

Universidad Nacional de la Matanza

Departamento:

Ingeniería e Investigaciones Tecnológicas

Cátedra:

Sistemas de Numeración (3625)

JEFE DE CÁTEDRA:

Mg. Artemisa Trigueros

UNIDAD NRO. 2
SISTEMAS DE NUMERACIÓN

CICLO LECTIVO:

2023

Unidad 2: Sistemas de Numeración

Índice

PARTE A. SISTEMAS DE NUMERACIÓN	4
2.A.1. Introducción	4
2.A.2. Sistemas de Numeración Posicionales y No Posicionales.....	5
2.A.2.1 Características de un sistema posicional.....	5
2.A.2.2. Valor Absoluto y Relativo	8
2.A.2.3. Pasaje de una base a base 10.....	9
2.A.2.4. Base 10 a otra base	11
Parte entera	11
Parte fraccionaria.....	12
CASO 1: “Parte Fraccionaria llega a cero”	12
CASO 2: “Parte Fraccionaria Periódica”	13
CASO 3: “Truncar las cifras fraccionarias”	13
2.A.3. Bases 10 como intermediaria.....	15
2.A.4. Pasaje Directo	15
2.A.4.1. Caso 1: Base Origen mayor que la Base Destino	15
2.A.4.2. Caso 2: Base Origen menor que la Base Destino	16
2.A.4.3. Importancia de aplicar pasaje directo	17
2.A.4. Comparación de números	17
2.A.5. Utilidad del Sistema Binario.....	18
2.A.4. Operaciones Aritméticas.....	21
2.A.4.1. Suma	21
2.A.4.2. Resta	22
2.A.4.3. Multiplicación.....	23
2.A.4.5. División	24
PARTE B. SISTEMAS NUMÉRICOS PARA APLICACIONES INFORMÁTICAS	26
2.B.1. Introducción	26
2.B.2. Formas de representar a los números enteros	26
2.B.2.1. Representación en binario puro. Enteros sin signo.....	27
2.B.2.2. Representación en signo y módulo. SM	28

2.B.3. Complemento de un número.....	30
2.B.3.1. COMPLEMENTO A LA BASE.....	30
2.B.3.2. COMPLEMENTO A LA BASE MENOS UNO	32
2.B.3.4. Representación en complemento a la base-1 (C_{B-1})	34
2.B.3.5. Representación en Complemento a la Base o Complemento a 2.....	37
2.B.5. Operaciones aritméticas con números signados	38
2.B.5.1. Suma en complemento a dos.....	40
2.B.5.2. Resta en complemento a dos.....	41
2.B.6. Overflow (desborde)	42
2.B.7. Operaciones aritméticas dentro de la ALU	43
2.B.8. Comparación de Números enteros signados.	44
2.B.9. Representación binaria de números reales	48
2.B.9.1. Representación en Punto Fijo	48
2.B.9.2. Representación exponencial. Punto Flotante	51
2.B.9.3. Normalización de la mantisa.....	52
2.B.9.4. Bit implícito u oculto	52
2.B.9.5. Representación del exponente.....	53
2.B.9.6. Representación en punto flotante dentro de la computadora	54
2.B.9.7. Rango de la representación IEEE 754. Simple precisión.	56
2.B.9.8. Análisis de Números almacenados en notación de punto flotante.....	57

PARTE A. SISTEMAS DE NUMERACIÓN

- Autora: Dra. Ing. Rocío A. Rodríguez -

2.A.1. Introducción

Según la Real Academia Española¹ un sistema de numeración puede ser definido como:

- “Sistema para expresar de palabra o por escrito todos los números con una cantidad limitada de vocablos y de caracteres o guarismos”.
- “Conjunto de símbolos y reglas utilizados para representar las cantidades”².

En base a la segunda definición se puede plantear a un sistema de numeración como se indica en la expresión 2.A.1.

$$N = (S, R)$$

N: Sistema de Numeración

S: Conjunto de Símbolos válidos dentro de dicho sistema

R: Conjunto de Reglas que permitirán formar números válidos

Expresión 2.A.1. Elementos de un Sistema de Numeración

La expresión 2.A.1. es válida para todo sistema de numeración. Cada sistema de numeración tendrá un conjunto de símbolos válidos y reglas de formación propias.

Existieron diversos sistemas de numeración mediante los cuales los egipcios, griegos, babilónicos, chinos, etc. podían representar las cantidades (ver la figura 2.A.2).

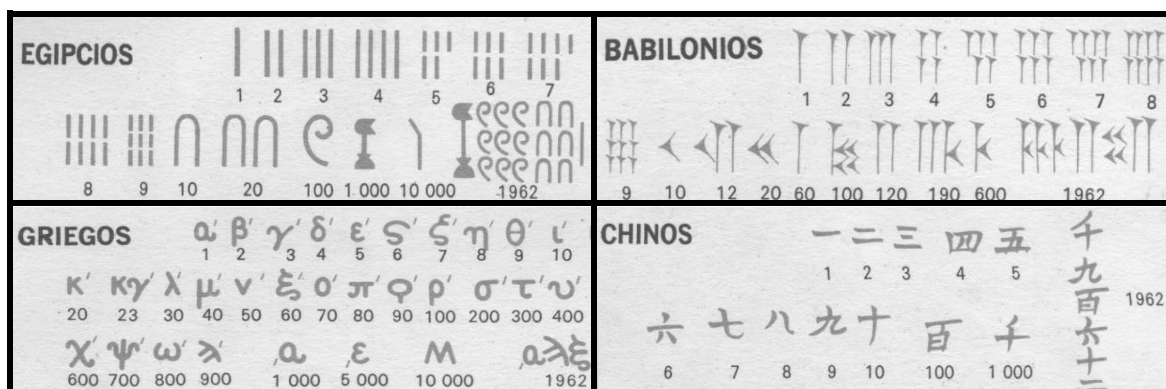


Figura 2.A.2. Representación de números egipcios, griegos, babilónicos y chinos³

Cada sistema de numeración utiliza sus propios símbolos. El sistema decimal es el sistema de numeración adoptado en Argentina sin embargo el sistema Romano está aún presente para la enumeración de diversos objetos, por ejemplo podría haber sido utilizado para numerar los capítulos de este libro. El sistema decimal también conocido como sistema

¹ Definición consultada en: <http://www.rae.es>

² Esta última definición es la adoptada por la mayor parte de la bibliografía

³ Esta imagen fue construida en base a un conjunto de ejemplos tomados de la página 12 del libro [BER74]: Bertha Morris Parker. “La fuente del Saber” Cuarta Edición. Editorial Sygmar S.A., (1974). Buenos Aires, Argentina

Arábigo (“...aunque originario en la India fue introducido en Europa por los árabes” [BER74]) cuenta con los siguientes símbolos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

El sistema romano utiliza otros símbolos: I, V, X, L, C, D, M (donde cada símbolo vale 1, 5, 10, 50, 100, 500 y 1000 respectivamente). Puede notarse que en el Sistema Romano no cuenta con una forma de representar el cero, sin embargo en otros sistemas de numeración como el Maya ya se incorporaba el cero como símbolo.

No sólo son importantes los símbolos sino también las reglas que permiten a través de esos símbolos construir los números. En el sistema de numeración romana: XI representa al número 11 mientras que IX representa al número 9. Cada símbolo tiene un valor de referencia tanto I como X son símbolos válidos del sistema de numeración. Por otra parte ambos números están compuestos por los mismos símbolos sin embargo el resultado final es distinto porque se aplica reglas de formación. En el sistema de numeración romano el símbolo I colocado a la derecha de la X está sumando su valor y en cambio colocado a la izquierda lo está restando.

2.A.2. Sistemas de Numeración Posicionales y No Posicionales

El sistema de numeración romano es no posicional dado que el valor de cada símbolo no depende de la posición en la que se encuentra. El número consignado en la tabla 2.1 vale 8 porque al símbolo V que equivale al 5, se le suma tres veces el valor del símbolo I que equivale al uno 1. No vale más una I que otra dentro de este número.

Tabla 2.A.1. Número Romano

Número Romano	V	I	I	I
Valor del Símbolo	5	1	1	1

En decimal el número 5111 vale cinco mil ciento once, puede observarse que también tiene tres símbolos iguales y que cada 1 no vale lo mismo, sino que su valor se ve condicionado por la posición que ocupa el símbolo dentro del número. A continuación se indica cuánto vale cada componente del número tomando en cuenta no sólo el símbolo sino también su posición (ver tabla 2.A.2).

Tabla 2.A.2. Número Decimal

Número Decimal	5	1	1	1
Valor de cada Símbolo	5000	100	10	1

Todos los sistemas de numeración que se utilizarán a lo largo del presente libro son posicionales y comparten las reglas de formación del sistema decimal.

2.A.2.1 Características de un sistema posicional

Un concepto importante es el de base de un sistema de numeración. La base de un sistema de numeración representa a la cantidad de símbolos admitidos por dicho sistema. El sistema decimal ó de base 10, posee 10 símbolos distintos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9; es importante notar que el 10 no es un símbolo del sistema sino que se ha generado por medio de combinar dos símbolos ya existentes.

Además todos los sistemas de numeración posicionales a los que se hace referencia tienen como símbolo inicial: 0. De forma que el sistema en base 2 sólo tendrá dos símbolos: 0, 1. Si se desea armar un sistema en base 3 bastará con agregar un símbolo al sistema anterior: 0, 1, 2. De esta manera es posible armar diversos sistemas cuya base sea menor que 10, utilizando parte de los símbolos de base 10. ¿Qué sucede si se quieren confeccionar sistemas de base mayor a 10?, en ese caso será necesario utilizar nuevos símbolos, el sistema Hexadecimal de base 16 utiliza letras para completar los símbolos faltantes, de este modo podrá utilizarse todos los símbolos del sistema Decimal (del 0 al 9) aquí hay 10 símbolos distintos y los 6 restantes utilizando las letras del alfabeto (de la A a la F). La tabla 2.A.3 muestra los símbolos que conforman distintos sistemas de numeración.

Tabla 2.A.3. Símbolos que conforman los sistemas de numeración

Sistemas	SIMBOLOS															
Base 2 (Binario)	0	1														
Base 3	0	1	2													
Base 4	0	1	2	3												
Base 8 (Octal)	0	1	2	3	4	5	6	7								
Base 9	0	1	2	3	4	5	6	7	8							
Base 10 (Decimal)	0	1	2	3	4	5	6	7	8	9						
Base 16 (Hexadecimal)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

A partir de ahora ante un número 435 debería ser importante preguntarse cuál es su base, sin conocer su base no será posible interpretar de que número se trata. Al ver este número sin su base es posible saber que el mismo no puede estar escrito por ejemplo en binario porque dicho sistema de numeración sólo tiene por símbolos válidos el 0 y 1; si el número tiene símbolos que no pertenecen a un sistema de numeración determinado dicho número no puede estar escrito en ese sistema de numeración. Por ende se puede afirmar que el 435 tiene que estar escrito en un sistema de numeración de base 6 ó cualquiera de base mayor a 6. Motivo por el cual los números estarán acompañados de su base la cual se indica como un subíndice detrás del mismo (se recomienda resolver el ejercicio 2.A.1).

Ejercicio 2.A.1 - Sugerido

Indique cuales de los siguientes números son inválidos (analizando los símbolos utilizados):

- a) $7A_{10}$ b) 523_9 c) 231_3 d) $A95_{16}$ e) 872_7 f) 462_5

Generar números en un determinado sistema de numeración posicional, es muy simple. En base 10, el primer número posible de escribir es el 0 con un solo dígito el último número posible de escribir será 9, una vez que se acabaron los símbolos estos pueden ser combinados con otros generándose el 10, 11 hasta el 99 en el cual se ha utilizado en ambas posiciones el mayor número del sistema de numeración esto implica que debe agregarse otro dígito de aquí en más los números van a estar constituidos con tres dígitos 100, 111 hasta el 999 en el cual en las tres posiciones existentes quedó el mayor símbolo de la base. Lo mismo puede hacerse en base 3, los primeros números a escribir serán el 0, 1 y 2; se acabaron los símbolos con lo cual deberá comenzarse a escribir números utilizando dos dígitos desde el 10 al 22; luego se empezará a escribir de a tres dígitos... De ésta forma se genera la tabla 2.A.4.

Tabla 2.A.4. Construcción de números en sistemas de numeración posicionales

Decimal Base 10	Binario Base 2	Octal Base 8	Hexadecimal Base 16
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15

El armar una tabla, en la cual cada columna muestra los números escritos en distintos sistemas de numeración, permite establecer equivalencias leyendo cualquier fila de la tabla 2.A.4, por ejemplo la última: $21_{10} = 10101_2 = 25_8 = 15_{16}$

Ejercicio 2.A.2 - Sugerido

Continúe la construcción de la tabla 2.4, realice tres filas más para mostrar que números representan a los números decimales 22, 23, 24

Al observar la tabla 2.A.4 es importante notar que:

- Diez escrito en base Diez es 10
- Dos escrito en base Dos es 10
- Ocho escrito en base Ocho es 10
- Dieciséis escrito en base Dieciséis es 10

Siempre la base de un sistema de numeración posicional expresada en dicho sistema será 10 (“uno cero”). No se deberá leer como Diez ya que ese 10 valdrá distinto según la base en la que se haya escrito.

Si bien es simple construir la tabla 2.A.4 es importante comenzar a suponer que debe existir otro mecanismo para poder conocer la equivalencia de un número 9142_{10} expresado en base 8. Para construir una tabla comenzando por el 0 decimal deberán realizarse luego 9142 renglones para determinar su equivalente en base 8. Esta tabla sirve

tan sólo a modo de ejemplo siempre que los números a representar no sean grandes. Por otra parte si sólo se aplicara el mecanismo de construir una tabla para poder realizar una conversión entre bases no sería posible conocer cuántos renglones serían necesarios construir si el número de origen está en otra base que no sea decimal, por ejemplo: 2103_4 .

2.A.2.2. Valor Absoluto y Relativo

En los sistemas de numeración posicionales cada símbolo tiene un valor absoluto que es el valor que tiene asignado el símbolo por ejemplo en decimal el símbolo 9 (vale nueve) pero también existe un valor relativo que hace que dentro de este número **59** ese símbolo valga distinto que dentro de este otro **93**. En el primer caso vale nueve y en el segundo caso vale noventa. El valor relativo es el valor que adquiere el símbolo por estar en una determinada posición dentro del número. Para profundizar esto se propone recordar algunas cuestiones del sistema decimal, la tabla 2.A.5 se construyó para ilustrar cuánto vale un símbolo escrito según en qué posición se encuentre.

Tabla 2.A.5. Sistema de Numeración Decimal - Posiciones⁴

Unidades de Mil	Centena	Decena	Unidad	Decima	Centésima	Milésima
1000	100	10	1	1/10	1/100	1/1000
10^3	10^2	10^1	10^0	10^{-1}	10^{-2}	10^{-3}

Es posible afirmar que si bien 519 y 915 a pesar de estar formados por los mismos signos y ambos escritos en base 10, no valen lo mismo. A partir de la tabla 1.3 se procede a descomponer uno de los números tal como se muestra en la expresión 2.A.2.

$$519_{10} = 5 \text{ centenas} + 1 \text{ decena} + 9 \text{ unidades} = 5 \times 100 + 1 \times 10 + 9 \times 1 = 5 \times 10^2 + 1 \times 10^1 + 9 \times 10^0$$

Expresión 2.A.2. Descomposición de un número decimal

A partir de lo realizado en la expresión 2.A.2. puede observarse que se partió de un número de tres cifras y el resultado se puede expresar por medio de tres términos, cada uno representa una componente del número en la cual está cada símbolo del número acompañado de la base del sistema de numeración elevada a un exponente que expresa la posición del símbolo dentro del número.

Por lo cual se puede afirmar que: Cada término ha quedado compuesto por el valor absoluto del símbolo y un valor relativo (base elevada a un exponente).

