UD1-2 Introducción a los SI

SI. 1º DAM.

Representación de la información

- Un **sistema de numeración** es:
  - Conjunto de símbolos y reglas de generación que permiten construir todos los números válidos en el sistema.
  - Estas reglas son diferentes para cada sistema de numeración.
  - Existe una regla común a todos: para construir números válidos en un sistema de numeración determinado sólo se pueden utilizar los símbolos permitidos en ese sistema.

Representación de la información

- Ejemplo

125_{10} ➡ Válido en el sistema decimal.

$12A_{10}$ ➡ No es válido en el sistema decimal ya que utiliza el símbolo A.

Representación de la información

- Los sistemas de numeración se clasifican por su **base**.
- **Base:** indica el número de dígitos que utiliza el sistema de numeración para representar un valor.

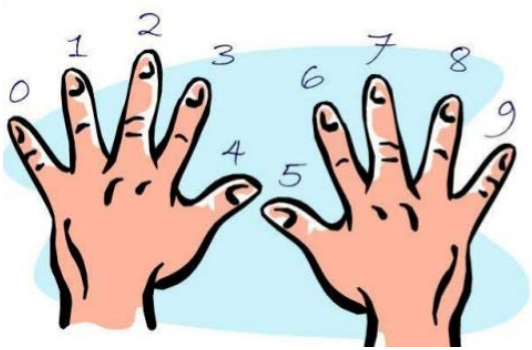
Sistema decimal (base 10)
Sistema binario (base 2)
Sistema octal (base 8)
Sistema hexadecimal (base 16)

El ordenador trabaja en base 2. La base 16 se utiliza de cara al programador, para compactar el número resultante de utilizar base 2.

Representación de la información

- **Sistema decimal**
  - Está compuesto por 10 símbolos **{0,1,…,9}**, por tanto la base es 10.
  - Ejemplo:

El número 1492,36 en decimal, puede expresarse como:
$1493{,}36_{10}=1*10^4+4*10^3+9*10^2+2*10^1+3*10^0+3*10^{-1}+6*10^{-2}$



Representación de la información

- **Sistema binario**
  - Tan solo utiliza dos dígitos **0** y **1**.

    El número 1100110 estaría definido en el sistema binario, mientras que el 1102 no.

  - El valor de cada dígito depende de la posición que tiene en el número.
  - Ejemplo

$$110101_2 = 1*2^5+1*2^4+0*2^3+1*2^2+0*2^1+1*2^0=32+16+4+1 = 53_{10}$$

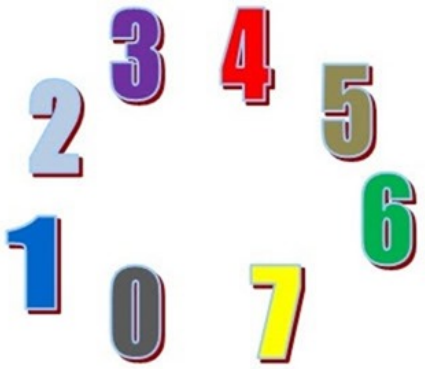
Representación de la información

- A cada dígito en el sistema de numeración binario se le denomina **BIT (Binary Digit)**.
- Estos bits se agrupan cada 8, 16, 32 formando las denominadas palabras. A 8 bits se le denomina **BYTE**.

Representación de la información

- **Sistema octal**
  - Es un sistema de numeración en base 8.
  - Utiliza los dígitos de **0 a 7**.
  - Ejemplo

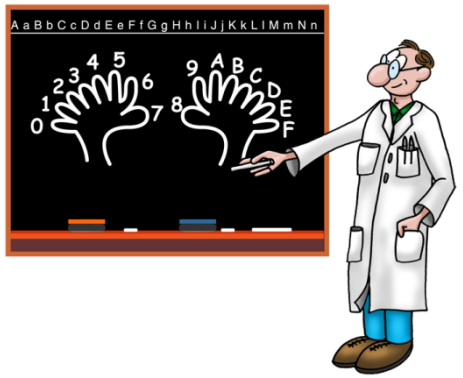
$5768_8 = 5*8^3+7*8^2+6*8^1+8*8^0 = 3064_{10}$



Representación de la información

- **Sistema hexadecimal**
  - Surgió para compactar la información binaria.
  - Un dígito hexadecimal representa 4 dígitos binarios.
  - Los dígitos que utiliza para representar un número son:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, E, F



