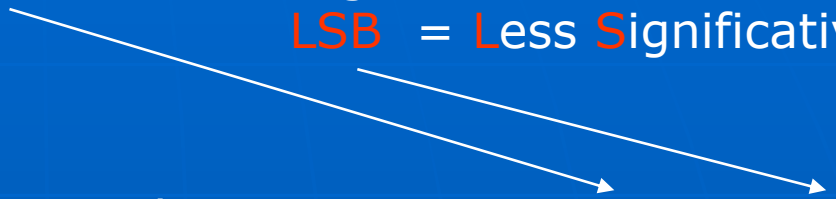
Sistemas Numéricos

SISTEMAS NUMÉRICOS

MSB = Most Significative Bit
LSB = Less Significative Bit



Clasificación según la base:

Binario:	La base es 2.	Ej.:	1001110
Decimal:	La base es 10.	Ej.:	78
Hexadecimal:	La base es 16.	Ej.:	4E

El mundo digital adopta el sistema de representación binario.
El mundo humano adopta el sistema de representación decimal.

Es necesario, entonces, poder realizar conversiones entre un sistema y el otro.

El hexadecimal está intimamente ligado al binario por ser ambas bases potencia de 2 (binario es 2^1 y hexadecimal es 2^4).

SISTEMAS NUMÉRICOS

Equivalencias

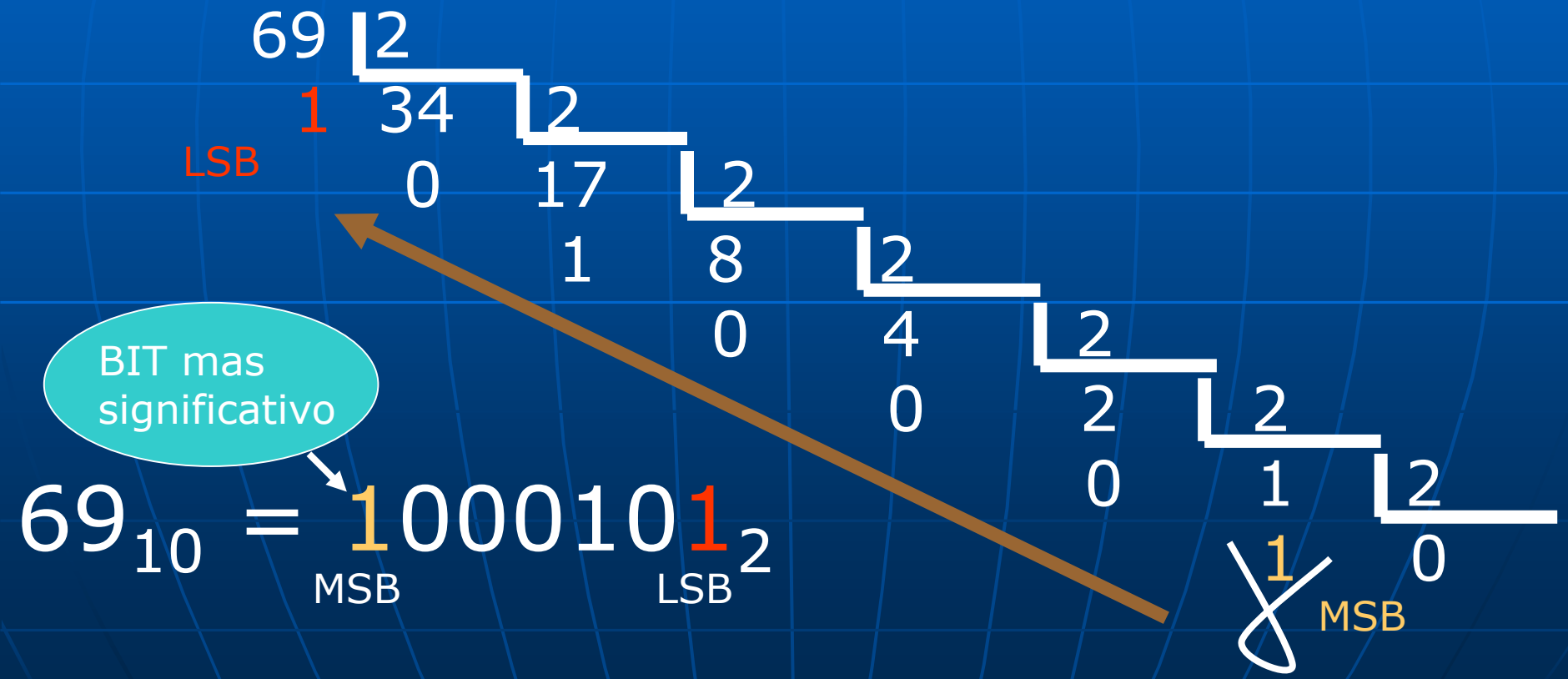
Sistema decimal	Sistema Binario	Sistema Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Números enteros:

Pasaje de base decimal a binario:

Se debe ir dividiendo por 2 en forma consecutiva los resultados obtenidos hasta que el resultado sea 0.

