**TEMA 2**

**REPRESENTACION INTERNA DE LOS DATOS**

* 1. Sistemas de numeración
     1. Sistema binario
  2. Conversión entre sistemas
  3. Representación de números enteros
     1. Módulo y signo
     2. Complemento a 1
     3. Complemento a 2
     4. Suma en complemento a 1
     5. Suma en complemento a 2
     6. Codificación alfanumérica

**INTRODUCCION**



* 1. **SISTEMAS DE NUMERACION**

**Qué es un Sistema de Numeración?**

Conjunto de símbolos y reglas que se utilizan para la representación de datos numéricos o cantidades.

Se caracteriza por:

* **Base**. Es la cantidad de símbolos distintos que utiliza.

Base 10 : {0,1,2,3,4,5,6,7,8,9}

Base 2: {0,1}

Base 8: {0,1,2,3,4,5,6,7}

* **Posicional**. El valor de cada símbolo depende de su ubicación respecto de la coma.
* **Notación**. **Xb** , indica que el número o cantidad X se encuentra en base b.

Ejemplo:

56710 indica que el número 567 se encuentra en base 10.

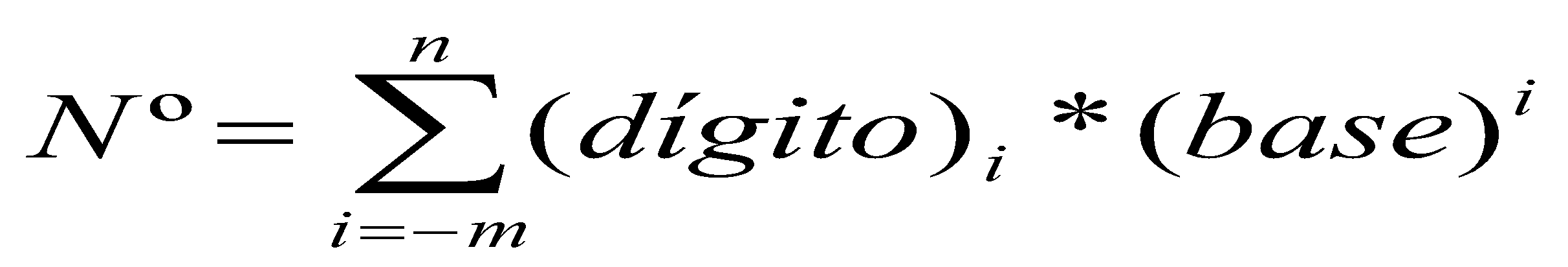
10112 indica que el número 1011 se encuentra en base 2.

1468 indica que el número 146 se encuentra en base 8.

**Teorema Fundamental de la Numeración (TFN)**

Permite convertir un valor expresado en cualquier Sistema de Numeración a uno en el Sistema Decimal.

Se encuentra expresado por la siguiente fórmula:

donde,

base: base original

i: posición respecto de la coma

n: número de dígitos a la derecha de la coma, menos 1

m: número de dígitos a la izquierda de la coma

dígito: cada uno de los componentes del número

Ejemplo:

111.23 = 2 \* 3-1 + 1 \*30 + 1 \* 31 + 1 \* 32 = **13.7**

85649 = 4 \* 90 + 6 \* 91 + 5 \* 92 + 8 \* 93 = **6295**

TENER PRESENTE

Los computadores efectuan operaciones

con números cuya precisión es finita y fija,

ya que la memoria disponible para almacenar un número

se fija en el momento del diseño.

**2.1.1 Sistema Binario**

En los circuitos electrónicos, desde el punto de vista lógico, se representa la presencia de tensión con un 1 (uno) y la ausencia con un 0 (cero).

Lógica positiva y Lógica negativa.

**Características**

Es el Sistema de Numeración utilizado internamente por los circuitos que configuran el hardware de los computadores.

* **Base: 2**
* **Dígitos: {0,1}** , denominados bits.

Para medir la cantidad de información representada en binario, se utilizan las siguientes medidas:

* Nibble o Cuarteto → 4 bits
* Byte u Octeto → 8 bits
* Kilobyte (KB) → 1024 bytes → 1024 \* 8 bits
* Megabyte (MB) → 1024 KB → 10242 \* 8 bits
* Gigabyte (GB) → 1024 MB → 10243 \* 8 bits
* Terabyte (TB) → 1024 GB → 10244 \* 8 bits

Se utiliza 1024 en lugar de 1000, ya que desde el punto de vista electrónico, es el múltiplo de 2 más cercano a 1000.

