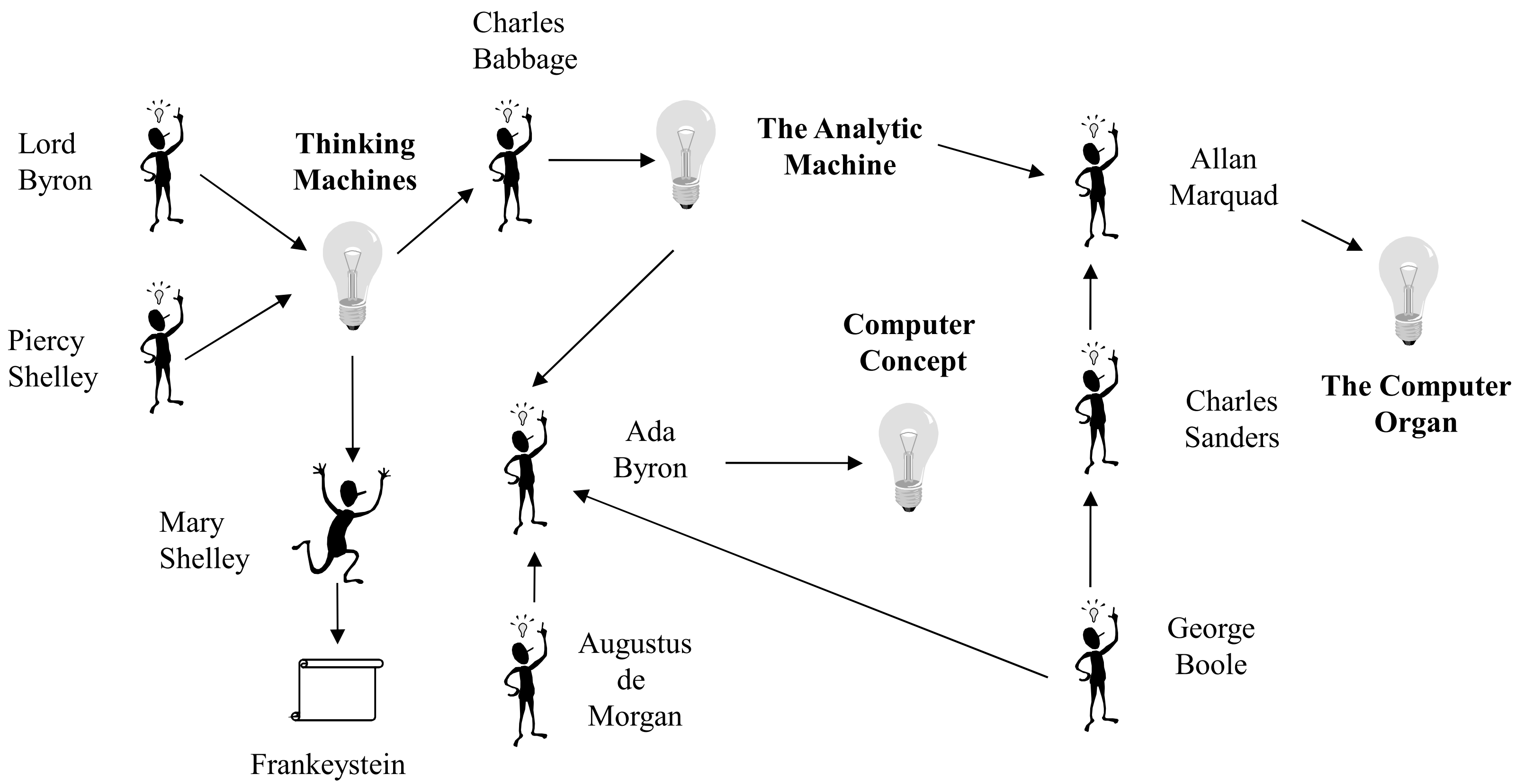
TECNOLÓGICO DE MONTERREY

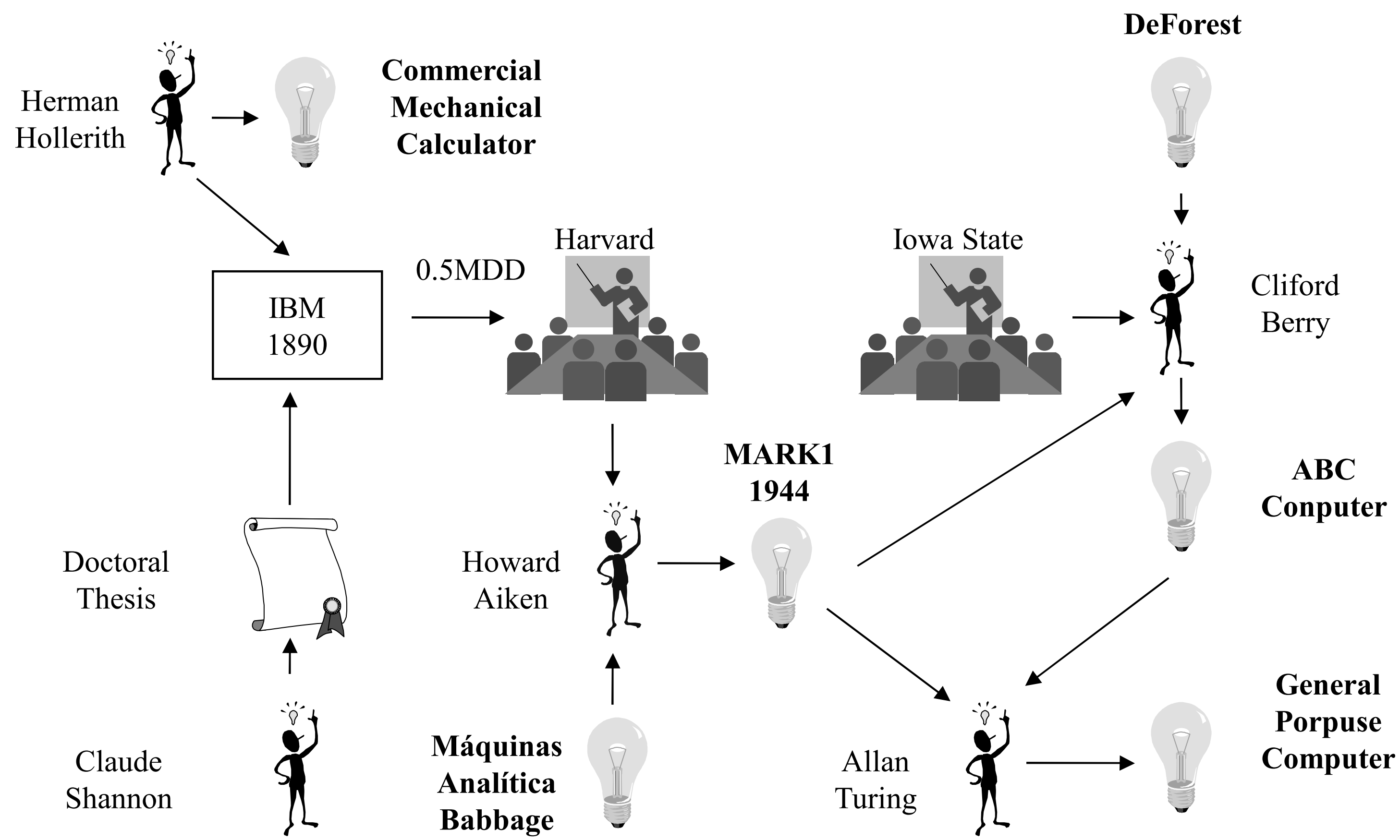
Subject 1

INTRODUCTION TO MICROCONTROLLERS

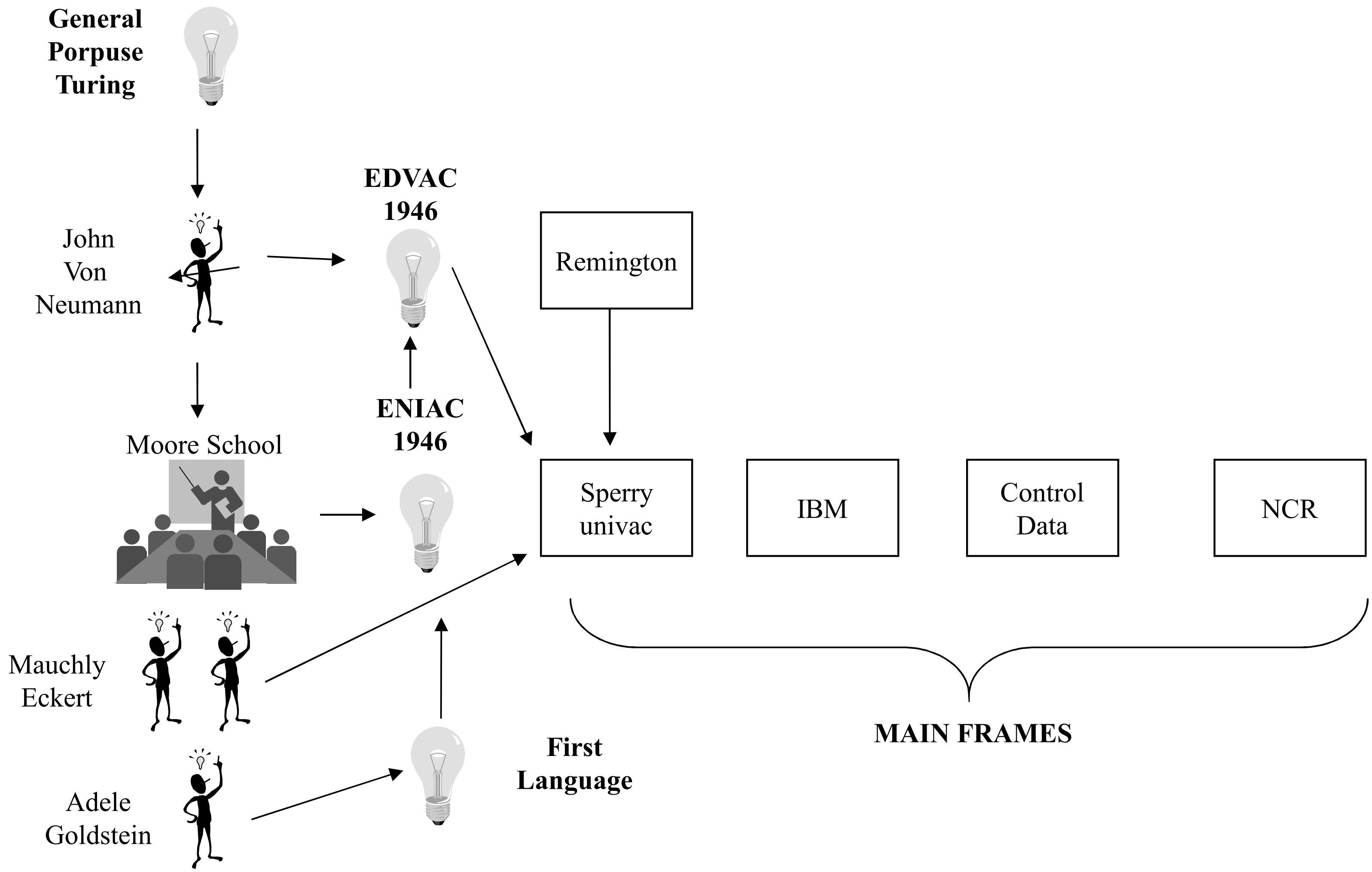
Initial concepts



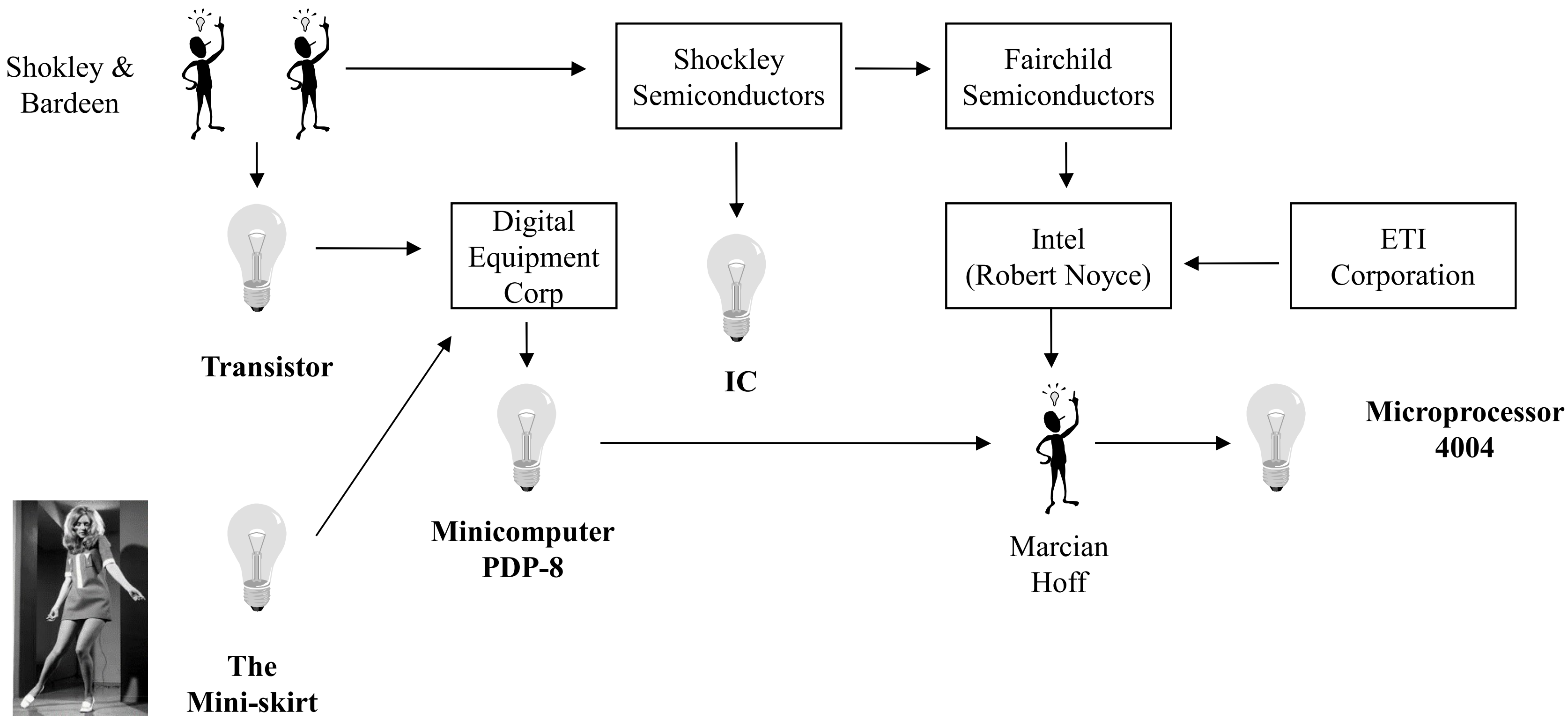
Mechanical computers



Computadoras electrónicas



The Microprocessor



Features of a Computer

