Aritmética Binaria

© Luis Entrena, Celia López, Mario García,
Enrique San Millán

Universidad Carlos III de Madrid

Índice

- Representación de números con signo
  - Sistemas de Signo y Magnitud, Complemento a 1 y Complemento a 2
  - Propiedades del sistema de Complemento a 2
- Aritmética Binaria
  - Adición y Sustracción
  - Multiplicación y División
- Representación de números reales

Representación de números con signo

- Los números con signo tienen dos partes: signo y magnitud
- Es necesario **codificar el signo** del número para poder representarlo utilizando sólo 0s y 1s:
  - Normalmente se codifica el signo con un 0 para números positivos y un 1 para números negativos
- Según diferentes formas de **codificar la magnitud** se obtienen 3 diferentes sistemas de representación:
  - Signo y magnitud
  - Complemento a 1
  - Complemento a 2

Representación Signo y Magnitud

- Los números en el sistema de Signo y Magnitud se codifican de la siguiente forma:
  - El MSB es el signo (0 si es positivo o 1 si es negativo)
  - El resto de los bits son la magnitud del número a representar, codificada en binario natural
- Ejemplos: $25_{10} = 11001_{BIN}$

$+25_{10} = 011001_{SM}$

$-25_{10} = 111001_{SM}$

Complemento a 1

- Los números en el sistema de Complemento a 1 se codifican:
  - Si el número es positivo:
    - El MSB es un 0 (signo)
    - El resto de los bits son la magnitud en binario natural
  - Si el número es negativo:
    - El MSB es un 1 (signo)
    - El resto de los bits son el complemento (a 1) de la magnitud
- Ejemplos:

$$+25_{10} = 011001_{Ca1}$$

$$-25_{10} = 100110_{Ca1}$$

Complemento a 2

- Los números en el sistema de Complemento a 2 se codifican:
  - Si el número es **positivo**:
    - El MSB es un 0 (signo)
    - El resto de los bits son la magnitud en binario natural
  - Si el número es **negativo**:
    - El MSB es un 1 (signo)
    - El resto de los bits son el complemento a 2 de la magnitud.
      - El complemento a dos de un número es su complemento + 1

$$Ca2(A) = \overline{A} + 1$$

      - Equivalentemente, se puede definir como (siendo n el número de bits):

$$Ca2(A) = 2^n - A$$

$$(2^n - A = 2^n - 1 + 1 - A = 11....11 - A + 1 = \overline{A} + 1)$$

- Ejemplos: $+25_{10} = 011001_{Ca2}$

  $-25_{10} = 100111_{Ca2}$

Complemento a 2

- No hay que confundir los conceptos de "operación de complementar a 2" y "representación en complemento a 2" de un número:
  - Operación de complementar a 2 de un número A:

$$Ca2(A) = 2^n - A = \overline{A} + 1$$

  - Representación en complemento a 2 de un número A:
    - Hay que distinguir si es positivo o negativo, sólo en el caso de que sea negativo dicha representación se obtiene aplicando la operación de complementar a 2.
- La representación en complemento a 2 es la más utilizada en los sistemas digitales para números con signo.

Extensión del número de bits

- Un mismo número se puede representar con diferente número de bits:

$$+25_{10} = 011001_{SM} = 0\color{red}{0000}11001_{SM} = 0\color{red}{0000000000}11001_{SM}$$

- Para extender el signo:
  - En SM:
    - Añadir ceros justo después del signo
  - En Ca1 y Ca2:
    - Si es positivo añadir ceros a la izquierda
    - Si es negativo añadir unos a la izquierda

$$+25_{10} = 011001_{Ca2} = \color{red}{0000}011001_{Ca2} = \color{red}{000000000}011001_{Ca2}$$

$$-25_{10} = 100111_{Ca2} = \color{red}{1111}100111_{Ca2} = \color{red}{1111111111}100111_{Ca2}$$

Complemento a 2

Codificación			SM	Ca1	Ca2
0	0	0	0	0	0
0	0	1	1	1	1
0	1	0	2	2	2
0	1	1	3	3	3
1	0	0	-0	-3	-4
1	0	1	-1	-2	-3
1	1	0	-2	-1	-2
1	1	1	-3	-0	-1

- En SM y en Ca1 hay dos codificaciones distintas que representan el 0
- En Ca2 la representación del 0 es única

Complemento a 2

- Otra propiedad del Ca2 es que el complemento a 2 del complemento a 2 de un número es el mismo número (la operación inversa del Ca2 es el propio complemento a 2):

$$\mathrm{Ca2}(\mathrm{Ca2}(\mathrm{A}_{\mathrm{Ca2}})) = \mathrm{A}_{\mathrm{Ca2}}$$

