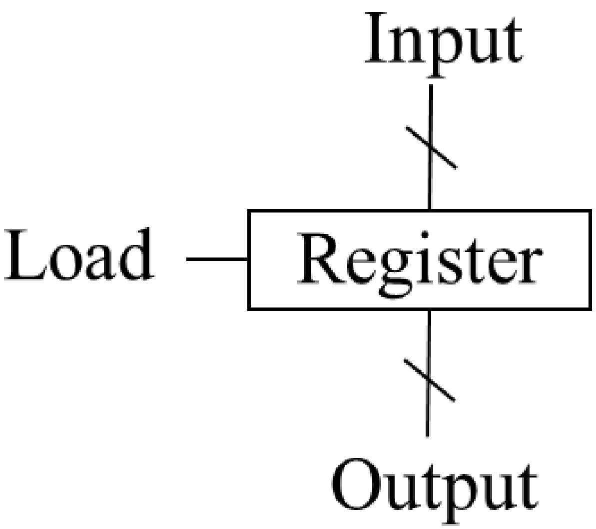
A	C
1	0
0	1

Combinational and sequential circuits

- **Combinational circuits**: the output depends only on the input values
  - Examples:
    - Decoders
    - Multiplexers
    - Arithmetic and logical operators
- **Sequential circuits**: Output depends on input and current state. Store information.
  - Examples:
    - Flip-flops
    - Registers

Register

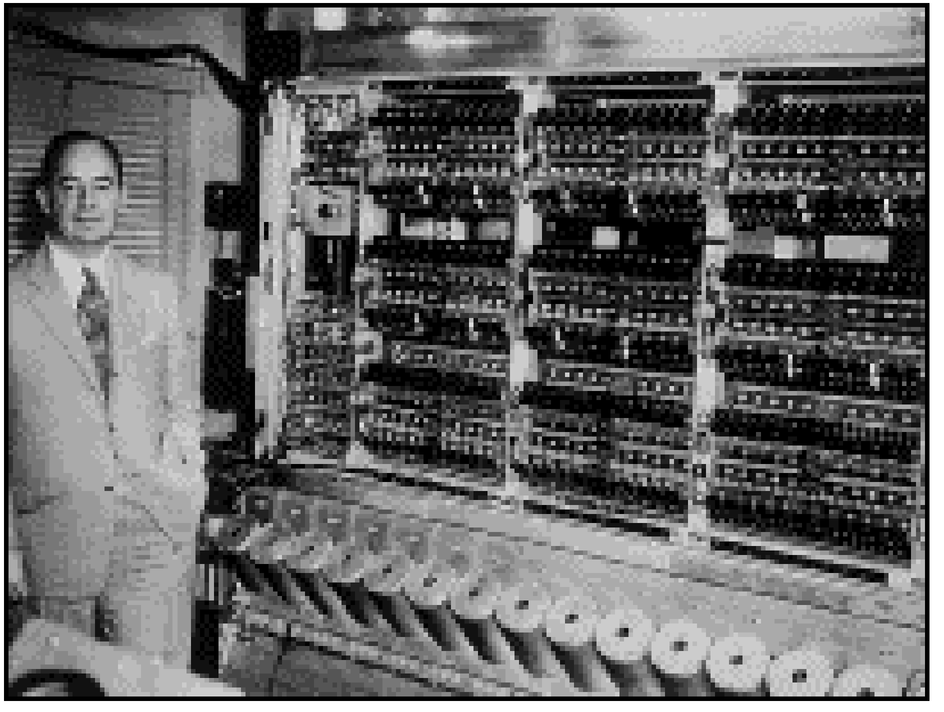
- Element that stores n bits (at the same time)



Content

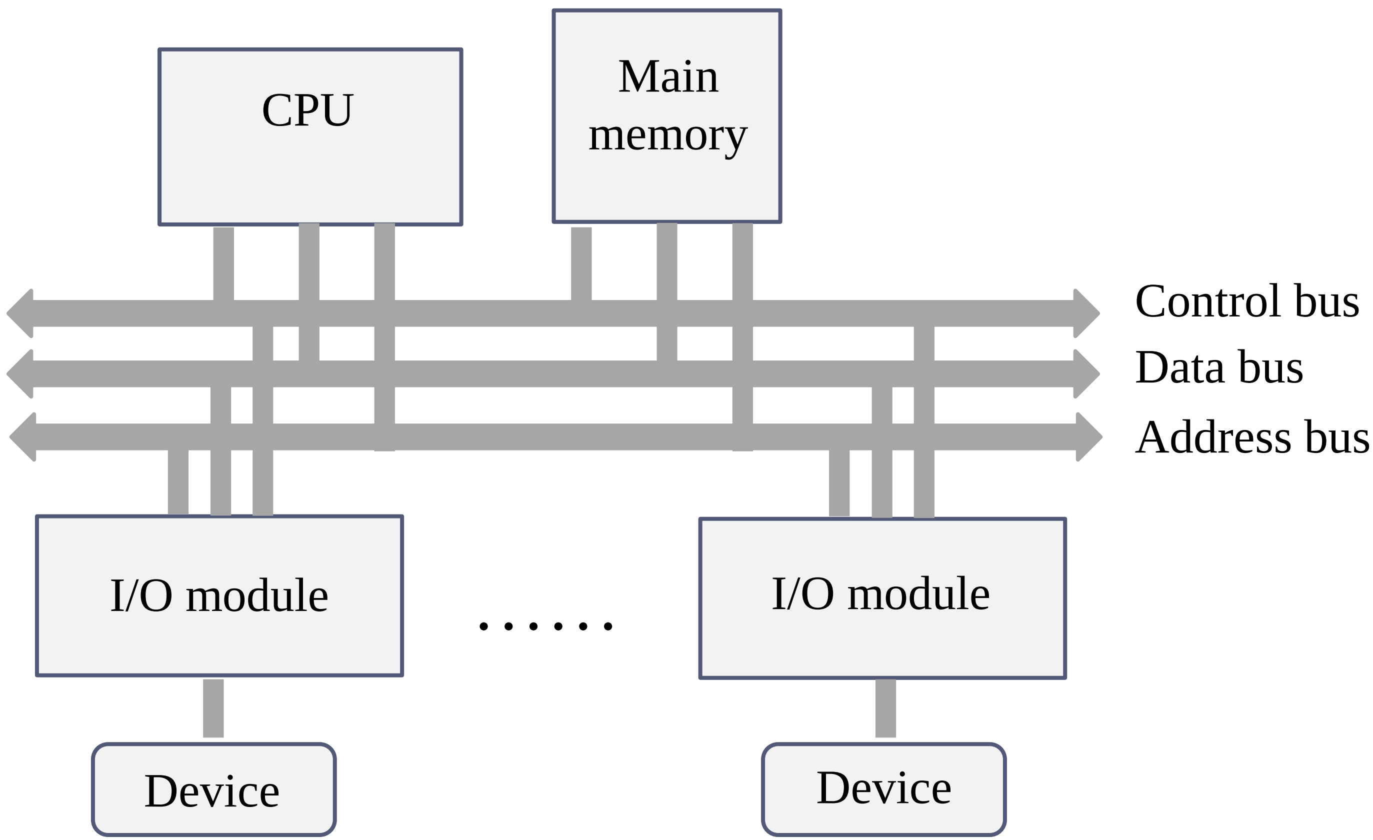
1. What is a computer?
2. Computer structure and computer architecture
3. Building blocks for a computer
4. **Von Neumann architecture**
5. Machine instructions and assembly programming
6. Execution steps of an instruction
7. Main characteristic parameters of a computer
8. Types of computers
9. Historic evolution

Von Neumann computer

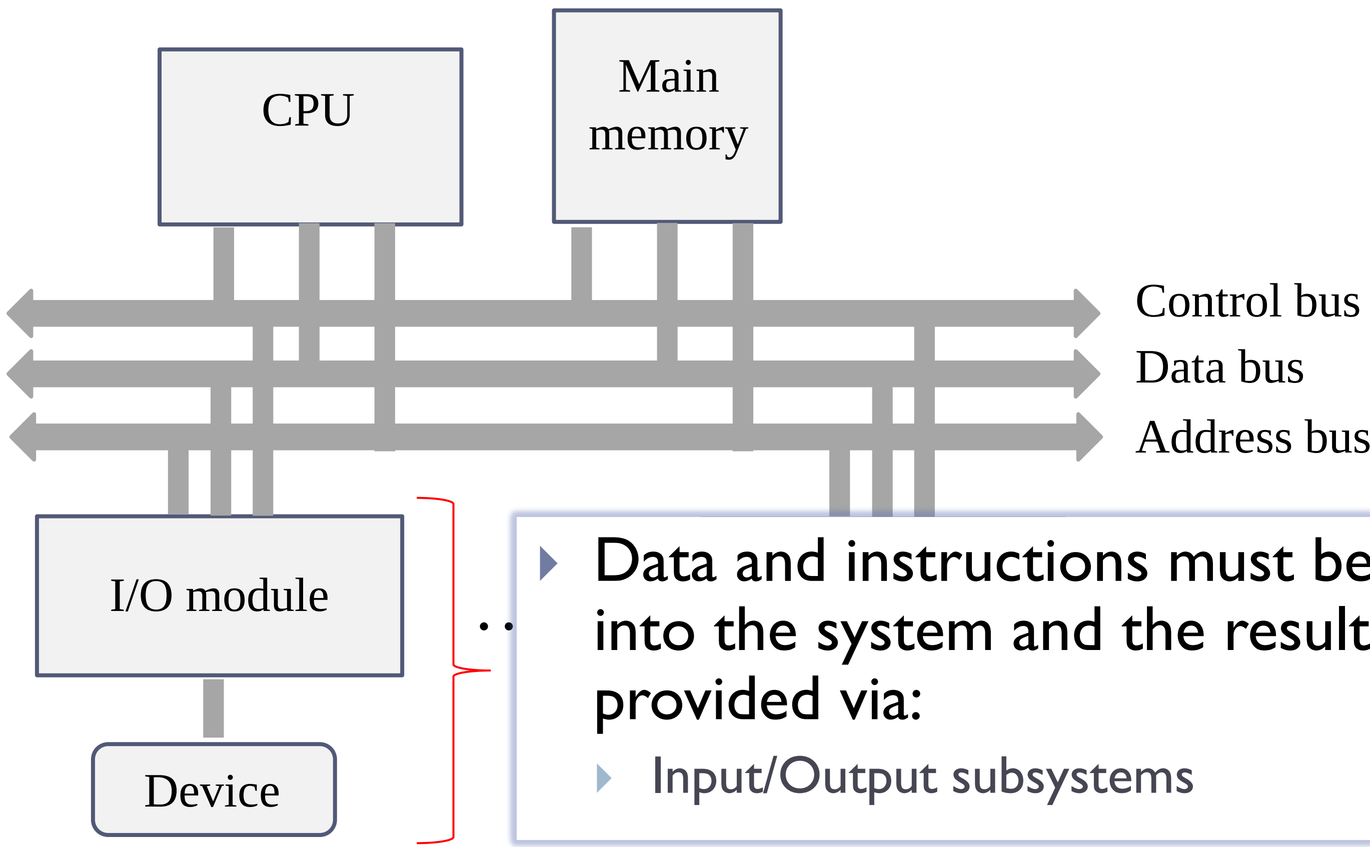


Machine capable of executing a series of elementary instructions (machine instructions) that are stored in memory (read and executed).

Von Neumann architecture

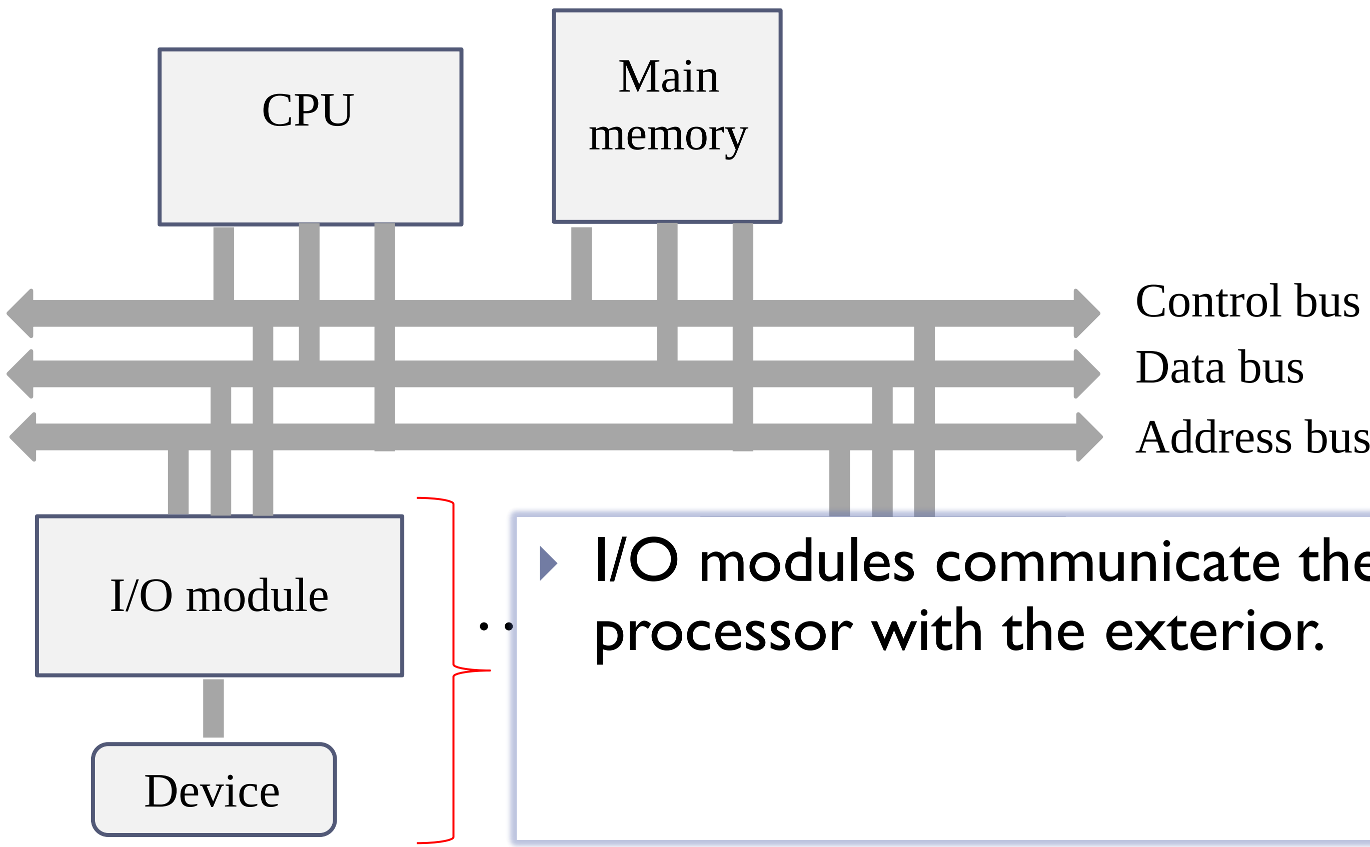


Von Neumann architecture (1/4)



- Data and instructions must be entered into the system and the results are provided via:
  - Input/Output subsystems

Von Neumann architecture (1/4)