Conversión de bases

- **Conversión de binario a octal**
  - Se agrupan los bits de 3 en 3
- Ejemplo

Pasar 010011_2 a octal

1. Separamos el número binario en grupos de 3 bits

$$010011_2 = 010 \mid 011$$

2. Cada binario de 3 dígitos se convierte a decimal

$$010 = 2$$
$$011 = 3$$

3. Unimos los valores consiguiendo así el valor octal

$$010011_2 = 23_8$$

Conversión de bases

- **Conversión de binario a hexadecimal**
  - Se agrupan los bits de 4 en 4
- Ejemplo

Pasar 11110011_2 a hexadecimal

1. Separamos el número binario en grupos de 4 bits

$$11110011_2 = 1111 \mid 0011$$

2. Cada binario de 4 dígitos se convierte a decimal

$$1111 = 15$$
$$0011 = 3$$

3. Unimos los valores consiguiendo así el valor hexadecimal (los número del 10-15 se sustituyen por las letras correspondientes)

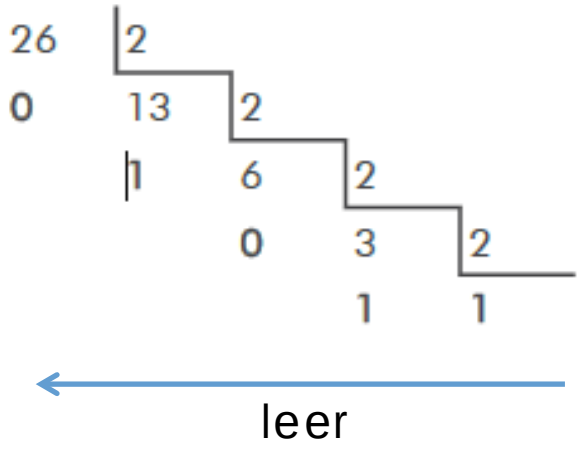
$$11110011_2 = F3_{16}$$

Conversión de bases

- **Conversión de decimal a cualquier base**
  - Método de divisiones sucesivas
- Ejemplo

Pasar 26_{10} a binario

$26_{10} = 11010_2$



Conversión de bases

- **Conversión de cualquier base a decimal**
  - Multiplicamos el dígito por la base elevada a la posición, empezando a contar de derecha a izquierda.
- Ejemplos

Pasar de binario a decimal

$100110_2 = 1*2^5+0*2^4+0*2^3+1*2^2+1*2^1+0*2^0 = 32+4+2 = 38_{10}$

Pasar de octal a decimal

$147_8 = 1*8^2+4*8^1+7*8^0 = 64+32+7 = 103_{10}$

Pasar de hexadecimal a decimal

$C3E_{16} = 12*16^2+3*16^1+14*16^0 = 3072+48+14 = 3134_{10}$

Ejercicio_01_SistNumeración

Prefijos binarios

Nombre	Abreviatura	Factor binario	
kilo	k	2^{10} =	1.024
mega	M	2^{20} =	1.048.576
giga	G	2^{30} =	1.073.741.824
tera	T	2^{40} =	1.099.511.627.776
peta	P	2^{50} =	1.125.899.906.842.624
exa	E	2^{60} =	1.152.921.504.606.846.976
zetta	Z	2^{70} =	1.180.591.620.717.411.303.424
yotta	Y	2^{80} =	1.208.925.819.614.629.174.706.176

Representación interna de la información

- Representar (o **codificar**) un número significa expresarlo en forma binaria.
- La representación de números en un ordenador es necesaria para que éste pueda almacenarlos y manipularlos.
- El problema es que un número matemático puede ser infinito (tan grande como se desee), pero la representación de un número en un ordenador debe ocupar un número de bits predeterminado.

Representación interna de la información

- **Representación de un número natural**
  - Un número natural es un número entero positivo o cero.
  - La elección de la cantidad de bits a utilizar para representarlo depende del intervalo de números que se utilizarán.
  - Para codificar los números naturales entre 0 y 255, todo lo que se necesita son 8 bits (un byte) ya que $2^8 = 256$.

Representación interna de la información

- **Representación de un número entero**
  - Los números enteros abarcan los números naturales, los número negativos y el cero.
  - El número se debe codificar de manera que se pueda distinguir si es positivo o negativo y de forma que siga las reglas de adición.