EJEMPLO:



Números enteros:

Pasaje de base binario a decimal :

Se debe desarrollar la fórmula que expresa un número decimal En función de los coeficientes binarios.

$$N_{10} = a_{n-1} \bullet 2^{n-1} + a_{n-2} \bullet 2^{n-2} + \ldots + a_3 \bullet 2^3 + a_2 \bullet 2^2 + a_1 \bullet 2^1 + a_0 \bullet 2^0$$

Como se denota los dígitos binarios desde 0 en adelante, un número binario de n bits tendrá designaciones desde 0 hasta (n-1), dando un total de n coeficientes.

EJEMPLO:

$$10110_2 => 1 \bullet 2^4 + 0 \bullet 2^3 + 1 \bullet 2^2 + 1 \bullet 2^1 + 0 \bullet 2^0$$
$$= 16 + 0 + 4 + 2 + 0$$
$$= 22_{10}$$

Números fraccionarios (no enteros):

Pasaje de base decimal a binario:

Se debe ir multiplicando por 2 la parte decimal. Los resultados obtenidos en cada multiplicación corresponden a los coeficientes binarios desde el a_{-1} hasta el a_{-n}.

$$N_{10} = a_{-1} \bullet 2^{-1} + a_{-2} \bullet 2^{-2} + \ldots + a_{-n} \bullet 2^{-n}$$

EJEMPLO:

0,125 => 0,125 x 2 = 0,250 ➔ $a_{-1} = 0$
0,250 x 2 = 0,500 ➔ $a_{-2} = 0$
0,500 x 2 = 1,000 ➔ $a_{-3} = 1$
0,000 x 2 = 0 (se termina la conversión)

$0{,}125_{10} = 0{,}001_2$

Números reales (con parte entera y decimal):

Pasaje de base decimal a binario:

Se deben realizar por separado las operaciones anteriormente descriptas.

EJEMPLO:

$22,125_{10}$ ➔ $22_{10} = 10110$

$0,125_{10} = 0,001$

$= 22_{10} + 0,125_{10} = 10110 + 0,001 = 10110,001_2$

Números reales (con parte entera y decimal):

Pasaje de base hexadecimal a binario:

Cada dígito hexadecimal está representado en binario por 4 bits.

EJEMPLO: $A3FD,CB2_{16} = 1010001111111101,110010110010_2$

Pasaje de base binario a hexadecimal :

Para la parte entera del número binario, se deben tomar grupos de a 4 bits desde la coma hacia la izquierda y completar con "ceros" si es necesario para obtener el dígito hexadecimal mas significativo.

Para la parte decimal, se debeyomar grupos de a 4 bits desde la Coma hacia la derecha y completar si es necesario con "ceros" para obtener el último dígito hexadecimal.

EJEMPLO: $101000101010111000110,101010101111111_2 =$

Representación de números en punto fijo

Existe un campo para la representación de la parte entera del Número binario y otra para la parte decimal.

EJEMPLO: $0111{,}001_2$ que equivale al $7{,}125_{10}$.

En general cada uno de ellos se ubica en posiciones de memoria diferentes.
Dependiendo de la extensión del número se pueden alojar en un "nibble" (4 bits), un "byte" (8 bits), un "word" (16 bits), "doble word" (32 bits), etc.
Mayor cantidad de bits empleados permiten representar un número en el sistema decimal mayor.

Representación de números en punto flotante

Existen en general al menos 3 campos para la representación de:

+ Mantisa.
+ Signo del número.
+ Exponente con el signo incluído.

Su utilización es importante cuando se requiere representar cantidades muy pequeñas y/o muy grandes, siendo muy engorroso emplear el formato en punto fijo por la cantidad de dígitos que se deberían emplear (y por lo tanto posiciones de memoria).

Uno de los formatos mas conocidos es el IEEE P754.

EJEMPLO: En IEEE P954 formato precisión simple, el número:
1|11000000|11000000000000000000000
equivale al número $-3{,}5_{10}$.
Nota: El símbolo "|" denota separación de campos.

Representación de números en punto fijo

Existen dos variantes:

+ Números sin signo.
+ Números con signo.

Números sin signo: Todos los bits se utilizan para representar el módulo del número.

Números con signo: Dado N bits, 1 de ellos se usa para representar el signo y los N-1 restantes para representar el módulo del número.

Existen tres métodos de representación:

+ Representación “Signo y módulo”.
+ Representación en “Complemento a 1 (Ca1)”.
+ Representación en “Complemento a 2 (Ca2)”.

Representación de números en punto fijo sin signo

Dado N bits para su representación tenemos:

Rango: 2^N

EJEMPLO: Con 8 bits ➔ Rango = 256.
Con 16 bits ➔ Rango = 65536.

Número mínimo a representar: 0.
Número máximo a representar: 2^N-1.

EJEMPLO: Con 8 bits ➔ Nº mínimo = 0.
Nº máx. = 255.

Con 16 bits ➔ Nº mínimo = 0.
Nº máx. = 65.535.

