Punto Flotante - Estándar IEEE 754

Organización de computadoras 2014

Universidad Nacional de Quilmes

1. Punto Flotante

Con los sistemas enteros es posible representar un rango de enteros positivos y negativos centrados en el cero. Con los sistemas de punto fijo, se permite también representar números con parte fraccionaria, pero esta aproximación tiene limitaciones: El error producido al representar un número de magnitud pequeña es más significativo que el mismo error al representar un valor de magnitud mayor. Por eso es importante relativizar el error, pensándolo como una proporción (o un porcentaje).

Para números decimales, las limitaciones anteriores se superan utilizando la notación científica, que permite escribir de modo abreviado números muy grandes:

$$620000000000000000000 = 6,2 \times 10^{20}$$

y números muy cercanos a cero:

$$0,00000000000000000062 = 6,2 \times 10^{-20}$$

Lo que se hizo fue mover dinámicamente la coma decimal a una posición conveniente y con el exponente se registra la posición de la coma.

Esta misma técnica puede aplicarse al sistema binario, representando los números con la forma

$$N = M \times 2^E$$

Entonces las expresiones que utilizan esta notación cuentan con dos partes. La **mantisa M** representa la parte significativa del número, y el **exponente E** que permite "recordar" dónde estaba la coma antes de abreviar.Además, la representación de números dentro de una computadora, utilizaría siempre la base 2.

Por ejemplo, suponer la cadena **11000000**. Su interpretación en BSS(8) es:

$$I_{bss(8)}(11000000) = 2^7 + 2^6 = 64 + 128 = 192$$

Si la cadena anterior se supone en un sistema de punto fijo bss(8,6) se tiene:

$$I_{bss(8,6)}(11000000) = \frac{(2^7 + 2^6)}{2^6}$$

Y la equivalencia entre ambos sistemas es la **escala** 2^6, que se conoce como **corrimiento de la coma** de 6 lugares:

$$I_{bss(8)}(11000000) = I_{bss(8,6)}(11000000) \times 2^6 = \frac{(2^7 + 2^6)}{2^6} \times 2^6$$

Aplicando equivalencias se puede simplificar:

$$= (\frac{2^7}{2^6} + \frac{2^6}{2^6}) \times 2^6 = (2^{7-6} + 2^{6-6}) \times 2^6 = (2^1 + 2^0) \times 2^6$$

De aquí podemos ver que se aplican dos sistemas enteros mas pequeños (bss(2) y bss(3)):

$$I_{bss(2)}(11) \times 2^6 = I_{bss(2)}(11) \times 2^{I_{bss(3)}(110)}$$

Este valor entonces puede codificarse con una mantisa en BSS(2) y un exponente en BSS(3), siendo `11` la mantisa y `110` el exponente. La base del exponente no hace falta registrarla porque es siempre 2.

Entonces, a la hora de almacenarlas, se concatenan ambas subcadenas siguiendo algún orden predeterminado para el sistema definido. Si es mantisa seguido de exponente: `11110`. Es importante destacar que podría elegirse concatenar en otro orden (`11011`).

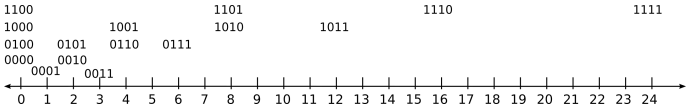
Así, con pocos bits de exponente se alcanza un rango considerablemente mas amplio que los sistemas de punto fijo. Por ejemplo, con 2 bits se puede representar (en bss) hasta el número 3, y por lo tanto se pueden "comprimir" 3 posiciones. Con 4 bits se puede representar (en bss) hasta el número 15, y por lo tanto se pueden "comprimir" 15 posiciones (o ceros), como sigue:

$$I(1100000000000000) = I(11) * 2^{15}$$

1.1. Rango y resolución

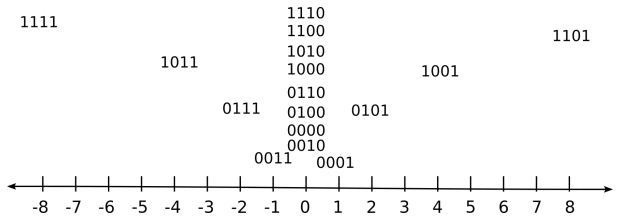
La definición del sistema de punto flotante tiene como característica importante que los valores representables se aglutinan en torno al cero y se dispersan hacia los extremos del rango. Esto implica que la resolución, a diferencia de los sistemas de punto fijo, sea variable.

Por ejemplo, considerar un sistema con exponente en bss(2) y mantisa en bss(2) organizados sus bits en el orden mencionado. Gráficamente, la distribución de las cadenas sobre la recta de números es como sigue, permitiendo la representación de números mas grande que en el sistema bss(4).



