

Informática I - Electrónica

Código de Materia 950452
R1091 Viernes Turno Noche

Oscar Paniagua
Gaston Coustau



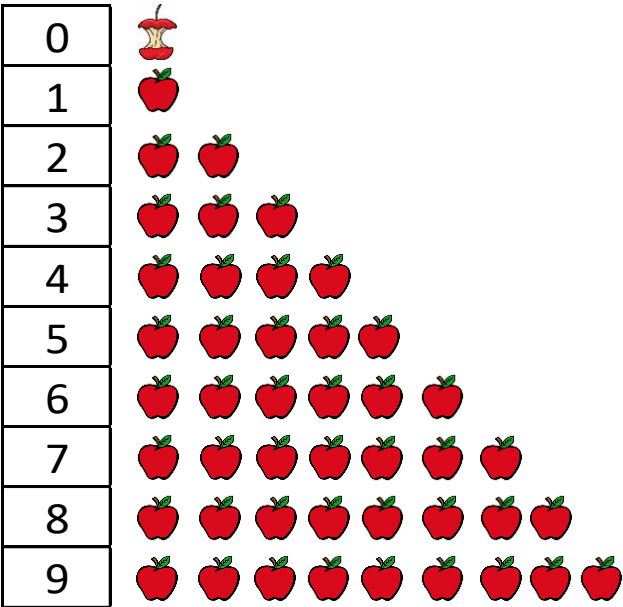
UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES




Premio Nacional a la
Calidad 2016-2019


Sistemas de numeración | ¿cómo contamos?


- A la hora de representar una cantidad, asignamos un símbolo de nuestro sistema de numeración a cada una de las cantidades a representar:




¿Y que pasa si quiero representar una cantidad mayor que la cantidad de simbolos que tiene mi sistema de numeracion?







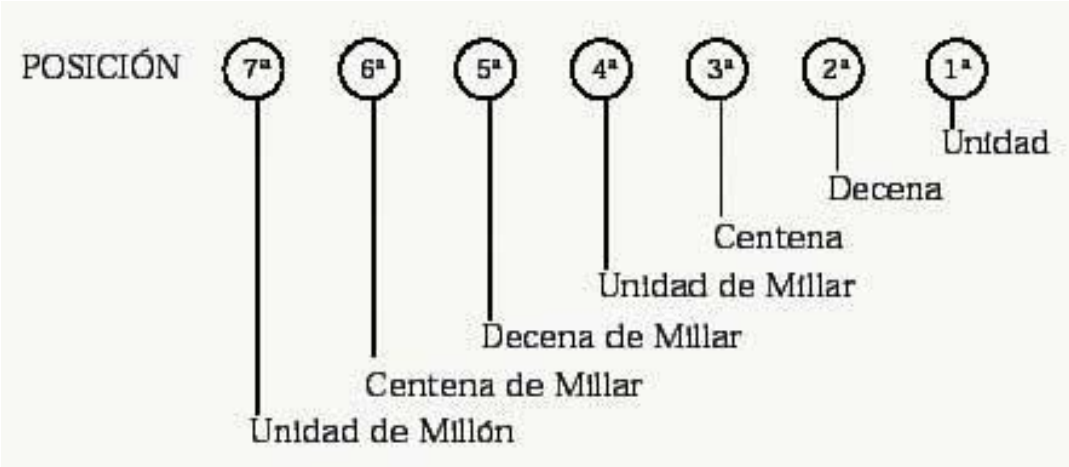
Utilizamos más simbolos 😊



10

Sistemas Posicionales

- Son aquellos donde la posición de un dígito cambia el peso que este representa en la cantidad
- Nuestro sistema Decimal es el mejor ejemplo

