**SISTEMAS NUMÉRICOS**

**REPRESENTACIÓN DE NÚMEROS BINARIOS CON SIGNO.**

**SUMA BINARIA:** Las operaciones de suma binaria se realizan de la siguiente forma:

| 0 | + | 0 | = | 0 |  |
| --- | --- | --- | --- | --- | --- |
| 0 | + | 1 | = | 1 |  |
| 1 | + | 0 | = | 1 |  |
| 1 | + | 1 | = | 0 | Llevo 1 |

**Ejemplo:**  Dado los números binarios:  W=1111100012; T=11011101012;   Obtener W+T

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

Matemáticas Binarias y Representaciones de Signos

Representar números con bits es una cosa. Realizar operaciones con ellos es un asunto completamente diferente. Este capítulo trata algunas de las operaciones matemáticas básicas que las computadoras realizan en números binarios junto con las representaciones binarias que soportan esas operaciones. Estos conceptos ayudarán a los programadores a comprender mejor las limitaciones de hacer cálculos matemáticos con un procesador y, por lo tanto, permitirles manejar mejor problemas como los límites superior e inferior de variables tipos, desbordamiento matemático y conversión de tipos.

**3.1 Suma binaria**

Independientemente del sistema de numeración, la suma de dos números con múltiples dígitos se realiza agregando los dígitos correspondientes de una sola columna juntos para producir un resultado de un solo dígito. Por ejemplo, 3 sumado a 5 usando el sistema de numeración decimal es igual a 8. El 8 es colocado en la misma columna del resultado de donde provienen los 3 y 5. Todos estos dígitos, 3, 5 y 8, existen en el sistema de numeración decimal,y por lo tanto puede permanecer en una sola columna.

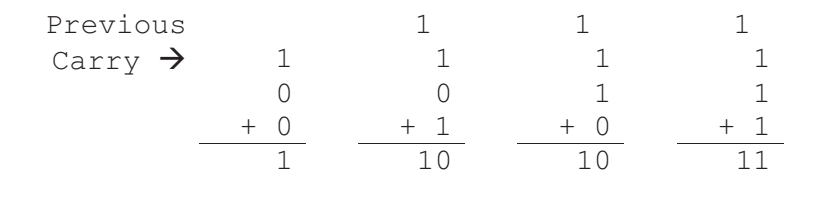
En algunos casos, el resultado de la adición en una sola columna puede ser más de 9, por lo que es necesario colocar un desbordamiento '1' o llevarlo al columna inmediatamente a la izquierda. Si sumamos 6 a 5, por ejemplo, obtenemos 11 que es demasiado grande para caber en un solo dígito decimal. Por lo tanto, 10 es restado del resultado dejando 1 como el nuevo resultado para esa columna. La resta de 10 se compensa colocando un carry en la siguiente columna más alta, el lugar de los diez. Otra forma de decir esto es que 6 sumado a 5 es igual a 1 con un acarreo de 1. Es importante tener en cuenta que la adición de dos dígitos en decimal nunca puede dar como resultado un valor mayor que 18. Por lo tanto, el transporte a la siguiente posición más alta nunca será más grande de 1.

La suma binaria funciona de la misma manera, excepto que estamos limitados a dos dígitos Tres de las operaciones de suma, 0 + 0, 0 + 1 y 1 + 0, dan como resultado 0 o 1, dígitos que ya existen en el sistema de numeración binario. Esto significa que no se necesitará llevar.

Sin embargo, agrega 1 a 1 da como resultado un decimal 2, un dígito que no existe en binario. En este caso, necesitamos crear un carry o overflow eso irá a la siguiente columna.

La siguiente posición de bit más alta representa 21 = 2. Tal como lo hicimos con decimal, restamos una instancia de la siguiente posición de bit más alta de nuestro resultado. En el caso de 1 + 1 = 2, restamos 2 de 2 y obtenemos 0.

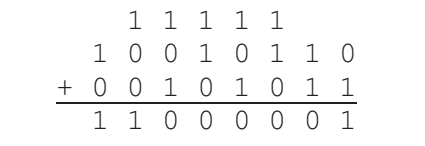
Por lo tanto, 0 es el resultado que se coloca en la columna actual, y el la resta de 2 se convierte en un acarreo a la siguiente columna. Por lo tanto, 1 + 1 en binario es igual a 0 con un acarreo de 1. Cada una de las posibles adiciones binarias de dos variables se muestra en la Figura 3-1.