En forma general independientemente de los símbolos que conforman al número ó de la base del sistema de numeración:

$$\sum_{i=-n}^M a_i \times B^i$$

Expresión 2.A.3. Teorema fundamental de la numeración

Siempre se procede a hacer una sumatoria en donde:

- **a** representa a un símbolo dentro del número
- **B** la base del sistema de numeración
- **i** la posición del símbolo dentro del número

⁴ Esta tabla muestra de forma práctica como expresar en potencias de 10 las distintas posiciones, ha sido tomada de <http://es.wikipedia.org/wiki/Bit>

Es importante notar que cada término de la sumatoria está construido por el producto de dos componentes a_i representa el símbolo (valor absoluto) y B^i es la base del sistema de numeración elevada a la posición que tiene el símbolo dentro del número (valor relativo). El teorema fundamental de la numeración permitirá descomponer un número tal como se muestra en la figura 2.A.2

Se quiere aplicar el teorema fundamental de la numeración para el número $519,6_{10}$

5	1	9	,	6
a_2	a_1	a_0		a_{-1}

$$\sum_{i=-1}^2 a_i \times B^i = a^2 \times B^2 + a^1 \times B^1 + a^0 \times B^0 + a^{-1} \times B^{-1} = 5 \times 10^2 + 1 \times 10^1 + 9 \times 10^0 + 6 \times 10^{-1}$$

Figura 2.A.2. –Descomposición de un número

2.A.2.3. Pasaje de una base a base 10

El teorema fundamental de la numeración puede ser aplicado para realizar la conversión de una base a base 10. En la figura 2.A.3 se plantea el caso de un número en base 4 el cual quiere expresarse en base 10. Para realizar esta conversión se aplica el teorema fundamental de la numeración. Nótese que no se está descomponiendo al número ya que se están expresando los valores en base 10, por ejemplo 4 (valor de la base) está expresado en decimal ya que el símbolo 4 no pertenece a base 4. La suma de todas las componentes del número escritas en decimal dará origen a un resultado en decimal.

Se quiere aplicar el teorema fundamental de la numeración para el número $2103,1_4$

a_3	a_2	a_1	a_0		a_{-1}
2	1	0	3	,	1

$$\sum_{i=-n}^M a_i \times B^i = \sum_{i=-1}^3 a_i \times 4^i$$

$$\sum_{i=-1}^3 a_i \times 4^i = 2 \times 4^3 + 1 \times 4^2 + 0 \times 4^1 + 3 \times 4^0 + 1 \times 4^{-1} = 128 + 16 + 0 + 3 + 0,25 = 147,25_{10}$$

(para el caso particular de este ejemplo)

Figura 2.A.3. – Pasaje de base 4 a base 10

De este modo se puede realizar el pasaje desde un sistema posicional a decimal sin inconveniente alguno. A continuación en la tabla 2.A.6 se presenta la conversión del número 10 escrito en diversas bases a decimal.

Tabla 2.A.6. Conversión del número 10 expresado en distintas bases a decimal

Valor de Origen	Cálculo	Resultado
10_2	$1 \times 2^1 + 0 \times 2^0$	2
10_3	$1 \times 3^1 + 0 \times 3^0$	3
10_4	$1 \times 4^1 + 0 \times 4^0$	4
...
10_8	$1 \times 8^1 + 0 \times 8^0$	8
10_9	$1 \times 9^1 + 0 \times 9^0$	9
...
10_{16}	$1 \times 16^1 + 0 \times 16^0$	16

A partir de la tabla 2.A.6 es posible decir que 10 escrito en una determinada base dará por resultado el valor de su base expresado en decimal. Del mismo modo podrá decirse que la base escrita en su base será 10. En forma general: $10_B = 0 \times B^0 + 1 \times B^1 = B$.

Siempre la base de un sistema de numeración posicional expresado en dicho sistema será 10 (“uno cero”). Esto puede ser observado también en la tabla 2.4. En dicha tabla se ha sombreado la base escrita en su base a lo largo de todas las columnas.

Para expresar un número hexadecimal (base 16) a decimal (base 10), se procede del mismo modo. Como puede observarse en el ejercicio 2.3 la base escrita en decimal será 16, cada uno de los símbolos numéricos del 0 al 9 en hexadecimal coinciden con decimal y a las letras A, B, C, D, E, F se escribirá su equivalencia en decimal (mostrada en la tabla 2.4). Se propone observar lo realizado en el ejercicio 2.A.3

Ejercicio 2.A.3 - Resuelto

Se quiere aplicar el teorema fundamental de la numeración para el número $A5C,B1_{16}$

Resolución:

$$\begin{array}{cccccc}
 & a_2 & a_1 & a_0 & & a_{-1} & a_{-2} \\
 & A & 5 & C & , & B & 1 \\
 \\
 \sum_{i=-2}^2 a_i \times 16^i = & A \times 16^2 & + 5 \times 16^1 & + C \times 16^0 & + B \times 16^{-1} & + 1 \times 16^{-2} = \\
 & 10 \times 16^2 & + 5 \times 16 & + 12 \times 1 & + 11 \times 1/16 & + 1/16^2 = \\
 & 2560 & + 80 & + 12 & + 0,6875 & + 0,00390625 = \\
 & \text{Resultado} = 2652,69140625_{10}
 \end{array}$$

A continuación se en el ejercicio 2.A.4 se proponen algunos pasajes los cuales requieren simples cálculos para ser efectuados.

Ejercicio 2.A.4 - Sugerido

Indicar el resultado en base 10 de los siguientes números:

- a) 162_8 b) 31_4 c) 100_2 d) 101101_2

2.A.2.4. Base 10 a otra base

El método que se empleará para realizar el pasaje de decimal (base 10) a otra base consiste en⁵:

1. Tomar la parte entera y dividir sucesivamente por el valor de la base destino
2. Tomar la parte fraccionaria y multiplicar sucesivamente por la base destino

Parte entera

Primeramente se presenta a modo de ejemplo como pasar de decimal a binario, para ello se toma el número 535_{10} y se muestra el procedimiento aplicado en la figura 2.4. Se procede a realizar divisiones que den por resultado un cociente entero se toma el número origen 535 se lo divide por la base destino 2 y se obtiene por cociente 266 y el resto arrojado es 0. Al cociente obtenido se lo vuelve a dividir por la base destino y así sucesivamente. Cabe destacar que a medida que se van aplicando divisiones el cociente que será obtenido en cada una de ellas será menor que el obtenido anteriormente (esto sucederá con todas las bases destinos). En el momento en que el cociente obtenido es inferior a la base destino originará un próximo cociente en cero de modo que se habrá finalizado el procedimiento. Se puede notar en la figura 2.4 que al dividir $2/2$ esto da cociente 1 y resto 0. El cociente 1 se somete nuevamente a división $1/2$ pero como se busca un cociente entero dará 0 y de resto 1 (destacado en la figura 2.5), a partir de allí todas las divisiones próximas serán $0/2$ dando cociente 0 y resto 0, allí no tiene sentido alguno continuar, dándose por finalizado el procedimiento (parte sombreada de la figura 2.A.4).

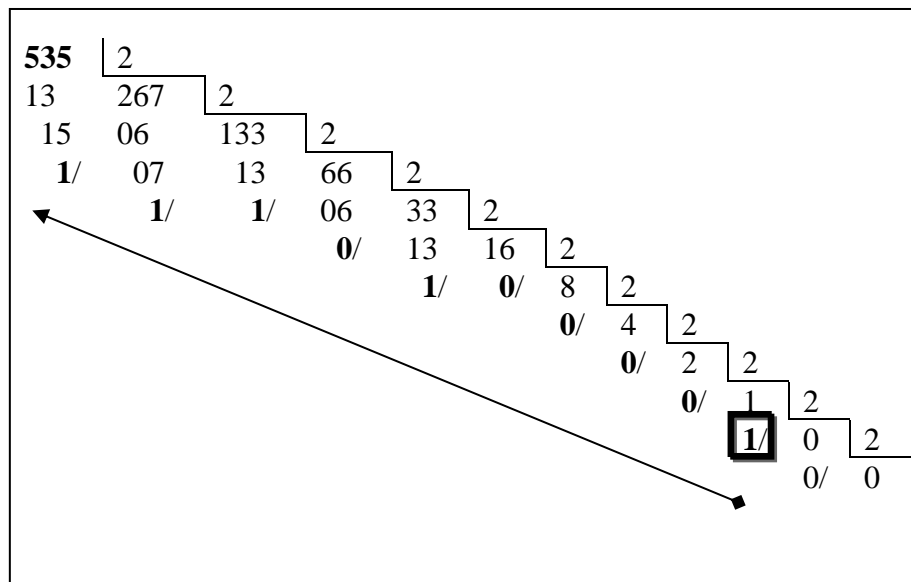


Figura 2.A.4. – Pasaje de base 10 a base 2

⁵ El fundamento de aplicar el método que se describe de forma práctica tiene basamento en el teorema fundamental de la numeración presentado previamente. La demostración formal de la aplicación de dicho teorema que da por origen la aplicación de este método práctico se encuentra en diversos libros entre ellos [MAN98]

El resultado de pasar 535_{10} a base 2 se consigna tomando todos los restos obtenidos en forma inversa (comenzando por el **1** destacado en la figura 2.4 último resto significativo obtenido): 1000010111 cabe destacar que si se hubiesen considerado los ceros arrojados como otros restos para conformar el resultado este hubiese sido ~~00~~1000010111 (los ceros delante de la cifra entera no aportan valor siendo el mismo número que el obtenido sin haberlos considerado).

Ejercicio 2.A.5 - Sugerido

Verifique que sea correcto lo realizado en el procedimiento anterior.

Se propone tomar el resultado obtenido en binario 1000010111, pasarlo a decimal y corroborar que de por resultado el número de partida 535_{10}

Parte fraccionaria

Todo número podrá analizarse descomponiéndose en su parte entera y en su parte fraccionaria, por ejemplo: $535,25$ será: $535 + 0,25$. Este número en base 10 para ser expresado en otra base será necesario realizar primeramente la conversión de la parte entera del mismo (en la figura 2.A.4 se muestra la conversión a base 2) y luego se añadirá a dicho resultado la conversión resultante con la parte fraccionaria (lo cual se explicará a continuación).

CASO 1: “Parte Fraccionaria llega a cero”

$535,25_{10}$: Se calculará 0,25 en base 2, para lo cual se multiplica sucesivamente por 2 (como se muestra en la figura 2.A.6). Cada cuenta efectuada estará compuesta por una parte entera y una parte fraccionaria, la parte entera será la que conformará cada uno de los dígitos del resultado a obtener. Puede observarse que $0,25 \times 2 = 0,50$ la parte entera es 0 y sirve para conformar el resultado, se continua con la parte fraccionaria el proceso (restandose previamente la parte entera) siempre cada multiplicación tendrá por primer factor 0,ParteFraccionaria del resultado anterior. Es importante notar que la segunda cuenta ha dado por resultado 1,00 para la multiplicación siguiente se tomará 0, 00 con lo cual todos los valores siguientes a tomar serán 0 (los ceros al final de la cifra, en la parte fraccionaria carecen de valor, esto debiera hacer notar que se acabó el procedimiento).

Se calculará 0,25 en base 2, para lo cual tal como se muestra en la figura 2.A.6 se multiplica sucesivamente por 2.

0,25	0,50	0,00	0,00	...
<u>x 2</u>	<u>x 2</u>	<u>x2</u>	<u>x2</u>	...
<u>0,50</u>	<u>1,00</u>	<u>0,00</u>	<u>0,00</u>	...

Figura 2.A.6. Pasaje de base 10 a base 2

Por lo tanto:

$0,25_{10}$ será $0,01000_2$ (Los ceros detrás de la parte fraccionaria no son significativos)

$535,25_{10}$ será $1000010111,01_2$ (se le añade la parte entera calculada anteriormente)

CASO 2: “Parte Fraccionaria Periódica”

525,3₁₀: Se calculará 0,3 en base 2, para lo cual tal como se muestra en la figura 2.A.7 el procedimiento realizado.

0,3	0,6	0,2	0,4	0,8	0,6	...
x 2	x 2	x2	x2	x 2	x2	...
<u>0,6</u>	<u>1,2</u>	<u>0,4</u>	<u>0,8</u>	<u>1,6</u>	<u>1,2</u>	...

Figura 2.A.7. – Pasaje de base 10 a base 2

En la figura 2.A.7 puede observarse que la segunda cuenta efectuada se repite a lo largo del procedimiento. Esto causará que se repitan también las siguientes cuentas a efectuar, por lo cual no es necesario continuar el procedimiento. Debe advertirse que el resultado es periódico: **01001**

Por lo tanto:

0,3₁₀ será 0,0**1001**₂

535,3₁₀ = 1000010111,0**1001**₂

CASO 3: “Truncar las cifras fraccionarias”

En realidad, este caso podría haberse aplicado a cualquier conversión, incluso a las planteadas en los casos anteriores. Establecida la precisión con la que se desea trabajar es posible indicar cuantas cifras fraccionarias se desean obtener por ejemplo 3 cifras, el resto de ellas serán “ignoradas” (aunque más adelante el procedimiento pueda concluir en un número periódico o que llegue a cero), a este proceso se lo denomina truncamiento. Se plantea a continuación un ejemplo en el que se toman 6 cifras fraccionarias.

535,28₁₀: Se calculará 0,28 en base 2, para lo cual se multiplica sucesivamente por 2 (como se muestra en la figura 2.A.5). Cada cuenta efectuada estará compuesta por una parte entera y una parte fraccionaria, la parte entera será la que conformará cada uno de los dígitos del resultado a obtener. Puede observarse que $0,28 \times 2 = 0,56$ la parte entera es 0 y sirve para conformar el resultado, se continua con la parte fraccionaria el proceso (restandose previamente la parte entera) siempre cada multiplicación tendrá por primer factor 0,ParteFraccionaria del resultado anterior. Es importante notar que la segunda cuenta ha dado por resultado 1,12 para la multiplicación siguiente se tomará 0, 12

0,28	0,56	0,12	0,24	0,48	0,96	...
x 2	x 2	x2	x2	x 2	x2	...
<u>0,56</u>	<u>1,12</u>	<u>0,24</u>	<u>0,48</u>	<u>0,96</u>	<u>1,92</u>	...

Figura 2.A.5. – Pasaje de base 10 a base 2

Cada multiplicación arroja un resultado del cual se tomará en cuenta la parte entera del mismo (dígito subrayado en la figura 2.A.5). Se han efectuado 6 multiplicaciones podría

haberse continuado, el interés de continuar dependerá de cuantos dígitos fraccionarios requiera el resultado.

Por lo tanto:

$0,28_{10}$ será $0,010001_2$

$535,28_{10}$ será $1000010111,010001_2$ (se le añade la parte entera calculada anteriormente)

A partir de este ejemplo se propone analizar el ejercicio 2.A.6 el cual está resuelto a continuación.

Ejercicio 2.A.6 - Resuelto

Verifique el resultado obtenido en la parte fraccionaria (0,28)

Cuáles son las conclusiones que puede alcanzar a partir de dicho resultado

Resolución:

$0,28_{10} = 0,010001_2$

0,	0	1	0	0	0	1
	1/2	1/4	1/8	1/16	1/32	1/64

$$1/4 + 1/64 = 0,25 + 0,015625 = \mathbf{0,265625}$$

El resultado obtenido no fue 0,28 esto es causado por el truncamiento efectuado, el no haber considerado todas las cifras en la figura 2.5 (multiplicaciones siguientes no efectuadas), a continuación se muestra sombreadas las cifras fraccionarias calculadas previamente, calculándose tres cifras más. Esto permitirá comprobar que el considerar el peso de dichas cifras harán que el número final será mayor más próximo al valor esperado.

0,28	0,56	0,12	0,24	0,48	0,96	0,92	0,84	0,68
x 2	x2	x2	x2	x2	x2	x 2	x 2	x 2
<u>0,56</u>	<u>1,12</u>	<u>0,24</u>	<u>0,48</u>	<u>0,96</u>	<u>1,92</u>	<u>1,84</u>	<u>1,68</u>	<u>1,36</u>

0,	0	1	0	0	0	1	1	1	1
	1/2	1/4	1/8	1/16	1/32	1/64	1/128	1/256	1/512

Al valor obtenido previamente **0,265625** calculado previamente con las cifras sombreadas se le deberá añadir el proveniente de las tres cifras contempladas adicionalmente.

$$= \mathbf{0,265625} + 0,0078125 + 0,00390625 + 0,001953125 =$$

$$= \mathbf{0,265625} + 0,013671875 = \mathbf{0,279296875}$$

Si se mostraran en el cálculo final tres cifras fraccionarias considerando únicamente las multiplicaciones sombreadas sería: 0,265 en cambio realizando 3 multiplicaciones más sería: 0,279 notablemente más próximo a 0,28

Ejercicio 2.A.7 – SugeridoExpresar el número $163,6875_{10}$ en Base 16**2.A.3. Bases 10 como intermediaria**

Aplicando lo visto anteriormente será posible expresar un número que esté proveniente en una base origen a una base destino (cualquiera sean estas bases). Por ejemplo: Se cuenta con un número en base 5 y quiere pasarse a base 8, para ello se aplicarán los dos métodos vistos: (1) Base origen: 5 \rightarrow Base destino: 10; (2) Base origen: 10 \rightarrow Base destino: 8

Es decir cuando ni la base origen ni la destino es decimal deberá utilizarse la base 10 como intermediaria realizándose dos pasajes.

