Aritmética Binaria

- Todas las operaciones aritméticas conocidas en el sistema de numeración decimal, pueden también realizarse en cualquier otro sistema de numeración, para ello se aplican las mismas reglas de la aritmética común.
- Estudiaremos las cuatro operaciones básicas: suma, resta, multiplicación y división, aplicadas al sistema de numeración binario.

Suma Binaria

- En la suma binaria se siguen los mismos pasos generales. No obstante, sólo pueden ocurrir cuatro casos cuando se suman los dos dígitos binarios (bits) en cualquier posición. Éstos son:

```
    0 + 0 = 0
    1 + 0 = 1
    1 + 1 = 10 = 0 + acarreo de 1 hacia la siguiente posición
1 + 1 + 1 = 11 = 1 + acarreo de 1 hacia la siguiente posición
```

- Lo último ocurre cuando los 2 bits son 1 y hay un acarreo de la posición anterior. He aquí varios ejemplos de la suma de dos números binarios (los equivalentes decimales están entre paréntesis):

```
  011 (3)        1001 (9)        11.011 (3.375)
+ 110 (6)      + 1111 (15)     + 10.110 (2.750)
---------      ----------      ---------------
 1001 (9)       11000 (24)      110.001 (6.125)
```

Suma Binaria

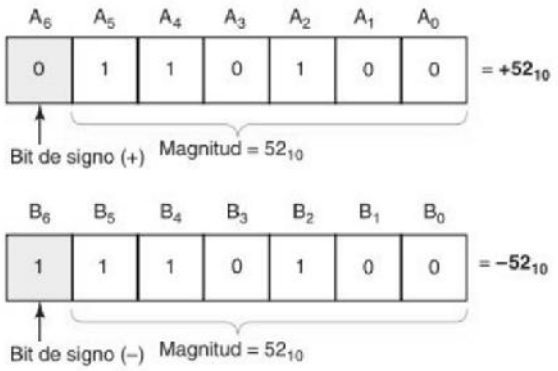
- No es necesario considerar la suma de más de dos números binarios a la vez, ya que en todos los sistemas digitales los circuitos que se encargan de realizar la suma sólo pueden manejar dos números al mismo tiempo. Cuando se van a sumar más de dos números, los primeros dos se suman y luego el resultado se suma al tercer número, y así sucesivamente.
- Ésta no es una desventaja grave, ya que por lo general las computadoras digitales modernas pueden realizar una operación de suma en varios nanosegundos.
- La suma es la operación aritmética más importante en los sistemas digitales.

Representación de números con signo

- En las computadoras digitales, los números binarios se representan mediante un conjunto de dispositivos de almacenamiento binario (por ejemplo, flip-flops). Cada dispositivo representa un bit. Por ejemplo, un registro FF de seis bits puede almacenar números binarios que varían desde 000000 hasta 111111 (de O a 63 en decimal).
- Esto representa la *magnitud del número. Debido a que la mayoría de las computa*doras y calculadoras digitales maneja números tanto negativos como positivos, se requiere de algún medio para representar el *signo del número (+ o -). Por lo general* lo que se hace es agregar otro bit al número; a este bit se le llama **bit de signo.**
- En general, la convención común es que un 0 en el signo representa a un número positivo y un 1 en el bit de signo representa a un número negativo. El registro *A contiene los bits 0110100. El 0 en el bit más a la izquierda (A6) es el bit de signo que representa al signo +. Los otros seis bits son la magni*tud del número 110100_2, el cual es equivalente al 52 decimal.
- Entonces, el número almacenado en el registro *A es +52. De manera similar, el número almacenado en el* registro *B es -52 ya que el bit de signo es 1, que representa al -.*

Representación de números con signo

FIGURA Representación de números con signo en la forma de signo-magnitud.



Representación de números con signo

- El bit de signo se utiliza para indicar la naturaleza positiva o negativa del número binario almacenado. Los números de la figura consisten de un bit de signo y de seis bits de magnitud. Los bits de magnitud son el equivalente binario real del valor decimal que se está representando. A éste se le conoce como sistema de signo-magnitud para representar números binarios con signo.
- Aunque el sistema de signo-magnitud es simple, las calculadoras y computadoras no lo utilizan de manera usual porque la implementación del circuito es más compleja que en otros sistemas. El sistema más común que se utiliza para representar números binarios con signo es el sistema de complemento a 2. Antes de ver cómo se hace esto, primero debemos ver cómo se forma el complemento a 1 y el complemento a 2 de un número binario.