**Figura 3-2** Cuatro resultados posibles de agregar dos bits con Acarreo.

El segundo y tercer caso son similares al último caso presentado en Figura 3-1 donde se suman dos 1 para obtener un resultado de 0 con un llevar. Sin embargo, el último caso en la Figura 3-2 tiene tres 1 sumados lo que equivale a 310. Restar 2 de este resultado coloca un nuevo resultado de 1 en la columna actual y envía un carry a la siguiente columna. Y justo como en suma decimal, el carry en binario nunca es mayor que 1.

Ahora intentemos agregar números binarios con varios dígitos. El ejemplo que se muestra a continuación presenta la adición de 100101102 y 001010112. Los valores resaltados son los acarreos de la anterior suma de la columna, y al igual que en la suma decimal, se agregan al siguiente dígito / bit más significativo.



**3.3 Complementos binarios**

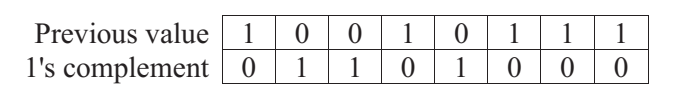
En aritmética decimal, cada número tiene un complemento aditivo, es decir, un valor que cuando se agrega al número original da como resultado un cero.

Por ejemplo, 5 y -5 son complementos aditivos porque 5 + (-5) = 0.

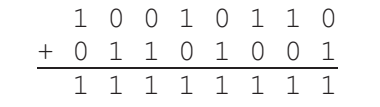
Esta sección describe los dos métodos principales utilizados para calcular el Complementos a un valor binario.

3.3.1 Complemento a uno

Cuando se le pide crear un patrón de unos y ceros que al agregar a un valor binario resulte cero, la mayoría de las personas responden con, "simplemente voltear cada bit en el valor original". Esta "inversión" de cada bit, sustituyendo 1 por todos los 0 y 0 por todos los 1, resulta en el Complemento a 1 del valor original. Un ejemplo se muestra a continuación.



El Complemento a 1 de un valor es útil para algunos tipos de funciones digitales, pero no proporciona muchos beneficios si está buscando el complemento aditivo. Vea lo que sucede cuando agregamos un valor a su Complemento a 1.



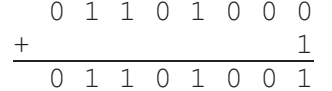
Si los dos valores fueran complementos aditivos, el resultado debería ser

cero, ¿verdad? Bueno, eso nos lleva al Complemento a 2.

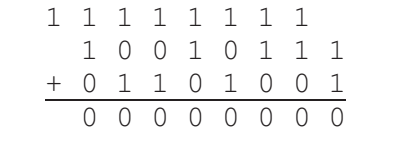
3.3.2 Complemento a dos

El resultado de agregar un número de n bits a su complemento es siempre un número de n bits con unos en cada posición. Si agregamos 1 a eso como resultado, nuestro nuevo valor es un número de n bits con ceros en cada posición y un desbordamiento o transporte a la siguiente posición más alta, la (n + 1)th columna que corresponde a 2n. Para nuestro ejemplo de 8 bits anterior, el el resultado de agregar 100101102 a 011010012 es 111111112. Agregar 1 a este número nos da 000000002 con un desbordamiento de 1 a la novena o 28 columnas. Si nos restringimos a 8 bits, este desbordamiento puede ser ignorado.

Esto nos da un método para encontrar el complemento aditivo, llamado representación del Complemento a 2. El Complemento a 2 de un el valor se encuentra primero tomando el Complemento a 1, luego incrementando ese resultado en 1. Por ejemplo, en la sección anterior, determinamos que el Complemento a 1 de 100101112 es 011010002. Si agregamos 1 a esto valor, obtenemos:



Por lo tanto, el Complemento a 2 de 100101112 es 011010012. Veamos qué sucede cuando intentamos agregar el valor al Complemento a 2.

