**PRACTICA 2: FUNDAMENTOS DE LOS COMPUTADORES**

**Ejercicio 1:**

**A) +159:** 15910=100111112 n=8 🡪 0010011111 en Complemento a 2

**B) +224:** 22410=111000002 n=8 🡪 0011100000 en Complemento a 2

**C) -99:** 9910=1100011**2** n=8 🡪 1110011100+1=1110011101 en Complemento a 2

**D) -187:** 18710=101110112 n=8 🡪 1101000100+1=1101000101 en Complemento a 2

**-A) -159:** 15910=100111112 n=8 🡪 1101100000+1=1101100001 en Complemento a 2

**A+B)**

0010011111 Resultado: 0101111111

+ 0011100000

0101111111

**C-A)**

1110011101 Resultado: 1011111110

+ 1101100001

1011111110

**B+D)**

0011100000 Resultado: 0000100101

+ 1101000101

0000100101

**D-A)**

1101000101 Resultado: 1010100110

+ 1101100001

1010100110

**Ejercicio 2:**

**Rango de representación:**

-[(2n-2-p),2-p]<=x<=[2-p,(2n-2-p)]

**Identifica los números siguientes:**

**i) 0 1011110000 00000:**

0 🡪 Signo positivo

10111100002 🡪 75210 Resultado: 752.010

000002 🡪 010

**ii) 1 1010100001 10000:**

1 🡪 Signo negativo

10101000012 🡪 67310  Resultado: -673.5

100002 🡪 0.510

**iii) 1 010110110 00110:**

1 🡪 Signo negativo

0101101102 🡪 36410  Resultado: -364.187510

001102 🡪 0.1875010

**Ejercicio 3:**

**a) 118.75:**

Signo positivo 🡪 0

118 🡪 1110110 Resultado: 0 1110110 110

0.75 🡪 1100

**b) -71.25:**

Signo negativo 🡪 1

71 🡪 1000111 Resultado: 1 100011 01

0.25 🡪 01

**c) 0.15:**

Signo positivo 🡪 0

0 🡪 00000000 Resultado: 0 0 0010011

0.15 🡪 0010011

**d) -11.1:**

Signo negativo 🡪 1

11 🡪 1011 Resultado: 1 1011 0001

0.1 🡪 0001

**Ejercicio 4:**

**a) 112.415:**

Signo positivo 🡪 0

112 🡪 1110000

0.415 🡪 0.0110101000

112.415 🡪 1110000.0110101000

Aplicando la normalización:

1.1100000110101 x 26

6+127=133 🡪 10000101

Resultado: 0 10000101 11000001101010000000000

**b) -2.76 x 102:**

Signo negativo 🡪 1

276 🡪 100010100

Aplicando la normalización:

1.00010100 x 28

8+127=135 🡪 10000111

Resultado: 1 10000111 00010100000000000000000

**c) 1020:**

Signo positivo 🡪 0

2exp=1020

exp=(20 x log10)/(log2) 🡪 exp=66.4385619

266 x 20.4385619

66+127=193 🡪 11000001

1.355252718 🡪 1.01011010111100011101011

Resultado: 0 11000001 01011010111100011101011

**d) 2-42:**

Signo positivo 🡪 0

-42+127=85 🡪 01010101

En este caso, la mantisa es todo 0 porque el numero es 1.0 x 2-42­

Resultado: 0 01010101 00000000000000000000

**Ejercicio 5:**

**a) 1 01110001 1011100100000000000000:**

Signo negativo 🡪 1

113-127=-14

-1.10111001 x 2-14

1.10111001 🡪 0.72265625

M=1.72265625

-1.72265625 x 2-14 = -0.00010514259

Resultado: -1.051425934 x 10-4

**b) 0 11010010 1000111011000000000000:**

210-127=83

1.1000111011 🡪 1.5576171875

1.5576171875 x 283 = 15.06434908 x 1024

Resultado: 1.506434908 x 1025

**c) 0 00000000 00000000000000010111001:**

Signo positivo 🡪 0

0-126=-126

M=0.0000220537

0.0000220537 x 2-127

Este número esta desnormalizado.

**d) 0 11111111 001100000000000000000000:**

Como la mantisa es M!=0 y el exponente E=1, podemos decir que es un caso especial y que se define como NaN (Not a Number)

**e) 1 11111111 000000000000000000000:**

Como la mantisa M=0 y el exponente E=1, podemos decir que es un caso especial y que se define como infinito. Para saber el signo, deberemos mirar el bit de signo. Al ser un 1, podemos afirmar que ese numero es el -∞

**Ejercicio 6:**

**a)**

Para obtener el código de Hamming, usaremos la siguiente ecuación:

2p≥1+p+n donde n es el número de bits.

Despejando y sustituyendo, tenemos como resultado que p=3

**b)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **P1** | **P2** | **D1** | **P3** | **D2** | **D3** | **D4** |
| **Recibida** | **1** | **0** | **1** | **0** | **1** | **1** | **0** |
| **P1** | **0** |  | **1** |  | **1** |  | **0** |
| **P2** |  | **0** | **1** |  |  | **1** | **0** |
| **P3** |  |  |  | **0** | **1** | **1** | **0** |

Como podemos observar, la cadena no es correcta. El bit erróneo es el primero. La cadena corregida sería 0010110.

**Ejercicio 7:**

**a) 051F:**

051F=10100011111

10100011111 / 110001

110001 110001

0110011

110001

0000101111

110001

011110

10100011111

- 11110

10100000001 / 110001

110001 110001

0110010

110001

0000110001

110001

000000

**b) Recibiendo la cadena 11000001110101:**

11000001110101 / 110001

110001 100001101

00000101110

110001

0111111

110001

00111001

110001

001000

11000001111101 / 110001

110001 100001101

00000101111

110001

0111101

110001

00110001

110001

000000

Como el resto de la división nos ha dado 0, podemos afirmar que no ha habido ningún error en la transmisión del mensaje.