Fundamentos de Computadores (FCO)

Tema 5: Representación de la información

GRADO EN INFORMÁTICA

Contenido

1 – Números naturales

1) ¿Cuál es el rango representable con 5 dígitos en base 10? Indíquelo en base 10.

SOLUCIÓN:

Número de valores representables = 10^5 = 100000
Rango: [0; 99999]

2) ¿Cuál es el rango representable con 5 dígitos en base 2? Indíquelo en base 10 y base 2.

SOLUCIÓN:

Número de valores representables = 2^5 = 32
Rango: [0; 31] = [00000; 11111]

3) ¿Cuál es el rango representable con 5 dígitos en base 8? Indíquelo en base 10 y base 8.

SOLUCIÓN:

Número de valores representables = 8^5 = 32768
Rango: [0; 32767] = [0; 77777]

4) ¿Cuál es el rango representable con 5 dígitos en base 16? Indíquelo en base 10 y base 16.

SOLUCIÓN:

Número de valores representables = 16^5 = 1048576
Rango: [0; 1048575] = [0; FFFFF]

5) Convierta la cantidad 12 representada en base 10 a base 2.

SOLUCIÓN:

12_{10} : 2 = 6 : 2 = 3 : 2 = 1 $12_{10} = 1100_2$
r:0 r:0 r: 1

6) Convierta la cantidad 0'6875 representada en base 10 a base 2.

SOLUCIÓN:

0'6875 x 2 = 1'375 0'375 x 2 = 0'75 0'75 x 2 = 1'5 0'5 x 2 = 1'0
$0'625_{10}$ = 0'1011

7) Convierta la cantidad 101101 representada en base 2 a base 10.

SOLUCIÓN:

$101101_2 = 1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 = 32 + 8 + 4 + 1 = 45_{10}$

8) Convierta la cantidad 0'1001101 representada en base 2 a base 10.

SOLUCIÓN:

$0'1001101_2 = 1x2^{-1} + 0x2^{-2} + 0x2^{-3} + 1x2^{-4} + 1x2^{-5} + 0x2^{-6} + 1x2^{-7} = 0{,}6015625_{10}$

9) Convierta la cantidad 0xa1d representada en base 16 a base 10.

SOLUCIÓN:

$0xA1D_{16} = Ax16^2 + 1x16^1 + Dx16^0 = 10x256 + 1x16 + 13x1 = 2589_{10}$

10) Convierta la cantidad 0x0'd1a representada en base 16 a base 10.

SOLUCIÓN:

$0x0'D1A_{16} = Dx16^{-1} + 1x16^{-2} + Ax16^{-3} = 13x0{,}0625 + 1x0{,}00390625 + 10x0{,}000244140625 =$
$0{,}81884765625_{10}$

11) Convierta la cantidad 1101101 representada en base 2 a base 16.

SOLUCIÓN:

$1101101_2 = \underline{0110}1101_2 = 6D_{16}$

12) Convierta la cantidad 0'1001101 representada en base 2 a base 16.

SOLUCIÓN:

$0'1001101_2 = 0'\underline{1001}1010_2 = 0'9A_{16}$

13) Convierta la cantidad 32'875 representada en base 10 a base 2.

SOLUCIÓN:

$32'865_{10}$ 32_{10} = (divisiones :2) = 100000_2 $32'865_{10} = 100000'111_2$
0'87510 = (multiplicaciones x2) = $0'111_2$

14) Convierta la cantidad 10110'10010101 representada en base 2 a base 10.

SOLUCIÓN:

$10110'10010101_2$	10110_2 = (PPB2) = 22_{10} 10010101_2 = (PPB2) = $0'58203125_{10}$	10110'10010101 = $22'58203125_{10}$

2 – Números enteros

1) ¿Cuál es el rango representable en signo y magnitud si utilizamos 8 bits? Indíquelo en base 2 y base 10.

