RAPPRESENTAZIONE BINARIA DEI NUMERI

Andrea Bobbio

Anno Accademico 1996-1997

Sistemi di Numerazione

Il valore di un numero può essere espresso con diverse rappresentazioni.

◇ *non posizionali:*
ad esempio la numerazione romana.

◇ *posizionali:*
viene associato un peso a ciascuna posizione all'interno della rappresentazione del numero.
Il valore del numero è ottenuto facendo la somma dei prodotti di ciascuna cifra per il relativo peso (ad esempio, numerazione decimale).

Sistemi Posizionali

Regola Generale

Sia r la base della numerazione.
Un numero frazionario assoluto $N_{(r)}$ viene rappresentato con una successione di cifre d_i che soddisfano le seguenti regole:

L'insieme delle cifre d_i costituisce un insieme di simboli diversi di cardinalità r, tali che

$$0 \leq d_i \leq r - 1$$

Per cui:

$$N_{(r)} = d_{n-1}d_{n-2}\cdots d_1 d_0 \,.\, d_{-1}d_{-2}\cdots d_{-m}$$

Il valore di tale numero è dato da:

$$V(N_{(r)}) = \sum_{i=-m}^{n-1} d_i \cdot r^i$$

Numerazione Binaria

Base: $r = 2$

Simboli base: $d_i \in \{0, 1\}$

ESEMPIO:

$$N_{(2)} = 11001$$

$$V(N_{(2)}) = (1 \cdot 2^4) + (1 \cdot 2^3) + (0 \cdot 2^2) + (0 \cdot 2^1) + (1 \cdot$$

$$= 25_{(10)}$$

Sequenza dei pesi associati alle varie posizioni nei numeri binari:

$\cdots$	2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}	2^{-3}	$\cdots$
$\cdots$	(8)	(4)	(2)	(1)		$(\frac{1}{2})$	$(\frac{1}{4})$	$(\frac{1}{8})$	$\cdots$

Vantaggi Sistemi Posizionali

◇ Esprimono qualunque valore finito.

◇ Sono irridondanti (ad ogni valore corrisponde una sola sequenza di simboli e viceversa).

◇ Le regole formali per le operazioni aritmetiche sono indipendenti dalla base della rappresentazione.

Numero Massimo Rappresentabile con n Cifre

Il numero massimo rappresentabile con n cifre in base r risulta:

$$\begin{aligned} N_{max} &= (r-1)\,r^{n-1} + (r-1)\,r^{n-2} + \cdots \\ &\quad + (r-1)\,r + (r-1)\,r^0 \\ &= \sum_{i=0}^{n-1} (r-1) \cdot r^i \\ &= r^n - 1 \end{aligned}$$

Un generico numero N_r di n cifre in base r è compreso nel seguente intervallo:

$$r^{n-1} - 1 < N_r \leq r^n - 1$$

Numero di Cifre per Rappresentare un Numero in Base r

Dato un numero N_r in base r, il numero di cifre necessario per la sua rappresentazione può essere ricavato nel seguente modo:

$$r^{n-1} < N_r + 1 \leq r^n$$

$$n - 1 < \log_r (N_r + 1) \leq n$$

Da cui:

$$n = \lceil \log_r (N_r + 1) \rceil$$

Numero di Cifre per Rappresentare un Numero in Base r

ESEMPIO

Sia dato il numero decimale $N_{10} = 415_{10}$

Con base $r = 2$ si ottiene:

$$n = \lceil \log_2 (415 + 1) \rceil = \lceil 8.70 \rceil = 9$$

Da cui:

$$415_{10} \rightarrow (\,1\,1\,0\,0\,1\,1\,1\,1\,1\,)_2$$

Con base $r = 7$ si ottiene:

$$n = \lceil \log_7 (415 + 1) \rceil = \lceil 3.09 \rceil = 4$$

Da cui:

$$415_{10} \rightarrow (\,1132\,)_7$$

Conversione di Base (da base r → a base 10)

Basta esprimere le cifre d_i e le potenze della base r in base 10 e applicare la definizione di numero posizionale:

$$N_{(r)} = \sum_{i=-m}^{n-1} d_i \cdot r^i$$

ESEMPIO

$$(3246)_7 = 3 \cdot 7^3 + 2 \cdot 7^2 + 4 \cdot 7 + 6 = (1161)_{10}$$

NOTA:

Se si considerano separatamente la parte intera e la parte frazionaria del numero da convertire si possono applicare algoritmi iterativi più agevoli da programmare.

$$N_r = I_r \,.\, F_r$$

Algoritmo iterativo per la conversione da base r → a base 10 (parte intera)

$$I_r = d_{n-1}r^{n-1} + d_{n-2}r^{n-2} + \ldots + d_2r^2 + d_1r^1 + d_0r^0$$

$$= ((\cdots((d_{n-1}r + d_{n-2})r + d_{n-3})r + \ldots + d_1)r + d_0$$

Esempio:

$$(3246)_7 = (((3 \cdot 7 + 2) \cdot 7 + 4) \cdot 7 + 6 = (1161)_{10}$$

Algoritmo:

1. Inizio;
2. Partire dalla cifra più significativa;
3. Se raggiunta la cifra meno significativa andare a 7);
4. Moltiplicare per la base;
5. Sommare la cifra successiva;
6. Andare a 3);
7. Fine.

Algoritmo iterativo per la conversione da base $r \rightarrow$ a base 10 (parte frazionaria)

$$F_r = .d_{-1}r^{-1} + d_{-2}r^{-2} + \ldots + d_{-(m-1)}r^{-(m-1)} + d_{-m}r^{-m}$$

$$= .\frac{1}{r}(d_{-1} + \frac{1}{r}(d_{-2} + \frac{1}{r}(d_{-3} + \ldots + \frac{1}{r}(d_{-(m-1)} + \frac{1}{r}d_{-m}))\cdots))$$

Esempio:

$$(0.101)_2 = .\frac{1}{2}\left(1 + \frac{1}{2}\left(0 + \frac{1}{2}1\right)\right)$$

Algoritmo:

1. Inizio;
2. Partire dalla cifra meno significativa;
3. Dividere per la base;
4. Se raggiunta cifra più signif. andare a 8);
5. Sommare la cifra precedente;
6. Dividere per la base;
7. Andare a 4);
8. Fine.

Conversione da sistema decimale a sistema in base r

Si considerano separatamente la parte intera e la parte frazionaria:

$$N_r = I_r . F_r$$

◇ Conversione parte intera *(algoritmo per divisioni)*

$$I_r = d_{n-1}r^{n-1} + d_{n-2}r^{n-2} + \ldots + d_1r^1 + d_0r^0$$

◇ Conversione parte frazionaria *(algoritmo per prodotti)*

$$F_r = .d_{-1}r^{-1} + d_{-2}r^{-2} + \ldots + d_{-m+1}r^{-m+1} + d_{-}$$

Algoritmo iterativo per la conversione da base 10 $\rightarrow$ a base r (parte intera)

$$
\begin{aligned}
I_{10} &= d_{n-1}r^{n-1} + d_{n-2}r^{n-2} + \ldots + d_2r^2 + d_1r^1 + d_0r^0 \\
&= ((\cdots((d_{n-1}r + d_{n-2})r + d_{n-3})r + \ldots + d_1)r + d_0
\end{aligned}
$$

Si chiami Q_0 il quoziente intero I_{10}/r;
d_0 è il resto intero della divisione I_{10}/r.

Si può quindi scrivere iterativamente:

$$
\begin{aligned}
I_{10} &= Q_0\, r + d_0 \\
Q_0 &= Q_1\, r + d_1 \\
Q_1 &= Q_2\, r + d_2 \\
\ldots &\quad \ldots \\
Q_{n-2} &= 0 \cdot r + d_{n-1}
\end{aligned}
$$

Algoritmo iterativo per la conversione da base 10 $\rightarrow$ a base r (parte intera)

L'algoritmo si basa sul calcolo iterativo del quoziente intero diviso per la base. Il resto della divisione fornisce una cifra del numero nella nuova base a partire dalla cifra meno significativa.

Algoritmo:

1. Inizio;
2. Dividere il numero decimale per la base di arrivo r e prendere il quoziente intero;
3. Il quoziente è il nuovo dividendo;
4. Il resto è una cifra nella nuova base a partire dalla cifra meno significativa ;
5. Se quoziente $\neq 0$ torna a 2);
6. Fine.

