Compás
capacitación e investigación

2025

Matemáticas Discretas:
La Base Lógica y Teórica de la Computación

Edison Loján Cueva
Maritza Alexandra Pinta

Matemáticas Discretas:
La Base Lógica y Teórica de la Computación

Edison Loján Cueva
Maritza Alexandra Pinta

ISBN: 978-9942-53-024-0 DOI: http://doi.org/10.48190/9789942530240

Compás
capacitación e investigación

Primera edición, 7/11/2025

ISBN: 978-9942-53-024-0
DOI: http://doi.org/10.48190/9789942530240
Distribución online
Acceso abierto

Cita
Loján, E., Pinta, M. (2025) Matemáticas Discretas: La Base Lógica y Teórica de la Computación. Editorial Grupo Compás

Este libro es parte de la colección de la Univesidad Técnica de Machala y ha sido debidamente examinado y valorado en la modalidad doble par ciego con fin de garantizar la calidad de la publicación.

DEDICATORIA

A Dios, fuente inagotable de sabiduría, por ser guía y fortaleza en cada paso del camino.

A mi familia, pilar fundamental de mi vida, por su amor, paciencia y constante apoyo, especialmente en las largas jornadas de estudio, investigación y enseñanza. Su presencia ha sido inspiración en cada uno de mis logros.

A mis estudiantes, quienes con sus preguntas, desafíos y entusiasmo renovaron mi compromiso con la enseñanza. A ellos dedico cada página de este libro, con la esperanza de que encuentren en estas líneas las herramientas que les permitan comprender y transformar el lenguaje profundo de la computación.

Y a todos quienes creen que las matemáticas no son solo números, sino un lenguaje lógico que estructura el pensamiento y moldea el futuro.

INTRODUCCIÓN

En el corazón de toda innovación tecnológica, desde las redes sociales que conectan a miles de millones de personas hasta los sistemas de inteligencia artificial más avanzados, reside un lenguaje silencioso pero fundamental: el de las matemáticas discretas. A menudo pasan desapercibidas, pero son los cimientos sobre los que se construyen los algoritmos de recomendación de Netflix y Spotify, la seguridad criptográfica de blockchain y las bases de datos que impulsan el comercio electrónico. Cada clic, cada transacción digital y cada interacción en línea son posibles gracias a los principios discretos de la lógica, la teoría de gráficos y la combinatoria.

Matemáticas Discretas: La Base Lógica y Teórica de la Computación se propone revelar este lenguaje oculto. A lo largo de estas páginas, exploraremos los conceptos esenciales que cimentan la computación moderna, incluyendo los fundamentos de la lógica, la aritmética del computador y la teoría de conjuntos. A diferencia de las matemáticas continuas, que se enfocan en lo que es infinitamente divisible, las matemáticas discretas estudian estructuras finitas y contables, lo que las hace perfectas para el mundo digital.

Este libro está diseñado para estudiantes y profesionales que deseen fortalecer su comprensión teórica para aplicarla en campos de alta demanda como el desarrollo de software, la ciencia de datos, la ciberseguridad y la inteligencia artificial. No solo aprenderás definiciones y procedimientos, sino que también desarrollarás un pensamiento "discreto" para resolver problemas complejos y comprender la estructura que sostiene el mundo digital.

Te invitamos a recorrer este camino de descubrimiento para desentrañar los secretos detrás de la computación y, en el proceso, dominar uno de los lenguajes más poderosos del conocimiento humano en la era actual.

OBJETIVOS GENERALES

Proporcionar a estudiantes universitarios, docentes y profesionales del área de Tecnologías de la Información una comprensión sólida, estructurada y aplicada de los principios fundamentales de las Matemáticas Discretas, abordando temas esenciales como los fundamentos teóricos, la aritmética del computador, la lógica proposicional y la teoría de conjuntos, con el fin de desarrollar habilidades analíticas y de razonamiento lógico que permitan modelar, resolver problemas computacionales y comprender la estructura matemática que subyace en los sistemas digitales y algoritmos informáticos.

ESTRUCTURA GENERAL DEL LIBRO

A continuación, se presenta un resumen de lo que se tratará en cada capítulo del libro:

Capítulo I - Fundamentos de Matemáticas Discretas

Este libro explora de manera clara y progresiva los fundamentos de los sistemas de numeración, esenciales en el campo de la computación. Comienza con una introducción conceptual e histórica, seguida por la definición y características de los distintos sistemas numéricos, destacando su base, valor posicional y notación. Se abordan las conversiones entre sistemas decimal, binario, octal y hexadecimal, además de las operaciones aritméticas en binario, claves para la lógica computacional. También se analiza el uso de complementos decimales y binarios como herramientas de optimización en cálculos digitales. En conjunto, este capítulo proporciona una base sólida para comprender cómo las matemáticas discretas sustentan el lenguaje operativo de las tecnologías digitales modernas.

Capítulo II - Aritmética del Computador

El capítulo "Aritmética del Computador" ofrece una visión detallada de los principios matemáticos fundamentales que sustentan el tratamiento numérico en sistemas computacionales. Inicia con una revisión de conceptos matemáticos básicos, distinguiendo entre números exactos y aproximados, para luego abordar la importancia de los dígitos significativos como medida de precisión en cálculos digitales. Se estudian técnicas de redondeo y truncamiento, así como el uso del valor absoluto en contextos computacionales. La unidad también introduce las formas exponenciales y su aplicación en la representación eficiente de números, preparando el terreno para el análisis de cómo las

computadoras representan internamente tanto números enteros como reales. A través de estos temas, se proporciona al lector una comprensión integral de cómo los datos numéricos son procesados, almacenados y utilizados dentro de los sistemas informáticos.

Capítulo III - Lógica y Tablas de Verdad

El capítulo **"Lógica y Tablas de Verdad"** del libro *Matemáticas Discretas: El Lenguaje Secreto de la Computación* aborda los fundamentos esenciales del razonamiento lógico aplicado al campo de la informática y la matemática, comenzando con los principios de la lógica matemática y su relación con la estructuración del pensamiento formal. Se introduce el concepto de proposiciones, el uso de operadores lógicos (como la conjunción, disyunción, negación, implicación y doble implicación), y se enseña a construir y analizar tablas de verdad. Además, se profundiza en la jerarquía de los operadores, la identificación de tautologías, contradicciones y contingencias, así como en el estudio de la equivalencia lógica y el álgebra de proposiciones. Todo esto se complementa con ejercicios propuestos que permiten aplicar los conceptos aprendidos, desarrollando habilidades críticas para el análisis y diseño de algoritmos, estructuras de control y validación de sistemas lógicos en el ámbito de las tecnologías de la información.

Capítulo IV - Teoría de Conjuntos

El capítulo "Teoría de Conjuntos" del libro Matemáticas Discretas: El Lenguaje Secreto de la Computación introduce los conceptos fundamentales que rigen la construcción y manipulación de conjuntos, los cuales son pilares esenciales en la lógica matemática y la programación. Se abordan las definiciones básicas, las distintas formas de determinar un conjunto (por extensión y comprensión), así como su clasificación en conjuntos finitos, infinitos, vacíos y universales. Además, se desarrollan las operaciones clásicas entre conjuntos —como la unión, intersección, diferencia y complemento— junto con las leyes del álgebra de conjuntos que rigen dichas operaciones. También se incluye el principio de dualidad, que permite establecer simetrías útiles en la resolución de problemas. Finalmente, se explora el principio de conteo y su aplicación en conjuntos finitos, sentando las bases para el análisis combinatorio y la optimización en el desarrollo de algoritmos y estructuras de datos.

ESTRATEGIAS METODOLOGICAS

Para garantizar una comprensión efectiva y profunda de los contenidos de este libro, se emplearán diversas estrategias metodológicas fundamentadas en el enfoque constructivista, el aprendizaje activo y la aplicación contextual de los conceptos. A continuación, se describen las estrategias principales utilizadas en cada capítulo:

1. **Aprendizaje basado en problemas (ABP):**

   Cada capítulo iniciará con una situación problema relacionada con entornos reales de la informática, como el diseño de algoritmos, estructuras de datos o sistemas de codificación. Estas situaciones permitirán contextualizar los conceptos teóricos y promover el desarrollo del pensamiento lógico-matemático.

2. **Uso de representaciones gráficas y visuales:**

   Se emplearán diagramas de Venn, tablas de verdad, circuitos lógicos, gráficos y líneas de tiempo que faciliten la comprensión visual de los contenidos, especialmente en los capítulos de *Teoría de Conjuntos* y *Lógica y Tablas de Verdad*.

3. **Ejercicios resueltos y propuestos con retroalimentación guiada:**

   Cada sección incluye ejemplos detalladamente resueltos y actividades complementarias con distintos niveles de dificultad, lo que permite reforzar el aprendizaje autónomo y autoevaluar el progreso.

4. **Simulación y herramientas digitales:**

   Se incorporarán herramientas interactivas como calculadoras binarias, simuladores de lógica digital y software matemático (por ejemplo, GeoGebra o SageMath) para representar la *Aritmética del Computador* y explorar visualmente el funcionamiento de los sistemas numéricos.

5. **Conexiones interdisciplinarias:**

   Los temas se vincularán con áreas de la computación como programación, arquitectura de computadoras y estructuras de datos, demostrando cómo los *Fundamentos de Matemáticas Discretas* son clave para el desarrollo de soluciones informáticas eficientes.

6. **Desarrollo de proyectos individuales y colaborativos:**

Se propondrán mini proyectos como la creación de tablas de verdad para algoritmos condicionales, diseño de conjuntos para modelar bases de datos, o implementación de conversiones numéricas mediante pseudocódigo, promoviendo la integración y aplicación práctica de los conocimientos.

7. **Autoevaluación y reflexión crítica:**

   Al final de cada capítulo se incluirán secciones de autoevaluación, rúbricas de comprensión conceptual, preguntas de análisis y síntesis que inviten al estudiante a reflexionar sobre su proceso de aprendizaje y su aplicabilidad en su futura práctica profesional.

Tabla de Contenido

Capítulo I
Fundamentos de Matemáticas Discretas

Introducción

Las matemáticas discretas constituyen el pilar lógico y numérico sobre el cual se construyen los sistemas computacionales modernos. Este capítulo introduce los fundamentos esenciales de los sistemas de numeración, conocimientos imprescindibles para comprender cómo las computadoras representan, manipulan y almacenan información. A lo largo de este recorrido, se explicarán qué son los sistemas de numeración, sus principales características, y cómo se convierten entre distintas bases numéricas, como el sistema decimal, binario, octal y hexadecimal. Así mismo, se abordarán las operaciones binarias y el concepto de complementos, tanto decimales como binarios, fundamentales para realizar cálculos en arquitecturas digitales. También se explorará la codificación para computadoras, enfatizando cómo los datos son representados mediante bits y bytes. Este capítulo no solo proporcionará una base teórica sólida, sino que también facilitará el desarrollo de habilidades prácticas que serán clave en la resolución de problemas computacionales, fortaleciendo el pensamiento lógico y abstracto de los estudiantes de tecnologías de la información.

Objetivos de la unidad

- Asimilar los principios esenciales de los sistemas de numeración aplicados a la computación, incluyendo las bases: binaria, decimal, octal y hexadecimal.
- Desarrollar destrezas en la conversión y operación entre diferentes sistemas numéricos, empleando métodos de complementos y técnicas de codificación digital para representar información de manera eficiente.
- Aplicar de forma práctica los conocimientos adquiridos en la resolución de problemas relacionados con la transformación y manipulación de datos numéricos, fortaleciendo el razonamiento lógico y el pensamiento computacional.

Preguntas de Enfoque

1. ¿Por qué es fundamental internalizar los sistemas de numeración en el contexto de la computación y las tecnologías de la información?
2. ¿Qué criterios permiten clasificar y comparar distintos sistemas de numeración?
3. ¿Cómo se realiza la conversión entre los sistemas decimal, binario, octal y hexadecimal, y cuál es su importancia en la programación y arquitectura de computadoras?
4. ¿En qué consisten los complementos binarios y decimales y cómo se utilizan para representar números negativos o realizar restas?
5. ¿Cómo se reflejan estos conceptos en aplicaciones reales como la manipulación de datos, almacenamiento en memoria o transmisión de información?

"Los números binarios son el alfabeto del pensamiento digital; entenderlos es hablar el lenguaje de las máquinas."
— George Boole

Fundamentos de Matemáticas Discretas

Aparición de los Números Naturales: El hombre desde tiempos remotos se vio en la necesidad de contabilizar todo lo que producía, los animales que le pertenecían, sus recursos y/o necesidades, etc.; lo cual permitió el descubrimiento paulatinamente de civilización en civilización de lenguajes como el de numeración. A continuación, mencionaremos civilizaciones que contribuyeron al desarrollo de los sistemas de numeración (Jiménez de Parga, 2024)

- **Egipto:** Muy adelantados en cuanto a las demás civilizaciones de la época porque:

  - Construcción de Pirámides
  - Construcción de Templos

  Inventaron el sistema de numeración **pictográfico.**

  1
  10
  100
  1000
  10000
  100000
  1000000

- **Babilonios:**

  1 cuña vertical
  10 cuña horizontal

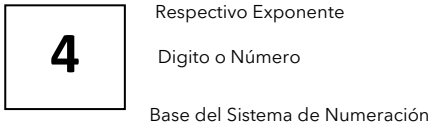
- **Grecia:**
  - **Tenemos a Thales de Mileto. -** Primero en calcular las pirámides de Egipto y la distancia a la luna.
  - **Pitágoras. -** Teorema de Pitágoras.
  - **Euclides. -** Fue el primero de los geómetras; ideas con lo que se basó la geometría (paralelas). Escritos del primer libro de geometría Elementos.
  - **Aristóteles**
  - **Platón**

- **Roma:**
  Sistema de numeración romano.

1	I
5	V
10	X
50	L
100	C
500	D
1000	M

Definiciones Básicas:

Todo número, independiente del sistema de numeración posee las siguientes características:



Base. - La base de un sistema de numeración, hace referencia al número de dígitos que posee dicho sistema de numeración.

Notación o Forma Expandida de un Número. - La Notación o Forma Expandida de un Número nos sirve para obtener el equivalente decimal de un número independientemente del sistema de numeración al que pertenece dicho número.

Sintaxis:

$d_1 * b^1 + d_2 * b^2 + d_3 * b^3 + \ldots\ldots\ldots\ldots\ldots\ldots\ldots + d_n * b^n$

d= dígito b= base del sistema de numeración

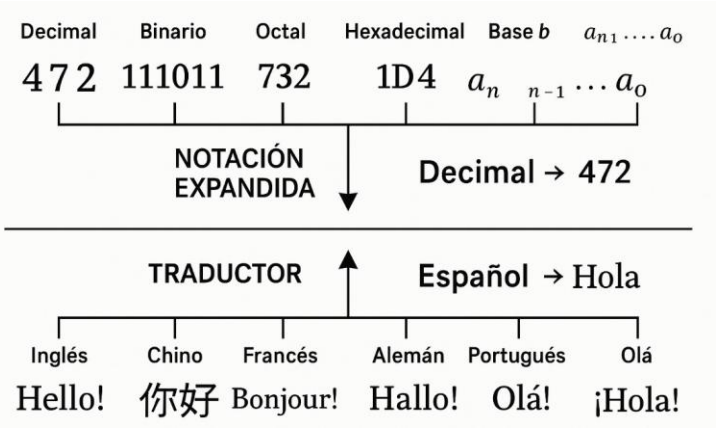


Figura 1: Analogía de la forma expandida de un número con los idiomas.

Fuente: Elaboración propia.

Sistema de Numeración Decimal

Es el sistema con el cual más familiarizados nos encontrados debido a que ha sido la fuente de aprendizaje de nuestros conocimientos matemáticos, geométricos, físicos etc. Tiene diez dígitos denotados por los símbolos.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

La base del sistema de numeración decimal es igual a 10, es el único sistema que puede obviar colocar su base, en otras palabras, si un número carece de base se considera por defecto que este número pertenece al sistema de numeración decimal (García, 2015).

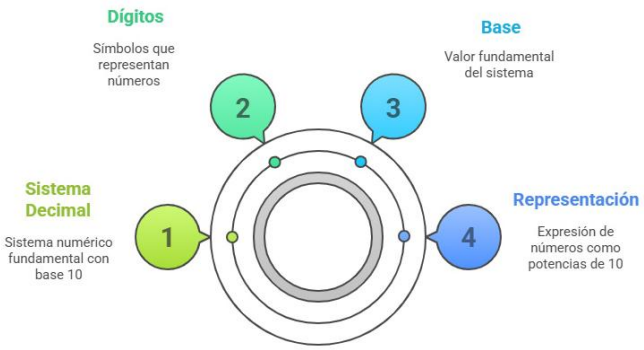


Figura 2: Estructura del sistema decimal.
Fuente: Elaboración propia.

Cualquier entero positivo y cualquier valor fraccionario, representado en el sistema decimal como una cadena de dígitos decimales, puede expresarse también como una suma de potencias de 10, con cada potencia ponderada por un digito. Por ejemplo.

3 2 1 0 -1 -2 -3

$$\mathbf{8253{,}526_{10}} = 8 * 10^3 + 2 * 10^2 + 5 * 10^1 + 3 * 10^0 + 5 * 10^{-1} + 2 * 10^{-2} + 6 * 10^{-3}$$

$$= 8000 + 200 + 50 + 3 + 5/10 + 2/100 + 6/100$$

$$= \mathbf{8253{,}526_{10}}$$

Ejercicios propuestos:

Conversión de los siguientes números decimales a sus equivalentes en números
romanos.

- 1231523432874328
- 89768234632,63536
- 762332645460,4566
- 98765453476234,34

- 4564567792343,568

Realizar la conversión de los siguientes números decimales a través de la forma o notación expandida.