210 = 1024

El **Byte** es considerado como la unidad básica de medida de información representada en este sistema.

**Aritmética Binaria**

Para comenzar a familiarizarse con la aritmética binaria, inicialmente, sólo se considerarán operandos positivos.

* **Suma**

Dos números binarios se suman comenzando por el bit de la derecha y sumando los bits correspondientes a los dos operandos. Si se generó acarreo, éste se lleva un lugar a la izquierda, tal como en la suma decimal.

Para efectuar la suma, considerar la siguiente tabla:

|  |
| --- |
| Suma |
| 0 + 0 = 0  0 + 1 = 1  1 + 0 = 1  1 + 1 = 0 acarreo 1 |

Ejemplo:

0 1 1 1 0 0 1 1 115

1 0 0 0 0 0 0 0 128

1 1 1 1 0 0 1 1 243

**1**

1 1 0 0 0 1 1 1 199

0 0 1 1 0 1 0 0 52

1 1 1 1 1 0 1 1 251

**1 1 1**

0 1 1 1 1 0 1 1 123

0 1 1 0 0 0 1 0 98

1 0 0 0 0 0 0 0 128

1 0 1 0 1 1 1 0 1 349

* **Resta**

La resta se efectua realizando restas parciales entre los dígitos de igual posición.

Si el segundo dígito excede al primero, se sustrae una unidad del dígito siguiente a la izquierda en el minuendo:

* Si es 1, se convierte en 0, equivaliendo la unidad extraída a 1 \* 2 en el minuendo de la resta parcial que se está realizando.
* Si es 0, se busca la unidad en los sucesivos dígitos del minuendo, teniendo en cuenta que la unidad se multiplica por 2 a cada desplazamiento a la derecha

Para efectuar la resta, considerar la siguiente tabla:

|  |
| --- |
| Resta |
| 0 - 0 = 0  0 - 1 = 1 buscar unidad  1 - 0 = 1  1 - 1 = 0 |

Ejemplo:

1 0 1 1 1 0 1 1 187

1 0 1 0 0 0 0 1 161

0 0 0 1 1 0 1 0 26

0 2 0 2

1 0 1 1 1 0 1 1 187

0 1 0 0 0 1 1 1 71

0 1 1 1 0 1 0 0 116

2 2 2 2 2

0 -1-1 0 -1

1 1 0 0 1 0 0 1 201

1 0 0 0 1 0 1 0 138

0 0 1 1 1 1 1 1 63

* **Multiplicación**

La multiplicación binaria opera de la misma forma que la multiplicación decimal, pero en este caso la suma final de los productos parciales se realiza en binario.

Para efectuar la multiplicación, considerar la siguiente tabla:

|  |
| --- |
| Multiplicación |
| 0 \* 0 = 0  0 \* 1 = 0  1 \* 0 = 0  1 \* 1 = 1 |

Ejemplo:

1 1 0 1 1 1 0 1 \* 1 1 0 1 221 \* 13

1 1 0 1 1 1 0 1 2873

0 0 0 0 0 0 0 0

1 1 0 1 1 1 0 1

1 1 0 1 1 1 0 1

1 0 1 1 0 0 1 1 1 0 0 1

0 1 1 1 1 0 0 0 \* 1 0 0 120 \* 4

0 0 0 0 0 0 0 0 480

0 0 0 0 0 0 0 0

0 1 1 1 1 0 0 0

0 1 1 1 1 0 0 0 0 0

* **División**

La división binaria opera de la misma forma que la multiplicación decimal, pero en este caso las multiplicaciones y restas internas se realizan en binario.

Ejemplo:

1 0’ 1’ 0’ 1’ 0’ 1’ 1’ / 1 0 = 1 0 1 0 1 0 1

1 0

0 0 1 0 1 7’ 1’ / 2 = 8 5

1 0 1 1

0 0 1 0 1

1 0

0 0 1 1

1 0

0 1

1 1 0’ 1’ 1’ 1’ 1’ 0’ / 1 0 1 = 1 0 1 1 0 0

1 0 1

0 0 1 1 1 2 2’ 2’ / 5 = 4 4

1 0 1 2 2

0 1 0 1 2

1 0 1

0 0 0 1 0

1 1 0 1’ 1’ 1’ 1’ 0’ / 1 1 1 = 1 1 1 1 1

1 1 1

1 1 0 1 2 2’ 2’ / 7 = 3 1

1 1 1 1 2

1 1 0 1 5

1 1 1

1 1 0 1

1 1 1

1 1 0 0

1 1 1

1 0 1

**2.1.2 Otros sistemas**

* **Sistema Octal**
* **Base: 8**
* **Dígitos: {0, 1, 2, 3, 4, 5, 6, 7}**
* **Sistema Hexadecimal**
* **Base: 16**
* **Dígitos: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}**
  1. **CONVERSIÓN ENTRE SISTEMAS**

