Laurea in Ingegneria Gestionale

Corso di Fondamenti di Informatica
A.A. 2017/2018

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Rappresentazione dell'Informazione - 1

Le seguenti slide sono rielaborate da slide analoghe del Prof. Diego Calvanese (Università di Bolzano)

Rappresentazione binaria dell'informazione

Per informazione intendiamo tutto quello che viene manipolato da un calcolatore:

- numeri (naturali, interi, reali, . . .)
- caratteri
- immagini (statiche, dinamiche)
- suoni
- programmi

In un calcolatore tutte le informazioni sono rappresentate in **forma binaria**, come sequenze di **0** e **1**.

Per **motivi tecnologici**: Nei calcolatori, gli elementi in grado di conservare informazioni sono "bistabili", cioè stabili in due stati (0 e 1), e le transizioni di stato sono forzate da un segnale di ingresso.

In sostanza, distinguere tra due valori di una grandezza fisica è più semplice che non ad esempio tra dieci valori.

Rappresentazione di numeri naturali

- Un numero naturale è un oggetto matematico, che può essere **rappresentato** mediante una **sequenza di simboli** di un alfabeto fissato.
- È importante distinguere tra numero e sua rappresentazione: il **numerale** "$2\,3\,4$" è la rappresentazione del numero 234.
- Si distinguono **2 tipi di rappresentazione**:

  additiva: ad es. le cifre romane

  posizionale: una cifra contribuisce con un valore diverso al numero a seconda della posizione in cui si trova

Noi consideriamo solo la rappresentazione posizionale.

Un numero è rappresentato da una **sequenza finita di cifre** di un certo **alfabeto**:

$$c_{n-1}c_{n-2}\cdots c_1c_0 \;=\; N_b$$

c_0 ..... viene detta cifra **meno significativa**

c_{n-1} ..... viene detta cifra **più significativa**

Il numero b di cifre diverse (dimensione dell'alfabeto) è detto **base** del sistema di numerazione. Ad ogni cifra è associato un valore compreso tra 0 e $b - 1$.

Base	**Alfabeto**	**Sistema**
2	$0, 1$	binario
8	$0, \ldots, 7$	ottale
10	$0, \ldots, 9$	decimale
16	$0, \ldots, 9, \mathtt{A}, \ldots, \mathtt{F}$	esadecimale

Il significato di una sequenza di cifre (il numero N che essa rappresenta) dipende da b:

$$c_{n-1} \cdot b^{n-1} + c_{n-2} \cdot b^{n-2} + \cdots + c_1 \cdot b^1 + c_0 \cdot b^0 \;=\; \sum_{i=0}^{n-1} c_i \cdot b^i \;=\; N$$

Esempio: Il numerale $\mathtt{101}$ rappresenta numeri diversi a seconda del sistema usato:

Sistema	**Base** b	$(101)_b$	**Valore** (in decimale)
binario	2	$(101)_2$	5
ottale	8	$(101)_8$	65
decimale	10	$(101)_{10}$	101
esadecimale	16	$(101)_{16}$	257

Intervallo di rappresentazione con n cifre in base b: **da** 0 **a** $b^n - 1$

Esempio:

3 cifre in base 10 :	da 0 a	$999 = 10^3 - 1$
8 cifre in base 2 (1 byte) :	da 0 a	$255 = 2^8 - 1$
16 cifre in base 2 (2 byte) :	da 0 a	$65\,535 = 2^{16} - 1$
32 cifre in base 2 (4 byte) :	da 0 a	$4\,294\,967\,296 = 2^{32} - 1$
2 cifre in base 16 :	da 0 a	$255 = 16^2 - 1$
8 cifre in base 16 :	da 0 a	$4\,294\,967\,296 = 16^8 - 1$

Conversioni di base

Conversione da base b a base 10

Usando direttamente

$$c_{n-1} \cdot b^{n-1} + c_{n-2} \cdot b^{n-2} + \cdots + c_1 \cdot b^1 + c_0 \cdot b^0 \;=\; \sum_{i=0}^{n-1} c_i \cdot b^i \;=\; N$$

esprimendo le cifre e b in base 10 (e facendo i conti in base 10)

Esercizio: Scrivere l'algoritmo di conversione da base 2 a base 10 e codificarlo in Python (soluzione disponibile nel file `conversioneBinarioDecimale.py`).