- 23152367874328
- 89768434632,6346
- 762326485460,4546
- 9876644769234,343
- 456458792343,5668

Sistema de Numeración Binario

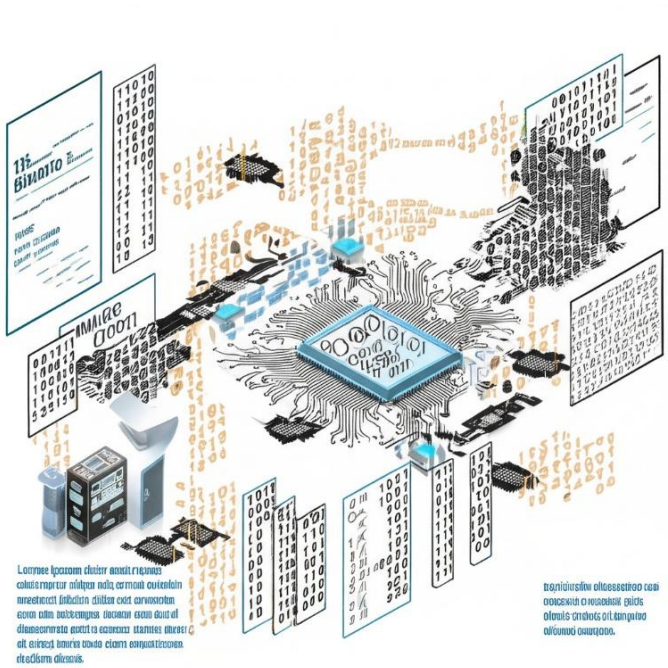


Figura 3: Introducción al sistema binario.
Fuente: Elaboración propia.

En el vasto universo de las matemáticas, existen diversos sistemas para representar cantidades. El sistema decimal, con el que estamos más

familiarizados en nuestra vida cotidiana, utiliza diez símbolos distintos (los dígitos del 0 al 9) como base para construir cualquier número. Sin embargo, en el corazón de la computación y la electrónica digital reside un sistema mucho más simple, pero increíblemente poderoso: el **sistema de numeración binario**.

El sistema de numeración binario es un sistema posicional que utiliza solo dos dígitos: el 0 y el 1, la base del sistema de numeración es igual a 2. Cada dígito binario se conoce como "bit" (binary digit) y representa una potencia de 2, de forma similar a como el sistema decimal utiliza potencias de 10. En este sistema, el valor de cada posición aumenta exponencialmente comenzando con 2^0, 2^1, 2^2 y así sucesivamente. Es el lenguaje base que utilizan las computadoras para representar y procesar datos, ya que los circuitos electrónicos trabajan de manera eficiente con dos estados: encendido (1) y apagado (0).

$$110010_2$$

Conversión binaria a decimal

Recordemos que para obtener el equivalente decimal de un número binario se usa la forma o notación expandida de un número, por lo tanto, cualquier número binario se puede escribir en notación expandida (Lipschutz,1995):

Convertir: $110101_2 \Longrightarrow X_{10}$

$\mathbf{1^5 1^4 0^3 1^2 0^1 1^0 =}$ $1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 53_{10}$

Convertir: $101,1101_2 \Longrightarrow Y_{10}$

$\mathbf{1^2 0^1 1^0, 1^{-1} 1^{-2} 0^{-3} 1^{-4} =}$ $1*2^2 + 0*2^1 + 1*2^0 + 1*2^{-1} + 1*2^{-2} + 0*2^{-3} + 1*2^{-4} =$

$5{,}8125_{10}$

Convertir: 10110110110100011010 1,101$_2$ $\Rightarrow$ Z_{10}

$1^{20}0^{19}1^{18}1^{17}0^{16}1^{15}1^{14}0^{13}1^{12}1^{11}0^{10}1^{9}0^{8}0^{7}0^{6}1^{5}1^{4}0^{3}1^{2}0^{1}1^{0}$,$1^{-1}0^{-2}1^{-3}1^{-4}0^{-5}$ =

$1*2^{20}+0*2^{19}+1*2^{18}+1*2^{17}+0*2^{16}+1*2^{15}+1*2^{14}+0*2^{13}+1*2^{12}+1*2^{11}+0*2^{10}+1*2^{9}+0*$

$2^{8}+0*2^{7}+0*2^{6}+1*2^{5}+1*2^{4}+0*2^{3}+1*2^{2}+0*2^{1}+1*2^{0}+1*2^{-1}+0*2^{-2}+1*2^{-3}+1*2^{-4}+0*2^{-5}=$

$1497653,6875_{10}$

Conversión decimal a binario

Para realizar la conversión de un numero decimal a su equivalente binario, lo primero que debemos identificar en dicho número es si posee parte entera (P_E) y parte fraccionaria (P_F). En dicha representación binaria se debe calcular la parte entera y la parte fraccionaria separadamente (Lipschutz,1995).

$$\mathbf{1497653,6875_{10} \Longrightarrow \quad X_2}$$

a) Para convertir P_E = **1497653** a su equivalente binario, (la operación que rige es la división) dividimos la P_E y cada cociente sucesivo por 2, tomando en cuenta que cuando el cociente es exacto el residuo es 0 y cuando el cociente es inexacto el residuo es 1. Además, cuando el cociente es inexacto la próxima división es solo con su parte entera y la parte fraccionaria ya no se utiliza, la extracción de los dígitos binarios va en sentido ascendente o sea de abajo hacia arriba.

P_E	**1497653**	**2**
	748826,5	1
	374413	0
	187206,5	1
	93603	0
	46801,5	1
	23400,5	1
	11700	0
	5850	0
	2925	0
	1462,5	1
	731	0
	365,5	1
	182,5	1
	91	0
	45,5	1
	22,5	1
	11	0
	5,5	1
	2,5	1
	1	0

$$\mathbf{1497653_{10}} = 101101101101000110101_2$$

b) Para convertir $P_F = \mathbf{0, 6875}$ en su equivalente binario (la operación que rige es la multiplicación) multiplicamos 0, 6875 y cada parte fraccionaria sucesivamente por 2, extrayendo como digito binario, la parte entera del resultado y para la siguiente multiplicación se prosigue con la parte fraccionaria obtenida, la extracción de los dígitos binarios va en sentido descendente o sea de arriba hacia abajo.

P_F

0,6875 * 2 = **1**,375	1
0,375 * 2 = **0**,75	0
0,75 * 2 = **1**,5	1
0,5 * 2 ⇒	**1**

$$\mathbf{0, 6875_{10}} = 1011_2$$

Ejercicios propuestos:

Conversión binaria a decimal y viceversa

- $10101010111111010101010101,11101_2$
- $11101010111101001101010101011,1110_2$
- $01010111110111101010101001000,101_2$
- $10101001010001111001 1,111001_2$
- $1101010111111010101,111101_2$

Adición de números binarios

Para sumar dos o más dígitos binarios debemos tener presente las siguientes combinaciones de sumas binarias básicas:

0+0=0
0+1=1
1+0=1
1+1=0 llevando 1
1+1+1=1 llevando 1

Ejemplo:

$100110101_2 + 11010101_2$

$$\begin{array}{r} \color{red}{11\ 111\ 1\ \ 1} \\ 100110101_2 \\ +\ \ 11010101_2 \\ \hline 1000001010_2 \end{array}$$

Para realizar la suma binaria operamos las cantidades como si fuesen cifras decimales: comenzamos a sumar desde la derecha, en nuestro ejemplo, 1+1=10, entonces escribimos 0 y "llevamos" 1 (Esto es lo que se llama el *arrastre o carrie*). Se suma este 1 a la siguiente columna: 1+0+0=1, y seguimos hasta terminar todas las columnas (exactamente como en el sistema de numeración decimal).

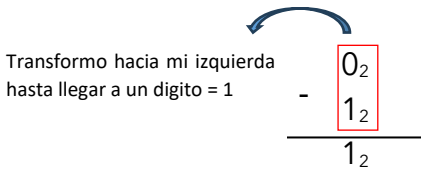
Resta de números binarios

El algoritmo de la resta, en binario, es el mismo que en el sistema de numeración decimal. Los términos que intervienen en la resta se llaman minuendo, sustraendo y diferencia. Las combinaciones básicas de la resta binaria son:

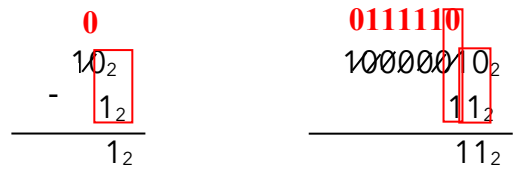
0 – 0 = 0
0 – 1 = 1 transformación o prestando un 1 de la siguiente columna
1 – 0 = 1
1 – 1 = 0

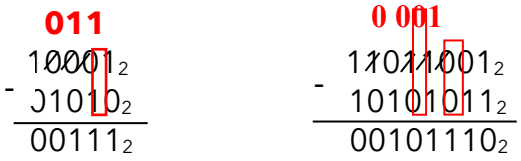
En la resta binaria la combinación binaria 0 – 1 = 1 provoca un proceso de transformación, el cual consiste en buscar un digito igual a 1 hacia mi izquierda

(hasta ahí dura este proceso de transformación), esta operación se realiza transformando los dígitos binarios ceros por unos y unos por ceros (aplicamos su complemento binario o lo que lo que es lo mismo restamos uno a cada digito que involucra este proceso). En una resta se pueden dar n procesos de transformación; a continuación, detallaremos algunas combinaciones posibles de este proceso:



Ejemplos:





Ejercicios propuestos:

Adición y resta de números binarios

- $10101010111111101010101010101,11101_2$ + / - $11101010110101001010110100101 1,1110_2$
- $11101010101111101010110100101 1,1110_2$ + / - $01010111110101010101100100 0,101_2$
- $11010111110101010111101010101001000,101_2$ + / - $10101001010000011111110011,111001_2$

- $10101001010000011111110011,111001_2$ + / - $11010101111110101010100101,11110001_2$

- $11101010101111101010101001000101 1,1110_2$ + / - $11010101111110101010100101,11110001_2$

Producto de números binarios

La regla para la multiplicación decimal tanto entera como fraccionaria también es válida para la multiplicación binaria. En efecto, la multiplicación binaria es más sencilla, ya que al multiplicar un número por el bit 0 o 1 da respectivamente 0 el mismo número, para lo cual tenemos que considerar las siguientes combinaciones posibles de multiplicaciones binarias.

0*0=0
0*1=0
1*0=0
1*1=1

Por ejemplo, multipliquemos $10110_2 * 1001_2$:

10110_2
x 1001_2

10110
00000
+ 00000
10110

11000110_2

Ejercicios propuestos:

Producto de Números Binarios

- $11101010101111101010010110100101 1,1110_2 * 1011,11_2$

- $10101010111111111101010101010101,11101_2 * 111,001_2$

- $01010111110101010111101010101001000,101_2 * 1010,10_2$
- $10101001010000011111110011,111001_2 * 10110_2$
- $11010101111110101010100101,11110001_2 * 10101,1_2$

División de números binarios

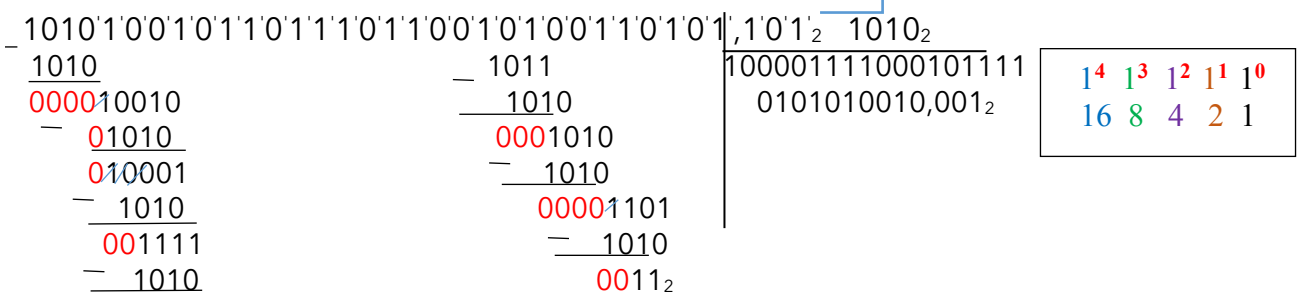
En la división binaria aplicamos las reglas de la división decimal tanto en la parte entera como como en la parte fraccionaria, además aplicamos la forma o notación expandida de un número debido a que esta operación me permite convertir cualquier número a su equivalente decimal (el sistema de numeración que nosotros dominamos), por lo que es fundamental que traslademos todo cálculo al sistema en el cuál podemos definir con certeza si un número nos alcanza para dividir o no, caso contrario con la experticia adquirida de la forma o notación expandida de un número la podemos simplificar por la siguiente representación de índices binarios, lo cual simplificará notablemente nuestro trabajo:

$1^4\ 1^3\ 1^2\ 1^1\ 1^0$
16 8 4 2 1

En cada una de las restas que vamos obteniendo en la división binaria comprobamos que se cumpla a cabalidad la regla básica de la división, la misma que nos exige que la diferencia de cada resta obtenida en la división debe de ser menor al divisor, caso contrario no podemos continuar con la división.

Ejemplo:

Equivale a 10 decimal



```
 01010
-1010
 ----
 00001110
  -1010
  -----
   010011
   -1010
   -----
    010010
    -1010
    -----
     010000
     -1010
     -----
      001101
      -1010
      -----
       001101
       -1010
       -----
        001100
        -1010
        -----
         0010
```

Ejercicios propuestos:

División de Números Binarios

- $10101010111111101001011111110111010010010101010101010111101110101010101010101,11101_2 / 1010_2$
- $11101010101111101010101110101010010100101011010010111110100101010010101,10100101111102 / 1101_2$
- $01010111110101010111101010101101010111111101001011111110101011111110,1001011111_2 / 0110_2$
- $11010101111110101010010101011111010101011101010101111101,0$ $10100101010010102 / 1010_2$

Complementos

Los complementos aritméticos surgen en dos contextos distintos, pero estrechamente relacionados entre sí. Mientras las personas utilizamos los símbolos + y – para representar números positivos y negativos, las computadoras trabajan exclusivamente con bits sucesivos. Aunque se puede designar un bit específico para indicar el signo de un número (por ejemplo, 0 para positivo y 1 para negativo), muchas arquitecturas computacionales optan por representar los números negativos mediante su complemento aritmético relacional.

Este concepto también se emplea en las operaciones de resta, ya que permite transformar una sustracción en una suma sucesiva, simplificando así el proceso y eliminando la necesidad de recurrir al préstamo y devolución entre columnas, por lo que deducimos que los complementos tanto decimales o binarios afectan exclusivamente al sustraendo.

Existen dos tipos principales de complementos: el complemento a la base menos uno (C_{b-1}) y el complemento a la base (C_b). En el sistema decimal, estos se conocen como complemento a nueves y complemento a dieces, respectivamente. En el sistema binario, sus equivalentes son el complemento a unos y el complemento a doces (Lipschutz,1995).

COMPLEMENTOS

- **DECIMAL**
  - $\mathbf{C_{b-1} = C_{10-1} = C_{9\ (nueves)} =}$ Se obtiene restando 9 a cada del sustraendo.
  - $\mathbf{C_b = C_{10\ (dieces)} =}$ Se obtiene sumando 1 al C_{b-1}.
- **BINARIO**
  - $\mathbf{C_{b-1} = C_{2-1} = C_{1\ (unos)} =}$ Se obtiene restando 1 a cada del sustraendo.
  - $\mathbf{C_b = C_{2\ (doces)} =}$ Se obtiene sumando 1 al C_{b-1}.

Complementos decimales

En el sistema de numeración decimal se obtiene el C_{b-1} o Complemento a nueves restando 9 a cada dígito del sustraendo, y el C_b o Complemento a dieces del sustraendo, el cual se obtiene sumando 1al complemento a nueves del sustraendo. En los complementos decimales existen 3 pasos de resolución detallados a continuación:

1. Resta común y corriente.
2. Sumamos al minuendo el C_{b-1} del sustraendo y finalmente usamos el método lleve la punta alrededor para obtener la respuesta.
3. Sumamos al minuendo el C_b del sustraendo y finalmente eliminamos el último digito para obtener la respuesta.

Por ejemplo: Resolver Y = A – B

A = 8 4 3 2 6 8 7 4 3 2
B = 4 2 6 1 3 4 5 2 6 8

a) Diferencia común y corriente:

```
  8 4 3 2 6 8 7 4 3 2
- 4 2 6 1 3 4 5 2 6 8
-------------------
  4 1 7 1 3 4 2 1 6 4
```

b) Sumando al minuendo el C_{b-1} del sustraendo:

```
                - 9 9 9 9 9 9 9 9 9 9
Sustraendo        4 2 6 1 3 4 5 2 6 8
-------------------------------------
C b-1             5 7 3 8 6 5 4 7 3 1

Minuendo          8 4 3 2 6 8 7 4 3 2
C b-1           + 5 7 3 8 6 5 4 7 3 1
-------------------------------------
              1 4 1 7 1 3 4 2 1 6 3
```

Usamos el método que se conoce con el nombre de *lleve la punta alrededor:*

$$\begin{array}{r} 4\,1\,7\,1\,3\,4\,2\,1\,6\,3 \\ +\quad\quad\quad\quad\quad\quad\quad\quad 1 \\ \hline 4\,1\,7\,1\,3\,4\,2\,1\,6\,4 \end{array}$$

c) Sumamos al minuendo el C_b del sustraendo:

$$\begin{array}{lr} & 9\,9\,9\,9\,9\,9\,9\,9\,9\,9 \\ \text{Sustraendo} & -\quad\quad 4\,2\,6\,1\,3\,4\,5\,2\,6\,8 \\ C_{b-1} & \quad 5\,7\,3\,8\,6\,5\,4\,7\,3\,1 \end{array}$$

Obtengo el C_b, sumando 1 al C_{b-1}.

$$\begin{array}{lr} C_{b-1} & 5\,7\,3\,8\,6\,5\,4\,7\,3\,1 \\ & +\quad\quad\quad\quad\quad 1 \\ C_b & \quad 5\,7\,3\,8\,6\,5\,4\,7\,3\,2 \end{array}$$

$$\begin{array}{lr} \text{Minuendo} & +\,8\quad\quad 8\,4\,3\,2\,6\,8\,7\,4\,3\,2 \\ C_b & \quad\quad 5\,7\,3\,8\,6\,5\,4\,7\,3\,2 \\ & \overline{1}\,4\,1\,7\,1\,3\,4\,2\,1\,6\,4 \end{array}$$

Eliminando el último digito desde la izquierda para obtener la respuesta.

