JOSÉ ALFREDO JIMÉNEZ MURILLO

MATEMÁTICAS PARA LA COMPUTACIÓN

3ª EDICIÓN

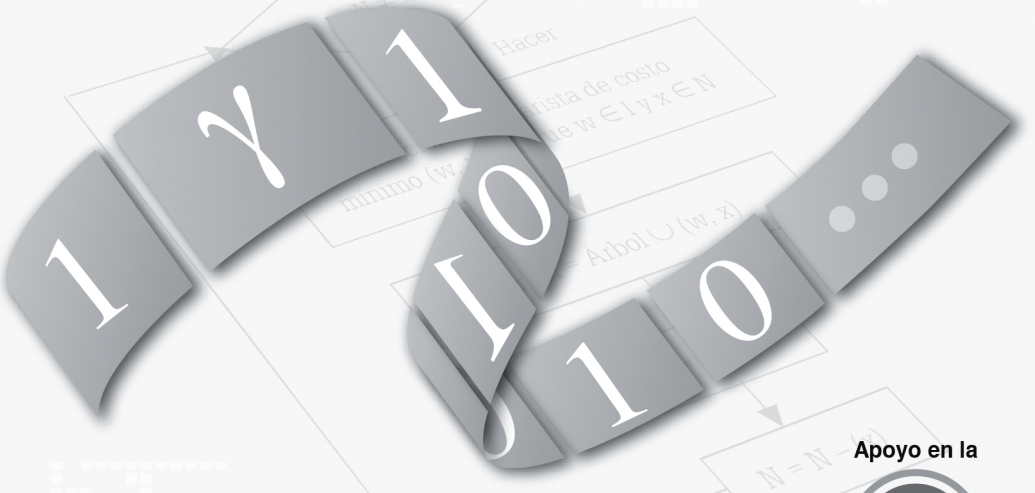
Apoyo en la web

Alfaomega

JOSÉ ALFREDO JIMÉNEZ MURILLO

MATEMÁTICAS PARA LA COMPUTACIÓN

3ª EDICIÓN



Apoyo en la web

Alfaomega

Director Editorial
Marcelo Grillo Giannetto
mgrillo@alfaomega.com.mx

Jefe de Ediciones
Francisco Javier Rodríguez Cruz
jrodriguez@alfaomega.com.mx

Supervisión Editorial
Ana Edith Hernández Velázquez
ehernandez@alfaomega.com.mx

Datos catalográficos
Jiménez Murillo, José Alfredo
Matemáticas para la Computación 3ª Edición

Alfaomega Grupo Editor, S.A. de C.V., México

ISBN 978-607-622-511-0

Formato: 21 × 24 cm Páginas: 576

Matemáticas para la Computación 3ª Edición
José Alfredo Jiménez Murillo

Tercera edición: Alfaomega Grupo Editor, México, agosto de 2015

Pitágoras 1139, Col. Del Valle, 03100, México D.F.

Miembro de la Cámara Nacional de la Industria Editorial Mexicana
Registro No. 2317

Pág. Web: **http://www.alfaomega.com.mx**
E-mail: **atencionalcliente@alfaomega.com.mx**

ISBN: 978-607-622-511-0

Empresas del grupo:

México: Alfaomega Grupo Editor, S.A. de C.V.-Pitágoras 1139, Col. Del Valle, México, D.F.-C.P. 03100.
Tel.: (52-55) 5575-5022 - Fax: (52-55) 5575-2420 / 2490. Sin costo: 01-800-020-4396
E-mail: atencionalcliente@alfaomega.com.mx

Colombia: Alfaomega Colombia S.A. - Calle 62, No. 20-46, Barrio San Luis, Bogotá, Colombia.
Tel.: (57-1) 746-0102 / 210-0415 - E-mail: cliente@alfaomega.com.co

Chile: Alfaomega Grupo Editor, S.A. - Av. Providencia 1443, Oficina 24, Santiago, Chile
Tel.: (56-2) 2235-4248 - Fax: (56-2) 2235-5786. E-mail: agechile@alfaomega.cl

Argentina: Alfaomega Grupo Editor Argentino, S.A. - Paraguay 1307 P.B. Of. 11, C.P. 1087, Buenos Aires, Argentina.
Tel./Fax: (54-11) 4811-8887 y 4811 7183. E-mail: ventas@alfaomegaeditor.com.ar

Mensaje del Editor

Una de las convicciones fundamentales de Alfaomega es que los conocimientos son esenciales en el desempeño profesional, ya que sin ellos es imposible adquirir las habilidades para competir laboralmente. El avance de la ciencia y de la tecnología hace necesario actualizar continuamente esos conocimientos, y de acuerdo con esta circunstancia Alfaomega publica obras actualizadas, con alto rigor científico y técnico, y escritas por los especialistas del área respectiva más destacados.

Consciente del alto nivel competitivo que debe de adquirir el estudiante durante su formación profesional, Alfaomega aporta un fondo editorial que se destaca por sus lineamientos pedagógicos que coadyuvan a desarrollar las competencias requeridas en cada profesión específica.

De acuerdo con esta misión, con el fin de facilitar la comprensión y apropiación del contenido de esta obra, cada capítulo inicia con el planteamiento de los objetivos del mismo y con una introducción en la que se plantean los antecedentes y una descripción de la estructura lógica de los temas expuestos, asimismo a lo largo de la exposición se presentan ejemplos desarrollados con todo detalle y cada capítulo concluye con un resumen y una serie de ejercicios propuestos.

Además de la estructura pedagógica con que está diseñado el contenido de nuestros libros, Alfaomega hace uso de los medios impresos tradicionales en combinación con las Tecnologías de la Información y las Comunicaciones (TIC) para facilitar el aprendizaje. Correspondiente a este concepto de edición, todas nuestras obras tienen su complemento en una página Web en donde el alumno y el profesor encontrarán lecturas complementarias así como programas desarrollados en relación con temas específicos de la obra.

Los libros de Alfaomega están diseñados para ser utilizados en los procesos de enseñanza aprendizaje, y pueden ser usados como textos en diversos cursos o como apoyo para reforzar el desarrollo profesional, de esta forma Alfaomega espera contribuir así a la formación y al desarrollo de profesionales exitosos para beneficio de la sociedad, y espera ser su compañera profesional en este viaje de por vida por el mundo del conocimiento.

Acerca del autor

José Alfredo Jiménez Murillo. Originario de Villa Morelos Michoacán, México, el Maestro Jiménez Murillo obtuvo el título de Ingeniero Industrial Mecánico en Diseño de Manufactura con el trabajo de tesis "Diseño industrial de engranes por computación" en el Instituto Tecnológico de Morelia (ITM), Morelia Mich. Méx., asimismo obtuvo el grado de Maestría en Ciencias en Cómputo Aplicado en el Centro de Estadística y Cálculo (CEC) del Colegio de Postgraduados (CP), en Montecillo Estado de México, con la tesis "Diseño y simulación del circuito de control para una etiquetadora (DYSCCE)", además de recibir un reconocimiento por la gran trayectoria académica mostrada durante su estancia en ese centro de Investigación.

Posteriormente obtuvo una segunda Maestría en Ciencias en la Enseñanza de las Ciencias, en el área de Matemáticas en el Centro Interdisciplinario de Investigación y Docencia en Educación Técnica (CIIDET) de la Ciudad de Querétaro Qro. Méx., con la tesis "Método didáctico para la enseñanza-aprendizaje de las relaciones y funciones", y también recibió un reconocimiento como el mejor promedio de su generación.

En sus más de 30 años de experiencia profesional ha dictado innumerables conferencias e impartido varios cursos relacionados con el área de las ciencias exactas y la computación, ha sido maestro de varias materias en el campo de las matemáticas, la física y la computación en los niveles de licenciatura y posgrado.

Tiene una trayectoria distinguida en el Sistema de Institutos Tecnológicos de la República Mexicana, donde ha participado en el desarrollo de programas de asignaturas del plan de estudio de diferentes ingenierías. Ha sido asesor de distintos trabajos de titulación a nivel licenciatura y posgrado y sus líneas de investigación son las Matemáticas discretas e Ingeniería del software. Actualmente es maestro titular de Matemáticas discretas en el Instituto Tecnológico de Morelia.

A la memoria de mis padres:

Marcelo y Josefina

Con todo mi amor para mis hijos:

Eréndira Miriam
Paola y
Marcelo

Agradecimientos

Antes que nada agradezco a los alumnos que me ayudaron a corregir errores y ambigüedades de esta obra, sus comentarios y sugerencias fueron y son de gran valor para mejorar y enriquecer esta edición. Imposible citarlos a todos, pero mi reconocimiento más sincero para cada uno de ellos.

Agradezco al Tecnológico Nacional de México, en especial al Instituto Tecnológico de Morelia por las facilidades otorgadas para la realización de la primera y segunda edición de este libro.

También agradezco a Alfaomega Grupo Editor que me permitió compartir y publicar este material. Mi reconocimiento para todos aquellos que de una manera u otra, han hecho posible la edición de este libro, en especial a Carlos Umaña, Director de Alfaomega, a Marcelo Grillo, Director editorial de Alfaomega, y especialmente a Javier Rodríguez Cruz, Editor de este libro, con quien fue un placer trabajar.

Deseo agradecer también a maestros de Instituciones Educativas que han utilizado este libro en su aula de clase. A aquellos que se tomaron la molestia de leer, revisar y emitir sugerencias que le dan valor a este libro, en especial a los siguientes profesores:

Gabriel Hurtado Chong	Universidad Nacional Autónoma de México
Maria Elena Markiewicz	Universidad de Río Cuarto. Argentina
Jorge I. López Chalé	Instituto Tecnológico de Morelia. México.

Contenido

Prefacio

Las matemáticas siempre han sido una de las disciplinas que le cuesta más trabajo entender a los estudiantes. Si se observan las estadísticas de reprobación en las carreras relacionadas con la computación, también la materia de programación es una aduana muy difícil de librar. Sin embargo, al mismo tiempo ambas disciplinas constituyen un campo importante, apasionante y ameno. La computación y las matemáticas tienen gran relación entre sí, solamenta hay que recordar que las computadoras fueron creadas inicialmente para realizar con mayor rapidez operaciones matemáticas, y la computación no se podría entender sin la participación de las matemáticas. Es cierto que actualmente la computación apoya a todas las actividades que se desarrollan diariamente en la administración, educación, medicina, ingeniería e investigación.

Simplemente no se podría entender el funcionamiento adecuado de una empresa sin la ayuda de la computadora, ¿qué haríamos si se tuviera que regresar al tiempo en que todo se procesaba manualmente? ¿Qué pasaría si no se contara con el correo electrónico, hojas de cálculo, procesadores de texto, lenguajes de programación e internet? Si bien es cierto que todos estos elementos son parte de las acciones que se pueden llevar a cabo en la computadora, también lo es que las matemáticas proporcionaron el soporte necesario para desarrollar todas estas herramientas computacionales. Ramas de las matemáticas como sistemas numéricos, métodos de conteo, conjuntos, matrices, lógica matemática, álgebra boolena, relaciones y funciones, son la base para el diseño de todo lo que se maneja en la computadora. Es por eso que surgen las matemáticas para la computación, mismas que permiten entender el aspecto formal de la relación matemáticas-computadora.