Se denomina conversión, entre números representados en diferentes sistemas de numeración, a la transformación de una cantidad desde un sistema a otro.

**2.2.1 Conversión Decimal-Binario**

* **Divisiones sucesivas por 2**

Método utilizado para convertir un número entero decimal a uno binario.

Consiste en dividir sucesivamente el número decimal y los sucesivos cuocientes por 2, hasta que el cuociente tome el valor 0.

La unión de todos los restos obtenidos, escritos en orden inverso, forma el número en binario.

Ejemplo:

|  |  |  |
| --- | --- | --- |
| 195 | 2 | 1 |
| 97  48  24  12  6  3  1  0 | 2  2  2  2  2  2  2 | 1  0  0  0  0  1  1 |
| 321 | 2 | 1 |
| 160  80  40  20  10  5  2  1  0 | 2  2  2  2  2  2  2  2 | 0  0  0  0  0  1  0  1 |

* **Multiplicaciones sucesivas por 2**

Método utilizado para convertir un número fraccionario en uno binario.

Consiste en multiplicar sucesivamente sólo la parte fraccionaria del número por 2.

Los dígitos binarios lo contituyen la parte entera del resultado de la multiplicación.

La unión de todos los enteros obtenidos, escritos en el orden natural, forma el número en binario resultante.

Repetir el proceso hasta que se presente alguno de los siguientes casos:

* La parte fraccionaria se haga 0.
* Se repita en forma periódica una cierta secuencia de dígitos.
* La cantidad de dígitos sea suficiente como para no exceder un cierto error.

Ejemplo:

|  |  |  |
| --- | --- | --- |
| 0.75 | 2 | 1.5 |
| 0.5  0.0 | 2 | 1.0 |

|  |  |  |
| --- | --- | --- |
| 0.45 | 2 | 0.90 |
| 0.90  0.80  0.60  0.20  0.40  0.80  0.60  0.20  0.40 | 2  2  2  2  2  2  2  2  2 | 1.80  1.60  1.20  0.40  0.80  1.60  1.20  0.40  0.80 |

|  |  |  |
| --- | --- | --- |
| 0.453 | 2 | 0.906 |
| 0.906  0.812  0.624  0.248  0.496  0.996  0.984  0.968  0.936  0.872  0.744  0.488 | 2  2  2  2  2  2  2  2  2  2  2  2 | 1.812  1.624  1.248  0.496  0.992  1.984  1.968  1.936  1.872  1.744  1.488  0.976 |

* **Restas sucesivas de las potencias de 2**

Método utilizado para convertir un número decimal con o sin parte fraccionaria.

Requiere el uso de una tabla de las potencias de 2 (positivas y negativas).

**(Tema de estudio para los alumnos)**

**2.2.2 Conversión Decimal-Octal**

* **Divisiones sucesivas por 8**

Método utilizado para convertir un número entero decimal en uno octal.

Consiste en dividir sucesivamente el número decimal y los sucesivos cuocientes por 8, hasta que el cuociente tome el valor 0.

La unión de todos los restos obtenidos, escritos en orden inverso, forma el número en octal.

Ejemplo:

|  |  |  |
| --- | --- | --- |
| 195 | 8 | 3 |
| 24  3  0 | 8  8 | 0  3 |

|  |  |  |
| --- | --- | --- |
| 321 | 8 | 1 |
| 40  5  0 | 8  8 | 0  5 |

* **Divisiones sucesivas por 8**

Método utilizado para convertir un número fraccionario en uno octal.

Consiste en multiplicar sucesivamente sólo la parte fraccionaria del número por 8.

Los dígitos octales corresponden a la parte entera del resultado de la multiplicación.

La unión de todos los enteros obtenidos, escritos en el orden natural, forma el número octal resultante.

Repetir el proceso hasta que se presente alguno de los siguientes casos:

* La parte fraccionaria se haga 0.
* Se repita en forma periódica una cierta secuencia de dígitos.
* La cantidad de dígitos sea suficiente como para no exceder un cierto error.

