Operaciones en Sistemas Octal y Hexadecimal y Códigos de Corrección de Errores en Computadoras

1.- Introducción

La representación, almacenamiento y transmisión confiable de datos son aspectos esenciales en los sistemas digitales modernos. Para ello, se utilizan diversos sistemas numéricos y esquemas de corrección de errores (ECC) que permiten manipular eficientemente los datos binarios y mitigar errores debidos a fallas del hardware o interferencias externas. Esta investigación aborda dos pilares técnicos, operaciones en sistemas numéricos octal y hexadecimal, y estructuras y arquitecturas de códigos de corrección de errores, especialmente en memorias y almacenamiento digital [1]-[5].

2.- Operaciones en Octal y Hexadecimal

2.1.- Fundamentos

El sistema **octal** (base 8) considera los dígitos del 0 a 7 y el **hexadecimal** (base 16) usa los dígitos del 0 al 9 y las letras de la A hasta la F, que representan los valores 10 a 15 respectivamente. Estos sistemas permiten una representación compacta de los datos binarios y son habituales en la depuración, el direccionamiento de memoria, los microcontroladores y la programación de sistemas [2].

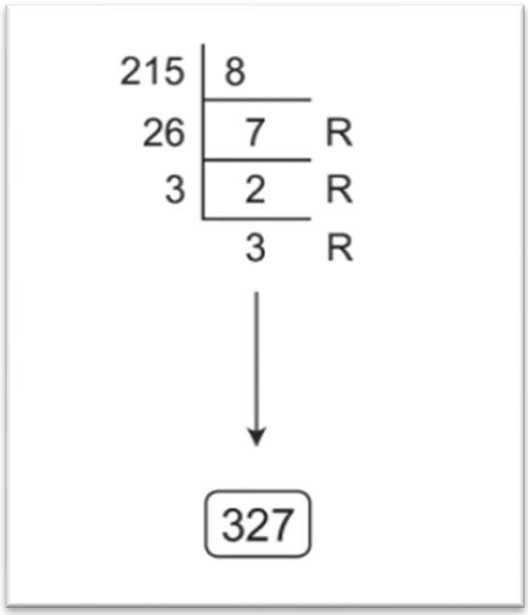
2.2.- Conversión entre bases

Decimal a octal o hexadecimal: Se aplica división sucesiva por la base correspondiente, registrando los residuos de abajo hacia arriba.

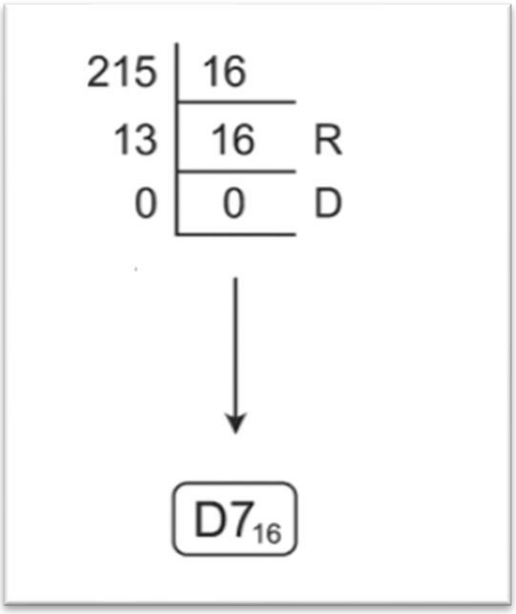
Entre binario y octal/hexadecimal: Se agrupan bits en grupos de tres (base 8) o cuatro (base 16) [2].

Por ejemplo:

Decimal 215 → Octal: 327_8 → 215 ÷ 8 → residuos = 3,2,7



Decimal 215 → Hexadecimal: $D7_{16}$ → residuos = 13 (D), 7



2.3.- Representación digital en hardware

Los datos en las computadoras se almacenan de forma interna como combinaciones de bits, ya sea en registros o en memorias. La representación numérica en sistemas octales y hexadecimales favorece la interacción de los seres humanos con sistemas de bajo nivel. Aun así, los diagramas lógicos también dejan entrever cómo como las unidades aritmético-lógicas (ALU) de procesan tales valores [2].

