Ejercicio 1:
Convertir los siguientes números en hexadecimal a binario de 32 bits:

a) 0xABCDEF00 b) 0x123456 c) 0x8E3FC581 d) 0x10A6F2B

a) 0xABCDEF00 = 1010 1011 1100 1101 1111 0000 0000

b) 0x123456 = 0x00123456 = 0000 0000 0001 0010 0011 0100 0101 0110

c) 0x8E3FC581 = 1000 1110 0011 1111 1100 0101 1000 0001

d) 0x10A6F2B = 0000 0001 0000 1010 0110 1111 0010 1011

Ejercicio 2:
Convertir los siguientes números en binario a decimal y a hexadecimal:

Binario	Decimal	Hexadecimal
$(1110011110000011)_2$	59267	0xE783
$(101101110011010010101111)_2$	6003247	0x5B9A2F
$(10110011011011100001000)_2$	11483.7578125	0x2CDB.C20
$(10001111110100011111.000001101)_2$	589087.02539625	0x8FD1F.068

$14.16^3+7.16^2+8.16+3$
$= 14.4096+7.256+8.16+3$
$= 57344+1792+128+3$

57344
1792
128
3
59267

0x5B9A2F = $5.16^5+11.16^4+9.16^3+10.16^2+2.16+15$
$= 5242880+720896+36864+2560+32+15$
$= 6003247$

Ejercicio 3:
Suponiendo que se tienen registros de 16 bits, convertir a binario **sin** signo los siguientes números en base 10:

a) 123_{10} b) 59_{10} c) $255,46_{10}$ d) $98,019_{10}$

a) 123_{10} =
$123 = 2.61+1$
$61 = 2.30+1$
$30 = 2.15$
$15 = 2.7+1$
$7 = 2.3+1$
$3 = 2.1+1$
$1 = 2.0+1$
$\Rightarrow 1111011$

b) $59 = 2.29+1$
$29 = 2.14+1$
$14 = 2.7+0$
$7 = 2.3+1$
$3 = 2.1+1$
$1 = 2.0+1$
$\Rightarrow 59_{10} = 111011_2$

c) $255,46_{10} \Rightarrow$
$255 = 2.127+1 \Rightarrow 1111\ 1111$
$127 = 2.63+1$
$63 = 2.31+1$
$31 = 2.15+1$
$15 = 2.7+1$
$7 = 2.3+1$
$3 = 2.1+1$
$1 = 2.0+1$

$0,46.2 = 0,92$
$0,92.2 = 1,84$
$0,84.2 = 1,68$
$0,68.2 = 1,36$
$0,36.2 = 0,72$
$0,72.2 = 1,44$
$0,44.2 = 0,88$
$0,88.2 = 1,76$
$0,76.2 = 1,52$
$0,52.2 = 1,04$
$0,04.2 = 0,08$
$\approx .01110111110$

Notar que tenemos la restriccion de 16 bits por lo tanto solo usaremos los 16 más significantes, que en este caso corresponden a la parte entera del número.

d) $98{,}019_{10}$ ⇒ $98 = 2 \cdot 49 + 0$ ⇒ 110 0010.0 Pues ya ocupé así los 16 bits

$49 = 2 \cdot 24 + 1$
$24 = 2 \cdot 12 + 0$
$12 = 2 \cdot 6 + 0$
$6 = 2 \cdot 3 + 0$
$3 = 2 \cdot 1 + 1$
$1 = 2 \cdot 0 + 1$

0,19.2 = 0.38
,38.2 = 0.76

Ejercicio 4:

Suponiendo que un microprocesador utiliza registros de 8 bits y representación de números negativos en complemento a 2, muestre el contenido de estos registros al codificar en binario los siguientes números **con** signo:

a) -76_{10} b) -43_{10} c) $+64_{10}$ d) -121_{10}

a) $76 = 2 \cdot 38 + 0$ ⇒ 1001100 = 76
$38 = 2 \cdot 19 + 0$
$19 = 2 \cdot 9 + 1$
$9 = 2 \cdot 4 + 1$
$4 = 2 \cdot 2 + 0$
$2 = 2 \cdot 1 + 0$
$1 = 2 \cdot 0 + 1$

