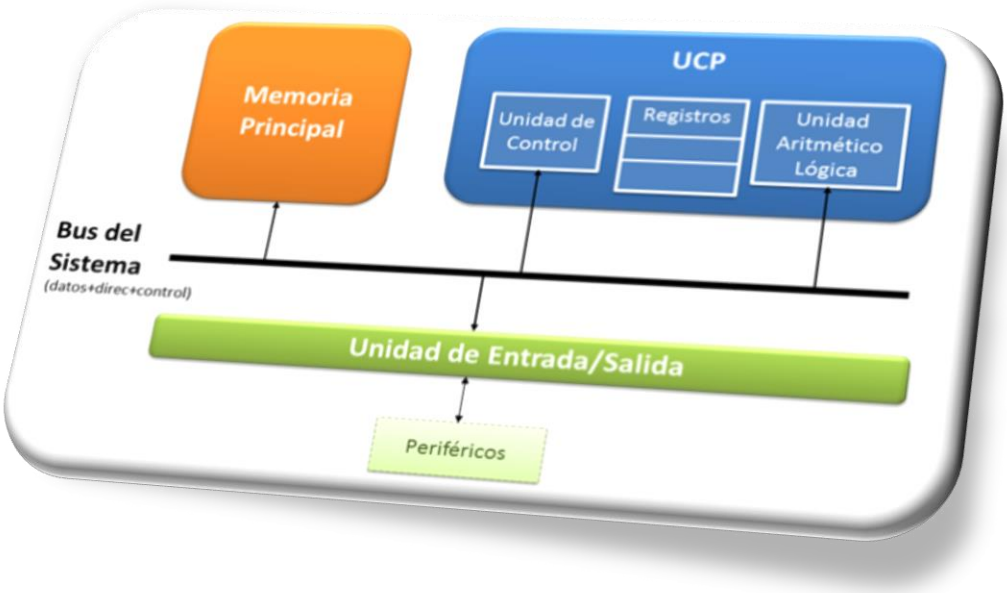
SISTEMAS INFORMÁTICOS

UD 1: INTRODUCCIÓN A LOS SISTEMAS INFORMÁTICOS

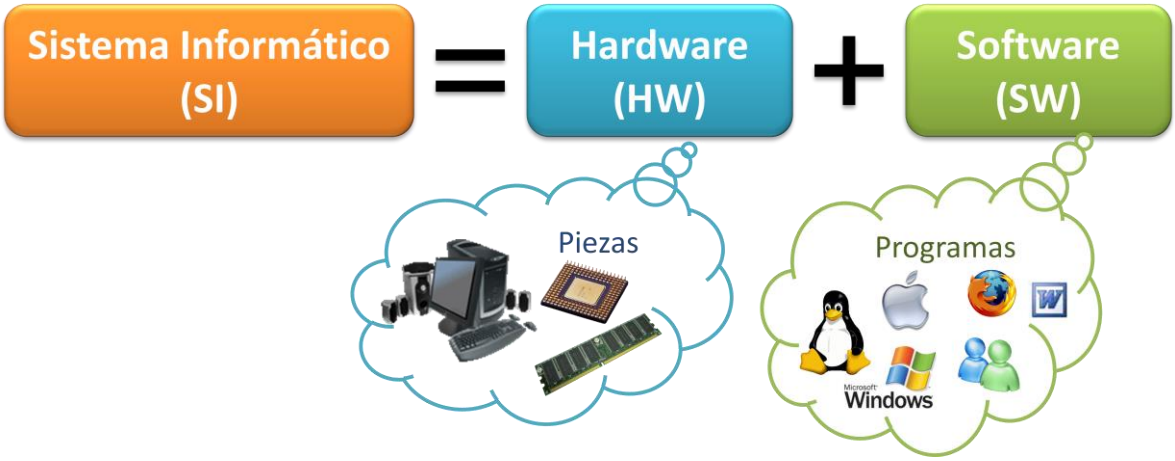


ÍNDICE

1. El Sistema informático

El **ordenador** es la herramienta que nos permite el tratamiento automático de la información, es decir, que nos permite organizar, procesar, transmitir y almacenar la información.

Un **sistema informático**, en mayor o menor medida, es precisamente esto, un conjunto de elementos de **hardware** y **software** interconectados para el tratamiento automático de la información (de ahí la palabra informá-tica). Por otro lado, también se podría incluir el elemento humano (por ejemplo, cuando la persona que usa el sistema, interviene en él introduciendo datos).



