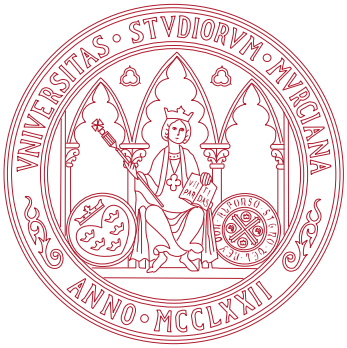

Universidad de Murcia

Facultad de Informática

Universidad Politécnica de Cartagena

Escuela Técnica Superior de Ingeniería de Telecomunicación

TÍTULO DE GRADO EN
CIENCIA E INGENIERÍA DE DATOS

Fundamentos de Computadores

Tema 2: Representación de la información

Boletín de autoevaluación de teoría / problemas

CURSO 2023 / 24

Índice general

Ejercicios resueltos

1. Sean los dos números expresados en octal 75_8 y 54_8 . ¿Cuál será el resultado de hacer el AND lógico, bit a bit en las posiciones correspondientes de ambos (expresado también en octal)?

2. ¿El número $1230)_4$ expresado en base 4 se corresponde con qué número expresado en base decimal?

3. El siguiente ejercicio explora las conversiones de números binarios con y sin signo a números decimales. Dado el número $(00100100100100100100100100100100)_2$:

   a) Si está representado en C2 (complemento a 2), ¿qué número decimal representa?

   b) Si está representado como un entero sin signo, ¿qué número decimal representa?

   c) Si está representado en notación sesgada, ¿qué número decimal representa?

   d) Por último, ¿qué número hexadecimal representa?

4. Tenemos el número 100_{10}. Responde a los siguientes apartados:

   a) ¿Cuál es su representación en base 2?

   b) ¿Cuál es su representación en base hexadecimal?

   c) ¿Cuál es su representación en base octal?

   d) Ahora dividimos el número por 4. ¿Cual es su representación en base hexadecimal?

   e) Si el número fuera negativo -100_{10}, ¿cómo lo representarías en complemento a 2 (C2) con un tamaño de 8 bits?

5. Asumiendo formato IEEE 754 en simple precisión, ¿qué número decimal es representado por la siguiente ristra de bits **1 01111101 00100000000000000000000**?

6. El formato que vamos a usar en este problema es un formato en coma flotante normalizado IEEE 754 de 8 bits, con 1 bit para el signo, 4 bits para el exponente y 3 bits para la mantisa. Es idéntico al formato de 32 y 64 bits visto en clase en cuanto al significado de los campos y codificación de números especiales. Asume que usamos redondeo al número par más cercano especificado en el estándar IEEE. Se pide:

   a) Codifica el número $0{,}0011011_2$ en formato IEEE-754 de 8 bits.

   b) Codifica el número $16{,}0_{10}$ en formato IEEE-754 de 8 bits.

   c) Decodifica el siguiente número en formato IEEE-754 de 8 bits 01010101 a su valor decimal.

   d) Codifica el número $+1{,}0_2$ en formato IEEE-754 de 8 bits.

   e) Codifica el número $0{,}0_2$ en formato IEEE-754 de 8 bits.

7. Vamos a practicar en este ejercicio varios problemas de redondeo, asumiendo que usamos redondeo al número par más cercano especificado en el estándar IEEE.

   a) Dado el número binario $+0{,}100101110$ redondéalo usando los tres últimos bits, de tal forma que tenga tan sólo 6 dígitos significativos después de la coma.

   b) Dado el número binario $+0{,}10010$ redondéalo usando los últimos 2 bits, de tal forma que tenga tan sólo 3 dígitos significativos después de la coma.

   c) Dado el número binario $-0{,}10010111$ redondéalo usando los dos últimos bits, de tal forma que tenga tan sólo 6 dígitos significativos después de la coma.

