Coc 3

**Conjunto universal**. También denominado conjunto unidad es el que incluye la totalidad de los elementos con una propiedad en común.

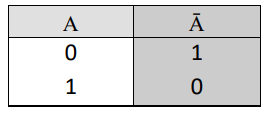
**Conjunto particular**. Reunión de elementos pertenecientes al conjunto universal, pero que además poseen alguna característica particular que los distingue del resto.

**Conjunto vacío**. Aquel que no posee ningún elemento. Se representa por 0.

**Conjunto complementario** de otro conjunto A (también denominado conjunto negado o inverso).

Está constituido por todos los elementos del conjunto universal que no pertenecen al conjunto A.

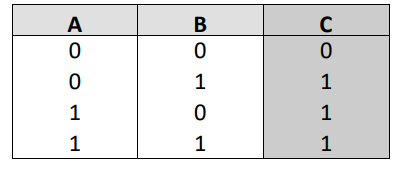
# **La complementación lógica**



# La suma lógica

se representa mediante el signo "+"

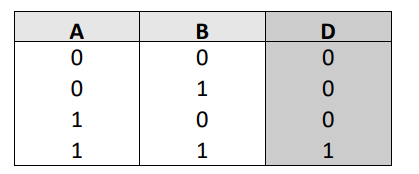
funciona como | en EPA



# **El producto lógico**

se representa mediante el símbolo \*

funciona como & en EPA



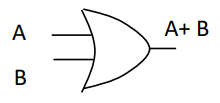
. Ley distributiva.

Respecto de la suma: A+BC = (A+B)(A+C)

Respecto del producto: A(B+C) = AB+AC

# **PUERTAS LÓGICAS**

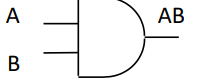
# Las puertas OR



Desarrollan la suma booleana

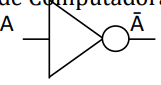
# Las puertas AND

Corresponden al producto booleano de las variables de entrada



A \* b

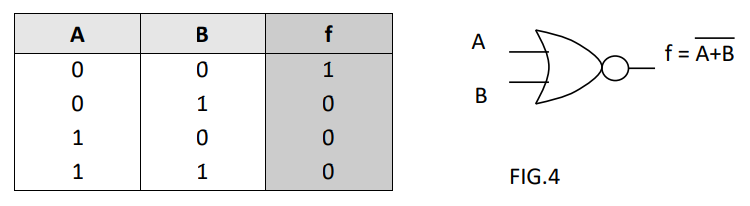
# Las puertas NOT



# Las puertas NOR

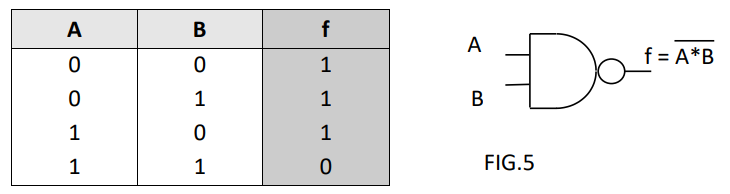
Realizan la función inversa de una operación suma lógica, es decir, es la equivalente a una puerta OR complementada. La función lógica será por tanto:





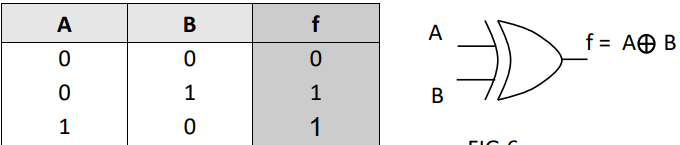
# Las puertas NAND

La función f es equivalente a una puerta AND complementada



# Las puertas OR EXCLUSIVAS (xor)

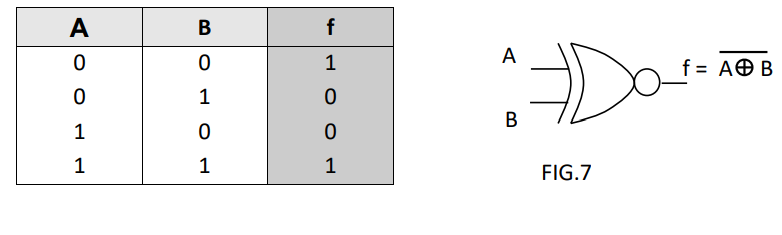
la salida de esta puerta es 1 lógico siempre que ‘una y solo una’ de sus entradas tenga el nivel lógico 1, es decir sus entradas tienen que poseer valores distintos.





