Esercitazione sulle Rappresentazioni Numeriche

Esistono 10 tipi di persone al mondo: quelli che conoscono il codice binario e quelli che non lo conoscono

Alessandro Pellegrini

Cosa studiare prima

- Conversione da un numero da binario a decimale e viceversa
- Definizione delle rappresentazioni:
  - senza segno
  - con modulo e segno
  - in complemento alla base
  - in virgola mobile
- Operazioni elementari (somma, sottrazione, moltiplicazione e divisione) tra numeri nelle diverse rappresentazioni (naturalmente la coppia di numeri su cui si opera ha la stessa rappresentazione per i due operandi)
- definizione di overflow ed underflow

000. Conversione da binario a decimale

Per rappresentare un numero (intero) in base binaria si possono utilizzare soltanto due cifre: 0 e 1. In linea generale, un qualsiasi numero in base due ha un valore decimale che può essere calcolato secondo il seguente esempio.

Considerando $N = 101011.1011_2$:

2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
1	0	1	0	1	1	1	0	1	1

da cui si ha: $N = 1\cdot2^5+0\cdot2^4+1\cdot2^3+0\cdot2^2+1\cdot2^1+1\cdot2^0+1\cdot2^{-1}+0\cdot2^{-2}+1\cdot2^{-3}+1\cdot2^{-4} = 43.6875_{10}$

001. Conversione da decimale a binario

Per convertire un numero da decimale a binario, è necessario distinguere la parte intera dalla parte frazionaria. È infatti necessario convertirle separatamente, utilizzando due regole pratiche differenti, e poi unire insieme il risultato.

Regola pratica per la conversione della parte intera In generale, per convertire un numero decimale in un'altra base, è sufficiente dividere ripetutamente il numero per la nuova base finché non si ottiene 0 come risultato, e scrivere poi i resti ottenuti, a partire dalla posizione meno significativa. Ad esempio, per convertire il numero 57_{10} in base due è sufficiente dividerlo ripetutamente per due:

$$
\begin{array}{r|lll}
\multicolumn{2}{c}{/2} & & \\
57 & 28 & R=1 & \uparrow \\
28 & 14 & R=0 & \\
14 & 7 & R=0 & \text{ordine} \\
7 & 3 & R=1 & \\
3 & 1 & R=1 & \\
1 & 0 & R=1 & \\
0 & & &
\end{array}
$$

da cui si ottiene che $57_{10} = 111001_2$

Regola pratica per la conversione della parte frazionaria Per convertire la parte frazionaria di un numero decimale in binario, è sufficiente:

- moltiplicare la parte frazionaria per 2
- scrivere il valore della parte intera ottenuta
- ripetere il procedimento sulla nuova parte frazionaria ottenuta, finché non si ottiene una parte frazionaria nulla
- Prendere le parti intere dalla più significativa alla meno significativa

Nota: nel caso dei numeri periodici, si potrebbe non ottenere mai una parte frazionaria nulla.

Ad esempio, per convertire 0.6875_{10} in binario:

$$
\begin{array}{r|lll}
\multicolumn{2}{c}{*2} & & \\
0.6875 & 1.375 & U=1 & \\
0.375 & 0.75 & U=0 & \text{ordine} \\
0.75 & 1.5 & U=1 & \\
0.5 & 1.0 & U=1 & \\
0 & & & \downarrow
\end{array}
$$

da cui si ottiene che $0.6875_{10} = 0.1011_2$.

Se si volesse convertire il numero 57.6875_{10} sarebbe sufficente unire i due risultati (per la parte intera e per la parte frazionaria), ottenendo 111001.1011_2.

È importante osservare come, passando da una base all'altra, non tutte le proprietà dei numeri vengono sempre conservate. Un esempio evidente è la *periodicità*. Questo significa che non si può affermare con assoluta certezza che se un numero frazionario è rappresentabile con un numero finito di cifre in una determinata base, questo sarà rappresentabile comunque con un numero finito di cifre (per quanto differente) in un'altra base: si commette, cioè, un *errore di approssimazione*.

