DENTRO LA SCATOLA

Rubrica a cura di

Fabio A. Schreiber

Il Consiglio Scientifico della rivista ha pensato di attuare un'iniziativa culturalmente utile presentando in ogni numero di Mondo Digitale un argomento fondante per l'Informatica e le sue applicazioni; in tal modo, anche il lettore curioso, ma frettoloso, potrà rendersi conto di che cosa sta "dentro la scatola". È infatti diffusa la sensazione che lo sviluppo formidabile assunto dal settore e di conseguenza il grande numero di persone di diverse estrazioni culturali che - a vario titolo - si occupano dei calcolatori elettronici e del loro mondo, abbiano nascosto dietro una cortina di nebbia i concetti basilari che lo hanno reso possibile. La realizzazione degli articoli è affidata ad autori che uniscono una grande autorevolezza scientifica e professionale a una notevole capacità divulgativa. Il primo di essi, pubblicato in questo numero, esce a firma del Prof. Luigi Dadda, uno dei Padri Fondatori dell'Informatica italiana e tutt'ora attivo ricercatore presso il Politecnico di Milano. Il Prof. Dadda è stato anche uno dei fondatori dell'AICA e per lunghi anni direttore responsabile di "Rivista di Informatica", il suo organo ufficiale.

Fondamenti dell'aritmetica digitale: i codici numerici

Luigi Dadda

1. Introduzione

Questo primo articolo sui fondamenti dell'aritmetica digitale tratterà dei modi di rappresentazione dei numeri nella forma adatta ai calcolatori digitali. È facile che sorga nel lettore la curiosità sulla storia dei numeri stessi. Per soddisfare, almeno in parte, tale possibile desiderio, sono riportati nella bibliografia gli indirizzi di una seria scelta di siti Web.
Se si fa riferimento specificamente alla rappresentazione dei numeri all'interno dei calcolatori un punto fondamentale di tale storia é segnato dalla adozione generalizzata del sistema binario, per la più facile realizzabilità dei circuiti logici e di memoria [10].

2. Sistemi numerici posizionali

2.1. Numeri interi

Un numero intero X può essere scritto come una stringa di simboli (*cifre* o *digits*):

$$X = x_{n-1}\, x_{n-2} \ldots\ldots x_1\, x_0$$

scelti da un insieme $(0, 1, \ldots, b^{-1})$; b è la base o radice. Ciascuna cifra, x_i, ha un "peso" b^i, cosicchè:

$$X = x_{n-1} \cdot b^{n-1} + x_{n-2} \cdot b^{n-2} + \ldots\ldots\ldots + x_1 \cdot b^1 + x_0 \cdot b_0 \qquad (1)$$

Poiché il valore o peso di una cifra x_i dipende dalla sua posizione nel numero, i sistemi numerici con base sono anche detti "posizionali". I valori della base b più comunemente utilizzati nei calcolatori elettronici sono: 10 (base decimale), 2 (base binaria), 8 (base ottale), 16 (base esadecinale). Per $b = 2$ per e $b = 8$ le cifre sono normalmente rappresentate con gli stessi simboli usati nel sistema decimale (rispettivamente 0, 1 e 0, 1, 2, 3, 4, 5, 6, 7); per $b = 16$ si adottano i simboli 0,....,9 con l'aggiunta di altri sei: A, B, C, D, E, F.
Si noti che, poiché uno stesso simbolo può essere usato per rappresentare numeri in basi diverse, se si usano basi diverse è necessario indicare la base di ogni numero. Per esempio: 145_{10}, 101_2, 176_8, $1D6_{16}$.
È, infatti, importante notare che il numero 101 potrebbe non essere in base 2 ma in una qualsiasi delle altre basi sopra citate (dieci, otto, sedici) perché i simboli 0 e 1 appartengono a tutte e quattro le basi. La combinazione 101 potrebbe, quindi, rappresentare valori del tutto diversi:

$101_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 4 + 0 + 1 = 5_{10}$
$101_{10} = 1 \cdot 10^2 + 0 \cdot 10^1 + 1 \cdot 10^0 = 100 + 0 + 1 = 101_{10}$
$101_8 = 1 \cdot 8^2 + 0 \cdot 8^1 + 1 \cdot 8^0 = 64 + 0 + 1 = 65_{10}$
$101_{16} = 1 \cdot 16^2 + 0 \cdot 16^1 + 1 \cdot 16^0 = 256 + 1 = 257_{10}$

Tutti i calcoli sopra riportati sono stati eseguiti con numeri nella base dieci. Essi sono anche esempi di conversione di numeri interi nelle basi 2, 8, 16 nella rappresentazione equivalente in base dieci.

