Fondamenti di Informatica
Esercitazione 1 (Soluzioni)

19 settembre 2023

Codifica Binaria

1.1 Sono dati i seguenti interi **senza segno**:

$$x = (111001)_2$$
$$y = (27)_{10}$$

Si effettuino (a mano) le seguenti operazioni:

1. Convertire x in base 10

   Usiamo la definizione:
   $(111001)_2 = 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^0 = 32 + 16 + 8 + 1 = (57)_{10}$

2. Convertire y in base 2. Quanti bit sono necessari per rappresentarlo?

   Usiamo il metodo delle divisioni successive:

27	2
13	1
6	1
3	0
1	1
0	1

   Leggendo i resti dal basso, $(27)_{10} = (11011)_2$; occorrono 5 bit.
   Verifica (conversione inversa usando la definizione): $16 + 8 + 2 + 1 = 27$.

3. Calcolare la somma $x + y$ in aritmetica binaria **senza segno**

(1)	(1)	(1)		(1)	(1)		
	1	1	1	0	0	1	+
	0	1	1	0	1	1	=
1	0	1	0	1	0	0	

4. Scrivere x e y in base 8 e in base 16.

$(\underbrace{111}_{7}\underbrace{001}_{1})_2 = (71)_8$
$(\underbrace{0011}_{3}\underbrace{1001}_{9})_2 = (39)_{16}$
$(\underbrace{011}_{3}\underbrace{011}_{3})_2 = (33)_8$
$(\underbrace{0001}_{1}\underbrace{1011}_{11=B})_2 = (1B)_{16}$

Complemento a due (CP2)

1.2 Si vogliono memorizzare delle temperature in gradi centigradi. Sappiamo che la temperatura sul pianeta Terra è compresa tra -90 e 60 (inclusi). Ipotizzando di rappresentare le temperature con la **codifica CP2**:

1. Quanti bit sono necessari?

   Il valore assoluto del numero negativo più piccolo possibile è 90. Richiede (senza segno) 7 bit.
   Il numero positivo più grande possibile è 60. Richiede (senza segno) 6 bit.
   Un numero di bit sufficienti è quindi $m = \max(7, 6) + 1 = 8$.
   Con $m = 8$ possiamo rappresentare tutti i numeri nell'intervallo:
   $[-2^7, 2^7 - 1] = [-128, 127]$, estremi inclusi.
   Con un bit in meno ($m = 7$), potremmo rappresentare tutti i numeri nell'intervallo: $[-64, 63]$, il che non sarebbe sufficiente. Il numero di bit necessari è quindi 8.

2. Quali sono le temperature massima e minima effettivamente memorizzabili?

   Vedi il punto precedente.

3. Quante temperature si possono memorizzare avendo a disposizione 500 byte di memoria?

   Una memoria di 500 byte corrisponde a $500 \times 8 = 4000$ bit.
   Siccome una temperatura richiede 8 bit, possiamo memorizzare $4000/8 = 500$ temperature.

4. (*Bonus*) Come cambiano le risposte se si vogliono memorizzare temperature del pianeta Marte, sapendo che sono sempre comprese tra -128 e 20 (inclusi)?

   Il valore assoluto del numero negativo più piccolo possibile è 128. Richiede (senza segno) 8 bit.
   Il numero positivo più grande possibile è 20. Richiede (senza segno) 5 bit.
   Un numero di bit sufficienti è quindi $m = \max(8, 5) + 1 = 9$.
   Con $m = 9$ possiamo rappresentare tutti i numeri nell'intervallo:

$[-256, 255]$, estremi inclusi.
Tuttavia, con un bit in meno ($m = 8$), potremmo rappresentare tutti i numeri nell'intervallo $[-128, 127]$, che è sufficiente!
Servono quindi 8 bit.

