Met4OP : METODOLOGÍA DE ANÁLISIS EN OPINIÓN PÚBLICA

Nociones preliminares

UBA
Universidad de Buenos Aires

Sistemas de Numeración

Conjunto de principios y reglas que permiten representar datos numéricos.

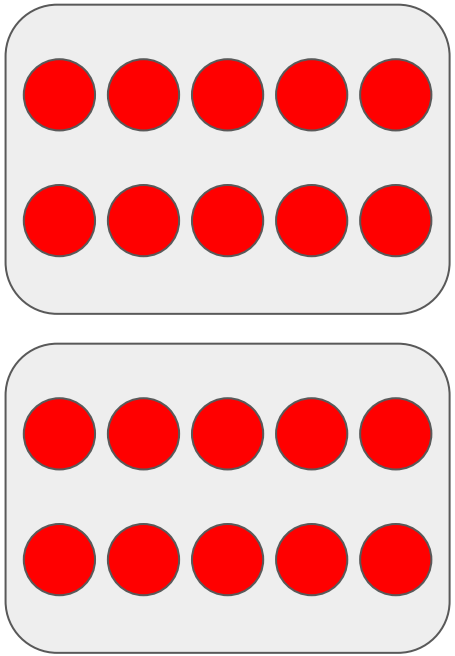
Principios:

1-ORDEN: se cuenta de derecha a izquierda (no confundir con el lugar).

2-BASE: número entero mayor que la unidad que nos indica la forma de agrupamiento.

3-POSICIONAL: toda cifra posee un valor posicional dado por el mismo número.

Ejemplo de Base



$21_{(10)}$

Sistema Decimal

Utiliza diez símbolos (0,1,2,3,4,5,6,7,8 y 9) y es el más conocido por nosotros.

Las operaciones que en él se pueden dar son las aritméticas(suma, resta, multiplicación, división, potenciación, etc.) y lógicas (Unión - disyunción, Intersección - conjunción, negación, Diferencia, Complemento, etc.).

Las relaciones entre los números del sistema decimal son: mayor que, menor que, igual y a nivel lógico son pertenencia y contenencia.

$$234{,}21_{10} = 2*10^{2} + 3*10^{1} + 4*10^{0} + 2*10^{-1} + 1*10^{-2}$$

Sistema Binario

Utiliza dos símbolos (0,1). Las computadoras trabajan internamente con 2 niveles: hay o no hay de tensión, hay o no hay corriente, pulsado o sin pulsar, etc.

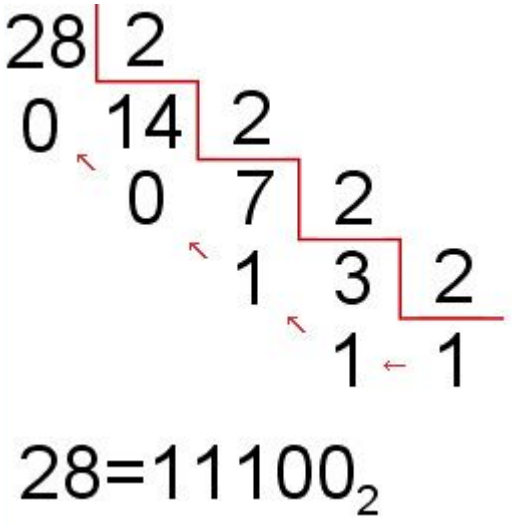
Cada dígito o número en este sistema se denomina bit (contracción de binary digit).

$$\mathbf{111000}_{2} = \mathbf{56}_{10}$$

Pasar de Decimal a Binario

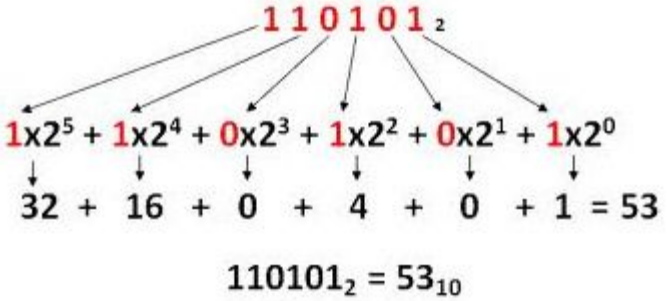
Para representar un número en un sistema diferente al decimal, se emplea el método de:

