Capítulo 3: PUNTO FLOTANTE

3.1 Introducción

Se denominan ***Números en Punto Flotante*** a las representaciones internas del procesador que modelan a los números reales. En forma externa, se representan números con punto decimal tal como 3.1415926 o en notación científica 9.512×10^{-5}, con un solo dígito a la izquierda del punto decimal; es decir, 9.512×10^{-4}.

3.2 Normalización de números

Se dice que el número está normalizado si el dígito a la izquierda del punto no es cero. En el ejemplo anterior, 95.12×10^{-5}. De manera normalizada quedaría: 9.512×10^{-4}. En el caso de computadores, se emplea números en sistema binario, y con un número finito de dígitos.

3.3 Propósito de números en punto flotante

Como la memoria de los computadores es limitada, no puedes almacenar números con precisión infinita, no importa si usas fracciones binarias o decimales, en algún momento tienes que cortar. Pero ¿cuánta precisión se necesita? ¿Y *dónde* se necesita? ¿Cuántos dígitos enteros y cuántos fraccionarios?

La idea es descomponer los números en dos partes:

- Una **mantisa** (también llamada coeficiente o significando) que contiene los dígitos del número, donde las mantisas negativas representan números negativos; es decir, la mantisa corresponde a la parte fraccionaria que es la diferencia entre el número y la parte entera del número. Veamos este ejemplo, en el número decimal 13.8543, la parte entera es 13 y la mantisa=13.8543-13= 0,8543. Pero, cuando el número decimal es negativo; esto es, -13.8543, la parte entera es -14 y la mantisa=-13.8543-(-14)= 0.1457.
- Un **exponente** que indica dónde se coloca el punto decimal (o binario) en relación al inicio de la mantisa. Cuando el exponente es negativo representará a un número menor que uno.

Este formato cumple todos los requisitos:

- Puede representar números de órdenes de magnitud enormemente dispares (limitado por la longitud del exponente).
- Proporciona la misma precisión relativa para todos los órdenes (limitado por la longitud de la mantisa).
- Permite cálculos entre magnitudes: multiplicar un número muy grande y uno muy pequeño conserva la precisión de ambos en el resultado.
- Los números de punto flotante decimales normalmente se expresan en notación científica con un punto explícito siempre entre el primer y el segundo dígitos. El

exponente o bien se escribe explícitamente incluyendo la base, o se usa una **E** para separarlo de la mantisa.

Parte entera	Mantisa	Exponente	Notación científica	Valor en punto fijo
1	0.5	4	1.5×10^{4}	15000
5	0	-3	5×10^{-3}	0.005
6	0.667	-11	6.667E-11	0.0000000000667
-3	0.999	2	-2.001×10^{2}	-200.1

Tabla 1: formato de configuración de punto flotante

3.4 Cifras significativas

Las *cifras significativas* representan el uso de una o más escalas de incertidumbre en determinadas aproximaciones. Se dice que 4.7 tiene 2 cifras significativas, mientras que 4.70 tiene 3. Para distinguir los ceros que son significativos de los que no lo son, estos últimos suelen indicarse como potencias de 10 en notación científica, por ejemplo 5724 será 5.724×10^{3}, con 4 cifras significativas.

Cuando no pueden ponerse más cifras significativas, cierta cantidad de ellas, por ejemplo de tres cifras simplemente, a la tercera cifra se le incrementa un número, si la cifra predecesora es 5 con otras cifras o mayor que 5. Si dicha cifra es menor simplemente se deja igual. Ejemplo 5.3689 consta de 5 cifras significativas, si sólo se pueden mostrar tres cifras, se le suma una unidad a la cifra 6 (6+1=7) ya que la cifra que la precede 8 es mayor que 5, así que queda 5.37 y si el número es menor que cinco: así 5.36489 y se redondea queda 5.36, no aumenta por que la cifra 4 es menor que 5.

