CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UNIDAD 1.
INFORMACIÓN DE REPRESENTACIÓN

Los sistemas informáticos
CFGS DAW

Alfredo Oltra

alfredo.oltra@ceedcv.e s

2019/2020

Versión: 190924.0933

Licencia

nomenclatura

A lo largo de Este tema se utilizarán Distintos Símbolos para distinguir Elementos Importantes Dentro del contenido. Símbolos Estós hijo:

- Importante

- Atención

- interesante

ÍNDICE DE CONTENIDO

1. Introducción..... ..... 4 ....
1.1 Pieza de información y la información 4
1.2 representación interna de datos 4
2. Sistemas numeral 5
2.1 Código binario 5
2.1.1 Cómo convertir un número decimal en un número binario 5
2.1.2 Cómo convertir un número binario en un número decimal 7
2.1.3 El número máximo de valores para representar 8
2.1.4 Operaciones con números binarios 8
2.1.5 Los números negativos 9
2.1.6 Los números reales 11
2.1.7 álgebra de Boole 14
2,2 Octal 15
2.2.1 Cómo convertir un número binario en octal 15
2.2.2 Cómo convertir un número octal en binario ....dieciséis
2.3 hexadecimal ....dieciséis
2.3.1 Cómo convertir números binarios en hexadecimal ....dieciséis
2.3.2 Cómo convertir un número hexadecimal en binario 17
2.3.3 Cómo convertir números hexadecimales en números octales 17
3. Representación alfanumérica 17
3.1 numéricos y alfanuméricos de datos 17
3.2 Representación interna 18
4. Sistema de unidades 19
5. El material adicional 20
6. Bibliografía 20

UD01. INFORMACIÓN DE REPRESENTACIÓN

1. INTRODUCCIÓN

1.1 Pieza de información y la información

Ordenadores (o más correctamente sistemas de información) son máquinas diseñadas para el procesamiento de información o, en otras palabras, para obtener los resultados de la aplicación de las operaciones en un conjunto de datos. Pero, ¿cuál es la información? ¿Qué es una pieza de información? Y lo que es una operación ?. Tomemos un ejemplo:

La temperatura es de 30º

- Pieza de información: la representación formal de un concepto, en este caso: " ***30*** "
- Información: el resultado de la interpretación de los datos: " ***Hace calor*** "
- Operación: regla aplica para obtener información: "Un ***s la temperatura es superior a 23, hace calor*** "

1.2 representación interna de datos

Por lo tanto, tenemos que almacenar y manejar en los datos y las operaciones de las computadoras. Y para ello, necesitan utilizar el código binario.

- Todo tipo de datos, ambos números o letras, se almacenan utilizando este sistema.

Este sistema se basa en el uso de sólo dos dígitos, 0 y 1, a diferencia del sistema decimal que utiliza diez (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Esto se debe a que los ordenadores sólo saben estos dos valores numéricos resultantes de la detección o no de un cierto potencial, de un número de voltios. Por lo tanto, un ordenador sabe que hay una 0 cuando el potencial medido en un miembro interior tiene un valor cercano a 0 voltios. De lo contrario, se detecta un 1.

- En términos eléctricos, el potencial podría asimilarse a la fuerza en el que la corriente eléctrica pasa a través de un cable.

- En general, los valores de 1 por lo general corresponden a potencial alrededor de 3 o 5 voltios.

Todos los elementos informáticos manejan esta numeración e interpretación de la información del sistema. Podría decirse que los ordenadores en realidad no saben nada en absoluto. Sólo saben acerca de 0 y 1 y de cómo realizar algunas operaciones básicas con ellos (+, -, * ...), aunque más rápido.

2. sistemas de numeración

Un sistema de numeración es un conjunto de **símbolos ordenados** se utiliza para representar cantidades. El número de símbolos se llama **base del sistema.**

En el mundo real, estamos acostumbrados a utilizar el sistema decimal (base 10), que conjunto de símbolos ordenados son 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Cualquier número, representado en cualquier sistema de numeración, se puede dividir en dígitos. Por ejemplo, 128 se puede dividir en 1, 2, 8 o 34,76 en 3, 4, 7, 6. A partir de estos dígitos y con su posición y la base del sistema es posible obtener de nuevo el número:

$$128 = 1 * 10_2 + 2 * 10_1 + 8 * 10_0$$

$$34{,}76 = 3 * 10_1 + 4 * 10_0 + 7 * 10_1 + 6 * 10\text{-}_2$$

Podemos ver que un número decimal se puede representar como adiciones de potencias de 10 (la base del sistema decimal). Si generalizamos, una serie **norte** expresado en un sistema de numeración ***si*** seria como:

$$N = a_{n-1}\ un_{n-2} \ldots un_1\ un_0, un_{-1}\ un_{-2} \ldots un_{-p+1}\ un_{-peg}$$

dónde:

N: número para representar

a: los símbolos que nuestro sistema de numeración incluye (enteros de 0 a B-1)

Los dígitos antes de la coma (,) [1] son la parte entera. Los dígitos después de la coma (,) son la parte fraccionaria.

