

Sistemi di numerazione

Andrea Passerini
passerini@dsi.unifi.it

Conoscenze informatiche e relazionali
Corso di laurea in Scienze dell'Ingegneria Edile

Premessa

- Cosa intendiamo quando scriviamo (ad es.) ?

1945732

unmilionenovecentoquarantacinquemilasettecentotrentadue

- Questo si ottiene come:
due + trenta + settecento + cinquemila + \dots + unmilione
- Ogni cifra viene moltiplicata per un peso. Da destra a sinistra i pesi sono 1, 10, 100, 1000, \dots , 1000000, ossia $10^0, 10^1, 10^2, 10^3, \dots, 10^6$
- Il totale si ottiene sommando il valore di ciascuna cifra moltiplicato per il peso corrispondente alla sua posizione.

Sistemi di numerazione posizionali

Ingredienti

- Un numero naturale b detto *base*. (e.g. 10)
- Un insieme ordinato di $b - 1$ simboli distinti detti *cifre* (e.g. 0,1,2,3,4,5,6,7,8,9)
- Un *codice di interpretazione* per determinare il numero rappresentato da una stringa di cifre.
- Un insieme di procedure (algoritmi) per le quattro operazioni aritmetiche $+, -, \times, /$

Codice di interpretazione

- Un numero intero rappresentato in base b con n cifre è una stringa di cifre:

$$(c_{n-1} \cdots c_0)_b$$

- Ad ogni posizione nella stringa è associato un peso. Da destra a sinistra i pesi sono:

$$b^0, \dots, b^{n-1}$$

- Ogni cifra rappresenta il numero di volte in cui deve essere considerato il peso corrispondente alla posizione in cui si trova la cifra stessa

Forma polinomiale

- Possiamo dunque definire la seguente relazione detta *forma polinomiale*

$$(c_{n-1} \cdots c_0)_b = c_0 \times b^0 + \cdots + c_{n-1} \times b^{n-1}$$

- analogamente possiamo scrivere numeri frazionari:

$$(.c_{-1} \cdots c_{-m})_b = c_{-1} \times b^{-1} + \cdots + c_{-m} \times b^{-m}$$

- o numeri con parte intera e parte frazionaria:

$$(c_{n-1} \cdots c_0.c_{-1} \cdots c_{-m})_b = c_{n-1} \times b^{n-1} + \cdots + c_0 \times b^0 + c_{-1} \times b^{-1} + \cdots + c_{-m} \times b^{-m}$$

Nota

Il sistema di numerazione in base 10, o *sistema decimale*, è il sistema comunemente usato, ed i numeri in base 10 sono rappresentati di norma senza l'indicazione della base.

Sistema di numerazione in base 2

- E' il sistema di numerazione con la base più piccola possibile
- In questo caso le cifre sono $\{0,1\}$
- Si parla di cifra binaria (*binary digit* o *bit*)
- Il bit è l'unità minima di informazione.
- Conversione binario \rightarrow decimale tramite la sua forma polinomiale:

$$\begin{aligned}(1011.01)_2 &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} \\ &= (11.25)_{10}\end{aligned}$$

Conversione decimale \rightarrow binario di numeri interi

- Problema formulabile come segue: dato l'intero decimale N , determinare la stringa di bit $c_{n-1} \cdots c_2 c_1 c_0$ tale che

$$N = c_0 \times 2^0 + c_1 \times 2^1 + c_2 \times 2^2 + \cdots + c_{n-1} \times 2^{n-1}$$

- Si applica il metodo delle *divisioni successive*.
- Dividiamo entrambi i membri dell'uguaglianza per 2:

$$N/2 = c_0 \times 2^{-1} + c_1 \times 2^0 + c_2 \times 2^1 + \cdots + c_{n-1} \times 2^{n-2} = R \times 2^{-1} + Q$$

dove $R = N \bmod 2$ e $Q = \lfloor N/2 \rfloor$ sono rispettivamente resto e quoziente della divisione.

- Si ricava così che:

$$c_0 = R = N \bmod 2$$

$$Q = \lfloor N/2 \rfloor = c_1 \times 2^0 + c_2 \times 2^1 + \cdots + c_{n-1} \times 2^{n-2}$$

Conversione decimale \rightarrow binario di numeri interi

- Se $Q \neq 0$ possiamo ripetere la procedura:

$$\begin{aligned}Q &= c_1 \times 2^0 + c_2 \times 2^1 + \dots + c_{n-1} \times 2^{n-2} \\Q/2 &= c_1 \times 2^{-1} + c_2 \times 2^0 + \dots + c_{n-1} \times 2^{n-3} \\&= R' \times 2^{-1} + Q'\end{aligned}$$

- da cui

$$\begin{aligned}c_1 &= R' = Q \bmod 2 \\Q' &= \lfloor Q/2 \rfloor = c_2 \times 2^0 + \dots + c_{n-1} \times 2^{n-3}\end{aligned}$$

- Iterando finché si ottiene una divisione con quoziente nullo.

