Problemas sobre representación de la información (Ejemplos resueltos)

March 15, 2023

Problema 1. Representa en binario natural los siguientes números:

- $(122)_{10}$.
- $(141)_{10}$.
- $(5824)_{10}$.

Proof. Solución

Utilizaremos el método de los residuos. En la siguiente tabla puedes ver la secuencia de cocientes y restos. Cada fila es una división:

122_{C_2}	122_{R_2}	141_{C_2}	141_{R_2}	5824_{C_2}	5824_{R_2}
61	0	70	1	2912	0
30	1	35	0	1456	0
15	0	17	1	728	0
7	1	8	1	364	0
3	1	4	0	182	0
$1 < 2$	1	2	0	91	0
		$1 < 2$	0	45	1
				22	1
				11	0
				5	1
				2	1
				$1 < 2$	0

Entonces, construimos la representación binaria natural, basada en la notación posicional para la base 2, tomando como bit más significativo el resultado del último cociente y a continuación todos los restos. De este modo:

$$(122)_{10} = (1111010)_2$$
$$(141)_{10} = (10001101)_2$$
$$(5824)_{10} = (1011011000000)_2$$

□

Problema 2. Pasar a base octal y a base hexadecimal los números del problema anterior.

Proof. Solución Este problema se puede resolver de dos formas.

Utilizando el método de los residuos: En base octal:

122_{C_8}	122_{R_8}	141_{C_8}	141_{R_8}	5824_{C_8}	5824_{R_8}
15	2	17	5	728	0
$1 < 8$	7	$2 < 8$	1	91	0
				11	3
				$1 < 8$	3

Por tanto:

$$(122)_{10} = (172)_8$$
$$(141)_{10} = (215)_8$$
$$(5824)_{10} = (13300)_8$$

□

En base hexadecimal:

$122_{C_{16}}$	$122_{R_{16}}$	$141_{C_{16}}$	$141_{R_{16}}$	$5824_{C_{16}}$	$5824_{R_{16}}$
$7 < 16$	$10(A)$	$8 < 16$	$13(D)$	364	0
				22	$12(C)$
				$1 < 16$	6

Por tanto:

$$(122)_{10} = (7A)_16$$
$$(141)_{10} = (8D)_16$$
$$(5824)_{10} = (16C0)_16$$

A partir de la codificación binaria: Puesto que:

$$(122)_{10} = (1111010)_2$$
$$(141)_{10} = (10001101)_2$$
$$(5824)_{10} = (1011011000000)_2$$

Para pasar a octal tomamos los bits de tres en tres de derecha a izquierda y completamos con ceros al final:

$$(122)_{10} = ((001)(111)(010))_2$$
$$(141)_{10} = ((010)(001)(101))_2$$
$$(5824)_{10} = ((001)(011)(011)(000)(000))_2$$

Ahora traducimos cada grupo de tres bits a binario natura. El resultado es la cifra octal que le corresponde:

$$(122)_{10} = ((1)(7)(2))_8$$
$$(141)_{10} = ((2)(1)(5))_8$$
$$(5824)_{10} = ((1)(3)(3)(0)(0))_8$$

Para pasar a hexadecimal tomamos los bits de cuatro en cuatro de derecha a izquierda y completamos con ceros al final:

$$(122)_{10} = ((0111)(1010))_2$$
$$(141)_{10} = ((1000)(1101))_2$$
$$(5824)_{10} = ((0001)(0110)(1100)(0000))_2$$

Ahora traducimos cada grupo de cuatro bits a binario natural. El resultado numérico debe traducirse a la cifra hexadecimal que le corresponde:

$$(122)_{10} = ((7)(A))_16$$
$$(141)_{10} = ((8)(D))_16$$
$$(5824)_{10} = ((1)(6)(C)(0))_16$$

Observamos como la representación octal y hexadecimal acorta notablemente la longitud de la representación en binario natural.

Problema 3. Asumir una representación binaria natural con número ilimitado de bits fraccionarios para codificar los siguientes números:

- 13.5
- 24.0125
- 5.3
- 2.2

Proof. Solución:
Comenzamos con 13.5. La parte entera es 13. Usamos el método de los residuos y obtenemos:

$$13 = 6 \times 2 + 1$$
$$6 = 3 \times 2 + 0$$
$$3 = 1 \times 2 + 1$$

$$(13)_{10} = (1101)_2$$

Ahora debemos convertir la parte fraccionaria 0.5. Para ello hemos de ir multiplicando por 2 la parte fraccionaria y quedándonos con la parte entera hasta que se repita una parte fraccionaria o esta sea 0.

P. fracc.	Op.	Resultado
0.5	$\times 2$	1.0
0.0	Fin	

Entonces, tomamos la partes enteras de la columna de resultados:

$$(0.5)_{10} = (0.1)_2$$

Por tanto, $(13.5)_{10} = (1101.1)_2$
Pasemos ahora 24.0125. La parte entera 24 es $(11000)_2$, veamos ahora la parte fraccionaria:

P. fracc.	Op.	Resultado
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	Repetido	

Entonces, $0.0125 = 0.000\overline{00011}$.

Así pues $24.0125 = (11000.000\overline{00011}$. Es importante recordar que hacen falta un número infinito de bits para representar con exactitud el número aunque éste admita una representación exacta en base 10.

En el caso de 5.3 tenemos $(5)_{10} = (101)_2$. La parte fraccionaria:

P. fracc.	Op.	Resultado
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
0.2	Repetido	

Por tanto $5 = (101.01\overline{0011})_2$.