Representación de números en punto fijo con signo

Signo y módulo

Dado N bits de representación,
el bit de signo se ubica a la izquierda (bit MSB).
El resto de los N-1 bits forman el módulo.

EJEMPLO: Con 8 bits
- ➔ 10001001= - 9.
- ➔ 00001001= +9.
- ➔ 11111111= -127.
- ➔ 01111111= +127.
- ➔ 00000000= +0.
- ➔ 10000000= - 0.

Con 16 bits
- ➔ 1000000000001001 = -9.
- ➔ 0000000000001001 =+9.

Se puede observar que existe una doble representación del “0”.

Representación de números en punto fijo con signo

Signo y módulo

Dado N bits para su representación tenemos:

Rango: 2^N

EJEMPLO: Con 8 bits ➔ Rango = 256.
Con 16 bits ➔ Rango = 65536.

Número máximo positivo a representar: $+(2^{N-1} -1)$

EJEMPLO: Con 8 bits ➔ Nºmáx. = +127.
Con 16 bits ➔ Nºmáx. = +32677.

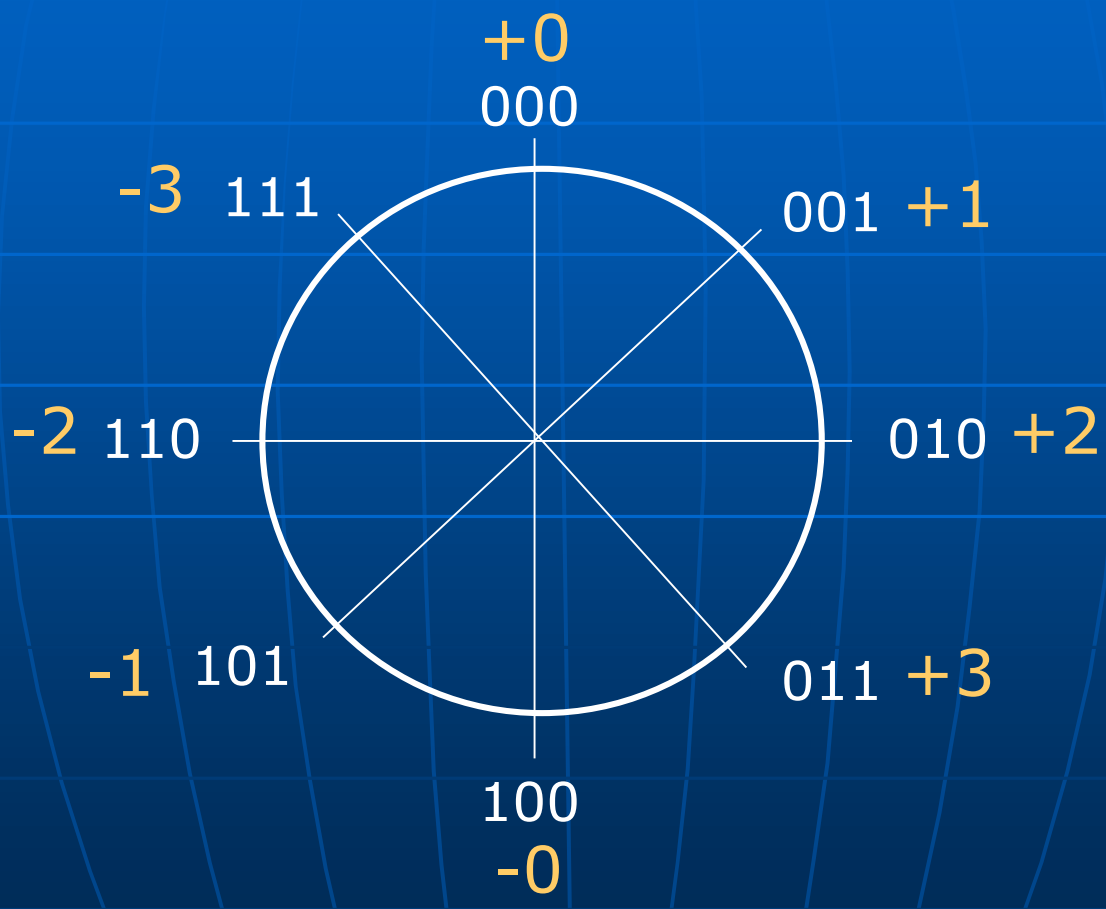
Número máximo negativo a representar: $-(2^{N-1} -1)$

EJEMPLO: Con 8 bits ➔ Nºmáx. = -127.
Con 16 bits ➔ Nºmáx. = -32677.

NOTA: Hay doble representación del "0".

Representación de números en punto fijo con signo

Representación en Signo y Módulo en 3 bits



Representación de números en punto fijo con signo

Complemento a 1 (Ca1)

Dado N bits de representación,
el bit de signo se ubica a la izquierda (bit MSB).
El resto de los N-1 bits forman el módulo.
Si el número es positivo su representación es idéntica a la del sistema "Signo y Módulo".
Si el número es negativo se debe usar la siguiente regla de conversión:

$N^{(-)} = 2^n-1-N$ donde: $N^{(-)}$ es el número negativo a buscar.
N es el número positivo de partida.
n es el número de bits de representación.

