DISCA DEPARTAMENT D'INFORMÀTICA DE SISTEMES I COMPUTADORS

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCOLA TÈCNICA SUPERIOR D'ENGINYERS DE TELECOMUNICACIÓ

Fundamentos de Computadores
Grado en Ingeniería Técnica de Telecomunicación

Tema 2. Representación de enteros y reales

Objetivos

Comprender y aprender a utilizar los sistemas de representación binaria de los números enteros con signo, particularmente la codificación en complemento a dos.

Comprender la importancia de la representación en complemento a dos desde el punto de vista de la simplificación de la unidad aritmética de enteros.

Comprender la representación en coma flotante de los números, con sus ventajas e inconvenientes.

Aprender a codificar los números en coma flotante en el estándar IEEE-754, tanto en el caso normalizado como en el no normalizado.

1. Números enteros con signo

1.1. Codificación del signo

Codificación del signo

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.

Sistemas de codificación más habituales

No hay un único sistema de codificación para los números enteros con signo, los sistemas más utilizados son los siguientes (n será el número de bits disponibles):

- *Signo-magnitud.* El bit MSB=0 indica positivo y MSB=1 negativo, los demás bits codifican la magnitud del número.
- *Complemento a uno.* El signo se codifica con el MSB como en signo-magnitud. En los números positivos los demás bits son la magnitud del número. Los números negativos se representan pasando a binario $(2^n-1)-|x|$. http://hdl.handle.net/10251/5233
- *Complemento a dos.* El signo se codifica con el MSB como en signo-magnitud. En los números positivos los demás bits son la magnitud del número. Los números negativos se representan pasando a binario $2^n-|x|$.
- *Exceso Z.* La palabra binaria correspondiente a Z (que será 2^{n-1} o $2^{n-1}-1$) representa el cero, los valores mayores que Z son positivos y los menores que Z son negativos. En este caso MSB=0 significa negativo y MSB=1 significa positivo.

Correspondencia entre decimal y binario

Dada una palabra binaria que codifica un entero con signo su interpretación requiere conocer el sistema utilizado. La representación de un número entero con signo depende del sistema o convenio empleado y del número de bits disponibles

Tabla 1. Representación de números enteros con signo usando 3 bits

	Signo-Magnitud	Complemento a 1	Complemento a 2	Exceso 3
000	0	0	0	-3
001	1	1	1	-2
010	2	2	2	-1
011	3	3	3	0
100	0	-3	-4	1
101	-1	-2	-3	2
110	-2	-1	-2	3
111	-3	0	-1	4

1.2. Representación en signo-magnitud

Definición

El bit MSB es el bit de signo, el signo + se codifica con MSB=0 y el signo - con MSB=1, los demás bits representan la magnitud del número.

- El cero tiene dos representaciones. De la definición se tiene que el cero tendrá dos representaciones:

$$+0 = 0\ 00\ldots00$$
$$-0 = 1\ 00\ldots00$$

- Rango de representación. Dados n bits el rango de representación será:

$$[-(2^{n-1}-1),\ 2^{n-1}-1]$$

Vídeo poliTube: representación en signo y magnitud

http://hdl.handle.net/10251/5233

http://hdl.handle.net/10251/27535

1.3. Representación en exceso Z

Definición

Dados n bits para la representación se toma como valor de $Z=2^{n-1}$ o $Z=2^{n-1}-1$, se tiene entonces que:

- Z representa el cero
- Los valores mayores que Z son positivos (MSB=1)
- Los valores menores que Z son negativos (MSB=0)
- Rango de representación. Como Z representa al cero entonces el rango sin signo $[0, 2^n-1]$ pasa a ser el rango de representación:

$$[-Z, (2^n-1)-Z]$$

Codificación y decodificación

Dada una representación en exceso Z con n bits:

- La codificación de un número decimal se realiza con los siguientes pasos:
  1. Se suma Z al número decimal.
  2. El valor decimal resultante del paso anterior se pasa a binario como número sin signo.
  3. Si se obtiene un número de bits menor que n se completa con ceros a la izquierda.
- La decodificación de un número en exceso Z se realiza con los siguientes pasos:
  1. Se pasa el número en exceso Z a decimal aplicando el polinomio característico a todos los bits (incluyendo el bit de signo).
  2. Se resta Z al valor decimal obtenido en el paso anterior.