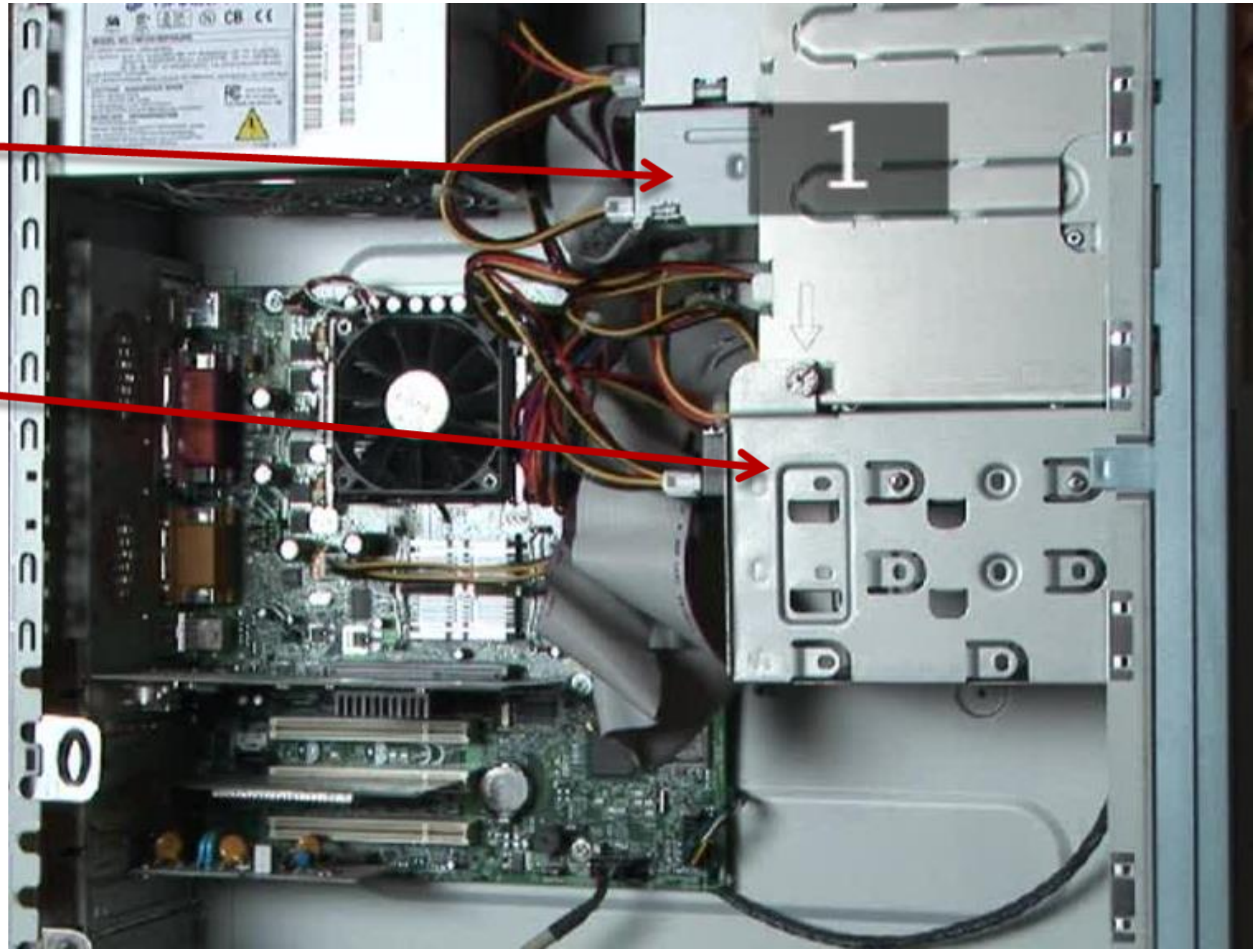
- I/O modules communicate the processor with the exterior.

Example of I/O module + devices

storage

CD-ROM/
DVD-ROM/
BluRay/…

Hard disk

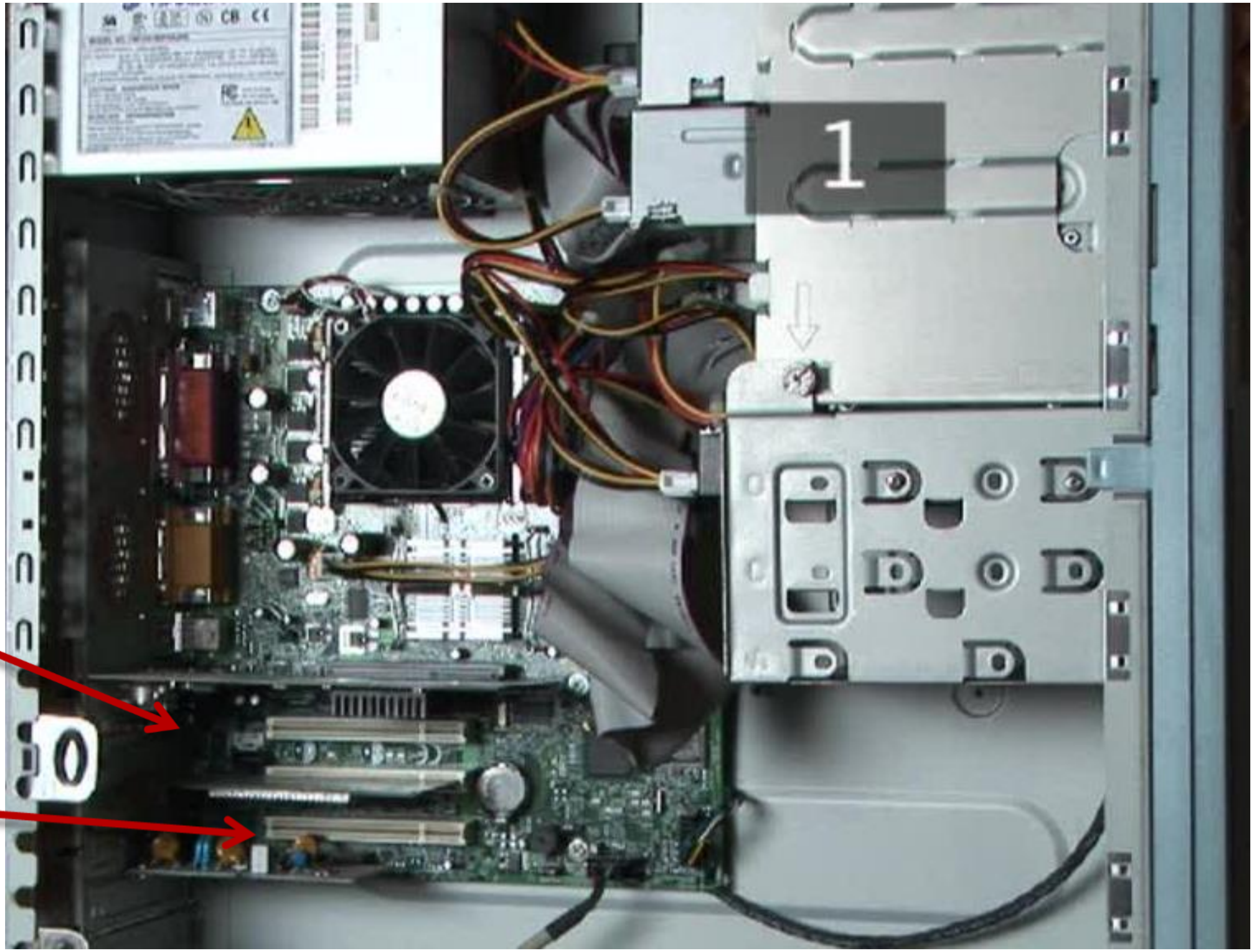


http://www.videojug.com/film/what-components-are-inside-my-computer

Example of I/O module + devices communication

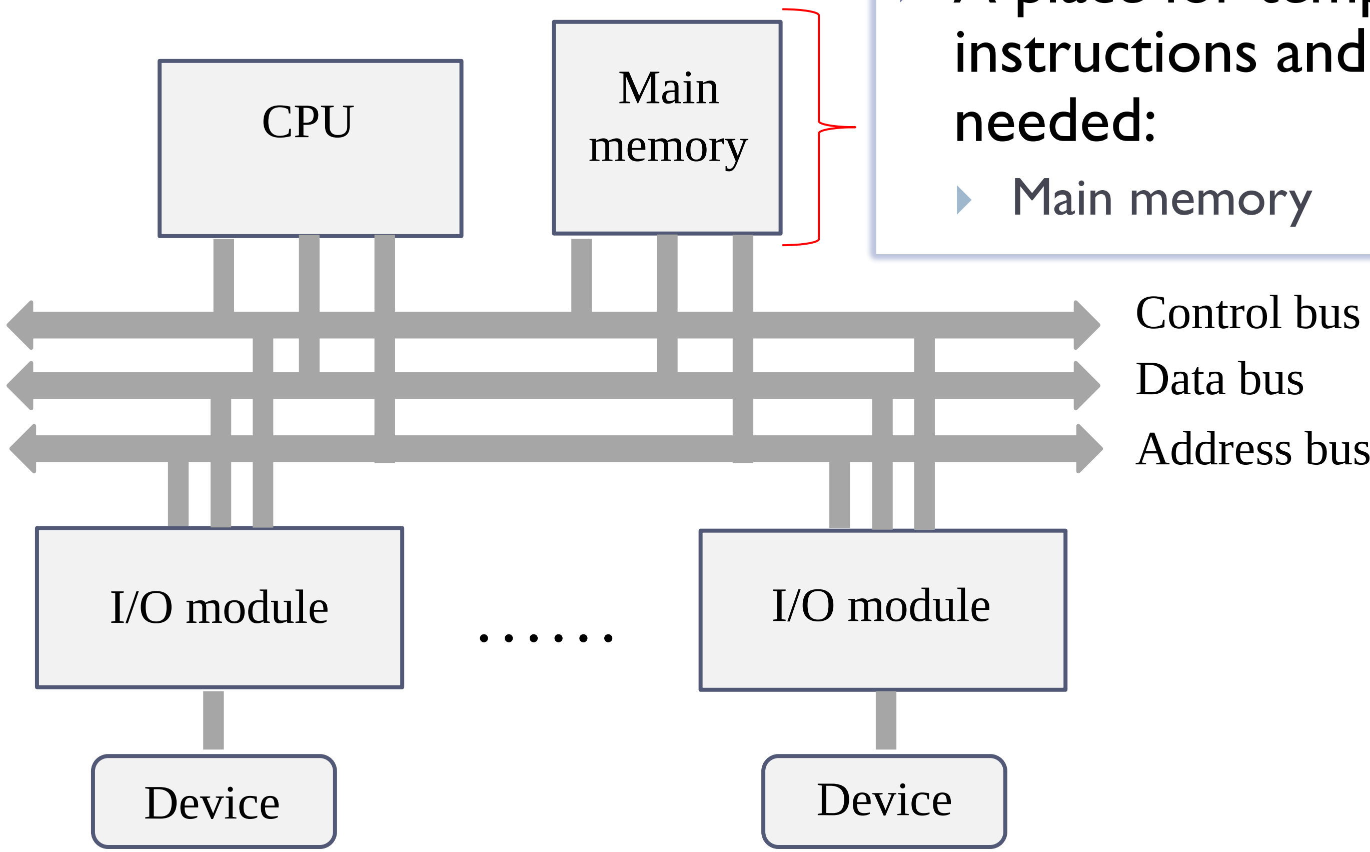
Network card

Sound card



http://www.videojug.com/film/what-components-are-inside-my-computer

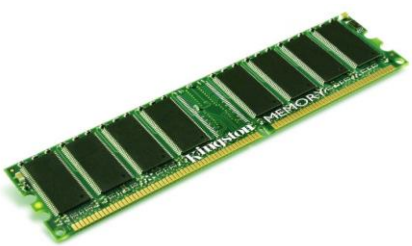
Von Neumann architecture (2/4)