3.- Códigos de corrección de errores

3.1.- Justificación técnica

Los ECC permiten detectar y corregir errores sin recurrir a retransmisión, aumentando la confiabilidad en memorias volátiles (SRAM, DRAM) y no volátiles (NAND, NOR). La elección del esquema ECC depende del equilibrio entre tasa de corrección, complejidad del decodificador y eficiencia energética [3]-[5].

3.2.- Tipos de ECC abordados

3.2.1.- Códigos superpuestos 2D

Plantean una técnica de corrección de errores mediante códigos de corrección de errores de tipo bipartito que se implementa superponiendo códigos 2D mediante la aplicación de bits de paridad a filas y columnas de una matriz de datos. Esta técnica mejora la cobertura de errores de tipo múltiple en memorias SRAM, sin necesidad de duplicar el coste en términos de la ejecución. Se presentan las métricas de latencia y overhead [3].

3.2.2.- Códigos Polar con decodificación avanzada

Proponen mejorar la decodificación de códigos polar sobre canales ISI bidimensionales, utilizando un decodificador SSC. Se introduce un posprocesamiento basado en perturbación aleatoria y algoritmos genéticos, que permite recuperar mensajes válidos, incluso cuando los esquemas tradicionales fallan [1].

3.2.3.- Códigos BCH optimizados para memorias Flash

Muestran una arquitectura optimizada para la decodificación BCH en los tipos de memoria Flash NOR y NAND. Se hace uso del paralelismo, de la técnica de *pipeline* y de multiplicadores de campo finito con compartición XOR, con el fin de que la latencia y la complejidad hardware sean reducidas al mínimo [4].

3.2.4.- Arquitecturas emergentes para ECC

Un análisis más reciente [5] profundiza en la comparación entre distintas arquitecturas ECC (Hamming, BCH, LDPC, Polar), evaluando el *compromiso* entre área, latencia y consumo energético. Se destaca la importancia de seleccionar arquitecturas adaptativas según el contexto (p. ej., IoT vs servidores).

4.- Comparativa de esquemas ECC

Código ECC	Aplicación	Ventajas principales	Limitaciones técnicas
Superpuestos 2D	*SRAM y matrices de datos*	*Corrección cruzada en 2D con baja redundancia*	*Complejidad estructural creciente*
Polar + SSC/GA	*Almacenamiento 2D*	*Decodificación eficiente incluso bajo ISI*	*Alta dependencia de ajustes finos*
BCH optimizado	*Flash NAND/NOR*	*Alta capacidad de corrección, bajo retardo*	*Diseño complejo del decodificador*
Hamming clásico	*DRAM, registros simples*	*Eficiencia en hardware y simplicidad*	*Solo detecta/corrige errores simples*

5.- Conclusiones

El estudio de los sistemas numéricos octal y hexadecimal brinda herramientas básicas para entender el funcionamiento interno de los computadores, mediante su uso para la representación en binario y el direccionamiento. Mientras tanto, la evolución en la corrección de errores refleja la evolución de arquitecturas adaptativas basadas en la precisión, la velocidad y la eficiencia energética aplicable a la tecnología de almacenamiento que se utilice. Desde los códigos clásicos hasta los híbridos como los que se han visto, Polar y BCH optimizados, la eficiencia computacional depende, de forma crítica, de estos métodos [1]- [5].

6.- Bibliografía

[1] N. K. Gerrar, S. Zhao, and L. Kong, “Error correction in data storage systems using polar codes,” *IET Communications*, vol. 15, no. 14, pp. 1859–1868, 2021, doi: 10.1049/cmu2.12197.

[2] M. Shabnam, S. Mahat, and M. K. Patil, “Number System for Digital Computers,” *International Journal of Scientific & Technology Research (IJSTR)*, vol. 9, no. 4, 2020, [Online]. Available: www.ijstr.org

[3] A. R. Fritsch and P. Alegre, “ESCOLA POLITÉCNICA PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO MESTRADO EM CIÊNCIA DA COMPUTAÇÃO OVERLAPPING ERROR CORRECTION CODES ON TWO-DIMENSIONAL STRUCTURES,” 2025.

[4] S. Nabipour and J. Javidan, “Enhancing Data Storage Reliability and Error Correction in Multilevel NOR and NAND Flash Memories through Optimal Design of BCH Codes,” 2023.

[5] L. Parrini *et al.*, “Error Detection and Correction Codes for Safe In-Memory Computations,” 2024.