UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Representación de números enteros: el convenio “complemento a dos”

Apellidos, nombre	Martí Campoy, Antonio (amarti@disca.upv.es)
Departamento	Informàtica de Sistemes i Computadors
Centro	Escola Tècnica Superior d’Enginyeria Informàtica

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

1 Resumen de las ideas clave

En este artículo se trata la problemática de la representación de los números enteros en los computadores. Así mismo, se presentará una posible solución a este problema, que recibe el nombre de representación en complemento a dos. Los conocimientos previos que necesitas para abordar este artículo se presentan en la tabla 1.

Tabla 1. Conocimientos previos

Conocimientos previos
1. Sistemas de numeración posicionales
2. Sistema de numeración binario
3. Cambios de base, especialmente binario
4. Aritmética básica en base 2

2 Objetivos

Una vez acabes de leer este artículo docente y reproduzcas los ejemplos presentados, deberás ser capaz de **representar** números enteros en binario **aplicando** el convenio llamado complemento a dos. Además podrás **calcular** el rango de representación para un tamaño de bits determinado. También serás capaz de **realizar** operaciones aritméticas de suma y resta de números enteros en binario y de extensión de signo utilizando la representación en complemento a dos. Por último, podrás **razonar** sobre las ventajas y desventajas de este convenio de representación de números enteros.

3 Introducción

En la vida cotidiana los números enteros se representan mediante los 10 símbolos (del 0 al 9) de la base decimal, junto con los símbolos "+" y "-" para identificar a los números positivos y negativos, respectivamente.

A la hora de representar números enteros en un computador (para almacenarlos, operarlos o comunicarlos) el problema que surge es que en los circuitos digitales sólo se pueden utilizar dos valores, normalmente representados por los símbolos 0 y 1. No cabe la posibilidad de representar un tercer y cuarto símbolo para distinguir un número positivo de otro negativo.

Así surge la necesidad de crear y definir convenios para codificar el signo de un número entero utilizando únicamente los símbolos 0 y 1 disponibles en los circuitos digitales.

Antes de explicar el convenio complemento a dos, objeto de este artículo, recordarte que los números se almacenan en circuitos digitales llamados registros,

y que su longitud es fija. Es decir, cuando hablemos de un número entero representado en binario y en complemento a dos deberemos indicar el número total de bits utilizados.

4 El convenio complemento a dos

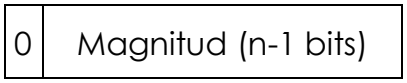
El nombre de este convenio se debe a que se utiliza la operación aritmética de complemento a dos para representar los números negativos. Por ello es muy **importante** que no confundas la operación aritmética de complemento a dos (le **hacemos** el complemento a dos a un número) con la representación de un número entero **en** complemento a dos (codificamos o representamos un número siguiendo el convenio).

También podrás encontrar referencias a este convenio como Ca2 y C'2, entre otras.

4.1 Definición

En este convenio se diferencia la forma en que se representa un número entero positivo de uno negativo. Consideramos que utilizamos n bits para representar los números enteros. El convenio, un acuerdo arbitrario, dice que:

- Si el número es **positivo** se representa su magnitud con n-1 bits, y se añade un 0 a la izquierda:



- Si el número es **negativo** se procede de la siguiente manera. Se representa su equivalente positivo, utilizando n-1 bits para la magnitud y añadiendo un cero a la izquierda, tal como se ha descrito anteriormente. Una vez tenemos la representación del equivalente positivo, le hacemos el complemento a dos, y esta será la representación del número negativo:

Dos cosas importantes. La primera es recordarte que el Ca2 de un número se puede calcular o realizar invirtiendo sus bits, es decir, cambiando unos por ceros y ceros por unos, luego sumando uno y descartando el acarreo final.

La segunda cosa importante es que al utilizar la representación **en** complemento a dos el bit de mayor peso, el de la izquierda, indica el signo del número y recibe el nombre de **bit de signo**.

Ejemplo: utilizando 8 bits (n = 8), representa el número +21 siguiendo el convenio de complemento a dos.

En primer lugar se convierte la magnitud o valor absoluto, 21, a binario natural con n-1=7 bits, completando con ceros los bits de mayor peso si fuera necesario:

$$21_{10} = 0010101_2$$

Como el número que queremos representar es positivo, se añade un cero a la izquierda:

$$+21_{10} = \mathbf{0}0010101_2$$

Ejemplo: utilizando 8 bits (n = 8), representa el número –26, siguiendo el convenio de complemento a dos.

Dado que el número a representar es negativo, necesitamos obtener en primer lugar la representación de su equivalente positivo. Convertimos la magnitud o valor absoluto, 26, a binario natural con n-1=7 bits, completando con ceros los bits de mayor peso si fuera necesario:

$$26_{10} = 0011010_2$$

Ahora añadimos un cero a la izquierda para obtener la representación de +26:

$$+26_{10} = \mathbf{0}0011010_2$$

Como lo que realmente queremos representar es -26, que es negativo, le hacemos el complemento a dos a +26:

$$\text{Ca2}\,(+26_{10}) = \text{Ca2}\,(\mathbf{0}0011010_2) = \mathbf{1}1100101_2 + 1 = \mathbf{1}1100110 = -26_{10}$$

(recuerda, al hacer la suma en el Ca2 descarta el acarreo final)

Ejemplo: obtén el valor decimal correspondiente a 010010_2 y 101000_2 sabiendo que están representados complemento a dos utilizando 6 bits (n = 6).