Vídeo poli Tube : representación en exceso Z

http://hdl.handle.net/10251/10112

1.4. Representación en complemento a dos

Definición y rango de representación

Dados n bits para la representación, el bit MSB=0 indica signo + y MSB=1 indica signo -. Los demás bits se codifican de forma diferente dependiendo de si el número es positivo o negativo:

- Números positivos. Los bits a la derecha del MSB codifican el módulo del número en binario.
- Números negativos. La representación de $x<0$ (incluyendo el bit de signo) se obtiene representando en binario:

$$2^n - |x|$$

- Rango de representación. El rango de representación en complemento a dos con n bits es:

$$[-2^{n-1}, +(2^{n-1}-1)]$$

Vídeo poli Tube : Introducción a la representación en complemento a dos

http://polimedia.upv.es/pub/visor/pub.swf?file=/AGONZALEZTELLEZ/c2_intro

2. Propiedades de la representación en complemento a dos

2.1. Codificación y decodificación

Codificación y decodificación de números positivos

La codificación y decodificación de los números positivos en complemento a dos se hace igual que en signo magnitud.

Codificación de números negativos

Dado un número decimal negativo x, su representación en complemento a dos con n bits se obtiene representando en binario:

$$2^n - |x|$$

Es necesario comprobar previamente que $|x| <= 2^{n-1}$, es decir, que está dentro del rango.

Decodificación de números negativos

Dado un número binario w en complemento a dos con n bits y bit de signo igual a 1, el número decimal (x) que representa se obtiene haciendo en decimal:

$$w_{10} - 2^n$$

w_{10} es el valor decimal que se obtiene aplicando el polinomio característico a w ASIGNANDO PESO NUMÉRICO AL BIT DE SIGNO.

2.2. Métodos alternativos para la transformación C2

La transformación C2 se suele aplicar a los números directamente en binario. Hay dos métodos para la obtención de C2(x) que no requieren cálculos cuando x está en binario.

- Método A. Invertir todos los bits en la representación binaria de x y sumarle 1.

  Ejemplo:

Teniendo en cuenta que 65=01000001 y que -65_{C2_8}= 10111111

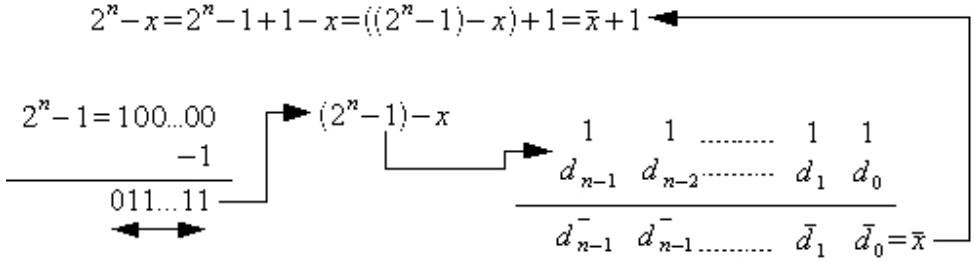
$$C2(01000001) = 10111110 + 1 = 10111111$$

- Método B. Dejar sin cambiar todos los bits a la derecha del 1 de menor peso (este incluido) e invertir todos los demás.