Ejercicios propuestos:

Complementos Decimales

- 989832758973454,25435 – 3453455787676,45436
- 57566978998745,45435 – 456657756768,67878
- 345546778697809,45456546 – 54353453435,456456
- 998787867867343,6565 – 7654342343342,8767678
- 34234678887653334- 786756745453453,9878979

Complementos binarios

En el sistema de numeración binario se obtiene el C_{b-1} o Complemento a unos restando 1 a cada dígito del sustraendo, y el C_b o Complemento a doces del sustraendo, el cual se obtiene sumando 1al complemento a unos del sustraendo. En los complementos binarios existen 3 pasos de resolución detallados a continuación:

1. Resta común y corriente.

2. Sumamos al minuendo el C_{b-1} del sustraendo y finalmente usamos el método lleve la punta alrededor para obtener la respuesta.

3. Sumamos al minuendo el C_b del sustraendo y finalmente eliminamos el último digito para obtener la respuesta.

Ejemplo: Dados A y B, resolver A-B

$A = 1\,1\,1\,1\,0\,0\,0\,0_2$
$B = 1\,0\,0\,0\,1\,1\,1\,0_2$

a) diferencia común y corriente:

$$\begin{array}{rr} - & 1\,1\,1\,1\,0\,0\,0\,0_2 \\ & 1\,0\,0\,0\,1\,1\,1\,0_2 \\ \hline & 0\,1\,1\,0\,0\,0\,1\,0_2 \end{array}$$

b) Sumando al minuendo el C_{b-1} del sustraendo:

$$\begin{array}{lrr} & - & 1\,1\,1\,1\,1\,1\,1\,1_2 \\ \text{Sustraendo} & & 1\,0\,0\,0\,1\,1\,1\,0_2 \\ C_{b-1} & \hline & 0\,1\,1\,1\,0\,0\,0\,1_2 \end{array}$$

$$\begin{array}{lrr} & & 1\,1\,1\,1\,0\,0\,0\,0_2 \\ \text{Minuendo} & + & 0\,1\,1\,1\,0\,0\,0\,1_2 \\ C_{b-1} & \hline & 1\,0\,1\,1\,0\,0\,0\,0\,1_2 \end{array}$$

Usando el método que se conoce con el nombre de *lleve* ***la punta alrededor***:

$$\begin{array}{r} 0\,1\,1\,0\,0\,0\,0\,1_2 \\ +\quad\quad\quad 1 \\ \hline 0\,1\,1\,0\,0\,0\,1\,0_2 \end{array}$$

c) Sumando al minuendo el C_b del sustraendo.

$$\begin{array}{lr} & 1\,1\,1\,1\,1\,1\,1\,1_2 \\ \text{Sustraendo} & -\quad 1\,0\,0\,0\,1\,1\,1\,0_2 \\ C_{b-1} & \hline 0\,1\,1\,1\,0\,0\,0\,1_2 \end{array}$$

Obtengo el C_b, sumando 1 al C_{b-1}

$$\begin{array}{lr} C_{b-1} & 0\,1\,1\,1\,0\,0\,0\,1_2 \\ & +\quad\quad\quad 1 \\ \hline C_b & 0\,1\,1\,1\,0\,0\,1\,0_2 \end{array}$$

$$\begin{array}{lr} \text{Minuend} & +\quad 1\,1\,1\,1\,0\,0\,0\,0_2 \\ C_b & 0\,1\,1\,1\,0\,0\,1\,0_2 \\ \hline & 1\,0\,1\,1\,0\,0\,0\,1\,0_2 \end{array}$$

Ejercicios propuestos:

Complementos Binarios

- $10101010111111101110101010101010101,11101_2$ - $11101010101111101010010110101011,1110_2$
- $11101010101111101011001010110100 1011,1110_2$ - $01010111110101010111101010101001000,101_2$
- $01010111110101010111101010101001000,101_2$ - $10101001010000011111110011,111001_2$
- $10101001010000011111110011,111001_2$ - $11010101111110101010010 1,11110001_2$

- $11101010101111101010101001010110010 11,1110_2$ - $11010101111110101010010 1,11110001_2$

Sistema de numeración octal

Figura 4: Introducción al sistema octal.
Fuente: Elaboración propia.

Los dígitos del sistema de numeración octal son 0, 1, 2, 3, 4, 5, 6, 7. La base del sistema de numeración es igual a 8 (b_8). Se dice que cada dígito octal tiene una única representación binaria de 3 bits debido a la división de la base del sistema octal en este caso 8 para la base del sistema de numeración en el cual basa su funcionamiento el ordenador (sistema binario) o sea 2 (Jiménez, 2014).

8 / 2

8	2
4	0
2	0
1	0

→ 3 bits de Representación Binaria
Combinación Inicial

Tabla1: Método obtención de equivalente binarios. equivalente binarios.

Dígitos Octales	Equivalente Binario
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Fuente: Elaboración propia.

Tabla 2: Dígitos octales vs

$$\begin{array}{l} 000 \Rightarrow 0 \\ +\ 1 \\ \hline 001 \Rightarrow 1 \\ +\ 1 \\ \hline 010 \Rightarrow 2 \\ +\ 1 \\ \hline 011 \Rightarrow 3 \\ +\ 1 \\ \hline 100 \Rightarrow 4 \\ \vdots \\ 111 \Rightarrow 7 \end{array}$$

Partimos de la primera combinación en este caso 000=0 y sumamos una posición binaria o sea 1 hasta llegar al último digito octal con su equivalencia binaria en este caso 111=7.

Fuente: Elaboración propia.

Conversión octal a decimal

En el sistema de numeración octal al realizar la conversión a su equivalente decimal usamos la operación denominada forma o notación expandida de un número, la cual nos permite obtener el equivalente decimal de un número sin importar a que sistema de numeración pertenece dicho número, ejemplo:

$$74601204321{,}1356_8 \Rightarrow X_{10}$$

Utilizando la forma o notación expandida tenemos:

$$7^{10}\ 4^{9}\ 6^{8}\ 0^{7}\ 1^{6}\ 2^{5}\ 0^{4}\ 4^{3}\ 3^{2}\ 2^{1}\ 1^{0}\ ,\ 1^{-1}\ 3^{-2}\ 5^{-3}\ 6^{-4}{}_8 =$$

$$7*8^{10} + 4*8^{9} + 6*8^{8} + 0*8^{7} + 1*8^{6} + 2*8^{5} + 0*8^{4} + 4*8^{3} + 3*8^{2} + 2*8^{1} + 1*8^{0} + 1*8^{-1} +$$

$$3*8^{-2} + 5*8^{-3} + 6*8^{-4} = 8154056913{,}18310546875_{10}$$

$$\mathbf{74601204321{,}1356_8} = 8154056913{,}18310546875_{10}$$

Conversión decimal a octal

Para realizar la conversión del sistema de numeración decimal a su equivalente octal usamos las mismas reglas que empleamos en la conversión decimal a binario tanto en su parte entera como en su parte fraccionara, ejemplo:

$8154056913{,}18310546875_{10}$ ⇨ Y_8

Parte Entera

8154056913	**8**
1019257114,125	**1**
127407139,25	**2**
15925892,375	**3**
1990736,5	**4**
248842	**0**
31105,25	**2**
3888,125	**1**
486	**0**
60,75	**6**
7,5	**4**

8154056913_{10} = 74601204321_8

Parte Fraccionaria

0, 18310546875 * 8 = **1**,46484375 — **1**
0,46484375*8 = **3**,71875 — **3**
0,71875*8 = **5**,75 — **5**
0,75*8 = — **6**

$0{,}18310546875_{10}$ = $0{,}1356_8$

Ejercicios propuestos:

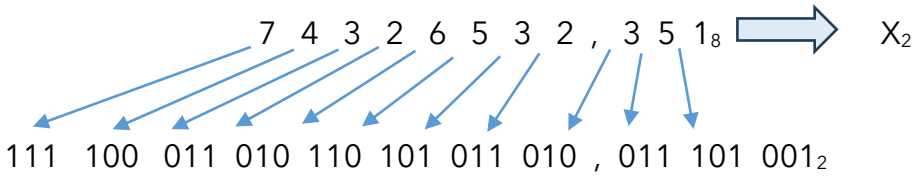
Conversión Octal a Decimal y Viceversa

- $74645472654327723,24344_8$
- $242347634524234523735,45_8$
- $234235462342463,3454358$
- $45676534534545343534,235_8$
- $745654345,32132135647_8$

Conversión octal a binario

En el sistema de numeración octal al realizar la conversión a su equivalente binario se recomienda reemplazar cada dígito octal a su equivalente binario (Tabla1), sin importar si lo hacemos de derecha a izquierda o viceversa, sea en su parte entera o en su parte fraccionaria el resultado es el mismo.

Ejemplo:



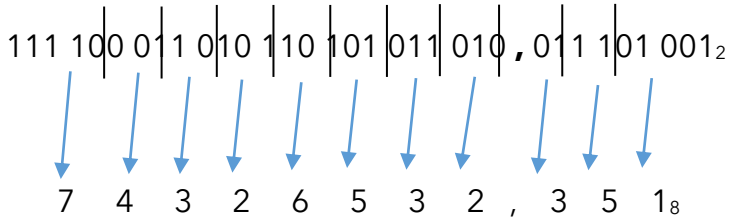
$$\mathbf{7\,4\,3\,2\,6\,5\,3\,2\,,\,3\,5\,1_8} = 111100011010110101011010,011101001_2$$

Conversión binaria a octal

En el sistema de numeración octal al realizar la conversión de un número binario a su equivalente octal, se recomienda agrupar en segmentos de 3 bits de representación, para lo cual partimos observando si el número posee parte

fraccionaria, si este es el caso agrupamos en la parte entera de derecha a izquierda a partir de la coma del número en sectores de 3 bits y en su parte fraccionaria agrupamos de izquierda a derecha en sectores de 3 bits, si al momento de agrupar nos encontramos con sectores incompletos los rellenamos con ceros en la parte entera de derecha a izquierda y en la parte fraccionaria de izquierda a derecha. Finalmente usamos la tabla 1 donde pasamos los sectores binarios agrupados a sus equivalentes en números octales.

$11110001101011010101011010, 011101001_2$ ⇨ Y_8



$11110001101011010101011010, 011101001_2$ = 7 4 3 2 6 5 3 2 , 3 5 1_8

Ejercicios propuestos:

Conversión Octal a Binario y Viceversa

- $74645472654374733632 7723,24434_8$
- $24234763452346234523735,455_8$
- $234235465474565463,345_8$
- $45676536453534,23435_8$
- $745654345645345,32136647_8$

Aritmética Octal

Tabla 3: Comprobación de la suma octal

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

Fuente: Elaboración propia.

Adición octal

Cuando sumamos 2 dígitos octales, los sumamos como si fueran dígitos decimales, luego verificamos si dicha suma es mayor al último digito del sistema de numeración (suma > 7), si la suma es mayor a 7 restamos la base del sistema de numeración (b= 8) y llevamos un 1 a la próxima columna.

$$\begin{array}{r} 4_8 \\ +\ 6_8 \\ \hline 10 \\ -\ 8 \\ \hline 12_8 \end{array} \qquad \begin{array}{r} 7_8 \\ +\ 5_8 \\ \hline 12 \\ -\ 8 \\ \hline 14_8 \end{array}$$

$$\begin{array}{r} {\scriptstyle 1\quad\ 1\ 1\ 1\ 1\ 1\ 1\ 1} \\ 7\ 4\ 3\ 6\ 5\ 1\ 2\ 6\ 4\ 7_8 \\ +\ 5\ 3\ 2\ 6\ 5\ 7\ 6\ 4\ 3\ 2_8 \\ \hline 12\ 7\ 6\ 13\ 11\ 9\ 9\ 11\ 8\ 9 \\ -8\ \ \ -8\ -8\ -8\ -8\ -8\ -8\ -8 \\ \hline \mathbf{1\ 4\ 7\ 6\ 5\ 3\ 1\ 1\ 3\ 0\ 1_8} \end{array}$$

Diferencia octal

Cuando realizamos una diferencia de 2 dígitos octales, se obtiene primero el complemento a la base menos 1 (C_{b-1}) del sustraendo o lo que es lo mismo complementos a 7 (el cual se obtiene restando 7 a cada dígito del sustraendo), después se obtiene el complemento a la base (C_b) del sustraendo o lo que es lo mismo el complemento a 8 (el cual se obtiene sumando 1 al C_{b-1}); y finalmente el C_b le sumamos al minuendo para obtener la respuesta (aplicando la regla de la suma octal).

Ejemplo: A - B

$A = 7\,4\,3\,6\,5\,1\,2\,6\,4\,7_8$ => *Minuendo*

$B = 2\,3\,4\,6\,7\,5\,6\,2\,3\,5_8$ => *Sustraendo*

		7	7	7	7	7	7	7	7	7	7	
-		2	3	4	6	7	5	6	2	3	5_8	=> *Sust.*
		5	4	3	1	0	2	1	5	4	2	=> C_{b-1}
+											1	
		5	4	3	1	0	2	1	5	4	3	=> C_b
	1	**1**						**1**	**1**	**1**		
+		7	4	3	6	5	1	2	6	4	7_8	=> *Min.*
		5	4	3	1	0	2	1	5	4	3_8	=> C_b
		13	8	6	7	5	3	4	12	9	10	
		- 8	- 8						- 8	- 8	- 8	
	1	**5**	**0**	**6**	**7**	**5**	**3**	**4**	**4**	**1**	**2_8**	

Ejercicios propuestos:

Adición y Resta de Números Octales

- $7464547265433 6723,243434_8$ + / - $24234763452337252,4545_8$
- $7426537236327723,2434_8$ + / - $2344756723565463,345435_8$
- $74647343723 6327723,243434_8$ + / - $4567653453453 5434,234325_8$

- $6543747343723632773,434_8$ + / - $7456545643345345,3213213_8$
- $456765345345334, 4325_8$ + / - $23423546547525,3455_8$

Sistema de Numeración Hexadecimal

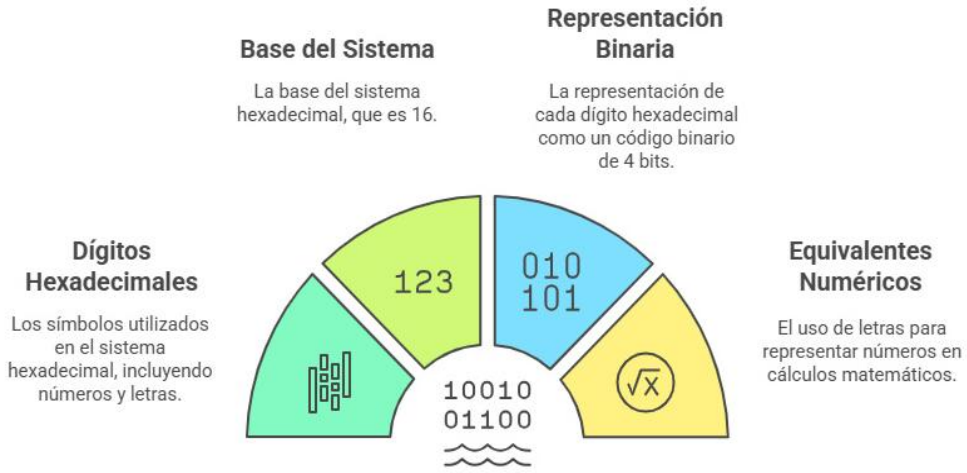


Figura 5: Estructura del sistema hexadecimal
Fuente: Elaboración propia.

Los dígitos del sistema de numeración hexadecimal (abreviadamente Hex o hex) son 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. La base del sistema de numeración es igual a 16 (b_{16}). Se dice que cada dígito hexadecimal tiene una única representación binaria de 4 bits debido a la división de la base del sistema hexadecimal en este caso 16 para la base del sistema de numeración en el cual basa su funcionamiento el ordenador (sistema binario) o sea 2.

En el sistema de numeración hexadecimal para los cálculos matemáticos sus dígitos alfabéticos se reemplazan por sus equivalentes numéricos (10=>A, 11=>B, etc.), en cambio sus caracteres alfabéticos se emplean para visualizar o representar un resultado (A=>10, B=>11, etc.) (Lipschutz,1995).

A => 10
B => 11
C => 12
D =>13
E => 14
F => 15

16 / 2

16	2
8	0
4	0
2	0
1	0

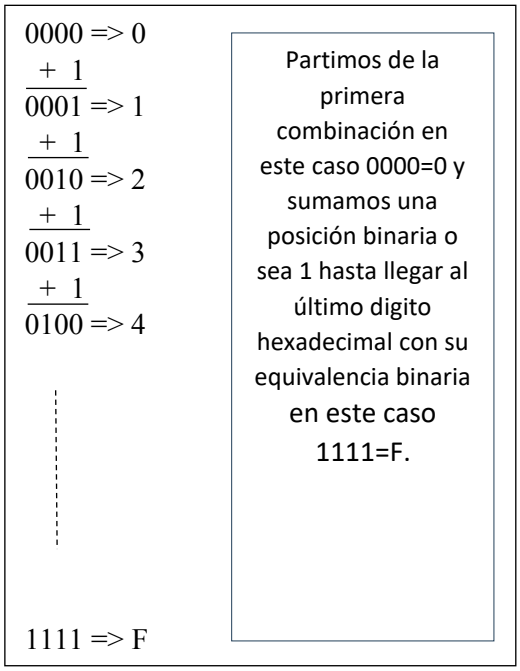
4 bits de Representación Binaria - Combinación Inicial

Tabla4: dígitos hex vs equivalente binarios. binarios.

