Problemas sobre la representación de punto flotante (Ejemplos resueltos)

March 15, 2023

Problema 1. Atención: el siguiente problema se basa en una simplificación de las reglas del IEEE754. Al no usarse las reglas reales, los resultados son solo correctos con las simplificaciones expuestas.

Cierta representación de números en punto flotante utiliza normas similares a las usadas por el IEEE754 para los números normalizados, salvo que utiliza un exponente de 3 bits y una mantisa de 4 bits. No se consideran números denormalizados, ni símbolos especiales salvo el 0. Obtén en esta representación el mayor número positivo, el menor número negativo, el menor número positivo distinto de 0 u el mayor número negativo distinto de 0.

Proof. Solución

- El mayor positivo es: 01111111.

  El signo es positivo, y el bit oculto junto a a la mantisa es 1.1111.

  Recordemos además que el exponente se codifica en exceso Z. Esto es, si el binario normalizado es $+1.1111 \times 2^x$, entonces el exponente codificado es $y = x + Z$, donde el exceso $Z = 2^{n-1} - 1$. Como en este caso $n = 3$, resulta que $Z = 3$. Como $y = 7 = x + 3$, resulta que $x = 4$.

  Por tanto, el binario normalizado es: $+1.1111 \times 2^4 = (11111)_2 = (31)_{10}$.

- El menor positivo diferente de cero es: 00000001. No podemos poner todos los bits a 0 porque en ese cso se correspondería con el símbolo especial 0.

  El signo es positivo, y el bit oculto junto a la mantisa es 1.0001.

  Recordemos además que el exponente se codifica en exceso Z. Esto es, si el binario normalizado es $+1.0001 \times 2^x$, entonces el exponente codificado es $y = x + Z$, donde el exceso $Z = 2^{n-1} - 1$. Como en este caso $n = 3$, resulta que $Z = 3$. Entonces $0 = x + 3$, lo que implica que $x = -3$.

  Por tanto, el binario normalizado es: $+1.0001 \times 2^{-3} = (0.0010001)_2 = 2^{-3} + 2^{-7}$.

- Los valores correspondientes a los negativos son los opuestos. El mayor valor negativo sería: $-2^{-3} - 2^{-7}$. y el menor valor negativo es -31.

□

Problema 2. Atención: el siguiente problema se basa en una simplificación de las reglas del IEEE754. Al no usarse las reglas reales, los resultados son solo correctos con las simplificaciones expuestas.

Vuelve a realizar los cálculos del problema anterior considerando además los símbolos especiales NaN, infinito positivo e infinito negativo.

Proof. Solución

- El mayor positivo es: 01101111. En este caso el exponente no puede ser 111 porque entonces sería un símbolo especial NaN.

  El signo es positivo, y el bit oculto junto a a la mantisa es 1.1111.

  Recordemos además que el exponente se codifica en exceso Z. Esto es, si el binario normalizado es $+1.1111 \times 2^x$, entonces el exponente codificado es $y = x + Z$, donde el exceso $Z = 2^{n-1} - 1$. Como en este caso $n = 3$, resulta que $Z = 3$. Como $y = 6 = x + 3$, resulta que $x = 3$.

  Por tanto, el binario normalizado es: $+1.1111 \times 2^3 = (1111.1)_2 = (15.5)_{10}$.

- El menor positivo diferente de cero es: 00000001. No podemos poner todos los bits a 0 porque en ese cso se correspondería con el símbolo especial 0.

  El signo es positivo, y el bit oculto junto a la mantisa es 1.0001.

  Recordemos además que el exponente se codifica en exceso Z. Esto es, si el binario normalizado es $+1.0001 \times 2^x$, entonces el exponente codificado es $y = x + Z$, donde el exceso $Z = 2^{n-1} - 1$. Como en este caso $n = 3$, resulta que $Z = 3$. Entonces $0 = x + 3$, lo que implica que $x = -3$.