2.A.4. Pasaje Directo

El pasaje directo permite en forma rápida poder expresar un número que se encuentra en una determinada base a otra. Es aplicable cuando la base origen y destino se relacionan por medio de una potencia entera y positiva. En la tabla 2.A.7 se muestran algunas bases entre las que es posible aplicar pasaje directo.

Tabla 2.A.7. Bases entre las que es posible aplicar pasaje directo

Base Origen	Base Destino	Relación
2	4	$2^2 = 4$
3	9	$3^2 = 9$
4	16	$4^2 = 16$

Es importante notar que si bien entre la base 8 y 16 no hay pasaje directo (ya que no existe potencia entera positiva la cual permita elevar al número 8 y obtener por resultado 16) sería posible realizar pasaje directo de base 8 a base 2 y luego otra vez pasaje directo de base 2 a 16.

Se recomienda realizar el ejercicio 2.A.9 en base a la definición de pasaje directo.

Ejercicio 2.A.9 – Sugerido

Indique en qué casos puede aplicarse pasaje directo

- a) Base 8 a base 5 b) Base 2 a Base 16 c) Base 2 a Base 8

2.A.4.1. Caso 1: Base Origen mayor que la Base Destino

Si se desea expresar al número $31,203_4$ en base 2, primeramente se analizará si es posible realizar pasaje directo, para lo cual se toma la base menor 4, luego surge la pregunta ¿A qué valor debe elevarse dicha base para alcanzar a la otra base? $2^2 = 4$ esto deberá leerse:

por cada 2 símbolos en base 2 deberá escribirse 1 en base 4. En la tabla 2.A.8 se han anotado todos los símbolos que componen a la base 4 y luego sus equivalentes en base 2, es posible notar que con dos símbolos en base 2 se han podido expresar todos los símbolos válidos de base 4. Es importante destacar que todos los números en la segunda columna deben indicarse con dos dígitos por ello se ha antepuesto un cero en las dos primeras filas.

Tabla 2.A.8. Tabla de símbolos en Base 8 y su equivalencia en Base 4

Base 4	Base 2
0	00
1	01
2	10
3	11

El proceso consistirá en sustituir cada uno de los símbolos en base 8 provenientes del número original por dos en base 4 (localizándolos en la tabla 2.A. 8), tal como se muestra en la figura 2.A.8.

3	1	,	2	0	3
<u>11</u>	<u>01</u>	,	<u>10</u>	<u>00</u>	<u>11</u>

Figura 2.A.8. Pasaje Directo entre base 4 a base 2

2.A.4.2. Caso 2: Base Origen menor que la Base Destino

Si se desea expresar el número $110,1_2$ a base 4 también será posible aplicar pasaje directo siendo la relación entre las bases: $2^2 = 4$. Cabe aclarar que en este caso se cuenta con cada uno de los símbolos del número en base 2 y por cada 2 de ellos deberán escribirse un símbolo en base 4.

En este caso se quiere agrupar de a dos un número con tres cifras en la parte entera, de forma que será necesario agregar un cero (que no altere el valor del número) para que puedan conformarse dos grupos en los cuales haya dos símbolos en cada uno de ellos. Lo mismo sucede con la parte fraccionaria como hay un sólo dígito será necesario agregar un cero para poder conformar un grupo de dos dígitos. En la figura 2.A.9 se muestran dos ceros agregados, si se observa el número resultante este será $1100,01$ el cual difiere del número original $110,1$ ha cambiado el valor con lo cual de aplicar el pasaje se estaría expresando otro número en base 4 y no el deseado.

1	1	0	0	,	0	1	1100,01
---	---	---	--------------	---	--------------	---	---------

Figura 2.A.9. Agrupamiento inválido

Para no alterar el valor al agregar ceros en un número será necesario que en la parte entera se inserten delante (a la izquierda) y en la parte fraccionaria los ceros se agreguen detrás (a la derecha), esto se observa fácilmente en decimal $5,2$ es lo mismo que escribir $05,20$ estos ceros agregados no aportan valor en el número escrito.

A continuación en la figura 2.A.10 se muestra el agrupamiento correctamente realizado, el número que ha quedado luego de agregar ceros es equivalente al de partida. Si se quiere podría comenzarse a agrupar tomando en cuenta el sentido indicado por medio de flechas en la figura 2.A.10 y luego agregar los ceros necesarios para conformar el último grupo. Luego de agrupar correctamente tan sólo queda escribir cada grupo a que dígito se corresponde en base 4, lo cual puede realizarse observando la tabla 2.A.8.

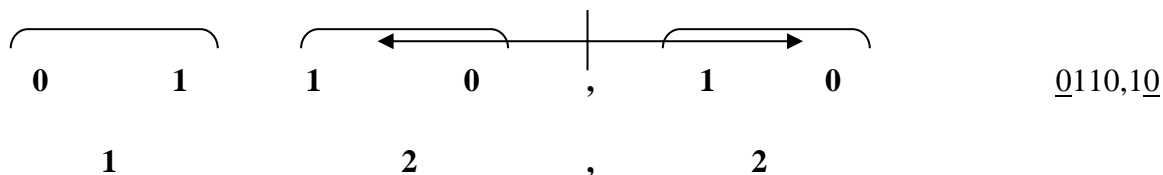


Figura 2.A.10. Pasaje Directo entre base 2 a base 4

Se propone realizar el ejercicio 2.A.10 para afianzar el método de pasaje directo.

Ejercicio 2.A.10 - Sugerido

Se proponen aplicar pasaje directo para:

- 1) $10110111,101_2$ a BASE 16
- 2) 123_4 a BASE 16

2.A.4.3. Importancia de aplicar pasaje directo

Es importante aplicar pasaje directo, siempre que sea posible, debido a que el tiempo que insume aplicar pasaje directo es inferior al invertido utilizando la base 10 como intermediaria. Cuando se utiliza a la base 10 como intermediaria es posible que sea necesario realizar una gran cantidad de cálculos matemáticos y en caso de no considerar todas las cifras fraccionarias en los cálculos intermedios el resultado final será aproximado. En cambio esto último no ocurre al aplicar pasaje directo.

Por un momento es importante pensar los distintos pasos a realizar sin aplicar pasaje directo para expresar el número en hexadecimal $A7CB8,3CD_{16}$ a base 4 y los cálculos matemáticos que serían necesarios realizar. Para comparar con lo planificado utilizando como intermediaria a la base 10, se propone resolver aplicando pasaje directo dicho ejercicio planteado a continuación.

Ejercicio 2.A.11 - Sugerido

Expresar el número $A7CB8,3CD_{16}$ en base 4

2.A.4. Comparación de números

Con lo visto previamente es posible realizar comparación de números para determinar uno de tres resultados posibles: Mayor, Menor ó Igual. Claramente la respuesta será una sola de estas tres posibilidades. Siendo sistemas de numeración posicionales al ver un número entero sin signo (como lo son los números de la PARTE A de esta Unidad), se realizará una comparación en magnitud.

Se tomará primeramente en consideración a modo de ejemplo los números en decimal: $A = 345_{10}$; $B = 534_{10}$

Los números A y B están compuestos por los mismos símbolos sin embargo $A < B$ esto se da porque para números de igual cantidad de dígitos se compara dígito a dígito comenzando por el dígito más significativo. En este caso la decisión se toma analizando la columna de las centenas, si el dígito en la columna de las centenas no es el mismo en ambos números entonces ya se podrá tomar la decisión. Para comparar números de distintas longitudes se pueden imaginar con ceros a la izquierda de la parte entera lo que provocará que siempre al comparar sea mayor el número que tiene mayor longitud. Lo mismo sucederá en otras bases, por ejemplo, en base 8: $A = 345_8$; $B = 534_8$ se seguirá manteniendo la misma relación $A < B$.

Finalmente se propone pensar para números de longitud de 4 bits: ¿cuántas comparaciones podrían necesitarse hacer entre bits decidir el resultado al compararlos? Entonces se puede plantear los números en base 2 (binario) $A = 1101_2$; $B = 1100_2$ recién en el bit menos significativo se deduce que $A > B$ esto requiere haber comparado los tres dígitos anteriores previamente es decir que se realizaron 4 comparaciones para llegar al resultado.

Por supuesto no se pueden comparar números que estén en distintas bases por ejemplo: $A = 101101_2$; $B = 233_4$ en este caso deberán expresarse ambos números en la misma base para luego realizar la comparativa, se decide por ejemplo pasar el número de base 2 a base 4 resultando: $10\ 11\ 01 = 231_4$ entonces se concluye que $A = 231_4$; $B = 233_4$ entonces $A < B$

2.A.5. Utilidad del Sistema Binario

De los sistemas de numeración presentados en este capítulo es el binario aquel que cobrará fundamental importancia para los siguientes capítulos. El sistema binario es el utilizado para almacenar información en una computadora.

En electrónica digital se utilizan dispositivos y circuitos en los que sólo existen dos estados posibles de información. Por este motivo, cualquier dato deberá ser representado como una secuencia de dos valores, que pueden considerarse como sí – no, abierto – cerrado, encendido – apagado (on – off) o cualquier otra pareja de símbolos. Por motivos prácticos se adoptan los símbolos 0 y 1, cuya sucesión proporciona números binarios más fáciles de tratar que las otras secuencias mencionadas, mediante los cuales es posible realizar operaciones aritméticas y lógicas. Un número binario, por ejemplo el 10101, está compuesto por una determinada cantidad de dígitos binarios denominados Bit (**B**inary **d**igit), en este caso el número está constituido por 5 bits.

Sin embargo, en informática no siempre se habla de bits. Sucede que en algunas ocasiones la cantidad de bits es tan grande que es conveniente expresar dicha cantidad de otro modo. Una persona que va a comprar un pendrive (medio de almacenamiento extraíble) seguramente no se le ocurriría solicitarlo pidiendo que tenga una capacidad de 64.000.000.000 bits, probablemente será más común escuchar que la capacidad de dicho dispositivo es de 64 GB. Lo mismo ocurre cuando se solicita un Kilo de pan y no 1.000

gramos. Si bien en matemática el Kilo es equivalente a 1.000, en informática resulta en algunas ocasiones conveniente expresar todo por medio de potencias de dos siendo $2^{10} = 1024$ el valor más próximo a 1.000. Para poder diferenciar un Kilo considerando 1.000 al Kilo considerando 1.024 la ISO (Organización Internacional de Estandarización) en una de sus normativas que data del 1999 (IEC 60027-2) ha dispuesto expresar KB en caso de tratarse de 1000 y KiB en caso de tratarse de 1024, esa i en el medio hará la diferencia, adoptándolo también el IEEE posteriormente (1542-2002).

En la tabla 2.A.13 se presentan las equivalencias y denominaciones. La unidad más pequeña es el bit (un bit podrá ser un 0 ó un 1), un conjunto de 4 bits recibe el nombre de Nigle. Un conjunto de 8 bits se denominan Byte, de forma que $\frac{1}{2}$ Byte = 1 Nigle. A partir del Byte se conforman el resto de las unidades 1000 Bytes = 1 KByte y a partir de allí cada 1000 (10^3) existe otra unidad. Para facilitar la comprensión de las unidades, se utiliza una aproximación entre las potencias de 2 y las de 10, ya que $2^{10} = 1024$ es aproximadamente $1.000 = 10^3$ (ver tabla 2.A.14).

Tabla 2.A.13. Nombres de las unidades en función de la cantidad de bits y bytes

Unidad	Abreviatura	Equivalencia entre unidades	Cantidad de Bytes en Potencias de 10
Bit		1 bit	
Nibble		4 bits	
Byte	B	8 bits	
Kilo	K	1000 Bytes	10^3
Mega	M	1000 KBytes	10^6
Giga	G	1000 MBytes	10^9
Tera	T	1000 GBytes	10^{12}
Peta	P	1000 TBytes	10^{15}
Exa	E	1000 PByte	10^{18}
Zetta	Z	1000 EByte	10^{21}
Yotta	Y	1000 ZByte	10^{24}

Tabla 2.A.14. Nombres de las unidades en función de la cantidad de bits y bytes

Unidad	Abreviatura	Equivalencia entre unidades	Cantidad Bytes en Potencias de 2
Bit		1 bit	
Nibble		4 bits	
Byte	B	8 bits	
Kilo	K	1024 Bytes	2^{10}
Mega	M	1024 KiBytes	2^{20}
Giga	G	1024 MiBytes	2^{30}
Tera	T	1024 GiBytes	2^{40}
Peta	P	1024 TiBytes	2^{50}
Exa	E	1024 PiByte	2^{60}
Zetta	Z	1024 EiByte	2^{70}
Yotta	Y	1024 ZiByte	2^{80}

A modo de ejemplo se muestra una captura de pantalla de resultados en donde puede observarse un archivo cuyo tamaño viene dado por 148 KiB.

```

Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\codigoc\camposBit.exe
- Output Size: 148,66796875 KiB
- Compilation Time: 0,69s

```

Figura 2.A.11. Captura del resultado de compilación en DevC++

Se recomienda observar el ejercicio 2.A.12 propuesto a continuación utilizando la tabla 2.A.13

Ejercicio 2.A.12 - Resuelto

¿Cuántos bits representan 16 ZiB?

Expresa el resultado por medio de una potencia de 2

Resolución:

$$\begin{array}{ccccccccccccccc}
 \textcircled{16} \text{ ZiB} = & \textcircled{16} & \times & (2^{10} & \times & 2^{10} & \times & 2^{10} & \times & 2^{10} & \times & 2^{10} & \times & 2^{10} & \times & 2^{10} & \times & 8) \text{ bits} \\
 & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\
 & \text{EB} & & \text{PB} & & \text{TB} & & \text{GB} & & \text{MB} & & \text{KB} & & \text{B} & & & &
 \end{array}$$

“Producto de potencias de igual base” se suman los exponentes de forma que podría expresarse el cálculo anterior como:

$$16 \text{ ZiB} = (16 \times 2^{70} \times 8) \text{ bits}$$

El número **8** que se ha incluido para poder pasar la cantidad de Bytes a bits, también se puede expresar como 2^3 , incluso 16 también podría ser expresado como 2^4 .

$$16 \text{ ZiB} = (16 \times 2^{70} \times 8) \text{ bits} = (2^4 \times 2^{70} \times 2^3) \text{ Bits} = 2^{77} \text{ bits}$$

Resulta por supuesto más conveniente decir 16ZiB en vez de 2^{77} para expresar dicha cantidad en bits.

A continuación se presentan operaciones aritméticas las cuales serán efectuadas únicamente en binario.

2.A.6. Operaciones Aritméticas

Todos los sistemas de numeración posicionales permiten realizar operaciones aritméticas. Las reglas aprendidas desde el colegio primario serán aplicadas en todos los sistemas de numeración posicionales. Es importante advertir que no es posible sumar cantidades que no expresen una misma cosa (por ejemplo: 7 caballos + 5 libros), es decir que no será posible tomar un número en base 6 e intentar sumarle un número en base 8 (para esto será necesario previamente expresarlos en una misma base). Del mismo modo si se suman dos números en base 6 el resultado deberá estar en base 6 (por ejemplo: si se suman 7 naranjas + 3 naranjas, el resultado no podrá ser 10 peras).

2.A.6.1. Suma

Para realizar una suma primeramente se deberán encolumnar los números a sumar, tal como se muestra en la figura 2.12.

$$\begin{array}{r}
 \\
 + \\
 \hline
 1
 \end{array}$$

Figura 2.A.12. Suma en base 2

Se pueden resolver sumas en distintas bases, aplicando las mismas reglas y metodología que para el sistema decimal, sólo será necesario analizar el resultado obtenido al sumar cada columna, si el resultado obtenido es un símbolo perteneciente a la base en la cual se está sumando se procederá a escribirlo, caso contrario deberá ser convertido expresándolo en la base destino. En el caso de base 2 en la cuenta realizada en la figura 2.A.12 se presenta la suma en binario para lo cual se ha encolumnado los dígitos previamente, se comienza como en decimal por la columna de menor peso (la de más a la derecha) $0+1=1$ y eso es lo que se escribe como resultado de la columna, se continúa por la columna adyacente $1+1=2$ pero ese 2 se debe escribir en binario que es **10**, se pone el 0 y se lleva el **1** a la siguiente columna, entonces en la columna de más a la derecha estará **1+1+1** (en donde el 1 resaltado en negrita es un acarreo de la columna anterior), esto da 3 en decimal lo cual en binario es **11** así que se pone el 1 en la columna y se acarea el otro 1 a la columna siguiente.