The ability to be programmed to operate on data without human intervention

The ability to store and retrieve data

Elements that form the computer

The Processor o CPU:

- The brain that controls and calculates

Input devices:

- Feed the computer with programs and the data

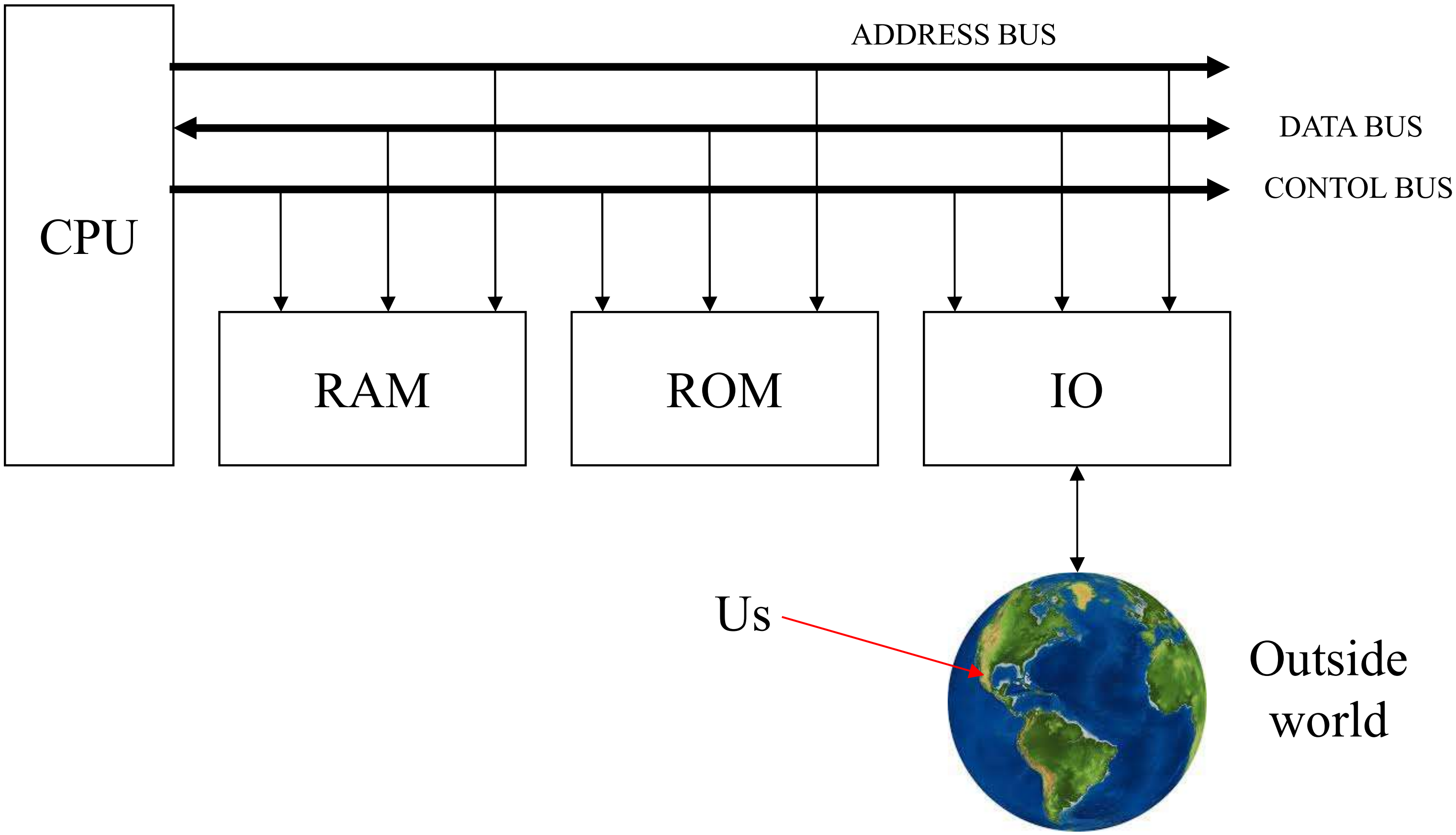
Output devices:

- Provide feedback and the outcome of the calculation to its user

Memory devices

- Store data and program

Arquitectura



CPU

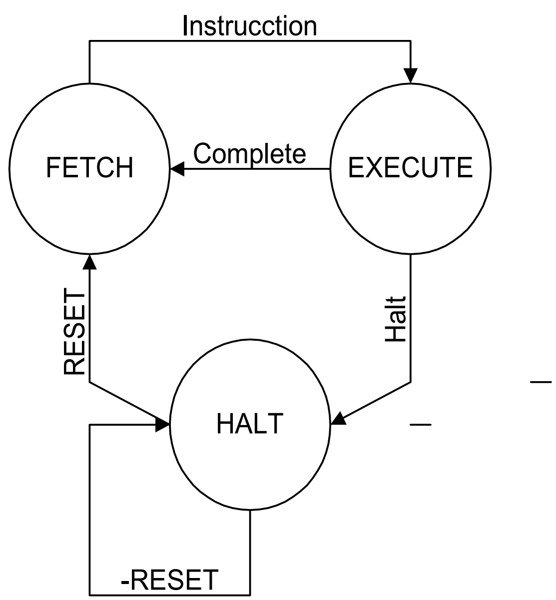
Central processing Unit

Manages the system activities

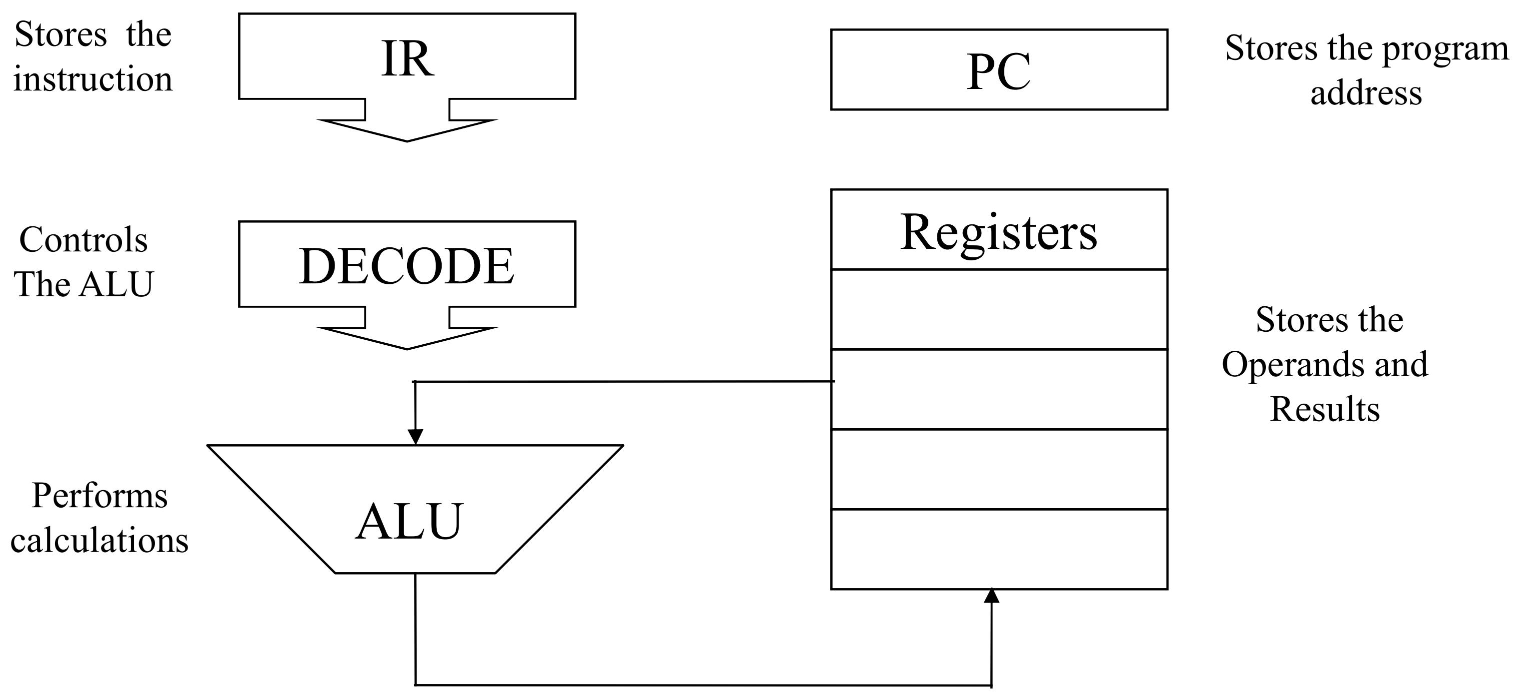
Performs the extraction (fetch) and execution of the instructions

Is a “simple” state machine that always performs the same action

In the execution there are “micro-actions” that tunes the cycle



Components of the CPU

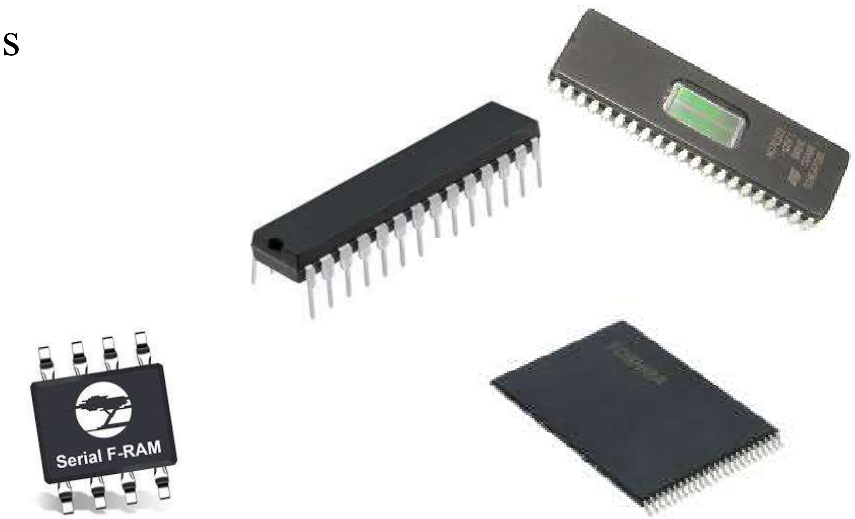


Read Only Memory

Stores programs that cannot change, in computers, its stores what is called the BIOS in the majority of embedded devices it stores the main program

Semiconductor type of ROM’s

- EPROM (UV)
- OTP
- EEPROM
- FLASH
- FRAM

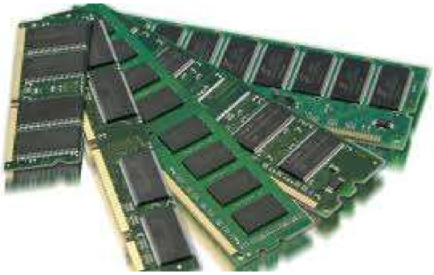


Random Access Memory (RAM)