⇒ $-76 = -1001100$
$= 01001100$

⇒ $-76 = 10110011 + 1$
10110100

b) -43_{10} ⇒ $43 = 2 \cdot 21 + 1$ ⇒ 43 = 101011
$21 = 2 \cdot 10 + 1$
$10 = 2 \cdot 5 + 0$
$5 = 2 \cdot 2 + 1$
$2 = 2 \cdot 1 + 0$
$1 = 2 \cdot 0 + 1$

$-43_{10} = -00101011$
$= 43_{16} =$ 11010101

010100
1
010101

c) 64_{10} ⇒

$64 = 2 \cdot 32 + 0$ ⇒ $64_{10} =$ 01000000
$32 = 2 \cdot 16 + 0$
$16 = 2 \cdot 8 + 0$
$8 = 2 \cdot 4 + 0$
$4 = 2 \cdot 2 + 0$
$2 = 2 \cdot 1 + 0$
$1 = 2 \cdot 0 + 1$

d) -121_{10} ⇒ $121 = 2 \cdot 60 + 1$
$60 = 2 \cdot 30 + 0$
$30 = 2 \cdot 15 + 0$
$15 = 2 \cdot 7 + 1$
$7 = 2 \cdot 3 + 1$
$3 = 2 \cdot 1 + 1$
$1 = 2 \cdot 0 + 1$

01111001
⇒ 10000110
10000111 = -121_{10}

Ejercicio 5:

Convertir los siguientes valores binarios de 8 bits en formato de complemento a dos a decimal:

a) 10010110 b) 11111011 c) 11100000 d) 00011110

a) 10010110
⇒ +1 = 10010110 = 2+4+16+128 ? = 22 − 128 = $(-106)_{10}$

b) 11111011 ⇒ 00000100 + 1 = 00000101 = $(-5)_{10}$

c) 11100000 ⇒ 00011111 + 1 = 00100000 = (−32)

d) $00011110 = (2+4+8+16)_{10} = (30)_{10}$

Ejercicio 6:

Suponga que los registros A y B del microprocesador del ejercicio 4 (registros de 8 bits) contienen los valores 0x80 y 0xD0 respectivamente.

a) ¿Qué valor contiene el registro C después de ejecutar la operación C = A + B? ¿El resultado que se guarda en C es el esperado?

b) ¿Qué valor contiene el registro C después de ejecutar la operación C = A - B? ¿El resultado que se guarda en C es el esperado?

c) En base al análisis de las operaciones anteriores, ¿cuál es la ventaja de la representación de números negativos mediante su complemento a 2, por sobre la representación binaria regular + un bit de signo?

a) $C = A + B$

$\Rightarrow$ $1000\,0000 + 1101\,0000 = 1\,0101\,0000$

Obtenido: 0x50

esperado: 0x150

En C se guarda 0x50 pues sólo almacena 8 bits.

128 + 208 = 336

b) $C = A - B = A + (-B)$

$B = 11010000 \Rightarrow -B = 00101111 + 1 = 00110000$

$\Rightarrow$ $1000\,0000 + 0011\,0000 = 1011\,0000 = 0xB0$

El resultado obtenido es el esperado.

c) *La ventaja es la facilidad que crea a la hora de realizar operaciónnes aritméticas, pues nos libera de la necesidad de precuparnos del signo y transformar todo en una suma.*

Ejercicio 7:

Expresar los siguientes números decimales en su representación binaria (negativos en complemento a 2) considerando los tamaños de los registros donde serán alojados según la tabla.

