# Representación de la información

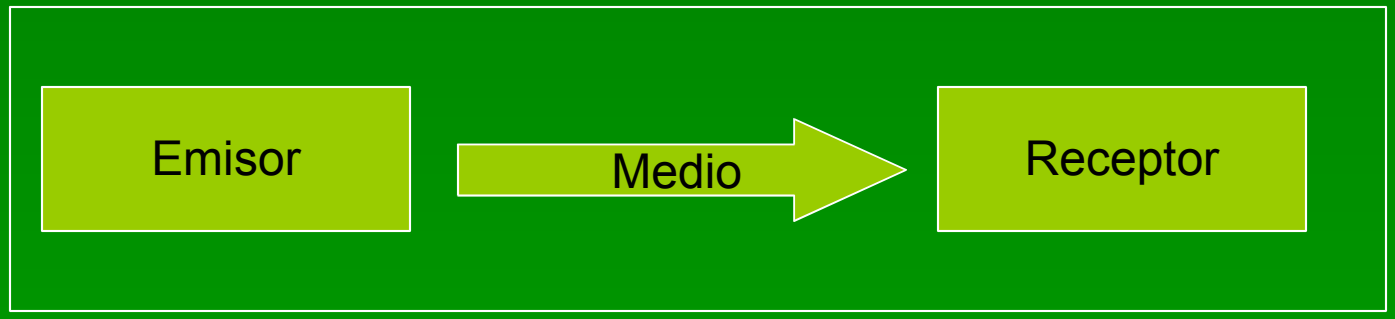
# Contenidos

1. Informática e información.
2. Sistemas de numeración.
3. Sistema binario, operaciones aritméticas en binario,
4. Sistemas octal y hexadecimal.
5. Conversiones entre sistemas.
6. Representación interna de la información: números enteros.
7. Magnitud y signo, complemento a 1, complemento a 2 y representación en exceso o sesgada.
8. Representación de números reales: coma fija y coma flotante.
9. Representación de datos alfabéticos: códigos BCD, ASCII, EBCDIC y Unicode.

# 1. Informática e información

- **La informática** es la ciencia que estudia el tratamiento automático y racional de la información, con el fin de obtener de ella la máxima utilidad.
- **Información** es sinónimo de conocimiento, de noticia, de datos, etc. Es la representación de hechos, objetos, valores, ideas, etc., que permite la comunicación entre emisor y receptor y la adquisición del conocimiento

- Un **sistema de comunicación** está formado por los siguientes elementos básicos:




  - Emisor, fuente o transmisor: es el que genera o emite la información
  - Receptor: es el que recibe la información
  - Medio o canal: vía de transmisión de la información

- Emisor y receptor pueden intercambiar sus papeles o realizar ambos papeles simultáneamente.
- La transmisión de información entre el ser humano y el ordenador es una comunicación en la que:
  - el emisor es una persona

intercambiables

  - el receptor es el ordenador
  - el medio o canal son los periféricos de entrada/salida del ordenador ( periféricos)

- La transmisión de información entre el ser humano y la computadora puede hacerse:
  - Mediante caracteres alfanuméricos
  - Mediante sonidos
  - Mediante vídeos
  - Mediante gráficos e imágenes
  - En general, cualquier tipo de datos enviados por un periférico
- En cada caso el canal es diferente y diferente la representación del emisor, receptor y canal. Ello implica una traducción o decodificación cuando los códigos utilizados por el emisor, el canal y el receptor son distintos

# Simbología y codificación

- Dado un conjunto idóneo de símbolos ( por ejemplo un vocabulario) y establecidas las reglas (por ejemplo una gramática), los símbolos pueden ser manejados como sustitutos de los objetos que representan. A la asociación entre objeto y símbolo se le denomina **codificación**
- **Ejemplos:**
  - El lenguaje
  - El alfabeto Morse
  - Los jeroglíficos egipcios
- **Código** es el conjunto de condiciones y convenios que permiten transformar la información de una representación a otra, es decir

**CODIGO = REGLAS + SIMBOLOS**

- La representación interna de la información en los ordenadores se da en forma de impulsos eléctricos en forma de **señales biestables:**
  - Activado-desactivado
  - Encendido-apagado
  - Abierto-cerrado
  - Tensión-no tensión
- Es decir, hay impulso o no lo hay
- Por ello utilizaremos el **código binario:**
  - 1 hay impulso
  - 0 no hay impulso

# 2. Sistemas de numeración

- **Sistema de numeración** es el conjunto de símbolos utilizados para la representación de las cantidades, así como las reglas de la representación.
- **Base** es el número de símbolos que utiliza un sistema de numeración y se caracteriza por ser el coeficiente que determina cuál es el valor de cada dígito dependiendo de su posición