Algoritmo iterativo per la conversione da base 10 $\rightarrow$ a base r (parte intera)

ESEMPIO:

Convertire 25_{10} in binario.

$$
\begin{array}{rl}
25 : 2 & \\
\hline
12 : 2 & \text{resto} \Rightarrow 1 \\
\hline
6 : 2 & \text{resto} \Rightarrow 0 \\
\hline
3 : 2 & \text{resto} \Rightarrow 0 \\
\hline
1 : 2 & \text{resto} \Rightarrow 1 \\
\hline
0 & \text{resto} \Rightarrow 1
\end{array}
$$

$$25_{10} \Rightarrow 1\,1\,0\,0\,1_{\,2}$$

Algoritmo iterativo per la conversione da base 10 $\rightarrow$ a base r (parte intera)

ESEMPIO:

Convertire 9852_{10} in ottale.

$$
\begin{array}{rl}
9852 : 8 & \\
\hline
1231 : 8 & \text{resto} \Rightarrow 4 \\
\hline
153 : 8 & \text{resto} \Rightarrow 7 \\
\hline
19 : 8 & \text{resto} \Rightarrow 1 \\
\hline
2 : 8 & \text{resto} \Rightarrow 3 \\
\hline
0 & \text{resto} \Rightarrow 2
\end{array}
$$

$$9852_{10} \Rightarrow 23174_{8}$$

Algoritmo iterativo per la conversione da base 10 $\rightarrow$ a base r (parte frazionaria)

$$F_{10} = .d_{-1}r^{-1} + d_{-2}r^{-2} + \ldots + d_{-(m-1)}r^{-(m-1)} + d_{-m}r^{-m}$$

$$= .\frac{1}{r}(d_{-1} + \frac{1}{r}(d_{-2} + \frac{1}{r}(d_{-3} + \ldots + \frac{1}{r}(d_{-(m-1)} + \frac{1}{r}d_{-m}))\cdots))$$

Si moltiplica la parte frazionaria per la base di arrivo; la parte intera del risultato fornisce una cifra nella nuova base a partire dalla cifra più significativa.

Si itera il procedimento.

$$F_{10} \cdot r = d_{-1} + X_{-1}$$

$$X_{-1} \cdot r = d_{-2} + X_{-2}$$

$$\ldots \quad \ldots$$

$$X_{-(m-1)} \cdot r = d_{-m} + 0$$

Algoritmo iterativo per la conversione da base 10 $\rightarrow$ a base r (parte frazionaria)

Algoritmo:

1. Inizio;
2. Moltiplicare la parte frazionaria del numero decimale per la base di arrivo;
3. Separare parte intera e parte frazionaria;
4. La parte intera dà una cifra nella nuova base a partire dalla cifra più significativa ;
5. Se non ottengo 0 o non raggiungo la precisione richiesta torna a 2);
6. Fine.

L'algoritmo termina naturalmente se si raggiunge una iterazione in cui $X_i = 0$, oppure deve essere arrestato o quando si è raggiunta la precisione richiesta o quando si è raggiunto il numero di cifre prestabilito.

Conversione numeri frazionari

ESEMPIO:

Convertire il numero $(0.3125)_{10}$ in base 2.

$$\begin{aligned} 0.3125 \cdot 2 &= 0.625 \Rightarrow \text{bit } 0 \\ 0.625 \cdot 2 &= 1.25 \Rightarrow \text{bit } 1 \\ 0.25 \cdot 2 &= 0.5 \Rightarrow \text{bit } 0 \\ 0.5 \cdot 2 &= 1.0 \Rightarrow \text{bit } 1 \\ 0 \cdot 2 &= 0 \end{aligned}$$

$$(0.3125)_{10} \Rightarrow (0\,.\,0101)_2$$

Convertire il numero $(0.2)_{10}$ in base 2.

$$\begin{aligned} 0.2 \cdot 2 &= 0.4 \Rightarrow \text{bit } 0 \\ 0.4 \cdot 2 &= 0.8 \Rightarrow \text{bit } 0 \\ 0.8 \cdot 2 &= 1.6 \Rightarrow \text{bit } 1 \\ 0.6 \cdot 2 &= 1.2 \Rightarrow \text{bit } 1 \\ 0.2 \cdot 2 &= \cdots \end{aligned}$$

$$(0.2)_{10} \Rightarrow (0\,.\,\overline{0011})_2$$

Approssimazione numeri frazionari

In generale, un numero frazionario decimale con un limitato numero di cifre può corrispondere ad un numero frazionario in una base r qualunque con un numero infinito di cifre.

In questo caso, si deve arrestare l'algoritmo di conversione da base 10 a base r o quando si è raggiunta una precisione prestabilita o quando si è raggiunto il numero di cifre prestabilito (registro di conversione finito).

Approssimazione numeri frazionari

ESEMPIO:

Convertire $(0.3)_{10}$ in binario con una precisione inferiore a 10^{-2}.

Si applichi iterativamente l'algoritmo di conversione:

$$\begin{aligned}
0.3 \cdot 2 &= 0.6 \Rightarrow \text{bit } 0 \ [\text{peso } 0.5] \\
0.6 \cdot 2 &= 1.2 \Rightarrow \text{bit } 1 \ [\text{peso } 0.25] \\
0.2 \cdot 2 &= 0.4 \Rightarrow \text{bit } 0 \ [\text{peso } 0.125] \\
0.4 \cdot 2 &= 0.8 \Rightarrow \text{bit } 0 \ [\text{peso } 0.0625] \\
0.8 \cdot 2 &= 1.6 \Rightarrow \text{bit } 1 \ [\text{peso } 0.03125] \\
0.6 \cdot 2 &= 1.2 \Rightarrow \text{bit } 1 \ [\text{peso } 0.015625] \\
0.2 \cdot 2 &= 0.4 \Rightarrow \text{bit } 0 \ [\text{peso } 0.007815]
\end{aligned}$$

Per raggiungere un'approssimazione inferiore al limite prescritto si devono calcolare 7 cifre binarie, in quanto solo la settima cifra corrisponde a un peso inferiore a 10^{-2}.

$$(0.3)_{10} \Rightarrow (0\,.\,0100110)_2$$

Conversione numero frazionario completo

$$12.5_{(10)}$$

$12_{(10)}$ $\qquad$ $.5_{(10)}$

$\Downarrow$ $\qquad$ $\Downarrow$

$1100_{(2)}$ $\qquad$ $.1_{(2)}$

$$1100.1_{(2)}$$

$$1101.01_{(2)}$$

$1101_{(2)}$ $\qquad$ $.01_{(2)}$

$\Downarrow$ $\qquad$ $\Downarrow$

$13_{(10)}$ $\qquad$ $.25_{(10)}$

$$13\,.\,25_{(10)}$$

Sistema ottale

Base: $r = 8$

Simboli base: $d_i \in \{0, 1, 2, 3, 4, 5, 6, 7\}$

ESEMPIO:

$$N_8 = 32.4_{(8)} = 3 \cdot 8^1 + 2 \cdot 8^0 + 4 \cdot 8^{-1} = 26.5_{(10)}$$

Conversione binario $\Longleftrightarrow$ ottale

Un numero in base r può essere scritto nella seguente forma simbolica generale:

$$N = \cdots d_8 r^8 + d_7 r^7 + d_6 r^6 + d_5 r^5 + d_4 r^4 + d_3 r^3 + d_2 r^2 + d_1 r^1 + d_0 r^0$$

Si raggruppino i termini a 3 a 3:

$$N = \cdots (d_8 r^2 + d_7 r + d_6) r^6 + (d_5 r^2 + d_4 r + d_3) r^3 + (d_2 r^2 + d_1 r + d_0)$$

Si operi un cambiamento di base: $R = r^3$

Ne consegue che: $D = d'' r^2 + d' r + d$
è una cifra nella nuova base R, poichè vale la relazione:

$$0 \leq D \leq R - 1$$

Conversione da binario a ottale

Si raccolgono i bit a gruppi di 3, partendo dalla virgola:

- ◇ procedendo verso sinistra per la parte intera, completando eventualmente l'ultima tripletta con 0 non significativi;
- ◇ procedendo verso destra per la parte frazionaria, completando eventualmente l'ultima tripletta con 0 non significativi.

