I. FUNDAMENTOS

3. Representación de la información

Universidad de La Laguna

Curso de Acceso a la Universidad para Mayores de 25 y 45 (CAM 25-45)

Introducción a la Informática

Coromoto León Hernández
Gara Miranda Valladares
Carlos Segura González

Curso 2011/2012

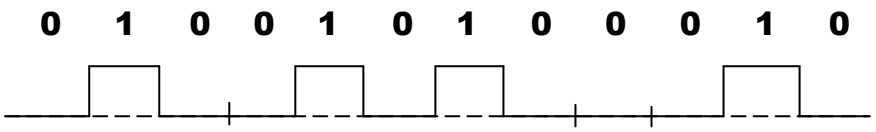
Contenido

1. Representación de la información en los sistemas informáticos.

Como ya hemos visto hasta el momento, la informática se centra principalmente en el tratamiento o procesamiento de la información. Para poder manipular la información los ordenadores necesitan utilizar un sistema de representación que les permita realizar distintas operaciones sobre los datos, así como almacenarlos físicamente.

Los humanos representamos la información mediante un conjunto de símbolos (alfabeto) que nos permiten crear palabras que se asocian a determinados conceptos. Agrupando y combinando estas palabras o conjuntos de símbolos somos capaces de expresarnos y de transmitir la información deseada. Para manejar información numérica utilizamos un sistema de representación compuesto por diez dígitos distintos. Este hecho no es algo casual: en los humanos es totalmente natural contar con los diez dedos de las manos.

Sin embargo, esta simbología que entendemos los humanos no es válida para los ordenadores. Los ordenadores son sistemas electrónicos formados por transistores cuyas líneas pueden estar en dos estados distintos: activas o inactivas. Eso significa que a pesar de que una línea eléctrica puede conducir un rango continuo de distintos voltajes, los sistemas electrónicos de un computador discretizan ese continuo y sólo consideran dos posibles estados, traduciéndose en unos y ceros. Esto quiere decir que los ordenadores utilizan sólo dos símbolos (“0” y “1”) para representar internamente cualquier tipo de información que se vaya a manipular, ya sean números, textos, imágenes o sonidos.



A este sistema de representación que utiliza dos dígitos para representar la información se le denomina **sistema binario**. Dado que los sistemas binarios forman parte de los denominados sistemas de numeración posicionales, vamos a ver algunas cuestiones generales sobre este tipo de sistemas.

2. Sistemas de numeración posicionales.

Un sistema de numeración en base ***b*** es una representación de números mediante un alfabeto compuesto por *b* símbolos, dígitos o cifras distintas. En estos sistemas todo número está representado por un conjunto de cifras, contribuyendo cada una de ellas con un valor que depende de:

- la cifra en sí,
- y de la posición que ocupa la cifra en cuestión dentro del número.

Para ver un ejemplo, recurriremos al sistema de numeración más conocido: el de base 10. A este sistema se le denomina sistema decimal porque cada cifra puede tomar diez posibles valores: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. Como veremos en el ejemplo siguiente, cada cifra del número **15.842** aporta un valor al número dependiendo no sólo de la cifra en sí, sino también dependiendo de la posición que ocupa la cifra dentro del número:

$$\begin{array}{r} 10.000 \\ 5.000 \\ 800 \\ 40 \\ 2 \\ \hline 15.842 \end{array}$$

Podemos decir entonces que el número 15.842 puede obtenerse como la suma de las cantidades 10.000, 5.000, 800, 40 y 2. Esto es:

$$\begin{aligned} 15.842 &= 10.000 + 5.000 + 800 + 40 + 2 = \\ &= 1 \times 10.000 + 5 \times 1.000 + 8 \times 100 + 4 \times 10 + 2 \times 1 = \\ &= 1 \times \mathbf{10^4} + 5 \times \mathbf{10^3} + 8 \times \mathbf{10^2} + 4 \times \mathbf{10^1} + 2 \times \mathbf{10^0} \end{aligned}$$

Como vemos en la descomposición anterior, cada cifra del número tiene un peso específico que se va multiplicando por la base, en este caso 10, cada vez que nos desplazamos hacia la izquierda.