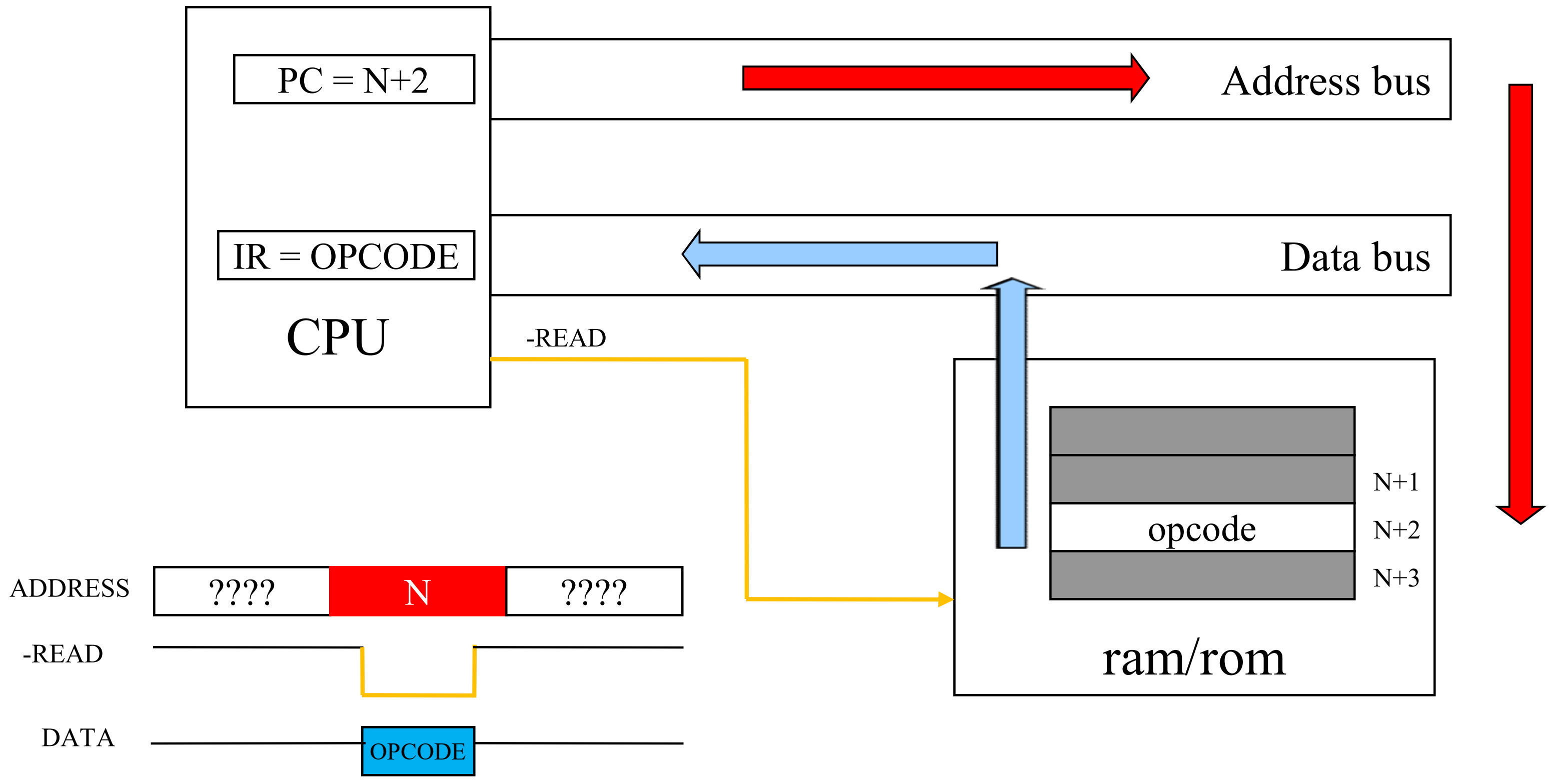
Is a volatile memory (does not retain data in the absence of power). Can be read and written, it stores temporary data and also programs

Semiconductor type of RAM's

- SRAM
  - Static, can be used as a RAM and with battery backup can act as a ROM, is very power efficient bur is very complex and "big" from the semiconductor point of view
- DRAM
  - Never stores data after power
  - Is very small and simple but requires controllers to refresh the data to operate this consumes lots of power



The “Fetch” cycle



Address Bus

- Collective group of signals to direct a specific element of the memory and devices to the CPU (RAM,ROM, IO)
- Its unidirectional, that is, goes always from the CPU to the RAM, ROM or IO
- The capacity of addressing of the CPU is given by 2^n, where n is the number of signals (wires) to the element

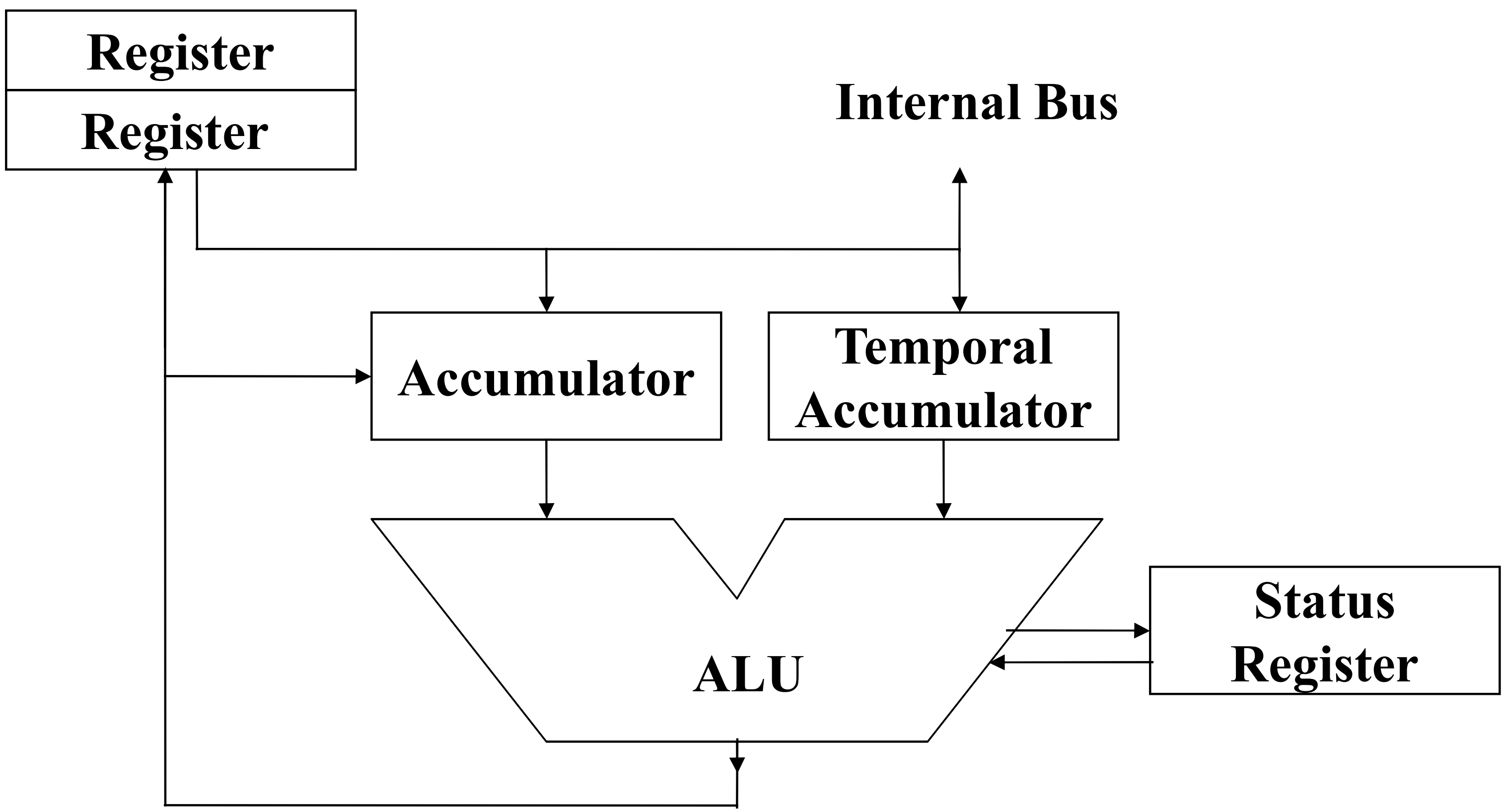
Data bus

- Is a bidirectional collective group of signals that goes in and out to the CPU
- It's the path where the data is written and read from the memories and IO (peripheral)
- Its size indicates the processing capacity or the CPU
  - It is related to the internal datapath with (number of bits) that the CPU can process, more number of signals (width) more "horse power"
- 4,8,18,32 and 64 bits (128 is till today to massive)

Control Bus

- Provide all signals required to read and write to the memories and IO
- Can also provide signals to control process via hardware

ALU


Register
Register
Internal Bus
Accumulator
Temporal Accumulator
ALU
Status Register

