Sistemas de numeración para números enteros

Organización de computadoras 2018

Universidad Nacional de Quilmes

En apuntes anteriores vimos como utilizar binario para representar números naturales. En este escrito veremos como trabajar con números enteros. En decimal, solemos utilizar el signo "−"para indicar que un número es negativo, pero este no es representable en las computadoras como tal, donde solo se cuenta con 0s y 1s. Veremos entonces, como salvar esto de distintas maneras, cada una con sus particularidades.

1. Signo - Magnitud

La idea detras de este sistema es suplir la incapacidad de escribir el signo '−' indicando de alguna forma si el mismo está presente o no. Obviamente, para eso vamos a utilizar la herramienta con la que disponemos: un bit.

Por convención se suele utilizar que si el primer bit de una cadena (el de mas a la izquierda) es un 1, la cadena esta representando a un numero negativo, y si es un cero, la cadena representa a un número positivo. Este bit se denomina **signo**. Los bits restantes reciben el nombre de **magnitud** y su valor se determina con el mecanismo del sistema binario sin signo(BSS).

Por lo anterior, este sistema recibe el nombre **Signo-Magnitud** (SM). Cuando acotamos la cantidad de bits a n, se lo denota **SM(n)**, donde el primer bit es el signo, y la magnitud es de $n - 1$ bits.

1.1. Interpretar

Para interpretar una cadena en Signo Magnitud se utiliza la interpretación del sistema Binario Sin Signo sobre los bits de la magnitud.

Por ejemplo, si tenemos una cadena que es `1010`, para interpretarla lo que hacemos es separar el primer bit, en este caso `1` por un lado; y el resto de la cadena, `010`, por el otro.

El primer bit lo interpretamos según lo que planteamos en el parrafo anterior: `1` es negativo, `0` es positivo. En el ejemplo, la cadena comienza con 1 así que su signo va a ser $-$.

El resto de la cadena, lo interpretamos como BSS.

$$I(010) = 0 * 2^0 + 1 * 2^1 + 0 * 2^2 = 2$$

Luego, uniendo ambas partes, el resultado es -2.

1.2. Representar

Dualmente, la representación en signo magnitud se apoya sobre el mecanismo de representación del sistema Binario Sin Signo, pero este último permite representar sólo números positivos, por lo que es nncesario **tomar el valor absoluto del valor dado para representar**.

Supongamos por ejemplo que queremos representar el número -5 en SM(4), es decir 1 bit para el signo y 3 para la magnitud.

Lo primero que tenemos que hacer es definir el signo, y como en este caso el número es negativo el valor del bit de signo que va a tener la cadena resultante es **1**. Luego tomamos el valor absoluto del número (en vez de -5 vamos a representar el número 5) y procedemos a representarlo como en BSS(**3**), obteniendo como resultado `101`

La cadena final en SM(4) se obtiene juntando ambas partes:

$$R_{SM(4)}(\text{-}5) = \mathbf{1}1101$$

1.3. Rango

El rango es el intervalo de números representables en un sistema con una cierta cantidad de bits (n bits).

El número mínimo que podemos representar en un sistema de signo magnitud, va a ser negativo, es decir va a comenzar con 1. Su magnitud va a ser la mas grande posible. De esta manera, para n bits, el mínimo va a ser:

$$\underbrace{1}_{signo}\ \underbrace{1...,1}_{n-1 magnitud}$$
$$-(2^0 + ... + 2^{n-2})$$
$$-(2^{n-1} - 1)$$

El numero máximo lo vamos a obtener utilizando signo positivo, es decir 0; y nuevamente la mayor magnitud.

$$\underbrace{0}_{signo}\ \underbrace{1...,1}_{n-1 magnitud}$$
$$2^0 + ... + 2^{n-2}$$
$$2^{n-1} - 1$$

Es decir, que en este caso, el rango es $[-(2^{n-1} - 1), 2^{n-1} - 1]$.

