Sistemas de Punto Flotante

Organización de computadoras 2018

Universidad Nacional de Quilmes

1. Motivación/introducción

Para representar números reales en una computadora es necesario tener una manera de codificar la coma raíz. La forma mas simple se implementa conviniendo la posición de la coma en un lugar fijo, separando de esta manera la cadena en dos partes: parte entera y parte fraccionaria, y eso es lo que se conoce como **punto fijo**. De esta manera, para poder representar la parte fraccionaria en necesario sacrificar el rango de alcance de la representación. Es decir cuantos mas bits fraccionarios posea un sistema, mayor precisión y menor rango poseerá (para una misma cantidad total de bits).

Al haber una cantidad limitada de bits de la parte fraccionaria, también se limita la precisión y por lo tanto existe un error máximo en la representación de un número real denominado **error absoluto**. Este error puede ser nulo para los números racionales [1], pero siempre existe en los irracionales.

Por otro lado, 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). Por ejemplo, suponer que una persona llega 10 minutos tarde a una clase. Eso es mucho más "grave"que llegar 10 más tarde de lo previsto luego de un viaje de 2000 kilómetros. Digamos que 10 minutos de retraso en un tiempo total mas grande resulta aceptable.

Esto indica una limitación en los sistemas de punto fijo, donde al representar valores muy pequeños (cercanos al cero) o valores muy grandes se comete el mismo **error absoluto máximo**, y este no tiene la misma importancia en los extremos del rango como entorno al cero. Como un segundo ejemplo, considerar un sistema de punto fijo cuya resolución sea de 0,5

1. Si se intenta representar el valor *100.000,3* se lo aproxima con el valor *100.000,5* con un error de 0,2.

2. Si se intenta representar el valor *0,26* se lo aproxima con el valor *0,5* con un error de 0,24.

En el primer caso el error obtenido es despreciable, mientras que en el segundo es muy importante

Con la intención de diseñar un sistema mas flexible en cuanto al error de representación, se presentan los sistemas de **punto flotante**. Estos permiten

[1] Un número x es racional si existen $a \in \mathbb{Z}$ y $b \in \mathbb{Z}$ tales que $x = \frac{a}{b}$. Los racionales tiene una cantidad finita de cifras o una periodicidad en su parte fraccionaria.

tener una resolución variable, teniendo una resolución pequeña en la zona cercana al cero y una resolución grande en las zonas alejadas del cero.

Para esto, los valores se representan mediante dos campos: una **mantisa** y un **exponente**:

$$m \times 2^e$$

Ejemplos:

$$6200000000000000000000 = 62 * 10^{20}$$

$$0,00000000000000000062 = 62 * 10^{-20}$$

Si se quisiera aprovechar esto en la representación de números dentro de una computadora, se utilizaría base 2: Por ejemplo, suponer la cadena `11000000`. Su interpretación en punto fijo BSS(8,0) es:

$$I(11000000) = 2^7 + 2^6 = 64 + 128 = 192$$

Si se busca "comprimirla" debemos correr la coma 6 lugares, es decir remover los 0s de la derecha y escalar una cadena más chica (`11`).

$$I(11000000) = I(11) * 2^6 = (2^2 + 2^1) * 2^6 = 3 * 64 = 192$$

Entonces al almacenar esta información se tiene una mantisa `11` y un exponente `110`, que representa el 6:

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

La base del exponente no hace falta registrarla porque es siempre 2. Entonces a la hora de almacenarlas, se concatenan ambas subcadenas: `11110` (mantisa/exponente), aunque podría elegirse concatenar en otro orden (`11011`). Así, con pocos bits de exponente se pueden guardar valores muy grandes. Considerar un sistema con 2 bits de mantisa, que permite representar (en bss) hasta el número 3, y por lo tanto se pueden "comprimir" 3 posiciones (o ceros). Con 4 bits se puede representar (en bss) hasta el número 15, y por lo tanto se pueden "comprimir" 15 posiciones (o ceros).