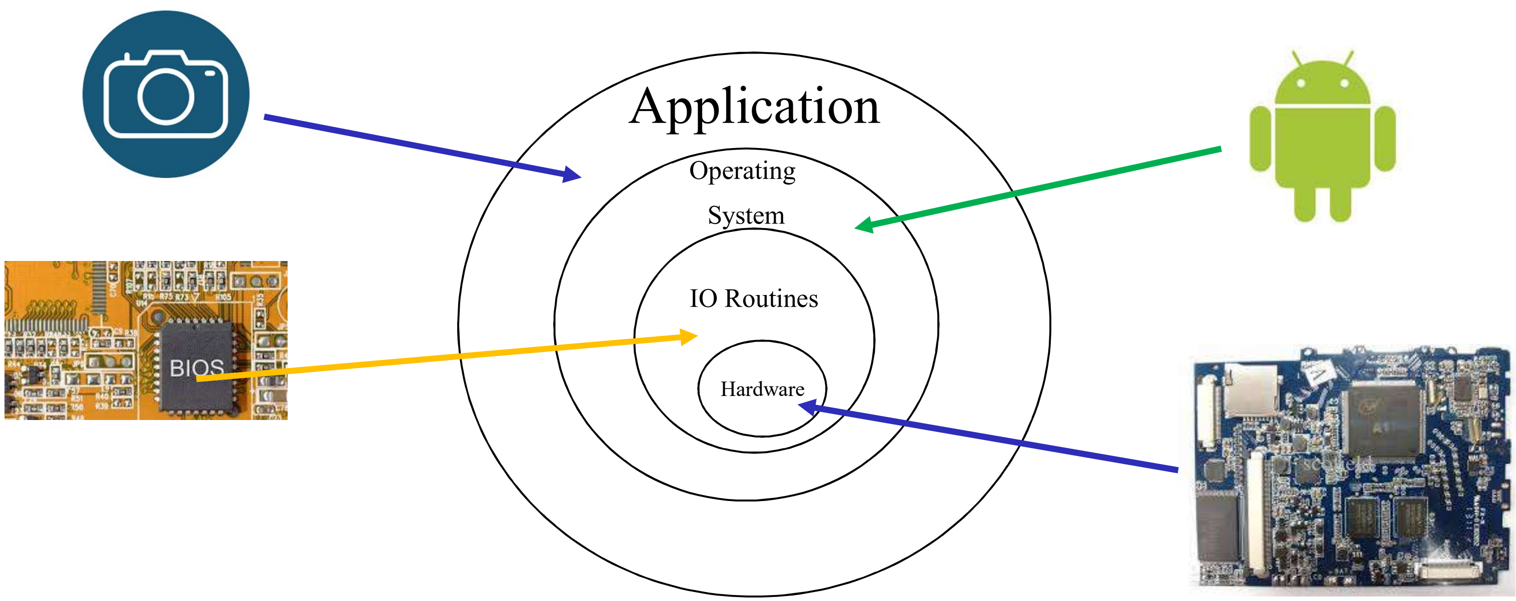
Instructions

A + B

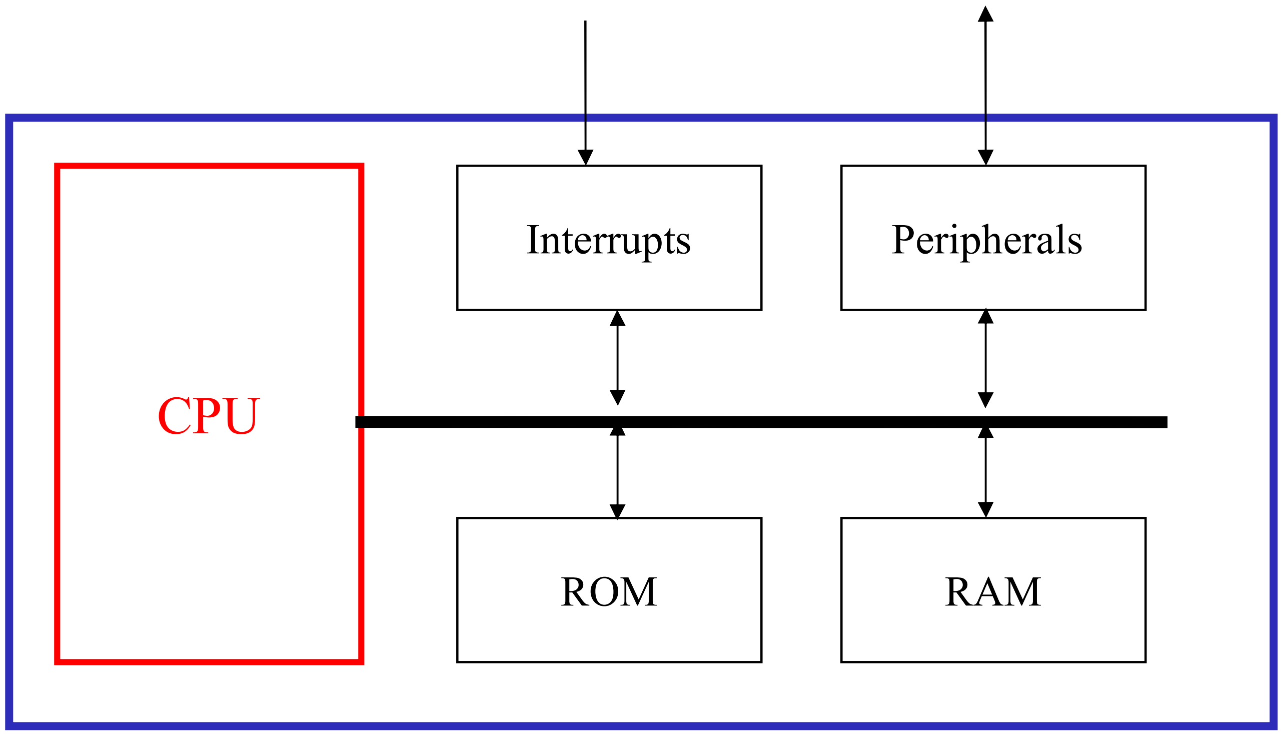
Operation Code | Operand 1 | Operand n

Programs

It is the sequence of instructions that define the operation of the computer to perform a task



Architecture of the microcontroller

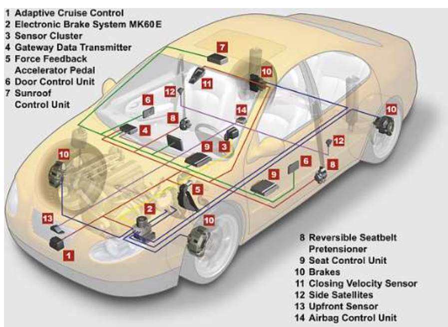


Microcontroller is a singe chip computer
Microprocessor the **CPU** or a computer

Difference of MPU and a MCU from the application point of view

- Microprocessor (uP or MPU):
  - Focused o very high data volume data processing, it is used as the CPU of the computer.
- Microcontroller (uC or MCU):
  - Medium or small applications focused single purpose application. It replaced conventional combinational and sequential logic
  - EMBEDDED SYSEMS

Applications of an MCU in embedded systems



Differences between the MPU and MCU from the instructions point of view

MPU or uP:

- Complex instructions that are focused on the massive data processing with a very flexible addressing schemes.
- Uses a very big set of instructions.

MCU or Uc:

- Instructions more simple that are focused on the control of I/O.
- Basic mathematical operations and is a very small set of instructions compared with MPU
- The main goal is that the program is small enough to fit on a single ROM from the theoretical point of view

Differences between the MPU and MCU

The main difference between a MPU used in a computer and the MCU is that the MCU will theoretically always perform the same program designed to perform a specific task that interacts with the human and machine world. This is called EMBEDDED CONTROL

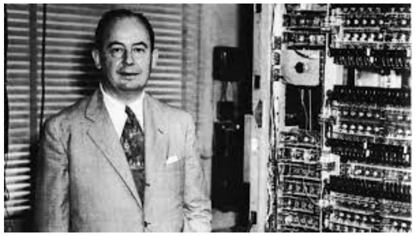
Programs for MCU's are called FIRMWARE

The relation between RAM/ROM
MCU = Low
MPU = High

Microcontroller feature taxonomy

- Architecture
  - Harvard, Von-Neumman, Multi-core
- Instruction types
  - RISC o CISC
- Manufacture technology (who cares??)
  - CMOS,NMOS etc.

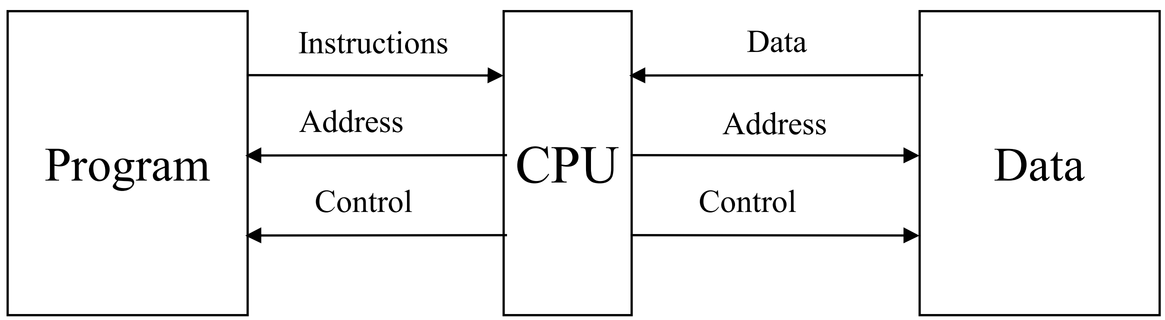
The Von Neumann Architecture



- Uses a single bus for data and instructions
- Programs and data reside on the same memory space (from the architectural point of view)
- It sequentially make a fetch of the instructions and the data using the same hardware resources.
- In theory and in practice is a slower architecture, but is simpler to code applications.

The Harvard architecture

- It has a independent data and instructions including the addressing bus.
- It can perform a fetch while an instruction is already in execution.
- Faster to execute an instruction but complex from the hardware point of view.