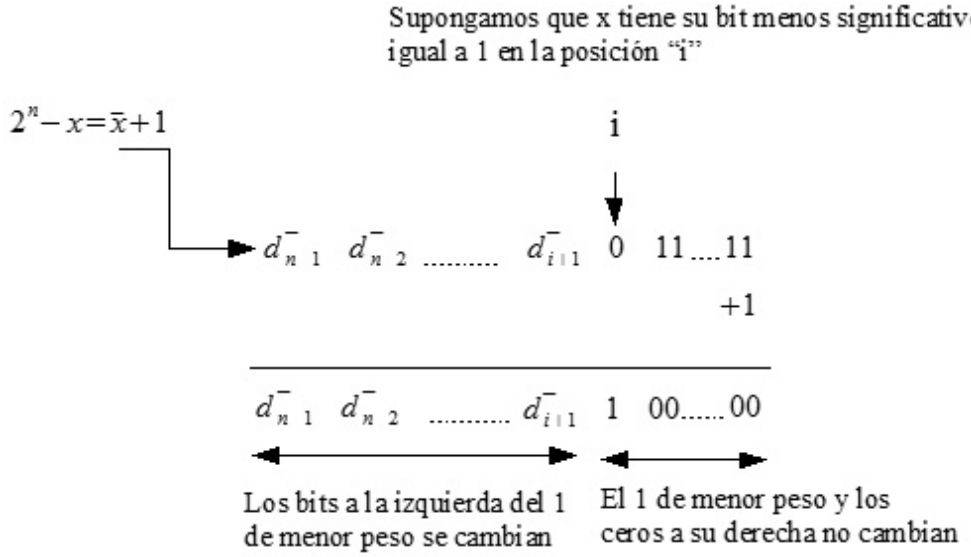
  Ejemplo:

$$C2(01000001) = 1011111\ 1 = 10111111$$

Demostración del método A



Demostración del método B



Vídeo poli Tube : Cambio de signo en complemento a dos

http://polimedia.upv.es/pub/visor/pub.swf?file=/AGONZALEZTELLEZ/c2_cambio_signo

2.3. Propiedades

Representación de cero

Sólo hay una representación del cero lo cual es una ventaja del complemento a dos frente al signo-magnitud y al complemento a uno.

$0_{10} = 00...00_{C2,n}$

Extensión de signo

Al ampliar el número de bits de la representación, la extensión de signo se realiza del siguiente modo:

- Números positivos. Se añaden ceros a la izquierda.
- Números negativos. Se añaden unos a la izquierda.

Demostración.

$$Dado\ x<0 \quad x_{C2,n}=2^n-|x| \quad hay\ que\ obtener \quad x_{C2,m}=2^m-|x|$$

$$2^m-|x|=2^m-2^n+2^n-|x|$$

$m-n+1$ bits, n bits

$2^m = 1\ 0\ \ldots\ 0\ 0\ \ldots\ 0$

$2^n = \quad\quad 1\ 0\ \ldots\ 0$

$2^m-2^n \rightarrow 0\ 1\ \ldots\ 1\ 0\ \ldots\ 0$

$m-n$ bits, n bits

$2^m-2^n = 1\ \ldots\ 1\ 0\ \ldots\ldots\ldots\ 0$

$2^n-|x| = \quad\quad d_{n-1}d_{n-2}\ldots d_1\ d_0$

$1\ \ldots\ 1\ d_{n-1}d_{n-2}\ldots d_1\ d_0$

Cambio de signo

La transformación C2 tiene como efecto un cambio de signo en la representación de complemento a dos. Sea v > 0 y definimos la transformación C2(x) como C2(x) = 2^n -x , entonces se tiene que:

- C2(v) = -v. La representación binaria de C2(v), siendo v>0, nos da la representación de -v en complemento a dos

  Demostración. C2(v) = 2^n - v = 2^n - |-v | = $-v_{C2\_n}$

- C2(-v) = v. La representación binaria de C2($-v_{C2\_n}$), siendo v>0, nos da la representación de v en complemento a dos

  Demostración. C2(-v) = 2^n - ($-v_{C2\_n}$) = 2^n - (2^n - | v |) = v = v_{C2_n}

2.4. Aritmética

Realización de sumas

La principal ventaja de la representación de números enteros en Complemento a dos es que al sumar dos números representados en dicho convenio, el resultado es correcto y también está representado en el mismo convenio. Y ni importa si los operandos son positivos o negativos o de signo distinto, ni el orden. La suma de dos números representados en C2 da como resultado el valor de la suma representado también en C2.

