1) Dados los números binarios A =00001100 y B =10111100, responder a las siguientes cuestiones:
   - Considerando que ambos números están codificados en signo-magnitud, hallar A+B y A-B. (0,5 puntos)

A = 12; B=-60

A+B = -48 = 100110000

A-B = 72 = 011000000

   - Independientemente de su codificación, hallar el Ca2 (complemento a 2) de ambos números. (0,5 puntos)

Ca2(A) = 11110100

Ca2(B) = 01000100

   - Considerando que ambos números están codificados en Ca2 (complemento a 2), hallar A+B y A-B. Indicar si hay desbordamiento. (0,5 puntos)

A=12; Ca2(B) = 01000100 -> B= -68

A + B = 00001100

10111100

11001000 -> Ca2= 00111000 -> 56 -> A-B =-56

No hay desbordamiento

==========

2) (1.5 puntos) Dados los números decimales 364 y 112:

- (0,5 puntos) Convertir ambos números a binario, expresándolos con 10 bits.

  364=; 112=00 0111 0000

- (0,5 puntos) Expresar el número (-112) en complemento a 2.

  Ca2(112) = 11 1001 0000

- (0,5 puntos) Realizar la operación (364 – 112) en complemento a 2.

  0101101100
  1110010000
  ~~1~~0 011111100 No hay overflow. El acarreo se ignora en Ca2

==========

Ca2 Casos posibles. Ejemplos

1º caso: resultado positivo correcto.

```
7-5=+7+(-5)          0111      +7
                     1011 +    -5 +
                     ------    ------
                     0010      +2
```

2º caso: resultado negativo correcto.

```
5-7=+5+(-7)          0101      +5
                     1001 +    -7 +
                     ------    ----
                     1110
                               -2
```

3º caso: resultado positivo con desbordamiento (OVERFLOW).

```
7+5=12               0111      +7
                     0101 +    +5
                     ------
                     ~~1~~100      FALSO, bit de signo -
```

4º Caso: resultado negativo con desbordamiento (OVERFLOW).

```
-7-5 = -7+(-5)          1001        -7
                        1011 +      -5
                        ------
                        ~~0~~100        FALSO, bit de signo +
```

Si usamos 1 bit más no se produce desbordamiento

```
7+5=12                   00111          +7
                         00101 +        +5 +
                        -------        -----
                         01100          +12 CORRECTO

-7-5 = -7+(-5)           11001
                         11011 +        -7
                        -------         -5 +
                        ~~1~~10100      -----
                                        -12 CORRECTO
```

¿Cómo saber si se ha producido desbordamiento en una operación en complemento a 2?

Viendo si los dos últimos acarreos son iguales. Si son iguales, no se ha producido desbordamiento. Si son distintos, sí se ha producido desbordamiento.

Ejemplo 1:

```
                        11110 ACARREOS
7-5=+7+(-5)              0111        +7
                         1011 +      -5 +
                        -------     -------
                        ~~1~~0010       +2 CORRECTO
```

Ejemplo 2:

```
                        01110 ACARREOS
7+5=12                   0111        +7
                         0101 +      +5 +
                        -------     -------
                        ~~0~~1100       -4 INCORRECTO
```

Operación	Nº Dec	Num C2 bin	
	47	00101111	
+	32	00100000	
	79	01001111	Acarreo pero no overflow

Operación	Nº Dec	Num C2 bin	
	105	01101001	
+	43	00101011	
		10010100	
	148	carry1-1-11	Overflow cambió el signo
			con 8 bits represento de -128 a 127

Operación	Nº Dec	Num C2 bin	
	-54	11001010	11001010 (es 54 complementado)
+	20	00010100	
	-34	11011110	no hay ni acarreos ni overflow

Operación	Nº Dec	Num C2 bin	
	-98	10011110	
+	50	00110010	
	-48	11010000	Hay acarreo

Operación	Nº Dec	Num C2 bin	
	100	01100100	
+	-27	11100101	
			hay acarreo y no overflow el uno de acarreo
	73	101001001	se quita

Operación	Nº Dec	Num C2 bin	
	-45	11010011	
+	-5	11111011	
	-50	101001110	hay acarreo y overflow cambia de signo
			11001110 =-50 en compemento a 2
			es 78 en decimal que es congruente con -50 en módulo 2
		01001110	128 -50 = 78

Complemento a 1.

La operación en complemento a 1 es igual que en complemento a 2 con la única diferencia de que además de sumar los dos operandos, se suma el acarreo que resulte de la operación como si fuera el LSB de un tercer operando.

Ejemplo 1:

```
00000 ACARREOS
 0110        +6
 0001 +      +1
 ------
 00111
    0 +
 ------      ------
 ~~0~~0111       +7
```

Ejemplo 2:

```
11000 ACARREOS
 1101        -2
 1110  +     -1
 ------
 11011
    1 +
 ------      ------
 ~~1~~1100       -3
```

Punto flotante

Convertir el número C19E0000 en formato IEEE754 en su equivalente decinmal:

Convertimos a binario

C19E0000 = 1100 0001 1001 1110 0000 0000 0000 0000

signo (1)
exponente (10000011)
mantisa (001 1110 0000 0000 0000 0000)

la formula es

(-1)^s * 1,mantisa * 2^(e-127)

s (1) significa, por tanto, signo negativo

exponente 10000011 es 131, como se representa en exceso a 2^(n-1)-1 hay que restar 127 para volver a tener el exponente real, es por tanto 4.

mantisa (hacia la derecha se van multiplicando los bits por 2^-1, 2^-2, 2^-3,2^-4, ...)