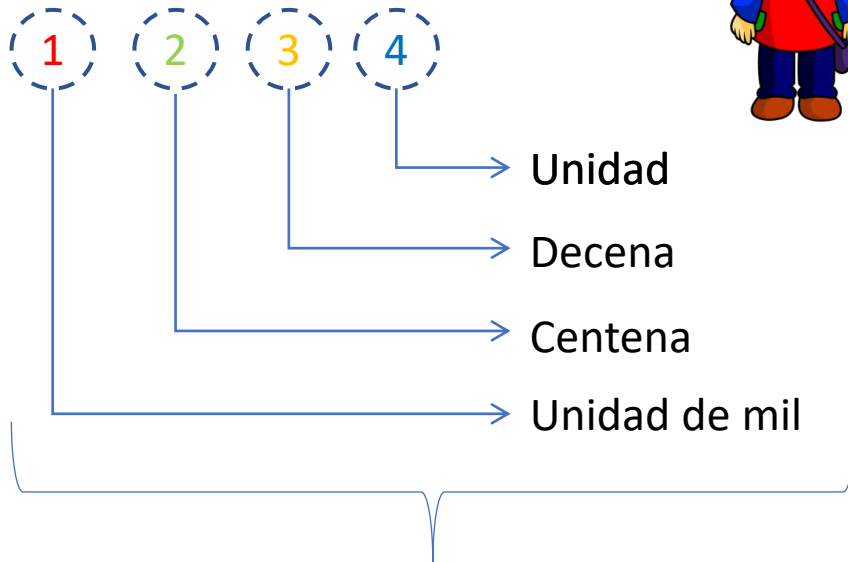


Sistemas **NO** Posicionales

- Son aquellos donde la cantidad representada por cada dígito es independiente de su ubicación
- Un claro ejemplo es el sistema numérico Romano

Numerales romanos						
Fundamentales				Secundarios		
I	X	C	M	V	L	D
1	10	100	1000	5	50	500

Volviendo a la primaria...
descompongamos el número
1234



$$1234 = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0$$

$$N = \sum_{i=-k}^{n-1} d_i \cdot 10^i$$

0, 1, 2, 3, 4, 5, 6, 7, 8, 9 → Base del sistema decimal

Generalizando → $\sum_{i=-k}^{n-1} d_i b^i$

Dígito: símbolo permitido con un valor determinado

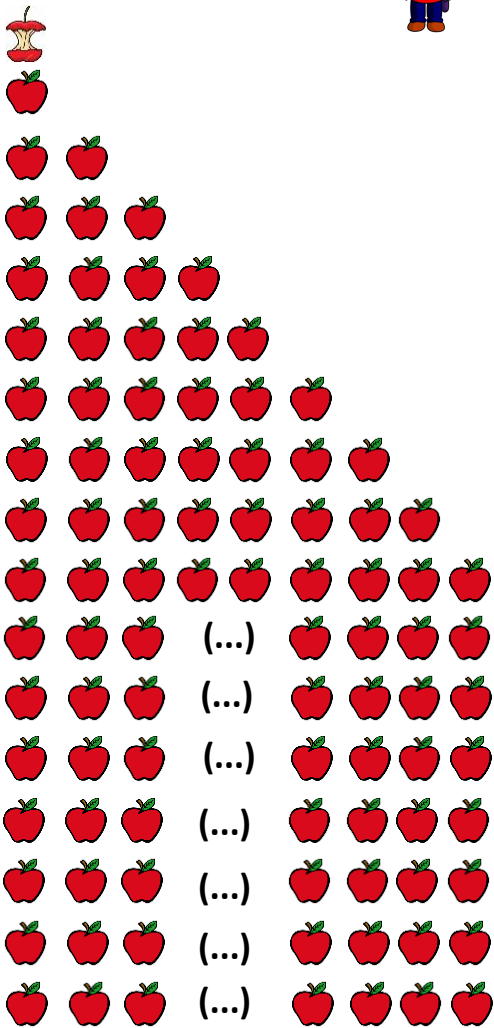
Base: cantidad de símbolos permitidos

Posición: ubicación del dígito dentro del número conformado

https://es.wikipedia.org/wiki/Sistema_de_numeraci%C3%B3n

Sistemas de numeración | otras bases

Cantidad



Decimal (Base = 10)

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Binaria (Base = 2)

0
1
10
11
100
101
110
111
1000
1001
1010
1011
1100
1101
1110
1111
10000

Octal (Base = 8)

0
1
2
3
4
5
6
7
10
11
12
13
14
15
16
17
20

Hexadecimal (Base = 16)

0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
10

Bit: **B**inari **di**gi**T**. Es la mínima unidad que se puede almacenar en un circuito electrónico.



