SISTEMA DI RAPPRESENTAZIONE BINARIA DEI NUMERI

E. Giordani

LEMS- Laboratorio Elettronico per la Musica Sperimentale
Conservatorio di Musica G. Rossini- Pesaro

Introduzione

Tutti i *calcolatori elettronici digitali* operano su base numerica e ciò significa che la rappresentazione interna di tutte le informazioni è di tipo numerica, sia che l'informazione sia propriamente un numero o qualche cosa di diverso. La base di rappresentazione dei dati è di tipo binario cioè attraverso un sistema di numerazione che è costituito da due soli simboli, a differenza del sistema decimale che fa uso invece di dieci simboli.

Questa circostanza deriva dal fatto che le macchine elettroniche numeriche sono costituite da unità fisiche elementari che sono in grado di operare con soli due livelli stabili. In origine queste unità elementari erano dei semplici *relais* , vale a dire interruttori comandati elettricamente in maniera elementare: assenza di corrente / interruttore aperto, presenza di corrente/ interruttore chiuso. Nel giro di poco tempo questa tecnologia primordiale è stata soppiantata dapprima dalle valvole termoioniche, quindi successivamente dai transistor e in tempi relativamente recenti dai circuiti integrati a larga scala di integrazione (VLSI), capaci cioè di contenere al loro interno centinaia di migliaia di transistor. Sebbene la tecnologia costruttiva sia profondamente mutata, il principio di rappresentazione dei dati è rimasto sostanzialmente invariato.

Rappresentazione binaria

Esistono due sistemi di rappresentazione per esprimere il valore di un numero: sistemi *non posizionali* e sistemi *posizionali*. Ad esempio, la numerazione romana appartiene al primo tipo mentre la tradizionale numerazione decimale (di origine araba) appartiene alla seconda tipologia. In questo ultimo caso, ogni cifra occupa una posizione alla quale è associato un peso e il valore del numero si ottiene sommando i prodotti di ogni cifra per il peso associato. A sua volta, ogni peso è dato da una quantità ottenuta elevando a potenze intere e progressive la base del sistema: nella numerazione decimale i pesi sono formati da termini del tipo 10^N dove N=0,1,2,3,….. mentre le cifre vanno da 0 a 9 (in totale 10 cifre), cioè tante quante ne prevede la base del sistema

In modo del tutto corrispondente, il sistema posizionale che utilizza solo due cifre è detto sistema binario, ove i pesi sono formati da termini del tipo 2^N dove N=0,1,2,3,…..

Ci occuperemo quindi di esporre alcuni elementi fondamentali dell' aritmetica binaria a partire dalla rappresentazione dei numeri a base 2.

L'unità minima.di rappresentazione numerica binaria è data da una *variabile* che può assumere due soli stati stabili: zero (0) / uno (l). In genere i due stati stabili corrispondono a due diversi livelli stabili di tensione (ad esempio 0, + 5 volt) presenti in un punto preciso di ciascuno delle centinaia di migliaia di dispositivi a semiconduttore che costituiscono gli attuali sistemi di calcolo. Tale variabile prende il nome di BIT , derivato dalla combinazione delle parole inglesi **B**inary d**I**gi**T** (cifra binaria) e può essere considerata l'unità minima di informazione.

Se consideriamo inizialmente il più familiare sistema di numerazione decimale, possiamo ad esempio esprimere il numero 735 attraverso la logica posizionale:

$$735 = 7 \cdot 100 + 3 \cdot l0 + 5 \cdot l$$

ovvero più formalmente

$$735 = 7 \cdot 10^2 + 3 \cdot 10^1 + 5 \cdot 10^0$$

Possiamo quindi rappresentare il numero 735 attraverso la somma pesata di tre cifre 7, 3, 5, ciascuna moltiplicata per la base del sistema (10) elevata alla potenza intera il cui grado rappresenta la posizione relativa della cifra stessa a partire da 0.

In maniera del tutto analoga possiamo esprimere una quantità numerica disponendo di due soli simboli: 0, l:

$$1101 = l \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + l \cdot 2^0$$

La sequenza dei pesi si può estendere agli esponenti negativi: in tale modo è possibile rappresentare numeri non interi:

$$1101.1011 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \quad + \quad 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} =$$

$$1 \cdot \mathbf{8} + 1 \cdot \mathbf{4} + 0 \cdot \mathbf{2} + 1 \cdot \mathbf{1} \quad + \quad 1 \cdot \mathbf{1/2} + 0 \cdot \mathbf{1/4} + 1 \cdot \mathbf{1/8} + 1 \cdot 1\mathbf{/16}$$

