Sistemas de numeración

Ing Florencia Ferrigno

Mayo 2016

UTN - FRBA

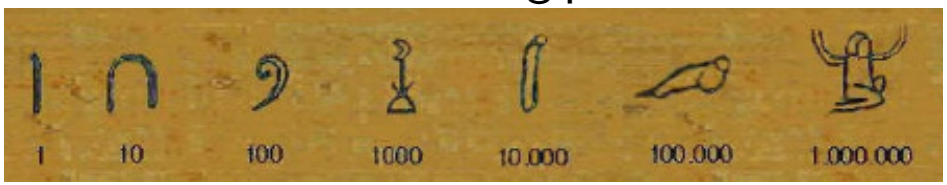
Que es un sistema de numeración?

Que es un sistema de numeración?

- Conjunto de símbolos y reglas que permiten representar datos numéricos.

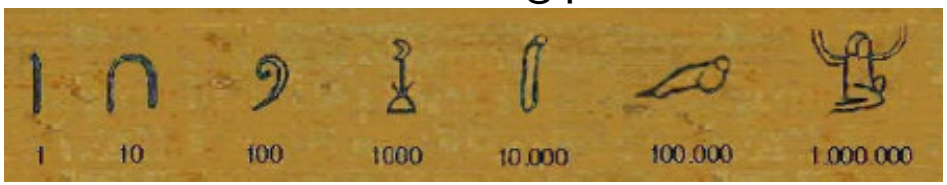
Que es un sistema de numeración?

- Conjunto de símbolos y reglas que permiten representar datos numéricos.
- Los sitemas de numeración se pueden clasificar en:
  - **No posicionales**: Son los sistemas más antiguos, donde no importa la posición del símbolo, este siempre vale lo mismo. Como por ejemplo el sistema de numeración egipcio.



Que es un sistema de numeración?

- Conjunto de símbolos y reglas que permiten representar datos numéricos.
- Los sitemas de numeración se pueden clasificar en:
  - **No posicionales**: Son los sistemas más antiguos, donde no importa la posición del símbolo, este siempre vale lo mismo. Como por ejemplo el sistema de numeración egipcio.



  - **Semi posicionales**: No son estrictamente posicionales y algunos de los símbolos tienen el mismo valor independientemente del lugar que ocupen. Un ejemplo de este sistema es la numeración romana.

I	V	X	L	C	D	M
1	5	10	50	100	500	1.000

Que es un sistema de numeración?

- **Posicionales**:
  - En estos sistemas la posición de cada dígito determina su valor tanto globalmente como individualmente. Nuestro sistema de numeración es posicional
  - Ejemplo: **375**: El 3 representa las centenas, por lo tanto son 300 unidades. Sin embargo el mismo 3 en el número **213** representa las unidades, por lo tanto hay 3 unidades

Base de representación

- El número de símbolos permitidos en un sistema de numeración se lo denomina base del sistema de numeración.

Base de representación

- El número de símbolos permitidos en un sistema de numeración se lo denomina base del sistema de numeración.
- Cuando decimos que un sistema es de base **b**, significa que disponemos de b elementos para representar los números.

Base de representación

- El número de símbolos permitidos en un sistema de numeración se lo denomina base del sistema de numeración.
- Cuando decimos que un sistema es de base **b**, significa que disponemos de b elementos para representar los números.
- Entonces cuando hablamos que nuestro sistema de numeración es decimal, significa que su base es 10, por lo tanto contamos con 10 elementos para representar los números: 0 1 2 3 4 5 6 7 8 9

Teorema fundamental de la numeración

- Establece la forma general de construir números en un sistema de numeración posicional.