ESEMPIO:

$$\underbrace{\_1}_{}\underbrace{1\,0\,1}_{}\underbrace{1\,0\,0}_{}\,.\,\underbrace{1\,0\,1}_{}\underbrace{1\_}_{}\ {}_{(2)}$$

$$\Downarrow$$

$$331.54\ {}_{(8)}$$

Conversione da ottale a binario

Si rappresenta ciascuna cifra ottale in base 2 su 3 bit.

ESEMPIO:

$$327.35\ {}_{(8)}$$

$$\Downarrow$$

$$\overbrace{0\,1\,1}\,\overbrace{0\,1\,0}\,\overbrace{1\,1\,1}\,.\,\overbrace{0\,1\,1}\,\overbrace{1\,0\,1}\ {}_{(2)}$$

Sistema esadecimale

Base: $r = 16$

Simboli base:

$d_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$

ESEMPIO:

$$\begin{aligned} N_{(16)} &= E7A.C \\ &= E \cdot 16^2 + 7 \cdot 16 + A + C \cdot 16^{-1} \\ &= 14 \cdot 16^2 + 7 \cdot 16 + 10 + 12 \cdot 16^{-1} \\ &= 3706.75_{(10)} \end{aligned}$$

Conversione da binario a esadecimale

Si raccolgono i bit a gruppi di 4, partendo dalla virgola:

- ◇ procedendo verso sinistra per la parte intera, completando eventualmente l'ultima quadripletta con 0 non significativi;
- ◇ procedendo verso destra per la parte frazionaria, completando eventualmente l'ultima quadripletta con 0 non significativi.

ESEMPIO:

$$\underbrace{11}\underbrace{0111}\underbrace{1101} \cdot \underbrace{110}_{(2)}$$

$$\Downarrow$$

$$37D.C_{(16)}$$

Conversione da esadecimale a binario

Si rappresenta ciascuna cifra esadecimale in base 2 su 4 bit.

ESEMPIO:

$$A2E.B5_{(16)}$$

$$\Downarrow$$

$$\overbrace{1\,0\,1\,0}\,\overbrace{0\,0\,1\,0}\,\overbrace{1\,1\,1\,0}\,.\,\overbrace{1\,0\,1\,1}\,\overbrace{0\,1\,0\,1}_{(2)}$$

Addizione binaria

Siano x_i e y_i due variabili binarie, e sia s_i la loro somma e c_i il riporto. Vale la seguente regola:

x_i	y_i	s_i	c_i
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

ESEMPIO:

$$\begin{array}{cccc} 1 & 0 & 0 & 1 \; + \\ 0 & 0 & 1 & 1 \\ \hline 1 & 1 & 0 & 0 \end{array}$$

Addizione binaria

Se si deve tener conto anche dei possibili riporti che arrivano dalle cifre precedenti si adotta la seguente tabella:

x_i	y_i	c_{i-1}	s_i	c_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

ESEMPIO:

$$\begin{array}{cccccc} & 1 & 0 & 1 & 1 & + \\ & 0 & 1 & 1 & 1 & \\ \hline 1 & 0 & 0 & 1 & 0 & \end{array}$$

Sottrazione binaria

x_i	y_i	d_i	b_i
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

ESEMPIO:

$$\begin{array}{cccccc} 1 & 0 & 1 & 1 & - \\ 0 & 1 & 1 & 1 & \\ \hline 0 & 1 & 0 & 0 & \end{array}$$

Sottrazione binaria

Se si deve tener conto anche dei possibili prestiti concessi alle cifre precedenti si adotta la seguente tabella:

x_i	y_i	b_{i-1}	d_i	b_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

ESEMPIO:

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

Moltiplicazione binaria

Si applicano le stesse regole della moltiplicazione decimale.

ESEMPIO:

$$\begin{array}{ccccccccc} & & & 1 & 1 & 0 & 0 & \times \\ & & & 1 & 0 & 0 & 1 & \\ & & \hline & 1 & 1 & 0 & 0 & \\ & & 0 & 0 & 0 & 0 & - & \\ & 0 & 0 & 0 & 0 & - & & \\ 1 & 1 & 0 & 0 & - & & & \\ \hline 1 & 1 & 0 & 1 & 1 & 0 & 0 & \end{array}$$

Divisione binaria

Si applicano le stesse regole della divisione decimale.

ESEMPIO:

$$
\begin{array}{rrrrrrrcrrrr}
1 & 0 & 1 & 0 & 1 & 0 & 0 & : & 1 & 1 & 0 & 0 \\
\cline{9-12}
 & 1 & 1 & 0 & 0 & & & & & 1 & 1 & 1 \\
\cline{2-5}
 & 1 & 0 & 0 & 1 & 0 & & & & & & \\
 & & 1 & 1 & 0 & 0 & & & & & & \\
\cline{3-6}
 & & 0 & 1 & 1 & 0 & 0 & & & & & \\
 & & & 1 & 1 & 0 & 0 & & & & & \\
\cline{4-7}
 & & & & & & 0 & & & & &
\end{array}
$$

Somme e sottrazioni in base qualunque

Operazioni in base $r = 16$.

ESEMPIO:

$$
\begin{array}{rrrrr}
 & D & F & A & A \; + \\
 & B & B & B & B \\
\hline
1 & 9 & B & 6 & 5
\end{array}
\qquad\qquad
\begin{array}{rrrrr}
 & D & F & A & A \; - \\
 & B & B & B & B \\
\hline
 & 2 & 3 & E & F
\end{array}
$$

Operazioni in base $r = 8$.

ESEMPIO:

$$
\begin{array}{rrrrr}
 & 5 & 0 & 3 & 0 \; + \\
 & 2 & 7 & 7 & 6 \\
\hline
1 & 0 & 0 & 2 & 6
\end{array}
\qquad\qquad
\begin{array}{rrrrr}
 & 5 & 0 & 3 & 0 \; - \\
 & 2 & 7 & 7 & 6 \\
\hline
 & 2 & 0 & 3 & 2
\end{array}
$$

Numeri a Rappresentazione Finita

I numeri trattati dagli elaboratori hanno una lunghezza finita. Questo fatto comporta alcune conseguenze non presenti nell'aritmetica ordinaria:

◇ L'insieme dei numeri è limitato.

◇ La rappresentazione dei numeri è a precisione finita.

◇ Un insieme di numeri a precisione finita non forma un insieme chiuso rispetto alle ordinarie operazioni aritmetiche $+$, $-$, $*$, $/$.

In un insieme finito di rappresentazione, possono verificarsi tre forme di violazione:

⇒ Un numero è troppo grande (overflow).

⇒ Un numero è troppo piccolo (underflow).

⇒ Un numero non appartiene all'insieme dei numeri rappresentabili.

Il Riporto (carry)

Data una somma su numeri finiti a n bit, assume notevole importanza il riporto di peso 2^n (riporto finale), che andrebbe a collocarsi sulla posizione di peso $(n+1)$.

Viene normalmente indicato con C *(carry)* e memorizzato in una opportuna posizione di un registro di stato.

Nel caso di numeri assoluti, un riporto sull'ultima cifra $C = 1$ indica una condizione di *tracimazione* (o *overflow*); in tal caso il risultato della somma non è più esprimibile su n bit, in quanto non appartiene all'intervallo $[\,0,\, 2^n - 1\,]$.

Somme con tracimazione

Si debbano eseguire le seguenti somme su numeri binari a 6 bit.