Ejemplo:

|  |  |  |
| --- | --- | --- |
| 0.75 | 8 | 6.00 |
| 0.00 |  |  |

|  |  |  |
| --- | --- | --- |
| 0.45 | 8 | 3.60 |
| 0.60  0.80  0.40  0.20  0.60  0.80  0.40  0.20 | 8  8  8  8  8  8  8  8 | 4.80  6.40  3.20  1.60  4.80  6.40  3.20  1.60 |

|  |  |  |
| --- | --- | --- |
| 0.453 | 8 | 3.624 |
| 0.624  0.992  0.936  0.488  0.904  0.232  0.856  0.848  0.784  0.272  0.176  0.408 | 8  8  8  8  8  8  8  8  8  8  8  8 | 4.992  7.936  7.488  3.904  7.232  1.856  6.848  6.784  6.272  2.176  1.408  3.264 |

* **Restas sucesivas de las potencias de 8**

Método utilizado para convertir un número decimal con o sin parte fraccionaria.

Requiere el uso de una tabla de las potencias de 8 (positivas y negativas).

**2.2.3 Conversión Decimal-Hexadecimal**

* **Divisiones sucesivas por 16**

Este método se aplica de forma similar a los casos de conversiones en binario y octal.

* **Multiplicaciones sucesivas por 16**

Este método se aplica de forma similar a los casos de conversiones en binario y octal.

**2.2.4 Conversión Binario-Decimal**

* **Conversión de enteros**

Escribir el número binario en forma vertical (dígito de la derecha debe quedar en la parte superior y el dígito de la izquierda en la parte inferior).

Repetir para cada uno de los dígitos:

* Sumar el dígito al producto de 2 por el resultado de la operación anterior (para el primer dígito el producto es 0).
* El resultado de la última operación corresponderá al valor decimal buscado.

Ejemplo:

1 1 1 0 1 1 1 02 = 23810

0 + 2 \* 119 = 238

1 + 2 \* 59 = 119

1 + 2 \* 29 = 59

1 + 2 \* 14 = 29

0 + 2 \* 7 = 14

1 + 2 \* 3 = 7

1 + 2 \* 1 = 3

1 + 2 \* 0 = 1

1 1 0 0 0 1 0 12 = 19710

1 + 2 \* 119 = 197

0 + 2 \* 59 = 98

1 + 2 \* 29 = 49

0 + 2 \* 14 = 24

0 + 2 \* 7 = 14

0 + 2 \* 3 = 6

1 + 2 \* 1 = 3

1 + 2 \* 0 = 1

* **Sumas de las potencias de 2**

Este método corresponde a la aplicación del Teorema Fundamental de la Numeración (TFN).

**2.2.4 Conversión Binario-Octal**

Este método consiste en agrupar los dígitos binarios de 3 en 3.

Desde el punto decimal hacia la izquierda para los enteros y hacia la derecha para los dígitos fraccionarios.

Buscar, para cada agrupación, el correspondiente dígito octal.

Ejemplo:

1 0 1 1 0 0 0 1 0 . 1 1 1 0 0 0

5 4 2 . 7 0

0 1 0 1 0 1 1 1 1 . 1 1 1 1 1 0

2 5 7 . 7 6

**2.2.4 Conversión Binario-Hexadecimal**

Este método consiste en agrupar los dígitos binarios de 4 en 4.

Desde el punto decimal hacia la izquierda para los enteros y hacia la derecha para los dígitos fraccionarios.

Buscar, para cada agrupación, el correspondiente dígito hexadecimal.

Ejemplo:

1 0 1 1 0 1 1 0 0 0 1 0 . 1 1 1 0 0 0 1 1

B 6 2 . E 3

0 0 0 1 0 0 1 0 1 1 1 1 . 1 1 1 1 1 0 0 0

1 2 F . F 8

**2.2.5 Conversión Octal-Decimal**

Aplicación del Teorema Fundamental de la Numeración (TFN).

**2.2.6 Conversión Hexadecimal- Decimal**

Aplicación del Teorema Fundamental de la Numeración (TFN).

**2.2.7 Conversión Base X a Base Y**

Este método de conversión se utiliza cuando no es factible aplicar directamente los métodos de conversión antes descritos.

Paso 1: Llevar el valor en **base x** **a** **base 10**, aplicando el TFN.