Ejemplo.

$$\begin{array}{rrcr} & 1\,1\,0\,0\,1\,1\,0\,1_{c2} & = & -51_{10} \\ + & \underline{0\,0\,0\,0\,0\,0\,1\,0_{c2}} & = & \underline{+\,+2_{10}} \\ & 1\,1\,0\,0\,1\,1\,1\,1_{c2} & = & -49_{10} \end{array}$$

Eso sí, como veremos más adelante, hay que tener en cuenta el posible desbordamiento.

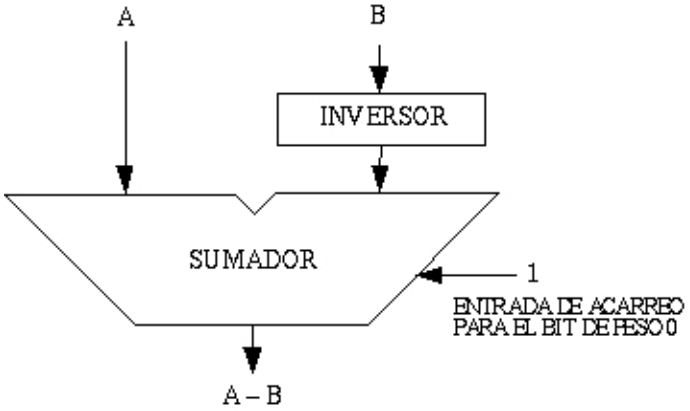
Conversión de restas en sumas

Si v y w pertenecen al rango de representación, entonces v - w = v + C2(w). Se obtiene el mismo resultado haciendo v-w aplicando la tabla de la resta binaria que haciendo v+C2(w) aplicando la tabla de la suma binaria

La representación en complemento a dos permite utilizar un circuito sumador como restador. Para ello sólo hay que hacer el complemento a dos del sustraendo lo cual se lleva a cabo realizando las dos operaciones siguientes:

- Invirtiendo los bits del sustraendo (un inversor por bit)
- Poniendo a uno la entrada de acarreo del bit de menor peso (en los sumadores esta entrada se fija a cero)

Figura 1. Restador en complemento a dos



Demostración

$$x+c2(y)=x+2^n-y=2^n+x-y$$

$$Si\ x-y\geqslant 0\rightarrow 2^n+x-y=x-y$$

$$Si\ x-y<0\rightarrow 2^n+x-y=2^n-(y-x)=2^n-|x-y|=(x-y)_{C2_n}$$

Nota

Cuando se usa el circuito anterior, el último bit de acarreo (C4) puede diferir del que se obtiene cuando hacemos la suma a mano, ya que el circuito realiza la operación A + Ca1(B) +1, mientras que a mano hacemos A + Ca2(B). El resultado es el mismo, pero el bit de acarreo final puede diferir.

Detección de desbordamiento

En la representación en complemento a dos cuando se realiza una suma la condición de que hay desbordamiento, es decir, el resultado está fuera de rango, es:

C_i XOR C_{i-1} = 1 , siendo C_i y C_{i-1} los dos últimos acarreos.

La operación de bits XOR se denomina OR-EXCLUSIVA y tiene la siguiente tabla de verdad:

C_i	C_{i-1}	C_i XOR C_{i-1}
0	0	0
0	1	1
1	0	1
1	1	0

Es decir, HABRÁ DESBORDAMIENTO CUANDO LOS DOS ÚLTIMOS ACARREOS SEAN DIFERENTES.

