TÉCNICAS DIGITALES 1

REPRESENTACIÓN Y OPERACIONES ARITMÉTICAS CON NÚMEROS EN PUNTO FIJO

AUTORES:

ING. GUILLERMO A. FERNÁNDEZ
DR. ING. FERNANDO BOTTERÓN

- DEPARTAMENTO DE ELECTRÓNICA -

UNIVERSIDAD NACIONAL DE MISIONES – FACULTAD DE INGENIERÍA
OBERÁ – MISIONES – ARGENTINA

INTRODUCCIÓN

Comercialmente existen numerosos dispositivos digitales, tales como microprocesadores (uP), microcontroladores (uC), procesadores y controladores digitales de señal (DSP y DSC), etc., que son capaces de efectuar operaciones aritméticas complejas. Muchas de estas operaciones requieren de la utilización de números reales. Debido a que los dispositivos mencionados sólo operan con números binarios, y en este sistema no pueden representarse todos los números reales, los resultados obtenidos sólo son aproximaciones. Para las operaciones aritméticas, los números reales pueden expresarse de forma aproximada a través de la representación en los formatos de "Punto Flotante" y "Punto Fijo". Los dispositivos más complejos y costosos poseen una arquitectura interna (circuito interno) preparada para trabajar con números en formato de "Punto Flotante", formato que se estudiará mas adelante. Los dispositivos más sencillos y asequibles comercialmente sólo están preparados para trabajar con números enteros, a través de los cuales pueden expresarse números con decimales en un formato denominado "Punto Fijo". La representación de "Punto Fijo" es llamada así debido a que los números poseen una cantidad fija de cifras para la parte decimal. Los mismos pueden comprenderse como números enteros que están afectados por determinado factor de escala, donde la coma decimal (o punto decimal, en inglés) siempre permanece en un mismo lugar (para números afectados por un mismo factor de escala). A continuación se indica un ejemplo de lo mencionado.

Ejemplo 1: *Ejemplos de números decimales con punto fijo.*

$$\frac{1234}{100} = 12{,}34 \quad ; \quad \frac{-12345}{100} = -123{,}45$$

Como podemos observar, el factor de escala es "100", por lo tanto el número obtenido en cada caso siempre posee dos cifras decimales. Es decir, en los dos casos la coma o punto decimal se encuentra fijo. Si en ambos casos del ejemplo anterior, el factor de escala fuera "1", el resultado sería un número con el punto fijo en el extremo derecho (parte decimal nula), es decir un número entero.

Los números enteros, pueden representarse de diferentes formas, tales como: Signo-Magnitud, Complemento a 1 y Complemento a 2, siendo esta última representación de especial importancia, ya que es la utilizada en las operaciones aritméticas efectuadas por los dispositivos mencionados en un principio.

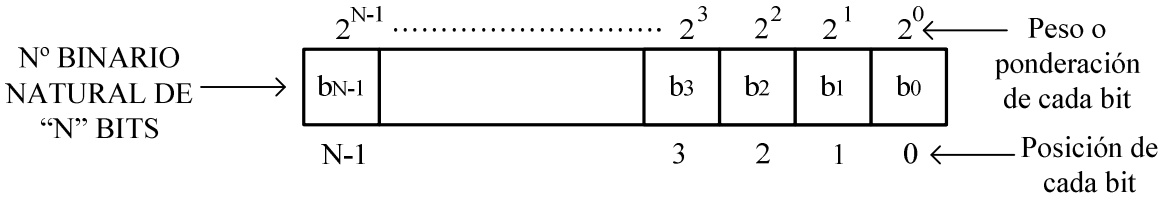
A continuación se estudiará la representación binaria de distintos conjuntos de números, la conversión de los mismos al sistema decimal y viceversa, como así también las operaciones aritméticas básicas. Algunas de las operaciones aritméticas son efectuadas de forma tal que permitan comprender el funcionamiento interno de los dispositivos digitales. Como los dispositivos digitales efectúan operaciones con números representados por palabras que poseen ancho o tamaño acotado de N bits, en todos los casos serán definidos por una cantidad acotada de bits, produciendo en las operaciones aritméticas resultados que estarán dentro de cierto rango.

Por último se estudiará la representación de números mediante el formato de "Punto Fijo", resaltando el "Formato Q" el cual puede utilizarse para operaciones aritméticas en dispositivos digitales de bajo costo.

REPRESENTACIÓN BINARIA DE LOS NÚMEROS

REPRESENTACIÓN DE NÚMEROS NATURALES

Los números naturales son aquellos que permiten contar elementos, es decir son 0, 1, 2, 3,….. Como podemos apreciar, en el conjunto de números naturales no hay números con signo. En el sistema binario, este tipo de números son representados por combinaciones de "ceros" y "unos". Estas combinaciones son expresadas a través de palabras que poseen un número definido dígitos o bits. Los números naturales están representados en binario puro, donde cada bit posee un peso o ponderación, como se muestra a continuación:



El peso de cada bit es creciente de derecha a izquierda (como en el sistema decimal, de unidad hacia decena, centena, etc.). Si el número binario representado por una palabra de N bits puede convertirse al sistema decimal mediante la siguiente general:

$$x_{10} = b_{N-1} \times 2^{N-1} + b_{N-2} \times 2^{N-2} + ... + b_1 \times 2^1 + b_0 \qquad [1]$$

Donde b_{N-1} es el bit más significativo (MSB) o de más peso y b_0 el menos significativo (LSB) o de menos peso. Una palabra con N bits, permite representar 2^N números binarios naturales diferentes, con un rango definido por:

$$0 \leq x \leq 2^N - 1 \quad ó \quad \left[0 \; ; \; 2^N - 1\right] \qquad [2]$$

Ejemplo 2: *Convertir los siguientes números binarios naturales al sistema decimal, en cada caso determinar el rango representable con la cantidad de bits utilizado.* $A_2 = 110101$; $B_2 = 11101101$.

Utilizando la expresión [1], cada número puede convertirse al sistema decimal:

$$A_2 = 110101 \rightarrow A_{10} = 1\times 2^5 + 1\times 2^4 + 0\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1\times 2^0$$
$$A_{10} = 32 + 16 + 4 + 1 \Rightarrow \underline{A_{10} = 53}$$

$$B_2 = 11101101 \rightarrow B_{10} = 1\times 2^7 + 1\times 2^6 + 1\times 2^5 + 0\times 2^4 + 1\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1\times 2^0$$
$$B_{10} = 128 + 64 + 32 + 8 + 4 + 1 \Rightarrow \underline{B_{10} = 237}$$

A través de la expresión [2], puede obtenerse el rango de representación en cada caso:

Para $A_2 = 110101 \rightarrow N = 6 \Rightarrow \left[0 \; ; \; 2^6 - 1\right] = \left[0 \; ; \; 63\right]$

Para $B_2 = 11101101 \rightarrow N = 8 \Rightarrow \left[0 \; ; \; 2^8 - 1\right] = \left[0 \; ; \; 255\right]$

Para efectuar la conversión inversa de un número natural, es decir del sistema decimal al sistema binario, debe recurrirse a la utilización divisiones sucesivas por 2. Realizando dichas divisiones hasta obtenerse el menor resultado entero posible. Para comprender el procedimiento, veamos el siguiente ejemplo.