Cuando la cifra a redondear está precediendo a 5, simplemente aumente en 1 dicha cifra. Por ejemplo para redondear a 3 cifras los números 12.45 y 0.1855, se observa que en el primer número, el dígito 3 que precede al 5, se incrementa en 1 cifra quedando 12.5 y el segundo número 0.1855, se mantiene su valor, 0.186 o 1.86×10^{-1}.

El uso de las cifras significativas considera que el último dígito de aproximación es incierto, por ejemplo, al determinar el volumen de un líquido con una probeta cuya resolución es de 1 ml, implica una escala de incertidumbre de 0.5 ml. Así se puede decir que el volumen de 6 ml será realmente de 5.5 ml a 6.5 ml. El volumen anterior se representará entonces como (6.0 ± 0.5) ml. En caso de determinar valores más próximos se tendrían que utilizar otros instrumentos de mayor resolución, por ejemplo, una probeta de divisiones más finas y así obtener (6.0 ± 0.1) ml o algo más satisfactorio según la resolución requerida.

3.5 Estándar IEEE 754

El *estándar IEEE 754* ha sido definido por el Instituto de Ingenieros Eléctricos y Electrónicos (*Institute of Electrical and Electronics Engineers*, *IEEE*) y establece dos

formatos básicos para representar a los números reales en la computadora digital: precisión simple y precisión doble.

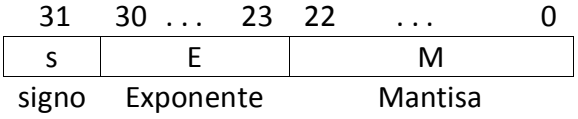
Formato	Bits totales	Bits significativos	Bits del exponente	Número más pequeño	Número más grande
Precisión simple	32	23 + 1 signo	8	~1.2×10^{-38}	~3.4×10^{38}
Precisión doble	64	52 + 1 signo	11	~5.0×10^{-324}	~1.8×10^{308}

Tabla 2: estándar de IEEE 754

La representación del exponente a exceso es $2^{n-1}-1$, mientras que, para la mantisa, normalmente se utiliza Signo Magnitud. Además, la mantisa se suele normalizar colocando el punto decimal a la derecha del bit más significativo.

3.6 Precisión Simple en el Estándar IEEE 754

Para escribir un número en el Estándar IEEE 754 en precisión simple, se escribe el número real usando 32 bits (4 bytes): 1 bit para el signo (s) del número, 23 bits para la mantisa (m) y 8 bits para el exponente (E), que se distribuyen de la siguiente forma:



En la siguiente tabla se resumen los cálculos que hay que realizar para deducir el valor en base 10 de un número escrito en el estándar IEEE 754 con precisión doble:

Signo (s)	*Exponente (exp)*	*Mantisa (m)*	*Valor en base 10*
0 ó 1	0 < **exp** < 255	Indiferente	$(-1)^{s} \cdot 1,\mathbf{m} \cdot 2^{\mathbf{exp}-127}$
0	**exp** = 255	**m** = 0	$+\infty$
1	**exp** = 255	**m** = 0	$-\infty$
0 ó 1	**exp** = 255	**m** ≠ 0	NaN
0 ó 1	**exp** = 0	**m** = 0	0
0 ó 1	**exp** = 0	**m** ≠ 0	$(-1)^{s} \cdot 0,\mathbf{m} \cdot 2^{-126}$

Figura 3: representación de un número real con precisión simple en el estándar IEEE 754.