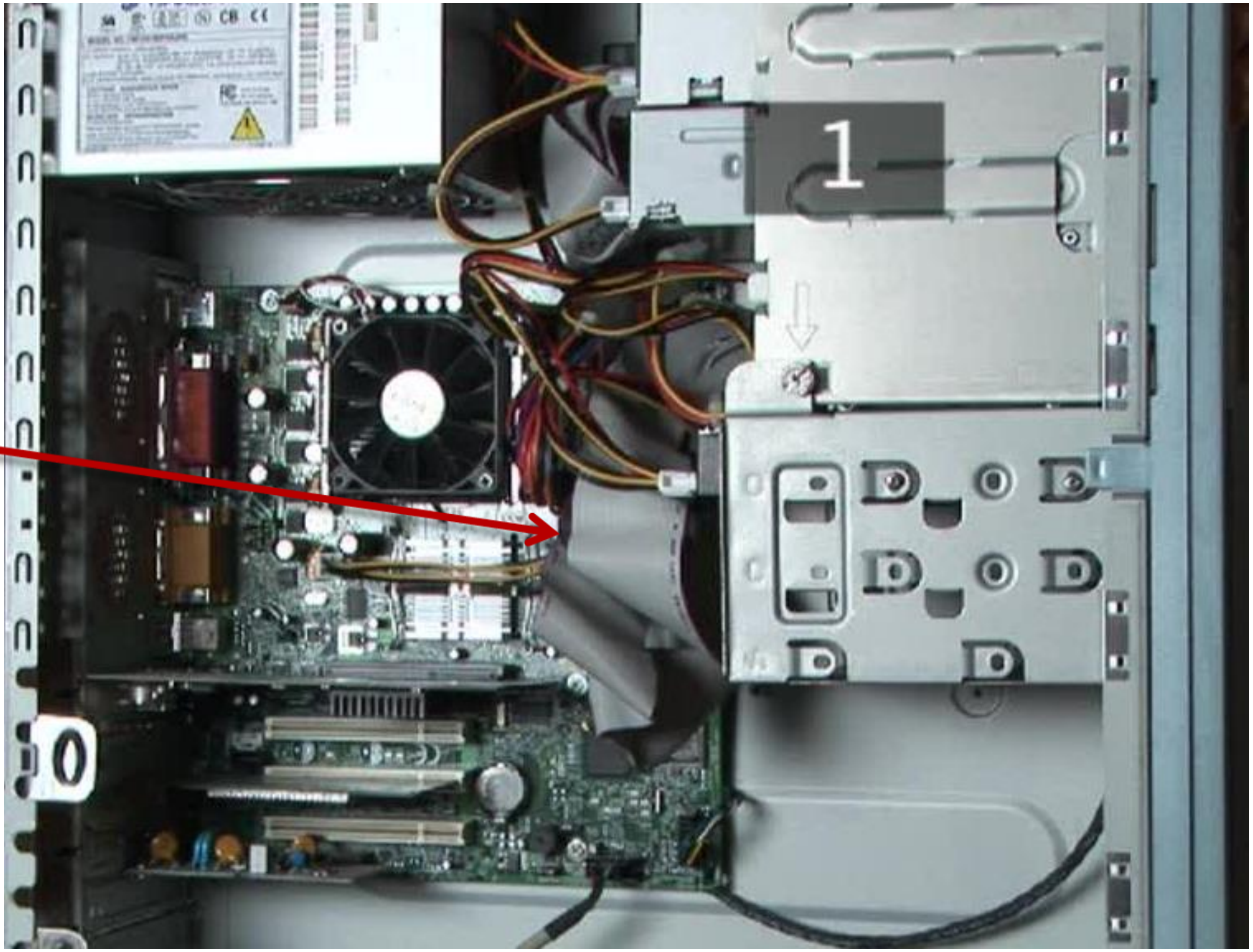


- A place for temporary store instructions and data is needed:
  - Main memory

Example: main memory

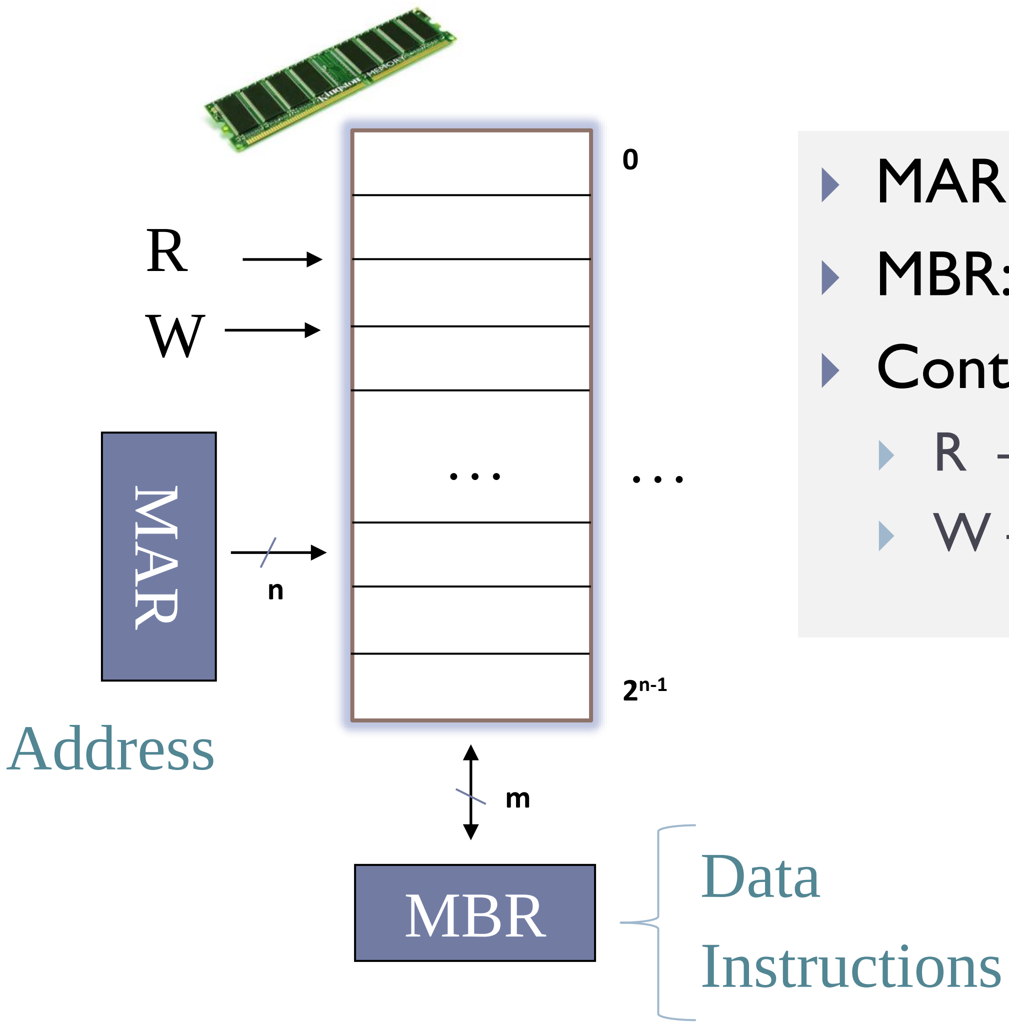
Main memory





http://www.videojug.com/film/what-components-are-inside-my-computer

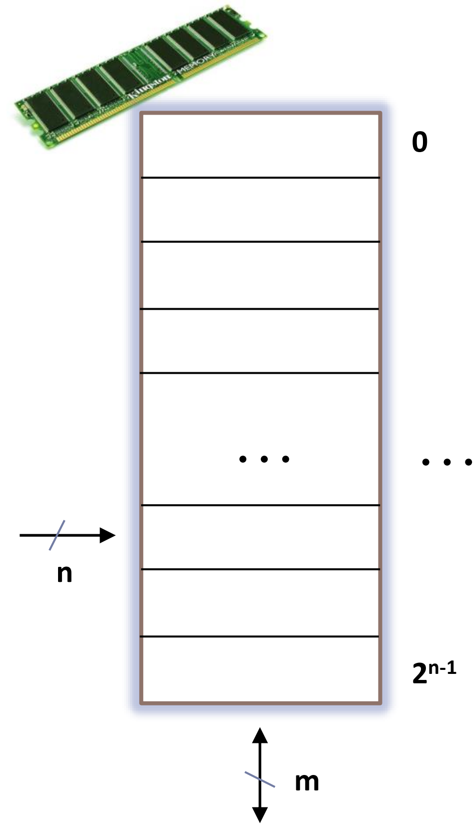
Main memory elements



- MAR: Memory Address Register
- MBR: Memory Buffer Register
- Control signals
  - R – Read from memory
  - W – Write to memory

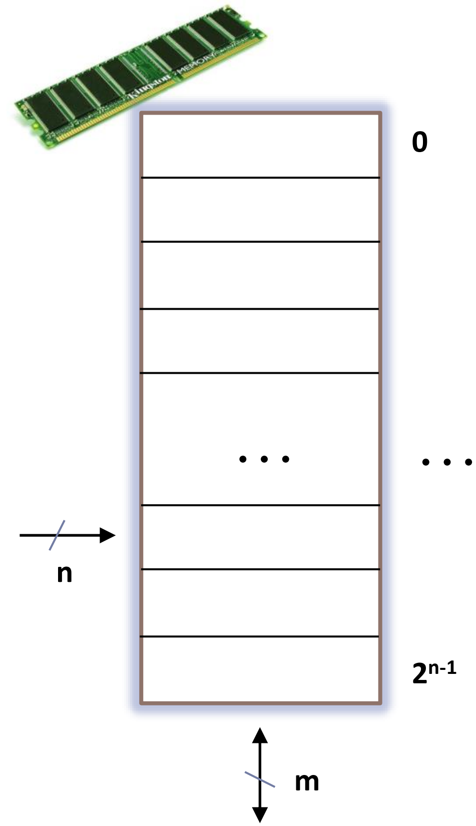
Address space vs. word size

Address Space:
Number of locations



Size of each location:
Number of bits per location

Address space vs. word size



Address Space:

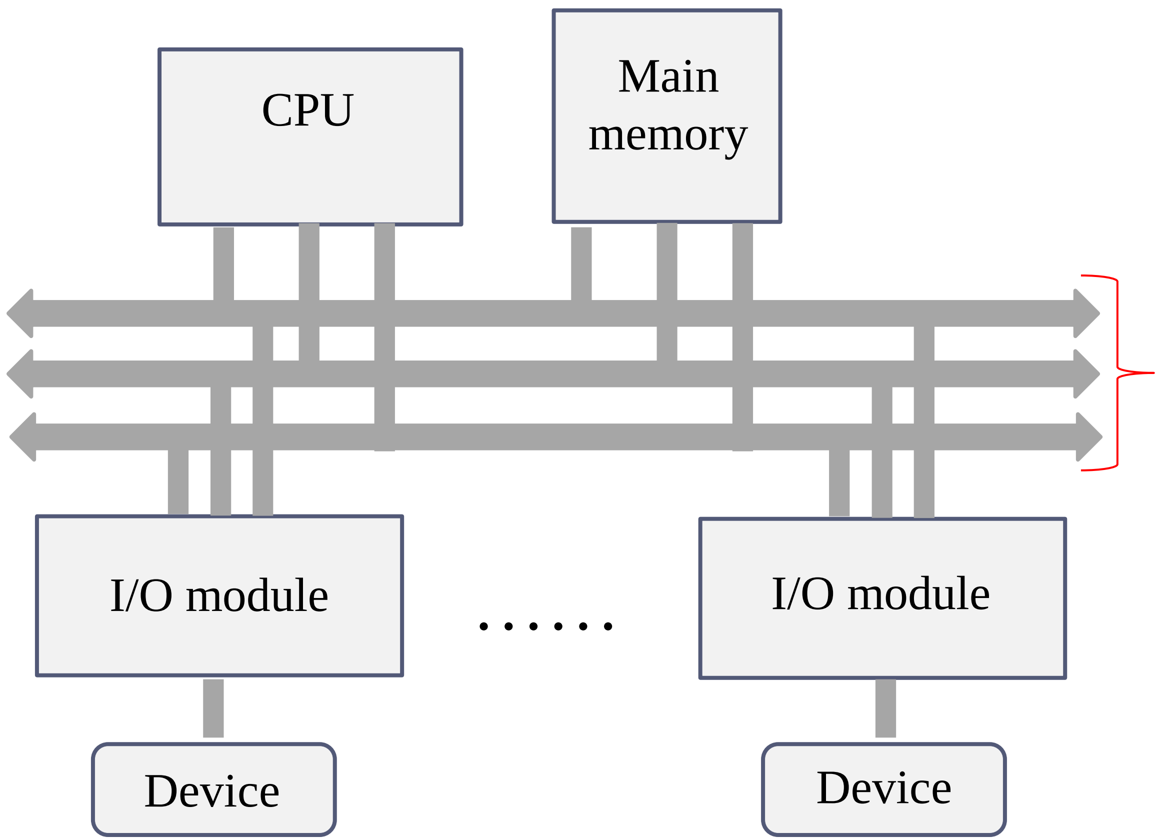
Number of locations

- Address with n bits
- 2^n locations

Size of each location:

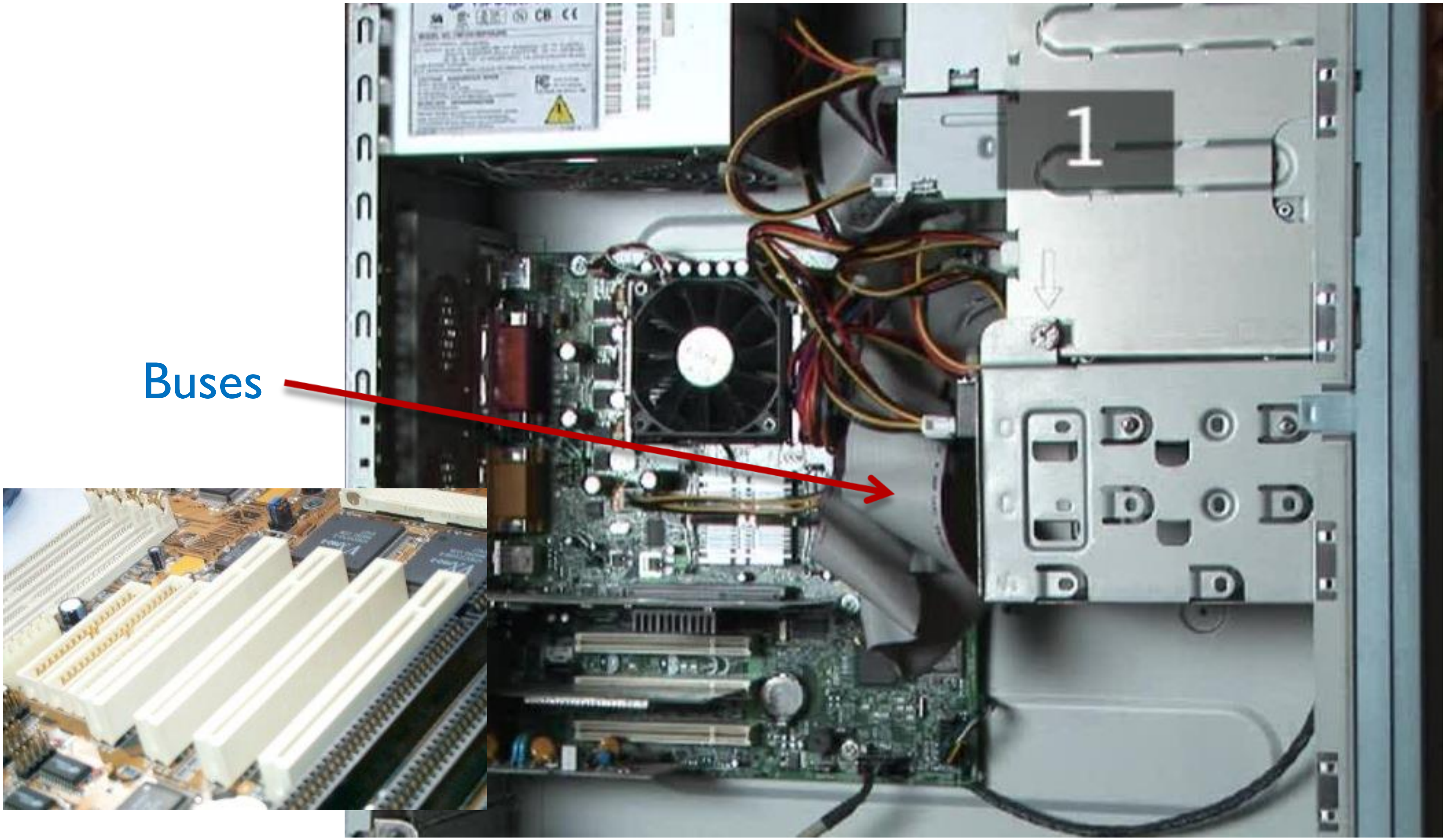
Number of bits per location

Von Neumann architecture (3/4)



- The different parts of the computer must communicate each other:
  - Buses

Example of buses

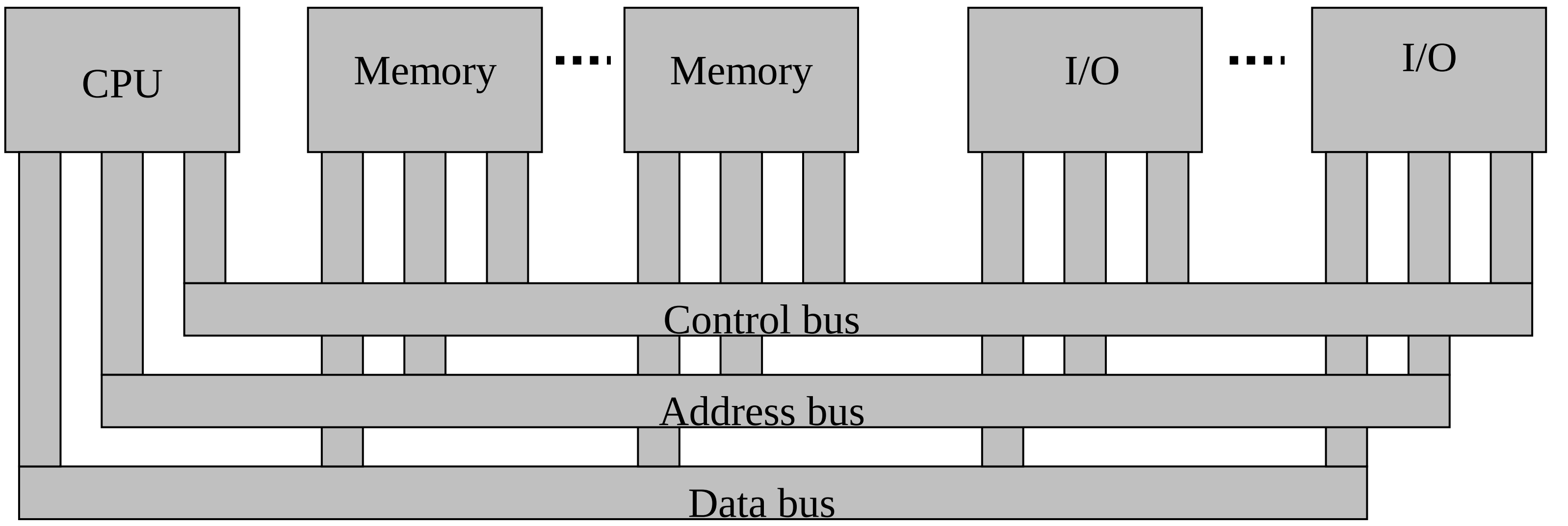


http://www.videojug.com/film/what-components-are-inside-my-computer

Buses

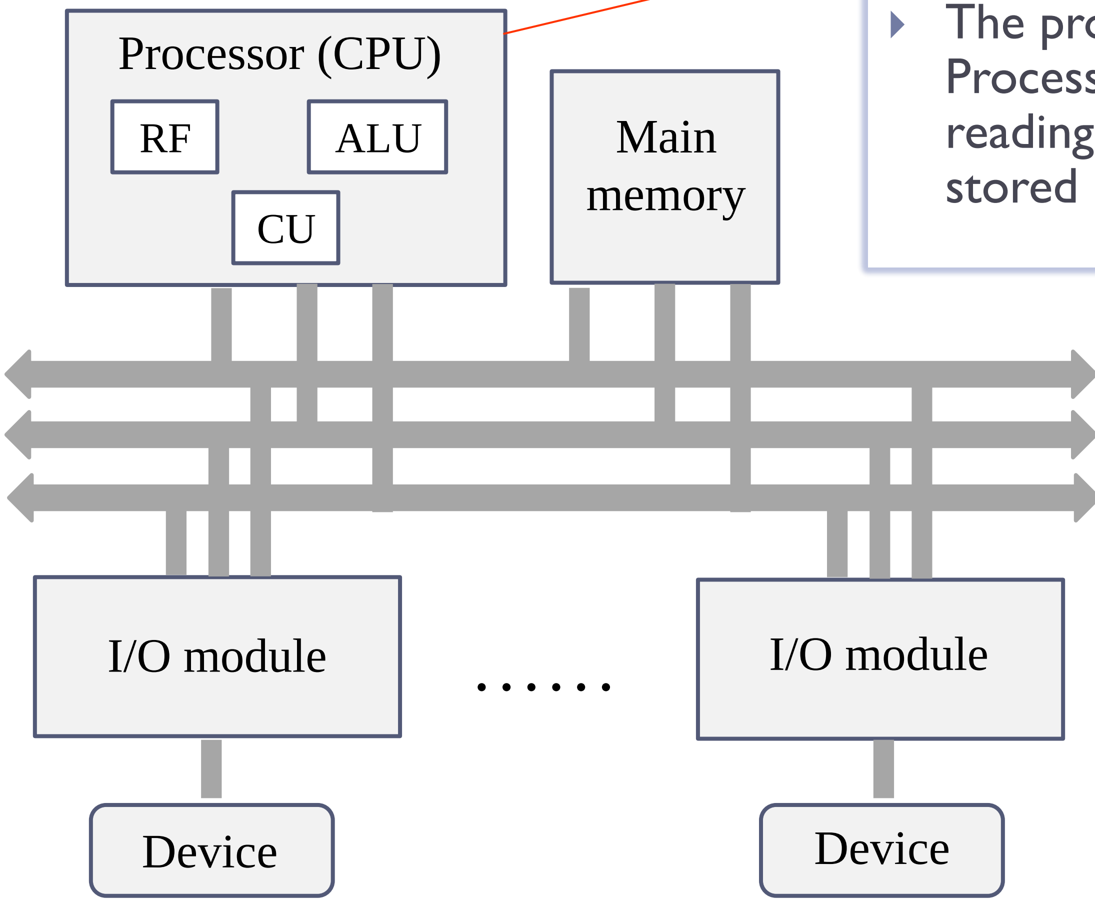
- A bus is a communication path between two or more elements (CPU, memory, ...) for the transmission of information between them.
- A bus usually consists of several communication lines, each transmitting one bit.
  - The width of the bus represents the size at which the computer works (example: a 32-bit computer has 32 buses).
- Three main types: data, address and control.

