



1



Universidad Nacional Autónoma de México
Facultad de Ingeniería
Computación para Ingenieros
Tema 4
MANEJO INTERNO DE DATOS



2

4. Manejo interno de datos

Objetivo: Describir cómo se almacenan los datos en los distintos medios de un sistema de cómputo, así como, manipular los datos para minimizar los diferentes errores que pueden suscitarse en su almacenamiento.



4. Manejo interno de datos

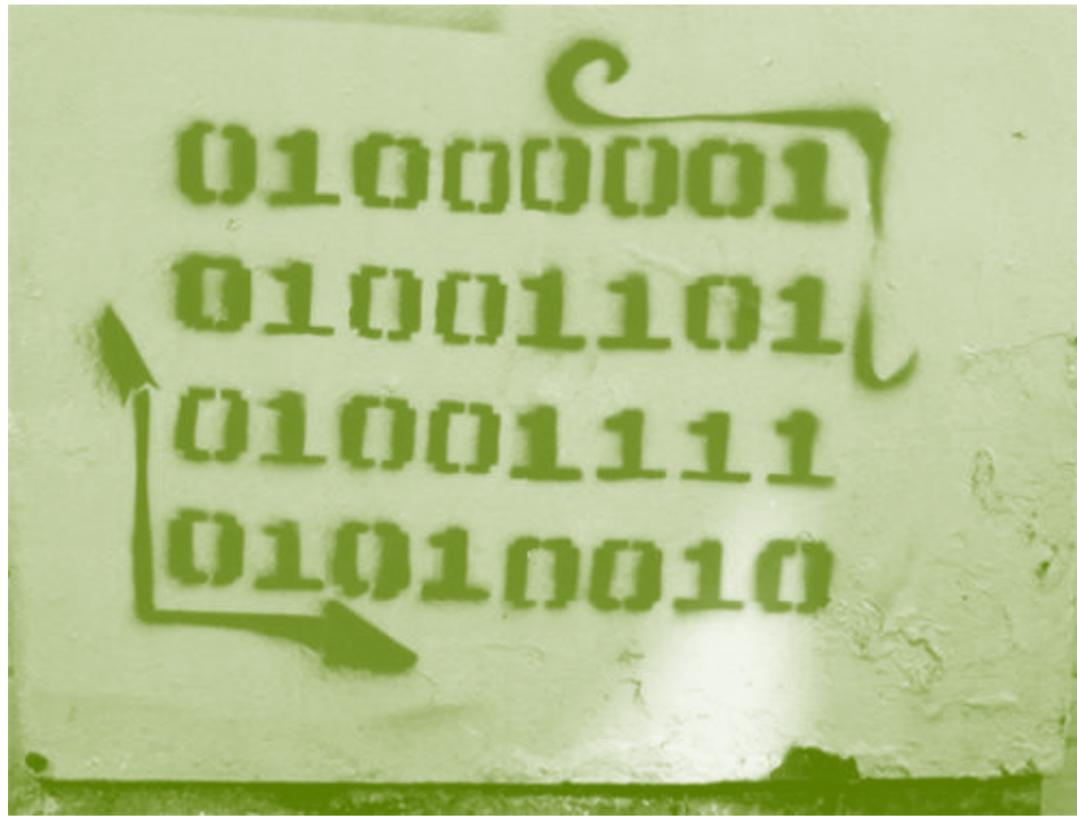
4.1 Dispositivos y unidades de almacenamiento: bit, byte y palabra.

4.2 Representación de datos tipo texto (códigos ASCII y EBCDIC).

4.3 Representación numérica: magnitud y signo, y complemento a r.

4.4 Tipos de errores en la manipulación de cantidades.

4.5 Formatos de archivos de texto, imágenes, video, audio, etc.



4.1 Dispositivos y unidades de almacenamiento



4.1 Dispositivos y unidades de almacenamiento

Un dispositivo de almacenamiento es elemento electromecánico o electrónico, capaz de almacenar información.

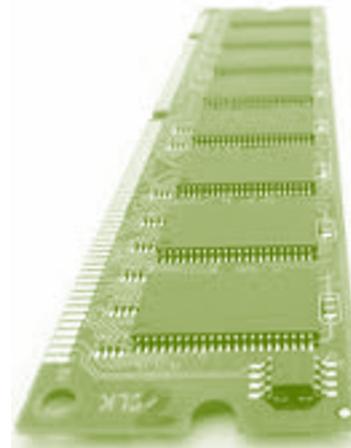
Muchos dispositivos tienen una interfaz electrónica especial, llamada controlador. Un controlador ayuda a comunicar un dispositivo electrónico con un equipo de cómputo.



6

Almacenamiento Primario

Es un dispositivo electrónico que permite guardar información, siempre y cuando posea un suministro constante de energía eléctrica. En estos dispositivos se encuentran las instrucciones y datos al momento de ejecución.





Almacenamiento Secundario

Es un dispositivo electrónico o electromecánico que permite almacenar la información en forma permanente. Debido a que operan a velocidades electromecánicas, son más lentos que los dispositivos de almacenamiento primario.





Lenguaje máquina y sistema binario

El lenguaje con el que trabaja el procesador se denomina *lenguaje máquina*. Cada procesador tiene su propio lenguaje.

Para efectuar una operación, el procesador necesita una secuencia de señales eléctricas almacenadas como unos y ceros (*sistema binario*) en la memoria.

Una y solo una secuencia de señales (conjunto de bits) realiza una determinada operación.



Las instrucciones que el procesador ejecuta constan de dos partes: código de operación y código de operandos.



El código de operación indica la instrucción a realizar. El código de operandos indican la(s) dirección(es) de memoria en la que se encuentra el operando, sobre el/(los) que se aplicará la operación.



Para programar en lenguaje se debe conocer la arquitectura física de la computadora (registros, palabras de memoria, etc.).

La estructura del lenguaje máquina está totalmente adaptada a los circuitos de la computadora y muy alejada del lenguaje de alto nivel.



Conversión de números

Como ya se mencionó, las computadoras ocupan sistema base 2 para representar instrucciones y cantidades.

Empero, la representación de cantidades en sistema binario utiliza demasiados dígitos, por lo que se optó por manejar dichas cantidades mediante las bases octal y hexadecimal.



- **El sistema binario representa cantidades utilizando dos dígitos: 0 y 1.**
- **El sistema octal representa cantidades utilizando ocho dígitos: 0, 1, 2, 3, 4, 5, 6, 7.**
- **El sistema hexadecimal representa cantidades utilizando dieciséis dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.**



Conversión de números enteros de base *diez* a base *b*

Un número entero en base diez puede ser representado de la siguiente manera:

$$d_n d_{n-1} \dots d_2 d_1 d_0$$

donde, *d* hace referencia a un número decimal (dígitos de 0 a 9) y *n* hace referencia a la posición del número a partir del punto decimal (parte entera).



Conversión de números enteros de base *diez* a base *b*

Para representar un número entero base 10 a base *b*, es necesario dividir el número decimal (dividendo) entre la base (divisor) hasta que el resultado (cociente) sea cero. Los residuos que se van obteniendo con la división forman el número convertido.



Ejemplo 4.1

Convertir el número 1992_{10} a base 2:

1992	2
996	0
498	0
249	0
124	1
62	0
31	0
15	1
7	1
3	1
1	1
0	1

$$\begin{array}{r} 996 \\ 2 \overline{) 1992} \\ 0 \\ \hline 62 \\ 2 \overline{) 124} \\ 0 \\ \hline 3 \\ 2 \overline{) 7} \\ 1 \end{array}$$

$$\begin{array}{r} 498 \\ 2 \overline{) 996} \\ 0 \\ \hline 31 \\ 2 \overline{) 62} \\ 0 \\ \hline 1 \\ 2 \overline{) 3} \\ 1 \end{array}$$

$$\begin{array}{r} 249 \\ 2 \overline{) 498} \\ 0 \\ \hline 15 \\ 2 \overline{) 31} \\ 1 \\ \hline 0 \\ 2 \overline{) 1} \\ 1 \end{array}$$

$$\begin{array}{r} 124 \\ 2 \overline{) 249} \\ 1 \\ \hline 7 \\ 2 \overline{) 15} \\ 1 \end{array}$$



Ejemplo 4.1

1992	2
996	0
498	0
249	0
124	1
62	0
31	0
15	1
7	1
3	1
1	1
0	1

El resultado de convertir el número 1992_{10} a base 2 es:

$$1992_{10} = 11111001000_2$$



NOTA. Los lenguajes de programación poseen una operación que se llama módulo cuya expresión es `%` y regresa como resultado el residuo de una división, es decir:

$$\begin{array}{r} 996 \\ 2 \overline{) 1992} \\ 0 \end{array}$$

La operación $1992 / 2$ da como resultado 996. La operación $1992 \% 2$ da como resultado 0.



Ejemplo 4.2

Convertir el número 1992_{10} a base 8:

$$\begin{array}{r|l} 1992 & 8 \\ \hline 249 & 0 \\ 31 & 1 \\ 3 & 7 \\ 0 & 3 \end{array}$$

$$249$$

$$8 \overline{)1992} \quad 0$$

$$8 \overline{)31} \quad 3 \quad 7$$

$$31$$

$$8 \overline{)249} \quad 1$$

$$8 \overline{)3} \quad 0 \quad 3$$



Ejemplo 4.2

El resultado de convertir el número 1992_{10} a base 8 es:

1992	8
249	0
31	1
3	7
0	3

$$1992_{10} = 3710_8$$



20

Ejemplo 4.3

Convertir el número 1992_{10} a base 16:

$$\begin{array}{r|l} 1992 & 16 \\ \hline 124 & 8 \\ 7 & 12 \\ 0 & 7 \end{array}$$

$$\begin{array}{r} 124 \\ 16 \overline{) 1992} \\ 8 \\ \hline 124 \\ 16 \overline{) 124} \\ 12 \\ \hline 0 \\ 16 \overline{) 7} \\ 7 \end{array}$$



Ejemplo 4.3

El el resultado de convertir el número 1992_{10} a base 16 es:

1992	16
124	8
7	12
0	7

$$1992_{10} = 7C8_{16}$$

NOTA: Hay que recordar que el sistema hexadecimal utiliza los dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.