Representación de números en punto fijo con signo

Complemento a 1 (Ca1)

Dado N bits para su representación tenemos:

Rango: 2^N

EJEMPLO: Con 8 bits ➔ Rango = 256.
Con 16 bits ➔ Rango = 65536.

Número máximo positivo a representar: $+(2^{N-1} -1)$

EJEMPLO: Con 8 bits ➔ Nºmáx. = +127.
Con 16 bits ➔ Nºmáx. = +32677.

Número máximo negativo a representar: $-(2^{N-1} -1)$

EJEMPLO: Con 8 bits ➔ Nºmáx. = -127.
Con 16 bits ➔ Nºmáx. = -32677.

NOTA: Hay doble representación del "0".

Representación de números en punto fijo con signo

Complemento a 1 (Ca1)

EJEMPLOS de números positivos:

1) Representar el número $+115_{10}$ con 8 bits en Ca1:

El módulo del número es 1110011.
Colocando el signo tendremos: → 01110011.

2) Representar el número $+15_{10}$ con 8 bits en Ca1.

El módulo del número es: 1111.
Colocando el signo tendremos: → 00001111.

NOTA: En este último ejemplo se debió llenar con ceros hasta completar los 8 bits.

Representación de números en punto fijo con signo

Complemento a 1 (Ca1)

EJEMPLO de números negativos:

Representar el número -115_{10} con 8 bits en Ca1:

El módulo del número es 1110011.

Empleando la definición: ➔ $2^n - 1 - N = 2^8 - 1 - 1110011$.

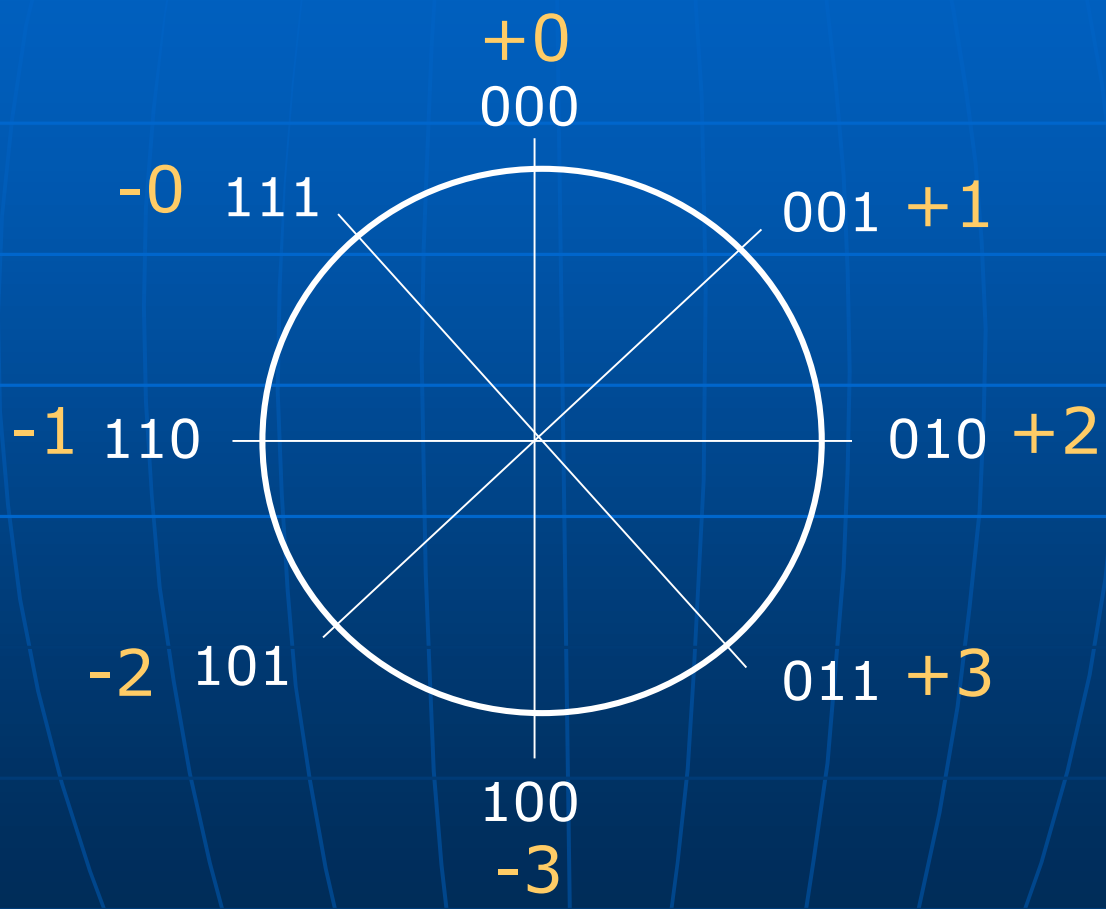
$$
\begin{array}{rl}
 & 2^8 = 100000000 \\
- & 1 = \quad\quad\quad 1 \\
\hline
 & 11111111 \\
- & 1110011 \\
\hline
\end{array}
$$

➔ RESULTADO: 10001100.

