Sistemas de numeración en la tecnología digital

Aarón CdC

December 13, 2021

Revisión 2

1 Sistema binario

Para explicar lo que es el sistema binario, tomemos por ejemplo un sistema de numeración que nosotros conocemos muy bien y que usamos a diario, que es el sistema decimal. Cuando nosotros contamos en decimal, contamos con un único dígito desde el número 0 hasta el número 9. Pero como es un sistema decimal, sólo tenemos diez números para contar por cifra. Así que cuando queremos pasar del 9 al siguiente valor, tenemos que añadir otro dígito a la izquierda. Que en éste caso sería el 10, y que representa cuántas vueltas hemos dado al primer dígito. Así volveríamos a contar desde el 10 al 19, y tendríamos que sumar 1 al segundo dígito para ir al siguiente número que es el 20. Repite ésto hasta el 99, y tendrás que añadir otro dígito a la izquierda para pasar al número 100. Y así constantemente.

El sistema de numeración binario es igual, pero al ser un sistema binario, sólo tenemos dos dígitos para contar (0 y 1). Así que la siguiente cifra después del 1, sería el 10. Después contaríamos 10, 11, 100, 101, 110, 111, 1000, y así constantemente.

DECIMAL	BINARIO
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
18	10010
19	10011
20	10100

Figure 1: Tabla con los 20 primeros números decimales y su representación binaria.

2 Conversión de binario a decimal

Convertir un valor binario en un valor decimal es sencillo. Dado que cada bit puede tener 2 valores, el valor de cada bit se representa en base 2 con exponente según la posición del bit (de derecha a izquierda), menos 1. Es decir, empezando por 2 elevado a 0 (Porque $2^0 = 1$). El valor de cada bit será 0 o 1, multiplicado por 2 elevado a la posición del bit.

$$bitval = \text{BIT} \in \{0 \vee 1\} * 2^{pos-1}$$

El valor del número binario en decimal será la suma de los valores de todos los bits cuyo valor sea 1.

$$\sum[\ bitval1\ ,\ bitval2\ \ldots\ bitvaln\ ]$$

La siguiente imagen ilustra el proceso de conversión de un número en binario (10110101) a su equivalente en decimal:

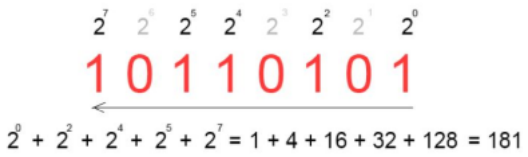

Figure 2: Proceso de conversión de binario a decimal.

En el caso del número binario 10110101, tomamos el valor de todos los bits que se encuentren en 1, que son los bits 0,2,4,5,7. El valor de éstos bits son 1,4,16,32 y 128 respectivamente. Si hacemos el sumatorio de todos éstos valores (sumamos todos los valores), el resultado es 181 en decimal.

3 Conversión de decimal a binario

Supongamos que queremos representar el valor 213 decimal en binario.

Para representar un valor en binario, primero debemos encontrar el número de bits necesarios para representarlo. Como el valor de cada bit es una potencia de 2, debemos encontrar la potencia de dos cuyo valor sea igual o inmediatamente superior al número que queramos representar. En éste caso, al ser 213, la potencia inmediatamente superior será 2 elevado a 8 (256). Es decir, 9 bits (recuerda, el primer exponente es 0). Pero, ¡Ojo! Porque hay que tener en cuenta si el valor decimal es igual o inferior a la potencia.

Si el valor de la potencia fuese igual al número que queremos representar, serían 9 bits. El último bit será un 1 y el resto serán ceros (100000000). Esto sería si quisiéramos representar el valor 256.

Pero como en éste caso el número es inferior ($213 < 256$), el último bit sería 0. En este caso, el número de bits será el exponente, menos 1. Ya que el último bit no cuenta. Es decir, que necesitamos 8 bits.

0 0 0 0 0 0 0 0

En el siguiente paso, vamos recorriendo todos los bits de izquierda a derecha. Es decir, de mayor valor (último), a menor valor (primero). El objetivo es colocar todos los bits a 1, siempre y cuando al hacerlo la suma del valor de los bits no exceda el valor decimal que queremos representar. Es decir, que la suma de los valores de cada bit no exceda, en éste caso, 213.