8. Vamos a hacer varios ejercicios acerca del estándar IEEE 754.

   a) Decodifica la representación **0 1010 101** en formato de 8-bits IEEE 754 a su valor decimal.

   b) En el formato de 32-bits IEEE 754, ¿cómo se codifica el cero negativo?

   c) En el formato de 32-bits IEEE 754, ¿cómo se codifica el infinito positivo?

   d) Dados los números **1 1100 100** y **1 1100 101** en formato de 8-bits IEEE 754, decir qué número es mayor poniendo su valor numérico en base 10.

   e) Usando el formato de 32-bits IEEE 754 de simple precisión, muestra cual es la representación de -11/16 ($-0,6875_{10}$).

9. Dada la tira de caracteres *01234*

   a) Traduce la tira a su valor ASCII decimal

   b) Traduce la tira a su valor Unicode de 16-bits (usando notación hexadecimal y expresando los respectivos *code points*).

   c) Por último, traduce a texto los siguientes valores de caracteres expresados en ASCII (hexadecimal): **41 44 44**

10. Tenemos el número fraccionario decimal $22,22_{10}$. Responde a los siguientes apartados:

    a) ¿Cual es su representación en base 2? (Utiliza 10 bits en total para representar el número).

    b) ¿Cual es su representación en formato en coma flotante normalizado IEEE 754 de 8 bits, con 1 bit para el signo, 4 bits para el exponente y 3 bits para la mantisa? Asume que usamos redondeo al número par más cercano especificado en el estándar IEEE utilizando dos bits.

11. Tenemos el siguiente byte de información: 10100001. Se pide:

    a) Si representa un número entero SIN signo, ¿cómo codificaría el ordenador la misma información en 2 bytes de tamaño? Nota: Da la respuesta en hexadecimal.

    b) Si representa un número entero CON signo, en formato signo y magnitud, ¿cómo codificaría el ordenador la misma información en 2 bytes de tamaño? Nota: Da la respuesta en hexadecimal.

    c) Si representa un número entero CON signo, en formato complemento a dos (C2), ¿cómo codificaría el ordenador la misma información en 2 bytes de tamaño? Nota: Da la respuesta en hexadecimal. Decir también el número entero en decimal representado.

    d) Si representa un carácter, en ISO 8859-1, ¿de qué carácter se trata?

    e) Si el mismo carácter anterior lo queremos representar en UNICODE UTF-16 con formato Big endian, ¿cómo se representaría?. Nota: Da la respuesta en hexadecimal.

    f) Si el mismo carácter anterior lo queremos representar en UNICODE UTF-8, ¿cómo se representaría? Nota: Da la respuesta en hexadecimal.

Ejercicios propuestos

1. El entero negativo menor que puede representarse en 8 bits es (usando representación en complemento a 2):

   a) -256.

   b) -128.

   c) -127.

2. El número obtenido al desplazar una secuencia de bits 5 posiciones a la izquierda (añadiendo ceros por la derecha) es:

   a) El número original multiplicado por 5.

   b) El número original multiplicado por 32.

   c) El número original dividido por 5.

3. Sea el número de 16 bits A1AC, expresado en C2 de 16 bits. Si quisiéramos extenderlo a 32 bits, representando el mismo valor entero, tendríamos el siguiente valor de 32 bits (todas las secuencias de bits expresadas en hexadecimal):

   a) 0000A1AC

   b) 1111A1AC

   c) FFFFA1AC

4. El resultado de hacer la operación lógica OR (bit a bit entre los bits de posiciones correspondientes), entre los bytes $F9)_{16}$ y $A3)_{16}$ será:

   a) $F7)_{16}$

   b) $FB)_{16}$

   c) $9C)_{16}$

5. Sea el número de 16 bits 8CAF, expresado en C2 de 16 bits. Si quisiéramos extenderlo a 32 bits, representando el mismo valor entero, tendríamos el siguiente valor de 32 bits (todas las secuencias de bits expresadas en hexadecimal):

   a) 00008CAF.

   b) 111181AB.

   c) Ninguna de las respuestas anteriores es correcta.

6. Disponemos de un alfabeto de exactamente quinientos símbolos, que querríamos representar mediante un código de E/S con longitud mínima de bits. ¿Cuántos bits necesitaremos para representar un símbolo de dicho alfabeto?

   a) 8 bits.

   b) 9 bits.

   c) 500 bits.