Ejercicio 4.1

Representar el siguiente resultado (número) en las bases 2, 8 y 16.

Por equipo, sumar el día de nacimiento por el mes de nacimiento de cada integrante y, al resultado, sumarle 1000, es decir:

$$\text{Num} = (d_1*m_1 + d_2*m_2 + d_3*m_3 + d_4*m_4 + d_5*m_5) + 1000$$



Conversión de números reales de base *diez* a base *b*

Para representar un número real base 10 a base *b*, es necesario dividir la parte entera del número decimal (dividendo) entre la base (divisor) hasta que el resultado (cociente) sea cero y multiplicar la parte fraccionaria del número (multiplicando) por la base (multiplicador). Los residuos obtenidos forman el número convertido.



Ejemplo 4.4

Convertir el número 7.25_{10} a base 2:

7	2
3	1
1	1
0	1

0.25	2
0.5	0
0.0	1

Por lo tanto, el resultado de convertir el número 7.25_{10} a base 2 es 111.01_2 .



Conversión de números enteros de base b a base diez

Un número entero en base diez está representado de la siguiente manera:

$$d_n d_{n-1} \dots d_2 d_1 d_0$$

donde, d hace referencia a un número decimal (dígito de 0 a 9) y n hace referencia a la posición del número a partir del punto decimal (parte entera).



La magnitud de un número esta determinada por la sumatoria del dígito multiplicado por la base elevada a la posición del dígito, es decir:

$$\text{Número} = d_n b^n + d_{n-1} b^{n-1} + \dots + d_2 b^2 + d_1 b^1 + d_0 b^0$$



La fórmula anterior se puede simplificar como:

$$\text{Número} = \sum_{i=0}^n d_i \times b^i$$

donde:

Número = El número en base diez.

d = Dígito iésimo de la cantidad a transformar.

b = Base elegida.

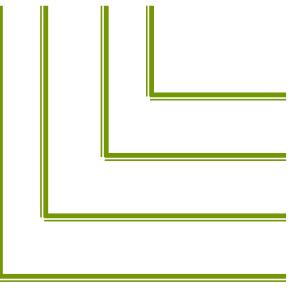
i = Posición que inicia en 0 y termina en n (el número de dígitos menos uno).



Ejemplo 4.5

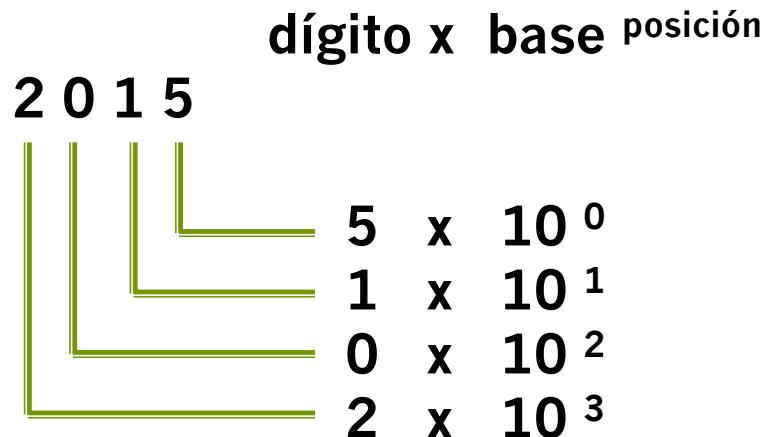
Se tiene el número 2015_{10} , comprobar que si se utiliza la fórmula mencionada ($d_n b^n + d_{n-1} b^{n-1} + \dots + d_2 b^2 + d_1 b^1 + d_0 b^0$) se obtiene el mismo número.

dígito x base ^{posición}

2 0 1 5	
	5 x 10^0
	1 x 10^1
	0 x 10^2
	2 x 10^3



Ejemplo 4.5



El resultado se obtiene desarrollando los productos y realizando la suma, por tanto:

$$\text{número} = 5 \times 10^0 + 1 \times 10^1 + 0 \times 10^2 + 2 \times 10^3$$

$$\text{número} = 5 \times 1 + 1 \times 10 + 0 \times 100 + 2 \times 1000$$

$$\text{número} = 5 + 10 + 0 + 2000 = 2015$$

Con lo anterior se comprueba la fórmula.



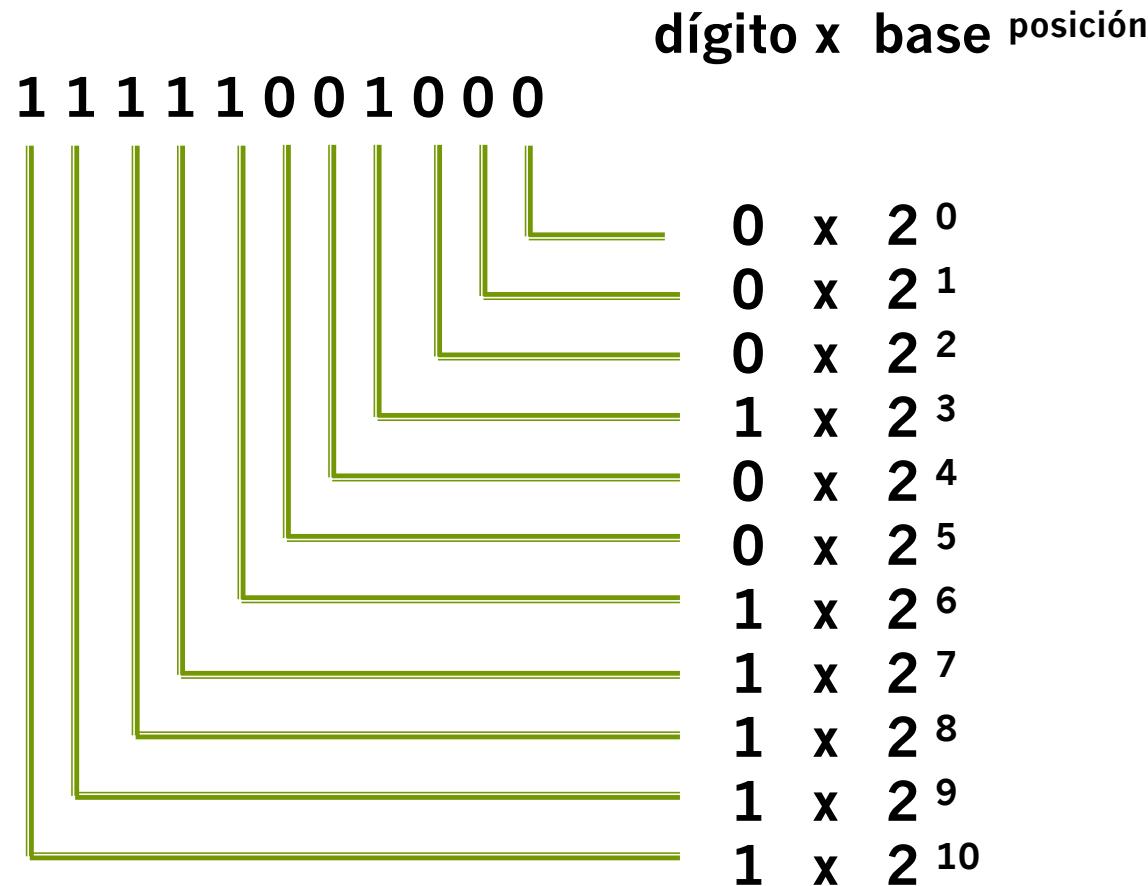
La tabla de valores por posición para definir cantidades de sistema binario a sistema decimal es:

... 2^4 2^3 2^2 2^1 2^0 . 2^{-1} 2^{-2} 2^{-3} 2^{-4} ...

... 16 8 4 2 1 . 1/2 1/4 1/8 1/16 ...

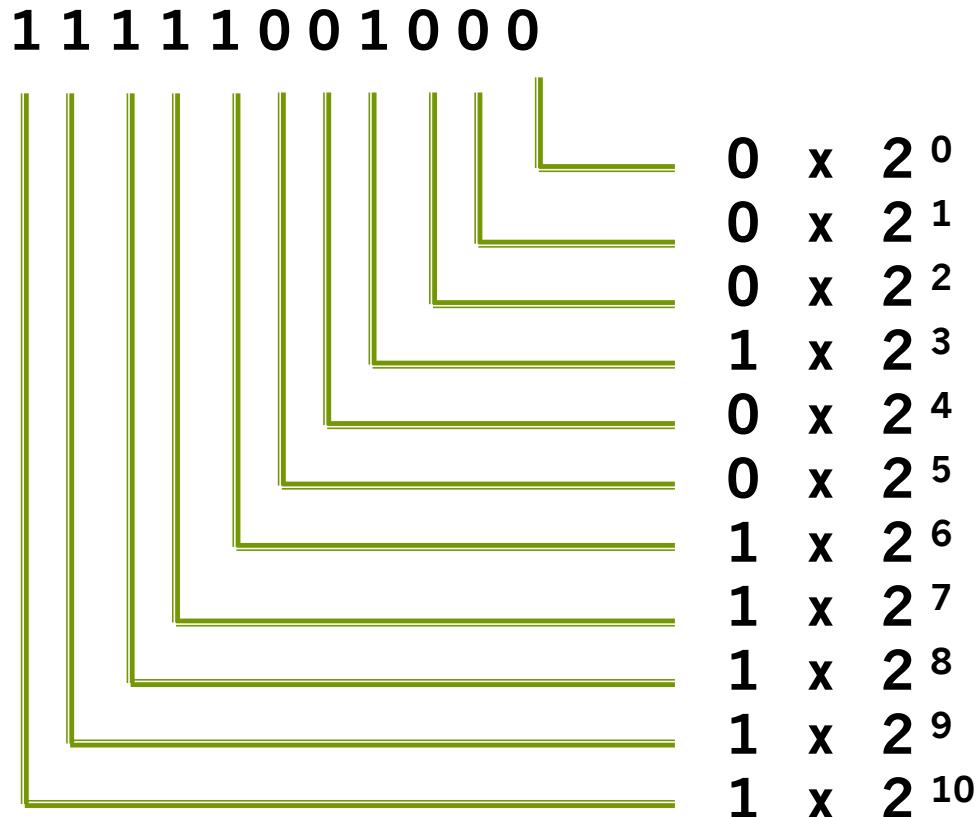
**Ejemplo 4.6**

Se tiene el número 11111001000_2 , obtener la equivalencia en sistema decimal.