NOTA: Una manera mecánica de realizar esta conversión es la de Invertir el valor de todos los dígitos (1 por 0 y 0 por 1).

Representación de números en punto fijo con signo

Representación en Complemento a 1 en 3 bits



Representación de números en punto fijo con signo

Complemento a 2 (Ca2)

Dado N bits para su representación tenemos:

Rango: 2^N

EJEMPLO: Con 8 bits ➔ Rango = 256.
Con 16 bits ➔ Rango = 65536.

Número máximo positivo a representar: $+(2^{N-1}-1)$

EJEMPLO: Con 8 bits ➔ Nºmáx. = +127.
Con 16 bits ➔ Nºmáx. = +32677.

Número máximo negativo a representar: $-(2^{N-1})$

EJEMPLO: Con 8 bits ➔ Nºmáx. = -128.
Con 16 bits ➔ Nºmáx. = -32678.

NOTA: No hay doble representación del "0" y se puede representar un número adicional en los negativos.

Representación de números en punto fijo con signo

Complemento a 2 (Ca2)

EJEMPLOS de números positivos:

1) Representar el número $+115_{10}$ con 16 bits en Ca2:

El módulo del número es 1110011.
Colocando el signo tendremos: → 0000000001110011.

2) Representar el número $+15_{10}$ con 16 bits en Ca2.

El módulo del número es: 1111.
Colocando el signo tendremos: → 0000000000001111.

NOTA: Tanto en "Signo y módulo", "Ca1" y "Ca2" los números positivos se representar de manera idéntica …!!!!!

Representación de números en punto fijo con signo

Complemento a 2 (Ca2)

EJEMPLO de números negativos:

Representar el número -115_{10} con 8 bits en Ca2:

El módulo del número es 1110011.

Empleando la definición: ➔ $2^n - N = 2^8 - 1 - 1110011$.

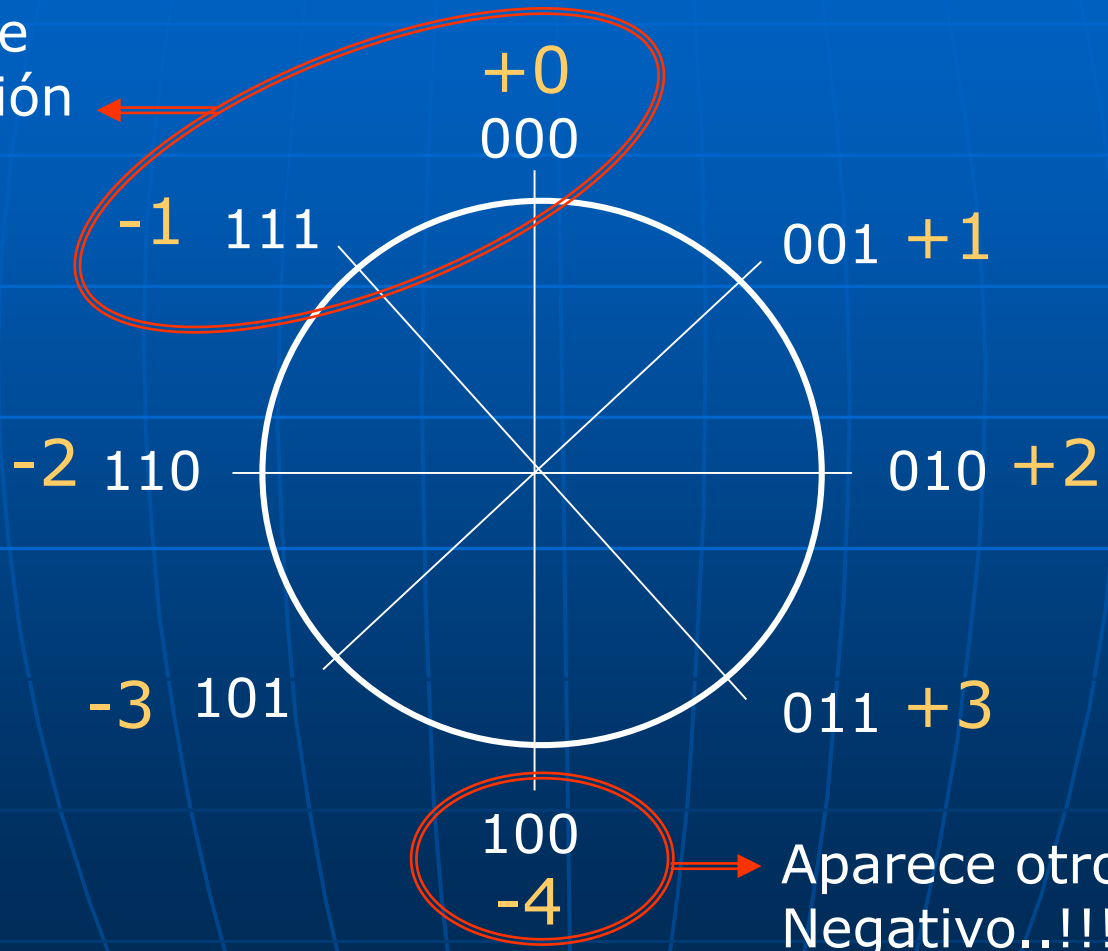
$$2^8 = 100000000$$
$$- \quad 01110011$$
$$\text{➔ RESULTADO:} \quad 10001101.$$

