PROBLEMAS RESUELTOS: Codificación

1. Calcule la codificación del valor 267 en los sistemas posicionales de base 2, 3, 4, 8 y 16.

SOLUCIÓN:

Aplicamos el algoritmo de codificación de enteros para la base 2:

$267 \div 2 \Rightarrow Q = 133 \quad R = 1 \Rightarrow \text{'1'}_2$
$133 \div 2 \Rightarrow Q = 66 \quad R = 1 \Rightarrow \text{'1'}_2$
$66 \div 2 \Rightarrow Q = 33 \quad R = 0 \Rightarrow \text{'0'}_2$
$33 \div 2 \Rightarrow Q = 16 \quad R = 1 \Rightarrow \text{'1'}_2$
$16 \div 2 \Rightarrow Q = 8 \quad R = 0 \Rightarrow \text{'0'}_2$
$8 \div 2 \Rightarrow Q = 4 \quad R = 0 \Rightarrow \text{'0'}_2$
$4 \div 2 \Rightarrow Q = 2 \quad R = 0 \Rightarrow \text{'0'}_2$
$2 \div 2 \Rightarrow Q = 1 \quad R = 0 \Rightarrow \text{'0'}_2$
$1 \div 2 \Rightarrow Q = 0 \quad R = 1 \Rightarrow \text{'1'}_2$

Ahora para la base 3:

$267 \div 3 \Rightarrow Q = 89 \quad R = 0 \Rightarrow \text{'0'}_3$
$89 \div 3 \Rightarrow Q = 29 \quad R = 2 \Rightarrow \text{'2'}_3$
$29 \div 3 \Rightarrow Q = 9 \quad R = 2 \Rightarrow \text{'2'}_3$
$9 \div 3 \Rightarrow Q = 3 \quad R = 0 \Rightarrow \text{'0'}_3$
$3 \div 3 \Rightarrow Q = 1 \quad R = 0 \Rightarrow \text{'0'}_3$
$1 \div 3 \Rightarrow Q = 0 \quad R = 1 \Rightarrow \text{'1'}_3$

Podríamos seguir aplicando el algoritmo para el resto de bases pero como son potencias de 2 es más sencillo codificarlas agrupando dígitos de la codificación binaria. Para la base 4 tendremos que agrupar de dos en dos comenzando por el LSB (*Least Significant Bit*):

$\text{'11'}_2 \Rightarrow \text{'3'}_4$
$\text{'10'}_2 \Rightarrow \text{'2'}_4$
$\text{'00'}_2 \Rightarrow \text{'0'}_4$
$\text{'00'}_2 \Rightarrow \text{'0'}_4$
$\text{'01'}_2 \Rightarrow \text{'1'}_4$

Agrupando la codificación binaria de tres en tres dígitos obtenemos la codificación octal:

$\text{'011'}_2 \Rightarrow \text{'3'}_8$
$\text{'001'}_2 \Rightarrow \text{'1'}_8$
$\text{'100'}_2 \Rightarrow \text{'4'}_8$

Finalmente, agrupando la codificación binaria de cuatro en cuatro dígitos obtenemos la codificación hexadecimal:

$\text{'1011'}_2 \Rightarrow \text{'B'}_{16}$
$\text{'0000'}_2 \Rightarrow \text{'0'}_{16}$
$\text{'0001'}_2 \Rightarrow \text{'1'}_{16}$

La tabla siguiente resumen las codificaciones obtenidas para cada base.

	base 2	base 3	base 4	base 8	base 16
267	100001011_2	100220_3	10023_4	413_8	$10B_{16}$

Los dígitos de la codificación en base 16 también se pueden obtener agrupando de dos en dos los de la codificación en base 4.

$'23'_4 = 2 \cdot 4^1 + 3 \cdot 4^0 = 11 \Rightarrow 'B'_{16}$
$'00'_4 = 0 \cdot 4^1 + 0 \cdot 4^0 = 0 \Rightarrow '0'_{16}$
$'01'_4 = 0 \cdot 4^1 + 1 \cdot 4^0 = 1 \Rightarrow '1'_{16}$

Nótese también que cuanto mayor es la base menor espacio requiere la codificación.

2. Calcule la codificación del valor 0,189 en los sistemas posicionales de base 2, 3 y 4. Obtenga 4 decimales en cada caso.