CISC

- Complex Instruction Set Computer
- Big instruction set (more than 250)
- Instructions are specialized on specific and complete tasks
- Instructions are specialized for specific taks on the memory space (ROM, RAM and IO)

RISC

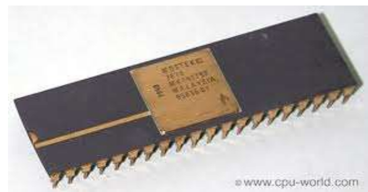
- Reduced Instruction Set Computer
- Simpler instructions, less than 200.
- Useful in Harvard type architectures.
- Orthogonal instructions
- The efficiency of the program depends on the compiler (not a factor anymore)

Evolution

TMS1000c

1974

MK3870



8051

1980

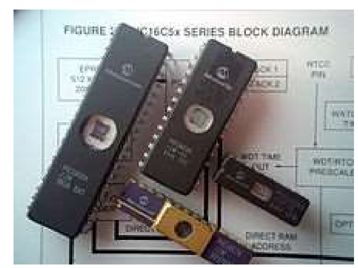
8048

1976

HC11

1985

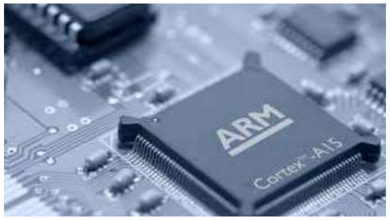
PIC16



1990-(1975)

Evolution: Systems on Chip (Soc)

ARM-MCU



1998

MIPS

2005

Difference between microcontrollers.

- Architecture
- Memory sizes
- Programming flexibility (OTP or MTP)
- Hardware protection and encryption
- Power supply
- Speed and power and its relation.
- Processing capacity (horsepower)
- Low power modes
- Protections
- Peripherals
- COST !!!

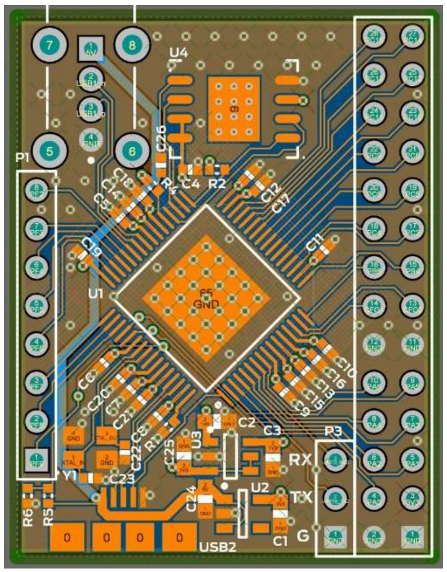
New tendency

- The increase of rich features in products demand nowadays much more computing power
- IoT has also introduced new elements as part of the ecosystem like:
  - Edge computing
  - Protocol gateways
  - Digital assistants
- These elements are now also present in vehicles, industrial, medical, applicances and many other niches

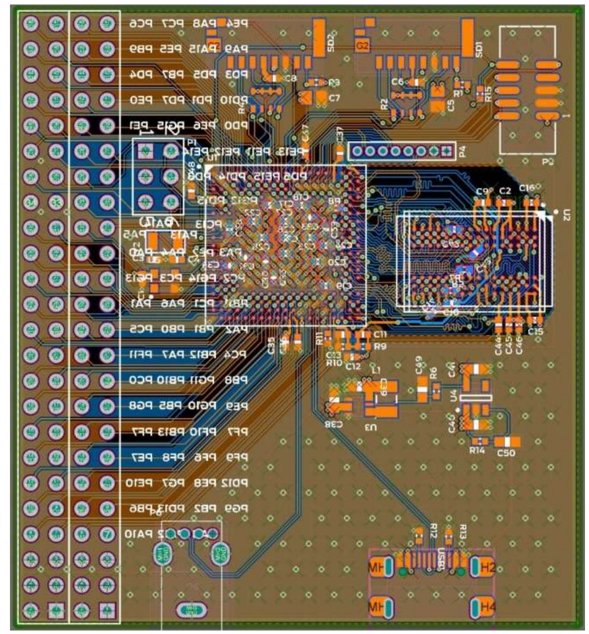
The System In Package (SIP)

- Uses a high power single or multicore MPU
- Includes internal or easy to add external DDR
- Rich set of peripheral interfaces like MIPI, SDIO, Ethernet,
- Provides MMU that allows the use of operating systems like Linux
- Is basically a miniaturized computer in an integrated circuit (multi-die, multi-chip)

The System In Package (SIP)

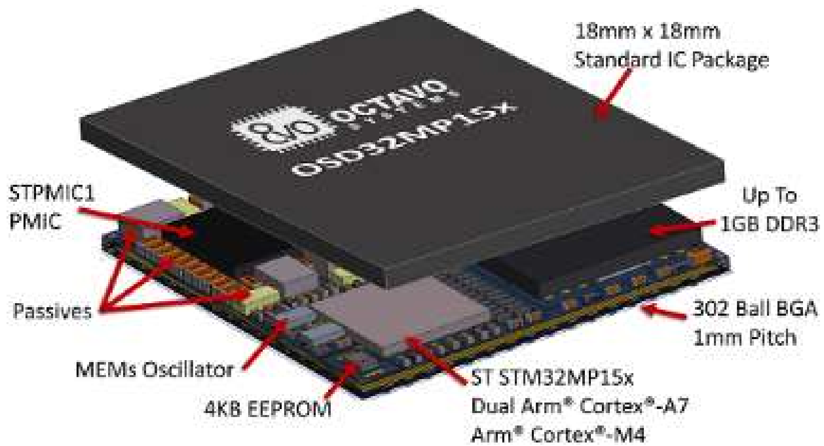


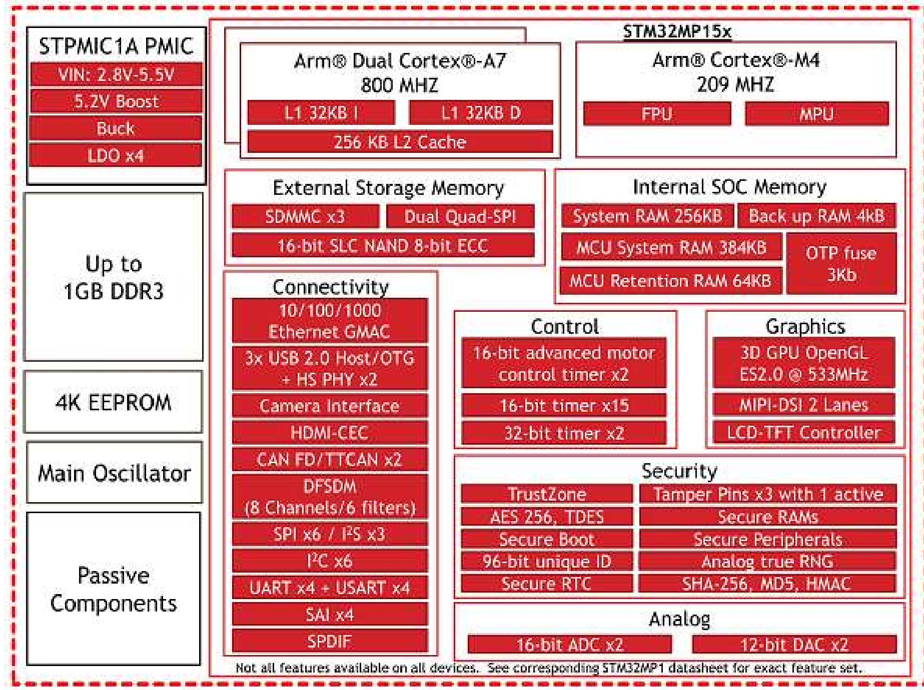
NUC980 $5.0



STM32MP1 $12

The System In Package (SIP)





OSD32MP157C-512M-IAA $43.00

Recommend to read:

https://jaycarlson.net/embedded-linux/

TECNOLÓGICO DE MONTERREY

Teacher, my final grade was 69.9 did I pass the course?

- ☐ A) Yes
- ☐ B) No
- ☐ C) Almost

Programming languages

Assembly language

Is the mnemonic representation of the machine language, generally each written instruction represents a CPU instruction .

- Very fast (you talk to God directly)
- Very compact
- Is currently used in very small microcontrollers or on libraries or sections of code that require speed.
- Is a MUST learn for all embedded system developers
- Is very hard to maintain

Programming languages

High level interpreters.

- Is a translator between a high level language like BASIC, JAVA, PHYTON.. but there is a middle man that interprets the code and translates it to the CPU on real time.
  - Easy to maintain and code
  - Programs are slow (thanks to the middle man)