ESEMPIO:

$$\begin{array}{cccccc} 0 & 1 & 0 & 1 & 0 & 1 \; + \\ 0 & 1 & 1 & 0 & 1 & 0 \\ \hline 1 & 0 & 1 & 1 & 1 & 1 \end{array}$$

$C = 0 \quad \Rightarrow \quad$ risultato corretto

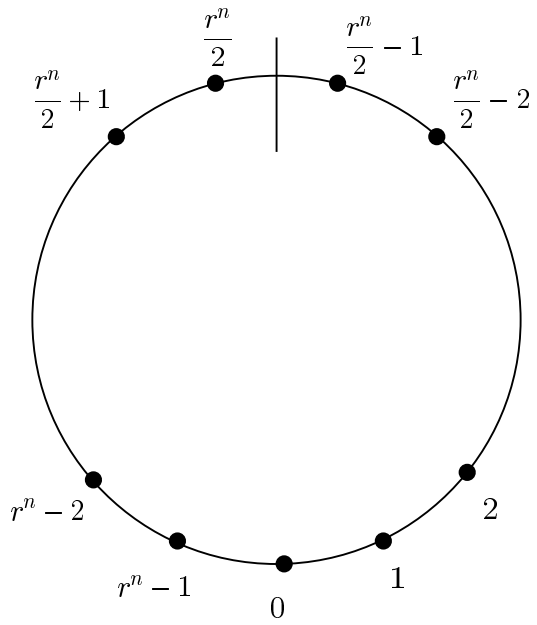
$$\begin{array}{cccccc|c} & 1 & 1 & 0 & 1 & 0 & 1 \; + \\ & 1 & 0 & 0 & 1 & 1 & 1 \\ \hline 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{array}$$

$C = 1 \quad \Rightarrow \quad$ Condizione di tracimazione: risultato errato

Numeri a Rappresentazione Finita

I numeri trattati dagli elaboratori hanno una lunghezza finita.

Per tener conto della lunghezza finita dei registri, si preferisce sostituire la rappresentazione lineare con una a cerchio.



RAPPRESENTAZIONE NUMERI BINARI RELATIVI

Anno Accademico 1996-1997

Numeri Relativi

Nella rappresentazione usuale si contraddistingue un numero positivo da uno negativo anteponendo al numero stesso un simbolo specifico.

$+ \ 3$	$+ \ 4328$
$- \ 3$	$- \ 4328$

I simboli $+$ e $-$ non vanno confusi con gli operatori di somma e sottrazione.

In un sistema di rappresentazione strettamente binario non possono essere introdotti altri simboli rispetto a 0 e 1.

Rappresentazione Numeri Relativi

Si può rappresentare un numero positivo o negativo in base r senza aggiungere nuovi simboli agli r simboli usuali della rappresentazione.

Sia dato il numero:

$$N_r = d_{n-1}\, d_{n-2} \cdots d_1\, d_0$$

Si può associare l'informazione del segno alla cifra più significativa, con la seguente convenzione:

$$d_{n-1} = \begin{cases} 0 & \text{per i numeri } \textit{positivi} \\ r-1 & \text{per i numeri } \textit{negativi} \end{cases}$$

Rappresentazione Numeri Relativi

Esistono tre modi fondamentali di rappresentazione di numeri relativi:

- ◇ *in Modulo e Segno (M&S)*;
- ◇ *in complemento alla base* $(\overline{\overline{N}})$;
- ◇ *in complemento alla base diminuita* $(\overline{N})$.

Rappresentazione in Modulo e Segno

Nella rappresentazione in *Modulo e Segno*, si associa il segno alla cifra più significativa e si riservano le restanti $(n-1)$ cifre per rappresentare il modulo del numero.

Sia dato il numero:

$$N_r = d_{n-1}\, d_{n-2} \cdots d_1\, d_0$$

- Il valore della cifra di segno è dato da:

$$d_{n-1} = \begin{cases} 0 & \text{per i numeri } \textit{positivi} \\ r-1 & \text{per i numeri } \textit{negativi} \end{cases}$$

- Le restanti $(n-1)$ cifre rappresentano il *modulo* del numero.

Rappresentazione in Modulo e Segno

Nella rappresentazione in *Modulo e Segno*, si adotta la seguente convenzione.

◇ *Numeri positivi:*

$$+N_r = 0\,|\,N\,| = 0\, d_{n-2} \cdots d_1\, d_0$$

◇ *Numeri negativi:*

$$-N_r = (r-1)\,|\,N\,| = (r-1)\, d_{n-2} \cdots d_1\, d_0$$

Numeri Binari in Modulo e Segno

Per i numeri binari in *Modulo e Segno*, bisogna prefissare la lunghezza in bit della rappresentazione, e per il segno consegue la seguente convenzione.

$$\text{bit di segno} = \begin{cases} 0 & \text{per i numeri } \textit{positivi} \\ 1 & \text{per i numeri } \textit{negativi} \end{cases}$$

ESEMPIO

$6_{10} \rightarrow \quad 1\ 1\ 0$ *Binario puro*

$+6_{10} \rightarrow 0\ 1\ 1\ 0$ *Binario positivo M&S*

$-6_{10} \rightarrow 1\ 1\ 1\ 0$ *Binario negativo M&S*

Per calcolare il valore vero di un numero bisogna separare il bit più significativo di segno dagli altri:

$$N = 1\,|\,0\ 0\ 1\ 1 = -[0\ 0\ 1\ 1] = -3_{10}$$

Numeri Binari in Modulo e Segno

ESEMPIO : numeri binari a 6 cifre.

$+12_{10} \rightarrow 0\ 0\ 1\ 1\ 0\ 0$

$-12_{10} \rightarrow 1\ 0\ 1\ 1\ 0\ 0$

ESEMPIO : numeri binari a 8 cifre.

$+71_{10} \rightarrow 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1$

$-71_{10} \rightarrow 1\ 1\ 0\ 0\ 0\ 1\ 1\ 1$

Intervalli di Rappresentazione in M&S

In una rappresentazione in *M&S* su n cifre sono rappresentabili i numeri compresi nell'intervallo:

$$-(r^{n-1} - 1) \leq N \leq +(r^{n-1} - 1)$$

Infatti, non considerando la cifra di segno, rimangono $(n - 1)$ cifre per il modulo, per cui il modulo massimo è:

$$N = r^{n-1} - 1$$

Nota: lo zero assume due diverse rappresentazioni:

$$+0 \rightarrow 0\ 0\ 0 \cdots 0\ 0$$

$$-0 \rightarrow 1\ 0\ 0 \cdots 0\ 0$$

Intervalli di Rappresentazione in M&S

Un numero binario in *M&S* su n cifre rappresenta i numeri compresi nell'intervallo:

$$-(2^{n-1} - 1) \leq N \leq +(2^{n-1} - 1)$$

Sono pertanto rappresentabili

$$2 \cdot (2^{n-1} - 1) + 1 = 2^n - 1$$

numeri diversi.

Poichè su n bit sono rappresentabili 2^n combinazioni si spreca la combinazione legata alla doppia rappresentazione dello zero.

Corrispondenza decimale binario in M&S

Valori rappresentati su 5 bit in *M&S*.

Decimale Rappresentato	Binario in *M&S*	Valore Rappresentazione
+15	0 1 1 1 1	15
+14	0 1 1 1 0	14
+13	0 1 1 0 1	13
...	...	...
...	...	...
+2	0 0 0 1 0	2
+1	0 0 0 0 1	1
+0	0 0 0 0 0	0
−0	1 0 0 0 0	16
−1	1 0 0 0 1	17
−2	1 0 0 1 0	18
...	...	...
...	...	...
−13	1 1 1 0 1	29
−14	1 1 1 1 0	30
−15	1 1 1 1 1	31

Calcolo valore vero in M&S

Il valore decimale di un numero binario in *M&S* su n cifre è dato da:

$$V(N) = (-1)^{d_{n-1}} \cdot \sum_{i=0}^{n-2} d_i \, r^i$$

ESEMPIO : numeri binari a 6 cifre.