# A. Teorema fundamental de la numeración

$$N_i = \sum_{i=-d}^{n} (digito)_i * (base)_i$$

- i: posición respecto a la coma. Para los dígitos a la derecha, la i es negativa, empezando en -1; para los de la izquierda, es positiva empezando en 0
- d : número de dígitos a la derecha de la coma
- n : número de dígitos a la izquierda de la coma -1
- Dígito : cada uno de los componentes del número
- Base : base del sistema de numeración (Pág.. 10 y 11)

# A. El sistema binario

- El **sistema binario** utiliza sólo dos dígitos **0 y 1** para representar las cantidades. Su base es 2. Cada dígito se llama **bit** (binary digit)
- El valor de los bits viene determinado por una potencia de base 2 o **peso** que depende de la posición que ocupa el bit (Pág. 11)

## Conversión de un número decimal a binario

- Para convertir un número decimal a binario se divide sucesivamente el número decimal y los cocientes que se van obteniendo por 2 hasta que el cociente sea menor que 2. La unión del último cociente y todos los restos obtenidos escritos en orden inverso será el número expresado en binario
- Ejemplo: el número decimal $25_{(10}$ será $11001_{(2}$ en base dos

# Cantidad de bits para representar un número decimal

| Número decimal | Número de bits |
|---|---|
| Menor que 2 | 1 |
| Menor que 4 | 2 |
| Menor que 8 | 3 |
| Menor que 16 | 4 |
| Menor que 32 | 5 |
| Menor que 64 | 6 |
| Menor que 128 | 7 |
| … | … |
| Menor que $2^n$ | n |

## Conversión de una fracción decimal a binario

- Para convertir una fracción decimal a binario multiplicamos sucesivamente la parte fracción por 2 hasta que dé 0 como resultado. La parte entera de cada multiplicación formará los bits del número binario

## Conversión de una fracción binaria a decimal

- Para convertir una fracción decimal a binaria se utiliza el teorema fundamental de la numeración

- Casos prácticos Pág.. 13

# Suma y resta en binario

- **Suma binaria**: En binario, la cifra más alta es el 1, por lo tanto, cuando en la suma encontramos dos unos resulta 1 + 1 = 10, entonces se deja el 0 y se arrastra el 1 para ser sumado a la izquierda. Debido al 1 de arrastre pueden juntarse tres unos, con lo que obtenemos 1 + 1 + 1 = 11 luego dejaremos un 1 y arrastramos otro 1 a la izquierda.

```
  1 1 1
+ 10110
  11100
--------
 110010
```

- **Resta binaria**:

```
Decimal
  54583055
- 19947053
----------
  34636002

Binario
  11010011
- 01101010
----------
  01101001
```

## Multiplicación binaria

- Se realiza como la multiplicación decimal, con la diferencia de que luego se hacen las sumas en binario

| | |
|---|---|
| 0 * 0 = 0 | 1 * 0 = 0 |
| 0 * 1 = 0 | 1 * 1 = 1 |

Errata Pág. 15

## División binaria

- Se realiza como la división decimal, pero las multiplicaciones y restas internas se hacen en binario

# A. El sistema octal

- El **sistema octal** tiene como base de numeración 8; utiliza pues 8 símbolos: 0, 1, 2, 3, 4, 5, 6 y 7
- Es un **sistema posicional**, es decir, un mismo dígito tiene distinto valor según la posición que ocupe
- **Conversión de un número decimal a octal:** Lo más sencillo son las divisiones sucesivas
- **Conversión de un número octal a decimal:** utilizaremos el sistema fundamental de la numeración
- **Conversión de una fracción decimal a octal:** método de multiplicaciones sucesivas, lo único que cambia es la base
- Conversión de una fracción octal a decimal: utilizaremos el sistema fundamental de la numeración
- Caso práctico pág. 17

# A. El sistema hexadecimal

- El **sistema hexadecimal** tiene como base de numeración 16; es decir, utiliza 16 símbolos para representar las cantidades, que son: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F, donde

| Símbolo | Valor asignado |
|---|---|
| A | 10 |
| B | 11 |
| C | 12 |
| D | 13 |
| E | 14 |
| F | 15 |

- Es un **sistema posicional**
- **Conversión de un número decimal a hexadecimal:** Lo más sencillo son las divisiones sucesivas y para restos entre 10 y 15 utilizaremos las letras correspondientes
- **Conversión de un número hexadecimal a decimal:** utilizaremos el sistema fundamental de la numeración
- **Conversión de una fracción decimal a hexadecimal:** método de multiplicaciones sucesivas, lo único que cambia es la base
- Conversión de una fracción hexadecimal a decimal: utilizaremos el sistema fundamental de la numeración
- Ejemplos pág. 18

