Codifiche numeriche

Stefano Lodi
Insegnamento di Informatica
Alma Mater Studiorum Università di Bologna
stefano.lodi@unibo.it

Sistemi di numerazione

- Il concetto di numero è verosimilmente antecedente all'utilizzo di qualunque particolare notazione ed è indipendente dalla particolare notazione usata
  - "MCMXCV" e "1995" rappresentano lo stesso numero
- Si dividono in posizionali e non posizionali

non posizionali Il valore di un simbolo non dipende dalla posizione

- Esempio: in "MCMXCV", 'M' vale mille indipendentemente dalla posizione

posizionali Il valore di un simbolo dipende dalla posizione

- Esempio: in "1995", '9' vale novecento o novanta, e ciò è determinato solo dalla sua posizione

Sistemi di numerazione posizionali

- Un insieme finito di simboli $\{d_1, d_2, \ldots, d_B\}$, la cui cardinalità B è la *base* del sistema di numerazione
  - Esempio: $\{a, b, c, d, e, f, g, h\}$, $B = 8$
- Una corrispondenza biunivoca tra l'insieme di simboli e i valori numerici $\{0, 1, \ldots, B-1\}$ che associa a ciascun simbolo d_i il suo valore numerico $\hat{d}_i$
  - Esempio:

$\hat{a} =$ due	$\hat{b} =$ tre	$\hat{c} =$ sette	$\hat{d} =$ otto
$\hat{e} =$ cinque	$\hat{f} =$ uno	$\hat{g} =$ sei	$\hat{h} =$ quattro

- Il valore $\hat{\delta}$ di una stringa qualunque di simboli $\delta = d_1 d_2 \ldots d_n$ si determina con la seguente formula

$$\hat{\delta} = \sum_{i=1}^{n} \hat{d}_i B^{n-i} = \hat{d}_1 B^{n-1} + \hat{d}_2 B^{n-2} + \cdots + \hat{d}_{n-1} B^1 + \hat{d}_n B^0$$

Sistemi posizionali notevoli

- In informatica, sono prevalentemente impiegati i seguenti sistemi

sistema	B	simboli
binario	2	$\{0, 1\}$
ottale	8	$\{0, 1, 2, 3, 4, 5, 6, 7\}$
decimale	10	$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
esadecimale	16	$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$

- Il sistema di numerazione impiegato si indicherà come indice (scritto nel sistema in base 10) della sequenza di cifre
  $43798_{10} = AB16_{16} = 125426_8 = 1010101100010110_2$

Conversione tra basi

- Problema: convertire una stringa che rappresenta un numero x espresso in un base (di partenza) nella stringa che rappresenta x in un'altra base (di arrivo)
- È possibile sia operando l'aritmetica della base di partenza che quella della base di arrivo

Uso della base di arrivo

- Consideriamo un numero in base esadecimale e convertiamolo in base decimale

1. Riscriviamo il numero per esteso utilizzando la formula appena vista

$$7D1_{16} = 7_{16} \cdot 10_{16}^{2} + D_{16} \cdot 10_{16}^{1} + 1_{16} \cdot 10_{16}^{0}$$

2. Trasformiamo le singole cifre

$$7_{16} = 7_{10} \; D_{16} = 13_{10} \; 1_{16} = 1_{10}$$

3. Sostituiamo nella formula iniziale, sostituendo anche le cifre che rappresentano la base ed eseguiamo i calcoli

$$7D1_{16} = 7_{10} \cdot 16_{10}^{2} + 13_{10} \cdot 16_{10}^{1} + 1_{10} \cdot 16_{10}^{0} = 2001_{10}$$

Formula di Horner

- La trasformazione utilizzando la base di arrivo può essere effettuata con somme e moltiplicazioni mediante la *formula di Horner*; è una formula generale per il calcolo del valore di un polinomio di grado n

$$\begin{aligned} p(x) &= a_n x^n + a_{n-1} x^{n-1} + \cdots a_1 x^1 + a_0 x^0 \\ &= (\ldots((a_n x + a_{n-1})\, x + a_{n-2})\, x + \cdots + a_1)\, x + a_0 \end{aligned}$$