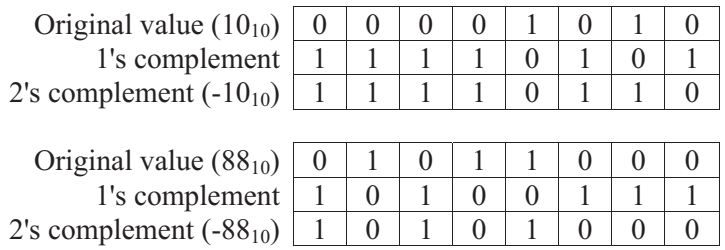


¡El resultado es cero! Bien, la mayoría de ustedes captaron el hecho de que yo no he desplegado el último acarreo que habría hecho el resultado 1000000002.

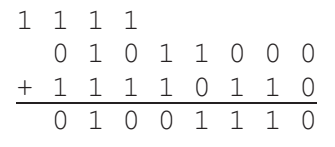
Esto no es un problema, porque en el caso de la aritmética con signo, el acarreo tiene un propósito diferente al de agregar un dígito adicional representando el próximo poder de dos. Siempre y cuando nos aseguremos de que dos números que se agregan tienen el mismo número de bits, y que nosotros mantenga el resultado en el mismo número de bits, luego cualquier acarreo que vaya más allá de eso debe descartarse.

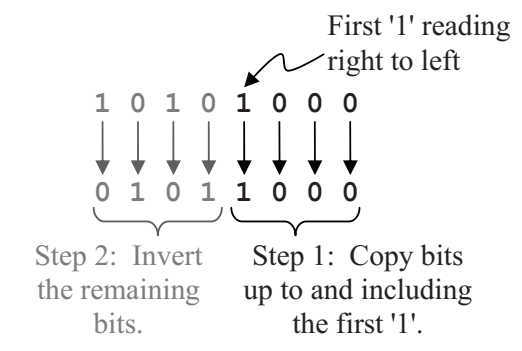
En realidad, descartado no es el término correcto. En algunos casos le haremos uso al acarreo como una indicación de un posible error matemático. Sin embargo, no se incluirá en el resultado de la adición. Esta es simplemente la primera de muchas "anomalías" que deben observarse cuando se trabaja con un número limitado de bits.

A continuación se muestran dos ejemplos más de Complementos a 2.



Ahora veamos si la representación del Complemento a 2 se destaca cuando se debe realizar sumas. Si 8810 = 010110002 y -1010 = 111101102, entonces la suma de estos dos números debe ser igual a 7810 = 010011102.





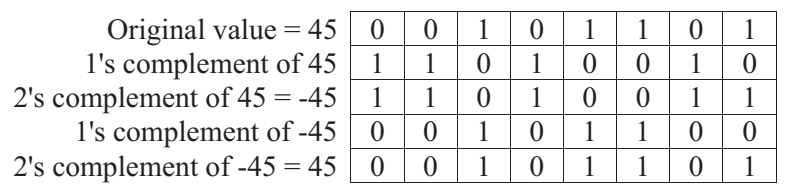
**Figura 3-3** Simplificación del Complemento a dos

Este resultado coincide con el resultado del ejemplo anterior.

En decimal, el negativo de 5 es -5. Si tomamos el negativo una segunda vez, volvemos al valor original, por ejemplo, el negativo de -5 es 5. ¿Es lo mismo para tomar el Complemento a 2 de un Complemento a 2 de un número binario? Bien, veamos.

El valor binario para 4510 es 001011012. Observe lo que sucede cuando

tomamos el complemento a 2 dos veces.



¡Funcionó! La segunda vez que se tomó el Complemento a 2, el patrón de unos y ceros volvió a sus valores originales. Resulta que esto es cierto para cualquier número binario de un número fijo de bits.

3.3.3 Bit más significativo como indicador de signos

Como se dijo anteriormente, el complemento a 2 se utiliza para permitir que la computadora

pueda representar el complemento aditivo de un número binario, es decir, números negativos. Pero hay un problema. Como hemos mostrado anteriormente en esta sección,

tomando el Complemento a 2 de 4510 = 001011012 nos da -4510 = 110100112. Pero con lo visto hasta el momento, el valor de ocho bits 110100112 mostró ser igual a 27 + 26 + 24 + 21 + 20 = 128 + 64 + 16 + 2 + 1 = 21110. Así pues, ¿acabamos de probar que -4510 es igual a 21110? O tal vez 001011012 es en realidad -21110.

Resulta que cuando se utiliza el Complemento a 2 de la representación binaria, la mitad de los patrones de bits binarios deben perder su asociación positiva para representar números negativos. ¿Así 110100112 es -4510 o 21110?