NOTA: Una manera mecánica de realizar esta conversión es la de Explorar el número de derecha a izquierda.
Se copian tal como están hasta que aparezca el primer "1".
Ese se deja y a partir de allí se invierten todos los demás dígitos.
Otra forma es invertir todos los bits y luego sumar "1" al resultado.

Representación de números en punto fijo con signo

Representación en Complemento a 2 en 3 bits

No hay doble representación del cero..!!!



Aparece otro nº adicional Negativo..!!!

Operaciones matemáticas

Los circuitos digitales complejos tales como microprocesadores emplean circuitos aritméticos que realizan operaciones matemáticas en el sistema de representación binario.
Esto tiene un fundamento físico ya que cada dígito binario se puede representar con sólo dos estados lógicos 0 y 1, cada uno asociado a un determinado rango de tensiones.
Las operaciones básicas son las de:

- Suma.
- Resta.
- Multiplicación.
- División.

Estas pueden ser (similar al caso del sistema decimal) en formato punto fijo ó punto flotante (notación científica).

Otras operaciones (raíces cuadradas, integrales, derivadas, etc.) se realizan empleando combinaciones de las anteriores.

SISTEMAS NUMÉRICOS

OPERACIONES MATEMÁTICAS CON NÚMEROS ENTEROS sin signo:

SUMA:

Las regla es la siguiente: 0+0=0; 0+1=1;1+0=1;1+1=0 y me llevo 1.

EJEMPLO:

$$\begin{array}{rl} & 1\ 11\ 1\ 1 \\ & 1011011 = 91_{10} \\ + & \quad 1111 = 15_{10} \\ \hline & 1101010 = 106_{10} \end{array}$$

NOTA: Cada uno de los "1" que aparecen arriba se denominan "carry" ó acarreo. Son el resultado del desborde de la operación realizada en cada posición de bit cuando se suma al menos dos "unos".

Pueden existir mas de un acarreo en una misma posición de bit, por ejemplo al sumar 4 "unos":

➔ Hacer:

$$\begin{array}{r} 111 \\ +011 \\ 011 \end{array}$$

En qué posición está El doble desborde?

SISTEMAS NUMÉRICOS

OPERACIONES MATEMÁTICAS CON NÚMEROS ENTEROS sin signo:

RESTA:

La regla es la siguiente: 0-0=0;0-1=1 y pido prestado un 1;1-0=1;1-1=0.

EJEMPLO:

$$\begin{array}{rll} & 11 & \\ & 1011011 & = 91_{10} \\ - & 1111 & = 15_{10} \\ \hline & 1001100 & = 76_{10} \end{array}$$

NOTA: Cada uno de los "1" que aparecen arriba se denominan "borrow" ó préstamo. Son el resultado del desborde de la operación realizada en cada posición de bit cuando a un 0 se le resta un 1.

OPERACIONES MATEMÁTICAS CON NÚMEROS ENTEROS sin signo:
MULTIPLICACIÓN:

EJEMPLO:

$$1011011 = 91_{10}$$

$$\times \quad 1111 = 15_{10}$$

```
  1 11
 1 1 11 1 1
 1 1011011
   1
   1 1011011
 1 1 1011011
   1011011
-----------------
10101010101
```

$$10101010101 = 1365_{10}$$

OPERACIONES MATEMÁTICAS CON NÚMEROS ENTEROS sin signo:
DIVISIÓN:

EJEMPLO:

$1011011 = 91_{10}$

$1111 = 15_{10}$

```
1011011 | 1111
 1111     110,0001.....
-----
 00111 1
   1111
  -----
  000001
   00001 0
    0001 00
     001 000
      010 000
        .
        .
```

En este caso el resultado puede tener infinitas cifras ya que no da un número divisible por potencia de 2.

Propuesta: Hacer 111/111000
Porqué el número de decimales es finito?

OPERACIONES MATEMÁTICAS CON NÚMEROS ENTEROS con signo:
SUMA Y RESTA en Complemento a 1 (Ca1):

Las regla es la siguiente: Siempre se suma; por lo tanto si se tiene que realizar por ejemplo la operación: 15-4, en realidad se hace 15+(-4). El -4 se debe hallar haciendo el complemento a 1 de +4.

EJEMPLO 1: Realizar la operación 15 – 4 en 5 bits en CA1.

+15 = 01111

+4 = 00100 ➔ $-4 = 2^n - 1 - (+4)$ ➔ -4 = 11011

```
  1 1 1 1
  01111  = +15(10)
+ 11011  = -  4(10)
-------------------
1 01010  = +10(10)
    + 1
-------------------
  01011  = +11(10)
```

➔ CORRECTO..!!

