13. Construya las tablas de multiplicación para los números base 3.

14. Multiplique 0111 por 0011 en binario.

15. Escriba un programa que tome un número decimal con signo como cadena ASCII e imprima su representación en complemento a dos en binario, octal y hexadecimal.

16. Escriba un programa que tome dos cadenas de 32 caracteres ASCII que contengan ceros y unos; cada cadena representa un número binario de 32 bits en complemento a dos. El programa deberá imprimir su suma como una cadena de 32 caracteres ASCII con ceros y unos.

B

NÚMEROS DE PUNTO FLOTANTE

En muchos cálculos el intervalo de números que se usan es muy grande. Por ejemplo, en un cálculo astronómico podrían intervenir la masa del electrón, 9×10^{-28} gramos, y la masa del Sol, 2×10^{33} gramos. Este intervalo es mayor que 10^{60}. Tales números podrían representarse con

00000000000000000000000000000000.00000000000000000000000000000009
20000000000000000000000000000000.00000000000000000000000000000000

y todos los cálculos podrían efectuarse llevando 34 dígitos a la izquierda del punto decimal y 28 posiciones a la derecha. Ello permitiría tener 62 dígitos significativos en el resultado. En una computadora binaria se podría usar aritmética de múltiple precisión para tener suficiente significancia. Sin embargo, la masa del Sol no se conoce ni siquiera con una precisión de cinco dígitos significativos, mucho menos 62. De hecho, pocas mediciones de cualquier tipo se pueden (o tienen que) efectuar con una precisión de 62 dígitos significativos. Aunque sería posible llevar todos los resultados intermedios con 62 dígitos significativos y luego desechar 50 o 60 de ellos antes de imprimir los resultados finales, hacerlo sería un desperdicio de tiempo de CPU y de memoria.

Lo que se necesita es un sistema para representar números en el que el intervalo de los números que pueden expresarse sea independiente del número de dígitos significativos. En este apéndice veremos un sistema así, que se basa en la notación científica de uso común en física, química e ingeniería.

B.1 PRINCIPIOS DE PUNTO FLOTANTE

Una forma de separar el intervalo y la precisión es expresar los números en la conocida notación científica

$$n = f \times 10^e$$

donde f es la **fracción** o **mantisa** y e es un entero positivo o negativo llamado **exponente**. La versión para computadora de esta notación se llama **punto flotante**. He aquí algunos ejemplos de números expresados en esta forma:

3.14	$= 0.314$	$\times 10^{1}$	$= 3.14$	$\times 10^{0}$
0.000001	$= 0.1$	$\times 10^{-5}$	$= 1.0$	$\times 10^{-6}$
1941	$= 0.1941$	$\times 10^{4}$	$= 1.941$	$\times 10^{3}$

El intervalo está determinado por el número de dígitos del exponente, y la precisión está determinada por el número de dígitos de la fracción. Puesto que hay más de una forma de representar un número dado, normalmente se escoge una forma como estándar. A fin de investigar las propiedades de este método de representación de los números, consideremos una representación, R, con una fracción de tres dígitos y signo dentro del intervalo $0.1 \le |f| < 1$ o cero, y un exponente de dos dígitos con signo. La magnitud de estos números varía entre $+0.100 \times 10^{-99}$ y $0.999 \times 10^{+99}$, un intervalo de casi 199 órdenes de magnitud, y sin embargo sólo se necesitan cinco dígitos y dos signos para almacenar un número.

Podemos usar números de punto flotante para modelar el sistema de números reales de las matemáticas, aunque hay varias diferencias importantes. La figura B-1 muestra un esquema muy exagerado de la línea de los números reales. La línea real se divide en siete regiones:

1. Números negativos grandes menores que -0.999×10^{99}.
2. Números negativos entre -0.999×10^{99} y -0.100×10^{-99}.
3. Números negativos pequeños con una magnitud menor que 0.100×10^{-99}.
4. Cero.
5. Números positivos pequeños con una magnitud menor que 0.100×10^{-99}.
6. Números positivos entre 0.100×10^{-99} y 0.999×10^{99}.
7. Números positivos grandes mayores que 0.999×10^{99}.