Por tanto, un sistema informático se puede definir como una serie de **elementos físicos (hardware)** capaz de realizar muchas tareas a gran velocidad y con gran precisión. Para que estos elementos físicos realicen un proceso determinado, es necesario que en él se ejecuten un conjunto de **órdenes o instrucciones (software)**, componentes no físicos que pongan en funcionamiento todos esos componentes físicos.

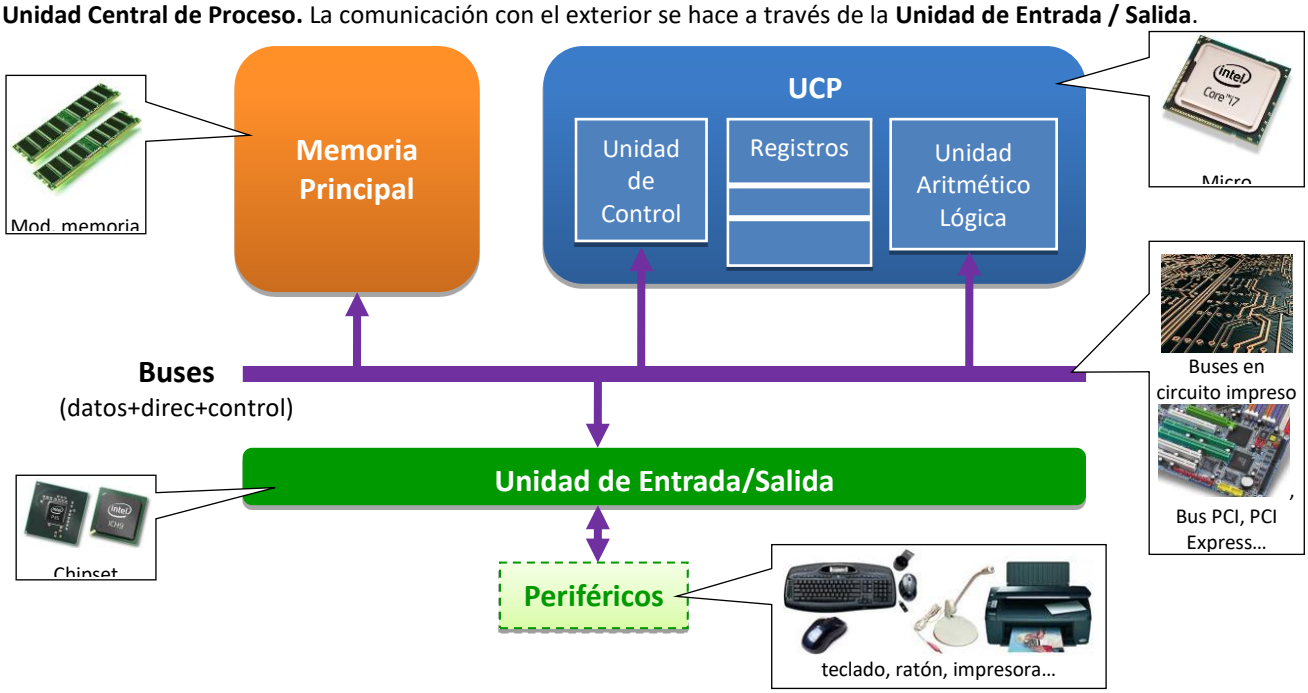
Podemos hacer una clasificación de los SI en:

De este modo, encontramos desde los **superordenadores** utilizados por grandes organizaciones en la realización de tareas muy avanzadas de tipo científico, militar, tecnológico…, pasando por los **macroordenadores** y **servidores** que proporcionan servicios diversos a otros equipos, hasta llegar a los **ordenadores personales** y **dispositivos de bolsillo** (móviles modernos, PDA…) que usamos en el ámbito doméstico o profesional.

2. Componentes físicos: Hardware

2.1. Arquitectura Von Neumann

El modelo de arquitectura básica empleada en los ordenadores digitales fue establecido por **Von Neumann** en 1945. Esta máquina es capaz de ejecutar una serie de instrucciones u órdenes elementales llamadas **instrucciones máquina**, que deben estar almacenadas en una unidad de almacenamiento llamada **Memoria Principal**, para poder ser leídas y ejecutadas por la **Unidad Central de Proceso.** La comunicación con el exterior se hace a través de la **Unidad de Entrada / Salida**.