Prendiamo ad esempio in considerazione il numero $(0.1)_{10}$, questo non è un numero periodico. Andando ad effettuare la conversione in base 2, otteniamo:

$$
\begin{array}{c|cc}
\multicolumn{2}{c}{\quad *2} & \\
0.1 & 0.2 & U = 0 \\
0.2 & 0.4 & U = 0 \\
0.4 & 0.8 & U = 0 \\
0.8 & 1.6 & U = 1 \\
0.6 & 1.2 & U = 1 \\
0.2 & 0.4 & U = 0
\end{array}
\quad \Big\downarrow \text{ ordine}
$$

e così via. La conversione, pertanto, è tale per cui $(0.1)_{10} = (0.0\overline{0011})_2$, ovverosia $(0.1)_{10}$ corrisponde ad un numero periodico in base 2.

010. Somma e sottrazione di due numeri interi

Somma e sottrazione, nel sistema binario, seguono le stesse regole del sistema decimale. Pertanto, se vogliamo sommare 110_2 con 10_2 procediamo così:

$$
\begin{array}{rc}
110 & + \\
10 & = \\
\hline
1000 &
\end{array}
\qquad
\begin{array}{r}
6 \\
2 \\
\hline
8
\end{array}
$$

Allo stesso modo, se vogliamo sottrarre 11_2 da 1110_2, procediamo così:

$$
\begin{array}{rc}
1110 & - \\
11 & = \\
\hline
1011 &
\end{array}
\qquad
\begin{array}{r}
14 \\
3 \\
\hline
11
\end{array}
$$

011. Rappresentazione di numeri interi con segno

Data una parola di lunghezza n, 1 bit viene utilizzato per rappresentare il segno, $n-1$ bit vengono utilizzati per rappresentare il numero in valore assoluto. Il valore 0 assume il significato di segno +, il valore 1 assume il significato di segno -. Ad esempio, utilizzando 8 bit si ha:

1	0	0	0	1	1	0	0	= -12

0	0	0	0	1	1	0	0	= 12

Esisteranno quindi due codifiche per il valore 0:

0	0	0	0	0	0	0	0	= +0

1	0	0	0	0	0	0	0	= -0

Questa rappresentazione assume anche il nome di *grandezza e segno*.7

100. Rappresentazione di numeri interi in complemento a 2

Nel complemento a 2, data una parola di n bit, si possono rappresentare 2^n numeri. Essi verranno divisi in due metà, una positiva ed una negativa. Si potranno rappresentare, quindi, tutti quei numeri nell'intervallo $[-2^{n-1}, +2^{n-1}-1]$, considerando lo 0 nella metà dei numeri positivi.

Pertanto, i numeri positivi sono rappresentati normalmente (rappresentazione binaria dei numeri positivi), con il bit più significativo pari a 0. I numeri negativi si ottengono come complemento a 2 del numero positivo corrispondente, ed hanno il bit più significativo pari a 1.

Il complemento a 2 di un numero N in base 2 rappresentato utilizzando n cifre binarie è dato da:

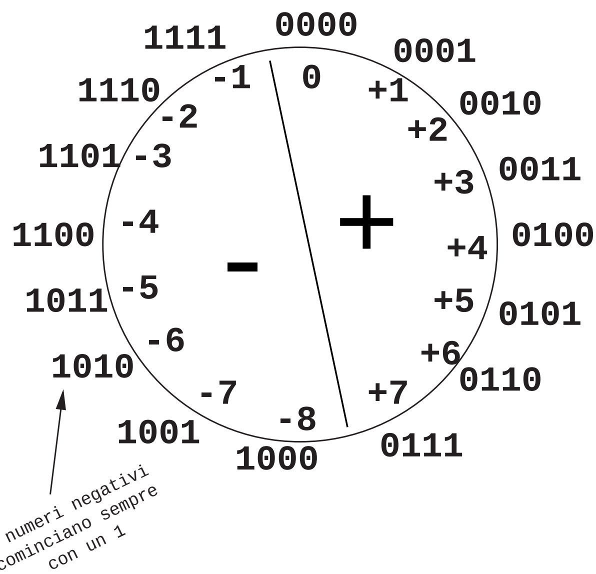
$$C_2 = 2^n - N$$

Ad esempio, considerando $N = 10100$ e $n = 8$:

$$\begin{array}{rcc} C_2 = & 10000000 & - \\ & 00010100 & = \\ \hline & 11101100 & \end{array}$$

Regola pratica: per calcolare il complemento a due, si parte dal primo bit meno significativo (a destra) e si lasciano immutati tutti i bit fino al primo 1, dopo si invertono tutti gli altri.

Visivamente, utilizzando 4 bit:



Alcuni esempi:

$$\begin{array}{|c|c|c|c|c|c|c|c|}\hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline\end{array} = 0$$

$$\begin{array}{|c|c|c|c|c|c|c|c|}\hline 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline\end{array} = 2^{n-1} - 1$$

$$\begin{array}{|c|c|c|c|c|c|c|c|}\hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline\end{array} = \text{-1}$$

In questa rappresentazione, il bit più significativo identifica il gruppo di appartenenza (0: numeri positivi; 1: numeri negativi), ma non va confuso con il segno della rappresentazione in modulo e segno. È una rappresentazione più vantaggiosa di quella in modulo e segno, perché consente di eseguire somme e sottrazioni come un'unica operazione.

Se vogliamo rappresentare il numero -67 in complemento a due con 8 bit, applicando la regola pratica, otteniamo:

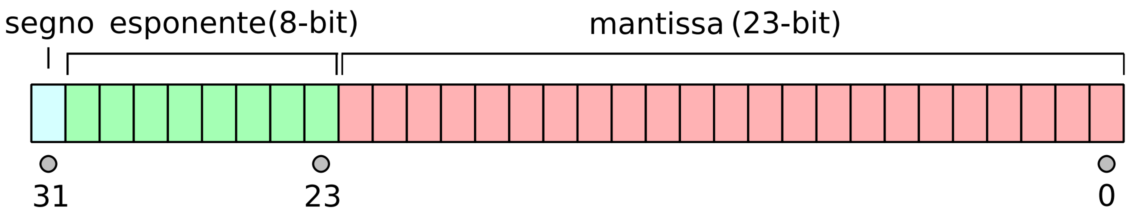
0	1	0	0	0	0	1	1	= 67

1	0	1	1	1	1	0	1	= -67

Esercizio: Cosa succede se si prova a calcolare il complemento a due, utilizzando soli quattro bit, del numero negativo -8 (1000)?

101. Rappresentazione di un numero in virgola mobile

Il formato standard per la rappresentazione dei numeri in virgola mobile è chiamato IEEE 754. Con parole di 32 bit, un numero in virgola mobile viene rappresentato come segue:



Il numero è quindi composto di tre campi:

1. *segno* s, di 1 bit
2. *esponente* e, di 8 bit
3. *mantissa* m, di 23 bit

Il **segno** ha valore 1 per indicare i numeri negativi, 0 per i positivi.
L'**esponente** e può individuare $2^8 = 256$ possibili valori. I valori 0 e 255 sono riservati per rappresentare famiglie di numeri particolari. Pertanto, i 254 valori restanti vengono suddivisi nell'intervallo $[-126, 127]$. I numeri negativi non vengono rappresentati utilizzando il complemento a 2, ma viene utilizzato un *bias* di 127. Pertanto, data una rappresentazione e di un esponente, il valore ad esso associato può essere calcolato come $E = e - 127$.
La **mantissa** è di solito normalizzata tra 1 e 1.9999998808 (corrispondente alla rappresentazione in base due $(1.11111111111111111111111)_2$). Nella rappresentazione di base, la cifra prima della virgola (che è quindi sempre 1) viene omessa dalla rappresentazione, perché considerata implicita.