Teorema fundamental de la numeración

- Establece la forma general de construir números en un sistema de numeración posicional.
- Sea:
  - **N**: número válido en el sistema de numeración
  - **b**: base del sistema de numeración
  - **d**: cualquier símbolo permitido en el sistema de numeración
  - **n**: número de dígitos de la parte entera (la posición que ocupa)
  - **,**: coma fraccionaria
  - **k**: número de dígitos de la parte decimal

Teorema fundamental de la numeración

- Entonces un número cualquiera se lo puede definir como:
- $N = d_{(n-1)} \ldots d_1 d_0, d_{-1} \ldots . d_{-k}$

Teorema fundamental de la numeración

- Entonces un número cualquiera se lo puede definir como:
- $N = d_{(n-1)}...d_1 d_0, d_{-1}....d_{-k}$
- Por lo tanto tambien podríamos definir N como:
- $\sum_{i=-k}^{n-1} d_i * b^i = N$

Teorema fundamental de la numeración

- Entonces un número cualquiera se lo puede definir como:
- $N = d_{(n-1)}...d_1 d_0, d_{-1}....d_{-k}$
- Por lo tanto tambien podríamos definir N como:
- $\sum_{i=-k}^{n-1} d_i * b^i = N$
- Es decir que el número 123,45 lo podremos expresar como:
- $1 * 10^2 + 2 * 10^1 + 3 * 10^0, 4 * 10^{-1} + 5 * 10^{-2}$

Bases con las cuales vamos a trabajar

- **Decimal**: Sistema de base 10. Los símbolos permitidos para expresar números serán: 0 1 2 3 4 5 6 7 8 9

Bases con las cuales vamos a trabajar

- **Decimal**: Sistema de base 10. Los símbolos permitidos para expresar números serán: 0 1 2 3 4 5 6 7 8 9
- **Binario**: Sistema de base 2. Sus símbolos serán únicamente: 0 1.
  A estos símbolos se los conoce como bit, que es el anacrónimo de la palabra en inglés *binary digit*

Bases con las cuales vamos a trabajar

- **Decimal**: Sistema de base 10. Los símbolos permitidos para expresar números serán: 0 1 2 3 4 5 6 7 8 9
- **Binario**: Sistema de base 2. Sus símbolos serán únicamente: 0 1.
  A estos símbolos se los conoce como bit, que es el anacrónimo de la palabra en inglés *binary digit*
- **Octal**: Sistema de base 8 y sus símbolos serán: 0 1 2 3 4 5 6 7

Bases con las cuales vamos a trabajar

- **Decimal**: Sistema de base 10. Los símbolos permitidos para expresar números serán: 0 1 2 3 4 5 6 7 8 9
- **Binario**: Sistema de base 2. Sus símbolos serán únicamente: 0 1.
  A estos símbolos se los conoce como bit, que es el anacrónimo de la palabra en inglés *binary digit*
- **Octal**: Sistema de base 8 y sus símbolos serán: 0 1 2 3 4 5 6 7
- **Hexadecimal**: Sistema de base 16. 0 1 2 3 4 5 6 7 8 9 A B C D E F

Bases con las cuales vamos a trabajar

- **Decimal**: Sistema de base 10. Los símbolos permitidos para expresar números serán: 0 1 2 3 4 5 6 7 8 9
- **Binario**: Sistema de base 2. Sus símbolos serán únicamente: 0 1.
  A estos símbolos se los conoce como bit, que es el anacrónimo de la palabra en inglés *binary digit*
- **Octal**: Sistema de base 8 y sus símbolos serán: 0 1 2 3 4 5 6 7
- **Hexadecimal**: Sistema de base 16. 0 1 2 3 4 5 6 7 8 9 A B C D E F

Sistema Binario

- Como dijimos antes es un sistema en el cual los números se representan usando dos cifras: 0 1
- Es muy útil para representar los dos únicos estados o niveles de tensión que pueden adquirir los circuitos que conforman la computadora

Sistema Binario

- Como dijimos antes es un sistema en el cual los números se representan usando dos cifras: 0 1
- Es muy útil para representar los dos únicos estados o niveles de tensión que pueden adquirir los circuitos que conforman la computadora
- Al dígito de numeración binaria se lo conoce como bit y es la unidad mínima de información ya que dentro de la computadora lo mínimo que se puede almacenar es un bit.