Resulta que 110100112 es uno de los patrones de bits destinados a representar un número negativo, por lo que en la notación del Complemento a 2, 110100112 = -4510.

Pero cómo podemos decir si un patrón de bits binarios representa un positivo o

un número negativo?

De la descripción anterior del Complemento a 2 simplificado, usted puede ver que a excepción de dos casos, el MSB de los Complemento a 2 es siempre el inverso del valor original. Los dos casos donde esto no es verdad es cuando todos los bits del número excepto el bit más significativo igual a 0 y el bit más significativo es un 0 o un 1. En ambos casos, el Complemento a 2 es igual al valor original.

En todos los demás casos, cuando aplicamos el método simplificado siempre encontrará un 1 antes de llegar a la MSB al leer de derecha a izquierda.

Puesto que cada pedacito después de este será invertido, entonces el más significativo debe ser invertido alternando su valor original. Si el original tiene un cero en el MSB, entonces su complemento de 2 debe tener un uno y viceversa. Debido a esta característica, el MSB de un valor se utilizará para indicar si un número es positivo o negativo y llamado un poco señal.

Un valor binario con un 0 en la posición MSB se considera positivo y un valor binario con un 1 en la posición MSB se considera negativo.

Esto hace que sea vital para declarar el número de bits que un binario utiliza. Si esta información no se da, entonces el ordenador o el usuario que mira un número binario no sabrá qué bit es el MSB.

Dado que el MSB se está utilizando para indicar el signo de un número binario, no se puede utilizar para representar una potencia de 2, es decir, si un número dice que representa un valor complementario a 2, sólo n-1 de sus n bits puede ser utilizado para determinar la magnitud, ya que el MSB se utiliza para el signo.

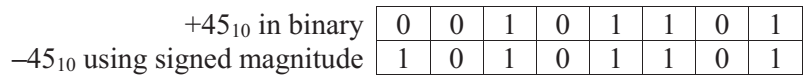
Esto reduce a la mitad el número de enteros positivos con n bits que pueden representar.

¿Y los casos especiales? Bueno, un número binario con todos los ceros es igual a un decimal 0. Tomando el negativo de cero todavía nos da cero. El otro caso es un poco más complicado. En la sección de mínimos y máximos, se verá que un valor de n-bit con un MSB igual a uno y todos los demás bits igual a cero es un número negativo, específicamente, -2 (n-1). El mayor número positivo representado en el complemento de 2 tiene un MSB de 0 con todos los bits restantes ajustados a uno. Este valor es igual a 2 (n-1) - 1. Por lo tanto, ya que 2 (n-1) > 2 (n-1) - 1, podemos ver que no hay equivalente positivo a el número binario 100… 002.

3.3.4 Signo-Magnitud

Una segunda forma, menos útil para representar números binarios positivos y negativos es tomar el MSB y usarlo como un bit de signo, al igual que un plus o menos señal, y dejar los bits restantes para representar la magnitud.

La representación se llama representación de signo-magnitud. Por ejemplo, -45 y +45 serían idénticos en binario excepto para el MSB que se establecería en un 1 para -45 y un 0 para +45. Esto se muestra a continuación para una representación de 8 bits.



3.3.5 MSB y número de bits

Dado que el MSB es necesario para indicar el signo de un valor binario, es vital que sepamos cuántos bits está siendo representado un número en particular con lo que sabremos exactamente dónde está la MSB. En otras palabras, los ceros principales de un valor binario pueden haber sido eliminados haciendo que parezca que el valor binario es negativo ya que comienza con un uno.

Por ejemplo, si se asume que el valor binario 100101002 es de 8 bits número usando la representación del Complemento a 2, luego convirtiéndolo a decimal nos daría -10810. (Discutiremos la conversión decimal más adelante en este capítulo. ) Si, sin embargo, era un número de 10-bits, entonces el MSB sería 0 y se convertiría en el positivo valor 14810.

3.3.6 Cuestiones que rodean la conversión de números binarios

Dado que las computadoras no utilizan un número infinito de bits para representar valores, el software debe saber dos cosas antes de que pueda interpretar un valor binario: el número de bits y el tipo de representación binaria que se va a utilizar. Esto suele ser confuso para el principiante.

Identificar 101001102 como un número de 8 bits no es suficiente. Tenga en cuenta que el

MSB es igual a 1. Por lo tanto, este valor representa un número en binario sin signo, otro número en el Complemento a 2, y un tercero en signo-magnitud.