¿3,3V o 0V?
¿Prendido o apagado?
¿1 o 0?

IMPORTANTE:

Toda la información se almacena en Bits en los dispositivos electrónicos. Es por eso que decimos que “todo es un número”

Byte: Conjunto de 8 bits

1	0	1	0	1	0	0	1
---	---	---	---	---	---	---	---

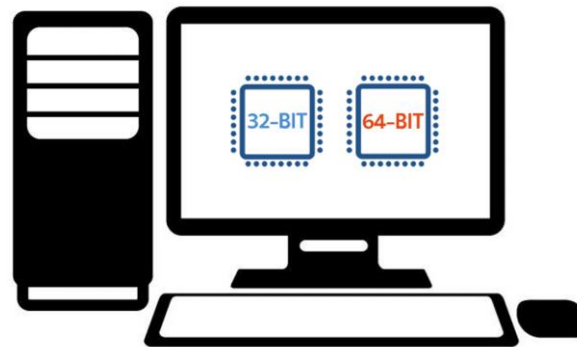
Nibble alto

Nibble bajo

Nibble: Conjunto de 4 bits de un byte

Word: Conjunto de bits que pueden ser accedidos en una única operación de lectura o escritura.

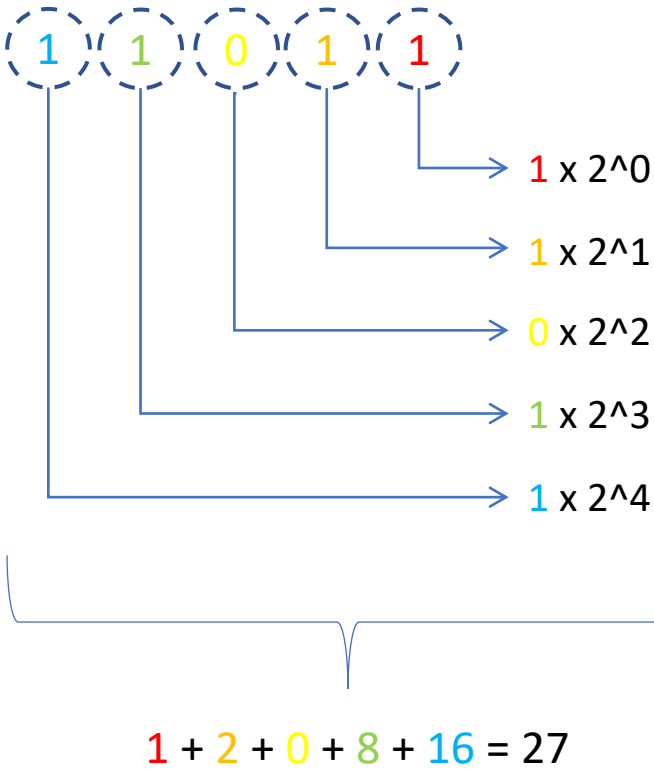
Tamaño del bus de datos.



