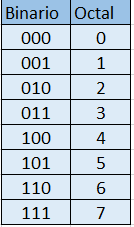
GENESIS GUTIERREZ

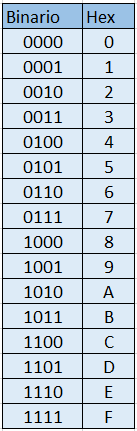
**OPERACIONES EN OCTAL Y HEXADECIMAL**

En el sistema octal, cada número está compuesto por dígitos que van del 0 al 7, y cada posición representa una potencia de 8. Esta forma de numeración simplifica la representación de valores binarios, ya que tres bits binarios corresponden exactamente a un dígito octal. De manera similar, el sistema hexadecimal utiliza dígitos del 0 al 9 y letras de la A a la F, donde cada posición es una potencia de 16, y cada dígito hexadecimal equivale a cuatro bits binarios [1].

Tanto el sistema octal como el hexadecimal se usan porque permiten representar números binarios largos de forma más compacta y legible, lo que es útil en programación y hardware.



*Sistema Octal*



*Sistema Hexadecimal*

**UTILIDAD DEL SISTEMA HEXADECIMAL EN PROGRAMACIÓN**

El sistema hexadecimal se utiliza frecuentemente para representar instrucciones en lenguaje ensamblador. Gracias a que cada instrucción máquina está formada por múltiples bits, escribirlas en binario sería muy largo y poco práctico. En cambio, expresarlas en hexadecimal permite comprimir esa información y trabajar de forma más eficiente en entornos de bajo nivel [2].

**HEXADECIMAL Y LA RELACIÓN CON LOS BYTES**

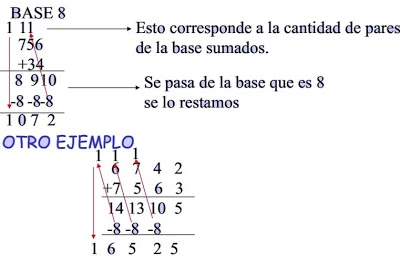
Una ventaja adicional del sistema hexadecimal es su alineación natural con los bytes, ya que un byte (8 bits) puede representarse como dos dígitos hexadecimales. Esto hace que el trabajo con direcciones de memoria y valores almacenados en registros sea más intuitivo. Por eso, muchos lenguajes y depuradores muestran valores de memoria en hexadecimal: es una forma directa de ver el contenido real de un byte [2].

Entender que “A1” equivale a un byte específico ayuda a interpretar qué instrucciones o datos están siendo manejados en una dirección concreta de memoria.

**OPERACIONES BÁSICAS EN OCTAL Y HEXADECIMAL**

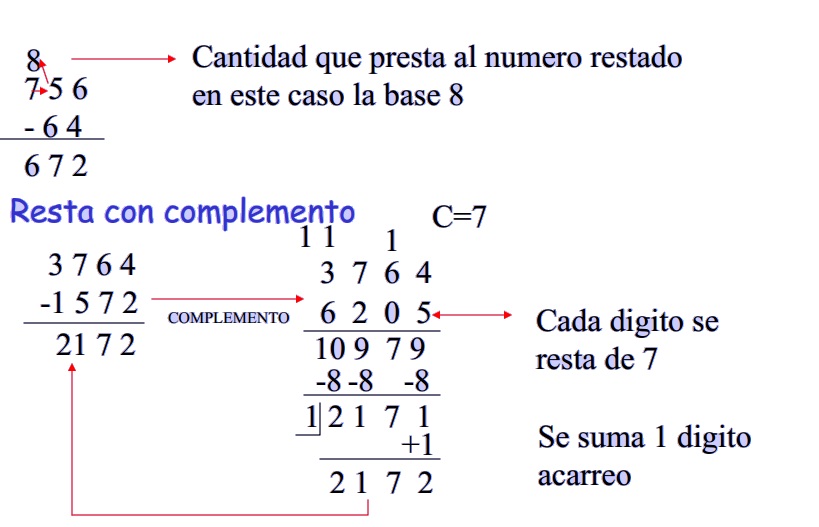
En operaciones aritméticas, tanto en octal como en hexadecimal, las reglas básicas se adaptan al valor de su base. Por ejemplo, en octal, si al sumar dos dígitos se pasa de 7, se lleva una unidad a la siguiente posición. En hexadecimal ocurre lo mismo si se supera la F (que representa el 15 decimal) [1].

En la figura (#) se muestra un ejemplo de suma octal. Aquí se observa cómo al sumar dos dígitos que dan un resultado mayor a 7, se lleva 1 y se ajusta el resultado a base 8.



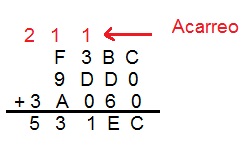
*Suma Octal*

En la figura (#) se muestra un ejemplo de resta octal. Se puede ver que cuando no se puede restar directamente, se pide prestado del dígito de la izquierda, teniendo en cuenta la base octal.



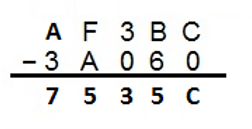
*Resta Octal*

En la figura (#) se muestra un ejemplo de suma hexadecimal. Se puede ver que cuando el resultado supera F (15), se genera un acarreo, y se ajusta el valor al sistema hexadecimal.



*Suma Hexadecimal*

En la figura (#) se muestra un ejemplo de resta hexadecimal. En este caso, la operación se puede realizar directamente sin necesidad de pedir prestado, simplemente restando los valores correspondientes en hexadecimal.



*Resta Hexadecimal*

Además de facilitar la representación binaria, los sistemas octal y hexadecimal permiten transformar cifras complejas a otras bases de forma directa y estructurada. Por ejemplo, al convertir de hexadecimal a binario, se toma cada dígito y se traduce directamente en un grupo de cuatro bits, mientras que en el caso del octal se usa un grupo de tres bits.

**APLICACIÓN PRÁCTICA EN ARQUITECTURA DE COMPUTADORES**

En la arquitectura de computadores, el uso de números en base 8 y 16 es esencial para la representación de instrucciones máquina, direcciones de memoria y manipulación de registros. Al utilizar sistemas como el hexadecimal, se puede comprender más fácilmente el comportamiento del procesador y los datos almacenados. Este sistema actúa como una especie de “puente” entre el binario que entiende la máquina y el ensamblador que programa el humano [2].

**BIBLIOGRAFIA:**

[1] A. C. Jha, “Positional Number System,” *NUTA Journal*, vol. 7, no. 1–2, pp. 1–9, Dec. 2020, doi: 10.3126/nutaj.v7i1-2.39924.

[2] S. B. Mir *et al.*, “Introducción a la arquitectura de computadores con QtARMSim y Arduino,” 2019. Accessed: Jun. 07, 2025. [Online]. Available: https://dspace.itsjapon.edu.ec/jspui/bitstream/123456789/435/1/Introduccion-ARM-Arduino.pdf