Ejemplo 3.1: exprese en formato IEEE 754 de precisión simple, el valor del número 45.25

Solución

Pasando la mantisa a base 2 y normalizando, se tiene: $45=101101_2$ y $0.25=0.01_2$

$45.25=45+0.25=101101_2 + 0.01_2=101101.01_2=1.0110101_2*2^5$

Veamos cómo se obtiene el valor del número:

- Como el signo es positivo, el valor del signo es 0.
- Representando en forma binaria el exponente en exceso 2^{n-1}-1 donde n (cantidad de cifras significativas) es 8 y e (exponente representacion externa) que es 5 en este ejemplo. Por lo tanto, el exponente en exceso es 2^{8-1}-1=127. Por consiguiente:
  E= **exponente representacion externa + exponente en exceso**
  =5+127=132=10000100_2
  101101.01_2
- Eliminando el bit implícito de la mantisa (el 1 antes de la fracción binaria), esta quedaría: 01101010000000000000000
- Por lo tanto, la representación final correponde

Signo	Exponente (E)	Mantisa (m)
0	10000100	01101010000000000000000

Ejemplo 3.2: Para escribir el número
101110.$0101011101000011111000011111000100 11_2$

en el estándar IEEE 754 con precisión simple, con exponente en Exceso a 2^{n-1}-1 y mantisa m y signo s, determine su número hexadecimal correspondiente

Primero tendremos que normalizarlo:

101110.$0101011101000011111000011111000100 11_2$
=$1.01110010101110100001111100001111100010011_2$ x 2^5

El exponente, en Exceso a 2^{n-1}-1 en base 10 será:

5 + (2^{8-1}-1)=5 + (2^7-1)=5 + (128-1)=132=10000100 exponente con exceso a 127

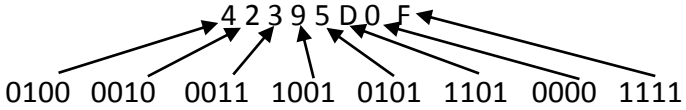
De la mantisa solo se cogerán los 23 bits más significativos:
1.01110010101110100001111

Como el resto de bits no pueden representarse, ya que no caben en la mantisa, entonces se descartarán. Sin embargo, cuando la mantisa se normaliza situando el punto decimal a la derecha del bit más significativo, dicho bit siempre vale 1. Por tanto, se puede prescindir de él, y coger en su lugar un bit más de la mantisa de la parte descartada. De esta forma, la precisión del número representado es mayor. Así, los bits de la mantisa serán: 01110010101110100001111

Al bit omitido se le llama bit implícito. Por otra parte, el bit de signo vale 0, ya que, el número es positivo. En consecuencia, el número se puede representar como:

31	30 ... 23	22 . . . 0
0	10000100	01110010101110100001111
Signo	Exponente	Mantisa

Los programadores, para representar los números reales en este formato, suelen utilizar el Sistema Hexadecimal; efectivamente, tome toda la cadena de bits anterior y agrúpelos de a 4 bits para representar cada cifra hexadecimal.

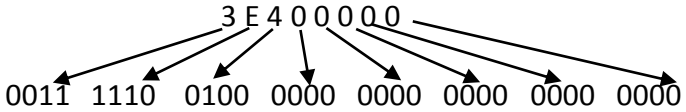


Así pues,
$101110.01010111010000111110000111110001001 1_2$
$=1.01110010101110100001111100001111x2^5=42395D0F_H$ en precisión simple.

En este caso, los números no son exactamente iguales, ya que, con precisión simple no han podido representarse todos los bits de la mantisa.

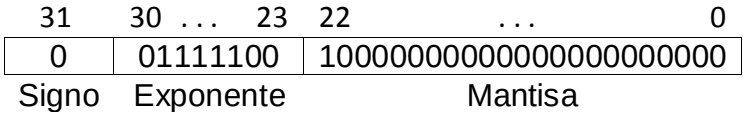
Ejemplo 3.3: Dado el número $3E400000_H$ del estándar IEEE 754 con precisión simple, exponente en Exceso a 2^{n-1}-1 y mantisa m con 1 bit implícito, signo s, averigüe a qué número representa en base 10.
En efecto, pueden realizarse los siguientes pasos:

1º) Convertir $3E400000_H$ a base 2:



2º) Obtener los bits del signo, de la mantisa y del exponente:
Signo (positivo)=0; n (8 bits)=01111100; mantisa (23 bits)=