Ejemplo 4.6



$$\begin{aligned}
 \text{Número} = & 1 \times 2^{10} + 1 \times 2^9 + 1 \times 2^8 + 1 \times 2^7 + 1 \times 2^6 \\
 & + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 \\
 & + 0 \times 2^1 + 0 \times 2^0
 \end{aligned}$$



Ejemplo 4.6

$$\begin{aligned}\text{Número} = & 1 \times 2^{10} + 1 \times 2^9 + 1 \times 2^8 + 1 \times 2^7 + 1 \times 2^6 \\ & + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 \\ & + 0 \times 2^1 + 0 \times 2^0\end{aligned}$$

$$\begin{aligned}\text{Número} = & 1 \times 1024 + 1 \times 512 + 1 \times 256 + 1 \times 128 \\ & + 1 \times 64 + 0 \times 32 + 0 \times 16 + 1 \times 8 \\ & + 0 \times 4 + 0 \times 2 + 0 \times 1\end{aligned}$$

$$\text{Número} = 1024 + 512 + 256 + 128 + 64 + 8 = 1992$$

Por lo tanto:

$$11111001000_2 = 1992_{10}$$



Ejemplo 4.7

Se tiene el número 0.01_2 , obtener la equivalencia en sistema decimal.

dígito x base posición

$$\begin{array}{r} 0.01 \\ \hline 1 & x & 2^{-2} \\ 0 & x & 2^{-1} \\ 0 & x & 2^0 \end{array}$$

$$\begin{aligned} \text{Número} &= 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} \\ \text{Número} &= 0 + 0 + 1/4 = 0.25 \end{aligned}$$

Por lo tanto 0.01_2 es igual a 0.25_{10} .



La tabla de valores por posición para definir cantidades de sistema octal a sistema decimal es:

... 8^4 8^3 8^2 8^1 8^0 . 8^{-1} 8^{-2} 8^{-3} 8^{-4} ...

... 4096 512 64 8 1 . 1/8 1/64 1/512 1/4096 ...

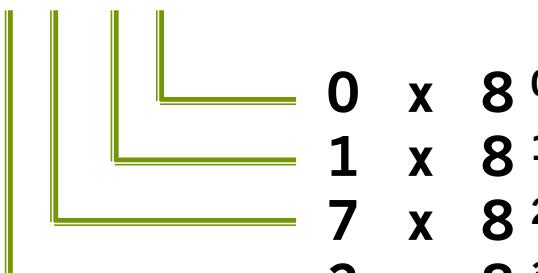


Ejemplo 4.8

Se tiene el número 3710_8 , obtener la equivalencia en sistema decimal.

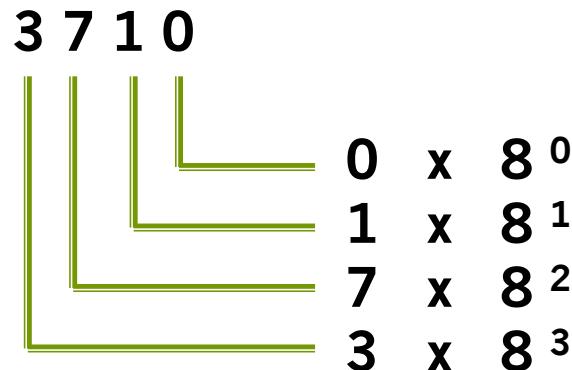
dígitos x base posición

3 7 1 0	
	0 x 8^0
	1 x 8^1
	7 x 8^2
	3 x 8^3





Ejemplo 4.8



$$\text{Número} = 3 \times 8^3 + 7 \times 8^2 + 1 \times 8^1 + 0 \times 8^0$$

$$\text{Número} = 3 \times 512 + 7 \times 64 + 1 \times 8 + 0 \times 1$$

$$\text{Número} = 1536 + 448 + 8 + 0$$

$$\text{Número} = 1992$$

Por lo tanto:

$$3710_8 = 1992_{10}$$



La tabla de valores por posición para definir cantidades de sistema hexadecimal a sistema decimal es:

... 16^3 16^2 16^1 16^0 . 16^{-1} 16^{-2} 16^{-3} ...

... 4096 256 16 1 . 1/16 1/256 1/4096 ...



Ejemplo 4.9

Se tiene el número $7C8_{16}$, obtener la equivalencia en sistema decimal.

dígito x base posición

$$\begin{array}{r} 7 \ C \ 8 \\ \text{---} \\ 8 \quad x \quad 16^0 \\ C \quad x \quad 16^1 \\ 7 \quad x \quad 16^2 \end{array}$$



Ejemplo 4.9

$$\begin{array}{c} 7 \text{ C } 8 \\ \text{---} \\ 8 \quad x \quad 16^0 \\ \text{C} \quad x \quad 16^1 \\ 7 \quad x \quad 16^2 \end{array}$$

$$\text{Número} = 7 \times 16^2 + \text{C} \times 16^1 + 8 \times 16^0$$

$$\text{Número} = 7 \times 256 + \text{C} \times 16 + 8 \times 1$$

$$\text{Número} = 1792 + 192 + 8$$

$$\text{Número} = 1992$$

Por lo tanto:

$$7\text{C}8_{16} = 1992_{10}$$

Ejercicio 4.2

Encontrar el número decimal de los siguientes valores (por equipo):

Equipo	Base 2	Base 8	Base 16
1	1 0101 0101 0111	13463	0x1857
2	1 0011 0001 1111	12527	0x1733
3	1 0011 0100 0100	11437	0x1557
4	1 0010 1111 0101	11504	0x131F
5	1 1001 0111 1100	11365	0x1344
6	1 1000 0011 0110	14574	0x12F5
7	1 1000 0101 0111	14066	0x197C
8	1 0111 0011 1011	14127	0x1836



Relación entre los sistemas binario, octal, decimal y hexadecimal

Considerando las tablas de posición para los sistemas binario, octal y hexadecimal se tiene:

(512) 2^9	(256) 2^8	(128) 2^7	(64) 2^6	(32) 2^5	(16) 2^4	(8) 2^3	(4) 2^2	(2) 2^1	(1) 2^0
(512) 8^3			(512) 8^2			(8) 8^1			(1) 8^0
	(512) 16^2				(16) 16^1				(1) 16^0



De la tabla anterior se puede observar lo siguiente:

- **Si se agrupan 3 dígitos binarios se obtiene un dígito octal.**
- **Si se agrupan 4 dígitos binarios se obtiene un dígito hexadecimal.**



Conversión de sistema binario a sistema octal

Dado un número binario, se separa en conjuntos de tres elementos (a partir del punto). A cada conjunto se le asocian los valores por posición 4, 2, 1, y con ellos se obtiene el dígito octal equivalente.



Ejemplo 4.10

Dado el número 11111001000_2 , obtener el número equivalente en sistema octal.

4	2	1	4	2	1	4	2	1	4	2	1
0	1	1	1	1	1	0	0	1	0	0	0
3			7			1			0		

Por lo tanto, el número 11111001000_2 es igual al número 3710_8 .



Conversión de sistema binario a sistema hexadecimal

Dado un número binario, se separa en conjuntos de cuatro elementos (a partir del punto). A cada conjunto se le asocian los valores por posición 8, 4, 2, 1, y con ellos se obtiene el dígito hexadecimal equivalente.



Ejemplo 4.11

Dado el número 11111001000_2 , obtener el número equivalente en sistema hexadecimal.

8	4	2	1	8	4	2	1	8	4	2	1
0	1	1	1	1	1	0	0	1	0	0	0
7		C				8					

Por lo tanto, el número 11111001000_2 es igual al número $7C8_{16}$.



Conversión de sistema octal a sistema binario

Dado un número octal, se asocia a cada dígito octal tres dígitos binarios; la unión de estos grupos de dígitos formarán la cantidad equivalente en sistema binario.



49

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



Conversión de sistema hexadecimal a sistema binario

Dado un número hexadecimal, se asocia a cada dígito hexadecimal cuatro dígitos binarios; la unión de estos grupos de dígitos formarán la cantidad equivalente en sistema binario.



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



Conversión de sistema hexadecimal u octal a otra base

Dado un número hexadecimal u octal, si se quiere obtener su equivalente en otra base, se toma como referencia el sistema binario (por facilidad) y de ahí se convierte a sistema deseado.

Ejemplo 4.12

Dado el número $7C8_{16}$, obtener el equivalente en base 4.

7	C	8								
0	1	1	1	1	0	0	1	0	0	0
1	3	3	0	2	0					

Por lo tanto, el número $7C8_{16}$ es igual a 133020_4 .