La presente obra tiene como objetivo fundamental que los alumnos que cursan alguna carrera relacionada con la computación, aprendan con facilidad los conocimientos básicos matemáticos necesarios para entender el principio matemático usado en la creación de herramientas computacionales, se espera que el joven que incursiona en el siempre interesante mundo de la computación, tenga una visión más clara de los aspectos que se toman en cuenta para el desarrollo y manejo de estructuras de datos, bases de datos, circuitos electrónicos y lenguajes de programación, no para desarrollar un software al final del curso, pero si para tener una mejor visión de todo aquello que ayudó a desarrollar estas herramientas computacionales que hoy usamos y de las cuales somos muy dependientes. Para entender el contenido del libro no es necesaria una plataforma muy fuerte de matemáticas, ya que se tratan todos los temas con palabras y conceptos que los estudiantes pueden entender, sin olvidar el aspecto formal propio de las matemáticas.

En la presente obra se pretende que el alumno vincule los conocimientos matemáticos con la compu-

tación, usando los esquemas de conocimientos con los cuales los alumnos llegan a la licenciatura, con la finalidad de que el alumno le de significado al material presentado en los diferentes temas en cada uno de los capítulos se resuelven problemas representativos e ilustrativos, pero se dejan para que el alumno resuelva otros algunas veces con mayor dificultad e igualmente interesantes, con la finalidad de que el alumno relacione lo aprendido en el aula con el nuevo problema a resolver, lo cual habla de un conocimiento significativo y el aprendizaje basado en competencias. Se recomienda al maestro asignar proyectos que tengan relación con el material del libro, para que los alumnos los desarrollen fuera del salón de clases, con el fin de que el alumno realice trabajo en equipo, ya que se pretende que estos proyectos tengan mayor dificultad, en donde implique el trabajo de más de una persona. La presente obra tiene las bases para que el alumno pueda aumentar su estructura cognoscitiva, al relacionar las matemáticas con la computación.

Este libro está integrado por nueve capítulos. El capítulo uno es *sistemas numéricos* y tiene como finalidad la representación y operación de cantidades en los sistemas binario, octal y hexadecimal como una forma de comunicación entre el ser humano y la computadora, se aborda la forma de llevar acabo operaciones aritméticas básicas en diferentes sistemas numéricos, la suma de dos cantidades en complemento a dos, adicionalmente en esta edición se tiene la multiplicación de dos cantidades usando el algoritmo de Booth. El segundo capítulo es *métodos de conteo*, mismos que permiten evaluar y mejorar el software, ya que en computación se busca desarrollar software cada vez más eficiente y compacto, que permita disminuir el número de interacciones, comparaciones y ciclos, tendientes a optimizar los recursos, tiene información de combinaciones, permutaciones, binomio de Newton y se agrega el problema del palomar. El tercer capítulo es *conjuntos*, ya que son las bases necesarias para entender todo el material relacionado con la computación, se establece la relación de los conjuntos con lógica matemática y álgebra booleana. En el cuarto capítulo se aborda *lógica matemática*, con el objetivo de que el alumno aprenda a resolver problemas usando como herramientas fundamentales la reflexión, vinculación y el sentido común, pero teniendo como herramientas las reglas de inferencia, equivalencias lógicas y tautologías, en este capítulo se aborda la validez de proposiciones por medio de métodos deductivos como ocurre con la demostración por el método directo y contradicción, pero también se usa inducción matemática para demostrar si una proposición es cierta, como una alternativa para probrar algoritmos, se incluye información del método de Quine para demostrar la validez de una proposición. El quinto capítulo es *álgebra booleana*, cuya finalidad es que el alumno adquiera las bases necesarias para entender, representar y manejar circuitos electrónicos básicos partiendo de la consideración de que la computadora está integrada con ellos, se simplifican funciones booleanas por medio de teoremas del álgebra booleana y mapas de Karnaugh, así como la representación de expresiones booleanas usando para ello bloques lógicos.

El sexto capítulo es *relaciones y funciones*, con el objetivo de que el estudiante aprenda la representación y manejo de las relaciones, sabiendo de antemano que en las relaciones y funciones son escenciales en bases de datos, estructuras de datos y programación, se incluyó material de diagramas de Hasse. El séptimo capítulo es *grafos*, partiendo del hecho de que los grafos son una representación gráfica de las redes de comunicación, incluyendo por supuesto las redes de computadoras, se abordan circuitos famosos como el circuito de Euler y Hamilton, se manejan problemas en donde las aristas tienen pesos, distancias o costos, como ocurre con el algoritmo de Dijktra, que permiten eliminar aristas costosas, se

tratan temas importantes como coloración de grafos planos y se utilizan los grafos de similaridad como una forma de discriminar información con características semejantes como ocurre en el reconocimiento de patrones. El octavo capítulo es *árboles*, los cuales son grafos no dirigidos conexos, sin ciclos ni lazos que permiten la estructuración de los datos necesaria en la computación para accesar de manera más rápida y eficiente la información, permiten la evaluación de expresiones matemáticas, se usa para la compactación de información como ocurre con el código de Huffman, en este capítulo se aborda el recorrido de árboles y la búsqueda de información a lo ancho y en profundidad. El noveno capítulo es *introducción a los lenguajes formales*, con la finalidad de que el alumno adquiera las bases necesarias para comprender asignaturas posteriores que tienen relación directa con matemáticas para la computación, en este capítulo se tiene material de gramáticas, lenguajes regulares, árboles de derivación, autómatas finitos determinísticos y no determinísticos, máquinas de estado finito, representación BNF y máquinas de Turing.

Material Web de apoyo

Además del contenido descrito, este libro cuenta con material Web de apoyo que permitirá reforzar, complementar, y enriquecer los conocimientos del alumno y auxiliar al maestro en su noble tarea. Para el alumno los tipos específicos de material Web incluidos son los siguientes:

- Mapas conceptuales.
- Simuladores.
- Lecturas complementarias.

Para el profesor los tipos de material Web disponibles son los anteriores, además de los siguientes:

- Presentaciones en Power Point.
- Solución detallada de todos los problemas propuestos.

José Alfredo Jiménez Murillo

Plataforma de contenidos interactivos

Para tener acceso al material de la plataforma de contenidos interactivos de **Matemáticas para la computación 3ª Edición**, siga los siguientes pasos:

1. Ir a la página: http://libroweb.alfaomega.com.mx

2. Ir a la sección de catálogo y seleccionar la imagen de la portada del libro, al dar doble clic sobre ella tendrá acceso al material descargable.

NOTA: se recomienda respaldar los archivos descargados de la página web en un soporte físico.

Capítulo 1

Sistemas numéricos

La mayoría de las ideas fundamentales de la ciencia son esencialmente sencillas, y por regla general pueden ser expresadas en un lenguaje comprensible para todos.

Albert Einstein

Competencia del capítulo

Aprender a representar y realizar operaciones aritméticas básicas en diferentes sistemas numéricos, así como la suma en complemento a 2 y la multiplicación de dos cantidades por medio del algoritmo de Booth, con la finalidad de comprender la forma en que la computadora lleva a cabo el procesamiento de información.

Competencias específicas

- Representar cantidades en cualquier sistema numérico, incluyendo los sistemas binario, octal y hexadecimal.
- Realizar las operaciones aritméticas básicas en diferentes sistemas numéricos, incluyendo los sistemas de numeración binario, octal y hexadecimal.
- Sumar dos cantidades en complemento a 2, para ilustrar la forma en que la computadora lleva a cabo operaciones aritméticas.
- Multiplicar dos cantidades por medio del algoritmo de Booth.

1.1 Introducción

De acuerdo con la historia, se cree que los primeros pobladores utilizaban rayas, círculos, figuras de animales u objetos para representar cantidades. Por ejemplo, una manada de siete animales podría estar representada por siete rayas o siete figuras de ese animal, pero para representar cantidades cada vez mayores se usó la agrupación de varios símbolos en uno solo, con la finalidad de compactar la información.

Por ejemplo, los egipcios empleaban símbolos para representar cantidades y algunos de ellos son $| = 1$, $\cap = 10$, $? = 100$; utilizando éstos, la representación de 134 es la siguiente:

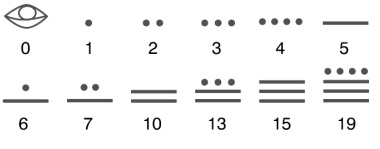
$$? \cap \cap \cap |||| = 134$$

Un sistema como el anterior se conoce como *sistema aditivo*, y en él se suman los valores de todos los símbolos para obtener la cantidad total, sin embargo este sistema es impráctico para la representación de cantidades grandes o muy pequeñas, ya que se necesitarían muchos símbolos para su representación.

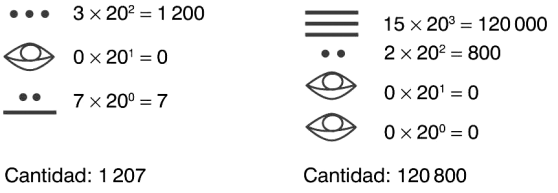
Otro sistema aditivo es el sistema de numeración romano en el cual los símbolos I, V, X, L, C, D y M representan cantidades, y una línea sobre el símbolo implica una multiplicación del número por mil. En ambos casos se suman los valores de los caracteres de acuerdo con sus propias reglas, pero en éstas no importa la posición sino únicamente el símbolo y es por eso que se les llama *sistemas de numeración aditivos*.

Se cree que los babilonios fueron uno de los primeros pueblos en usar un sistema posicional para la representación de cantidades, ya que con base en el movimiento de los astros empleaban un sistema sexagesimal (60 caracteres diferentes, en donde cada uno representa un número) para indicar cantidades. Su sistema aún se utiliza para la medición de horas, minutos y segundos, sin embargo tiene problemas con la representación del cero.

Otro sistema posicional es el sistema numérico maya, en el que se estableció un símbolo para representar el número cero, necesario para el buen funcionamiento de todo sistema posicional, con lo cual la cultura maya hizo una aportación valiosa a la ciencia. Este sistema tiene una base de 20, y los 20 símbolos distintos correspondientes se obtienen a partir de la combinación de los que se consideran los tres símbolos básicos para la representación de cantidades. Los siguientes son algunos de los símbolos de este sistema:



Hasta este momento parece que se trata de un sistema numérico aditivo, ya que se suman las rayas y los puntos para obtener los diferentes símbolos utilizados, sin embargo a partir del 20 se utilizan los diferentes símbolos considerando la posición que ocupan, de forma que al multiplicar el símbolo por potencias de 20 (según su posición) y sumar los resultados parciales se obtiene la cantidad a representar. Se puede notar que para representar cantidades se coloca un símbolo encima del otro, asignando la posición 0 al que está en la parte más baja, al que le sigue hacia arriba la posición 1 y así sucesivamente. Como se muestra a continuación, el número que corresponde a las siguientes representaciones es el que se obtiene luego de sumar el valor de los símbolos utilizados:



Como se ve en esta representación, la posición del símbolo utilizado juega un papel importante.

Actualmente los sistemas para la representación de cantidades son posicionales, ya que éstos tienen muchas ventajas en relación con los aditivos. Ejemplos de sistemas posicionales son los sistemas numéricos decimal, binario, octal y hexadecimal. Una característica de los sistemas posicionales es que el valor del símbolo lo determina la posición que ocupa, y la base del sistema, que es la cantidad de símbolos diferentes usados en un sistema numérico. En este libro se trata preferentemente la representación, conversión y operaciones aritméticas en los sistemas *decimal*, *binario*, *octal* y *hexadecimal*, pero es importante mencionar que el procedimiento de representación, conversión y operación es el mismo, independientemente del sistema numérico.