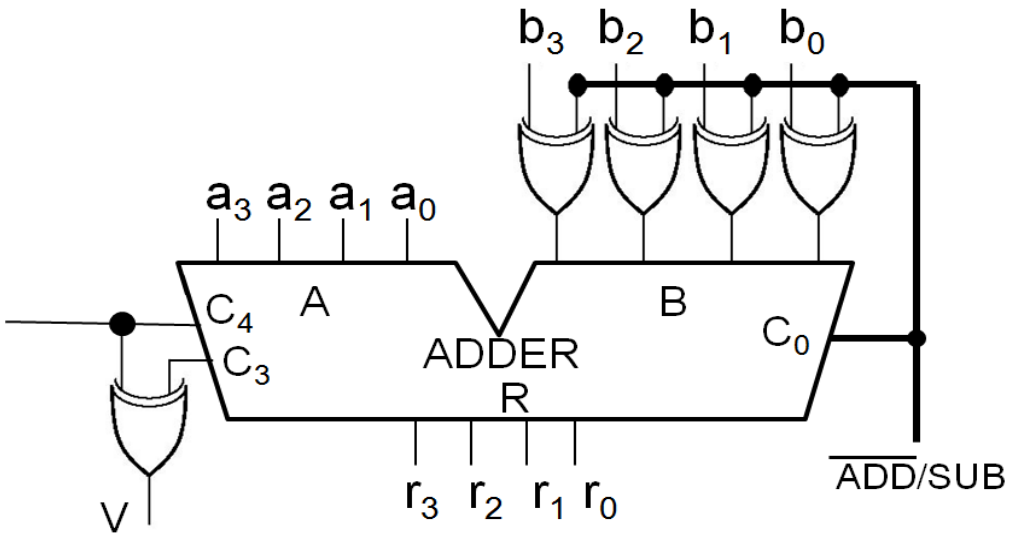
Cuando se produce desbordamiento, el resultado no es correcto y se dice que no hay resultado o que el resultado no es válido.

Ejemplo.

$$\begin{array}{rll} \boxed{01}\,0\,0\,0\,1\,0\,0\,0 & \text{«} & \textit{Acarreos} \\ 0\,1\,0\,0\,1\,1\,0\,1_{c2} & = & +77_{10} \\ +\ \ 0\,1\,0\,0\,1\,0\,1\,0_{c2} & = & \underline{+\,+74_{10}} \\ 1\,0\,0\,1\,0\,1\,1\,1_{c2} & \neq & +151_{10} \end{array}$$

Circuito sumador/restador en Ca2

El circuito siguiente permite realizar sumas y restas de números representados en Complemento a 2. Si la señal $\overline{ADD}/SUB$ toma valor 0, el circuito realiza una suma. Si la señal $\overline{ADD}/SUB$ toma valor 1, el circuito realizará una resta (a – b).



Vídeo poli Tube : Aritmética en complemento a dos

http://polimedia.upv.es/pub/visor/pub.swf?file=/AGONZALEZTELLEZ/c2_aritmetica

Vídeo poli Tube : representación y aritmética de números enteros en complemento a dos

http://hdl.handle.net/10251/5234

3. Números en coma flotante

3.1. Representación en coma flotante

Caracterización de la coma flotante

La representación en coma fija se caracteriza por que los dígitos tiene un peso fijo. La referencia para fijar el peso es el punto (o coma) decimal, de forma que el dígito que está a la izquierda del punto decimal tiene peso 0.

En la representación en coma flotante el peso del dígito situado a la izquierda del punto decimal es variable. El peso de dicho dígito se especifica mediante el exponente.

Componentes de la representación

La representación en coma flotante tiene los siguientes componentes:

- Signo. Determina si el número es positivo o negativo.
- Mantisa. Incluye los dígitos más significativos del número y se normaliza poniendo el punto decimal a la izquierda o a la derecha del dígito de mayor peso.
- Exponente. Determina el peso del dígito que está a la izquierda del punto decimal (puede ser positivo o negativo).

Ejemplos.

El número -25.323 representado en coma flotante con la coma a la izquierda del dígito más significativo queda de la siguiente manera:

$$-25.323 = -0.25323*10^{2}$$

El exponente es 2 ya que es el peso del dígito a la izquierda del punto decimal.

El número 3425245002300032002.0 representado en coma flotante dejando sólo los seis dígitos más significativos y poniendo el punto decimal a la derecha del dígito más significativo (o el dígito más significativo a la izquierda del punto decimal) queda de la siguiente manera:

$$3425245002300032002.0 = 3.42524*10^{18}$$

El exponente es 18 ya que es el peso del dígito a la izquierda del punto decimal.

El número 0.000000000000742 representado en coma flotante colocando la coma a la derecha del dígito más significativo queda de la siguiente forma:

$$0.000000000000742 = 7.42*10^{-13}$$

El exponente es -13 ya que es el peso del dígito a la izquierda del punto decimal.

