I formati Floating Point e conversione di numeri frazionari in binario

Esercizio 1

Convertire il numero 12.125 in binario.

Parte intera

La conversione è quella solita.

```
12  0  12:2=6  (resto 0)
 6  0   6:2=3  (resto 0)
 3  1   3:2=1  (resto 1)
 1  1   1:2=0  (resto 1)
 0
```

12 (base 10) --> **1100** (base 2)

Parte decimale

Si procede con moltiplicazioni successive.

```
0.125 * 2 = 0.250 (riporto 0)
0.250 * 2 = 0.500 (riporto 0)
0.500 * 2 = 1.000 (riporto 1) --> si considera solo
più la parte decimale
0.000 * 2 = FINE
```

Ordinando i riporti al contrario, si ottiene il risultato.

0.125 (base 10) --> **0.001** (base 2)

Il risultato sarà: 12.125 (base 10) --> **1100. 001** (base 2)

Esercizio 2

Convertire il numero 17.55 in binario (precisione 8 cifre decimali, arrotondamento per troncamento).

Parte intera: si usano i metodi già visti. 17 (base 10) --> 10001 (base 2)

Parte decimale: si usano le moltiplicazioni successive.

```
0.55 * 2 = 1.10 (riporto 1) --> si considera solo più "0.10"
0.10 * 2 = 0.20 (riporto 0)
```

```
0.20 * 2 = 0.40 (riporto 0)
0.40 * 2 = 0.80 (riporto 0)
0.80 * 2 = 1.60 (riporto 1) --> si considera solo più "0.60"
0.60 * 2 = 1.20 (riporto 1) --> si considera solo più "0.20"
0.20 * 2 = 0.40 (riporto 0)
0.40 * 2 = 0.80 (riporto 0)
--> si sono ormai calcolate 8 cifre --> si tronca il risultato
```

Il risultato, a 8 cifre binarie frazionali, è quindi pari a : **10001.10001100** (base 2) .

Esercizio 3

Convertire il numero decimale 17.55 con una precisione di 1/100 (decimale)

Il numero è già stato convertito nell'esercizio precedente. Rimane questa volta da capire quante sono le cifre frazionarie che è necessario ricavare.

Si ricorda la seguente proprieta': la precisione di un numero binaario è data dal valore 1/(2^n), dove n è il numero di cifre frazionarie.

Una precisione di 1/100 decimale si ottiene quindi con 1/128, ossia 1/(2^7).

Il numero di bit frazionari necessari è quindi 7.

Il risultato, a 7 cifre binarie frazionali, è quindi pari a : **10001.1000110** (base 2).

Esercizio 4

Convertire i seguenti numeri in binario, con una precisione di 8 cifre decimali.

Decimale	**Binario**
23.466	*10111.01110111*
61.625	*111101.10100000*
13.543	*1101.10001011*
55.110	*110111.00011100*
19.999	*10011.11111111*
22.001	*10110.00000000*
41.700	*101001.10110011*

Esercizio 5

Convertire il numero binario 1101.00101100 in decimale.

Il procedimento è analogo alla conversione della sola parte intera.

Parte intera: `1*2^0 + 0*2^1 + 1*2^2 + 1*2^3 = 1 + 0 + 4 + 8 = 13`

Parte decimale: `0*2^-1 + 0*2^-2 + 1*2^-3 + 0*2^-4 + 1*2^-5 + 1*2^-6 + 0*2^-7 + 0*2^-8 = 0 + 0 + 0.125 + 0 + 0.03125 + 0.015625 + 0 + 0 = 0,171875`

--> 1101.00101100 (Base 2) = **13.171875** (base 10)

Esercizio 6

Convertire i seguenti numeri frazionari binari in decimale.

Binario	Decimale
11101.11010111	*29.83984375*
10001.00001101	*17.05078125*
00011.10011001	*3.59765625*
11000.11111001	*24.97265625*
10011.00100100	*19.140625*
10010.01001001	*18.28515625*

Esercizio 7

Convertire il numero decimale 11.876 in rappresentazione reale Floating-Point (virgola mobile) su 32 bit. Indicare, in particolare, come vengono rappresentati mantissa ed esponente.

$$-1^{S} \times 2^{E-M} \times (+1.F)$$

Mantissa

La trasformazione prevede la conversione in binario del numero 11.876.