1.3 Sono dati i seguenti interi:

$$x = (10100101)_{CP_2}$$
$$y = (-62)_{10}$$

Si effettuino (a mano) le seguenti operazioni, precisando sempre il bit di carry e il bit di overflow:

1. Convertire x in base 10

   Usiamo la definizione:
   $(10100101)_{CP2} = -2^7 + 2^5 + 2^2 + 1 = -128 + 32 + 4 + 1 = (-91)_{10}$

2. Convertire y in codifica CP2. Quanti bit sono necessari per rappresentarlo?

   Scriviamo prima la codifica binaria di $|y| = 62$:

62	2
31	0
15	1
7	1
3	1
1	1
0	1

   Quindi, $(63)_{10} = (111110)_2$; occorrono 6 bit per il numero senza segno, quindi 7 per il numero in CP2.
   Aggiungiamo uno 0 a sinistra di 111110 (per utilizzare effettivamente 7 bit) ed eseguiamo l'operazione di complemento a 2. Prima complementiamo le cifre di 0111110:

   $$\begin{array}{cccccccc} 0 & 1 & 1 & 1 & 1 & 1 & 0 & ! \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 1 & \end{array}$$

   Poi sommiamo 1:

   $$\begin{array}{cccccccc} & & & & & (1) & & \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & + \\ & & & & & & 1 & \\ \hline 1 & 0 & 0 & 0 & 0 & 1 & 0 & \end{array}$$

   Quindi, $(-62)_{10} = (1000010)_{CP2}$

3. Calcolare la differenza $x - y$ in aritmetica **CP2**.

Questo equivare a calcolare $(-91)_{10} + (62)_{10}$. Dovremmo quindi negare (con l'operazione di complemento a 2) il secondo operando e poi effetuare la somma. In questo caso sappiamo già, dal punto 2, la rappresentazione binaria di $(62)_{10}$. Quindi:

$$
\begin{array}{cccccccccc}
 & & & (1) & (1) & (1) & (1) & & & \\
 & & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & + \\
 & & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & = \\
\hline
[0] & (0) & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 &
\end{array}
$$

Per allineare i due numeri, abbiamo aggiunto zeri alla sinistra del secondo addendo, in quanto è un numero **positivo**.
Il bit di carry è (0) e viene ignorato.
Essendo una somma di un numero positivo e uno negativo, il bit di overflow è sicuramente $[0]$.

4. Calcolare la somma $x + y$ in aritmetica **CP2**.

   Questo equivare a calcolare $(-91)_{10} + (-62)_{10}$. In CP2:

$$
\begin{array}{ccccccccccc}
 & & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & + \\
 & & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & = \\
\hline
[1] & (1) & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 &
\end{array}
$$

   Per allineare i due numeri, abbiamo aggiunto uni alla sinistra del secondo addendo, in quanto è un numero **negativo**.
   Il bit di carry è (1) e viene ignorato.
   Abbiamo sommato due numeri negativi e ottenuto un numero positivo. Il risultato è inconsistente, quindi settiamo il bit di overflow a $[1]$.

Numeri Reali

1.4 Sono dati i seguenti numeri reali:

$$
\begin{aligned}
x &= 0.90625 \\
y &= 0.768 \\
z &= 14.63
\end{aligned}
$$

Si effettuino (a mano) le seguenti operazioni:

1. Convertire x e y in rappresentazione binaria usando 4 bit

   Usiamo il metodo delle moltiplicazioni successive:

$$\begin{array}{r|l} 0.90625 \text{ x } 2 & = 1 + 0.8125 \\ 0.8125 \text{ x } 2 & = 1 + 0.625 \\ 0.625 \text{ x } 2 & = 1 + 0.25 \\ 0.25 \text{ x } 2 & = 0 + 0.5 \end{array}$$

Leggendo i bit dall'alto, 0.90625 0.1110; usando 4 bit, con un errore massimo di 2^{-4}.
Verifica (conversione inversa usando la definizione): $0.5+0.25+0.125 = 0.875$.
Verifica Errore: $0.90625 - 0.875 = 0.03125 < 0.0625(2^4)$

$$\begin{array}{r|l} 0.768 \text{ x } 2 & = 1 + 0.536 \\ 0.536 \text{ x } 2 & = 1 + 0.1072 \\ 0.1072 \text{ x } 2 & = 0.2144 \\ 0.2144 \text{ x } 2 & = 0 + 0.4288 \end{array}$$