“Divisiones Sucesivas”

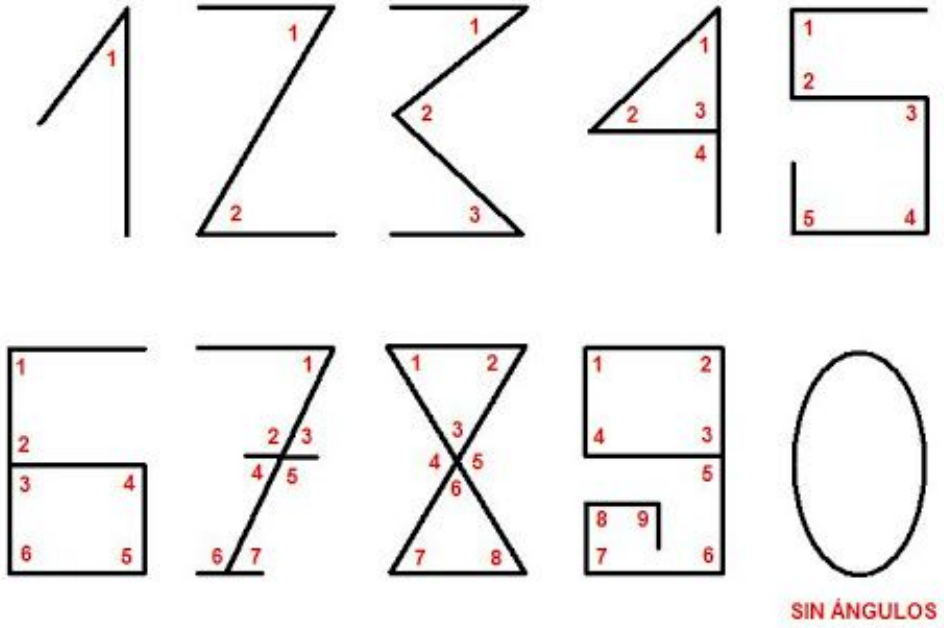


Pasar de Binario a Decimal

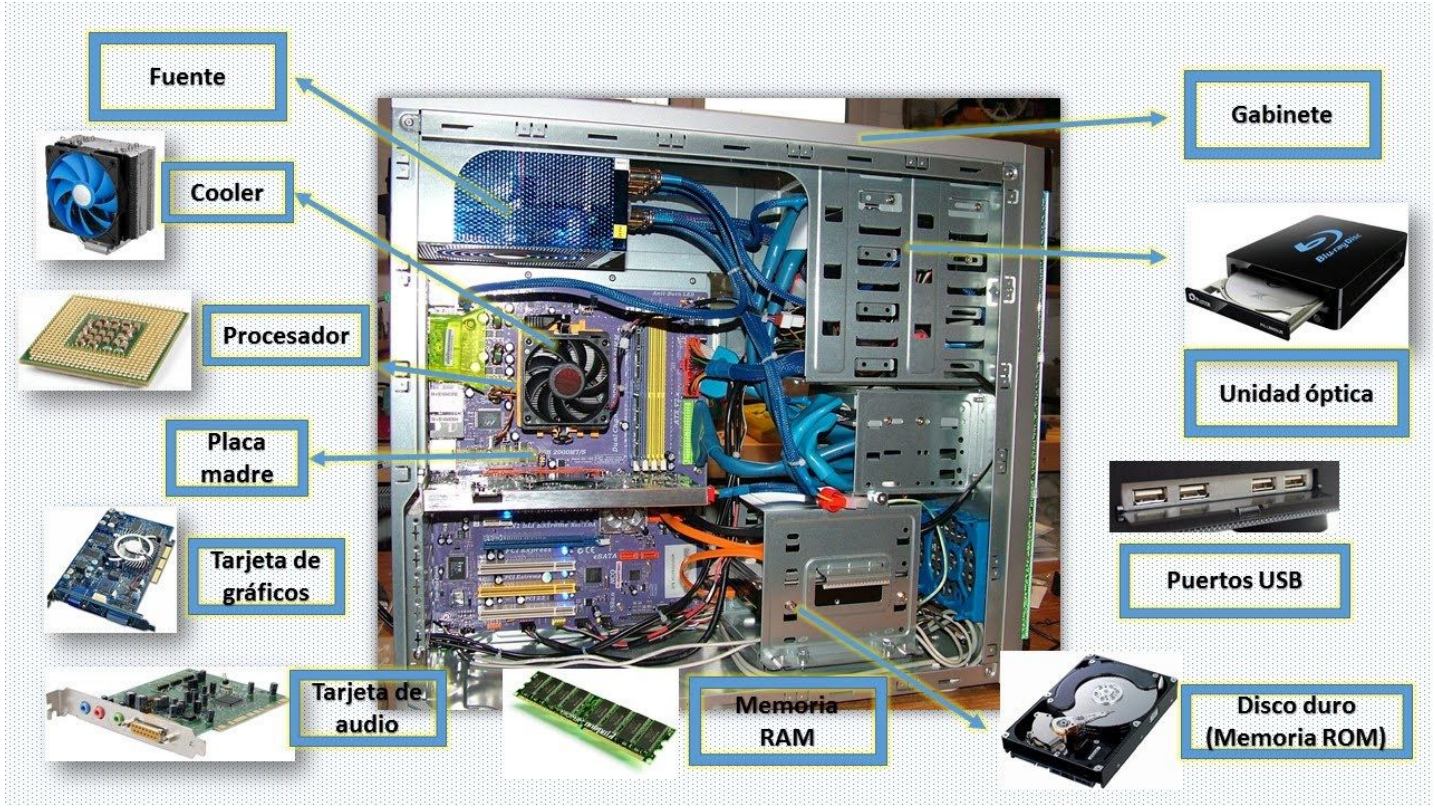
- Escribir el número **binario** y listar las potencias de 2 de derecha a izquierda.
- Escribir los dígitos del número **binario** debajo de sus potencias correspondientes.
- Conectar los dígitos del número **binario** con sus potencias correspondientes.
- Escribe el valor final de cada potencia de dos.
- Suma los valores finales.