Sistema Binario

- Como dijimos antes es un sistema en el cual los números se representan usando dos cifras: 0 1
- Es muy útil para representar los dos únicos estados o niveles de tensión que pueden adquirir los circuitos que conforman la computadora
- Al dígito de numeración binaria se lo conoce como bit y es la unidad mínima de información ya que dentro de la computadora lo mínimo que se puede almacenar es un bit.
- La unidad de información de base utilizada en informática es el byte y equivale a un conjunto de 8 bits por lo cual también es común escuchar referirse al byte como **octeto**

Sistema Binario

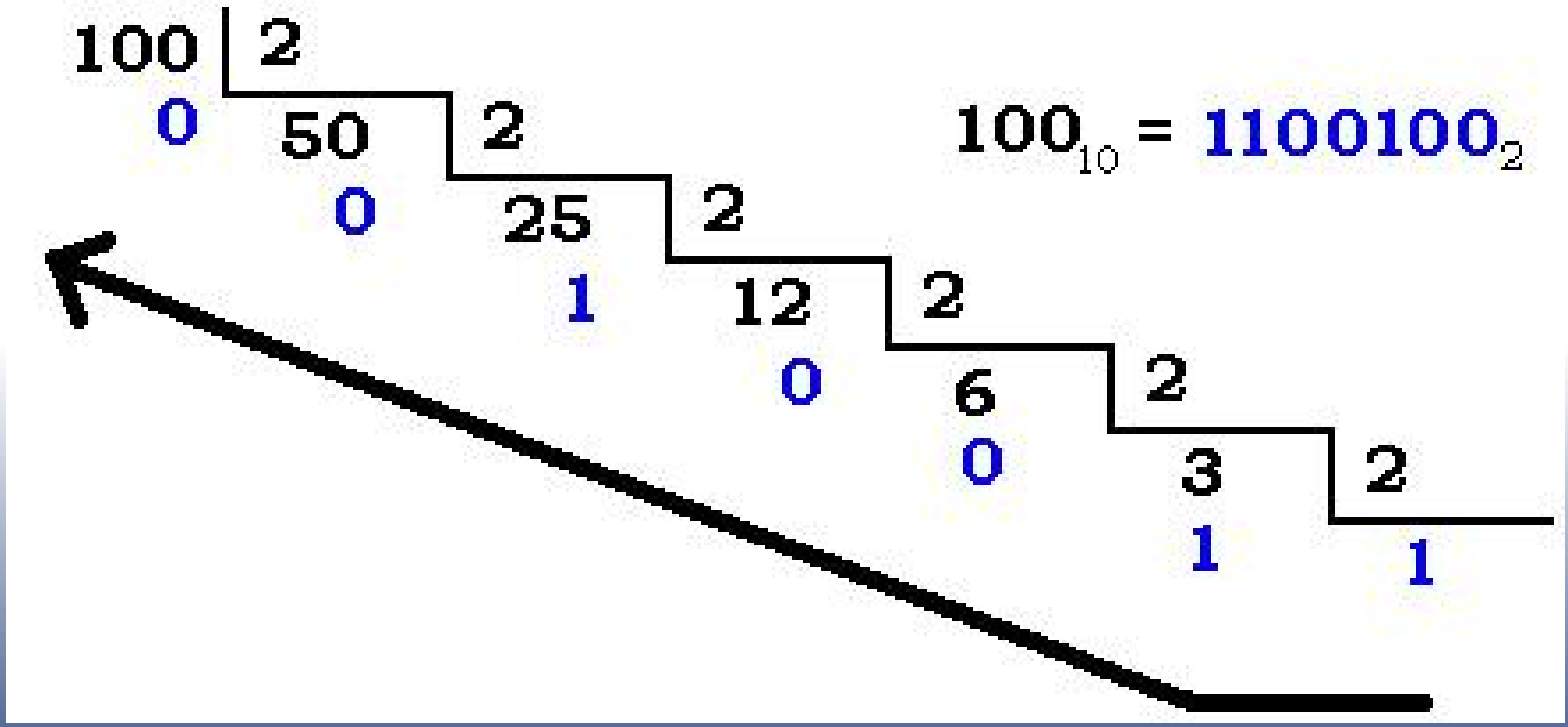
- **Palabra**: Conjunto de bits que pueden ser accedidos por la CPU en un proceso de lectura o escritura. Su tamaño depende de la arquitectura de la máquina.