7. Supóngase que el resultado de una operación IEEE 754 ha sido 1011 1111 1101 1101 1101 1011 1011 1101 (simple precisión), y los bits de redondeo y retenedor (es decir, posiciones de mantisa -24 y -25) han tomado, respectivamente, los valores 1 y 1. El resultado final debidamente redondeado será:

a) 1011 1111 1101 1101 1101 1011 1011 1110.

b) 1011 1111 1101 1101 1101 1011 1011 1100.

c) Ninguna de las anteriores respuestas es correcta.

8. El formato que vamos a usar en las siguientes cuestiones es un formato en coma flotante normalizado IEEE 754 de 8 bits, con 1 bit para el signo, 4 bits para el exponente y 3 bits para la mantisa. Es idéntico al formato de 32 y 64 bits visto en clase en cuanto al significado de los campos y codificación de números especiales. Asume que usamos redondeo al número par más cercano especificado en el estándar IEEE. Se pide:

   a) Codifica el número 0,011011 en base 2 en formato IEEE-754 de 8 bits.

   b) Codifica el número 15.0 en base decimal en formato IEEE-754 de 8 bits.

   c) Decodifica el siguiente número en formato IEEE-754 de 8 bits 11010101 a su valor decimal.

   d) Codifica el número -1.0 en binario en formato IEEE-754 de 8 bits.

   e) Codifica el número 0.0 en binario en formato IEEE-754 de 8 bits.

9. Considere de nuevo el formato en coma flotante normalizado IEEE 754 de 8 bits, con 1 bit para el signo, 4 bits para el exponente (sesgo S=7) y 3 bits para la mantisa, idéntico al formato de 32 y 64 bits visto en clase en cuanto al significado de los campos y codificación de números especiales. Sea entonces el número fraccionario 25/4 (dado en decimal). Responda a los siguientes apartados:

   a) ¿Cual es su representación en base 2? (utiliza como mucho 8 bits para representar el número)

   b) ¿Cual es su representación en formato en coma flotante normalizado IEEE 754 de 8 bits comentado? Asuma que usamos redondeo al número par más cercano especificado en el estándar IEEE utilizando dos bits.

10. El número de bits necesarios para almacenar los píxeles de una imagen PPM en formato RGB, con cada valor de R, G y B entre 0 y 255 (sin contar la cabecera) es.

    a) Número de filas * Número de columnas * 255

    b) Número de filas * Número de columnas * 3 * 8

    c) Número de filas * Número de columnas * 256 * 3

11. Dada la tira de caracteres *hello*:

    a) Traduce la tira a su valor ASCII decimal.

    b) Traduce la tira a su valor Unicode de 16-bits (usando notación hexadecimal y expresando los respectivos *code points*).

    c) Por último, traduce a texto los siguientes valores de caracteres expresados en ASCII (hexadecimal): **4D 49 50 53**.

12. Realizar la resta de los números decimales $16 - 84$ usando la representación en C2 de 8 bits. Para ello:

    a) Expresa los números $16)_{10}$ y $-84)_{10}$ en C2 de 8 bits.

    b) Suma las dos ristras de 8 bits resultantes para obtener otros 8 bits de resultado (despreciando, en su caso, el posible acarreo adicional).

    c) Comprueba, realizando las cuentas pertinentes, que la ristra de bits resultante se corresponde efectivamente con la representación en C2 del número $-68)_{10}$.

13. Repetir el ejercicio anterior, manteniendo todo igual, pero esta vez para realizar la operación de suma de los números decimales negativos -100 y -84 (e.d. realizar la operación $-100-84$), de nuevo usando la representación en C2 de 8 bits. ¿Es correcto el resultado obtenido? En caso negativo, explica qué es lo que está pasando.

Soluciones a los ejercicios resueltos

1. Este ejercicio es muy fácil. Lo primero que hacemos es pasar cada número octal a binario:

   $75_8 = 111101_2$

   $54_8 = 101100_2$

   Y ahora hacemos el AND lógico bit a bit, con lo que obtenemos:

   111101_2 AND 101100_2 = 101100_2 que representado en octal da el número: 54_8.