1.2 Sistema decimal

El sistema decimal se usa en forma rutinaria para la representación de cantidades mediante los siguientes 10 caracteres diferentes:

$$0, \quad 1, \quad 2, \quad 3, \quad 4, \quad 5, \quad 6, \quad 7, \quad 8, \quad 9$$

Con estas cifras se pueden expresar cantidades hasta el 9. Para expresar cantidades más allá de este número es necesario introducir la representación posicional, es decir, a cada cifra se le asigna un valor posicional determinado de acuerdo con el lugar que ocupa dentro del número. Por ejemplo: el número decimal 836.74 se compone en la parte entera de la cifra 8 con el valor posicional 100, la cifra 3 con el valor posicional 10 y la cifra 6 con el valor posicional 1, y en la parte fraccionaria de la cifra 7 con el valor posicional 0.1 y la cifra 4 con el valor posicional 0.01. Así se tiene que:

$$836.74 = 8 \times 100 + 3 \times 10 + 6 \times 1 + \frac{7}{10} + \frac{4}{100}$$

Usando exponentes esto se puede representar como:

$$836.74 = 8 \times 10^2 + 3 \times 10^1 + 6 \times 10^0 + 7 \times 10^{-1} + 4 \times 10^{-2}$$

A esta forma de representación se le llama *representación exponencial*. La representación exponencial es especialmente importante porque por medio de ella se puede convertir una cantidad representada en

cualquier sistema numérico al sistema decimal, como se estudiará más adelante. El valor de la posición lo determina el exponente en una sucesión ascendente de derecha a izquierda para los enteros a partir del punto decimal (el 6 se encuentra en la posición 0, el 3 en la posición 1 y el 8 en la posición 2), y de izquierda a derecha para la parte fraccionaria (el 7 está en la posición -1 y el 4 en la posición -2), usando como base el número 10, debido a que se está en el sistema decimal. También se dice que la base de este sistema aritmético es 10, tomando en cuenta los 10 símbolos disponibles para representar cantidades.

Sistema decimal

De acuerdo con la antropología, el origen del sistema decimal se encuentra en el hecho de que los seres humanos tenemos diez dedos en las manos.

1.3 Sistemas binario, octal y hexadecimal

1.3.1 Sistema binario

En el sistema binario sólo hay dos cifras: 0 y 1. Como sucede en el sistema decimal, en el sistema binario también se utilizan exponentes para expresar cantidades mayores. Mientras que en el sistema decimal la base es 10, en el sistema binario la base es 2.

Como se mencionó anteriormente, la representación exponencial se utiliza para convertir una cantidad de un sistema numérico cualquiera al sistema decimal. A continuación se muestra la forma de hacer esto.

Ejemplo 1.1

Convertir el número binario 10011.01 a decimal.

Solución. Expresando el número propuesto en notación exponencial y realizando las operaciones correspondientes, se obtiene la siguiente conversión de binario a decimal:

$$\begin{aligned} 10011.01_{(2)} &= 1\times 2^4 + 0\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0 + 0\times 2^{-1} + 1\times 2^{-2} \\ &= 16 + 0 + 0 + 2 + 1 + 0 + 0.25 = 19.25_{(10)} \end{aligned}$$

Como el 0 y el 1 son caracteres válidos en el sistema decimal y en otros sistemas de mayor base, de aquí en adelante se indicará el sistema en que se encuentra un número expresando su base como un subíndice entre paréntesis, como se hizo en el ejemplo anterior en el que la cantidad binaria está indicada como $10011.01_{(2)}$.

Toda cantidad multiplicada por cero es cero, como se mostró en el caso anterior, sin embargo a partir de ahora esto será suprimido.

Si se desea convertir una cantidad que tiene una parte entera y otra fraccionaria de base diez a base dos, la parte entera se divide sucesivamente entre 2 y los restos resultantes se toman en orden contrario a como se encontraron. La parte fraccionaria se multiplica por 2 y el entero del resultado conforma la parte fraccionaria en el orden en que fueron encontrados. Este procedimiento se ilustra en el siguiente ejemplo.

Ejemplo 1.2

Convertir el número $28.37_{(10)}$ a binario.

Solución. Parte entera:

			Resto	
$28/2$	$=$	14	0	↑
$14/2$	$=$	7	0	↑
$7/2$	$=$	3	1	↑
$3/2$	$=$	1	1	↑
$1/2$	$=$	0	1	↑

Los restos se toman en orden inverso a como fueron encontrados.

Parte fraccionaria:

			Entero	
0.37×2	$=$	0.74	0	↓
0.74×2	$=$	1.48	1	↓
0.48×2	$=$	0.96	0	↓
0.96×2	$=$	1.92	1	↓
0.92×2	$=$	1.84		

Los enteros se toman en el mismo orden en que fueron encontrados.

Se podría seguir aproximando para determinar más dígitos en la parte fraccionaria y obtener así un

resultado más exacto, sin embargo para ilustrar el procedimiento es suficiente con cuatro dígitos después del punto que separa a la parte entera de la parte fraccionaria. De esta forma, el resultado es:

$$28.37_{(10)} = 11100.0101_{(2)}$$

Sistema binario

El antiguo matemático Pingala presentó la primera descripción que se conoce de un sistema de numeración binario en el siglo III a. de C., lo cual coincidió con su descubrimiento del número cero.

El sistema binario moderno fue documentado en su totalidad por Leibniz en el siglo XVII, en su artículo ***Explication de l'Arithmétique Binaire***. Aquí Leibniz usó el 0 y el 1, al igual que el sistema de numeración binario actual.

Gottfried Wilhelm von Leibniz (1646-1716)

Fue un filósofo, matemático, jurista y político alemán que nació en Leipzig y que en el área de las matemáticas descubrió el cálculo infinitesimal, independientemente de Newton, e inventó el sistema de numeración binario en que se basan casi todas las arquitecturas de computación actuales.

1.3.2 Sistema octal

La reglas descritas para los sistemas decimal y binario, también son aplicables al sistema octal. En los siguientes ejemplos se ilustra este planteamiento.

Ejemplo 1.3

Convertir $631.532_{(8)}$ a binario.

Solución. Primero se convierte el número dado a decimal y luego de decimal a binario.

Para convertir una cantidad de cualquier sistema numérico a decimal, se plantea su representación en notación exponencial y se realizan las operaciones. En este caso particular se tiene que:

$$\begin{aligned} 631.532_{(8)} &= 6\times 8^2 + 3\times 8^1 + 1\times 8^0 + 5\times 8^{-1} + 3\times 8^{-2} + 2\times 8^{-3} \\ &= 409.6758_{(10)} \end{aligned}$$

La conversión del número obtenido a binario es la siguiente:

Parte entera			Resto	Parte fraccionaria			Entero
$409/2$	$=$	204	1 ↑				
$204/2$	$=$	102	0				
$102/2$	$=$	51	0				
$51/2$	$=$	25	1	0.6758×2	$=$	1.3516	1 ↓
$25/2$	$=$	12	1	0.3516×2	$=$	0.7032	0
$12/2$	$=$	6	0	0.7032×2	$=$	1.4064	1
$6/2$	$=$	3	0	0.4064×2	$=$	0.8128	0
$3/2$	$=$	1	1				
$1/2$	$=$	0	1 ↑				

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

La conversión de octal a binario y de binario a octal es relativamente fácil si se utiliza una tabla de equivalencias. En esta tabla se puede apreciar que se utilizan tres dígitos en binario, para cada número en octal, debido a que la cantidad mayor válida en el sistema octal es el número 7, que ocupa tres bits, por lo tanto, todos deberán usar la misma cantidad de bits.

Sistema octal

El sistema de numeración octal usa 8 dígitos (0, 1, 2, 3, 4, 5, 6, 7) que tienen el mismo valor que en el sistema de numeración decimal.
Este sistema es muy usado en la computación por tener una base que es potencia exacta de 2, además de que esta característica hace que la conversión a binario o viceversa sea bastante simple.

Ejemplo 1.4

Convertir $631.532_{(8)}$ a binario usando la tabla de equivalencias anterior.

Solución. En la siguiente tabla se presenta la conversión pedida:

6	3	1	.	5	3	$2_{(8)}$
110	011	001	.	101	011	$010_{(2)}$

Como se puede observar en el ejemplo 1.4, usando la tabla de equivalencias el resultado es igual al obtenido al aplicar el método general usado en el ejemplo 1.3, sin embargo se debe mencionar que cuando se usa el método general algunas veces existen diferencias en los resultados, ya que al convertir la cantidad de octal a decimal y posteriormente a binario se pierde exactitud por el redondeo y también debido a que se acordó anteriormente encontrar solamente los primeros cuatro dígitos en la parte fraccionaria. A pesar de esto, si existiera alguna diferencia ésta se presentaría en la parte fraccionaria y no en la parte entera.

Ejemplo 1.5

Convertir $1101010000011110101101 0.0001101_{(2)}$ a octal usando tablas y verificar dicho resultado usando el método general.

Solución. Cuando se usan tablas, se deben separar los bits de la cantidad binaria en bloques de tres en tres, a partir del punto decimal hacia la izquierda en la parte entera, y del punto decimal a la derecha

en la parte fraccionaria. Si los bloques no se completan, se agregan ceros en los extremos como se indica a continuación:

011	010	100	000	111	101	011	010	.	000	110	$100_{(2)}$
3	2	4	0	7	5	3	2	.	0	6	$4_{(8)}$

Para verificar este resultado usando el método general, primero se convierte de binario a decimal de forma que:

$$\begin{aligned}
11010100000111101011010.0001101_{(2)} &= 1\times 2^{22}+1\times 2^{21}+1\times 2^{19}+1\times 2^{17}+1\times 2^{11}+1\times 2^{10}\\
&\quad +1\times 2^{9}+1\times 2^{8}+1\times 2^{6}+1\times 2^{4}+1\times 2^{3}+1\times 2^{1}\\
&\quad +1\times 2^{-4}+1\times 2^{-5}+1\times 2^{-7}\\
&= 4194304+2097152+524288+131072+2048+1024\\
&\quad +512+256+64+16+8+2+0.0625+0.0312+0.0078\\
&= 6950746.1015_{(10)}
\end{aligned}$$

A continuación se convierte de decimal a octal:

Parte entera			Resto	Parte fraccionaria			Entero
$6950746/8$	$=$	868843	2 ↑				
$868843/8$	$=$	108605	3				
$108605/8$	$=$	13575	5	0.1015×8	$=$	0.8120	0
$13575/8$	$=$	1696	7	0.8120×8	$=$	6.4960	6
$1696/8$	$=$	212	0	0.4960×8	$=$	3.9680	3
$212/8$	$=$	26	4	0.9680×8	$=$	7.7440	7 ↓
$26/8$	$=$	3	2				
$3/8$	$=$	0	3				

Al comparar los resultados obtenidos por el método general y mediante la tabla de equivalencias, se observa que la parte entera coincide totalmente y que solamente existe una pequeña diferencia en la parte fraccionaria, debido a errores de redondeo.