El último bit en éste caso tendría un valor de $2^7 = 128$. Como 128 es menor que 213, pondremos éste bit a 1.

1 0 0 0 0 0 0 0

El siguiente bit tiene un valor de $2^6 = 64$. $128 + 64 = 192$. Como 192 es menor que 213, ponemos éste bit también a 1.

1 1 0 0 0 0 0 0

El siguiente bit tendría un valor de 32. $192 + 32 = 224$. 224 es mayor que 213, con lo cuál éste bit se queda en 0.

1 1 ***0*** *0 0 0 0 0*

Seguimos repitiendo éste proceso con el resto de bits. El siguiente serían 16. $192 + 16 = 208$, $208 < 213$, con lo que se pondría a 1. El siguiente serían 8. $208 + 8 = 216$, $216 > 213$ (se queda a 0). El siguiente serían 4. $208 + 4 = 212$, $212 < 213$ (1). El siguiente serían 2. $212 + 2 = 214$, $214 > 213$ (0). Y por último $212 + 1 = 213$. $213 = 213$ (1).

1 1 ***0*** *1* ***0*** *1* ***0*** *1*

El valor resultante (11010101) es el valor binario de 213.

4 Aritmética binaria

La aritmética en binario funciona exactamente de la misma manera que en cualquier otro sistema de numeración, pero teniendo en cuenta de que cada dígito sólo tiene un rango entre 0 y 1, y que cada vuelta que demos añadirá o sustraerá 1 al acarreo del siguiente dígito.

Por ejemplo, si sumamos 1 a la cifra binaria 110, el resultado sería 111. Pero si sumamos 1 a la cifra 1011, el resultado sería 1100.

Por poner otro ejemplo, si quisiéramos sumar 101 (5) a la cifra 1011 (11), al igual que en decimal, sumamos cada dígito de menor a mayor (de derecha a izquierda), teniendo en cuenta el acarreo (lo que nos "llevamos" de la suma del anterior dígito para añadírselo al siguiente). En éste caso, empezamos por los bits más a la derecha, que sería 1+1=10. Como el valor es 10, nos llevamos 1 al acarreo.

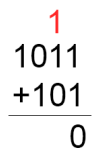

Figure 3: Proceso de suma binaria. Los números en rojo simbolizan el acarreo.

El siguiente bit sería 1+0, pero tenemos un 1 en el acarreo, así que el resultado sería 1+0+1 =10. Igualmente nos llevamos 1 al acarreo. Repetimos ésta acción con el resto de bits. 0+1+1=10, 1+1=10, y el último acarreo se lo añadimos al final al resultado. El resultado serían 10000 (16).

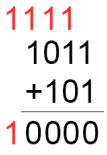

Figure 4: Suma binaria.

De igual manera, es posible realizar operaciones de resta, multiplicación y división de manera similar a como lo hacemos en decimal.

5 Números negativos en binario y Complemento a dos (C2)

Representar un valor negativo en binario es muy sencillo. Normalmente cuando representamos un valor negativo en decimal, usamos el símbolo "-". Por ejemplo, -20 sería un valor de 20 en negativo (por debajo de 0).

En binario, añadimos lo que se denomina un bit de signo. En éste caso, el valor 10100 (20), pasaría a ser 110100, siendo el último bit el bit de signo. De éste modo, todos los valores negativos empiezan con un uno.

Ésto tiene más sentido cuando trabajamos con un número de bits fijo. Y por eso, por norma general, solemos trabajar con un número de bits fijo. Siendo el último el bit de signo. Por ejemplo, cuando queremos representar un valor

negativo con un único byte (8 bits), el bit de mayor valor sería el de signo. Pudiendo así representar valores de entre -127 (10000000) hasta 127 (01111111), siendo el primer valor negativo (-1) 11111111.

Cuando trabajamos con valores de bits fijos, podemos invertir esos valores usando una técnica conocida como el complemento a dos.