SOLUCIÓN:

Antes de empezar a codificar vamos a conocer si el número decimal es exacto o no en cada sistema. Para ello encontramos su fracción irreducible y descomponemos el denominador en sus factores primos:

$0,189 = \dfrac{189}{1000} \rightarrow 1000 = 2^3 \times 5^3 \rightarrow$ el producto de los factores primos 2×5 del denominador no se encuentra en las bases 2, 3, ni 4 por lo que la codificación no es exacta en ninguna de estas bases.

Aplicamos el algoritmo de codificación de números fraccionarios a la base 2:

$0,189 \times 2 = 0,378 \Rightarrow '0'_2$
$0,378 \times 2 = 0,756 \Rightarrow '0'_2$
$0,756 \times 2 = 1,512 \Rightarrow '1'_2$
$0,512 \times 2 = 1,024 \Rightarrow '1'_2$

Otro método: Puesto que nos indican el número máximo de decimales, podemos resolver la codificación multiplicando el valor a codificar por la base elevada al número de decimales y codificar la parte entera del número resultante. En este caso, para base 2 será $[0,189 \cdot 2^4] = [3,024] = 3 = 0011_2 \Rightarrow 0,0011_2$. El hecho de que hayamos obviado decimales ya nos indica que la codificación no es exacta.

En base 3 tendremos:

$0,189 \times 3 = 0,567 \Rightarrow '0'_3$
$0,567 \times 3 = 1,701 \Rightarrow '1'_3$
$0,701 \times 3 = 2,103 \Rightarrow '2'_3$
$0,103 \times 3 = 0,306 \Rightarrow '0'_3$

Utilizando el **otro método** para 4 decimales: $[0,189 \cdot 3^4] = [15,309] = 15 = 0120_3 \Rightarrow 0,0120_3$. Aquí también comprobamos que la codificación no es exacta ya que el producto tiene parte decimal que no representamos.

En base 4 podríamos asociar de dos en dos los dígitos de la representación binaria pero nos faltarían 2 decimales por lo que aplicamos el algoritmo de codificación desde el comienzo:

$0,189 \times 4 = 0,756 \Rightarrow '0'_4$
$0,756 \times 4 = 3,024 \Rightarrow '3'_4$
$0,024 \times 4 = 0,096 \Rightarrow '0'_4$
$0,096 \times 4 = 0,384 \Rightarrow '0'_4$

Utilizando el **otro método** para 4 decimales: $[0,189 \cdot 4^4] = [48,384] = 48 = 0300_4 \Rightarrow 0,0300_4$. Aquí también comprobamos que la codificación no es exacta ya que el producto tiene parte decimal que no representamos.

3. Determine la codificación del valor 673 en binario limitado a 4, 8 y 12 dígitos indicando el desbordamiento.

SOLUCIÓN:

En un sistema posicional limitado a n dígitos hemos de indicar si hay o no desbordamiento ya que, excedido el tamaño disponible, lo codificado es la clase de congruencia y no el valor.

Para ello tenemos dos caminos: indicaremos el desbordamiento si el tamaño requerido para la codificación es mayor que el disponible o bien si una vez realizado el algoritmo de codificación durante n iteraciones el cociente final es distinto de cero.
Veamos ambos caminos aquí. Comenzaremos determinando el tamaño que necesita el valor 673 para ser representado en binario:

$$\mathit{TAM}(V, b) = \lfloor \log_b V \rfloor + 1$$

Es decir;

$$\mathit{TAM}(673, 2) = \lfloor \log_2 673 \rfloor + 1 = 10 \text{ bits}$$

Esto significa que en sistemas limitados a 4 y 8 bits, $\mathsf{des} = \text{'1'}$ ya que no podrá ser codificado hasta que no haya al menos 10 bits. Ahora calculamos su codificación en 12 iteraciones del algoritmo:

$673 \div 2 \Rightarrow Q = 336 \quad R = 1 \Rightarrow \text{'1'}_2$
$336 \div 2 \Rightarrow Q = 168 \quad R = 0 \Rightarrow \text{'0'}_2$
$168 \div 2 \Rightarrow Q = 84 \quad R = 0 \Rightarrow \text{'0'}_2$
$84 \div 2 \Rightarrow \mathbf{Q = 42} \quad R = 0 \Rightarrow \text{'0'}_2 \quad \Rightarrow \mathsf{d}=\text{'1'} \quad$ para 4 bits
$42 \div 2 \Rightarrow Q = 21 \quad R = 0 \Rightarrow \text{'0'}_2$
$21 \div 2 \Rightarrow Q = 10 \quad R = 1 \Rightarrow \text{'1'}_2$
$10 \div 2 \Rightarrow Q = 5 \quad R = 0 \Rightarrow \text{'0'}_2$
$5 \div 2 \Rightarrow \mathbf{Q = 2} \quad R = 1 \Rightarrow \text{'1'}_2 \quad \Rightarrow \mathsf{d}=\text{'1'} \quad$ para 8 bits
$2 \div 2 \Rightarrow Q = 1 \quad R = 0 \Rightarrow \text{'0'}_2$
$1 \div 2 \Rightarrow \mathbf{Q = 0} \quad R = 1 \Rightarrow \text{'1'}_2 \quad \Rightarrow \mathsf{d}=\text{'0'} \quad$ a partir de 10 bits
$0 \div 2 \Rightarrow Q = 0 \quad R = 0 \Rightarrow \text{'0'}_2$
$0 \div 2 \Rightarrow Q = 0 \quad R = 0 \Rightarrow \text{'0'}_2$

Vemos que después de 4 iteraciones el cociente no se anula por lo que indicamos el desbordamiento ($\mathsf{des} = \text{'1'}$). Lo mismo sucede a las 8 y nuevamente lo señalamos con $\mathsf{des} = \text{'1'}$. A la décima iteración el codiente se anula y a partir de ese momento ya no hay desbordamiento ($\mathsf{des} = \text{'0'}$). En resumen tenemos:

	4	d	8	d	12	d
676	0001	1	1010 0001	1	0010 1010 0001	0

4. Codifique el valor entero -21 en signo-magnitud, complemento a dos, complemento a uno y exceso a 128 siendo el tamaño de representación $n = 8$ bits. Indique el desbordamiento.

SOLUCIÓN:

Aplicaremos la definición de cada sistema de representación.

En signo magnitud, el operador $\mathit{SM}(Z, n)$ devuelve:

$$\mathit{SM}(Z,n) = \begin{cases} \left.\begin{array}{l} \mathit{BIN}\,(|Z|, n-1) \\ \text{si } Z \geq 0 \quad \mathsf{s} = \text{'0'} \\ \text{si } Z < 0 \quad \mathsf{s} = \text{'1'} \end{array}\right\} = \mathsf{Z}_{SM,n} \\ \\ \mathsf{des} = \text{'0' si } Z \in \mathsf{rango}(\mathcal{SM}_n) \\ \mathsf{des} = \text{'1' si } Z \notin \mathsf{rango}(\mathcal{SM}_n) \end{cases}$$

$$\mathit{SM}(-21, 8) = \mathit{BIN}\,(|-21|, 8-1) = \mathit{BIN}(21, 7) \quad \mathsf{s} = \text{'1'}$$

$21 \div 2 \Rightarrow Q = 10 \quad R = 1 \Rightarrow \text{'1'}_2$
$10 \div 2 \Rightarrow Q = 5 \quad R = 0 \Rightarrow \text{'0'}_2$

$5 \div 2 \Rightarrow Q = \ 2 \quad R = 1 \Rightarrow \text{'1'}_2$
$2 \div 2 \Rightarrow Q = \ 1 \quad R = 0 \Rightarrow \text{'0'}_2$
$1 \div 2 \Rightarrow Q = \ 0 \quad R = 1 \Rightarrow \text{'1'}_2$
$0 \div 2 \Rightarrow Q = \ 0 \quad R = 0 \Rightarrow \text{'0'}_2$
$0 \div 2 \Rightarrow \mathbf{Q = 0} \quad R = 0 \Rightarrow \text{'0'}_2 \quad \Rightarrow$ d='0'

De donde obtenemos:

$SM(-21, 8) = 1\ 001\,0101_{SM,8}$

En complemento a dos, el operador $C2(Z, n)$ devuelve:

$$C2(Z,n) = \left\{ \begin{array}{l} \left.\begin{array}{ll} \text{si } Z \geq 0 & BIN(Z,n) \\ \text{si } Z < 0 & BIN\left(2^n - |Z|, n\right) \end{array}\right\} = \mathsf{Z}_{C2,n} \\ \\ \text{des} = \text{'0' si } Z \in \text{rango}(\mathcal{C}2_n) \\ \text{des} = \text{'1' si } Z \notin \text{rango}(\mathcal{C}2_n) \end{array} \right.$$

$$C2(-21, 8) = BIN\left(2^8 - |-21|, 8\right) = BIN(256 - 21, 8) = BIN(235, 8)$$

$235 \div 2 \Rightarrow Q = 117 \quad R = 1 \Rightarrow \text{'1'}_2$
$117 \div 2 \Rightarrow Q = \ 58 \quad R = 1 \Rightarrow \text{'1'}_2$
$\ 58 \div 2 \Rightarrow Q = \ 29 \quad R = 0 \Rightarrow \text{'0'}_2$
$\ 29 \div 2 \Rightarrow Q = \ 14 \quad R = 1 \Rightarrow \text{'1'}_2$
$\ 14 \div 2 \Rightarrow Q = \ \ 7 \quad R = 0 \Rightarrow \text{'0'}_2$
$\ \ 7 \div 2 \Rightarrow Q = \ \ 3 \quad R = 1 \Rightarrow \text{'1'}_2$
$\ \ 3 \div 2 \Rightarrow Q = \ \ 1 \quad R = 1 \Rightarrow \text{'1'}_2$
$\ \ 1 \div 2 \Rightarrow \mathbf{Q = \ \ 0} \quad R = 1 \Rightarrow \text{'1'}_2 \quad \Rightarrow$ d='0'

De donde obtenemos:

$C2(-21, 8) = 1110\,1011_{C2,8}$

En complemento a uno, el operador $C1(Z, n)$ devuelve:

$$C1(Z,n) = \left\{ \begin{array}{l} \left.\begin{array}{ll} \text{si } Z \geq 0 & BIN(Z,n) \\ \text{si } Z < 0 & BIN\left(2^n - 1 - |Z|, n\right) \end{array}\right\} = \mathsf{Z}_{C1,n} \\ \\ \text{des} = \text{'0' si } Z \in \text{rango}(\mathcal{C}1_n) \\ \text{des} = \text{'1' si } Z \notin \text{rango}(\mathcal{C}1_n) \end{array} \right.$$

$$C1(-21, 8) = BIN\left(2^8 - 1 - |-21|, 8\right) = BIN(256 - 1 - 21, 8) = BIN(234, 8)$$

$234 \div 2 \Rightarrow Q = 117 \quad R = 0 \Rightarrow \text{'0'}_2$
$117 \div 2 \Rightarrow Q = \ 58 \quad R = 1 \Rightarrow \text{'1'}_2$
$\ 58 \div 2 \Rightarrow Q = \ 29 \quad R = 0 \Rightarrow \text{'0'}_2$
$\ 29 \div 2 \Rightarrow Q = \ 14 \quad R = 1 \Rightarrow \text{'1'}_2$
$\ 14 \div 2 \Rightarrow Q = \ \ 7 \quad R = 0 \Rightarrow \text{'0'}_2$
$\ \ 7 \div 2 \Rightarrow Q = \ \ 3 \quad R = 1 \Rightarrow \text{'1'}_2$
$\ \ 3 \div 2 \Rightarrow Q = \ \ 1 \quad R = 1 \Rightarrow \text{'1'}_2$
$\ \ 1 \div 2 \Rightarrow \mathbf{Q = \ \ 0} \quad R = 1 \Rightarrow \text{'1'}_2 \quad \Rightarrow$ d='0'

De donde obtenemos:

$C1(-21, 8) = 1110\,1010_{C1,8}$

En exceso a 2^{n-1}, el operador $EX2^{n\text{-}1}(Z,\, n)$ devuelve:

$$EX2^{n\text{-}1}(Z,n) = \begin{cases} BIN(Z+2^{n-1},n) = \mathsf{Z}_{EX\,2^{n\text{-}1},n} \\ \\ \mathsf{des} = \text{'0' si } Z \in \mathsf{rango}(\mathcal{EX}2^{n\text{-}1}{}_n) \\ \mathsf{des} = \text{'1' si } Z \notin \mathsf{rango}(\mathcal{EX}\text{-}2^{n\text{-}1}{}_n) \end{cases}$$