  Por tanto, el binario normalizado es: $+1.0001 \times 2^{-3} = (0.0010001)_2 = 2^{-3} + 2^{-7}$.

- Los valores correspondientes a los negativos son los opuestos. El mayor valor negativo sería: $-2^{-3}-2^{-7}$. y el menor valor negativo es -15.5.

□

Problema 3. Atención: el siguiente problema se basa en una simplificación de las reglas del IEEE754. Al no usarse las reglas reales, los resultados son solo correctos con las simplificaciones expuestas. En este caso las simplificaciones se reducen al número de bits usados para el exponente y la mantisa.

Con los datos del problema 1, añadiendo los números denormalizados calcula: el menor número denormalizado positivo, el mayor número denormalizado positivo, el menor número normalizado positivo.

Solución.

- El mayor denormalizado positivo es: 00001111. El signo es positivo, y el bit oculto junto a a la mantisa es 0.1111.

  Recordemos además que el exponente se codifica en exceso Z. Para los números denormalizados $Z = 2^{n-1} - 2$. Como $n = 3$, resulta que $Z = 2$. Esto es, si el binario denormalizado es es $+0.1111 \times 2^x$, entonces el exponente codificado es $y = x + Z$. Como en este caso $n = 3$, resulta que $Z = 2$. Como $y = 0 = x + 2$, resulta que $x = -2$.

  Por tanto, el binario fraccionario es:

$$+0.1111 \times 2^{-2} = (0.001111)_2 = 2^{-3} + 2^{-4} + 2^{-5} + 2^{-6}$$

- El menor positivo diferente de cero denormalizado es es: 00000001. No podemos poner todos los bits a 0 porque en ese cso se correspondería con el símbolo especial 0.

  El signo es positivo, y el bit oculto junto a la mantisa es 0.0001. Para los números denormalizados $Z = 2^{n-1} - 2$. Como $n = 3$, resulta que $Z = 2$. Esto es, si el binario denormalizado es es $+0.0001 \times 2^x$, entonces el exponente codificado es $y = x + Z$. Como en este caso $n = 3$, resulta que $Z = 2$. Como $y = 0 = x + 2$, resulta que $x = -2$.

  Por tanto, el binario fraccionario es:

$$+0.0001 \times 2^{-2} = (0.000001)_2 = 2^{-6}$$

- El menor positivo diferente de cero normalizado es: 00010000.

  El signo es positivo, y el bit oculto junto a la mantisa es 1.0000. Para los números normalizados $Z = 2^{n-1} - 1$. Como $n = 3$, resulta que $Z = 3$. Esto es, si el binario normalizado es $+1.0000 \times 2^x$, entonces el exponente codificado es $y = x + Z$, por tanto $y = 1 = x + 2$, resulta que $x = 1$.

Por tanto, el binario fraccionario es:

$$+1.0000 \times 2^1 = (10.00)_2 = (2)_{10}$$

□

Problema 4. En el formato de punto flotante con las normas IEEE754, pero con 3 bits para el exponente y 4 para la mantisa, obtén la representación de los números 0.30 y el número 0.15.

Proof. Comencemos con 0.30. En primer lugar obtenemos la representación en binario fraccionario:

$$0.3 \times 2 = 0.6$$

$$0.6 \times 2 = 1.2$$

$$0.2 \times 2 = 0.4$$

$$0.4 \times 2 = 0.8$$

$$0.8 \times 2 = 1.6$$

$$0.6 \times 2 = 1.2$$

Por tanto $0.3 = 0.010011001\ldots$.

Lo normalizamos: $0.010011001 = 1.0011001 \times 2^{-2}$. El exceso es Z=3. Por tanto, $y = -2 + 3 = 1$. Así pues, el número codificado es 00010011.

En cuanto al 0.15, observar que es la mitad de 0.30. Por tanto, el binario fraccionario es:

$$0.15 = 0.0010011001\ldots$$

Lo normalizamos: $0.0010011001 = 1.0011001 \times 2^{-3}$. Para normalizados el exceso es $Z = 3$. Por tanto $y = -3 + 3 = 0$. El exponente codificado es 0, por lo tanto no puede ser normalizado, es denormalizado.