# Las puertas NOR EXCLUSIVAS (XNOR)

Contrario a las or exclusivas, va a ser 1 siempre que sus entradas sean iguales





# **CIRCUITOS COMBINACIONALES**

Un circuito combinacional es un conjunto de puertas lógicas interconectadas, cuya salida, en un momento dado, es función solamente de los valores de las entradas en ese instante

# Preguntas

1. La razón técnica por la cual las computadoras son sistemas binarios es que solo hay dos posibilidades, que haya (1, v), o que no haya (0, f) corriente eléctrica.
2. La ventaja que daría esto es que los chips serian multiuso, y modificables, es decir no servirían para realizar una única función.

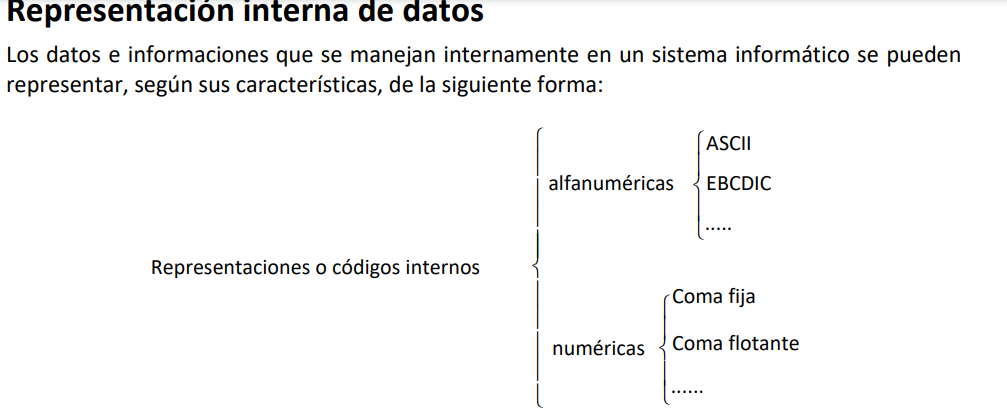
Si, existe y se llama pld, dispositivo lógico programable.

1. Yo creo que es porque cada letra del comienzo se ramifica en 8 cada una, 4 positivas y 4 negativas respectivamente

2)

D = AꚛB

C= A\*B



**Códigos Alfanuméricos**

Una computadora puede trabajar internamente con un conjunto de caracteres que nos

permitirán manejar datos, informaciones, instrucciones, órdenes de control, etc. Este conjunto

de caracteres podemos subdividirlo en los siguientes grupos:

 caracteres alfabéticos

 letras mayúsculas (A..Z sin la Ñ)

 letras minúsculas (a..z sin la ñ)

 cifras decimales: los números 0, 1, ..., 9

 caracteres especiales

 caracteres como el . , ; : \* @, etc.

 órdenes de control. Equivalen a las teclas enter, tabulación, esc, etc.

En general cada carácter se maneja internamente en una computadora por medio de un

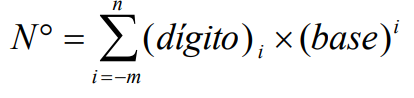
conjunto de 8 bits mediante un sistema de codificación binario que denominaremos código de

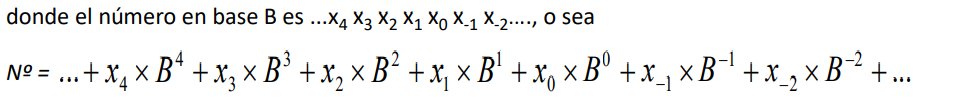
caracteres.

**Teorema Fundamental de la Numeración**

Se trata de un teorema que relaciona una cantidad expresada en cualquier sistema de numeración posicional con la misma cantidad expresada en el sistema decimal. Supongamos una cantidad expresada en un sistema cuya base es B y representamos por xi cada uno de los dígitos que contiene dicha cantidad, donde el subíndice i indica la posición del dígito con respecto a la coma fraccionaria, la posición se numera en forma creciente hacia la izquierda y decreciente hacia la derecha de la coma (posición 0), en ambos casos de a 1.