Nell'esempio, la quantità (in decimale) espressa dal numero 1101.1011 è data dalla somma di tutti i termini dei quali è formata e cioè:
8 + 4 + 0 + 1 = 13 per la parte intera
0.50 + 0+ 0.125 + 0.0625 = 0.68756 per la parte decimale
La quantità assoluta espressa da un qualsivoglia numero dipende dal sistema di numerazione adottato. Benché 0 e l siano simboli che possono appartenere sia al sistema di numerazione decimale che a quello binario, non c'è ambiguità nella rappresentazione, purché sia definito a priori su quale base si sta operando. Infatti il numero 1101 nel sistema decimale esprime una quantità numerica diversa rispetto al sistema binario. li numero 735 non pone questa ambiguità poiché il sistema binario non dispone dei simboli 7, 3 e 5 , ma potrebbe rappresentare un numero in base 8 (cifre da 0 a 7).

La quantità assoluta espressa da un qualsivoglia numero dipende dal sistema di numerazione adottato. Benché 0 e 1 siano simboli che possono appartenere sia al sistema di numerazione decimale che a quello binario, non c'è ambiguità nella rappresentazione, purchè sia definito a priori su quale base si sta operando. Infatti il numero 1101 nel sistema decimale esprime una quantità numerica diversa rispetto al sistema binario. Il numero 735 non pone questa ambiguità poiché il sistema binario non dispone dei simboli 7, 3, e 5, ma potrebbe rappresentare un numero in una base contenente 8 diversi simboli (da 0 a 7).
All'interno di una formalizzazione rigorosa ogni numero andrebbe quindi rappresentato esplicitando la propria base: $1101|_2$ e $735|_{10}$. Nella pratica l'ambiguità è eliminata dal contesto e solo quando si opera in contesti misti può essere utile impiegare tale formalismo.
E' prassi comune esprimere verbalmente i numeri binari come sequenze di zeri e uni e quindi il numero 1101 non si legge *millecentouno* (come normalmente avverrebbe se la base fosse decimale) bensì *uno uno zero uno.*
La cifra più a sinistra (cioè quella elevata alla potenza 3 nell' esempio) viene detta **MSB** (Most Significant Bit o cifra più significativa o bit più pesante) mentre la cifra più a destra (cioè quella elevata alla potenza O nell'esempio) viene detta **LSB** (Least Significant Bit o cifra meno significativa o bit più leggero).
La numerazione naturale si ottiene, analogamente a quanto avviene nel sistema decimale e similmente a tutte le altre basi, per mezzo di una successione ordinata di tutti i simboli disponibili (due nel sistema binario, dieci nel sistema decimale) a partire dalla posizione meno significativa. Esauriti i simboli in tale posizione si aumenta di una unità la posizione successiva e si ricomincia la successione dei simboli daccapo:

000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Questa sequenza di numeri binari, ciascuno costituito da tre cifre binarie (3 bit) consente di generare 8 diverse combinazioni per cui se includiamo lo zero, con 3 soli bit potremmo contare fino a 7 unità. Ciò conduce alla regola generale secondo la quale disponendo di un numero N di cifre binarie è possibile rappresentare 2^N combinazioni e quindi rappresentare come massima quantità positiva il numero 2^{N-1}. Ad esempio, con 16 bit si possono ottenere 2^{16} = 65536 combinazioni diverse e quindi esprimere come massima quantità positiva il numero 65535.
Attraverso questa relazione è immediata la corrispondenza tra le due basi di rappresentazione, o almeno per tutti i numeri rappresentati esattamente dalle potenze crescenti di 2 ossia 1,2,4,8, 16,32, 64, 128, 256, 512,1024 e così via. E' però più interessante vedere come sia possibile passare da una base all'altra attraverso un procedimento generalizzato.
Se riprendiamo il solito numero binario 1101, attraverso la sua rappresentazione esplicita, siamo in grado di esprimere immediatamente il valore corrispondente nel sistema decimale:

$$1101 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

Sarà infatti sufficiente eseguire la somma di prodotti e il risultato sarà il valore cercato:

$$110|_2 = 8 + 4 + 0 + 1 = 13|_{10}$$

In sostanza vanno considerati solo i contributi delle cifre diverse da zero e si sommano i relativi pesi. Nella pratica, specie se i numeri sono piuttosto grandi si può ricorrere alle tavole di conversione o ancora più agevolmente impiegare una calcolatrice tascabile di tipo scientifico dotata di rappresentazione binaria e relativa funzioni di conversione.
Vediamo ora come sia possibile convertire un numero decimale nel corrispondente binario prendendo come numero di partenza proprio il numero 13. Il procedimento consiste nel dividere successivamente per 2 il numero di partenza fino a che il quoziente intero risulti diverso da zero e nel formare il numero binario coi resti delle divisioni successive *presi in ordine inverso:*