10000000000000000000000



3º) Pasar el exponente a base 10:
$(01111100_2 - (2^{8-1} - 1)) = 124 - (2^7 - 1)$
$= 124 - (128 - 1) = 124 – 127 = -3$

4º) Escribir el número en notación científica. Para ello, la mantisa se debe escribir con el bit implícito (1), seguido del punto decimal (.) y de los bits de la mantisa (10000000000000000000000), teniendo en cuenta que los ceros por la derecha se pueden despreciar. Por otra parte, el número es positivo, ya que el bit de signo es 0. Por tanto, el número es: $1.10000000000000000000000x2^{-3}=1.1 \times 2^{-3}$

5º) Expresar el número en base 10.

$1.1 \times 2^{-3} = ((2^{0} + 2^{-1}) \times 2^{-3}) = ((1 + 0.5) \times 0.125) = (1.5 \times 0.125) = 0.187510$

Por tanto, $3E400000_H$ (precisión simple) = 1.1×2^{-3} = +**0.187510**

3.8 Precisión Doble en el Estándar IEEE 754

Para escribir un número real en precisión doble, se emplean 64 bits (8 bytes): 1 bit para el signo (s) del número, 52 bits para la mantisa (m) y 11 bits para el exponente (E).

En la siguiente tabla se resumen los cálculos que hay que realizar para deducir el valor en base 10 de un número escrito en el estándar IEEE 754 con precisión doble:

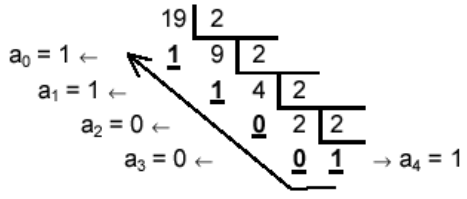
Signo (s)	*Exponente (exp)*	*Mantisa (m)*	© carlospes.com *Valor en base 10*
0 ó 1	0 < **exp** < 2047	Indiferente	$(-1)^{s} \cdot 1,\mathbf{m} \cdot 2^{\mathbf{exp}-1023}$
0	**exp** = 2047	**m** = 0	$+\infty$
1	**exp** = 2047	**m** = 0	$-\infty$
0 ó 1	**exp** = 2047	**m** ≠ 0	NaN
0 ó 1	**exp** = 0	**m** = 0	0
0 ó 1	**exp** = 0	**m** ≠ 0	$(-1)^{s} \cdot 0,\mathbf{m} \cdot 2^{-1022}$

Figura 4: representación de un número real con precisión doble en el estándar IEEE 754.

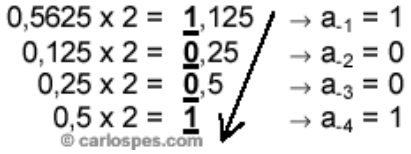
Ejemplo 3.4: Si se quiere escribir el número 19.5625 en el estándar IEEE 754 con precisión doble, exponente en Exceso a 2^{n-1}-1 y mantisa m con 1 bit implícito, signo s, dado en hexadecimal.

Los pasos a seguir son:

1º) Cambiar 19.5625 a base 2. Primero la parte entera:



y, a continuación, la parte fraccionaria:



De modo que, $19.5625 = 10011.1001_2$

2º) Normalizar el número binario obtenido, colocando el punto decimal a la derecha del bit más significativo: $10011.1001_2 = 1.00111001 \times 2^4$

3º) Escribir en base 10 el exponente en Exceso a $2^{n-1} - 1$, donde n=11: $4 + (2^{11-1} - 1 = 4 + (2^{10} - 1) = 4 + (1024 - 1) = 1027 = 10000000011_2$ Exponente en exceso a 1023