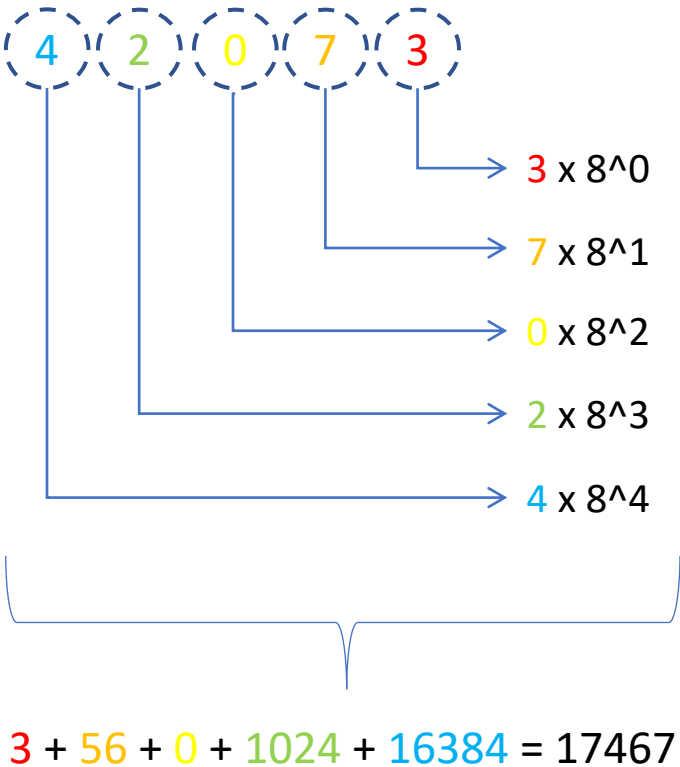
Nombre	Abrev.	Factor binario	Tamaño en el SI
bytes	B	$2^0 = 1$	$10^0 = 1$
kilo	k	$2^{10} = 1024$	$10^3 = 1000$
mega	M	$2^{20} = 1\,048\,576$	$10^6 = 1\,000\,000$
giga	G	$2^{30} = 1\,073\,741\,824$	$10^9 = 1\,000\,000\,000$
tera	T	$2^{40} = 1\,099\,511\,627\,776$	$10^{12} = 1\,000\,000\,000\,000$
peta	P	$2^{50} = 1\,125\,899\,906\,842\,624$	$10^{15} = 1\,000\,000\,000\,000\,000$
exa	E	$2^{60} = 1\,152\,921\,504\,606\,846\,976$	$10^{18} = 1\,000\,000\,000\,000\,000\,000$
zetta	Z	$2^{70} = 1\,180\,591\,620\,717\,411\,303\,424$	$10^{21} = 1\,000\,000\,000\,000\,000\,000\,000$
yotta	Y	$2^{80} = 1\,208\,925\,819\,614\,629\,174\,706\,176$	$10^{24} = 1\,000\,000\,000\,000\,000\,000\,000\,000$