Paso 2: Llevar el valor en **base 10 a base y**, aplicando el método de las divisiones sucesivas por **y**.

Ejemplo:

4217 🡪 X3

Paso N°1: Aplicando TFN

4217 = 1 \* 70 + 2 \* 71 + 4 \* 72

= 1 + 14 + 196

= 21110

Paso N°2: Aplicando método de las divisiones sucesivas por 3.

|  |  |  |
| --- | --- | --- |
| 211 | 3 | 1 |
| 70  23  7  2  0 | 3  3  3  3 | 1  2  1  2 |

* 1. **REPRESENTACION DE NÚMEROS ENTEROS**

Los computadores utilizan cuatro métodos para la representación interna de los números enteros (positivos y negativos).

* Módulo y signo (Signo/magnitud)
* Complemento a 1 (C-1)
* Complemento a 2 (C-2)
* Exceso 2n-1

Para estas representaciones:

* Se utiliza el sistema binario.
* Se considera, como realmente ocurre, que se tiene un número limitado de bits para cada dato numérico, indicado por **n**.
* **Rango de Representación**, conjunto de números representables en un método.

**2.3.1 Módulo y signo**

Si se considera n bits:

* El bit que está más a la izquierada representa el signo, 0 (positivo) y 1 (negativo).
* Los n-1 bits restantes representan el módulo del número.

Ejemplo:

n = 8

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|  |  |  |  |  |  |  |  |
| **S** | **Magnitud** | | | | | | |

n = 16

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **15** | **14** | **13** | **.** | **.** | **2** | **1** | **0** |
|  |  |  |  |  |  |  |  |
| **S** | **Magnitud** | | | | | | |

**Rango de representación**

**- 2n-1 + 1 ≤ X ≤ 2n-1 - 1**

- 28-1 + 1 ≤ X ≤ 28-1 - 1

- 27 + 1 ≤ X ≤ 27 - 1

- 128 + 1 ≤ X ≤ 128 – 1

- 127 ≤ X ≤ 127

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
|  | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| **S** | **127** | | | | | | |

Ejemplo:

n=8

x=38

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **+** |  |  |  |  |  |  |  |
| **0** | **0** | 1 | 0 | 0 | 1 | 1 | 0 |
|  |  | | | | | | |

x=- 38

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **-** |  |  |  |  |  |  |  |
| **1** | **0** | 1 | 0 | 0 | 1 | 1 | 0 |
|  |  | | | | | | |

Ventaja: Posee rango simétrico. Igual cantidad de enteros positivos y negativos.

Desventaja: Posee 2 representaciones para el valor cero.

**2.3.2 Complemento a 1 (C-1)**

Si se considera n bits:

* El bit que está más a la izquierada representa el signo, 0 (positivo) y 1 (negativo).
* Los n-1 bits restantes representan el módulo del número.
* Números positivos, se consideran igual que en Signo/Magnitud.
* Números negativos, complementar todos los dígitos, incluido el signo.

**Rango de representación**

**- 2n-1 + 1 ≤ X ≤ 2n-1 - 1**

- 127 ≤ X ≤ 127

Ejemplo:

n=8

x=38

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **+** |  |  |  |  |  |  |  |
| **0** | 0 | 1 | 0 | 0 | 1 | 1 | 0 |

x=- 38

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **-** |  |  |  |  |  |  |  |
| **1** | 1 | 0 | 1 | 1 | 0 | 0 | 1 |

Ventaja: Posee rango simétrico. Igual cantidad de enteros positivos y negativos.

Desventaja: Posee 2 representaciones para el valor cero.

**2.3.3 Complemento a 2 (C-2)**

Si se considera n bits:

* El bit que está más a la izquierada representa el signo, 0 (positivo) y 1 (negativo).
* Los n-1 bits restantes representan el módulo del número.
* Números positivos, se consideran igual que en Signo/Magnitud.
* Números negativos, requiere 2 pasos:

1. Aplicar C-1
2. Al resultado obtenido en (1), se le suma 1 en binario, despreciando el último acarreo, si existe.

**Rango de representación:**

**- 2n-1 ≤ X ≤ 2n-1 - 1**

- 128 ≤ X ≤ 127

Ejemplo:

n=8

x=38

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **+** |  |  |  |  |  |  |  |
| **0** | 0 | 1 | 0 | 0 | 1 | 1 | 0 |

x=- 38

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **C-1** |  |  |  |  |  |  |  |
| **1** | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| **+** | | | | **1** | | | |
| **1** | 1 | 0 | 1 | 1 | 0 | 1 | 0 |

Ejemplo:

n=8

x=0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **+** |  |  |  |  |  |  |  |
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

x=- 0

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **C-1** |  |  |  |  |  |  |  |
| **1** | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **+** | | | | **1** | | | |
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Desventaja: Posee rango asimétrico.