2.2. Unidad Central de Proceso

La **Unidad Central de Proceso** o microprocesador (UCP o CPU): se encarga de la ejecución de los programas almacenados en memoria y está formada por:

- La **Unidad de Control** (UC): decide y controla las operaciones que se van a realizar en cada momento. Su función es decodificar las instrucciones del programa en ejecución y generar todas las señales necesarias para que puedan ser ejecutadas.
- La **Unidad Aritmético Lógica:** realiza las operaciones aritméticas (sumas, restas...) y lógicas (And, Not, Or...).
- Los **Registros:** son un almacén temporal durante la ejecución de una instrucción. Se trata de memorias muy pequeñas de acceso muy rápido.

2.3. Memoria principal

La **Memoria Principal:** contiene las instrucciones de los programas para ser ejecutados por la CPU y los datos a usar.

Se organiza como un conjunto de celdas consecutivas de longitud fija (palabra), enumeradas de forma que cada celda es directamente accesible por su número denominado dirección. La unidad mínima de información que almacena es el **bit** (con dos estados: 0 y 1).

Su capacidad se suele expresar en:

Medida	Simbología	Equivalencia	Equival. (Bytes)	Medida	Simbología	Equivalencia	Equival. (Bytes)
Byte	B	8bits		**Terabyte**	TB	1024 GB	2^{40} B
Kilobyte	KB	1024 B	2^{10} B	**Petabyte**	PB	2014 TB	2^{50} B
Megabyte	MB	1024 KB	2^{20} B	**Exabyte**	EB	2014 PB	2^{60} B
Gigabyte	GB	1024 MB	2^{30} B	**Zettabyte**	YB	1024 EB	2^{70} B

2.4. Unidad de Entrada/Salida

La **Unidad de Entrada/Salida** permite la comunicación de la CPU y la Memoria Principal con el exterior: por ejemplo, impresoras, ratones y/o monitores. Entre sus **funciones** destacamos:

- **Direccionamiento**: identifica los dispositivos asignándoles una dirección única, llamada **puerto**.
- **Sincronización**: sincroniza las distintas velocidades de los dispositivos con la CPU.
- **Transferencia**: realiza el intercambio de información entre los dispositivos y la CPU.

En la actualidad, la **Unidad de Entrada/Salida** equivale al **Chipset**.

2.5. Dispositivos periféricos

Se pueden **clasificar** según la dirección de la información con la que trabajan en:

- **Entrada**: introducen información al ordenador. Ej.: teclado, ratón, escáner, micrófono, webcam…
- **Salida**: extraen información del ordenador. Ej.: monitor, impresora, altavoces…
- **Entrada/salida**: permiten introducir y extraer información. Ej.: pantalla táctil, impresoras multifunción (impresora + escáner), memorias auxiliares (discos duros, lectores de CD o DVD, memorias flash, pendrives…)

2.6. Buses

Los **Buses:** son los encargados de la interconexión entre los distintos elementos que componen el ordenador. Según la información que transporten se pueden clasificar en los siguientes **tipos**:

- **Datos**: llevan información (datos o instrucciones) a los distintos elementos
- **Direcciones**: llevan la dirección de memoria de la que se va a leer o almacenar información
- **Control**: lleva señales de control a cada elemento desde la UC, indicando cual debe ponerse en funcionamiento

2.7. Funcionamiento. Ejecución de una instrucción

Ejecución de una instrucción:

1. La UC **trae de memoria principal la próxima instrucción a ejecutar**, apuntada por el CP y se incrementa dicho contador para que apunte a la siguiente instrucción.
2. **Decodifica** e interpreta la instrucción leída para **determinar el código de operación y los datos**.
3. Genera las señales de control para **ejecutar la instrucción**.
4. Una vez obtenido el **resultado**, **se escribe en memoria**.

Actividad de clase: veamos un ejemplo y analicemos el funcionamiento haciendo clic aquí

3. Componentes lógicos: datos y software de un SI

Como se ha indicado en el punto 1, la parte **Software de un SI** son los **programas** y los **datos**.