Decimal	Binario		
	Byte	HalfWord	Word
113	0111 0001	0X0071	0x00000071
-63	11 000001	0xFFC1	0xFFFFFFC1
319	0011 1111	0x013F	0x0000013F
-128	1000 0000	0xFF80	0xFFFF FF80
65535	111	0xFFFF	0x0000 FFFF
-149744	0001 0000	0xB710	0xFFFD B710

$113 = 2.56+1 = 2^7 + 2^6 + 2^5 + 2^4 = 01110001$

$-63 \Rightarrow 63 = 2.31+1 \Rightarrow 11\,000000 + 1 = 11\,000001$

$319 = 0100111111$

$-128 = 1\,1000\,0000$

Ejercicio 8:

Convertir los siguientes números decimales a formato IEEE 754 de precisión simple (normalizados):

a) 5678_{10} b) $306{,}59375_{10}$ c) $723{,}125_{10}$
d) $18{,}1953125_{10}$ e) -3020.993_{10} f) -0.000892_{10}

IEEE 754 FLOATING-POINT STANDARD ④

$(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent} - \text{Bias})}$
where Single Precision Bias = 127,
Double Precision Bias = 1023

IEEE 754 Symbols

Exponent	Fraction	Object
0	0	± 0
0	≠ 0	± Denorm
1 to MAX - 1	anything	± F1. Pt. Num.
MAX	0	± ∞
MAX	≠ 0	NaN

IEEE Single Precision and Double Precision Formats: S.P. MAX = 255, D.P. MAX = 2047

S	Exponent	Fraction
31	30 23	22 0

S	Exponent	Fraction
63	62 52	51 0

a) 5678_{10}

$5678 = 0001\ 0110\ 0010\ 1110$

$S = 0$ $\quad 12 + 127 = 139$

$139 = 1000\ 1011$

0 1000 1011 011 0001 0111 0000 0000 0000

b) $306{,}59375_{10} = 100110010.10011$

$S = 0$ $\quad E = 8 + 127 = 135 = 1000\ 0111$

0 10000111 001 1001 0100 1100 0000 0000

c) $-3020.993 = -101111001100.111111100011010101$

$S = 1$ $\quad E = 11 + 127 = 138 = 1000\ 1010$

1 1000 1010 0111 1001 1001 1111 1100 011

Ejercicio 9:

Convertir los siguientes en formato IEEE 754 de precisión simple (normalizados) a números decimales:

a) 1 1 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 b
b) 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 1 0 0 0 0 0 b
c) 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 b
d) 0 0 1 1 1 0 0 1 0 0 1 0 0 1 1 0 1 0 1 1 1 0 0 1 0 0 1 1 1 1 0 1 b
e) 1 0 1 1 1 0 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0 0 0 0 1 0 1 0 1 0 0 1 b
f) 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 b

a) 1 | 1 0 0 0 1 0 1 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 b

S = 1

E = 10001011 = 128 + 8 + 2 + 1 = 139 ⇒ E = 139 − 127 = 12

F: 1,000000000000 11000000000 · 2^{12} = 1000000000000,1100000000

∴ 4096,75

0,11 = $2^{-1} + 2^{-2} = \frac{1}{2} + \frac{1}{4}$ = 0,5 + 0,25 = 0,75

El resto de ejercicios son mecánicamente iguales.

Ejercicio 10:

Investigar cómo se escriben los símbolos especiales (“NaN”, “+Infinito”, “-Infinito”, “+0”, “-0”) en formato IEEE 754 de precisión simple (normalizados).

Zero:

The number zero is represented specially:
sign = 0 for positive zero, 1 for negative zero.
biased exponent = 0.
fraction = 0.

Positive and negative infinity:

Positive and negative infinity are represented thus:
sign = 0 for positive infinity, 1 for negative infinity.
biased exponent = all 1 bits.
fraction = all 0 bits.

NaN:

Some operations of floating-point arithmetic are invalid, such as taking the square root of a negative number.The act of reaching an invalid result is called a floating-point exception.An exceptional result is represented by a special code called a NaN, for "Not a Number". All NaNs in IEEE 754-1985 have this format:

sign = either 0 or 1.
biased exponent = all 1 bits.
fraction = anything except all 0 bits (since all 0 bits represents infinity).