Recomendación

Es importante tener cuidado cuando se encuentra el resto de una división. En el caso de este ejemplo se dividió 108605/8 = 13575.625, y el resto resultó de multiplicar la parte fraccionaria del cociente por la base, esto es, 0.625 × 8 = 5

1.3.3 Sistema hexadecimal

La base numérica del sistema hexadecimal es 16 y para representar cantidades en él se utilizan los diez dígitos del sistema decimal (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) así como las seis primeras letras del alfabeto (A, B, C, D, E, F). Con esto pueden formarse números según el principio de valor posicional como en los demás sistemas aritméticos.

Los caracteres válidos en hexadecimal son del 1 al 15, con la particularidad de que a las letras se les asigna el siguiente valor: A = 10, B = 11, C = 12, D = 13, E = 14 y F = 15.

Ejemplo 1.6

Convertir $E8A7.3D_{(16)}$ a octal.

Solución. El número dado primero se convierte a decimal:

$$E8A7.3D_{(16)} = 14 \times 16^3 + 8 \times 16^2 + 10 \times 16^1 + 7 \times 16^0 + 3 \times 16^{-1} + 13 \times 16^{-2} = 59559.2383_{(10)}$$

Ahora el número obtenido se convierte a octal:

Parte entera			Resto	Parte fraccionaria			Entero
59559/8	=	7444	7 ↑				
7444/8	=	930	4	0.2383×8	=	1.9064	1
930/8	=	116	2	0.9064×8	=	7.2512	7
116/8	=	14	4	0.2512×8	=	2.0096	2
14/8	=	1	6	0.0096×8	=	0.0768	0 ↓
1/8	=	0	1				

De igual manera que en la conversión de binario a octal, se puede obtener la siguiente tabla de equivalencias de binario a hexadecimal.

Hexadecimal	Binario
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

Nuevamente el número mayor del sistema numérico es el que manda en relación con cuántos bits se deberán usar para representar cada uno de los caracteres. En este caso F = 15 es el símbolo mayor y ocupa cuatro bits, por lo tanto todos los símbolos deberán representarse por cuatro bits.

Ejemplo 1.7

Convertir $E8A7.3D_{(16)}$ a octal usando tablas de equivalencias.

Solución. A diferencia del método general, en que el sistema intermedio es el sistema decimal, cuando se utilizan tablas de equivalencias el sistema intermedio es el binario, por lo que primero se pasa la cantidad al sistema binario poniendo los cuatro bits correspondientes a cada uno de los caracteres del sistema hexadecimal:

E	8	A	7	.	3	D_{16}
1110	1000	1010	0111	.	0011	$1101_{(2)}$

A continuación se pasa de binario a octal, agrupando la información en bloques de tres bits, ya que la tabla de equivalencia octal-binario utiliza solamente tres bits para cada uno de los caracteres. En caso de no completarse los bloques de tres bits, se deberán agregar los ceros necesarios en los extremos:

001	110	100	010	100	111	.	001	111	010_2
1	6	4	2	4	7	.	1	7	$2_{(8)}$

Como se puede observar, el resultado es semejante al obtenido en el ejemplo 1.6 y la variación, en caso de existir, es muy pequeña.

Sistema hexadecimal

El uso del sistema hexadecimal está estrechamente relacionado con la informática y con las ciencias de la computación, ya que las computadoras suelen utilizar el byte u octeto como unidad básica de memoria.

1.4 Generalización de las conversiones

De la misma manera en que fueron creados los sistemas posicionales decimal, binario, octal y hexadecimal, es posible crear nuestro propio sistema usando los dígitos necesarios del 0 al 9, y también en el caso de que se requieran las letras del alfabeto.

Las siguientes cantidades están expresadas en sistemas posicionales inexistentes, pero que podrían ser perfectamente válidos ya que respetan todas las reglas de los sistemas posicionales:

$20541.32_{(7)}$	Aquí la base es 7 y los caracteres válidos van del 0 al 6.
$7G5A90.HB_{(18)}$	En este caso, además de poder usar los dígitos del 0 al 9 es posible utilizar las letras A = 10, B = 11, C = 12, D = 13, E = 14, F = 15, G = 16, H = 17, ya que en base 18 los caracteres válidos van del 0 al 17.

Se puede notar que el número menor siempre es el 0 y que el mayor es el que corresponde a (base − 1).

Esas cantidades expresadas en cualquier sistema numérico pueden ser convertidas a otro sistema existente o no, de tal forma que se puede establecer que para pasar de un sistema X cualquiera a decimal

se representa en notación exponencial, y para pasar de decimal a un sistema W cualquiera se divide la parte entera entre la base a la que se desea convertir y la parte fraccionaria se multiplica por la base en cuestión, como se muestra en el diagrama de la figura 1.1.

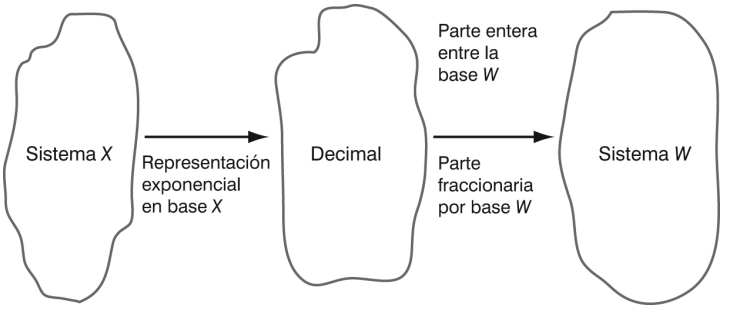


Figura 1.1 Conversión entre diferentes sistemas numéricos.

Ejemplo 1.8

Convertir $\text{CD057.EC}_{(15)}$ a base 20.

Solución. Hay que observar que en este caso no es posible usar tablas de equivalencias, por lo tanto no queda otra opción que usar el método general y convertir primero a decimal utilizando la representación exponencial y posteriormente a base 20, de acuerdo con el diagrama de la figura 1.1:

$$\begin{aligned}\text{CD057.EC}_{(15)} &= 12\times 15^4 + 13\times 15^3 + 5\times 15^1 + 7\times 15^0 + 14\times 15^{-1} + 12\times 15^{-2}\\ &= 651457.9866_{(10)}\end{aligned}$$

A continuación se hace la conversión a base 20:

Parte entera			Resto	Parte fraccionaria			Entero
$651457/20$	$=$	32572	17	0.9866×20	$=$	19.732	19
$32572/20$	$=$	1628	12	0.732×20	$=$	14.64	14
$1628/20$	$=$	81	8	0.64×20	$=$	12.8	12
$81/20$	$=$	4	1	0.8×20	$=$	16.0	16
$4/20$	$=$	0	4				

Finalmente se puede concluir que:

$$\text{CD057.EC}_{(15)} = \text{418CH.JECG}_{(20)}$$

Aquí se debe de observar que en la base 20 pueden existir caracteres del 0 al 19, y por tanto se tiene que A = 10, B = 11, C = 12, D = 13, E = 14, F = 15, G = 16, H = 17, I = 18, J = 19.

1.5 Operaciones básicas

Las operaciones básicas de suma, resta, multiplicación y división que se realizan en el sistema decimal, también se pueden llevar a cabo en cualquier sistema numérico aplicando las mismas reglas y teniendo en cuenta la base en que se encuentran los números con que se efectúa la operación.

Es importante observar que las cantidades que se estén operando se deben de encontrar en la misma base, y en caso de no ser así lo primero que se debe de hacer es la conversión correspondiente.

A continuación se realizarán operaciones básicas de suma, resta, multiplicación y división en los sistemas decimal, binario, octal y hexadecimal. El sistema decimal permite ilustrar el procedimiento a seguir en cada una de las operaciones aritméticas, gracias a la familiaridad que se tiene con él, y los sistemas binario, octal y hexadecimal son de gran utilidad en el área de la computación.

Operación matemática

Una operación matemática es la acción de un operador que aplica sobre una selección de elementos de un conjunto, y que los relaciona con otro elemento de un conjunto final. La aplicación de un operador en este caso se conoce como ley de composición.

1.5.1 Suma

Ejemplo 1.9

Suma en el sistema decimal:

$$
\begin{array}{cccccccccc}
 & & & 4 & 5 & 6 & . & 7 & 8_{(10)} & \\
+ & 1 & 7 & 8 & 2 & 0 & . & 6 & 4 & 9_{(10)} \\
\hline
 & 1 & 8 & 2 & 7 & 7 & . & 4 & 2 & 9_{(10)}
\end{array}
$$

Explicación por columna

$0 + 9 = 9$	El 9 es un dígito válido de base 10, por lo que se queda tal cual.
$8 + 4 = 12$	El 12 no es válido en decimal, ya que es una combinación del 1 y el 2. Cuando ocurre esto se deberá dividir entre la base (10), colocando el resto debajo de la linea y sumando el cociente a los números de la siguiente columna de la izquierda.
$1 + 7 + 6 = 14$	El 14 no es válido por lo que se divide entre la base y se procede como se hizo anteriormente.
$1 + 6 + 0 = 7$	Dígito válido en decimal
$5 + 2 = 7$	Dígito válido.
$4 + 8 + = 12$	Aqui hay que dividir entre la base.
$1 + 0 + 7 = 8$	Dígito válido.

Los espacios vacíos de los extremos, como el de arriba del 9 en el ejemplo 1.9, se consideran como 0, por esta razón se sumó $0 + 9$. Esto mismo sucede en todos los sistemas numéricos, ya que el 0 es el dígito válido más pequeño. Cuando el resultado de la suma es un símbolo válido en ese sistema numérico normalmente no se divide entre la base, pero podría dividirse ya que el cociente es 0 y por lo tanto no afecta a la siguiente columna de la izquierda y el resto es el mismo número, por ejemplo, si el resultado es 6, el cual es válido en el sistema decimal, el cociente de dividir 6 entre 10 es 0 y el resto es 6.

Ejemplo 1.10

Suma en el sistema hexadecimal:

$$\begin{array}{rcccccccc} & A & 6 & F & C & 9 & . & 7 & B & 2_{(16)} \\ + & 4 & E & 7 & D & 0 & . & 7 & 3 & E_{(16)} \\ \hline & F & 5 & 7 & 9 & 9 & . & E & F & 0_{(16)} \end{array}$$

Explicación por columna

$2 + 14 = 16$	Al dividir 16 entre la base se obtiene el cociente 1 y resto 0.
$1 + 11 + 3 = 15$	Dígito válido: 15 = F
$7 + 7 = 14$	Dígito válido: 14 = E
$9 + 0 = 9$	Dígito válido.
$12 + 13 = 25$	Al dividir 25 entre la base se obtiene el cociente 1 y resto 9.
$1 + 15 + 7 = 23$	Al dividir 23 entre la base se obtiene el cociente 1 y resto 7.
$1 + 6 + 14 = 21$	Al dividir 21 entre la base se obtiene el cociente 1 y el resto 5.
$1 + 10 + 4 = 15$	Dígito válido 15= F

Generalización de la suma

Como se puede observar, el procedimiento para llevar a cabo la suma en los diferentes sistemas numéricos no cambia, sino que sólo hay que tener en cuenta la base en que se realiza la operación.

