Sistemas de Numeración

Descomponer un número por el peso de cada dígito.

1349

$9 \times 10^0 = 9$ Unidad

$4 \times 10^1 = 40$ Decena

$3 \times 10^2 = 300$ Centena

$1 \times 10^3 = 1000$ Unidad de Mil

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

El decimal es un sistema de numeración posicional

Posee diez símbolos o unidades que representan diez cantidades distintas

Al agotarse los símbolos, se agrega una columna a la izquierda del número

El peso de cada símbolo depende de la posición en la que se encuentre.

10 es la base del sistema

Generalizando la descomposición...

1349

$$9 \times 10^{0} = 9$$
$$4 \times 10^{1} = 40$$
$$3 \times 10^{2} = 300$$
$$1 \times 10^{3} = 1000$$

$$\text{Peso} = \text{Unidad} \times \text{Base}^{\text{Posición}}$$

0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000
17	10001

Qué pasa si tengo solamente dos símbolos, “1” y “0”?

Qué pasa cuando se me agotan los símbolos disponibles?

Y así sucesivamente...

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	
17	

Este sistema de numeración se llama binario

Es posicional al igual que el sistema decimal.

Cada símbolo o unidad representa una cantidad.

La posición del símbolo define el peso de cada columna

10 es la base del sistema.

(no es “diez”! es “uno-cero”)

Ejemplo

$$6_{10} = 0110_{2}$$

Peso = Unidad x Base $^{\text{Posición}}$

$$0 \times 2^{0} = 0$$
$$1 \times 2^{1} = 2$$
$$1 \times 2^{2} = 4$$
$$0 \times 2^{3} = 0$$

$$0+4+2+0 = 6$$

Entonces, cuánto “pesa” cada columna en binario?

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)

Un número binario de 8 bits es la definición de 1 byte

Cual es el rango de un número binario?

Mínimo valor: 0 en todas las columnas
Máximo valor: 1 en todas las columnas

8 bits	1 byte	0 ... 255
16 bits	2 bytes	0 ... 65,535
32 bits	4 bytes	0 ... 4,294,967,296

$0 \sim 2^n\text{-}1$

donde n es el número de dígitos o bits del número.

* válido sólo para números positivos

Como pasar de binario a decimal y vice-versa?

Ejemplo: representar 1001010b en decimal.

128	64	32	16	8	4	2	1
1	1	0	0	1	0	1	0

128+64+8+2 = **202**

Ejemplo: representar 97d en binario.

128	64	32	16	8	4	2	1
0	**1**	**1**	0	0	0	0	**1**

64+32+1 = **97**

Si necesito números más grandes... agrego más bits!

Y los números signados?

En binario no existe signo negativo.
El signo se representa con el bit más significativo.

S	2^6	2^5	2^4	2^3	2^2	2^1	2^0
--	**64**	**32**	**16**	**8**	**4**	**2**	**1**
Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)

Si S=0 es positivo, si S=1 es negativo.

Cómo se conoce el valor decimal un número signado?

Si es positivo, se hace **igual que un número no signado**.

Si es negativo, se utiliza el **CA2 (Complemento A 2)**

Ejemplo: representar **-**97d en binario signado

```
      97₁₀ = 01100001₂
=> CA1 = 10011110
=> CA2 =        + 1
         --------
       = 10011111₂ = -97₁₀
```

$$97_{10} = 01100001_2$$
$$\Rightarrow CA1 = 10011110$$
$$\Rightarrow CA2 = +1$$
$$= 10011111_2 = -97_{10}$$

El CA2 es el "equivalente binario" a multiplicar por -1

Y el rango de un número signado?

Puedo representar la misma cantidad de números (2^n), pero en un intervalo diferente.

No signado : $0 \dots 2^n-1$

Signado: $-2^{n-1} \dots 2^{n-1}-1$

8 bits	1 byte	0 ... 255	-128 ... 127
16 bits	2 bytes	0 ... 65,535	-32,768 ... 32,767
32 bits	4 bytes	0 ... 4,294,967,296	-2,147,483,648 ... 2,147,483,647

1 bit	---	0 ... 1	-1 ... 0

Como sé si un número binario es signado o no?

No lo sé.
Alguien me lo tiene que decir.