El Teorema Fundamental de la Numeración dice que el valor decimal de una cantidad expresada en otro sistema de numeración está dado por la fórmula:





**Sistemas Decimal, Binario y Hexadecimal**

El sistema que ha usado el hombre para contar desde hace bastante tiempo es el denominado sistema decimal, adoptado por contar con los diez dedos de la mano. El sistema decimal es uno de los denominados posicionales, que utiliza un conjunto de 10 símbolos, xi Є {0,...9}. Un valor determinado o cantidad, que se denomina número decimal, se puede expresar por la fórmula del Teorema anterior, donde la Base es 10. Ejemplo. ¿Cuál es la interpretación de la representación de la cantidad 3,1416?



La suma de números binarios es igual que la de decimales

En la resta es igual, cada vez que se pide al compañero el numero se multiplica x2 la cantidad de veces que se pase.

**Rango de representación. Valores mínimo y máximo.**

Se denomina rango de representación en un sistema determinado al conjunto de números representables con el mismo. Un sistema de base b y números de n dígitos tiene un rango igual a b n .

El valor mínimo representable se obtiene cuando los n dígitos del número son iguales al símbolo de menor valor del sistema, por ejemplo con 4 dígitos, 0000 coincide como mínimo en base 2, 10 o 16.

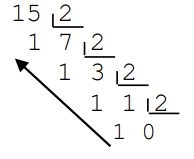
El valor máximo representable se obtiene cuando los n dígitos del número son iguales al símbolo de mayor valor del sistema, por ejemplo con 4 dígitos, los máximos serán 11112, 999910 o FFFF16.

**Conversión decimal-binario**

El método de conversión de un número decimal a un número binario consiste en efectuar, sobre la parte entera del número decimal, divisiones sucesivas de los cocientes por el número 2, hasta que el cociente tome el valor 0. La unión de todos los restos obtenidos, escritos en orden inverso, nos proporciona ahora el número decimal inicial expresado en sistema binario.

Ejemplo.

Convertir el número decimal 15 a binario.

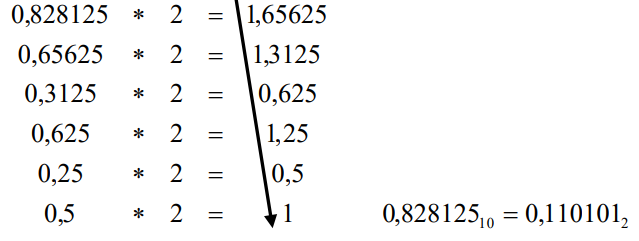


Leyendo los restos, del último obtenido al primero de ellos, tenemos: 1111(2 = 15(10

Para convertir una fracción decimal a su equivalente binario se debe multiplicar dicha fracción por dos, obteniendo en la parte entera del resultado el primero de los dígitos binarios de la fracción que buscamos. A continuación, se repite el proceso con la parte fraccionaria del resultado anterior, obteniendo en la parte entera del nuevo resultado el segundo de los dígitos buscados. El proceso se repite hasta que desaparezca la parte fraccionaria de los resultados parciales (se haga 0) o hasta que tengamos los suficientes dígitos binarios.

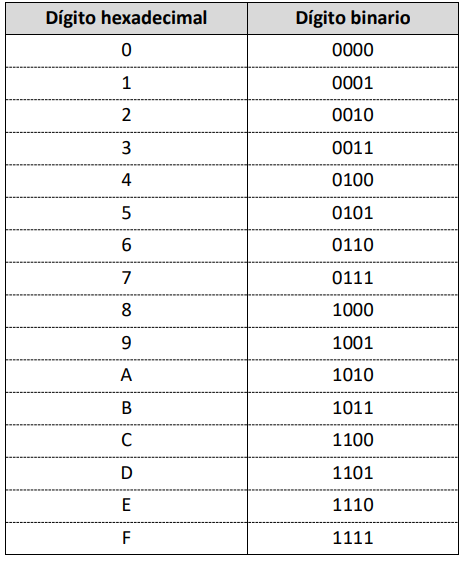
Ejemplo.