2. Para calcularlo, al ser en base 4 aplicamos la formula:

   $N_{10} = 1 * 4^3 + 2 * 4^2 + 3 * 4^1 + 0 * 4^0 = 108$

   Por tanto, es el número 108 expresado en base 10.

3. Tenemos las siguientes respuestas para los diversos apartados:

   a) La forma de conversión de un número en C2 a decimal es:
   Número = $x_{31} * (-2)^{31} + x_{30} * 2^{30} + ... + x_1 * 2^1 + x_0 * 2^0 =$
   $= 1 * 2^{29} + 1 * 2^{26} + 1 * 2^{23} + 1 * 2^{20} + 1 * 2^{17} + 1 * 2^{14} + 1 * 2^{11} + 1 * 2^8 + 1 * 2^5 + 1 * 2^2$
   $= 613{,}566{,}756_{10}$

   b) Al estar representado como un entero sin signo, aplicamos los pesos que tienen en base 2 a cada uno de los dígitos:
   Número = $x_{31} * 2^{31} + x_{30} * 2^{30} + ... + x_1 * 2^1 + x_0 * 2^0 =$
   $= 1 * 2^{29} + 1 * 2^{26} + 1 * 2^{23} + 1 * 2^{20} + 1 * 2^{17} + 1 * 2^{14} + 1 * 2^{11} + 1 * 2^8 + 1 * 2^5 + 1 * 2^2$
   $= 613{,}566{,}756_{10}$
   Como vemos, obtenemos el mismo valor que en el apartado del C2 debido a ser número positivo dentro del rango del C2.

   c) Sabemos que la notación sesgada representa números en el rango: $[-2^{n-1} \dots 2^{n-1} - 1]$, siendo el valor del sesgo 2^{n-1}, y por tanto el cero representado por $1000 \dots 00_2$.
   Por lo tanto, al valor que represente el número le tenemos que restar el sesgo, que en este caso es de 2^{31}. Entonces:
   Número = $(Valor\_entero\_sin\_signo - 2^{31})_{10} = 613{,}566{,}756_{10} - 2{,}147{,}483{,}648_{10} = -1{,}533{,}916{,}892_{10}$

   d) El número $(00100100100100100100100100100100)_2$ en base binaria es el siguiente número en base hexadecimal: 24924924_{16}

4. Tenemos las siguientes respuestas para los diversos apartados:

   a) Como ya se explicó, para pasar de decimal a binario se va dividiendo por 2 el número. Los restos y el último cociente son las cifras binarias, en orden LSB → MSB (de derecha a izquierda).
   Por tanto:
   100 / 2 = 50 (R=0) / 2 = 25 (R=0) / 2 = 12 (R=1) / 2 = 6 (R=0) / 2 = 3 (R=0) / 2 = **1** (R=1)
   Y el número obtenido es: 1100100_2, que podemos comprobar que corresponde a 100:
   $1100100_2 = 1 * 2^6 + 1 * 2^5 + 0 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 0 * 2^0 = 64 + 32 + 4 = 100_{10}$

   b) El procedimiento más sencillo es usar la representación binaria, y pasar esta a la hexadecimal, agrupando los bits en grupos de 4. Tenemos:
   $1100100_2 = 0110\ \ 0100 = 64_{16}$

c) Vamos a seguir utilizando el procedimiento de pasar de la representación binaria a la octal, agrupando los bits en grupos de 3. Tenemos:
$1100100_2 = 001\ \ 100\ \ 100 = 144_8$

d) Dividir por 4 es desplazar el número binario 2 veces a la derecha. Una vez hecho esto, lo podemos representar en hexadecimal. Entonces:
$1100100_2/4 = 11001_2 = 0001\ \ 1001 = 19_{16}$

e) Como sabemos, la representación de un número x negativo en C2 es igual a $\bar{x} + 1$. Entonces:
$x = 100_{10} = 01100100_2$ (en representación de 8 bits).
Entonces: $\bar{x} = 10011011_2$ y resulta:
$C2(x) = \bar{x} + 1 = 10011100_2$
Por tanto, el número $-x = -100_{10}$ representado en C2 es igual a 10011100_2 (en representación de 8 bits).