0*0,5
0*0,25
1*0,125
1*0,0625
1*0,03125
1*0,015625

0,234375 en decimal

como está normalizada (se supone un 1 a la izquierda de la coma) es en realidad 1,234375. Todo junto:

(-1)^1* 1,234375 * 2^4 = -1,234375*16 = -19,75

Calcular el valor de 11000110101101000000000000000000, sabiendo que sigue la representación IEEE 754

El signo es negativo porque el bit de mayor peso es 1

El valor nominal del exponente 10001101 teniendo en cuenta la posición de cada dígito es141

Tenems en cuenta que para m=8, sesgo=$2^{m-1}-1=127$

El valor del exponente es 141−127=14141−127=14.

Los dígitos almacenados de la mantisa son 01101000000000000000000 y teniendo en cuenta el bit implícito, que es uno, la mantisa es 1,01101000000000000000000

Número

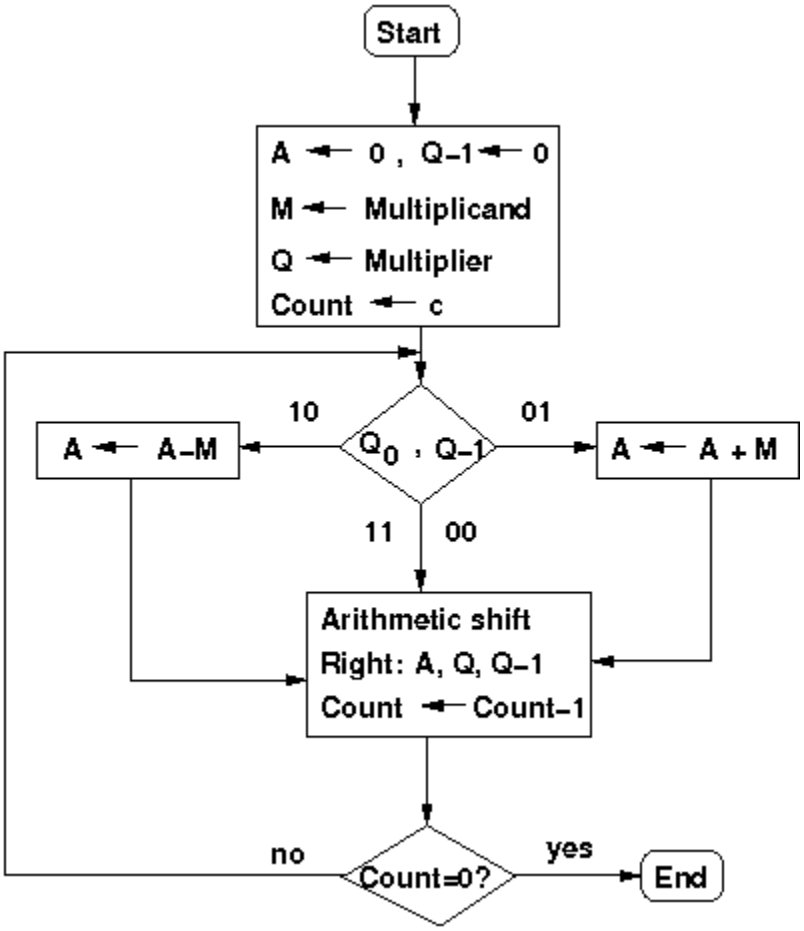
Por lo tanto, si escribimos el número este es
$-1,01101\times2^{14}\longrightarrow-(1+2^{-2}+2^{-3}+2^{-5})\times2^{14}=\boxed{-23040}$

Representar el número -24,50 utilizando el estándar de coma flotante de simple precisión IEEE 754. Expresar dicha representación en binario y en hexadecimal

$24,5 = 11000.1 = 1,10001 \times 2^4$

Signo = 1, número negative
Exponente = 4 + 127 = 131 = 10000011
Mantisa = 1000100000 .. 00000
En binario es: 11000000111000100000 …….00000
En Hexadecimal es: C1C40000

Multiplicador de Booth



A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial values	
1001	0011	0	0111	A ← A − M	First cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second cycle
0101	0100	1	0111	A ← A + M	Third cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth cycle

Figure 9.13 Example of Booth's Algorithm (7 × 3)

Booth's Algorithm Work out Example (-12 x -11 =132)

Comments	SC	Multiplicand M	Product A	Product Q	Q_e
Initial (-12 x -11)	0	10100	00000	10101	0
Q_0Q_e = 10; P←P-M	0	10100	01100	10101	0
Ar.sh.r . PQ Q_e	0	10100	00110	01010	1
SC ← SC+1	1	10100	00110	01010	1
Q_0Q_e = 01; P←P+M	1	10100	11010	01010	1
Ar.sh.r . PQ Q_e	1	10100	11101	00101	0
SC ← SC+1	2	10100	11101	00101	0
Q_0Q_e = 10; P←P-M	2	10100	01001	00101	0
Ar.sh.r . PQ Q_e	2	10100	00100	10010	1
SC ← SC+1	3	10100	00100	10010	1
Q_0Q_e = 01;	3	10100	11000	10010	1
Ar.sh.r . PQ Q_e	3	10100	11100	01001	0
SC ← SC+1	4	10100	11100	01001	0
Q_0Q_e = 10; P←P-M	4	10100	01000	01001	0
Ar.sh.r . PQ Q_e	4	10100	00100	00100	1
SC ← SC+1	5	10100	**00100**	**00100**	1

-12 x -11 = 132