

## **Operaciones Fundamentales en Binario**

### **Base 2 y su importancia en el procesamiento de datos en computadoras.**

La base 2, conocida comúnmente como sistema binario, es una forma válida de contar utilizando únicamente dos números (el 0 y el 1). Aunque parezca de lo más sencillo, en realidad es un sistema muy potente. Cada número se obtiene a partir de esos dos dígitos y cada posición corresponde a una potencia de 2.

Aunque hoy lo utilicemos sobre todo en los ordenadores, tiene unas raíces muy antiguas. Los egipcios ya utilizaban un sistema de multiplicación mediante duplicaciones que seguía el principio básico del sistema binario: utilizar sumas de potencias de 2. Luego, en torno al siglo XVII, un filósofo llamado Leibniz se dio cuenta de que se podían representar todos los números utilizando solo unos y ceros. Le pareció tan elegante que incluso lo vinculó a pensamientos filosóficos sobre el todo y la nada[1].

Constituye la base del procesamiento de datos en las computadoras, ya que permite representar y manipular la información de forma eficiente y óptima. Este sistema se fundamenta en la lógica binaria, que a su vez se basa en el álgebra de Boole. Posibilita que los dispositivos electrónicos (desde microchips hasta supercomputadoras) ejecuten instrucciones y realicen operaciones lógicas y aritméticas con gran velocidad y precisión. Por ello, resulta impensable imaginar el desarrollo del procesamiento de datos sin recurrir al sistema binario.

Además, la lógica binaria es esencial en áreas como las telecomunicaciones, la criptografía, la transmisión eficiente de información, el diseño de algoritmos de cifrado, los circuitos digitales y la ejecución de instrucciones complejas. Incluso tecnologías emergentes como la computación cuántica dependen de esta forma de codificación, lo que evidencia que su relevancia no solo se mantiene, sino que continúa creciendo[2].

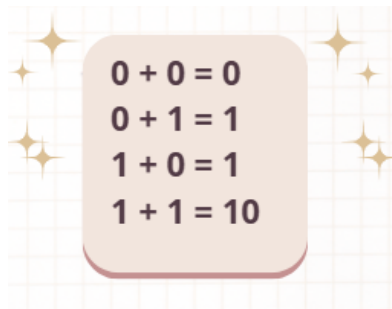
### **Operaciones aritméticas básicas**

#### **Suma**

La suma binaria constituye una de las operaciones básicas de la aritmética digital. Según el libro *Cryptographic Arithmetic: Algorithms and Hardware Architectures*, se consideran dos bits que se deben sumar, junto con un bit de acarreo proveniente de la suma en la posición anterior. El resultado produce un bit de suma y un bit de acarreo, el cual debe ser transferido a la siguiente posición. Para realizar la suma de números que ocupan más de un bit, es necesario interconectar full adders.

Existen varios tipos de sumadores:

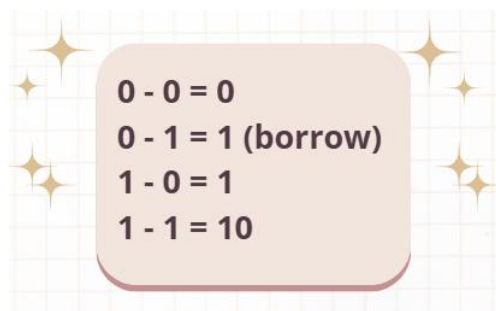
- Sumadores en serie, que suman un bit a la vez
- Sumadores de propagación de acarreo (carry-ripple)
- Sumadores de prefijo paralelo (parallel-prefix), que conectan los full adders de forma que la propagación de los acarreos se realiza simultáneamente, gracias a su naturaleza paralela[3].


$$\begin{array}{l} 0 + 0 = 0 \\ 0 + 1 = 1 \\ 1 + 0 = 1 \\ 1 + 1 = 10 \end{array}$$

## Resta

Por norma general, la resta binaria suele llevarse a cabo utilizando el complemento a dos del número a restar. El mismo libro señala que la ventaja de este método es que permite reutilizar el circuito de suma para que la unidad aritmética pueda realizar también restas, en lugar de diseñar un circuito cuya única finalidad sea ejecutar esta operación.