Una diferencia importante entre el conjunto de los números que pueden representarse con tres dígitos en la fracción y dos en el exponente, y los números reales, es que los primeros no se pueden usar para expresar números en las regiones 1, 3, 5 o 7. Si el resultado de una operación aritmética es un número en las regiones 1 o 7 —por ejemplo, $10^{60} \times 10^{60} = 10^{120}$— ocurrirá un **error de desbordamiento** y la respuesta será incorrecta. La razón está en la naturaleza finita de la representación de los números y es inevitable. Así mismo, un resultado

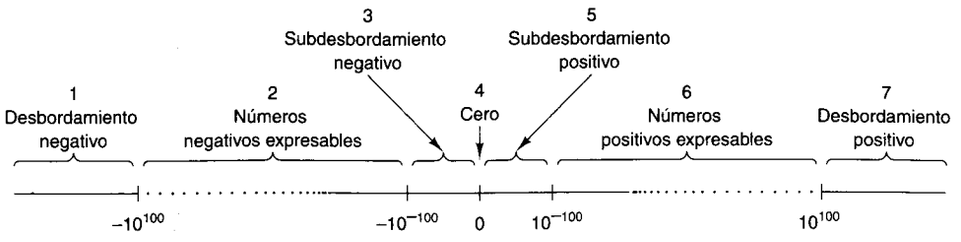

Figura B-1. La línea de los números reales se puede dividir en siete regiones.

en la región 3 o 5 tampoco puede expresarse. Esta situación se llama **error de subdesbordamiento**. Este tipo de error es menos grave que el de desbordamiento, porque en muchos casos 0 es una aproximación satisfactoria para números de las regiones 3 y 5. Un saldo en el banco de 10^{-102} dólares no es mucho mejor que un saldo de 0.

Otra diferencia importante entre los números de punto flotante y los números reales es su densidad. Entre cualesquier dos números reales x y y hay otro número real, por más cercano que esté x a y. Esta propiedad proviene del hecho de que para cualesquier números reales distintos, x y y, $z = (x + y)/2$ es un número real que está entre ellos. Los números reales forman un continuo.

Los números de punto flotante, en cambio, no forman un continuo. Se pueden expresar exactamente 179,100 números positivos en el sistema de cinco dígitos y dos signos que usamos antes, 179,100 números negativos y 0 (que se puede expresar de muchas formas), para un total de 358,201 números. Del número infinito de números reales que hay entre -10^{+100} y $+0.999 \times 10^{99}$, sólo 358,201 se pueden especificar con esta notación. Dichos números se simbolizan con los puntos de la figura B-1. Es muy posible que el resultado de un cálculo sea uno de los otros números, aunque esté en la región 2 o 6. Por ejemplo, $+0.100 \times 10^{3}$ dividido entre 3 no se puede expresar *con exactitud* en nuestro sistema. Si el resultado de un cálculo no se puede expresar en la representación numérica que se está usando, lo que obviamente debe hacerse es usar el número más cercano que sí pueda expresarse. Este proceso se llama **redondeo**.

La distancia entre números adyacentes que pueden expresarse no es constante a lo largo de las regiones 2 o 6. La separación entre $+0.998 \times 10^{99}$ y $+0.999 \times 10^{99}$ es inmensamente mayor que la distancia entre $+0.998 \times 10^{0}$ y $+0.999 \times 10^{0}$. Sin embargo, si expresamos la distancia entre un número y su sucesor como un porcentaje del número, no hay variación sistemática dentro de la región 2 o 6. En otras palabras, el **error relativo** introducido por el redondeo es aproximadamente el mismo para números pequeños que para números grandes.

Aunque la explicación anterior se refirió a un sistema de representación con una fracción de tres dígitos y un exponente de dos dígitos, las conclusiones que se sacan son válidas también para otros sistemas de representación. Modificar el número de dígitos de la fracción o exponente simplemente desplaza los límites de las regiones 2 y 6, y cambia el número de puntos expresables y que aquéllas contienen. Incrementar el número de dígitos en la fracción aumenta la densidad de puntos y por tanto mejora la precisión de las aproximaciones. In-

crementar el número de dígitos del exponente aumenta el tamaño de las regiones 2 y 6 al encoger las regiones 1, 3, 5 y 7. En la figura B-2 se muestran las fronteras aproximadas de la región 6 para números decimales de punto flotante con diversos tamaños de la fracción y el exponente.