2.2. Numeri frazionari

Per rappresentare numeri frazionari si usa la notazione posizionale con pesi costituiti da potenze negative intere della base:

$$X = 0 \cdot x_{-1} x_{-2} \ldots x_{-n} = x_{-1} \cdot b^{-1} + x_{-2} \cdot b^{-2} + \ldots + x_{-n} \cdot b^{-n}$$

Un numero binario frazionario sarà facilmente convertibile nell'equivalente decimale disponendo dei valori:

$b^{-1} = 0.5_{10}$, $b^{-2} = 0.25_{10}$, $b^{-3} = 0.125_{10}$, $b^{-4} = 0.0625_{10}$, $b^{-5} = 0.03125_{10}$ ecc.

Per esempio:

$0.11001_2 = 0.5 + 0.25 + 0 + 0 + 0.03125 = 0.78125_{10}$

2.3. Numeri misti

Sono costituiti da parte intera e parte frazionaria. La loro conversione si ottiene convertendo separatamente le rispettive parti intere e quelle frazionarie.
La conversione in base 2 dei numeri in base 8 e in base 16 si può molto facilmente eseguire esprimendo ciascuna cifra ottale (esadecimale) con il gruppo di 3 (4) bit, che rappresentano la cifra nel sistema binario per esempio:

$325.47_8 = 011\ 010\ 101.100\ 111_2$
$325.47_{16} = 0011\ 0010\ 0101.0100\ 0111_2$
$3B8.45D_{16} = 0011\ 1011\ 0100.0100\ 0101\ 1101_2$

3. Conversione di numeri decimali negli equivalenti binari

3.1. Decimali interi

Un qualsiasi numero binario intero può esprimersi nella seguente forma:

$$X = \{[(x_{n-1} \cdot 2 + x_{n-2})\ 2 + x_{n-3}]\ 2 + \ldots\ldots\ldots + x_1\}\ 2 + x_0$$

Per esempio:

$$X = 110101_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = \{[((1 \cdot 2 + 1) \cdot 2 + 0) \cdot 2 + 1] \cdot 2 + 0\} \cdot 2 + 1$$

Da ciò può derivarsi la seguente *regola per la conversione* di un intero decimale nella forma binaria:
1. Si divide per due l'intero: il resto vale x_0, la *prima* cifra binaria (di peso 1).
2. Si divide il quoziente per due: il resto vale x_1, la *seconda* cifra binaria.
3. Si ripete il passo precedente, ottenendo via via le altre cifre della rappresentazione, fino ad ottenere un quoziente nullo.

Esempio:

$X = 54_{10}$
54:2 = 27 con resto 0 = x_0
27:2 = 13 con resto 1 = x_1
13:2 = 6 con resto1 = x_2
6:2 = 3 con resto 0 = x_3
3:2 = 1 con resto 1 = x_4
1:2 = 0 con resto 1 = x_5
e quindi: $X = 110110_2$

3.2. Decimali frazionari

Con procedimento analogo al precedente, si ottiene la seguente regola di conversione:
1. Si moltiplica per due il frazionario dato: la parte intera del prodotto, che può essere solo *0* oppure *1*, vale x_{-1}, la cifra binaria più significativa (di peso 2^{-1}).
2. Si applica alla parte frazionaria del precedente raddoppio lo stesso precedente procedimento, fino ad ottenere una parte frazionaria nulla o ad individuare un numero periodico (si osservi che, in questo caso, un numero decimale frazionario con un numero finito di cifre significative può generare un numero binario frazionario con un numero di cifre infinito).

Esempi:

$X = 0.75_{10}$
$0.75 \cdot 2 = 1.50$; parte intera 1 = x_{-1}
$0.50 \cdot 2 = 1.00$; parte intera 1 = x_{-2}
0.00 e quindi: $X = 0.11_2$

$X = 0.1_{10}$

Si ottiene:

$X = 0.0001100110011001100110011...._2 = 0.\underline{0011}_2$ cioè un numero periodico.