Primero, hagamos la conversión de 101001102 asumiendo que es un 8-bit, binario sin signo.

101001102 = 27 + 25 + 22 + 21 = 128 + 32 + 4 + 2 = 16610

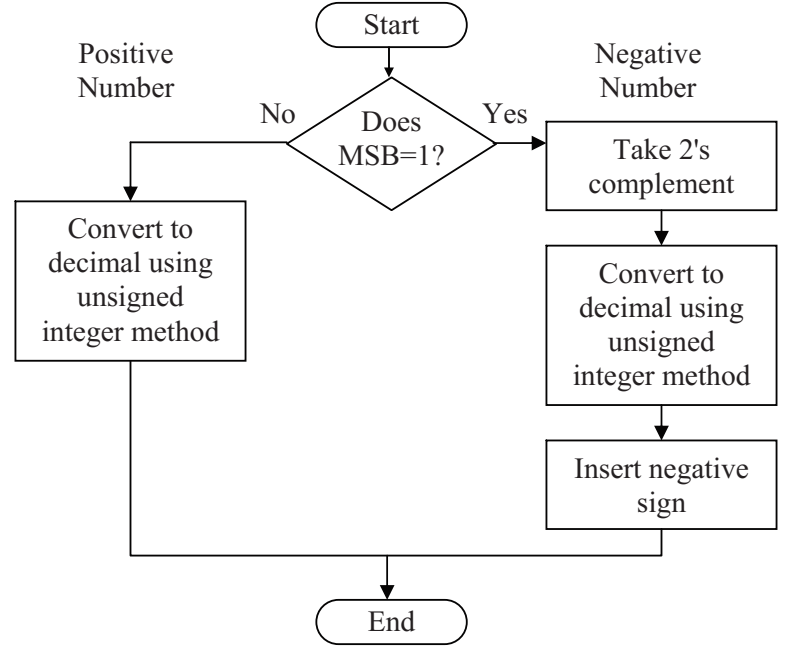
Ahora vamos a hacer la conversión en complemento a 2. Antes de hacerlo, vamos a examinar el proceso. Primero, si el MSB es igual a 0, entonces el valor es un número positivo. En la notación del Complemento a 2, números positivos parecen binarios sin signo y deben ser tratados exactamente igual cuando se realiza una conversión al decimal.

Si el MSB es igual a 1, entonces el valor representado por este patrón de unos y ceros es negativo. Para convertirlo en un número negativo, alguien tuvo que aplicar el proceso de tomar los Complementos a 2 al valor positivo original. Por lo tanto, debemos eliminar el signo negativo antes de hacer la conversión.

Se mostró anteriormente como una segunda aplicación que el Complemento a 2 de los 2 procesos de conversión devuelve el número a su positivo original valor. Si toma el Complemento a 2 de un número negativo vuelve a su valor positivo, entonces el valor positivo se puede convertir a decimal usando el mismo proceso usado para un valor binario sin signo. Añadiendo un signo negativo al resultado decimal completa la conversión. La figura 3-4 presenta un diagrama de flujo que muestra este proceso gráficamente.

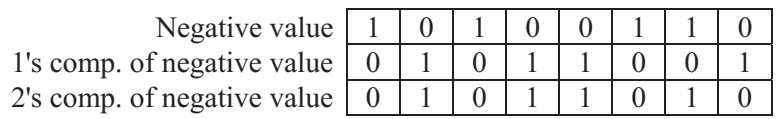
Un segundo método para convertir el valor del Complemento a 2 de n-bit a

decimal es realizar la conversión como lo haría un binario sin signo excepto que el dígito MSB se trata como -2(n-1) en lugar de 2(n-1). Por ejemplo, el MSB de un valor complementario de 8-bit 2 representaría -27 = -128.



**Figura 3-4** Convertir un número de complemento de dos a un decimal

En el caso de 101001102, el MSB es un 1. Por lo tanto, es un número negativo. Siguiendo la rama derecha del diagrama de flujo en la Figura 3-4, vemos que debemos tomar el complemento de los dos para encontrar el positivo contrapartida de nuestro número negativo.



Ahora que tenemos la contraparte positiva para el complemento de los 2 valores del número negativo 101001102, lo convertimos a decimal sólo como hicimos con el valor binario sin signo.