Dígitos en la fracción	Dígitos en el exponente	Cota inferior	Cota superior
3	1	10^{-12}	10^{9}
3	2	10^{-102}	10^{99}
3	3	10^{-1002}	10^{999}
3	4	10^{-10002}	10^{9999}
4	1	10^{-13}	10^{9}
4	2	10^{-103}	10^{99}
4	3	10^{-1003}	10^{999}
4	4	10^{-10003}	10^{9999}
5	1	10^{-14}	10^{9}
5	2	10^{-104}	10^{99}
5	3	10^{-1004}	10^{999}
5	4	10^{-10004}	10^{9999}
10	3	10^{-1009}	10^{999}
20	3	10^{-1019}	10^{999}

Figura B-2. Límites superior e inferior aproximados de los números decimales de punto flotantes expresables (no normalizados).

En las computadoras se usa una variación de esta representación. Por eficiencia, la exponenciación es base 2, 4, 8 o 16, no 10, y entonces la fracción consiste en una cadena de dígitos binarios, base 4, octales o hexadecimales. Si el primero de esos dígitos a la izquierda es cero, todos los dígitos pueden desplazarse una posición a la izquierda y el exponente reducirse en 1, sin alterar el valor del número (siempre que no haya subdesbordamiento). Se dice que una fracción cuyo dígito de la izquierda no es cero está **normalizada**.

Los números normalizados generalmente son preferibles a los no normalizados porque sólo hay una representación normalizada, mientras que hay muchas representaciones no normalizadas. En la figura B-3 se dan ejemplos de números de punto flotante normalizados para dos bases de exponenciación. En estos ejemplos se muestra una fracción de 16 bits (incluido el bit de signo) y un exponente de siete bits empleando notación exceso de 64. El punto base está a la izquierda del bit de la extrema izquierda de la fracción, es decir, a la derecha del exponente.

B.2 ESTÁNDAR DE PUNTO FLOTANTE IEEE 754

Hasta cerca de 1980, cada fabricante de computadoras tenía su propio formato de punto flotante. Sobra decir que todos eran diferentes. Peor aún, algunos de ellos efectuaban aritmética incorrecta porque la de punto flotante tiene algunas sutilezas que no son obvias para el diseñador de hardware ordinario.

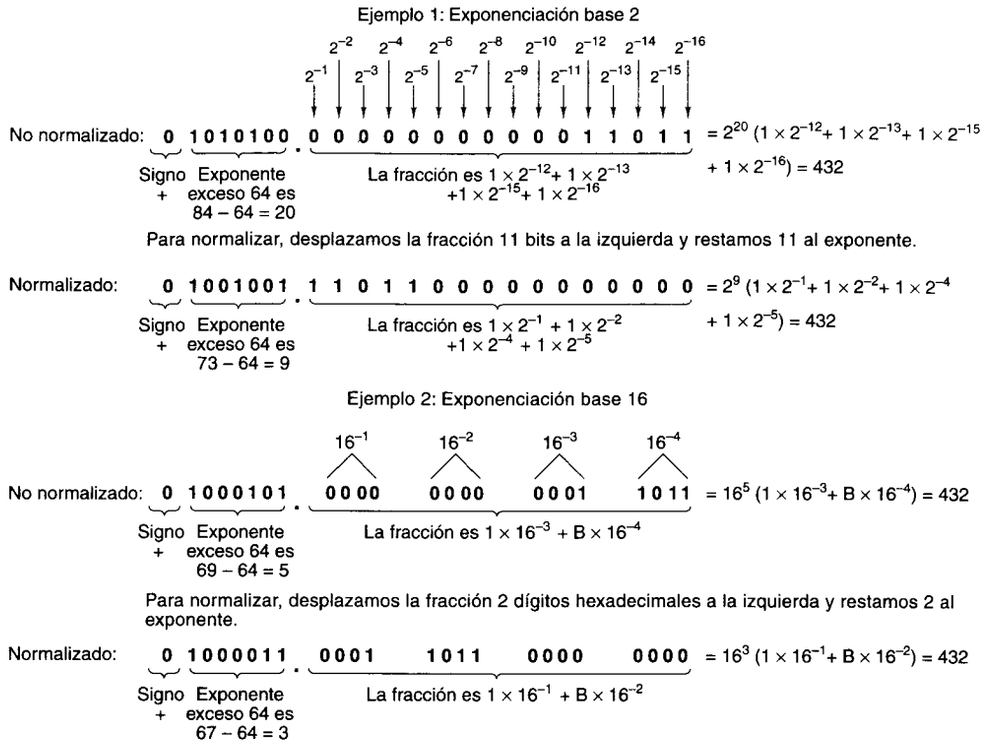

Figura B-3. Ejemplos de números de punto flotante normalizados.