$010010_2 \rightarrow$ dado que su bit de mayor peso (bit de signo) es 0, sabemos que se trata de un número positivo. Siguiendo el convenio, retiramos el cero de mayor peso y nos queda la magnitud $10010_2 = 18_{10}$ por lo que $010010_2 = +18_{10}$

$101000_2 \rightarrow$ dado que su bit de mayor peso (bit de signo) es 1, sabemos que se trata de un número negativo. En este caso aprovechamos que la operación de complemento a dos es reversible (Ca2(Ca2(x)) = x) para obtener el equivalente positivo, y de este modo poder obtener la magnitud:

$$\text{Ca2}(101000_2) = 010111_2 + 1 = 011000_2 = +24_{10}$$

Eliminamos el bit de signo y queda la magnitud $11000_2 = 24_{10}$

Por lo que en complemento a dos tenemos que $101000_2 = -24_{10}$

Quiero que recuerdes dos cosas importantes. Una, que no sabemos leer números negativos representados en complemento a dos y por eso tenemos que encontrar el equivalente positivo, que sí sabemos leer. Y dos, que en ese proceso no debemos olvidar indicar el signo al final de la conversión.

4.2 Rango

El rango de un sistema o convenio de representación es el conjunto de valores diferentes que pueden representarse. Estudiaremos el rango de forma separada para los números positivos y negativos:

Positivos		Negativos	
$000\cdots000_2$	+0	−0	$000\cdots000_2$
$000\cdots001_2$	+1	−1	$111\cdots111_2$
$011\cdots110_2$	$+(2^{n-1}-2)$	$-(2^{n-1}-1)$	$100\cdots001_2$
$011\cdots111_2$	$+(2^{n-1}-1)$	$-(2^{n-1})$	$100\cdots000_2$

En esta tabla hay varias cosas interesantes. Para verlo más fácilmente vamos a ver unos ejemplos con 4 bits:

Positivos		Negativos	
0000_2	+0	−0	0000
0001_2	+1	−1	1111_2
0110_2	+6	−7	1001_2
0111_2	+7	−8	1000_2

Estas son las cosas a las que has de prestar atención:

- Sólo hay un cero y es positivo. Si hacemos Ca2(0000) = 1111 + 1 = 0000 (recuerda, el acarreo final se descarta). Es decir, el equivalente negativo del cero es él mismo.
- El número 1000 es negativo, y si buscamos su equivalente positivo nos encontramos que Ca2(1000) = 0111 +1 = 1000, que vuelve a ser negativo. Cuando sepas sumar será fácil demostrar que este valor corresponde con -8. Y si hablamos de n bits, corresponde con $-(2^{n-1})$
- Los números negativos no están ordenados de la misma forma que si fueran naturales.

Con todo lo anterior podemos concluir que el rango de representación para n bits en complemento a dos es:

$$\begin{array}{ll} \text{Rango en binario:} & [100\cdots000,\quad 111\cdots111,\quad 000\cdots000,\quad 011\cdots111] \\ \text{Rango en decimal:} & [-(2^{n-1}),\quad -1,\quad 0,\quad +(2^{n-1}-1)] \end{array}$$

El rango es asimétrico, es decir, incluye un valor más para los números negativos que para los positivos, y sólo incluye una representación para el cero. También, como hemos dicho anteriormente, los números negativos no están en el orden "natural", lo que complica su interpretación por parte de los humanos.

4.3 Suma y resta

La operación de suma de números representados en complemento a dos se realiza usando las reglas de suma de binario natural, independientemente del signo de los operandos y **descartando** el acarreo final. Es decir, da igual que se sumen dos positivos o dos negativos, o un positivo y un negativo, simplemente se

suman. Y además, el resultado de la suma se encuentra representado en complemento a dos. Esta es la principal ventaja de este convenio de representación y la razón de que el 100% de los sistemas informáticos lo utilicen para la representación de números enteros.

La operación de resta se realiza mediante una suma, a la que se le cambia el signo al sustraendo. Es decir, A - B = A + (-B). Y como hemos visto antes, cambiar el signo a un número representado en complemento a dos es muy sencillo, se consigue haciéndole el complemento a dos.

A continuación se muestra como ejemplo una suma y una resta de números representados en binario complemento a dos con 4 bits (y sus equivalentes en decimal):

Binario Ca2 con 4 bits	0100 + 1101 **0001**	El acarreo final es 1, pero se descarta	
Decimal	+4 + −3 +1		
Binario Ca2 con 4 bits	1101 − 0010	Hacemos Ca2(0010) y se convierte en suma	1101 + 1110 **1011**
Decimal	−3 − +2 −5	Ca2(**1011**) = 0100+1 = 0101 **1011** = -5_{10}	

4.4 Desbordamiento en la suma y resta

Al realizar una suma de números enteros es posible que el resultado exceda el rango de representación. En este caso se dice que no hay resultado o que el resultado no es representable.