Ejemplo 3: *Convertir los siguientes números naturales al sistema binario, utilizando el método de la división sucesiva por 2.* $A_{10} = 22$; $B_{10} = 13$.

$\frac{22}{2} = 11$; *resto* 0 $\qquad$ $\frac{13}{2} = 6$; *resto* 1

$\frac{11}{2} = 5$; *resto* 1 $\qquad$ $\frac{6}{2} = 3$; *resto* 0

$\frac{5}{2} = 2$; *resto* 1 $\qquad$ $\frac{3}{2} = \boxed{1}$; *resto* 1

$\frac{2}{2} = \boxed{1}$; *resto* 0

Para obtener el equivalente binario de cada número, debe tomarse como MSB el resultado de la última división y los restos de las divisiones anteriores. Por lo tanto queda:

$A_{10} = 22 \rightarrow A_2 = \boxed{1}0110$; $B_{10} = 13 \rightarrow B_2 = \boxed{1}101$

Operaciones Aritméticas

Para efectuar operaciones aritméticas, hay que tener en cuenta que las mismas deben arrojar resultados que se encuentren dentro del conjunto de números naturales. Por ejemplo, no podrá efectuarse una resta donde el minuendo es menor al sustraendo (ya que daría un número negativo, no representable en los números naturales), tampoco se podría realizar una división cuyo resultado arroje decimales, el resultado sólo se puede expresar mediante números naturales.

Los dispositivos digitales capaces de realizar operaciones aritméticas, efectúan las mismas con números representados por palabras, las cuales poseen una cantidad acotada de bits. Por lo tanto, los números utilizados en las operaciones deberán arrojar resultados representables por estas palabras. Es decir, si el dispositivo opera con palabras de 4 bits, sólo podrá representar correctamente resultados que van de 0 a 15 (esto es relativo, ya que en algunos casos pueden utilizarse dos o mas palabras para representar los resultados). A continuación veremos las operaciones elementales con números naturales, a partir de las cuales obtendremos conceptos importantes que son aplicables a otros conjuntos de números.

Suma: La suma de dos números binarios naturales se efectúa bit a bit, desde el bit menos significativo hacia el más significativo, generándose el bit de acarreo siempre que es necesario. Para efectuar la suma debe tenerse en cuenta que: 0+0=0 ; 0+1=1 ; 1+0=1 ; 1+1=10 (2 en binario) y 1+1+1=11 (3 en binario).

Ejemplo 4: *Sumar los números naturales A+B, en los siguientes casos: A)* $A_2=0011$ *y* $B_2=1100$*; B)* $A_2=1101$ *y* $B_2=1110$.

A)

```
    1 <-------------- Acarreo
  0 0 1 1   (3)     entre bits
+ 1 0 1 0   (10)
-----------
  1 1 0 1   (13)
```

B)

```
    1 <-------------- Acarreo
  1 1 0 1   (13)    entre bits
+ 1 1 1 0   (14)
-----------
1 | 1 0 1 1 (27)
  ^
  |
  BIT de Acarreo
  (CARRY BIT)
```

En la primera suma, cuando se efectúa 1+1 (en los bits menos significativos), el resultado es 10. Por lo tanto se anota "0" en el resultado y el "1" es acarreado a la siguiente columna de bits. En la segunda suma puede apreciarse que el resultado no puede representarse con una palabra de 4 bits, lo cual queda indicado por el bit de acarreo (CARRY BIT). Este bit está presente en dispositivos digitales que realizan operaciones aritméticas, sirve para indicar que el resultado ha sobrepasado el rango de representación (en éste caso de 0 a 15).

Resta: Esta operación se efectúa restando bit a bit los números comenzando desde el bit menos significativo hacia el más significativo. Para efectuar la resta debe tenerse en cuenta que: 0–0=0 ; 1–0=1 ; 1–1=0 y 0–1=1 ya que debe pedirse prestado "1" al siguiente bit de mas peso para efectuar 10–1=1. Para comprende esto, veamos el siguiente ejemplo.

Ejemplo 5: *Realizar la resta de números naturales A–B, siendo A_2=11000 y B_2=00011.*

```
                 0 1 1 1 <-------- Bits que se
Minuendo ---->   1 X 0 0 0  (24)    pidieron
Sustraendo ---> - 0 0 0 1 1  (3)     prestado
                -----------
                  1 0 1 0 1  (21)
```

Como puede verse en el ejemplo, cuando no puede efectuarse la resta se "pide prestado" al bit más significativo siguiente. Debe recordarse que la resta de números naturales, puede realizarse siempre que el minuendo es mayor o igual al sustraendo.

Multiplicación: Esta operación puede efectuarse como en el sistema decimal o repitiendo el multiplicando desplazado hacia la izquierda, conforme a la posición que ocupen los unos del multiplicador. Luego deben sumarse todos los valores desplazados. A este método se denomina de suma sucesiva. Veamos el siguiente ejemplo.

Ejemplo 6: *Realizar las siguientes multiplicaciones entre números naturales:*
A) A_2=0011× B_2=0101; B) A_2=1110 × B_2=0011; C) A_2=0011 × B_2=1000.

A)

```
Multiplicando ---> 0 0 1 1  (3)
Multiplicador ---> × 0 1 0 1  (5)
                   -----------
                     0 0 1 1
                   + 0 0 0 0
                   -----------
                   0 0 1 1
                 + 0 0 1 1
                   -----------
                 0 0 | 1 1 1 1  (15)
```

B)

```
  1 1 1 0  (14)
× 0 0 1 1  (3)
-----------
  1 1 1 0
+ 1 1 1 0
-----------
1 0 | 1 0 1 0  (42)
```

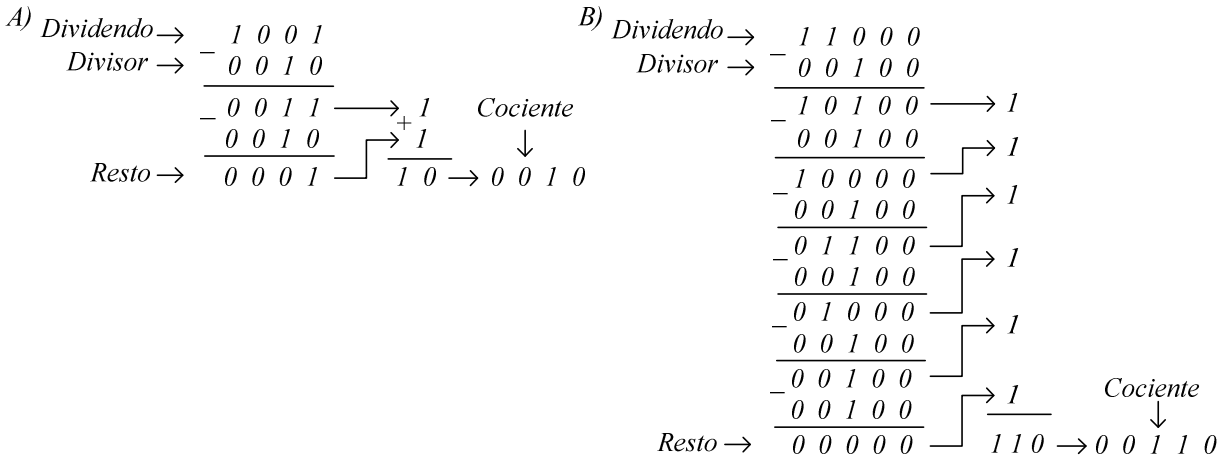
C)

```
  0 0 1 1  (3)
× 1 0 0 0  (8)
-----------
  0 0 0 0
+ 0 0 0 0
-----------
  0 0 0 0 0
+ 0 0 0 0
-----------
  0 0 0 0 0 0
+ 0 0 1 1
-----------
0 0 1 | 1 0 0 0  (24)
```

Si el dispositivo digital que efectúa las multiplicaciones del ejemplo, opera con palabras de 4 bits, no podría representar el resultado de la segunda multiplicación, ya que el mismo excede el rango de los números con los 4 bits (de 0 a 15). En el tercer caso del ejemplo, puede observarse al resultado como el multiplicando desplazado hacia la izquierda en tres posiciones. El desplazamiento de un número hacia la izquierda se produce cada vez que es efectuada la multiplicación del mismo por la unidad seguida de ceros (como en el sistema decimal). En el sistema binario la unidad seguida de ceros es representada por el equivalente binario a 2^n, siendo "n" el número de ceros (n=1;2;3;…).