Por último, $2.2 = (10.\overline{0011})_2$.

□

Problema 4. Asumir una representación binario natural en punto fijo con el punto fraccionario separando una parte de 8 bits y una parte fraccionaria de 3 bits para completar la siguiente tabla:

Base decimal	**Binario natural fraccionario**	Error
125.25		
232.13		
	$(11010.011)_2$	

Solución. En este caso tenemos un número fijo de bits para la parte entera y para la parte fraccionaria. El número 125 en binario natural es $(1111101)_2$. Como la parte entera debe tener 8 bits en la parte fija, tenemos que añadir un bit 0 en el lado más significativo $(01111101)_2$.

Calculamos la parte fraccionaria:

P. fracc.	Op.	Resultado
0.25	$\times 2$	0.5
0.5	$\times 2$	1.0
0.0	Fin	

Por tanto: $(0.25)_{10} = (0.01)_2$. Como necesitamos 3 bits para la parte fraccionaria debemos añadir un bit más, esta vez en el lado menos significativo: $(0.25)_{10} = (0.010)_2$. Por tanto:

$$(125.25)_{10} = (01111101.010)_2$$

La conversión en este caso es exacta ya que el resto de bits en la parte menos significativa es 0. Esto se puede comprobar fácilmente ya que $(0.010)_2 = 0 \times 2^{-1} + 1 \times 2^{-2} = 0.25$.

Hacemos lo mismo con 232.13. En este caso, $(232)_{10} = (11101000)_2$. La parte fraccionaria:

P. fracc.	Op.	Resultado
0.13	$\times 2$	0.26
0.26	$\times 2$	0.52
0.52	$\times 2$	1.04

Aunque como sabemos el proceso de conversión continuaría debemos detenernos ya que sólo disponemos de tres bits.

En este caso $(0.001)_2 = 2^{-3} = 0.125$. Por tanto hay un error: $e = 0.13 - 0.125 = 0.05$. Se trata de un error de precisión inherente a disponer sólo de un número finito de bits para representar la parte fraccionaria del número.

Finalmente tenemos la conversión a decimal de $(11010.011)_2$ asumiendo una interpretación en binario natural. En este caso simplemente aplicamos la regla posicional:

$$(11010.011)_2 = 2^4 + 2^3 + 2^1 + 2^{-2} + 2^{-3} = 26.375$$

En este caso el error es 0 porque no hay ninguna limitación establecida para la representación en base decimal.

□

Problema 5. Obtén todos los códigos binarios y su significado en base decimal para las representaciones módulo-signo, complemento a 1 y complemento a 2 para un celda de 4 bits con 0 bits en la parte fraccionaria.

Proof. En módulo signo tenemos que reservar un bit para el signo: un 1 es negativo y un 0 es positivo. El resto de bits nos proporciona el módulo del número usando una interpretación en binario natural, entonces:

Código	Interpetación módulo signo
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

En la interpretación de complemento a 1, los códigos cuyo bit más significativo es 0 tienen la misma interpretación que en binario natural. El resto de códigos puede interpretarse en binario natural, después de cambiar los bits y devolver el resultado con signo negativo.

Código	Interpetación de compl. a 1
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-7
1001	-6
1010	-5
1011	-4
1100	-3
1101	-2
1110	-1
1111	-0

En la interpretación de complemento a 2, nuevamente todos los códigos cuyo bit más significativo es 0 pueden interpretarse del mismo modo que en binario natural. El resto de códigos puede interpretarse en binario natural, después de copiar cada bit desde el menos significativo hasta el primer bit a 1, a partir de ahí se debe cambiar cada bit.

Código	Interpetación de compl. a 1
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

□

Problema 6. Obtener el código binario para una celda de 5 bits que bajo una interpretación de complemento a 2 representa a cada uno de los siguientes números:

- -8
- -16
- 9
- 16

Proof. Solución

Este tipo de problemas requiere en primer lugar el rango de la representación para comprobar en cada caso si el número se encuentra dentro o fuera del rango. Para complemento a 2, el rango se puede calcular con la siguiente fórmula:

$$R_{C_2} = \{-2^{n-1}, ..., 2^{n-1} - 1\}$$

donde n es el número de bits de la celda.

En este caso $n = 5$ por lo que $R = \{-2^4, ..., 2^4 - 1\} = \{-16, ..., 15\}$.

El valor -8 está en el rango, así que se puede proceder a la obtención del código. Al ser un número negativo tenemos que obtener en primer lugar su representación en binario natural sobre la celda de 5 bits, esto es: $8 = (01000)_2$. Ahora hay que realizar la operación NEG sobre la combinación de bits resultante, es decir:

$$-8 =_{C_2} NEG(01000)$$

Recordemos que NEG implica copiar los bits del menos significativo al más significativo hasta encontrar el primer 1. A partir de ahí hay que cambiar los restantes bits. Por tanto:

$$-8 = (11000)_{C_2}$$

El valor -16 también está en el rango. Procedemos del mismo modo:

$$-16 =_{C_2} NEG(10000)$$

$$-16 = (10000)_{C_2}$$

El valor 9 también está en el rango. Como es un número positivo entonces simplemente hay que usar la representación de binario natural sobre la celda de 5 bits. Entonces:

$$9 = (01001)_{C_2}$$

Finalmente el valor 16 resulta que no está en el rango. Por tanto no podemos obtener una representación bajo la interpretación de complemento a 2 en una celda de 5 bits para el 16.

□