El complemento a dos consiste en dos pasos: primero invertimos el valor de todos y cada uno de los bits (los unos pasan a ser ceros y los ceros pasan a ser unos). Y despues, al resultado de ésta operación, le sumamos un 1. El resultado será el valor invertido.

Por ejemplo, el número 27 en binario usando 8 bits con signo se representaría como 00011011.

Para representar su valor en negativo, primero invertimos todos los bits. Los ceros pasan a ser unos, y vice versa.

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

Al valor invertido, le sumamos 1.

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

El resultado (11100101) sería el equivalente en binario de -27. Puedes volver a realizar estos dos pasos sobre éste número (11100101) para comprobar que, en efecto, el resultado será otra vez 00011011 (+27).

6 Restas en binario usando C2

Cuando ambas cifras son positivas, podemos realizar una resta invirtiendo el valor de la segunda cifra y sumando ambos valores. Por ejemplo, para restar $30 - 24$, primero convertimos 24 (00001100) en su forma negativa usando C2 (11110100), y después sumamos (00011110 + 11110100) Sería el equivalente de realizar la operación $30 + -24$.

Pero, ¿Qué pasa con el acarreo final? Como tenemos un número fijo de bits (8 bits), en éste caso el acarreo se ignora, ya que queda fuera de los límites de bits disponibles. Es decir, que el resultado final de la operación sería 00000110 (6).

¿Por qué es ésto importante? Porque éste es el método que se usa en los componentes como los procesadores a la hora de restar.

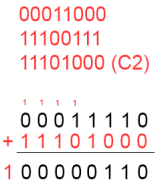

Figure 5: Proceso de resta usando complemento a dos. Primero se invierte una de las dos cifras, y despues se suman.

7 Numeración en Octal

De igual forma que en binario tenemos un sistema de numeración con dos estados por dígito, en octal tenemos hasta 8 valores por dígito, del 0 al 7. Es decir, contamos del 0 al 7 para pasar al 10. Los valores de cada dígito son en éste caso 8 elevado a la posición del dígito (empezando siempre por 0). Es decir, $8^0 = 1$, $8^1 = 8$, $8^2 = 16$, etcétera.

La razón por la que se usa la numeración octal es porque a veces queremos usar grupos de por ejemplo 3 bits, con valores del 0 al 7. El ejemplo más claro de ésto es el sistema de permisos en UNIX (Y los sistemas basados en Linux), donde el valor de los permisos se representa en octal y cada cifra representa los permisos para un usuario o grupo específico.

En el sistema de permisos UNIX, hay 3 bits donde cada bit representa un parámetro (lectura, escritura y ejecución). Los bits que se encuentren a 1 permiten al usuario o grupo realizar dicha acción. Hay 3 grupos con 3 conjuntos de 3 bits cada uno (usuario, grupo y "otros").

Como son 3 bits, el rango máximo que podemos representar numéricamente por grupo es del 0 al 7. Por eso se usa el sistema octal en éste caso.

8 Sistema de numeración hexadecimal

El sistema hexadecimal es el más importante, después del binario. Al igual que en los anteriores casos, cada dígito nos permite contar desde 0 hasta, en éste caso, 15. Para pasar al 10.

Pero, ¿Cómo representamos cifras superiores al 9 en éste sistema? Como nuestra numeración sólo dispone de 10 dígitos del al 9, usamos caracteres alfanuméricos para representar las cifras del 10 al 15. Es decir, A representaría el 10, B el 11, C el 12, D el 13, E el 14, y F el 15. De éste modo, decimos que podemos contar desde 0, hasta F, para pasar a 10.

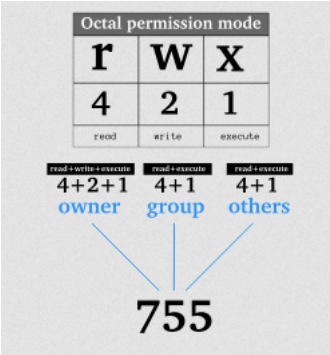

Figure 6: Permisos en UNIX

En informática se usa el hexadecimal para representar el valor de cada byte con sólo dos cifras. Del 0 al FF (255). A la hora de representar multitud de valores en una pantalla, es mucho más fácil de organizar y representar de ésta manera, ayudando a que su lectura sea más simple.