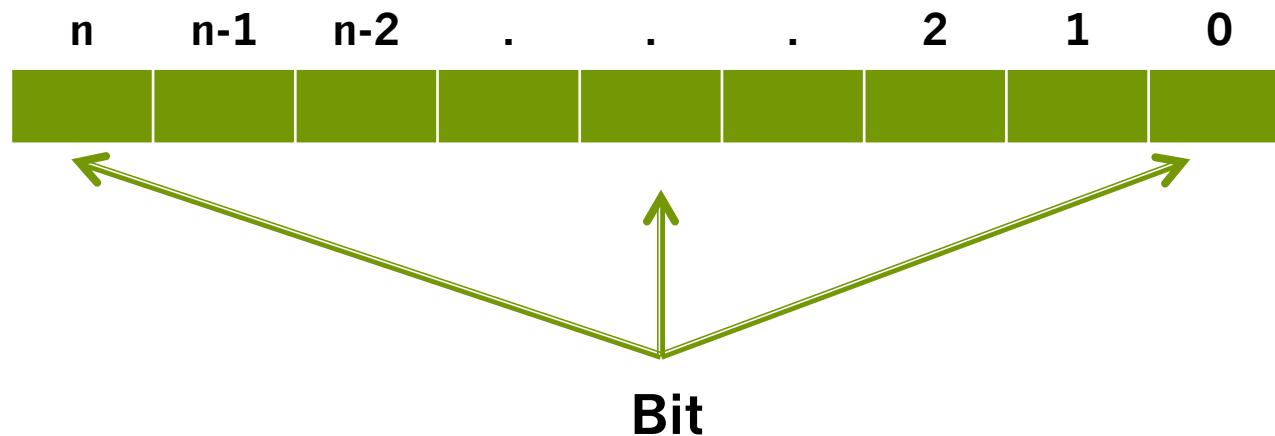
Unidades de medida de almacenamiento

La unidad central de proceso toma instrucciones y datos de la memoria primaria en grupos de n bits llamados *palabra* de computadora.

Un *bit* es el nombre que recibe un dígito en binario, el cual solo puede tomar dos posibles estados: cero y uno. Un *byte* es un conjunto de 8 bits.



La longitud de una palabra de computadora es el número de bits que la componen. En la mayoría de los equipos de cómputo esta longitud oscila entre 8 y 64 bits.





**Un bit puede representar dos estados distintos (2^1).
Una palabra de computadora de longitud n puede
representar 2^n estados distintos.**



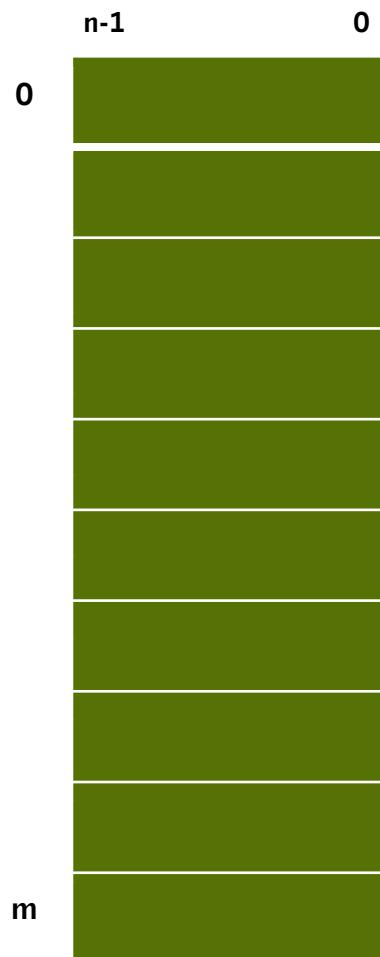
	n-1	n-2	n-3	.	.	.	2	1	0
0									
1									
2									
3									
...									
m-3									
m-2									
m-1									

La memoria, físicamente, está constituida por un conjunto de m palabras de longitud n. A cada palabra de computadora se le asocia una dirección.



La representación de información en la memoria se realiza mediante un cierto número de bits, dependiendo del tipo de dato.

Para escribir (almacenar) o leer (recuperar) de una palabra de computadora, existe el registro de dirección, el registro de datos y líneas de control de flujo de datos.



Registro de dirección



Registro de datos



Control E/S



Unidad	2^n	# bytes	Equivalencia
Kilobyte (KB)	2^{10}	1024	1024 bytes
Megabyte (MB)	2^{20}	1048576	1024 KB
Gigabyte (GB)	2^{30}	1073741824	1024 MB
Terabyte (TB)	2^{40}	1099511627776	1024 GB
Petabyte (PB)	2^{50}	1125899906842624	1024 TB
Exabyte (EB)	2^{60}	1152921504606846976	1024 PB
Zettabyte (ZB)	2^{70}	1180591620717411303424	1024 EB
Yottabyte (YB)	2^{80}	1208925819614629174706176	1024 ZB

Unidades de almacenamiento



61



4.2 Representación de datos tipo texto



4.2 Representación de datos tipo texto

Uno de los conjuntos de símbolos que más se utilizan en un procesador son aquellos que se introducen a través del teclado.

Desde la aparición de los primeros procesadores ha existido la necesidad de tener una codificación para estos símbolos. Dado que las computadoras intercambian entre sí infinidad de datos, esta codificación es deseable que sea idéntica para todos ellos.



Código ASCII

Una de las codificaciones de letras que más trascendencia ha tenido en los últimos años es la codificación ASCII (American Standard code for Information interchange).

Esta codificación incluye letras, dígitos y códigos especiales para la transmisión de mensajes entre computadoras.



64

Utiliza grupos de 7 bits por carácter ($2^7=128$) lo que permite describir el alfabeto con letras mayúsculas y minúsculas y símbolos comunes.

El código ASCII extendido usa 8 bits por carácter, lo que añade otros 128 caracteres posibles. Este juego de códigos más amplio permite agregar símbolos de otros lenguajes y varios símbolos gráficos.



El código ASCII fue publicado como estándar por primera vez en 1967 y fue actualizado por última vez en 1986.

En la actualidad define códigos para 33 caracteres no imprimibles, de los cuales la mayoría son caracteres de control obsoletos que tienen efecto sobre como se procesa el texto. Además, define 95 caracteres imprimibles (empezando por el carácter espacio).





Código EBCDIC

EBCDIC (Extended Binary Coded Decimal Interchange Code) es un código binario que representa caracteres alfanuméricos, controles y signos de puntuación. Fue desarrollado por IBM en 1964.

Cada carácter está compuesto por 8 bits, por lo tanto, EBCDIC define un total de 256 caracteres.



Copyright BetaSoftText.com
PrintSoftText.com



Unicode

Unicode es un estándar industrial cuyo objetivo es proporcionar el medio por el cual un texto en cualquier forma e idioma pueda ser codificado para el uso informático.

La codificación Unicode se ha transformado en un estándar adoptado por las principales empresas de hardware y software. Con el paso del tiempo se espera que sea la única representación utilizada.



El estándar pretende ser lo más genérico posible, y por tanto, en lugar de fijar un único tamaño para la representación, su codificación la divide en tres posibles formas: 8 bits, 16 bits y 32 bits.

Estas codificaciones son diferentes pero todas son parte del estándar y se conocen con los nombres de “UTF-8”, “UTF-16” y “UTF-32” respectivamente.



Imagen	Símbolo	Código
z	z minúscula	0x007A
水	agua en chino	0x6C34
♪	clave de sol	0xD834 0xDD1E

Símbolos codificados con Unicode



Una vez definida la codificación de todas las letras y símbolos adicionales, las cadenas de estos símbolos se representan mediante una secuencia de códigos en los que cada número corresponde con una letra.

Esta codificación es utilizada por los editores de texto plano o lenguajes de programación para la representación de texto.



Codificación de instrucciones

Una de las codificaciones más importantes es la que utiliza el procesador para su conjunto de instrucciones. Cada instrucción con sus operandos se considera un elemento.

Para la codificación de las instrucciones es preciso escoger tanto el número de bits como la correspondencia entre instrucciones y su codificación binaria.



La estructura de una instrucción está formada por:

Operación Número (8 bits) Número (8 bits)

add 0x23 0x34

sub 0x4F 0xBA

mul 0x12 0xFF

div 0xFF 0x08



Las instrucciones están representadas por un número dentro del procesador, por ejemplo, 00 hace referencia a la instrucción add.

Si se quiere sumar dos cantidades, la instrucción y codificación sería la siguiente:

Instrucción: ADD 0x27 0xB2

Codificación: 0000 1001 1110 1100 1000 0000

Representación: 0x09EC80



Programa

```
.data
msg: .asciz "Hello world\n"
.text
.globl main
main: push $msg
      call printf
      add 4, %esp
      ret
```

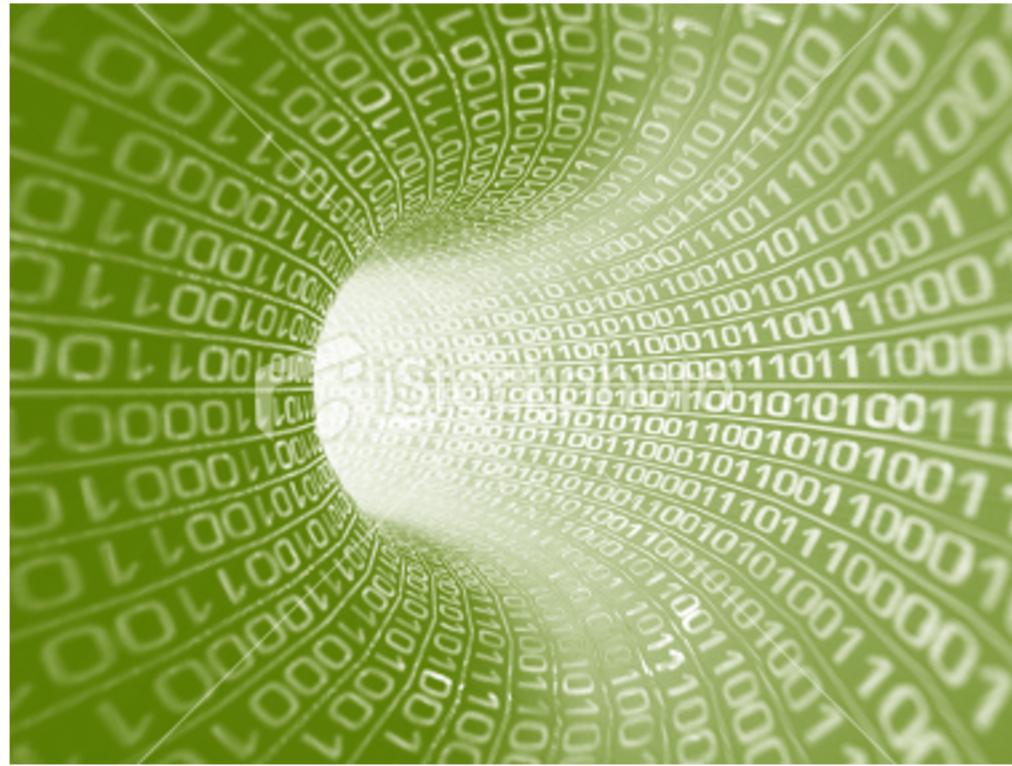
Codificación (ASCII)

```
20 20 20 20 20 20 2E 64 61 74 61 0A
6D 73 67 3A 20 20 2E 61 73 63 69 7A 20 22 48 65 6C 6C...
20 20 20 20 20 20 2E 74 65 78 74 0A
20 20 20 20 20 20 2E 67 6C 6F 62 6C 20 73 74 61 72 74...
6D 61 69 6E 3A 20 70 75 73 68 20 24 6D 73 67 0A
20 20 20 20 20 20 63 61 6C 6C 20 70 72 69 6E 74 66 0A
20 20 20 20 20 20 61 64 64 20 24 34 2C 20 25 65 73 70....
20 20 20 20 20 20 72 65 74 0A
```



77





4.3 Representación numérica: magnitud y signo, y complemento a r



4.3 Representación numérica: magnitud y signo, y complemento a r

Para representar un número entero en la computadora se utiliza el sistema de numeración binaria (base 2) debido a que la memoria solo puede almacenar dos dígitos: cero o uno.