Características de la representación en coma flotante

Las características principales de la representación en coma flotante son:

- Amplía el rango de representación de los números enteros. Esto se consigue representando sólo los dígitos de más peso y ajustando el exponente con un criterio de normalización para saber cuál es el peso de los dígitos representados.
- Permite representar números fraccionarios. El punto decimal se puede desplazar hacia la derecha hasta colocarse en los dígitos diferentes de cero cuando el número es fraccionario, esto permite representar número fraccionarios que pueden llegar a ser muy próximos a cero.
- Es una representación aproximada. En coma flotante se pone el énfasis en la capacidad para poder representar los dígitos más significativos del número, lo cual obliga en general a renunciar a representar todos los dígitos diferentes de cero del número. El error es un aspecto muy importante de la coma flotante, sobre todo cuando se hacen cálculos.

Ejemplos.

En una calculadora que dispone de una longitud de palabra de 10 dígitos decimales ¿cuál es el mayor número entero que se puede representar? ¿Cuál es el mayor número que se puede representar utilizando 8 dígitos para la mantisa y 2 dígitos para el exponente? ¿Qué ventaja y que desventaja presenta la segunda opción?

El mayor número entero es: 9999999999

El mayor número entero dedicando dos dígitos al exponente es: $99999999*10^{99}$

Ventaja: el número $99999999*10^{99}$ es mucho más grande que 9999999999 = $99999999.99*10^{2}$

Desventaja: el número $99999999.99*10^{2}$ se representa en el segundo formato como $99999999*10^{2}$ lo cual supone que se pierden los dos dígitos significativos de menor peso, se produce pues un error en la representación

En una calculadora que dispone de una longitud de palabra de 10 dígitos decimales, ¿cuál es el número más pequeño que se puede representar suponiendo que todos los dígitos son fraccionarios? ¿Cuál es el número más pequeño que se puede representar utilizando 8 dígitos para la mantisa y 2 dígitos para el exponente?

Si todos los dígitos son fraccionarios el número más pequeño es: .0000000001

Dedicando 8 dígitos a la mantisa y dos al exponente: .00000001*10-99

En este caso está claro que .00000001*10-99 es mucho más pequeño que .0000000001 = .00000001*10-2

Ahora el inconveniente del error se da en ambos casos ya que en cualquier base siempre hay fracciones de la unidad que sólo se pueden representar de forma aproximada (número infinito de dígitos fraccionarios)

4. Formatos IEEE-754

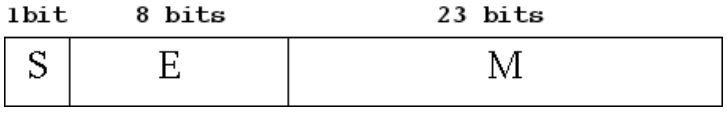
4.1. Definición

Formatos para precisión simple y precisión doble

El Institute of Electric and Electronic Engineers ha propuesto el estándar IEEE-754 para la representación binaria de coma flotante. El estándar tiene dos formatos que plantean el compromiso entre eficiencia (ocupación de memoria y tiempo de cálculo) y capacidad de representación (rango y precisión): cuantos más bits menor eficiencia pero mayor capacidad de representación.

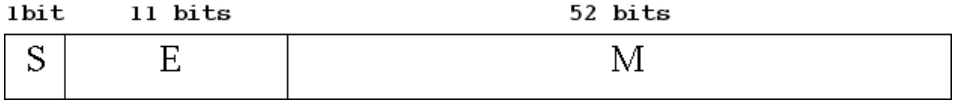
- Precisión simple. Ocupa 32 bits: 1 bit para codificar el signo (S), 8 bits para codificar el exponente (E) y 23 bits para codificar la mantisa (M).

Figura 2. Distribución de los bits en los campos de IEEE754 en precisión simple



- Precisión doble. Ocupa 64 bits: 1 bit para codificar el signo (S), 11 bits para codificar el exponente (E) y 52 bits para codificar la mantisa (M)

Figura 3. Distribución de los bits en los campos de IEEE754 en precisión doble



4.2. Rango normalizado

Codificación del signo

En ambos formatos IEEE-754 el bit MSB es el bit de signo. El signo se codifica separado e independiente de la mantisa.