Con operandos representados en complemento a dos se produce desbordamiento al realizar una suma si el último y el penúltimo acarreo son distintos. Una simple puerta lógica or-exclusiva (xor) permite detectar la condición de desbordamiento. A continuación tienes un ejemplo donde se produce desbordamiento. En este ejemplo los acarreos aparecen en letra cursiva:

Binario Ca2 con 4 bits	*1000* 1101 + 1010 0111	Se produce desbordamiento y no hay resultado	¿-3 + -9 = +7?

4.5 Extensión de signo

En algunos casos es necesario operar datos con diferentes tamaños. Para aumentar el número de bits con que se representa un dato se realiza la operación llamada extensión de signo.

En el caso de la representación en complemento a dos, la extensión de signo se realiza replicando el bit de signo.

Ejemplo: dados los números enteros 0010_2 y 1110_2 representados en complemento a dos con 4 bits, extiende el signo para representarlos con 8 bits.

$$\mathbf{0}010_2 \quad = \quad \mathbf{00000}010_2$$
$$\mathbf{1}110_2 \quad = \quad \mathbf{11111}110_2$$

5 Ejercicios

A continuación tienes unos pocos ejercicios. Es muy conveniente que cojas lápiz y papel y los resuelvas. Recuerda que estas aprendiendo, por lo que puedes, y aún diría más, debes consultar las secciones anteriores de este documento para resolver los ejercicios. También tienes las soluciones de los ejercicios, pero te pido encarecidamente que no las mires hasta que no hayas intentado resolver todos los ejercicios

5.1 Enunciados

1. Representa el número -66_{10} en binario complemento a dos con 8 bits.
2. Representa el número $+99_{10}$ en binario complemento a dos con 8 bits.
3. Indica la representación decimal de 10110001_2 sabiendo que está representado en complemento a dos con 8 bits.
4. Indica la representación decimal de 00101001_2 sabiendo que está representado en complemento a dos con 8 bits.
5. ¿Cuál es el rango de representación en complemento a dos con 8 bits? Expresa el rango en decimal.
6. Dados los números enteros representados en complemento a dos con 8 bits A = 10000101_2 y B = 01011011_2, realiza las operaciones A + B, A – B y B – A, indicando si el resultado es correcto o no.
7. Realiza la extensión de signo a 16 bits de 11000101_2 sabiendo que está representado en complemento a dos con 8 bits.
8. Realiza la extensión de signo a 16 bits de 01110110_2 sabiendo que está representado en complemento a dos con bits.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

5.2 Soluciones

1. Representa el número -66_{10} en binario complemento a dos con 8 bits. Sol: 10111110_2
2. Representa el número $+99_{10}$ en binario complemento a dos con 8 bits. Sol: 01100011_2
3. Indica la representación decimal de 10110001_2 sabiendo que está representado en complemento a dos con 8 bits. Sol: -79_{10}
4. Indica la representación decimal de 00101001_2 sabiendo que está representado en complemento a dos con 8 bits. Sol: $+41_{10}$
5. ¿Cuál es el rango de representación en complemento a dos con 8 bits? Expresa el rango en decimal. Sol: $[-2^7 , +(2^7 - 1)] = [-128 , +127]$
6. Dados los números enteros representados en complemento a dos con 8 bits A = 10000101_2 y B = 01011011_2, realiza las operaciones A + B, A – B y B – A, indicando si el resultado es correcto o no. Sol: A + B = 11100000; A -- B = desborda; B –A = desborda.
7. Realiza la extensión de signo a 16 bits de 11000101_2 sabiendo que está representado en complemento a dos con 8 bits. Sol: 1111111111000101_2
8. Realiza la extensión de signo a 16 bits de 01110110_2 sabiendo que está representado en complemento a dos con bits. Sol: 0000000001110110_2

6 Conclusiones

Los circuitos digitales sólo pueden almacenar dos símbolos, por lo que es necesario establecer un acuerdo o convenio para utilizar estos dos símbolos, el 0 y el 1, para representar el signo de un número entero. El convenio llamado representación en complemento a dos es sencillo y presenta una aritmética muy sencilla y útil. Construir un único circuito sumador/restador para números representados en complemento a dos es muy sencillo. Esta es una de las razones por la que es el convenio de representación de enteros utilizado en todos los computadores modernos.

Recuerda, **representar** en complemento a dos no significa **hacerle** el complemento a dos a todos los números, sólo a aquellos que indica el convenio, los negativos.

7 Bibliografía

7.1 Libros:

[1] Pedro de Miguel Anasagasti. "*Fundamentos de los computadores*", 9ª ed. Madrid, Thomson-Paraninfo. 2004, 2007

[2] John F. Wakerly. "*Diseño digital : principios y prácticas*". Madrid. Pearson Educación. 2001

7.2 Recursos electrónicos:

[3] Martí Campoy, Antonio. "*Representación de enteros: Complemento a 2*", Universitat Politècnica de València, 2009. http://hdl.handle.net/10251/5234