Dem: $\mathrm{Ca2}(\mathrm{Ca2}(A_{Ca2})) = 2^{\mathrm{n}} - (\mathrm{Ca2}(A_{Ca2})) = 2^n - (2^n - A_{Ca2}) = A_{Ca2}$

- De lo cual se deduce esta otra propiedad:

$$-\mathrm{A}_{\mathrm{Ca2}} = \mathrm{Ca2}\,(\mathrm{A}_{\mathrm{Ca2}})$$

Dem: Si A_{Ca2} es positivo, entonces por la propia definición de representación en Ca2

Si A_{Ca2} es negativo, entonces $-A_{Ca2}$ se obtiene haciendo la operación inversa del Ca2, pero como $\mathrm{Ca2}(\mathrm{Ca2}(\mathrm{A}_{\mathrm{Ca2}})) = \mathrm{A}_{\mathrm{Ca2}}$ se tiene que $-\mathrm{A}_{\mathrm{Ca2}} = \mathrm{Ca2}\,(\mathrm{A}_{\mathrm{Ca2}})$

- Ejemplo:

Partiendo de un número positivo:	00001001	(+9)
Realizando la operación de Ca2:	11110111	(-9)
Realizando la operación de Ca2:	00001001	(+9)

Complemento a 2

- Otra forma de calcular el complemento a 2 de un número:
  - Empezando por el LSB, dejar iguales los bits hasta encontrar el primer 1 e invertir el resto:
  - Ca2(11100100) =00011100

    Comprobación: Ca2(11100100) = 00011011 + 1 = 00011100

- Otra forma de convertir de Ca2 a decimal:
  - Considerar el peso del signo como negativo
  - Ejemplos:
    - $1110_2 = 1*(-2^3) + 1*2^2 + 1*2^1 + 0*2^0 = -8 + 4 + 2 = -2_{10}$
    - $0110_2 = 0*(-2^3) + 1*2^2 + 1*2^1 + 0*2^0 = 4 + 2 = 6_{10}$
  - Demostración:
    - Si A es positivo el peso del signo será cero, y por tanto la magnitud es la suma del resto de pesos
    - Si A es negativo el peso será -2n, y por tanto el número que se obtendrá es:

      -2n + A = -Ca2(A) = -(-A) = A

Suma binaria

- Las sumas de números naturales en binario se realizan igual que en decimal:

```
   1001   (9)
+  1101  (13)
  10110  (22)
```

- Utilizando la representación de los números en Ca2, el método es válido también para números con signo, si se siguen las siguientes reglas.
  - Operandos con el mismo número de bits
  - Se descarta el acarreo final
  - Si los dos operandos tienen el mismo signo, y el resultado de la operación tiene signo diferente el resultado no es válido. Se dice que hay desbordamiento (**"overflow"**):
    - Esto sucede porque haría falta un bit adicional para poder representar el resultado.

Suma binaria en Ca2: Ejemplos

- Dos números positivos: (+9) + (+4)

  0 1001$_{Ca2}$ (+9)
  \+ 0 0100$_{Ca2}$ (+4)
  0 1101$_{Ca2}$ (+13)

- Número positivo grande y número negativo pequeño: (+9) + (-4)

  0 1001$_{Ca2}$ (+9)
  \+ 1 1100$_{Ca2}$ (-4)
  ~~1~~ 0 0101$_{Ca2}$ (+5)

- Números iguales de signo contrario: (+9) + (-9)

  0 1001$_{Ca2}$ (+9)
  \+ 1 0111$_{Ca2}$ (-9)
  ~~1~~ 0 0000$_{Ca2}$ (0)

- Dos números negativos: (-9) + (-4)

  1 0111$_{Ca2}$ (-9)
  \+ 1 1100$_{Ca2}$ (-4)
  ~~1~~ 1 0011$_{Ca2}$ (-13)

- Número positivo pequeño y número negativo grande: (-9) + (+4)

  1 0111$_{Ca2}$ (-9)
  \+ 0 0100$_{Ca2}$ (+4)
  1 1011$_{Ca2}$ (-5)

- Overflow: (+1) + (+1)

  0 1$_{Ca2}$ (+1)
  \+ 0 1$_{Ca2}$ (+1)
  1 0$_{Ca2}$ (-2)

Resta binaria

- Para realizar restas binarias se utiliza la propiedad vista anteriormente del complemento a 2:

$$-A_{Ca2} = Ca2\,(A_{Ca2})$$

  Y por tanto:

$$A_{Ca2} - B_{Ca2} = A_{Ca2} + Ca2\,(B_{Ca2})$$

  O equivalentemente:

$$A_{Ca2} - B_{Ca2} = A_{Ca2} + \overline{B_{Ca2}} + 1$$

- En los sistemas digitales se representan los números con signo en Ca2, y no se utilizan restadores (no hacen falta), sólo sumadores