$\diamond$ $N = 011001$

$$V(N) = (-1)^0 \cdot (1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^0)$$

$$= +25_{10}$$

$\diamond$ $N = 111001$

$$V(N) = (-1)^1 \cdot (1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^0)$$

$$= -25_{10}$$

Negazione di un numero in M&S

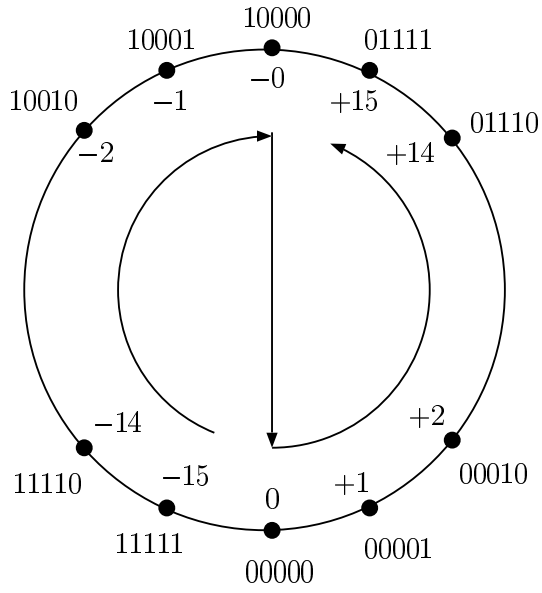
L'opposto di un numero si calcola:

1. Lasciando invariato il modulo;
2. Invertendo la codifica del segno (nel sistema binario, invertendo il primo bit).

ESEMPIO

$$\begin{aligned} \mathbf{0}11001 &\equiv +25_{10} \\ \Downarrow & \\ \mathbf{1}11001 &\equiv -25_{10} \\ \Downarrow & \\ \mathbf{0}11001 &\equiv +25_{10} \end{aligned}$$

Cerchio delle rappresentazioni in M&S



Cerchio delle rappresentazioni in M&S

Dalla rapresentazione sul cerchio dei numeri binari in *M&S* su 5 bit si possono trarre le seguenti osservazioni:

1. Esistono due zeri distanti fra loro una semicirconferenza;

2. Andando da 00000 (0_{10}) a 01111 (15_{10}) vengono rappresentati concordemente i numeri da +0 a +15;

3. Andando da 10000 (16_{10}) a 11111 (31_{10}) vengono rappresentati discordemente (in ordine decrescente) i numeri da −0 a −15;

La rappresentazione in *M&S*, all'interno degli elaboratori, semplifica le operazioni di moltiplicazione e divisione, ma complica notevolmente quelle di addizione e sottrazione.

Complemento alla Base

Il concetto di complemento alla base è intimamente connesso alla rappresentazione di un numero (in qualunque base r) su un numero limitato di cifre.

Per poter definire il complemento alla base di un numero si deve prefissare il numero di cifre k della sua rappresentazione.

Data una base r, in un registro a k posizioni si può rappresentare qualunque numero $MOD\,(\,r^k\,)$.

ESEMPIO: contatore decimale a k cifre.

Se il contatore segna il numero N, può essere pensato come rappresentativo di qualunque numero N':

$$N' = M \cdot r^k + N$$

con M intero.

Aritmetica in Complemento alla Base

ESEMPIO: contatore decimale a 4 cifre.

Per fare l'operazione:

$$\begin{array}{r} 8\ 7\ 2\ 5\ - \\ 3\ 5\ 1\ 2 \\ \hline 5\ 2\ 1\ 3 \end{array}$$

Si può calcolare:

$$\begin{array}{r} 1\ 0\ 0\ 0\ 0\ - \\ 3\ 5\ 1\ 2 \\ \hline 6\ 4\ 8\ 8 \end{array}$$

E successivamente calcolare:

$$\begin{array}{r} 8\ 7\ 2\ 5\ + \\ 6\ 4\ 8\ 8 \\ \hline 1\ 5\ 2\ 1\ 3 \end{array}$$

Che $MOD\,(10^4)$ fornisce il valore richiesto.

Complemento alla Base

Dato un numero N di k cifre in base r, si definisce complemento alla base di N il numero:

$$\overline{\overline{N}} = r^k - N$$

ESEMPIO:

Con: $r = 10 \;\; ; \;\; k = 2 \;\; ; \;\; N = 64$

$$\overline{\overline{N}} = 10^2 - 64 = 36$$

Con: $r = 2 \;\; ; \;\; k = 5 \;\; ; \;\; N = 01011$

$$\overline{\overline{N}} = \begin{array}{r} 1\ 0\ 0\ 0\ 0\ 0\ - \\ 0\ 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 1\ 0\ 1 \end{array}$$

Complemento alla Base Diminuita

Dato un numero N di k cifre in base r, si definisce complemento alla base diminuita di N il numero:

$$\overline{N} = (r^k - 1) - N$$

ESEMPIO:

Con: $r = 10 \;\; ; \;\; k = 2 \;\; ; \;\; N = 64$

$$\overline{N} = 99 - 64 = 35$$

Con: $r = 2 \;\; ; \;\; k = 5 \;\; ; \;\; N = 01011$

$$\overline{N} = \begin{array}{r} 1\;1\;1\;1\;1\;- \\ 0\;1\;0\;1\;1 \\ \hline 1\;0\;1\;0\;0 \end{array}$$

Complemento alla Base Diminuita

Il complemento alla base diminuita di un numero in base r si ottiene calcolando cifra per cifra il complemento a $(r - 1)$.

NOTA:

Il complemento alla base diminuita di un numero binario è detto complemento a 1.

Il complemento a 1 di un numero binario si ottiene complementando il numero bit a bit.

Relazione fra Numeri in Complemento

Dato un numero $N_{(r)}$ di k cifre in base r, esiste la seguente relazione fra il complemento alla base $\overline{\overline{N}}_{(r)}$ e il complemento alla base diminuita $\overline{N}_{(r)}$.

$$\overline{\overline{N}}_{(r)} = r^k - N_{(r)}$$

$$\overline{N}_{(r)} = (r^k - 1) - N_{(r)} = r^k - N_{(r)} - 1$$

$$\overline{\overline{N}}_{(r)} = \overline{N}_{(r)} + 1$$

Vale la relazione:

$$\overline{\overline{(\overline{N}_{(r)})}} = r^k - \overline{\overline{N}}_{(r)} = r^k - (r^k - N_{(r)}) = N_{(r)}$$

$$\overline{(\overline{N}_{(r)})} = N_{(r)}$$

Rappresentazione dei Numeri Interi Relativi in Complemento alla Base

Utilizzando il concetto di *complemento alla base*, è possibile introdurre una rappresentazione dei numeri interi relativi, detta **rappresentazione in complemento alla base**.

La rappresentazione in complemento alla base adotta la seguente convenzione:

◇ *Numeri Positivi*:

$$+N \Longrightarrow 0\,|\,N\,|$$

◇ *Numeri Negativi*:

$$-N \Longrightarrow (r-1)\overline{|\,N\,|}$$

Dove $\overline{|\,N\,|}$ rappresenta il complemento alla base del modulo del numero.

Rappresentazione dei Numeri Interi Relativi in Complemento alla Base

Dalla relazione

$$\overline{\overline{N}}_{(r)} = \overline{N}_{(r)} + 1$$

deriva la seguente *regola pratica*.

Sia dato il numero positivo su n cifre:

$$+N_r = 0\ d_{n-2} \cdots d_1\, d_0$$

Il suo complemento alla base è dato da:

$$\overline{\overline{N}}_{(r)} = (r-1)\ \overline{d}_{n-2}\, \overline{d}_{n-3} \cdots \overline{d}_1\, \overline{d}_0 + 1$$

dove:

$$\overline{d}_i = (r-1) - d_i$$

Rappresentazione dei Numeri Interi Relativi in Complemento a 2

Vale la seguente assunzione.

◇ *Numeri positivi:*

Sono rappresentati dal loro modulo e hanno il bit più significativo di segno a 0 (e quindi coincidono con la rappresentazione in *M&S*).

$$+N \implies 0\,|\,N\,|$$

◇ *Numeri negativi:*

Sono rappresentati dal complemento a due del corrispondente numero positivo, segno compreso.

I numeri negativi hanno il bit più significativo di segno sempre a 1.

$$-N \implies 1\overline{|\,N\,|} = \overline{|\,+N\,|}$$

Regole per la complementazione a 2

Da decimale a complemento a 2.

Se $N \geq 0 \Rightarrow 0\,|$ binario puro

Se $N < 0 \Rightarrow$ 1) Prendere binario puro
segno compreso
2) Complementare bit a bit
3) Sommare $+1$

Da complemento a 2 a decimale.

Se $N \geq 0 \Rightarrow 0\,|$ binario puro

Se $N < 0 \Rightarrow$ 1) Prendere binario
segno compreso
2) Complementare bit a bit
3) Sommare $+1$
4) Convertire numero binario
5) Aggiungere il segno $-$

Conversione decimale $\Rightarrow$ complemento a 2

Applicando la precedente *regola pratica* ai numeri binari, consegue che il complemento a 2 di un numero binario si ottiene complementando (in senso logico) ciascuna cifra, compresa quella rappresentativa del segno, e aggiungendo 1 al risultato.