Al lenguaje C también.

Suma binaria

```
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10
```

Ejemplo

```
 01001110      78
+10011000     +152
--------     ----
 11100110     230
```

Resta binaria

Se computa como una suma de el minuendo más el CA2 del sustraendo (o bien: sumar el sustraendo x -1)

Ejemplo

```
  01001110          01001110          78
 -01101110         +10010010        -110
                   ---------        ----
CA1=10010001        11100000         -32
CA2=10010010

                  CA1=00011111
                  CA2=00100000=32d
```

Multiplicación y división

Multiplicar x $BASE^n$ = Deplazar n posiciones a la izquierda.

Dividir x $BASE^n$ = Desplazar n posiciones a la derecha.

Válido para todos los sistemas de numeración posicionales.

DEC	BIN	OCT
0	0000	0
1	0001	1
2	**0010**	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	**10**
9	1001	11
10	1010	12
11	1011	13
12	1100	14
13	1101	15
14	1110	16
15	1111	17
16	10000	20
17	10001	21

El Octal es un sistema de numeración posicional de base 8

Cumple con todas las reglas simbólicas y aritméticas anteriormente mencionadas

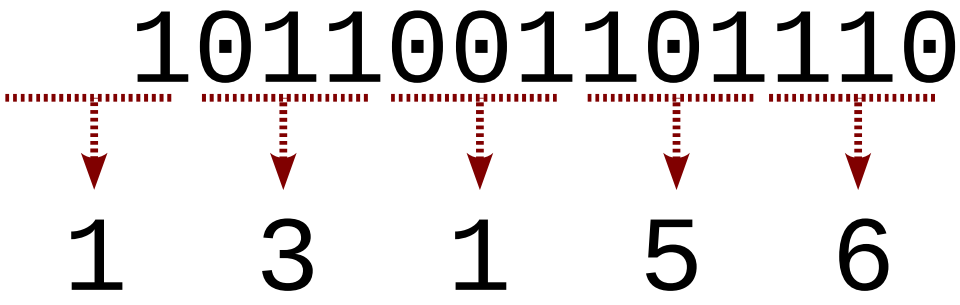
Como su base es 2^3, la conversión con el sistema binario es muy simple, agrupando de a 3 bits

10 es la base del sistema

DEC	BIN	OCT
0	0000	0
1	0001	1
2	**0010**	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	**10**
9	1001	11
10	1010	12
11	1011	13
12	1100	14
13	1101	15
14	1110	16
15	1111	17
16	10000	20
17	10001	21

Ejemplo

Convertir de binario a octal



13156_8

DEC	BIN	OCT	HEX
0	0000	0	0
1	0001	1	1
2	**0010**	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	**10**	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	**10**
17	10001	21	11

Hexadecimal es un sistema de numeración posicional de base 16

Cumple con todas las reglas simbólicas y aritméticas anteriormente mencionadas

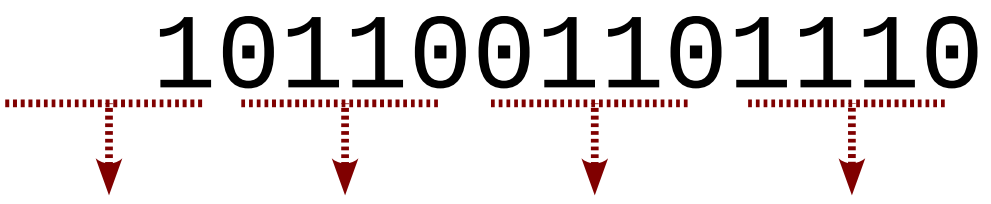
Como su base es 2^4, la conversión con el sistema binario es muy simple, agrupando de a 4 bits

10 es la base del sistema

DEC	BIN	OCT	HEX
0	0000	0	0
1	0001	1	1
2	**0010**	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	**10**	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	**10**
17	10001	21	11

Ejemplo

Convertir de binario
a hexadecimal



$$166E_{16}$$

La base es distinta

El número es el mismo

1011001101110_{2}

13156_{8}

$166E_{16}$

5742_{10}

En lenguaje C

```
int n;
-------
n=013156;
n=0x166e;
n=5742;
```

Preguntas?