Sistemas de numeración | Base X a Base 10

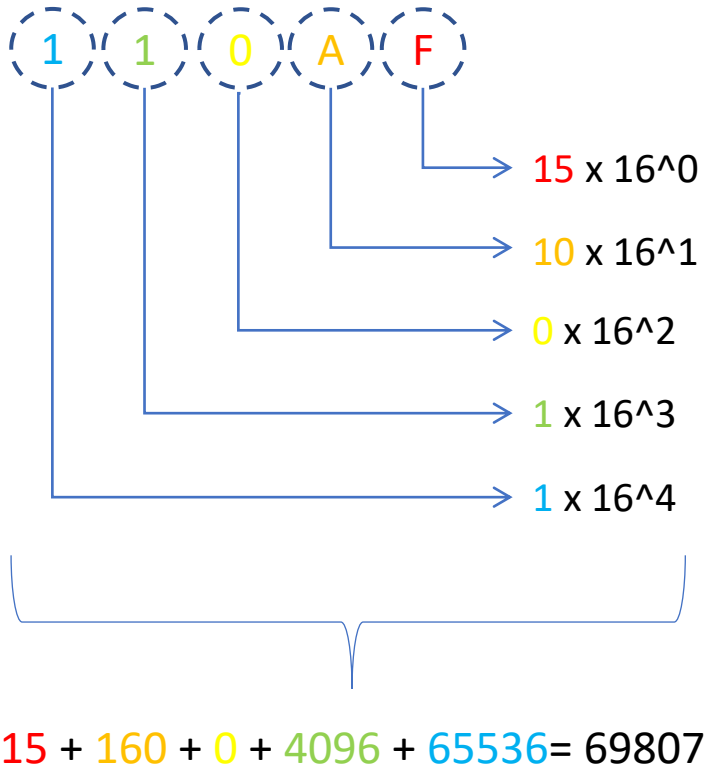
Binario



Octal

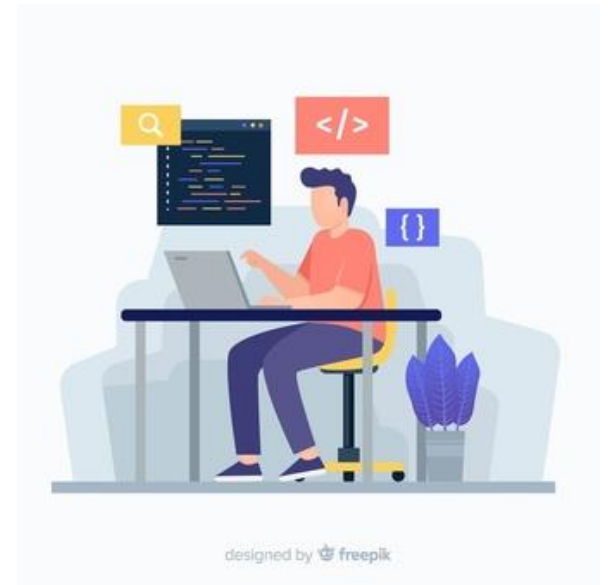


Hexadecimal




```
0100000000010100011011000000100101100011
11000101110100010001111111110100000100
001010010110000110101110110110110010001
110110000010101100100010000111000100111
01001100101101001101101001111011101110
000110100110100110110100011010
10010011010010011001010001110
10001001int main()
01010001{
111001100 printf("Hello World");
0100000111 return 42;
000110100010001110001101000011010
1001001101111010111011110000001010001110
100010010001010110010011101110100010111
01010001110011010101110001010100011000
1110011000001011111010100111110001100
010000011111101010010011010101110110110
```

¡A programar!



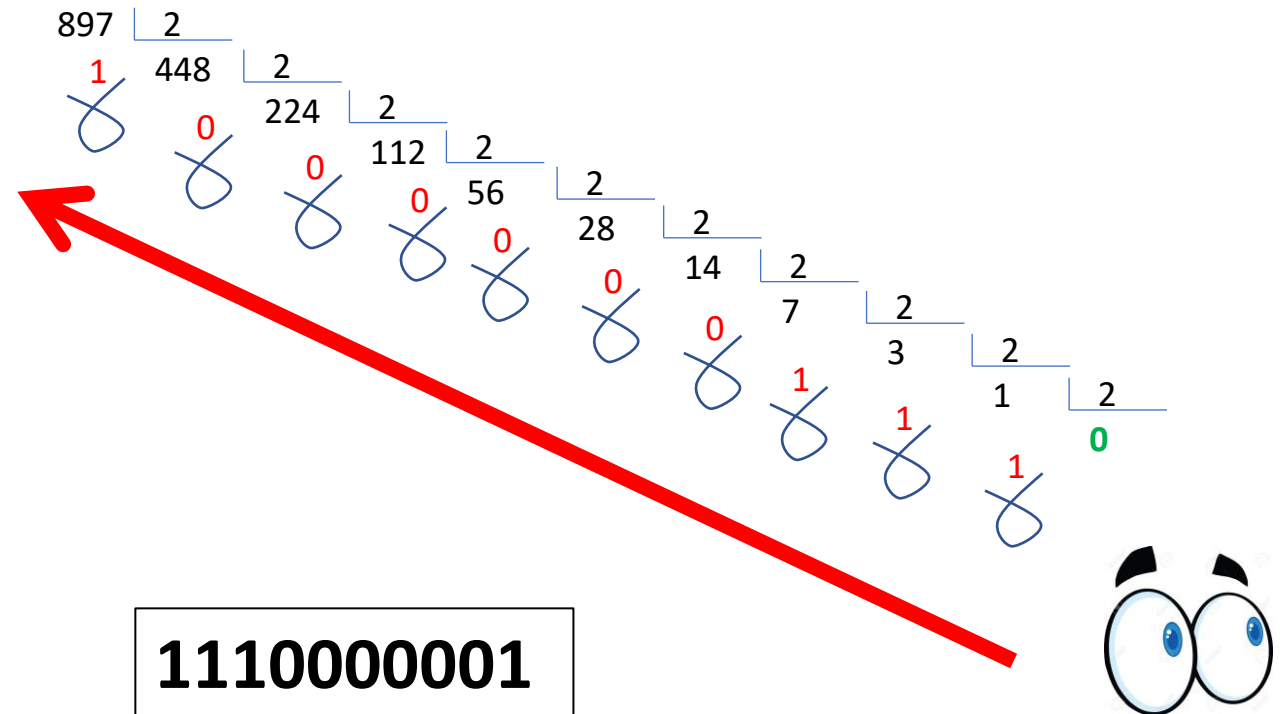
- Realizar una función que reciba un numero positivo en base 2 en formato string y devuelva la cantidad o -1 en caso de error
- Realizar una función que reciba un numero positivo en base 8 en formato string y devuelva la cantidad o -1 en caso de error
- Realizar una función que reciba un numero positivo en base 16 en formato string y devuelva la cantidad o -1 en caso de error
- Realizar una función que imprima un número en bases octal, decimal y hexadecimal