División: Esta operación puede realizarse restando sucesivamente el divisor del dividendo. Las restas son realizadas hasta obtener "0" o no poder efectuar más la operación. Por cada resta realizada se incrementa en uno el cociente. En el siguiente ejemplo es indicado el método. Debe recordarse que en el conjunto de números naturales no podrán resolverse de forma exacta aquellas divisiones donde el dividendo no es múltiplo del divisor.

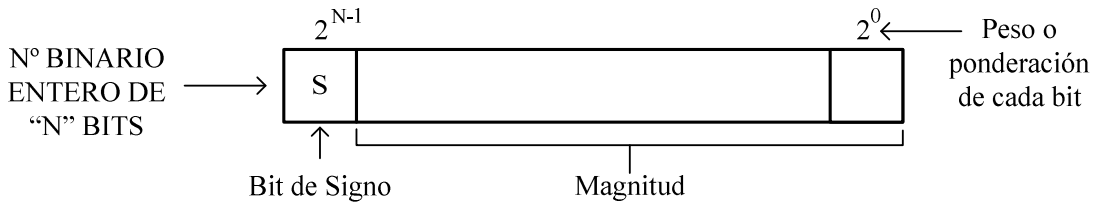
Ejemplo 7: *Realizar las siguientes divisiones entre números naturales:*
A) $A_2=1001 \div B_2=0010$*; B)* $A_2=11000 \div B_2=00100$*.*



Como puede apreciarse en el ejemplo anterior, para el primer caso no puede obtenerse el resultado exacto ya que el dividendo no es múltiplo del divisor. En el otro caso puede observarse que al efectuar la división por la unidad seguida de ceros, el resultado corresponde al dividendo desplazado hacia la derecha tantos bits como ceros posee el divisor. Más adelante se estudiará mejor el desplazamiento o corrimiento de un número.

REPRESENTACIÓN DE NÚMEROS ENTEROS

El conjunto de los números enteros está conformado por los números naturales y sus opuestos (incluyendo el "0"), es decir son números con signo (…;-2;-1;0;+1;+2;…). Para representarlos en el sistema binario, el bit mas significativos de la palabra es utilizado como bit de signo. De esta forma la representación de los números enteros en el sistema binario, mediante palabras de N bits, queda:



La forma indicada posee dos variantes para la representación de los números enteros: "Signo-Magnitud" y "Complemento a 2", siendo esta última la utilizada por los dispositivos digitales que poseen capacidad para efectuar operaciones aritméticas.

En la representación binaria de "Signo-Magnitud", el bit de signo permite representar a números positivos y negativos de acuerdo a lo siguiente:

S = 0; número entero positivo.
S = 1; número entero negativo.

La magnitud (o valor absoluto) del número es representada directamente en binario puro.

Ejemplo 8: *En signo-magnitud, a qué enteros corresponden los siguientes números: A_2=10111 y B_2=01100.*

Para A_2: 1 0111 = – 7 *Para B_2: 0 1100 = + 12*

Otra forma de representar a los números binarios enteros es en "complemento a 2". En esta representación, los números positivos son expresados como si estuvieran en "Signo-Magnitud", mientras que los números negativos en complemento a 2. El complemento a 2 de un número binario se obtiene invirtiendo bit a bit el número (a esto se denomina complemento a 1) y luego sumándole "1". Cabe aclarar que en la representación de complemento a 2 el MSB siempre es el bit de signo y su representación es igual que en "Signo-Magnitud". El siguiente ejemplo muestra como obtener el complemento a 2.

Ejemplo 9: *Realizar el complemento a 2 de A_2=10110 y B_2=01100.*

$A_2 \longrightarrow$ 1 0 1 1 0

\+ 0 1 0 0 1 ← Complem. a 1

1

0 1 0 1 0 ← Complem. a 2

$B_2 \longrightarrow$ 0 1 1 0 0

1 1

\+ 1 0 0 1 1 ← Complem. a 1

1

1 0 1 0 0 ← Complem. a 2

Cuando un número binario entero es negativo y se encuentra en complemento a 2, para obtener la magnitud del mismo basta con volver a complementarlo a 2. En el ejemplo anterior, el número negativo A_2=10110 se trata de – 10, como lo muestra su complemento a 2. Por otra parte si se desea saber el opuesto de un número, basta con expresarlo de forma positiva y luego complementarlo a 2. El segundo caso del ejemplo anterior indica lo mencionado, se tiene B_{10}= +12 (01100) y su opuesto será B_2= 10100 (–12).

Para obtener algunas conclusiones acerca de la representación en signo-magnitud y en complemento a 2, veamos los números que pueden expresarse a través de una palabra de 4 bits.

Combinaciones con 4 bits	Nº Naturales	Signo-Magnitud	Nº Enteros en CA2
0000	0	+ 0	0
0001	1	+ 1	+ 1
0010	2	+ 2	+ 2
0011	3	+ 3	+ 3
0100	4	+ 4	+ 4
0101	5	+ 5	+ 5
0110	6	+ 6	+ 6
0111	7	+ 7	+ 7
1000	8	– 0	– 8
1001	9	– 1	– 7
1010	10	– 2	– 6
1011	11	– 3	– 5
1100	12	– 4	– 4
1101	13	– 5	– 3
1110	14	– 6	– 2
1111	15	– 7	– 1

Como puede apreciarse en la tabla anterior, en signo-magnitud, el número "0" posee dos representaciones. Esto no es deseable, ya que para efectuar operaciones aritméticas se pretende que cada combinación binaria corresponda a un único número. También puede apreciarse que la representación de números con signo, depende de la interpretación que se da a cada combinación binaria, las combinaciones no son modificadas según el complemento que se utilice.

El complemento a 2, además de servir para la expresar números con signo, permite efectuar la resta de dos números como si fuera una suma. En los dispositivos digitales, esto permite aprovechar para realizar restas, los mismos circuitos que son utilizados para efectuar sumas. Por tal motivo, la representación en complemento a 2 es la utilizada en las operaciones aritméticas de dispositivos digitales tales como: microprocesadores, microcontroladores, etc. Luego se explicarán las operaciones aritméticas utilizando números representados de ésta forma.