Porqué dio erróneo el resultado?

Rta: Se pasó dos veces por la representación del "0".

Solución: Sumar un "1"

OPERACIONES MATEMÁTICAS CON NÚMEROS ENTEROS con signo:
SUMA Y RESTA en Complemento a 2 (Ca2):

Las regla es la siguiente: Siempre se suma; por lo tanto si se tiene que realizar por ejemplo la operación: 15-4, en realidad se hace 15+(-4). El -4 se debe hallar haciendo el complemento a 2 de +4.

EJEMPLO 2: Realizar la operación 15 – 4 en 5 bits en CA2.

+15 = 01111
+4 = 00100 ➔ -4 = 2^n – (+4) ➔ -4 = 11100

```
    1 1
  01111  = +15(10)
+ 11100  = -  4(10)
-------------------
1 ← 01011  = +11(10)
```

Sólo es un carry;
no afecta el resultado

Porqué dio bien el resultado?

Rta: Se pasó sólo una vez por la representación del "0".

SISTEMAS NUMÉRICOS

OPERACIONES MATEMÁTICAS CON NÚMEROS ENTEROS con SIGNO:
RESÜMEN:

Operaciones de suma y resta:

Ca1 tiene la desventaja de tener que analizar si en ciertos casos el resultado dá mayor o menor que cero para saber si debe sumar o nó el carry mas significativo.
Ca2 no tiene ese problema lo que simplifica la lógica.
Ca1 tiene la ventaja que permite complementar un número simplemente invirtiendo el estado lógico de cada bit (se resuelve con una compuerta OR-Exclusiva).
Ca2 debe complementar con un paso extra: inversión de bits y suma de un "1".

Operaciones de multiplicación y división:

No se realizan en Ca1 o Ca2 dado lo complejo de las operaciones.
Se realizan en Signo y Módulo, poniendo el signo al resultado en paralelo con la operación realizada.

Representación de números en punto fijo

Formato BCD:

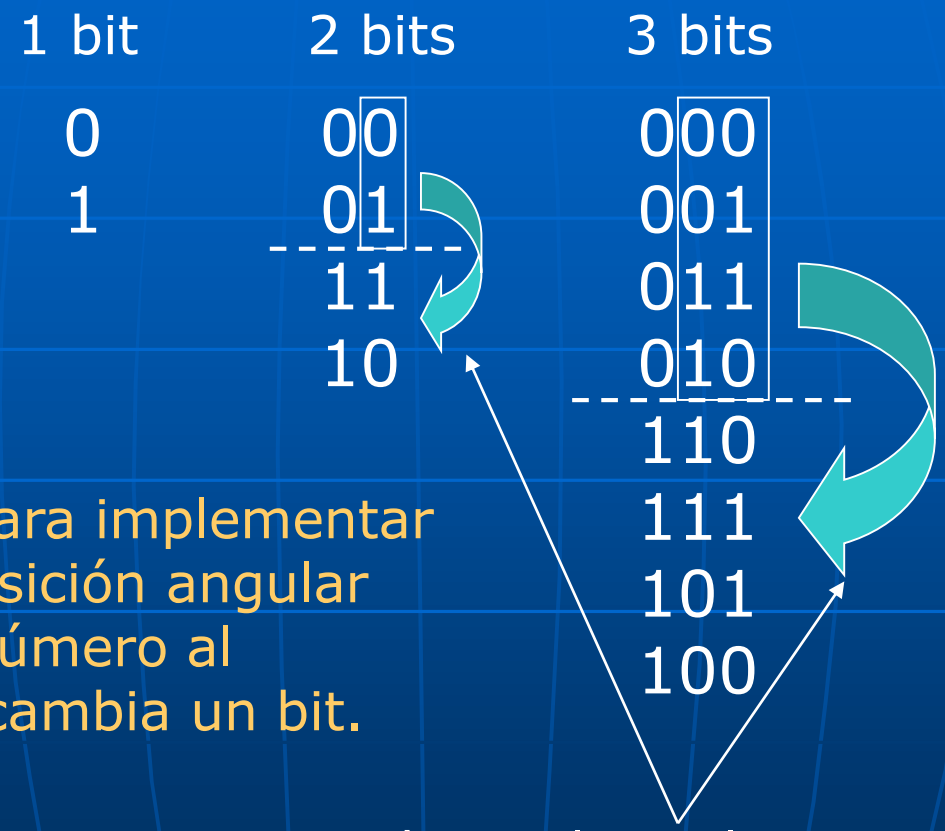
Sistema de representación de números decimales donde cada dígito se empaqueta en formato binario de 4 bits. Útil en sistemas de visualización con display.

EJEMPLO:

$$987{,}023_{10} = \underbrace{1001}_{9}\underbrace{1000}_{8}\underbrace{0111}_{7},\underbrace{0000}_{0}\underbrace{0010}_{2}\underbrace{0011}_{3}$$

Código de Gray:

Código que emplea dígitos binarios (0 y 1) pero con una secuencia particular.