Leggendo i bit dall'alto, 0.768 0.1100; usando 4 bit, con un errore massimo di 2^{-4}.
Verifica (conversione inversa usando la definizione): $0.5+0.25+0.+0 = 0.75$.
Verifica Errore: $0.768 - 0.75 = 0.018 < 0.0625(2^4)$

2. Convertire x e y in rappresentazione binaria usando 5 bits

Usiamo il metodo delle moltiplicazioni successive:

$$\begin{array}{r|l} 0.90625 \text{ x } 2 & = 1 + 0.8125 \\ 0.8125 \text{ x } 2 & = 1 + 0.625 \\ 0.625 \text{ x } 2 & = 1 + 0.25 \\ 0.25 \text{ x } 2 & = 0 + 0.5 \\ 0.5 \text{ x } 2 & = 1 + 0 \end{array}$$

Leggendo i bit dall'alto, 0.90625 0.1110; usando 4 bit, con un errore massimo di 2^{-5}.
Verifica (conversione inversa usando la definizione): $0.5 + 0.25 + 0.125 + 0.03125 = 0.90625$.
Verifica Errore: $0.90625 - 0.90625 = 0. < 0.0625(2^{-4})$

$$\begin{array}{r|l} 0.768 \text{ x } 2 & = 1 + 0.536 \\ 0.536 \text{ x } 2 & = 1 + 0.1072 \\ 0.1072 \text{ x } 2 & = 0.2144 \\ 0.2144 \text{ x } 2 & = 0 + 0.4288 \\ 0.4288 \text{ x } 2 & = 0 + 0.8576 \end{array}$$

Leggendo i bit dall'alto, 0.768 0.11000; usando 5 bit, con un errore massimo di 2^{-5}.
Verifica (conversione inversa usando la definizione): $0.5 + 0.25 + 0. + 0 + 0 = 0.75$.
Verifica Errore: $0.768 - 0.75 = 0.018 < 0.03125(2^{-5})$

3. Convertire z in rappresentazione a virgola fissa con 5 bit di parte intera e 6 bit di parte reale.

   Usiamo la definizione:
   $14 = 8+4+2 = 1*2^3+1*2^2+1*2^1 = 1110_2$. Segno positivo, qindi aggiungiamo uno 0 all'inizio 14_{10} = 01110_2

0.63 x 2	$= 1 + 0.26$
0.26 x 2	$= 0 + 0.52$
0.52 x 2	$= 1 + 0.04$
0.04 x 2	$= 0 + 0.08$
0.08 x 2	$= 0 + 0.16$
0.16 x 2	$= 0 + 0.32$

   Leggendo i bit dall'alto, 14.63 01110.101000; usando 5 bit di parte intera e 6 bit di parte reale, con un errore massimo di 2^{-6}.
   Verifica (conversione inversa usando la definizione): $0.5++0.125+0.+0.+0. = 0.6125$.
   Verifica Errore: $0.63 - 0.6125 = 0.0175 < 0.015625(2^{-6})$

4. Convertire z in rappresentazione a virgola mobile secondo lo standard IEEE 754-1985 a precisione singola

   (a) Definisco il bit di segno. $z > 0$ => $S = 0$

   (b) Codifico in virgola fissa in base 2, parte frazionaria e parte intera (gia fatto in parte)

0.63 x 2	$= 1 + 0.26$
0.26 x 2	$= 0 + 0.52$
0.52 x 2	$= 1 + 0.04$
0.04 x 2	$= 0 + 0.08$
0.08 x 2	$= 0 + 0.16$
0.16 x 2	$= 0 + 0.32$
0.32 x 2	$= 0 + 0.64$
0.64 x 2	$= 1 + 0.28$
0.28 x 2	$= 0 + 0.56$

   14.63 $01110.101000010...$

   (c) Porto il numero in forma normalizzata in base 2
   14.63 $1.110101000010....$ x 2^3

   (d) Definisco M a 23 bit come la mantissa senza il primo bit (sempre 1) M = 1.110101000010....

(e) Calcolo $E = n+127 = 3+127 = 130 = 128+2 = 2^7+2^1- > 10000010$ (8 bit)