Sistema Binario

- **Palabra**: Conjunto de bits que pueden ser accedidos por la CPU en un proceso de lectura o escritura. Su tamaño depende de la arquitectura de la máquina.
- **Tamaño de la palabra** para representar un número en binario: La cantidad de dígitos necesarios para representar un número en el sistema binario es mayor que en el sistema decimal. Como regla general con una palabra de n dígitos se puede representar como máximo 2^n números. El número más grande que se podrá representar será: $2^n - 1$

Conversión de decimal a binario

- **Método de las divisiones sucesivas**:Se divide el número decimal por 2, cuyo resultado se vuelve a dividir por 2 y así sucesivamente hasta que el resultado sea menor a 2. Es decir cuando el número a dividir sea 1 o 0.
- **Ejemplo:** Vamos a convertir el número 100 en base 10 a binario.



Método de las divisiones sucesivas

- **En líneas generales**: Se divide el número (en base x) por la base a la cual se quiere convertir (en base y), cuyo resultado se vuelve a dividir por esa misma base y así sucesivamente hasta que el resultado sea menor a la misma. Y por último el resultado será el cociente (el último) y los restos en sentido inverso al que fueron calculados.

Sistema Binario

Conversiónes rápidas

- **Nibble**:También conocido como **cuarteto**. Es un conjunto de 4 bits. Su interés surge por que cada cifra hexadecimal requiere 4 bits para ser representada en binario. Esto surge dado que $2^4 = 16$ que es la cantidad de símbolos válidos para la base hexadecimal.
- Sabiendo esto, ahora cuando se quiera convertir un número binario a hexa, deberemos agrupar los bits en nibbles y convertir cada uno individualemente. Siempre teniendo en cuenta que si ese nibble da mayor a 9 se deberán usar las letras.
- Recordar la siguiente tabla:

Conversiones rápidas

Binario	Decimal	Octal	Hexa
1010	10	12	0A
1011	11	13	0B
1100	12	14	0C
1101	13	15	0D
1110	14	16	0E
1111	15	17	0F

Conversiones rápidas

- **De binario a hexa**:

Binario	1010	0100	1110	1101	0111	1001	0010	1011
Hexadecimal	A	4	E	D	7	9	2	B

- **De Binario a Octal**:Se toman grupos de 3 bits y se hace la conversión directa. Tener en cuenta que al ser grupos de 3 bits, no podremos tener números mayores a 7!

Binario	110	100	110	101	111	001	010	011
Octal	6	4	6	5	7	1	2	3

- **De Octal a Binario**:Cada uno de los dígitos en octal se convierte de manera directa a un número binario de 3 bits.

Octal	6	4	6	5	7	1	2	3
Binario	110	100	110	101	111	001	010	011

Números con signo

- En matemática para representar números negativos, basta con representarlos con el signo menos. Sin embargo en los sistemas binarios esa no es una opción.
- Existen cuatro métodos para representar los números menores a cero:
  - **Signo y Magnitud**
  - **Complemento a 1**
  - **Complemento a 2**
  - **Binario desplazado**

Signo y Magnitud

- Este primer enfoque se basa en asignar el **bit más significativo** (MSB) para representar el signo del número. Por convención se asignó el 0 para representar los positivos y el 1 para los negativos
- Los n-1 bits se utilizarán para representar al número. Es decir representan la magnitud.

Signo y Magnitud

- Sea una palabra de 8 bits, tendremos un bit para el signo y 7 bits para representar la magnitud. Es decir que nos da la posiblidad de representar $2^8 = 256$ números, tendremos $2^7 = 128$ números positivos y otros $2^7 = 128$ números negativos?
  $127_{10} = 01111111$
  $0_{10} = 00000000$
  $0_{10} = 10000000$
  $-127_{10} = 11111111$
- La principal desventaja que tiene este sistema es la doble representación del cero.

Complemento a 1

- El complemento a uno se basa en invertir cada uno de los bits, en otras palabras cambiar los unos por los ceros y viceversa.(Es decir, complementarlos)
- En este sistema de representación también se asigna el MSB para el signo, y continuando con la misma convención el cero representará a los positivos y el 1 a los negativos. Los *n-1* bits restantes serán para la magnitud
- Ejemplo: $Ca_1(001010110)_2 = (110101001)_2$
- Sin embargo no se soluciona la doble representación del cero.