Significación de los números



Partes de la Computadora



Funcionamiento de la memoria

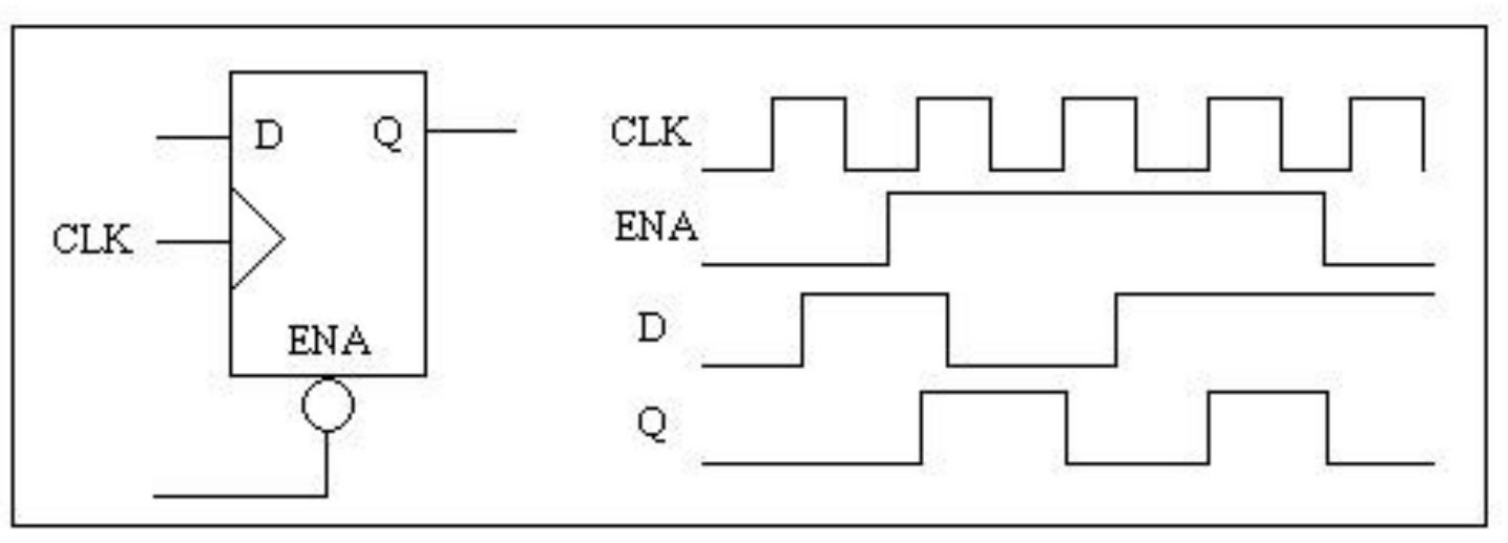
Un modo sencillo de comprender el funcionamiento de la memoria es asemejarlo con un gran panel constituido por un conjunto de casillas o células, denominadas posiciones de memoria, en las que se almacenan los datos.

La computadora debe saber exactamente dónde se encuentra cada dato en la memoria, por lo que identifica cada posición de la memoria mediante un número denominado dirección de memoria.

Cada posición de memoria almacena un byte, lo que hace pensar en la gran cantidad de posiciones que serán necesarias para poder almacenar instrucciones y datos; este es el motivo por el que las computadora cada vez disponen de mayor cantidad de memoria.

Celda de Memoria

Circuito más pequeño que es capaz de almacenar la información de un bit. El dispositivo más simple, el Flip-Flop D. :



Conceptos importantes

Palabra de memoria: Es un grupo de bits (ó celdas de memoria) que representan un dato o una instrucción. El “ancho” de la palabra puede ser desde 1 hasta N bits

Byte: Es un grupo de 8 bits (unidad mínima de memoria que representa un 0 o un 1).

Nibble: Es la mitad de un Byte, es decir una palabra de 4 bits.

Bit: unidad mínima de información empleada. Puedo representar valores booleanos como verdadero o falso y abierto o cerrado.

Almacenamiento

1 bit	Puede tomar dos valores: 1 ó 0
1 byte	8 bits
1 kilobyte (Kb)	2^{10} bytes
1 megabyte (Mb)	2^{10} Kilobytes: 2^{20} bytes
1 gigabyte	2^{10} megabytes: 2^{30} bytes
1 terabyte	2^{10} gigabytes: 2^{40} bytes
1 petabyte	2^{10} terabytes: 2^{50} bytes
1 exabyte	2^{10} petabytes: 2^{60} bytes
1 zettabyte	2^{10} exabytes: 2^{70} bytes
1 yottabyte	2^{10} zettabytes: 2^{80} bytes

Más conceptos importantes

Char: Ocupa en la memoria 1 byte (8 bits) y permite representar en el sistema numérico binario 2^8 valores = 256. El tipo char puede contener los valores positivos, igual que negativos. El rango de valores es de -128 a 127.

Int: tiene el tamaño de 4 bytes (32 bits). El valor mínimo es de —2 147 483 648, el valor máximo es de 2 147 483 647.

float: presenta el número de punto flotante. Float se usa para representar números reales y se escribe con un punto decimal que divide las partes entera y fraccionaria.

Los valores flotantes de Python se representan como valores de doble precisión de 64 bits. El valor máximo que puede tener cualquier número de coma flotante es aproximadamente 1.8 x 10308. Cualquier número mayor que este será indicado por la cadena **inf** en Python.