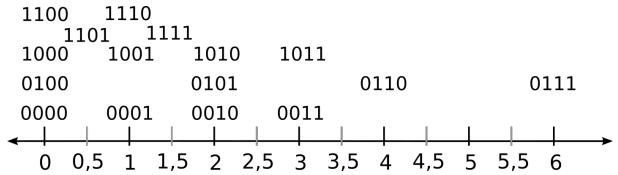
Sistema A: [e: bss(2)][m: bss(2)]

Similarmente, si el sistema de la mantisa es un sistema con signo, como sm(2) se tiene la siguiente distribucion



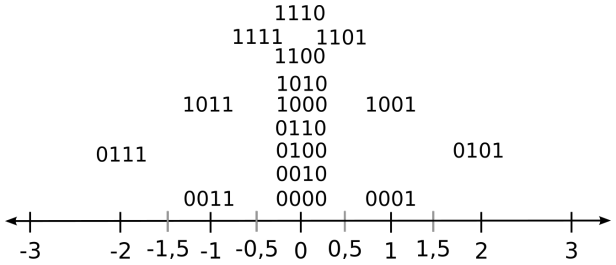
Sistema B: [e: bss(2)][**m: sm(2)**]

Por otro lado, si el exponente es el que permite representar negativos, en una expresión $N = M \times 2^E$ se permiten valores fraccionarios. Por ejemplo el siguiente sistema tiene mantisa bss(2) y exponente sm(2).



Sistema C: [**e: sm(2)**][m: bss(2)]

Finalmente, mediante el uso de sistemas con signo en ambas partes, se consigue una distribución balanceada a ambos lados del cero, pero que también permite representar fraccionarios



Sistema D: [e: sm(2)][m: sm(2)]

Si se comparan los sistemas **A** y **C** se ven claramente un aspecto en común y una diferencia. Ambos se distribuyen en los enteros positivos pero **C** permite representar fraccionarios (gracias al exponente con signo) y es una escala comprimida de **A**.

Si se comparan los sistemas **B** y **D** se ve que ambos son simétricos con respecto al cero (gracias a la mantisa con signo), pero **D** permite fraccionarios y por lo tanto es una escala de **C**.

1.2. Normalización

Cualquier número en punto flotante puede expresarse de ditintas formas. Por ejemplo, considerar un sistema con mantisa en sm(8) y ca2(5), con el formato (s)(e)(m). Interpretar:

- $I(0000000000100) = 4 \times 2^0 = 2^2$
- $I(0000010000010) = 2 \times 2^1 = 2^2$
- $I(0000100000001) = 1 \times 2^2 = 2^2$

Sin embargo, no es deseable tener múltiples representaciones para el mismo valor, por dos motivos. El primero, no desperdiciar cadenas, el segundo, no tener redundancia pues esto implica que la lógica de comparación entre cadenas sea mas compleja.

La solución a este problema es la **normalización**, esto es, establecer una regla para seleccionar las *cadenas válidas*, descartando las demás. En general se establece como regla que las cadenas válidas deben *comenzar con 1 en la mantisa*. En el ejemplo anterior la única cadena normalizada para representar el 2^2 es `0 11100 1000000`.

$$N = I_{sm}(01000000) \times 2^{I_{ca2}(11100)} = 2^6 \times 2^{-4} = 2^2$$

Facilmente pueden verse algunas desventajas de este criterio de normalización. Por un lado, la mitad de las

cadenas se descartan y por otro lado, ese bit que debe asegurarse en valor 1 ocupa un espacio innecesario y podría no escribirse. Esto lleva a la solución de dejar implícito ese primer bit de la mantisa y *agregar su peso a la hora de interpretar*.

Esto permite tener un bit mas disponible en la mantisa, consiguiendo cubrir un rango más amplio si la mantisa es entera, o más precisión, si la mantisa es fraccionaria.

1.2.1. Interpretación en un sistema normalizado con bit implícito

Suponer el caso donde se debe interpretar en un sistema de punto flotante con mantisa fraccionaria normalizada en el sistema signo-magnitud de 10 bits, con un bit implícito, esto quiere decir que el total son 11 bits. El exponente se asume en un sistema *sm(5)*. Además, el formato de la cadena distribuye los bits como sigue:

Magnitud Mant 9b	Signo Mant 1b	Signo Exp 1b	Magnitud Exp 4b

Por ejemplo, al interpretar la cadena `010001011 1 0 1110` se debe interpretar la mantisa en el sistema $SM(10{+}1, 10)$, donde se denota el bit implícito con la expresión $+1$.

