Tema 2_1

SISTEMAS DE REPRESENTACIÓN

SI | 23-24

Índices

Sistemas de numeración posicionales

Un código o sistema de numeración es un conjunto de símbolos y reglas que se utilizan para representar cantidades.

En todos los sistemas de numeración existe un elemento que caracteriza al propio sistema y se le da el nombre de base del sistema de numeración. La base del sistema de numeración, es el número de símbolos dis-tintos que se utilizan para poder representar la información en ese sistema determinado, por ejemplo el sistema decimal tiene base 10, es decir utiliza 10 sím-bolos distintos (del 0 al 9), el sistema binario tiene base 2 y utiliza 2 símbolos (0 y 1).

Otro concepto que debemos conocer cuando hablamos de un sistema de numeración es el rango de representación, que es el conjunto de cantidades posibles que podemos represen-tar dado un número de cifras determinado (n).

El rango de representación se determina elevando la base del sistema de representación al número de cifras que se vayan a utilizar en la codificación.

Por ejemplo, en un sistema de representación de 4 cifras (posiciones) en base 2, su rango de representación es: 24 = 16, es decir, 16 es el número de combina-ciones distintas que podemos hacer con 4 posiciones y dos símbolos distintos de representación.

Este tipo de sistema de numeración recibe el nombre de sistema de numeración posicional.

Sistema de numeración posicional: Es aquel que al representar una cantidad mediante una cadena de símbolos, el significado de cada uno de los símbo-los que la forman varían en función de la posición que ocupen dentro de la cadena.

Por ejemplo: si tenemos los números 84 y 48, vemos que el 84 tiene una cadena de símbolos el 8 y el 4, mientras que el 48 tiene una cadena de símbolos compuesta por el 4 y el 8. Cada uno de esos símbolos tiene un valor distinto dependiendo de su posición dentro de la cadena que forma el número.

En el 84 el 4 representa las unidades y el 8 las decenas. En el 48 el 8 representa las unidades y el 4 las decenas. Lo cual significa que en función de la posición que ocupe el número dentro de la cadena, su valor varía.

A continuación vamos a ver algunos sistemas de numeración posicionales:

EL SISTEMA DECIMAL

Es el que entendemos y utilizamos todos los humanos de forma habitual, es un sistema de numeración en base 10. Utiliza 10 símbolos (del 0 al 9) para representar cualquier cantidad.

Su rango de representación será: 10^n , donde n es el número de cifras o posiciones que se vayan a utilizar.

Este sistema de numeración es posicional, el dígito más a la derecha representa las unidades y queda multiplicado por 1 (para las unidades sería 10^0), el siguiente dígito representa a las decenas quedando multiplicado por 10 (10^1) así sucesivamente.

EJEMPLO: Descomposición del número decimal 433.

$4 \times 10^2 + 3 \times 10^1 + 3 \times 10^0 = 400 + 30 + 3 = 433$.

Por ser el sistema que conocemos todos será el utilizado para conocer cualquier cantidad representada en otro sistema de numeración.

Es muy cómodo para los hombres pero muy ineficiente para los ordenadores.

EL SISTEMA BINARIO

El ordenador utiliza internamente este sistema de numeración. Es un sistema de numeración en base 2. Utiliza únicamente 2 símbolos (el 0 y el 1) para representar cualquier cantidad. Cada uno de los dígitos que componen el número representado en este sistema se le denomina Binary digit.

El valor posicional de un dígito dentro de un número binario se basa en la progresión de potencia de 2.

EJEMPLO: La representación del número decimal 13 en binario es 1101, siendo su descomposición la siguiente:

$13\ (\text{base}10) = 1101(\text{base}\ 2) = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 8+4+0+1 = 13$

La **lógica binaria** trabaja con variables y operaciones **binarias.**

Las variables sólo tomarán dos valores discretos:

- 1 y 0
- V (verdadero) y F (falso);
- sí y no

0	1	10	11	100	101	110	111	1000	1001	1010
0	1	2	3	4	5	6	7	8	9	10
1011	1100	1101	1110	1111	10000	10001	10010	10011	10100	
11	12	13	14	15	16	17	18	19	20	

SISTEMA OCTAL

Es un sistema de numeración en base 8 que utiliza 8 símbolos (del 0 al 7) para representar cualquier cantidad.