A pesar de que el almacenamiento en memoria se realiza con números binarios, las operaciones se pueden realizar en una base diferente.



Suma y resta en base b

Es posible realizar operaciones básicas para una base dada de manera similar a como se realizan en base 10, lo único que hay que tomar en cuenta es que el número de dígitos permitidos lo establece la base en la que se esté trabajando.



La operaciones básicas entre números en base 10 utiliza dígitos de 0 a 9 (10 dígitos).



Ejemplo 4.13

Obtener el resultado de sumar 2894_{10} con 1587_{10} .

$$\begin{array}{r} 1 & 1 & 1 \\ 2 & 8 & 9 & 4 \\ + & 1 & 5 & 8 & 7 \\ \hline 4 & 4 & 8 & 1 \end{array}$$

El resultado de sumar 2894_{10} con 1587_{10} es 4481_{10} .

**Ejemplo 4.14**

Obtener el resultado de sumar los números 8765_{10} , 1974_{10} y 2875_{10} .

$$\begin{array}{r} & 1 & 1 \\ & 1 & 1 & 1 & 1 \\ 8 & 7 & 6 & 5 \\ + & 1 & 9 & 7 & 4 \\ \hline 2 & 8 & 7 & 5 \\ \hline 1 & 2 & 7 & 1 & 4 \end{array}$$

El resultado de la suma entre 8765_{10} , 1974_{10} y 2875_{10} es 12714_{10} .



Ejemplo 4.15

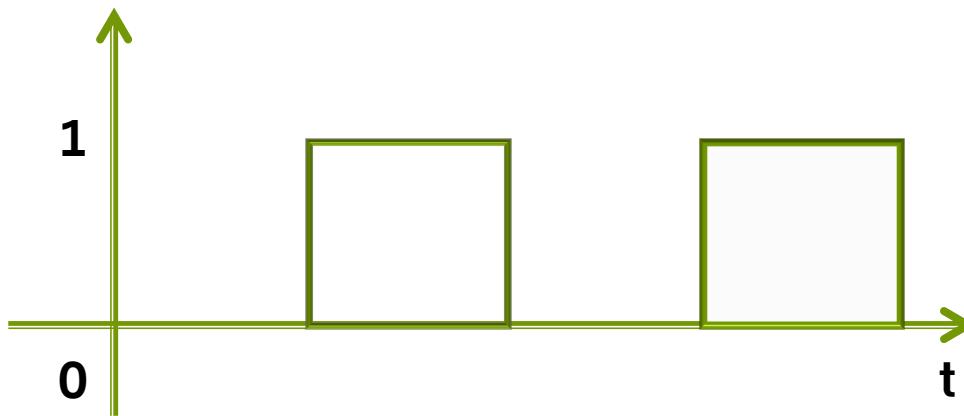
Obtener el resultado de restar 765_{10} a 1234_{10} .

$$\begin{array}{r} & 1 & 1 & 1 \\ & 1 & 2 & 3 & 4 \\ - & 7 & 6 & 5 \\ \hline & 0 & 4 & 6 & 9 \end{array}$$

El resultado de restar 765_{10} a 1234_{10} es 469_{10} .



Para representar números en base 2 solo es posible utilizar los dígitos de 0 y 1 (2 dígitos).



Función escalón unitario

Ejemplo 4.16

86



Obtener el resultado de sumar 1011_2 con 1010_2 .

$$\begin{array}{r} & 1 & 1 \\ & 1 & 0 & 1 & 1 \\ + & 1 & 0 & 1 & 0 \\ \hline & 1 & 0 & 1 & 0 & 1 \end{array}$$

El resultado de sumar 1011_2 con 1010_2 es 10101_2 .



Ejemplo 4.17

Obtener el resultado de sumar 1111_2 con 1110_2 con 1001_2 .

$$\begin{array}{r} & & 1 \\ & 1 & 1 & 1 & 1 & 1 \\ + & & 1 & 1 & 1 & 1 \\ & 1 & 1 & 1 & 0 \\ \hline & 1 & 0 & 0 & 1 & 0 \end{array}$$

El resultado de la suma entre 1111_2 , 1110_2 y 1001_2 es 100110_2 .

**Ejemplo 4.18**

Obtener el resultado de restar 1111_2 con 1110_2 con 1001_2 .

$$\begin{array}{r} 1111 \\ + \\ 1001 \\ \hline 0110 \end{array}$$

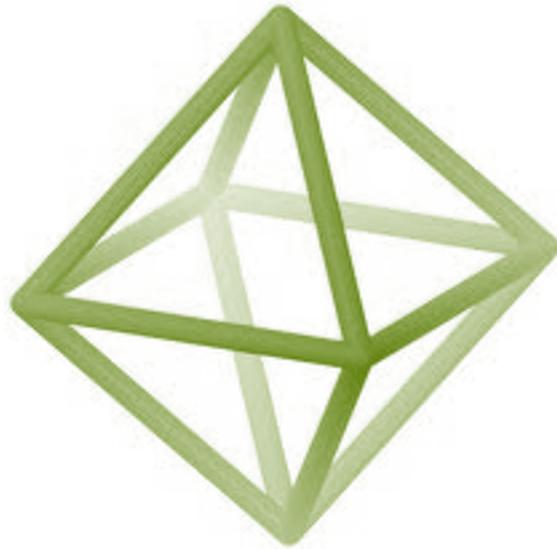
El resultado de la resta entre 1111_2 y 1001_2 es 0110_2 .

Ejemplo 4.19

Obtener el resultado de restar 1001_2 con 1111_2 .

$$\begin{array}{r} 1001 \\ - \quad \dots 1111 \\ \hline \dots 111010 \end{array}$$

**El resultado de la resta entre 1001_2 y 1111_2 es:
111010₂. El resultado está dado en complemento.**



Para representar números en base 8 es posible utilizar dígitos de 0 a 7 (8 dígitos).



Ejemplo 4.20

Obtener el resultado de sumar 6742_8 con 5742_8 .

$$\begin{array}{r} & 1 & 1 & 1 \\ & 6 & 7 & 4 & 2 \\ + & 5 & 7 & 4 & 2 \\ \hline & 1 & 4 & 7 & 0 & 4 \end{array}$$

El resultado de sumar 6742_8 con 5742_8 es 14704_8 .



```

0x0048196ab811c9318c90c9ab160x32dc39e40e62e06c200w76128c138a63dd3e7d0a
bedf4475ac8x797cfef0x84425ae0c9bd7290b0c351a757fb3a326c6bb8accaf7130e30a
31df8840x81f8fecb0x5742b05270x323c1c1d0x7451f6260xa65934ee0xAb2b224c0a101
3144750x962f0" "hFa0x390919cc0x1d3d07df0uc526cb900wf1253ed0wz216
ff693c0x8h" "55363890xaccea9460x1e8d87c98ac04f78abn7181d
77480x" "r215f7abv. "28a0c0x5357f140xf5soebf50xa914126f0wc4271A
390xp" "0x88d77fc70x>
0xca6fa18a0xc90bef0b0x9cd63506b4w139411
52f165de80x2049f5a20x4 77722910wbdf6ch
5cf 50x99e7c9b20x5t
xe5f253180x9f3634850ne9056af68wz2ed04
7d67bb150x2d59d4030x96cb5138ufbc2632
3d9 3private key0x4
xb4ab46e00x3cc916d10w6ae16a529wde594
d5 fa0x265787170x8
x88e5fe230xae1598b00wxc43f54080w93bac
2f5 ad98379760x536a5f0a0x19909ca30x15be00
779 130xccae0e5d0x<
0x6686fd720x8e907eae0x19867703bcdbe8a1
9576 80xee9025e30
1a86 3508da6P
0x3b445af80x6751dd5a0bx5d645eb0xf7bf4
1a163b3aa0x609edc440x957234830x15e1f1
0x6ab1e94e610w63ced2940xaf8c5ed99wz34470
1a87 3508da6P
e61f0x1ab37bf30w988557e70xc4a68a5b0x542fb4
0x76e67d3b0w
0x4f1980e0x1eccecd60x56931f4d0x7768c9b8xdeb4929
0x223f32060x955e7df30xc1d14fa50x49a8aca00xcc29d15c0x1fe399ac0x6a2d9691
30xdb33e9860x8d6315940xfc67e7cd8x3caeccc9d0xbdb8237690xf0d675940x8cbe33d
ad0x77fa5c4e0x9188a7580xcaeef8900xf73242ad0xeff8e0xf8d50x9f8eb4340x684d7216
50x2e4a03280x88ff2d450x24ddf7480x3c78d8910xdefbcc720w85f1b5d70x22f9cbf80
50x2d7869920xfe0250060x23b70d260xf1c561520x23e87380w67aa4fa70x151cfan30
70x7fd503f0xda7b17680x4e7c3dac0xfdde1dba0x61081dd0xcf6504350xde4cf8010w
hxc6a89f9680x27f72add0xeffe3dd330xddab30d90x86e3fd160x252ecbf50x386161820w
0x4a912w320w3217e1fc0w84db57660x47f724820x36d4c25d0w6d1c6dbf0xb2f28e2d0w

```

Para representar números en base 16 es posible utilizar dígitos de 0 a 9 y letras de la A a la F (16 dígitos).