13 : 2 = 6 R = 1
6 : 2 = 3 R = 0
3 : 2 = 1 R = 1
1 : 2 = 0 R = 1

da cui risulta appunto che $13|_{10} = 1101|_2$

Un sistema alternativo ma assai efficace consiste nel valutare a mente le somme parziali successive attraverso un procedimento iterativo che accumula i pesi decrescenti :

Test: 13 < di 16 ?
Sì. Allora si pone 1 nella posizione immediatamente inferiore (posizione $3 = 2^3 = 8$)

8	4	2	1
1	-	-	-

Test: 8< 13 ?
Sì. Allora si pone 1 nella posizione immediatamente inferiore (posizione $2 = 2^2 = 4$) e si fa la somma parziale : 8 + 4 = 12

8	4	2	1
1	1	-	-

Test: 12< 13 ?
Sì. Allora si pone 1 nella posizione immediatamente inferiore (posizione $2 = 2^1 = 4$) e si fa la somma parziale : 12 + 2 = 14. Il risultato è maggiore di 13 quindi si deve porre a zero questo bit. La somma parziale ritorna quelle precedente cioè 12

8	4	2	1
1	1	0	-

Test: 12<13 ?
Sì. Allora si pone 1 nella posizione immediatamente inferiore (posizione $2 = 2^0 = 1$) e si fa la somma parziale: 12 + 1 = 13.
Quando la somma parziale uguaglia il numero di partenza la conversione è completa
.

8	4	2	1
1	1	0	1

Come spesso accade in questi casi è meno intuitiva la spiegazione del procedimento rispetto alla sua effettiva semplicità applicativa.
Prima di estendere la rappresentazione binaria ai numeri relativi, occorre dire come si esegue l' operazione di addizione tra numeri binari. Poiché in un calcolatore si ha a disposizione un numero limitato di bit, occorre che il risultato della somma sia contenuto all'interno del massimo numero esprimibile con quella specifica quantità di bit.
In ogni caso l'addizione si esegue come si è imparato a farla elementarmente, tenendo conto che i riporti si hanno quando la somma supera 1. Ad esempio si ha:

11(riporti)			
110110	+	32+16+0+4+2+0	= 54 +
000110	=	0+0 +0+4+2+0	= 6 =
111100			60

Quando la somma supera la capacità massima disponibile (in questo esempio 6 bit, 2^6-1= 63) si verifica il fenomeno dell'*overfiow.* In questi casi, se ciò non è opportunamente segnalato, il risultato dell'addizione perde di significato. Ad esempio se avessimo sommato al numero 54 il numero l0 (ovvero $1010|_2$) si sarebbe avuto:

110110	+	(32+16+0+4+2+0	= 54 +)
001010	=	0+0 +8+0+2+0	= 10=)
1000000			0

In questo caso, per esprimere il risultato corretto (che è 64) occorre una cifra addizionale. Nelle applicazioni reali, è sempre possibile prevedere l'ordine di grandezza del risultato in modo tale che possa essere espresso in modo corretto.

Rappresentazione binaria dei numeri relativi

La rappresentazione di un numero relativo in generale necessita di un'informazione sul segno e poiché tale segno può essere o positivo o negativo, tale informazione, nella rappresentazione binaria, può essere associata ad uno specifico bit, assumendo che il valore 0 rappresenti il + (numero positivo) mentre 1 rappresenti il - (numero negativo). In pratica disponendo di N bit si utilizzeranno N-l bit per rappresentare il valore del modulo e 1 bit per rappresentare il segno. Tale rappresentazione prende appunto il nome di rappresentazione in *modulo e segno.*
Supponendo ad esempio di disporre di soli 3 bit, il primo bit a sinistra sarà impiegato per esprimere il segno ed i restanti 2 bit per il modulo. Così facendo, delle 2^3 = 8 combinazioni, 4 avranno uno 0 a sinistra e 4 un l a sinistra:

positivi		negativi	
+ 0	000	100	- 0
+ 1	001	101	- 1
+ 2	010	110	- 2
+ 3	011	111	- 3