El programa que se ejecuta cuando se arranca un ordenador es el **Sistema Operativo** (SO), el cual es el responsable del manejo y control del funcionamiento del equipo. Una vez cargado el SO ya es posible la ejecución de aplicaciones.

Cada aplicación o **programa consta de** una serie de **instrucciones**, la cuales son cargadas en MP y ejecutadas por la UCP tal y como se ha explicado en el punto anterior.

Un **programa** está formado por un conjunto de ficheros escritos en algún **lenguaje de programación** (java, C#, javascript, Python, ...).

A continuación, veamos cómo se representan internamente los datos.

3.1. Datos. Representación interna. Sistemas de numeración

Un ordenador es una máquina capaz de procesar información, o más concretamente, **datos** mediante la ejecución de una serie de **instrucciones**. Un dato es la representación de la información de manera adecuada para su tratamiento por un ordenador. Ambos tipos de información, datos e instrucciones, deben estar **representados internamente** de un modo comprensible para el computador y, por tanto, reconocible por sus circuitos electrónicos: el **sistema binario**, cuya unidad mínima es el **bit** (binary digit) y solo puede tener dos valores diferentes: **0 y 1**. Estos valores se representan empleando la tensión eléctrica: normalmente, el valor 0 indica ausencia de tensión eléctrica (0 V) y el 1 indica presencia de tensión (+5 V).

3.1.1. Sistemas de numeración

Se define un **sistema de numeración** como el conjunto de símbolos y reglas que se utilizan para representar datos numéricos.

Un sistema de numeración **en base b** utiliza, para representar los números, un alfabeto compuesto por b símbolos distintos.

Los sistemas de numeración que se van a tratar son **posicionales**, es decir, son sistemas en los que cada digito dependerá del símbolo utilizado y de la posición que éste ocupe en el número.

Los sistemas posicionales más usados son: Decimal (base 10), Binario (base 2), Octal (base 8) y Hexadecimal (base 16).

Ejemplo: En el sistema de numeración decimal, se tiene b = 10 y el alfabeto está constituido por diez signos distintos {0, 1, 2, ..., 9}.

Actividad de clase: encontrar un ejemplo de dos números distintos del sistema decimal que usan los mismos dígitos

Los sistemas posicionales están basados en el **teorema fundamental de la numeración** (TFN), que relaciona una cantidad expresada en cualquier sistema de numeración con la misma cantidad expresada en el sistema decimal.

De este modo, si la expresión de un número N en base b es:

$N_b = ... n_3 n_2 n_1 n_0 n_{-1} n_{-2} n_{-3} ...$

Entonces, su valor decimal sería:

$N_{10} = ... + n_2 * b^2 + n_1 * b^1 + n_0 * b^0 + n_{-1} * b^{-1} + n_{-2} * b^{-2} + ...$

Por tanto, para la conversión de un número en cualquier base a decimal, se utilizará este teorema.

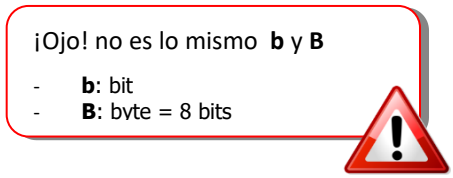
En general, los ordenadores efectúan las operaciones aritméticas utilizando una **representación** para los datos numéricos basada en el **sistema binario**. También se utilizan como **códigos intermedios** los **sistemas octal y hexadecimal**, ya que requieren menos símbolos que el binario para representar su valor y además, la conversión entre estos sistemas y el binario es directa.

Sistema Binario:

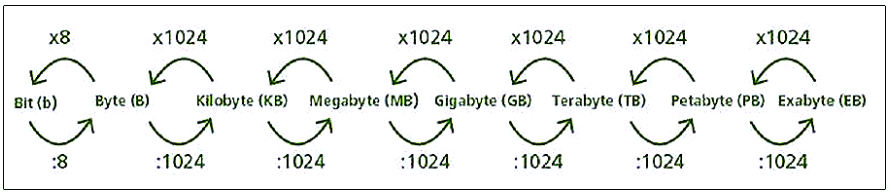
- Tiene base 2: dos signos distintos → 0 y 1. Cada uno de estos signos recibe el nombre de bit (binary digit)
- Ventajas: Se utiliza en los circuitos electrónicos para caracterizar los dos estados en que puede encontrarse.
- Inconveniente: utiliza más dígitos que cualquier otro sistema para representar una cantidad
- Múltiplos del bit (b):
  - Byte (B): 8 bits
  - Kilobyte (KB): 1024 B=2^{10}B
  - Megabyte (MB): 1024=2^{20}B
  - Gigabyte (GB): 1024 MB=2^{30}B
  - Terabyte (TB): 1024 GB=2^{40}B
  - Petabyte (PB): 1024 TB=2^{50}B
  - Exabyte (EB): 1024 PB=2^{60}B
  - Zettabyte (ZB): 1024 EB=2^{70}B
  - Yottabyte (YB): 1024 ZB=2^{80}B