$$EX2^{n\text{-}1}(-21,8) = BIN(-21+2^{8-1},8) = BIN(-21+128,8) = BIN(107,8)$$

$$\begin{array}{rrl} 107 \div 2 \Rightarrow Q = 53 & R = 1 \Rightarrow \text{'1'}_2 & \\ 53 \div 2 \Rightarrow Q = 26 & R = 1 \Rightarrow \text{'1'}_2 & \\ 26 \div 2 \Rightarrow Q = 13 & R = 0 \Rightarrow \text{'0'}_2 & \\ 13 \div 2 \Rightarrow Q = 6 & R = 1 \Rightarrow \text{'1'}_2 & \\ 6 \div 2 \Rightarrow Q = 3 & R = 0 \Rightarrow \text{'0'}_2 & \\ 3 \div 2 \Rightarrow Q = 1 & R = 1 \Rightarrow \text{'1'}_2 & \\ 1 \div 2 \Rightarrow Q = 0 & R = 1 \Rightarrow \text{'1'}_2 & \\ 0 \div 2 \Rightarrow \mathbf{Q = 0} & R = 0 \Rightarrow \text{'0'}_2 & \Rightarrow \mathsf{d}=\text{'0'} \end{array}$$

De donde obtenemos:

$EX2^{n\text{-}1}(-21,8) = 0110\,1011_{EX\,2^{n\text{-}1},8}$

La tabla siguiente recopila las diferentes codificaciones.

	$\mathcal{SM}_n$	d	$\mathcal{C}2_n$	d	$\mathcal{C}1_n$	d	$\mathcal{EX}2^{n\text{-}1}{}_n$	d
-21	1 001 0101	0	1110 1011	0	1110 1010	0	0110 1011	0

5. Tenemos la siguiente representación: $111000_{2,6}$. Interprete dicha representación en binario puro, signo-magnitud, complemento a dos, complemento a uno y exceso a 2^{n-1} calculando el valor codificado en cada caso.

SOLUCIÓN:

Aplicando el teorema fundamental de la numeración obtenemos el valor de la representación en binario puro:

$$VAL\text{-}BIN(111000) = \sum_{i=0}^{n-1} d_i \cdot 2^i = 2^5 + 2^4 + 2^3 = 56$$

No obstante, dado que la representación tiene 3 ceros a la derecha podemos tomar sencillamente la representación 111_2 sobre 3 bits y multiplicarla por 2^3, es decir, $7 \times 8 = 56$.

Para la representación en signo-magnitud asumimos que el bit de signo está en el MSB (*Most Significant Byte*). El valor representado es negativo y corresponde a la magnitud $3 \times 8 = 24$, es decir:

$$VAL\text{-}SM(111000) = (-1)^s \sum_{i=0}^{n-2} d_i \cdot 2^i = -(2^4 + 2^3) = -24$$

En complemento a dos podemos aplicar el operador *VAL-C2* pero conociendo el valor binario y el límite de representación sobre 6 bits es inmediato obtener $56 - 64 = -8$, es decir:

$$VAL\text{-}C2(111000) = \sum_{i=0}^{n-2} d_i \cdot 2^i - d_{n-1} \cdot 2^{n-1} = 2^4 + 2^3 - 2^5 = -8$$

En complemento a uno podemos aplicar el operador *VAL-C1* pero conociendo el valor binario y el límite de representación sobre 6 bits es inmediato obtener $56 - 64 + 1 = -7$, es decir:

$$VAL\text{-}C1(111000) = \sum_{i=0}^{n-2} d_i \cdot 2^i - d_{n-1} \cdot 2^{n-1} + d_{n-1} = 2^4 + 2^3 - 2^5 + 1 = -7$$

Finalmente, si interpretamos la representación en exceso a 2^{n-1} podemos obtenerlo complementando el MSB y calculándolo en complemento a dos, es decir, $3 \times 8 = 24$:

$$\mathit{VAL\text{-}EX2^{n\text{-}1}}(111000) = \sum_{i=0}^{n-2} d_i \cdot 2^i - \overline{d}_{n-1} \cdot 2^{n-1} = 2^4 + 2^3 = 24$$

6. Represente el valor $-57{,}864$ en coma fija sobre 10 bits en complemento a dos sabiendo que la coma está en el peso 2^3. Si es el caso, indique el error cometido al codificar.