Por tanto, en general se puede establecer, que si al sumar dos dígitos, el resultado de la suma sobrepasa al dígito mayor de un sistema numérico determinado, entonces el resultado se debe dividir entre la base del sistema y el residuo de esa división se pone debajo de la línea y el cociente se suma a la columna siguiente izquierda.

Ejemplo 1.11

$$\begin{array}{rrrrrrrrrr} & & H & 7 & 2 & 5 & . & 4 & A_{(18)} \\ + & & & G & B & D & . & 7 & H\ 2_{(18)} \\ \hline & 1 & 0 & 5 & E & 0 & . & C & 9\ 2_{(18)} \end{array} \qquad \begin{array}{rrrrrrrrrr} & & & 2 & 3 & 0 & 3 & 1 & 1 & . & 2\ 1_{(5)} \\ + & 3 & 1 & 3 & 2 & 0 & 0 & 3 & . & 3 & 2\ 1_{(5)} \\ \hline & 3 & 4 & 1 & 2 & 3 & 2 & 0 & . & 0 & 3\ 1_{(5)} \end{array}$$

$$\begin{array}{rrrrrrrrrr} & 9 & 8 & 0 & B & 3 & . & 2 & 5_{(13)} \\ + & & A & 7 & 2 & 8 & . & 0 & 9_{(13)} \\ \hline & A & 5 & 8 & 0 & B & . & 3 & 1_{(13)} \end{array} \qquad \begin{array}{rrrrrrrrrr} & & & K & 0 & J & 7 & . & L & 2_{23} \\ + & 2 & 7 & C & M & E & . & F & A_{23} \\ \hline & 3 & 4 & D & I & M & . & D & C_{23} \end{array}$$

1.5.2 Resta

Ejemplo 1.12

Resta en el sistema decimal:

$$\begin{array}{rrrrrrrrr} & 8 & 1 & 2 & 7 & . & 5 & 8 & 0_{10} \\ - & 5 & 8 & 3 & 1 & . & 9 & 6 & 4_{10} \\ \hline & 2 & 2 & 9 & 5 & . & 6 & 1 & 6_{10} \end{array}$$

Explicación por columna

$(0 + 10) - 4 = 6$	Cuando el sustraendo es mayor al minuendo, como ocurre en la primera columna, se deberá sumar la base al minuendo y después llevar a cabo la sustracción (minuendo + 10) − sustraendo = resultado.
$8 - (6 + 1) = 1$	Cuando en la columna anterior se sumó 10 al minuendo, en la columna siguiente de la izquierda se deberá sumar 1 al sustraendo (minuendo − (sustraendo + 1)) = resultado. Si después de sumar 1 al sustraendo éste es mayor que el minuendo, entonces se deberá sumar la base al minuendo antes de llevar a cabo la resta.
$(5 + 10) - 9 = 6$	Debido a que sustraendo > minuendo, se suma la base al minuendo y se realiza la resta.
$7 - (1 + 1) = 5$	Como en la columna anterior se le sumó la base 10 al minuendo, en esta columna se le deberá sumar 1 al sustraendo.
$(2 + 10) - 3 = 9$	Como sustraendo > minuendo, se suma la base al minuendo.
$(1 + 10) - (8 + 1) = 2$	Como se le sumó la base a la columna anterior, se deberá aumentar en 1 el sustraendo y como sustraendo < minuendo, se deberá sumar la base al minuendo antes de llevar a cabo la resta. El orden en que se llevan a cabo los incrementos en este caso es muy importante.
$8 - (5 + 1) = 2$	Sumar 1 al sustraendo, ya que en la columna anterior se le sumó la base al minuendo, y después llevar a cabo la resta.

Al efectuar la resta es necesario revisar si el sustraendo es mayor que el minuendo, ya que en caso afirmativo se debe sumar la base al minuendo antes de llevar a cabo la resta de dos dígitos de una columna cualquiera. Una vez comenzada la operación de resta cuando al minuendo se le suma la base, entonces al sustraendo de la columna izquierda próxima se le deberá sumar 1 (ya que en este caso la base es 10) antes de hacer la comparación entre el minuendo y sustraendo. En el caso de otro sistema numérico, lo que se le suma al minuendo debe de ser la base que corresponda (8 en octal, 16 hexadecimal

o 2 en binario), sin embargo cuando se le suma la base al minuendo invariablemente será 1 lo que se incrementa el sustraendo de la columna izquierda próxima, independientemente del sistema numérico de que se trate (ya que ese 1 significa que se le sumó una vez la base en la columna anterior).

Ejemplo 1.13

Resta en el sistema octal:

$$\begin{array}{rccccccccc} & 4 & 1 & 0 & 7 & 2 & . & 1 & 4_{(8)} & \\ - & 3 & 6 & 0 & 4 & 3 & . & 7 & 1 & 3_{(8)} \\ \hline & 0 & 3 & 0 & 2 & 6 & . & 2 & 2 & 5_{(8)} \end{array}$$

Explicación por columna

$(0 + 8) - 3 = 5$	Como sustraendo > minuendo, hay que sumar la base al minuendo.
$4 - (1 + 1) = 2$	Sumar 1 al sustraendo debido a que se sumó la base al minuendo en la columna anterior.
$(1 + 8) - 7 = 2$	Sumar la base al minuendo, ya que sustraendo > minuendo.
$(2 + 8) - (3 + 1) = 6$	Primero sumar 1 al sustraendo, ya que se aumentó la base en la columna anterior, después sumar la base al minuendo, ya que sustraendo > minuendo.
$7 - (4 + 1) = 2$	Sumar 1 al sustraendo debido a que se sumó la base al minuendo en la columna anterior.
$0 - 0 = 0$	Sin cambio, ya que no se cumple que sustraendo > minuendo ni se sumó la base en la columna anterior.

$(1 + 8) - 6 = 3$	Sumar la base al minuendo, ya que sustraendo > minuendo.
$4 - (3 + 1) = 0$	Sumar 1 al sustraendo, ya que se sumó la base al minuendo.

Generalización de la resta

En forma general se puede decir que si en la primera columna se cumple la condición sustraendo > minuendo, entonces se deberá sumar la base al minuendo y después se realizará la resta; en caso de que no se cumpla la condición, solamente se hace la resta. A partir de la segunda columna se sumará 1 al sustraendo si en la columna anterior se sumó la base al minuendo (en caso contrario se deja tal cual) y después se observará si sustraendo > minuendo, si esto es verdadero se le agrega la base al minuendo para finalmente llevar a cabo la resta correspondiente en cada una de las columnas hasta terminar.

Ejemplo 1.14

$$\begin{array}{rcccccccc} & 8 & 5 & A & C & 3 & . & 7 & 0 & E_{(17)} \\ - & & B & 5 & C & 6 & . & F & 2 & {}_{(17)} \\ \hline & 7 & B & 4 & G & D & . & 8 & F & E_{(17)} \end{array} \qquad \begin{array}{rcccccccc} & 4 & 0 & 5 & 8 & 2 & 7 & 1 & . & 6_{(9)} \\ - & 1 & 5 & 8 & 3 & 5 & 8 & 2 & . & 2\ 8\ 4_{(9)} \\ \hline & 2 & 3 & 6 & 4 & 5 & 7 & 8 & . & 3\ 0\ 5_{(9)} \end{array}$$

$$\begin{array}{rcccccccc} & 2 & 3 & 1 & K & A & . & 4 & 3_{(21)} \\ - & & 1 & G & 7 & H & . & C & 2_{(21)} \\ \hline & 2 & 1 & 6 & C & D & . & D & 1_{(21)} \end{array} \qquad \begin{array}{rcccccccc} & 7 & 4 & 1 & 0 & 0 & . & A & 5_{(14)} \\ - & & 5 & C & 5 & 3 & . & C & 2_{(14)} \\ \hline & 6 & C & 2 & 8 & A & . & C & 3_{(14)} \end{array}$$

1.5.3 Multiplicación

La forma en que se multiplica en decimal es la misma en que se llevan a cabo las multiplicaciones en otros sistemas numéricos, la única diferencia es la base.

Ejemplo 1.15

Multiplicación en el sistema decimal

		8	0	5	7	.	2	$3_{(10)}$
$\times$				5	3	.	$7_{(10)}$	
		5	6	4	0	0	6	1
	2	4	1	7	1	6	9	
4	0	2	8	6	1	5		
4	3	2	6	7	3	.2	5	$1_{(10)}$

Explicación por columna

$7 \times 3 = 21$ Como el 21 no es un dígito válido en decimal, se divide entre la base para obtener cociente = 2 y resto = 1. El resto se coloca debajo de la línea y el cociente se suma en el producto de la siguiente columna.

$7 \times 2 + 2 = 16$ Como el 16 no es un dígito válido en el sistema decimal, se realiza lo mismo que en el caso anterior para obtener cociente = 1 y resto = 6.

El procedimiento seguido en el sistema decimal es el que se realiza en cualquier sistema numérico, tomando en cuenta que cuando la cantidad resultante no es un dígito válido en dicho sistema entonces se debe dividir entre la base; en octal se debe dividir entre 8, en hexadecimal entre 16, en vigesimal entre 20 y así sucesivamente. La forma en que se suman en el sistema decimal los resultados obtenidos en las diferentes columnas, es la misma en que se suman en otro sistema. Como se muestra en el ejemplo 1.15, en la primera columna se baja el 1 porque no tiene otro dígito con qué sumarse, en la segunda columna se suma $6 + 9 = 15$ y como el 15 no es un dígito válido en decimal entonces se debe dividir entre la base

para obtener cociente = 1 y resto = 5, el resto se pone debajo de la línea y el cociente se suma a los dígitos de la siguiente columna de la izquierda, y así se continúa hasta terminar.

Ejemplo 1.16

Multiplicación en el sistema binario:

		1	0	0	1	1	.	0	$1_{(2)}$
x						1	.1	0	$1_{(2)}$
			1	0	0	1	1	0	1
		0	0	0	0	0	0	0	
	1	0	0	1	1	0	1		
1	0	0	1	1	0	1			
1	1	1	1	1	.0	1	0	0	1

Entre menor sea la base del sistema numérico es más sencillo realizar operaciones aritméticas, ya que el número de dígitos válidos también se reduce en la misma proporción. Como se muestra en el ejemplo 1.16, en el sistema binario solamente hay posibilidades de 0 ó 1 y esto simplifica el trabajo.

En cualquier sistema, al multiplicar una cantidad por 1 se obtiene la misma cantidad, por esa razón en el sistema binario al multiplicar 1 por el multiplicando resulta el mismo multiplicando y al multiplicar 0 por el multiplicando resulta una fila de ceros. Al sumar las columnas se pueden observar casos como el que ocurre en la tercera columna de derecha a izquierda, donde $1 + 0 + 1 = 2$, pero como el 2 no es un dígito válido en binario se debe dividir entre la base, obteniéndose cociente = 1 y resto = 0 por lo que el resto se coloca debajo de la línea y el cociente se suma con los dígitos de la siguiente columna, de forma que en la cuarta columna se deben sumar $1 + 1 + 0 + 0 + 1 = 3$, donde el primer 1 es el cociente de la línea anterior. Como el 3 no es un dígito válido en binario, se divide entre la base y se obtiene cociente = 1 y resto = 1. En lo que sigue se procede de la misma manera hasta terminar de sumar todas las columnas. El punto que separa a la parte entera de la fraccionaria se coloca también de manera semejante a como se realiza en el sistema decimal: se cuentan los decimales tanto del multiplicando como del multiplicador.