Si generalizamos este funcionamiento, podríamos decir que en un sistema de numeración en base $b > 1$, todo entero N positivo tiene una única representación de la forma:

$$N = n_p b^p + n_{p-1} b^{p-1} + \ldots + n_1 b^1 + n_0 b^0$$

donde $0 \leq n_i < b$ para todo $i = 0, 1, \ldots, p.$

3. El sistema binario.

En el sistema de numeración binario la base es 2 (*b = 2*), por lo que se utiliza un alfabeto de tan sólo dos elementos {0, 1} para representar cualquier número. Los elementos de este alfabeto se denominan cifras binarias o bits.

En la tabla siguiente representamos los números enteros binarios que se pueden representar con 3 bits y que representan a los números decimales del 0 al 7:

Binario	Decimal
0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	7

Utilizando 3 bits hemos sido capaces de representar 8 números decimales distintos: {0, 1, 2, 3, 4, 5, 6, 7}. Si en lugar de utilizar sólo 3 bits, pasamos a utilizar 4 bits, podríamos representar un total de 16 números decimales distintos:

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

Si nos fijamos en las tablas anteriores, vemos que el número de bits utilizados en la representación determina el número de valores distintos que podemos representar:

- Con 3 bits podemos representar 8 valores distintos ($8 = 2^3$)
- Con 4 bits podemos representar 16 valores distintos ($16 = 2^4$)

Generalizando podemos decir que si utilizamos *n* bits, podremos representar 2^n valores distintos, que van desde el *0* al $2^n - 1$.

3.1. Convertir de binario a decimal.

Para convertir un número binario a un número decimal, basta con aplicar la fórmula general presentada para la representación de números en un sistema posicional, pero en este caso, teniendo en cuenta que la base es 2.

Como ejemplo vamos a ver cómo transformamos el número binario de seis dígitos **110101** a decimal:

$$110101 = 1 \times \mathbf{2^5} + 1 \times \mathbf{2^4} + 0 \times \mathbf{2^3} + 1 \times \mathbf{2^2} + 0 \times \mathbf{2^1} + 1 \times \mathbf{2^0} =$$

$$= 1 \times \mathbf{32} + 1 \times \mathbf{16} + 0 \times \mathbf{8} + 1 \times \mathbf{4} + 0 \times \mathbf{2} + 1 \times \mathbf{1} =$$

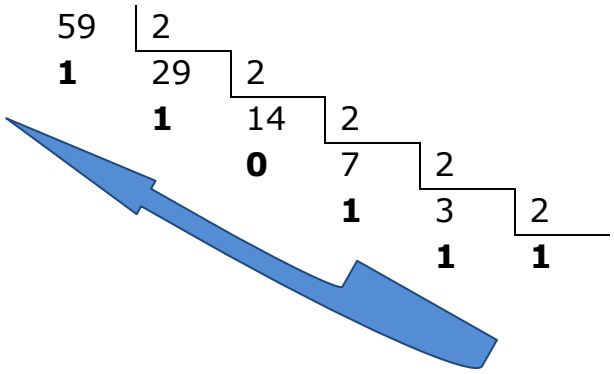
$$= 32 + 16 + 4 + 1 = 53$$

Si nos fijamos en el ejemplo, transformar de binario a decimal es bastante sencillo, pues basta con sumar los pesos en cuyas posiciones haya un 1.

3.2. Convertir de decimal a binario.

Para convertir un número entero decimal a binario hay que dividir el número de partida entre la base 2 (sin obtener decimales en el cociente) y seguir dividiendo entre 2 los cocientes que se vayan obteniendo sucesivamente. Los residuos de estas divisiones y el último cociente (que serán siempre menor que la base, esto es, 1 o 0) colocados en orden inverso al que han sido obtenidos determinarán la representación binaria del número decimal original.