SOLUCIÓN:

Definición para la representación de reales en complemento a dos:

$$\mathit{C2}(R,n,q) = \left\{ \begin{array}{ll} \text{si } R \geq 0 & \mathit{BIN}\left(\dfrac{Z}{2^q}, n, q\right) = \mathit{BIN}\left(R, n, q\right) \\ \text{si } R < 0 & \mathit{BIN}\left(\dfrac{2^n}{2^q} - \dfrac{|Z|}{2^q}, n, q\right) = \mathit{BIN}\left(2^{n-q} - |R|, n, q\right) \end{array} \right\} = \mathsf{R}_{C2,n,q}$$

Aplicamos la definición para los negativos:

si $R < 0 \Rightarrow \mathit{C2}(R,n,q) = \mathit{BIN}(2^{n-q} - |R|.n, q)$

es decir,

$\mathit{C2}(-57{,}864, 10, 3) = \mathit{BIN}(2^7 - 57{,}864, 10, 3) = \mathit{BIN}(70{,}136, 10, 3)$

La codificación la podemos obtener multiplicando el valor complementado por el peso de la coma: $70{,}136 \times 2^3 = [561{,}088] = 561 = 10\,0011\,0001_2$, es decir,

$\mathit{C2}(-57{,}864, 10, 3) = \mathit{BIN}(70{,}136, 10, 3) = 100\,0110{,}001_{C2,10,3}$

Otro método: Alternativamente podemos codificar la parte entera por un lado y la fraccionaria por otro.

Sabemos que la representación no es exacta porque al multiplicar por el peso de la coma nos ha salido un número con parte fraccionaria. Veamos cuál es el valor efectivamente codificado:

$$\frac{\mathit{VAL\text{-}C2}(10\,0011\,0001)}{2^3} = \frac{-463}{8} = -57{,}875$$

Se produce, entonces, un error por defecto de $-0{,}011$ que está por debajo de la precisión del sistema de representación, es decir, $|-0{,}011| < 2^{-3} = 0{,}125$.

7. Represente el valor 3,91905 en coma flotante con 10 bits para la mantisa normalizada en signo-magnitud y 6 bits para el exponente en exceso a 2^{n-1}. Recuerde que una mantisa es una representación del tipo 0,..., es decir, sin parte entera.

SOLUCIÓN:

Como la mantisa está en signo-magnitud normalizada usaremos la siguiente expresión para calcular el exponente:

$$E = \lfloor \log_2 |V| \rfloor + 1 = \lfloor \log_2 |3{,}91905| \rfloor + 1 = 2$$

cuya codificación es: $10\,0010$.

La mantisa será:

$$\frac{V}{2^E} = \frac{3{,}91905}{2^2} = 0{,}9797625$$

cuya codificación es: 0 1111 1010 1.

El valor realmente codificado es 3,9140625.

8. Represente el valor 12,4071 en coma flotante con 10 bits para la mantisa normalizada en signo-magnitud y 6 bits para el exponente en exceso a 2^{n-1}. Recuerde que una mantisa es una representación del tipo 0,..., es decir, sin parte entera.

SOLUCIÓN:

Como la mantisa está en signo-magnitud normalizada usaremos la siguiente expresión para calcular el exponente:

$$E = \lfloor \log_2 |V| \rfloor + 1 = \lfloor \log_2 |12{,}4071| \rfloor + 1 = 4$$

cuya codificación es: 10 0100.

La mantisa será:

$$\frac{V}{2^E} = \frac{12{,}4071}{2^4} = 0{,}77544375$$

cuya codificación es: 0 1100 0110 1.

El valor realmente codificado es 12,40625.

9. Codifíque el número $-2{,}759 \times 10^{12}$ en simple precisión IEEE 754.

SOLUCIÓN:

Sabemos que en simple precisión IEEE 754 se codifica el significante sobre 24 bits en signo-magnitud normalizado y el exponente en exceso a $2^{n-1} - 1$ sobre 8 bits. Recuerde que un significante es una representación del tipo 1,..., es decir, tiene como parte entera la unidad.

Determinamos primero el exponente:

$$\lfloor \log_2 V \rfloor = \lfloor \log_2 | -2,759 \times 10^{12} | \rfloor = 41$$

cuya codificación será: 1010 1000.

Ahora determinamos el significante:

$$\frac{V}{2^E} = \frac{-2,759 \times 10^{12}}{2^{41}} = -1,254647941$$

cuya codificación será: 1 0100 0001 0011 0000 1001 110.

NOTA: la codificación tiene un error de $-64{,}000$.