Como el procedimiento para multiplicar en cualquier sistema es el mismo que el que se utiliza en el sistema decimal, lo único que se debe tener presente es la base en que se está trabajando.

Ejemplo 1.17

		5	9	A	C	.	1	$2_{(14)}$
×						B	.8	$9_{(14)}$
		3	9	3	D	A	A	4
	3	3	8	2	C	9	2	
4	6	9	7	6	C	8		
4	A	2	A	D	.A	D	C	$4_{(14)}$

		A	5	3	7	5	.	$C_{(13)}$
×					4	.B	9	$8_{(13)}$
		6	5	3	2	7	8	5
	7	2	8	6	2	1	4	
8	A	6	0	4	0	2		
3	2	8	1	3	A	9		
3	C	0	3	4	B	.0	A	C $5_{(13)}$

		C	5	9	D	A	.	$2_{(15)}$
×					6	B	.4	$5_{(15)}$
		4	1	D	4	8	5	A
	3	4	7	9	9	A	8	
9	1	2	4	0	6	7		
4	E	3	E	7	0	C		
5	8	8	A	5	9	2	.A	D $A_{(15)}$

	F	3	A	C	7	.	$3_{(17)}$	
×				0	.5	D	$4_{(17)}$	
	3	9	E	8	F	B	C	
B	A	D	3	8	8	5		
4	8	1	2	B	1	F		
5	2	F	8	C	.2	4	G	$C_{(17)}$

1.5.4 División

Se sabe que la división involucra operaciones de resta y multiplicación, por lo que es más complicada que las tres operaciones aritméticas anteriores. En este caso lo que se recomienda es usar lo que se conoce como división desarrollada, la cual permite realizar primero la multiplicación y después la resta, ya que de otra forma el tratar de llevar a cabo tanto la multiplicación como la resta en un sistema numérico con el que no se está familiarizado podría ser muy complicado.

Ejemplo 1.18

Dividir en el sistema decimal:

$$7\ .6\ 9_{(10)} \quad \overline{)\ 4\ 3\ 2\ 5\ 0\ .1\ 8\ 2_{(10)}}$$

Divisor Dividendo

En una división en el sistema decimal el dividendo puede tener o no punto decimal, pero el divisor no debe tenerlo o bien lo debe tener al final. En este ejemplo se debe recorrer el punto decimal dos posiciones, para mandar el punto decimal al extremo derecho del divisor. Si se recorre el punto decimal cierto número de posiciones en el divisor, también se debe recorrer esas mismas posiciones en el dividendo. En todos los casos en que sea necesario, esto se debe de hacer antes de llevar a cabo la división, independientemente del sistema numérico de que se trate.

				5	**6**	**2**	**4**	**.2**	1_{10}	← Cociente
7	6	$9_{(10)}$	4	3	2	5	0	1	8	$.2_{(10)}$
			3	8	4	5				
			0	4	8	0	0			
				4	6	1	4			
				0	1	8	6	1		
					1	5	3	8		
					0	3	2	3	8	
						3	0	7	6	
							1	6	2	2
							1	5	3	8
							0	0	8	4	0
									7	6	9
Resto →									0	7	1

Después de recorrer el punto decimal al lado derecho del dígito menos significativo del divisor, y de recorrer ese mismo número de posiciones el punto decimal en el dividendo, se procede a llevar a cabo la división.

Como el divisor tiene tres dígitos y en este caso no cabe ninguna vez en tres dígitos del dividendo ya

que $769_{(10)} > 432_{(10)}$, se toman cuatro dígitos del dividendo y se encuentra que cociente = 5, después se multiplica este cociente por el divisor, $5 \times 769_{(10)} = 3845_{(10)}$, y dicho resultado se resta de los cuatro dígitos del dividendo para obtener $4325_{(10)} - 3845_{(10)} = 0480_{(10)}$, luego se baja el siguiente dígito, se encuentra el cociente, se multiplica dicho cociente por el divisor y se resta del dividendo y así sucesivamente hasta terminar. Es importante mencionar que si después de bajar un dígito más del dividendo, el divisor no cabe ninguna vez en dicha cantidad, entonces se pondrá 0 como cociente y se bajará otro dígito. También, una vez que se han bajado todos los dígitos del dividendo y se desean más decimales, se deberán agregar más ceros a la cantidad que resulta de la resta. Todo esto ya se sabe, porque se está familiarizado con el sistema decimal y es válido en todos los sistemas.

Ejemplo 1.19

División en el sistema hexadecimal. Dividir $1AF578.2B_{(16)}$ entre $A.7E2_{(16)}$, obtener una cifra después del punto hexadecimal y hacer la comprobación correspondiente de la división:

					2	**9**	**1**	**B**	**D**	**.D**$_{(16)}$	
A	7	E	2.$_{(16)}$	1	A	F	5	7	8	2	B	0$_{(16)}$	
				1	4	F	C	4					
				0	5	F	9	3	8				
					5	E	6	F	2				
					0	1	2	4	6	2			
							A	7	E	2			
							7	C	8	0	B		
							7	3	6	B	6		
							0	9	1	5	5	0	
								8	8	6	7	A	
									8	E	D	6	0
									8	8	6	7	A
										6	6	E	6

En este ejemplo se puede observar que el dividendo no tiene parte fraccionaria, ya que al recorrer el punto hexadecimal incluso fue necesario agregar un 0, de forma que para obtener un dígito más después del punto hexadecimal fue necesario agregar un 0 al resto. Para comprobar la división es necesario multiplicar el divisor por el cociente y sumar el resto al resultado, como se muestra a continuación:

				2	9	1	B	D	.D$_{(16)}$
×						A	7	E	2$_{(16)}$
				5	2	3	7	B	A
		2	3	F	8	6	1	6	
	1	1	F	C	3	0	B		
1	9	B	1	6	A	2			
1	A	F	5	7	7	C	4	1	A$_{(16)}$
				+		6	6	E	6
1	A	F	5	7	8	2	B	0	.0

Notar que el punto que separa la parte entera de la parte fraccionaria se coloca después de que se suma el residuo al resultado de multiplicar el cociente por el divisor.

Ejemplo 1.20

División en otros sistemas. El procedimiento para llevar a cabo las divisiones en cualquier sistema no cambia, ya que se trata de sistemas posicionales y lo único que debe tenerse en cuenta es la base en que se está trabajando.

							B	**7**	**D**	**.6$_{(17)}$**
9	D	4.$_{(17)}$	6	A	0	C	8	7	.9$_{(17)}$
			6	5	9	A			
			0	4	8	2	8		
				4	0	7	B		
				0	7	B	E	7	
					7	8	2	1	
					0	3	C	6	9
						3	7	B	7
						0	4	C	2

$$
\begin{array}{cccc|ccccccccc}
 & & & & & & & \mathbf{4} & \mathbf{A} & \mathbf{9} & \mathbf{A} & \mathbf{.B}_{(12)} \\
7 & 6 & 1 & 3._{(12)} & 3 & 0 & 9 & 8 & 5 & 6 & 7 & 2_{(12)} & \\
 & & & & \underline{2} & \underline{6} & \underline{0} & \underline{5} & \underline{0} & & & & \\
 & & & & 0 & 6 & 9 & 3 & 5 & 6 & & & \\
 & & & & & \underline{6} & \underline{3} & \underline{1} & \underline{0} & \underline{6} & & & \\
 & & & & & 0 & 6 & 2 & 5 & 0 & 7 & & \\
 & & & & & & 5 & 7 & 6 & B & \underline{3} & & \\
 & & & & & & 0 & 6 & A & 1 & 4 & 2 & \\
 & & & & & & & \underline{6} & \underline{3} & \underline{1} & \underline{0} & \underline{6} & \\
 & & & & & & & 0 & 7 & 0 & 3 & 8 & 0 \\
 & & & & & & & & \underline{6} & \underline{A} & \underline{7} & \underline{1} & \underline{9} \\
 & & & & & & & & 0 & 1 & 8 & 6 & 3
\end{array}
$$

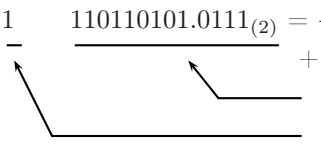
1.6 Suma de dos cantidades en complemento a 2

Las operaciones que la computadora realiza internamente se llevan a cabo en una forma muy particular. En principio el sistema numérico utilizado es el binario y la operación básica es la suma. En computación las cantidades se representan por un conjunto de bits (ceros y unos), usando un bit exclusivo para distinguir las cantidades negativas de las positivas, el cual recibe el nombre de "bit de signo". La convención más común para el signo es 0 = positivo y 1 = negativo.

Existen tres formas de representar cantidades: magnitud verdadera, complemento a 1 y complemento a 2; cada una de estas formas tiene su utilidad dentro de la computación.

Magnitud verdadera

En la representación en magnitud verdadera se muestran los bits en forma real, y una característica de este tipo de representación es que se puede saber fácilmente a cuánto equivale ese conjunto de bits en el sistema decimal usando para ello la representación exponencial como se presenta en la siguiente expresión:



$$= -\,(1\times 2^{8} + 1\times 2^{7} + 1\times 2^{5} + 1\times 2^{4} + 1\times 2^{2} + 1\times 2^{0} + 1\times 2^{-2} + 1\times 2^{-3} + 1\times 2^{-4}\,) = \text{-}437.44_{(10)}$$

Complemento a 1

Como en el sistema binario solamente existen como dígitos válidos el 0 y el 1, se dice que el complemento de 0 es 1 y el complemento de 1 es 0. El complemento de un número en binario se obtiene complementando cada uno de los bits, sin considerar el signo, como se muestra a continuación:

1	1 0 1 0 1 1 1 0 0 1 0 0 1 . 0 1 $_{(2)}$	Magnitud verdadera
1	0 1 0 1 0 0 0 1 1 0 1 1 0 . 1 0 $_{(2)}$	Complemento a 1

0	1 0 0 0 1 0 0 1 1 . 1 0 0 $_{(2)}$	Magnitud verdadera
0	0 1 1 1 0 1 1 0 0 . 0 1 1 $_{(2)}$	Complemento a 1

Como se puede observar, para obtener el complemento a 1 de una cantidad expresada en binario es suficiente cambiar todos los ceros por unos y los unos por ceros, pero en ningún momento se cambia el bit de signo, que en este caso es el bit de la extrema izquierda.

Complemento a 2

El complemento a 2 se obtiene sumando 1 al bit menos significativo del complemento a 1, como se muestra en el siguiente caso:

1	0	1	0	1	0	0	0	1	1	0	1	1	0	.	1	$0_{(2)}$	Complemento a 1
													+			1	
1	0	1	0	1	0	0	0	1	1	0	1	1	0	.	1	$1_{(2)}$	Complemento a 2

0	0	1	1	1	0	1	1	0	0	.	0	1	$1_{(2)}$	Complemento a 1
									+				1	
0	0	1	1	1	0	1	1	0	0	.	1	0	$0_{(2)}$	Complemento a 2

Una multiplicación es una sucesión de sumas y una división es una sucesión de restas. Como se mencionó anteriormente, la computadora no realiza restas, ni multiplicaciones ni divisiones, sino únicamente sumas. Cuando las dos cantidades a sumar son positivas se suman tal cual, pero cuando alguna de ellas es negativa (lo que equivale a restar una cantidad de otra) entonces la cantidad negativa se complementa a 2 y después se suma a la otra cantidad, de forma que una resta se convierte en una suma.