Para convertir al sistema decimal, un número binario entero representado en complemento a 2, puede utilizarse la siguiente expresión:

$$x_{10} = -b_{N-1} \times 2^{N-1} + b_{N-2} \times 2^{N-2} + ... + b_1 \times 2^1 + b_0 \qquad [3]$$

Donde los coeficientes "*bi*" (con $i = 0;1;2; \ldots ;N-1$) corresponden a los bits del número binario, siendo b_{N-1} el bit mas significativo (MSB) y b_0 el menos significativo (LSB).

Ejemplo 10: *Determinar el equivalente decimal de los siguientes números binarios enteros, representados por una palabra de 8 bits:* $A_2 = 10100101$ *y* $B_2 = 01101101$.

Como los números poseen ocho bits (N=8), la expresión general [3] queda de la siguiente forma:

$$x_{10} = -b_7 \times 2^7 + b_6 \times 2^6 + ... + b_1 \times 2^1 + b_0$$

Aplicando la anterior a ambos números, puede realizarse la conversión deseada.

$$A_2 = 10100101 \rightarrow A_{10} = -1\times 2^7 + 0\times 2^6 + 1\times 2^5 + 0\times 2^4 + 0\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1$$
$$A_{10} = -128 + 32 + 4 + 1 \Rightarrow \underline{A_{10} = -91}$$

$$B_2 = 01101101 \rightarrow B_{10} = -0\times 2^7 + 1\times 2^6 + 1\times 2^5 + 0\times 2^4 + 1\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1$$
$$B_{10} = 64 + 32 + 8 + 4 + 1 \Rightarrow \underline{B_{10} = 109}$$

Mediante una palabra de N bits podrán representarse 2^N números binarios enteros que están en un rango representado por:

$$-2^{N-1} \le x \le +2^{N-1} - 1 \quad \acute{o} \quad \left[-2^{N-1} \; ; \; +2^{N-1} - 1\right] \qquad [4]$$

Ejemplo 11: *¿Cuál es el rango para una palabra de 16 bits que puede representar a: A) Números Enteros ; B) Números Naturales (incluido el "0")?.*

A) Los números enteros, son números con signo, por lo tanto son representados en complemento a 2. Entonces, a partir de la expresión [4] puede obtenerse el rango.

$$\left[-2^{N-1} \; ; \; +2^{N-1} - 1\right] = \left[-2^{16-1} \; ; \; +2^{16-1} - 1\right] = \left[-2^{15} \; ; \; +2^{15} - 1\right] = \left[-32768 \; ; \; +32767\right]$$

B) Los números naturales no poseen números negativos, es decir son todos positivos, por lo tanto el rango será:

$$\left[0 \; ; \; 2^N - 1\right] = \left[0 \; ; \; 2^{16} - 1\right] = \left[0 \; ; \; 65535\right]$$

Antes de tratar las operaciones aritméticas veamos en el siguiente ejemplo una particularidad del bit de signo en la representación en complemento a 2.

Ejemplo 12: *Determinar la representación de los números A_{10}=+5 y B_{10}=–6, mediante una palabra de 8 bits.*

$$A_2 = 0\,0\,0\,0\,0\,1\,0\,1 \qquad +6 \rightarrow 0\,0\,0\,0\,0\,1\,1\,0$$
$$1\,1\,1\,1\,1\,0\,0\,1$$
$$\underline{\qquad\qquad\qquad 1}$$
$$B_2 = 1\,1\,1\,1\,1\,0\,1\,0$$

En el ejemplo puede apreciarse que en ambos casos el bit de signo es repetido hasta completar los 8 bits, a esto se denomina "extensión del bit de signo". Todo número debe ser representado de esta forma para poder realizar correctamente las operaciones aritméticas. Es decir, los números expresados con N bits, deben poseer el bit de signo extendido para poder completar los N bits (se los debe completar con "ceros" si es positivo, o con "unos" si es negativo).

Operaciones Aritméticas

A continuación veremos las operaciones aritméticas elementales, en las mismas son considerados números enteros representados por palabras de N bits, por lo cual los resultados deberán permanecer dentro del rango establecido por la expresión [4].

Suma y Resta: La suma de dos números binarios enteros se efectúa de igual forma que en el caso de los números binarios naturales (bit a bit). La resta se efectúa sumando al minuendo el complemento a 2 de sustraendo (es decir el opuesto del sustraendo). Tanto como para la suma y la resta, los números deben sumarse incluyendo el bit de signo. En el siguiente ejemplo es explicado el procedimiento para ambas operaciones.

Ejemplo 13: *Suponiendo que un dispositivo digital opera con palabras de 5 bits, realizar las siguientes operaciones con $A_{10}=10$, $B_{10}=3$, $C_{10}=12$: A) A+B ; B) A–B; C) B–C; D) –A–C; E) A+C .*

$A_2 = 01010 \rightarrow -A_2 = 10110$
$B_2 = 00011 \rightarrow -B_2 = 11101$
$C_2 = 01100 \rightarrow -C_2 = 10100$

A+B

$$\begin{array}{rr} & 0\ 1\ 0\ 1\ 0 = 10 \\ + & 0\ 0\ 0\ 1\ 1 = 3 \\ \hline & 0\ 1\ 1\ 0\ 1 = 13 \end{array}$$

A–B=A+(–B)

$$\begin{array}{rr} & 0\ 1\ 0\ 1\ 0 = 10 \\ + & 1\ 1\ 1\ 0\ 1 = -3 \\ \hline & 1 \mid 0\ 0\ 1\ 1\ 1 = 7 \end{array}$$

Acarreo que se ignora

B–C=B+(–C)

$$\begin{array}{rr} & 0\ 0\ 0\ 1\ 1 = 3 \\ + & 1\ 0\ 1\ 0\ 0 = -12 \\ \hline & 1\ 0\ 1\ 1\ 1 = -9 \end{array}$$

–A–C=(–B)+(–C)

$$\begin{array}{rr} & 1\ 0\ 1\ 1\ 0 = -10 \\ + & 1\ 0\ 1\ 0\ 0 = -12 \\ \hline & 1 \mid 0\ 1\ 0\ 1\ 0 \\ & \uparrow \quad \downarrow \\ & \text{Acarreo} \quad 10 \\ & \text{(Desbordamiento)} \end{array}$$

A+C

$$\begin{array}{rr} & 0\ 1\ 0\ 1\ 0 = 10 \\ + & 0\ 1\ 1\ 0\ 0 = 12 \\ \hline & 1\ 0\ 1\ 1\ 0 \\ & \downarrow \\ & -10 \end{array}$$

A partir del ejemplo anterior y considerando que el resultado debe ser representado en N bits, podemos concluir:

- Si se suman dos números de igual signo, puede producirse un acarreo que representa un resultado fuera de rango (como en –A–C). Si no se produce el acarreo mencionado, la suma puede arrojar un resultado con un signo que no corresponde (como en A+C). En ambos casos estamos ante un resultado erróneo.

- Si se suman dos números con distintos signos, puede producirse un acarreo, el cual se descarta (como en A–B) obteniéndose así un resultado correcto. Esto es debido a que la operación en realidad es una resta y nunca podría dar fuera del rango de representación con los N bits.