Dígitos Hexadecimales	Equivalentes Binarios
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Fuente: Elaboración propia.

Tabla5: método obtención de equivalente



Fuente: Elaboración propia.

Conversión hexadecimal a decimal

En el sistema de numeración hexadecimal al realizar la conversión a su equivalente decimal usamos la operación denominada forma o notación expandida de un número, la cual nos permite obtener el equivalente decimal de un número sin importar a que sistema de numeración pertenece dicho número, ejemplo:

$$FAB7424DCE,AB019_{16} \Rightarrow X_{10}$$

Reemplazamos los caracteres alfabéticos por sus equivalentes numéricos y luego usamos la forma o notación expandida de un número:

$15^{9}\ 10^{8}\ 11^{7}\ 7^{6}\ 4^{5}\ 2^{4}\ 4^{3}\ 13^{2}\ 12^{1}\ 14^{0}\ ,\ 10^{-1}\ 11^{-2}\ 0^{-3}\ 1^{-4}\ 9^{-5}{}_{16} =$

$15*16^{9} + 10*16^{8} + 11*16^{7} + 7*16^{6} + 4*16^{5} + 2*16^{4} + 4*16^{3} + 13*16^{2} + 12*16^{1} + 14*16^{0}$

$+ 10* 16^{-1} + 11*16^{-2} + 0*16^{-3} + 1*16^{-4} + 9*16^{-5} =$

$1076816399822{,}667992591857910156251_{0}$

$$\mathbf{FAB7424DCE,AB019_{16}} = 1076816399822{,}667992591857910156251_{0}$$

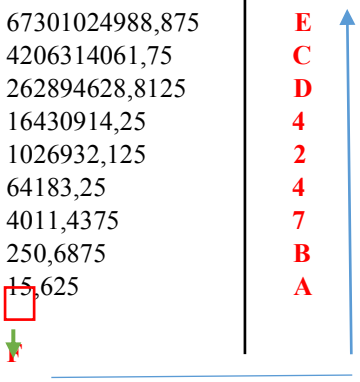
Conversión decimal a hexadecimal

Para realizar la conversión del sistema de numeración decimal a su equivalente hexadecimal usamos las mismas reglas que empleamos en la conversión decimal a octal tanto en su parte entera como en su parte fraccionara, ejemplo:

$1076816399822{,}667992591857910156251_{0}$ ⇨ Y_{16}

Parte Entera

1076816399822	16
67301024988,875	E
4206314061,75	C
262894628,8125	D
16430914,25	4
1026932,125	2
64183,25	4
4011,4375	7
250,6875	B
15,625	A
1	



1076816399822$_{10}$ = FAB7424DCE$_{16}$

Parte Fraccionaria

0, 66799259185791015625 * 16 = **10**,6878814697265625 **A**

0, 6878814697265625 * 16 = **11**,006103515625 **B**

0, 006103515625 * 16 = **0**,09765625 **0**

0, 09765625 * 16 = **1**,5625 **1**

0, 5625 * 16 = **9**

0, 6679925918579101562$_{10}$ = 0,AB019$_{16}$

Ejercicios propuestos:

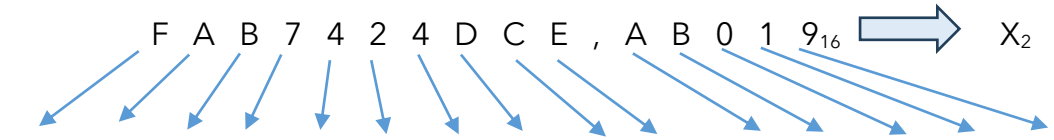
Conversión hexadecimal a decimal y viceversa

- $A758B697125C36E712,56D532_{16}$
- $8A682DC263C7127E676,72F6E71_{16}$
- $92A374B623956F2,653D5A6376_{16}$
- $A67B67269462B4C5D623,7645B_{16}$
- $BCD767D1E2F531C2,6D5A293B_{16}$

Conversión hexadecimal a binario

En el sistema de numeración hexadecimal al realizar la conversión a su equivalente binario se recomienda reemplazar cada dígito hexadecimal a su equivalente binario (Tabla3), sin importar si lo hacemos de derecha a izquierda o viceversa, sea en su parte entera o en su parte fraccionaria el resultado es el mismo.

Ejemplo:



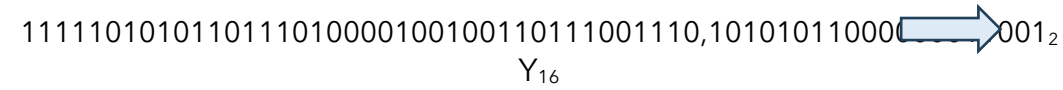
1111 1010 1011 0111 0100 0010 0100 1101 1100 1110 , 1010 1011 0000 0001 1001_2

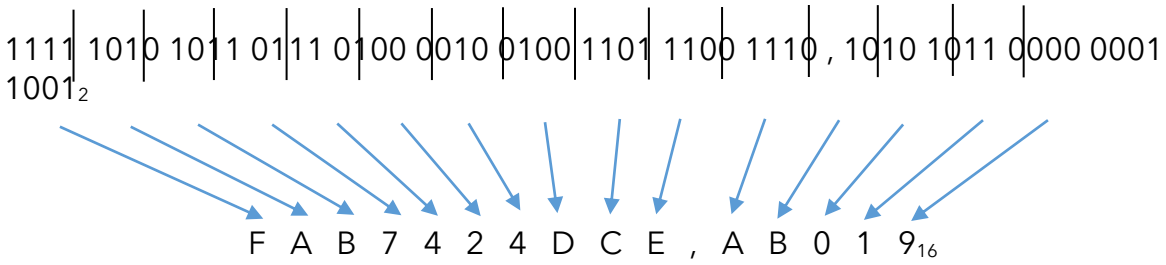
FAB7424DCE,AB019$_{16}$ =
1111010101101110100001001001101110011 10,10101011000000011001$_2$

Conversión binaria a hexadecimal

En el sistema de numeración hexadecimal al realizar la conversión de un número binario a su equivalente hexadecimal, se recomienda agrupar en segmentos de 4 bits de representación, para lo cual partimos observando si el número posee parte fraccionaria, si este es el caso agrupamos en la parte entera de derecha a izquierda a partir de la coma en sectores de 4 bits y en su parte fraccionaria agrupamos de izquierda a derecha en sectores de 4 bits.

Si al momento de agrupar nos encontramos con sectores incompletos los rellenamos con ceros en la parte entera de derecha a izquierda y en la parte fraccionaria de izquierda a derecha. Finalmente usamos la tabla 3 donde pasamos los sectores binarios agrupados a sus equivalentes en números octales.





1111010101101110100001001001101110011 10,10101011000000011001$_2$ =
FAB7424DCE,AB019$_{16}$

Ejercicios propuestos:

Conversión hexadecimal a binario y viceversa

- AB75E7F3671213C1,56D7E6F12F$_{16}$
- B6D2E6F867A625B371,72A16D$_{16}$
- C2D37246A3B25B6A2,658691A$_{16}$
- D8697D6B792B4A5D, 4F3E5639BA$_{16}$
- A9786A7D1DA1C2B,6B59C2D5$_{16}$

Aritmética Hexadecimal

Tabla 6: Comprobación de la suma hexadecimal

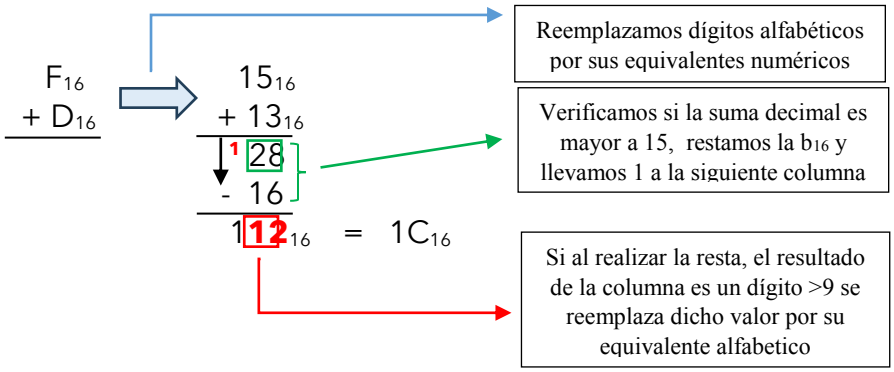
+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

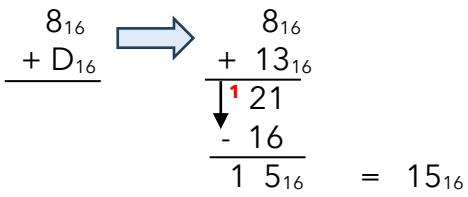
Fuente: Elaboración propia.

Adición hexadecimal

Cuando sumamos 2 dígitos hexadecimal, los sumamos como si fueran dígitos decimales, luego verificamos si dicha suma es mayor al último digito del sistema de numeración (suma > 15), si la suma es mayor a 15 restamos la base del sistema de numeración (b= 16) y llevamos un 1 a la próxima columna.

Ejemplo:





Resolver: Y= A + B

A= FAB7424DCE,AB019$_{16}$

B= 9FAB12CB1A,12CDE$_{16}$

```
   1  1  1  1     1     1              1
     15 10 11  7  4  2  4  3 12 14 , 10 11  0  1  9₁₆
+
      9 15 10 11  1  2 12 11  1 10 ,  1  2 12 13 14₁₆
-------------------------------------------------------
     25 26 22 18  5  5 16 14 14 24   11 13 12 15 23
-
     16 16 16 16    16       16             16
-------------------------------------------------------
   1  9  A  6  2  5  5  0  E  E  8 , B  D  C  F  7₁₆
```

A + B= $19A62550EE8,BDCF7_{16}$

Diferencia Hexadecimal

Cuando realizamos una diferencia de 2 dígitos hexadecimales, se obtiene primero el complemento a la base menos 1 (C_{b-1}) del sustraendo o lo que es lo mismo complementos a 15 (el cual se obtiene restando 15 a cada dígito del sustraendo), después se obtiene el complemento a la base (C_b) del sustraendo o lo que es lo mismo el complemento a 16 (el cual se obtiene sumando 1 al C_{b-1}); y finalmente el C_b le sumamos al minuendo para obtener la respuesta (aplicando la regla de la suma hexadecimal).

Resolver: Y= A - B

A= $FAB7424DCE,AB019_{16}$ => *Minuendo*

B= $9FAB12CB1A,12CDE_{16}$ => *Sustraendo*

```
   15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
-
    9 15 10 11  1  2 12 11  1 10 ,  1  2 12 13 14₁₆ => Sust.
```

$$
\begin{array}{rcccccccccccccccl}
+ & 6 & 0 & 5 & 4 & 14 & 13 & 3 & 4 & 14 & 5 & , & 14 & 13 & 3 & 2 & 1 & => & C_{b-1} \\
 & & & & & & & & & & & & & & & & 1 & & \\
\hline
 & 6 & 0 & 5 & 4 & 14 & 13 & 3 & 4 & 14 & 5 & , & 14 & 13 & 3 & 2 & 2 & => & C_b
\end{array}
$$

$$
\begin{array}{rcccccccccccccccl}
 & {\scriptstyle 1} & & {\scriptstyle 1} & & {\scriptstyle 1} & & & & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & & & & & & \\
+ & 15 & 10 & 11 & 7 & 4 & 2 & 4 & 3 & 12 & 14 & , & 10 & 11 & 0 & 1 & 9_{16} & => & Min. \\
 & 6 & 0 & 5 & 4 & 14 & 13 & 3 & 4 & 14 & 5 & , & 14 & 13 & 3 & 2 & 2 & => & C_b \\
\hline
 & 21 & 11 & 16 & 12 & 18 & 15 & 7 & 8 & 27 & 20 & & 25 & 24 & 3 & 3 & 11 & & \\
- & 16 & & 16 & & 16 & & & & 16 & 16 & & 16 & 16 & & & & & \\
\hline
 & 1 & 5 & B & 0 & C & 2 & F & 7 & 8 & B & 4 & , & 9 & 8 & 3 & 3 & B_{16} &
\end{array}
$$

A – B = $15B0C2F78B4,9833B_{16}$

Ejercicios propuestos:

Suma y resta hexadecimal

- $D8697D6BEBA5D, 4F3EA_{16}$ +/ - $B6D2E6F86B71,72A1_{16}$
- $FA682D67E676,72F1D6_{16}$ +/ - $92A3747439F2,65336_{16}$
- $A67B67269483623,764A4B_{16}$ +/ - $BCD767D31C2,6D5926A93B_{16}$
- $D8697D6B4A5D, 4F3EBA_{16}$ +/ - $A758B6E712,5678D2_{16}$
- $A9786A7D1A1C2B,6B5262A3D5_{16}$ +/ - $BCD767D1E2F31C2,6D592293B_{16}$

Sistema de Numeración Base b

Los dígitos del sistema de numeración $base_b$ son: cualquier número entero positivo(Z^+) hasta …+∞. La base del sistema de numeración $base_b$ es igual a: cualquier número entero positivo excepto:

- 10 es la base del sistema de numeración decimal.
- 2 es la base del sistema de numeración binario.
- 8 es la base del sistema de numeración octal.
- 16 es la base del sistema de numeración hexadecimal.

El sistema de numeración $base_b$ solo se relaciona con el sistema de numeración decimal, es decir se puede realizar la conversión $base_b$ a decimal y viceversa (Lipschutz,1995).

Conversión $base_b$ a decimal

En el sistema de numeración $base_b$ al realizar la conversión a su equivalente decimal usamos la operación denominada forma o notación expandida de un número, la cual nos permite obtener el equivalente decimal de un número sin importar a que sistema de numeración pertenece dicho número, ejemplo:

$$340124332{,}2134_{b=5} \Rightarrow X_{10}$$

En este ejemplo el número pertenece al sistema de numeración $base_b$ por lo tanto sus dígitos son: 0,1,2,3,4 y su base es = 5.

$3^{8}\ 4^{7}\ 0^{6}\ 1^{5}\ 2^{4}\ 4^{3}\ 3^{2}\ 3^{1}\ 2^{0}\ ,\ 2^{-1}\ 1^{-2}\ 3^{-3}\ 4^{-4}\ _{b=5} =$

$3*5^{8} + 4*5^{7} + 0*5^{6} + 1*5^{5} + 2*5^{4} + 4*5^{3} + 3*5^{2} + 3*5^{1} + 2*5^{0} + 2*\ 5^{-1} + 1*5^{-2} + 3*5^{-3}$

$+ 4*5^{-4} = 1489342,4704_{10}$

$$\mathbf{340124332,2134_{b=5}} = 1489343,4704_{10}$$

Conversión decimal a base$_b$

Para realizar la conversión del sistema de numeración decimal a su equivalente base$_b$ usamos las mismas reglas que empleamos en la conversión decimal a hexadecimal tanto en su parte entera como en su parte fraccionara, ejemplo:

$$1489342,4704_{10} \Rightarrow Y_{b=5}$$

Parte Entera

1489342	**5**
297868,4	**2**
59573,6	**3**
11914,6	**3**
2382,8	**4**
476,4	**2**
95,2	**1**
19	**0**
3,8	**4**

$1489342,4704_{10}$ = $340124332_{b=5}$

Parte Fraccionaria

0, 4704 * 5	=	**2**,352	**2**
0, 352 * 5	=	**1**,76	**1**
0, 76 * 5	=	**3**,8	**3**
0, 8 * 5	=		**4**

$0,4704_{10}$ = $0,2134_{b=5}$

Ejercicios propuestos:

Conversión base$_b$ a decimal y viceversa

- $875346313145467223454654 7,23643_{b=9}$
- $4320948349870202035345,23534_{b=9}$
- $3534234234342534534524,21444534_{b=6}$
- $6364324324343345535345 3,3534312_{b=7}$
- $8772647823342423423423,23450_{b=9}$

En la actualidad, los sistemas de numeración y la codificación digital son la base de tecnologías emergentes como la computación cuántica, donde la información se representa mediante qubits en lugar de bits, ampliando exponencialmente la capacidad de cálculo. Asimismo, el manejo de bases numéricas continúa siendo esencial en áreas como la ciberseguridad, donde los sistemas criptográficos utilizan operaciones modulares y conversiones numéricas para proteger la información en entornos digitales.

Resumen del capítulo

El capítulo Fundamentos de Matemáticas Discretas del libro *"Matemáticas Discretas: El Lenguaje Secreto de la Computación"* explora los principios básicos que sustentan el manejo y representación de la información numérica en el ámbito computacional. Inicia con una introducción a los sistemas de numeración, definiendo su propósito y características esenciales, para luego profundizar en el proceso de conversión entre diferentes bases numéricas. Se aborda el sistema binario como la base del lenguaje de las computadoras, incluyendo sus operaciones fundamentales y el uso de complementos –tanto decimales como binarios– como métodos para facilitar cálculos aritméticos y representar números negativos. Además, se estudian sistemas numéricos alternativos como el octal y el hexadecimal, relevantes en programación y arquitectura de computadoras, así como las distintas formas de codificación que permiten interpretar y manipular información digital. Este capítulo proporciona

una base sólida para comprender cómo los números y datos son estructurados, procesados y comunicados en el entorno digital.