5. La palabra dada de 32 bits representa el siguiente número decimal:

$$Numero = (-1)^1 * (2^{(125-127)}) * (1,001_2) = (-1) * (0,01001_2) = -0,28125_{10}$$

6. *a*) El número $0,0011011_2$ es equivalente a $1,1011_2 * 2^{-3}$, que es a la vez igual a $(-1)^0 * (1,1011)_2 * 2^{(4-7)}$
El número $(0,1011)_2$ tenemos que redondearlo, pues la mantisa solo puede contener 3 bits. Como sólo hay 1 bit de redondeo, y es 1, significa que el número par más cercano es el que está sumándole 1 a la parte de mantisa que nos quedaba, es decir, $101_2 + 1_2 = 110_2$.
Y obtenemos que el número pedido en el formato de 8 bits IEEE 754 es el: **0 0100 110**.

b) El número $16,0_{10}$ es igual a $10000,0_2$, que se expresar también como $(-1)^0 * (1,000)_2 * 2^{(11-7)}$. Como la mantisa ya es de 3 bits, no hace falta el redondeo, por lo que el número pedido en el formato de 8 bits IEEE 754 es el: **0 1011 000**.

c) La mantisa tiene el valor 101, lo cual representa el número $1,101_2$, que representa el número decimal $1,625_{10}$. Por tanto, el valor decimal es el siguiente: $(-1)^0 * (2)^{(10-7)} * 1,625 = +13,0$

d) El número $1,0_2$ es equivalente a $1,000_2 * 2^0$, que es a la vez igual a $(-1)^0 * (1,000)_2 * 2^{(7-7)}$, por lo que el número pedido en el formato de 8 bits IEEE 754 es el: **0 0111 000**.

e) Esta pregunta es sencillísima, pues por convención, en el formato IEEE 754 el 0 es representado por la ristra de bits **0 0000 000**.

7. *a*) El número $+0,100101110$ lo descomponemos entre los bits a ser mantenidos $+0,100101$ y los bits que nos a servir para el redondeo 110. Al ser el número positivo y estar el valor 110 por encima de la mitad (100), el valor más cercano se obtiene sumándole uno a los bits que habíamos mantenido. Por tanto, el número redondeado con 6 dígitos es $+0,100101 + 0,000001 = +0,100110$.

b) En este caso, el número $+0,10010$ lo descomponemos entre los bits a ser mantenidos $+0,100$ y los bits que nos a servir para el redondeo 10. El valor 10 nos da un empate entre el número $+0,100$ y el número $+0,101$. El estándar IEEE de redondeo dice que en caso de empate tenemos que optar por el número par más cercano (cuya último bit (LSB) siempre acaba en 0). Por lo tanto, en este caso debemos simplemente truncar el número a los tres bits que nos habían pedido, obteniendo el número redondeado $+0,100$.

c) El número $-0,10010111$ lo descomponemos entre los bits a ser mantenidos $-0,100101$ y los bits que nos a servir para el redondeo 11. Al ser el número positivo y estar el valor 11 por encima de la mitad (10), el valor más cercano se obtiene restándole uno a los bits que habíamos mantenido. Por tanto, el número redondeado con 6 dígitos es $-0,100101 - 0,000001 = -0,100110$.