Esempio

- Convertire in binario il numero decimale $(61)_{10}$

61	$\text{mod } 2 = 1 = c_0$	(least significant bit),	$\lfloor 61/2 \rfloor =$	30
30	$\text{mod } 2 = 0 = c_1$		$\lfloor 30/2 \rfloor =$	15
15	$\text{mod } 2 = 1 = c_2$		$\lfloor 15/2 \rfloor =$	7
7	$\text{mod } 2 = 1 = c_3$		$\lfloor 7/2 \rfloor =$	3
3	$\text{mod } 2 = 1 = c_4$		$\lfloor 3/2 \rfloor =$	1
1	$\text{mod } 2 = 1 = c_5$	(most significant bit)	$\lfloor 1/2 \rfloor =$	0

- quindi $(61)_{10} = (111101)_2$

Conversione decimale \rightarrow binario di numeri frazionari

- Problema formulabile come segue: dato il numero frazionario decimale F , determinare la stringa di bit $c_{-1}c_{-2}c_{-3} \dots c_{-m}$ tale che

$$F = c_{-1} \times 2^{-1} + c_{-2} \times 2^{-2} + c_{-3} \times 2^{-3} + \dots + c_{-m} \times 2^{-m}$$

- Si applica il metodo delle *moltiplicazioni successive*.
- Moltiplichiamo entrambi i membri dell'uguaglianza per 2:

$$F \times 2 = c_{-1} + c_{-2} \times 2^{-1} + c_{-3} \times 2^{-2} + \dots + c_{-m} \times 2^{-m+1} = N + F'$$

dove $N = c_{-1}$ è la parte intera del risultato ed

$F' = c_{-2} \times 2^{-1} + c_{-3} \times 2^{-2} + \dots + c_{-m} \times 2^{-m+1}$ è la parte frazionaria.

Conversione decimale \rightarrow binario di numeri frazionari

- Si itera il procedimento sulla parte frazionaria finchè si verifica una delle due condizioni:
 1. Si raggiunge il numero massimo di cifre binarie con cui si intende rappresentare il numero frazionario (si ottiene un'*approssimazione per difetto* del numero decimale).
 2. Si ottiene come risultato di una moltiplicazione un numero con parte frazionaria nulla.

Operazioni aritmetiche binarie

- Tabella per la sottrazione binaria:

$$0 - 0 = 0$$

$$0 - 1 = 1 \quad \text{con prestito di } 1$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

- Esempio

$$\begin{array}{r} \\ 1 \\ \\ \hline 0 \end{array}$$

Sistema di numerazione in base 8 (ottale)

- Cifre usate: $\{0,1,2,3,4,5,6,7\}$
- Conversione ottale \rightarrow decimale tramite la forma polinomiale
- Conversione decimale \rightarrow ottale tramite divisioni (o moltiplicazioni per la parte frazionaria) successive per la base (8).

Esempi

- Conversione ottale \rightarrow decimale di $(754)_8$

$$(754)_8 = 4 \times 8^0 + 5 \times 8^1 + 7 \times 8^2 = (492)_{10}$$

- Conversione decimale \rightarrow ottale di $(678)_{10}$

678	$\text{mod } 8 = 6 = c_0$	$\lfloor 678/8 \rfloor$	=	84
84	$\text{mod } 8 = 4 = c_1$	$\lfloor 84/8 \rfloor$	=	10
10	$\text{mod } 8 = 2 = c_2$	$\lfloor 10/8 \rfloor$	=	1
1	$\text{mod } 8 = 1 = c_3$	$\lfloor 1/8 \rfloor$	=	0

quindi $(678)_{10} = (1246)_8$

Conversione tra binario ed ottale

- Si noti che ciascuna cifra ottale può essere rappresentata con tre bits:

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

Conversione tra binario ed ottale

Conversione binario \rightarrow ottale

1. Si raggruppano i bit a gruppi di tre da destra verso sinistra per la parte intera, da sinistra verso destra per la parte frazionaria.
2. Si aggiungono se necessario bit 0 a sinistra (per la parte intera) ed a destra (per la parte frazionaria) del numero.
3. Si sostituisce ogni gruppo di tre cifre binarie con la corrispondente cifra ottale.

$$\text{e.g. } (1011.1001)_2 \rightarrow (001\ 011\ .\ 100\ 100)_2 \rightarrow (13.44)_8$$

Conversione ottale \rightarrow binario

1. Si sostituisce ogni cifra ottale con il corrispondente gruppo di tre cifre binarie

Sistema di numerazione in base 16 (esadecimale)

- Cifre usate: {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}
- Dove vale la seguente conversione:

<u>cifra esadecimale</u>	<u>numero decimale</u>
A	10
B	11
C	12
D	13
E	14
F	15

- La conversione tra esadecimali e decimali è analoga a quelle già viste (forma polinomiale per passare a decimale, divisioni o moltiplicazioni per la base 16 per passare a esadecimale).