- La formula di Horner si può calcolare con n moltiplicazioni e n somme
- Applicando la formula di Horner all'espressione di $\hat{\delta}$ si ottiene una formula di conversione con l'utilizzo della base di arrivo

$$\hat{\delta} = \sum_{i=1}^{n} \hat{d}_i B^{n-i} = (\ldots((\hat{d}_1 B + \hat{d}_2)\, B + \hat{d}_3)\, B + \cdots + \hat{d}_{n-1})\, B + \hat{d}_n$$

Uso della base di partenza

- Si utilizzano le operazioni dell'aritmetica intera div e mod
- Consideriamo un numero rappresentato in base decimale e convertiamolo in base binaria
- Vale il seguente sistema, per interi positivi

$$\begin{cases} \hat{\delta}_n = \hat{\delta} \\ \hat{\delta}_{i-1} = \hat{\delta}_i \text{ div } B \\ \hat{d}_i = \hat{\delta}_i \text{ mod } B \end{cases}$$

1972 / 2	986	0
986 / 2	493	0
493 / 2	246	1
246 / 2	123	0
123 / 2	61	1
61 / 2	30	1
30 / 2	15	0
15 / 2	7	1
7 / 2	3	1
3 / 2	1	1
1 / 2	0	1

$$1972_{10} = 11110110100_2$$

Rappresentazione degli interi

- Il componente funzionale elementare di un moderno elaboratore elettronico è un particolare circuito, il *flip-flop*, che può trovarsi in uno di due distinti stati, e lo conserva fino a quando impulsi forniti ai suoi ingressi non lo modificano
- Se si interpretano convenzionalmente gli stati come le cifre 0 e 1 del sistema di numerazione binaria, una successione di flip-flop realizza un semplice dispositivo per la memorizzazione di numeri in notazione binaria
- Nomenclatura
  - Una singola cifra binaria è denominata *bit* (*binary digit*)
  - 8 bit formano un *byte*
- Problema: rappresentare un intero con segno, con lo scopo di memorizzarlo utilizzando successioni di flip-flop
- Esistono due possibilità

  Modulo e segno
  Complemento a due

Rappresentazione in modulo e segno

- Supponiamo di disporre di 4 byte = 32 bit
- I bit sono convenzionalmente numerati da 0 a 31 da destra verso sinistra, in tal modo maggiore e l'indice, maggiore è la significatività della cifra corrispondente

1	0	1	...	0	1	1	0
31	30	29	...	3	2	1	0

- Si utilizza il primo bit per rappresentare il segno
- I bit 0–30 rappresentano il valore assoluto del numero
- Si possono rappresentare i seguenti intervalli di valori

positivi 2^{31} valori positivi distinti, da $+0$ a $2^{31}-1$
negativi 2^{31} valori negativi distinti,da -0 a $-(2^{31}-1)$

Rappresentazione in complemento a due

- Il *complemento a due* di un numero binario si ottiene
  1. Rappresentando il numero in forma binaria
  2. invertendo tutti i bit, cioè sostituendo simultaneamente 0 con 1 e viceversa; tale operazione è chiamata *complemento a uno*
  3. Aggiungendo il valore 1
- Nella *codifica in complemento a due* i numeri positivi sono rappresentati in modulo e segno e i negativi in complemento a due
- L'intervallo di rappresentazione dei positivo resta invariato: da 0 a $2^{31} - 1$

0	0	0	...	0	0	0	0	÷	0	1	1	...	1	1	1	1

- Per i negativi, l'intervallo è da -2^{31} a -1 (-0 non è rappresentabile; la sequenza 100...000 che lo rappresentava in modulo e segno ora rappresenta -2^{31}

1	0	0	...	0	0	0	0	÷	1	1	1	...	1	1	1	1