Conclusiones

- Los sistemas de numeración son la base del lenguaje computacional permiten comprender los sistemas decimal, binario, octal y hexadecimal es esencial para interpretar cómo las computadoras representan, procesan y almacenan información digital.

- Las operaciones y conversiones entre sistemas fortalecen el pensamiento lógico, permitiendo aprender a realizar conversiones, sumas, restas, multiplicaciones y divisiones en diferentes bases numéricas permite desarrollar habilidades prácticas y abstractas clave en programación y arquitectura de computadoras.

- El uso de complementos y codificación digital optimiza el procesamiento de datos, los complementos binarios y decimales simplifican cálculos aritméticos, mientras que la codificación mediante bits y bytes permite representar eficientemente la información en sistemas digitales.

Autoevaluación / Cuestionario

Selecciona la opción correcta

1. ¿Qué es un sistema de numeración?

- Un conjunto de símbolos matemáticos utilizados para realizar operaciones aritméticas complejas.
- Un método para representar cantidades utilizando un conjunto de símbolos y reglas.
- Una forma de codificar información alfanumérica para su almacenamiento en computadoras.

- Un diagrama que muestra la relación entre diferentes conjuntos de números.

2. ¿Cuál es la importancia de la base en un sistema de numeración?

- Determina la complejidad de las operaciones aritméticas.
- Define el número de dígitos o símbolos únicos utilizados para representar cantidades.
- Indica la precisión con la que se pueden representar los números fraccionarios.
- Establece el orden en el que se deben leer los dígitos de un número.

3. El proceso general para convertir un número del sistema decimal a cualquier otra

base implica:

- Multiplicar el número decimal por la nueva base y tomar los residuos.
- Dividir sucesivamente el número decimal por la nueva base y registrar los residuos en orden inverso.
- Sumar el número decimal con potencias crecientes de la nueva base.
- Restar sucesivamente la nueva base del número decimal hasta llegar a cero.

4. **¿Cuál es el equivalente decimal del número binario 110101_2?**

- 21
- 29
- 53
- 105

5. ¿Cómo se realiza la suma de dos números binarios?

a) Sumando los dígitos de cada posición como en el sistema decimal, sin acarreo.

b) Sumando los dígitos de cada posición y acarreando 1 cuando la suma es 2 o mayor.

c) Multiplicando los dígitos correspondientes y sumando los resultados.

d) Realizando una operación OR lógica entre los dígitos correspondientes.

6. ¿Cuál es la utilidad principal de los complementos a 1 y a 2 en el sistema binario?

a) Simplificar la multiplicación y división de números binarios.

b) Facilitar la representación de números fraccionarios en binario.

c) Simplificar la resta de números binarios mediante la operación de suma.

d) Aumentar la capacidad de almacenamiento de números binarios en la memoria.

7. ¿Cuál es la base del sistema de numeración octal y cuál es su relación con el sistema binario?

a) Base 10; cada dígito octal corresponde a 4 bits binarios.

b) Base 8; cada dígito octal corresponde a 3 bits binarios.

c) Base 16; cada dígito octal corresponde a 4 bits binarios.

d) Base 8; cada dígito octal corresponde a 4 bits binarios.

8. ¿Cuál es el equivalente decimal del número hexadecimal $A3_{16}$? (Recuerda que A representa 10 en decimal)

a) 153

b) 163

c) 179

d) 195

9. ¿Cuál de las siguientes afirmaciones es verdadera?

a) Cada dígito octal representa 2 bits

b) Cada dígito hexadecimal representa 4 bits

c) El sistema binario es más compacto que el hexadecimal

d) El sistema decimal es el más usado en computadoras

10. ¿Cuál es la relación entre los sistemas de numeración y el procesamiento de datos en computadoras?

A) Sirven para graficar ecuaciones

B) Permiten el diseño de impresoras

C) Facilitan la representación y manipulación de información digital

D) Ayudan a optimizar motores

Referencia Bibliográfica

Lipschutz, S (1995). Matemáticas para Computación, Primera Edición. McGrawHill. Interamericana. ISBN: 970-10-0266-0 (BIC00013).

Jiménez, José (2014). Matemáticas para la Computación. Editorial Alfaomega México. Segunda Edición. ISBN : 978-607-707-719-0.

García Merayo, Felíx (2015). Matemática Discreta. España: Paraninfo. ISBN: 978-84-283-3568-3 (BIC01140).

Jiménez de Parga, C, (2024). Lógica matemática y computacional: teoría y ejercicios resueltos. Tébar Flores. ISBN: 978-84-7360-992-0.

Capítulo II
Aritmética del Computador

Introducción

En el ámbito de la computación, los números no solo representan cantidades abstractas, sino que constituyen la base sobre la cual se procesan, almacenan y comunican los datos. El capítulo "Aritmética del Computador" tiene como propósito fundamental introducir al lector en los conceptos clave que permiten comprender cómo los números son manipulados internamente por los sistemas digitales. Se explorarán temas esenciales como los números aproximados y los dígitos significativos, herramientas que permiten evaluar la precisión de los cálculos computacionales. Además, se estudiarán técnicas fundamentales como el redondeo y el truncamiento, procesos indispensables para limitar errores y controlar la exactitud en operaciones aritméticas.

También se analizará el uso del valor absoluto, así como las diferentes formas de representación numérica, entre ellas la notación científica, la notación exponencial normalizada y la forma exponencial binaria, todas ellas cruciales para expresar números extremadamente grandes o pequeños en entornos computacionales. Se abordarán las representaciones internas de números enteros y reales, mostrando cómo la arquitectura de una computadora influye en la manera en que interpreta los datos. Finalmente, se estudiará la aritmética de números enteros y reales, junto con la identificación y gestión de errores, elementos esenciales para garantizar la confiabilidad en los procesos numéricos dentro del entorno digital.

Este capítulo proporciona una base sólida para comprender los mecanismos internos del cálculo en los computadores, permitiendo al lector desarrollar un pensamiento analítico y preciso en el diseño de algoritmos y en la interpretación de resultados numéricos en programación y ciencias computacionales.

Objetivos de la unidad

- Asimilar la naturaleza de los números aproximados, reconociendo la importancia de los dígitos significativos en la representación de cantidades con precisión limitada.
- Dominar las diferentes formas de notación exponencial (científica, normalizada y binaria) para representar y comparar números de manera eficiente en entornos computacionales.
- Describir la representación interna de números enteros y reales en la memoria de la computadora, identificando las diferencias fundamentales entre ambas representaciones.
- Relacionar los conceptos de la aritmética del computador con los fundamentos de la matemática discreta, valorando su aplicación práctica en el análisis y resolución de problemas de la ciencia de la computación.

Preguntas de Enfoque

- ¿Por qué es importante distinguir entre números exactos y números aproximados en los cálculos computacionales?
- ¿Qué función cumplen los dígitos significativos al representar cantidades en contextos científicos y computacionales?
- ¿En qué situaciones se utiliza la notación científica o exponencial normalizada en el ámbito computacional?
- ¿Qué ventajas ofrece la notación binaria en la representación exponencial dentro de los sistemas de cómputo?
- ¿Qué tipos de errores pueden surgir en las operaciones aritméticas realizadas por un computador y cómo pueden minimizarse?
- ¿Cuál es la relación entre la teoría matemática y su implementación computacional en el manejo de números?

"El arte de la computación no está en los números que usamos, sino en cómo los representamos."
– ***Donald E. Knuth***

Números Aproximados y Dígitos Significativos

Figura 6: Introducción a la aritmética del computador
Fuente: Elaboración propia.

Cualquier dispositivo de cálculo puede manejar no solo un número limitado de cifras, para expresar ciertas magnitudes:

Peso	Olor
Estatura	Sabor
Edad	Grosor

Ejemplo: El peso de una hoja de papel en diferentes balanzas.

De ahí que el valor registrado en un dispositivo de cálculo será solo una aproximación del verdadero valor. La exactitud de un número aproximado A se mide por la cantidad de dígitos significativos que tiene A y decimos dígitos significativos por cuanto en ciertos casos algunos números utilizan los ceros únicamente como referencia, para colocar el punto decimal, sin que el nivel de aproximación sea superior (Lipschutz, 1995).

7,9mm a mts.	0,0079mts.	$7,9x10^{-3}m$
7,9mm a Km.	0,0000079Km.	$7,9x10^{-6}Km$
0,4832dm a Hm.	0,000000483Hm.	$4,83x10^{-7}Hm$
4,78cm. a Dm.	0,00478Dm.	$4,78x10^{-3}Dm$

Para determinar cuándo una cifra es significativa aplicamos las siguientes reglas:

1. Cualquier valor no nulo (esto es >0), siempre es significativo.

2. El valor cero siempre es significativo si se encuentra entre dos cifras no nulas.

$\sqrt{3} = 1{,}732\mathbf{0}5\mathbf{0}8\mathbf{0}8$

Los ceros si son significativos por encontrarse entre 2 cifras no nulas.

3. El cero nunca es significativo si esta precedido por cifras significativas o que tienen valor.

3,69**000**

En un número cualquiera $A = 72{,}9002$; se llama dígito **+ significativo** al primer dígito significativo desde la izquierda (7), se llama dígito **- significativo** al último dígito significativo de la derecha (2). El **total de cifras significativas** o número de dígitos significativos es el total de cifras que hay entre el dígito más significativo y el menos significativo, incluidos estos dos.

Número de cifras significativas de *A= (6)*

Ejercicio:

De los siguientes números determine el dígito más significativo, el menos significativo y el total de cifras significativas:

	+ Significativo	**- Significativo**	**# Cifras Significativas**
10034,96	1	6	7
80,000092	8	2	8
5,26000	5	6	3
0,008640	8	4	3
44500	4	5	3

Ejercicios propuestos:

De los siguientes números determine el dígito más significativo, el menos significativo y el total de cifras significativas:

- 768327823489327,000200324032
- 0,0000000000043204023042
- 1232,2340000000000000
- 0,00000000000023230000000000
- 32432,00002343423423
- 000000000000,123
- 000,23232435434000000
- 000,454540000000001
- 00000,1312312300002

Redondeo de Números

Redondeo: frecuentemente necesitamos expresar un número mediante una cantidad limitada de cifras, una forma de hacerlo es el redondeo, para lo cual se adopta un número determinado de cifras significativas. Existe la posibilidad de eliminar las cifras menos significativas, tomando el valor de una cifra como referente para el redondeo, la cual se le denomina **dígito de prueba o dígito de precisión**. Para el redondeo se utiliza el siguiente símbolo ≈que se lee aproximación o igual.

Cuando no se especifica la cantidad de posiciones decimales o fraccionarias de un número, se asume por defecto que el ejercicio va a contener 2 posiciones decimales o fraccionarias de precisión (Chapra & Canale, 2010).

$A = 7{,}94\,\mathbf{7}65$

dígito de prueba

Para lo cual dentro del redondeo se adoptan las siguientes reglas:

1. **Aproximación por defecto.**

   Si el dígito de prueba es menor que 5, simplemente se eliminan las cifras menos significativas.

   235,934962 ≈ 235,93

   dp

   Aprox.

   x

   Defecto

2. **Aproximación por exceso.**

   Si el dígito de prueba es mayor que 5 ó 5 seguido de cifras no nulas, se eliminan las cifras menos significativas del valor que queda y se suma 1 a la última cifra de la derecha.

   6,247451 ≈ 6,25

   dp

   Aprox.

   x

   Exceso

   25,9050012 ≈ 25,91

   dp

   Aprox.

   x

   Exceso

3. **Sume si es impar**

   Si el dígito de prueba es igual a 5 ó es 5 seguido de ceros igualmente se suma 1, solo si la cifra que queda es impar.

   76,835 ≈ 76,84

   dp

   Sume

   si es

   Impar

   Sume

   si es

   Impar

$$94{,}60\cancel{500} \approx 94{,}60$$

dp

Es importante mencionar que a partir de ahora si se suprime dígitos menos significativos de un número, es necesario definir que regla se ha empleado para dicho proceso.

Truncamiento

El truncamiento es otra forma de eliminar los dígitos menos significativos de un número, el cual consiste en eliminar dichos dígitos menos significativos sin alterar a los dígitos que quedan en su parte entera o en su parte fraccionaria.

$\pi = 3{,}1415\cancel{9265}4$
$\pi = 3{,}1416$ Redondeo por Aprox. x Exceso
$\pi = 3{,}1415$ Truncamiento

Valor Absoluto

Se llama valor absoluto de un número a aquel que se define independientemente de su signo y que se representa mediante dos barras que abarcan dicho número $|a| \Rightarrow$ y que se lee valor absoluto de a.

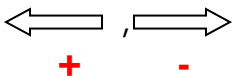
$|-7| = 7$

$|8-9| = 1$

$$\left|\frac{(a^3-b^3)}{x^2+2x-3}\right| = \left|\frac{(a-b)(a^2+ab+b^2)}{(x+3)(x-1)}\right| = \frac{(a-b)(a^2+ab+b^2)}{(x+3)(x-1)}$$

Formas Exponenciales

Existen números grandes o pequeños que requieren formas alternativas de notación o de representación, esto generalmente se expresa cuando el número es positivo o negativo.



En la forma exponencial cuando se recorre la coma o cuando se eliminan las cifras menos significativas de derecha a izquierda, el número de posiciones recorridas o eliminadas se adicionan al valor del exponente del número original y cuando se recorre la coma o cuando se eliminan las cifras menos significativas de izquierda a derecha, el número de posiciones recorridas o eliminadas se disminuyen al valor del exponente del número original (Burden, R. & Faires, 2011).

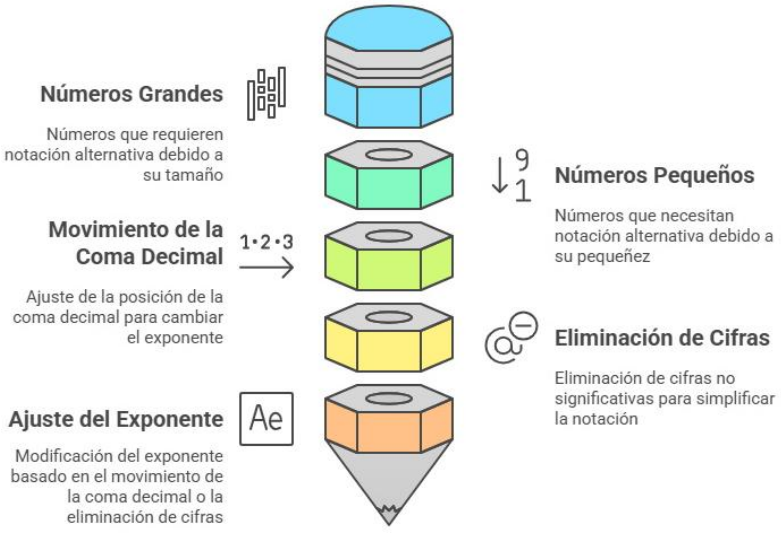


Figura 7: Introducción a las formas exponenciales
Fuente: Elaboración propia.

Ejemplo:

120’000.000 = 12000000 * 10^0

Eliminando 3 cifra menos significativas => 3bits de recorrido

120.000 x 10^3

Eliminando 6 cifra menos significativas => 6 bits de recorrido

120 * 10^6

Eliminando 7 cifra menos significativas y recorriendo la coma en toda su extensión:

0,12 * 10^9 => 9 bits de recorrido

Estas son formas exponenciales de las cuales hay dos formas de uso generalizado:

Notación Científica

En ella un número cualesquiera ***A*** se expresa en la forma:

$$A = m * 10^n$$

Donde ***m*** va a estar dentro del rango mayor o igual a 1 y menor a 10.

$$1 \leq m < 10$$

Y ***n*** es un número entero cualesquiera que puede ser positivo o negativo.

Otra forma de definir notación científica sería en eliminar las cifras menos significativas de un número o recorrer la coma hasta lograr en dicha cantidad la mínima expresión entera de un número, o sea 1,….. (cualquier valor); 2,….. (cualquier valor), etc.

Ejemplo:

	Notación Científica
14500	$1{,}45 \times 10^{4}$
1930	$1{,}93 \times 10^{3}$
6,93	$6{,}93 \times 10^{0}$
0,00013	$1{,}3 \times 10^{-4}$
0,023	$2{,}3 \times 10^{-2}$
0,16	$1{,}6 \times 10^{-1}$

Notación Exponencial Normalizada

En este caso también el número ***A*** se expresa de la forma:

$$A = m * 10^n$$

Pero en este caso el valor de ***n*** se llama mantisa y va en el rango mayor o igual a 0.1 y menor a 1.

$$0{,}1 \leq m < 1$$

Y en consecuencia se diferencia de la Notación Científica en una cifra más 1.

Otra forma de definir notación exponencial normalizada sería en eliminar las cifras menos significativas de un número o recorrer la coma hasta lograr en dicha cantidad la mínima expresión total de un número, o sea 0,….. (cualquier valor).

Ejemplo:

	Notación Científica	**Notación Exponencial Normalizada**
14500	$1{,}45 \times 10^4$	$0{,}145 \times 10^5$
1930	$1{,}93 \times 10^3$	$0{,}193 \times 10^4$
6,93	$6{,}93 \times 10^0$	$0{,}693 \times 10^1$
0,00013	$1{,}3 \times 10^{-4}$	$0{,}13 \times 10^{-3}$
0,023	$2{,}3 \times 10^{-2}$	$0{,}23 \times 10^{-1}$
0,16	$1{,}6 \times 10^{-1}$	$0{,}16 \times 10^0$

Forma Exponencial Binaria

Los números binarios también pueden ser normalizados, y siguen la misma regla anterior, con la única diferencia es que la base del sistema de numeración binario es 2.