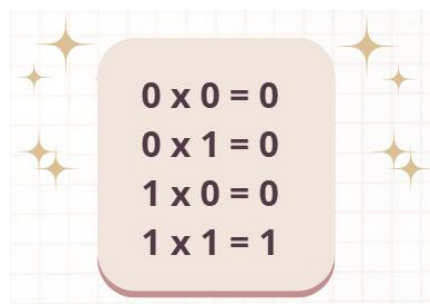
El complemento a dos se obtiene invirtiendo todos los bits del sustraendo y sumando uno. Esta nueva representación se puede sumar al minuendo, y el circuito es capaz de gestionar automáticamente los préstamos (*borrows*) en los casos en que la resta se haría de forma convencional. Por lo tanto, la implementación de la resta resulta práctica y eficiente[3].


$$\begin{array}{l} 0 - 0 = 0 \\ 0 - 1 = 1 \text{ (borrow)} \\ 1 - 0 = 1 \\ 1 - 1 = 10 \end{array}$$

## Multiplicación

La multiplicación binaria se puede definir como la suma de productos parciales. Se trata de observar cada uno de los bits del multiplicador: si un bit vale uno, el multiplicando se suma desplazado; si un bit vale cero, se suma cero.

Los productos parciales se van sumando y, durante el proceso, es importante manejar correctamente los acarreos que se producen al sumar. Para mejorar el rendimiento de la operación, se pueden emplear multiplicadores de radix alto para procesar varios bits a la vez, o bien estructuras de tipo paralelo que permiten realizar la multiplicación de forma más rápida[3].


$$\begin{array}{l} 0 \times 0 = 0 \\ 0 \times 1 = 0 \\ 1 \times 0 = 0 \\ 1 \times 1 = 1 \end{array}$$

## **División**

La división binaria se lleva a cabo de forma análoga a la división larga en decimal. Según el libro, esta se realiza mediante sucesivas restas del divisor a un residuo parcial. Durante este procedimiento, en cada paso se determina si el divisor puede ser restado (1) o no (0).

Existen 2 métodos fundamentales:

- La división restauradora, en la que se realizan recuperaciones tan pronto como el resultado de la resta es negativo.
- La división no restauradora, en la que el tratamiento se deja hasta el final del cálculo.

Cabe destacar que, en ambos casos, el control de los borrows es esencial para gestionar el residuo y obtener el cociente correcto[3].

## Ejemplos Prácticos:

# OPERACIONES BINARIAS

## SUMA

$$\begin{array}{r} 11100010 \\ + 10011101 \\ \hline \end{array}$$

CARRY → 10111111

$$\begin{array}{r} 01011011 \\ + 00100101 \\ \hline \end{array}$$

10000000

## RESTA

BORROW

$$\begin{array}{r} 10101000 \\ - 00110100 \\ \hline \end{array}$$

01110100

$$\begin{array}{r} 10000000 \\ - 00011110 \\ \hline \end{array}$$

01100010

## MULTIPLICACIÓN

00010110  
× 1101

00010110  
00000000  
+ 0001011000  
0001011000  
000100011110

00001111  
× 1010

00000000  
000011110  
+ 0000000000  
00001111000  
00010010110

## DIVISIÓN

11110000 | 1010  
01010  
00000  
00000  
00000  
0000

11010011 | 1011  
00100  
01000  
10001  
01101  
0010

## **Bibliografía**

- [1] M. Moyon, "Binary Numeration: From Ancient Egypt to a 19th Century French Mathematical Recreation," *REMATEC*, vol. 19, no. 47, Mar. 2024, doi: 10.37084/REMATEC.1980-3141.2024.n47.e2024005.id607.
- [2] H. Li, "Binary System: Foundation of Modern Computing," vol. 14, no. 1, Mar. 2024.
- [3] A. R. Omondi, *Cryptography Arithmetic*, 1st ed., vol. 77. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-34142-8.