Se plantea en la figura 2.A.13 el caso de la de 4 números de distinta cantidad de dígitos los cuales han sido previamente alineados.

$$\begin{array}{r}
 \\
 \\
 \\
 + \\
 \hline
 1
 \end{array}$$

Figura 2.A.13. Suma en binario

Se propone realizar la suma en decimal por columna y luego pensar como se expresa dicho resultado en binario (en binario solo será válido el valor arrojado por una columna que de por resultado 0 ó 1, todos los resultados restantes deberán

ser convertidos a binario). En la primer columna sombreada en la figura 2.14, se suma $1+0+1+0=2$ en decimal luego se debe pensar como se escribe el 2 en base 2 (la base escrita en su base es 10), con lo cual se anota el 0 en esa columna y se acarrea un 1 a la columna siguiente. En la columna siguiente se deberá sumar el acarreo a los símbolos propios de la columna: $1+(1+1+0+1)=4$, el 4 decimal se escribe 100 en binario, se anota 0 en la columna y se acarrea 10 a la columna siguiente (observe que en este caso el acarreo consta de dos dígitos). Nótese que ese 10 surge de un acarreo de una operación en base 2 no representa al número diez decimal sino que representa al dos decimal, con lo cual la siguiente columna a sumar será: $2+(0+0+1)=3$ que se escribe 11 con lo cual se anota el 1 en dicha columna y se acarrea el otro 1 a la columna siguiente. En la última columna a sumar a quedado $1+(1+1)=3$ que se escribe 11, se anota un 1 en esa columna y se acarrea un 1 a la siguiente. Ver Figura 2.A.14.

$$\begin{array}{r}
 1 \quad 1 \quad 10 \quad 1 \\
 \downarrow \\
 \begin{array}{r}
 1 \quad 1 \quad 1 \quad 0 \quad 0 \\
 1 \quad 0 \quad 1 \quad 0 \quad 0 \\
 1 \quad 0 \quad 0 \quad 1 \quad 0 \\
 1 \quad 1 \quad 1 \quad 0 \quad 0 \\
 \hline
 1 \quad 1 \quad 1 \quad 0 \quad 0
 \end{array}
 \end{array}$$

Figura 2.A.14. Suma en binario

2.A.6.2. Resta

Al igual que para la suma será necesario contemplar todas las reglas que se aplican en decimal para realizar una resta. No es necesario explicar cómo se resta en decimal: $26 - 7$ sin embargo es importante partir de los detalles que encierra esta cuenta para poder luego ocuparse en restas en otras bases. En el colegio primario la maestra explicaba en el pizarrón, más o menos lo siguiente: “No puedo restar 6 con el 7 entonces le pide al compañero, el compañero tiene 2 queda en 1 y le pasa 1 a la columna que le pidió prestado”. Acá surgen un gran interrogante (ver figura 2.A.15), ¿porque si se le pidió 1 al compañero y la columna que lo pidió tenía 6 no queda en 7 sino en 16, porque el 1 se anota delante del número?

$$\begin{array}{r}
 1 \\
 \cong \quad 16 \\
 - \quad 7 \\
 \hline
 \end{array}$$

Figura 2.A.15. Método de resta

Sucede que el número 26 está compuesto por 2 decenas y 6 unidades. Al quitar una decena se está sacando 10 unidades las cuales pueden sumarse a la columna de las unidades y el numero seguirá representando el mismo valor.

2 decenas + 6 unidades \equiv 1 decena + 16 unidades

Esto explica porque siempre es posible quitarle a una columna y otorgarle lo equivalente a lo quitado a otra columna de menor peso.

Podría entonces pensarse que si saca 1 decena se le otorga a la columna de las unidades 10, si se saca 1 centena se le otorgan 10 a las columnas de las decenas. Dada esta

explicación es posible comprender que éste 10 que reciben las columnas, en las que no era posible efectuar la resta, se debe a que se está trabajando en decimal. Este proceso de “pedir prestado” se denomina Borrow.

Cada vez que no se puede efectuar la resta se quita de la columna inmediata “una base” y esta entrega a la que lo solicito “una base” (proceso de Borrow). Es decir en base 3, cada vez que se quite 1 la columna anterior recibirá 3. Entonces se plantea como sería la resta en base 2.

- **Resta en base 2:** Se desea realizar $101_2 - 11_2$ (siendo el 101 el minuendo y el 11 el sustraendo). En la figura 2.A.17 se muestra el cálculo a realizar, al restar la primer columna 1-1 el resultado es 0, la siguiente columna no puede ser restada no es posible a 0 quitarle 1 con lo cual se recurre a la columna siguiente que tiene 1 y queda en 0, la columna anterior recibe la base que es 2 que escrita en su base se consignará como 10, la cuenta a resolver entonces será $2+0=2$ y ese valor menos 1, el resultado de la columna será entonces 1, finalmente en la última columna ha quedado $1-1=0$ (ese cero podría no anotarse ya que los ceros a la izquierda de la parte entera de un número no tienen valor siendo lo mismo 10_2 que 010_2). Ver Figura 2.A.16.

$$\begin{array}{r}
 0 \quad 2 \rightarrow 10 \\
 \begin{array}{r}
 1 \quad 0 \quad 1_2 \\
 - \quad 1 \quad 1_2 \\
 \hline
 0 \quad 1 \quad 0_2
 \end{array}
 \end{array}$$

Figura 2.A.16. Resta en base 2

Sólo en está primer resta se aclarará el valor de la base en decimal (en la segunda columna de la cuenta) para las futuras restas solo se pondrá 10 ya que la base escrita en su base es 10, y el lector deberá recordar que eso en decimal se leerá como el valor de la base en cuestión.

2.A.6.3. Multiplicación

Para comenzar se plantea un ejemplo sencillo, se desea realizar: 101×11 (ambos números expresados en base 2), el cual se muestra en la figura 2.A.18. Cabe destacar que el método a realizar es el mismo que en decimal, como hay dos dígitos en el segundo factor habrá dos multiplicaciones parciales las cuales luego deben ser sumadas y el resultado de esa suma también será expresado en base 2.

$$\begin{array}{r}
 \begin{array}{r}
 1 \quad 0 \quad 1_2 \\
 \times 1 \quad 1_2 \\
 \hline
 1 \quad 0 \quad 1 \\
 1 \quad 0 \quad 1 \quad - \\
 \hline
 1 \quad 1 \quad 1 \quad 1_2
 \end{array}
 \end{array}$$

Figura 2.A.18. Multiplicación en base 2

2.A.6.5. División

Como el resto de las operaciones aritméticas presentadas la división se puede realizar en todos los sistemas posicionales aplicando las mismas reglas que en el sistema decimal.

Se presenta el caso de una división en decimal, $19/5$ en donde 19 es el dividendo y 5 es el divisor. El cociente da 3 y el resto da 4. Esto no requiere esfuerzo alguno debido a que el lector está acostumbrado a realizar operaciones en base 10. Sin embargo resulta necesario aplicar los mecanismos de resolución utilizados en decimal para resolver divisiones en otras bases.

Se propone a continuación dos formas de resolver la división:

1. **Por tanteo:** Consiste en analizar por qué número debe multiplicarse al divisor para obtener un resultado lo más cercano al dividendo sin sobrepasarlo ($5 \times ? \leq 19$). De todos los números que cumplen con la condición el 3 es el que arroja un resultado más próximo, dicho valor es el cociente de la división y el resto será la diferencia entre el dividendo y el resultado al realizar la productoria entre el divisor y el número escogido: $19 - (5 \times 3) = 19 - 15 = 4$.
2. **Restas Sucesivas:** Permite ver cuántas veces el divisor cabe en el dividendo. ¿Cuántas veces es posible restarle 5 al número 19? Si se realizan las restas sucesivas se verá que la cantidad de veces es 3 (cociente de la división) y en la última resta quedan 4 unidades (resto de la división)

En la figura 2.A.19 se presentan ambos métodos en la resolución de la división $19/5$ en base 10.

$\begin{array}{r} 19 \overline{) 5} \\ -15 \\ \hline 4 \end{array}$	$\begin{array}{r} 19 \overline{) 5} \\ 4 \\ \hline 19 \\ -15 \\ \hline 4 \end{array}$	$\begin{array}{r} 19 \\ -5 \\ \hline 14 \\ -5 \\ \hline 9 \\ -5 \\ \hline 4 \end{array}$
Por tanteo	Restas Sucesivas	
Cociente 3	Cociente 3	
Resto 4	Resto 4	

Figura 2.A.19. División en base 10 - Métodos

La explicación anterior es algo trivial cuando se trata del sistema de numeración decimal, ¿porque se explicará en este libro cuentas del tipo $10/3$ si es evidente que el resultado será 3 y el resto 1?. Sin embargo ¿será tan evidente el resultado si estos números estuviesen expresados en base 7?

Como las operaciones aritméticas, para el alcance de esta unidad, se efectuarán en base 2 se explicará directamente las dos posibilidades en dicha base.

A continuación se aplicará los métodos de división para resolver una división en base 2, tomando como ejemplo $1101/100$:

1. Restas sucesivas: En la figura 2.A.20 se procede a realizar la misma división por medio de restas sucesivas. Cabe destacar que dichas restas deberán efectuarse en base 2. Pudo efectuarse 3 restas (resultado del cociente) y en la última de ellas se produce un resto de 1. Ahora bien no se puede escribir 3

como resultado del cociente ya que debe expresarse en base 2 entonces se indica como cociente 11

1101	100	1101
1	11	- 100
Por restas sucesivas		1001
Cociente: cantidad de restas realizadas. Resultado: 3		- 100
Resto: Valor que queda en la última cuenta sin poder ser restado nuevamente. Resultado: 1		101
		- 100
		1

Figura 2.A.20 División en base 2- Restas sucesivas

2. Por tanteo: Como es binario sólo se puede poner como cifras 0 ó 1, lo cual simplifica la cuestión de saber a cuanto le está. En la **figura 2.A.21** se muestra el procedimiento como el denominador tiene 3 dígitos se tomarán 3 del numerador para ver si cabe uno en otro si cabe le está a 1 sino cabe deberá tomarse un dígito adicional. Luego se realiza la resta y se baja el siguiente dígito nuevamente se prueba si le está en cuyo caso irá uno caso contrario 0.

1101	100
- 100	11
101	
- 100	
1	
Por tanteo	
Cociente	11
Resto	1

Figura 2.A.21. División en base 2- Por tanteo

Ambos métodos son válidos pudiéndose elegir uno u otro en forma indistinta.

PARTE B. SISTEMAS NUMÉRICOS PARA APLICACIONES INFORMÁTICAS

Representación interna de la información

- Autora: C.C. Mabel Cilenti -

2.B.1. Introducción

Los caracteres utilizados en los lenguajes naturales humanos y los números decimales son de uso corriente y comprendidos por las personas, ellos conforman lo que se denomina “representación de datos externa”, pero la computadora por su naturaleza electrónica, no puede entender y usar esos símbolos y números directamente, los datos para ser tratados deben estar en forma binaria. Con representación interna de datos se hace referencia a los distintos métodos de representar el lenguaje natural y los números decimales en binario dentro de la computadora, como fue explicado en 2.A.5.

2.B.2. Formas de representar a los números enteros

Los números enteros son aquellos que no poseen parte fraccionaria, se los representa en notación de **punto fijo** (en Argentina “la coma”), ubicando el punto decimal (“la coma”) siempre en un lugar fijo que es a la derecha de la cifra menos significativa, de allí su nombre de “punto fijo”. Por ejemplo: el número entero 532 es igual al 532. representado en punto fijo. Nótese la ubicación del punto fijo a la derecha del dígito menos significativo, en este caso el 2.

Independientemente del método utilizado para la representación de los números se debe conocer n , la cantidad (fija) de bits que la computadora utiliza para almacenar información a esta cantidad de bits que la computadora lee o graba todos juntos en una sola acción se la denomina: **longitud de palabra**. Esta puede ser de 8 bits, 16 bits, 32 bits, 64 bits, etc.

Se pueden considerar las siguientes representaciones de números enteros:

- Binario puro (enteros sin signo)
- Signo y módulo (también denominado signo y valor absoluto).
- Complemento a la base
- Complemento a la base menos 1

Como el sistema de numeración a emplear es binario (base 2), en las dos últimas representaciones puede sustituirse la palabra Base por el valor de la base, es decir 2. De esta forma existen dos maneras de mencionar a dichas representaciones:

Complemento a la base (C_B)= Complemento a dos (C_2)

Complemento a la base menos 1 (C_{B-1})= Complemento a 1 (C_1)

En cada representación se tendrá en cuenta dos parámetros importantes:

- **Capacidad de representación:** la cantidad de tiras de bits distintas que se pueden representar. Por ejemplo si se tiene un sistema restringido a 4 bits, existen $2^4 = 16$ representaciones distintas.
- **Rango o intervalo de representación:** dado por el número más pequeño y el más grande representables. Por ejemplo, en binario sin signo, con 4 dígitos es $[0,15]$.

2.B.2.1. Representación en binario puro. Enteros sin signo

Existen algunas magnitudes que son siempre positivas, por ejemplo la edad de una persona: **18** años, la dirección de una vivienda: Rivadavia **1234**, el número de DNI: **12345678**.

Los enteros sin signo (siempre positivos) poseen un rango entre 0 y el infinito positivo, las computadoras no pueden almacenar todos los enteros en este intervalo, para ello necesitarían un número infinito de bits, lo que implicaría una computadora con una capacidad de almacenamiento infinita, cosa imposible, por lo tanto el máximo entero dependerá de la longitud de palabra utilizada.

En esta convención la representación del valor numérico coincide con su expresión en binario, siendo siempre positivo (ver ejemplo 2.B.1).

Ejemplo 2.B.1. – Representación en binario puro (sin signo)

Representar los números decimales 22 y 197 considerando una palabra de $n = 8$ bits

0	0	0	1	0	1	1	0
Magnitud = 22_{10}							
1	1	0	0	0	1	0	1
Magnitud = 197_{10}							

La cantidad de números que se pueden representar con n bits es 2^n , por ejemplo: para $n = 8$ bits existen 256 representaciones distintas. Pero como los enteros incluyen al 0 y la cantidad total de números que se pueden representar es 2^n , el rango se expresa:

En general para palabras de n bits el rango para ENTEROS SIN SIGNO es desde 0 a $2^n - 1$

Para palabras de 8 bits el rango es desde 0 a $2^8 - 1$

Para palabras de 8 bits el rango es desde 0 a 256 - 1

Para palabras de 8 bits el rango es desde 0 a 255

La tabla 2.B.1 muestra según la cantidad de bits que se utilicen para el almacenamiento, el rango de representación en binario puro y algunos ejemplos.

Tabla 2.B.1. Rango de representación en binario puro

Cantidad de bits	Rango de representación
n	$[0 \dots 2^n - 1]$
8	$[0 \dots 255]$
16	$[0 \dots 65535]$

Ejercicio 2.B.1. – Sugerido

Representar el número decimal 191 en binario puro y $n = 8$ bits.

2.B.2.2. Representación en signo y módulo. SM

Para la representación de números enteros con signo se asocia los dos posibles valores del signo (+ y -) a los dos dígitos (0 y 1) del sistema binario mediante la utilización de un bit.

En esta convención el MSB (bit más significativo, o sea el bit más a la izquierda en la palabra) representa el signo, por convención 0 para + y 1 para el signo -, en el resto de los $n-1$ bits va el valor absoluto o módulo del número en binario. Para $n = 8$ bits, el formato es el presentado en la Figura 2.B.1.

SIGNO	MÓDULO o VALOR ABSOLUTO DEL NÚMERO						

Figura 2.B.1. Formato de representación en Signo y Módulo para 8 bits.

El ejemplo 2.B.2. muestra las representaciones del mismo módulo con signo positivo y negativo. Obsérvese que la única diferencia entre ambas representaciones es el signo (dígito más significativo, ubicado a la izquierda).

Ejemplo 2.B.2. – Representación en módulo y signo

Representar el valor +35 y -35, con $n = 8$ bits, incluido el signo

$$\begin{array}{rcl}
 \text{Valor } +35_{10} & = & \mathbf{0} \ 0100011_2 \\
 \text{Valor } -35_{10} & = & \mathbf{1} \ 0100011_2
 \end{array}$$

\uparrow └──────────┘
Signo Valor absoluto

Esta forma es muy simple de implementar, pero de baja utilidad ya que si bien admite números signados, presenta algunos inconvenientes:

- Tiene dos representaciones para el cero, una positiva y otra negativa, (para $n = 8$ bits) **00000000** y **10000000**. No tiene sentido contar con dos combinaciones para representar el cero, tampoco tiene sentido matemático contar con un signo para expresar al cero como positivo o negativo. Por ello es preciso descartar una de estas combinaciones optándose por no utilizar la de signo negativo. Por lo tanto si bien la cantidad de combinaciones distintas son 2^n , 256 para el caso de 8 bits, solo pueden representarse $2^8 - 1 = 255$ combinaciones.
- No permite realizar operaciones aritméticas, es decir, si bien el MSB indica el signo del valor representado, al hacer una operación aritmética dicho bit debe ser tomado como un bit más, sin hacer diferenciación en la operación. Esta situación se muestra en la figura 2.B.1 donde al realizar la suma de los números $+36$ y -36 en esta convención, se obtiene un resultado no nulo (-72), cuando el correcto es cero.