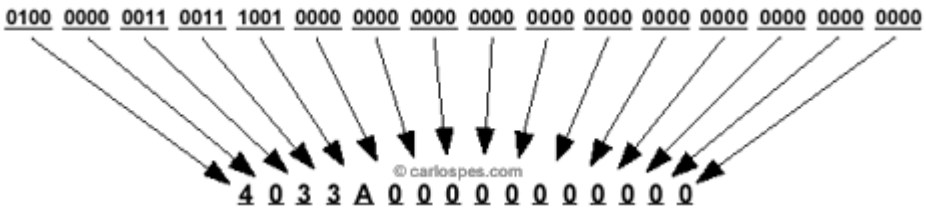
4º) Establecer la mantisa utilizando bit implícito. Para ello, se cogen los ocho bits que están a la derecha del punto (00111001) y el resto de la mantisa (52 bits) se rellena con ceros:

00111001000000000000000000000000000000000000000000000000

5º) Expresar el número en el estándar IEEE 754 con precisión doble. En este caso, hay que tener en cuenta que el bit de signo vale 0, ya que, el número es positivo:

63	62 . . . 52	51 . . . 0
0	10000000011	00111001000000000000000000000000000000000000000000000000
signo	Exponente	Mantisa

6º) Representar el número en hexadecimal:



De tal forma que,

$19.5625 = 10011.1001_2 = 1.00111001 \times 2^4 =$ **4033A00000000000H** (precisión doble)

Casos especiales en el Estándar IEEE 754

Tanto en precisión doble como en precisión simple, existen algunos casos especiales que dependen de los valores del signo, del exponente y de la mantisa:

Signo (s)	*Exponente (exp)*	*Mantisa (m)*	*Significado* © carlospes.com
Positivo (0)	Todo unos (11…1)	Todo ceros (00...0)	Más infinito (+∞)
Negativo (1)	Todo unos (11…1)	Todo ceros (00...0)	Menos infinito (-∞)
0 ó 1	Todo unos (11…1)	Distinta de todo ceros	No es un número (*Not a Number*, **NaN**)
0 ó 1	Todo ceros (00...0)	Todo ceros (00...0)	Representa al cero (0)
0 ó 1	Todo ceros (00...0)	Distinta de todo ceros	Número muy pequeño cercano al cero (0)

Figura 5: tabla de condiciones especiales en el estándar IEEE 754

3.9 Rangos de representación en el Estándar IEEE 754

Los dos últimos casos merecen especial atención, ya que, cuando todos los bits del exponente son ceros (00...0), esto quiere decir que no se está utilizando bit implícito. Si, además, la mantisa es todo ceros (00...0), el número representado es el cero (0), pero si la mantisa es distinta de todo ceros, el número que se está representando es muy pequeño, de forma que, el exponente valdrá -126 ó -1022, dependiendo de si el número está escrito en precisión simple o doble, respectivamente.

Los rangos de representación en el estándar IEEE 754 con precisión simple y doble, exponente en Exceso a 2^{n-1}-1 y mantisa en Signo Magnitud con bit implícito, son los siguientes:

$$((2^{-23} - 2) \cdot 2^{127})_{10} \leq x \leq ((2 - 2^{-23}) \cdot 2^{127})_{10}$$

Figura - Rango de representación en el estándar IEEE 754 con precisión simple.

$$((2^{-52} - 2) \cdot 2^{1023})_{10} \leq x \leq ((2 - 2^{-52}) \cdot 2^{1023})_{10}$$

Figura - Rango de representación en el estándar IEEE 754 con precisión doble.

Ambos rangos de representación son discontinuos, es decir, no se pueden representar todos los números reales que existen entre dos cualesquiera de ellos. Esto es debido a que entre dos números reales cualesquiera siempre existen infinitos números, sin embargo, sólo se dispone de un número determinado de bits para representar a los números reales. Por esta razón, en las computadoras digitales no se pueden representar a todos los números reales. Por ejemplo, con precisión simple, alrededor del número cero (0) existen infinitos números reales, mayores que -2^{-126} y menores que 2^{-126}, que no son representables. Gráficamente:

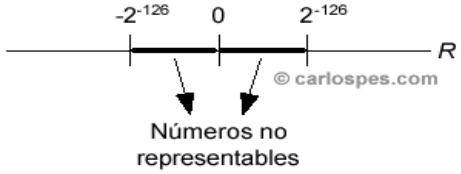


Figura - Números no representables en el estándar IEEE 754 con precisión simple.