A fin de rectificar esta situación, a fines de los años setenta el IEEE formó un comité para estandarizar la aritmética de punto flotante. La meta no sólo era poder intercambiar datos de punto flotante entre diferentes computadoras, sino también proporcionar a los diseñadores de hardware un diseño que se sabía era correcto. Los trabajos resultantes dieron pie al estándar IEEE 754 (IEEE, 1985). Casi todas las CPU actuales (incluidas las Intel, SPARC y JVM que estudiamos en este libro) tienen instrucciones de punto flotante que se ajustan al estándar de punto flotante IEEE. A diferencia de muchos estándares, que tienden a ser términos medios con los que nadie está contento, éste no está nada mal, en parte porque fue en gran medida el trabajo de una sola persona, el profesor de matemáticas de Berkeley William Kahan. En el resto de la sección describiremos el estándar.

El estándar define tres formatos: precisión sencilla (32 bits), doble precisión (64 bits) y precisión extendida (80 bits). El formato de precisión extendida pretende reducir los errores de redondeo; se usa primordialmente dentro de unidades aritméticas de punto flotante, por lo que no hablaremos más de él. Los formatos tanto de precisión sencilla como doble usan la base 2 para las fracciones y notación en exceso para los exponentes. Los formatos se muestran en la figura B-4.

Ambos formatos comienzan con un bit de signo para el número en su totalidad: 0 para positivo y 1 para negativo. Luego viene el exponente, que usa exceso en 127 en el caso de la

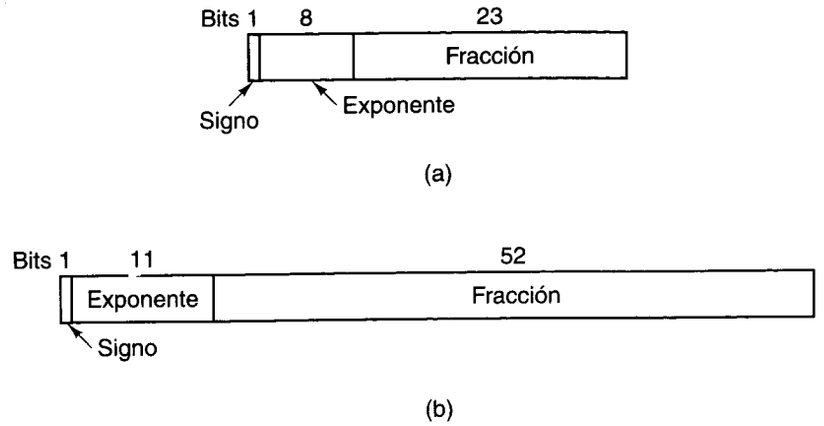

Figura B-4. Formatos de punto flotante IEEE. (a) Precisión sencilla. (b) Doble precisión.

precisión sencilla y exceso en 1023 para la doble precisión. Los exponentes mínimo (0) y máximo (255 y 2047) no se usan con números normalizados; tienen usos especiales que se describen más adelante. Por último, tenemos las fracciones, de 23 y 52 bits, respectivamente.

Una fracción normalizada comienza con un punto binario, seguido de un bit 1, y luego el resto de la fracción. Siguiendo una práctica iniciada en el PDP-11, los autores del estándar se dieron cuenta de que el bit inicial de la fracción no tiene que almacenarse, ya que puede simplemente darse por hecho que está presente. Por tanto, el estándar define la fracción de una forma un poco distinta a lo acostumbrado. La fracción consiste en un bit 1 implícito, un punto binario implícito, y 23 o 52 bits arbitrarios. Si todos los bits de la fracción son ceros, la fracción tiene el valor numérico 1.0; si todos son unos, la fracción es numéricamente un poco menos que 2.0. A fin de evitar confusiones con una fracción convencional, la combinación del 1 implícito, el punto binario implícito y los 23 o 52 bits explícitos se llama **significando** en vez de fracción o mantisa. Todos los números normalizados tienen un significando, s, dentro del intervalo $1 \leq s < 2$.