$+ 36_{10}$	00100100₂	
$+ - 36_{10}$	+ 10100100₂	
0_{10}	11001000₂	(es el -72) ₁₀

Figura 2.B.2. Representación SM, no válida para operaciones aritméticas.

La cantidad de valores que se pueden representar con n bits sigue siendo 2^n , pero ahora la mitad serán positivos y la mitad negativos.

En la figura 2.B.3 se procede a escribir todas las combinaciones posibles con 3 bits (8 combinaciones), al considerar que el MSB representa el signo puede observarse que las 4 primeras se corresponderían con números positivos y las 4 restantes con números negativos.

	MSB			
+	0	1	1	$= +3$
	0	1	0	$= +2$
	0	0	1	$= +1$
	0	0	0	$= +0$
-	1	0	0	$= -0$ (combinación que se descarta)
	1	0	1	$= -1$
	1	1	0	$= -2$
	1	1	1	$= -3$

Figura 2.B.3. Representación SM, no válida para operaciones aritméticas

El rango de representación es diferente con respecto al visto para enteros sin signo, debido a que se ha utilizado un bit para indicar el signo del número, quedando $n-1$ bits para representar su valor absoluto, por lo tanto el máximo valor posible es $2^{n-1} - 1$, y el menor negativo es $-(2^{n-1} - 1)$.

En la Tabla 2.B.2. se indica el rango de representación en signo y módulo según la cantidad de bits.

Tabla 2.B.2. Rango de representación en signo y módulo

Cantidad de bits	Rango de representación
N	$[-(2^{n-1}-1).., +0 .. 2^{n-1}-1]$
8	$[-127,+127]$
16	$[-32767,+32767]$

Ejercicio 2.B.3 - Sugerido

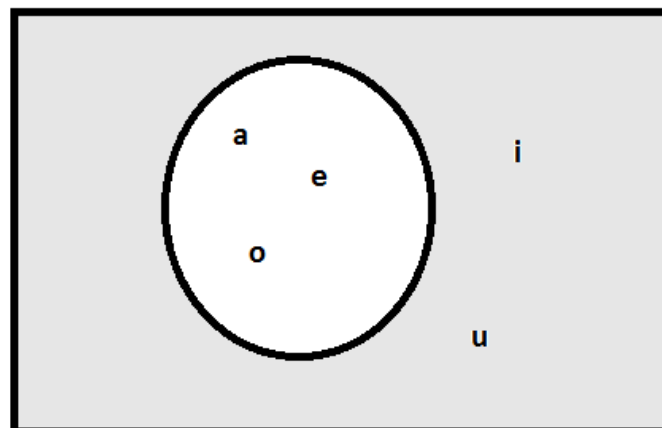
Indique cuál es el mínimo número de bits necesarios para representar en binario el número decimal -256 en signo y módulo.

2.B.3. Complemento de un número.

2.B.3.1. COMPLEMENTO A LA BASE

El concepto de complemento se refiere a lo que le falta a un conjunto para alcanzar el todo.

Por ejemplo: el complemento de {a, e, o} con respecto a todas las vocales es {i, u}. Lo que le falta a: {a, e, o} para “completar” todas las vocales es {i, u}. Ver Figura 2.B.4.



vocales

Figura 2.B.4. Complemento.

El **complemento a la base** $C_{N,B}$ de un número positivo N de base B, es la diferencia entre la base elevada al número de cifras empleada para la representación, y el valor que se desea representar, esto se plasma en la expresión 2.B.1.

$$C_{N,B} = B^n - N$$

N: número a representar, entero o fraccionario

B: base del sistema de numeración

n: cantidad de cifras empleadas en la representación del número

Expresión 2.B.1. Definición de complemento a la base

NÚMERO = 3			COMPLEMENTO A LA BASE = 7							10
0	1	2	3	4	5	6	7	8	9	

Figura 2.B.4.: Complemento a la Base del número 3 ($C_{3,10} = 7$)

El complemento se calcula siempre por exceso, o sea tomando potencias de B que sean superiores a N.

En el Ejemplo 2.B.4 mostrado a continuación se presentan tres casos para los cuales se ha calculado el complemento a la base.

Ejemplo 2.B.4. – Complemento a la base

$$C_{287,10} = 10^3 - 287_{10} = 1000_{10} - 287_{10} = 713_{10}$$

$$C_{72,8} = 10^2 - 72_8 = 100_8 - 72_8 = 6_8$$

$$C_{1101,2} = 10^4 - 1101_2 = 10000_2 - 1101_2 = 0011_2$$

Tal como se ha comprobado en la Parte A, la base de un sistema de numeración expresada en dicha base será siempre 10 (“uno cero”), por ello B se expresa como 10 en todos los casos, por facilidad n (exponente al que se eleva la base se ha representado en decimal independientemente de la base en la que se esté trabajando). Tomando el tercer caso presentado en el ejemplo 3.3 el exponente 4 escrito en binario sería 100.

Es importante notar por otra parte que la resta a realizar deberá llevarse a cabo en la base que se está trabajando, ver figura 2.B.5.

1	0	0	0	0	2
-	1	1	0	1	2
0	0	1	1	2	

Figura 2.B.5. Cálculo del Complemento a la Base del número 1101₂

Si bien esta cuenta no resulta difícil de resolver, se provee a continuación una regla práctica que permite obtener el complemento a la base de un número binario sin necesidad de realizar una cuenta:

Se recorre el número a complementar de derecha a izquierda, hasta el primer bit en 1 inclusive no se modifican, y el resto de los bits se invierten, unos por ceros y ceros por unos. En la figura 2.B.6 se presentan algunos ejemplos.

A = 011011	A = 010100
←----	←----
C _{A,2} = 100101	C _{A,2} = 101100

Figura 2.B.6. Regla práctica para hallar el complemento a 2

2.B.3.2. COMPLEMENTO A LA BASE MENOS UNO

El **complemento a la base menos uno** de un número positivo, es la diferencia entre la base elevada al número de cifras destinadas a representar el número menos uno y el valor que se desea representar:

$$C_{N,B-1} = (B^n - 1) - N$$

N es el número a representar, entero o fraccionario y B es la base del sistema de numeración en que está representado el número, en este caso también el complemento es por exceso. En el ejemplo 2.B.5 se realizan complementos a la base menos uno de números expresados en distintas bases.

Ejemplo 2.B.5. – Complemento a la base menos uno

$$C_{287,10-1} = (10^3 - 1) - 287 = 999 - 287 = 712_{10}$$

$$C_{72,8-1} = (10^2 - 1) - 72 = 77 - 72 = 05_8$$

$$C_{1101,2-1} = (10^4 - 1) - 1101 = 1111 - 1101 = 0010_2$$

Regla práctica para obtener el complemento a la base menos uno de un número binario (complemento a 1): se recorre el número a complementar cambiando los unos por ceros y viceversa tal como se muestra en la figura 2.B.5.

$A = \begin{array}{c} 011011 \\ \leftarrow \text{----} \end{array}$	$A = \begin{array}{c} 010100 \\ \leftarrow \text{----} \end{array}$
$C_{A,B-1} = 100100$	$C_{A,B-1} = 101011$

Figura 2.B.5. Regla práctica para hallar el complemento a 1

Se puede calcular el complemento a la base de un número, obteniendo primero su representación en complemento a la base menos uno, y luego sumarle 1 en la posición del bit menos significativo (ver el ejemplo 2.B.6).

Ejemplo 2.B.6. - Complemento a la base, a través del complemento a la base menos uno

$$C_{287,10} = C_{287,10-1} + 1 = 712_{10} + 1 = 713_{10}$$

$$C_{72,8} = C_{72,8-1} + 1 = 05_8 + 1 = 06_8$$

$$C_{1101,2} = C_{1101,2-1} + 1 = 0010_2 + 1 = 0011_2$$

Comparar los resultados obtenidos con los del ejemplo 2.B.3.

Ejercicio 2.B.7. - Sugerido

- a) Calcule el complemento a la base de: $A1F_H$ y 101100100_2
 b) calcule el complemento a la base -1 de: 467_8 y 11100010_2

2.B.3.3 UTILIZACIÓN DEL COMPLEMENTO EN OPERACIONES DE RESTA

El término complemento de un número es aplicable a todos los sistemas de numeración posicionales, pero es utilizado fundamentalmente en el sistema binario para entregar una adecuada representación de números signados que permita operar con ellos, el mismo puede ser utilizado para realizar la operación de la resta a partir de una operación de suma. Si se desea restar dos números M y S podría plantearse lo mostrado en la tabla 2.B.3.

Tabla 2.B.3. Expresiones equivalentes para la resta entre dos números

Expresión de Partida:	$R = M - S$
La expresión no cambia si se suma y resta B^n (una potencia de la base, tal que sea mayor que M y S)	$R = M + B^n - B^n - S$
Reacomodamos los términos	$R = M + (B^n - S) - B^n$
El paréntesis de la expresión anterior representa al complemento a la base del número S	$R = M + C_{S,B} - B^n$

A partir de la tabla 2.B.3 se puede concluir que la resta entre M y S se puede realizar sumando a M el complemento a la base de S y eliminando luego la potencia de la base que se utilizó para el cálculo del complemento, como se muestra en la expresión 2.B.2.

$$R = M + C_{S,B} - B^n$$

Expresión 2.B.2. Resta utilizando complemento

A modo de ejemplo, en la figura 2.B.6 se realiza la resta de los números 128 y 39, expresados en el sistema decimal de numeración, utilizando el concepto de complemento a la base:

$$\begin{array}{rcl}
 428 - 39 & = & 389 \\
 428 + 61 & = & 489 - 10^2 = 389 \\
 \downarrow & & \text{(se resta el } 10^2 \text{ del complemento)} \\
 C_{39,10} = 10^2 - 39 & &
 \end{array}$$

Figura 2.B.6. Resta de dos números a través de la suma del complemento

Se restó del resultado obtenido, la potencia de la base utilizada para calcular el complemento (10^2).

En la figura 2.B.7 se puede observar la resta de los números binarios 11001 y 00101 utilizando el complemento a la base.

$$\begin{array}{rcl}
 11001 - 00101 & = & 10100 \\
 11001 + 11011 & = & \cancel{10100} \text{ se resta } 10^5 \\
 \downarrow & & \\
 C_{00101,2} = 10^{101} - 00101 & &
 \end{array}$$

Figura 2.B.7. Resta de dos números binarios en complemento a 2

Se restó del resultado obtenido, la potencia de la base utilizada para calcular el complemento (10^5), sencillamente se elimina el bit más a la izquierda que excede la cantidad de bits usados en la representación.

Si bien el cálculo de la resta utilizando la suma del complemento es más largo, en el caso de las computadoras que operan en binario, el resolver la resta por medio de la suma del número complementado permite un ahorro en la estructura circuital de la unidad de cálculo, dado que las operaciones de suma y resta se resuelven con el mismo circuito lógico.

Por todo lo expuesto, SÓLO se complementarán los números negativos.

Otra forma de resolver la resta entre M y S es sumar a M el complemento a la base menos 1 de S, eliminar luego la potencia de la base y al resultado obtenido sumarle 1 (ver expresión 2.B.3)

$$R = M + (B^n - 1) - (B^n - 1) - S$$

$$R = M + [(B^n - 1) - S] - B^n + 1$$

$$R = M + C_{S,B-1} - B^n + 1$$

Expresión 2.B.3. Resta a través del complemento a 1

Otro caso a considerar es aquél en el cual el sustraendo es mayor que el minuendo y por lo tanto el resultado de la resta es negativo. La figura 2.B.8 ejemplifica esta situación al restar el número decimales 128 de 39, utilizando el concepto de complemento a la base:

$$\begin{array}{r} 39 - 128 = -89 \\ 39 + 872 = 911 \\ \downarrow \\ C_{128,10} = 10^3 - 128 \end{array}$$

Figura 2.B.8. Resta utilizando Complemento a 2

En esta oportunidad, la potencia de la base utilizada en el cálculo del complemento (10^3) no aparece explícitamente en el resultado como en los casos anteriores, por lo tanto se debe restar del mismo para verificar el resultado de la resta: $911 - 1000 = -89$, lo que equivale a complementar el resultado obtenido.

Por lo tanto, cuando el resultado de una resta es negativo, la suma del minuendo y el complemento del sustraendo da como resultado el **complemento** del resultado real.

A modo de ejercitación resolver el ejercicio 2.B.8.

Ejercicio 2.B.8. – Sugerido

Realizar en binario las siguientes operaciones de números decimales:

- a) $235 - 372$
- b) $372 - 235$

2.B.3.4. Representación en complemento a la base-1 (C_{B-1})

Aquí también el bit más significativo (MSB) representa el signo, 0 para los positivos y 1 para los negativos. Para los números positivos en los (n-1) bits restantes se coloca

el valor absoluto o módulo igual que en SM y para los números negativos va el complemento a la base menos 1 del número que se está representando.

Solamente se utiliza la representación en complemento para los números negativos.


La tabla 2.B.4 proporciona ejemplos para el caso de $n=8$ bits.

Tabla 2.B.4. Representación en C_1

Número a representar	Signo	Módulo
$+37_{10}$	0	0100101
-37_{10}	1	1011010

A continuación la tabla.3.5 presenta todos los valores posibles a representar en C_1 con $n=3$ bits.

Tabla 2.B.5. Representación en C_1

Decimal	Representación
0	000
1	001
2	010
3	011
-3	100
-2	101
-1	110
-0	111
	

Se observa las siguientes características para esta representación:

- El bit de signo es 0 para los números positivos y 1 para los negativos.
- La cantidad de combinaciones posibles con n bits es 2^n , de las cuales la mitad corresponde a números positivos y la mitad a negativos.
- Existen dos representaciones para el 0, positivo y negativo. La combinación 111, correspondiente al -0 no es utilizada ya que no tiene significado matemático, por lo cual de las 2^n combinaciones posibles solo se pueden representar $2^n - 1$ valores numéricos.
- El complemento a uno de un número positivo, es su correspondiente número negativo y viceversa.
- La posibilidad de realizar operaciones aritméticas y obtener un resultado correcto, como se ejemplifica en la figura 2.B.11.


$+36_{10}$	00100100
$+ -13_{10}$	$+ 11110010$
$+23_{10}$	00010110
	

Figura 2.B.11. Operación aritmética en C_{B-1}

Se puede ver en el resultado la aparición, a la izquierda del bit de signo, de un bit extra (se agregó a los ocho bits originales con que se representaron los datos), denominado **bit de arrastre o carry**.

Dicho bit **debe ser descartado** al realizarse una operación de resta mediante la suma del complemento del sustraendo. Además al calcular dicho complemento se ha sumado una potencia de la base menos uno ($B^8 - 1$) y al eliminar este 1 (el carry), se saca una potencia de la base (B^8), o sea se saca uno más de lo que se sumó, por lo cual, para comprobar el resultado correcto se deberá **sumar 1 al resultado obtenido**.

Cuando al restar (suma por medio de complemento) se produce un Carry, como el resultado no puede tener más dígitos que los números que se están restando, este se descarta y dejamos de llamarlo Carry a este bit descartado para llamarlo **Borrow**. Si bien en la arquitectura de la computadora este bit seguirá siendo el mismo, la interpretación que le damos es otra. Por haberse añadido o prestado la base para realizar el complemento surge este bit adicional a la cuenta Borrow.

Verificación del resultado correcto:

- El resultado que obtuvo la ALU (Unidad Aritmética y Lógica: parte del computador que se encarga de resolver las operaciones aritméticas y lógicas) al realizar la operación es 00010110 y Carry.
- Como el signo es positivo (0) el resultado es directamente el valor binario representado 00010110₂ o sea 22₁₀.
- Por haber trabajado en C_{B-1} (se sumó B^{n-1}) y eliminado el Carry (B^n), se resta uno más de lo que se sumó, por lo tanto se debe sumar 1 al resultado de la ALU para una correcta interpretación: $22 + 1 = 23$ que es el valor decimal correcto.

La tabla 2.B.6 muestra el rango de representación de números enteros en $C-1$.