Bus interconnection diagram



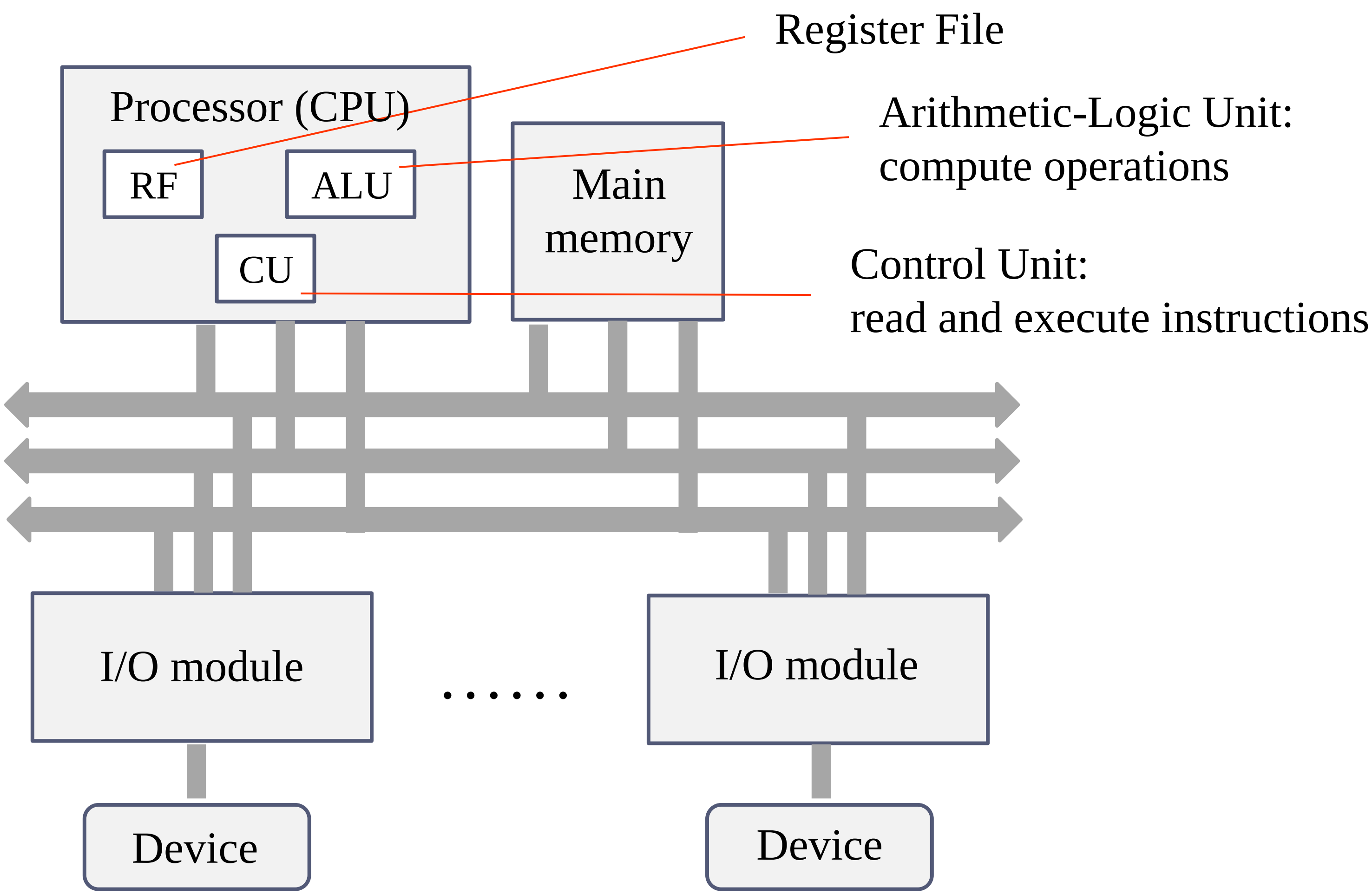
- Control bus: control and timing signals.
- Address bus: designates the source or destination of a data.
  - Its width determines the maximum memory capacity of the system.
- Data bus: data movement between components.

Von Neumann architecture (4/4)



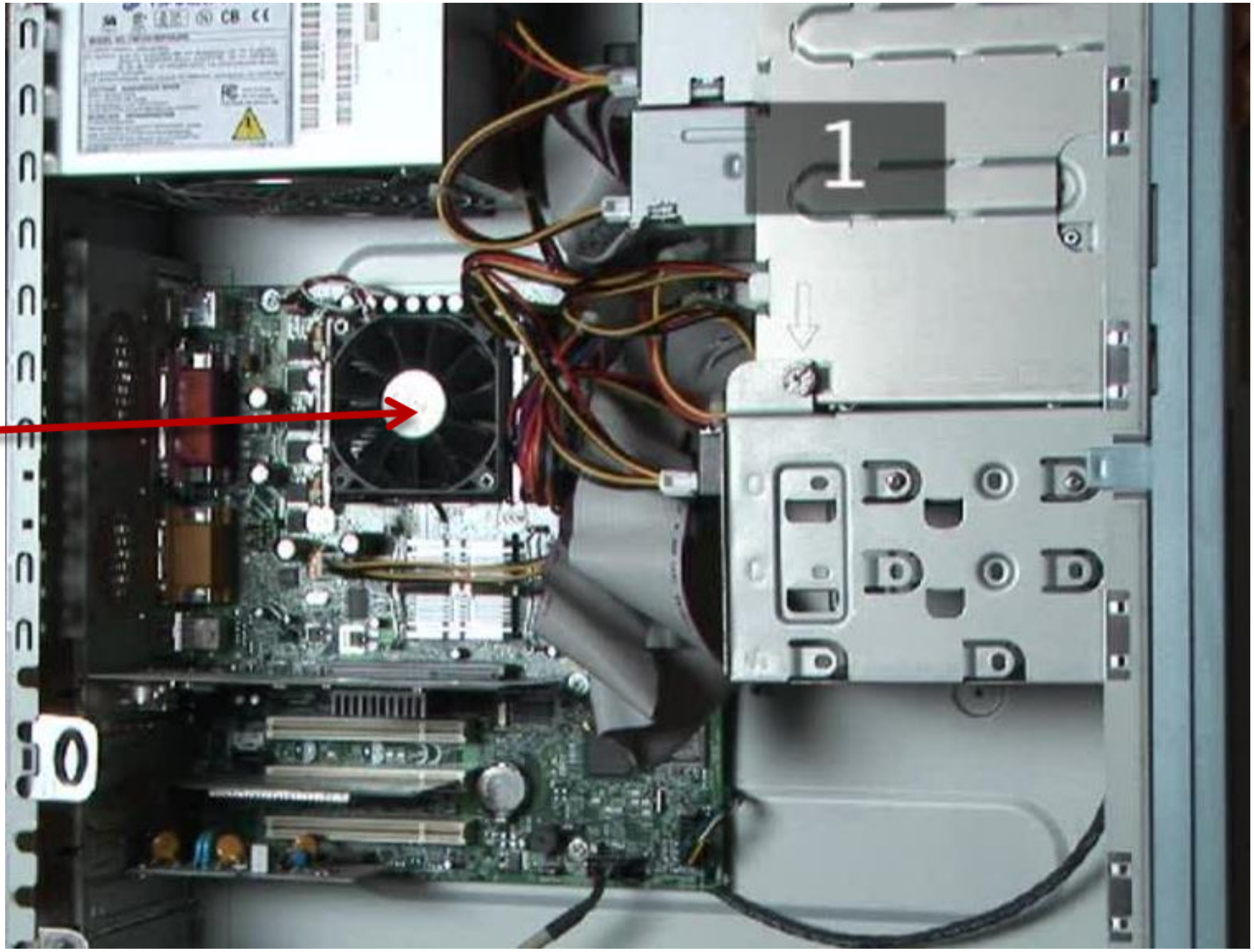
- The processor or CPU (Central Processing Unit) is responsible for reading and executing the instructions stored in the main memory.

Von Neumann architecture (4/4)



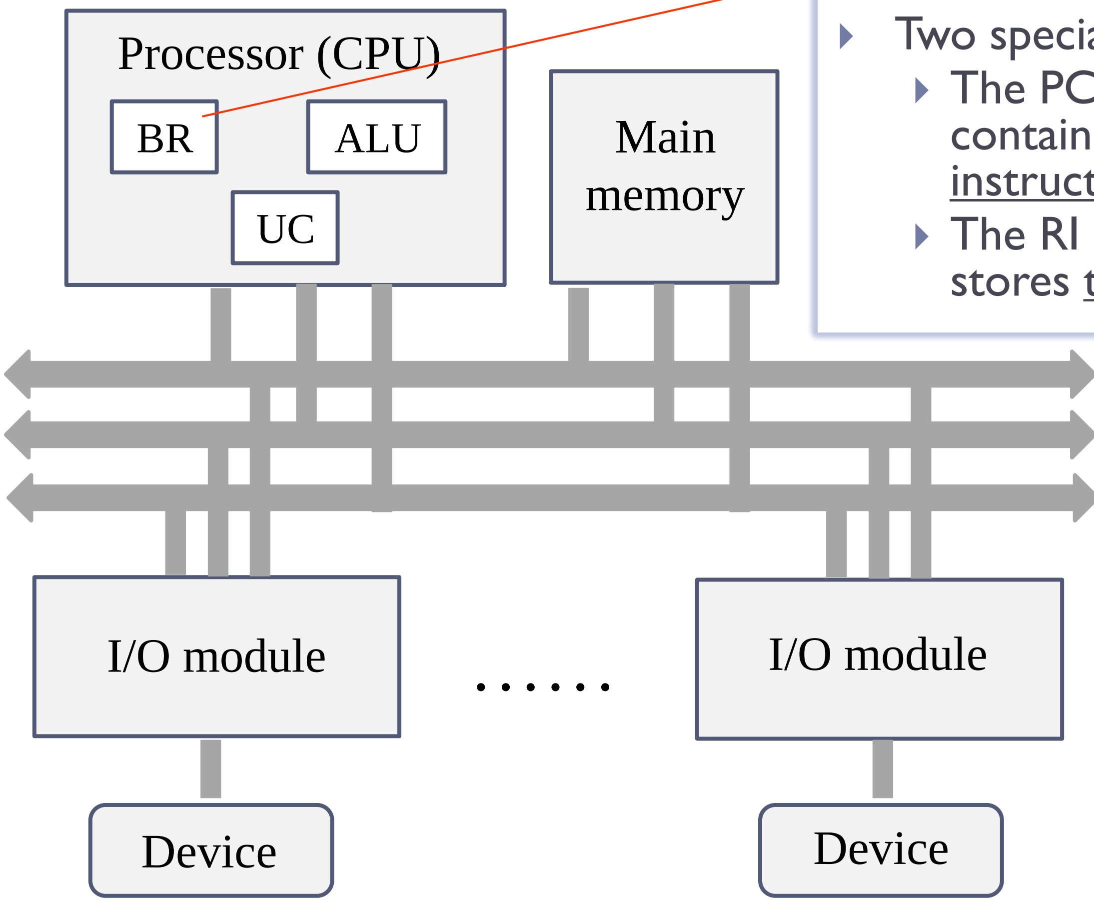
Example of CPU

CPU



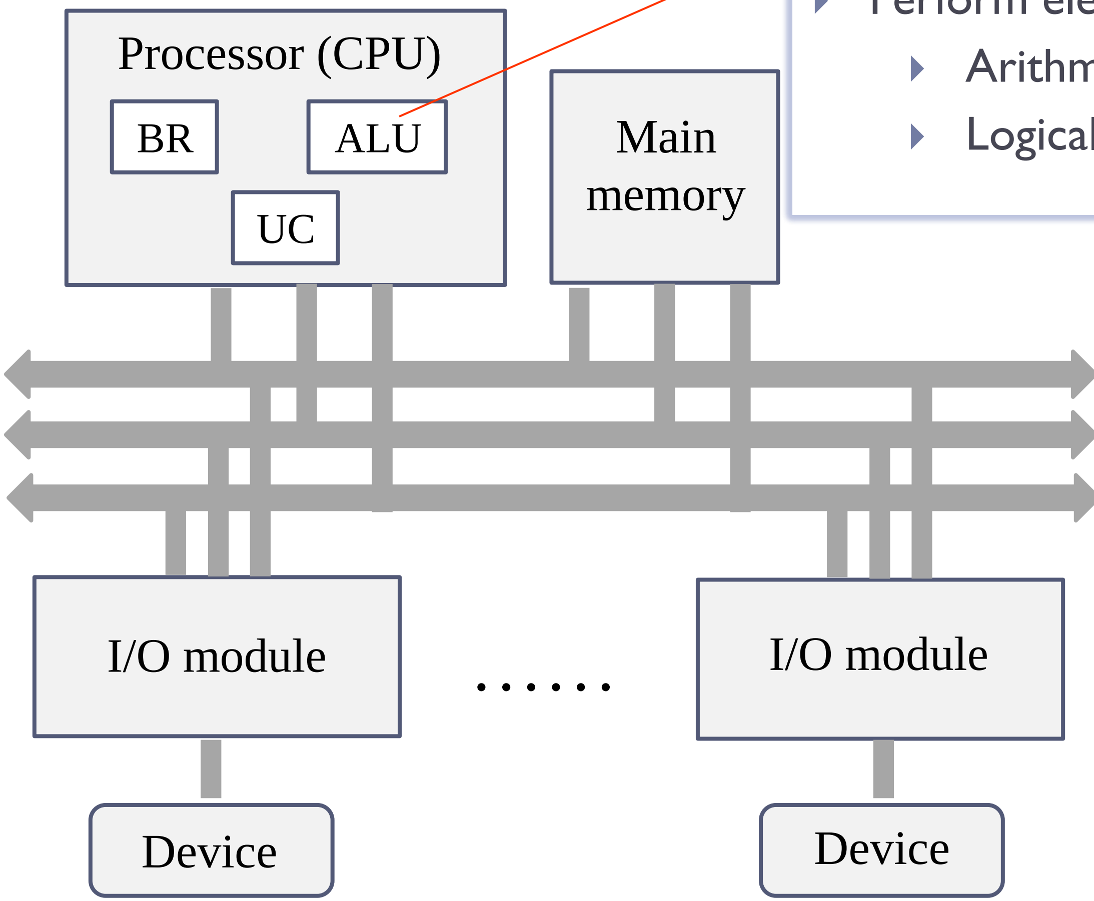
http://www.videojug.com/film/what-components-are-inside-my-computer