# 1. Representación interna de la información

- La Unidad Aritmético Lógica (UAL) es la parte del ordenador que se encarga de procesar todas las operaciones lógicas y de cálculo
- Es en las operaciones aritméticas donde es necesario efectuar traducción entre el formato de entrada/salida y el formato interno de representación de datos

- **Octeto, carácter o byte**: es la agrupación de 8 bits, el tamaño típico de información; con él se puede codificar el alfabeto completo (ASCII estándar)
- **Palabra**: tamaño de información manejada en paralelo por los componentes del sistema. El tamaño de la palabra puede ser el byte o un múltiplo del byte. Cuanto mayor sea el tamaño de la palabra, mayor será la potencia y precisión de cálculo de un ordenador

# A. Representación de números enteros

- El grupo del los números **enteros** comprende los números positivos ( números naturales), el cero y los números negativos
- Si la longitud de la palabra es de n bits, podremos representar $2^n$ números enteros positivos distintos y si reservamos un bit para el signo podremos representar $2^{n-1}$

# Relación entre bases binaria, octal y hexadecimal

- **Conversión hexadecimal a binario**, se sustituye cada dígito hexadecimal por su representación binaria utilizando 4 dígitos
- **Conversión binario a hexadecimal,** se agrupan los dígitos binarios de 4 en 4 desde la coma a la izquierda y desde la coma a la derecha
- **Conversión octal a binario,** se sustituye cada dígito octal por su representación binaria utilizando 3 dígitos
- **Conversión binario a octal,** se agrupan los dígitos binarios de 3 en 3 desde la coma a la izquierda y desde la coma a la derecha
- **Conversión de hexadecimal a octal,** primero se pasa de hexadecimal a binario, y luego de binario a octal
- **Conversión de octal a hexadecimal,** primero se pasa de octal a binario, y luego de binario a hexadecimal
- **Casos prácticos pág. 19**

## Signo y magnitud

- En esta representación, el bit situado más a la izquierda representa el signo, y su valor será **0 para el positivo y 1 para el negativo**
- **El rango de números representable es:**

$$-(2^{n-1}-1) \leq X \leq (2^{n-1}-1)$$

- **Calcular el rango para 8 bits**
- **El inconveniente de este sistema es que el cero tiene dos representaciones: +0 y -0**

## Complemento a 1

- El complemento a 1 utiliza el bit de más a la izquierda para el signo
- Los números positivos se representan con signo y magnitud
- Los números negativos se obtienen complementando todos los dígitos, es decir, **cambiando 0 por 1 y 1 por 0**
- Tiene el inconveniente de que el 0 tiene dos posibles representaciones

  +0 ⇨ 00000000 C a 1 ⇨ 11111111 (-0)

## Complemento a 2

- El complemento a 2 utiliza el bit de más a la izquierda para el signo
- Los números positivos se representan con signo y magnitud
- Los números negativos se obtienen de la siguiente forma:
  - Se hace el complemento a 1 del correspondiente número positivo
  - Al resultado obtenido se le suma 1, despreciando el acarreo final si existe
- La principal ventaja es que el cero tiene una representación única

## Representación sesgada o exceso a $2^{n-1}$

- No utiliza bit para el signo
- Todos los bits representarán un valor que corresponde al número representado más el exceso
- Para n bits viene dado por la fórmula $2^{n-1}$
- El número resultante será siempre positivo y se representará en binario natural
- El 0 tiene única representación, porque

$$0 + 2^{n-1} = -0 + 2^{n-1}$$

- Ejemplo pág. 22

## Suma en complemento a 1

- Dos números se suman igual que en binario
- Si aparece un acarreo en los bits de más a la izquierda, éste se suma la resultado
- Las restas no existen; lo que se hace es convertir el número que se resta a C a 1 y sumarlo al otro.
- Sólo se hacen sumas

## Suma en complemento a 2

- Dos números se suman igual que en binario, pero el último acarreo, si existe, se desprecia

## Error de desbordamiento

- Si se realizan sumas de números del mismo signo en C a 1 y en C a 2, y disponemos de un número determinado de bits, ocurrirá un error de desbordamiento (**Overflow**) y el resultado aparecerá con el signo contrario al de los sumandos

# A. Representación de números reales

- Hablamos de **números reales**, cuando nos referimos a cantidades con **parte entera** y **parte fraccionaria**
- Teniendo en cuenta que siempre dispondremos de un número finito de bits (**palabra**), los números reales deberán ser truncados la mayor parte de las veces
- El tamaño de la palabra depende del ordenador que se utilice: a mayor tamaño, mayor precisión en el cálculo

## Coma o punto fijo

- En esta representación, la coma ocupa una posición fija
- Existen tres formas de representar los números en coma fija:
  - **Binario puro**
  - **Decimal desempaquetado**
  - **Decimal empaquetado**