Multiplicación y división: La multiplicación puede efectuarse directamente con los números en complemento a 2, recordando que el resultado corresponde a los N bits menos significativos (siempre que multiplicando y multiplicador posean un ancho de N bits). Cabe mencionar que el resultado ya estará con el signo correcto, es decir si es negativo estará en complemento a 2. Para la división conviene expresar dividendo y divisor como enteros positivos, efectuar la operación como si fueran naturales y luego aplicar la regla de los signos para expresar el resultado como positivo o como negativo. La regla de los signos dice que si dividendo y divisor poseen distintos signos, el

resultado será negativo, en caso contrario será positivo. Hay que recordar que por tratarse de números enteros, la división exacta sólo podrá efectuarse cuando el dividendo es múltiplo del divisor y también mayor o igual que éste último.

Ejemplo 14: *Realizar las siguientes multiplicaciones: A) (–3)×(5). B) (–3)×(–4).*

Según la expresión [4], para efectuar estas operaciones es suficiente la utilización de números con N=5 bits. Por lo tanto:

```
  1 1 1 0 1 ← (–3)
×
  0 0 1 0 1 ← (5)
-----------
  1 1 1 0 1
+ 0 0 0 0 0
-----------
  0 1 1 1 0 1
+   1 1 1 0 1
-----------
1 0 0 1 0 0 0 1   → (–15)
```

```
  1 1 1 0 1 ← (–3)
×
  1 1 1 0 0 ← (–4)
-----------
  0 0 0 0 0
+ 0 0 0 0 0
-----------
  0 0 0 0 0 0
+   1 1 1 0 1
-----------
  1 1 1 0 1 0 0
+   1 1 1 0 1
-----------
  1 0 1 0 1 1 1 0 0
+   1 1 1 0 1
-----------
1 1 0 0 1 0 1 1 0 0   → (12)
```

Ejemplo 15: *Realizar la siguiente división utilizando el método de las restas sucesivas: (–15)÷(5).*

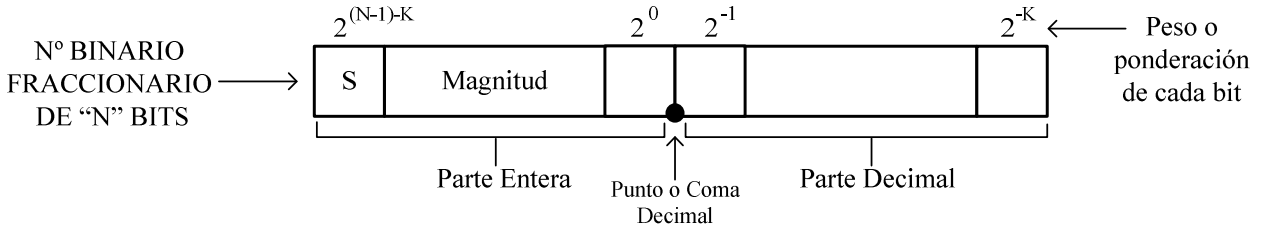
Según la expresión [4], para efectuar estas operaciones deben utilizarse números con N=5 bits. Para realizar la división, los números negativos deben expresarse como positivos. Por lo tanto:

```
  0 1 1 1 1 ← (15)
-
  0 0 1 0 1 ← (5)
-----------
  0 1 0 1 0 → 1
-
  0 0 1 0 1
-----------
  0 0 1 0 1 → 1      0 0 0 1 1
-
  0 0 1 0 1
-----------
  0 0 0 0 0 → 1
```

Aplicando la regla de los signos para la división, el resultado es negativo. Por lo tanto debe expresarse en complemento a 2: 11101

REPRESENTACIÓN DE NÚMEROS FRACCIONARIOS O DECIMALES

Los números fraccionarios expresados de forma decimal (de ahora en más números fraccionarios, para no confundir decimal con sistema decimal), poseen una parte entera y una parte decimal, ambas separadas por la coma (o el punto, en inglés). Los números fraccionarios, en el sistema binario pueden representarse mediante una palabra de N bits, de los cuales K bits son utilizados para la parte decimal. De esta forma el número binario fraccionario quedará de la siguiente forma:



Como es un número con signo, el mismo se encuentra representado en el sistema de complemento a 2. La parte entera utiliza el bit más significativo para el signo del número, con la misma convención utilizada para los números enteros. A la representación anterior se denomina de "Punto Fijo".

Para obtener el equivalente decimal del número binario fraccionario, se recurre a la siguiente expresión general:

$$x_{10} = -b_{(N-1)-K} \times 2^{(N-1)-K} + b_{(N-2)-K} \times 2^{(N-2)-K} + ... + b_1 \times 2^1 + b_0 + \\ + b_{-1} \times 2^{-1} + b_{-2} \times 2^{-2} + ... + b_{-K} \times 2^{-K} \qquad [5]$$

En la expresión anterior, los coeficientes "*bi*" (con (N–1)–K ≤ *i* ≤ –K) corresponden a los bits del número binario fraccionario.

Debido a que un número entero podría considerarse como un número fraccionario con el punto fijo en su extremo derecho, la expresión [5] se extiende también a los mismos, considerando los coeficientes $b_{-i}=0$ y K=0 (parte decimal nula).

Ejemplo 16: *Determinar el equivalente decimal para los siguientes números binarios fraccionarios:* A_2*= 011011,11 y* B_2*= 110,10111 .*

Para el número "A", que es positivo ya que su MSB=0, tenemos N=8 y K=2, por lo tanto a partir de la expresión general [5] quedará:

$$x_{10} = -b_5 \times 2^5 + b_4 \times 2^4 + b_3 \times 2^3 + b_2 \times 2^2 + b_1 \times 2^1 + b_0 + b_{-1} \times 2^{-1} + b_{-2} \times 2^{-2}$$

$$A_2 = 011011{,}11 \rightarrow A_{10} = -0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 + 1 \times 2^{-1} + 1 \times 2^{-2}$$

$$A_{10} = 16 + 8 + 2 + 1 + 1/2 + 1/4 \Rightarrow \underline{A_{10} = 27{,}75}$$

Para el número "B", que es negativo (el MSB=1), tenemos N=8 y K=5, por lo tanto a partir de la expresión general [5] quedará:

$$x_{10} = -b_2 \times 2^2 + b_1 \times 2^1 + b_0 + b_{-1} \times 2^{-1} + b_{-2} \times 2^{-2} + b_{-3} \times 2^{-3} + b_{-4} \times 2^{-4} + b_{-5} \times 2^{-5}$$

$$B_2 = 110{,}10111 \rightarrow B_{10} = -1 \times 2^2 + 1 \times 2^1 + 0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} + 1 \times 2^{-5}$$

$$B_{10} = -4 + 2 + 1/2 + 1/8 + 1/16 + 1/32 \Rightarrow \underline{B_{10} = -1{,}28125}$$

Con el fin de comprender algunas características más sobre la representación de los números binarios fraccionarios, veamos que sucede considerando distintas cantidades de bits para la parte decimal del número, con el equivalente decimal de las 16 combinaciones obtenidas a partir de una palabra de 4 bits.