Método de las divisiones



- Dividimos al numero por la nueva base y nos quedamos con el resto
- Reéitimos el procedimiento con el cociente hasta llegar a 0
- Leemos el numero convertido de atrás hacia adelante

Ejemplo: 897 de base 10 a base 2



Método de las restas



- 1. Buscamos el mayor numero posible que sea potencia de 2 y a su vez sea menor que el número a convertir
- 2. Le restamos dicho número a nuestro valor
- 3. Repetimos 1 y 2 hasta llegar a 0
- 4. Aquellas potencias que hayamos restado, las marcamos en orden con un 1
- 5. Aquellas potencias que no hayamos restado las completamos con 0

Ejemplo: 897 de base 10 a base 2

potencia	valor	
0	1	✓ ④
1	2	✗
2	4	✗
3	8	✗
4	16	✗
5	32	✗
6	64	✗
7	128	✓ ③
8	256	✓ ②
9	512	✓ ①
10	1024	

- 1) $897 - 512 = 385$
- 2) $385 - 256 = 129$
- 3) $129 - 128 = 1$
- 4) $1 - 1 = 0$

1110000001

```
0100000000010100011011000000100101100011
11000101110100010001111111110100000100
001010010110000110101110110110110010001
1101100000101011001000100001110001001111
010011001011010011011010011110111011110
0001101001101001101101000011010
100100110100100011001010001110
10001001int main()
01010001{
111001100 printf("Hello World");
001000001 return 42;
0001101000100011100011000011010
1001001101111010111011110000001010001110
100010010001010110010011101110100010111
01010001110011010101110001010100011000
1110011000001011111010100111110001100
010000011111101010010011010101110110
```

iA programar!



- Realizar una función que reciba una cantidad y un array de char vacío; y coloque en éste un string con el número convertido en binario utilizando el método de las divisiones
- Repita el ejercicio anterior, pero utilizando el método de las restas
- Modifique la función anterior para que reciba un tercer parámetro, que indique la base de destino a la cual se desea convertir el número

Sistemas de numeración | pasajes entre bases

- Existen métodos sencillos para realizar pasajes de numeros entre bases que guardan relaciones de potencia entre ellas
- La relacion de potencia entre las bases nos da la relacion directa que hay entre las cantidades de dígitos
- $8 = 2^3$
 - Esto implica que tomando grupos de 3 digitos binaros podemos convertirlos en un único dígito octal
- $16 = 2^4$
 - Esto implica que tomando grupos de 4 digitos binaros podemos convertirlos en un único dígito Hexadecimal