Las características numéricas de los números de punto flotante IEEE se dan en la figura B-5. Consideremos, por ejemplo, los números 0.5, 1 y 1.5 en formato de precisión sencilla normalizado. Éstos se representan en hexadecimal como 3F000000, 3F800000 y 3FC00000, respectivamente.

Uno de los problemas tradicionales con los números de punto flotante es cómo manejar el subdesbordamiento, el desbordamiento y los números no inicializados. El estándar IEEE aborda los problemas explícitamente, tomando su enfoque en parte de la CDC 6600. Además de los números normalizados, el estándar tiene otros cuatro tipos numéricos, que se describen a continuación y se muestran en la figura B-6.

Surge un problema cuando el resultado de un cálculo tiene una magnitud menor que el número de punto flotante normalizado más pequeño que puede representarse en el sistema. Antes, casi todo el hardware adoptaba uno de dos enfoques: igualar el resultado a cero y continuar, o causar una trampa de desbordamiento de punto flotante hacia cero. Ninguno de

Concepto	Precisión sencilla	Doble precisión
Bits del signo	1	1
Bits del exponente	8	11
Bits de la fracción	23	52
Total de bits	32	64
Sistema de exponente	Exceso en 127	Exceso en 1023
Intervalo del exponente	−126 a +127	−1022 a + 1023
Número normalizado más pequeño	2^{-126}	2^{-1022}
Número normalizado más grande	aprox. 2^{128}	aprox. 2^{1024}
Intervalo decimal	aprox. 10^{-38} a 10^{38}	aprox. 10^{-308} a 10^{308}
Número desnormalizado más pequeño	aprox. 10^{-45}	aprox. 10^{-324}

Figura B-5. Características de los números de punto flotante IEEE.

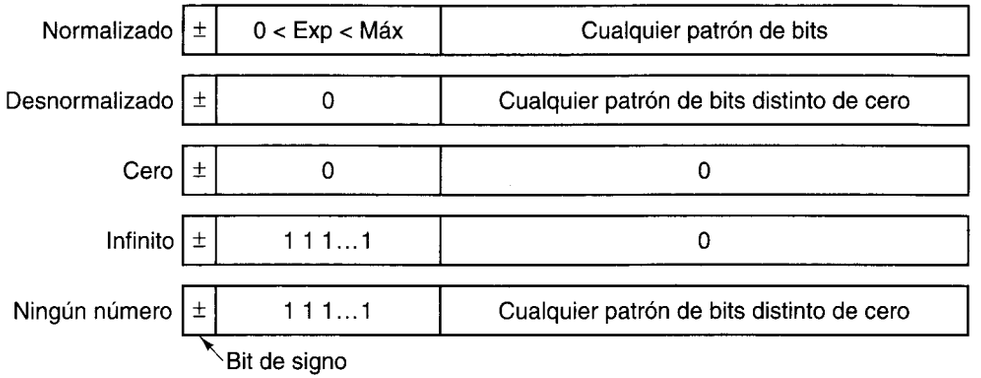

Figura B-6. Representaciones numéricas IEEE.

éstos es realmente satisfactorio, por lo que IEEE inventó los **números desnormalizados**. Estos números tienen un exponente cero y una fracción dada por los siguientes 23 o 52 bits. El bit implícito a la izquierda del punto binario ahora se convierte en 0. Los números desnormalizados se pueden distinguir de los normalizados porque estos últimos no pueden tener un exponente cero.

El número de precisión sencilla normalizado más pequeño tiene 1 como exponente y 0 como fracción, y representa 1.0×2^{-126}. El número desnormalizado más grande tiene 0 como exponente y sólo unos en la fracción, y representa cerca de $0.9999999 \times 2^{-127}$, que es casi la misma cosa. Sin embargo, una cosa que debemos tener presente es que este número sólo tiene 23 bits de significancia, en vez de 24 como todos los números normalizados.

Si los cálculos reducen aún más este resultado, el exponente sigue siendo 0, pero los primeros bits de la fracción se van convirtiendo en ceros, lo que reduce tanto el valor como el número de bits significativos de la fracción. El número desnormalizado más pequeño distinto

de cero consiste en un 1 en el bit de la extrema derecha, con todos los demás 0. El exponente representa 2^{-127} y la fracción representa 2^{-23}, de modo que el valor es 2^{-150}. Este esquema hace posible un subdesbordamiento "gradual", sacrificando significancia en lugar de saltar a 0 cuando el resultado no puede expresarse como número normalizado.