2.1 B código de ciertas piezas

El código binario es un sistema de numeración que base del sistema es 2 y sus símbolos son 0 y 1.

- Cada dígito de un número binario se llama **poco** y es la unidad más pequeña de información, en otras palabras, es lo mínimo que se puede representar

- Para evitar confusiones, es habitual para indicar el número del sistema de base a ser representado por un subíndice a la derecha. Por ejemplo 101 ($_{10}$ o 101 ($_2$

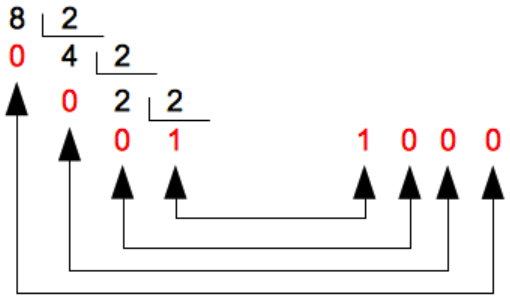
2.1.1 Cómo convertir un número decimal en un número binario

En general, para convertir un número decimal en otra base, tenemos que realizar divisiones sucesivas de la serie por la base. Al final, tenemos que obtener la

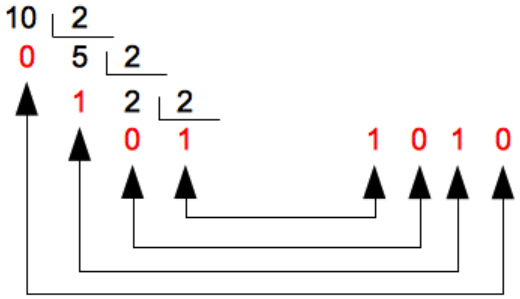
1 En la cultura Inglés, la separación entre la parte decimal y parte fraccionaria es un punto decimal (.)

desechos y el último cociente y los clasificó en la dirección opuesta. Consideremos el caso de convertir un decimal en binario con algunos ejemplos:

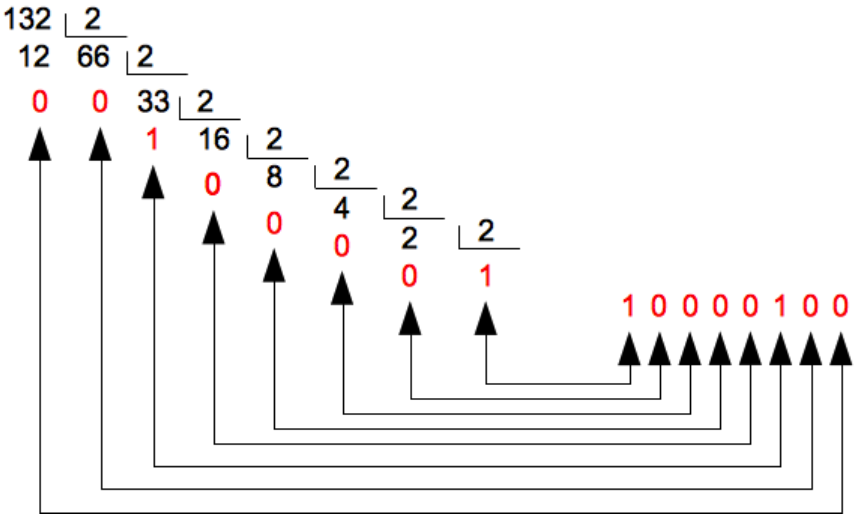
$8_{(10} \Rightarrow ?\ _{(2}$



$10_{(10} \Rightarrow ?\ _{(2}$

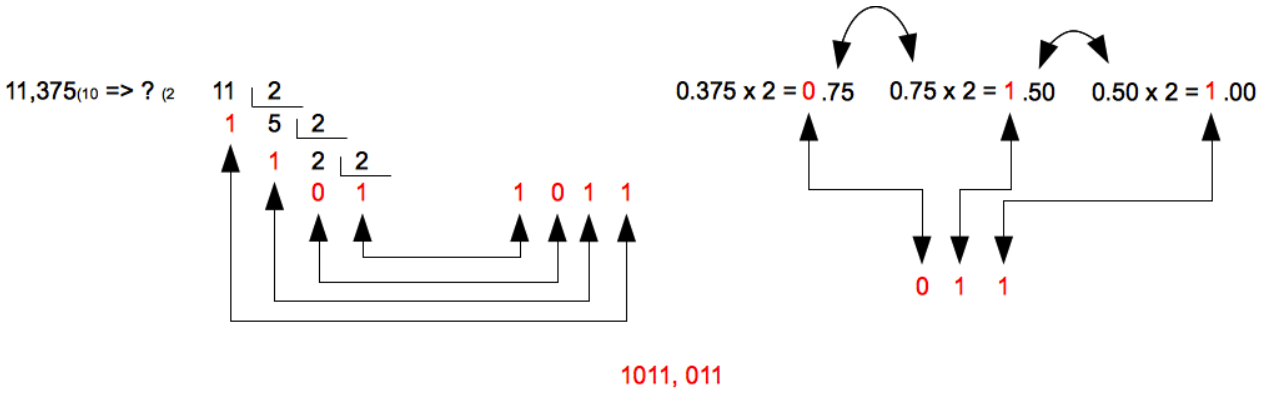


$132_{(10} \Rightarrow ?\ _{(2}$



En números con parte fraccionaria, el proceso es el mismo para la parte entera, pero la parte fraccionaria se calcula multiplicando por 2 sucesivamente y tomar la parte entera (en este caso en el orden correcto).

- El extremo izquierdo se mordió se le llama bit más significativo (MSB) y el extremo derecho se mordió se le llama bit menos significativo (LSB).



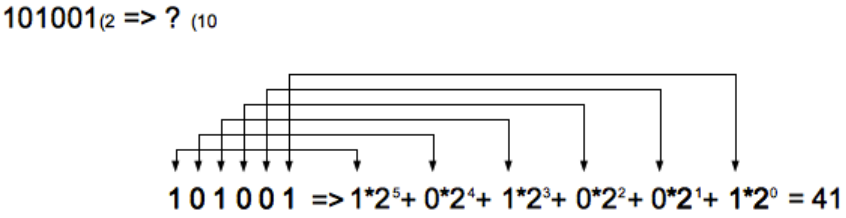
2.1.2 Cómo convertir un número binario en un número decimal

En este caso, el proceso es muy fácil. Como se explicó anteriormente, un número decimal se puede representar como adiciones de potencias de diez.

En general, se puede convertir el valor de un número representado en un sistema de numeración B [2] en el sistema decimal usando la siguiente fórmula:

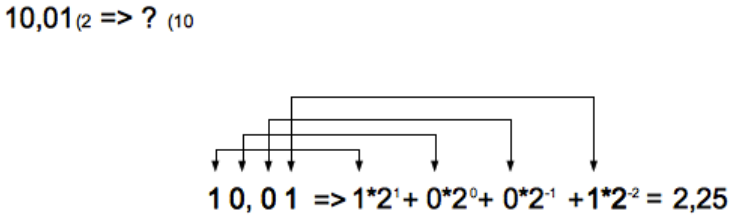
$$N = a_{n-1} si_{n-1} + un_{n-2} si_{n-2} + \dots + un_1 si_1 + un_0 si_0 + un_{-1} SI_{-1} + \dots + un_{-pag} SI_{-p} = \sum_{i=-p}^{n-1} un_{yo} si_{yo}$$

Vamos a utilizarlo para convertir en la base 2



El proceso consiste en cuatro pasos

1. Para escribir las cifras de números binarios multiplicado por 2.
2. Para escribir un signo más (+) entre cada uno de los productos.
3. Para escribir un exponente en cada 2, a partir de cero y desde el último número de la parte de número entero (en el extremo derecho si no hay parte fraccionaria) y el aumento de uno por uno a la izquierda y bajando a la derecha.
4. Para realizar la operación



2 Como veremos más adelante, B puede ser cualquier sistema de numeración

2.1.3 El número máximo de valores para representar

Una de las preguntas típicas al manipular un número binario es saber cuál es el valor decimal máximo que puede ser representado por un número determinado bits. La respuesta es fácil: 2_n, donde n es el número de bit. Por ejemplo, con 4 bits podemos representar 16 valores, de 0 a 15 (0.000 a 1.111)

2.1.4 Operaciones con números binarios

Además binaria y la resta siguen las siguientes reglas:

Adición: 0 + 0 = 0

1 + 0 = 1 0 + 1

= 1

1 + 1 = 0 (carry 1)

Sustracción:

De 0 - 0 = 0 1

- 0 = 1

A 0 - 1 = 1 (carry 1 a *sustraendo)*

1 - 1 = 0

El resultado de operaciones es el mismo que sus operaciones decimales relacionados, a excepción de los casos en que el resultado no tienen un valor en el sistema binario, es decir 1 + 1, que no pueden ser representados por 2 y 0-1, que puede no representan por -1. Aquí es donde arrastre es importante. Algunos ejemplos:

```
   11                  111111
  10011010                1011
+ 01001100            + 111101
  11100110              1001000
```

- Si queremos añadir dos números binarios, que además es mayor que el número máximo de representar el ordenador emite una ***Desbordamiento*** advertencia. Por ejemplo, si tenemos un equipo que trabaja con 8 bits, se puede representar de un 0 ($_{10}$ a 255 ($_{10}$. Si queremos añadir 10000000 ($_{2}$ (128 ($_{10}$)) plus 10000000 ($_{2}$ (128 ($_{10}$)) tenemos un problema porque el resultado es 100000000 ($_{2}$ (256 ($_{10}$) superior a 255. Por lo tanto una ***Desbordamiento*** ocurre.

```
  1 10000000
+ 10000000
  1 00000000
```

```
  101101              11101
    1                   11
- 10101             - 00111
  011000              10110
```

- En la resta, el traspaso no se suman al minuendo, pero sustraendo.

Multiplicación:

0 * 0 = 0 1 * 0

= 0 0 * 1 = 0 1

* 1 = 1

División: 0/0 = Indefinido

1/0 = sin límites 1/1 = 1

0/1 = 0

Ambos, multiplicación y división, los presentes no hay diferencia de las operaciones relacionadas en decimal, a menos que las operaciones auxiliares se realizan en binario.

- En la multiplicación, cuando añadimos, puede ser que tenemos en la misma columna más de dos de 1. En este caso, realizamos las adiciones en grupos de dos y vamos a llevar a las 1 de la siguiente columna.

- En la división, que comienza a recibir en el dividendo y el divisor el mismo número de cifras. Si no se puede dividir, intentamos conseguir una figura más en el dividendo.

Si la división es posible, entonces, el divisor sólo puede estar contenida una vez en el dividendo, es decir, la primera figura cociente es 1. En este caso, el resultado de multiplicar el divisor por 1 es el divisor en sí (el valor que restará).

```
101010      L110
- 110         111
  1001
    1
- 1 110
  00110
     110
     000
```

2.1.5 Los números negativos

Cuando necesitamos para representar un número binario negativo, tenemos varias opciones

aunque tres son los más importantes. Este rango indica que la manera de expresar que debería ser un acuerdo entre dos partes: la que genera el número y uno para leerlo. Si no es así, el valor real que se expresa sería un error.

magnitud con signo

Tal vez es el método más sencillo de entender. La idea es mantener el MSB para indicar el signo del número: 0 positivo, 1 negativo. Los bits restantes indican el valor del número en valor absoluto. Por ejemplo:

Decimal	Binario	binario positivo binario negativo	
5	101	***0*** 101	***1*** 101

Como se puede ver, necesitamos un bit para indicar el signo, de modo que lo que en valores normales de representación sería 0 a 15 en este caso, para utilizar el signo, se convierte de
-7-7 (1111 - 0111).

Este sistema es sencillo de entender pero difíciles de usar cuando se realizan operaciones matemáticas. Además tiene un problema: hay dos maneras de definir el 0 ($_{10}$: 0000 ($_2$ y 1000 ($_2$

complemento a uno

La segunda opción también utiliza el primer bit como indicador de señal, pero en este caso el número negativo se logra complementa número positivo (cambiando unos por ceros y viceversa).