Combinaciones con 4 bits	Con K=0 (Enteros)	Con K=1 (xxx,x)	Con K=2 (xx,xx)	Con K=3 (x,xxx)
0000	0	0	0	0
0001	1	0,5	0,25	0,125
0010	2	1,0	0,50	0,250
0011	3	1,5	0,75	0,375
0100	4	2,0	1,00	0,500
0101	5	2,5	1,25	0,625
0110	6	3,0	1,50	0,750
0111	7	3,5	1,75	0,875
1000	– 8	– 4,0	– 2,00	– 1,000
1001	– 7	– 3,5	– 1,75	– 0,875
1010	– 6	– 3,0	– 1,50	– 0,750
1011	– 5	– 2,5	– 1,25	– 0,625
1100	– 4	– 2,0	– 1,00	– 0,500
1101	– 3	– 1,5	– 0,75	– 0,375
1110	– 2	– 1,0	– 0,50	– 0,250
1111	– 1	– 0,5	– 0,25	– 0,125

De la tabla anterior puede concluirse lo siguiente:

- Los números fraccionarios aparecen en "saltos" determinados por 2^{-K}. Estos saltos establecen la resolución de la representación, es decir el valor más pequeño capaz de expresarse con la cantidad de bits utilizados para la parte decimal del número.

$$Re\,solución = 2^{-K} \qquad [6]$$

- A mayor cantidad de bits para la parte decimal, los "saltos" son mas chicos, pudiéndose así representar números mas pequeños. Esto quiere decir que habrá mejor resolución en la representación del número fraccionario (hay una mejor aproximación a los números reales).

- Los números fraccionarios obtenidos en cada columna pueden considerarse que provienen de realizar el cociente entre un número entero y 2^{K}. O el producto entre el número entero y 2^{-K}. A partir de lo mencionado y considerando la expresión [4], el rango para un número fraccionario de N bits, con K bits para la parte decimal, será:

$$-\left(2^{N-1}\right).2^{-K} \le x \le +\left(2^{N-1}-1\right).2^{-K} \;\; ó \;\; \left[-\left(2^{N-1}\right).2^{-K};\; +\left(2^{N-1}-1\right).2^{-K}\right]$$
$$ó \;\; \left[-\left(2^{N-1-K}\right);\; +\left(2^{N-1-K}-2^{-K}\right)\right] \qquad [7]$$

- La cantidad de bits utilizada para la parte entera, influye notablemente en el rango de la representación. Con más bits para la parte entera, el alcance de la representación es mayor ya que pueden expresarse números más positivos y más negativos.

- Para una palabra con una cantidad definida de bits, la mejora de la resolución va en detrimento del rango, y viceversa.

Ejemplo 17: *Si un número binario fraccionario es representado por una palabra de 8 bits y para la parte decimal se utilizan 4 bits, hallar: A) El menor número que puede expresarse. B) El rango de números que podrá ser representado. C) La cantidad de números que podrá representarse.*

A) Como K=4, a través de la expresión [6] puede obtenerse el número mas pequeño que podrá representarse:

$$Resolución = 2^{-K} = \frac{1}{2^4} = \frac{1}{16} = \underline{0{,}0625}$$

B) Siendo N=8 y K=4, a partir de la expresión [7] el rango será:

$$\left[-\left(2^{N-1}\right).2^{-K} \;;\; +\left(2^{N-1}-1\right).2^{-K}\right] = \left[-\left(2^{8-1}\right).2^{-4} \;;\; +\left(2^{8-1}-1\right).2^{-4}\right] = [-8 \;;\; +7{,}9375]$$

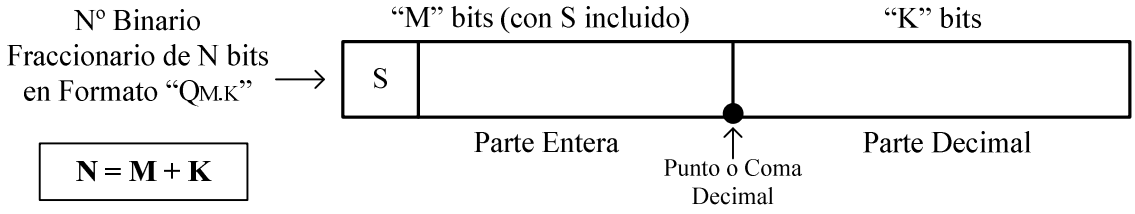
C) Como N=8, podrán representarse 2^8=256 números diferentes.

REPRESENTACIÓN BINARIA EN FORMATO Q

DEFINICIÓN DEL FORMATO

Ya hemos dicho que los dispositivos digitales como ser: microprocesadores, DSPs, etc., para sus operaciones aritméticas utilizan palabras con una cantidad de bits acotada. Esto quiere decir que si el dispositivo opera con palabras de N bits, los números sólo podrán representarse a través de N bits en un rango definido por esta cantidad.

El "Formato Q", o también denominado "Formato IQ", consiste en tomar a las palabras de N bits y ubicar un punto decimal "ficticio" con el fin de representar números que posean una parte entera y parte decimal, es decir números fraccionarios. Para denotar a los números en este formato, se recurre a lo siguiente:



Un número en formato "$Q_{M.K}$", está representado por una palabra de N bits (N=M+K), donde M bits son utilizados para la parte entera y K bits para la parte decimal. En esta representación, al igual que lo explicado en párrafos anteriores en los números fraccionarios, la parte entera está representada en complemento a 2, siendo el MSB utilizado para el signo del número (positivo S=0, negativo S=1). Una variante a la denotación indicada anteriormente es "Q_K", pero en esta forma abreviada debe aclararse de antemano con que longitud de palabra se está trabajando.

Para efectuar la conversión de un número binario en formato Q, al sistema decimal se utiliza la expresión general [5]. En cuanto a la resolución y el rango de los números en este formato, se adoptan los mismos criterios observados anteriormente (expresiones [6] y [7]).

Ejemplo 18: *Sea un dispositivo digital que opera con palabras de dato de 8 bits, determinar la cantidad de bits utilizados para la parte real y la parte decimal, como así también la resolución y el rango de la representación de números en los formatos: A) $Q_{2.6}$; B) $Q_{3.5}$; C) Q_7.*

A) *En el formato $Q_{2.6}$, la parte entera estará representada por M=2 bits y la parte decimal por K=6 bits. La resolución y el rango de la representación en este formato se obtienen a través de las expresiones [6] y [7] respectivamente.*

$$Resolución = 2^{-K} = 1/2^6 = 0{,}015625$$
$$Rango = \left[-\left(2^{8-1}\right).2^{-6} \;;\; +\left(2^{8-1}-1\right).2^{-6}\right] = \left[-2 \;;\; +1{,}984375\right]$$

B) *En el formato $Q_{3.5}$ tenemos M=3 bits y K=5 bits. La resolución y el rango serán:*

$$Resolución = 2^{-K} = 1/2^5 = 0{,}03125$$
$$Rango = \left[-\left(2^{8-1}\right).2^{-5} \;;\; +\left(2^{8-1}-1\right).2^{-5}\right] = \left[-4 \;;\; +3{,}96875\right]$$

C) *El dispositivo digital opera con palabras de N=8bits, por lo tanto en el formato Q_7 tenemos M=1 bit y K=7 bits. La resolución y el rango serán:*

$$Resolución = 2^{-K} = 1/2^7 = 0{,}0078125$$
$$Rango = \left[-\left(2^{8-1}\right).2^{-7} \;;\; +\left(2^{8-1}-1\right).2^{-7}\right] = \left[-1 \;;\; +0{,}9921875\right]$$

En ejemplos anteriores, hemos visto como convertir un número binario fraccionario al sistema decimal mediante la utilización de la expresión general [5], a continuación veremos el paso inverso, es decir como convertir un número fraccionario del sistema decimal, a su equivalente binario. Para esto haremos uso de números representados en formato Q, ya que en la práctica siempre será necesario trabajar con números que están representados por una cantidad finita de bits.