	Forma Exponencial Binaria
$1011{,}101_2$	$0{,}1011101 \times 2^4$
$0{,}00011011_2$	$0{,}11011 \times 2^{-3}$
$1{,}101_2$	$0{,}1101 \times 2^1$
$0{,}1101_2$	$0{,}1101 \times 2^0$

Ejercicios propuestos:

De los siguientes números obtener la notación científica y la forma exponencial normalizada.

- 7683278234327,00020032
- 0,00000000432023042
- 1232,234000000000
- 0,0000002323000000
- 32432,0004323
- 00000000,123
- 000,232324300000
- 00,454000001
- 000,13100002

De los siguientes números obtener la notación o la forma exponencial binaria.

- $010101101010{,}0001_2$
- $000000001111110101{,}01_2$
- $0{,}0000010101111111_2$
- $1110111{,}110101111_2$
- $0{,}0000010111000_2$
- $11101010010111{,}11_2$
- $0{,}000111_2$

- 00001111000_2
- $11111111,1111_2$
- 111111111111_2
- 906904564
- 874872
- 9486645
- 1323313

Representación Interna

Los números internamente son representados mediante codificaciones binarias, para lo cual se utiliza un número fijo de bits, que recibe el nombre de "***longitud de palabra o localización de memoria***", la misma que es igual a **32 bits** de representación, en todos los casos; entonces analizaremos lo siguiente (Santamaría, 2013):

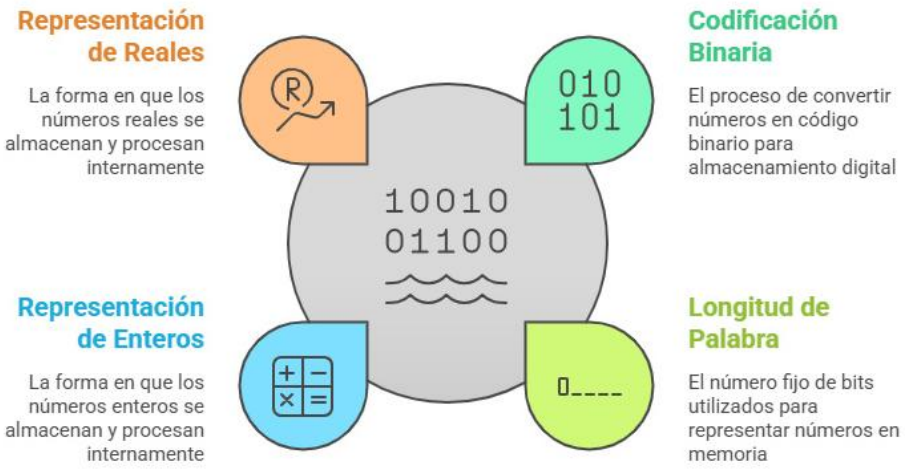


Figura 8: Representación interna de números
Fuente: Elaboración propia.

1.- Números enteros: $Z = \{-\infty..... -2,-1,0,1,2..... +\infty\}$

Llamado también de punto fijo, se representan internamente mediante su codificación binaria directa (forma binaria) si son positivos y mediante su complemento si son negativos, en la representación interna de un número entero primero van los bits de relleno más la codificación binaria directa. Ejemplo:

$$\mathbf{7953_{10}} \quad \Rightarrow \quad \mathbf{X_2}$$

7953	2
3976,5	1
1988	0
994	0
497	0
248,5	1
124	0
62	0
31	0
15,5	1
7,5	1
3,5	1
1,5	1
1	

$= \quad 1111100010001_2 = 13 \text{ bits} \Rightarrow 32 - 13 = 19 \text{ bits de relleno}$

+7953 0000000000000000000111110001000 1

Bits de relleno — Codificación Binaria Directa

Longitud de palabra

- 7953 11111111111111111110000011101110

Bits de relleno — Complemento

Longitud de palabra

Ejercicios propuestos:

Obtener la representación interna de los siguientes números enteros.

- 7674872
- 845932
- 9023845
- 87365894
- 906564

- 874772
- 94869845
- 13233123
- 9987434
- 11110111

2.- Números reales: R = {-∞,……,0,……..+∞}

Se presentan internamente mediante su forma exponencial y dividen a la localización de memoria en tres campos:

***Primer Campo*:** Tiene una longitud de 1 bit y sirve para representar el signo del número, cuando es positivo hacemos referencia al 0 y si es negativo al 1.

***Segundo Campo*:** Tiene una longitud de 7 bits y en él colocamos la representación interna del exponente.

***Tercer Campo*:** Tiene una longitud de 24 bits y sirve para representar la mantisa del número, que no es más que la parte fraccionaria del binario normalizado.

1 bit	7 bits	24 bits
SIGNO	**EXPONENTE**	**MANTISA**
+ = 0	+ = 0 CBD	1.- Codificación Binaria Directa
- = 1	- = 1 Complemento	2.- Bits de Relleno

Para representar el campo del exponente hay 2 formas, la cual deriva a la posibilidad de tener dos representaciones internas de un número real :

1. Codificación Binaria Directa CBD

Mediante la Codificación Binaria Directa del exponente si es positivo o mediante su complemento si es negativo.

Ejercicio: Obtener la representación interna del siguiente número real:

-3754

- **Primer paso**: convertir a binario el número

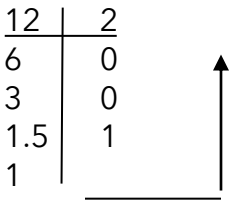
-3754	2	
1877	0	
938.5		1
469	0	
234.5		1
117	0	
58.5		1
29	0	
14.5		1
7	0	
3.5	1	
1.5	1	
	1	

$-3754 = 111010101010_2$

- **Segundo paso**: Normalizamos (forma exponencial binaria).

111010101010_2 => $0{,}111010101010 * 2^{12}$

- **Tercer paso**: Representación del exponte mediante CBD



$12 = 1100_2$

- **Cuarto paso**: Ubicamos los valores en los campos de la localización de memoria

Signo	Exponente	Mantisa
1	00011100	11101010101000000000000

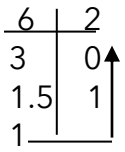
Ejercicio: Obtener la representación interna del siguiente número real:

$0{,}0000001101111_2$

En vista que el número ya se encuentra en el sistema de numeración binario, directamente normalizamos:

Normalizando: $0{,}1101111011 \times 2^{-6}$

Exponente:



$6 \Rightarrow 110_2$ 0000 = bits de relleno
1111 = complemento de los bits de relleno
0000110 codificación binaria directa cuando es +
1111001 mediante su complemento cuando es -

Signo	Exponente	Mantisa
0	1111001	11011110110000000000000

2. Representación del campo del exponente mediante la Característica

La mayoría de los ordenadores en el campo del exponente colocan la **característica**, la misma que es igual a:

$$N + 2^{t-1}$$

En dónde **N** es el verdadero valor del exponente en la forma normalizada, **t** es igual a la longitud del campo del exponente, entonces como en nuestro caso, siempre la longitud del campo del exponente es igual a siete, por lo que vamos a obtener siempre una constante.

$$2^{t-1} = 2^{7-1} = 2^{6} = 64$$

$$\text{CARACTERISTICA} = N + 64$$

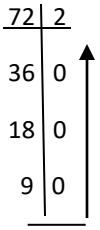
Ejercicio: Utilizando el método de la característica representar el siguiente número real:

$11011100{,}1101_2$

Normalizando nos quedaría: $0{,}110111001101 * 2^8$

Exponente utilizando la característica: $N+64=8+64=72$

$72 \Rightarrow X_2 = 1001000_2$



Signo	Exponente	Mantisa
0	1001000	11011100110100000000000

Ejercicio: Hallar la representación interna del siguiente número usando la característica.

$0{,}000000000111011100010 1_2$

Normalizando: $0{,}1110111000101 * 2^{-9}$

Exponente: $N+64=-9+64=55$

$55 \Rightarrow X_2 = 110111_2$

Signo	Exponente	Mantisa
0	0110111	11101110001010000000000

Ejercicio: Hallar la representación interna usando el método BCD y la característica.

$-0{,}000001111001101_2$

a) Exponente Binario

Normalizando: $0{,}1111001101 * 2^{-5}$

Exponente:

5	2
2.5	1
1	0

$5 = 101_2$ 0000 = bits de relleno

1111 = complemento de los bits de relleno

0000101 codificación binaria directa cuando es +

1111010 mediante su complemento cuando es -

Signo	Exponente	Mantisa
1	1111010	11110011010000000000000

b) Característica

Normalizando: $0{,}1111001101 \times 2^{-5}$

Exponente: $N + 64 = -5 + 64 = 59$

$55 \Rightarrow X_2 = 111011_2$

59	2
29,5	1
14,5	1
7	0

Signo	Exponente	Mantisa
1	0111011	11110011010000000000000

Ejercicios propuestos:

Obtener la representación interna de los siguientes números reales (empleando las 2 formas de representación).

- 7674872
- 84596432
- 90238345
- 8736034
- 9004564
- 874872
- 948645
- 132323
- 9987434
- 1111111
- $0{,}001010101111_2$
- $10101111111101010101111111_2$
- $1111111100011111{,}1110010_2$
- $10010101010101010010000000_2$
- $0{,}000000010101111111111_2$
- $000000001111111010101{,}001_2$
- $0{,}000000000000000001_2$
- 10100000000000000000_2
- $0{,}000000100001010101011_2$
- $11111101111110000000000_2$

Aritmética del Computador

Generalmente dependiendo del lenguaje de programación, el tratamiento que se da a las operaciones aritméticas con números enteros y reales es diferente.

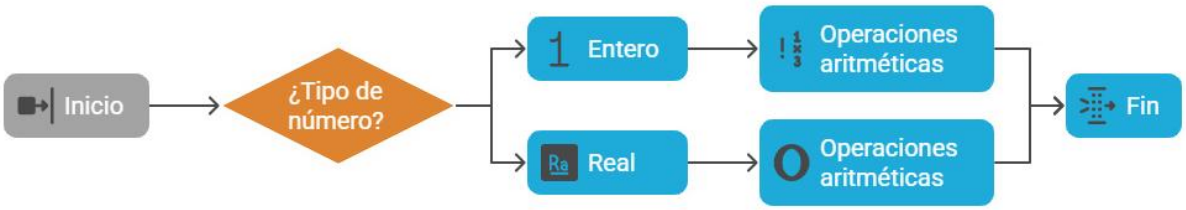


Figura 9: Tratamiento de operaciones aritméticas
Fuente: Elaboración propia.

Aritmética de números enteros

Existe una propiedad llamada de cerradura, según la cual una operación aritmética entre dos números del mismo tipo se resuelve de la siguiente manera:

Suma entera: Si se suma dos números enteros, se obtiene como resultado otro número entero. Z + Z = Z

Ejemplo:

- ✓ En procesamiento de imágenes digitales, los valores de intensidad de cada píxel (números enteros de 0 a 255) se suman al aplicar filtros o mejorar el brillo.

- ✓ En blockchain y criptomonedas, las sumas enteras son fundamentales para actualizar saldos de transacciones en tiempo real.

Resta entera: Se cumple la ley de la cerradura. Z - Z = Z.

Ejemplo:

- ✓ En realidad aumentada, la resta de coordenadas enteras se utiliza para calcular la posición relativa de objetos en una escena virtual.

- ✓ En sistemas de control industrial, la resta se emplea para determinar el error entre un valor medido y un valor de referencia, base de los algoritmos PID.

Producto entero: Siempre un producto de dos números enteros nos da otro número entero. Z * Z = Z.

Ejemplo:

- ✓ En redes neuronales artificiales, las operaciones de multiplicación de pesos (enteros o representados en punto fijo) con entradas forman la base del cálculo en hardware optimizado.

- ✓ En compresión de audio y video, los productos enteros se aplican en transformadas rápidas (como la DCT en JPEG o MP3) para manipular grandes volúmenes de datos de manera eficiente.

División entera: Es una operación aritmética parcialmente definida entre dos números enteros, pues no siempre se cumple que:

$$\frac{A+B}{C} = \frac{A}{C} + \frac{B}{C}$$

Ejemplo:

- ✓ En criptografía moderna, la división modular se utiliza en algoritmos como RSA para calcular claves seguras.

- ✓ En computación en la nube, los sistemas de planificación de recursos (ej. dividir cargas de trabajo entre servidores) aplican divisiones enteras para distribuir tareas en lotes discretos.

Aritmética de números reales

La operación con los números reales se efectúa en sus formas exponenciales normalizadas, así:

Suma Real: En la suma se distinguen dos casos:

1.- Cuando los sumandos tienen igual exponente, se suman las mantisas y se coloca al resultado el mismo exponente, finalmente se trunca y se redondea a P dígitos de precisión.

Ejemplo:

$0,4328 * 10^3 + 0,6473 * 10^3$

$$\begin{array}{rl} & 0,6473 \\ + & 0,4328 \\ \hline & 1,0801 \end{array} \Longrightarrow \quad 1,0801 * 10^3$$

$Normalización \Rightarrow 0,10801 \cdot 10^4$

$\text{Re}dondeo \Rightarrow 0,11 \cdot 10^4$ Aproximación por Exceso

$Truncamiento \Rightarrow 0,10 \cdot 10^4$

2.- Si tienen distinto exponente, se reajusta uno de los exponentes para llevarlo al otro valor y luego se aplica la regla anterior.

a. Cuando convierto de exponente mayor a exponente menor.
b. Cuando convierto de exponente menor a exponente mayor.

Ejemplo:

$0,2573 * 10^5 + 0,3680 * 10^2$

a) Exponente > a exponente <

$257,3 * 10^2 + 0,3680 * 10^2$

$$\begin{array}{rl} & 257,3 \\ + & 0,3680 \\ \hline & 257,6680 * 10^2 \end{array}$$

$Normalización \Rightarrow 0{,}2576680 * 10^5$
$Redondeo \Rightarrow 0{,}26 * 10^5$ Aproximación por Exceso
$Truncamiento \Rightarrow 0{,}25 * 10^5$

b) Exponente < a Exponente >

$0{,}2573 * 10^5 + 0{,}0003680 * 10^5$

$$\begin{array}{rl} & 0{,}2573 \\ + & 0{,}0003680 \\ \hline & 0{,}2576680 * 10^5 \end{array}$$

$Normalización \Rightarrow 0{,}2576680 * 10^5$
$Redondeo \Rightarrow 0{,}26 * 10^5$ Aproximación por Exceso
$Truncamiento \Rightarrow 0{,}25 * 10^5$

Ejemplos de Aplicaciones Actuales:

- En simulaciones climáticas, la suma de grandes cantidades de datos reales (temperaturas, niveles de CO_2, humedad) permite modelar escenarios de cambio climático.

- En economía digital, plataformas de comercio electrónico suman montos reales (precios con decimales, impuestos, descuentos) para calcular el costo final de transacciones en línea.

Resta Real: Se tienen los mismos casos que en la suma de números reales y se procede de igual manera, con la única diferencia que se restan las mantisas.

1.- Cuando tienen = Exponente

$0{,}6528 * 10^{-3} + 0{,}6319 * 10^{-3}$

$$\begin{array}{rll} & 0{,}6528 & \\ - & 0{,}6319 & \\ \hline & 0{,}0209 \Rightarrow & 0{,}0209 * 10^{-3} \end{array}$$

$Normalización \Rightarrow 0{,}209 * 10^{-4}$
$Redondeo \Rightarrow 0{,}21 * 10^{-4}$ Aproximación por Exceso
$Truncamiento \Rightarrow 0{,}20 * 10^{-4}$

2.- Cuando tienen ≠ Exponente

$0{,}7825 * 10^{1} - 0{,}2319 * 10^{-1}$

a) Exponente > a Exponente <

$78{,}25 * 10^{-1} + 0{,}2319 * 10^{-1}$

$$\begin{array}{r} 78{,}25 \\ - \quad 0{,}2319 \\ \hline 78{,}0181 * 10^{-1} \end{array}$$

$Normalización \Rightarrow 0{,}780181 * 10^{1}$
$Redondeo \Rightarrow 0{,}78 * 10^{1}$ Aproximación por Defecto
$Truncamiento \Rightarrow 0{,}78 * 10^{1}$

b) Exponente < a Exponente >

$0{,}7825 * 10^{1} + 0{,}002319 * 10^{1}$

$$\begin{array}{r} 0{,}7825 \\ - \quad 0{,}002319 \\ \hline 0{,}780181 * 10^{1} \end{array}$$

$Normalización \Rightarrow 0{,}780181 * 10^{1}$
$Redondeo \Rightarrow 0{,}78 * 10^{1}$ Aproximación por Defecto
$Truncamiento \Rightarrow 0{,}78 * 10^{1}$

Ejemplos de Aplicaciones Actuales:

- En gráficos por computadora, la resta de coordenadas reales permite calcular desplazamientos, trayectorias y animaciones en entornos 3D.

- En biomedicina, la resta de señales electrocardiográficas o de imágenes médicas sirve para identificar diferencias entre registros y detectar anomalías.

Multiplicación Real: Aquí no es importante que los exponentes sean iguales, únicamente multiplicamos las mantisas y sumamos los exponentes para obtener el resultado. Si hace falta normalizamos y finalmente redondeamos y/o truncamos a P dígitos de precisión.

Ejemplo:

$0{,}2679 * 10^{-5} \quad * \quad 0{,}9645 * 10^{3}$

$$\begin{array}{rl} & 0{,}2679 \qquad \text{Exponentes: } -5 + 3 = -2 \\ * & 0{,}9645 \\ \hline & 0{,}25838955 * 10^{-2} \end{array}$$

$Normalización \Rightarrow 0{,}25838955 * 10^{-2}$
$Redondeo \Rightarrow 0{,}26 * 10^{-2}$ Aproximación por Exceso
$Truncamiento \Rightarrow 0{,}25 * 10^{-2}$

Ejemplos de Aplicaciones Actuales:

- En aprendizaje automático (machine learning), las operaciones de multiplicación con números reales son esenciales en el cálculo de probabilidades y en la actualización de parámetros de modelos estadísticos.