Así pues, en la ecuación $y = x + Z$ necesariamente $y = 0$. Si calculamos Z para los números denormalizados tenemos $Z = 2^{n-1} - 2$, con lo que $Z = 2$. Entonces, despejando x, tenemos $x = -2$. Planteamos entonces la forma denormalizada, con el exponente -2.

$$0.15 = (0.0010011001)_2 = (0.10011001) \times 2^{-2}$$

Con lo cual:

- Signo. Es 0 al ser el número positivo.
- Exponente. Es 000 por ser denormalizado.
- Mantisa. Copiamos la mantisa de la expresión denormalizada:1001

El código es:

$$00001001$$

□

Problema 5. Supongamos un exponente con 4 bits y una mantisa con 5 bits. Asumir las reglas del estándar IEEE754 para codificar los números a) 0.025, b) 0.00625

Proof. Construimos el binario fraccionario para 0.025.

$$0.025 \times 2 = 0.05$$

$$0.05 \times 2 = 0.1$$

$$0.1 \times 2 = 0.2$$

$$0.2 \times 2 = 0.4$$

$$0.4 \times 2 = 0.8$$

$$0.8 \times 2 = 1.6$$

$$0.6 \times 2 = 1.2$$

$$0.2 \times 2 = 0.4$$

Entonces, $0.025 = 0.00000110011\ldots$. Normalizamos: $0.025 = 1.100110011 \times 2^{-6}$. Codificamos el exponente asumiendo $n = 4$ y por tanto $Z = 7$: $y = x + Z = -6 + 7 = 1$.

Así pues $0.025 = (0000110011)_F$.

Por otra parte, 0.00625 se codifica cono binario fraccionario:

$$0.00625 \times 2 = 0.0125$$

$$0.0125 \times 2 = 0.025$$

$$0.025 \times 2 = 0.05$$

$$0.05 \times 2 = 0.1$$

$$0.1 \times 2 = 0.2$$

$$0.2 \times 2 = 0.4$$

$$0.4 \times 2 = 0.8$$

$$0.8 \times 2 = 1.6$$

$$0.6 \times 2 = 1.2$$

$$0.2 \times 2 = 0.4$$

Por tanto, $0.00625 = 0.0000000110011$. Normalizamos:

$$0.00625 = 1.100110011\ldots \times 2^{-8}$$

Por tanto, como $Z = 7$, entonces $y = -8 + Z = -1$

Como el exponente de los normalizados no puede ser 0 o negativo entonces este número o es denormalizado o no es representable por underflow. Vamos a probar si es denormalizado. En este caso:

$$Z = 2^{n-1} - 2 = 6$$

Por tanto $y = x + Z$. Como $y = 0$ para los denormlizados, $x = -6$. Debemos escribir, el binario fraccionario de 0.00625 con el bit oculto 0 y el exponente $x = -6$. Para ello basta desplazar el punto fraccionario 6 lugares a la derecha.

$$0.00625 = 0.01100110011\ldots \times 2^{-6}$$

Entonces la codificación es: $0.00625 = (0000001100)_{F(4,5)}$.

□

Problema 6. Representar con las reglas del IEEE754 el 2.125 en precisión simple (exponente 8 bits, 32 bits en total).

Proof. Comenzamos por obtener el binario fraccionario:

$$(2.125)_{10} = (10.001)_2$$

Ahora normalizamos:

$$(10.001)_2 = (1.0001)_2 \times 2^1$$

Entonces:

- El bit de signo es 0 ya que el número es positivo.
- La mantisa es 00010000000000000000000. Se ha copiado la mantisa del binario fraccionario y se ha completado con 0 hasta 23 cifras.
- Para obtener el exponente usamos como siempre la fórmula $y = x + Z$. donde y es el valor decimal del exponente codificado interpretado en binario natural, x es el exponente real (en este caso $x = 1$) y Z es el exceso. En este caso, $Z = 2^{n-1} - 1$. Como $n = 8$, $Z = 127$. Por tanto $y = (128)_{10} = (10000000)_2$. El exponente es compatible con las restricciones para números normalizados, por lo que concluimos que efectivamente estamos en el rango de los números normalizados.