Multiplicación binaria

- Las multiplicaciones en binario se realizan de forma similar a las multiplicaciones en decimal:

```
       1001    (9)
       1101  (13)
___________________
       1001
      0000
     1001
    1001
___________________
    1110101 (117)
```

- En el caso de números con signo se utiliza la representación en complemento a 2:
  - Si los dos números son positivos, se realiza directamente la operación. Al resultado se añade un bit de signo 0 (el resultado es un número positivo).
  - Si los dos números son negativos, se complementan a 2, se multiplican y al resultado se añade un bit de signo 0 (el resultado es un número positivo).
  - Si uno de los números es negativo, se hace el Ca2 de ese número, se multiplica por el otro, y al resultado se le realiza el Ca2 y se le añade un bit de signo 1 (el resultado es negativo).

División binaria

- La división se realiza también como en sistema decimal:

```
 1001  /  11
-011      0011
────
 0011
  -11
────
    0
```

- En el caso de números con signo se utiliza la representación en complemento a dos y las mismas reglas que para la multiplicación en cuanto a signos.

Representación de números reales

- Los números reales se pueden representar de dos formas:
  - Punto fijo
  - Punto flotante
- Punto fijo:
  - La coma decimal se considera fija en un punto.
    - Ejemplo: Datos de 32 bits, utilizar 20 bits para la parte entera y 12 bits para los decimales
  - Fácil realizar las operaciones de sumas, restas, multiplicaciones y divisiones vistas hasta ahora
  - Notación: Qm,n
    - m: número de bits de la parte entera (opcional)
    - n: número de bits para la parte decimal
    - Se utiliza un bit adicional para el signo (en total hacen falta m+n+1 bits).
    - Ejemplos: Q16,16, Q.32, etc.

Representación de números reales

- Punto flotante:
  - La coma decimal es "flotante"
  - Se descompone el número en dos partes: mantisa y exponente:

    $$N = M \times b^E$$

    Ejemplos:
    - $2547{,}35_{10} = 2{,}54735 * 10^3$
    - $0{,}0035_{10} = 3{,}5 * 10^{-3}$
    - $111{,}0110_2 = 1{,}11011_2 * 2^2$
    - $0{,}001101_2 = 1{,}101_2 * 2^{-3}$
  - Se utiliza un número fijo de bits para la mantisa, otro para el exponente y otro adicional para el signo
  - Normalización: Fija la posición de la coma decimal en la descomposición (para tener una representación única)

Representación de números reales

- Standard IEEE 754: Precisión simple (32 bits)
  - Se utiliza 1 bit para el signo
  - Se utilizan 8 bits para el exponente (E)
  - Se utilizan 23 bits para la mantisa (M)
  - Números normalizados: $N = (-1)^s * 2^{E-127} * 1.M$
    - E puede tomar valores entre 1 y 254 (el exponente está desplazado por -127)
  - El cero se representa como todo ceros en los campos E y M (es una excepción)
  - Algunas otras excepciones: E = 255 denota infinito (utilizado en casos de overflow, por ejemplo)
  - También se pueden representar números no normalizados (E=0). La descomposición es diferente, la mantisa es 0.M
- Standard IEEE 754: Precisión doble (64 bits)
  - Representación similar
  - 1 bit para signo, 11 bits para exponente, 52 bits para mantisa

Representación de números reales

- Ejemplo: Representar -7.625_{10} en precisión sencilla

$-7.625_{10} = -111.101_2$

Descomponiendo en la forma $N = (-1)^s * 2^{E-127} * 1.M$:

$-111.101_2 = (-1)^1 * 1.11101 * 2^2$

$S = 1$

$M = 11101$

$2 = E - 127 \rightarrow E = 129_{10} = 10000001_2$

Por tanto el número representado con precisión sencilla:

1 10000001 11101000000000000000000

Representación de números reales

- Operaciones aritméticas con números en coma flotante: más complejas
  - Suma:
    - Modificar uno de los números para que tengan el mismo exponente (denormalizarlo)
    - Normalizar el resultado una vez realizada la operación
  - Resta:
    - Análogo a la suma. Añade complejidad de convertir a complemento a 2
  - Multiplicación:
    - Sumar exponentes
    - Multiplicar mantisas
    - Redondear y normalizar
  - División:
    - Análogo a la multiplicación

- Debido a la complejidad de las operaciones en coma flotante, muchos circuitos electrónicos digitales (microprocesadores por ejemplo) tienen módulos dedicados a realizar estas operaciones

Referencias

- Digital Systems Fundamentals. Thomas L. Floyd. Pearson Prentice Hall
- Introduction to Digital Logic Design. John P. Hayes. Addison-Wesley
- Digital Systems. Principles and Applications. Ronald J. Tocci. Pearson Prentice Hall.