Tabla 2.B.6. Rango de representación en Complemento a uno

Cantidad de bits	Intervalo
N	$[-(2^{n-1}-1)..+ 2^{n-1} -1]$
8	$[-127.....+127]$
16	$[-32767.....+32767]$

Ejercicio 2.B.9. - Sugerido

- Representar en 8 bits los números decimales +112 y -112 en C_{B-1} .
- Se cuenta con la siguiente representación 11011001 en Complemento a 1 y $n=8$ bits. ¿Cuál es el número decimal almacenado?
- Represente el número decimal -3 en C_{B-1} , con $n=5$ bits.
- ¿Cuál es el número decimal más grande y el más pequeño que se puede representar con $n=5$ bits en C_{B-1} ?

2.B.3.5. Representación en Complemento a la Base o Complemento a 2

También en esta convención los números positivos se representan en binario mediante el bit de signo en 0 y su valor absoluto (al igual que en SM y en complemento a 1). Para los negativos se coloca 1 en el bit de signo y el módulo en complemento a la base (complemento a 2). La tabla 2.B.7 muestra algunos ejemplos para el caso de $n=8$ bits.

Tabla 2.B.7. Rango de representación en Complemento a la base, C_2

Número a representar	Signo	Módulo
+ 17	0	0010001
-17	1	1101111

Se ilustra en la tabla 2.B.8 todos los valores posibles a representar en Complemento a dos con $n=3$ bits:

Tabla 2.B.8. Representación en Complemento a dos

Decimal	Representación
0	000
1	001
2	010
3	011
-4	100
-3	101
-2	110
-1	111



signo

Las propiedades de esta representación son:

- El bit de signo (el más significativo) es 0 para los números positivos y 1 para los negativos.
- Posee una única representación para el cero. Por ejemplo, con $n = 3$ bits, la representación del $+0$ es 000, para representar el -0 se debe calcular el C_2 de 000, o sea $B^3 - 000 = 1000 - 000 = 1000$, valor imposible de representar en 3 bits. Por lo tanto existe una única convención para el 0, y por supuesto es la que posee bit de signo en 0.
- La posibilidad de realizar operaciones aritméticas y obtener un resultado correcto (ver figura 2.B.10).

$$\begin{array}{r}
 + 36_{10} \quad 00100100 \\
 + - 36_{10} \quad + 11011100 \\
 \hline
 0_{10} \quad \swarrow 00000000 \\
 \quad \quad \quad \uparrow \\
 \quad \quad \quad \text{CARRY}
 \end{array}$$

Figura 2.B.10. Resta en complemento a 2

Nuevamente se observa la aparición en el resultado, del **bit de arrastre o carry**, representando un **Borrow** que debe ser descartado.

- Existen en esta convención, para n bits, 2^n valores a representar, de los cuales la mitad son positivos (considerando al cero positivo dado que su bit de signo es cero) y la mitad negativos.
- Los números positivos a representar comienzan en 0 y los negativos en -1, por lo tanto el valor absoluto del menor negativo supera en 1 al valor absoluto del mayor positivo, en la tabla 3.8 el mayor positivo es 3 y el menor negativo -4.
La tabla 2.B.9 define el rango de representación de números enteros en C_2 .

Tabla 2.B.9. Rango de representación en complemento a 2

Cantidad de bits	Intervalo
n	$[-(2^{n-1})..-1, +0 .. 2^{n-1} - 1]$
8	$[-128..-1, +0..+127]$
16	$[-32768..-1, +0..+32767]$

La convención de complemento a dos es la representación más utilizada en la actualidad.

Ejercicio 2.B.10. - Sugerido

- Representar en 8 bits los números decimales +12 y -12 en C_B .
- Se cuenta con la siguiente representación 11011001 en Complemento a 2 y $n=8$ bits. ¿Cuál es el número decimal almacenado?
- Represente el número decimal -11 en C_B , con $n=6$ bits.
- ¿Cuál es el número decimal más grande y el más pequeño que se puede representar en C_B y $n=4$ bits?

2.B.4. Rangos de representación

Ya se ha dicho que si se utilizan n bits para representar los números enteros, se obtienen 2^n combinaciones distintas. En la tabla 2.B.10 se resume el rango (el mínimo y el máximo valor) para cada una de las convenciones estudiadas anteriormente.

Tabla 2.B.10. Rangos de representación

Representación	Rango
Enteros sin signo	$[0 \dots +2^n - 1]$
Signo y módulo	$[-(2^{n-1} - 1) \dots +(2^{n-1} - 1)]$
Complemento a la base menos uno (C_1)	
Complemento a la base (C_2)	$[-2^{n-1} \dots +(2^{n-1} - 1)]$

2.B.5. Operaciones aritméticas con números signados

Se plantea a continuación diversos ejemplos que indican como se realizan las operaciones de suma y resta en los computadores, teniendo en cuenta las siguientes reglas:

- En el caso de suma o resta de número signados se deberá utilizar alguna de las convenciones que representan los negativos a través del complemento.

- Los bits de signo de cada operando se tratan de la misma forma que los bits de magnitud, además ambos operandos deben ser representados con la misma cantidad de bits de forma que los bits de signo queden encolumnados.
- Los números negativos se expresan a través de su complemento y como bits de signo 1. Por lo tanto cada vez que se obtenga como resultado de una operación 1 como bits de signo, se deberá complementar dicho resultado para lograr una correcta interpretación del resultado real.

En todos los casos se trabajará, a modo de ejemplo, con $n=8$ bits incluido el bit de signo.

Adelantando parte de los conceptos correspondientes a la Unidad 4, se explicará a continuación la suma de enteros dentro de la computadora.

La encargada de realizar la suma es un conjunto de circuitos llamados Unidad Aritmética y Lógica (ALU), según sus siglas en inglés.

La ALU posee dos entradas, una para cada sumando y obtiene una salida con el resultado de la operación solicitada por el programa. Además actualiza la información almacenada en una pequeña memoria llamada Registro de Estados. Dicho registro contiene información relevante sobre el resultado de la operación realizada.

La Figura 2.B.11 muestra el diagrama de la ALU y del Registro de Estados, donde se suman $1 + 2$.

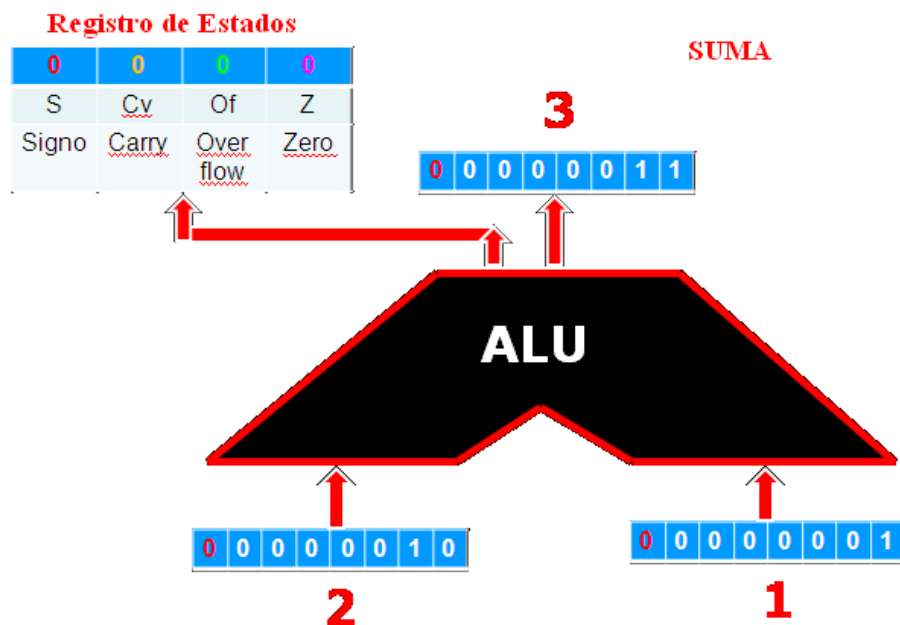


Figura 2.B.11. La ALU y el Registro de Estados

Es posible observar que el Registro de Estados, posee 4 ceros (en este caso), correspondientes a las 4 señales de Estado:

- Signo (S): 0 si el resultado es positivo, 1 si el resultado es negativo.
- Carry o Acarreo (Cy): 0 si no hubo Carry, 1 si hubo Carry
- Overflow (Of): 0 si no hubo Overflow, 1 si hubo Overflow.
- Zero (Z): 0 si el resultado NO dio cero, 1 si el resultado es cero.

Cada una de ellas será explicada en los ítems siguientes.

2.B.5.1. Suma en complemento a dos

Caso 1: La figura 2.B.12 ilustra la suma de dos números decimales positivos, +29 y +18.

$$\begin{array}{rcl}
 +29_{10} & \rightarrow & \boxed{0} \ 0011101 \\
 + & & \\
 +18_{10} & \rightarrow & \boxed{0} \ 0010010 \\
 + & & \\
 +47_{10} & \rightarrow & \boxed{0} \ 0101111 \\
 \hline
 & & \uparrow \text{ Bits de signo}
 \end{array}$$

Figura 2.B.12. Suma dos números positivos en Complemento a 2

En el resultado el bit de signo es 0, por lo tanto el resultado es positivo y su magnitud está expresada en binario puro.

Caso 2: Suma de un número positivo con otro negativo de módulo menor al primero. Sumar los números decimales +29 y -18, representar los operandos con 8 bits incluido el signo. El -18 se debe expresar en complemento a 2 (11101110) (ver figura 2.B.13).

$$\begin{array}{rcl}
 +29_{10} & \rightarrow & \boxed{0} \ 0011101 \\
 + & & \\
 -18_{10} & \rightarrow & \boxed{1} \ 1101110 \\
 + & & \\
 +11_{10} & \rightarrow & \boxed{0} \ 0001011 \\
 \hline
 & & \uparrow \text{ Bits de signo} \\
 & & \uparrow \text{ Este acarreo se elimina}
 \end{array}$$

Figura 2.B.13. suma de un número positivo con uno negativo

El bits de signo del resultado es 0, por lo tanto el resultado es positivo. Además se genera un acarreo en el MSB, el cual se descarta debido a que fue incluido cuando se calculó el complemento a 2 del 18.

Caso 3: Suma de un número positivo con otro negativo de módulo mayor al primero.

Esta situación se muestra en la figura 2.B.14, en la cual se suman los números decimales -29 y +18, representando los operandos con 8 bits incluido el signo. El -29 se debe expresar en complemento a 2 (11100011).

$$\begin{array}{rcl}
 +18_{10} & \rightarrow & \boxed{0} \ 0010010 \\
 + & & \\
 -29_{10} & \rightarrow & \boxed{1} \ 1100011 \\
 + & & \\
 -11_{10} & \rightarrow & \boxed{1} \ 1110101 \\
 \hline
 & & \uparrow \text{ Bits de signo}
 \end{array}$$

Figura 2.B.14: Suma de un número positivo con otro negativo

El bit de signo del resultado es 1, por lo tanto el resultado de la suma es negativo y está expresado en C-2. Para saber el verdadero valor se debe descomplementar a dos el

resultado 1110101, lo cual dará como resultado 00001011 (+11), o sea 1110101 es el -11.

Caso 4: Suma de dos números negativos.

Sumar los números decimales -29 y -18, representar los operandos con 8 bits incluido el signo. Ambos operandos deberán expresarse en complemento a 2 (ver figura 2.B.15).

$$\begin{array}{rcl}
 -29_{10} & \rightarrow & \boxed{1} \ 1100011 \\
 + \ -18_{10} & \rightarrow & \boxed{1} \ 1101110 \\
 \hline
 -47_{10} & \rightarrow & \boxed{1} \ 1010001 \\
 & \uparrow & \text{Bits de signo} \\
 & \uparrow & \text{Este acarreo se elimina}
 \end{array}$$

Figura 2.B.15. Suma de dos números negativos

El bit de signo del resultado es 1, por lo tanto el resultado es negativo y se encuentra en C-2. Para el cálculo del resultado real se deberá complementar el 1010001, obtenido 0101111 (47₁₀), o sea el -47₁₀.

Además se genera un acarreo en el MSB, el cual se descarta debido a que fue incluido cuando se calculó el complemento a 2.

2.B.5.2. Resta en complemento a dos

La operación de resta de números signados usando complemento a 2, se realiza a través de la operación de suma y no es distinta de los diversos casos planteados para la misma

Se recuerda que al restar un número binario (sustraendo) de otro número binario (minuendo), primero se obtiene el complemento del sustraendo y luego este último se lo suma con el minuendo. Como ejemplo en la figura 2.B.16 se resta +12 de +37, representados en 8 bits, incluido el signo y usando complemento a dos.

$$\begin{array}{rcl}
 +37_{10} & \rightarrow & \boxed{0} \ 0100101 \\
 - \ +12_{10} & \rightarrow + & \boxed{1} \ 1110100 \text{ (C-2 de 00001100)} \\
 \hline
 +25_{10} & \rightarrow & \boxed{0} \ 0011001 \\
 & \uparrow & \text{Bits de signo} \\
 & \uparrow & \text{Este acarreo se elimina}
 \end{array}$$

Figura 2.B.16. Resta en Complemento a 2

Al cambiar el sustraendo por su complemento a dos, se convierte a -12 y se está *sumando* -12 y +37 que es igual que restar +12 de +37.

Otro ejemplo, se muestra en la figura 2.B.17, en el cual se desea restar -12₁₀ de -37₁₀, representados en 8 bits, incluido el signo y usando C-2.

$$\begin{array}{rcl}
 -37_{10} & \rightarrow & \boxed{1} \ 1011011 \text{ (-37}_{10} \text{ en C-2)} \\
 - \ -12_{10} & \rightarrow + & \boxed{0} \ 0001100 \\
 \hline
 -25_{10} & \rightarrow & \boxed{1} \ 1100111 \text{ (-25}_{10} \text{ en C-2)}
 \end{array}$$

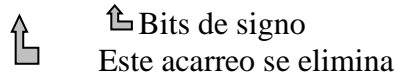


Figura 2.B.17. Resta en complemento a dos

Puede observarse en la figura 2.B.17 que para calcular $(-37) - (-12)$, aplicando la regla de los signos, es equivalente a realizar $(-37) + 12$.

Se propone realizar el ejercicio 2.B.10 para afianzar lo visto.

Ejercicio 2.B.10. – Sugerido.

Realizar en 8 bits incluido el signo y complemento a la base, las siguientes operaciones:

- | | |
|----------------------|---------------------|
| a) $(+97) + (+15)$ | b) $(-39) + (+121)$ |
| c) $(-123) + (+120)$ | d) $(-113) + (-8)$ |
| e) $(+195) - (+87)$ | f) $(-84) - (-23)$ |

2.B.6. Overflow (desborde)

En las operaciones realizadas en binario con números signados, se produce desborde (también llamado exceso o rebalse) si el resultado excede la capacidad de almacenamiento correspondiente a la cantidad de bits utilizados para representar los operandos.

Véase como ejemplo la suma de los números decimales $+70$ y $+82$, expresados ambos en formato de 8 bits incluido el signo y complemento a 2. En la operación representada en la figura 2.B.18, se puede observar que el resultado obtenido es incorrecto, pues el bit de signo representa un número negativo y se están sumando dos números positivos, por lo tanto nunca su suma puede arrojar un resultado negativo.

	$+ 70_{10}$	\rightarrow	0	1000110
$+$	$+ 82_{10}$	\rightarrow	0	1010010
	$+152_{10}$		1	0011000 (magnitud incorrecta es el -104_{10})

Bits de signo incorrecto (se suma dos número positivos, nunca puede dar un número negativo)

Figura 2.B.18. Suma con desborde

Esto no debería sorprender ya que en la notación de Complemento a dos, con 8 bits incluido el signo, el máximo número posible que puede representarse es el $+127$, por lo tanto es imposible representar el $+152$, que es la respuesta correcta.

Si bien el resultado “10011000” si se lo ve como un número entero sin signo es el $(+152)_{10}$, al estar trabajando en complemento a dos, el primer bit, el de signo está indicando un número negativo, y los 7 bits restantes representan al decimal 104 en Complemento a dos, o sea el resultado almacenado es el -104 , incorrecto.

Los dos operandos $(+70)_{10}$ y $(+82)_{10}$ se han podido representar correctamente en formato de 8 bits incluido el signo, no así el resultado de la suma de los mismos $(+152)_{10}$, cuyo módulo supera el rango permitido para esa representación que es $+127_{10}$, requiriendo más de 7 bits para su representación y por lo tanto *desborda* hacia la posición del bit de signo.