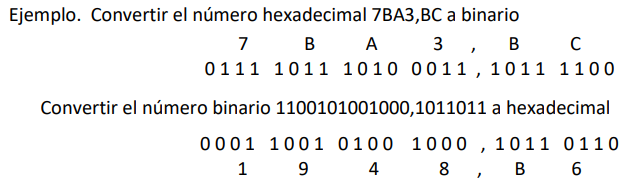
Se desea convertir la fracción 0,828125 a binario



**Conversión hexadecimal-binario y binario-hexadecimal**

Cada dígito hexadecimal tiene una representación binaria con cuatro dígitos según indica la siguiente Tabla





**Representación de números enteros**

Las computadoras utilizan cuatro métodos para la representación interna de números enteros (positivos y negativos); éstos son los siguientes:

▪ Módulo y signo

▪ Complemento a 1

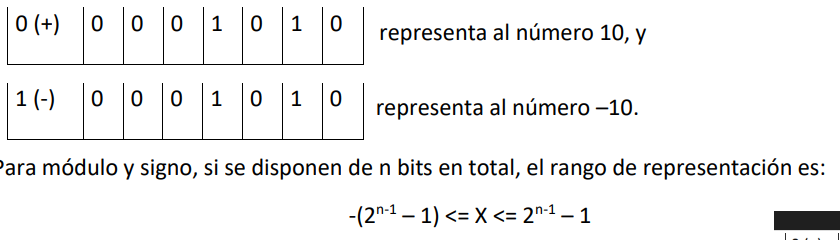
▪ Complemento a 2

▪ Exceso a 2n-1

Estas representaciones de números utilizan el sistema binario y se considera que tenemos un número limitado de bits para cada dato numérico. Este número de bits disponibles lo representamos por n. También se pueden representar mediante estos métodos números reales, como veremos más adelante.

**Módulo y signo**

En este sistema de representación, también llamado binario con signo, el bit que está situado más a la izquierda representa el signo, y su valor será 0 para el signo + y 1 para el signo -. El resto de bits (n-1) representan el módulo del número. Suponemos en principio que los números no poseen parte decimal, por lo que la coma se supone implícita a la derecha. Por ejemplo, supongamos que disponemos de 8 bits, y queremos representar los números 10 y –10. Veamos cuales son sus representaciones.



Para el caso de n = 8 bits, el rango de representación va desde –127 a 127.

La ventaja que presenta este sistema frente a otros es la de poseer rango simétrico (igual cantidad de números positivos que negativos), mientras que su mayor inconveniente es el de poseer dos representaciones para el número 0. El cual se representa tanto con un signo positivo (0) como con uno negativo (1) y el resto de los bits en 0

**Complemento a 1**

Este sistema de representación utiliza el bit de más a la izquierda para el signo, correspondiendo

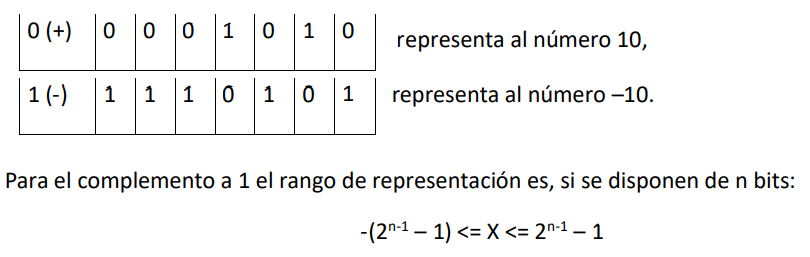
el 0 para el signo + y el 1 para el signo -. Para los números positivos, los n-1 bits de la derecha

representan el módulo (igual que en el sistema anterior). El negativo de un número se obtiene

complementando todos sus dígitos (cambiando ceros por uno y viceversa) incluido el signo.

Veamos la representación en complemento a 1 de los números 10 y –10 para el caso de n=8

bits.



Para el caso de n = 8 bits, el rango de representación va desde –127 a 127.

**Complemento a 2**

Este sistema de representación utiliza el bit de más a la izquierda para el signo, correspondiendo

el 0 para el signo + y el 1 para el signo -. Para los números positivos, los n -1 bits de la derecha

representan el módulo (igual que en los dos sistemas anteriores). El negativo de un número se

obtiene en dos pasos:

▪ Primer paso: se complementa el número en todos sus bits (cambiando ceros por uno

y viceversa), incluido el bit de signo, similar a complemento a 1.

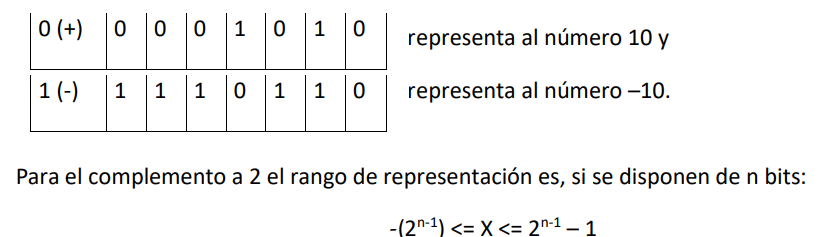
▪ Segundo paso: al resultado obtenido en el paso anterior se le suma 1 (en binario),

despreciando el acarreo del bit más significativo si existiera.