- S = 0 : el número es positivo
- S = 1 : el número es negativo

Codificación de la mantisa

En los formatos IEEE-754 la mantisa se codifica con el campo M como fraccionaria con bit implícito. La mantisa se normaliza en la forma 1.M, es decir que siempre habrá un 1 a la izquierda del punto decimal por lo que ese 1 no hace falta que sea representado (bit implícito). El campo M incluye los bits que están a la derecha del punto decimal.

Pasos para obtener el campo M de la representación IEEE-754 de un número decimal.

- Pasar el número a base 2. Para esto se pasa a binario la parte entera y la parte fraccionaria del número.
- Colocar la coma a la derecha del bit más significativo igual a 1, ajustando el exponente. Se intenta normalizar el número. Los casos en que no se puede normalizar se tratan más adelante.
- M corresponde a la parte fraccionaria obtenida. Si se ha conseguido normalizar, M incluye los bits a la derecha de la coma.

Ejemplos.

El campo M correspondiente a la representación en precisión simple del número 35.75 es:

$$0.75 * 2 = 1.5 \text{ --> } d_{-1} = 1$$

$$0.5 * 2 = 1.0 \text{ --> } d_{-2} = 1$$

$$0.75_{10} = 0.11_2$$

$$35.75_{10} = 100011.11_2 = 1.0001111*2^5 \text{ --> } M = 00011110000000000000000$$

El campo M correspondiente a la representación en precisión simple del número 45.625 es

$$45.625_{10} = 101101.101_2 = 1.01101101*2^5$$

$$M = 01101101000000000000000$$

Codificación de exponente

Antes de codificar el exponente hay que normalizar la mantisa. El valor del exponente que se codifica es el que se obtiene al ajustar el exponente en el proceso de normalización de la mantisa.

El campo E es la representación en exceso Z del exponente obtenido tras normalizar y referido a la base 2. Los valores del exceso que fija el estándar para los dos formatos son:

- Precisión simple : Z = 127
- Precisión doble : Z = 1023

Los rangos de representación para el exponente son:

- Precisión simple : [-126, +127]
- Precisión doble : [-1022, +1023]

Obsérvese que se han suprimido los valores extremos: E=00...00 y E=11...11. Estos valores se reservan para poder tratar el desbordamiento y la representación no normaliza.

Vídeo poli Tube : Codificación de binaria de enteros y coma flotante

http://polimedia.upv.es/pub/visor/pub.swf?file=/AGONZALEZTELLEZ/codificacion_binaria_int_float

Ejemplos.

El valor del campo E correspondiente a la codificación IEEE-754 en precisión simple del número 7.4 es:

$$7.410 = 111.01100110...2 = 1.1101100110...*22 \text{ --> } exp = 2$$

$$E = 2 + 127 = 129 = 10000001$$

El valor del campo E correspondiente a la codificación IEEE-754 en precisión simple del número 45.625 es:

$$45.62510 = 101101.0112 = 1.01101011*25 \text{ --> } exp = 5$$

$$E = 5 + 127 = 132 = 10000100$$

4.3. Casos especiales

Rango no normalizado

Ocurre cuando al normalizar la mantisa se obtiene para el exponente un valor menor que el mínimo. En este caso la mantisa no se puede normalizar, o sea no se puede poner como 1.M sino que quedará como 0.M

- Precisión simple : Mínimo valor para el exponente = 1 - 127 = -126
- Precisión doble : Mínimo valor para el exponente = 1 - 1023 = -1022

Los números no normalizados se pueden utilizar y se representan como:

- E = 000...0 Este valor de E está reservado para este propósito, si E=000...0 entonces el bit implícito es 0 y el valor del exponente es siempre el valor mínimo: -126 en precisión simple y -1022 en precisión doble.
- M contiene los bits a la derecha de la coma tras ajustar el exponente al valor mínimo representable.

El rango no normalizado ha de utilizarse con cuidado ya que conforme se van haciendo más pequeños los números se va perdiendo precisión, debido a que el peso de los dígitos es fijo.