Codificaciones con significado especial

- Infinito (E=255, m=0): representan cualquier valor de la región de overflow
- NaN (Not-a-Number) (E=255, m>0): se obtienen como resultado de operaciones inválidas
- Número desnormalizado (E=0, m>0): es un número sin normalizar cuyo bit implícito se supone que es 0. Al ser el exponente 0, permiten representar números en las regiones de underflow
- Cero (E=0,m=0): número no normalizado que representa al cero (en lugar de al 1)

3.10 Aritmética en punto flotante

La aritmética de punto flotante tiene un alto grado de elaboración, una de las refinaciones es el redondeo, esto cobra especial importancia en este sistema numérico que en sí es una aproximación del número real. Se emplean bits adicionales denominados de guarda y redondeo en los cálculos intermedios.

Las operaciones en punto flotante requieren de algoritmos especiales y de un hardware dedicado (tradicionalmente como un coprocesador, con registros independientes, que puede comunicar los resultados a la memoria)

3.10.1 Suma/Resta en punto flotante.

Para realizar esta operación puede proceder así:

1. Extraer signos, exponentes y magnitudes.
2. Tratar operandos especiales (por ejemplo, alguno de ellos a cero)
3. Desplazar la mantisa del número con exponente más pequeño a la derecha |E1-E2| bits
4. Fijar el exponente del resultado al máximo de los exponentes (podría suceder que se pierdan cifras de representación)
5. Si la operación es suma y los signos son iguales, o si la operación es resta y los signos son diferentes, sumar las mantisas. En otro caso restarlas. El resultado podría no quedar normalizado.
6. Detectar overflow de la mantisa
7. Normalizar la mantisa, desplazándola a la derecha o a la izquierda hasta que el dígito más significativo esté delante del punto.
8. Redondear el resultado y renormalizar la mantisa si es necesario. En este caso podría suceder que al redondear, el número se des-normalice (bit implícito se asume que es 0), en ese caso debe repetirse el paso de normalización; de lo contrario, permitiría representar números en las regiones de underflow (E>0 y m> 0).
9. Corregir el exponente en función de los desplazamientos realizados sobre la mantisa.
10. Detectar overflow o underflow del exponente

Ejemplo 3.5: si a= $0.4523x10^4$ y b = $0.0002115x10^0$, calcule a + b y a – b, almacenando solamente 4 dígitos en la mantisa y 2 en el exponente.

Solución:
a + b = $0.4523x10^4 + 0.0002115x10^0$
= $0.4523\ x10^4 + 0.00000002115\ x10^4 = 0.4523\ x10^4$ positivo=$\mathbf{+4.523x10^3}$.

Ahora, a - b = $0.4523x10^4$ positivo=$\mathbf{+4.523x10^3}$

Estos cálculos muestran claramente la pérdida de dígitos significativos en las operaciones de suma y resta en punto flotante. Observamos que en el caso de la resta no se ha producido una diferencia cancelativa, ya que el resultado tiene una exactitud igual a la precisión (4 dígitos) de la aritmética usada.

Lo usual es dedicar hardware especial para implementar el algoritmo anterior. En caso de no tenerlo, la solución es implementar una biblioteca, en base a las operaciones de la unidad aritmética entera, que realice el algoritmo anterior. En este caso, obviamente la realización de una simple suma de flotantes requiere la ejecución de varias instrucciones, alargando el tiempo de ejecución de los programas que empleen números reales.