Processor: registers



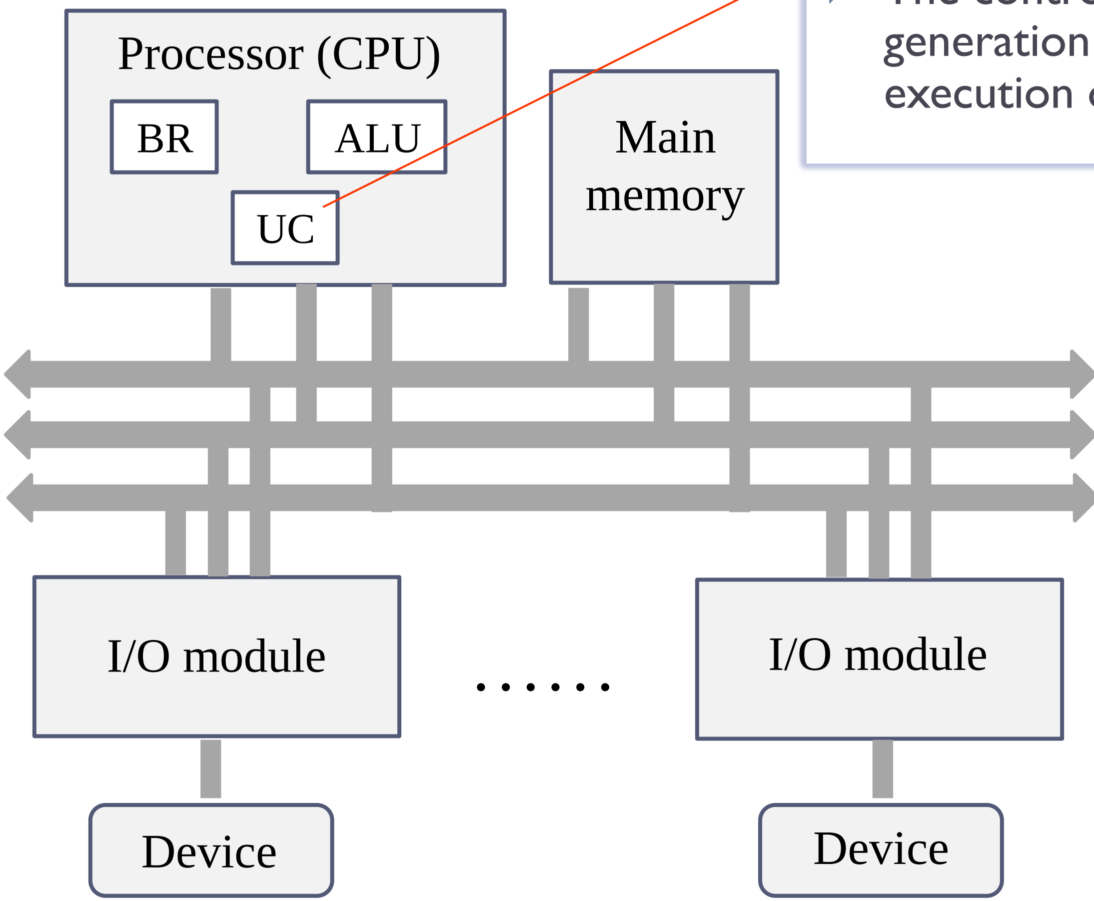
- Registers: store a sequence of bits.
- Two special registers:
  - The PC register (program counter) contains the address of the next instruction to be executed.
  - The RI register (instruction register) stores the instruction being executed.

Processor: arithmetic-logic unit



- Perform elementary operations on data:
  - Arithmetic
  - Logical

Processor: control unit



- The control unit is responsible for generation of control signals for the execution of instructions.

Content

Félix García-Carballeira, Alejandro Calderón Mateos

Program

- Consecutive sequence of machine instructions

```
00001001110001101010111101011000
10101111010110000000100111000110
11000110101011110101100000001001
01011000000010011100011010101111
```

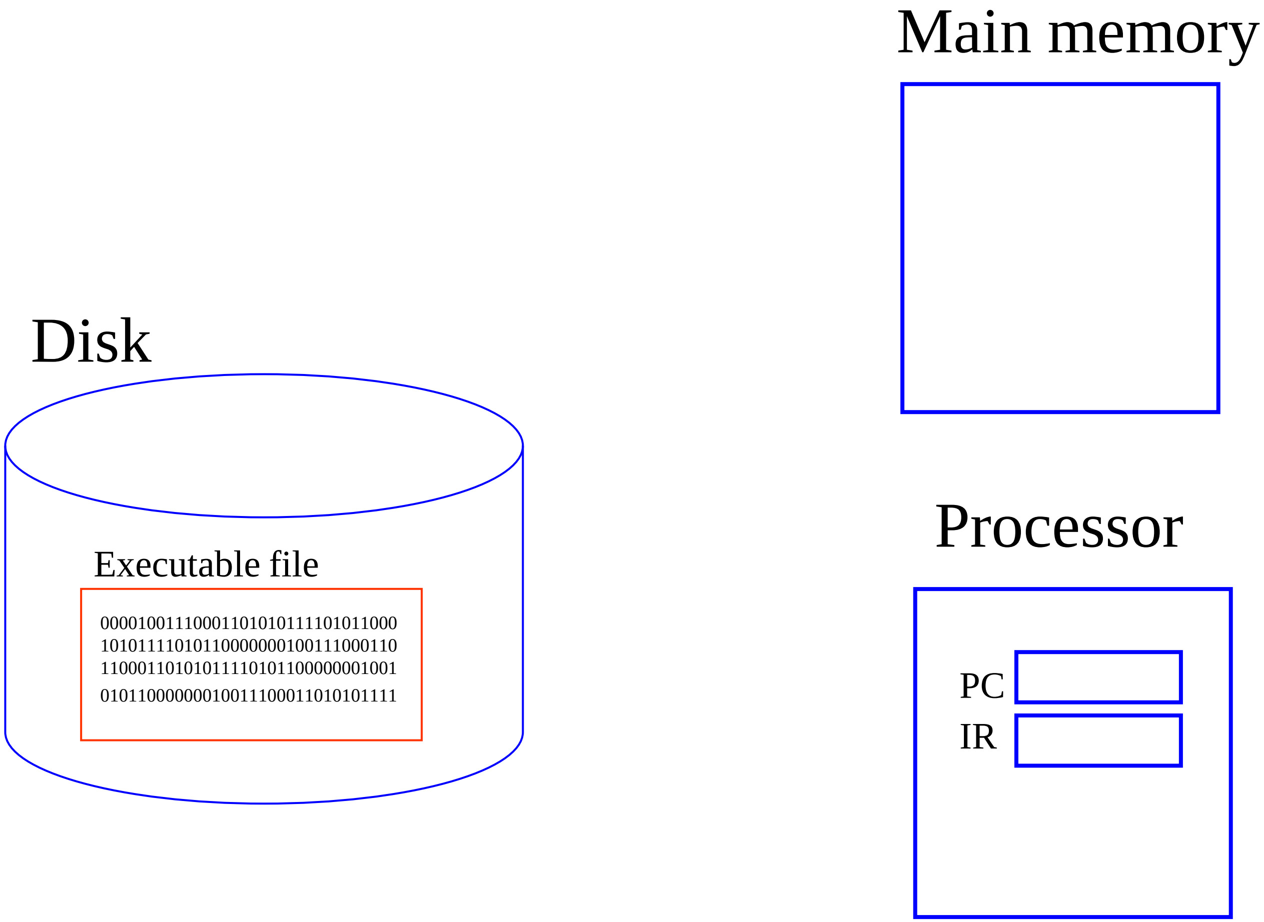
⋮

Program

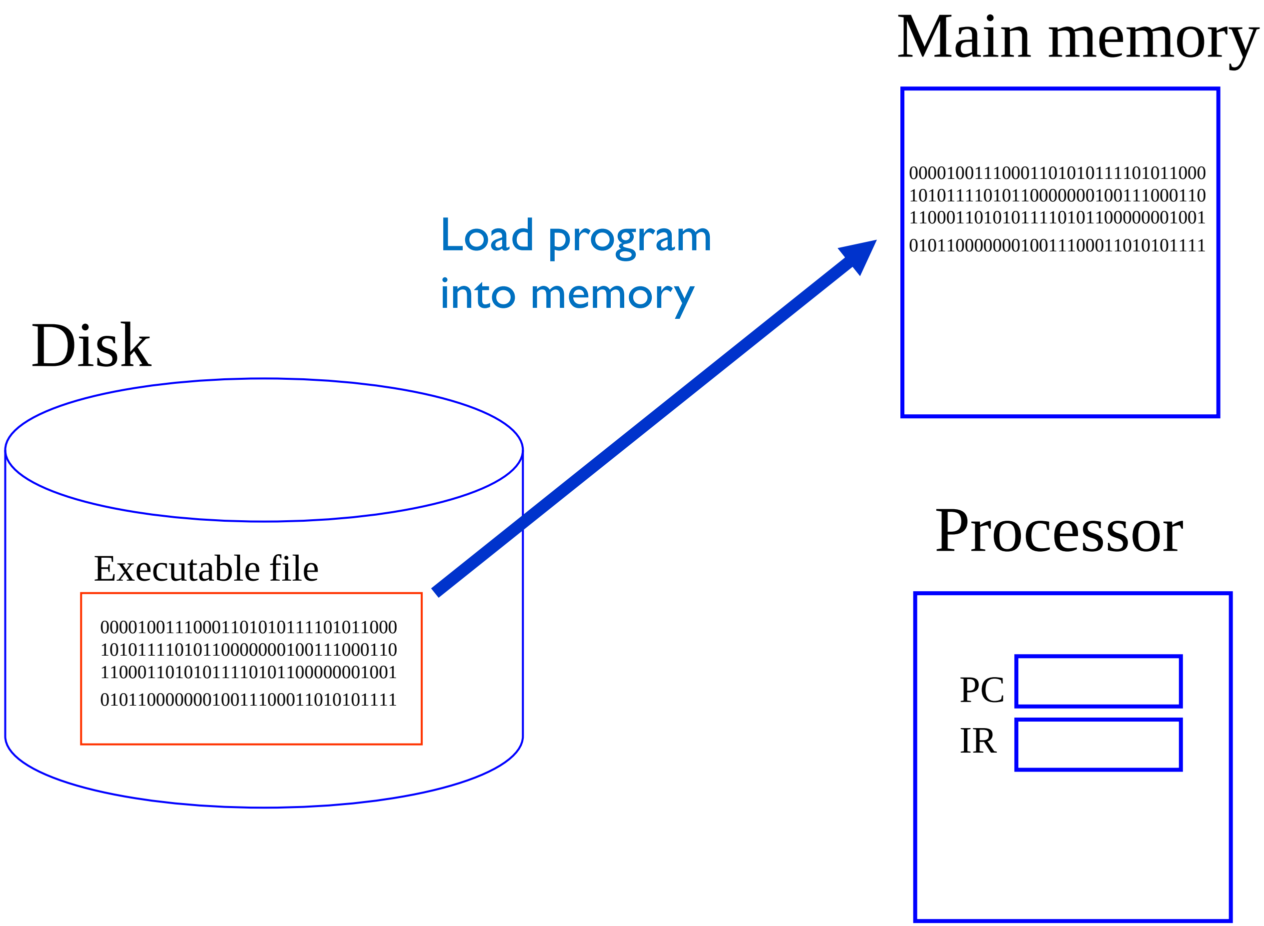
- Consecutive sequence of machine instructions
- Machine instruction: elementary operation that can be executed directly by a processor.
  - Binary coding

```
00001001110001101010111101011000        temp = v[k];
10101111010110000000100111000110        v[k] = v[k+1];
11000110101011110101100000001001        v[k+1] = temp;
01011000000010011100011010101111
              ⋮
```