Antes de explicar el método para efectuar la conversión mencionada, veamos el siguiente ejemplo para apreciar una particularidad.

Ejemplo 19: *Sea el número X_2=111.00111, para el mismo: A) Convertirlo en un número entero. B) Hallar la relación que hay entre los equivalentes del sistema decimal, para X_2 como fraccionario y como entero.*

A) *Para convertir a X_2 en un número entero, se debe correr el punto a su extremo derecho. Como en el sistema decimal, la coma o punto decimal se corre hacia la derecha al multiplicar a un número por la unidad seguida de ceros 10^n, en el sistema binario es lo mismo. Como habíamos visto, en el sistema binario la unidad seguida de ceros está representada por 2^n. Entonces, para convertir a X en entero, se lo debe multiplicar por $2^K=2^5$. De esta forma se obtiene:*

$$X'_2 = 111.00111 \times 2^5 = 11100111$$

B) *Utilizando la expresión [5] puede hallarse el equivalente en el sistema decimal para X_2 y X'_2 y la relación solicitada.*

$$X_2 = 111.00111 \rightarrow X_{10} = -0,78125$$
$$X'_2 = 11100111 \rightarrow X'_{10} = -25$$
$$\frac{X'_{10}}{X_{10}} = \frac{-25}{-0,78125} = 32 = 2^5 \quad \Rightarrow X'_{10} = 2^5.(-0,78125)$$

Como puede apreciarse en el ejemplo, para determinar a que número del sistema decimal corresponde un binario fraccionario convertido en entero, basta con multiplicar por 2^K a su equivalente en el sistema decimal. Es decir:

$$X'_{10} = 2^K . X_{10} \qquad [8]$$

Siendo "K" la cantidad de bits del equivalente binario de X_{10}.

El ejemplo anterior nos permitirá mostrar el método para convertir un número fraccionario del sistema decimal al sistema binario. El método que se explicará a continuación es general, puede aplicarse tanto a números positivos como negativos y consiste en:

P1 – Primeramente, obtener la cantidad de bits "K", que corresponde a la cantidad de bits decimales con que se trabajará. Luego aplicar la expresión [8], para obtener X'_{10}. Esto debe efectuarse, ya que si el número fraccionario es negativo, su parte decimal no puede pasarse directamente al complemento a 2. Si en el producto se obtiene un número con parte decimal, esta debe descartarse ya que sólo interesa la parte entera del mismo (X'_{10} es un entero que proviene del número binario entero X'_2).

P2 – Luego, el número entero obtenido debe convertirse a su equivalente binario. Hay que tener en cuenta que si el número es negativo, deberá complementarse a 2.

P3 – Por último debe colocarse el punto teniendo en cuenta el valor de K.

Para comprender mejor el método, a continuación se muestra un ejemplo.

Ejemplo 20: *Convertir al sistema binario los números A_{10} = 2,32 y B_{10} = –4,34. Utilizar para la representación binaria palabras de 8 bits y adoptar un formato Q tal que permita la mejor aproximación posible (es decir con la mejor resolución).*

Para obtener la mejor resolución en las representaciones, debe utilizarse la mayor cantidad posible de bits para la parte decimal del número binario (es decir, el mayor K posible). Teniendo en cuenta la expresión [5], para A_{10} la representación debe realizarse en formato Q_5 y para B_{10} en formato Q_4. A continuación, para efectuar la conversión aplicaremos los pasos explicados anteriormente.

Para A_{10} = 2,32 tenemos:

P1 – Se trabaja con K=5; entonces: $A'_{10} = 2,32 \times 2^5 = 74,24$. Descartamos la parte decimal, entonces $A'_{10} = 74$.
P2 – Convertimos el entero a su equivalente binario de N=8 bits, $A'_2 = 01001010$.
P3 – Colocamos el punto ocupando K=5 bits para la parte decimal, entonces:

$A_{10} = 2,32 \rightarrow A_2 = 010.01010$

Para $B_{10} = –4,34$ tenemos:

P1 – $B'_{10} = –4,34 \times 2^4 = –69,44$. Descartamos la parte decimal, entonces $B'_{10} = –69$.
P2 – Convertimos el entero a su equivalente binario de N=8 bits (en complemento a 2, ya que es negativo), $B'_2 = 10111011$.
P3 – Colocamos el punto ocupando K=4 bits para la parte decimal, entonces:

$$B_{10} = –4,34 \rightarrow B_2 = 1011.1011$$

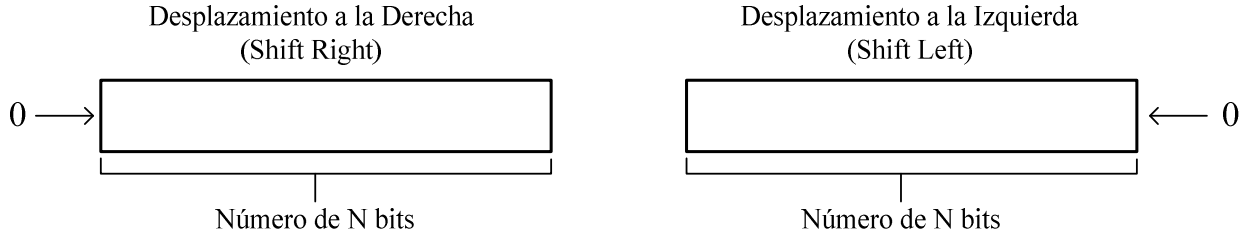
Si cada resultado obtenido en las conversiones del ejemplo anterior, vuelve a convertirse al sistema decimal mediante la expresión [5], veremos que no coinciden directamente con los valores originales de los números A_{10} y B_{10}, sólo son aproximaciones. Aparece un error, denomina "error de cuantización", este error será menor cuanto mas bits sean utilizados para la parte decimal (siempre que el número pueda representarse dentro del rango establecido por la expresión [7]).

Antes de estudiar las operaciones aritméticas básicas, veamos el efecto de corrimiento o desplazamiento de un número.

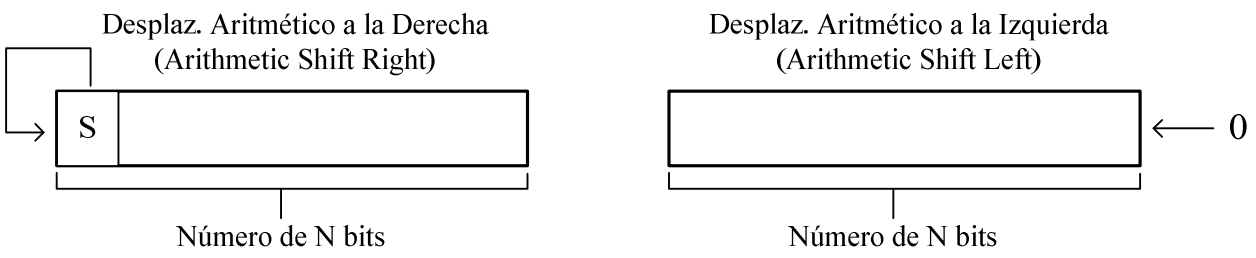
CORRIMIENTO O DESPLAZAMIENTO DE UN NÚMERO

El corrimiento o desplazamiento de un número es una operación en la que los N bits del mismo se corren hacia la izquierda o hacia la derecha. La longitud del desplazamiento se denomina "n", que corresponde a los "n" bits que se ha desplazado el número. Estos "n" bits constituyen un hueco, cuyo "relleno" depende del tipo de desplazamiento. Los desplazamientos que veremos son: Desplazamiento Lógico y Desplazamiento Aritmético.