Es interesante notar que en dicho intervalo no hay 2^n numeros distintos. Por ejemplo si $n = 3$, dicho intervalo nos queda $[-(2^{3-1} - 1), 2^{3-1} - 1] = [-3, 3]$ y en dicho intervalo hay 7 números:

$-3, -2, -1, 0, 1, 2, 3$.

En binario sin signo, con 3 bits teníamos 8 números representables diferentes. Dónde está el numero que nos falta?

Tanto las cadenas 100 como 000 representan al 0. Decimos entonces que hay una doble representación del 0. Esto no es deseable ya que por un lado no aprovecha completamente las cadenas y por otro tiende a complicar la aritmetica, pues nos obliga a considerar 2 tipos posibles de cero.

1.4. Aritmetica

Suma

La suma en SM se realiza de distintas maneras en función de los signos de las cadenas a sumar.

Si las cadenas a sumar tienen el mismo signo (ambas negativas o ambas positivas), la suma se realizará sumando las magnitudes como vimos para BSS y tomando como signo el signo del resultado.

Tomemos como ejemplo la suma `1101+1001`. El signo es el mismo de los operandos (`1`) y la magnitud se obtiene al sumar en BSS `101 + 001 = 110`. Por lo tanto el resultado de la suma es `1110`.

Si las cadenas a sumar tienen diferente signo, lo que vamos a hacer primero es identificar qué cadena tiene la mayor magnitud (llamaremos A a dicha cadena y B a la otra). El signo del resultado va a ser el signo que tenga A, y la magnitud resultado se obtiene restando a la magnitud de A la de B.

Supongamos el ejemplo `1101 + 0001`. La mayor magnitud la tiene la cadena `1101`, ya que su magnitud es `101` (5) y es mayor que la magnitud `001` (1). Es por esta razón que ya podemos afirmar que el signo del resultado va a ser `1`. Tenemos ahora que restar las magnitudes (a la mayor magnitud, la menor). Esta resta auxiliar la tenemos que hacer en BSS y nos queda: `101 - 001 = 100`. Entonces el resultado final es `1100`

Resta

El cálculo de una resta puede simplificarse con la siguiente equivalencia:

$$A - B = A + (-B)$$

dado que $-B$ es el inverso de B y es posible construirlo simplemente invirtiendo el bit de signo sobre B.

En pocas palabras, la operación se traduce en una suma y se deben analizar los casos que describimos antes.

Veamos como ejemplo la suma `1101 - 1001`, que se traduce a `1101 + 0001` y dicha suma la resolvemos como vimos anteriormente.

2. Complemento a 2

La idea de este sistema es lograr tener números negativos pero manteniendo la forma de sumar y restar que teníamos en BSS. Esto es muy deseable porque permite que una maquina no neceiste dentro de su ALU diferentes circuitos para sumar numeros naturales y enteros; lo cual permite abaratar costos.

Suponga que tiene un solo dígito en el sistema decimal (dec(1)) y se tiene que poder representar negativos (sin usar el signo "-"). Entonces se los ubica en una rueda como la de la figura 1 de manera que cada cadena (en este caso, dígito) quede aparejado con el opuesto a su complemento a la base (en este caso 10). Así, el 9 queda asociado al -1 pues: $10 + (-1) = 9$

Este aparejamiento tiene la particularidad que preserva las propiedades de las operaciones aritméticas, pues la suma entre cadenas sigue el mismo mecanismo

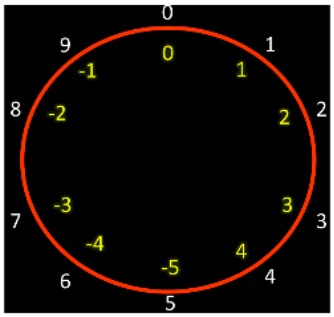


Figura 1: Distribución de las cadenas de; sistema Dec(1)

que en decimal, aunque las cadenas estén representando otros números y esa es una gran ventaja!. Por ejemplo en el sistema decimal restringido a un bit, 8+2 = 0 (c=1). Si se tiene en cuenta el valor representado por cada una: I(8)+ I(2) = -2+2=0