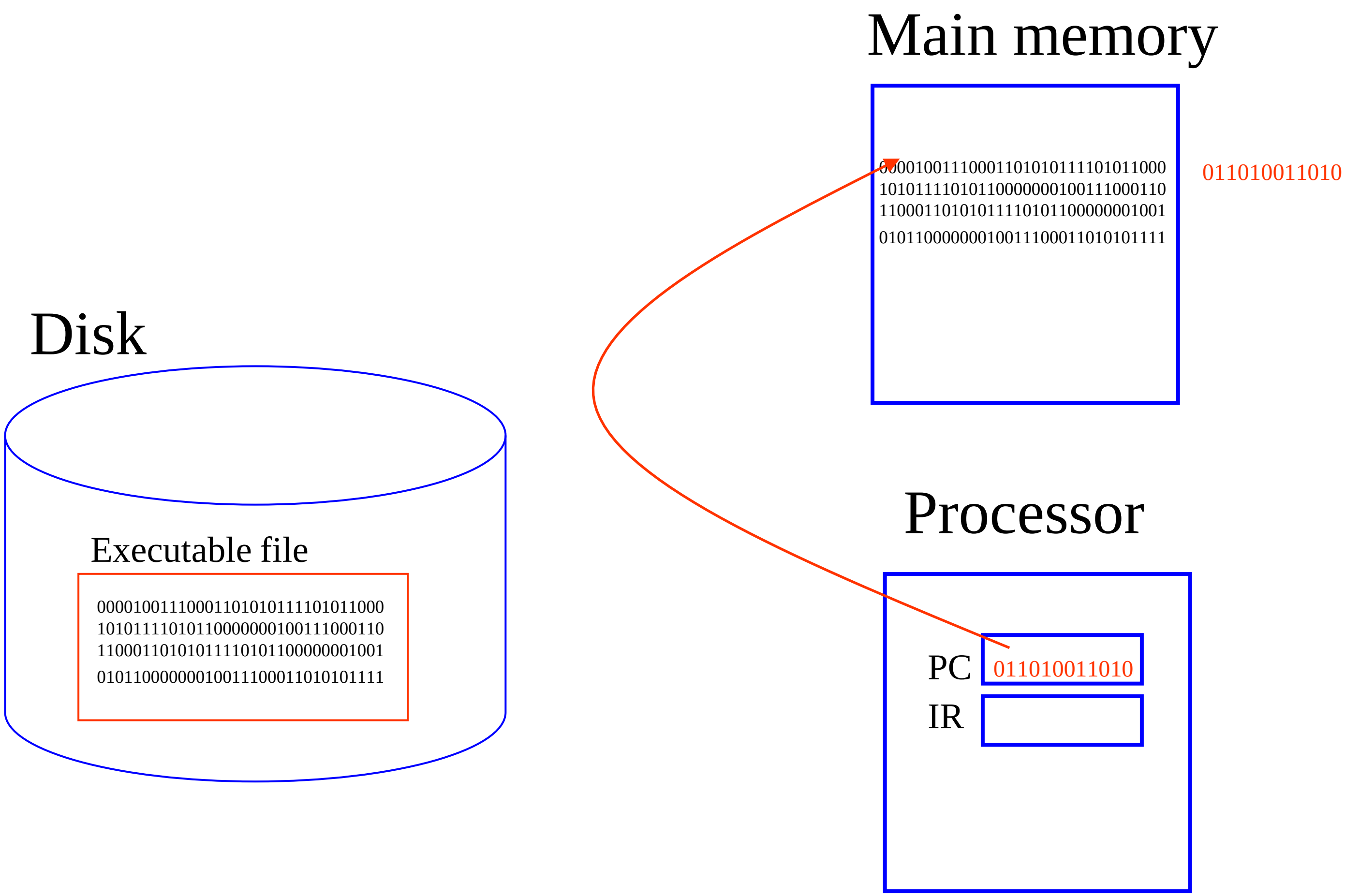
Program execution



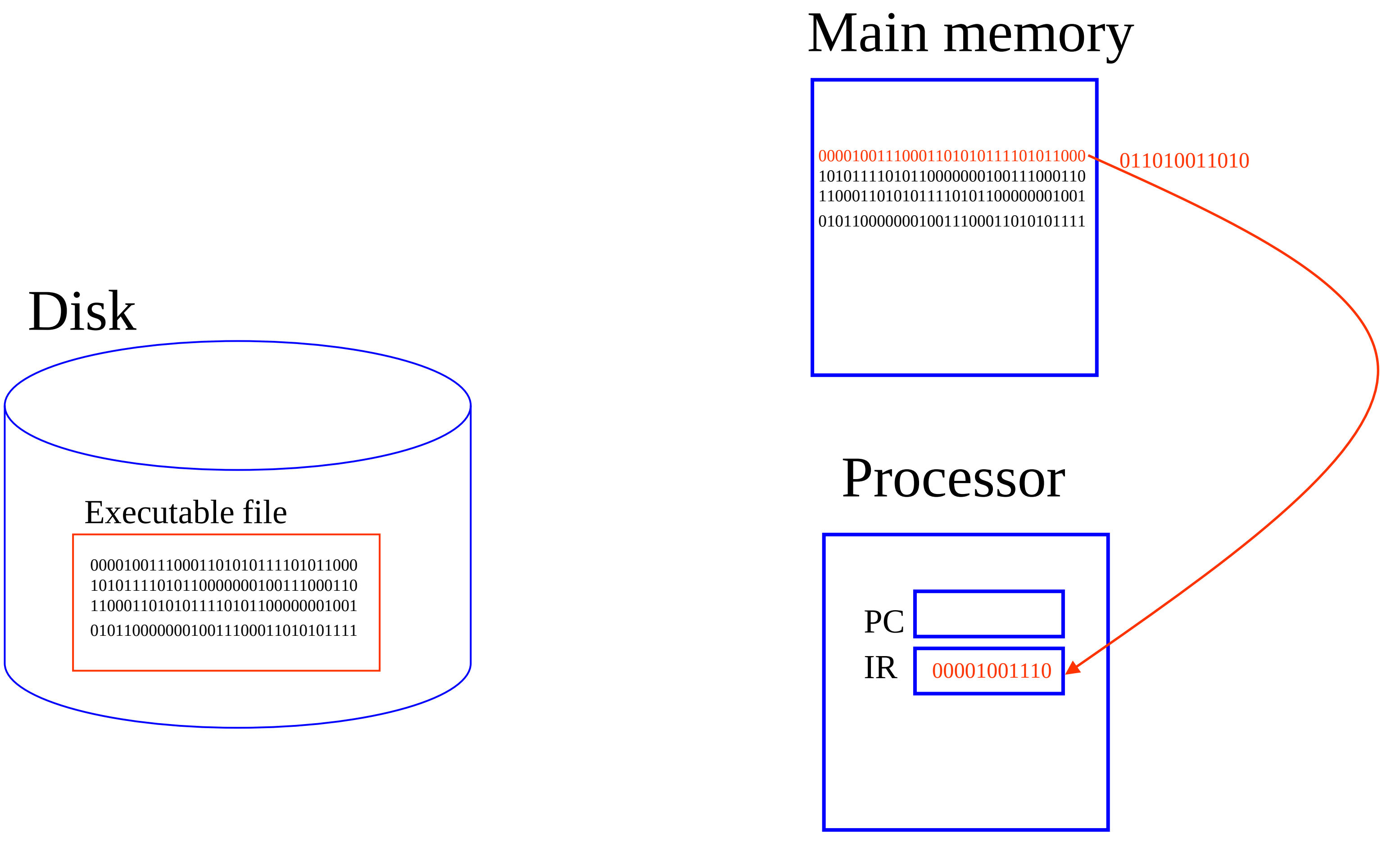
Program execution



Program execution

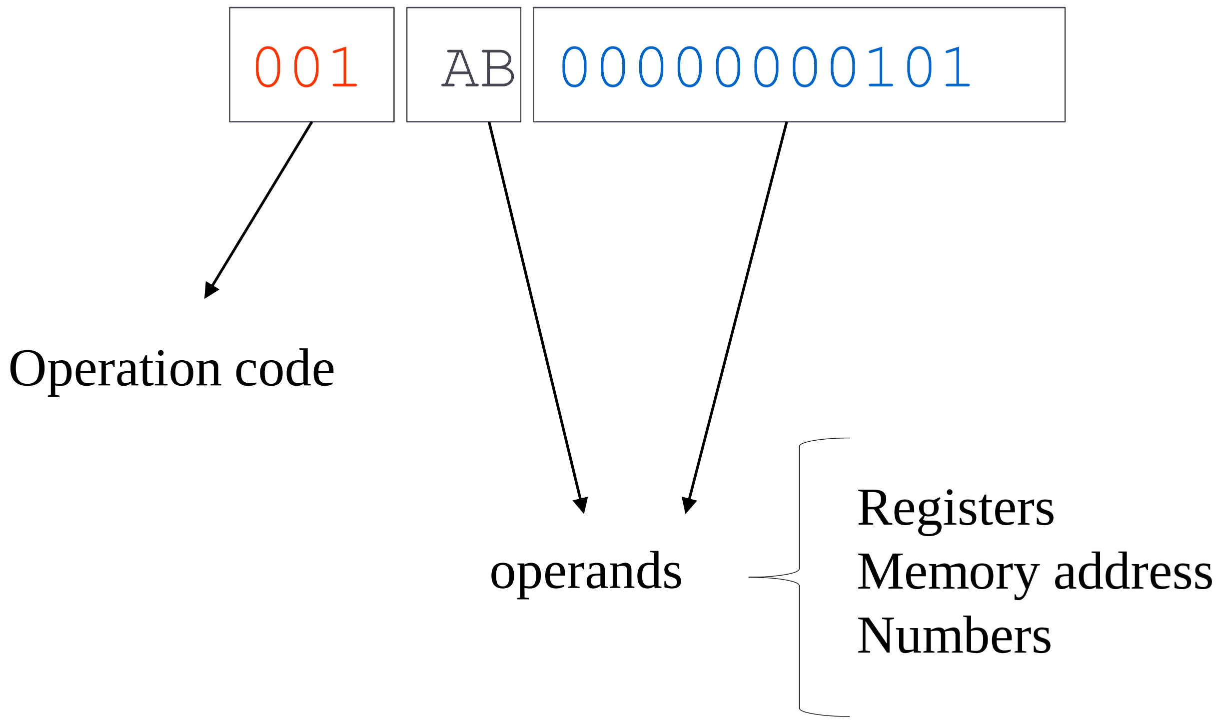


Program execution



Félix García-Carballeira, Alejandro Calderón Mateos

Format of a machine instruction



Example

- Instructions set with the following features:
  - Memory address location: 16 bits
  - Instruction size: 16 bits
  - Operation code: 3 bits
    - How many different instructions can execute this computer?
    - Number of general purpose registers: 4
    - Symbolic labels:
      - R0
      - R1
      - R2
      - R3
    - How many bits are needed to represent 4 registers?

Example

- Instructions set with the following features:
  - Memory address location: 16 bits
  - Instruction size: 16 bits
  - Operation code: 3 bits
    - How many different instructions can execute this computer? (8)
    - Number of general-purpose registers: 4
    - Symbolic labels:
      - R0 (00)
      - R1 (01)
      - R2 (10)
      - R3 (11)
    - How many bits are needed to represent 4 registers? (2 bits)

Example. Instructions set

Instruction	Description
000EFABCDXXXXXXX	Adds the register AB to CD and stores the result in EF
001AB00000000101	Stores in register AB the value 00000000101
010AB00000001001	Stores in register AB the value stored in the memory address 00000001001
011AB00000001001	Stores in the memory address 00000001001 the content of the register AB
1000000000001001	Jump to execute the instruction stored in the memory address 0000000001001
101ABCD000001001	Jump to execute the instruction sorted in memory address 000001001 if AB is equal to CD

With A,B, C, D, E, F = 0 o 1

Félix García-Carballeira, Alejandro Calderón Mateos

Examples

- Instruction that stores the value 5 in register 00
- Instruction that stores the value 7 in register 01
- Instructions that adds the register 00 to 01 and store the result in register 10
- Instruction that stores the above result in the memory address 1027 (in decimal)

Examples

Instruction	Description
000EFABCDXXXXXXX	Adds the register AB to CD and stores the result result in EF
001AB0000000101	Stores in register AB the value 0000000101
010AB0000001001	Stores in register AB the value stored in the memory address 0000001001
011AB0000001001	Stores in the memory address 0000001001 the content of the register AB
100000000001001	Jump to execute the instruction stored in the memory address 000000001001
101ABCD000001001	Jump to execute the instruction sorted in memory address 000001001 if AB is equal to CD

- Instruction that stores the value 5 in register 00
- Instruction that stores the value 7 in register 01
- Instructions that adds the register 00 to 01 and store the result in register 10
- Instruction that stores the above result in the memory address 1027 (in decimal)

Examples (solution)

- Instruction that stores the value 5 in register 00
  - 0010000000000101
- Instruction that stores the value 7 in register 01
  - 0010100000000111
- Instructions that adds the register 00 to 01 and store the result in register 10
  - 000100001XXXXXXX
- Instruction that stores the above result in the memory address 1027 (in decimal)
  - 0111010000000011

Example of program loaded in memory

Main memory

Address	Content
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Program generation and loading

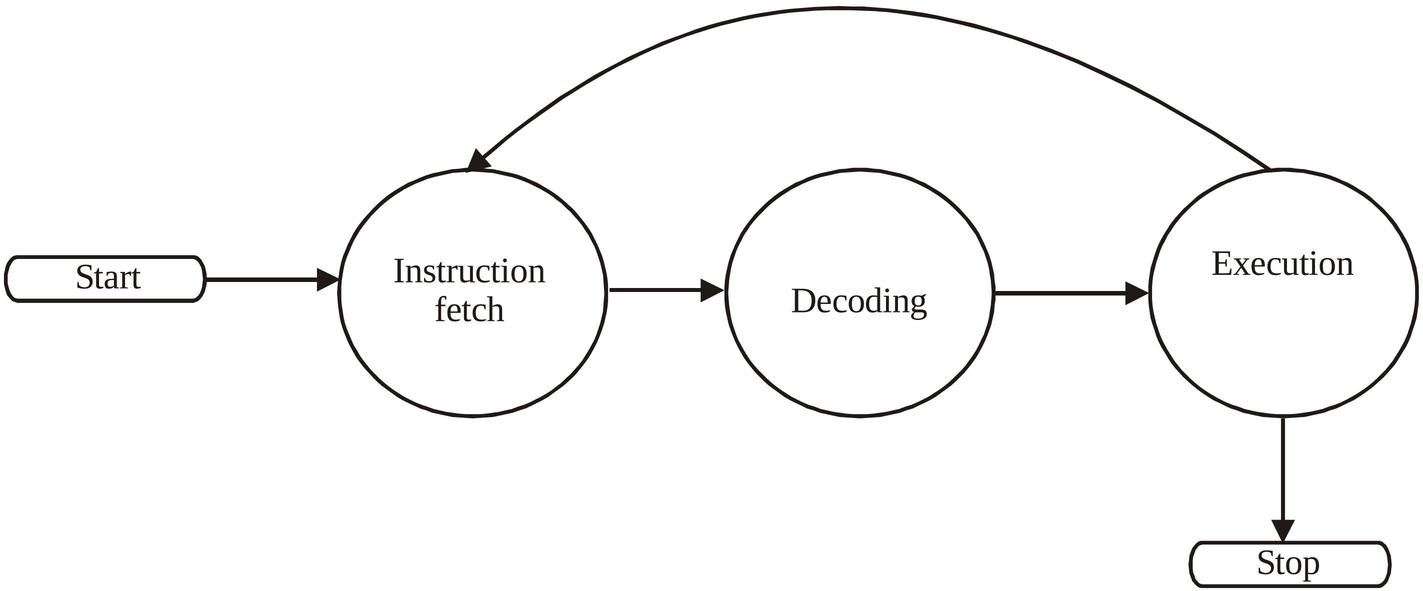
```
i=0;
s = 0;
while (i < 4)
{
   s = s + 1;
   i = i + 1;
}
```

```
      li R0, 0
      li R1, 4
      li R2, 1
      li R3, 0
lo1:  beq R0, R1, en1
      add R3, R3, R2
      add R0, R0, R2
      beq R0, R0, lo1
en1:  sw R3, 100000
```

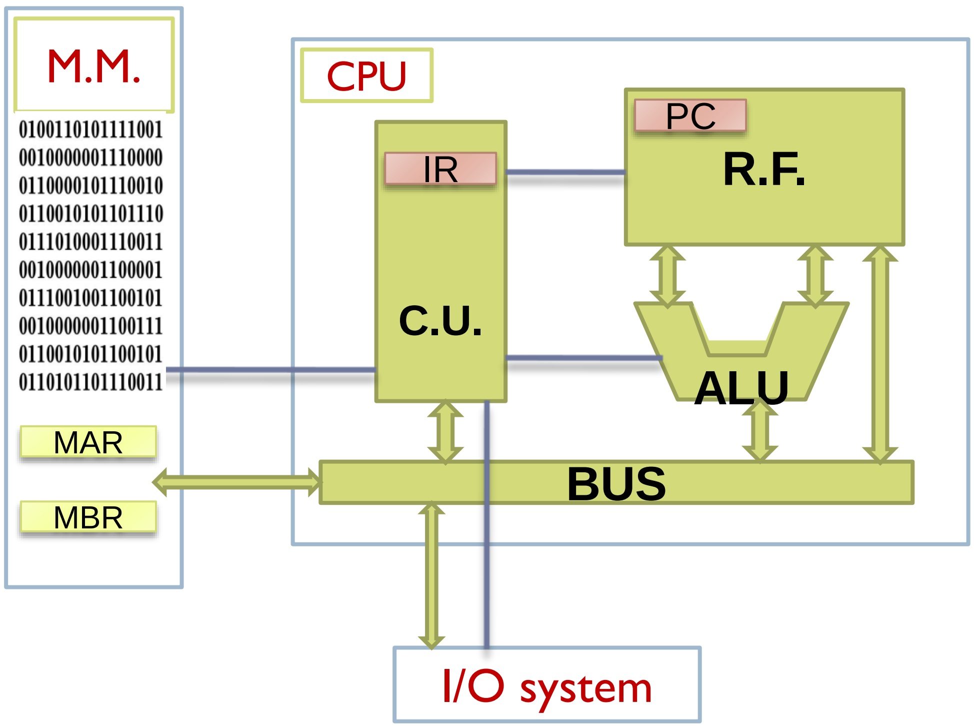
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Content

Steps to execute instructions

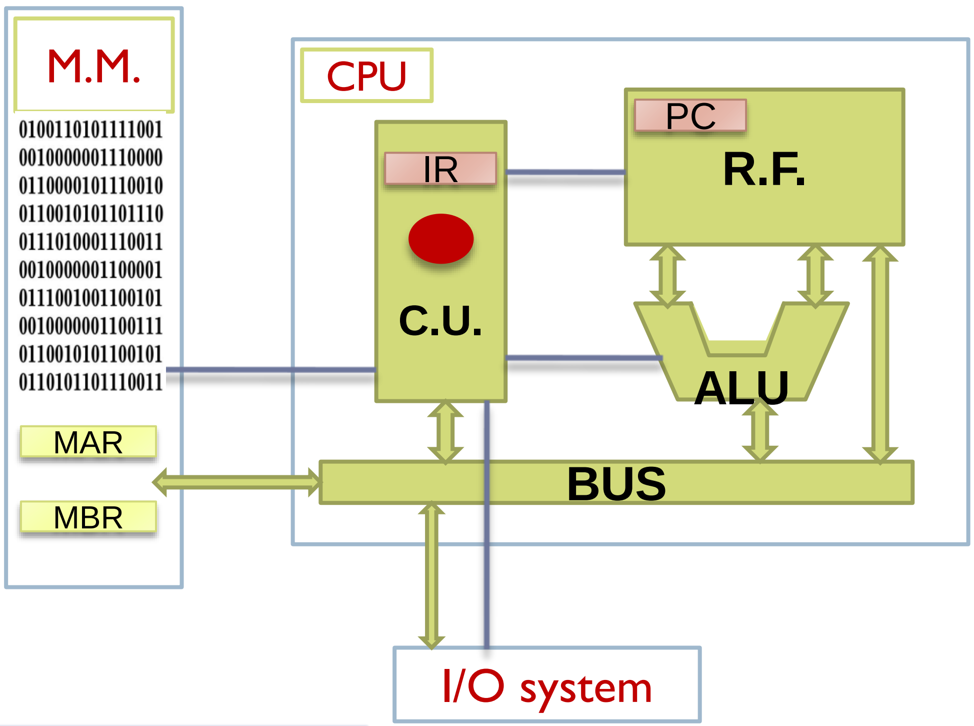


Execution stages: Instruction fetch



- Read from Main memory the instruction pointed by the PC
  - The PC contains the memory address where the instruction to be executed is stored.
  - The instruction read from M.M. is stored in IR.
- Increment PC
  - Increment the address stored in the PC so that it points to the next instruction
- Decode instruction
- Execute instruction

Execution stages: Decode

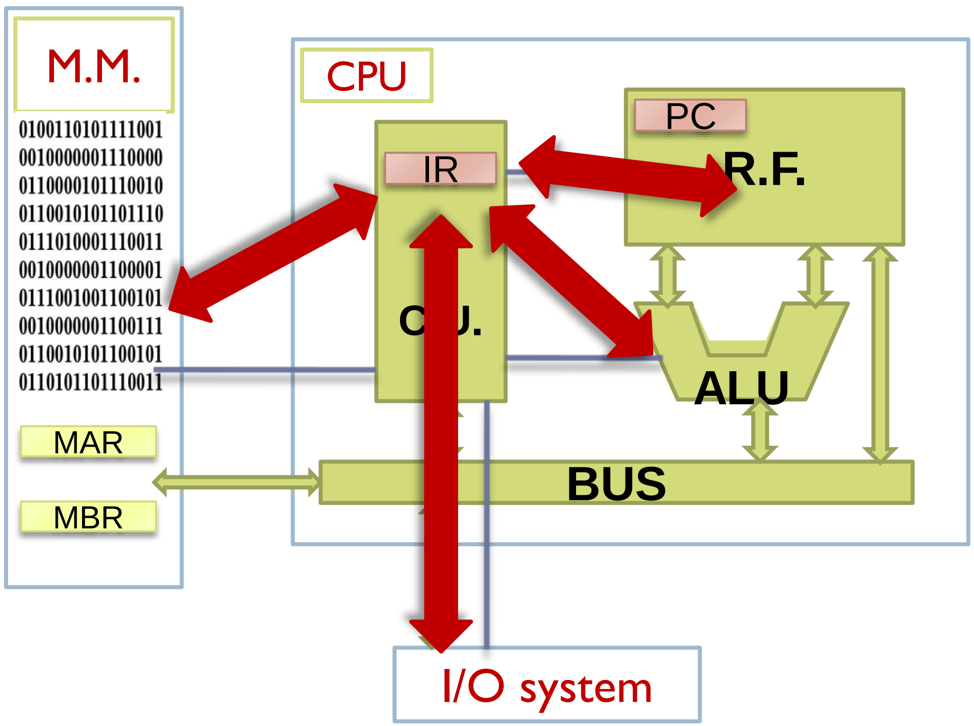


- Registers: store a sequence of bits.
- Two special registers:
  - The PC register (program counter) contains the address of the next instruction to be executed.
  - The RI register (instruction register) stores the instruction being executed.