Ventaja: Posee 1 representación para el valor cero.

**2.3.4 Suma en complemento a 1**

Los números se suman de igual forma que en binario.

El acarreo, producido en la suma parcial de los bits de más a la izquierda, se suma al resultado.

Ejemplos:

1. Sumar 2010 y 3510 en C-1 para n=8 bits.

C-1 2010 0 0 0 1 0 1 0 0

C-1 3510 0 0 1 0 0 0 1 1

C-1 5510 0 0 1 1 0 1 1 1

1. Sumar -2010 y 3510 en C-1 para n=8 bits.

C-1 -2010 1 1 1 0 1 0 1 1

C-1 3510 0 0 1 0 0 0 1 1

1 0 0 0 0 1 1 1 0

1

C-1 1510 0 0 0 0 1 1 1 1

1. Sumar 2010 y -3510 en C-1 para n=8 bits.

C-1 2010 0 0 0 1 0 1 0 0

C-1 -3510 1 1 0 1 1 1 0 0

C-1 -1510 1 1 1 1 0 0 0 0

1. Sumar 8010 y 6010 en C-1 para n=8 bits.



C-1 8010 0 1 0 1 0 0 0 0

C-1 6010 0 0 1 1 1 1 0 0

C-1 14010 1 0 0 0 1 1 0 0

* Los valores 80 y 60 son válidos para el rango de representación dado por n=8, sin embargo, el resultado de la suma no lo es.
* **El bit de signo del resultado es 1**, lo cual indica que la suma de dos valores positivos da como resultado un número negativo, por lo tanto **el resultado es incorrecto**.

1. Sumar -8010 y -6010 en C-1 para n=8 bits.



C-1 -8010 1 0 1 0 1 1 1 1

C-1 -6010 1 1 0 0 0 0 1 1

C-1 -14010 1 0 1 1 1 0 0 1 0

1 

0 1 1 1 0 0 1 1

* Los valores -80 y -60 son válidos para el rango de representación dado por n=8, sin embargo, el resultado de la suma no lo es.
* **El bit de signo del resultado es 0**, lo cual indica que la suma de dos valores negativos da como resultado un número positivo, por lo tanto **el resultado es incorrecto**.

**2.3.5 Suma en complemento a 2**

Los números se suman de igual forma que en binario.

El acarreo, producido en la suma parcial de los bits de más a la izquierda, se desprecia.

Ejemplos:

1. Sumar 2010 y 3510 en C-2 para n=8 bits.

C-2 2010 0 0 0 1 0 1 0 0

C-2 3510 0 0 1 0 0 0 1 1

C-2 5510 0 0 1 1 0 1 1 1

1. Sumar -2010 y 3510 en C-2 para n=8 bits.

C-1 -2010 1 1 1 0 1 0 1 1

+1 1

C-2 -2010 1 1 1 0 1 1 0 0

C-2 3510 0 0 1 0 0 0 1 1

C-2 1510 1 0 0 0 0 1 1 1 1

1. Sumar 2010 y -3510 en C-2 para n=8 bits.

C-1 -3510 1 1 0 1 1 1 0 0

+1 1

C-2 -3510 1 1 0 1 1 1 0 1

C-2 2010 0 0 0 1 0 1 0 0

C-2 -1510 1 1 1 1 0 0 0 1

**2.3.6 Codificación alfanumérica**

Los datos que se manejan internamente en el computador se pueden representar mediante códigos alfanuméricos.

El computador trabaja con un conjunto de caracteres que permiten manejar datos, instrucciones, órdenes de control, etc.

Este conjunto se subdivide en :

* Caracteres alfabéticos.
* Caracteres decimales
* Caracteres especiales

Caracteres: \*, @, ...

Cada caracter se maneja internamente por medio de un conjunto de 8 bits, mediante un sistema de codificación binario, denominado Código de Caracteres.

Cada computador tiene su código definido por el fabricante.

Por ejemplo: FIELDATA de 6 bits

ASCII de 7 bits

ASCII ext. de 8 bits

EBCDIC 8 bits