4. Numeri binari negativi

Il modo più facile per distinguere i numeri positivi da quelli negativi consiste nell'aggiungere a essi un apposito simbolo, il segno. Per esso basta un bit e una convenzione, per esempio 0 per +, 1 per –: è il metodo di rappresentazione detto del *segno e grandezza* (*sign-and-magnitude*) e in esso occorre disporre di due algoritmi distinti per la somma e per la sottrazione.
Un altro metodo di rappresentazione dei numeri negativi è, invece, basato sui *complementi*. Nel seguito, essi vengono esemplificati facendo riferimento a numeri binari di 4 bit (Tabella 1).
La tabella 1 contiene nella prima colonna le 16 configurazioni di 4 bit. La seconda colonna contiene i valori equivalenti, in base 10, delle predette configurazioni secondo la rappresentazione detta dei *complementi* a 2, *o*, *più in generale, complementi alla base*: il bit all'estrema sinistra ha il ruolo del segno. La terza colonna della tabella 1 mostra i valori delle configurazioni con primo bit di valore 1 nel caso di rappresentazione con i *complementi ad* 1, *o*, *più in generale, complementi alla* (*base* – 1). Il bit più a sinistra rappresenta ancora il segno e vi è anche uno "0" negativo.
L'utilità di ricorrere ai complementi deriva da due tipi di considerazioni: il primo è la facilità di ottenere il complemento a 1 o a 2 di un numero dato mediante dispositivi elettronici; il secondo sta nel fatto che, con l'utilizzo della rappresentazione in complemento dei numeri negativi, le operazioni di somma e sottrazione possono essere eseguite con lo stesso circuito. Tali argomenti verranno trattati in successivi articoli, dedicati alle operazioni aritmetiche e alle macchine per realizzarle.
Qui di seguito si vuol dare solamente un breve esempio di operazioni di somma/sottrazione che utilizzano i concetti esposti; si tenga presente che l'algoritmo di somma è analogo a quello usato nel caso decimale e che, in binario, valgono le seguenti regole: 0 + 0 = 0, 0 + 1 = 1, 1 + 1 = 0 con un riporto di 1. Nella rappresentazione in complemento a 2 il riporto viene trascurato:

	Complementi a 2 o complementi alla base	Complementi ad 1 o complementi alla (base –1)
0111	+7	
0110	+6	
0101	+5	
0100	+4	
0011	+3	
0010	+2	
0001	+1	
0000	0	
1111	–1	–0
1110	–2	–1
1101	–3	-2
1100	–4	–3
1011	–5	–4
1010	–6	–5
1001	–7	–6
1000	–8	–7

TABELLA 1
Esemplificazione di numeri negativi

0111 (7)	1101 (13)	0101 (+5)	1010 (–6)
0010 (2)	0011 (3)	1110 (–2)	0010 (+2)
1001 (9)	10000 (16)	(1)0011 (+3)	1100 (–4)
somme di numeri positivi		somme di numeri in complemento a 2	

5. Numeri decimali con cifre codificate in binario

I numeri binari sono certamente i più utilizzati nei calcoli scientifico-tecnici, in quanto le operazioni aritmetiche sono eseguibili con la massima velocità quando gli operandi sono rappresentati in tale forma . I dati di ingresso e i risultati sono, tuttavia, quasi sempre forniti o voluti in forma decimale, peraltro facilmente ottenibile con i metodi prima descritti.
Molti calcolatori, tuttavia, sono dedicati a calcoli di tipo amministrativo. Si possono sempre utilizzare allo scopo numeri binari, ma vi sono esigenze che richiedono di eseguire i calcoli direttamente nella forma decimale. Basti pensare al problema degli arrotondamenti, che devono soddisfare precisi requisiti anche legali, difficili da realizzare su numeri convertiti in binario. Si può ricordare l'esempio mostrato in preceden-