Parte intera

La parte intera di X è 11 (base 10) --> **1011** (base 2).

Parte frazionaria

Essendo la parte intera composta da 4 cifre, si dovranno calcolare 20 cifre decimali (la mantissa è composta da 24 cifre binarie).

0.876 (base 10) --> 0.11100000010000011000 (base 2, 20 cifre con approssimazione per troncamento)

La mantissa sarà pari a: 1011.11100000010000011000

Normalizzando, si avrà: 1.01111100000010000011000 * 2^3

Siccome il numero è positivo, e la prima cifra della mantissa (essendo sempre pari a "1") si omette, la mantissa sarà pari a **01111100000010000011000.**

Esponente

Avendo dovuto fare uno shift della mantissa di tre posizioni (si è moltiplicato per 2^3), l'esponente varrà 3.

Essendo però espresso in codice ECCESSO 127, il numero da inserire come esponente sarà 127+3=130.

Si converte quindi questo numero in binario puro con i metodi già visti, ottenendo 10000010.

Il risultato finale sarà:

Segno	Esponente	Mantissa
0	10000010	01111100000010000 011000

Esercizio 8

Convertire i seguenti numeri decimali frazionari in codifica standard IEEE 754 SP.

Decimale	Segno	Esponente	Mantissa
-12.72	1	*10000010*	*10010111000010100011101*
+14.375	0	*10000010*	*11001100000000000000000*
-46.188	1	*10000100*	*01110001100000010000011*
+7.99	0	*10000001*	*11111111010111000010100*
-12.56	1	*10000010*	*10010001111010111000010*
+5.54	0	*10000001*	*01100010100011110101110*
-2.21	1	*10000000*	*00011010111000010100011*
+0.07	0	*01111011*	*00011110101110000101000*

Esercizio 9

Dati due numeri A=+59 e B=-30 in base 10 rappresentarli entrambi in base 2 con notazione complemento a 2 ed effettuarne la somma algebrica.

Esprimere il risultato finale anche in complemento a 1.

Svolgimento:

1. **Calcolo del numero minimo di bit per rappresentarli entrambi.**

Se n è il numero di bit a disposizione, l'insieme dei numeri rappresentabili in complemento a 2 è dato dall'intervallo

$[-2^{n-1},+2^{n-1}-1]$

Nel caso specifico per rappresentare il maggiore in modulo (N=+59) sono necessari:

$n=\lceil \lg(N+1) \rceil +1$

cioè il "primo intero superiore" del lg(60)=6 aggiungendo un bit, quindi

n=7

2. **Rappresentazione del primo numero**

A=+59 è un numero positivo, la sua rappresentazione con notazione di complemento a 2 è identica a quella del binario puro, con la solita regola di trasformazione attraverso i resti delle divisioni INTERE successive per 2.

Esprimiamolo su 7 bit per uniformità con la rappresentazione di -30

NUMERO	DIVISIONE per 2	RESTO	POSIZIONE
59	29	1	0
29	14	1	1
14	7	0	2
7	3	1	3
3	1	1	4
1	0	1	5
0	0	0	6

Verifica $(0111011)_2 = 1*2^5+1*2^4+1*2^3+1*2+1=(59)_{10}$

3. **Rappresentazione del secondo numero**

Essendo B= -30, applicando la formula della notazione a complemento a 2, 2^n - N, dove n=7 e N è il modulo di B, otteniamo: 128 - 30= 98.

A questo punto sappiamo che la rappresentazione in notazione a complemento a 2 di -30 corrisponde alla rappresentazione di 98 in binario puro, e cioè 1100010.

4. **Calcolo della somma algebrica in complemento a 2**

	Riporto 1	**1 1**
(+59) +		0111011
(-30)		1100010
		0011101

Essendo i riporti sulle cifre più significative concordi, non si ha condizione di overflow.

Adesso dobbiamo verificare che il risultato sia corretto. Notiamo subito che essendo la cifra più significativa del risultato uguale a zero, il numero è positivo, come ci si aspetta.

((+ 59) + (-30)) in base 10 ha come risultato +29.