ESEMPIO

Con: $r = 2 \quad ; \; k = 6$

$$+21_{10} \Longrightarrow 0\ 1\ 0\ 1\ 0\ 1$$

$$-21_{10} \Longrightarrow \begin{array}{r} 1\ 0\ 1\ 0\ 1\ 0\ + \\ \hline 1\ 0\ 1\ 0\ 1\ 1 \end{array} \quad \begin{array}{r} \\ 1 \end{array}$$

Verifica:

$$\overline{\overline{N}} = \begin{array}{r} 1\ 0\ 0\ 0\ 0\ 0\ 0\ - \\ 0\ 1\ 0\ 1\ 0\ 1 \\ \hline 1\ 0\ 1\ 0\ 1\ 1 \end{array}$$

Conversione complemento a 2 $\Rightarrow$ decimale

ESEMPIO

Con: $r = 2 \quad ; \quad k = 6$

$$0\ 1\ 1\ 1\ 0\ 1 \qquad \Longrightarrow \quad +29$$

$$1\ 1\ 0\ 0\ 1\ 1 \qquad \Longrightarrow \quad -13$$

$$\begin{array}{r} 0\ 0\ 1\ 1\ 0\ 0\ + \\ \hline 0\ 0\ 1\ 1\ 0\ 1 \end{array} \qquad (\Longrightarrow \ +13)$$

Negazione binario in complemento a 2

La negazione di un numero binario N si ottiene facendo il complemento a 2 del numero segno compreso.

ESEMPIO

$$0\ 1\ 1\ 1\ 0\ 1 \qquad \Longrightarrow \qquad +29$$

$$\Downarrow \ \textit{inversione}$$

$$\begin{array}{r} 1\ 0\ 0\ 0\ 1\ 0\ + \\ \hline 1\ 0\ 0\ 0\ 1\ 1 \end{array} \qquad \Longrightarrow \qquad -29$$

$$1\ 1\ 0\ 1\ 1\ 0 \qquad \Longrightarrow \qquad -10$$

$$\Downarrow \ \textit{inversione}$$

$$\begin{array}{r} 0\ 0\ 1\ 0\ 0\ 1\ + \\ \hline 0\ 0\ 1\ 0\ 1\ 0 \end{array} \qquad \Longrightarrow \qquad +10$$

Regola per la complementazione a 2

Ulteriore regola pratica per il calcolo del complemento a 2.

Partendo dalla cifra meno significativa e procedendo verso sinistra, lasciare inalterati tutti gli zeri a destra del primo 1, lasciare inalterato tale 1, e complementare (in senso logico) i restanti bit fino al più significativo.

ESEMPIO

0 1 1 0 0
1 0 1 0 0

1 1 0 1 0
0 0 1 1 0

Intervallo di rappresentazione in complemento a 2

Nel caso di numeri binari su n bit, si ha:

$$-2^{n-1} \leq N \leq 2^{n-1} - 1$$

Vengono utilizzate tutte le possibili 2^n combinazioni, con una sola rappresentazione per lo zero, e un numero negativo in più rispetto ai positivi.

Calcolo valore vero in complemento a 2

Nel caso di numeri binari su n bit, il valore vero di un numero $N = d_{n-1}\, d_{n-2} \cdots d_0$ rappresentato in *complemento a due* può essere calcolato con la seguente espressione:

$$V(N) = -\, d_{n-1}\, 2^{n-1} + \sum_{i=0}^{n-2} d_i\, 2^i$$

Al bit di segno viene pertanto attribuito un peso pari a 2^{n-1}.

Dimostrazione

- Se $N \geq 0 \;\Rightarrow\; d_{n-1} = 0$; da cui:

$$V(N) = \sum_{i=0}^{n-2} d_i\, 2^i$$

- Se $N < 0 \;\Rightarrow\; d_{n-1} = 1$; da cui:

$$\begin{aligned} V(N) &= -\, 2^n + \overline{\overline{N}} \\ &= -\, 2^n + 2^{n-1} + \sum_{i=0}^{n-2} d_i\, 2^i \\ &= -\, 2^{n-1} + \sum_{i=0}^{n-2} d_i\, 2^i \end{aligned}$$

Calcolo valore vero in complemento a 2

ESEMPIO

◇ $N = 0\,0\,1\,1$

$$\begin{aligned} V(N) &= -0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ &= -0 + 2 + 1 \;= +3_{10} \end{aligned}$$

◇ $N = 1\,0\,1\,1$

$$\begin{aligned} V(N) &= -1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ &= -8 + 2 + 1 \;= -5_{10} \end{aligned}$$

◇ $N = 1\,0\,0\,0\,0$

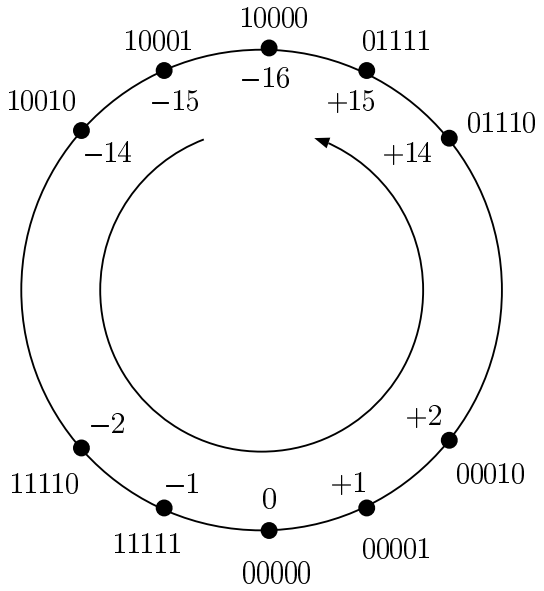
$$\begin{aligned} V(N) &= -1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 \\ &= -\, 16_{10} \end{aligned}$$

Corrispondenza decimale binario in complemento a 2

Valori rappresentati su 5 bit in complemento a due.

Decimale Rappresentato	Binario in Complemento a 2	Valore Rappresentazione
+15	0 1 1 1 1	15
+14	0 1 1 1 0	14
+13	0 1 1 0 1	13
...	...	...
...	...	...
+2	0 0 0 1 0	2
+1	0 0 0 0 1	1
0	0 0 0 0 0	0
−1	1 1 1 1 1	31
−2	1 1 1 1 0	30
...	...	...
...	...	...
−14	1 0 0 1 0	18
−15	1 0 0 0 1	17
−16	1 0 0 0 0	16

Cerchio delle rappresentazioni in Complemento a due



Vantaggi Rappresentazione in complemento a 2

Tenendo presente che si opera su registri di dimensione finita k, e quindi con una rappresentazione $MOD(r^k)$, le operazioni di sottrazione possono ridursi a operazioni di somma complementando a due il sottraendo.

Dalla definizione di complemento alla base si deduce che:

$$-M_{(r)} = \overline{\overline{M}}_{(r)} - r^k$$

Da cui:

$$N_{(r)} - M_{(r)} = N_{(r)} + \overline{\overline{M}}_{(r)} - r^k$$

Rappresentazione dei Numeri Interi Relativi in Complemento a 1

Vale la seguente assunzione.

◇ *Numeri positivi:*

Sono rappresentati dal loro modulo e hanno il bit più significativo di segno a 0 (e quindi coincidono con la rappresentazione in *M&S*).

$$+N \Longrightarrow 0\,|\,N\,|$$

◇ *Numeri negativi:*

Sono rappresentati dal complemento a 1 del corrispondente numero positivo, segno compreso.

I numeri negativi hanno il bit più significativo di segno sempre a 1.

$$-N \Longrightarrow 1\overline{|\,N\,|} = \overline{|\,+N\,|}$$

Rappresentazione dei Numeri Interi Relativi in Complemento a 1

Sia dato il numero positivo su n cifre:

$$+N_r = 0\ d_{n-2} \cdots d_1\, d_0$$

Vale la seguente *regola pratica*.