El valor posicional de un dígito dentro de un número en base octal se basa en la progresión de potencia de 8.

EJEMPLO: La representación del número decimal 78 en octal es 116, siendo su descomposición la siguiente:

$1 \times 8^2 + 1 \times 8^1 + 6 \times 8^0 = 64 + 8 + 6 = 78$.

0	1	2	3	4	5	6	7	10	11	12
0	1	2	3	4	5	6	7	8	9	10
13	14	15	16	17	20	21	22	23	24	
11	12	13	14	15	16	17	18	19	20	

SISTEMA HEXADECIMAL

Es un sistema de numeración en base 16 y utiliza 16 símbolos (del 0 al 9 y las letras A,B,C,D,E,F) para representar cualquier cantidad. Cada una de las letras representa un valor, A=10; B=11; C=12; 0=13; E=14; F=15.

El valor posicional de un dígito dentro de un número en base hexadecimal se basa en la progresión de potencia de 16.

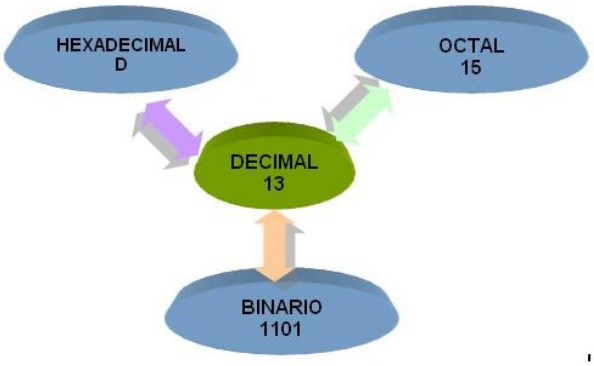
EJEMPLO: La representación del número decimal 78 en hexadecimal es 4E, siendo su descomposición la siguiente:

$4 \times 16^{\wedge}{}_1 + 14 \times 16^{\wedge}{}_0 = 64 + 14 = 78$.

0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
A	B	C	D	E	F	10	11	12	13	14
10	11	12	13	14	15	16	17	18	19	20

CONVERSIÓN DE DECIMAL A OTROS SISTEMAS DE NUMERACIÓN

A continuación vamos a ver cómo cambiar de un sistema de numeración a otro sistema que utilice diferente número de símbolos (base).



CONVERSIÓN DE CUALQUIER SISTEMA A DECIMAL

Consiste en transformar una cantidad dada o expresada en un sistema de numeración concreto en otra cantidad expresa-da en el sistema decimal y que ambas sean equivalentes.

Para la conversión de un número en cualquier base a deci-mal se aplicará el Teorema Fundamental de la Numeración que relaciona una cantidad expresada en cualquier sistema de numeración con su equivalente en base 10.

Teorema fundamental de la numeración:

Dado un número de n cifras con las cifras Xn,.... X2,X1,X0 que está en base B, su valor decimal equivalente viene representado por la fórmula:

Xn × Bn + ... + X2 × B2 + X1 × B1 + X0 × B0

Ejemplo: Convertir el número 56 que está expresado en octal (base 8), a decimal.(base 10)

6 x 80 + 5 x 81 = 6 + 40 = 46 (decimal)

CONVERSIÓN DE DECIMAL A OTROS SISTEMAS DE NUMERACIÓN

El procedimiento general para pasar de decimal a cualquier sistema de numeración, consiste en:

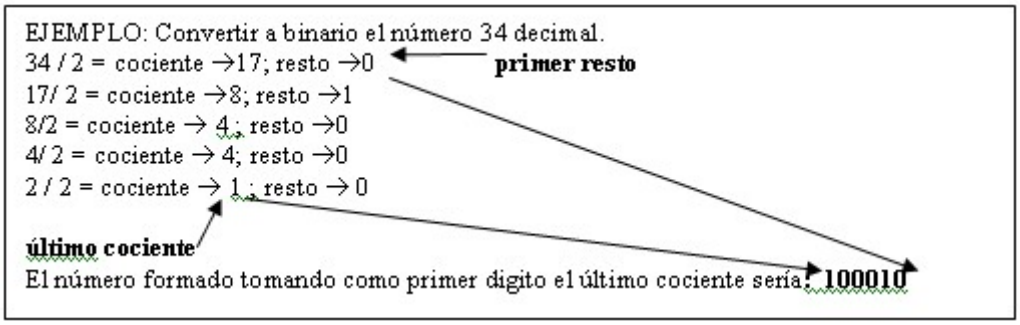
- Realizar divisiones sucesivas del número en decimal entre la base del sistema de numeración a la que queramos cambiar (si es a binario entre 2, si es a octal entre 8, si es a hexadecimal entre 16, ...), hasta que el cociente de la división sea menor que la base del sistema de numeración al que queremos convertir el número (si es a binario, hasta que el cociente sea menor que 1, si es a octal hasta que el cociente sea menor que 8, si es a hexadecimal, hasta que el cociente sea menor que 16.....)
- El número convertido a la base deseada, se forma tomando el último cociente, que será la cifra más a la izquierda, y los restos sucesivos, empezando por el último, de forma que la cifra más a la derecha del número será el primero de los restos.

Vamos a ilustrarlo con ejemplos de conversión entre los diferentes sistemas:

Conversión decimal-binario

Se realiza mediante divisiones sucesivas entre 2. Los pasos a seguir son:

1. Se divide el número entre 2 sucesivamente.
2. Los sucesivos cocientes se siguen dividiendo entre 2 hasta que el cociente sea menor que dos.
3. El número se formará cogiendo el último cociente y los distintos restos de derecha a izquierda, siendo el primer dígito (el dígito más a la izquierda) del número formado, el último cociente y el último dígito el primer resto.



Conversión binario-decimal

Se realiza aplicando el teorema fundamental de la numeración.

EJEMPLO: Convertir a decimal el número 100010 binario.

Teorema fundamental de la numeración $X_n \times B^n + ... + X_2 \times B^2 + X_1 \times B^1 + X_0 \times B^0$ donde B=2 (binario) y X_0=0, X_1=1, X_2=0, X_3=0, X_4=0, X_5=1, que son las cifras del número.

Aplicándolo obtenemos:

$0 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 0 \times 2^3 + 0 \times 2^4 + 1 \times 2^5 = 0 + 2 + 0 + 0 + 0 + 32 = 34$.

decimal-octal

Se realiza mediante divisiones sucesivas entre 8. Los pasos a seguir son:

1. Se divide el número dado entre 8.
2. Los sucesivos cocientes se siguen dividiendo entre 8 hasta que el cociente sea menor que 8.
3. El número se formará cogiendo el último cociente y los distintos restos de derecha a izquierda, Siendo el primer dígito del número formado el último

cociente y el último dígito del número formado el primer resto. (El procedimiento es el mismo que para pasar de decimal a binario)

EJEMPLO: Pasar a octal el número 32 decimal.

32/8 = cociente -> 4; resto -> 0

El número formado es el 40.

Conversión octal-decimal

Se realiza aplicando el teorema fundamental de la numeración:

EJEMPLO: Pasar a decimal el número 40 octal

$0 \times 80 + 4 \times 81 = 0 + 32 = 32$.

Conversión decimal-hexadecimal

Se realiza mediante divisiones sucesivas entre 16. Los pasos a seguir son exactamente iguales que para la conversión a binario y octal.