Conversione tra binario ed esadecimale

- Si noti che ciascuna cifra esadecimale può essere rappresentata con quattro bit:

0000	→ 0	0001	→ 1	0010	→ 2	0011	→ 3
0100	→ 4	0101	→ 5	0110	→ 6	0111	→ 7
1000	→ 8	1001	→ 9	1010	→ A	1011	→ B
1100	→ C	1101	→ D	1110	→ E	1111	→ F

- Le conversioni sono analoghe a quelle viste per la numerazione ottale:

- $(101111.01)_2 \rightarrow (0010\ 1111 . 0100)_2 \rightarrow (2F.4)_{16}$

- $(A3.E)_{16} \rightarrow (1010\ 0011 . 1110)_2 \rightarrow (10100011.111)_2$

Utilità dei sistemi ottale ed esadecimale

- Servono a rappresentare in maniera leggibile e concisa stringhe di bit.
- La numerazione ottale è stata introdotta in informatica quando i mainframe più diffusi usavano parole di 24 o 36 bit (divisibili per 3).
 - E' ancora diffusa per rappresentare i permessi sui file nei sistemi *Unix*: lettura (4), scrittura (2), esecuzione (1). Si sommano i valori dei permessi che si vogliono garantire (e.g. 6 = lettura e scrittura)
- Con la diffusione dei computer a 16, 32 e 64 bit (divisibili per 4) si è imposta la numerazione esadecimale.
 - Ad esempio si usa in HTML per rappresentare i colori a 24-bit nel formato RGB (#RRGGBB con RR valore della componente rossa, GG della verde e BB della blu, e.g. #FFFF00 = giallo)

Forma complemento

- Dato un intero N in base b con k cifre, il *complemento a b* di N , scritto $C_b(N)$ è il numero in base b tale che

$$N + C_b(N) = b^k$$

- Esempi:

- $b = 10, k = 4, N = 2553, C_{10}(2553) = 10000 - 2553 = 7447$
- $b = 2, k = 5, N = (10110)_2, C_2(10110) = (100000)_2 - (10110)_2 = (01010)_2$
- $b = 8, k = 3, N = (547)_8, C_8(547) = (1000)_8 - (547)_8 = (231)_8$

Calcolo di complementi

- Dato un intero N in base b con k cifre, il *complemento a $b-1$* di N , scritto $C_{b-1}(N)$ è il numero in base b tale che

$$N + C_{b-1}(N) = b^k - 1$$

- Si noti che per ogni base b , $b^k - 1$ è rappresentato con k cifre uguali a $b - 1$ (e.g. $10^3 - 1 = 999$)
- Quindi calcolare il complemento a $b - 1$ di qualsiasi numero in qualsiasi base non necessita mai operazioni di prendere a prestito:
 - $C_9(2553) = 9999 - 2553 = 7446$
 - $C_1(10110) = (11111)_2 - (10110)_2 = (01001)_2$
 - $C_7(547) = (777)_8 - (547)_8 = (230)_8$
- Vale la relazione $C_b(N) = C_{b-1}(N) + 1$, quindi basta sommare 1 al complemento a $b - 1$ per ottenere il complemento a b .

Regola pratica per il caso $b=2$

- $C_1(N) = (2^k - 1) - N$
- $(2^k - 1)$ in binario si scrive $1 \dots 1$ (k volte).
- Sottrarre N da $(2^k - 1)$ significa *negare* tutti i bits di N (sostituire 1 con 0 e viceversa).
- Per ottenere il complemento a 2 si somma 1 al risultato.
- Es. $(10100)_2$
 1. Si negano i bits, ottenendo $(01011)_2$
 2. Si somma 1, ottenendo $(01100)_2$

Sottrazioni con la forma complemento

- Vogliamo calcolare $N - M$ dove N ed M sono due numeri di k cifre in base b (se uno dei due ha meno cifre, gli si aggiungono zeri a sinistra).
- Valgono le seguenti relazioni:
$$N - M = N - M + b^k - b^k = N + (b^k - M) - b^k = N + C_b(M) - b^k$$
- Si può quindi calcolare la differenza sommando ad N il complemento a b di M e scartando la cifra più significativa del risultato (ossia b^k).
- Esempio: $(11001100)_2 - (11000010)_2$
 1. $C_2(11000010) = (00111101)_2 + (1)_2 = (00111110)_2$
 2. $(11001100)_2 + (00111110)_2 = (100001010)_2$
 3. Risultato = $(00001010)_2$