$$I_{sm(10+1,10)}(1010001011)$$

A esta interpretación se le debe agregar el bit implícito para poder interpretar en el sistema de punto fijo:

$$I_{sm(11,10)}(11010001011)$$

De ahora en mas, la interpretación corresponde al sistema $sm(11, 10)$:

$$-I_{bss(10,10)}(1010001011) = -(2^{-1}{+}2^{-3}{+}2^{-7}{+}2^{-9}{+}2^{-10})$$

Por otro lado, se debe interpretar el exponente:

$$I_{sm(5)}(01110) = 2^1 + 2^2 + 2^3 = 14$$

Finalmente, el valor del número representado es

$$N = -(2^{-1} + 2^{-3} + 2^{-7} + 2^{-9} + 2^{-10}) \times 2^{14}$$

Es interesante notar que la mantisa es equivalente a:

$$\frac{-I_{bss}(1010001011)}{2^{10}} = \frac{-(2^9 + 2^7 + 2^3 + 2^1 + 2^0)}{2^{10}} = \frac{-(651)}{2^{10}}$$

Por lo tanto:

$$N = \frac{-(651)}{2^{10}} \times 2^{14} = -651 \times 2^4$$

2. Estándar IEEE

El estándar IEEE 754 define representaciones para números de coma flotante con diferentes tipos de presición: simple y doble, utilizando anchos de palabra de 32 y 64 bits respectivamente. Estas representaciones son las que utilizan los procesadores de la familia x86, entre otros.
Estos sistemas, a diferencia de los anteriores, permiten representar también valores especiales, los cuales serán tratados posteriormente.

Precisión simple

En la representación de 32 bits, el exponente se representa en exceso de 8 bits, con un desplazamiento de 127, y la mantisa está representada en un sistema SM(24+1,23), es decir que:

- Se tienen 24 bits explicitos y uno implícito
- 23 bits son fraccionarios

Como es un sistema signo-magnitud, se tiene 1 bit de signo y 24 bits de magnitud. De los bits de la magnitud, 1 está implícito y los otros 23 son los que se usan explícitamente. De aqui que estos 23 bits son fraccionarios y el bit implícito es entero.

Además, el total de 32 bits se escriben con el siguiente formato:

S	Exponente: 8b	Magnitud: 23b

Precisión doble

De manera similar, en la representación IEEE de doble precisión, el bit mas significativo es utilizado para almacenar el signo de la mantisa, los siguientes 11 bits representan el exponente y los restantes 52 bits representan la mantisa. El exponente se representa en exceso de 11 bits, con un desplazamiento de 1023.

S	Exponente: 11b	Magnitud: 52b

Como en el caso de precisión simple, también se tiene una mantisa normalizada con un bit entero y los restantes

fraccionarios, es decir que tiene la forma "1,X", donde X es el valor de los bits fraccionarios. Además, como se tiene un bit implícito, el dígito 1 (entero) está oculto y por lo tanto no es almacenado en la representación, permitiendo así ganar precisión.

Sin embargo, los parámetros usados en las representaciones de simple y doble precisión son los que se describen en la siguiente tabla:

	P. simple	P. doble
Cant. total de bits	32	64
Cant. de bits de la mantisa (*)	24	53
Cant. de bits del exponente	8	11
Mínimo exponente (emin) (**)	-126	-1022
Máximo exponente (emax) (**)	127	1023

(* incluyendo el bit implícito)

(** emin es -126 en lugar de -127, que corresponde al minimo valor del exceso(8,127), ver siguiente sección)

Representación de valores especiales

Una cuestión de interés para los sistemas de numeración usados en las computadoras, es analizar qué sucede cuando una operación arroja como resultado un número indeterminado o un complejo. En estos casos el resultado constituye un valor especial para el sistema y se almacena como NaN (Not a Number) tal como ocurre al hacer, por ejemplo $\frac{\infty}{\infty}$ ó $\sqrt{-4}$.

A veces sucede que el resultado de una operación es muy pequeño y menor que el mínimo valor representable, en este caso se almacenará como +0 ó -0, dependiendo del signo del resultado. También se observa que al existir un 1 implícito en la mantisa no se puede represental el valor cero como un número normal, por lo que éste es considerado un valor especial.

Por otro lado, ante una operación que arroje un resultado excesivamente grande (en valor absoluto), este se almacenará como $+\infty$ ó $-\infty$.

De las situaciones mencionadas, surge la necesidad de una representación para los valores especiales.