Supóngase que se definen las variables A, B y C del tipo entero, por lo que ocupa 1 byte de memoria cada una de ellas. Si $A = 225$, $B = 76$ y en alguna línea de un programa se tiene que $C = A + B$, entonces lo que la computadora hace es lo siguiente: primero convierte los valores de A y B a binario, y luego realiza la suma de la siguiente forma

$$
\begin{array}{llllll}
+ & 2 & 2 & 5_{(10)} = & 0 & 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1_{(2)} \\
+ & & 7 & 6_{(10)} = & 0 & 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0_{(2)} \\
\hline
+ & 3 & 0 & 1_{(10)} & 1 & 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1_{(2)}
\end{array}
$$

Signo Magnitud

El resultado obtenido es $1\ 00101101_{(2)} = -45_{(10)}$ que es muy diferente al $+301_{(10)}$ esperado. Lo que ocurrió aquí es que se presentó un "desbordamiento" al querer guardar en la variable C definida de 8 bits, una cantidad mayor. Este error es común que ocurra durante el proceso de la programación, ya que se definen las variables de cierto tipo y con cierta capacidad y algunas veces se desea guardar en ellas una cantidad que sobrepasa esa capacidad. La finalidad de citar este caso es mostrar que se deben de considerar todos los elementos que se presentan en el momento en que la computadora realiza una operación aritmética.

Para resolver el problema de desbordamiento es conveniente definir las variables con una capacidad mayor, por ejemplo suponer que las variables A, B y C son enteras, pero ahora con una capacidad de 16 bits, que es lo que realmente ocurre en un programa cuando se definen variables con capacidad menor a la requerida.

$$
\begin{array}{llllll}
+ & 2 & 2 & 5_{(10)} = & 0 & 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1_{(2)} \\
+ & & 7 & 6_{(10)} = & 0 & 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0_{(2)} \\
\hline
+ & 3 & 0 & 1_{(10)} & 0 & 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1_{(2)}
\end{array}
$$

Signo Magnitud

Se observa ahora que $+301_{(10)} = 0\ 0000000100101101_{(2)}$ es el resultado correcto, con lo cual se evita el desbordamiento.

Es importante mencionar que el desbordamiento solamente ocurre cuando las dos cantidades que se están sumando son del mismo signo, ya que son los únicos casos en que el resultado puede requerir mayor

espacio. Cuando las cantidades a sumar son de signo contrario no se presenta el desbordamiento, pues el valor absoluto del resultado siempre será menor al valor absoluto de alguna de las cantidades que se suman.

Ejemplo 1.21

Sumar A = $-225_{(10)}$ con B = $+76_{(10)}$.

Cuando una cantidad es negativa, se deberá encontrar el complemento a 2 de esa cantidad y después realizar la suma, como se muestra a continuación.

$-\ 2\ 2\ 5_{(10)} =$	$1\ \ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1_{(2)}$	Magnitud verdadera
	$1\ \ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0_{(2)}$	Complemento a 1
$=$	$\qquad\qquad\qquad\qquad\qquad\qquad\qquad 1$	
	$1\ \ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1_{(2)}$	Complemento a 2

Nótese que para obtener el complemento a 1 se cambian todos los bits por su complemento, pero el bit de signo *no* se cambia. Para encontrar el complemento a 2 se le suma 1 al bit menos significativo del complemento a 1.

Ahora sí se procede a sumar el complemento a 2 de la cantidad negativa y la otra cantidad positiva.

$-\ 2\ 2\ 5_{(10)} =$	$1\ \ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1_{(2)}$	
$+\ \ 7\ 6_{(10)} =$	$0\ \ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0_{(2)}$	
$-\ 1\ 4\ 9_{(10)}$	$1\ \ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1_{(2)}$	

Signo Magnitud

El resultado obtenido es negativo, como se esperaba, pero la magnitud obtenida no es la correcta, ya que $1\ 01101011_{(10)} = -107_{(10)}$ es diferente de $-149_{(10)}$. En forma general se puede decir que *si el resultado de la suma es negativo, se deberá complementar a 2 el resultado.*

	$1\ \ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1_{(2)}$	Resultado negativo
	$1\ \ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0_{(2)}$	Complemento a 1
	$\qquad\qquad\qquad\qquad\qquad\qquad 1$	
$-\ 1\ 4\ 9_{(10)} =$	$1\ \ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1_{(2)}$	Complemento a 2

Ejemplo 1.22

Sumar A = $+225_{(10)}$ con B = $-76_{(10)}$.

Complementando a 2 la cantidad negativa se tiene:

$-\ 7\ 6_{(10)}$	=	1	0	1	0	0	1	1	0	$0_{(2)}$	Magnitud verdadera
		1	1	0	1	1	0	0	1	$1_{(2)}$	Complemento a 1
									1		
	=	1	1	0	1	1	0	1	0	$0_{(2)}$	Complemento a 2

Sumando el complemento encontrado a la cantidad positiva se obtiene:

$+\ 2\ 2\ 5_{(10)}$	=		0		1	1	1	0	0	0	0	$1_{(2)}$
$-\ \ 7\ 6_{(10)}$	=		1		1	0	1	1	0	1	0	$0_{(2)}$
$+\ 1\ 4\ 9_{(10)}$		1	0		1	0	0	1	0	1	0	$1_{(2)}$
		Acarreo	Signo		Magnitud							

En esta suma se puede observar que el resultado es positivo, por lo tanto es el resultado correcto, ya que solamente se complementan a 2 los resultados negativos. También se obtiene un acarreo, el cual se debe despreciar en todos los casos.

Ejemplo 1.23

Sumar A = $-225_{(10)}$ con B = $-76_{(10)}$.

Cuando una cantidad es negativa, se debe determinar el complemento a 2 de esa cantidad. En este caso las dos cantidades a sumar son negativas, por lo tanto se tiene que obtener el complemento a 2 de ambas antes de realizar la suma. Pero también se debe tomar en cuenta que la suma produce un desbordamiento, de tal forma que en el complemento ya se debe trabajar con los bits correctos. Complementando a 2 ambas cantidades y considerando el desbordamiento:

$-\ 2\ 2\ 5_{(10)}$	=	1	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	$1_{(2)}$	Magnitud verdadera
		1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	$0_{(2)}$	Complemento a 1
																		1	
	=	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	$1_{(2)}$	Complemento a 2

$-\ 7\ 6_{(10)}$	=	1	0	0	0	0	0	0	0	0	1	0	0	1	1	0	$0_{(2)}$	Magnitud verdadera
		1	1	1	1	1	1	1	1	1	0	1	1	0	0	1	$1_{(2)}$	Complemento a 1
																	1	
	=	1	1	1	1	1	1	1	1	1	0	1	1	0	1	0	$0_{(2)}$	Complemento a 2

Sumando:

$-\ 2\ 2\ 5_{(10)}$	=		1		1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	$1_{(2)}$
$-\ \ 7\ 6_{(10)}$	=		1		1	1	1	1	1	1	1	1	1	0	1	1	0	1	0	$0_{(2)}$
$-\ 3\ 0\ 1_{(10)}$		1	1		1	1	1	1	1	1	1	0	1	1	0	1	0	0	1	$1_{(2)}$

Acarreo Signo Magnitud

Si se convierte a decimal el resultado obtenido es posible observar que no es el esperado de -$301_{(10)}$. Sin embargo, se sabe que cuando el resultado de la suma es negativo se deberá complementar a 2. En este caso también se tiene acarreo, el cual se desprecia.

	1	1	1	1	1	1	1	0	1	1	0	1	0	0	1	$1_{(2)}$	Resultado negativo
	1	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	$0_{(2)}$	Complemento a 1
																	1	
$-\ 3\ 0\ 1_{(10)}$ =	1	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	$1_{(2)}$	Complemento a 2

Conclusiones de las sumas en complemento a 2

De la misma manera en que se sumaron dos cantidades enteras en complemento a 2, también es posible sumar dos cantidades con una parte entera y otra fraccionaria, siempre y cuando se tenga en cuenta que el desbordamiento sólo puede darse en la parte entera (ya que las operaciones se llevan a cabo de derecha a izquierda) y que para evitar éste se debe trabajar con el número de bits suficiente. Para encontrar el complemento a 1 se cambia cada bit por su complemento, independientemente de si se encuentra en la parte entera o en la parte fraccionaria. Para encontrar el complemento a 2 se suma un 1 en el bit menos significativo (el que esté más a la derecha) independientemente de si pertenece a la parte entera o bien a la parte fraccionaria, y sólo se complementan a 2 las cantidades negativas y los resultados negativos de las sumas.

1.7 Multiplicación de dos cantidades usando el algoritmo de Booth

El algoritmo de Booth fue desarrollado por Andrew Donald Booth con la finalidad de aumentar la velocidad al llevar a cabo la multiplicación de dos cantidades usando el concepto de complemento a dos. Para multiplicar dos cantidades, por ejemplo $1600_{(10)}$ por $750_{(10)}$, usando solamente sumas se tendría que sumar 1600 veces $750_{(10)}$ o bien $750_{(10)}$ veces $1600_{(10)}$ algo que es muy tardado, sin embargo por medio del algoritmo de Booth se puede realizar la multiplicación de dos cantidades con pocos pasos.

El número de pasos requeridos para llevar a cabo la multiplicación depende del número de bits que se utilicen en dicha operación, y el número de bits usados está directamente relacionado con los bits que requiere la mayor de las dos cantidades que se está multiplicando. No hay una regla, pero se ha observado que el número de bits mínimo de trabajo es:

$$\text{Bits de trabajo} = \text{Bits ocupados de la cantidad mayor} + 1$$

En la práctica es recomendable trabajar con 1, 2, 4, 8, 16, ..., 2^n bits. Por ejemplo si la cantidad mayor a multiplicar (ya sea el multiplicando o el multiplicador) ocupa seis bits, es recomendable trabajar con por lo menos siete bits, pero podría llevarse a cabo la operación con 8, 16, ... 2^n bits. El resultado de la multiplicación es correcto si se usan más bits de los necesarios, pero el número de pasos que se deben realizar está en función de los bits y también se incrementa. Se recomienda tomar en cuenta la fórmula anterior para usar solamente el menor número de bits requerido y por lo tanto el menor número de pasos necesarios para llevar a cabo la multiplicación.

Multiplicación por algoritmo de Booth

Para llevar a cabo la multiplicación por medio del algoritmo de Booth se utilizan tres cantidades:

- **M**: Multiplicando.
- **C**: Variable complementaria.
- **P**: Multiplicador, pero a la vez variable para guardar el resultado de la multiplicación.

Dependiendo del signo de **M** y **P** podrían complementarse dichas variables. El valor de **C** depende directamente de **M**. En la tabla siguiente se muestran los casos posibles.

Operación	C	M	P
$+\mathbf{M}$ $\times \quad +\mathbf{P}$	Complemento a dos de **M** con signo negativo.	No cambia.	No cambia.
$-\mathbf{M}$ $\times \quad +\mathbf{P}$	**M** con signo positivo.	Complemento a dos de **M**, con signo negativo.	No cambia.
$-\mathbf{M}$ $\times \quad -\mathbf{P}$	**M** con signo positivo.	Complemento a dos de **M**, con signo negativo.	Complemento a dos de **P**, con signo positivo.
$+\mathbf{M}$ $\times \quad -\mathbf{P}$	Complemento a dos de **M** con signo negativo.	No cambia.	Complemento a dos de **P**, con signo positivo.