9 Conversión de decimal a hexadecimal y hexadecimal a decimal

Para convertir un valor decimal a hexadecimal, primero convertimos el valor decimal en un valor binario, y después dividimos dicho valor en grupos de 4 bits, de derecha a izquierda. Si alguno de los grupos no tiene cuatro bits, le añadimos tantos ceros como sean necesarios a la izquierda.

Por ejemplo, para representar el número 137 en hexadecimal, primero lo convertimos a binario siguiendo los pasos ya mencionados.

$$137 = 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$

Dividimos la cifra en grupos de 4 bits, de derecha a izquierda. Recuerda, si alguno de los grupos a la izquierda no tuviese cuatro bits, añádele ceros a la izquierda hasta que tenga al menos 4 bits.

$$1\ 0\ 0\ 0\ 1\ 0\ 0\ 1$$
$$1\ 0\ 0\ 0 \quad 1\ 0\ 0\ 1$$

Como trabajamos ahora con grupos de 4 bits, el valor de cada grupo de cuatro bits (nibble) tiene un rango de valores del 0 al 15. En hexadecimal, ésto es del 0 al F. Simplemente representa el valor de cada nibble en hexadecimal.

$$1\ 0\ 0\ 0 = \boldsymbol{8}$$
$$1\ 0\ 0\ 1 = \boldsymbol{9}$$

El número 137 en hexadecimal sería 89.

Para representar los valores en hexadecimal, normalmente añadimos el prefijo 0x a la izquierda, o una 'H' a la derecha (89H, 0x89).

Para convertir cualquier valor hexadecimal a decimal, seguimos los pasos inversos. Por ejemplo, para pasar el número 0x5A a decimal, primero pasamos los dígitos 5 y A a binario.

$$\boldsymbol{5} = 0\ 1\ 0\ 1$$
$$\boldsymbol{A} = 1\ 0\ 1\ 0$$

Unimos ambas cifras binarias y pasamos el valor binario a decimal

$$0\ 1\ 0\ 1\ 1\ 0\ 1\ 0$$
$$2^1 + 2^3 + 2^4 + 2^6 = 2 + 8 + 16 + 64 = 90$$

10 Final del artículo

11 Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

Eres libre de:

- **Compartir** — Copiar y redistribuir el material en cualquier medio y formato.
- **Adaptar** — Remezclar, transformar, y añadir al material.

El licenciador no puede revocar estos derechos siempre y cuando sigas los términos de licencia. Bajo los siguientes términos:

- **Atribución** — Debes de otorgar los créditos apropiados, proveer un enlace a la licencia, e indicar si se han realizado modificaciones sobre la obra. Puedes hacer esto de cualquier forma razonable, pero no de una forma que sugiera que tienes el aval del autor original.
- **No Comercial** — No está permitido el uso de esta obra con fines comerciales (venta, monetización, NFTs, etcétera).
- **Compartir Igual** — Si remezclas, modificas o añades contenido basado en esta obra, debes distribuir tu contribución bajo la misma licencia de la obra original.
- **Sin restricciones adicionales** — No puedes aplicar ni añadir clausulas legales (Como un copyright o similar) o medios tecnológicos (DRM, etc) que restrinjan a terceras partes de hacer cualquier cosa que ésta licencia les permita hacer.

Para mayor información, visita la página web `https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode`

Esta publicación está creada por Aarón CdC para la web de ElInformatico (https://www.elinformati.co) y su distribución es gratuita. Eres libre de usar y distribuir esta publicación como desees, pero siempre bajo los términos de la licencia mencionada anteriormente. Así mismo, no podrás hacer un uso inadecuado de esta obra, de un modo que vulnere la ley o que supere la línea de lo que es moralmente aceptable.

El acceso a esta publicación debe ser siempre libre y gratuito, y no se podrá revocar el acceso al mismo por motivos culturales, políticos, de raza, de sexo o género, o por cualquier otro tipo de motivo discriminatorio.

Si no estás de acuerdo con los términos de esta licencia, deberás de detener su uso y borrar todas las copias de esta publicación de forma inmediata.