Relación de binario .Trabajo Realizado por Rafael Muñoz y Laura Calvente.

1. Realiza la conversión de estas representaciones binarias de **8 bits** a la base numérica que se indica.

A base decimal:

01011011 (signo-magnitud)

El primer bit es el que nos determina el signo del número al ser un 0 es positivo y para operar el numero lo dejamos aparte y pasamos el número que nos queda directamente a base decimal.

1011011=1·(2^6)+ 1·(2^4)+ 1·(2^3)+ 1·(2^1)+ 1·(2^0)= 91 en base decimal

10001101 (Ca1)

Combinamos ceros por unos para obtener el número original, como el bit primero es 1 el número es negativo.

01110010= 1·(2^6)+ 1·(2^5)+ 1·(2^4)+ 1·(2^2)= -114(le añadimos el negativo porque el número que buscamos es negativo.) en base 10.

11001011 (Ca2)

Como el Ca2 es la representación negativa de un número, realizamos la acción inversa para obtener el número original. Al Ca2 le restamos el Ca2 de 1 para obtener así el Ca1 del número original, y luego cambiamos ceros por unos.

```
1=00000001 → Ca1 de 1= 11111110
                      +00000001
                       --------------
                       11111111   Este es el Ca2 de 1
```

```
  11001011
+ 11111111
-----------------
```

1/11001010= Ca1 (el número que esta apartado por una barra y resaltado en azul no lo incluimos pues es un desbordamiento.) →00110101= numero original=1·(2^5)+ 1·(2^4)+ 1·(2^2)+ 1·(2^0)= -53 (le añadimos el negativo porque el número que buscamos es negativo.) en base 10.

A base hexadecimal:

10011001 (Sesgada)

En la representación sesgada, el exceso que se suma al número es de 2(^(n-1)) donde n es = número de bits del registro, en este cano n=8. Por tanto el exceso es 2^(8-1)=2^7=128.

Como el número viene dado por representación sesgada, tendremos que restar el exceso para obtener el número original.

Para poder restar, tenemos que convertir 128 en negativo (-). Para ello lo pasamos a Ca2.

Pasamos el número a binario dividiéndolo constantemente entre dos obteniendo restos que son 0 o 1 y luego partiendo desde el ultimo cociente apuntamos nuestro número que serán la lista seguida de los restos de derecha a izquierda.

128 en base 10= 1000000 (lo pasamos Ca1) → 01111111
+10000000

-128 en base 10=1 100000000 → Ca2

10011001
+10000000

1/00011001 = número original =1·(2^4)+ 1·(2^3)+ 1·(2^0)=25

Convertimos el resultado de binario a hexadecimal, separando en grupos de 4 bits, y hacemos la conversión directamente

0001 → 1
1001 → 9

El número en hexadecimal es 19.

HECHO EN CLASE
10011001 → 128+16+8+1→153-128=25 en base 10 = 19 en base hexadecimal.

10011111 (Ca2)

Obtenemos el número original, sumando el valor negativo de 1 para conseguir el Ca1 y luego cambiamos ceros por unos.

-1 en base 10=11111111

10011111
+111111111

1/10011110 → (cambiamos ceros por unos) 01100001= número original.

Para transformarlo en hexadecimal separamos en grupos de 4 en 4 y lo convertimos directamente
O110→6
0001→1
El número en base hexadecimal es 61.

2. Realiza estas operaciones aritméticas en base binaria con registros de 8 bits. Indica que representación utilizas.

0xA2 + 25 en base10

A2=10100010
25=00011001

```
  10100010
+ 00011001
 ----------------
  10111011 es el resultado. Utilizamos representación en signo-magnitud.
```

125 en base 10 – 12 en base 10

125=01111101
12=00001100 →convertimos 12 a valor negativo pasándolo a Ca2 →

```
0001100 → Ca1=11110011
              +00000001
             ----------------
               11110100 → ya tenemos el número en Ca2, ahora debemos realizar la suma aritmética
```

utilizando la representación signo-magnitud.

```
   011111001
+   11110100
--------------------
1/01110001 → el primer digito es 0 por lo que el número es positivo, este es el resultado final.
```