Programming languages

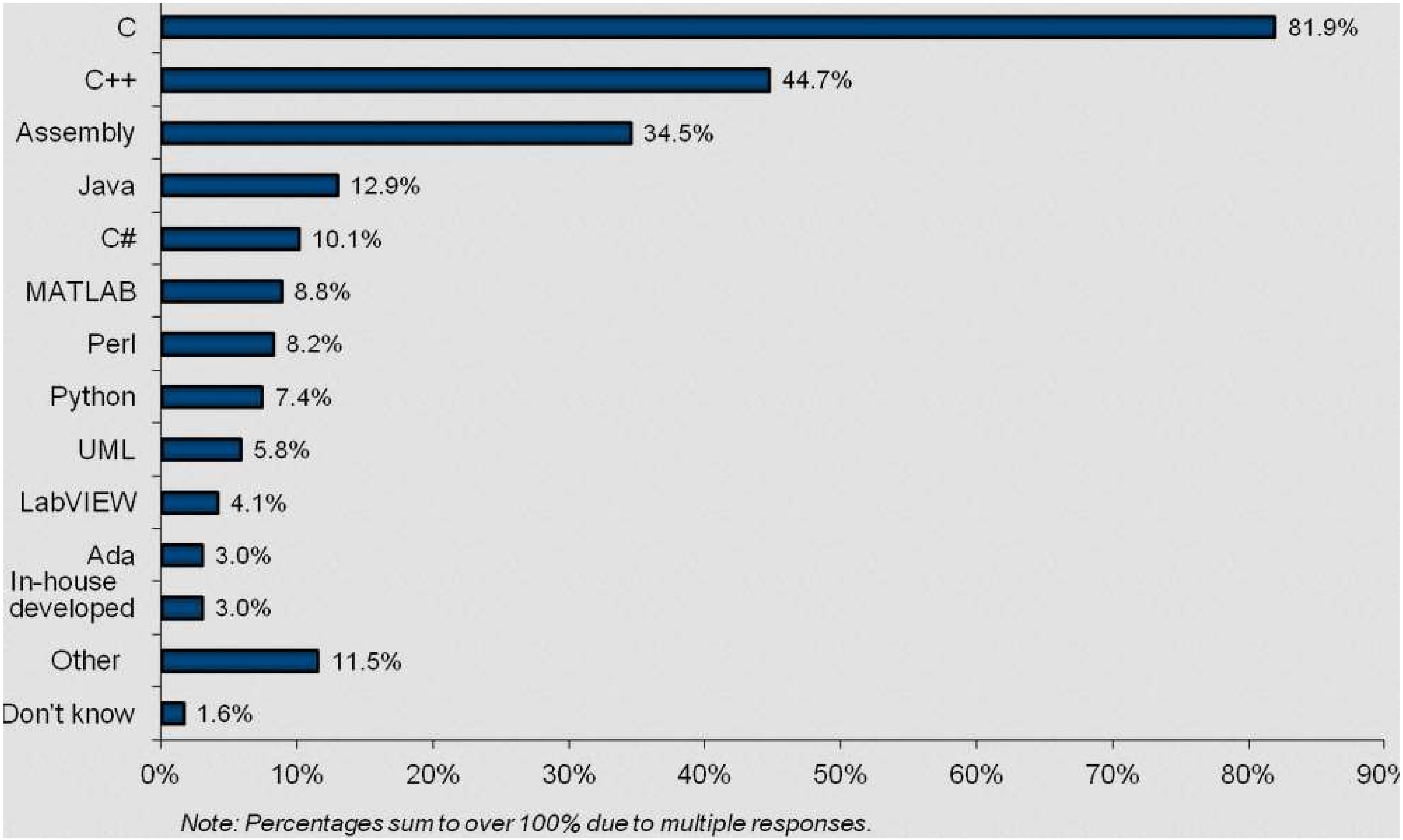
High level compilers

- Translates a high level languages like C, C++, JAVA to language machine (assembler) so there is no middle man.
  - Easy to develop
  - Hard to test (takes more time because the compiler)
  - Faster when the finally execute.
  - Super to use on MCU nowadays

Cross-assembler y Cross-compiler

When the assembler or the compiler will generate code that will be executed in a different platform where it was developed, the prefix "cross" is used to differentiate.
For microcontrollers, your development tool will always have the "cross" prefix. (if they don't, they are jerks.. many of them nowadays)

Preferred languages for MCU? (VDC Research)



A look of languages for microcontrollers:

ASSEMBLER

```
              bra     inicia          ;Brinca a inicio del programa
              org     0x0100                  ;Programa empezara en direcccion 0x:
inicia:
              call    inicializa_p    ;Subrutina de inicializacion de puertos
              bcf     PORTD,MOTOR     ;Apagar el motor (activo alto)
              bsf     PORTD,TORRETA    ;Encender torreta (activo alto)
              clrf    CUENTA          ;Hacemos la cuenta = 0
espera_boton: btfsc   PORTB,BOTON     ;Test bit and skip if clear (0)
              bra     espera_boton    ;Si es 1 continua preguntando
              bsf     PORTD,MOTOR     ;Encender el motor (activo alto)
              bcf     PORTD,TORRETA   ;Apagar torreta (activo alto)
espera_caja1: btfsc   PORTB,SENSOR    ;Test bit and skip if clear (0)
              bra     espera_caja1    ;Si es 1 continua preguntando
              movlw   0x01            ;Vamos a sumar 1 W = 1
              addwf   CUENTA,F        ;Sumamos CUENTA = CUENTA + W
espera_caja2: btfss   PORTB,SENSOR    ;Test bit and skip if set (1)
              bra     espera_caja2    ;Si es 0 continua preguntando

              cpfseq  CUENTA           ;Compare f with WREG, skip =
              bra     espera_caja1    ;No igual, espera la siguiente caja
              bsf     PORTD,TORRETA   ;Encender torreta (activo alto)
              bcf     PORTD,MOTOR     ;Apagar motor
              clrf    CUENTA          ;Hacemos la cuenta = 0
              bra     espera_boton    ;Esperar boton
```

A look of languages for microcontrollers:

HIGH LEVEL

```
main(void){
int cuenta = MAXIMO;               //Variable que almacena la cuenta de cajas
init_ports();                      //Inicializa los puertos
while(1){                         //Lazo principal se repite por siempre
    if(cuenta == MAXIMO){         //Verificamos si se llego a la maxima cuentas.

        MOTOR = 0;//Apagamos el motor
        TORRETA = 1;//Encendemos torreta
        cuenta = 0;//Hacemos 0 la cuenta
        //ESPERA A QUE PRESIONE BOTON (MIENTRAS SEA 1)
        while(BOTON)
        MOTOR = 1;//Enciene el motorwhile(BOTON
        TORRETA = 0;//Apagamos torreta
    } else {
        while(CAJA);// ESPERAMOS QUE LLEGUE LA CAJA
        cuenta++;//INCREMENTAMOS CUENTa
        while(!(CAJA));//* ESPERAMOS QUE PASE LA CAJA
    }//del if
} //while(1)
} //de main() TEMA_04_PIC_2.C
```

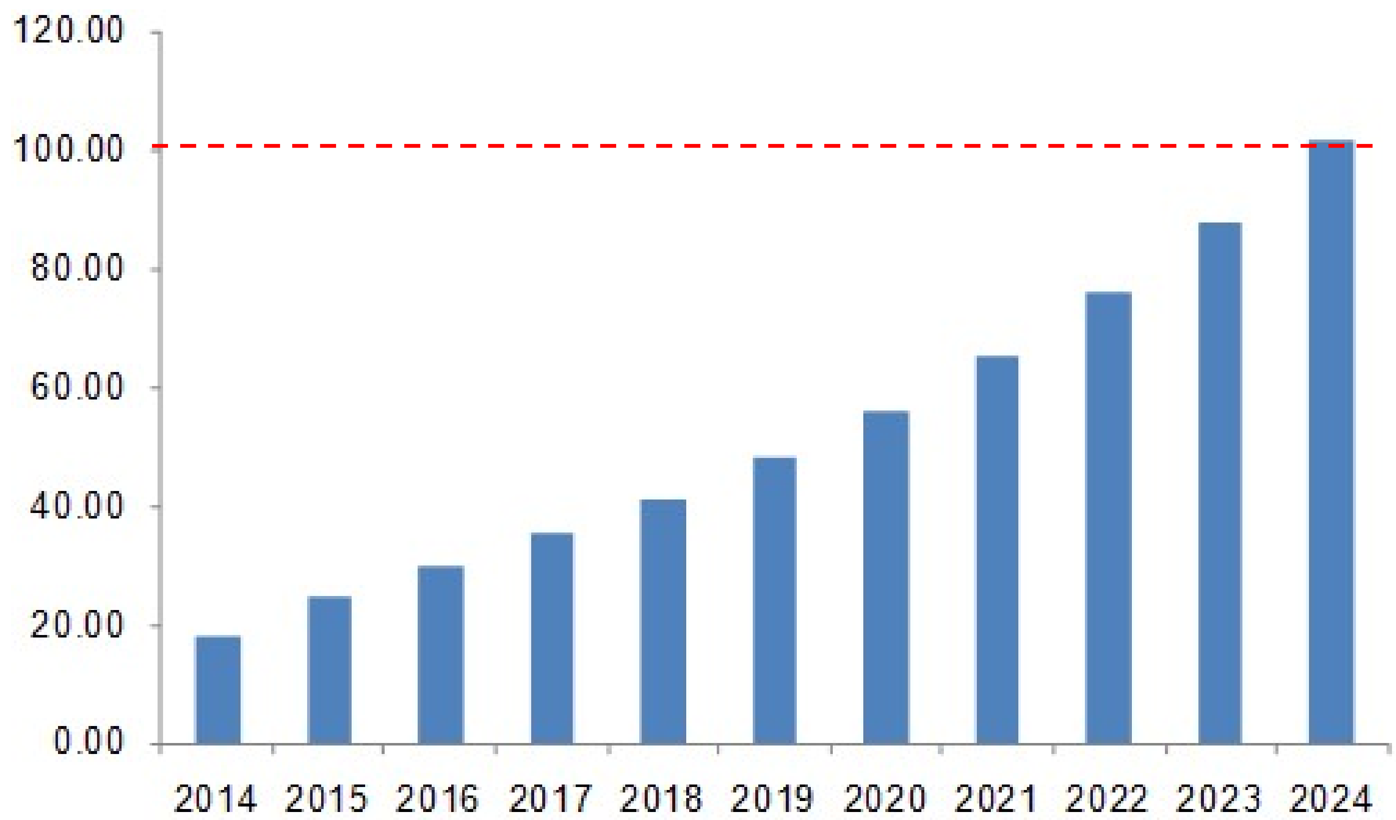
TECNOLÓGICO DE MONTERREY

¿Why should I learn to design and program with microcontrollers?