010110102 = 26+ 25+ 23+ 21= 64 + 16 + 8 + 2 = 9010

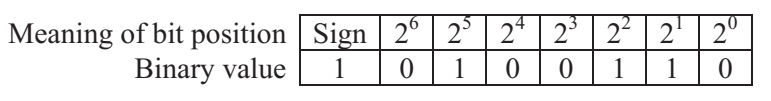
Dado que el valor del Complemento a 2 original fue negativo para empezar, el valor 101001102 en forma de Complemento a 2 de 8-bit, es -90.

Podemos duplicar este resultado usando el segundo método de conversión, es decir, convertir 101001102 utilizando el método binario sin signo mientras el MSB lo tomamos como -27. En este caso, hay un 1 en el -27, 25, 22, y 21 posiciones.

101001102 = -27 + 25 + 22 + 21 = –128 + 32 + 4 + 2 = –9010

A continuación, vamos a hacer la conversión suponiendo 101001102 está en 8-bit Signo-Magnitud donde el MSB representa el bit de signo. Al igual que en Complemento a 2, el MSB en 1 significa que el número es negativo.

La conversión de un número binario de Signo-Magnitud a decimal es diferente del Complemento a 2. En el caso de Signo-Magnitud, eliminamos el MSB y convertimos los bits restantes utilizando los mismos métodos de convertir binario sin signo a decimal. Cuando esté hecho, colocar un negativo delante del resultado decimal sólo si el MSB es igual a 1.



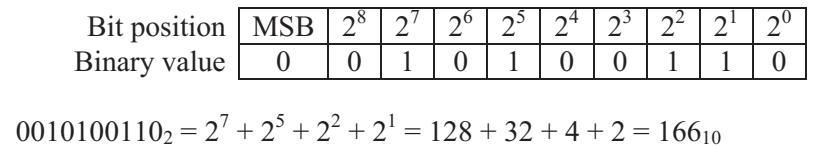
Para convertir este valor a un número positivo, elimine el bit de signo.

A continuación, calcule la magnitud tal como lo haríamos para el caso sin signo.

01001102 = 25 + 22 + 21 = 32 + 4 + 2 = 3810

Dado que el MSB del valor original fue igual a 1, el Signo-Magnitud era un número negativo para empezar, y tenemos que añadir un signo negativo. Por lo tanto, 101001102 en 8-bit, en Signo-Magnitud se representa igual a -3810.

Pero ¿qué pasa si este número binario era en realidad es un número de 10 bits y no un número de 8 bits? Bueno, si es un número de 10 bits (00101001102), el MSB es 0 y por lo tanto es un número positivo. Esto hace que nuestra conversión sea mucho más fácil. El método para convertir un valor binario positivo a un valor decimal es el mismo para las tres representaciones. La conversión dice algo como esto:



Esta discusión muestra que es posible para un patrón binario de unos y ceros pueda tener tres interpretaciones. Todo depende de cómo se le ha dicho a la computadora que interprete el valor.

En un lenguaje de programación como C, la forma en que un ordenador trata una variable depende de cómo se declare. Variables declaradas como int sin signo se almacenan en notación binaria sin signo. Variables declaradas como int se tratan como complemento de 2 o signo-magnitud dependiendo del procesador y/o compilador.

3.3.7 Mínimos y Máximos

Cuando se utiliza un número finito de posiciones de bits para almacenar información, es necesario poder determinar los valores mínimos y máximos que cada representación binaria puede manejar. Un fallo al hacer esto podría resultar en errores en el software que se crea. Esta sección calcula el mínimo y valores máximos para cada una de las tres representaciones discutidas en este y el capítulo anterior usando un número fijo de bits, n.

Vamos a empezar con la representación más básica, binario sin signo. El valor más pequeño que se puede representar con binario sin signo se produce cuando todos los bits son iguales a cero. Conversión desde binario a decimal en 0 + 0 + ... + 0 = 0. Por lo tanto, para un número de n bits:

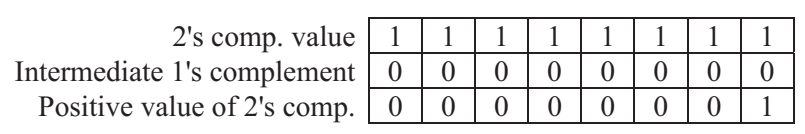
Número binario sin signo de n-bit mínimo = 0

El valor más grande que se puede representar con binario sin signo se alcanza cuando todos los n bits son iguales. Cuando convertimos este valor de binario a decimal, obtenemos 2n-1 + 2n-2 + ... + 20. Como era se muestra en el capítulo 2, añadiendo uno a esta expresión resulta en 2n.