EJEMPLO: Pasar a hexadecimal el número 31 decimal.

31/16 = cociente -> 1; resto -> 15 (Recordamos que 15 es F en hexadecimal) El número formado es el 1F.

Conversión hexadecimal-decimal

Se realiza aplicando el teorema fundamental de la numeración:

EJEMPLO: Pasar a decimal el número 1F hexadecimal. (Recordamos que 15 es F en hexadecimal)

$15 \times 160 + 1 \times 161 = 15 + 16 = 31$.

CONVERSIÓN ENTRE OTROS SISTEMAS DE NUMERACIÓN

Pasar de binario a octal

Agrupando los dígitos binarios en tríos usando la siguiente tabla de conversión:

000->0
001->1
010->2
011->3
100->4

101->5

110->6

111->7

Ejemplo: Pasar de binario a octal el numero 101010101011100001

101010101011100001 =>

- 101 010 101 110 100 001 =>
- 5 2 5 6 4 1 => El número octal es 525641

Pasar de octal a binario

La forma de pasarlo es Usar la Anterior tabla de conversión para convertir cada número octal en uno binario.

Ejemplo: Pasar de octal a binario: 125401 =>

- Usamos la conversión 1=001 / 2=010 / 5=101
- 4=100 / 0=000 / 1=001
- El número final es 001010101100000001

Pasar de binario a hexadecimal

La forma de pasarlo es agrupando los dígitos binarios en cuartetos.

Posteriormente usando la siguiente tabla de conversión:

0000->0 // 1000->8

0001->1 // 1001->9

0010->2 // 1010-> A (10)

0011->3 // 1011-> B (11)

0100->4 // 1100-> C (12)

0101->5 // 1101-> D (13)

0110->6 // 1110-> E (14)

0111->7 // 1111-> F (15)

Ejemplo pasar de binario a hexadecimal el número: 10101010101110000111

- 10101010101110000111 =>
- 1010 1010 1110 1000 0111 =>
- A A E 8 7 => El numero hexadecimal es AAE87

Pasar de hexadecimal a binario

La forma de pasarlo es Usar la Anterior tabla de conversión para convertir cada número hexadecimal en cuatro binarios.

Ejemplo pasar de hexadecimal a binario el número: 124ª

- 124A =>
- Usamos la conversión 1=0001 / 2=0010 / 4=0100 / A=1010
- El número final es 0001001001001010

LÓGICA BINARIA

Operaciones Binarias AND

La función AND es equivalente a la conjunción "Y" de nuestra lengua, denominada también multiplicación lógica. Al aplicar esta función sobre dos variables (A y B), el resultado (S) será el siguiente: El resultado será verdadero si y sólo si, A y B son verdaderos. Es decir:

AND

$$\begin{aligned} 0 \cdot 0 &= 0 \\ 0 \cdot 1 &= 0 \\ 1 \cdot 0 &= 0 \\ 1 \cdot 1 &= 1 \end{aligned}$$

Operaciones Binarias OR

La función OR es equivalente a la conjunción "O" de nuestra lengua, también denominada suma lógica. Al aplicar esta función sobre dos variables (A y B), el resultado (S) será el siguiente: Si al menos una de las dos variables tiene un valor verdadero (1), entonces el resultado será verdadero. Para esta función con dos variables son posibles cuatro combinaciones, o sea:

OR

$$\begin{aligned} 0 + 0 &= 0 \\ 0 + 1 &= 1 \\ 1 + 0 &= 1 \\ 1 + 1 &= 1 \end{aligned}$$

Operaciones Binarias NOT

Si la variable A es verdadera, el resultado será falso y viceversa:

NOT

$$\bar{0} = 1$$
$$\bar{1} = 0$$

PROPIEDADES BINARIAS

- Conmutativa
- Asociativa
- Distributiva
- Ley de Morgan
- Doble Negación

Propiedad conmutativa

El resultado no depende del orden.