za sulla conversione in binario di 0.1_{10}, che genera un numero binario di lunghezza infinita e che perciò richiede un inevitabile arrotondamento dato che tutti i calcolatori rappresentano ovviamente i numeri utilizzando un numero finito di bit.
Una rappresentazione usata a questo scopo è quella costituita dal cosiddetto codice BCD/8421, rappresentato nella prima colonna della tabella 2. BCD sta per *binary coded decimals*, ossia cifre decimali codificate in binario, e 8, 4, 2, 1 sono i pesi associati nell'ordine ai 4 bit. La somma dei pesi delle cifre binarie pari a 1 individua la cifra rappresentata. Per esempio, 0110 individua la cifra 6 in quanto la sommatoria dei pesi fornisce per l'appunto $0 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 = 6$.
Nella tabella 2 sono riportate altre tre rappresentazioni binarie delle dieci cifre decimali, che godono di utili proprietà sulle quali si tornerà in seguito.

6. Numeri rappresentati in virgola mobile

Il modo più semplice per rappresentare i numeri reali consiste nel rappresentarne parte intera e parte frazionaria separate dal punto (l'equivalente anglosassone della virgola, di fatto uno standard nel mondo dei calcolatori). Tale rappresentazione non è però adatta nei calcoli tecnico-scientifici, che possono mettere in gioco contemporaneamente numeri piccolissimi e numeri grandissimi. Infatti, a causa del numero finito di bit utilizzati per rappresentare i numeri all'interno dei calcolatori, si potrebbero perdere molte o tutte le cifre significative per i numeri molto piccoli (oppure, per i numeri molto grandi, si limiterebbe il valore massimo rappresentabile).
La soluzione di tale problema è offerta dalla rappresentazione in virgola mobile (*floating point, FP*).Con essa, il numero decimale 765.432 può essere rappresentato con 7.65432×10^2; l'esadecimale –001D4B4E con $-1.D4B4E \times 16^{-2}$, il binario 101.111 con 1.01111×2^2.
Un numero in virgola mobile è, quindi, individuato da: segno, base (spesso implicita), esponente (associato alla base), frazione o mantissa (l'insieme delle cifre significative).
Per i numeri FP. binari lo IEEE (*Institute of Electrical and Electronics Engineers*) ha definito, con lo Standard 754, una rappresentazione a 32 bit così costituita:

s eeeeeeee fffffffffffffffffffffff

cioè con 1 bit per il segno (s), 8 bit per l'esponente in base = 2 (le *e*), 23 bit per la frazione (le *f*) che si sottintende essere del valore 1.*fffffffffffffffffffffff* (il bit di valore 1 nella parte intera non viene fisicamente memorizzato). In realtà, l'esponente è rappresentato sommando l'esponente vero (che può essere positivo o negativo) alla costante (*bias*) : $0111\ 1111_2 = 127_{10}$, allo scopo di semplificare il confronto tra numeri.
Con la suddetta notazione possono essere rappresentati numeri nel campo $10^{-44.85} - 10^{38.53}$.
È stato definito anche lo standard per numeri FP binari con 64 bit (per maggiori ragguagli vedasi [4, 8]).

7. Altri modi e casi di codificazione numerica

Sono in uso (per esempio per rappresentare date e tempi) numeri posizionali con pesi che non sono potenze intere di una base.
Un sistema concettualmente importante di rappresentazioni dei numeri binari si basa sull'uso dei "residui", cioè dei resti della divisione del numero dato per un, opportuna serie di divisori prefissati. Anche questo argomento verrà trattato in un successivo articolo.
Quanto finora detto riguarda comunque i numeri in senso stretto, ma la codificazione binaria è richiesta anche per le informazioni che si

	BCD 8421	Biquinario	2 di 5	Gray
0	0000	01 00001	00011	0010
1	0001	01 00010	00101	0110
2	0010	01 00100	00110	0111
3	0011	01 01000	01001	0101
4	0100	01 10000	01010	0100
5	0101	10 00001	01100	1100
6	0110	10 00010	10001	1101
7	0111	10 00100	10010	1111
8	1000	10 01000	10100	1110
9	1001	10 10000	11000	1010