Representación de números con signo

- **Complemento de 1**
- El complemento a 1 de un número binario se obtiene al cambiar cada 0 por un 1 y cada 1 por un 0. En otras palabras, se cambia cada uno de los bits en el número por su complemento. El proceso se muestra a continuación.

```
1 0 1 1 0 1  número binario original
↓ ↓ ↓ ↓ ↓ ↓
0 1 0 0 1 0  se complementa cada bit para formar el complemento a 1
```

- Por lo tanto, decimos que el complemento de 1 de 101101 es 010010.

Representación de números con signo

- **Complemento de 2**
- El complemento de 2 de un número binario se forma al tomar el complemento de 1 de ese número y sumarle 1 a la posición del bit menos significativo. El proceso se ilustra a continuación para el número $101101_2 = 45_{10}$

```
  1 0 1 1 0 1    equivalente binario de 45
  0 1 0 0 1 0    se complementa cada bit para formar el complemento a 1
+         1      se le suma 1 para formar el complemento a 2
  0 1 0 0 1 1    complemento a 2 del número binario original
```

- Entonces, decimos que 010011 es la representación en complemento de 2 de 101101.

Representación de números con signo

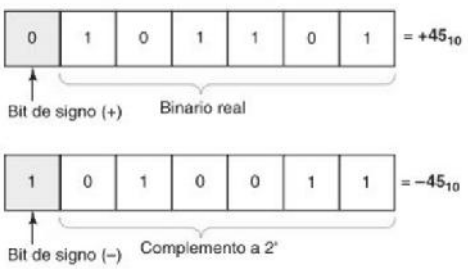
- Ejemplo representar el número 101100 en complemento de 2.

	101100	número binario original
	010011	complemento a 1
+	1	se le suma 1
	010100	complemento a 2 del número original

Representación de números con signo mediante complemento de 2

- El sistema de complemento a 2 para representar números con signo funciona así:
  - Si el número es positivo, la magnitud se representa en su forma binaria real y se coloca el bit de signo 0 enfrente del MSB. Esto se muestra en la figura para el número $+45_{10}$.
  - Si el número es negativo, la magnitud se representa en su forma de complemento a 2 y se coloca el bit de signo 1 enfrente del MSB. Esto se muestra en la figura para el número -45_{10}.

FIGURA Representación de números con signo en el sistema de complemento a 2.



Representación de números con signo mediante complemento de 2

- El sistema de complemento a 2 se utiliza para representar números con signo ya que, como veremos, nos permite realizar la operación de la resta, a partir de una operación de suma.
- Esto es importante ya que significa que una computadora digital puede utilizar los mismos circuitos tanto para sumar como para restar, lo cual redunda en un ahorro en el hardware.
- Represente cada uno de los siguientes números decimales con signo en forma de un número binario con signo en el sistema de complemento a 2. Use un total de cinco bits, incluyendo el bit de signo.

Representación de números con signo mediante complemento de 2

Solución

(a) El número es positivo, por lo que la magnitud (13) se representará en su forma de magnitud real, es decir: $13 = 1101_2$. Si agregamos el bit de signo 0 tenemos que

$$+13 = 01101$$

bit de signo ↑

(b) El número es negativo, por lo que la magnitud (9) deberá representarse en forma de complemento a 2:

$$\begin{array}{rll} 9_{10} = 1001_2 & & \\ 0110 & & \text{(complemento a 1)} \\ + \quad 1 & & \text{(se suma 1 al LSB)} \\ \hline 0111 & & \text{(complemento a 2)} \end{array}$$

Cuando agregamos el bit de signo 1, el número con signo completo se convierte en

$$-9 = 10111$$

bit de signo ↑

Representación de números con signo mediante complemento de 2

El procedimiento que acabamos de seguir requiere de dos pasos. Primero determinamos el complemento a 2 de la magnitud y después agregamos el bit de signo. Esto puede lograrse en un solo paso si incluimos el bit de signo en el proceso del complemento a 2. Por ejemplo, para encontrar la representación para −9, empezamos con la representación para +9, *incluyendo el bit de signo*, y tomamos el complemento a 2 de ese número para poder obtener la representación para −9.

```
+9 = 01001
     10110     (complemento a 1 de cada bit, incluyendo el bit de signo)
   +     1     (se suma 1 al LSB)
−9 = 10111     (representación de −9 en complemento a 2)
```