SOLUCIÓN:

[11111111; 10000000; 00000000; 011111111] = [-127; -0; +0; +127]

2) Represente la cantidad $+96_{10}$ en signo y magnitud con 8 bits. Represéntelo en base 2.

SOLUCIÓN:

$+96_{10} = 1100000_{2\ 7bits}$? $+96_{10} = 01100000_{sm}$

3) Represente la cantidad -96_{10} en signo y magnitud con 8 bits. Represéntelo en base 2.

SOLUCIÓN:

$-96_{10} = 1100000_{2\ 7bits}$? $-96_{10} = 11100000_{sm}$

4) ¿Cuál es el rango representable en complemento a 2 si utilizamos 9 bits? Indíquelo en base 2 y base 10.

SOLUCIÓN:

[100000000; 000000000; 111111111] = [-256; 0; +255]

5) Represente la cantidad $+45_{10}$ en complemento a 2 con 8 bits. Represéntelo en base 2

SOLUCIÓN:

$+45_{10}$ Representamos el valor absoluto con 7 bits |+45| = $0101101_{2\ 7\ bits}$
Como es positivo, añadimos un cero, y ya está $+45_{10}$ = **0**0101101_{C2}

6) Represente la cantidad -101_{10} en complemento a 2 con 8 bits. Represéntelo en base 2.

SOLUCIÓN:

-101_{10} Representamos el valor absoluto con 7 bits |-101| = $1100101_{2\ 7\ bits}$
Como es negativo, le añadimos un cero, y le hacemos el complemento a 2:
0$1100101_2$ Ca2(01100101) = 10011011_2 101_{10} = 10011011_{C2}

7) ¿Cuál es el rango representable en Exceso 9 con 6 bits? Indíquelo en base 2 y base 10

SOLUCIÓN:

[-9; 54] = [000000; 111111]

8) Represente la cantidad $+14_{10}$ en Exceso 31 con 6 bits. Represéntelo en base 2.

SOLUCIÓN:

$+14_{10} = 001110_2$ $001110_2 + 011111_2 = 101101_{Z31} = +14_{10}$ | $011111_2 = 31$

9) Represente la cantidad -14_{10} en Exceso 31 con 6 bits. Represéntelo en base 2.

SOLUCIÓN:

$-14_{10} = -001110_2$ $-001110_2 + 011111_2 = 010001_{Z31} = -14_{10}$ | $011111_2 = 31$

3 – Operaciones con enteros

1) Dados los números $A=00110011_{C2}$ y $B=01110100_{C2}$ realice la operación A+B en complemento a dos, indicando si el resultado es correcto o se produce desbordamiento, justificándolo correctamente.

SOLUCIÓN:

```
          1 ---------------------- Penúltimo carry
A          00110011
B          01110100          1 xor 0 = 1 Hay desbordamiento y por lo tanto no hay resultado.
+         010100111
          |                   Último carry
```

2) Dados los números $A=10110011_{C2}$ y $B=01110100_{C2}$ realice la operación A+B en complemento a dos, indicando si el resultado es correcto o se produce desbordamiento, justificándolo correctamente.

SOLUCIÓN:

```
          1 ---------------------- Penúltimo carry
A          10110011
B          01110100          1 xor 1 = 0 No hay desbordamiento y el resultado es: 00100111 C2
+         100100111
          | carry
```

3) Dados los números A=10110011_{C2} y B=11110100_{C2} realice la operación A+B en complemento a dos, indicando si el resultado es correcto o se produce desbordamiento, justificándolo correctamente.

SOLUCIÓN:

```
           1 ---------------------- Penúltimo carry
A           10110011
B           11110100        1 xor 1 = 0 No hay desbordamiento y el resultado es: 10100111 C2
+          110100111
           |                 Último carry
```

4) Dados los números A=00110011_{C2} y B=11110100_{C2} realice la operación A-B (resta) en complemento a dos, indicando si el resultado es correcto o se produce desbordamiento, justificándolo correctamente.