Se pueden definir reglas sencillas para detectar desbordamiento (overflow):

- Analizando los signos de los sumandos y del resultado.
 - Si se suman números que tienen el mismo signo y el resultado tiene el signo opuesto se produce overflow.
 - Si los números a sumar son de signos opuestos no puede ocurrir overflow, ya que la magnitud del resultado no supera la del operando mayor.
- Analizando los acarreo que se generan al realizar la operación.
 - En una operación de suma se produce desborde si los bits de acarreo de entrada y de salida de la posición del signo son distintos. En la figura 2.B.19, se presenta un ejemplo de la regla aquí enunciada

En la operación de suma, el acarreo de entrada \neq acarreo de salida esto indica que se produjo Overflow

Acarreo de salida		Acarreo de entrada (de la columna anterior)	
	0	1	
+ 70	→	0	1000110
+ 82	→	0	1010010
+152		1	0011000 (es el -104_{10} , incorrecto)
		↑	Bits de signo

Figura 2.B.19. Regla de overflow en términos de acarreo

En el caso de la resta de números en complemento a dos, se puede llegar a definir otro conjunto de reglas que permitan detectar desborde, pero debido a que la mayoría de los circuitos de resta realizan su tarea sumando al minuendo el complemento a dos del sustraendo, resulta de aplicación las mismas reglas presentadas para la suma.

La computadora tiene un circuito especial para detectar overflow al sumar dos números, el cual enviará una señal especial a la unidad de control indicando esta situación y lo incorrecto del resultado.

2.B.7. Operaciones aritméticas dentro de la ALU

Al momento de obtener el resultado de una operación aritmética se puede concluir que los circuitos de la ALU realizarán una SUMA siempre, ingresaran ambos números (una línea de control indicará si se tratará de una suma o resta) y eso desencadenará en todos los casos una suma de las tiras de bits ingresadas, si es una SUMA la tiras de bits serán sumadas tal como están a la entrada y si es una RESTA uno de los números ingresará en

complemento. Explicado en forma simple podemos decir que la ALU siempre suma, pero gracias al conocimiento de la operación a realizar SUMA o RESTA se podrá interpretar al CARRY como tal o como BORROW a descartar si se trata de una resta.

Ejercicio 2.B.11. – Sugerido con Resultado

Calcular el resultado al realizar la suma de los números signados (ya expresados en Complemento a la Base) que se muestran a continuación. Calcule como quedan los flags luego de cada operación indicando el valor, en el caso de Carry recordar que en la resta se trata de un Borrow (B):

1. 0101 0100 0001 0100
2. 0010 1011 0101 0110
3. 1011 0100 0101 1011
4. 1011 0010 1000 1101
5. 1100 0000 0100 0000

Resultados:

1. ALU: 0110 1000 Flags: S=1, Cy=0, Of=0, Z=0
2. ALU: 1000 0001 Flags: S=1, Cy=0, Of=1, Z=0
3. ALU: 0000 1111 Flags: S=0, Cy=1 (B), Of=0, Z=0
4. ALU: 0011 1111 Flags: S=0, Cy=1, Of=1, Z=0
5. ALU: 0000 0000 Flags: S=0, Cy=1(B), Of=0, Z=1

2.B.8. Comparación de Números enteros signados.

Al comparar dos números puede obtenerse como resultado una de tres alternativas (ver Figura 2.B.20) que sean iguales $A=B$, que el primero A número sea mayor que el segundo B ($A>B$), ó que el A sea menor que el B ($A<B$). En la figura 2.B.21 se presentan algunos casos a modo de ejemplo.



Figura 2.B.20. Resultados posibles de una comparación entre dos números

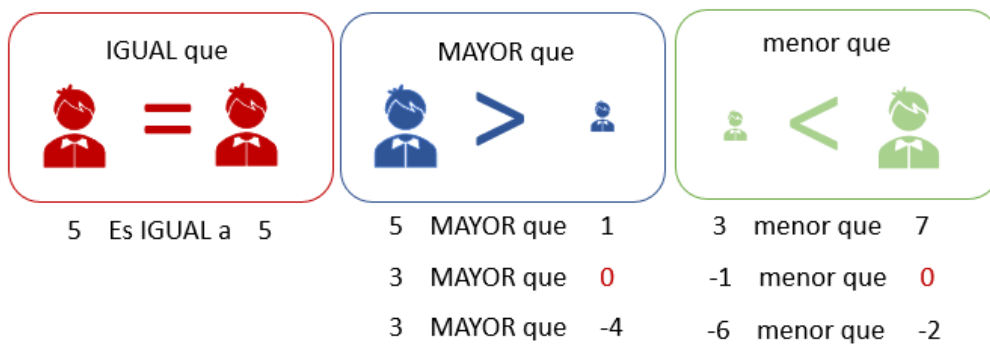


Figura 2.B.21. Ejemplos de valores a comparar, para cada uno de los resultados posibles

Tendremos presente que la máquina siempre suma. Hay ocasiones donde se necesita realizar una resta, la misma se hará a través del complemento del número que tiene signo negativo. Para comparar dos números A y B se deberá analizar primeramente el bit de signo, y luego el resultado obtenido. Este análisis también puede realizarse a través de los flags del registro de estado. En la Figura 2.B.22 se muestra un resumen de los flags del registro de estado.

S Signo

- Representa el signo del resultado.
- 0: resultado positivo
- 1: resultado negativo

Cy Carry

- Acarreo.
- 0: No hubo acarreo
- 1: si hubo acarreo

Of Overflow

- “Analizamos los signos de los operandos y del resultado”. SE PASO DE RANGO
 - $++ = -$ (operandos positivos, resultado negativo)
 - $-- = +$ (operandos negativos, resultado positivo)
- Si los bits de acarreo de entrada (carry in) y de acarreo de salida (carry out) del bit de signo MSB, son distintos, hay Overflow

Z Zero

- Escribe
 - 0: si el resultado **no** es cero
 - 1: si el resultado dio cero (0)

Figura 2.B.22. Resumen de los Flags del Registro de estado.

Los ejemplos mostrados a continuación por una cuestión de simplicidad se realizan con 4 bits incluido el bit de signo.

- **Analizar el Signo del resultado**

En la tabla 2.B.11 vemos los números positivos del [0 .. +7] y los números negativos [-1... -8] representados con Complemento a 2.

Tabla 2.B.11. Rangos de representación

Nros. Positivos		Nros Negativos Representación a través del C2	
0	0000	-8	1000
1	0001	-1	1111
2	0010	-2	1110
3	0011	-3	1101
4	0100	-4	1100
5	0101	-5	1011
6	0110	-6	1010
7	0111	-7	1001

Caso 1: A menor que B

Se consideran los números A= +3 y B=+4, se realiza la resta mostrada en la Figura 2.B.23 como el signo del resultado es negativo, se puede inferir que el número B es mayor que el número A lo que es equivalente a decir que A es menor que B.

$$\begin{array}{rcl}
 \text{A:} & +3 & 0011 \\
 \text{B:} & + & + \\
 & -4 & 1100 \\
 \hline
 & -1 & 1111 \\
 & & \text{Bit de Signo}
 \end{array}
 \quad A < B$$

Figura 2.B.23. Ejemplo de un caso de un número A menor que B

Si el Bit de signo es negativo significa que de los dos números evaluados A y B (siendo B negativo representado en C2). A es menor que B

Caso 2: A Mayor que B

Por lo explicado anteriormente, cabe pensar que si al realizar el cálculo anterior el bit de signo del resultado es “0”, se obtuvo un resultado positivo y entonces se concluye que el primer número sería Mayor que el segundo número a comparar. Pero debe tomarse en consideración también en este caso el flag Z. A continuación, ejemplos:

Se hace la comparación de A=4 con B= -3, siendo que los bits del resultado no son todos 0 se deduce que A>B porque el resultado es positivo.

A: + 4	0100	
B: + - 3	+ 1101	
+ 1	0001	
	 Bit de Signo	A > B

Figura 2.B.24. Ejemplo de un caso de un número A Mayor que B

Caso 3: A igual a B

Bit de signo positivo, se podría pensar que $A > B$, pero al observar todos los bits del resultado son cero ($Z=1$) entonces se concluye que los números eran iguales. A continuación se presenta el caso de $A=4$ y $B=-4$.

A: + 4	0100	
B: + - 4	+ 1100	
0	0000	
	 Bit de Signo	A = B

Figura 2.B.25. Ejemplo de un caso de un número A Igual que B

Con lo expuesto anteriormente se puede concluir que es posible por medio de la resta (suma con complemento) comparar dos números signados, analizando para ello los flags de signo (S) y cero (Z).

- **Comparación de dos números negativos**

Con lo visto anteriormente se deduce que si los números a analizar tienen signos distintos se puede prescindir de realizar comparativas (bit de signo =0) > (bit de signo =1). Caso contrario si son 2 positivos o 2 negativos se puede efectuar la comparación por medio de una resta (suma de complementos), por practicidad si son dos negativos en caso de estar complementados se vuelve a aplicar el complemento y se someten al mismo proceso mostrado previamente.

Ejemplos:

1. Si se trata de: $A = -5$ y $B = -4$.
 Se consideran entonces $(+)5 - (+4) = +1$.
 Se deberán interpretar al revés, si el signo del resultado es positivo como el de magnitud más grande es el primero (A), sería más chico en la recta de negativos. Se concluye que A es menor que B. Dado que los estamos comparando por magnitud, una vez analizado el resultado se considera su signo original.
2. Dado los números: $A = -3$ y $B = -4$.
 Se consideran entonces: $(+)3 - (+4) = -1$ - Al ser el signo el signo del resultado negativo se interpretará que A es Mayor que B.

En la recta numérica (Figura 2.B.26) se puede observar que los números cuanto más a la izquierda están más pequeños serán y cuanto más a la derecha se ubican más grande serán.

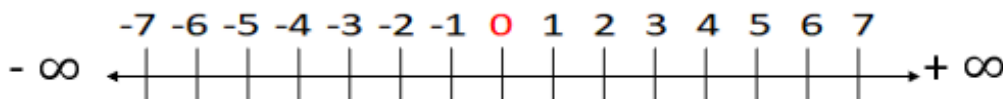


Figura 2.B.26. Recta numérica

2.B.9. Representación binaria de números reales

Habitualmente utilizamos los números reales como forma de representar las magnitudes. En binario estas pueden ser representadas de formas:

- Representación en Punto Fijo
- Representación en Punto Flotante

2.B.9.1. Representación en Punto Fijo⁶

Un ejemplo de representación en punto fijo puede ser: la medición de una longitud (123,45 cm), el cálculo de un peso (45,23 gr), un valor monetario (\$ 580,50), una temperatura: (-3,5°C).

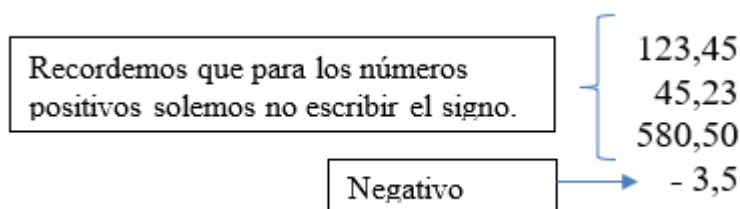


Figura 2.B.27. Ejemplo de magnitudes representados en punto fijo

Si quisiéramos unificar todas estas mediciones con la misma cantidad de decimales, entonces tendríamos: el signo, 3 dígitos decimales para la parte entera y 2 dígitos decimales. Observemos en la tabla 2.B.12 que se utilizan 3 campos.

Tabla 2.B.12. Representación de números reales

	Signo	Parte Entera	Parte Decimal
Medición	Positivo (+)	123,	45
Peso	Positivo (+)	045,	23
Importe	Positivo (+)	580,	50
Temperatura	Negativo (-)	003,	50

⁶ Colaboración para esta parte de la guía de Lic. Claudia Gabriela Alderete.

(Se podría tener solo **2 campos** para la representación, en ese caso, no tendríamos números signados, **Solo serían positivos**).

En binario sucede lo mismo. Se pueden representar números sin signo (solo positivos) o números con signo (positivos o negativos).

En el caso de los números signados se utiliza la convención de 0 para representar los números positivos y 1 para los negativos.

Considerando un Formato Numérico Digital llamado: Q (Q s,n,m). En este caso:

s: representa el signo.

n: la cantidad de bits para la parte entera

m: la cantidad de bits para representar la parte fraccionaria.

Cuando se quiere representar números en binario en una computadora, se tendrá una determinada cantidad de bits totales, por ejemplo 16 bits (ver tabla 2.B.13).

Tabla 2.B.13. Representación de números reales

Formato		Signo	Parte entera	Parte Fracc.	Total de Bits.
Q 1.0.15	Con signo	1	0	15	16 bits
Q 1.7.8	Con signo	1	7	8	16 bits
Q 0.0.16	Sin signo	0	0	16	16 bits
Q 0.0.16	Sin signo	0	0	16	16 bits
Q 0.1.15	Sin signo	0	1	15	16 bits

Ejercicio: En una arquitectura con un total de 16 bits, realizar la suma de dos números: $+24,4_{10}$ y $-16,2_{10}$ en formato **Q 1.8.3** y **Complemento a 2** para los negativos.

1. Paso a binario.

$+24,125_{10}$ $+11000,001_2$
 $-16,5_{10}$ Como el número es negativo, primero lo paso a positivo, entonces ...
 $+16,5_{10}$ $+10000,1_2$

2. Paso ambos números al formato especificado:

Q 1.8.3: 1 bit para signo, 8 bits para la parte entera y 3 bits para la parte fraccionaria.

Si es necesario se agregan ceros.

Si la parte fraccionaria excede el número de lugares, se trunca (se corta).

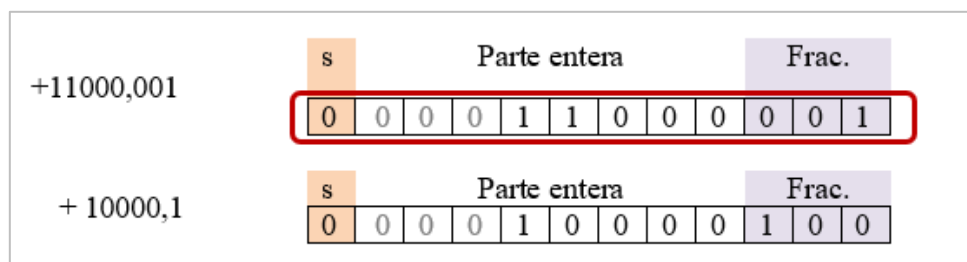


Figura 2.B.28. Pasar al formato especificado

Nótese que al pasar al formato NO SE ESCRIBE LA COMA.

Porque en la computadora “la coma” no se guarda.

3. Si hay un número negativo lo complemento (como en nuestro ejemplo).

$+16,5_{10}$	s	Parte entera								Frac.		
	0	0	0	0	1	0	0	0	0	1	0	0
$-16,5_{10}$	s	Parte entera								Frac.		
	1	1	1	1	0	1	1	1	1	1	0	0

Complemento a 2

Figura 2.B.29. Complemento SOLO en los números negativos.

4. Realizo la suma.

	1	1	1	1	1							
$+24,125_{10}$	0	0	0	0	1	1	0	0	0	0	0	1
$-16,500_{10}$	1	1	1	1	0	1	1	1	1	1	0	0
$7,625_{10}$	0	0	0	0	0	0	1	1	1	1	0	1

Figura 2.B.30. Suma en binario

5. Interpreto el número.

Restablezco el lugar que tenía la coma.

Sabiendo que el primer bit el más significativo, es para el signo.

Los 8 siguientes para la parte entera.

Los últimos 3 bits (los menos significativos, los de la derecha) corresponden a la parte fraccionaria.

Si es negativo, volvería a hacer el complemento.

Como el resultado obtenido en el bit de signo tiene 0, significa que es positivo. Entonces solo hago el pasaje de binario a decimal.

En binario sería así:

+ 0 0 0 0 0 1 1 1, 1 0 1₂

Resultado en Decimal:

+ 7, 625₁₀

Figura 2.B.31. Resultado en binario y en decimal

2.B.9.2. Representación exponencial. Punto Flotante

La representación de números en formato de punto fijo ubica la coma decimal en una posición establecida en la representación del número, dejando una cantidad determinada de dígitos tanto para la parte entera como para la fraccionaria. Cuando se requiere almacenar números fraccionarios o números enteros, muy grandes (por ejemplo millón de millones) o fracciones muy pequeñas, con muchos ceros después de la coma decimal y antes del primer dígito no nulo, se necesita contar con palabras de muchos bits, o sea se requiere una buena cantidad de hardware, lo cual implicaría que las operaciones de cálculo pueden llegar a realizarse de manera más lenta.

La representación de números en punto flotante, utilizada para números reales, (aunque también sirve para números enteros) permite representar un amplio rango de valores con poca cantidad de dígitos binarios. Consiste en la representación exponencial de los números reales.