90 de base 10 - 0x12

90=01011010
12=10010 → lo convertimos a Ca2, para ello primero lo convertimos a Ca1=01101 y ahora le sumamos uno para obtener el Ca2

```
  00010010 (le añadimos ceros a la izquierda porque estamos trabajando en 8 bits)
+ 00000001
 ---------------
  00010011 →número en Ca2 ahora hacemos la suma aritmética en signo-magnitud
```

```
  01011010
+  00010011
 ----------------
   01101101 → como el primer digito es 0 el número es positivo. Este es el resultado definitivo.
```

-0x60 + 37 en base 10

60=1100000 → pasamos el número a complemento Ca2, para ello primero realizamos el Ca1=0011111 y le sumamos 1 →

```
  0011111
+ 0000001
  ------------
  0100000 → ya tengo el Ca2
```

37=100101

Ahora realizamos la suma aritmética de los dos números en signo magnitud.

```
  00100101 (le ponemos 0 a la izquierda hasta obtener 8 dígitos porque estamos trabajando en 8 bits
+ 00100000
  ---------------
  01000101 → como el primer digito es 0 el número es positivo. Este es el resultado final.
```

3. Realiza estas operaciones aritméticas en base binaria con registros de 8 bits. Utiliza la representación en valor absoluto. Usamos para ambas la representación de valor absoluto

24 en base 10 x 8 en base 10

24=00011000 en decimal
8=00001000=2^3

```
00011000
X       2^3
-------------
11000000 es el resultado.
```

112 en base 10 : 4 en base 10

112=01110000
4=0000011100=2^2

```
01110000
X       2^2
 -------------
00011100 es el resultado.
```

4. Halla la expresión lógica correspondiente a este enunciado y obtén su tabla de verdad.

"Si no sabes realizar el problema de coma flotante, y tampoco tienes muchos negativos, no podrás aprobar las prácticas del primer trimestre de SI"

Saber realizar problemas= P
Tener muchos negativos = N
Aprobar= A
Expresión lógica= ˜A=˜P^˜N

P	N	˜A=˜P^˜N
0	0	0
0	1	1
1	0	1
1	1	1

5. Realiza estas operaciones lógicas en base binaria con registros de 8 bits.

117 en base 10 v 0x42

117 en base 10=01110101
42 en hexadecimal=1000010

  01110101
V 01000010

   01110111 es el resultado.

0x33 (circulo con un más)131 en base 10

33 en hexadecimal=00110011
131 en base 10 = 10000011

00110011
(Signo exclusión) 10000011

10110000 este es el resultado.

6. Convierte estos valores a la representación binaria coma-flotante de 32 bits.

0xA1996

A1996= 1010 0001 1001 1001 0110= 1´010 0001 1001 1001 0110·(2^19)
S=0 porque el número es positivo
Exponente= 19 +128= 147 en base 10 =10010011
M=01000011001100101100000 (no la pasamos a Ca1 porque es positiva) →
Por tanto el número sería

s	exponente	mantisa
0	10010011	01000011001100101100000

-0,03410

0´034 en base 10=0000,000010001000000100000110=1,0001000000100000110·(2^(-5))
S=1 porque es un número negativo.
Exponente= -5 +128 = 123 en base 10 = 01111011
M=00010000001000001100000 → lo pasamos a Ca1 y entontes m= 11101111110111110011111
Por tanto el número quedaría de la siguiente manera:

s	exponente	mantisa
1	01111011	11101111110111110011111

7. Tenemos un archivo de texto plano de 125 KiB en el sistema de codificación UTF-8 utilizando caracteres unicode puros ¿Cuántos caracteres contendrá?

Un KIB son 1024 bits, por tanto para saber cuántos bits hay en 125 KIB solo tenemos que hacer la regla de tres y nos saldrán 128000 bits. Como sabemos que en Unicode un carácter se almacena cada dos bits solo lo tenemos que dividir entre dos para obtener el número de caracteres que tendrá el archivo, resultando 64000.

8. Calcula los límites de una representación binaria de 12 bits en signo magnitud indicando como se representa el cero, el valor máximo y valor mínimo de la representación.

n=12 bits

nº de representación 4096

rango= (-2047 ≤ X ≤ 2047)

0 en binario ---> hay que poner las dos opciones positivo y negativo → 000000000000 y 100000000000. Estos serían los valores máximos y mínimos.