Representación de números con signo mediante complemento de 2

(c) El valor decimal 3 puede representarse en binario con sólo usar dos bits. No obstante, la declaración del problema requiere una magnitud de cuatro bits precedida por un bit de signo. Por lo tanto, tenemos:

$$+3_{10} = 0011$$

En muchas situaciones el número de bits se fija con base en el tamaño del registro que almacenará los números binarios, por lo que tal vez debamos agregar 0s para poder llenar el número requerido de posiciones de bits.

```
+2 = 00010
     11101     (complemento a 1)
   +     1     (se suma 1)
−2 = 11110     (representación de −2 en complemento a 2)
```

(d) Empezamos por escribir +2 mediante el uso de cinco bits:

(e) Empezamos con +8:

```
+8 = 01000
     10111     (se complementa cada bit)
   +     1     (se suma 1)
−8 = 11000     (representación de −8 en complemento a 2)
```

Extensión de signo

- En el ejemplo anterior tuvimos que utilizar un total de cinco bits para representar los números con signo.
- El tamaño de un registro *(número de flip-flops) determina* el número de dígitos binarios que se almacenan para cada número. La mayoría de los sistemas digitales hoy en día almacenan números en registros con tamaños en múltiplos pares de cuatro bits.
- En otras palabras, los registros de almacenamiento están compuestos de 4, 8, 16, 32 o 64 bits. En un sistema que almacena números de ocho bits, siete de ellos representan la magnitud y el MSB representa el signo.
- Si necesitarnos almacenar un número positivo de cinco bits en un registro de ocho bits, sólo basta con agregar ceros a la izquierda. El MSB (bit de signo) sigue siendo 0, lo cual indica un número positivo.

0000 1001

0s a la izquierda que se agregaron — valor binario para el 9

Extensión de signo

¿Qué ocurre si tratamos de almacenar números negativos de cinco bits en un registro de ocho bits? En la sección anterior vimos que la representación binaria en

1 0111

complemento a 2 de cinco bits para el −9 es 10111.
Si anexáramos 0s a la izquierda, éste ya no sería un número negativo en el formato de ocho bits. La manera apropiada de extender un número negativo es anexando 1s a la izquierda. Así, el valor almacenado para el 9 negativo es

111 1 0111

- magnitud en complemento a 2
- signo en formato de cinco bits
- extensión del signo al formato de ocho bits

Negación

- La negación es la operación de convertir un número positivo en su equivalente negativo, o un número negativo en su equivalente positivo.
- Cuando los números binarios con signo se representan en el sistema de complemento a 2, para realizar la negación sólo basta con realizar la operación de complemento a 2.
- Para ilustrar esto empecemos con el +9 en su forma binaria de ocho bits. Su representación con signo es 00001001.
- Si sacamos su complemento a 2 obtendremos 11110111, el cual representa al valor con signo -9.
- De igual forma podemos empezar con la representación de -9, que es 11110111, y sacar su complemento a 2 para obtener 00001001, el cual representa a +9.

Negación

Empezamos con	00001001	+9
complemento a 2 (se cambia el signo)	11110111	−9
se niega otra vez	00001001	+9

Por lo tanto, para cambiar el signo de un número binario con signo lo complementamos a 2.

Esta negación cambia el número a su equivalente con signo opuesto.

Ejercicio: Cada uno de los siguientes números es binario con signo de cinco bits en el sistema de complemento a 2. Determine el valor decimal en cada caso:

(a) 01100 (b) 11010 (c) 10001

Negación

Solución

(a) El bit de signo es 0, por lo que el número es *positivo* y los otros cuatro bits representan la magnitud real del número. Esto es, $1100_2 = 12_{10}$. Por lo tanto, el número decimal es +12.

(b) El bit de signo de 11010 es un 1, por lo que sabemos que el número es negativo, pero no podemos saber cuál es la magnitud. Podemos encontrarla si cambiamos el signo (complementamos a 2) al número para convertirlo en su equivalente positivo.

	11010	(número negativo original)
	00101	(complemento a 1)
+	1	(se suma 1)
	110	(+6)

Como el resultado de la negación es 0010 = +6, el número original 11010 debe ser equivalente a −6.

(c) Siga el mismo procedimiento que en (b):

	10001	(número negativo original)
	01110	(complemento a 1)
+	1	(se suma 1)
	1111	(+15)

Por lo tanto, 10001 = −15.