- **Decimal desempaquetado**
  - En este sistema cada dígito de la cantidad a representar ocupa un byte, de forma que:

    1111 XXXX

    Bits de zona (F) Bits de dígito
  - El cuarteto de la izquierda del último dígito representa el signo, 1100 para el + y 1101 para el – (C y D respectivamente en hexadecimal)
  - Ejemplo:

    1111 0001 1111 0110 1111 0111 1100 1000

    F 1 F 6 F 7 + 8

    es decir +1678

- **Decimal empaquetado**
  - Cada dígito se representa en un cuarteto sin los bits de zona, excepto el primer dígito de la derecha que lleva a la derecha los bits del signo
  - 1100 para el + y 1101 para el – (C y D en hexadecimal)
  - Ejemplo:

0000 0001 0110 0111 1000 1100

0 1 6 7 8 +

0 1 6 7 8 C

## Coma o punto flotante

$$\text{Número} = \text{Mantisa} * \text{Base}^{\text{Exponente}}$$

- Donde la **mantisa** no tiene parte entera y
- El primer dígito a la derecha de la coma es significativo, es decir, distinto de 0; excepto en la representación del número 0
- Esta notación se conoce como **Notación Científica o Exponencial**
- En esta notación el ordenador debe almacenar:
  - La mantisa y su signo
  - La base
  - El exponente

- **Simple precisión**

  - El número se almacena en una palabra en binario puro, asignándose un número de bits al exponente y el resto a la mantisa. La base es 2

- **Doble precisión**

  - Igual que en simple precisión pero utilizando una doble palabra para almacenar los números
  - Si la palabra es de 32 bits, este método utilizará 64 bits (doble palabra). Si consideramos un bit para el signo, más los 8 bits para el exponente, la mantisa ocupará 55

| 0 | 00000000 | 00000000……000 |
|---|---|---|
| Signo | Exponente | Mantisa (55 bits) |
| 63 | 62….55 | 54 53 ……..10 |

# A. Representación de datos alfabéticos y alfanuméricos, códigos en Entrada/Salida

- Los códigos de E/S permitirán traducir la información que nosotros entendemos a una representación que la máquina pueda interpretar y procesar
- Los datos entran y salen del ordenador a través de los periféricos de entrada y salida respectivamente
- Códigos estándares de entrada/salida: BCD, EBCDID, ASCII y Unicode

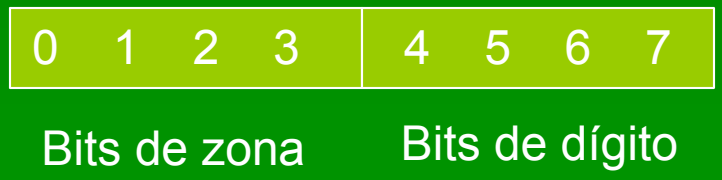
## BCD

- BINARY CODED DECIMAL o decimal codificado en binario o CBD (decimal binario codificado)
- BCD divide cada octeto en dos mitades o cuartetos, cada uno de los cuales almacena en binario puro una cifra

| Digito decimal | BCD natural |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |

# EBCDIC

- Código BCD_extendido de caracteres decimales codificados en binario para el intercambio de información
- Representa los caracteres alfanuméricos con 8 bits
- Podemos representar hasta $2^8 = 256$ caracteres distintos
- Este código está dividido en dos zonas



- Los bits de zona indican la representación de los distintos caracteres; así pues dos caracteres distintos pueden tener los mismos bits de dígito, pero distintos bits de zona

| **Bits de zona** | **Caracteres representados**<br>**Bits de dígito** |
|---|---|
| 1111 ( F ) | Los números |
| 1100 ( C) | Mayúsculas de A..I |
| 1101 ( D ) | Mayúsculas de J..R |
| 1110 ( E ) | Mayúsculas de S..Z |
| 1000 ( 8 ) | Minúsculas de a..i |
| 1001 ( 9 ) | Minúsculas de j..r |
| 1010 ( A ) | Minúsculas de s..z |
| 01-- | Caracteres especiales |
| 00-- | No tienen asignación |

# ASCII

- Código estadounidense estándar para el intercambio de información
- Utiliza grupos de 7 bits por carácter, es decir, puede representar hasta $2^7$ =128 caracteres distintos
- El ASCII extendido utiliza 8 bits por carácter, lo que añade 128 caracteres más
- Es el código más extendido y el utilizado en MS-DOS, Windows y Unix/Linux

## Unicode

- El Unicode es una norma de codificación universal para ordenadores con sistema operativo Windows NT y navegadores como Internet Explorer y Netscape 4.0 y siguientes
- Utiliza 16 bits, por lo que puede representar hasta $2^{16}$ caracteres distintos, lo que permite representar cualquier carácter de cualquier idioma