Un valor numérico N , en una base B cualquiera, puede representarse como se indica en la expresión 2.B.4

$$N = m \times B^e$$

N : Valor numérico a representar

m : mantisa del número

B : base del sistema de numeración en que está expresado N .

e : exponente necesario para llegar al número N

Expresión 2.B.32. Representación exponencial de los números reales

En la tabla 2.B.11 se presentan algunos ejemplos:

Tabla 2.B.11. Ejemplos de representación exponencial de números reales

Número a representar	Base	Mantisa	Exponente	Representación exponencial
+52487,25	10	+0,05248725	6	$0,05248725 \times 10^6$
+52487,25	10	+0,5248725	5	$0,5248725 \times 10^5$
+52487,25	10	+5,248725	4	$5,248725 \times 10^4$
+52487,25	10	+52,48725	3	$52,48725 \times 10^3$
-0,00000000000000028	10	-28	-16	-28×10^{-16}
+1000000000000000	2	+1	+12	$1 \times 10^{+1100}$
-0,0000000000000001	2	-1	-14	-1×10^{-1110}

Debe quedar claro que el signo del número se observa en el signo de la mantisa y no en el signo del exponente que siempre indica si el número es menor o mayor a la unidad.

Esta forma de representar números permite expresar valores que pueden ser muy grandes o muy pequeños de una manera simple y además realizar operaciones aritméticas fácilmente con solo observar algunas reglas para operar en este formato:

- Para sumar o restar números en formato de punto flotante los números deben tener el mismo exponente, procediendo luego a sumar o restar las mantisas.

- Para multiplicar números en formato de punto flotante, el resultado tiene como exponente, la suma de los exponentes de los operandos y como mantisa el producto de sus mantisas.
- Para dividir números en formato de punto flotante, el resultado tiene como exponente, la resta de los exponentes de los operandos y como mantisa la que se obtiene al dividir las mantisas entre sí.
- Tanto en el caso de la multiplicación como en la división, no es necesario que los operandos tengan igual exponentes.

Esta representación presenta el problema que un mismo número puede tener distintas representaciones. Por ejemplo, las siguientes formas numéricas son todas equivalentes:

$$2593,2 \times 10^0 = 259,32 \times 10^1 = 25,9320 \times 10^2 = 2,5932 \times 10^3 = 0,25932 \times 10^4$$

2.B.9.3. Normalización de la mantisa

Para evitar el uso de representaciones múltiples para un mismo número, la representación de punto flotante trabaja con **formas normalizadas de la mantisa** siendo las frecuentemente usadas:

- La coma decimal ubicada a la izquierda del dígito no nulo más significativo, o sea la parte entera de la mantisa nula $0,1 \leq m < 1$. En el ejemplo planteado anteriormente correspondería a la expresión $0,25932 \times 10^4$.
- La coma decimal ubicada a la derecha del dígito no nulo más significativo, o sea la parte entera de la mantisa de un solo dígito $1 \leq m < 10$. En el ejemplo planteado anteriormente correspondería a la expresión $2,5932 \times 10^3$.

Si al realizar cualquiera de las operaciones anteriores, se obtiene un resultado con mantisa no normalizada, la misma deberá ser normalizada, multiplicando y dividiendo el resultado por una potencia de la base que permita la normalización.

2.B.9.4. Bit implícito u oculto

Si la representación en punto flotante se realiza en el sistema de numeración binario y la normalización de la mantisa utilizada es con la coma a la izquierda del dígito no nulo más significativo (en binario 1), luego de la coma siempre se tendrá un 1, por lo tanto la mayoría de los métodos no almacenan dicho bit, lo “recortan” antes de empaquetar el número para almacenarlo y lo incorporan cuando lo desempaquetan para volver a su representación en mantisa y exponente.

Este bit oculto, no guardado, se denomina **bit implícito u oculto**. Al utilizar esta forma se logra almacenar un bit adicional a la derecha de la mantisa, mejorando la precisión. Por ejemplo: si en un formato determinado la mantisa ya normalizada se representa, como 0,11011 la mantisa a ser almacenada será 11011 y el bit más significativo no se guarda, se sobreentiende.

Si en cambio la normalización utilizada es con coma a la derecha del primer bit no nulo, más significativo, lo que se tiene siempre es un 1 antes de la coma, y este sería el bit a

ocultar. Si como resultado de la normalización se tuviera una mantisa 1,101110 el patrón almacenado sería 101110.

En cualquiera de las dos formas normalizadas, en el caso de no ocultar el bit que es siempre 1, se dice que se trabaja con **primer bit no implícito o explícito**.

2.B.9.5. Representación del exponente

El exponente podría ser representado a través de su complemento a dos, lo cual implica la utilización de un bit para el signo, además de los inconvenientes enunciados con anterioridad en este tipo de representación. Para salvar estos problemas el método que se utiliza para almacenar el exponente es la **representación en exceso**.

Representación en exceso N de un valor numérico $X = X + \text{exceso } N$

siendo el exceso N (para una representación que utiliza n bits) igual a 2^{n-1} o una unidad menor, $2^{n-1} - 1$.

Expresión 2.B.33. Representación exponencial de los números reales

La representación definida en la expresión 2.B.5 convierte a todos los exponentes en números positivos. A continuación, en la tabla 2.B.12 se ejemplifica para $n = 8$ bits.

Tabla 2.B.12. Representación en exceso

Valor a representar	Exceso	Valor excedido	Representación en exceso
$+20_{10}$	$2^{n-1} = 2^{8-1} = 2^7 = 128$	$+20 + 128 = 148$	1 0 0 1 0 1 0 0
-20_{10}	$2^{n-1} = 2^{8-1} = 2^7 = 128$	$-20 + 128 = 108$	0 1 1 0 1 1 0 0
$+93_{10}$	$2^{n-1} - 1 = 2^{8-1} - 1 = 2^7 - 1 = 127$	$+93 + 127 = 220$	1 1 0 1 1 1 0 0
-93_{10}	$2^{n-1} - 1 = 2^{8-1} - 1 = 2^7 - 1 = 127$	$-93 + 127 = 34$	0 1 0 1 1 1 1 0

La tabla 2.B.13 muestra todos los valores posibles a representar, con $n = 3$ bits, en exceso 4 (exceso $2^{n-1} = 2^2$).

Tabla 2.B.13. Representación en exceso 4

Decimal	Representación
0	100
1	101
2	110
3	111
-4	000
-3	001
-2	010
-1	011

Esta representación es asimétrica y tiene una única representación para el 0, en ella el rango coincide con el de complemento a 2, salvo el bit más significativo (compararla con la tabla 2.B.5).

En la representación excedida los números aparecen ordenados si se los mira en una representación binaria sin signo, toda representación de un número negativo es menor que

cualquiera de un número positivo. Esto último tiene fundamental importancia al facilitar la comparación de los exponentes de los números en punto flotante para igualarlos, en caso de ser necesario para realizar su suma o resta.

Independientemente el formato de punto flotante que se utilice, debe ser conocido por toda persona que pretenda almacenar o recuperar datos que se hallan representados en ese formato.

2.B.9.6. Representación en punto flotante dentro de la computadora

Dado que existen distintas formas de representar números en formato de punto flotante, el Instituto de Ingenieros Electrónicos y Eléctricos (IEEE: Institute of Electrical and Electronics Engineers) es quién ha tomado el liderazgo en la normalización de formatos de punto flotante, formalizando hacia fines de la década de los sesenta la norma IEEE 754. (<http://www.ieee.org.ar>)

Esta norma considera dos formatos básicos: **simple precisión** y **doble precisión**.

En la figura 2.B.21 se muestran ambos formatos.

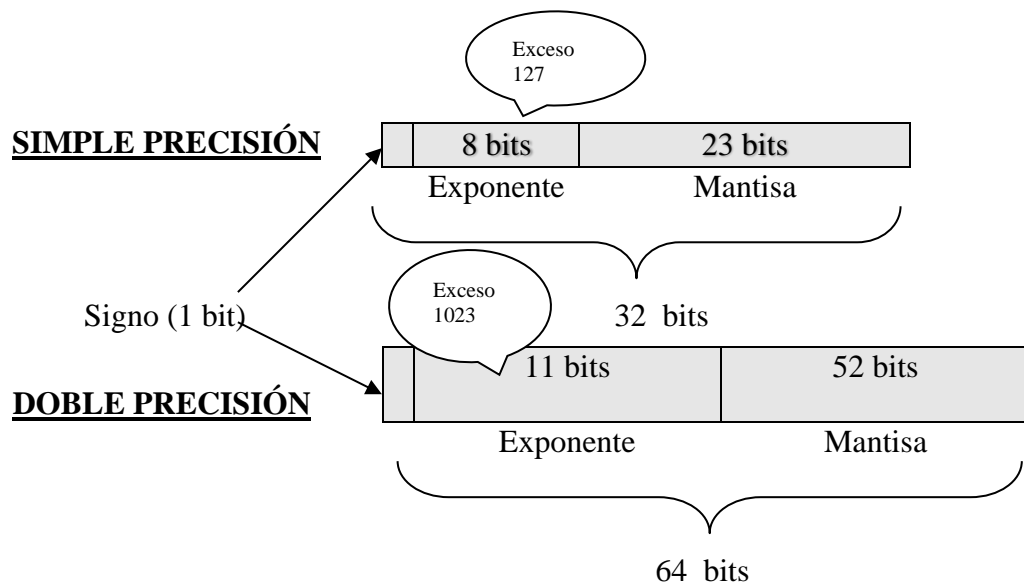


Figura 2.B.34. Formato IEEE 754 para punto flotante simple y doble precisión

SIMPLE PRECISIÓN

- **Total de bits:** 32
- **Signo:** 1er bit de la izquierda (0 positivo, 1 negativo)
- **Exponente:** 8 bits en exceso 127
- **Mantisa:** 23 bits, en binario, normalizada con la coma a la derecha del primer dígito significativo, con dicho bit implícito (oculto).

DOBLE PRECISIÓN

- **Total de bits:** 64
- **Signo:** 1er bit de la izquierda (0 positivo, 1 negativo)
- **Exponente:** 11 bits en exceso 1023
- **Mantisa:** 52 bits, en binario, normalizada con la coma a la derecha del primer dígito significativo, con dicho bit implícito (oculto).

Se presentará a continuación como sería el procedimiento para almacenar un número en punto flotante bajo la notación del IEEE 754 simple precisión.

Los bits que forman la palabra se empaquetarán de tal forma que el bit de signo (1 bit) quede a la izquierda, seguido por los bits que representan el exponente (8 bits) y a continuación los bits que representan la mantisa (23 bits). No se almacena ni la base del sistema, ni la coma fraccionaria.

S	EXPONENTE								MANTISA																						

Se quiere hallar la representación del número $-439,75_{10}$ utilizando la norma del IEEE simple precisión.

Paso 1 – PASAR A BINARIO:

Convertir el número a representar (en este caso el $-439,75_{10}$) al sistema de numeración binario utilizando los métodos de conversión estudiados anteriormente.

$$-439,75_{10} = -110110111,11_2$$

Paso 2 – CORRER LA COMA:

Ubicar la coma, a la derecha del bit más significativo (el 1 que se encuentra más a la izquierda), dejándolo expresado en formato de punto flotante:

$$-110110111,11_2 = -1,101101111 \times 10^8$$

Paso 3 – CALCULAR EXPONENTE NORMALIZADO:

El exponente es 8 (1000), pero como se almacena en exceso 127, el exponente a guardar saldrá de la siguiente operación:

$$\begin{array}{rcl} \text{Exponente:} & 1000 & (+8) \\ \text{Exceso 127:} & + \underline{111\ 1111} & (+127) \\ \text{Exponente en exceso 127:} & 1000\ 0111 & (+135) \end{array}$$

Paso 4 - CONTAR BITS DE MANTISA: Contar la cantidad de bits disponibles para la mantisa.

Mantisa: $1,101101111$, se almacenan 23 bits. Como la norma usa primer bit implícito, el 1 a la izquierda de la coma no se guarda $1,101101111$ (en este caso hay 10 bits a almacenar **se completarán los 13 restantes con ceros a la derecha de la parte fraccionaria** para que el número no varía, podría haber sucedido que existan más bits que los que se puedan almacenar en ese caso deberá truncarse)

Paso 5: EXPRESAR EL NUMERO NORMALIZADO

El valor a representar es negativo, por lo tanto **bit de signo (S) → 1**

A continuación del signo se vuelca el exponente normalizado (calculado en el paso 3)

Finalmente la mantisa calculada en el paso 4, en este caso se requirió **agregar ceros** para completar los 23 bits de la mantisa

S	EXPONENTE								MANTISA																							
1	1	0	0	0	0	1	1	1	1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Esta cadena de bits es el número $-439,75_{10}$ representado en el formato de punto flotante especificado.

2.B.9.7. Rango de la representación IEEE 754. Simple precisión.

Este formato permite expresar números en el rango de 2^{-127} al 2^{+128} , en decimal de 10^{-38} a 10^{+38}

- El valor más pequeño corresponde a la siguiente representación binaria:

1 00000001 000000000000000000000000

Signo= - Exponente= 1 Mantisa = $1/2^{23} = 2^{-23}$

Mínimo valor = $-(1 + 2^{-23}) \times 2^{1-127}$
 $= -(1 + 2^{-23}) \times 2^{-126}$, en decimal $-1,17549449 \cdot 10^{-38}$

- El valor máximo corresponde a la siguiente representación binaria

0 11111110 111111111111111111111111

Signo= + Exponente= 254 Mantisa = $1/2^1 + 1/2^2 + \dots + 1/2^{23} = 0,999999999$

Máximo valor = $(1 + 0,999999) \times 2^{254-127}$
 $= (1,999999) \times 2^{127}$, en decimal $3,40282346 \cdot 10^{38}$

Ejercicio 2.B.11. – Sugerido.

Si en IEEE simple precisión se ha almacenado una cadena de 32 bits todos ellos en cero, cual es el número en decimal almacenado.

Resolución: Dado que el signo es 0 se trata de un número positivo, tomando en consideración que el 1 está implícito deberá agregarse un 1 delante de la mantisa entonces el número será 1,00000.... El exponente deberá calcularse como el Exponente Normalizado menos el Exceso que se ha agregado para almacenarlo el cual es 127: Entonces el número es: $+1,0 \times 2^{-127}$

Notese que como siempre hay un 1 implícito no se puede guardar al cero, pero esta cadena de todos ceros está muy cercana a ser 0 dado que ese 1 implícito estará 126 lugares detrás de la coma.

MANTISA: Se quita el 1 que está implícito y se toma la parte restante: **11001** como es periódico se completa los bits restantes con la parte periódica 1001

S	EXPONENTE	MANTISA
0	1 0 0 0 0 0 0 0 0	1 0 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1

Ahora bien si analizamos el número que quedó almacenado para saber que número representa deberíamos hacer el proceso inverso:

SIGNO: 0 ENTONCES ES POSITIVO

EXPONENTE 128- EXCESO (127) = +1

MANTISA: AGREGAMOS 1, delante de los bits almacenados.

$$+ 1,10 \mathbf{1001 \ 1001 \ 1001 \ 1001 \ 1001 \ 10} \times 2^{+1}$$

Se corre la coma un lugar agrandando el número:

$$+ 11,0 \mathbf{1001 \ 1001 \ 1001 \ 1001 \ 1001 \ 10}$$

Ahora faltaría hacer el pasaje a base 10 para saber que numero representa, tomamos por un lado la parte entera $11_2 = 3_{10}$ y por otro lado se debe calcular la parte fraccionaria del número (se muestra el cálculo en la Figura 2.B.22). A lo que se le debe sumar la parte fraccionaria calculada mediante los pesos: 0,999984 dando por resultado **3,999984**

BITS	PESO
0	1/2
1	1/4
0	1/8
0	1/16
1	1/32
1	1/64
0	1/128
0	1/256
1	1/512
1	1/1024
0	1/2048
0	1/4096
1	1/8196
1	1/16384
0	1/32768
0	1/65536
1	1/153274
1	1/326570
0	1/653140
0	1/1306280
1	1/2612560
1	1/5225120
1	1/10450240
	0,2999984

Figura 2.B.35. Cálculo de la parte fraccionaria

Se almacenó el número **3,3₁₀** y el número recuperado fue **3,2999984** es importante entender porque se produce esta diferencia como efecto del truncamiento. De hecho, un programador que ha guardado el valor 3,3 en una variable tipo FLOAT (en notación de punto flotante) al compararla con el valor 3,3 en una CONSTANTE (escrito en el cuerpo del código) le resultado será que no es igual.