Caso especial en la representación de complemento de 2

- Siempre que un número con signo tiene un 1 en el bit de signo y todos los bits de su magnitud son 0, su equivalente decimal es -2^N en donde N es el número de bits en la magnitud. Por ejemplo,

$$1000 = -2^3 = -8$$
$$10000 = -2^4 = -16$$
$$100000 = -2^5 = -32$$

- y así en lo sucesivo. Observe que en este caso especial, al sacar el complemento a 2 de estos números se produce el valor con el que comenzamos, ya que estamos en el límite negativo del intervalo de números que pueden representarse mediante esta cantidad de bits. Si extendemos el signo de estos números especiales, el procedimiento normal de negación funciona sin problemas. Por ejemplo, al extender el número 1000 (-8) a 11000 (8 negativo de cinco bits) y sacar su complemento a 2 obtenemos 01000 (8), que es la magnitud del número negativo.

Caso especial en la representación de complemento de 2

- Así, podemos afirmar que el intervalo completo de valores que pueden representarse en el sistema de complemento a 2 con N bits de magnitud es

$$-2^N \text{ to } +(2^N - 1)$$

- Hay un total de 2N+1 valores distintos incluyendo el cero.

Valor decimal	Binario con signo mediante el uso del complemento a 2
$+7 = 2^3 - 1$	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
0	0000
−1	1111
−2	1110
−3	1101
−4	1100
−5	1011
−6	1010
−7	1001
$-8 = -2^3$	1000

Caso especial en la representación de complemento de 2

¿Cuál es el intervalo de valores decimales *sin signo* que pueden representarse en un byte?

Solución

Recuerde que un byte es de ocho bits. Aquí nos interesan los números sin signo, por lo que no hay bit de signo y todos los ocho bits se utilizan para la magnitud. Por lo tanto, los valores variarán desde

$$00000000_2 = 0_{10}$$

hasta

$$11111111_2 = 255_{10}$$

Esto nos da un total de 256 valores distintos, los cuales podríamos haber previsto, ya que $2^8 = 256$.

Caso especial en la representación de complemento de 2

¿Cuál es el intervalo de valores decimales *con signo* que pueden representarse en un byte?

Solución

Como el MSB se va a utilizar como el bit de signo, se tienen siete bits para la magnitud. El valor negativo más grande es

$$10000000_2 = -2^7 = -128_{10}$$

El valor positivo más grande es

$$01111111_2 = +2^7 - 1 = +127_{10}$$

En consecuencia, el intervalo es de −128 a +127; esto nos da un total de 256 valores distintos, incluyendo el cero. De manera alternativa, y como hay siete bits de magnitud ($N = 7$), entonces hay $2^{N+1} = 2^8 = 256$ valores distintos.

Caso especial en la representación de complemento de 2

Cierta computadora almacena los siguientes dos números con signo en su memoria, utilizando el sistema de complemento a 2:

$$00011111_2 = +31_{10}$$
$$11110100_2 = -12_{10}$$

Mientras ejecuta un programa, a la computadora se le pide que convierta cada número en su signo opuesto; esto es, debe cambiar el +31 a −31 y el −12 a +12. ¿Cómo hará esto?

Solución

Ésta es la operación de cambio de signo, en la que un número con signo puede cambiar su polaridad con sólo realizar la operación de complemento a 2 sobre el número *completo*, incluyendo el bit de signo. Los circuitos de la computadora tomarán el número con signo de la memoria, encontrarán su complemento a 2 y pondrán el resultado de vuelta en la memoria.

Ejercicios

1. Represente cada uno de los siguientes valores como un número con signo de ocho bits en el sistema de complemento a 2.
   (a) +13 (b) − 7 (c) −128
2. Cada uno de los siguientes números es binario con signo en el sistema de complemento a 2. Determine el equivalente decimal para cada uno de ellos.
   (a) 100011 (b) 1000000 (c) 01111110
3. ¿Qué intervalo de valores decimales con signo pueden representarse en 12 bits (incluyendo el bit de signo)?
4. ¿Cuántos bits se requieren para representar valores decimales que varían desde −50 hasta +50?
5. ¿Cuál es el valor decimal negativo más grande que puede representarse mediante un número de dos bytes?
6. Realice la operación de complemento a 2 en cada uno de los siguientes números:
   (a) 1000 (b) 10000000 (c) 1000
7. Defina la operación de negación.