Decimal	Binario	Positivo Negativo binaria	
5	0101	***0*** 101	***1010***

En esta opción se requiere para dar el número de bits para codificar, de tal manera que si en el ejemplo anterior usamos 8 bits para codificar:

Decimal	Binario	Positivo Negativo binaria	
5	101	***0*** 0000101	***1*** 1111010

Este método tiene el mismo problema que magnitud con signo: hay dos maneras de definir el 0 ($_{10}$: 0000 ($_2$ y 1111 ($_2$

complemento a dos

Aunque complemento a uno simplifica las operaciones matemáticas, que hacen mucho más con el uso de complemento a dos. Es por eso que es el método más utilizado.

complemento a dos consiste en aplicar un complemento queridos y luego, añadir 1. Por ejemplo, complemento a dos de 5 codificados con 8 bits es:

5 ($_{10}$ → 101 ($_2$ → (codificado en 8 bits) 00000101 ($_2$ → (de 1 complemento) 1111010 ($_2$ →
(+1) 11111011

Decimal	Binario	Positivo Negativo binaria	
5	101	***0*** 0000101	***1*** 1111011

¿Qué número decimal representa un número en complemento a dos ?. Fácil. Tenemos para preformas el mismo proceso:

11111011 ($_2$ → (de 1 complemento) → 00000100 ($_2$ → (+1) 00000101 ($_2$ → 5 ($_{10}$

La gran ventaja del método de complemento a dos es que permite la resta como si fueran añade. Esto se debe **restar dos números binarios es lo mismo que sumar al minuendo el complemento del sustraendo .. 101101 ($_2$ (45 ($_{10}$) - 010101 ($_2$ (21 ($_{10}$) • 010101 ($_2$ (del complemento a 1) → 101010 ($_2$ →**

→ (+1) 101011 • 101101 + 101011

```
  101101
+ 101011
```

1 011000 ($_2$ (24 ($_{10}$) el último traspaso 1 se rechaza

El exceso-K o binario compensar

Dependiendo del número de bits disponibles, de rango medio está dedicado para números negativos y la otra mitad (menos 1) a los aspectos positivos (el valor cero está en el medio). La nueva gama será [-K, K-1], donde **podemos calcular por K = 2^{n-1}** [3].
Una vez que tenemos el rango admisible, el número más pequeño es quien tiene todos sus bits a 0. Veamos un ejemplo:

Tenemos 3 bits para representar el número para que podamos representar 2^3 números, el intervalo [0,7]. En este caso, K será $2^{3-1} = 2^2 = 4$, por lo que la gama con números negativos será [-4,3]. El número más pequeño -4 será 000 y el más grande 3 estará 111. La junta será completa:

- 4	- 3	- 2	- 1	0	1	2	3
000	001	010	011	100	101	110	111

Si tenemos un número en *El exceso-K* y sabemos que su valor decimal, tenemos que restar el valor del exceso al valor decimal. Por ejemplo, si n = 8 y es $K = 2^{n-1} = 128$,

11001100 → 204; 204-128 = 76 o 00111100 → 60; 60-128 = -68

2.1.6 Los números reales

Cuando escribimos un número real en un papel que utilizamos una coma decimal (o punto decimal, que depende de la cultura) para distinguir entre parte entera y una parte fraccionaria. En un ordenador, el espacio para representar este tipo de números es

3 Hay otra versión de este método con $K = 2^{n-1} - 1$

dividida en dos áreas: una para la parte entera y una para la parte fraccionaria. Hay dos maneras para denotar el tamaño de estas áreas (campos), y por lo tanto, la posición de coma: ***punto fijo*** y ***punto flotante. Punto fijo***

En esta notación, asignamos un tamaño fijo a la parte entera y la parte fraccionaria del número, en otras palabras, un lugar fijo a la coma. La ventaja es que el proceso para realizar operaciones básicas es el mismo que los números enteros. Sin embargo, este método no se aprovecha de la capacidad del formato de representación utilizado. Por ejemplo, un ordenador con 8 bits para representar números, podría utilizar 5 bits para la parte entera y 3 para fraccional parte b_7 si $_6$ si $_5$ si $_4$ si $_3$, si $_2$ si $_1$ si $_0$.

En este caso el número máximo para representar será 01111.111 y el mínimo (positivo) 00000,001. Si la coma decimal estaría en una posición de flotación, el rango de números positivos representan ma sería 011111111 -

0,0000001

Punto flotante

El rango de números que pueden ser representados en el formato de punto fijo es insuficiente para muchas aplicaciones, particularmente para aplicaciones científicas que a menudo utilizan números muy grandes y muy pequeñas. Para representar una amplia gama de números utilizando relativamente pocos dígitos, es bien conocido en el sistema decimal, la representación científica o notación exponencial. Por ejemplo, 0,00000025 =

$2{,}5 * 10^{-7}$. en general, para cualquier sistema de numeración, un número real puede ser expresada como:

$$N = M * B^{mi} \text{ o } N = (M; B; E)$$

dónde:

M: mantisa B: E

Base: exponente

La representación interna que hace que este formato en los ordenadores que se conoce como punto flotante.