Otro ejemplo: 7+4=1 (con acarreo 1) entonces

$$I(7) + I(4) = -3 + 4 = 1$$

Esta idea la llevamos al sistema binario, para lo cual ubicamos las cadenas en un círculo ordenadas alfabéticamente, y luego le vamos asignando números positivos en sentido horario y negativos en sentido anti-horairo, como se muestra en la figura 2. De esta manera conseguimos que algunas cadenas (las mas pequeñas en sentido alfabético) queden asociadas a valores positivos y otras a valores negativos (las mayores en sentido alfabético)

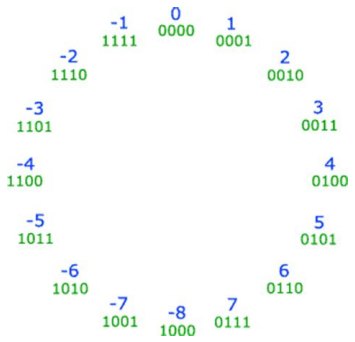


Figura 2: Distribución de las cadenas de 3 bits sobre los valores enteros en un sistema CA2(3)

Notación: Para decir que trabajamos con un sistema de complemento a 2 de n bits, escribimos $CA2(n)$.

Vamos a utilizar las cadenas que empiezan en 0 para los numeros positivos y las cadenas que comiencen en 1 para los negativos. Es importante notar que

no vamos a utilizar ese bit para signo como en sistema anterior, pero si ocurre que el bit nos dará información a la hora de interpretar.

Para una cadena definimos la operacion ***Complemento a 2*** como la cadena resultante de invertir los bits de la original y sumarle 1 al resultado. Por ejemplo, el complemento a 2 de la cadena 0001 es 1111, ya que primero invierto los bits obteniendo 1110 y luego sumo 1. El complemento a 2 de una cadena de n bits nos da otra, tal que si las sumamos obtenemos como resultado la cadena de n ceros. Por ejemplo, con 4 bits: $0001 + 1111 = 0000$.

2.1. Representar

A la hora de representar se tiene en cuenta si el número es positivo (o cero), o si es negativo. Los números positivos se representan en binario sin signo y los números negativos se representan con el **complemento a dos de la cadena** que representa su **valor absoluto**.

Por ejemplo, en CA2(4), la representación del 3 es `0101`, es decir, igual que en BSS.

Para representar -3 en CA2(4), primero representamos el 3 en BSS (es decir 0101) y luego tomamos el complemento a 2 de dicha cadena `0101` $\rightarrow$ `1010 + 1` $\rightarrow$ `1011`.

2.2. Interpretar

Al momento de interpretar se recorre el camino inverso a la representación, por lo tanto también se debe determinar si estamos frente a una cadena que representa un número negativo o un número positivo. Si la cadena comienza en 0, sabemos que se trata de un número positivo, y en ese caso simplemente interpretamos como si fuera BSS. En caso contrario, si la cadena comienza en 1, sabemos que se trata de un número negativo. En este último caso lo que hacemos es tomar el complemento a dos de dicha cadena, interpretar el resultado en BSS y finalmente le agregamos el signo negativo.

Por ejemplo, la cadena `1111` comienza con 1, por lo que representa un número negativo.

1. Calculamos su complemento a dos: $1111 => 0000 + 1 = 0001$.
2. Interpretamos la cadena resultante 0001 en BSS, lo cual nos da 1
3. Ahora, como sabemos que se trataba de un negativo, agregamos el signo, siendo el resultado final -1.