Suma en Complemento de 2

- Cómo se realizan las operaciones de suma y resta en los equipos digitales que utilizan la representación de complemento a 2 para los números negativos ?.
- En los diversos casos a considerar, es importante observar que el bit de signo de cada número se opera sobre la misma forma que los bits de magnitud.

Operaciones en Complemento de 2

Caso I: dos números positivos. La suma de dos números positivos es simple. Considere la suma de +9 y +4:

```
+9 → 0 1001    (primer sumando)
+4 → 0 0100    (sumando)
     0 1101    (suma = +13)
     ↑
     └ bits de signo
```

Observe que los bits de signo del **primer sumando** y del **sumando** son ambos 0 y que el bit de signo de la suma es 0, lo cual indica que la suma es positiva. Observe, además, que el primer sumando y el sumando se ajustan para tener el mismo número de bits. Esto debe hacerse *siempre* en el sistema de complemento a 2.

Operaciones en Complemento de 2

Caso II: número positivo y número negativo más pequeño. Considere la suma de +9 y −4. Recuerde que el −4 estará en su forma de complemento a 2. Por lo tanto, +4 (00100) debe convertirse en −4 (11100).

```
     ┌ bits de signo
     ↓
+9 → 0 1001    (primer sumando)
−4 → 1 1100    (sumando)
   1 0 0101
     ↑
     └ Este acarreo se descarta; el resultado es 00101 (suma = +5).
```

En este caso el bit de signo del sumando es 1. Observe que los bits de signo también participan en el proceso de la suma. De hecho, se genera un acarreo en la última posición de la suma. *Este acarreo siempre se descarta*, por lo que la suma final es 00101, que equivale a +5.

Operaciones en Complemento de 2

Caso III: número positivo y número negativo más grande. Considere la suma de −9 y +4:

```
−9 →   10111
+4 →   00100
       11011      (suma = –5)
       ↑
       └── bit de signo negativo
```

Aquí la suma tiene un bit de signo en 1, lo cual indica un resultado negativo. Como el resultado de la suma es negativo se encuentra en su forma de complemento a 2, por lo que los últimos cuatro bits (1011) en realidad representan el complemento a 2 del resultado. Para encontrar la magnitud real, debemos negar (complementar a 2) el 11011; el resultado es 00101 = +5. De esta forma, 11011 representa a −5.

Operaciones en Complemento de 2

Caso IV: dos números negativos

```
−9 →   10111
−4 →   11100
     1 10011
     ↑ └─ bit de signo
     └── Este acarreo se descarta; el resultado es 10011 (suma = –13).
```

Nuevamente el resultado final es negativo y está en forma de complemento a 2 con un bit de signo de 1. Al cambiar de signo (complementar a 2) este resultado se produce 01101 = +13.

Operaciones en Complemento de 2

Caso V: números iguales y opuestos

$$\begin{array}{rl} -9 \rightarrow & 10111 \\ +9 \rightarrow & 01001 \\ \hline 0 \quad 1 & 00000 \end{array}$$

Se descarta; el resultado es 0000 (suma = +0).

Es obvio que el resultado es +0, como se esperaba.

Resta en complemento de 2

- La operación de resta mediante el uso del sistema de complemento a 2 en realidad implica la operación de la suma y no es distinta de los diversos casos para la suma . Cuando se resta un número binario (el sustraendo) de otro número binario (el minuendo) se utiliza el siguiente procedimiento:
  - *Negar el sustraendo. Esto cambiará el sustraendo por su valor equivalente del signo opuesto.*
  - *Sumar éste al minuendo. El resultado de esta suma representará la diferencia entre el sustraendo y el minuendo.*
- Una vez más, como en todas las operaciones aritméticas de complemento a 2 es necesario que ambos números tengan la misma cantidad de bits en su representación.

Resta en complemento de 2

Consideremos el caso en donde +4 se va a restar de +9.

minuendo (+9) → 01001
sustraendo (+4) → 00100

Se cambia el signo al sustraendo para producir 11100, el cual representa a −4. Ahora se suma éste al minuendo.

```
  01001   (+9)
+ 11100   (−4)
1 00101   (+5)
↑
└ Se descarta, por lo que el resultado es 00101 = +5.
```

Cuando el sustraendo se cambia por su complemento a 2, en realidad se convierte en −4 por lo que estamos *sumando* −4 y +9, que es lo mismo que restar +4 de +9. Este caso es igual al caso II de la sección 6-3. Entonces, cualquier operación de resta se convierte en una de suma cuando se utiliza el sistema de complemento a 2. Esta característica del sistema de complemento a 2 lo ha convertido en el sistema de más amplio uso de todos los métodos disponibles, ya que permite realizar la suma y la resta mediante el mismo circuito.