32 BITS v.s 64 BITS

La cantidad de bits (32 ó 64) se refiere al tipo de unidad central de proceso o CPU, al sistema operativo, los drivers y el software. Todos ellos utilizan una misma arquitectura para de esta forma actuar en la misma sintonía.

	32 BITS	64 BITS
Software	x86	x64/ x86-64 o AMD64
Almacenamiento	32 bits	64 bits
RAM Máxima Gestionable	4 GB	16 Exabytes
Retrocompatibilidad	No	Si
Temporalidad	Antiguos	Actuales

Overflow y Underflow

Son el resultado de una carencia de espacio de representación. Sucede cuando asignamos un valor que está fuera del rango del tipo de datos declarado de la variable.

Puede observarse más fácilmente en tipos de datos como los números enteros (int) y puntos flotantes (float).

Existe un salto entre la matemática teórica y el análisis numérico. El número almacenado en una computadora existe de manera discreta. Cuando hacemos un cálculo que da como resultado un dígito adicional, no podemos agregarlo a nuestro resultado, por lo que obtenemos un Overflow o Underflow. Simplemente no podemos representarlo porque no tengo más dígitos disponibles.

Overflow

Indica que hemos realizado un cálculo que resultó en un número mayor que el número más grande que podemos representar.

Supongamos que tenemos un número entero almacenado en 1 byte. El mayor número que podemos almacenar en un byte es 255. Esto es 11111111 en binario. Si le sumamos 2 (00000010 en binario) el resultado es 257, que es 100000001. El resultado tiene 9 bits, mientras que la representación posible llega a 8. La computadora descartará el bit más significativo (MSB) y se quedará con el resto, lo que es igual a $r\%2^n$ donde r= resultado, n=número de bits disponible y % es el operador módulo (lo veremos más adelante).

Underflow

Indica que hemos realizado un cálculo que resultó en un número menor que el número más bajo que podemos representar. La convención del punto flotante presenta técnicas para representar números fraccionarios.

Si intentamos asignar el valor 10^{-1000} (que es muy cercano a 0) a una variable de tipo float habrá un underflow. En muchas ocasiones es posible continuar el cálculo reemplazando el resultado por cero.

Un poco sobre lógica

Una **proposición** es un enunciado que tiene valor de verdad. Puede ser verdadero o falso, pero no ambos.

Proposiciones simples: son oraciones declarativas. Estas suelen representarse con las letras minúsculas p, q, r, s, t,..., que se denominan variables proposicionales.
Ej: Los gatos son animales.

Proposiciones compuestas: son proposiciones simples enlazadas con conectores como “y”, “o”, “si... entonces...”, “a menos que”, “pero”, “ni... ni...”, “... si y sólo si...”
Ej: Los perros y los gatos son animales.

Conjunción

La conjunción es una proposición compuesta, en donde se utiliza el conector “y”. Se simboliza: p∧q

Ejemplo: p= Los peces nadan. q= Los peces respiran por branquias. Haciendo conjunción (p∧q) queda: L*os peces nadan y respiran por branquias*.

Tabla de verdad de la conjunción:

p	***q***	***p∧q***
V	V	V
V	F	F
F	V	F
F	F	F

Disyunción

La disyunción es una proposición compuesta, en donde se utiliza el conector “o”. Se simboliza: $p \vee q$

Ejemplo: p= Pedro come pan. q= Pedro es alto; haciendo disyunción ($p \vee q$) queda: *Pedro come pan o es alto*.

Tabla de verdad de la disyunción:

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F

Disyunción excluyente

Es la misma disyunción solo que, será verdadera si, y solo si lo es una de las dos proposiciones es verdadera, pero no ambas. Se simboliza: $p \veebar q$

Ejemplo: p= Pedro viajará a Buenos Aires q= Pedro viajará a Rosario. Aplicando disyunción ($p \veebar q$) queda: *Pedro viajará a Buenos Aires o a Rosario*.