Il complemento alla base diminuita è dato da:

$$\overline{N}_{(r)} = (r-1)\ \overline{d}_{n-2}\, \overline{d}_{n-3} \cdots \overline{d}_1\, \overline{d}_0$$

dove:

$$\overline{d}_i = (r-1) - d_i$$

Regola per la complementazione a 1

La complementazione a 1 di un numero binario si ottiene complementando singolarmente ogni bit del numero.

ESEMPIO

1 0 1 1 0 0 1

⇓

0 1 0 0 1 1 0

0 1 1 0 1 0 0

⇓

1 0 0 1 0 1 1

Intervallo di rappresentazione in complemento a 1

Coincide con l'intervallo di rappresentazione dei numeri in *M&S*.

Nel caso di numeri binari su n bit, si ha:

$$-(2^{n-1} - 1) \leq N \leq 2^{n-1} - 1$$

Anche in questo caso si ha una doppia rappresentazione per lo zero:

$$+0 = 0\,0\,0 \cdots 0$$

$$-0 = 1\,1\,1 \cdots 1$$

Calcolo valore vero in complemento a 1

Nel caso di numeri binari su n bit, il valore vero di un numero $N = d_{n-1}\,d_{n-2} \cdots d_0$ rappresentato in *complemento a uno* può essere calcolato con la seguente espressione:

$$V(N) = -\,d_{n-1}\,(2^{n-1} - 1) + \sum_{i=0}^{n-2} d_i\,2^i$$

Al bit di segno viene pertanto attribuito un peso pari a $(2^{n-1} - 1)$.

Dimostrazione

- Se $N \geq 0 \;\Rightarrow\; d_{n-1} = 0$; da cui:

$$V(N) = \sum_{i=0}^{n-2} d_i\,2^i$$

- Se $N < 0 \;\Rightarrow\; d_{n-1} = 1$; da cui:

$$\begin{aligned} V(N) &= -(2^n - 1) + \overline{N} \\ &= -2^n + 1 + 2^{n-1} + \sum_{i=0}^{n-2} d_i\,2^i \\ &= -2^{n-1} + 1 + \sum_{i=0}^{n-2} d_i\,2^i \end{aligned}$$

Calcolo valore vero in complemento a 1

ESEMPIO

◇ $N = 0\,1\,0\,1$

$$V(N) = -0 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

$$= +5_{10}$$

◇ $N = 1\,1\,0\,1$

$$V(N) = -1 \cdot (2^3 - 1) + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

$$= -7 + 4 + 1 = -2_{10}$$

◇ $N = 1\,0\,0\,0\,0$

$$V(N) = -1 \cdot (2^4 - 1) + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$$

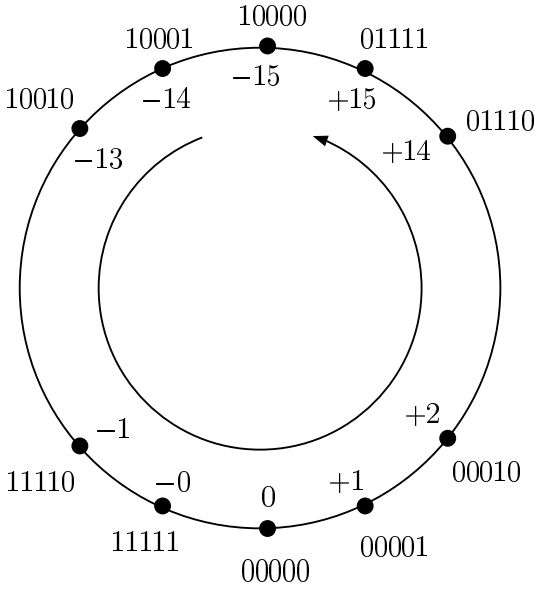
$$= -15_{10}$$

Corrispondenza decimale binario in complemento a 1

Valori rappresentati su 5 bit in complemento a uno.

Decimale Rappresentato	Binario in Complemento a 1	Valore Rappresentazione
+15	0 1 1 1 1	15
+14	0 1 1 1 0	14
+13	0 1 1 0 1	13
...	...	...
...	...	...
+2	0 0 0 1 0	2
+1	0 0 0 0 1	1
0	0 0 0 0 0	0
−0	1 1 1 1 1	31
−1	1 1 1 1 0	30
−2	1 1 1 0 1	29
...	...	...
...	...	...
−13	1 0 0 1 0	18
−14	1 0 0 0 1	17
−15	1 0 0 0 0	16

Cerchio delle rappresentazioni in Complemento a uno



Confronto Rappresentazioni

Confronto fra i valori rappresentati su 5 bit nelle tre rappresentazioni introdotte.

Binario	*M& S*	*Compl 2*	*Compl 1*
0 0 0 0 0	0	0	0
0 0 0 0 1	1	1	1
0 0 0 1 0	2	2	2
...	...	...	...
...	...	...	...
0 1 1 0 1	13	13	13
0 1 1 1 0	14	14	14
0 1 1 1 1	15	15	15
1 0 0 0 0	−0	−16	−15
1 0 0 0 1	−1	−15	−14
1 0 0 1 0	−2	−14	−13
...	...	...	...
...	...	...	...
1 1 1 0 1	−13	−3	−2
1 1 1 1 0	−14	−2	−1
1 1 1 1 1	−15	−1	−0

SOMME E SOTTRAZIONI SU NUMERI BINARI RELATIVI

Anno Accademico 1996-1997

Addizione numeri binari in *M&S*

Si calcolando separatamente il modulo e il segno del risultato secondo la seguente tabella.

$S(X)$	$S(Y)$	$S(R)$	$\mid R \mid$
0	0	0	$\mid X \mid + \mid Y \mid$
1	1	1	$\mid X \mid + \mid Y \mid$
0	1	$\begin{cases} 0 \text{ se } \mid X \mid \geq \mid Y \mid \\ 1 \text{ se } \mid X \mid < \mid Y \mid \end{cases}$	$\begin{cases} \mid X \mid - \mid Y \mid \\ \mid Y \mid - \mid X \mid \end{cases}$
1	0	$\begin{cases} 0 \text{ se } \mid X \mid \leq \mid Y \mid \\ 1 \text{ se } \mid X \mid > \mid Y \mid \end{cases}$	$\begin{cases} \mid Y \mid - \mid X \mid \\ \mid X \mid - \mid Y \mid \end{cases}$

Addizione numeri binari in *M&S*

Eseguire la seguente addizione con $n = 6$ bit.

ESEMPIO:

$$X = 110100 \quad \Rightarrow \quad -20_{10}$$

$$Y = 001011 \quad \Rightarrow \quad +11_{10}$$

$$\left.\begin{array}{r} S(X) = 1,\ S(Y) = 0 \\ |\,X\,| > |\,Y\,| \end{array}\right\} \Rightarrow \begin{cases} S(R) = 1 \\ |\,R\,| = |\,X\,| - |\,Y\,| \end{cases}$$

$$\begin{array}{lcl} 1\,0\,1\,0\,0 - & \longrightarrow & |\,X\,| \\ \underline{0\,1\,0\,1\,1} & \longrightarrow & |\,Y\,| \\ 0\,1\,0\,0\,1 & \longrightarrow & |\,R\,| \end{array}$$

$$R = 101001 \quad \Rightarrow \quad -9_{10}$$

Addizione numeri binari in *M&S*

Eseguire la seguente addizione con $n = 6$ bit.

ESEMPIO:

$$X = 100111 \quad \Rightarrow \quad -7_{10}$$

$$Y = 110010 \quad \Rightarrow \quad -18_{10}$$

$$\left.\begin{array}{r} S(X) = 1,\ S(Y) = 1 \end{array}\right\} \Rightarrow \begin{cases} S(R) = 1 \\ |\,R\,| = |\,X\,| + |\,Y\,| \end{cases}$$

$$\begin{array}{lcl} 0\,0\,1\,1\,1 + & \longrightarrow & |\,X\,| \\ \underline{1\,0\,0\,1\,0} & \longrightarrow & |\,Y\,| \\ 1\,1\,0\,0\,1 & \longrightarrow & |\,R\,| \end{array}$$

$$R = 111001 \quad \Rightarrow \quad -25_{10}$$

Addizione numeri binari in *M&S*

Occorre prestare attenzione al fatto che nelle operazioni sui moduli degli addendi non si verifichino condizioni di tracimazione *(overflow)*.