Este esquema cuenta con dos ceros, positivo y negativo, determinados por el bit de signo. Ambos tienen un exponente de 0 y una fracción de 0. Aquí, también, el bit a la izquierda del punto binario es implícitamente 0 en lugar de 1.

El desbordamiento no puede manejarse de forma gradual. No quedan más combinaciones de bits. En vez de ello, se proporciona una representación especial para infinito, que consiste en un exponente solamente de unos (lo cual no está permitido en los números normalizados) y una fracción de 0. Este número puede usarse como operando y se comporta según las reglas matemáticas usuales para el infinito. Por ejemplo, infinito más cualquier cosa es infinito, y cualquier número finito dividido entre infinito es cero. Así mismo, cualquier número finito dividido entre cero da infinito.

¿Y qué pasa con infinito dividido entre infinito? El resultado no está definido. Para manejar este caso se proporciona otro formato especial, llamado **ningún número** (**NaN**, *Not a Number*), que también puede usarse como operando con resultados predecibles.

PROBLEMAS

1. Convierta los números siguientes al formato IEEE de precisión sencilla. Represente los resultados como ocho dígitos hexadecimales.

   a. 9
   b. 5/32
   c. –5/32
   d. 6.125

2. Convierta los siguientes números de punto flotante IEEE de precisión sencilla de hexadecimal a decimal:

   a. 42E48000H
   b. 3F880000H
   c. 00800000H
   d. C7F00000H

3. El formato de los números de punto flotante de precisión sencilla en la 370 tiene un exponente de 7 bits en el sistema exceso de 64, y una fracción que contiene 24 bits más un bit de signo, con el punto binario en el extremo izquierdo de la fracción. La base de exponenciación es 16. El orden de los campos es bit de signo, exponente, fracción. Exprese el número 7/64 como número normalizado en este sistema, en hexadecimal.

4. Los siguientes números binarios de punto flotante consisten en un bit de signo, un exponente base 2 en exceso de 64 y una fracción de 16 bits. Normalícelos.

   a. 0 1000000 0001010100000001
   b. 0 0111111 0000001111111111
   c. 0 1000011 1000000000000000

5. Para sumar dos números de punto flotante hay que ajustar los exponentes (desplazando la fracción) de modo que sean iguales. Luego se pueden sumar las fracciones y normalizar el resultado, si es necesario. Sume los números IEEE de precisión sencilla 3EE00000H y 3D800000H y exprese el resultado normalizado en hexadecimal.

6. La Compañía de Computadoras Económicas decidió sacar una máquina que maneje números de punto flotante de 16 bits. El Modelo 0.001 tiene un formato de punto flotante con un bit de signo, un exponente de 7 bits en exceso de 64, y una fracción de 8 bits. El Modelo 0.002 tiene un bit de signo, un exponente de 5 bits en exceso de 16, y una fracción de 10 bits. Ambas usan exponenciación base 2. ¿Cuáles son los números normalizados más grande y más pequeño en cada modelo? ¿Cuántos dígitos decimales de precisión tiene aproximadamente cada uno? ¿Compraría alguno de ellos?

7. Hay una situación en la que una operación con dos números de punto flotante puede causar una reducción drástica del número de bits significativos en el resultado. ¿Cuál es?

8. Algunos chips de punto flotante tienen incorporada una instrucción de raíz cuadrada. Un posible algoritmo es iterativo (por ejemplo, el de Newton-Raphson). Los algoritmos iterativos necesitan una aproximación inicial y luego la mejoran continuamente. ¿Cómo se puede obtener una raíz cuadrada aproximada rápida de un número de punto flotante?

9. Escriba un procedimiento que sume dos números de punto flotante IEEE de precisión sencilla. Cada número se representa con un arreglo booleano de 32 elementos.

10. Escriba un procedimiento para sumar dos números de punto flotante de precisión sencilla que usan base 16 para el exponente y base 2 para la fracción pero no tienen un bit 1 implícito a la izquierda del punto binario. Un número normalizado tiene 0001, 0010, ..., 1111 como los cuatro bits de extrema izquierda de la fracción, pero no 0000. Un número se normaliza desplazando la fracción a la izquierda 4 bits y sumando 1 al exponente.