Ejemplos.

¿Cómo se representa en precisión simple 0.000011001_2*2^{-122} ?

$$0.000011001_2*2^{-122} = 1.1001_2*2^{-127} \text{ --> } -127 < -126 \text{ --> no se puede normalizar}$$

$$0.000011001_2*2^{-122} = 0.11001_2*2^{-126}$$

$$S = 0 ; M = 1100100...0 ; E = 00000000$$

¿Cómo se representa en precisión doble 0.000000101_2*2^{-1018} ?

$$0.000000101_2*2^{-1018} = 1.01_2*2^{-1025} \text{ --> } -1025 < -1022 \text{ --> no se puede normalizar}$$

$$0.000000101_2*2^{-1018} = 0.00101_2*2^{-1022}$$

$$S = 0 ; M = 0010100...0 ; E = 00000000000$$

Desbordamiento

Ocurre cuando se sobrepasa el valor máximo representable para el exponente. Cuando el valor del exponente resultante del ajuste de la mantisa supera el valor máximo, el número no se puede representar.

- Precisión simple : Valor máximo para el exponente = 254 - 127 = 127
- Precisión doble : Valor máximo para el exponente = 2046 - 1023 = 1023

Cuando ocurre desbordamiento el número no se puede utilizar y se representa con E = 111...11. Este valor de E está reservado para este propósito.

Ejemplos.

¿Cuál es la representación del número 1000.011_2*2^{126} en precisión simple?

$$1000.011_2*2^{126} = 1.000011_2*2^{129} \text{ --> } 129 > 127$$

Hay desbordamiento

¿Cuál es la representación del número 1110000.1001_2*2^{1020} en precisión doble?

$$1110000.1001_2*2^{1020} = 1.1100001001_2*2^{1026} \text{ --> } 1026 > 1023$$

Hay desbordamiento

4.4. Rango de representación

Rango de IEEE 754 en precisión simple normalizado

El número más grande que se puede representar es aquel con la mantisa y el exponente más grande posible:
E = 11111110 = 254-127 = 127 y M= 11111111111111111111111 que resulta en:
$1.11111111111111111111111_2 * 2^{127} = (2^1 - 2^{-23}) * 2^{127} = 2^{128} - 2^{114} = 3{,}4026 * 10^{38}$

El número más pequeño es el de menor mantisa y exponente:
E = 00000001 = 1 - 127 = -126 y M = 00000000000000000000000 que resulta en:
$1.0_2 * 2^{-126} = 2^{-126} = 1{,}1755*10^{-38}$

El rango IEEE 754 en precisión simple normalizado es:

$$+/-[1{,}1755*10^{-38}; 3{,}4026 * 10^{38}]$$

Rango de IEEE 754 en precisión simple no normalizado
Como el exponente es fijo (-126) el número más grande que se puede representar es aquel con la mantisa más grande:
M=11111111111111111111111 que corresponde con:
$0.11111111111111111111111_2*2^{-126} = (2^0-2^{-23})*2^{-126} = 2^{-126}-2^{-149} = 1{,}1755*10^{-38}$
(este valor está muy cerca del valor menor normalizado)

El número más pequeño es el de menor mantisa distinta de 0:
M = 00000000000000000000001 que corresponde con:
$0.00000000000000000000001_2 * 2^{-126} = 2^{-149} = 1{,}4013*10^{-45}$

El rango IEEE 754 en precisión simple normalizado es:

$$+/-[1{,}4013*10^{-45}; 1{,}1755*10^{-38}]$$

Rango de IEEE 754 simple precisión completo en binario

$$0 \cup \left[\pm 2^{-149}, \pm 2^{-126}\right[\cup \left[\pm 2^{-126}, \pm 2^{+128}\right[\approx \left[\pm 2^{-149}, \pm 2^{+128}\right]$$

Además, existen los casos especiales:

- E = 0 y M = 0 representa el cero (+ y -)
- E = 11111111 y M = 0 representa Infinito (+ y -)
- E = 11111111 y M != 0 representa NaN (Not a Number)