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

Entonces la mantisa (m) y el exponente (e) son cadenas binarias que codifican valores en sistemas de numeración enteros. La mantisa, además, admite algún sistema de punto fijo.

2. Interpretación

Para interpretar una cadena en un sistema de punto flotante es necesario conocer la especificación del sistema. Suponer por ejemplo un sistema cuya mantisa está codificada en $BSS(4)$ y el exponente en $BSS(3)$, organizado de la siguiente manera.

mantisa BSS(4)	exponente BSS(3)

Es decir que las cadenas del sistema en cuestión tienen 7 bits de longitud, y para interpretar cualquiera de ellas se la **debe segmentar para interpretar por separado mantisa y exponente, aplicando las reglas de interpretación de los sistemas correspondientes**. Por ejemplo, interpretar la cadena `1101101` requiere interpretar la cadena `1101` en $BSS(4)$ y la cadena `101` en el sistema $BSS(3)$, para finalmente calcular

$$m \times 2^e$$

Interpretación de la mantisa:

$$m = I_{bss(4)}(1101) = 2^3 + 2^2 + 2^0 = 13$$

Interpretación del exponente:

$$e = I_{bss(3)}(101) = 2^2 + 2^0 = 5$$

Por último se reemplazan los valores obtenidos en la formula

$$m \times 2^e$$

obteniéndose:

$$13 \times 2^5 = 13 \times 31 = 403$$

3. Rango y resolución

Para analizar el rango de un sistema, de manera general, se construye la cadena que representa al número más chico y la que representa al número más grande. En particular a los sistemas de punto flotante, se debe tener en cuenta el exponente, el cual puede tener un sistema distinto al de la mantisa. El valor máximo tiene entonces la mantisa máxima y el máximo exponente, mientras el valor mínimo se compone con la mantisa mínima y el exponente máximo.

Para ejemplificar, considerar el siguiente sistema

`e:bss(2)`	`m:bss(2)`

Como el sistema $BSS(2)$ no admite números negativos, la mantisa mínima es: `00`, cuya interpretación es:

$$M_{min} = I_{bss}(00) = 0$$

La mantisa máxima es: `11`, y su interpretación:

$$M_{max} = I_{bss}(11) = 2^1 + 2^0 = 3$$

El exponente máximo es: `11`, y representa:

$$E = I_{bss}(11) = 2^1 + 2^0 = 3$$

Por lo tanto el rango es el siguiente:

$$Rango = [M_{min} \times 2^E; M_{max} \times 2^E] = [0 \times 2^3; 3 \times 2^3] = [0; 24]$$

En la siguiente tabla se detallan las interpretaciones de todas las cadenas del sistema anterior y al graficar las cadenas con respecto a los valores que representan se obtiene una distribución como la de la figura 1.

cadena	exponente	mantisa	$N = M \times 2^E$
0000	$I_{bss}(00) = 0$	$I_{bss}(00) = 0$	$N = 0 \times 2^0 = 0$
0001	$I_{bss}(00) = 0$	$I_{bss}(01) = 1$	$N = 1 \times 2^0 = 1$
0010	$I_{bss}(00) = 0$	$I_{bss}(10) = 2$	$N = 2 \times 2^0 = 2$
0011	$I_{bss}(00) = 0$	$I_{bss}(11) = 3$	$N = 3 \times 2^0 = 3$
0100	$I_{bss}(01) = 1$	$I_{bss}(00) = 0$	$N = 0 \times 2^1 = 0$
0101	$I_{bss}(01) = 1$	$I_{bss}(01) = 1$	$N = 1 \times 2^1 = 2$
0110	$I_{bss}(01) = 1$	$I_{bss}(10) = 2$	$N = 2 \times 2^1 = 4$
0111	$I_{bss}(01) = 1$	$I_{bss}(11) = 3$	$N = 3 \times 2^1 = 6$
1000	$I_{bss}(10) = 2$	$I_{bss}(00) = 0$	$N = 0 \times 2^2 = 0$
1001	$I_{bss}(10) = 2$	$I_{bss}(01) = 1$	$N = 1 \times 2^2 = 4$
1010	$I_{bss}(10) = 2$	$I_{bss}(10) = 2$	$N = 2 \times 2^2 = 8$
1011	$I_{bss}(10) = 2$	$I_{bss}(11) = 3$	$N = 3 \times 2^2 = 12$
1100	$I_{bss}(11) = 3$	$I_{bss}(00) = 0$	$N = 0 \times 2^3 = 0$
1101	$I_{bss}(11) = 3$	$I_{bss}(01) = 1$	$N = 1 \times 2^3 = 8$
1110	$I_{bss}(11) = 3$	$I_{bss}(10) = 2$	$N = 2 \times 2^3 = 16$
1111	$I_{bss}(11) = 3$	$I_{bss}(11) = 3$	$N = 3 \times 2^3 = 24$