Desplazamiento Lógico: Consiste en efectuar un desplazamiento del número a izquierda o derecha, completando con "ceros" los "n" bits del hueco. La siguiente figura expresa éste tipo de corrimiento.



Desplazamiento Aritmético: Consiste en efectuar un desplazamiento del número a izquierda o derecha. Cuando el desplazamiento es hacia la derecha, los "n" bits del hueco se rellenan con el bit de signo. Cuando el desplazamiento es hacia la izquierda, completando con "ceros" los "n" bits del hueco se rellenan con "0" (como en el desplazamiento lógico). En la siguiente figura se observa este desplazamiento.



El desplazamiento aritmético hacia la derecha permite "extender el bit de signo", ya que efectúa el desplazamiento "copiando" dicho bit. Esto es sumamente útil para efectuar operaciones aritméticas con signo.

Ejemplo 21: *Para los números de 8 bits A_2=11001000 y B_2=01111000 realizar los siguientes desplazamientos con una longitud n=3: A) Desplazamiento lógico hacia derecha e izquierda. B) Desplazamiento aritmético hacia derecha e izquierda.*

		A_2=11001000	B_2=01111000
A)	*Desp. Lógico a Derecha*	***000***11001	***000***01111
	Desp. Lógico a Izquierda	01000***000***	11000***000***
B)	*Desp. Aritmético a Derecha*	***111***11001	***000***01111
	Desp. Aritmético a Izquierda	01000***000***	11000***000***

Como puede apreciarse, en el desplazamiento aritmético hacia la derecha, hay una extensión del bit de signo.

El efecto de desplazar un número hacia la izquierda o derecha es el mismo que se obtiene al multiplicar o dividir el mismo por la unidad seguida de ceros 2^n (siendo "n" el número de ceros). La multiplicación produce un desplazamiento de "n" bits hacia la izquierda, mientras que la división un desplazamiento de "n" bits hacia la derecha. Esto lo podemos apreciar en el siguiente ejemplo.

Ejemplo 22: *Multiplicar y dividir al número entero A_2=11001000, por $2^3 = 1000_2$.*

MULTIPLICACIÓN

```
  1 1 0 0 1 0 0 0
×   0 0 0 1 0 0 0
-----------------
  0 0 0 0 0 0 0 0
+ 0 0 0 0 0 0 0
-----------------
  0 0 0 0 0 0 0 0
+ 0 0 0 0 0 0 0
-----------------
  0 0 0 0 0 0 0 0
+ 1 1 0 0 1 0 0 0
-----------------
  1 1 0 0 1 0 0 0 0 0 0 0
    Resultado
```

DIVISIÓN

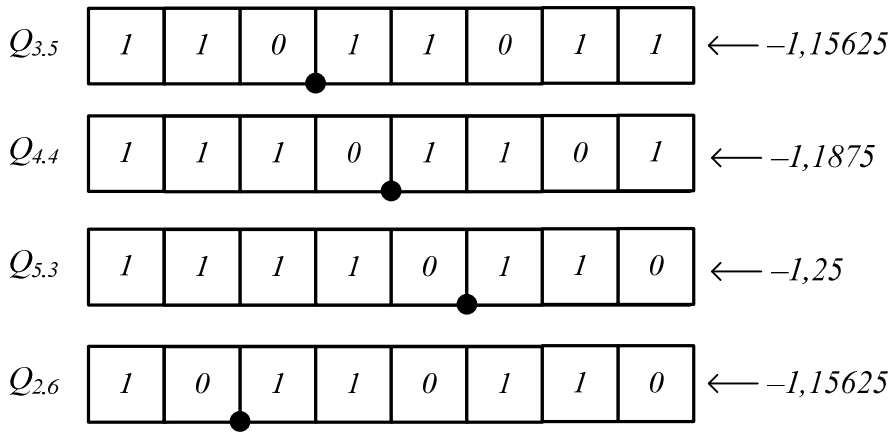
```
1 1 0 0 1 0 0 0   → 0 0 1 1 1 0 0 0
0 0 1 1 0 1 1 1   − 0 0 0 0 1 0 0 0
+           1     ------------------
-----------------   0 0 1 1 0 0 0 0 → 1
0 0 1 1 1 0 0 0   − 0 0 0 0 1 0 0 0
                  ------------------
                    0 0 1 0 1 0 0 0 → 1
                  − 0 0 0 0 1 0 0 0
                  ------------------
                    0 0 1 0 0 0 0 0 → 1
                  − 0 0 0 0 1 0 0 0
                  ------------------
                    0 0 0 1 1 0 0 0 → 1
                  − 0 0 0 0 1 0 0 0
                  ------------------
                    0 0 0 1 0 0 0 0 → 1
                  − 0 0 0 0 1 0 0 0
                  ------------------
                    0 0 0 0 1 0 0 0 → 1
                  − 0 0 0 0 1 0 0 0
                  ------------------
                    0 0 0 0 0 0 0 0 → 1

0 0 0 0 0 1 1 1
        ↓
1 1 1 1 1 0 0 0
+           1
---------------
1 1 1 1 1 0 0 1
  Resultado
```

Con el ejemplo anterior puede comprobarse que la multiplicación y la división de un número por la unidad seguida de ceros, produce el desplazamiento aritmético del número hacia uno u otro extremo.

Considerando los números binarios fraccionarios, el desplazamiento aritmético permite convertir un número que se encuentra en un formato $Q_{M.K}$ a otro formato $Q_{S.P}$. El siguiente ejemplo muestra como realizar dicha conversión.

Ejemplo 23: *Pasar A_2=110.11011= –1,15625 del formato $Q_{3.5}$ a los formatos: $Q_{4.4}$; $Q_{5.3}$ y $Q_{2.6}$.*



Para efectuar este tipo de conversiones, debe tenerse en cuenta que el número pueda representarse dentro del formato Q deseado. Es decir, debe considerarse el rango determinado por la expresión [7]. En el ejemplo puede observarse que al disminuir la cantidad de bits "K" de la parte decimal, se produce mayor error en la representación del número.

BIBLIOGRAFÍA

- Ronald J. Tocci; "Sistemas digitales: Principios y Aplicaciones"; octava edición, año 2003.
- Mario C. Ginzburg; "Introducción a las técnicas digitales con circuitos integrados"; octava edición, año 1998.
- Hyeokho Choi; "Fixed Point Arithmetic"; año 2003.
  http://www.ict.kth.se/courses/IL2206/0506/docs/Choi_Fixed-Point.pdf
- Randy Yates; "Fixed-Point Arithmetic: An Introduction"; año 2001.
  http://personal.atl.bellsouth.net/y/a/yatesc/fp.pdf
- Erwan Simon; "Application Report SPRA588: Implementation of a speed field oriented control of 3-phase PMSM Motor using TMS320F240"; año 1999.
  http://www.datasheetarchive.com/datasheet-pdf/019/DSA00333179.html
- Erick L. Oberstar; "Fixed Point Representation and Fraccional Math"; año 2005.
  http://www.superkits.net/whitepapers/Fixed%20Point%20Representation%20&%20Fractional%20Math.pdf