Por ejemplo, en decimal (B = 10) 259,75 ($_{10}$ = $0{,}25975 * 10^3$ o (0,25975; 10; 3) o, en código binario (B = 2)

259,75 ($_{10}$ → 100000011,11 ($_2$ → $0{,}10000001111 * 2^9$ ($_2$ → $0{,}10000001111 * 2^{1001}$ ($_2$
→ (0,10000001111; 1001)

Los números representables gama para un valor dado de B, se fija por el número de bits del exponente E, mientras que la exactitud se determina por el número de bits de M.

Normalización

El mismo valor real se puede representar de infinitas formas de notación exponencial. Por ejemplo, 2,5 se puede representar como $0{,}25 * 10^1$, $0025 * 10^2$, $250 * 10$

2, ... Para evitar confusiones, debemos elegir uno de estos formatos como el estándar para la representación de punto de un número real flotante. La forma elegida se llama ***forma normalizada*** y es uno que mantiene la más alta precisión en la representación de los números. Esto se logra cuando el punto binario se encuentra inmediatamente a la izquierda del primer dígito significativo, de modo que no se desperdicia espacio representa la ausencia de dígitos significativos. Por ejemplo:

- 2,5 representado en forma normalizada es de 0,25 * 10 1
- (0,000011101; 2; 0111) → (0,11101; 0011) 4 → exceso-k exponente (0,11101; 1011)

> • En general, para representar los exponentes negativos *exceso-k* se utiliza el método. En el otro lado, para representar negativo *mantisas* método de magnitud con signo

- (100,11110; 2; 0010) → (0,10011110; 2; 0101) → exceso-k exponente (0,10011110; 1101)
- (101.001; 2; 0100) → (0,1010010; 2; 0111) → exceso-k exponente (0,10011110; 1111)

IEEE754

El formato más popular para la representación de puntos flotantes en binario fue desarrollado por el ***Instituto de Ingenieros Eléctricos y Electrónicos (*** IEEE) y es llamado el IEEE754. Este formato puede representar casos especiales, tales como valores infinitos y los resultados no definidos, ***Yaya*** o ***No un número*** resultados. Propone 3 formatos:

La mitad de precisión. Se utiliza 16 bits

Se utiliza 16 bits: un bit para el signo, 5 para el exponente, 10 para la precisión mantisa simple.

Se utiliza 32 bits: un bit para el signo, 8 para el exponente, 23 para la mantisa de precisión doble.

Se utiliza 64 bits: un bit para el signo, 11 para el exponente, 55 para la mantisa

> • Los tres formatos utilizan mantisa normalizada, por lo que el primer bit de la izquierda en la mantisa será un 1 (el primer dígito significativo

4 Eliminamos el valor base, porque se supone que la base del sistema es 2

tiene que ser un 1). Debido a esto, tres formatos no codifican 1 en la mantisa, a pesar de que se tiene en cuenta cuando se opera con el número. En otras palabras, en estos formatos el MSB está a la izquierda de la coma decimal y que sólo guarda los bits en el lado derecho. Para representar el exponente los usos estándar, el método de exceso-K con $K = 2^{n-1} - 1$

- Para entender mejor esta operación es muy recomendable reloj la píldora 02 ***Convertir número real de URL código binario***

2.1.7 álgebra de Boole

Además de las operaciones matemáticas (+, -, * /), en números binarios puede aplicar operaciones booleanas o lógicos: AND, OR, XOR, no ...

- Para comprender mejor estas operaciones es digno de nombrar al menos 1 ***cierto*** y 0 como ***falso***

NO:

Se puede representar de varias maneras: NO, ¬

NO 0 = 1 NO 1 =

0

Y:

Se puede representar de varias maneras: Y, Y, ^, *

0 y 0 = 0 1 0 Y = 0

0 y 1 = 0 1 Y 1 = 1

En otras palabras, el resultado será verdadero (1) sólo cuando los dos dígitos eran ***cierto.*** Como se puede ver, el resultado es el mismo que el de multiplicar.

```
  10011010          1011
Y 01001100      Y 111101
  00001000          001001
```

O:

Puede representarse de varias maneras: O, O, v

0 O 0 = 0 1 O 0

= 1 0 OR 1 = 1 1

O 1 = 1

En este caso, el resultado será ***cierto*** tan pronto como uno de los dígitos era ***cierto.***

```
  10011010                1011
O 01001100            O 111101
  11011110                111111
```

XOR:

0 XOR 0 = 0 1

XOR 0 = 1 0 XOR

1 = 1 1 XOR 1 = 0

En este caso, el resultado será ***cierto*** cuando uno y sólo uno de los dígitos eran ***cierto.***

```
  10011010                1011
XOR 01001100          XOR 111101
  11010110                110110
```

2.2 octal

Junto a binario, hay otros dos sistemas de numeración interesante cuando trabajan en temas relacionados con la tecnología de la información: el octal y hexadecimal. Esto se debe a que de ellos son fáciles de convertir a binario.