- En ingeniería aeroespacial, productos de números reales permiten calcular trayectorias de satélites o vehículos autónomos en tiempo continuo.

División Real: Se dividen las mantisas y se restan los exponentes para obtener la respuesta. Si hace falta normalizamos y finalmente redondeamos y/o truncamos a P dígitos de precisión.

Ejemplo:

$0{,}9318 * 10^{-4} / 0{,}1653 * 10^{2}$

$$\begin{array}{rl} & 0,\ 9318 \qquad \text{Exponentes: } -4 - 2 = -6 \\ / & 0,\ 1653 \\ \hline & 5{,}637023593 * 10^{-6} \end{array}$$

$Normalizaación \Rightarrow$ 0, 5637023593 * 10^{-5}
$Redondeo \Rightarrow$ 0,56 * 10^{-5} Aproximación por Defecto
$Truncamiento \Rightarrow$ 0,56 * 10^{-5}

Ejemplos de Aplicaciones Actuales:

- En procesamiento de imágenes, la división real se aplica al normalizar valores de píxeles (por ejemplo, dividir entre 255 para llevarlos al rango 0–1 en redes neuronales).

- En finanzas algorítmicas, se utiliza para calcular indicadores como el retorno sobre inversión (ROI) o ratios de riesgo/beneficio en tiempo real.

Errores

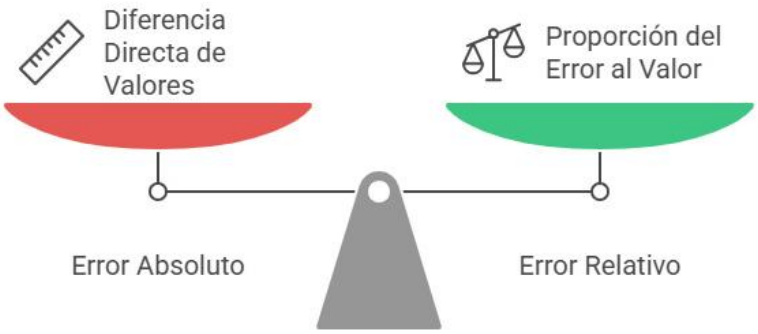


Figura 10: Comparando métricas de error en cálculos
Fuente: Elaboración propia.

Todo dispositivo de cálculo almacena un número limitado de cifras, lo que quiere decir que los valores almacenados en realidad, en muchos casos son aproximaciones de los verdaderos valores, ello da pie a la existencia de los siguientes valores:

A = valor real

A'= valor aproximado

Se llama **ERROR ABSOLUTO** a la diferencia que existe entre el valor real y el valor aproximado.

$$e = A - A'$$

Ejemplos de Aplicaciones Actuales:

- En medicina digital, al comparar la medida real de la glucosa en sangre (120.4 mg/dL) con la medición aproximada de un sensor portátil (119.8 mg/dL), el error absoluto es 0.6 mg/dL.

- En visión por computadora, al detectar la posición real de un objeto (100.0 px en el eje X) frente al valor detectado por un algoritmo (98.7 px), el error absoluto es 1.3 px.

Se llama **ERROR RELATIVO** al cociente que existe entre el error absoluto y el valor real.

$$r = \frac{e}{A} = \frac{A - A'}{A}$$

El error relativo es tan pequeño que es preferible expresarlo en términos de porcentajes. El porcentaje una vez declarado nunca puede ser negativo.

Ejemplos de Aplicaciones Actuales:

- En cálculo de trayectorias de drones, si la posición real es 250.0 m y el valor aproximado es 249.5 m, el error absoluto es 0.5 m, y el error relativo es 0.5/250 = 0.002 (0.2%).

- En finanzas algorítmicas, al predecir el precio de una acción (valor real: 87.50 USD, estimado: 87.00 USD), el error absoluto es 0.50 USD, mientras que el error relativo es 0.50/87.50 = 0.0057 (0.57%).

Sea **A = 0,9284567**

Obtener el error absoluto, el error relativo, truncado A' donde P = 4.

A'= 0,9284

e = A –A' = 0,9284567- 0,9284 = 0,0000567

$$r = \frac{e}{A} = \frac{0,0000567}{0,9284567} = 0,000061069$$ 0,000061069 * 100 = 0,0061069 %

La representación de números enteros y reales no solo constituye un fundamento teórico, sino que también tiene aplicaciones directas en el desarrollo de inteligencia artificial y aprendizaje automático, donde los algoritmos requieren optimizar la precisión de los cálculos numéricos en contextos de big data. Además, la elección de representaciones numéricas eficientes es clave en tecnologías como la computación en la nube y el procesamiento de gráficos en 3D, donde la velocidad y exactitud en la manipulación de datos determinan el rendimiento de los sistemas.

Resumen del capítulo

Este capítulo aborda los fundamentos de la aritmética utilizada en los sistemas computacionales, comenzando con la distinción entre números exactos y aproximados, y el análisis de dígitos significativos, redondeo y truncamiento, elementos esenciales para manejar la precisión en los cálculos. Se estudia el valor absoluto como base para comprender errores y desviaciones, así como las diferentes formas de representar los números mediante notación científica, exponencial normalizada y binaria. Además, se examina cómo los computadores representan internamente tanto números enteros como reales, y cómo se efectúan operaciones aritméticas dentro del hardware. Finalmente, se analizan los errores que pueden surgir durante el procesamiento numérico, destacando su impacto en la confiabilidad de los resultados computacionales.

Conclusiones

- La precisión numérica es clave en el procesamiento digital. La distinción entre números exactos y aproximados, junto con el uso de dígitos significativos, redondeo y truncamiento, permite controlar la precisión en los cálculos computacionales y minimizar errores.

- La representación interna de números influye en la eficiencia del sistema. Los números enteros y reales se codifican en la memoria mediante estructuras binarias específicas, como la codificación directa, el complemento y la característica, lo que determina cómo se almacenan y procesan los datos.

- La aritmética computacional requiere normalización y gestión de errores Las operaciones con números reales se realizan en forma exponencial normalizada, y es fundamental aplicar técnicas de redondeo, truncamiento y análisis de errores absolutos y relativos para garantizar resultados confiables.

Autoevaluación / Cuestionario

Selecciona la opción correcta

1. ¿Qué se entiende por dígitos significativos en un número aproximado?

A) Los ceros a la izquierda del número
B) Todos los dígitos después del punto decimal
C) Los dígitos que aportan información real sobre la precisión del número ★
D) Los dígitos que se repiten en una secuencia

2. ¿Cuál es la principal diferencia entre redondeo y truncamiento?

A) El redondeo elimina todos los ceros
B) El truncamiento siempre aumenta el valor
C) El redondeo ajusta el valor según el dígito siguiente, el truncamiento simplemente

corta ★
D) No hay diferencia, son sinónimos

3. ¿Qué representa el valor absoluto de un número?

A) Su valor negativo
B) Su valor sin signo ★
C) Su valor en binario
D) Su valor en notación científica

4. ¿Cuál de las siguientes es una forma válida de notación científica?

A) 0.000123
B) 123 × 10^0
C) 1.23×10^{-4} ★
D) 123.0

5. ¿Qué característica define a la notación exponencial normalizada?

A) El exponente siempre es positivo
B) La base es 2
C) El número tiene un solo dígito distinto de cero antes del punto decimal ★
D) Se usa solo para números enteros

6. ¿Qué base se utiliza en la forma exponencial binaria?

A) Base 10
B) Base 8
C) Base 16
D) Base 2 ★

7. ¿Cuál es una limitación de la representación interna de números reales en los computadores?

A) No se pueden representar números negativos
B) Solo se pueden representar números enteros
C) No todos los números reales pueden representarse exactamente ★
D) Los números reales no pueden sumarse

8. ¿Qué estándar se utiliza comúnmente para representar números reales en los computadores?

A) ASCII
B) IEEE 754 ★
C) UTF-8
D) ISO 9001¿Cuál de las siguientes afirmaciones es verdadera?

9. ¿Qué tipo de error ocurre cuando se pierde precisión al representar un número?

A) Error lógico
B) Error de sintaxis
C) Error de redondeo ★
D) Error de ejecución

10. ¿Cuál de los siguientes conceptos ayuda a medir la magnitud de un error?

A) Notación binaria
B) Valor absoluto ★
C) Truncamiento
D) Dígitos significativos

Referencia Bibliográfica

Lipschutz, S (1995). Matemáticas para Computación, Primera Edición. McGrawHill. Interamericana. ISBN: 970-10-0266-0 (BIC00013).

Chapra, S. & Canale, R. (2010). Métodos numéricos para ingenieros (6.ª ed.). McGraw-Hill.

Burden, R. L. & Faires, J. D. (2011). Análisis numérico (9.ª ed.). Cengage Learning.

Santamaría, P. (2013). Representación de los números en la computadora (Versión 1.8.1). Facultad de Ciencias Astronómicas y Geofísicas, Universidad Nacional de La Plata.

Capítulo III
LÓGICA Y TABLAS DE VERDAD

Introducción

La lógica proposicional es la base del razonamiento formal y una herramienta fundamental en la computación, las matemáticas y la filosofía. En este capítulo exploraremos cómo las proposiciones, operadores lógicos y estructuras argumentativas se utilizan para representar y analizar afirmaciones complejas. Iniciaremos con definiciones básicas que nos permitirán identificar y manipular proposiciones, para luego estudiar operadores como la conjunción, disyunción, negación, condicional y bicondicional, junto con sus respectivas tablas de verdad. También abordaremos conceptos fundamentales como tautologías, contradicciones y contingencias, que ayudan a clasificar el valor lógico de una expresión. A través del estudio de la equivalencia lógica y las leyes del álgebra de proposiciones, aprenderemos a simplificar y transformar expresiones lógicas. Finalmente, analizaremos la validez de argumentos mediante técnicas de deducción formal, incluyendo las formas recíproca, inversa y contrarrecíproca de las proposiciones condicionales. Este capítulo sienta las bases para comprender cómo las computadoras "piensan" lógicamente, y cómo podemos diseñar algoritmos y estructuras de decisión claras y efectivas.

Objetivos de la unidad

- Asimilar los conceptos esenciales de la lógica proposicional, incluyendo proposiciones, conectivos lógicos y estructuras básicas de razonamiento, para fundamentar el análisis lógico en contextos computacionales.
- Construir y analizar tablas de verdad que permitan evaluar con precisión el valor lógico de proposiciones simples y compuestas.
- Aplicar e interpretar las leyes del álgebra de proposiciones, así como los conceptos de tautología, contradicción y contingencia, en la resolución de problemas lógicos.

- Determinar la validez de argumentos formales, utilizando reglas de inferencia y equivalencias lógicas para evaluar razonamientos en contextos computacionales.

Preguntas de Enfoque

- ¿Qué papel juega la lógica proposicional en la construcción del pensamiento computacional?
- ¿Cómo se relacionan los operadores lógicos con las operaciones básicas de la computación digital?
- ¿De qué manera una tabla de verdad permite determinar si una proposición es una tautología o una contradicción?
- ¿Cómo se puede demostrar que dos proposiciones son lógicamente equivalentes?
- ¿Cómo se estructura un argumento lógico válido y cuáles son sus elementos fundamentales?
- ¿Qué utilidad tiene el análisis lógico en la toma de decisiones dentro de sistemas computacionales?

"La lógica es el arte de razonar bien: un lenguaje universal que le da estructura a la verdad."
– ***Gottlob Frege***

Lógica y Tablas de Verdad

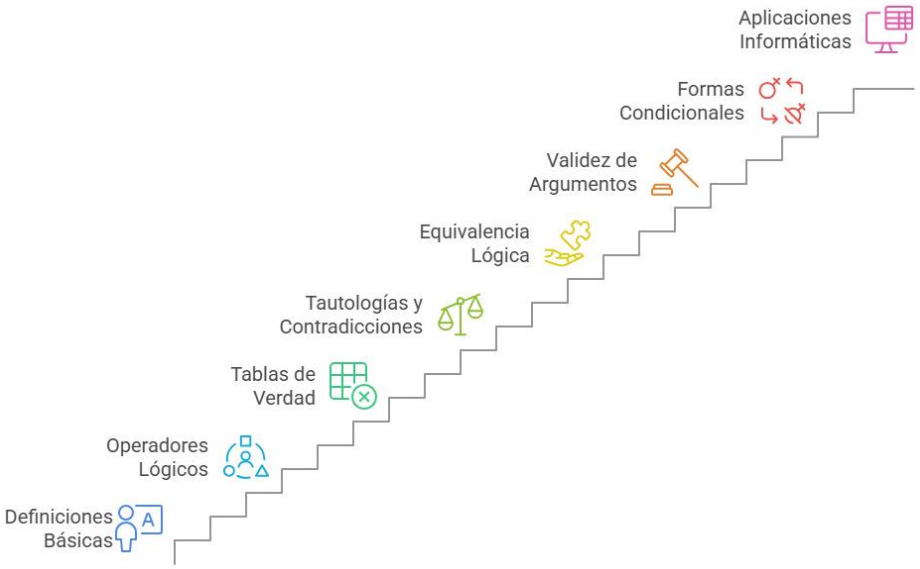


Figura 11: Camino del álgebra proposicional
Fuente: Elaboración propia.

Un ordenador se diferencia de las calculadoras comunes y corrientes, ya que en ella es posible realizar uno u otro evento según sean las condiciones y/o necesidades del usuario.

Definiciones

Juicio: Es un pensamiento que afirma o niega algo.

Enunciado: Es la expresión verbal o escrita de un juicio.

Razonamiento: Es la inferencia (deducción) de un enunciado a partir del análisis de varios enunciados llamados premisas.

Proposición: Es un enunciado que forma parte de un razonamiento y es siempre verdadero o falso (García, 2015).

Ejemplo:

2 + 7 = 6
José es ingeniero de sistemas

Mi dirección es Juan León Mera 4-40 y Vargas Machuca

Existen la siguiente clasificación de proposiciones:

Numéricos
Alfabéticas
Alfanuméricos

Valor de Verdad: Como toda proposición tiene un valor, a esto se le conoce como valor de verdad.

Proposición Simple: Es aquella que posee una sola afirmación o negación.

Ejemplo:

2 + 7 = 6	F
Juan es ingeniero de sistemas	V

Proposición Compuesta: Es aquella que está formada por varias proposiciones simples, enlazadas mediante operadores o conectivos lógicos.

Ejemplo:

Juan es ingeniero de sistemas **y** le gusta el fútbol

Operadores Lógicos

Figura 12: Operadores lógicos
Fuente: Elaboración propia.

Dentro del uso de operadores lógicos tenemos:

2^n ➔ *Me sirve para determinar el número de valores de verdad del ejercicio.*

Donde ***n*** es el número proposiciones que tiene el ejercicio (Lipschutz, 1995).

Ejemplo: cuando hay dos proposiciones q y p.

Se asignan en orden alfabético así: 4/2 = cada 2 valores se asignan a p
2/2 = cada 1 valor se asigna a q

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

$2^n = 2^2 = 4$ filas de valores de verdad

Conjunción

Se representa por $\wedge$, se lee "y", y en los lenguajes de programación se escribe "and".
En la conjunción se obtiene un valor de verdad **verdadero** si y solo si las dos o más proposiciones poseen un valor de verdad verdadero. Su tabla de verdad es la siguiente:

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

Disyunción

En la disyunción encontramos dos clases:

1) **Disyunción Inclusiva:** Se representa con $\vee$, y se lee "o" y en los lenguajes de programación se escribe OR.
   En la disyunción inclusiva se obtiene un valor de verdad **falso** si y solo si las dos o más proposiciones poseen un valor de verdad falso, y su tabla de verdad es la siguiente:

p	***q***	***p* v *q***
V	V	V
V	F	V
F	V	V
F	F	F

2) **Disyunción Exclusiva**

   Se representa por el símbolo $\vee$, se lee "ó" y en los lenguajes de programación se escribe XOR.