Por tanto:

$$(2.125)_{10} = (01000000000010000000000000000000)_{IEEE754}$$

□

Problema 7. Calcular en decimal el número cuyo código binario en IEEE754 es el 0x00700000

Proof. Se trata de un problema de decodificación donde la cadena binario nos la dan en hexadecimal. Por tanto, lo primero es pasar a binario:

$$(00700000)_{16} = (00000000011100000000000000000000)_2$$

Es decir:

- Bit de signo: 0.
- Exponente: 000000000
- Mantisa: 11100000000000000000000

Como vemos, todos los bits del exponente son cero. Esto quiere decir que se trata de un número denormalizado:

- Signo. Al ser el bit se signo 0 el número es positivo.
- Mantisa. Copiamos la mantisa, considerando como bit oculto el 0 ya que estamos en el rango de los denormalizados: 0.111.
- Exponente. Aplicamos la fórmula $y = x + Z$. El valor decimal del exponente codificado interpretado en binario natural es $y = 0$. El exceso viene dado por $Z = 2^{n-1} - 2$ al tratarse de un número denormalizado. Por tanto, como $n = 8$, resulta que $Z = 126$. De ahí deducimos que $x = -126$.

Por lo tanto, el número es:

$$+(0.111)_2 \times 2^{-126} = 2^{-127} + 2^{-128} + 2^{-129}$$

□

Problema 8. Obtén la representación del número 1.84×10^{-39} en el formato IEEE754 precisión simple.

Proof. Como siempre tendremos que comenzar obteniendo el binario fraccionario. Sin embargo, aquí no podemos usar la técnica de los residuos en la parte fraccionaria porque tendríamos que hacer muchas multiplicaciones hasta llegar al primer bit a 1.

Una forma de resolverlo es la siguiente:

La estrategia consiste en buscar cuantas veces hay que multiplicar por 2 para llegar a obtener el primer bit a 1. Esto se obtiene con la fórmula:

$$1.84 \times 10^{-39} \times 2^n > 1$$

Vamos a despejar n.

$$2^n > \frac{1}{1.84} \times 10^{39}$$

$$n > log_2(\frac{1}{1.84} \times 10^{39})$$

$$n > 128.67$$

El siguiente entero es 129. Así, pues si multiplicamos por 2, 129 veces obtendremos el primer bit a 1. Podemos entonces escribir:

$$1.84 \times 10^{-39} = (2^{129} \times 1.84 \times 10^{-39}) \times 2^{-129}$$

Hemos multiplicado y dividio por 2^{129}, de forma que:

$$1.84 \times 10^{-39} = 2^-129 \times 1.2522$$

Ahora obtenemos el binario fraccionario de 1.2522.

$$1.2522 = (1.010000001)_2$$

Por tanto:

$$1.84 \times 10^{-39} = 2^-129 \times (1.010000001)_2$$

Si observamos, el número ya está normalizado. Si intentamos codificarlo como normalizado, para un exponente de 8 bits sabemos que $Z = 127$.

Aplicando la ecuación $y = x + Z$, tenemos que $y = -2$. Como es negativo, sabemos que el número no puede ser del rango de los normalizados.

Intentamos codificarlo entonces como denormalizado. En ese caso $y = 0$, $Z = 2^{n-1} - 2 = 126$, por lo que $x = -126$.

La forma denormalizada sería:

$$1.84 \times 10^{-39} = (0.001010000001)_2 \times 2^{-126}$$

En esta forma:

- Signo=0
- Exponente= 00000000
- Mantisa = 00101000000010000000000

El código sería: 00000000000101000000100000000000

□