Hay que observar que **C** es el complemento de **M**, solamente cuando **M** es positiva y $\mathbf{C} = +\mathbf{M}$ cuando **M** es negativa. **M** y **P** por su parte se complementan a dos solamente cuando son negativas, pero **P** siempre tendrá signo positivo independientemente de si se complementa o no.

Andrew Donald Booth (1918-2009)

Ingeniero eléctrico, físico e informático que inventó la memoria de tambor magnético para guardar información en las computadoras. Famoso también por inventar el algoritmo de Booth para la multiplicación.

Trabajó como físico matemático en el equipo de rayos X con la compañía británica productora de caucho. En 1944 obtuvo un doctorado en "cristalografía" en la Universidad de Birmingham del Reino Unido. Constructor de una de las primeras computadoras electrónicas, Booth fue fundador del departamento de Birkbeck encargado de la automatización automática numérica y llevó a cabo trabajos en la traducción automática; también se desempeñó como presidente de la Universidad de Lakehead de 1972 hasta 1978.

Ejemplo 1.24

Supóngase que se desea multiplicar

(a) $+17_{(10)} \times -14_{(10)}$ **(b)** $+17_{(10)} \times +14_{(10)}$

(c) $-17_{(10)} \times -14_{(10)}$ **(d)** $-17_{(10)} \times +14_{(10)}$

Se sabe que $17_{(10)} = 10001_{(2)}$ y $14_{(10)} = 1110_{(2)}$. El multiplicando se representa por cinco bits y el multiplicador por cuatro. Por lo tanto el número de bits requeridos para realizar la multiplicación es:

$$\text{Bits de trabajo} = 5 + 1 = 6$$

El número de bits de trabajo es también el número de pasos que se deben realizar en la multiplicación.

En el inciso **(a)** se tiene que: $(+\mathbf{M}) \times (-\mathbf{P})$ por lo tanto es el último de los casos de la tabla, de tal manera que **C** es el complemento a dos de **M** con signo negativo, como se muestra a continuación:

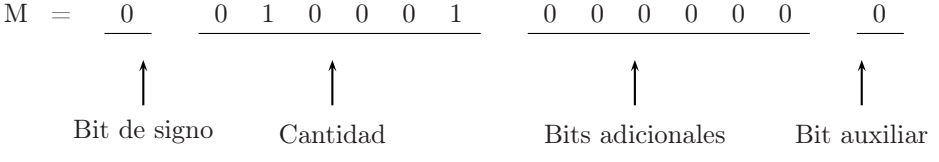
M	=	1	0	1	0	0	0	1	Multiplicando
		1	1	0	1	1	1	0	Complemento a uno de M
							+	1	
C	=	1	1	0	1	1	1	1	Complemento a dos de M

Nota: hay que observar cómo al complementar a dos el signo no cambia.

Se puede observar en la tabla que para este caso **M** no cambia y **P** es el complemento a dos del multiplicador **P** con signo positivo:

P	=	0	0	0	1	1	1	0	Multiplicador
		0	1	1	0	0	0	1	Complemento a uno de P
							+	1	
P	=	0	1	1	0	0	1	0	Complemento a dos de P

Si se trabaja con n bits se deberá agregar ese mismo número de bits a las tres cantidades en cuestión **M**, **C** y **P**, además del bit de signo a la extrema izquierda y un bit auxiliar (cero) a la extrema derecha como se muestra a continuación.



Los bits adicionales se ponen a la derecha de la cantidad en las variables **M** y **C** y a la izquierda de la cantidad en la variable **P**, como se muestra:

M	=	0	0	1	0	0	0	1	0	0	0	0	0	0
C	=	1	1	0	1	1	1	1	0	0	0	0	0	0
P	=	0	0	0	0	0	0	0	1	1	0	0	1	0

Nota: si se estuviera trabajando con ocho bits se tendrían que agregar ocho bits adicionales en el lugar correspondiente, a cada una de las variable **M**, **C** y **P**, además del bit de signo a la izquierda y del bit auxiliar a la derecha.

El número de pasos de la multiplicación usando el algoritmo de Booth depende del número de bits con los que se trabaje (en este caso son seis bits, por lo tanto el número de pasos a llevar a cabo son solamente seis). Las operaciones a realizar dependerán de los dos últimos bits de la cantidad **P**. Si los bits que están más a la derecha de **P** (el bit auxiliar y el que está junto a él) son:

0	0	No se realiza ninguna operación, solamente se recorren los bits de **P** una posición a la derecha.
0	1	Realizar $\mathbf{P} = \mathbf{P} + \mathbf{M}$ y después hacer el corrimiento a la derecha.
1	0	Realizar $\mathbf{P} = \mathbf{P} + \mathbf{C}$ y posteriormente llevar a cabo el corrimiento a la derecha.
1	1	Solamente llevar a cabo el corrimiento a la derecha de los bits de **P**.

Nota: cuando se lleva a cabo el corrimiento, el bit de la extrema derecha (o auxiliar) se pierde y el bit nuevo que se agrega a la izquierda es igual al bit de signo. Esto significa que se agrega un cero si había un cero en el bit de signo o bien se agrega un uno si había un uno.

a) Para llevar a cabo la multiplicación

$$\begin{array}{rr} & + \quad 17_{(10)} \\ \times & - \quad 14_{(10)} \\ \hline \end{array}$$

usando el algoritmo de Booth cuyas cantidades son **M**, **C** y **P**, los pasos son los siguientes:

- **Paso 1:** como los bits a la extrema derecha de **P** son 0 y 0, solamente se realiza el corrimiento a la derecha de la cantidad **P** para obtener:

$$\mathbf{P} = \quad 0 \quad 0\ 0\ 0\ 0\ 0\ 0 \quad 0\ 1\ 1\ 0\ 0\ \underline{1 \quad 0}$$

- **Paso 2:** como los bits de la extrema derecha de **P** son 1 y 0, realizar $\mathbf{P} = \mathbf{P} + \mathbf{C}$ y posteriormente llevar a cabo el corrimiento:

  Realizando $\mathbf{P} = \mathbf{P} + \mathbf{C}$

$$\begin{array}{rcccccccccccccccc} & \mathbf{P} & = & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ + & \mathbf{C} & = & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline & \mathbf{P} & = & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{array}$$

Haciendo el corrimiento de un bit a la derecha, de esta **P** encontrada se tiene que la nueva **P** es:

$$\begin{array}{lcccccccccccccccc} \text{P} = & & 1 & & 1 & 1 & 0 & 1 & 1 & 1 & & 1 & 0 & 1 & 1 & 0 & \underline{0} & & \underline{1} \end{array}$$

- **Paso 3:** Como los dos bits a la extrema derecha de **P** son 0 y 1, realizar **P** = **P** + **M** y posteriormente llevar a cabo el corrimiento

$$\begin{array}{lcccccccccccccccc} & \text{P} = & & 1 & & 1 & 1 & 0 & 1 & 1 & 1 & & 1 & 0 & 1 & 1 & 0 & 0 & & 1 \\ + & \text{M} = & & 0 & & 0 & 1 & 0 & 0 & 0 & 1 & & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\ & \mathbf{P} = & 1 & 0 & & 0 & 0 & 1 & 0 & 0 & 0 & & 1 & 0 & 1 & 1 & 0 & 0 & & 1 \end{array}$$

Acarreo

El acarreo se desprecia y se hace el corrimiento a la derecha:

$$\begin{array}{lcccccccccccccccc} \mathbf{P} = & & 0 & & 0 & 0 & 0 & 1 & 0 & 0 & & 0 & 1 & 0 & 1 & 1 & \underline{0} & & \underline{0} \end{array}$$

- **Paso 4:** debido a que los dos bits a la extrema derecha de **P** son 0 y 0, solamente se realiza el corrimiento a la derecha de **P**.

$$\begin{array}{lcccccccccccccccc} \mathbf{P} = & & 0 & & 0 & 0 & 0 & 0 & 1 & 0 & & 0 & 0 & 1 & 0 & 1 & \underline{1} & & \underline{0} \end{array}$$

- **Paso 5:** como los bits a la derecha de **P** son 1 y 0 **P** = **P** + **C** y después realizar el corrimiento a la derecha.

$$\begin{array}{lcccccccccccccccc} & \text{P} = & & 0 & & 0 & 0 & 0 & 0 & 1 & 0 & & 0 & 0 & 1 & 0 & 1 & 1 & & 0 \\ + & \text{C} = & & 1 & & 1 & 0 & 1 & 1 & 1 & 1 & & 0 & 0 & 0 & 0 & 0 & 0 & & 0 \\ \hline & \text{P} = & & 1 & & 1 & 1 & 0 & 0 & 0 & 1 & & 0 & 0 & 1 & 0 & 1 & 1 & & 0 \end{array}$$

Realizando el corrimiento se tiene:

$$\begin{array}{lcccccccccccccccc} \text{P} = & 1 & & 1 & 1 & 1 & 0 & 0 & 0 & & 1 & 0 & 0 & 1 & 0 & \underline{1} & & \underline{1} \end{array}$$

- **Paso 6:** ya que los bits a la derecha de **P** son 1 y 1, solamente se hace el corrimiento a la derecha.

$$\begin{array}{lcccccccccccccccc} \text{P} = & 1 & & 1 & 1 & 1 & 1 & 0 & 0 & & 0 & 1 & 0 & 0 & 1 & \underline{0} & & \underline{1} \end{array}$$

Finalmente se elimina el bit auxiliar y el resultado es el producto de multiplicar $(+17_{(10)}) \times (-14_{(10)}) = -238_{(10)}$. Y es posible comprobarlo haciendo el complemento a dos de dicho resultado.

P =	1		1	1	1	1	0	0		0	1	0	0	1	0	
	1		0	0	0	0	1	1		1	0	1	1	0	1	Complemento a uno
														+	1	
P =	1		0	0	0	0	1	1		1	0	1	1	1	0	Complemento a dos

Nota: si el resultado de la multiplicación es negativo, se deberá complementar a dos dicho resultado.

b) Para multiplicar:

$$\begin{array}{rr} & + \ 17_{(10)} \\ \times & + \ 14_{(10)} \\ \hline \end{array}$$

Es el primer caso de la tabla en donde **C** es el complemento a dos de **M** con signo negativo.

M =	1		0	1	0	0	0	1	Multiplicando
	1		1	0	1	1	1	0	Complemento a uno de M
							+	1	
C =	1		1	0	1	1	1	1	Complemento a dos de M

M no cambia y **P** tampoco sufre cambio alguno. De tal manera que con los bits adicionales, el bit de signo y el bit auxiliar se tienen las tres variables **M**, **C** y **P** que se utilizarán en la multiplicación:

M =	0		0	1	0	0	0	1		0	0	0	0	0	0	0
C =	1		1	0	1	1	1	1		0	0	0	0	0	0	0
P =	0		0	0	0	0	0	0		0	0	1	1	1	0	0

- **Paso 1:** Como los bits a la extrema derecha de **P** son 0 y 0, realizar solamente el corrimiento a la derecha de **P**