TABELLA 2
Due rappresentazioni binarie delle cifre decimali

esprimono tramite gli alfabeti delle varie lingue (codificazioni "alfanumeriche").
Sono, inoltre, necessari codici binari speciali per rappresentare altre forme di espressione, come il suono e le figure.
Un aspetto generale della informazione codificata è quello della sua integrità, non solo durante la sua elaborazione ma anche per quanto riguarda la trasmissione a distanza e la memorizzazione. La corruzione dell'informazione può manifestarsi con la perdita di uno o più bit o con la loro alterazione. La teoria dell'informazione ha sviluppato metodi che permettono la rivelazione di errori e anche la loro correzione (entro certi limiti). Furono proposti e usati alcuni codici destinati alle informazioni puramente numeriche, ma presto il problema venne affrontato per l'informazione in generale, anche nella forma alfanumerica. I codici "biquinario" e "due di cinque" della tabella 2 offrono, proprio per la ridondanza che li caratterizza, una sia pur modesta capacità di rivelazione di errori: il primo perché tutte le configurazioni sono costituite da due gruppi di bit e in ciascun gruppo un solo bit ha il valore 1; il secondo perché tutte le configurazioni hanno 2 e 2 soli bit con valore 1.

Bibliografia

[1] Ancient Egiptian Mathematics
http://home.clara.net/beaumont/egypt/maths/

[2] Chinese Numbers
http://www.mandarintools.com/numbers.html
(con programma in linea per la conversione automatica).

[3] Evolution of Arabic (Roman) Numerals from India
http://www.gosai.com/chaitanya/saranagati/html/vishnu_mjs/math/math_4.html

[4] Hollasch, S.: IEEE Standard, 754-Floating Point Numbers
http://research.microsoft.com/~hollasch/cgindex/coding/ieeefloat.html

[5] Kailash Srivastava, Numbersystem, some clarification
http://manaskriti.com/InteractInn/10119801.html

[6] IEEE-754 Floating-Point Conversion
http://babbage.cs.qc.edu/courses/cs341/IEEE-754.html

[7] Indian numerals
http://www-gap.dcs.st-and.ac.uk/~history/HistTopics/Indian_numerals.html

[8] Maya Mathematics
http://www.michielb.nl/maya/math.html
(con programma in linea per la conversione automatica).

[9] Melville, Ancient and Classical Mathematics
http://it.stlawu.edu/%7Edmelvill/323/index.html

[10] von Neumann, John: The Principles of Large-Scale Computing Machines. Reprinted in: *Ann. Hist. Comp.*, Vol. 3, n. 3, 1946, p. 263-273.

[11] The Arabic numeral system
http://www-gap.dcs.st-and.ac.uk/~history/HistTopics/Arabic_numerals.html

Luigi Dadda laureato in Ingegneria Elettrotecnica. Titolare della cattedra di Elettrotecnica dal 1962. È stato Rettore del Politecnico di Milano dal 1 Novembre 1972 al 31 Ottobre 1984. Nel 1954 ha vinto una borsa di Studio della National Sciences Foundation per svolgere ricerche presso il California Institute of Technology di Los Angeles, dove ha partecipato alla realizzazione di un calcolatore elettronico che nel 1954 è stato installato al Politecnico di Milano. La sua attività di ricerca si é indirizzata inizialmente nel campo dei modelli e dei calcolatori analogici, poi in quello dei calcolatori elettronici dove, in particolare, si é occupato delle unità aritmetiche proponendo soluzioni originali per i moltiplicatori paralleli. Si é in seguito dedicato allo studio delle reti di Petri, proponendone l'impiego per il progetto di sistemi di controllo di grande complessità. Più recentemente ha esteso le ricerche sull'area della elaborazione di segnale con lo studio di nuovi sistemi di convolutori. Oltre all'attività scientifica e didattica, si é anche dedicato alla guida e al coordinamento della ricerca sia in sede nazionale che internazionale. Il Centro di Calcolo e il Laboratorio di Calcolatori Elettronici del Dipartimento di Elettronica del Politecnico di Milano da lui diretto, ha svolto ricerche avanzate sui sistemi di calcolo, sull'architettura di microcalcolatori, sui linguaggi di programmazione, sulle banche di dati e sulle reti di calcolatori. Nel 1980/82 ha presieduto la Commissione per la Scienza e la Tecnologia presso la Presidenza del Consiglio dei Ministri. È fondatore e direttore della "Rivista di Informatica".
luigi.dadda@polimi.it