El octal es un sistema de numeración con una base de sistema de igual a 8 (símbolos 0,1, 2, 3, 4, 5, 6, 7). Su base es una potencia exacta de la base del sistema binario de $2_3 = 8$ o, en otras palabras, con tres dígitos binarios (con tres bits) podemos representar todos los dígitos octales.

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

2.2.1 Cómo convertir un número binario en octal

El proceso consiste en crear grupos de bits de tres, a partir de la mano derecha,

y sustituirlos por el valor octal relacionados

1101011 (2 => 1 101 011 => 1 5 3 (8

2.2.2 Cómo convertir un número octal en binario

El proceso se invierte a la anterior: se convierte a binario cada uno de los números de número octal

7 4 0 2 (8 => 111 100 000 010 (2 => 111 100 000 010 (2

2.3 hexadecimal

Su base de sistema es de 16. A medida que el número de símbolos utilizados en el sistema es mayor que 10, se deben utilizar 6 caracteres, en este caso de la A a F. Por lo tanto, el conjunto ordenado de los símbolos es: 0, 1, 2, 3 , 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Decimal	Binario	hexadecimal
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	8
9	1001	9
10	1010	UN
11	1011	si
12	1100	C
13	1101	re
14	1110	mi
15	1111	F

2.3.1 Cómo convertir números binarios en hexadecimal

El proceso es similar al proceso binario-octal, excepto en este caso grupos son de cuatro en cuatro.

1101011 (2 => 110 1011 => 6 si (dieciséis

2.3.2 Cómo convertir un número hexadecimal en binario

El proceso se invierte a la anterior: se convierte a binario cada uno de los números de número octal

7 F 0 UN (16 = 0111 1111 0000 1010 (2 = 111 1111 0000 1010 (2

2.3.3 Cómo convertir números hexadecimales en números octales

Nos convertimos en una representación binaria y agrupar los bits en grupos de tres a cuatro patas o, lo que es el destino sistema de numeración

6 si (16 = 110 1011 => 1101011 (2 = 1 101 011 => 1 5 3 (8

> • La conversión entre octal o hexadecimal y decimal o viceversa, se puede realizar siguiendo los métodos para convertir entre binario y decimal, pero multiplicando por poder de 8 o 16 o dividiendo por estos números y para obtener los residuos (en hexadecimal si el resto es mayor que 9 obtenemos los valores relativos a A..F).
>
> Sin embargo, por lo general es más práctico para llevar a cabo directamente la conversión en binario y después, convertido en el sistema solicita.

3. representación alfanumérica

3.1 numéricos y alfanuméricos de datos.

Un dato es numérico, si es posible realizar operaciones matemáticas. Por el contrario, un dato es alfanumérico Si no puede realizar operaciones matemáticas en él.

numérico: ***¿Cuantos años tienes?*** **45**
alfanumérico: ***¿Cuál es tu nombre?*** **" Roberto "**

> • Con el fin de diferenciar claramente entre los dos tipos de datos, es común el uso de comillas simples o dobles para indicar que los datos son alfanumérico.

Es habitual pensar que los datos numéricos son números y los datos alfanuméricos son sólo letras. Pero esto no es correcto. Por ejemplo:

¿Cual es tu dirección? **" Avenida de las Palmeras 34 "**
¿Cuál es su número de teléfono móvil? **" 555341273"**

En el primer caso, **Avenida de las Palmeras 34,** está compuesto de letras y números, y en el segundo, **555341273,** solamente por los números, pero no operables (que no tiene sentido para agregar o multiplicar dos números de teléfono).

3.2 Representación interna

Los caracteres alfanuméricos para representar los ordenadores se basan en tablas, de manera que cada una de las entradas de la tabla (cada número) corresponde a un símbolo alfanumérico. A lo largo de la historia de la informática, se han realizado varias tablas que siempre se han caracterizado por el número de bits utilizados para representar cada carácter. Uno de los mejores ejemplos de la tabla ASCII. El número de bits es 7, lo que dejó espacio para 128 caracteres ($2_7 = 128$)

Como se puede observar en la siguiente tabla, cada número está relacionado con un personaje. Por ejemplo, 73 ($_{10}$ es una "I", 105 ($_{10}$ es una "i" o 50 ($_{10}$ es un "2". Las primeras entradas están reservadas para los caracteres no imprimibles, los que no son visibles, como el tabulador (9 ($_{10}$) o retorno de carro (15($_{10}$).