Ejemplo 4.21

Obtener el resultado de sumar EB7C_{16} con 19BB_{16} .

$$\begin{array}{r} 1 \ 1 \ 1 \ 1 \\ \text{E} \ \text{B} \ 7 \ \text{C} \\ + \ 1 \ 9 \ \text{B} \ \text{B} \\ \hline 1 \ 0 \ 5 \ 3 \ 7 \end{array}$$

El resultado de sumar EB7C_{16} con 19BB_{16} es 10537_{16} .



Complemento aritmético

El complemento aritmético (a') de un número real se refiere a la cantidad que le falta a dicho número para ser igual a una unidad del orden inmediato superior.



El complemento aritmético a^r (o complemento a la base) de un número real se obtiene a partir de la siguiente fórmula:

$$a^r = r^n - |N|$$

donde:

a^r : complemento aritmético de un número real base r .

r : es la base del número.

n : número de dígitos de la parte entera del número.

N : el número dado.



Ejemplo 4.22

Dado el número 789_{10} , obtener su complemento aritmético a la base a^r (Complemento a^{10}).

$$\begin{aligned}a^r &= r^n - |N| \\c &= 10^3 - |789| \\c &= 1000 - 789 = 211\end{aligned}$$

$$a^r = 211_{10}$$



Ejemplo 4.23

Obtener el resultado de restar 1001_2 con 1111_2 .

$$\begin{array}{r} 1001 \\ - \quad \dots 1111 \\ \hline \dots 111010 \end{array}$$

El resultado de la resta está dado en complemento a la base (complemento a^2), por lo tanto, se debe aplicar la fórmula de complemento aritmético para obtenerlo en magnitud y signo, ya que:

$$n = (a^r)^r$$



Ejemplo 4.23

Dado complemento aritmético 111010_2 , obtener el número en magnitud y signo.

$$\begin{aligned}n &= (a^r)^r \\a^r &= r^n - |N| \\n &= 2^6 - |111010|\end{aligned}$$

$$\begin{array}{r} 100000 \\ - 111010 \\ \hline 0000110 \end{array}$$



Ejemplo 4.23

Para un número en complemento el bit de signo es parte del número.

En el ejemplo anterior, el bit más significativo del número está prendido (1), por lo tanto, se puede afirmar que el número es negativo.

Entonces, la magnitud y signo del número 111010_2 es -110_2 .

10
0

Ejercicio 4.3

Obtener el complemento aritmético de los siguientes números:

Equipo	Base 2	Base 10
1	11100101	83456
2	11110000	71912
3	11000011	70007
4	11100001	49567
5	10000111	49234
6	11111000	61816
7	10001111	57362
8	11101111	56253



El complemento aritmético menos uno ($a^r - 1$ o complemento a la base disminuida) de un número real se calcula con base en la siguiente fórmula:

$$a^r - 1 = r^n - r^m - |N|$$

donde:

$a^r - 1$: complemento aritmético de un número real base r .

r : es la base del número.

n : número de dígitos de la parte entera del número.

m : número de dígitos de la parte fraccionaria del númer.

N : el número dado.



10
2

Ejemplo 4.24

Dado el número 789_{10} , obtener su complemento aritmético a la base a^r-1 (Complemento $a^{10}-1$).

$$\begin{aligned}a^r-1 &= r^n - r^m - |n| \\c &= 10^3 - 10^0 - |789| \\c &= 1000 - 1 - 789 = 210\end{aligned}$$

$$a^r-1 = 210_{10}$$

10
3

Ejemplo 4.25

Dado el número 1001.01_2 , obtener su complemento aritmético a la base a^r-1 (Complemento a^2-1).

$$\begin{aligned}a^{r-1} &= r^n - r^m - |n| \\c &= 2^4 - 2^{-2} - |1001.01| \\c &= 10000 - 0.01 - 1001.01 \\c &= 10000.0 - 1001.1 \\c &= 00110.1\end{aligned}$$

$$a^{r-1} = 110.1_2$$

10
4

Ejercicio 4.4

Obtener el complemento a la base disminuida de los siguientes números:

Equipo	Base 2	Base 10
1	10000111	49234
2	11111000	61816
3	10001111	57362
4	11101111	56253
5	11100101	83456
6	11110000	71912
7	11000011	70007
8	11100001	49567

10
5

Realizar la siguiente operación: 123 - 98.

Magnitud y signo

$$\begin{array}{r} 123 \\ - 98 \\ \hline 025 \end{array}$$

Complemento a^2

$$\begin{array}{r} 123 \\ + 902 \\ \hline 1025 \end{array}$$

Complemento a^2-1

$$\begin{array}{r} 123 \\ + 901 \\ \hline 1024 \\ \rightarrow 1 \\ \hline 025 \end{array}$$



Realizar la siguiente operación: 123 - 125.

Magnitud y signo

$$\begin{array}{r} 123 \\ - 125 \\ \hline -002 \end{array}$$

Complemento a^2

$$\begin{array}{r} 123 \\ + 875 \\ \hline 998 \\ -002 \end{array}$$

Complemento a^{2-1}

$$\begin{array}{r} 123 \\ + 874 \\ \hline 997 \\ +1 \\ -002 \end{array}$$



Representación de un número entero en memoria

Dependiendo de la arquitectura de la computadora, los números enteros pueden ocupar 16 ó 32 bits en memoria, donde el primer bit registra el signo y los restantes registran la capacidad del entero.

10
8

Un número entero que ocupa 32 bits se distribuye en la memoria de la siguiente manera:

**1 bit para el signo del número
31 bits para el número**

+/-	n ₃₀	n ₂₉	n ₂₈	n ₂₇	n ₂₆	n ₂₅	n ₂₄	n ₂₃	n ₂₂	n ₂₁	n ₂₀	n ₁₉	n ₁₈	n ₁₇	n ₁₆
n ₁₅	n ₁₄	n ₁₃	n ₁₂	n ₁₁	n ₁₀	n ₉	n ₈	n ₇	n ₆	n ₅	n ₄	n ₃	n ₂	n ₁	n ₀

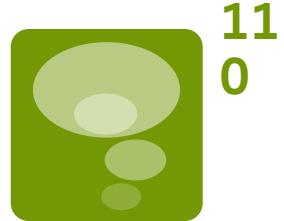
10
9

Ejemplo 4.26

Convertir el número $28,345_{10}$ a binario y mostrar su representación en memoria (32 bits).

$$28,345_{10} = 110111010111001_2$$

$28\ 345$	/ 2
14 172	1
7 086	0
3 543	0
1 771	1
885	1
442	1
221	0
110	1
55	1
27	1
13	1
6	1
3	0
1	1
0	1



Ejemplo 4.26

La representación en memoria del número $28,345_{10}$ es:

$$28,345_{10} = 110111010111001_2$$

+

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	1	1	1	0	1	0	1	1	1	1	0	0	1



Es importante tener en cuenta que un número entero puede superar la capacidad de almacenamiento determinada y ello provoca pérdida de información (parcial o total).



Representación de un número entero en complemento a^r en memoria

Dependiendo de la arquitectura de la computadora, un dato puede ser almacenado en memoria con su complemento (especialmente cuando el número es negativo).



Ejemplo 4.27

**Se desea representar en memoria el número entero -
 110111010111001_2 en complemento a^r.**

Para representar el complemento del número es necesario expresarlo con el número máximo de elementos que puede ser almacenados en memoria, es decir, para el caso de una arquitectura de 32 bits el número quedaría expresado como sigue:

$0000000000000000110111010111001_2$



Ejemplo 4.27

Una vez que se posee el número completo se obtiene el complemento a' :

$$(a) 0000000000000000110111010111001_2 = \\ = 1111111111111001000101000111_2 (a')$$



Ejemplo 4.27

La representación en memoria del número entero $0000000000000000110111010111001_2$ en complemento a^r es:

$$11111111111111001000101000111_2$$

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	1	0	0	0	1	0	1	0	0	0	1	1	1	1	1



Representación de un número real en la memoria

En la práctica se está constantemente trabajando con números reales, sin embargo, estos números no siempre se pueden representar de manera exacta debido a que el número de dígitos está limitado por el tamaño de palabra de cada máquina.

Por lo tanto, los números necesariamente son redondeados o truncados, y, con ello, se provoca un error.



Para realizar el registro de un número real en memoria se puede utilizar la notación de punto flotante normalizado o la notación de punto flotante no normalizada.



Un número A puede ser expresado en notación científica no normalizada, es decir, expresar el número como una potencia de 10:

$$A = C \times 10^n$$

Donde $C \geq 1$ y n es un entero positivo, negativo o cero ($n \in \mathbb{Z}$).

Por ejemplo, si se desea expresar el número 836.238 en la forma de punto flotante no normalizada el número quedaría: 836238×10^{-3} .



Un número A puede ser expresado en notación científica normalizada, es decir, expresar el número como una potencia de 10:

$$A = C \times 10^n$$

Donde $C < 1$ y n es un entero positivo, negativo o cero ($n \in \mathbb{Z}$).

Por ejemplo, si se desea expresar el número 836.238 en la forma de punto flotante normalizada el número quedaría: 0.836238×10^3 .



Los número binarios también pueden ser representados en notación científica normalizada (notación de punto flotante normalizada) de la siguiente manera:

$$A = M \times 2^n$$

donde n es un entero positivo, negativo o cero (expresado en binario) y M es la mantisa (dígitos significativos del número), que debe ser menor a 1.