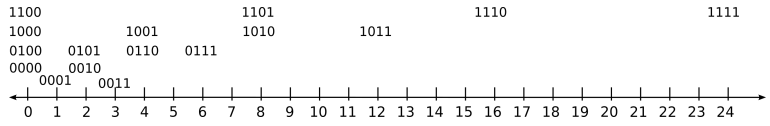


Figura 1: Mantisa en BSS(2) y exponente en BSS(2)

Como es posible apreciar en la figura anterior, muchas cadenas representan al mismo valor (discusión que se retomará en la sección **??**) y además los valores representables no son equidistantes, como si ocurre en los sistemas enteros y de punto fijo. Esta característica recibe el nombre de **resolución variable**. En el sistema de ejemplo desarrollado, la resolución varía entre 1 (resolución mínima del sistema) y 8 (resolución máxima).

Como se puede comprobar, el rango y la resolución (o rango de resoluciones) de cualquier sistema de punto flotante está determinado por los sistemas subyacentes. Por ejemplo, considerar el caso donde tanto mantisa como exponente permiten representar un rango positivo (ver figura 1) en contraposición a un sistema donde la mantisa permite representar números negativos (ver figura 2).

Podría concluirse que el rango del sistema está relacionado con el de la mantisa pues si esta es simétrica con respecto al cero, el sistema de punto flotante también lo es. Por otro lado, es importante analizar cómo el sistema del exponente afecta al rango y la resolución del sistema. Considerar dos sistemas con mantisa BSS donde el primero tiene exponentes positivos (ver figura 1) y el otro tiene un sistema en el exponente que permite representar números negativos (ver figura 3).

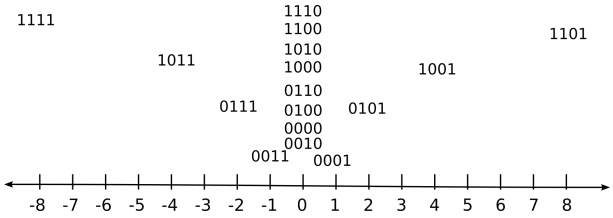


Figura 2: Mantisa en SM(2) y exponente en BSS(2)

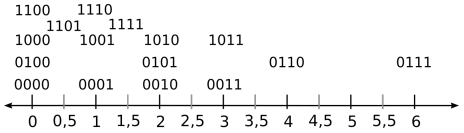


Figura 3: Mantisa en BSS(2) y exponente en SM(2)

En la siguiente tabla se presenta una comparación entre ambos:

Mantisa	Exponente	Mínimo valor representable	Máximo valor representable	Resolución mínima	Resolución máxima
$BSS(2)$	$BSS(2)$	0	24	1	8
$BSS(2)$	$SM(2)$	0	6	0,5	2

Es posible entonces concluir que el exponente, cuando su sistema permite representar números negativos, permite resoluciones menores a los enteros. Por último ver el gráfico de la figura 4, que es similar al de la figura 2 pero mas aglutinado en torno al cero.