Complemento a 1

- Hay que prestar especial atención al operar aritmeticamente, ya que al sumar en Co_1 hay que tener en cuenta el *acarreo*
- Por ejemplo: si queremos sumar $2_{10} + (-1_{10}) = 1_{10}$ en complemento a 1 sería: $00000010_2 + 11111110_2 = 100000000_2$ Como ese carry no esta contemplado, parecería que el resultado de dicha suma es cero.
- Para obtener el resultado correcto, hay que hacer una corrección por carry: $00000010_2 + 11111110_2 = 00000000_2 + 00000001 = 00000001_2$ Que es el resultado correcto.

Complemento a 2

- El complemento a dos se basa realizar el complemento a 1 y sumarle 1
- Ejemplo: $Ca_2(001010110)_2 = Ca_1(001010110)_2 + (1)_2 = (110101010)_2$
- **Conversión rápida**:Tomamos el número binario y comenzando por la derecha (por el LSB- bit menos significativo-) copiamos bit a bit hasta encontrar el primer 1, a partir de ahi se niegan todos los bits.
- El complemento a 2 elimina la ambigüedad del cero.
  $00000000_2 = 0_{10}$
  $00000001_2 = 1_{10}$
  $01111110_2 = +126_{10}$
  $01111111_2 = +127_{10}$
  $10000000_2 = -128_{10}$
  $10000001_2 = -127_{10}$
  $11111111_2 = -1_{10}$

Complemento a 2

- La resta de números binarios se facilita con el uso del complemento a 2 ya que la resta se puede obtener como la suma del minuendo con el complemento a 2 del sustraendo.Se utiliza porque la unidad aritmético-lógica no resta números binarios, suma binarios negativos, por eso esta conversión al negativo.

Binario desplazado

- Consiste en sumarle a cada número signado un mismo valor y se le dice que esta en exceso por ese valor.
- Por eso también se lo conoce como **representación en exceso**.
- El valor que se suele tomar es $2^{n-1} - 1$.
- Para una palabra de 8 bits, el exceso será: $2^{8-1} - 1 = 2^7 - 1 = 127$
  $00000000_2 = -127_{10}$
  $00000001_2 = -126_{10}$
  $01111110_2 = -1_{10}$
  $01111111_2 = 0_{10}$
  $11111110_2 = +127_{10}$
  $11111111_2 = +128_{10}$

Números reales

- Hasta ahora vimos como representar números naturales en diferentes bases.
- Sin embargo en la vida real no todo se representa con números naturales.
  - Para un ingeniero construyendo un edificio no importa si tiene 10 metros de altura o si tiene 10,0001 metros
  - Para un diseñador de microchips, 0,0001 metros es una diferencia enorme
  - Un físico tiene que manejar valores como la velocidad de la luz (300.000.000 m/s) y la constante de gravitación universal (0.0000000000667 $N.m^2/kg^2$)
- Para satisfacer estas necesidades será necesario trabajar **números de precisión finita**

Números de precisión finita

- El rango de números reales comprende del $-\infty$ al $+\infty$
- Sin embargo, los registros de una computadora no cuentan con semejante resolución, tienen resolución finita.
- Por lo tanto se dice que la computadora sólo puede representar un subconjunto de los números reales.
- Para representar este subconjunto se utilizan los siguientes formatos:
  - **Punto Fijo**
  - **Coma flotante**

Números de Punto Fijo

- La representación será del tipo: $(a_n a_{n-1}...a_0, a_{-1}a_{-2}...a_{-m}) = (-1)^s * (a_n * 2^n + ... + a_0 * 2^0 + a_{-1} * 2^{-1} + a_{-2} * 2^{-2} + ... + a_{-m} * 2^{-m})$
- donde:
  - s es 0 si es positivo y 1 si es negativo.
  - a_i pertenece a los enteros y $0 < a_i < 1$ para todo i=-m,..-1,0,1,..n
- La distancia entre dos números consecutivos: 2^{-m}
- Deja de ser un rango continuo de números para transformarse en un rango discreto.