Ejemplo 4.28

Expresar los siguientes números en su forma científica normalizada.

$$A = M \times 2^n$$

$$\begin{aligned} & 11111.01_2 \\ & 11111.01_2 = 0.1111101_2 \times 2^{101} \end{aligned}$$

$$\begin{aligned} & -0.00000011101101_2 \\ & -0.00000011101101_2 = -0.11101101_2 \times 2^{-110} \end{aligned}$$



Un número flotante ocupa 32 bits (4 bytes) en la memoria y se distribuyen de la siguiente manera:

1 bit para el signo de la mantisa

1 bit para el signo del exponente

7 bits para el exponente entero (en binario)

23 bits para la mantisa (en binario)

\pm	\pm	e_6	e_5	e_4	e_3	e_2	e_1	e_0	m_0	m_1	m_2	m_3	m_4	m_5	m_6
m_7	m_8	m_9	m_{10}	m_{11}	m_{12}	m_{13}	m_{14}	m_{15}	m_{16}	m_{17}	m_{18}	m_{19}	m_{20}	m_{21}	m_{22}

NOTA: Como la mantisa siempre empieza con uno, no hay necesidad de almacenar este dígito.



Ejemplo 4.29

Convertir el número 31.25_{10} a sistema binario y dibujar su representación en memoria.

$$31.25_{10} = 0.3125_{10} \times 10^2 = 0.3125_{10} \times 100_{10}$$

$$\begin{array}{r|l} 0.3125 & * 2 \\ 0.6250 & 0 \\ 0.2500 & 1 \\ 0.5000 & 0 \\ 0.0000 & 1 \end{array}$$

$$\begin{array}{r|l} 100 & \% 2 \\ 50 & 0 \\ 25 & 0 \\ 12 & 1 \\ 6 & 0 \\ 3 & 0 \\ 1 & 1 \\ 0 & 1 \end{array}$$

12
4

Ejemplo 4.29

0.3125	*2	↓
0.6250	0	
0.2500	1	
0.5000	0	
0.0000	1	

100	%	2	↑
50	0	0	
25	0	1	
12	1	0	
6	0	0	
3	0	1	
1	1	1	
0	1	1	

$$31.25_{10} = 0.0101_2 \times 1100100_2 = 11111.01_2$$
$$11111.01_2 = 0.1111101_2 \quad 2^{101}$$

0	0	0	0	0	0	1	0	1	1	1	1	1	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



12
5

Ejemplo 4.30

Realizar la representación del número -0.000721619 en memoria en una palabra de 32 bits.

Al convertir el número a binario se obtiene:

$$\begin{aligned}-0.000721619_{10} &= \\ &= -0.00000000010101101001010110000100000000101_2\end{aligned}$$

Se pasa el número a su forma científica normalizada:

$$= -0.10101101001010110000100000000101_2 \times 2^{-1010}$$



Ejemplo 4.30

Como el exponente es negativo, se le aplica complemento a² a 8 cifras ya que el signo es parte del número (se puede aplicar complemento a²⁻¹ y sumarle uno) :

$$00001010_2 = 11110101_2 + 1_2 = 11110110_2$$

Por lo tanto, el número a almacenar en memoria es:

$$-0.10101101001010110000100000000101_2 \times 2^{11110110}$$

12
7

Ejemplo 4.30

La representación en memoria del número -
 $0.10101101001010110000100000000101_2 \times 2^{11110110}$ es:

1	1	1	1	1	0	1	1	0	0	0	1	0	1	1	0	1
0	0	1	0	1	0	1	1	0	0	0	0	0	1	0	0	0



12
8

Ejercicio 4.5

Se tiene una computadora que maneja palabras de 16 bits, ¿cuál sería el resultado de sumar 1000 veces el número fraccionario $1/100$?

Dibujar su representación en memoria. Considerar 2 bits para los signos, 5 bits para el exponente y 9 bits para la mantisa.



12
9



4.4 Tipos de errores en la manipulación de cantidades



4.4 Tipos de errores en la manipulación de cantidades

El error absoluto es la diferencia entre el resultado real obtenido (valor calculado o aproximado) y la previsión que se había hecho o que se tiene como cierta (valor real):

$$\text{Error} = |\text{Valor verdadero} - \text{Valor estimado}|$$

$$E_x = |x_r - x^*|$$



El error relativo de un número está dado por el cociente del error absoluto entre un valor real obtenido (valor calculado o aproximado) o un valor de referencia (valor real):

$$\text{Error relativo} = \text{Error absoluto} / \text{Valor estimado}$$
$$e_x = |E_x / X_r|$$

$$\text{Error relativo} = \text{Error absoluto} / \text{Valor real}$$
$$e_x = |E_x / X^*|$$



Exactitud

La exactitud se refiere a que tan cercano está el valor calculado o medido con respecto al valor real.

La inexactitud (sesgo) se define como una desviación sistemática del valor verdadero, por ejemplo, un cronómetro que se detiene medio segundo después de pulsar el botón.



Precisión

La precisión se refiere a que tan cercanos se encuentran (unos de otros) los valores calculados o medidos.

La imprecisión (incertidumbre) se refiere a la magnitud en la dispersión de los datos.



Baja exactitud
Alta precisión



Alta exactitud
Baja precisión

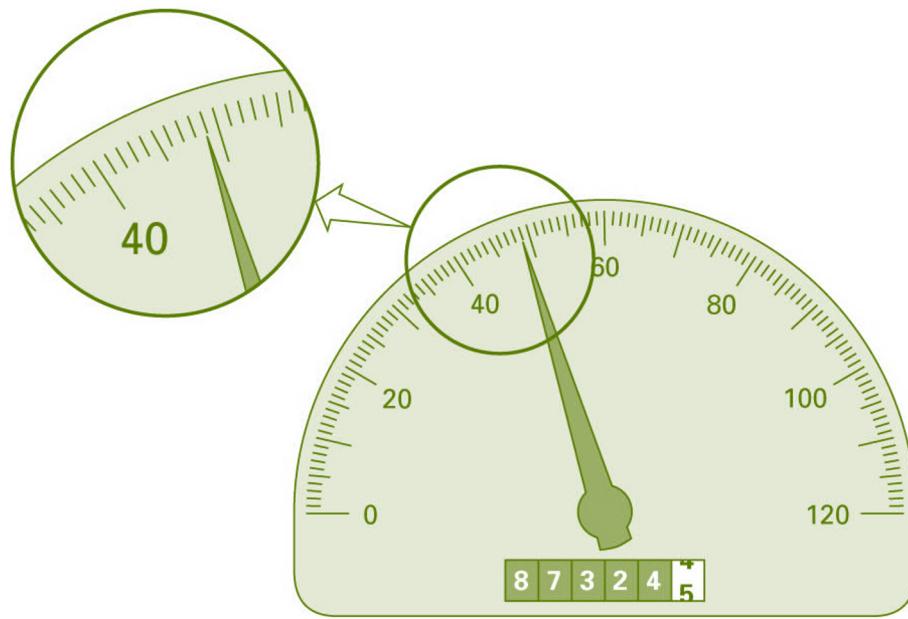


Alta exactitud
Alta precisión



Error de redondeo

El error de redondeo se atribuye tanto a la imposibilidad de almacenar todas las cifras de un número, como a la imprecisión de los instrumentos de medición con los cuales se obtienen los datos.



Cuando, en una medición práctica, se omiten cifras significativas en un número se le conoce como *error de redondeo*.



Cuando se redondea un número, es necesario observar la cifra que está a su derecha:

- 1) Si la cifra es mayor a 5, se suma 1 a la cifra anterior, es decir, la que está a la izquierda.
- 2) Si es menor a 5, la cifra anterior no se altera.
- 3) Si la cifra es igual a 5, se observa la cifra anterior, si ésta es un número par se deja la misma cifra, de lo contrario (si es impar), se deja en la cifra par siguiente.



Ejemplo 4.31

Redondear las siguientes cifras como se especifica:

$$72.36 \text{ (en décimas)} = 72.4$$

$$7.462 \text{ (en centésimas)} = 7.46$$

$$7.465 \text{ (en centésimas)} = 7.46$$

$$7.475 \text{ (en centésimas)} = 7.48$$

$$72.8 \text{ (a unidades)} = 73$$

$$116,500,000 \text{ (a millones)} = 116,000,000$$

$$117,500,000 \text{ (a millones)} = 118,000,000$$



NOTA:

En 1991 un misil iraquí impactó en un cuartel estadounidense, matando a 28 soldados.

La investigación demostró que el misil fue detectado por el radar, pero el sistema antimisiles Patriot no entró en funcionamiento.

El sistema Patriot tenía un error de redondeo (al calcular $1/10$) que provocaba un retraso de 0.000000095 segundos cada segundo. Tras 100 horas en funcionamiento, el sistema antimisiles acumulaba un error 0.34 segundos, tiempo suficiente para que un misil Scud iraquí avanzase 600 metros y escapase de la detección.



Error de truncamiento

Los errores de truncamiento se pueden presentar bajo las siguientes dos circunstancias:

- **Cuando un número se reemplaza por una expresión de éste más simple.**
- **Al calcular un número de manera iterativa, se genera un error al momento de detener el proceso iterativo.**



Cuando se trunca un número en una cifra determinada, se consideran igual a cero todas las cifras que siguen a la derecha de dicho número.



Ejemplo 4.32

Truncar:

1) **7.475 (en décimas) = 7.4**

$$E = 7.475 - 7.4 = 0.075$$

$$e = (0.075/7.475) * 100 = 1\%$$

2) **7.447 (en décimas) = 7.4**

$$E = 7.447 - 7.4 = 0.047$$

$$e = (0.075/7.447) * 100 = 0.63\%$$



La serie de Taylor proporciona una buena forma de aproximar el valor de una función en un punto en términos del valor de la función y sus derivadas en otro punto.

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a)^1 + \frac{f''(a)}{2!}(x-a)^2 + \dots + \frac{f^n(a)}{n!}(x-a)^n$$

La idea general es realizar operaciones según una fórmula general y, mientras más términos de la fórmula tenga la serie, más exacto será el resultado.