SOLUCIÓN:

Como se pide una resta, se hará la suma A + (-B)

-B = Ca2(B) = Ca2(11110100) = 00001100_{C2}

```
           0 ---------------------- Penúltimo carry
A           00110011
(-B)        00001100        0 xor 0 = 0 No hay desbordamiento y el resultado es: 00111111 C2
+          000111111
           |                 último carry
```

5) Dados los números A=11110011_{C2} y B=11110100_{C2} realice la operación A+B en complemento a dos, indicando si el resultado es correcto o se produce desbordamiento, justificándolo correctamente.

SOLUCIÓN:

```
           1 ---------------------- Penúltimo carry
A           11110011
B           11110100        1 xor 1 = 0 No hay desbordamiento y el resultado es: 11100111 C2
+          111100111
           |                 último carry
```

6) Dados los números A=00000011_{C2} y B=00000100_{C2} realice la operación A+B en complemento a dos, indicando si el resultado es correcto o se produce desbordamiento, justificándolo correctamente.

SOLUCIÓN:

```
           0 ---------------------- Penúltimo carry
A           00000011
B           00000100        0 xor 0 = 0 No hay desbordamiento y el resultado es: 000000111 C2
+          000000111
           |                 Último carry
```

7) Dados los números $A=00110011_{C2}$ y $B=01110100_{C2}$ realice la operación A-B en complemento a dos, indicando si el resultado es correcto o se produce desbordamiento, justificándolo correctamente.

SOLUCIÓN:

Como se pide una resta, se hará la suma A + (-B)
-B = Ca2(B) = Ca2(01110100) = 10001100_{C2}

A	00110011	Suma de un valor positivo y otro negativo, no desborda
(-B)	10001100	0 xor 0 = 0 No Hay desbordamiento
+	10111111	

8) Dados los números $A=01100011_{C2}$ y $B=01000000_{C2}$ realice la operación A+B en complemento a dos, indicando si el resultado es correcto o se produce desbordamiento, justificándolo correctamente.

SOLUCIÓN:

	1 ----------------------- Penúltimo carry	
A	01100011	
B	01000000	1 xor 0 = 1 Hay desbordamiento y por lo tanto no hay resultado.
+	010100011	
	|	último carry

9) Dados los números $A=01100011_{C2}$ y $B=01000000_{C2}$ realice la operación A-B en complemento a dos, indicando si el resultado es correcto o se produce desbordamiento, justificándolo correctamente.

SOLUCIÓN:

Como se pide una resta, se hará la suma A + (-B)
-B = Ca2(B) = Ca2(01000000) = 11000000_{C2}

	1 ----------------------- Penúltimo carry	
A	01100011	
(-B)	11000000	1 xor 1 = 0 No hay desbordamiento y el resultado es: 00100011_{C2}
+	100100011	
	|	último carry

10) Dados A = 101001_{Z31} y B = 100110_{Z31} diga si es cierto que A es mayor que B, y cuántas unidades de diferencia hay entre ellos.

SOLUCIÓN:

Comparando bit a bit A y B:
A = $10\mathbf{1}001_{Z31}$
B = $10\mathbf{0}110_{Z31}$ A es mayor que B, y la diferencia es:
A = $10\mathbf{1}001_{Z31}$
-B = $\underline{10\mathbf{0}110}_{Z31}$
000011_2 = **3**

11) Dados A = 001001_{Z31} y B=011100_{Z31} diga si es cierto que A es mayor que B, y cuántas unidades de diferencia hay entre ellos.

SOLUCIÓN:

Comparando bit a bit A y B:
A = $0\mathbf{0}1001_{Z31}$
B = $0\mathbf{1}1100_{Z31}$ B es mayor que A, y la diferencia es:
B = $0\mathbf{1}1100_{Z31}$
-A = $\underline{0\mathbf{0}1001}_{Z31}$
010011_2 = **19**

4 – Coma flotante

1) Dado el número real +33'703125, represéntelo en el formato IEEE754 de simple precisión. Escriba el nombre y el tamaño de los campos. Muestre el resultado en binario y en hexadecimal.