Coma flotante

- La notación en coma flotante (Que viene del inglés **floating point**) surgió de la necesidad de poder representar números racionales extremadamente grandes de manera eficiente
- Esta se basa en representar a los números en notación científica

Coma flotante

- La notación en coma flotante (Que viene del inglés **floating point**) surgió de la necesidad de poder representar números racionales extremadamente grandes de manera eficiente
- Esta se basa en representar a los números en notación científica
- Los números en coma flotante se conforman de 3 partes:
  1. El signo
  2. La mantisa: Representa la magnitud del número
  3. Exponente: indica el desplazamiento de la coma

Coma flotante

- De esta manera podríamos representar a un número decimal $3564021 = 0,3564021 * 10^8$

Coma flotante

- De esta manera podríamos representar a un número decimal $3564021 = 0,3564021 * 10^8$
- Por lo que podemos desacoplar
  1. Signo: $+$
  2. Mantisa: 3564021
  3. Exponente: 8

Coma flotante

- El formato que se utiliza para la representación de números binarios en coma flotante está definido por el éstandar 514-1985 ANSI/IEE
- El formato más simple de precisión (lo que en C definimos como **float**) utiliza 32 bits para su representación

signo	exponente	mantisa
1 bit	8 bits	23 bits

Mantisa

- La mantisa siempre se almacena de forma normalizada, es decir: $1011,011 -> 0,1011011 * 2^4$

Mantisa

- La mantisa siempre se almacena de forma normalizada, es decir: $1011,011 -> 0,1011011 * 2^4$
- Como podemos ver de forma intuitiva, siempre nos queda un 1 en el primer decimal por lo tanto

Mantisa

- La mantisa siempre se almacena de forma normalizada, es decir: $1011,011 -> 0,1011011 * 2^4$
- Como podemos ver de forma intuitiva, siempre nos queda un 1 en el primer decimal por lo tanto
- **Aprovechamos otro bit más!**, y normalizamos de manera que siempre nos quede un 1 en la parte entera y este es el valor de mantisa que guardamos $1011,011 \rightarrow 0,1011011 * 2^4 \rightarrow 1,011011 * 2^3$

Exponente

- Como bien sabemos los números en coma flotante son usados justamente para guardar decimales. Esto significa que deberíamos usar un bit del exponente para el signo
- Pero para ahorrarlo, utilizamos el exponente en el formato denominado **exceso**
- Este consiste en sumar un desplazamiento al valor real del número
  1. Para esto restamos 1 al entero más grande que podamos representar y luego lo dividimos por 2
  2. En nuestro caso como utilizamos 8 bits para el exponente nuestro mayor número posible seria 255
  3. Por lo tanto desplazamos $\frac{255-1}{2} = 127$

Exponente

- Representar en formato exceso nos permite entonces representar valores entre -126 y 127 de esta manera
  $exponente = -126 \rightarrow -126 + 127 = 1 \rightarrow 00000001$
  $exponente = -56 \rightarrow -56 + 127 = 1 \rightarrow 01000111$
  $exponente = 0 \rightarrow 0 + 127 = 1 \rightarrow 01111111$
  $exponente = 1 \rightarrow -126 + 127 = 1 \rightarrow 10000000$
  $exponente = 25 \rightarrow 25 + 127 = 152 \rightarrow 10011000$
  $exponente = 127 \rightarrow 127 + 127 = 254 \rightarrow 11111110$

Casos particulares

- El sistema en exceso presenta dos casos particulares de exponente
- El valor -127 (00000000) representa dos casos particulares
  1. Si la mantisa es = 0, representa el 0
  2. si la mantisa es != 0 se trata de un número no normalizado y el exponente es -126
- El valor 128 (11111111) representa dos casos particulares
  1. Si la mantisa es = 0, representa el infinito
  2. si la mantisa es != 0 se trata de una situación donde no se pudo realizar la operación. Por lo tanto el valor no es un número (**NaN**)

Ventajas

- El sistema nos pemite representar números de 128 bits con solo 32

Ventajas

- El sistema nos pemite representar números de 128 bits con solo 32
- El sistema nos permite representar números reales

Ventajas

- El sistema nos pemite representar números de 128 bits con solo 32
- El sistema nos permite representar números reales
- El sistema es flexible: Le podemos dedicar cualquier cantidad de btis tanto a la parte entera como a la fraccionaria segun convenga