2.3. Rango

Para calcular el rango, vamos a obtener las cadenas que nos dan el máximo y el mínimo. Para el primer caso, sabemos que necesitamos una cadena positiva y en particular a la que nos de el número mas grande. Dado que las cadenas positivas se comportan como binario sin signo, si trabajamos con complemento a 2 de n bits, la cadena que nos da al máximo es `011...11` (un 0 seguido de todos unos). Dicha cadena, si la interpretamos nos da:

$$2^0 + ... + 2^{n-2}$$

$$2^{n-1} - 1$$

Para el caso negativo, como al momento de interpretarlo vamos a aplicar la operación de complementar, lo que queremos es buscar al número cuyo complemento sea lo mas grande posible. De todas las cadenas negativas, esto lo vamos a obtener con `100...00` (un 1 seguido de todos ceros). Si complementamos

$$100.,00$$

$$011.,11 + 1$$

$$100.,00$$

Que luego al interpretar (ahora usando binario sin signo, porque ya complementamos) nos da:

$$2^{n-1}$$

Recordamos que era negativa, por lo que el resultado es:

$$-2^{n-1}$$

Finalmente, el rango es:

$$[-2^{n-1}, 2^{n-1} - 1]$$

Notar que en este caso, la cantidad de números representables si es de 2^n, ya que no hay doble representación de ningún número.

2.4. Aritmetica

La aritmetica en CA2 es lo mas fácil: Es exactamente igual a como la haciamos en BSS, tanto la suma como la resta[1]

3. Exceso

El sistema de exceso trabaja utilizando las reglas de interpretación y representación de BSS pero desplazando todas las interpretaciones hacia un costado de la recta númerica. Por ejemplo, si trabajamos con un sistema de exceso 4, lo que estamos diciendo es que a las cadenas las vamos a interpretar restandole 4 al valor que nos da la interpretación, es decir comenzando desde el -4 en lugar del 0.

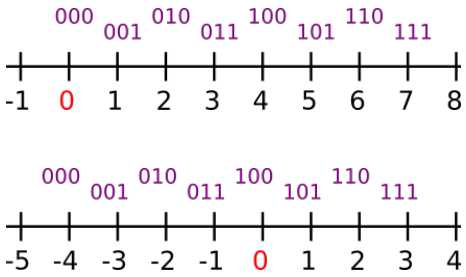
Para trabajar en un sistema de exceso, necesitamos conocer ademas de la cantidad de bits n, el valor del desplazamiento D, de esta forma hablamos de un sistema $EX(n, D)$.

La idea que atraviesa el sistema **exceso** es la de desplazar las cadenas sobre la recta numérica, con respecto a un sistema de base que es el BSS. Este desplazamiento lo llamamos exceso.

Tenemos la recta númerica donde se muestra la distribución de las cadenas de BSS(3):

En exceso de 4, la misma recta queda:

[1] En una próxima edición se incluirán ejemplos de sumas y restas en CA2. Mientras tanto tratá de comprobar lo que acá se dice



3.1. Representación

Si queremos representar un número, lo que vamos a hacer es sumarle el valor D y luego representar ese resultado en BSS.

Por ejemplo, si trabajo en EX(4,8) (es decir Exceso de 4 bits con 8 de desplazamiento), y quiero representar el -2, lo primero que hago es sumarle el exceso, es decir 8: $-2+8=6$ y luego represento al 6 en BSS de 4 bits: 0110

3.2. Interpretación

Al momento de interpretar, lo primero que hacemos es usar las reglas de binario sin signo. Una vez que obtuvimos el valor númerico de la cadena, le quitamos el exceso para obtener el verdadero numero representado en la cadena.

Por ejemplo, si trabajo en EX(4,8) (es decir Exceso de 4 bits con 8 de desplazamiento), y quiero interpretar la cadena 0111, lo que hago es interpretarla primero en BSS, lo cual nos da 7; y luego le resto el desplazamiento $7-8=-1$.

3.3. Rango

En Exceso de n bits, las cadenas que nos van a dar a los números mínimos y máximos son las mismas que en BSS: $\underbrace{00.,00}_{n}$ y $\underbrace{11.,11}_{n}$. Los valores que representan cada una dependen del valor del exceso D.

El minimo va a ser $0-D=-D$ mientras que el máximo va a ser 2^n-1-D.