SOLUCIÓN:

El formato de simple precisión de IEEE 754 es:

Signo (1 bit)	Exponente (8 bits)	magnitud (23 bits)

El campo Signo toma valor 1 si la cantidad representada es negativa, y toma valor 0 si es positiva.
El campo Exponente está representado en Exceso 127.
El campo magnitud es la mantisa, normalizada de la forma 1' y con la técnica del bit implícito.

El primer paso es convertir el número a representar a binario:

$+33'703125_{10} = +100001'101101 \times 2^0$

El segundo paso es normalizar la mantisa a la forma 1'x:

$+100001'101101 \times 2^0 = +1'00001101101 \times 2^5$

La magnitud, con 23 bits y con el bit implícito es: 00001101101000000000000

A continuación, representamos el exponente en Exceso 127:

$+5 = 00000101_2 \rightarrow 00000101_2 + 01111111_2 = 10000100_{z127} = +5$

Finalmente, representamos los diferentes campos de signo, exponente y magnitud (mantisa con el bit implícito)

0	10000100	00001101101000000000000

Agrupando los bit de cuatro en cuatro, la representación en hexadecimal es: 0x4206D000

2) Dado el número real -0,00030517578125, represéntelo en el formato IEEE754 de simple precisión. Escriba el nombre y el tamaño de los campos. Muestre el resultado en binario y en hexadecimal.

SOLUCIÓN:

El formato de simple precisión de IEEE 754 es:

Signo (1 bit)	Exponente (8 bits)	magnitud (23 bits)

El campo Signo toma valor 1 si la cantidad representada es negativa, y toma valor 0 si es positiva.
El campo Exponente está representado en Exceso 127.
El campo magnitud es la mantisa, normalizada de la forma 1' y con la técnica del bit implícito.

El primer paso es convertir el número a representar a binario:

$-0{,}00030517578125_{10}$ = $-0'00000000000101 \times 2^{0}$

El segundo paso es normalizar la mantisa a la forma 1'x:

$-0'00000000000101 \times 2^{0} = +1'01 \times 2^{-12}$

La magnitud, con 23 bits y con el bit implícito es: 01000000000000000000000

A continuación, representamos el exponente en Exceso 127:

$-12 = -00001100_2$ -> $-00001100_2 + 01111111_2 = 01110011_{Z127} = -12$

Finalmente, representamos los diferentes campos de signo, exponente y magnitud (mantisa con el bit implícito)

1	01110011	01000000000000000000000

Agrupando los bit de cuatro en cuatro, la representación en hexadecimal es: 0xB9A00000

3) La secuencia de dígitos en hexadecimal 42C90000 representa un número real codificado en el formato IEEE754 de simple precisión, ¿qué valor decimal se está representando?

SOLUCIÓN:

El primer paso es convertir el número a representar a binario:

$42C90000_{16}$= 0 10000101 $10010010000000000000000_2$

De acuerdo al formato IEE754 de simple precisión, los campos serán los siguientes:

0	10000101	10010010000000000000000

Signo = 0 (número positivo)

Exponente en exceso 127= 10000101_2=133_{10}

Exponente = 133_{10} – 127_{10} = 6_{10}

Mantisa= $1{,}10010010000000000000000_2$

Luego el número real será el siguiente:

$1{,}10010010000000000000000_2 * 2^6 = 1100100{,}1_2=$ $\mathbf{100{,}5_{10}}$

4) Represente el número real -520,8125 en el formato IEEE754 de simple precisión. Detalle todos los pasos realizados y exprese el resultado final en binario y en hexadecimal

SOLUCIÓN:

En primer lugar se convierte la cantidad a binario, por un lado la parte entera con divisiones sucesivas (o sabiendo que el número es igual a 512 + 8 = 2^9 + 2^3)