Resta en Complemento de 2

He aquí otro ejemplo que muestra la resta de +9 con −4:

```
  11100   (−4)
− 01001   (+9)
```

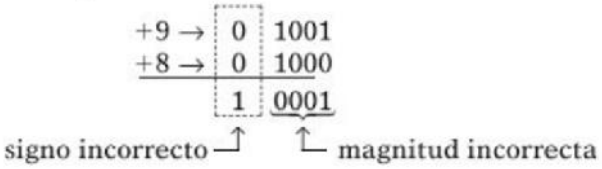
Se niega el sustraendo (+9) para producir 10111 (−9) y se suma éste al minuendo (−4):

```
  11100   (−4)
+ 10111   (−9)
1 10011   (−13)
↑
└ Se descarta
```

[ ] verificar los resultados de utilizar el procedimiento anterior para las siguientes restas: (a) +9 − (− 4); (b) −9 − (+4); (c) −9 − (−4); (d) +4 − (−4). Recuerde que cuando el resultado tiene un bit de signo de 1, es negativo y está en forma de complemento a 2.

Desbordamiento aritmético

- En cada uno de los ejemplos anteriores de sumas y restas, los números que se sumaron consistían de un bit de signo y cuatro bits de magnitud.
- Las respuestas también consistían de un bit de signo y cuatro bits de magnitud. Se descartó cualquier acarreo hacia la sexta posición de bit.
- En todos los casos considerados, la magnitud de la respuesta fue lo bastante pequeña como para ajustarla en cuatro bits. Veamos la suma de +9 y +8.



Desbordamiento aritmético

- La respuesta tiene un bit de signo negativo, lo cual es obvio que es incorrecto ya que estamos sumando dos números positivos. La respuesta deberá ser +17, pero la magnitud 17 requiere más de cuatro bits y, por lo tanto, se desborda hacia la posición del bit de signo.
- Esta condición de desbordamiento puede ocurrir sólo cuando se están sumando dos números positivos o dos números negativos, y siempre produce un resultado incorrecto. Para detectar el desbordamiento podemos comprobar si el bit de signo del resultado es el mismo que los bits de signo de los números que se van a sumar.
- Para realizar la resta en el sistema de complemento a 2, se niega el minuendo y se suma al sustraendo, por lo que el desbordamiento sólo puede ocurrir cuando el minuendo y el sustraendo tienen signos diferentes.
- Por ejemplo, si vamos a restar -8 de +9, el -8 se niega para convertirse en +8 y se suma a +9 como se muestra a continuación, y el desbordamiento produce un resultado negativo erróneo ya que la magnitud es demasiado grande.

Multiplicación de números binarios

- La multiplicación de números binarios se realiza de la misma forma que la multiplicación de números decimales. En realidad el proceso es más simple, ya que los dígitos multiplicadores son 0 o 1, por lo que siempre estamos multiplicando por 0 o 1 y ningún otro dígito. El siguiente ejemplo ilustra esto para los números binarios sin signo:

```
   1001      ← multiplicando = 9₁₀
   1011      ← multiplicador = 11₁₀
  ------
   1001   }
   1001   }  productos parciales
   0000   }
  1001    }
  ------
  1100011 }  producto final
```

Multiplicación de números binarios

- En este ejemplo el multiplicando y el multiplicador están en su forma binaria natural por lo que no se utilizan bits de signo.
- Los pasos que se siguen en el proceso son exactamente los mismos que en Ja multiplicación decimal.
- Primero se examina el LSB del multiplicador; en nuestro ejemplo es un 1. Este 1 multiplica el multiplicando para producir 1001, el cual se anota como el primer producto parcial.
- A continuación se examina el segundo bit del multiplicador. Es un 1, por lo que se escribe 1001 para el segundo producto parcial. Observe que este segundo producto parcial se *desplaza* una posición a la izquierda, en forma relativa al primer producto.
- El tercer bit del multiplicador es 0, por lo que se escribe 0000 como el tercer producto parcial; de nuevo, se desplaza una posición a la izquierda en forma relativa al producto parcial anterior.
- El cuarto bit del multiplicador es 1, por lo que el último producto parcial es 1001, desplazado una posición a la izquierda. Después los cuatro productos parciales se suman para producir el producto final.