En la disyunción exclusiva se obtiene un valor de verdad **verdadero** si y solo si únicamente cuando una de las dos o más proposiciones posee un valor de verdad verdadero. Su tabla de verdad es la siguiente:

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

Negación

Se representa por el símbolo **~** o **¬,** se lee "No", en los lenguajes de programación se escribe NOT. La negación consiste en negar la afirmación original o la proposición dada.

p	$\sim p$
V	F
F	V

Tabla de Jerarquía: Nos indica que el orden de ejecución es el siguiente:

1. ~
2. $\wedge$
3. $\vee$
4. $\vee$
5. $\rightarrow$
6. $\leftrightarrow$

Ejercicio:

$$[(\sim\sim p \vee q \wedge r) \wedge (\sim r \wedge \sim p) \vee (s)]$$

Proposiciones => *p,q,r,s* *n = 4*

$2^4 = 16$ (*p*) 16/2 = 8 (*q*) 8/2 = 4 (*r*) 4/2 = 2 (s) 2/1 = 1

[(~	~	*P*	∨	(*q*	∧	*r*))	∧	(~	*r*	∧	~	*p*)	∨	(*s*)]
V	F	V	F	V	V	V	F	F	V	F	F	V	V	V
V	F	V	F	V	V	V	F	F	V	F	F	V	F	F
V	F	V	F	V	F	F	F	V	F	F	F	V	V	V
V	F	V	F	V	F	F	F	V	F	F	F	V	F	F
V	F	V	V	F	F	V	F	F	V	F	F	V	V	V
V	F	V	V	F	F	V	F	F	V	F	F	V	F	F
V	F	V	V	F	F	F	F	V	F	F	F	V	V	V
V	F	V	V	F	F	F	F	V	F	F	F	V	F	F
F	V	F	V	V	V	V	F	F	V	F	V	F	V	V
F	V	F	V	V	V	V	F	F	V	F	V	F	F	F
F	V	F	V	V	F	F	V	V	F	V	V	F	V	V
F	V	F	V	V	F	F	V	V	F	V	V	F	F	F
F	V	F	F	F	F	V	F	F	V	F	V	F	V	V
F	V	F	F	F	F	V	F	F	V	F	V	F	F	F
F	V	F	F	F	F	F	F	V	F	V	V	F	V	V
F	V	F	F	F	F	F	F	V	F	V	V	F	F	F

Ejercicios propuestos:

- [{(<~~~~~~~~ p ∨ ~~~~~ q ∧ ~~~~r >) ∨ (r ∧ ~~~ r ∨ ~~~~r) ∧ ~~~q} ∨ {(~~~q ∧ ~~~ q ∨ q)} ∨ ~p]
- [{(<~~ q ∨ ~~~~~ s ∧ ~~~~p >) ∨ (r ∧ ~~~ q ∨ ~~~~s) ∧ ~~~q} ∧ {(~~~r ∧ q ∨ q) ∨ ~p ∧ ~~~q}]
- [{(~~~~p ∨ r ∧ ~~~ r ∨ ~~~~r) ∧ (~~~r ∧ q ∧ ~~~ q ∨ q) ∨ ~~~~~ s ∨ r ∧ ~~~ q) ∨ p ∨ ~~~~~ s}]

- [{(<~~~~~~~~ q∨ ~~~~~ r) ∧ ~~~~~~~~~~~~~~q} ∨ {(~~~~~~~~~p ∧ ~~~~~~~~~ p ∨ ~~~~~~q)}]

- [{({(~~~r ∧ ~~~ q ∨ ~~~~~~~~~p) ∧ (~~~s ∧ ~~~~~~~t ∧ ~~~ q ∨ ~~~~~~~~~q)} ∨ ~p]

Tautología y Contradicción

Dadas dos proposiciones que se analizan y cuando la columna de resultado es verdadera, se dice que es una Tautología; y cuando la columna de resultados es falsa, se denomina Contradicción o Falacia.

p	∨	~	p
V	V	F	V
F	V	V	F

Tautología

p	∧	~	p
V	F	F	V
F	F	V	F

Falacia

Cuando obtenemos verdadero y falso en la columna de resultado no son ni Tautología, ni Falacia.

Equivalencia Lógica y Algebra de Proposiciones

Equivalencia Lógica: Dos proposiciones compuestas como:

$$P(p,q,r,s,...) \qquad Q(p,q,r,s,...)$$

Se llaman lógicamente equivalentes si para igual valor de sus componentes, sus resultados son idénticos y se indica por $P \equiv Q$, que se lee P es equivalente con Q.

Demostrar: ~ (p ∧ q) ≡ ~ p ∨ ~ q

~	(p	∧	q)
F	V	V	V
V	V	F	F
V	F	F	V
V	F	F	F

~	p	∨	~	q
F	V	F	F	V
F	V	V	V	F
V	F	V	F	V
V	F	V	V	F

$$\text{——————} \equiv \text{——————}$$

Estas equivalencias se agrupan en una serie de leyes denominadas, **leyes del algebra de Proposiciones** (Jiménez de Parga, 2024).

Leyes del algebra de proposiciones:

1. Idempotencia:
   $p \wedge p \equiv p$
   $p \vee p \equiv p$

2. Asociativa
   $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$
   $p \vee (q \vee r) \equiv (p \vee q) \vee r$

3. Conmutativa
   $p \wedge q \equiv q \wedge p$
   $p \vee q \equiv q \vee p$

4. Distributiva
   $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
   $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$

5. Identidad
   $p \wedge f \equiv f$
   $p \vee f \equiv p$
   $p \wedge t \equiv p$
   $p \vee t \equiv t$

6. Complemento
   $p \wedge \sim p \equiv f$
   $p \vee \sim p \equiv t$
   $\sim f \equiv t$
   $\sim t \equiv f$

7. Involución:
   $\sim\sim p \equiv p$

8. Ley de Morgan:
   $\sim (p \wedge q) \equiv \sim p \vee \sim q$
   $\sim (p \vee q) \equiv \sim p \wedge \sim q$

Enunciado Condicional y Bicondicional

Muchos enunciados dentro del habla común o dentro de ciencias como las matemáticas, informática, utilizan el enunciado de la forma: **si *p* entonces *q***; el cual se denota por: $p \rightarrow q$ y se lee **p solo si q**; estos enunciados reciben el nombre de **condicional**. También el enunciado condicional se lee *p* implica *q*

Donde *p* toma el nombre de antecedente y *q* el de consecuente.

En un condicional se obtiene un valor de verdad falso, solo si el antecedente es verdadero y el consecuente es falso, y su tabla de verdad es la siguiente (Jiménez, 2014):

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

Otra forma común de enunciado es p si y solo si q, la misma que implica una doble dirección del condicional y cuya tabla de verdad es la siguiente:

p si y solo si *q* ⟹ $p \rightarrow q \wedge q \rightarrow p$

$(p$	$\rightarrow$	$q)$	$\wedge$	$(q$	$\rightarrow$	$p)$
V	V	V	V	V	V	V
V	F	F	F	F	V	V
F	V	V	F	V	F	F
F	V	F	V	F	V	F

Esta forma de doble condicional se indica con: p doble flecha q ($p \leftrightarrow q$), que se llama **bicondicional** y que será verdadero solo si ambos son verdaderos ó falsos. Su tabla de verdad es la siguiente:

p	q	$p \leftrightarrow q$
V	V	V
V	F	F
F	V	F
F	F	V

Método Intuitivo

Es el que percibe los sentidos por lo cual una equivalencia de mucha importancia es la obtenida a través del método intuitivo:

$$\sim p \vee q \qquad \equiv \qquad p \rightarrow q$$

Demostrando intuitivamente, nos quedaría de la siguiente manera:

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

$\equiv$

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

Recíproca, Inversa y Contrarrecíproca

Del condicional $p \rightarrow q$ deducimos las siguientes condicionales:

$q \rightarrow p$ $\qquad\qquad$ $\sim p \rightarrow \sim q$ $\qquad\qquad$ $\sim q \rightarrow \sim p$

Donde cada una de ellas reciben el nombre respectivamente de:

Recíproca $\qquad\qquad$ Inversa $\qquad\qquad$ Contrarrecíproca

Donde sus tablas de verdad son las siguientes:

				Condicional	Recíproca	Inversa	Contrarrecíproca
p	$\sim p$	q	$\sim q$	$p \rightarrow q$	$q \rightarrow p$	$\sim p \rightarrow \sim q$	$\sim q \rightarrow \sim p$
V	F	V	F	V	V	V	V

V	F	F	V	F	V	V	F
F	V	V	F	V	F	F	V
F	V	F	V	V	V	V	V

De esta tabla se deduce:

Condicional ≡ Contrarrecíproca

$$p \rightarrow q \quad \equiv \sim q \rightarrow \sim p$$

Recíproca ≡Inversa

$$q \rightarrow p \ \equiv \ \sim p \rightarrow \sim q$$

Argumentos

Un argumento es una relación entre un conjunto de proposiciones en este caso $P_1, P_2, P_3 \cdots P_n$ llamadas premisas y otra proposición q llamada conclusión, se denota por:

$$P_1, P_2, P_3 \cdots P_n \quad \vdash\!\!\!\!- \quad q$$

Se dice que un argumento es válido si las premisas dan como consecuencia la conclusión, en otras palabras, un argumento es válido si cada vez que las premisas son verdaderas, también lo es la conclusión; caso contrario el argumento no es válido y recibe el nombre de falacia. Una forma de demostrar la validez de un argumento consiste en calcular sus tablas de verdad y luego analizar si se cumple la condición antes dicha (Lipschutz, 1995). Por ejemplo.

$$P_1, P_2, P_3 \cdots P_n \quad \vdash\!\!\!\!- \quad q$$

Premisas verdaderas — *Todas* — Condición

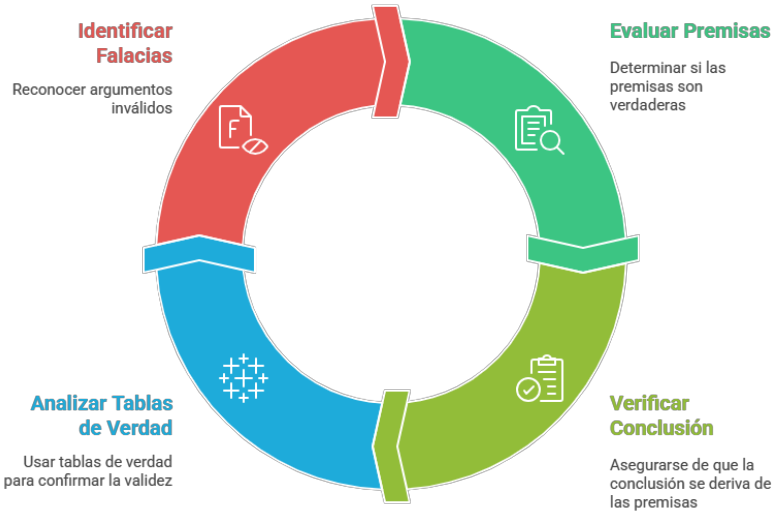


Figura 133: Ciclo de validez de un argumento
Fuente: Elaboración propia

Ejercicio:

Demostrar la validez del siguiente argumento:

p	$p \to q$	$\vdash$	q
V	V		V
V	F		F
F	V		V
F	V		F

Válido

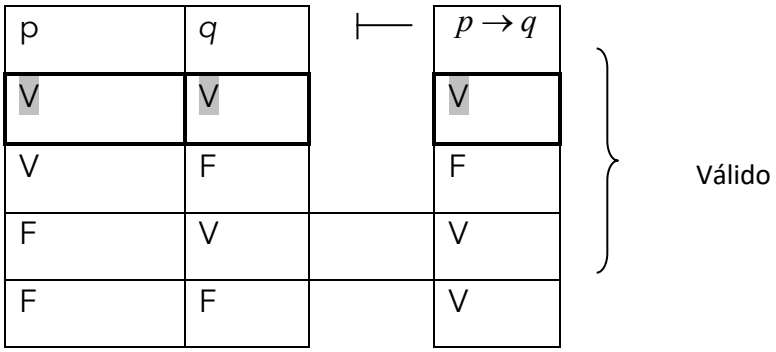
Ejercicio:

Demostrar la validez del siguiente argumento:

$p \to q$	q	$\vdash$	p
V	V		V
F	F		V
V	V		F
V	F		F

Falacia

Demostrar la validez del siguiente argumento:



La lógica proposicional sigue teniendo un papel fundamental en la era contemporánea, especialmente en el diseño de inteligencia artificial explicable (XAI), donde los razonamientos deben ser transparentes y verificables. Igualmente, las reglas de inferencia y equivalencias lógicas son aplicadas en la verificación formal de software y hardware, un campo esencial para garantizar la confiabilidad de sistemas críticos como vehículos autónomos, dispositivos médicos o protocolos de comunicación en redes 5G.

Resumen del capítulo

Este capítulo introduce los fundamentos de la lógica proposicional, base esencial para el razonamiento matemático y computacional. Se abordan las definiciones básicas que permiten comprender qué es una proposición y cómo se diferencian los enunciados verdaderos y falsos. Posteriormente, se exploran los operadores lógicos fundamentales como la conjunción, disyunción, negación, condicional y bicondicional, así como su representación mediante tablas de verdad.

El capítulo también analiza conceptos clave como tautología, contradicción y contingencia, proporcionando herramientas para evaluar la validez de proposiciones compuestas. Se estudia la equivalencia lógica y se introducen las

principales leyes del álgebra de proposiciones, como la ley de Morgan, distributiva, conmutativa y asociativa.

Además, se profundiza en la interpretación del condicional y bicondicional, junto con sus formas derivadas: recíproca, inversa y contrarrecíproca. Finalmente, se analiza la estructura de los argumentos lógicos, identificando premisas, conclusiones y cómo determinar si un argumento es válido o inválido mediante razonamientos formales y tablas de verdad.

Conclusiones

- La lógica proposicional es la base del pensamiento computacional. Comprender proposiciones, operadores lógicos y tablas de verdad permite estructurar razonamientos formales, fundamentales para el diseño de algoritmos y sistemas de decisión en computación.
- L as leyes del álgebra de proposiciones permiten simplificar expresiones lógicas. A través de equivalencias como la ley de Morgan, la distributiva, conmutativa y asociativa, se pueden transformar y optimizar expresiones lógicas para mejorar la eficiencia en programación y circuitos digitales.
- La validación de argumentos fortalece el razonamiento lógico. Evaluar la validez de argumentos mediante tablas de verdad y formas condicionales (recíproca, inversa y contrarrecíproca) es esencial para garantizar la coherencia en sistemas computacionales y en la resolución de problemas complejos.

Autoevaluación / Cuestionario

Selecciona la opción correcta

1. ¿Cuál de las siguientes opciones representa una proposición?

A) ¡Cierra la puerta!
B) ¿A qué hora llegas?
C) 3 es un número primo. ★
D) ¡Qué bonito día!

2. ¿Cuál es el operador lógico que representa la conjunción?

A) ∨
B) ¬
C) ∧ ★
D) →

3. Una tautología es una proposición que es...

A) Verdadera en todos los casos posibles. ★

B) Falsa en todos los casos.

C) Verdadera solo cuando las premisas lo son.

D) Verdadera solo si P y Q son verdaderas.

4. ¿Cuál es el valor de verdad de la proposición "P ∨ Q" si P es falsa y Q es verdadera?

A) Verdadero ★

B) Falso

C) Indeterminado

D) Contradictorio

5. ¿Cuál de las siguientes proposiciones es equivalente a "P → Q"?

A) ¬P ∧ Q

B) ¬P ∨ Q ★

C) P ∧ ¬Q

D) ¬Q → ¬P

5. ¿Qué ley lógica permite transformar ¬ (P ∧ Q) en ¬ P ∨ ¬ Q?

A) Ley distributiva

B) Ley de doble negación

C) Ley de Morgan ★

D) Ley asociativa

7. Si una proposición es siempre falsa sin importar los valores de verdad, se dice que es:

A) Tautología

B) Contingencia

C) Contradicción ★

D) Bicondicional

8. ¿Cuál es la forma recíproca del condicional "Si llueve, entonces hay nubes"?

A) Si no llueve, entonces no hay nubes.

B) Si hay nubes, entonces llueve. ★

C) Si no hay nubes, entonces no llueve.

D) Si hay nubes, entonces no llueve.

9. ¿Cuál de las siguientes opciones representa un argumento válido?

A) $P \rightarrow Q$, P, entonces $\neg Q$

B) $P \rightarrow Q$, Q, entonces P

C) $P \rightarrow Q$, $\neg Q$, entonces $\neg P$ ★

D) $P \vee Q$, $\neg P$, entonces Q ★

10. ¿Qué significa que dos proposiciones sean lógicamente equivalentes?

A) Que tienen diferente forma pero igual número de símbolos.

B) Que siempre tienen el mismo valor de verdad. ★

C) Que una es la negación de la otra.

D) Que no pueden coexistir en un mismo argumento.

Referencia Bibliográfica

Lipschutz, S (1995). Matemáticas para Computación, Primera Edición. McGrawHill. Interamericana. ISBN: 970-10-0266-0 (BIC00013).

Jiménez, José (2014). Matemáticas para la Computación. Editorial Alfaomega México. Segunda Edición. ISBN: 978-607-707-719-0.

García Merayo, Felíx (2015). Matemática Discreta. España: Paraninfo. ISBN: 978-84-283-3568-3 (BIC01140).

Jiménez de Parga, C. (2024). Lógica matemática y computacional: teoría y ejercicios resueltos. Tébar Flores.

Ing. Edison Luis Loján Cueva, Msig.
0000-0002-7092-1281
elojan@utmachala.edu.ec Filial: Universidad Técnica de Machala. Ingeniero de Sistemas, Magister en Sistemas de Información Gerencial y Especialista en Docencia Universitaria. Con más de dos décadas de experiencia en la enseñanza universitaria, ha ejercido la cátedra desde el año 2003 en instituciones como la Universidad Católica de Cuenca y actualmente en la Universidad Técnica de Machala, en donde continúa formando nuevas generaciones de profesionales. En su campo de experticia ha desarrollado una destacada labor docente e investigativa. Ha dirigido numerosos proyectos de tesis de grado y posgrado, contribuyendo al fortalecimiento académico y científico de la comunidad universitaria. Se distingue por su compromiso con la educación, su vocación por la enseñanza y su firme propósito de democratizar el conocimiento técnico a través de obras como esta, que buscan revelar la estructura lógica que sostiene el mundo de la computación.

Ing. Maritza Alexandra Pinta. PhD.
0000-0002-5697-0474
mpinta@utmachala.edu.ec. Filial: Universidad Técnica de Machala. Filial: Universidad Técnica de Machala. Ingeniero Civil, Licenciado En Ciencias De La Educación Especialidad De Matemáticas, Doctor En Ciencias De La Educación Mención En Investigación Y Planificación Educativa, Magister En Desarrollo De La Inteligencia Y Educación, egresada de Maestría en Big Data y Ciencias de Datos, Doctora Dentro Del Programa De Doctorado En Matemáticas por la universidad de Almería, España. Se desempeña actualmente como docente de Análisis Matemático y Algebra Lineal en la carrera de tecnologías de la Información de la Universidad Técnica de Machala. Ha participado como autora de libros de Análisis Matemático y Métodos Numéricos y en proyectos de investigación relacionados con las matemáticas y las Tecnologías de la Información.

ISBN: 978-9942-53-024-0

9 789942 530240

Compás
capacitación e investigación