La tracimazione si individua dal fatto che, nella somma fra i soli moduli, si ottiene un bit di riporto dalla cifra più significativa.

ESEMPIO ($n = 6$ bit):

$$X = 010111 \quad \Rightarrow \quad +23_{10}$$
$$Y = 001110 \quad \Rightarrow \quad +14_{10}$$

$$S(X)=0,\ S(Y)=0\ \} \ \Rightarrow \ \begin{cases} S(R) = 0 \\ \mid R \mid = \mid X \mid + \mid Y \mid \end{cases}$$

$$\begin{array}{rcl} 1\,0\,1\,1\,1 + & \longrightarrow & \mid X \mid \\ 0\,1\,1\,1\,0 & \longrightarrow & \mid Y \mid \\ \hline \mathbf{1}\,0\,0\,1\,0\,1 & \longrightarrow & \mid R \mid \end{array}$$

$C = 1 \quad \Rightarrow$ *overflow*: c'è stato riporto dal bit più significativo del modulo.

Sottrazione numeri binari in *M&S*

Si opera come nel caso della somma, calcolando separatamente il modulo e il segno del risultato secondo la seguente tabella.

$S(X)$	$S(Y)$	$S(R)$	$\mid R \mid$
0	0	$\begin{cases} 0 \text{ se } \mid X \mid \geq \mid Y \mid \\ 1 \text{ se } \mid X \mid < \mid Y \mid \end{cases}$	$\begin{cases} \mid X \mid - \mid Y \mid \\ \mid Y \mid - \mid X \mid \end{cases}$
0	1	0	$\mid X \mid + \mid Y \mid$
1	0	1	$\mid X \mid + \mid Y \mid$
1	1	$\begin{cases} 0 \text{ se } \mid X \mid \leq \mid Y \mid \\ 1 \text{ se } \mid X \mid > \mid Y \mid \end{cases}$	$\begin{cases} \mid Y \mid - \mid X \mid \\ \mid X \mid - \mid Y \mid \end{cases}$

Sottrazione numeri binari in *M&S*

Eseguire la seguente sottrazione con $n = 6$ bit.

ESEMPIO:

$$X = 110100 \quad \Rightarrow \quad -20_{10}$$

$$Y = 111011 \quad \Rightarrow \quad -27_{10}$$

$$\left.\begin{array}{r} S(X) = 1,\ S(Y) = 1 \\ |\,X\,| \leq |\,Y\,| \end{array}\right\} \Rightarrow \left\{\begin{array}{l} S(R) = 0 \\ |\,R\,| = |\,Y\,| - |\,X\,| \end{array}\right.$$

$$\begin{array}{lll} 1\ 1\ 0\ 1\ 1\ - & \longrightarrow & |\,Y\,| \\ \underline{1\ 0\ 1\ 0\ 0} & \longrightarrow & |\,X\,| \\ 0\ 0\ 1\ 1\ 1 & \longrightarrow & |\,R\,| \end{array}$$

$$R = 000111 \quad \Rightarrow \quad +7_{10}$$

Operazioni numeri binari in complemento a 2

Le operazioni di somma e sottrazione nella rappresentazione in complemento a 2 sono particolarmente agevoli, per le seguenti condizioni:

- ◇ Sia la somma che la sottrazione si riportano ad una oparazione di somma (eventualmente complementando un operando).

- ◇ Il bit di segno si tratta con le stesse regole con cui si trattano tutti gli altri bit del numero.

Addizione numeri binari in complemento a 2

Si applicano a tutti i bit le regole per i binari puri.

Contrariamente a quanto avveniva per i binari puri il bit di carry non fornisce alcuna informazione sulla correttezza del risultato.

ESEMPIO ($n = 6$ bit):

0 1 1 1 0 1	+	$\longrightarrow$	X	$+29_{(10)}$
1 0 1 1 0 0	=	$\longrightarrow$	Y	$-20_{(10)}$
1 0 0 1 0 0 1		$\longrightarrow$	R	$+9_{(10)}$

Il risultato è corretto anche se il riporto dalla cifra più significativa vale $C_n = 1$.

0 1 1 0 0 1	+	$\longrightarrow$	X	$+25_{(10)}$
0 1 0 1 0 0	=	$\longrightarrow$	Y	$+20_{(10)}$
0 1 0 1 1 0 1		$\longrightarrow$	R	$-19_{(10)}$

Il risultato non è corretto anche se il riporto dalla cifra più significativa vale $C_n = 0$.

Addizione numeri binari in complemento a 2

Condizione di tracimazione (overflow)

Si verifica una condizione di tracimazione quando il risultato dell'addizione non appartiene all'intervallo dei numeri ammessi nella rappresentazione a n bit:

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

Regola pratica

Si può riconoscere la condizione di tracimazione attraverso due possibili regole:

- ◇ I riporti generati in corrispondenza dei bit di peso n e di peso $(n-1)$ sono diversi.
- ◇ Gli addendi sono concordi e il bit di segno del risultato è diverso dal bit di segno degli addendi.

Addizione numeri binari in complemento a 2

ESEMPIO Verifica Tracimazione:

$n = 6$ bit

$$\begin{array}{rcll} 0\,1\,1\,1\,0\,1 + & \longrightarrow & X & +29_{(10)} \\ 1\,0\,1\,1\,0\,0 = & \longrightarrow & Y & -20_{(10)} \\ \hline \mathbf{1}\,0\,0\,1\,0\,0\,1 & \longrightarrow & R & +9_{(10)} \end{array}$$

Poichè $C_n = 1$ e $C_{n-1} = 1$ il risultato è corretto (l'operazione non può essere in *overflow* perchè il segno degli addendi è discorde).

$$\begin{array}{rcll} 0\,1\,1\,0\,0\,1 + & \longrightarrow & X & +25_{(10)} \\ 0\,1\,0\,1\,0\,0 = & \longrightarrow & Y & +20_{(10)} \\ \hline \mathbf{0}\,1\,0\,1\,1\,0\,1 & \longrightarrow & R & -19_{(10)} \end{array}$$

Poichè $C_n = 0$ e $C_{n-1} = 1$ il risultato è errato (esistono potenziali condizioni di *overflow* perchè il segno degli addendi è concorde).

Sottrazione numeri binari in complemento a 2

Viene ricondotta ad un operazione di somma ($MOD(2^n)$) addizionando al minuendo il complemento a 2 del sottraendo.

$$\begin{aligned} R &= X - Y \\ &= X + (-Y) \\ &= X + \overline{\overline{Y}} - 2^n \end{aligned}$$

Sottrazione numeri binari in complemento a 2

ESEMPIO ($n = 6$ bit):

Dati:

$$X = 010010 \quad \Rightarrow \quad +18_{10}$$

$$Y = 001011 \quad \Rightarrow \quad +11_{10}$$

Calcolare: $X - Y$

$$\begin{array}{rcl} 0\,1\,0\,0\,1\,0\ + & \longrightarrow & X \\ 1\,1\,0\,1\,0\,1\ = & \longrightarrow & \overline{\overline{Y}} \\ \hline \mathbf{1}\,0\,0\,0\,1\,1\,1 & \longrightarrow & +7_{10} \end{array}$$

Il risultato è corretto.

Sottrazione numeri binari in complemento a 2

ESEMPIO ($n = 6$ bit):

Dati:

$$X = 010010 \quad \Rightarrow \quad +18_{10}$$

$$Y = 001011 \quad \Rightarrow \quad +11_{10}$$

Calcolare: $-X + Y$

$$\begin{array}{rcl} 1\,0\,1\,1\,1\,0\ + & \longrightarrow & \overline{\overline{X}} \\ 0\,0\,1\,0\,1\,1\ = & \longrightarrow & Y \\ \hline 1\,1\,1\,0\,0\,1 & \longrightarrow & -7_{10} \end{array}$$

Il risultato è corretto.

Operazioni su numeri binari in complemento a 2

Osservazioni conclusive.

◇ Sommando due numeri positivi si ha sempre *carry* $= 0$ e si può avere o meno *overflow*.

◇ Sommando due numeri negativi si ha sempre *carry* $= 1$ e si può avere o meno *overflow*.

◇ Sommando due numeri di segno discorde non si ha mai *overflow* e si può avere ogni valore per il *carry*.