- El espacio también es un personaje: 32 ($_{10}$

El problema de esta tabla es su espacio limitado. Como se puede ver, tiene espacio para todos los vocablos latinos utilizados en lenguas anglosajonas, pero no podemos encontrar la ortografía como la ñ, ç o vocales acentuadas. Por lo tanto, se creó la tabla ASCII extendido de 8 bits (256 caracteres). Esta nueva tabla puede incorporar todos los vocablos latinos más algunos símbolos gráficos.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

- Hoy en día las tablas ASCII son casi obsoletos. La expansión de Internet y la globalización hacen tablas necesarias que incorporan no sólo los caracteres latinos, pero chino, árabe, coreano,

Ruso, hebreo ...

4. UNIDAD DE SISTEMA

Como se discutió anteriormente, el bit es la unidad de información más pequeña. Hoy en día no se trabaja a nivel de bit, pero en grupos de bits (en el apartado anterior hemos visto que un carácter se codifica utilizando 7 u 8 bits).

Debido a que todo el interior del equipo se encuentra en código binario, una manera fácil de manejar grupos es utilizar algún valor que es una potencia de 2, siendo el más básico 2^3 = 8. Un grupo de 8 bits se denomina **byte.**

Hoy en día no es usual utilizar el grupo de bytes, pero un múltiplo de la misma: ***kilobytes (*** kB), ***megabyte (*** MEGABYTE), ***Gigabyte (*** GB), ***terabyte (*** TB) .... En el ***Sistema Internacional*** estos múltiplos son potencias de 10 (que se basan en el sistema decimal), pero en potencias de 2 de computación se utilizan. Sin embargo, la tendencia es utilizar el Sistema Internacional, aunque cabe señalar que los valores son similares pero no iguales.

Se utilizan dos tipos diferentes de palabras para diferenciar entre las unidades del sistema. Cuando hablamos de kilobytes nos referimos al sistema decimal y cuando hablamos de ***kibibytes*** al sistema binario.

Las equivalencias se pueden ver en la siguiente tabla:

Sin nombre	SI	Binario	Nombre binario
Kilobytes (KB)	10^3 bytes = 1000 bytes	2^{10} bytes = 1024 bytes	Kibibyte (Kib)
Megabytes (MB)	10^6 bytes = 1000 kB	2^{20} bytes = 1024^2 bytes	Mebibyte (MIB)
Gigabyte (GB)	10^9 bytes = 1000 MB	2^{30} bytes = 1024^3 bytes	Gibibyte (GIB)
Terabyte (TB)	10^{12} bytes = 1000 GB	2^{40} bytes = 1024^4 bytes	Tebibyte (TiB)
Petabye (PB)	10^{15} bytes = 1000 TB	2^{50} bytes = 1024^5 bytes	Pebibyte (PIB)
Exabyte (EB)	10^{18} bytes = 1000 PB	2^{60} bytes = 1024^6 bytes	Exbibyte (BEI)
Zetabyte (ZB)	10^{21} bytes = 1000 EB	2^{70} bytes = 1024^7 bytes	Zebibyte (ZiB)

- Aunque por lo general se utiliza indistintamente y, aún hoy en día es más frecuente ***Sistema Internacional,*** los valores que están representados son diferentes: 1 MB son 1.000.000 bytes (un millón de bytes), mientras que 1 MiB son 1.048.576 bytes

- Es importante diferenciar entre kb y KB. El primero se refiere a kilobyte, mientras que el segundo kilobit, 8 veces menos.

- Aunque la mayoría de los nombres y abreviaturas de los múltiplos tienen letras mayúsculas, el kilo se define con una minúscula.

5. MATERIAL ADICIONAL

[1] Glosario.

[2] Videos acerca de las operaciones binarias. [3]

ejercicios.

6. BIBLIOGRAFÍA

[1] Wikipedia. Representaciones número con signo

https://en.wikipedia.org/wiki/Signed_number_representations [2] Wikipedia.

Representaciones número con signo

https://en.wikipedia.org/wiki/Signed_number_representations [3] Informáticos Sistemas. Isabel

Mª Jimenez Cumbreras. ***Garceta.*** 2012