¡Ojo! no es lo mismo **b** y **B**

- **b**: bit
- **B**: byte = 8 bits



Conversión de unidades



Sistema Octal:

Octal	Binario
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

- Tiene base 8: dígitos →del 0 al 7
- Cada símbolo en octal se representa por 3 símbolos binarios.
- Ventajas: La conversión entre números de la base binaria y la octal es inmediata, ya que $8=2^3$.

Sistema hexadecimal:

Hexadecimal	Binario
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Hexadecimal	Binario
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

- Base 16: 16 signos distintos →0..9A..F
- Cada símbolo en hexadecimal se puede representar mediante una combinación de 4 símbolos binarios.
- Ventajas: La conversión entre números de la base binaria y la hexadecimal es inmediata, ya que $16=2^4$.

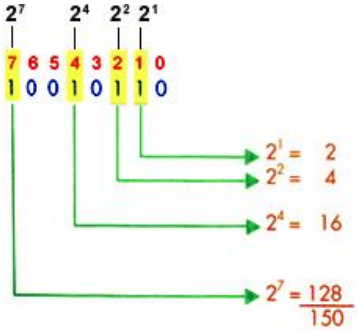
3.1.2. Cambios de base (conversiones)

Binario → decimal:

Se utiliza el TFN (teorema fundamental de la numeración), es decir, se multiplica el valor de cada dígito binario por la potencia de 2 de la posición que ocupe el dígito dentro del número:

Ejemplos:

10010110_2 a decimal:



$10010110_2 = \mathbf{150_{10}}$

$11001,01_2$ a decimal:

$11001,01_2 = 2^4+2^3+2^0+2^{-2} = 16 + 8 + 1 + 0.25 =$ **$25,25_{10}$**

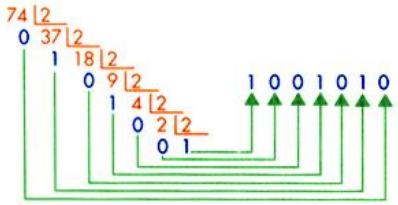
Decimal → binario:

Para convertir la **parte entera** de decimal a binario, este debe ser dividido por dos y se repite el proceso con sus cocientes hasta que el cociente tome el valor 1. La unión de todos los restos escritos en orden inverso encabezados por el último cociente nos dará el valor expresado en binario.

Para pasar la **parte fraccionaria** de decimal a binario, se multiplica por 2 y se va añadiendo al número la parte entera obtenida que se descartaría para la siguiente multiplicación.

Ejemplo: $74{,}375_{10}$ a binario

Primero pasamos a binario la parte entera:



Luego pasamos la parte fraccionaria:

0,375*2=**0**,750→Añado **0**

0,750*2=**1**,50→Añado **1**

0,5*2=**1**.0→Añado **1**

$74{,}375_{10} = \mathbf{1001010{,}011_2}$

Binario ↔ Octal:

La conversión entre la base binaria y la base octal es inmediata, ya que $8=2^3$.

Ejemplo: $\underbrace{111}_{7}\underbrace{011}_{3},\underbrace{001}_{1}{}_2 = \mathbf{73{,}1_8}$

Binario ↔ Hexadecimal:

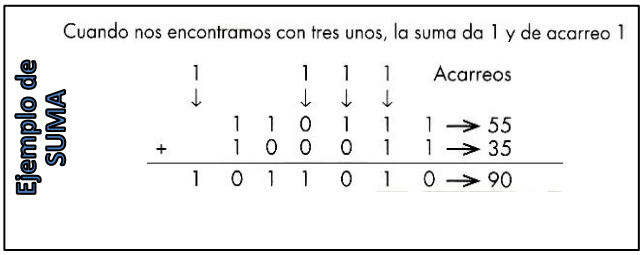
La conversión entre la base binaria y la hexadecimal es inmediata, ya que $16=2^4$.