Como ejemplo vamos a ver cómo convertir el número decimal 59 a binario:



Como podemos apreciar, la representación binaria del número decimal 59 es 111011:

$$111011 = 1 \times \mathbf{2^5} + 1 \times \mathbf{2^4} + 1 \times \mathbf{2^3} + 0 \times \mathbf{2^2} + 1 \times \mathbf{2^1} + 1 \times \mathbf{2^0} =$$

$$= 1 \times \mathbf{32} + 1 \times \mathbf{16} + 1 \times \mathbf{8} + 0 \times \mathbf{4} + 1 \times \mathbf{2} + 1 \times \mathbf{1} =$$

$$= 32 + 16 + 8 + 2 + 1 = 59$$

4. Unidades de información en los sistemas digitales.

Como hemos podido ver hasta el momento, la unidad mínima de información de un sistema digital es el **bit** (*Binary Digit*). Un bit es la cantidad mínima de información que puede almacenarse en un ordenador. Puede tener dos valores diferente: 0 ó 1.

La necesidad de codificar informaciones más complejas ha llevado a agrupar varios bits, apareciendo así las siguientes unidades:

- El byte es la cantidad de información que puede codificarse en 8 bits. Representa, por tanto, $2^8 = 256$ valores distintos. Se utiliza comúnmente como unidad básica de almacenamiento de información en los sistemas digitales.
- La palabra se define en relación con la computadora considerada, como la cantidad de información que la máquina puede manejar de una sola vez. Para evitar equívocos, se habla de palabras de 8 bits, 16 bits, 32 bits, etc.

Cuando se manejan grandes cantidades de información, no es viable seguir hablando de bytes, y por ello, se ha definido el conjunto de unidades siguientes:

Unidad	Abreviatura	Representa
Byte	B	2^0 = 1 byte
Kilo	Kb	2^{10} = 1024 bytes
Mega	Mb	2^{20} = 1024 Kb = 1.048.576 bytes
Giga	Gb	2^{30} = 1024 Mb = 1.073.741.824 bytes
Tera	Tb	2^{40} = 1024 Gb = 1.099.511.627.776 bytes
Peta	Pb	2^{50} = 1024 Tb = 1.125.899.906.842.624 bytes
Exa	Eb	2^{60} = 1024 Pb = 1.152.921.504.606.846.976 bytes
Zetta	Zb	2^{70} = 1024 Eb = 1.180.591.620.717.411.303.424 bytes
Yotta	Yb	2^{80} = 1024 Zb = 1.208.925.819.614.629.174.706.176 bytes

Teniendo en cuenta la tabla anterior, es sencillo pasar de una unidad a otra. Basta con que sepamos que para pasar de una unidad a su siguiente mayor magnitud basta con que dividamos la cantidad entre 1024, mientras que si, por el contrario, lo que queremos es pasar a la siguiente menor magnitud, tendremos que multiplicar por 1024.

Por ejemplo, si queremos saber cuántos Mb son 358.400 Kb, tendremos que dividir:

$$358.400 \text{ Kb} = 358.400 / 1024 = 350 \text{ Mb}$$

Por otro lado, si quisiéramos saber cuántos Gb son 23 Tb, tendríamos que multiplicar por 1024:

$$23 \text{ Tb} = 23 \times 1024 = 23.552 \text{ Gb}$$

5. Representación de números enteros.

Si se considera únicamente números enteros positivos, con N bits de espacio es posible representar los números de 0 a 2^N - 1. La forma más intuitiva de representarlos consiste en interpretar cada combinación mediante la cantidad que representa en binario. Por ejemplo, con 1 byte (es decir, con N = 8) se representarán los números del 0 al 255.

En el caso de los números enteros con signo, el sistema más común es el convenio de **magnitud y signo**. En este caso se reserva el primer dígito binario para codificar el signo. Suele representarse el signo positivo con un cero y el negativo con un uno, mientras que los restantes N - 1 dígitos se utilizan para representa el valor absoluto del número. Así, en este sistema el rango de valores que admite la representación es $[-(2^{N-1} - 1) .. 2^{N-1} - 1]$. Suponiendo que N = 8, podríamos representar el rango de valores [-127 .. +127]. Cabe destacar que en este case el número cero tiene dos representaciones: "-0" y "+0". Esto hace que desaprovechemos una codificación de todas las posibles.

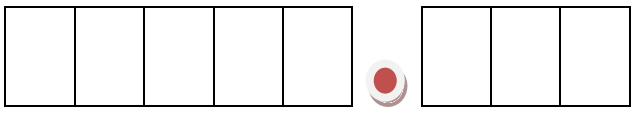
6. Representación de números reales.