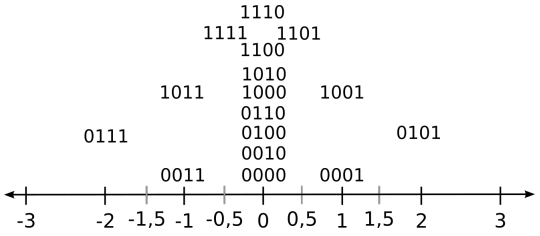


Figura 4: Mantisa: SM(2), Exponente: SM(2)

3.1. Mantisa entera vs mantisa fraccionaria

Calculemos la resolución máxima y mínima del siguiente sistema de punto flotante con

Mantisa: $SM(4+1,4)$

Exponente: $CA2(4)$

a) Para la resolución mínima necesito el mínimo exponente, es decir, 1000 en nuestro sistema. El cual representa, como ya vimos, al número -8.

Las cadenas que voy a utilizar son 0000 1000 y 0001 1000, las cuales son consecutivas y como las mantisas están normalizadas y tienen bit implícito, la primera vale

$$2^{-1} \times 2^{-8}$$

y la segunda

$$(2^{-1} + 2^{-4}) \times 2^{-8}$$

Para saber la resolución tengo que restar los valores y para eso distribuyo en el segundo valor, quedando:

$$2^{-1} \times 2^{-8} + 2^{-4} \times 2^{-8}$$

Y si resto los valores, me queda:

$$2^{-1} \times 2^{-8} + 2^{-4} \times 2^{-8} - 2^{-1} \times 2^{-8}$$

Por lo tanto, mi resolución mínima es:

$$2^{-4} \times 2^{-8}$$

b) Para la resolución máxima necesito el máximo exponente, es decir, 0111 en nuestro sistema. El cual representa, como ya vimos, al número 7.

Las cadenas que voy a utilizar son 0000 0111 y 0001 0111, las cuales son consecutivas y como las mantisas están normalizadas y tienen bit implícito, la primera vale

$$2^{-1} \times 2^{7}$$

y la segunda

$$(2^{-1} + 2^{-4}) \times 2^{7}$$

Para saber la resolución tengo que restar los valores y para eso distribuyo en el segundo valor, quedando:

$$2^{-1} \times 2^{7} + 2^{-4} \times 2^{7}$$

Y si resto los valores, me queda:

$$2^{-1} \times 2^{7} + 2^{-4} \times 2^{7} - 2^{-1} \times 2^{7}$$

Por lo tanto, mi resolución máxima es:

$$2^{-4} \times 2^{7}$$

4. 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 implícwito 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 suiguiente 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)

Nota:

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 representa 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.

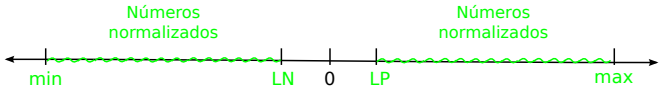
El exponente lo dice todo

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.

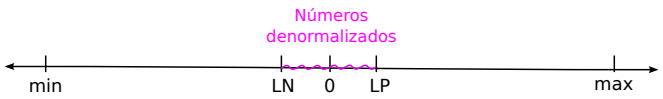
Adicionalmente pueden representarse valores subnormales o denormalizados, es decir números **no normalizados**, de la forma $\pm 0, X * 2^{\delta}$, que se extienden en el rango comprendido entre el mayor número normal negativo y el menor número normal positivo. Dicho exponente especial δ tiene el valor -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 saturado (`1..1`) y mantisa nula. 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	0..0	$\pm\infty$
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**
$$m = I_{bss(23,23)}(01000001011100000000000) =$$
$$= 2^{-2} + 2^{-8} + 2^{-10} + 2^{-11} + 2^{-12}$$

Referencias

[1] Williams Stallings, *Computer Organization and Architecture*, octava edicion, Editorial Prentice Hall, 2010. **Capitulo 8: Artimética del Computador, subcapítulo 8.2: Representación en coma fija.**