Octal (Base = 8)

0
1
2
3
4
5
6
7
10
11
12
13
14
15
16
17
20

Binaria (Base = 2)

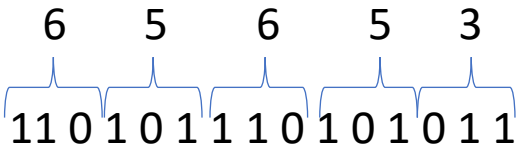
0
1
10
11
100
101
110
111
1000
1001
1010
1011
1100
1101
1110
1111
10000

Hexadecimal (Base = 16)

0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
10

Ejemplo a base octal

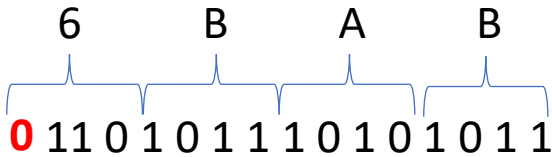
2	8
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7



El número convertido a base 8 es:
065653

Ejemplo a base Hexadecimal

2	16
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F



El número convertido a base 16 es:
0x6BAB

- Para representar numeros signados existen distintos formatos
 - **Signo y magnitud**
 - Binario desplazado
 - Complemento a 1
 - Complemento a 2

Signo y Magnitud

- La forma mas natural de pensarlo es **signo y magnitud**
- Del conjunto de bits disponibles, tomo uno para representar el signo. El resto representaran la magnitud
- Posee doble representacion del 0
- Los numeros más negativos parecen mayores que otros numeros. Se dificulta la comparación

10	2
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
-0	1000
-1	1001
-2	1010
-3	1011
-4	1100
-5	1101
-6	1110
-7	1111

Binario desplazado

- Buscando resolver las desventajas de SyM surge el binario desplazado
- Se coloca arbitrariamente el 0 al medio de la tabla
- No posee doble representacion del 0
- Los numeros negativos no aparentan ser mayores que los positivos
- Presenta una gran dificultad a la hora de realizar operaciones aritméticas

10	2
-7	0000
-6	0001
-5	0010
-4	0011
-3	0100
-2	0101
-1	0110
0	0111
1	1000
2	1001
3	1010
4	1011
5	1100
6	1101
7	1110

Complemento a 1

- Buscando resolver las desventajas de SyM y del binario desplazado
- Los numeros positivos se representan con su magnitud
- Los numeros negativos se representan mediante su complemento a 1 → cambiar los dígitos por su opuesto
- Doble representacion del 0

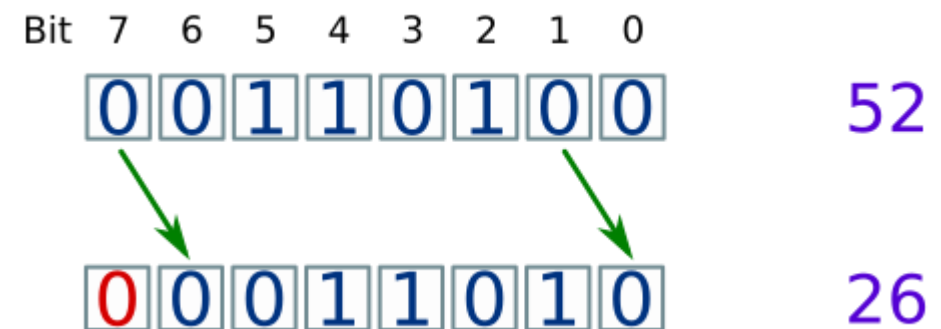
10	2
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
-0	1111
-1	1110
-2	1101
-3	1100
-4	1011
-5	1010
-6	1001
-7	1000