Gran utilidad para implementar sensores de posición angular ya que de un número al siguiente sólo cambia un bit.

Se rebaten las columnas sobre el la línea de trazos como en un espejo.

Representación de números en punto flotante:

Método de representación para números con signo en notación científica.

Formato IEEE P754: Normalizado por la IEEE tiene a su vez diferentes subformatos:

+ Precisión simple.
+ Precisión doble.
+ Precisión extendida.
+ Representación en BCD.
+ etc.

Representación de números en punto flotante:

Formato IEEE P754: Precisión simple.

Se divide el número a representar en 3 campos:

+ Signo del número (1 bit).
+ Fracción del significando (23 bits).
+ Exponente con signo incluído (8 bits).

Signo: "1" si el número es negativo, "0" si es positivo.
Fracción del significando: Se representa sólo la parte decimal de la mantisa ó significando. Se asume que la parte entera es igual a 1.
Exponente: Con 8 bits (desde 0 a 255) se fija el número 127 como el "bias" tal que representa al exponente "0".
Todo exponente positivo será mayor a 127 y negativo menor a 127.
Ejemplos:
Si el exponente dá +15 se debe poner 127+15 = 10001110.
Si el exponente dá -15 se debe poner 127–15 = 01110000.

NOTA: Se reservan las combinaciones del exponente 0….0 y 1…1 que Junto con otras del campo de fracción del significando sirven para avisar al sistema de ciertas condiciones alcanzadas.

Representación de números en punto flotante:

Formato IEEE P754: Precisión simple.

EJEMPLO:

Representar el número - $13{,}625_{10}$

Paso 1: Llevar el número a una notación en punto fijo:
13,625 = 1101,101.

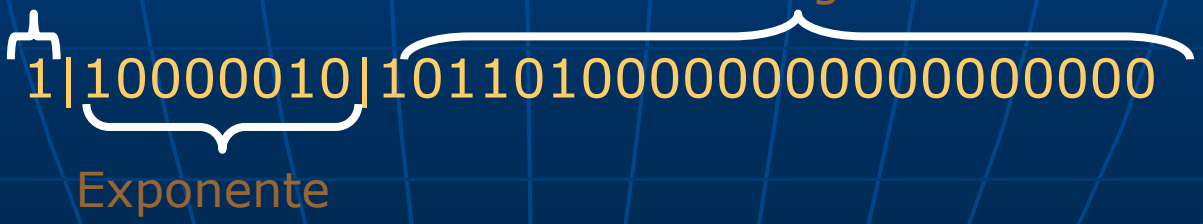
Paso 2: Expresarlo tal que la mantisa quede con parte entera igual a "1" ➔ $1{,}101101 \times 2^{+3}$.

Paso 3: En el campo de fracción de significado poner:
10110100000000000000000 (23 bits en total).

Paso 4. En el campo del exponente sumar 3 a 01111111 ya que es positivo ➔ 10000010.

Paso 5: Poner el campo del signo del número en "1".

RESULTADO:



Representación de números en punto flotante:

Formato IEEE P754: Precisión doble.

Se divide el número a representar en 3 campos:

+ Signo del número (1 bit).
+ Fracción del significando (52 bits).
+ Exponente con signo incluído (11 bits).

Es similar al de precisión simple, salvo que tiene mayor rango para representar números mas chicos y/o mas grandes.

Signo: "1" si el número es negativo, "0" si es positivo.
Fracción del significando: Se representa sólo la parte decimal de la mantisa ó significando. Se asume que la parte entera es igual a 1.
Exponente: Con 11 bits (desde 0 a 2047) se fija el número 1023 como el "bias" tal que representa al exponente "0".

NOTA: Se reservan las combinaciones del exponente 0….0 y 1…1 que Junto con otras del campo de fracción del significando sirven para avisar al sistema de ciertas condiciones alcanzadas.

Bibliografía:

Apuntes de teoría:

- "Sistemas de Representación Numéricos". S. Noriega.
- "Operaciones matemáticas con números binarios". S. Noriega.
- "Representación de números binarios en punto fijo y punto flotante". S. Noriega.

Libros:

- "Sistemas Digitales". R. Tocci, N. Widmer, G. Moss. Ed. Prentice Hall.
- "Diseño Digital". M. Morris Mano. Ed. Prentice Hall. 3ra edición.
- "Diseño de Sistemas Digitales". John Vyemura. Ed. Thomson.
- "Diseño Lógico". Antonio Ruiz, Alberto Espinosa. Ed. McGraw-Hill.
- "Digital Design:Principles & Practices". John Wakerly. Ed. Prentice Hall.
- "Diseño Digital". Alan Marcovitz. Ed. McGraw-Hill.
- "Electrónica Digital". James Bignell, R. Donovan. Ed. CECSA.
- "Técnicas Digitales con Circuitos Integrados". M. Ginzburg.
- "Fundamentos de Diseño Lógico y Computadoras". M. Mano, C. Kime. Ed. Prentice Hall.
- "Teoría de conmutación y Diseño lógico". F. Hill, G. Peterson. Ed. Limusa