0011101 è la rappresentazione in binario puro di: $1*2^4+1*2^3+1*2*^2+1=(29)_{10}$

IL RISULTATO E' QUINDI CORRETTO.

SE VOLESSIMO ESPRIMERE IL RISULTATO NELLA NOTAZIONE DEL COMPLEMENTO A 1, ESSA RIMARREBBE INVARIATA TRATTANDOSI DI UN NUMERO POSITIVO, CHE MANTIENE UNA RAPPRESENTAZIONE EQUIVALENTE A QUELLA DELLO STESSO NUMERO ESPRESSO IN BINARIO PURO.

Esercizi su conversione di numeri

- Dati i numeri **A** = **+39** e **B** = **-50** in base 10 rappresentarli entrambi in base 2 notazione complemento a 2, utilizzando il numero minimo di bit per rappresentare ENTRAMBI gli operandi.

Soluzione

1) **Numero minimo di bit per rappresentare entrambi i numeri.**

Se n è il numero di bit a disposizione, l'insieme dei numeri rappresentabili in complemento a 2 è dato dall'intervallo:

$[-2^{n-1}, +2^{n-1} - 1]$

Nel caso specifico, 6 bit non sono sufficienti. Infatti, applicando la formula si ottiene l'intervallo [-32, +31]

7 bit sono sufficienti, infatti si rappresentano i numeri nell'intervallo: [-64, +64]

2) **Rappresentazione del primo numero**

A è un numero positivo. Questo significa che le sue rappresentazioni come numero naturale ed in complemento a due sono uguali. Applichiamo allora la regola che ci consente di ricavare la rappresentazione in base 2 di un numero naturale:

NUMERO	DIVISIONE per 2	RESTO	POSIZIONE
39	19	1	0
19	9	1	1
9	4	1	2
4	2	0	3
2	1	0	4
1	0	1	5

La rappresentazione ottenuta è 100111. Aggiungendo un bit per rappresentare il numero in 7 bit si ha: **0100111**

3) **Rappresentazione del secondo numero**

B è un numero negativo. Esistono due algoritmi possibili per ottenerne la rappresentazione in complemento
a due.
Il primo algoritmo consiste nell'applicare la seguente formula 2^n-X, dove n è il numero di bit da utilizzare
(7 in questo caso) e X è il valore assoluto del numero. Nel caso in esame si ha: 128-50 = 78. A questo punto
sappiamo che la rappresentazione in complemento a due di –50 coincide con la rappresentazione di 78 considerato come numero naturale:

Si ha quindi che il risultato finale è: **1001110**

Il secondo algoritmo consiste nello svolgere i seguenti passi:

1. Calcolare la rappresentazione di 50 come numero naturale:

NUMERO	DIVISIONE per 2	RESTO	POSIZIONE
50	25	0	0
25	12	1	1
12	6	0	2
6	3	0	3
3	1	1	4
1	0	1	5

Il numero ottenuto è 110010 che in 7 bit diventa 0110010

2. Effettuare l'operazione di complementazione di ciascun bit: 1001101
3. Aggiungere un 1 alla cifra meno significativa:

1001101
1
1001110

Come si nota il risultato corrisponde a quello ottenuto con il primo algoritmo.

ESERCIZIO 3

Esprimi in **complemento** a **due** il numero decimale - 61,81 arrestandosi al 6° bit dopo la virgola.
Esprimi lo stesso numero normalizzato in virgola mobile.

0111101,110011 (+61,81) CA2: 1000010,001100+1=1000011,001100 (-61,81)

- Per passare alla forma in virgola mobile, bisogna dapprima normalizzare

questa è la forma non normalizzata: 1000011,001100

per normalizzare si deve spostare la virgola di sei posizioni a sinistra ottenendo:

$$1{,}000011001100 * 2^5$$

- Scriviamo ora il nello standard IEEE 754 con precisione singola (32 bit) e con eccesso M=128

La forma del numero sarà

$$-1^S * 2^{E-M} + 1.F$$

S= 1 (perché il numero è negativo)
E=N+M=5+128=133=10000101 (8 bit)
F=000011001100 (la parte che segue 1.)

1	1	0	0	0	0	1	0	1	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

1 bit	8 bit	23 bit – i meno significativi sono posti a 0