Representación interna de la información

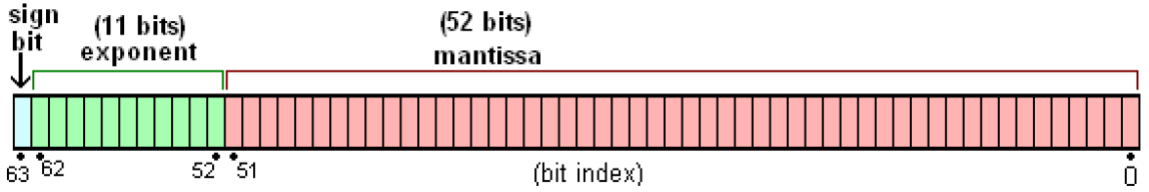
- **Representación de un número real**
  - **Coma fija**
    - La posición de la coma (coma decimal en base diez) es la que fija la potencia de la base por la que hay que multiplicar el dígito correspondiente.

$$10101{,}110_2 = 1*2^4+0*2^3+1*2^2+0*2^1+1*2^0+1*2^{-1}+1*2^{-2}+0*2^{-3} = 21{,}75$$

Ejercicio 7. h), i)

Representación interna de la información

- **Representación de un número real**
  - **Coma flotante**
    - Son grupos de bytes en los que una parte se emplea para guardar las cifras del número (mantisa) y otra para indicar la posición del punto flotante (exponente). Esto permite trabajar con números de muy elevado tamaño y con una mayor o menor precisión en función de los bits empleados para codificar la mantisa.



Representación interna de la información

- **BCD**
  - Consiste en emplear cuatro bits para codificar los dígitos del 0 al 9 (desperdiciando las seis combinaciones que van de la 1010 a la 1111).
  - Ventaja: la simplicidad de conversión a/de base 10, que resulta inmediata.

Ejercicios 6 y 12

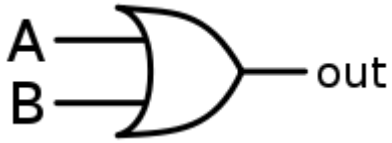
Operaciones lógicas

- Las puestas lógicas son la unidad básica sobre la que se diseña un circuito integrado y pueden tener una o varias entradas que se convertirán en una salida.

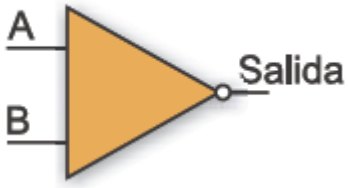
- Existen diversos tipos de puertas lógicas (para su estudio supondremos puertas de dos entradas):
  - **AND:** Devolverá como salida una tensión superior a cero en caso de que en ambas entradas el valor de tensión sea también superior a cero.

Operaciones lógicas

- **OR:** Devolverá como salida una tensión superior a cero siempre que alguna de las entradas o ambas tengan una tensión superior a cero.



- **NOT:** Invierte el valor de tensión de la entrada, es decir, si a la entrada aplicamos tensión, a la salida la tensión será cero y viceversa.



Operaciones lógicas

- **Tablas de verdad de las puertas lógicas:**

AND

Equivale al producto lógico y su tabla de verdad es la siguiente:

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

Operaciones lógicas

- **Tablas de verdad de las puertas lógicas:**

OR

Equivale a la suma lógica. Su tabla de verdad es la siguiente:

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

Operaciones lógicas

- **Tablas de verdad de las puertas lógicas:**

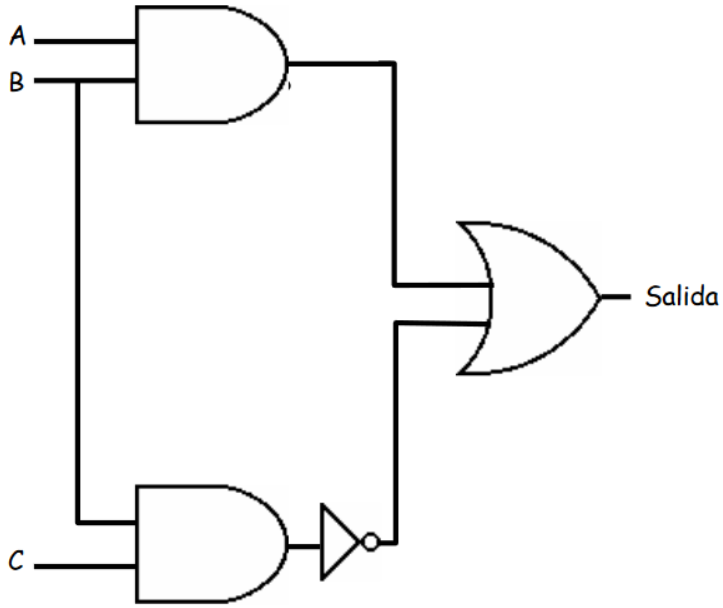
NOT

Es el operador de negación, dispone de la siguiente tabla de verdad

A	not A
0	1
1	0

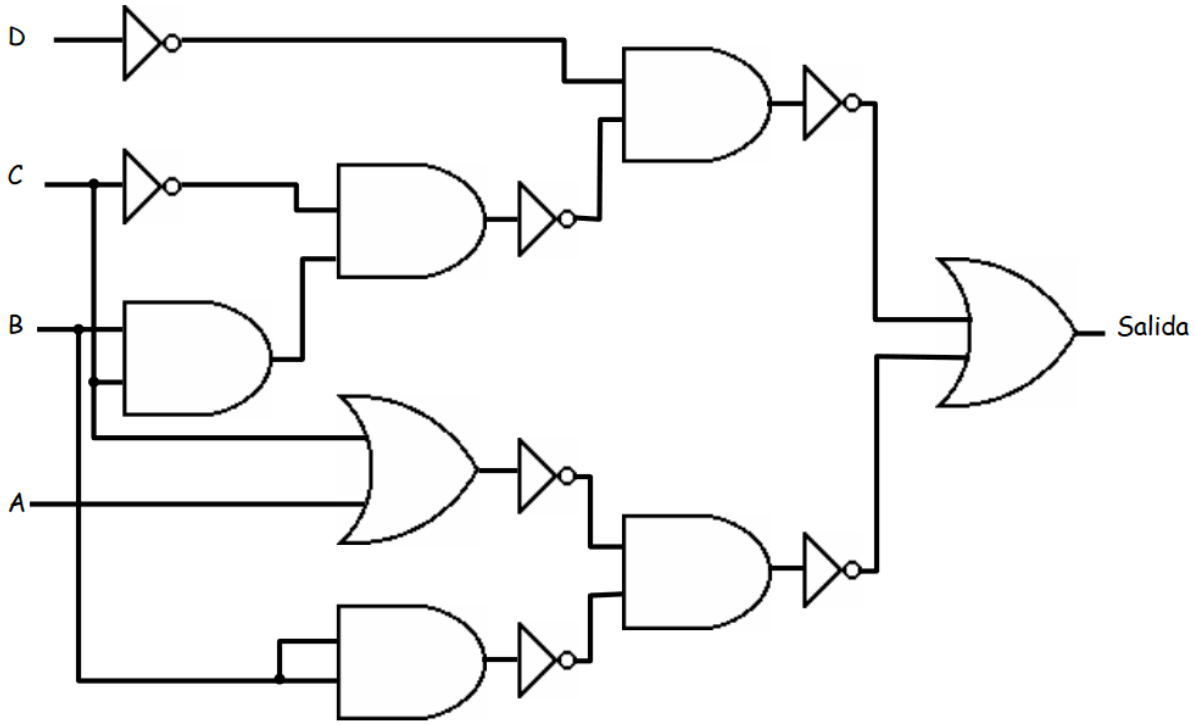
Operaciones lógicas

- Escribir la tabla de verdad del siguiente circuito compuesto por las siguientes puertas lógicas:



Operaciones lógicas

- Escribir la tabla de verdad del siguiente circuito compuesto por las siguientes puertas lógicas:



Datos alfanuméricos

- Internamente el ordenador no trabaja con letras y números, sino que trabaja con dígitos binarios o bits.
- Si, por ejemplo, introducimos a través del teclado una letra, el ordenador lo almacena en la memoria como una combinación de unos y ceros.
- Para después poderlo leer y entenderlo necesita utilizar un código que le indique a qué carácter corresponde ese conjunto de bits.

Datos alfanuméricos

- Esa codificación se realiza mediante los siguientes códigos, también llamados códigos de entrada/salida:
  - **Código BCD** (Binary Coded Decimal)
    - Consiste en emplear cuatro bits para codificar los dígitos del 0 al 9.
    - Se desperdician las seis combinaciones que van de la 1010 a la 1111.
    - La ventaja es la simplicidad de conversión a/de base 10, que resulta inmediata.

Datos alfanuméricos

- **Código ASCII**(American Standard Code for Information Interchange)
  - Asigna a cada carácter un valor numérico.
  - Se trata de un código de 7 bits con capacidad para 128 símbolos que incluyen todos los caracteres alfanuméricos del inglés, con símbolos de puntuación y algunos caracteres de control.
  - Posteriormente se amplió a 8 bits para incluir caracteres de otros idiomas diferentes al inglés.

Datos alfanuméricos

- **UNICODE** (del inglés 'universal' y 'code')
  - Es el estándar más utilizado en la actualidad.
  - Su objetivo es proporcionar el medio por el cual un texto en cualquier forma e idioma pueda ser codificado para el uso informático.