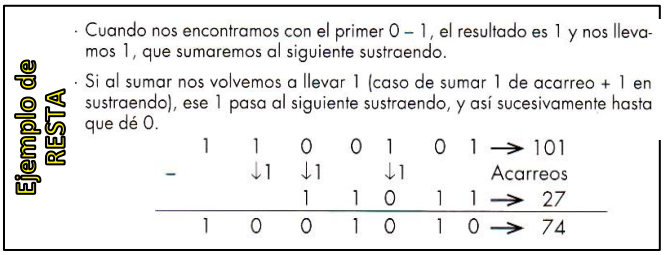
Ejemplo: $\underbrace{1101}_{D}\underbrace{0111}_{7},\underbrace{1110}_{E}\underbrace{0011}_{3}{}_2 = \mathbf{D7{,}E3_{16}}$

3.1.3. Operaciones en binario

Operaciones aritméticas

SUMA	RESTA	PRODUCTO	DIVISIÓN
0+0=0	0-0=0	0*0=0	0/0=No definido
0+1=1	0-1=1^1 (préstamo)	0*1=0	0/1=0
1+0=1	1-0=1	1*0=0	1/0=Imposible
1+1=0^1 (acarreo)	1-1=0	1*1=1	1/1=1





Operaciones lógicas

NAND: es la negación de AND.
NOR: es la negación de OR.
XOR: el resultado será 0 cuando ambos valores sean iguales y 1 cuando sean diferentes

A	B	NOT A	A AND B	A NAND B	A OR B	A NOR B	A XOR B
0	**0**	1	0	1	0	1	0
0	**1**		0	1	1	0	1
1	**0**	0	0	1	1	0	1
1	**1**		1	0	1	0	0

Ejemplo:
A = 11101100 y B = 00011101

A **AND** B = 00001100
A **OR** B = 11111101
A **NAND** B = 11110011
A **NOR** B = 00000010
A **XOR** B = 11110001

3.2. Software: SO, aplicaciones y lenguajes de programación

Según la **función** que realiza, podemos clasificar el software en:

- Software de **base** o **Sistema Operativo**: software que actúa como intermediario entre el usuario y el hardware, ocultando su complejidad con una interfaz fácil de usar. Sus objetivos son hacer cómodo el uso de la computadora y utilizar los recursos del ordenador de forma eficiente.

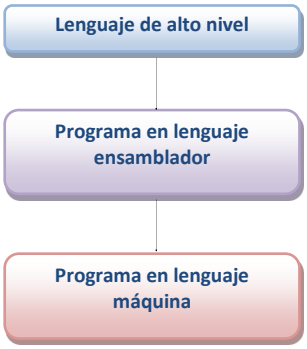
Componentes de un Sistema Informático

- Software de **aplicación** (aplicaciones): programas usados para hacer una tarea específica, como: Word, juegos, antivirus…

- Software de **programación**: programas que sirven para desarrollar otros programas. Por ejemplo, los entornos de desarrollo NetBeans, Eclipse o Microsoft Visual Studio, los cuales permiten construir programas utilizando **lenguajes de programación**. Un lenguaje de programación es un conjunto de códigos que siguiendo una sintaxis permite la introducción de órdenes en el ordenador.

  Por otro lado, los **lenguajes de programación** pueden ser:

  - De **bajo nivel**: cercanos al hardware
    - ✓ Lenguaje **máquina**: único comprensible directamente por el ordenador. Está formado por ceros y unos.
    - ✓ Lenguaje **ensamblador**: basado en el lenguaje máquina. Cada instrucción del lenguaje ensamblador es equivalente a una instrucción del lenguaje máquina, pero en vez de usar 0 y 1 se utilizan instrucciones básicas. Sus instrucciones varían dependiendo del microprocesador.

  - De **alto nivel**: más comprensible para las personas. Transportable de un ordenador a otro e independiente del hardware. Tiene que ser traducido al lenguaje máquina para que el ordenador los entienda. Ejemplos: C++, Java o C#.

Bibliografía

- Jiménez, I. (2012). “Sistemas Informáticos”. Garceta.
- Stallings, W. (2005). “Sistemas Operativos”. Pearson.