Pertanto, il valore decimale del numero rappresentato può essere calcolato (in generale) come:

$$(-1)^s \cdot 2^E \cdot 1.m$$

Esistono delle eccezioni, individuate dai due valori riservati dell'esponente. La rappresentazione di un numero in formato IEEE 754 è riassunta in questa tabella:

e	m	Valore del numero
$[1, 254]$	qualsiasi	$(-1)^s \cdot 2^{-126} \cdot 1.m$ (numeri normali)
0	$\neq 0$	$(-1)^s \cdot 2^{-126} \cdot 0.m$ (numeri subnormali o denormalizzati)
0	0	$(-1)^s$ (zero con segno)
255	0	$(-1)^s \cdot \infty$ (infinito, con segno)
255	$\neq 0$	`NaN` (not a number)

Conversione da decimale a virgola mobile

Prendiamo in considerazione il numero decimale $N = -5,828125_{10}$. Procediamo attraverso i seguenti passi:

a) Determinazione del segno

Poiché il numero è negativo, poniamo $s = 1$.

b) Conversione della parte intera

Procediamo come nel caso di numero intero qualsiasi:

$$
\begin{array}{r|l l}
\multicolumn{2}{c}{/2} & \\
5 & 2 & \mathrm{R} = 1 \\
2 & 1 & \mathrm{R} = 0 \\
1 & 0 & \mathrm{R} = 1 \\
0 & &
\end{array}
\quad \uparrow \text{ordine}
$$

da cui si ottiene che $5_{10} = 101_2$.

c) Conversione della parte frazionaria

Anche qui, la conversione segue sempre la stessa regola:

$$
\begin{array}{r|l l}
\multicolumn{2}{c}{*2} & \\
0.828125 & 1.65625 & \mathrm{U} = 1 \\
0.65625 & 1.3125 & \mathrm{U} = 1 \\
0.3125 & 0.625 & \mathrm{U} = 0 \\
0.625 & 1.25 & \mathrm{U} = 1 \\
0.25 & 0.5 & \mathrm{U} = 0 \\
0.5 & 1 & \mathrm{U} = 1 \\
0 & &
\end{array}
\quad \downarrow \text{ordine}
$$

da cui si ottiene che $0.828125_{10} = 0.110101_2$.

d) Normalizzazione della mantissa

Il numero che abbiamo calcolato fin'ora è 101.110101_2. Lo standard IEEE 754 richiede che, per essere rappresentato, il numero sia nella forma $1.m$, così da poter omettere l'unità 1 che risulta essere sempre implicita[1] . Pertanto vale l'uguaglianza:

[1] La scelta di questa rappresentazione consente di aumentare di un bit l'espressività della mantissa, proprio perché l'1 prima della virgola viene omesso.

$$101.110101 = 1.\underbrace{01110101}_{m} \cdot 2^2$$

In cui, avendo spostato la virgola verso sinistra di due posizioni, dobbiamo moltiplicare il numero per 2^2. La mantissa m è a questo punto determinata.

e) Rappresentazione dell'esponente

Il nostro numero è ora nella forma $1.m * 2^E$. Dobbiamo rappresentare il nostro esponente $E = 2$. Dobbiamo prima di tutto applicare il bias:

$$e = E + 127 = 2 + 127 = 129$$

E possiamo ora procedere a convertire questo valore in binario, come sempre:

	/2		
129	64	R = 1	↑ ordine
64	32	R = 0	
32	16	R = 0	
16	8	R = 0	
8	4	R = 0	
4	2	R = 0	
2	1	R = 0	
1	0	R = 1	
0			

e pertanto il nostro esponente sarà $e = 10000001$.

f) Rappresentazione complessiva

Abbiamo determinato i valori di s, e ed m. Il numero in formato IEE 754 è dato dalla concatenazione dei tre campi:

1	1	0	0	0	0	0	0	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= -5,828125
31								23																							0	

g) Rappresentazione in esadecimale della virgola mobile

Può essere utile adottare una notazione più compatta per rappresentare il numero. A questo scopo ci viene incontro la notazione in base 16, o esadecimale. Poiché ”$^4 = 16$, abbiamo che ogni quartetto di bit viene rappresentato da una singola cifra esadecimale. Pertanto, per effettuare la conversione è sufficiente raggruppare i bit in gruppi di quattro ed effettuare a conversione gruppo a gruppo:

1100	0000	1011	1010	1000	0000	0000	0000
C	0	B	A	8	0	0	0

Pertanto il nostro numero può essere scritto come $C0BA8000_{16}$ o, utilizzando due notazioni alternative, `C0BA8000H`[2] o `0xC0BA8000`.

È importante notare che il valore `0xC`, prima cifra del numero convertito, è composto da 4 bit che appartengono a due campi: il primo è il segno, i restanti sono i primi tre dell'esponente. Questo metodo di rappresentazione compatta non va confuso con la rappresentazione esadecimale in virgola fissa, in cui è necessario aggiungere degli zeri davanti alla parte intera o dopo la parte decimale per poter procedere alla conversione, come nel seguente esempio:

$$N = 101.1011001_2 \Rightarrow 0101.10110010_2 \Rightarrow 5.B2_{16}$$

Conversione da virgola mobile a decimale

Consideriamo il seguente numero rappresentato in formato IEEE 754:

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

31 23 0

Abbiamo che:

- $s = 1$
- $e = 2^7 + 2^0 = 129 \Rightarrow E = 129 - 127 = 2$
- $m = 1 + 2^{-2} = 1.25$, in cui l'1 sommato a 2^{-2} viene sommato poiché, essendo la mantissa normalizzata, l'1 è implicito.

Pertanto il numero rappresentato da questa parola di bit è dato da:

$$N = (-1)^s \cdot 2^E \cdot 1.m = (-1)^1 \cdot 2^2 \cdot 1.25 = -5$$

È dunque evidente che, nonostante il formato sia in virgola mobile, è anche possibile rappresentare numeri interi!

110. Errore assoluto ed errore relativo

Ogni volta che rappresentiamo un numero N in virgola mobile, in realtà rappresentiamo un numero N' che potrebbe non essere uguale ad N per via di un errore di approssimazione. È allora interessante vedere di quanto la nostra rappresentazione sta sbagliando.

Si può dunque introdurre l'*errore assoluto*, calcolato come $\varepsilon_A = N - N'$, ossia una grandezza algebrica che indica quanto la nostra approssimazione "ha perso" dell'informazione originale.

L'*errore relativo*, invece, è una grandezza adimensionale che ci permette di capire se l'errore che abbiamo commesso utilizzando la nostra approssimazione è piccolo oppure grande. Esso è dato da $\varepsilon_R = \frac{\varepsilon_A}{N} = \frac{N - N'}{N}$.

Vediamo un esempio in base 10. Prendiamo il numero $x = 3.5648722189$ e decidiamo di rappresentarlo utlizzando soltanto 4 cifre decimali:

[2] La `H` è l'abbreviazione di *hexadecimal*, esadecimale in inglese.

$$x \approx \bar{x} = 3.5648$$

Andiamo a calcolare qual è l'errore relativo che abbiamo introdotto con questa rappresentazione approssimata:

$$\varepsilon_R = \frac{x - \bar{x}}{x} = \frac{3.5648722189 - 3.5648}{3.5648722189} = 0,000020258$$

che ci indica che l'errore commesso è, di fatto, non molto grande.

Vale anche l'interessante relazione che ci dice che $-log_{10}(\varepsilon_R) \simeq$ numero di cifre senza errore nella nostra rappresentazione. Infatti:

$$-log_{10}(0,000020258) = 4.69$$