Veamos la representación en complemento a 2 de los números 10 y –10 para el caso de n = 8

bits.

# Es decir igual que el a 1 pero le sumamos 1 a lo que nos de la transformación al negativo

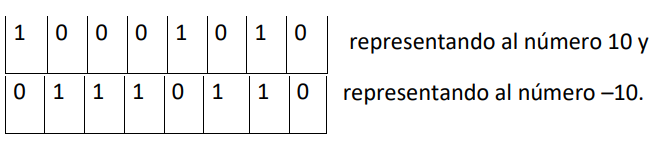


Para el caso de n = 8 bits, el rango de representación va desde –128 a 127

La principal ventaja es la de tener una única representación para el número 0, ya que el 0 positivo o negativo se representan igual.

**Exceso a 2n-1**

Por ejemplo, para n=8 bits el valor del exceso a utilizar será 128, con lo cual para representar un número deberá sumársele dicho exceso. De esta manera el número decimal 10, que veníamos representando, recibirá la adición del número 128, por lo que el número 138 expresado en binario lo representará. Por otro lado, el número decimal –10, se representará como el 118 (─10+128) en binario. De esta forma quedarán:

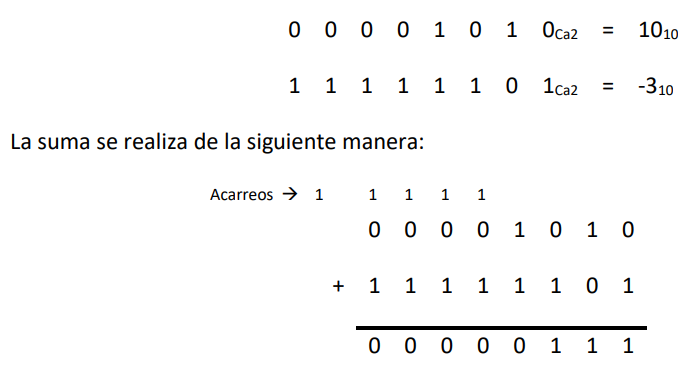


En este sistema el número 0 tiene una sola representación, la cual consiste en la representación binaria del valor del exceso que en este ejemplo es 100000002. El rango de representación en exceso a 2 n-1 es asimétrico y viene dado por:



Resulta interesante observar que todo número representado en exceso a 2n-1 tiene la misma representación que un complemento a 2 con el bit que define el signo cambiado de valor. Puede inferirse entonces, que el bit mas significativo representaría el signo de valor opuesto (el 0 un valor ‘-‘ y el 1 un valor ‘+’).

**Suma en complemento a 2**



El acarreo del bit mas alto se desprecia, por lo que ese 1 señalado no se toma en cuenta

y teniendo en cuenta que el acarreo del bit más significativa se desprecia, el resultado de 8 bits obtenido 000001112 interpretado como valor en Ca2 representa el valor +710 que es correcto.

**Flags**

En la UCP, existen bits llamados banderas o flags que luego de realizar una operación, cambiarán acorde al resultado de dicha operación.

Se presentan aquí 4 banderas, que son determinadas luego de realizar una operación de suma ó una resta y por eso las llamaremos banderas aritméticas:

▪ Z (cero): esta bandera toma el valor 1 indicando que el resultado de la operación fue cero. Para cualquier otro resultado el valor de esta bandera es cero.

▪ N (negativo): esta bandera toma el valor del bit más significativo del resultado. Dicho de otra manera si la bandera vale 1 es porque el resultado es negativo, y 0 si el resultado es positivo.

▪ C (carry): esta bandera toma el valor 1 indicando que hay ’acarreo’ en la suma ó ‘borrow’ en la resta. Cuando esta bandera toma el valor 1 indica una condición de fuera de rango en números sin signo.

▪ V (overflow): esta bandera vale 1, indicando una condición de desborde (fuera de rango) del resultado en números con signo (complemento a 2). Por condición de desborde se entiende que la cantidad de bits no alcanza para expresar el resultado.