Por lo tanto, para un número binario sin signo de n-bit, el máximo es:

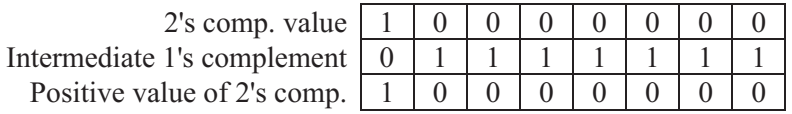
Número binario máximo n-bit sin signo = 2n-1

A continuación, vamos a examinar los valores mínimos y máximos para representación de complemento a 2 con n-bit. A diferencia del caso sin signo, el más bajo valor decimal que se puede representar con n-bits en representación de complemento a 2 no es obvio. Recuerde, Complemento a 2 utiliza el MSB como bit de signo. Dado que el valor más bajo será negativo, el MSB se debe establecer en 1 (un valor negativo). Pero ¿que se debe hacer con todos los bits restantes? Una inclinación natural es establecer todos los bits después de el MSB a uno. Este debe ser un gran número negativo, ¿verdad? Bueno, convertirlo a decimales resulta algo como el ejemplo de 8 bits de abajo:



Esto no es exactamente lo que esperábamos. Usando el método de complemento a 2 para convertir 1111112 a un número decimal resulta en -110. Esto no podría ser el valor más bajo que se puede representar con complemento a 2.

Resulta que el valor más bajo posible de Complemento a 2 es el MSB de 1 seguido de todos los ceros como se muestra en el ejemplo de 8 bits a continuación. Para la conversión al trabajo, debe seguir estrictamente la secuencia presentada en Figura 3-4 para convertir el valor del complemento de un 2 negativo a decimal.



Conversión del valor positivo a decimal utilizando el método sin signo muestra que 100000002 = -27 = -128. Traducir esto a n-bits nos da:

El valor máximo es un poco más fácil de encontrar. Es un número positivo, es decir, una MSB de 0. Los restantes n-1 bits se tratan como sin firmar representación de magnitud. Por lo tanto, para n bits:

Número máximo del complemento a 2 de n-bit = 2(n-1) - 1

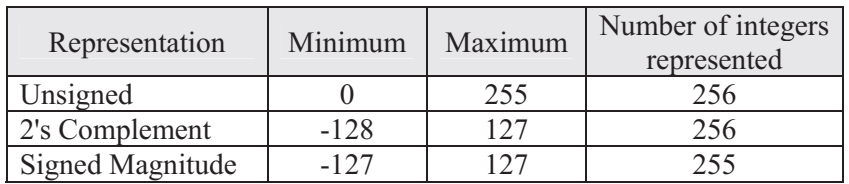
Por último, tenemos la representación de magnitud firmada. Para determinar la magnitud de un valor de Signo-Magnitud, ignorar la MSB y utilizar los bits n-1 restantes para convertir a decimales como si estuvieran en representación sin signo. Esto significa que los valores más grandes y más pequeños representados con un número de magnitud firmado n-bit es igual al positivo y valores negativos de un número binario sin signo (n-1)-bit.

Número de signo-magnitud mínimo de n-bit = -(2(n-1)- 1)

Número de signo-magnitud máximo de n-bit = (2(n-1)- 1)

Por ejemplo, en el cuadro 3-1 se comparan el mínimo y el máximo valor de un número de 8 bits para cada una de las representaciones binarias. La última columna muestra el número de valores enteros distintos posible con cada representación. Por ejemplo, hay 256 valores enteros entre 0 y 255, lo que significa que la representación binaria sin signo de 8 bits tiene 256 posibles combinaciones de 1 y 0, cada una de las cuales representa un diferente número entero en el rango.

**Tabla 3-1** Comparación de representación para números binarios de 8 bits



Así que ¿por qué signo-magnitud con 8-bit sólo puede representar 255 en lugar de 256? Es porque en la signo-magnitud 000000002 y 100000002 ambos representan el mismo número, un decimal 0.