520_{10} = 1000001000_2

y por otro lado, la parte fraccionaria con multiplicaciones sucesivas

0,8125 x 2 = $\underline{1}$,625
0,625 x 2 = $\underline{1}$,25
0,25 x 2 = $\underline{0}$,5
0,5 x 2 = $\underline{1}$,0

por lo que -520,8125 = $-1000001000{,}1101_2$

Se reescribe en forma ±1,M x 2^E:
$-1000001000{,}1101_2= -1000001000{,}1101_2 \times 2^0 = -1{,}0000010001101_2 \times 2^9$

Expresamos el exponente en exceso 127:
Exponente = 9, expresado en exceso 127, se representa mediante binario (9 + 127) = binario (136) = 10001000

Campo S (signo): 1 (negativo)
Campo E (exponente): 10001000
Campo M (parte fraccionaria de la mantisa normalizada):

00000100011010000000000 (teniendo en cuenta que el bit de la parte entera es el bit implícito y no se almacena y rellenando hasta completar los 23 bits de este campo

Los 32 bits juntos en el orden S, E, M:
1 10001000 00000100011010000000000

En hexadecimal (reuniendo todos los bits y agrupando de cuatro en cuatro):
1100 0100 0000 0010 0011 0100 0000 0000 0xC4023400

5 – Cuestiones de Ampliación

Operaciones en Ca2

Edite, compile, y ejecute el siguiente código para hacer ejercicios de representación y operaciones en Ca2.
El código corresponde a lenguaje C. Para compilar en Linux, desde la consola de órdenes, teclee:
gcc -o enteros enteros.c
Para ejecutar, teclee: ./enteros
En otras plataformas, utilice un compilador de C y un proyecto de consola.

```
////enters.c

#include <stdio.h>
#include <stdlib.h>


void main (void)
 {
  signed char a, b, q, c, resul;
  int check;
  //El tipo char no es un carácter, es un entero de 8 bits con signo

  double f;

  srandom (time(NULL));
  for (q=0; q< 100; q++)
   {
    printf ("Ejercicio %d ", q);
    a = (char) (255.0 * random() / RAND_MAX);
    b = (char) (255.0 * random() / RAND_MAX);
    f = 2.0 * random() / RAND_MAX;
    if (f<1)
     {
      printf ("Realice la operación %d - %d ", a, b);
      resul = a - b;
      check = a - b;
     }
```

```
    else
     {
      printf ("Realice la operación %d + %d ", a, b);
      resul = a + b;
      check = a + b;
     }
    printf ("representando el operando en Ca2\n y haciendo la operación
en Ca2\n");
    printf ("Pulse INTRO para ver la solución\n");
    c = getchar();
    if (check == resul)
     printf ("Resultado en hexadecimal: 0x%x\n\n\n", resul);
    else printf ("Desbordamiento!!!\n\n\n");
   }
 }
```

Representación en IEEE754

Edite, compile, y ejecute el siguiente código para hacer ejercicios de representación en IEEE754.
El código corresponde a lenguaje C. Para compilar en Linux, desde la consola de órdenes, teclee:
gcc -o ieee754 ieee754.c
Para ejecutar, teclee: ./ieee754
En otras plataformas, utilice un compilador de C y un proyecto de consola.

```
////ieee754.c

#include <stdio.h>
#include <stdlib.h>


void main (void)
 {
  signed char q, c, p;
  int pot;
  float resul;

  srandom (time(NULL));
  for (q=0; q< 100; q++)
   {
    printf ("Ejercicio %d \n", q);
    resul = 0;
    pot = 2;
    for (c = 0; c < 8; c++)
     {
      p = (char) (2.0 * random() / RAND_MAX);
      if (p) resul = resul + 1.0/(pot);
      pot = pot << 1;
     }
    resul = resul + (char) ((64.0 * random() / RAND_MAX) - 32);
    printf ("Convierte el número %f al formato ieee754 de simple
precisión, expresándolo en hexadecimal\n",resul);
    printf ("Pulse INTRO para ver la solución \n");
    c = getchar();
    memcpy (&pot,&resul,4);
    printf (" 0x%8.0x\n\n", pot);
   } }
```