Tabla de verdad disyunción excluyente

p	q	$p \veebar q$
V	V	F
V	F	V
F	V	V
F	F	F

Negación

Es el valor con el cual se niega la proposición, es decir, con el valor contrario. Se simboliza: ¬p

Ejemplo: p= Pedro come verduras. Con la negación (¬) queda: *Pedro no come verduras*.

Tabla de verdad de la negación

p	***¬p***
V	F
F	V

Condicional

El condicional es una proposición compuesta, en donde la primera premisa depende de la segunda. p implica q, Es decir si p entonces q. Se simboliza: p→q

Ejemplo: p= Pedro aprobará la materia q= Pedro estudia. Aplicando el condicional (q→p) queda: *Pedro aprobará la materia si estudia*; *Si Pedro estudia entonces aprobará la materia*.

Tabla de verdad del condicional

p	q	$p \rightarrow q$
V	V	V
V	F	F
F	V	V
F	F	V

Bicondicional

El bicondicional es una proposición compuesta, en donde las dos premisas dependen de ambas. Se simboliza: p↔q

Ejemplo: p= un polígono tiene cuatro lados q= un cuadrilátero tiene cuatro lados. Aplicando bicondicional (p↔q) queda: *Un polígono es un cuadrilátero sí y sólo si tiene cuatro lados*.

Tabla de verdad del bicondicional

p	***q***	***p↔q***
V	V	V
V	F	F
F	V	F
F	F	V

Operadores Lógicos

Los operadores son tokens que forman parte de las expresiones y definen la operación (de ahí el nombre) que se deben ejecutar sobre los datos.

Los operadores lógicos aplican parte del álgebra de boole a operandos de tipo lógico o booleano.

and: opera sobre dos valores lógicos obteniendo como resultado verdadero sólo en el caso de que ambas expresiones tengan el valor verdadero.

or: opera sobre dos valores lógicos obteniendo como resultado falso sólo en el caso de que ambas expresiones tengan el valor falso.

not: obtiene el complemento de una expresión lógica, si la expresión es True el resultado será False y viceversa. Este operador es de tipo *monario*: operadores que actúan sobre un solo operando para producir un nuevo valor.

Otras denominaciones para operadores lógicos

Operador	**Python**	**Algunas librerías**	**Otros lenguajes (C)**	**Significado**
and	a **and** b	**&**	a **&&** b	Verdadero si todos los operandos son verdaderos.
or	a **or** b	**|**	a **||** b	Verdadero si alguno de los operandos es verdadero.
not	**not** a	**!**	**!**a	Verdadero si el operando es igual a 0 o falso.

Ejemplo

- El gato es gris **y** peludo -- True
- El gato es blanco **y** peludo -- False

- El gato es blanco **o** peludo -- True
- El gato es blanco **o** negro -- False

- El gato **no** es blanco -- True
- El gato **no** es gris -- False

Ejercicio 1

a = 5

b = 10

c = 2

d = 10

(c < a) and (d = b) = ¿T o F?

Resolución ejercicio 1

a = 5

b = 10

c = 2

d = 10

(c < a) and (d = b) = ¿T o F?

T and T = T

Ejercicio 2

a = 5

b = 10

c = 2

d = 10

not(b = d) or (b < c) = ¿T o F?

Resolución ejercicio 2

a = 5

b = 10

c = 2

d = 10

not(b = d) or (b < c) = ¿T o F?

not T or F = ¿T o F?

F or F = F

Ejercicio 3

a = 5

b = 10

c = 2

d = 10

((a > c) and (a < b)) and (not (b = c) or (d < c)) = ¿T o F?

Resolución ejercicio 3

a = 5

b = 10

c = 2

d = 10

((a > c) and (a < b)) and (not (b = c) or (d < c)) = ¿T o F?

(T and T) and (not F or F) = ¿T o F?

T and (T or F) = ¿T o F?

T and T = T