El exponente lo dice todo

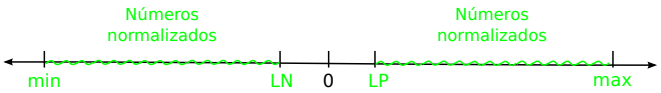
Es importante detenerse en la representación del exponente, que como se ha visto, utiliza el sistema Exceso con frontera no equilibrada (127 o 1023), lo que permite almacenar exponentes comprendidos en el rango [-127,128] en el sistema de precisión simple o [-1023,1024] en el sistema de precisión doble. Pues, puede verse en la tabla de la sección anterior que el rango entre emin y emax no cubre todo el rango disponible, y esto se debe a que se reservan las representaciones de emin-1 y emax+1 en ambas precisiones para representar valores especiales. Nótese que esta elección no es arbitraria: la cadena que representa emin-1 está compuesta de ceros y la cadena que representa el valor emax+1 está compuesta por unos, ambos fácilmente reconocibles.

Adicionalmente pueden representarse valores subnormales o denormalizados, es decir números **no normalizados**, de la forma $\pm 0, X * 2^{\delta}$, que se extienden en el rango comprendido entre el mayor número normal negativo y el menor número normal positivo. Dicho exponente especial δ tiene el valor -126.

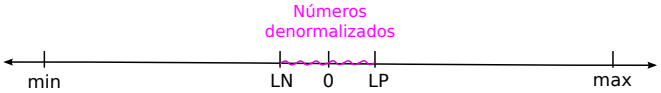
Nota: estos números desnormalizados no tienen bit implícito (ó es cero).

De esta manera, se definieron las siguientes clases de representaciones:

Números normalizados Las cadenas de esta clase se distinguen con un exponente que no sea nulo (cadena compuesta por 0s) ni saturado (cadena compuesta por 1s). Gráficamente, la clase de números normalizados se distribuye como sigue (LP=Límite positivo/ LN=Límite negativo):



Números denormalizados Las cadenas de esta clase tienen exponente nulo pero mantisa no nula. Gráficamente, la clase de números denormalizados se distribuye como sigue:



Ceros Esta clase incluye sólo dos cadenas: aquella compuesta con exponente y mantisa nulos, con ambos signos posibles. Esto permite representar el valor 0 positivo o negativo. Es importante notar que ninguna de las clases anteriores (normalizados o desnormalizados) permite construir por si misma el valor 0.

Infinitos Esta clase se identifica con exponente y mantisa saturados (`1..1`). Permite representar la situación en que el resultado está fuera del rango representable por los normalizados

Not a number (NaN) Esta clase se identifica con exponente saturado y es utilizada para representar los casos de error descriptos antes

La siguiente tabla resume cómo se distinguen las cadenas de las diferentes clases.

Exponente	Mantisa	Clase de número
0..0	0..0	± 0
0..0	$\neq$ 0.,0	Denormalizados: $\pm 0, X * 2^{emin}$
1..1	1..1	$\pm\infty$
1..1	$\neq$ 1.,1	NaN
[emin,emax]	cualquiera	Normalizados: $\pm 1, X * 2^e$

Ejemplos de interpretación

Cadena normalizada

Por ejemplo, se quiere interpretar la cadena en formato de precisión simple:

`1100 0010 0110 1011 1000 0000 0000 0000`

Para esto, es necesario separar los diferentes campos de la cadena:

1	10000100	110 1011 1000 0000 0000 0000
S	exponente:8b	Mantisa: 23b t

Dado que el exponente no es la cadena `00000000` ni la cadena `11111111`, se entiende que se lo debe interpretar como **un número en la clase normalizada**, por lo que se debe interpretar separadamente:

- Exponente: interpretar en Exc(8,127)

  $$e = I_{ex}(10000100) = I_{bss}(10000100) - 127 = 2^7 + 2^2 - 127 = 5$$

- Mantisa: Dado que hay un bit implícito cuyo peso es $2^0 = 1$.

  $$m = -(1 + I_{bss(23,23)}(11010111000000000000000)) =$$

  $$= -(1 + 2^{-1} + 2^{-2} + 2^{-4} + 2^{-6} + 2^{-7} + 2^{-8})$$

Cadena desnormalizada

El siguiente es un ejemplo de una cadena (en formato de precisión simple) cuyo exponente indica que es un número desnormalizado:

`0000 0000 0010 0000 1011 1000 0000 0000`

Separando los campos se obtiene:

0	00000000	010 0000 1011 1000 0000 0000
S	Exp:Exc(8,127)	Mant: BSS(23,23)

- Exponente: En este caso no se interpreta el exponente, sino que se usa el exponente especial

  $$e = -126$$

- Mantisa: Dado que **no hay bit implícito**

  $$m = I_{bss(23,23)}(01000001011100000000000) =$$

  $$= 2^{-2} + 2^{-8} + 2^{-10} + 2^{-11} + 2^{-12}$$