Multiplicación de números binarios

- La mayoría de los equipos digitales puede sumar sólo dos números binarios a la vez. Por esta razón los productos parciales que se forman durante la multiplicación no pueden sumarse todos juntos al mismo tiempo, sino que se suman de dos en dos; es decir, el primero se suma al segundo, después el resultado se suma al tercero y así en lo sucesivo.

Sumar	1001	← primer producto parcial
	1001	← segundo producto parcial desplazado a la izquierda
Sumar	11011	← suma de los primeros dos productos parciales
	0000	← tercer producto parcial desplazado a la izquierda
Sumar	011011	← suma de los primeros tres productos parciales
	1001	← cuarto producto parcial desplazado a la izquierda
	1100011	← suma de los cuatro productos parciales, que equivale al producto total final

Multiplicación en complemento de 2

- En las computadoras que utilizan la representación en complemento a 2, la multiplicación se lleva a cabo en la manera antes descrita, siempre y cuando tanto el multiplicando como el multiplicador se coloquen en forma binaria natural. Si los dos números a multiplicar son positivos, ya se encuentran en la forma binaria natural y se multiplican como están. Desde luego que el producto resultante es positivo y se le asigna un bit de signo de 0. Cuando los dos números sean negativos, deberán estar en forma de complemento a 2. Se saca el complemento a 2 de cada uno para convertirlo en un número positivo y después se multiplican esos dos números. El producto se mantiene como un número positivo y recibe un bit de signo de 0.
- Cuando uno de los números es positivo y el otro negativo, primero se convierte el número negativo en una magnitud positiva, sacando su complemento a 2. El producto se encontrará en forma de magnitud real. No obstante, el producto debe ser negativo ya que los números originales son de signos opuestos. Como consecuencia, el producto se cambia a su forma de complemento a 2 y se le asigna un bit de signo de 1.

División binaria

- El proceso para dividir un número binario (el dividendo) entre otro (el divisor) es el mismo que el que se utiliza para los números decimales, al cual, por lo general, se le conoce como "división larga".
- El proceso actual es más simple en binario, ya que cuando estamos comprobando cuántas veces "cabe" el divisor en el dividendo sólo hay dos posibilidades: 0 o 1. Para ilustrar esto, considere los siguientes ejemplos simples de división:

```
   0011          0010.1
11|1001       100|1010.0
   011           100
   0011           100
    11            100
     0              0
```

División Binaria

- En el primer ejemplo tenemos el número 1001_2 dividido entre 11_2, que en decimal equivale a 9/3. El cociente resultante es $0011_2 = 3_{10}$. En el segundo ejemplo, el número 1010_2 se divide entre 100_2, o 10/4 en decimal. El resultado es $0010.1_2 = 2.5_{10}$.
- En la mayoría de los equipos digitales modernos, por lo general, las restas que son parte de la operación de división se llevan a cabo mediante el uso de la resta con complemento a 2; es decir, se saca el complemento a 2 del sustraendo y después se suma.
- La división de números con signo se maneja de la misma forma que la multiplicación. Los números negativos se convierten en positivos mediante su negación y después se lleva a cabo la división.
- Si el dividendo y el divisor tienen signos opuestos, el cociente resultante se cambia a número negativo, para lo cual se saca su complemento a 2 y se le asigna un bit de signo de l.
- Si el dividendo y el divisor son del mismo signo, el cociente se deja como número positivo y se le asigna un bit de signo de 0.

Códigos de Numeración Binaria

- La representación de cantidades por medio de algún arreglo de dígitos se denomina número, código o "palabra".
- En el sistema de numeración binaria existen varias formas de codificar o de representar cantidades.

Código Binario Natural

- En este código, los bits a la izquierda del punto se denominan enteros y los de la derecha fraccionarios.
- Las ponderaciones son positivas y ascendentes hacia la izquierda a partir del punto y negativas y descendentes hacia la derecha del punto.
- La siguiente tabla muestra los números enteros de 4-bits [binario] con sus equivalentes en: octal, decimal, hexadecimal, BCD, EXC-a-3 y GRAY observe que en BCD existen 6-códigos binarios que no se utilizan.