- A+B = B+A
- A*B = B*A

Propiedad asociativa

El resultado no depende del modo de asociación.

- A+(B+C)=(A+B)+C=A+B+C
- A*(B*C)=(A*B)*C=A*B*C

Propiedad distributiva

Una operación se distribuye en una asociación

- A*(B+C)=A*B+A*C
- A+(B*C)=(A+B)*(A+C)

Leyes de Morgan

En ingeniería electrónica e informática, la ley de De Morgan se escribe comúnmente como:

$$\overline{A \cdot B} \equiv \overline{A} + \overline{B}$$
$$\overline{A + B} \equiv \overline{A} \cdot \overline{B}$$

Que se puede leer como:

La negación del producto es igual a la suma de los negados.

La negación de la suma es igual al producto de los negados.

Donde:

- El punto es el producto lógico (Y).
- La cruz es la suma lógica (O).
- La barra superior es la negación lógica (NO) de lo que está por debajo de la barra superior.

Doble negación

$$\overline{\overline{A}} = A$$

REPRESENTACIÓN DE NÚMEROS EN LOS SISTEMAS INFORMÁTICOS: NÚMEROS ENTEROS

Un bit es un dígito binario (base 2), que es la mínima cantidad de información representable. En los sistemas digitales se asigna un número fijo n de bits para representar un número, donde n es la longitud de una palabra (conjunto de n bits). Una palabra está compuesta por un número de bits que pueden ser tratados en su conjunto o simultáneamente en una operación del sistema digital, siendo sus longitudes más corrientes: 8, 16 ó 32 bits, según el sistema.

Con n bits se pueden representar 2^n combinaciones distintas y por lo tanto 2n números diferentes, por lo que existirán dos valores extremos, un máximo y un mínimo, que acotarán a todos los números representables.

Signo y magnitud

En este sistema de representación, se utiliza 1 bit para representar el signo, y el resto, en binario natural, para representar el número.

Por tanto utilizando este sistema, el rango de representación será:

Utilizando n bits => $-(2^{n-1}-1) \leq X \leq (2^{n-1}-1)$

Lo podemos ver mejor ilustrado con un ejemplo:

Si utilizamos 8 bits: n=8 bits, el rango de números que podemos representar utilizando este sistema será: $-(2^{8-1}-1) \leq X \leq (2^{8-1}-1)$, $-127 \leq X \leq 127$.

Puesto que 1 bit lo utilizamos para representar el signo, y los 7 restantes para representar los números.

Es decir, podemos representar los números enteros comprendidos entre el -127 y el 127.

A continuación vamos a codificar utilizando este sistema, los números 20 y -20: (utilizando 8 bits):

+20(10= **0**0010100

-20(10= **1**0010100

(El bit en negrita representa el signo del número: 0 para números positivos y 1 para números negativos)

Complemento a 1

Al igual que en el sistema anterior, en este sistema se utiliza el primer bit para codificar el signo del número (0 para números positivos / 1 para números negativos).

El número se representa en binario si es positivo, y en complemento a 1 si es negativo.

El complemento a 1, de un número binario se obtiene intercambiando los ceros por unos y los unos por ceros.

EJEMPLO: Representación de los números 20 y -20 en complemento a 1, utilizando 8 bits:

+20(10= **0**0010100(C1

-20(10= **1**1101011(C1

(El bit en negrita representa el signo del número)

Complemento a 2

Primer bit para signo, (0 positivo/1 negativo). El número positivo se representa en binario.

Si es negativo:

1. se complementa a 1 (incluso signo)
2. se le suma 1 en binario, (sin acarreo final)

EJEMPLO: Representación de los números 68 y -68 en complemento a 2, utilizando 8 bits:

$+68_{(10} = 01000100_{(C2}$

$-68_{(10}$=> 68: **0**1000100

C1: 10111011

+1: 1

10111100$_{(C2}$

(El bit en negrita representa el signo del número: 0 positivo, 1 negativo)