Ejemplo 3.6: sea a= $6.144x10^2$ y b= $9.975x10^4$. Calcule a + b y a - b, almacenando solamente 4 dígitos en la mantisa y hasta 2 dígitos en el exponente.
Solución:

Hagamos primero el alineamiento (números con igual potencia):
a= $6.144x10^2$=$0.06144x10^4$ y b= $9.97500x10^4$
Sumemos los coeficientes y deja la misma potencia:
a + b=$0.06144x10^4$ + $9.97500x10^4$=$10.03644x10^4$

Desarrollemos ahora la normalización:
$10.03644x10^4$=$1.003644x10^5$ positivo

Por ultimo redondeemos para 4 cifras significativas: +**$1.004x10^5$**

Ahora, a-b=**$-9.914x10^4$**

3.10.2 Multiplicación en punto flotante

Esta operación puede describir los siguientes pasos:

1. Se suman los exponentes. En esto debe considerarse que los exponentes estén con exceso 2^{n-1}-1; por lo tanto debe restarse el exceso para obtener el exponente correcto. Veamos: si por ejemplo en decimal se tiene 2 exponentes (5 y -7) con exceso a 2^{8-1}-1=127, proceda así:
   - con el exponente 5 haga 5 + 127= 132 y con el otro exponente (-7) póngalo -7 + 127= 120;
   - la suma de los exponentes resulta 132+120 = 252, que no es el exponente correcto, porque el exceso se estaría sumando 2 veces.
   - Debe realizarse: 252-127 = 125 que es el exponente para la representación (equivale al exponente 5+ (-7)=-2).
2. La mantisa del resultado es igual al producto de las mantisas. Este producto es sin signo. Dado que los dos operandos están comprendidos entre 1 y 2 el resultado r será: $1 \leq r < 4$ Este paso puede requerir una normalización, mediante desplazamiento a la derecha y ajuste del exponente resultado.
3. Si la mantisa resultado es del mismo tamaño que la de los operandos habrá que redondear. El redondeo puede implicar la necesidad de normalización posterior.
4. El exponente del resultado es igual a la suma de los exponentes de los operandos. Considerando que usamos una representación sesgada del exponente, al hacer esta suma estamos sumando dos veces el sesgo, y por tanto habrá que restar este sesgo una vez para obtener el resultado correcto.
5. Finalmente debe colocarse el signo.

Ejemplo 3.7: Multiplique $1.110x10^{10}$ por $9.200x10^{-5}$, almacenando solamente 4 dígitos en la mantisa y hasta 2 dígitos en el exponente. Se sugiere que proceda así:

P1: Calculemos el exponente del producto sumando los exponentes de los factores. Efectivamente, el nuevo exponente es e= 10 + (-5) = 5.

Ahora probemos con los exponentes desplazados para asegurarnos que obtenemos el mismo resultado: 10 + 127 = 137, y –5 + 127 = 122. Entonces, el exponente sería E= 137 + 122 = 259, lo cual es un error, debido a que el desplazamiento se realizaría 2 veces (una vez por cada exponente); por consiguiente que el exponente desplazado es: E= (137 + 122) - 127 = 259 – 127 = **132** exponente para la representación.

P2: Por último se deben multiplicar las mantisas: 1.110x9.200= 10212000.

Observe que hay tres dígitos a la derecha de cada factor, de manera que deberá haber 6 dígitos a la derecha del punto decimal del producto: 10.212000.

Suponiendo que sólo se pueden mantener 3 dígitos a la derecha del punto decimal, el producto resultante sería: $10.212x10^{5}$

P3: Este resultado no está normalizado, de manera que tenemos que normalizarlo para obtener: $1.0212\ x10^{6}$

Después de la multiplicación, el producto puede ser desplazado a la derecha, sumando 1 al exponente, o bien desplazado a la izquierda restándole 1 al exponente. Por lo tanto, en este paso debe verificarse la presencia de un sobre flujo (overflow) o bien de un bajo flujo (underflow).