Los humanos utilizamos fundamentalmente dos formas de representar los números reales: la notación tradicional o la científica. De cada una de ellas se deriva un tipo de representación utilizada en los ordenadores.

6.1. Coma fija.

Cuando representamos un real, normalmente usamos un carácter adicional (la coma) que separa la parte entera de la parte real. La representación en coma fija utiliza la misma idea, pero situando la coma en una posición fija dentro de la cadena o vector de bits que representan el número.

Por ejemplo, podemos usar 8 bits para representar un conjunto de números, donde los 5 primeros bits se utilizan para la parte entera, y los tres restantes para la parte real:



Parte entera Parte real

Utilizando esta representación, podríamos encontrarnos con los valores binarios 10101,110 ó 01100,001. Para convertir estos números a decimal, tenemos que tener en cuenta que los pesos de los valores que se encuentran a la derecha de la "coma" se van dividiendo entre la base (en el caso binario, la base es 2) conforme nos desplazamos a la derecha:

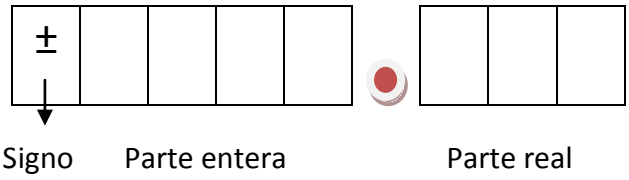
$$10101{,}110 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} =$$

$$= 16 + 4 + 1 + 1/2 + 1/4 = 21 + 0{,}5 + 0{,}25 = 21{,}75$$

$$01100{,}001 = 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} =$$

$$= 8 + 4 + 1/8 = 12 + 0{,}125 = 12{,}125$$

Nótese que esta representación puede extenderse a los números reales negativos, simplemente reservando el primer bit para la representación del signo, tal y como vimos para los números enteros:



Por otro lado, cuando se habla de la representación de números reales ya no podemos utilizar rangos para indicar los números que podemos representar con un conjunto determinado de bits. Es decir, en la representación anterior, donde tenemos el signo, cuatro bits para la parte entera y tres bits para la parte real, no podemos decir que se representa el rango de números reales del -15 al 15, pues por ejemplo, la representación utilizada no nos permite representar el número 8,2356. Por eso, lo que debemos decir es que la representación anterior no nos permite representar números mayores o iguales que 16, ni menores ni iguales que -16, ni números con más precisión de 0,125 (2^{-3}). Por este motivo se dice que el mayor inconveniente de esta representación es que tiene una muy limitada precisión, y que ésta es fija para todos los valores.

6.2. Coma flotante.

La coma flotante resuelve el problema de precisión de la representación en coma fija, permitiendo que la posición de la coma cambie. Esta codificación se basa en la notación científica:

$$0{,}00001 = 1{,}0 \times 10^{-5}$$

$$-5471000 = -5{,}471 \times 10^{6}$$

A esta codificación se le llama coma flotante (*floating point*) debido a que la coma decimal no se halla en una posición fija dentro de la secuencia de bits, sino que su posición se indica como una potencia de la base. Los números en coma flotante tienen tres componentes:

- El **signo**.
- La **mantisa**: indica la magnitud del número. En el caso del ejemplo anterior, las mantisas eran 1,0 y 5,471, respectivamente.
- El **exponente**: indica el valor al que está elevada la potencia. En el caso del ejemplo anterior, los exponentes eran -5 y 6, respectivamente.

Aunque la idea de la coma flotante es muy intuitiva, para su implementación es necesario tomar una serie de decisiones: cuántos bits utilizar para la representación de la mantisa, cuántos bits utilizar para la representación del exponente, cómo realizar las operaciones, cómo hacer los redondeos, etc. Por esto motivo, a lo largo de la historia de la informática se han implementado muchas versiones de este tipo de codificación. Una de las implementaciones más utilizadas es el estándar IEEE 754. Para la representación, este estándar requiere un total de 32 bits: 1 bit para el signo, 8 bits para el exponente y 23 bits para la mantisa.