8. *a*) El valor decimal es $(-1)^0 * (1,101_2) * 2^{10-7} = 13,0$

b) La representación del cero negativo en 32-bits IEEE 754 es la siguiente:
1 00000000 00000000000000000000000

c) La representación del infinito positivo en 32-bits IEEE 754 es la siguiente:
0 11111111 00000000000000000000000

d) Para ello, vamos a ver qué número decimal representa cada uno de ellos:
El primer número es: $(-1)^1 * (1{,}1)_2 * 2^{12-7} = -48,0$
El segundo número es: $(-1)^1 * (1{,}101)_2 * 2^{12-7} = -52,0$
Por tanto, el primer número es mayor que el segundo.

e) Aplicando el procedimiento explicado en clase, la representación de $-0,6875_{10}$ es:
1 01111110 01100000000000000000000000

9. *a*) La tira *01234* tiene el siguiente valor ASCII expresado en decimal: **48 49 50 51 52**

b) La tira *01234* se corresponde con los siguientes valores Unicode de 16-bits (usando notación hexadecimal): **U+0030, U+0031, U+0032, U+0033, U+0034**

c) La solución es: **ADD**

10. Tenemos las siguientes respuestas para los diversos apartados:

a) Como ya se explicó, para pasar de decimal a binario se va dividiendo por 2 la parte entera del número, siendo los restos y el último cociente las cifras binarias enteras, en orden LSB → MSB. Para pasar la parte fraccionaria del número de decimal a binario se va multiplicando por 2, siendo la partes enteras obtenidas las cifras binarias fraccionarias, en orden MSB → LSB.
Por tanto, para obtener la parte entera:
22 / 2 = 11 (R=0) / 2 = 5 (R=1) / 2 = 2 (R=1) / 2 = **1** (R=0)
Y la parte entera del número obtenido es: 10110_2.
Como vemos, necesitamos 5 bits para representar la parte entera. Por tanto nos quedan otros cinco para representar la parte fraccionaria. Aplicamos la teoría para obtener la parte fraccionaria:
0,22 * 2 = **0**,44 * 2 = **0**,88 * 2 = **1**,76 * 2 = **1**,52 * 2 = **1**,04
Y la parte fraccionaria del número obtenido es: 00111_2.
Por lo tanto, el número fraccionario decimal $22,22_{10}$ se corresponde al número binario: $10110,00111_2$.

b) Vamos a partir del número en formato binario. Tenemos que:
$10110,00111_2 = 1,011000111_2 * 2^4$
Como la mantisa sólo puede tener tres bits, tenemos que:
$1,011000111_2 = 1,011 + Redondeo(000111)$
Como no nos dicen lo contrario, hacemos uso del mecanismo base de quedarnos con 2 bits para el redondeo (00), que indica que hay truncar el número dejándolo con los tres dígitos. Por lo tanto, el número queda: $= (-1)^0 * (1,011)_2 * 2^{(4+7)}$, obteniendo la siguiente secuencia en formato en coma flotante normalizado IEEE 754 de 8 bits: **0 1011 011**

11. *a*) El número representado como entero SIN signo corresponde a $A1_{16}$. Si lo queremos representar con 2 bytes, entonces tenemos: $00A1_{16}$.

b) El número representado CON signo en formato signo y magnitud corresponde a -21_{16}. Si lo queremos representar con 2 bytes, al tratarse de un número negativo entonces tenemos: 8021_{16}.

c) El número representado CON signo en formato C2 corresponde a -95_{10}. Si lo queremos representar con 2 bytes, extendemos el signo y tenemos: $FFA1_{16}$.

d) En ISO 8859-1, el valor anterior ($A1_{16}$) corresponde al carácter: ¡

e) En Unicode con codificación UTF-16, tenemos 2 bytes (16 bits) para dicho carácter, que se representan por el *code point* $U + 00A1_{16}$. Como nos dicen que el formato es Big endian, entonces tenemos: $00A1_{16}$ que al separarlo en dos bytes queda como 0x00 y 0xA1.

f) En Unicode con codificación UTF-8, necesitaremos 2 bytes para dicho *code point*, mirando la tabla de las transparencias de teoría. El carácter solicitado en binario se representa como 10100001. Para la codificación como UTF-8, la tabla nos dice que para ese rango de números se codifica de la forma 110yyyyy 10xxxxxx. Comparando con nuestro número tenemos que $yyyyy = 00010$ y que $xxxxxx = 100001$. Por tanto, obtenemos la siguiente representación final: $11000010\ 10100001_2 = C2A1_{16}$, que al separarlo en dos bytes queda como 0xC2 0xA1.