P4: Puesto que asumimos que la mantisa era de 4 dígitos, se debe redondear al producto, de manera que: $1.0212x10^{6}$ es redondeado a $1.021x10^{6}$

P5: El signo del producto depende de los signos de los factores, si son iguales el producto es positivo y si son diferentes, el producto será negativo.

Efectivamente, en este caso el resultado de la mantisa es: **$+1.021x10^{6}$**

3.10.3 División en punto flotante

El algoritmo de división, es bastante más complejo, y produce una operación más lenta. Para acelerarla en algunos diseños se genera el recíproco; en otros se intenta obtener varios bits de la división en un solo paso. Un pequeño error en la implementación de este algoritmo causó una pérdida millonaria a la empresa Intel, en el diseño del procesador PENTIUM.

La norma IEEE 754, tiene símbolos especiales para representar más y menos infinito, y lo que denomina NaN (not a number) que ocurre en la división de cero por cero, o en una resta de infinito menos infinito. También se tratan números no normalizados que permiten extender el rango de representación.

AUTOEVALUACION 3

1. Obtenga el número decimal con **4** cifras significativas, asociado al número dado en IEEE 754: $11000000010100100111000100000000_2$
   A) -5.153 B) -3.288 C) -5.152 D) -3.289

2) Obtenga el número decimal con 5 dígitos de precisión en la mantisa y 2 en el exponente, asociado al número dado en IEEE 754: $11000000101001001110001000000000_2$
   A) 5.15259 B) 5.15258 C) -5.15258 D) -5.15259

3) Exprese en el formato IEEE 754 el número 57.23 en precisión simple.
   A) 0 10000100 11001001110000000000000
   B) 0 10000101 11001001110000000000000
   C) 1 10000100 11001001110000000000000
   D) 1 10000101 11001001110000000000000

TALLER 3

1. Normalice los siguientes números y expréselos con 4 cifras significativas:
   $32452.25661x10^{-3}$= _____
   $932032.00001233x10^{4}$= _____
   $0.000001234434x10^{7}$= _____
   $0.000001234434x10^{-5}$= _____
2. Represente a –0.75 en punto flotante de precisión simple expresado en formato IEEE 754.
   R/

31	30 . . . 23	22 . . . 0
1	0 1 1 1 1 1 1 0	10000000000000000000000
signo	Exponente	Mantisa

3. A partir de los datos en la siguiente tabla, complétela determinando el valor decimal, según las codificaciones con significado especial de la IEEE 754:

Signo (S)	Exponente (E)	Mantisa (m)	Valor decimal
1	255	0	
1	247	10000001111101010101111	
0	137	10000001111101010101111	
0	2047	010101011…11100111001111	
0	255	0	
0	1250	10000001111…101010101111	
1	2047	10000001111…101010101111	
1	2047	0	
1	255	0	

4. Sumar los números 0.5 y –0.4375 en binario, suponiendo 4 bits de precisión. **R/1.000$_2$ x 2^{-4}**
5. Sume los números: 88.7325x10^{-1} y 5.72382 x10^2 dado que se pueden almacenar 5 dígitos en la mantisa y hasta 2 dígitos en el exponente **R/5.8126x10^2**
6. Multiplicar 0.5 por - 0.4375 en binario, suponiendo 4 bits de precisión. **R/- 1.110$_2$ x 2^{-3}**
7. Represente los números resultantes de los numerales 4, 5 y 6 en punto flotante de precisión simple expresados en formato IEEE 754.
8. Resuelva los siguientes problemas

   Problema 1: exprese en decimal el número 1BACAFEA$_H$ del estándar IEEE 754 con precisión simple.

   Problema 2: exprese en decimal el número 80D3EE45$_H$ del estándar IEEE 754 con precisión simple.

   Problema 3: sume y multiplique los números: -38.7325x10^{-3} y 5.72325 x10^2 dado que se pueden almacenar 6 dígitos en la mantisa y hasta 2 dígitos en el exponente.