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

**Ejemplo 4.33**

La serie de Taylor para calcular e^x :

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \frac{x^6}{720} + \dots$$

Obtener una aproximación a e^x , para $x = 0.1$, utilizando cuatro elementos de la serie con 8 cifras significativas. Obtener el error relativo porcentual con el valor exacto de $e^{0.1}$.

$$e^{0.1} = 1.1051709180756$$

**Ejemplo 4.33**

$$e^{0.1} = 1.1051709180756, x = 0.1$$

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \frac{x^6}{720} + \dots$$

$$e^{0.1} \approx 1 + (0.1/1) + (0.1)^2/2 + (0.1)^3/6$$

$$e^{0.1} \approx 1 + 0.1 + 0.005 + 1.6666667 \times 10^{-4} = 1.10516667$$

$$E = \frac{1.1051709180756 - 1.10516667}{1.1051709180756} * 100 =$$

$$E = 0.0003816672$$



Ejercicio 4.6

Obtener las aproximación de e^x , para $x = 0.1$, utilizando cinco y seis elementos de la serie con 8 cifras significativas. Obtener el error relativo porcentual con el valor exacto de $e^{0.1}$.



Rango de valores para almacenar en memoria

Las computadoras poseen limitaciones físicas propias de la arquitectura: velocidad del procesador, capacidad de almacenamiento, etc.

Así mismo, los lenguajes de programación poseen un rango de bits específico dependiendo del tipo de dato que se desea almacenar.



La capacidad de almacenamiento de un número en memoria está dado por la siguiente fórmula:

$$\sum_{i=0}^n 2^i$$

Donde:

***i* es la potencia a la que se eleva la base 2.**

***n* es el número máximo de bits - 1 definido según el tipo de dato a almacenar.**



Ejemplo 4.34

Para número entero que ocupa 32 bits, en memoria su representación es la siguiente:

+/-	n ₃₀	n ₂₉	n ₂₈	n ₂₇	n ₂₆	n ₂₅	n ₂₄	n ₂₃	n ₂₂	n ₂₁	n ₂₀	n ₁₉	n ₁₈	n ₁₇	n ₁₆
n ₁₅	n ₁₄	n ₁₃	n ₁₂	n ₁₁	n ₁₀	n ₉	n ₈	n ₇	n ₆	n ₅	n ₄	n ₃	n ₂	n ₁	n ₀

Ejemplo 4.34

Su capacidad de almacenamiento está dada por:

$$\sum_{i=0}^n 2^i$$

$$2^0 + 2^1 + 2^2 + 2^3 + \dots + 2^{29} + 2^{30} + 2^{31} = 4\,294\,967\,313$$

Por lo tanto, en la memoria es posible representar un número entero en el rango -4,294,967,314 a 4,294,967,313.



Se tiene un tipo de dato que ocupa 3 bits, solo es posible almacenar 2^2 datos diferentes en memoria.

Magnitud y signo

0 1 1
0 1 0
0 0 1
0 0 0

1 0 0
1 0 1
1 1 0
1 1 1

Complemento a²

0 1 1
0 1 0
0 0 1
0 0 0

1 1 1
1 1 0
1 0 1
1 0 0

Complemento a²-1

0 1 1
0 1 0
0 0 1
0 0 0

1 1 1
1 1 0
1 0 1
1 0 0



4.5 Formatos de archivos de texto, imágenes, video, audio, etc.



4.5 Formatos de archivos de texto, imágenes, video, audio, etc.

Cómo ya se ha analizado, los componentes físicos de una computadora solo pueden almacenar dos datos (0 ó 1).

Por tanto, la información contenida en un archivo está almacenada en conjuntos de bits dentro de la computadora. Cuando se quiere recuperar la información almacenada, se necesitan leer estos bits.



Si la información se encuentra almacenada como 0 ó 1, ¿cómo se puede saber si el conjunto de bits se refiere a un archivo de texto, a uno de video, a una imagen o a un archivo de audio?



Formato de archivo

Un *formato de archivo* se refiere a la manera particular de codificar información para ser almacenada y, posteriormente, recuperada (decodificar).

Existen diferentes tipos de formatos para diversos tipos de información.



Para visualizar un archivo, se necesita una aplicación que permita leer, editar y guardar los datos contenidos en el mismo.

El formato de archivo permite discernir entre aplicaciones para poder editar un archivo, es decir, dependiendo del formato, es posible ver el contenido de un archivo con diferentes aplicaciones.



De manera general y con base en su facilidad de apertura, existen dos tipos de formatos de archivos: libre y propietario.



Formato libre (o abierto)

Son formatos que poseen una especificación de referencia bajo una licencia libre y pueden ser implementados por cualquiera sin restricciones legales de uso.

Habitualmente son publicados y patrocinados por organizaciones de estándares abiertos, aunque muchos son desarrollados por empresas.



Formato propietario (o cerrado)

Son formatos que tienen restricciones legales de uso, no pueden ser implementados por cualquiera ya que sus especificaciones no son públicas, están sujetos al pago de licencias y son controlados y definidos por intereses privados.



16
0

Texto plano

- **TXT (.txt): formato libre.**

Documentos

- **DOC (.doc): formato propietario de MS-Word.**
- **OpenDocument Text (.odt): formato libre.**
- **PDF (.pdf): formato libre de Adobe.**



Hoja de cálculo

- **XLS (.xls): formato propietario de MS-Excel.**
- **OpenDocument Spreadsheet (.ods): formato libre.**

Presentación

- **PPT (.ppt): formato propietario de MS-PowerPoint.**
- **OpenDocument Presentation (.odp): formato libre.**



E-book

- **Microsoft Reader (.lit): formato propietario.**
- **Kindle (.prc, .azw): formato propietario.**
- **Texto Plano (.txt): formato libre.**
- **Texto Enriquecido (.rtf): formato libre.**
- **HTML (.html): formato libre.**
- **PDF (.pdf): formato libre.**
- **DjVu (.djvu): formato libre.**
- **EPub (.epub): formato libre.**
- **MobiPocket (.mobi): formato libre.**
- **Open Ebook (.oeb): formato libre.**



Comprimido

- **tarballs (.tar.gz): formato libre.**
- **ZIP (.zip): formato libre.**
- **7 Zip (.7z): formato libre.**
- **GNU Zip (.gz): formato libre.**
- **BZip2 (.bz2): formato libre.**
- **RAR (.rar): formato propietario.**

CAD

- **DWG (.dwg): formato propietario de AutoCAD.**
- **DXF (.dxf): formato libre.**



Imagen de disco

- **ISO (.iso): formato libre.**
- **CUE (.cue): formato libre.**
- **BIN (.bin): formato libre.**
- **Clone CD Control File (.ccd): formato propietario.**
- **Direct Access Archive (.daa): formato propietario.**
- **Media Descriptor (.mds): formato propietario.**
- **Nero Burning ROM (.nrg): formato propietario.**



Gráfico de mapa de bits

- **PNG (.png)**: formato libre que consigue la mayor compresión sin pérdida de calidad.
- **GIF (.gif)**: formato libre utilizado en transparencias o animaciones. No tiene pérdida de calidad.
- **JPEG (.jpg)**: formato libre, ideal para fotos destinadas a Internet, ya que comprime con pérdida de calidad, pero ésta se puede ajustar.
- **WebP (.webp)**: formato libre creado recientemente por Google en base el códec VP8. Parecido a JPEG.



Diseño gráfico

- **Photoshop (.psd): formato propietario de Adobe Photoshop.**
- **eXperimental Computing Facility (.xcf): formato libre utilizado en GIMP.**

Gráfico vectorial

- **Macromedia Flash (.swf): formato propietario de Adobe.**
- **Metafile (.wmf): formato propietario de Microsoft.**
- **SVG (.svg): formato libre.**



Audio

- **MP3 (.mp3)**: formato propietario del Instituto Fraunhofer.
- **AAC (.m4a, .mp4)**: formato propietario utilizado por iTunes.
- **Ogg Vorbis (.ogg)**: formato libre.
- **FLAC (.flac)**: formato libre para audio sin pérdida de calidad.
- **Speex (generalmente dentro de un contenedor .ogg)** formato libre para grabación de voz.
- **WMA (.wma)**: formato propietario de MS-Windows Media Audio.
- **QuickTime (.mov)**: formato propietario de Apple.
- **RealAudio (.ra)**: formato propietario de RealNetworks.
- **AMR (.amr)**: formato propietario para grabar voz.



Video

- **MPEG (.mpg)**: formato propietario de MPEG.
- **DivX (.avi)**: formato propietario de DivX.
- **Ogg Theora (.ogv)**: formato libre.
- **XviD (.avi)** : formato libre.
- **WebM (.webm)** : formato libre.
- **WMV (.wmv)**: formato propietario de MS-Windows Media Video.
- **QuickTime (.mov)**: formato propietario de Apple.
- **RealVideo (.rm)**: formato propietario de RealNetworks.



Contenedor multimedia

Un contenedor multimedia es un tipo de archivo que almacena audio, vídeo, subtítulos, capítulos, meta-datos, etc.

- **MPEG-4 (.mpg)**, formato propietario de MPEG.
- **AVI (.avi)**: formato propietario de Microsoft.
- **Ogg (.ogg)**: formato libre.
- **Matroska (.mkv)** : formato libre.
- **QuickTime (.mov)**: formato propietario de Apple para sus códecs.
- **RealMedia (.rm)**: formato propietario de RealNetworks para sus códecs.



4. Manejo interno de datos

Objetivo: Describir cómo se almacenan los datos en los distintos medios de un sistema de cómputo, así como, manipular los datos para minimizar los diferentes errores que pueden suscitarse en su almacenamiento.

4.1 Dispositivos y unidades de almacenamiento: bit, byte y palabra.

4.2 Representación de datos tipo texto (códigos ASCII y EBCDIC).

4.3 Representación numérica: magnitud y signo, y complemento a r.

4.4 Tipos de errores en la manipulación de cantidades.

4.5 Formatos de archivos de texto, imágenes, video, audio, etc.