- Read from Main memory the instruction pointed by the PC
  - The PC contains the memory address where the instruction to be executed is stored.
  - The instruction read from M.M. is stored in IR.
- Increment PC
  - Increment the address stored in the PC so that it points to the next instruction
- **Decode instruction**
- Execute instruction

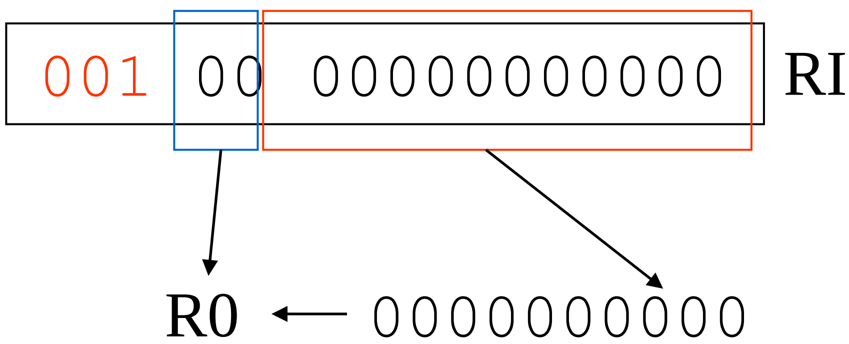
Félix García-Carballeira, Alejandro Calderón Mateos

Execution stages: Execution



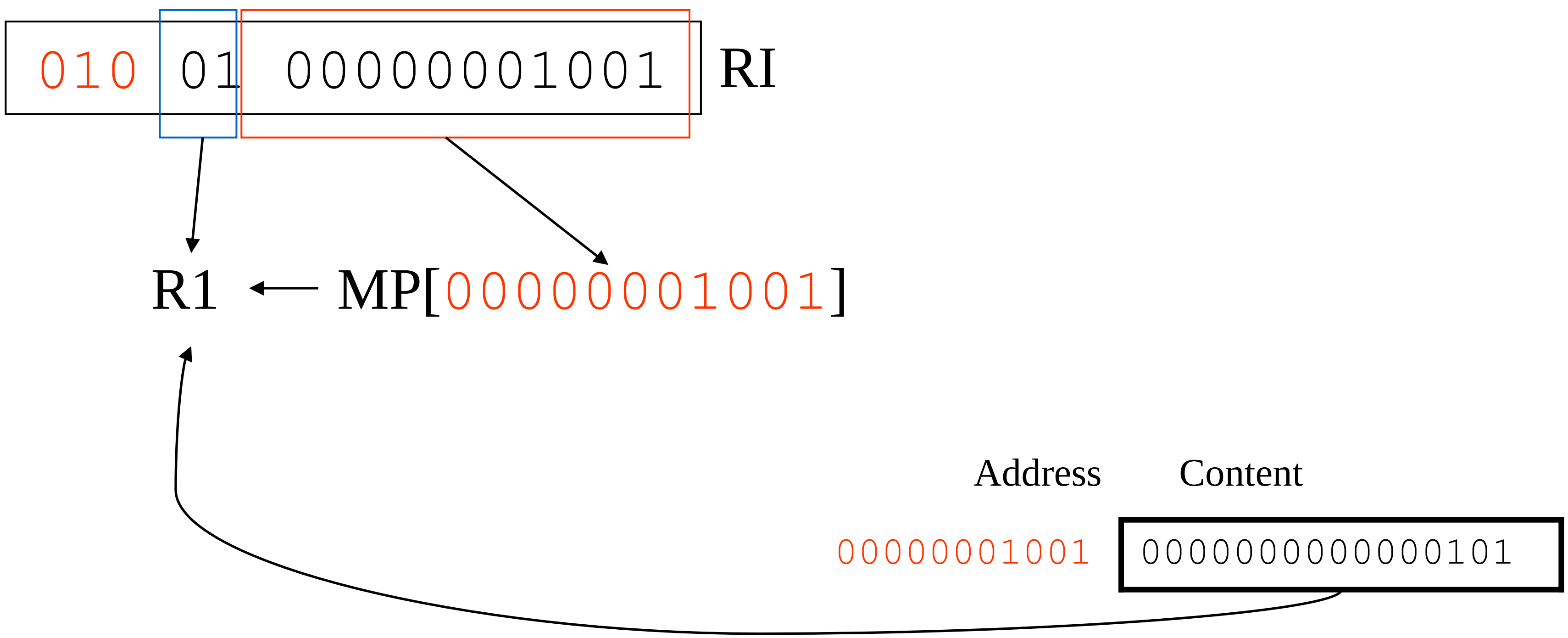
- Read from Main memory the instruction pointed by the PC
  - The PC contains the memory address where the instruction to be executed is stored.
  - The instruction read from M.M. is stored in IR.
- Increment PC
  - Increment the address stored in the PC so that it points to the next instruction
- Decode instruction
- **Execute instruction**

Example: instruction execution



Store in R0 the value 0

Example: instruction execution



Store in R1 the content of the memory address
00000001001

Example: instruction execution

100 0000000001000 RI

PC ← 0000000001000

Modify PC with the memory `0000000001000` to execute the instruction stored in the memory address `0000000001000`

Félix García-Carballeira, Alejandro Calderón Mateos

Example of program execution

Processor

PC	000100
RI	?
00	?
01	?
10	?
11	?

- Instruction fetch
- Point to the next instruction
- Instruction decoding
- Instruction execution
- Jump to fetch

Main memory

Dirección	Content
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Félix García-Carballeira, Alejandro Calderón Mateos

Example of program execution

Processor

PC	000100
RI	0010000000000000
00	?
01	?
10	?
11	?

- Instruction fetch
- Point to the next instruction
- Instruction decoding
- Instruction execution
- Jump to fetch

Main memory

Dirección	Content
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Félix García-Carballeira, Alejandro Calderón Mateos

Example of program execution

Processor

PC	000101
RI	0010000000000000
00	?
01	?
10	?
11	?

Main memory

Dirección	Content
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

- Instruction fetch
- Point to the next instruction
  - PC ⟵ PC + "1"
- Instruction decoding
- Instruction execution
- Jump to fetch

Example of program execution

Processor

PC	000101
RI	0010000000000000
00	?
01	?
10	?
11	?

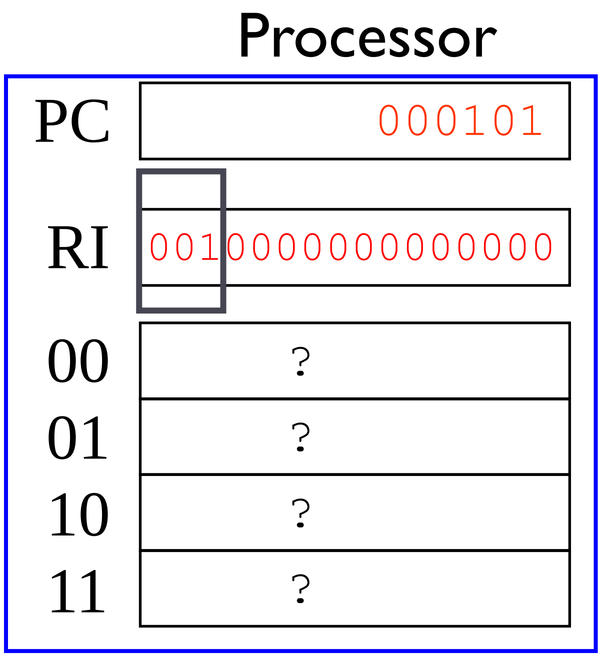
- Instruction fetch
- Point to the next instruction
- Instruction decoding
- Instruction execution
- Jump to fetch

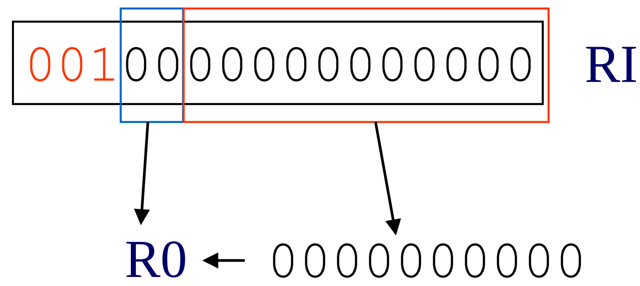
Main memory

Dirección	Content
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Félix García-Carballeira, Alejandro Calderón Mateos

Example of program execution

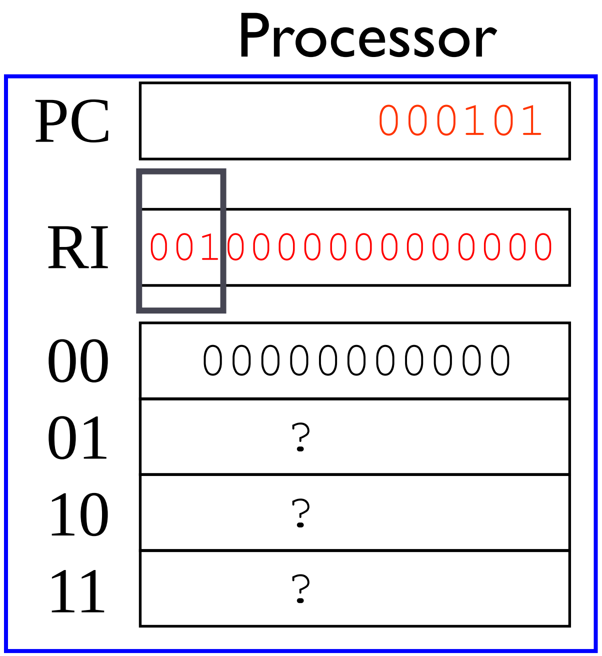


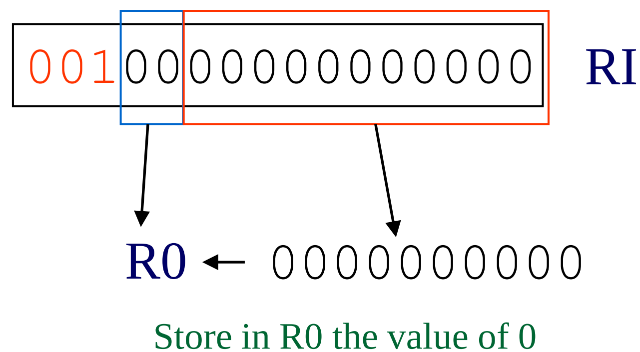


Store in R0 the value of 0

- Instruction fetch
- Point to the next instruction
- Instruction decoding
- Instruction execution
- Jump to fetch

Example of program execution





- Instruction fetch
- Point to the next instruction
- Instruction decoding
- Instruction execution
- Jump to fetch

Example of program execution

Processor

PC	000101
RI	0010000000000000
00	000000000000
01	?
10	?
11	?

- Instruction fetch
- Point to the next instruction
- Instruction decoding
- Instruction execution
- Jump to fetch

Main memory

Dirección	Content
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Félix García-Carballeira, Alejandro Calderón Mateos

Example of program execution

Processor

PC	000101
RI	0010100000000100
00	000000000000
01	?
10	?
11	?

- Instruction fetch
- Point to the next instruction
- Instruction decoding
- Instruction execution
- Jump to fetch

Main memory

Dirección	Content
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Example of program execution

Processor

PC	000110
RI	0010100000000100
00	000000000000
01	?
10	?
11	?

- Instruction fetch
- Point to the next instruction
  - PC ⟵ PC + "1"
- Instruction decoding
- Instruction execution
- Jump to fetch

Main memory

Dirección	Content
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Félix García-Carballeira, Alejandro Calderón Mateos

Example of program execution

Processor

PC	000110
RI	001010000000100
00	00000000000
01	?
10	?
11	?

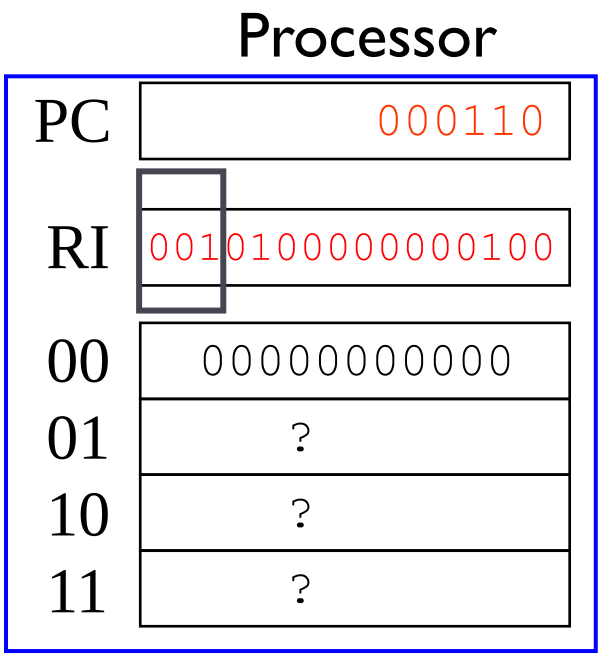
- Instruction fetch
- Point to the next instruction
- Instruction decoding
- Instruction execution
- Jump to fetch

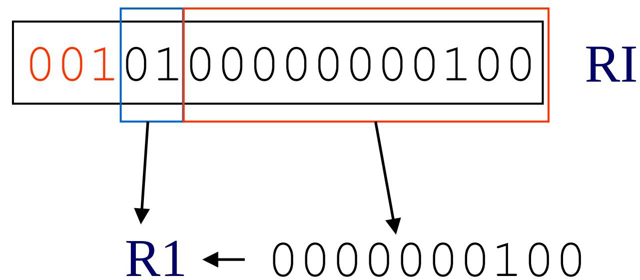
Main memory

Dirección	Content
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Félix García-Carballeira, Alejandro Calderón Mateos

Example of program execution

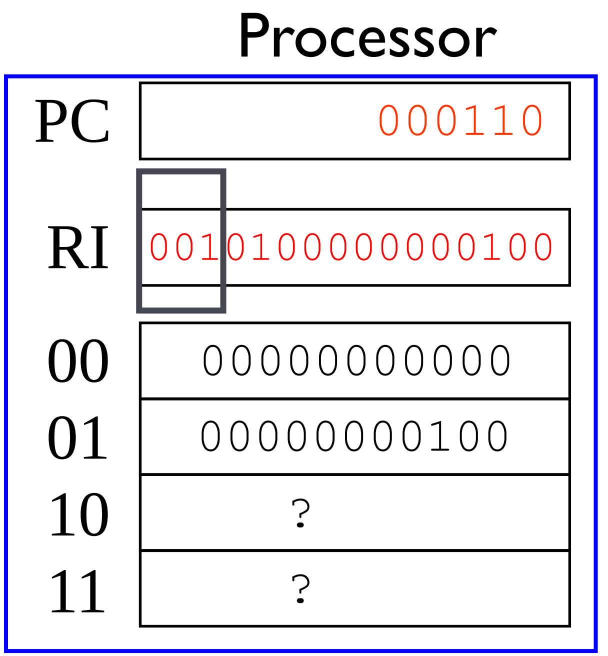


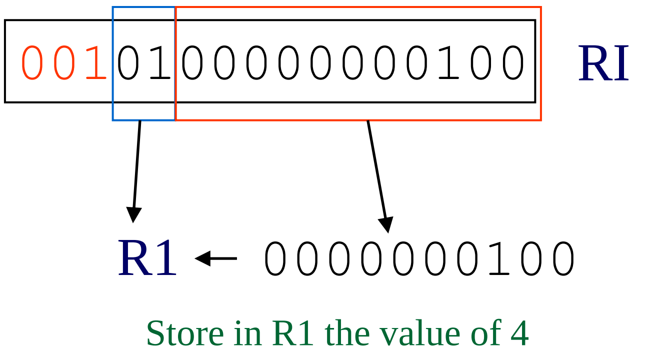


Store in R1 the value of 4

- Instruction fetch
- Point to the next instruction
- Instruction decoding
- Instruction execution
- Jump to fetch

Example of program execution





- Instruction fetch
- Point to the next instruction
- Instruction decoding
- Instruction execution
- Jump to fetch

Example of program execution

Processor

PC	000110
RI	0010100000000100
00	00000000000
01	00000000100
10	?
11	?