Market share

- Relation between MPU and MCU = 400:1
- Sustained wrought in an average or 8% per year
- Sales for 2020 are expected to be on the 60,000 MD
- The cost drops about 1% per year
- The current average cost is about $1.20
- The manufacturers will treat you as divas if you design with their parts.

Market share of MCU's



Ubiquity

- Medium level car = 50 MCU’s
- High end car = 100 MCU’s **
- Average US house= 200

** High level car code is between 10 and 100 million lines

Market share per brand

Leading MCU Suppliers ($M)

2016 Rank	Company	2015	2016	% Change	% Marketshare
1	NXP*	1,350	2,914	116%	19%
2	Renesas	2,560	2,458	-4%	16%
3	Microchip**	1,355	2,027	50%	14%
4	Samsung	2,170	1,866	-14%	12%
5	ST	1,514	1,573	4%	10%
6	Infineon	1,060	1,106	4%	7%
7	Texas Instruments	820	835	2%	6%
8	Cypress***	540	622	15%	4%

*Acquired Freescale in December 2015.

**Purchased Atmel in April 2016.

***Includes full year of sales from Spansion acquisition in March 2015.

Source: IC Insights, company reports

Job Market

- Big, and I mean BIG !!, embedded system programmers are very highly recognized in the market, I mean HIGHLY … in Mexico:

JOHN DEERE

Continental
The Future in Motion

SONY

NXP

dextra technologies®

CAT
CATERPILLAR

TECNOLÓGICO DE MONTERREY

Review of basic concepts of digital systems

Numeric bases (radix)

A numeral system can be constructed using a base with an exponent (a power)

The number of symbols required to represent the numbers for the numeral system is equal to the base of the number.

Decimal System

Example: The number 5346.72

5	3	4	6	.	7	2
10^3	10^2	10^1	10^0		10^{-1}	10^{-2}

The number of symbols required to represent a number on decimal base is 10 from 0 to 9

Binary system

In the binary system the base is 2

We use two digits to represent any number (0 or 1) and its called a bit.

A group of bits is a number that is usually called a "binary word"

Based on their length, they have been named (de-facto):

- 4 bits word= nibble
- 8 bits word= byte
- 16 bits word= word
- 32 bits word= double word
- 64 bits word= long

Binary system

The last digit (at the right) is called

- LEAST SIGNIFICANT BIT o LSB

The first digit (at the left) is called

- MORE SIGNIFIANT BIT o MSB

Example:

1010

MSB **LSB**

Binary system

In power representation

1	**0**	**1**	**0**	**.**	**0**	**1**
2^3	2^2	2^1	2^0		2^{-1}	2^{-2}

Binary system

To convert any binary number to decimal, you multiply each digit by the power given by its significance in the word and then you add them

1	**0**	**1**	**0**	**.**	**0**	**1**
2^3	2^2	2^1	2^0		2^{-1}	2^{-2}

$$1x2^3+0x2^2+1x2^1+0x2^0+0x2^{-1}+1x2^{-2} = 10.25\ d$$

Hexadecimal system (Hex)

In the computation world, the hexadecimal (base 16) is highly used since it provides a more compact representation of a binary world that is also easier to remember than an string of 1 and 0's

Hexadecimal system

To convert a binary number to a hex number is easy. You just group bits in nibbles (4 bits) staring from the LSB. Each nibble is then converted to a hexadecimal digit (0 to F)

Hexadecimal system

Example: The binary number 10100011

$$\overbrace{\underbrace{1\quad 0\quad 1\quad 0}_{\substack{\mathbf{A}\\ \text{Nibble}}}\quad \underbrace{0\quad 0\quad 1\quad 1}_{\substack{\mathbf{3}\\ \text{Nibble}}}}^{\text{Byte}}$$

Binary	Hex
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

BCD code

The BCD code (Binary Coded Decimal) also uses the same nibble grouping concept (4 bits) to represent a number.

It was basically conceived to interact as the input and output of the computers since culturaly the decimal system is the international standard to represent numbers.

It is really a sub-set of the Hex code where the digits A to F are not valid

BCD Code

Example: The binary number 010100101001

$$\underbrace{0\ 1\ 0\ 1}_{\mathbf{5}}\ \underbrace{0\ 0\ 1\ 0}_{\mathbf{2}}\ \underbrace{1\ 0\ 0\ 1}_{\mathbf{9}}$$

BE CARFULL !!, this number is 529 BCD however the direct conversion from the binary is also 1321 decimal

The ASCII code

ASCII: American Standard Code for Information Interchange. Designed to interact literally with computers and to communicate them (1963)

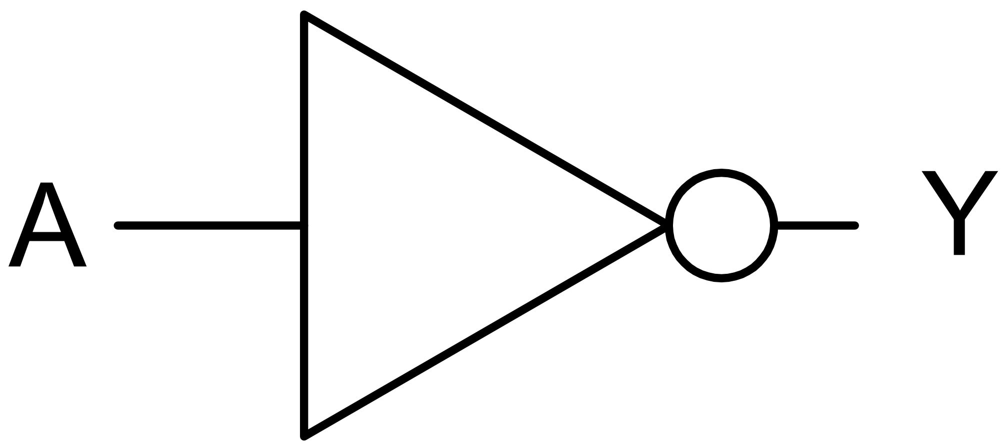
Each byte represent a symbol (Char) where most of them are printable and readable in the Latin alphabet*. The original standard had 128 symbols and control characters (7 bits) then it was extended to 8.
There are extensions and variations as the ASCCII 885 (Cyrillic), Greek (737) etc…

https://www.ascii-codes.com

NOT

$$Y = \overline{A}$$

A	Y
0	1
1	0

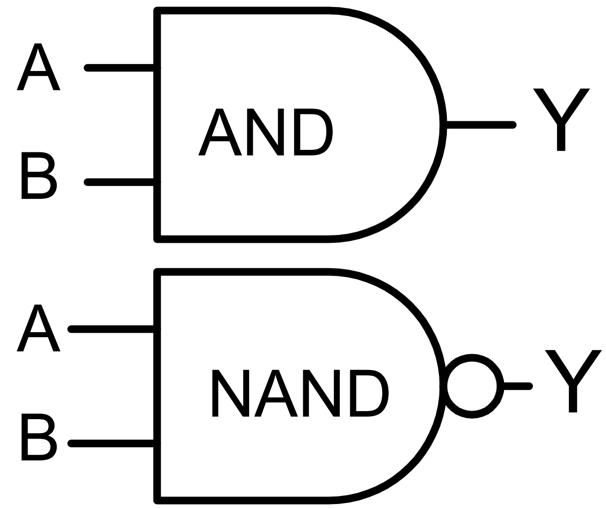


AND , NAND

$$Y = A \bullet B$$

$$\overline{Y} = A \bullet B$$

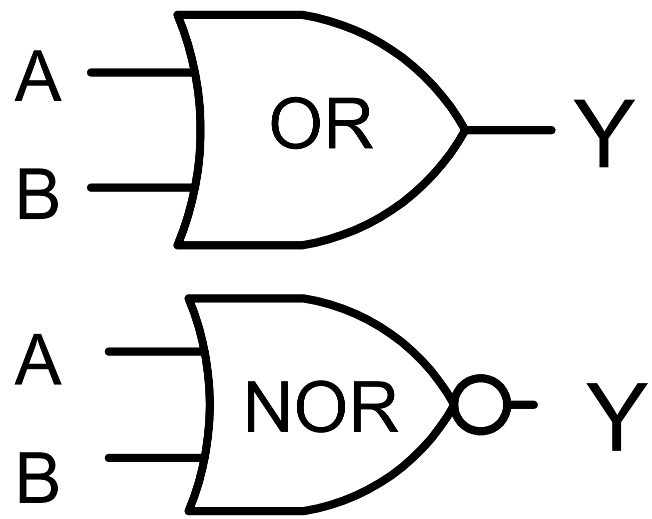
A	B	Y	Y
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0



OR , NOR

$$Y = A + B \qquad \overline{Y} = A + B$$

A	B	Y	Y
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0



XOR , XNOR

$$Y = A \oplus B \qquad \overline{Y} = A \oplus B$$

A	B	Y	$\overline{Y}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1