Complemento a 2

- Buscando resolver las desventajas de los anteriores
- Los numeros positivos se representan con su magnitud
- Los numeros negativos se representan mediante su complemento a 2 → realizar el complemento a 1 y sumar 1
- Tiene un valor mas hacia el negativo
- Éste es el mecanismo utilizado actualmente

10	2
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000

- Las operaciones aritméticas básicas son análogas en todas las bases
- Resulta interesante destacar que al multiplicar o dividir por la base, se “corre la coma” a la derecha o izquierda respectivamente.
 - Este concepto se aplica frecuentemente a nivel de bits desplazando los mismos a la izquierda o derecha para multiplicar o dividir por 2





Realizar las siguientes operaciones en todas las bases vistas:

- $1524 + 4573$
- $1024 * 8$
- $45218-4852$
- $54873+55481$
- $12548*16$
- $102458/2$

Con los conceptos incorporados, completar el siguiente cuadro y justificar la respuesta

tipo	rango
char	
unsigned char	
int	
unsigned int	

Verificar....



Punto fijo

- Una de las formas de representar a los números reales (con coma) es la denominada “punto fijo”
- El concepto radica en dejar reservada una determinada cantidad de bits para la parte fraccionaria
- Esto conlleva una pérdida de precisión
- Ej: representar el nro 25,69 en punto fijo, con 5 bits para parte entera y 3 para la parte decimal

32	16	8	4	2	1	0.5	0.25	0.125
0	1	1	0	0	1	1	0	1

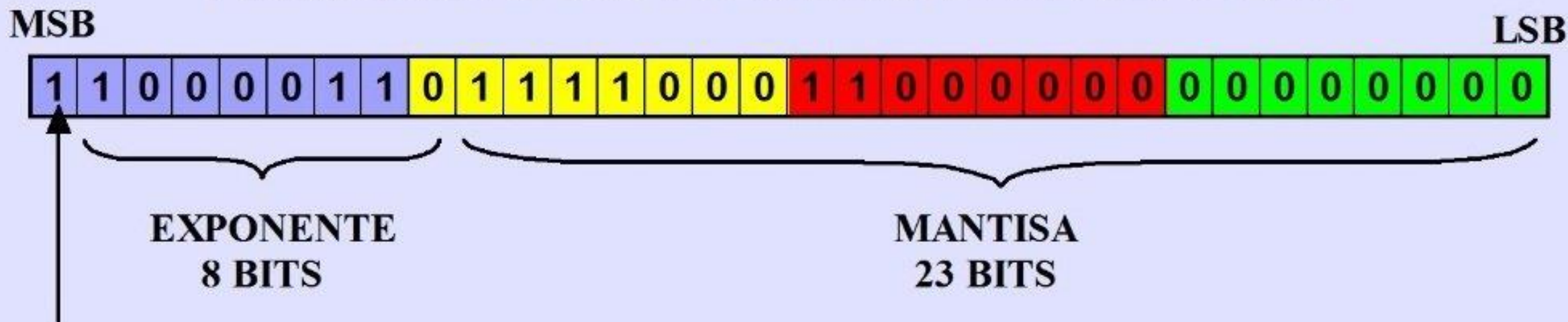
Nro representado: 25,625

Punto flotante

- Es la forma actualmente utilizada por las computadoras para representar los números reales
- El concepto es similar al anterior, solo que se reserva un número de bits para “mover” la coma de acuerdo a la necesidad
- El formato es similar al utilizado en la notación científica
- El signo se representa con un bit
- El exponente se representa con 8 bits en binario desplazado
- La mantisa son los 23 bits restantes, y le dan la precisión al número



FORMATO DE PUNTO FLOTANTE IEEE-754, 32 BITS



BIT DEL SIGNO
1= NEGATIVO
0=POSITIVO

EJEMPLO: -248.75
HEXADECIMAL: C3 78 C0 00

Clase	Exp	Fracción
Ceros	0	0
Números desnormalizados	0	distinto de 0
Números normalizados	1-254	cualquiera
Infinitos	255	0
NaN (Not a Number)	255	distinto de 0