- Instruction fetch
- Point to the next instruction
- Instruction decoding
- Instruction execution
- Jump to fetch

Main memory

Dirección	Content
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

Félix García-Carballeira, Alejandro Calderón Mateos

Example of program execution

Processor

PC	000110
RI	0010100000000100
00	00000000000
01	00000000100
10	?
11	?

Main memory

Dirección	Content
000100	0010000000000000
000101	0010100000000100
000110	0011000000000001
000111	0011100000000000
001000	1010001000001100
001001	0001111100000000
001010	0000000100000000
001011	1000000000001000
001100	0111100000100000

- And so on…

Félix García-Carballeira, Alejandro Calderón Mateos

Algorithm of the previous program

```
i=0;
s = 0;
while (i < 4)
{
  s = s + 1;
  i = i + 1;
}
```

The program stores in the memory address `0000010000` the value : 1 + 1 + 1 + 1

Félix García-Carballeira, Alejandro Calderón Mateos

Assembly languaje

- Uses symbolic and mnemonic codes to represent the machine instructions executed by a computer.

Assembly instructions	Machine instructions
`li R0, 0`	`0010000000000000`
`li R1, 4`	`0010100000000100`
`li R2, 1`	`0011000000000001`
`li R3, 0`	`0011100000000000`
`loop1: beq R0, R1, end1`	`1010001000001100`
`add R3, R3, R2`	`0001111100000000`
`add R0, R0, R2`	`0000000100000000`
`beq R0, R0, loop1`	`1000000000001000`
`end1: sw R3, 100000`	`0111100000100000`

Content

Characteristic parameters of a computer

- Regarding its architecture
  - Word and word size
- Storage
  - Size
  - Storage units
- Communications
  - Bandwidth
  - Latency
- Computer power
  - MIPS
  - MFLOPS

Word Width

- Number of bits handled in parallel inside the computer.
  - Influences the size of the registers (BR).
  - Therefore, also in the ALU
    - Two 32-bit sums are not the same as one 64-bit sum.
  - Therefore also on the width of the buses.
    - A 32-bit address bus 'only' addresses 4 GB
- A computer with a word width of n bits:
  - n-bit memory addresses
  - Registers store n bits
  - n-bit integers
- Typical sizes → 32 bits, 64 bits

Privileged sizes

- Word
  - Information handled in parallel inside the processor.
  - Typically 32/64 bits
- Half word
- Double word

- Octet, character or byte
  - Representation of a character
  - Typically 8 bits

Exercise

- Consider a hypothetical computer with a word width of 20 bits with 60 registers that addresses memory by bytes.

  Please answer the following questions:

  a) How many bits are used for memory addresses?
  b) What is the size of the registers?
  c) How many bits are stored in each memory location?
  d) How many memory locations can be addressed? Express the result in KB.
  e) How many bits are needed to identify the registers?

Memory size

- Main memory size (RAM)
  - Usual capacity: 512MB – 4 GB
  - Expressed in bytes
- Auxiliary memory size (storage capacity of secondary memory device)
  - Paper: a few bytes
  - Diskette: 1,44 KB
  - CD-ROM: 600 MB
  - DVD: 4.7GB
  - Blu-ray: 50 GB
  - Hard disk: 10 GB – 2 TB

Size units

- Usually in bytes:

Name	Abr	Factor	IS
Kilo	K	2^{10} = 1,024	10^{3} = 1,000
Mega	M	2^{20} = 1,048,576	10^{6} = 1,000,000
Giga	G	2^{30} = 1,073,741,824	10^{9} = 1,000,000,000
Tera	T	2^{40} = 1,099,511,627,776	10^{12} = 1,000,000,000,000
Peta	P	2^{50} = 1,125,899,906,842,624	10^{15} = 1,000,000,000,000,000
Exa	E	2^{60} = 1,152,921,504,606,846,976	10^{18} = 1,000,000,000,000,000,000
Zetta	Z	2^{70} = 1,180,591,620,717,411,303,424	10^{21} = 1,000,000,000,000,000,000,000
Yotta	Y	2^{80} = 1,208,925,819,614,629,174,706,176	10^{24} = 1,000,000,000,000,000,000,000,000

Units for size

- In communication, powers of 10 are used:
  - 1 Kb = 1000 bits
  - 1 KB = 1000 bytes

- In storage, some manufacturers do not use powers of two, but powers of 10:
  - kilobyte 1 KB = 1.000 bytes 10^3 bytes
  - megabyte 1 MB = 1.000 KB 10^6 bytes
  - gigabyte 1 GB = 1.000 MB 10^9 bytes
  - terabyte 1 TB = 1.000 GB 10^{12} bytes
  - …..

Félix García-Carballeira, Alejandro Calderón Mateos

Exercice

- How many bytes does a 200 GB hard disk have?

- How many bytes per second does my 20 Mb ADSL transmit?

Exercice (solution)

- How many bytes does a 200 GB hard disk have?
  - 200 GB = 200 * 10^9 bytes = 186.26 Gigabytes

- How many bytes per second does my 20 Mb ADSL transmit?
  - B $\longrightarrow$ Byte
  - b $\longrightarrow$ bit.
  - 20 Mb = 20 * 10^6 bits = 20 * 10^6 / 8 bytes = 2.38 Megabytes per second

Bandwidth

- Several interpretations:
  - Information throughput transmitted by a bus.
  - Information throughput transmitted by an I/O unit.
  - Information throughput that can be processed by a unit.
  - Number of bits transferred per unit of time.
- Unit:
  - Kb/s (Kilobits per second, not to be confused with KB/s)
  - Mb/s (Megabits per second, <u>not</u> megabytes per second)

Latency

- Various interpretations:
  - Elapsed time in issuing a request in a reliable messaging system.
  - Elapsed time between the issuance of a request and the performance of the associated action.
  - Elapsed time between the issuance of a request and the receipt of the response.
- Unit:
  - s. (seconds)

Computing power

- Measurement of computing power.

- Factors involved:
  - Instruction set.
  - CPU clock (1 GHz vs 2 GHz vs 4 GHz...)
  - Number of 'cores' (quadcore vs dualcore vs...)
  - Word width (32 bits vs 64 bits vs...)

- Typical ways of expressing computational power:
  - MIPS
  - MFLOPS
  - …

MIPS

- Millions of Instructions Per Second.
- Typical range: 10-100 MIPS
- Not all instructions take the same amount of time to execute Depends on which instructions are executed.
- Not 100% reliable as a measure of performance.

MFLOPS

- Millions of Floating Point Operations per Second.
- Scientific computing power.
- MFLOPS < MIPS
  - Floating operation more complex than normal operation
- Vector Computers: MFLOPS > MIPS
- Example: Itanium 2 → 3,5 GFLOPS

Vectors per second

- Computing power in graphics generation.
- Applicable to graphics processors.
- Can be measured in:
  - 2D vectors.
  - 3D vectors.

- Example: ATI Radeon 8500 → 3 Million.

Content

1. What is a computer?
2. Computer structure and computer architecture
3. Building blocks for a computer
4. Von Neumann architecture
5. Machine instructions and assembly programming
6. Execution steps of an instruction
7. Main characteristic parameters of a computer
8. **Types of computers**
9. Historic evolution

Félix García-Carballeira, Alejandro Calderón Mateos

Types of computers

- Desktop
- Personal mobile devices
- Servers
- Clusters
- Embedded

Types of computers

- Desktop
  - Designed to deliver good performance to users
  - Currently, most of them are portable
  - Design aspects:
    - Price-performance ratio
    - Power
    - Graphics performance

Types of computers

- Personal mobile devices
  - Wireless devices with multimedia user interface
  - Smartphones, tablets,…
  - Design aspects:
    - Price
    - Energy
    - Performance
    - Response time

Types of computers

- Servers
  - Used to run high performance or scale applications
  - Serve multiple users simultaneously
  - Design aspects:
    - Throughput (Processing rate)
    - Availability
    - Reliability
    - Energy
    - Scalability

Types of computers

- Clusters
  - A set of computers connected by a network that acts as a single, higher performance computer.
  - Used in supercomputers and large data centers.
  - Design aspects:
    - Price-performance
    - Throughput (Processing rate)
    - Availability
    - Reliability
    - Energy
    - Scalability

Types of computers

- Embedded
  - Computer inside another system to control its operation.
    - Washing machines, TVs, MP3 players, video game consoles, etc.
  - Design aspects:
    - Price
    - Energy
    - Application specific performance

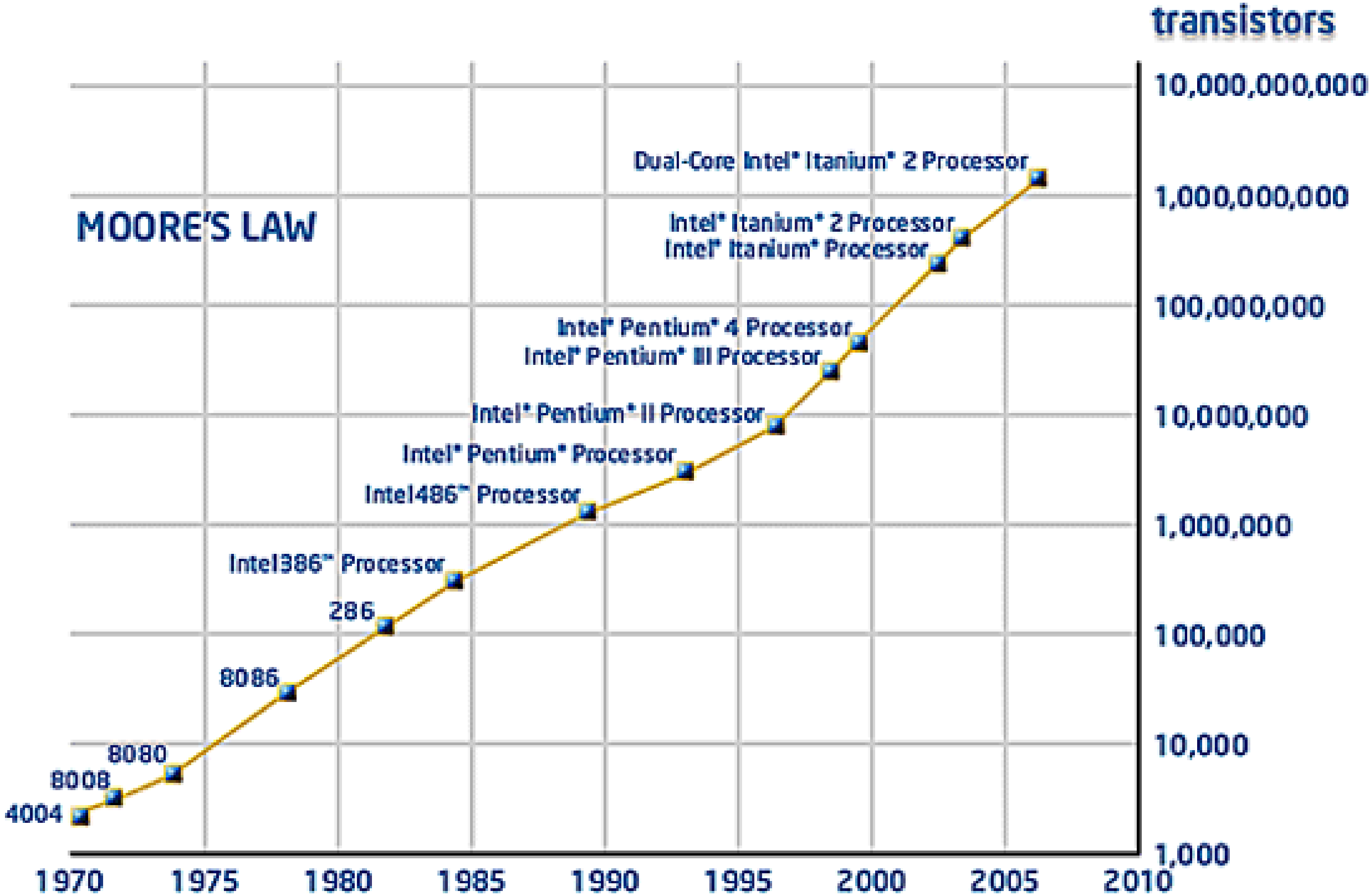
Content

1. What is a computer?
2. Computer structure and computer architecture
3. Building blocks for a computer
4. Von Neumann architecture
5. Machine instructions and assembly programming
6. Execution steps of an instruction
7. Main characteristic parameters of a computer
8. Types of computers
9. **Historic evolution**

Microprocessor

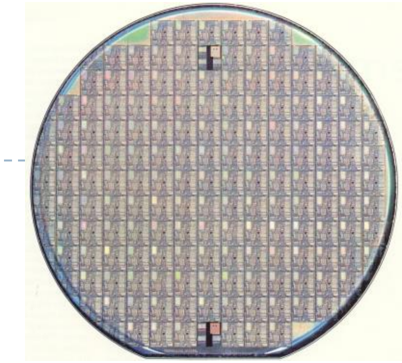
- A **microprocessor** incorporates the functions of a computer's central processing unit (CPU) on a single integrated circuit

Moore's law

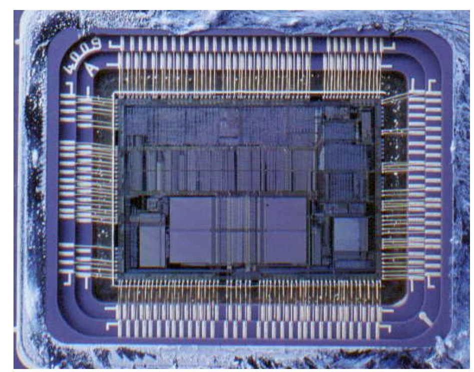


MOORE'S LAW
transistors
10,000,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
100,000
10,000
1,000
Dual-Core Intel® Itanium® 2 Processor
Intel® Itanium® 2 Processor
Intel® Itanium® Processor
Intel® Pentium® 4 Processor
Intel® Pentium® III Processor
Intel® Pentium® II Processor
Intel® Pentium® Processor
Intel486™ Processor
Intel386™ Processor
286
8086
8080
8008
4004
1970
1975
1980
1985
1990
1995
2000
2005
2010

Moore's law



- Double the density implies to reduce the dimensions the 30%
- In 1971 the 4004 Intel had 2300 transistors of 10 micrometers
- Nowadays there are microprocessors with less than 30 nanometers
- Moore's law need technology with a price that double every 4.4 years



Technology improvements

- Memory
  - DRAM capacity: 2x / 2 years (since 96);
    64x in the last decade.
- Processor
  - Speed: 2x / 1.5 years (since 85);
    100X in the last decade.
- Disks
  - Capacity: 2x / 1 year (since 97)
    250X in the last decade.

Historic evolution

- http://history.sandiego.edu/GEN/recording/computer1.html
- http://www.computerhope.com/history/
- http://www.computerhistory.org/
- http://www.computersciencelab.com/ComputerHistory/History.htm
- Museos de informática
- **In Google/Bing, look for: "Computer history"**