In questo modo si avrebbe, per i numeri positivi la successione 0, 1, 2, 3 e analogamente per i numeri negativi 0, -1, -2, -3. Risulta allora evidente l'ambiguità della rappresentazione dello zero poiché due sono possibili due diverse rappresentazioni dello stesso numero (+0,-0). E' come se esistesse uno zero positivo e uno zero negativo.
Per rimuovere tale ambiguità si ricorre ad un tipo di rappresentazione detta *in complemento a due* che consiste nel rappresentare i numeri negativi attraverso un'operazione di complementazione del corrispondente positivo. Sono possibili due forme di complementazione: complementazione a uno e complementazione a due, ma per motivi di ordine pratico relative ad operazioni di calcolo si preferisce quest' ultima. Per ottenere il complemento a due di un numero si devono compiere due passi:
a) complementazione[1] a 1 del numero stesso che consiste nel cambiare gli zeri in uni e viceversa
b) sommare 1 al complemento a uno

Ad esempio, se consideriamo ancora 3 bit, il numero 011 rappresenta +3. Allora per ottenere il valore corrispondente negativo si avrà:

[1] Per *complementazione* si deve intendere cambiare gli uni con gli zeri.

011	numero positivo (+3)
100	complemento a 1 (gli zeri vengono cambiati in uni e viceversa)
1	somma 1
____	
101	numero negativo in complemento a 2 (-3)

Si noti che se c'è un riporto nel bit di segno, questo non viene considerato.
Attraverso la complementazione a 2, nel caso di 3 bit possiamo rappresentare la seguente sequenza:

+ 3	011
+ 2	010
+ 1	001
0	000
- 1	111
- 2	110
- 3	101
- 4	100

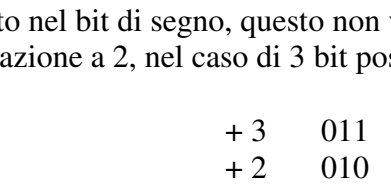
Si possono fare due osservazioni: la prima riguarda il tipo di disposizione circolare dei numeri binari. A partire dallo 0 verso l'alto la numerazione si raccorda tra il massimo positivo (+3) e il massimo negativo (-4).
La seconda, ben più rilevante della prima, è che attraverso la rappresentazione in complemento a 2 non c'è simmetria dal momento che il massimo numero negativo è di una unità superiore rispetto al corrispondente numero positivo. Questo è il prezzo che si deve pagare per non avere nessuna ambiguità nella rappresentazione dei numeri relativi. Allora dati N bit, la rappresentazione in complemento a 2 consentirà una numerazione bipolare i cui valori estremi (positivi e negativi) sono rispettivamente $+2^{N-1} - 1$ e -2^{N-1}. Ad esempio disponendo di 16 bit complessivi e volendo rappresentare numeri relativi si avranno come limiti estremi +32767, -32768. Questa apparente limitazione dovuta alla asimmetria, non porta nella pratica a nessun inconveniente. Naturalmente il procedimento di complementazione è ancora applicabile per ottenere il corrispondente positivo di un numero negativo dato.
La rappresentazione in complemento a due è molto diffusa e una certa classe di dati numerici viene rappresentata secondo tale formato.

Il problema della conversione

Abbiamo già visto brevemente come si può passare dal sistema decimale a quello binario e viceversa. Per i numeri relativi la conversione può essere fatta utilizzando procedure (algoritmi) che eseguono il calcolo separatamente per la parte intera e per quella frazionaria. Per la sola parte frazionaria si può procedere nel modo che segue: si moltiplica per 2 il numero iniziale e si pone uno 0 se la cifra intera è pari oppure 1 se la prima cifra è dispari. Si procede ad una nuova moltiplicazione fino al raggiungimento dell'annullamento della parte frazionaria dopodiché, si prenderà la sequenza ottenuta nell'ordine di successione e si arriverà al risultato finale.

Supponiamo che il numero decimale da convertire sia 0.3125 . Si avrà allora:

$0.3125 \cdot 2 = 0.625 \quad \Rightarrow$
$0.6250 \cdot 2 = 1.250 \quad \Rightarrow$
$0.2500 \cdot 2 = 0.500 \quad \Rightarrow$
$0.5000 \cdot 2 = 1.000 \quad \Rightarrow$
$0.0000 \cdot 2 = 0.000 \quad \Rightarrow 0$



0 1 0 1

Quindi $0.3125|_{10} \Rightarrow 0.0101|_2$

Come controprova, possiamo sommare i pesi corrispondenti. Si otterrà allora:

$0 \cdot \mathbf{1/2} + 1 \cdot \mathbf{1/4} + 0 \cdot \mathbf{1/8} + 1 \cdot 1\mathbf{/16} =$
$0 + 0.25 + 0 + 0.0625 = 0.3125$ (come si voleva dimostrare)

Non sempre però è possibile rappresentare correttamente la parte frazionara di un numero con un numero finito di cifre. In questo caso si deve arrestare la conversione giungendo così ad un risultato che approssima il numero desiderato. La rappresentazione sarà precisa ad un certo numero di cifre significative.