Conversione da base 10 a base b

In linea di pricipio potremmo usare lo stesso metodo precedente, ma le operazioni andrebbero fatte nella base di "arrivo", e quindi in base b. Vediamo invece un metodo di conversione basato du operazioni nella base di partenza, e quindi 10.

$$\begin{aligned} N &= c_0 + c_1 \cdot b^1 + c_2 \cdot b^2 + \cdots + c_{k-1} \cdot b^{k-1} \\ &= c_0 + b \cdot (c_1 + b \cdot (c_2 + \cdots + b \cdot (c_{k-1}) \cdots)) \end{aligned}$$

Vogliamo determinare le cifre $c_0, c_1, \ldots, c_{k-1}$

Consideriamo la divisione di N per b:

$$\begin{aligned} N &= R + b \cdot Q \qquad\qquad (0 \leq R < b) \\ &= c_0 + b \cdot (c_1 + b \cdot (\cdots)) \end{aligned}$$

$\Longrightarrow$ $R = c_0$ ovvero, il resto R della divisione di N per b dà c_0 (cifra meno significativa)
$Q = c_1 + b \cdot (\cdots)$

A partire dal quoziente Q si può iterare il procedimento per ottenere le cifre successive (fino a che Q diventa 0).

algoritmo converte N da base 10 a base b

$i \leftarrow 0$ # $\leftarrow$ indica l'assegnazione (denotata in Python da =)
while $N \neq 0$ # $\neq$ indica disuguaglianza (denotata in Python da !=)
do $c_i \leftarrow N$ mod b # mod indica la funzione modulo (denotata in Python da %)
$N \leftarrow N$ div b # div indica la divisione intera (denotata in Python da //)
$i \leftarrow i + 1$

N.B. Le cifre vengono determinate dalla meno significativa a quella più significativa.

Esempio: $(25)_{10} = (???)_2$

$N : b$	Q	R	cifra
$25 : 2$	12	1	c_0
$12 : 2$	6	0	c_1
$6 : 2$	3	0	c_2
$3 : 2$	1	1	c_3
$1 : 2$	0	1	c_4

$(25)_{10} = (11001)_2$

N.B. servono 5 bit (con cui possiamo rappresentare i numeri da 0 a 31)

Esercizio: Scrivere un progamma Python per la conversione da base 10 a base 2 (soluzione disponibile nel file `conversioneDecimaleBinario.py`).

Rappresentazione di numeri interi

dobbiamo rappresentare anche il **segno** $\Longrightarrow$ si usa uno dei bit (quello più significativo)

Rappresentazione tramite modulo e segno

- il bit più significativo rappresenta il segno
- le altre $n-1$ cifre rappresentano il valore assoluto
- **problemi**:
  - doppia rappresentazione per lo zero ($00\cdots00$ e $10\cdots00$)
  - le operazioni aritmetiche sono complicate

$\Longrightarrow$ invece della rappresentazione tramite modulo e segno si usa una rappresentazione in complemento

Rappresentazione in complemento

In quanto segue:

- b indica la base
- n indica il numero complessivo di cifre
- consideriamo solo numeri in valore assoluto $\leq b^n$

Residuo modulo b^n di un intero X:

$$|X|_{b^n} = X - \lfloor X/b^n \rfloor \cdot b^n$$

dove $\lfloor Y \rfloor$ indica la parte intera inferiore di Y

Esempio: $b = 10, \quad n = 2$

$$\begin{aligned} |25|_{10^2} &= 25 - \lfloor 25/10^2 \rfloor \cdot 10^2 = 25 - 0 = 25 \\ |-25|_{10^2} &= -25 - \lfloor -25/10^2 \rfloor \cdot 10^2 = -25 - (-1) \cdot 100 = 75 \end{aligned}$$

$\Longrightarrow$ per un numero positivo, il residuo è pari al numero stesso
per un numero negativo, il residuo è pari al complemento (del suo valore assoluto) rispetto a b^n

- ad un numero corrisponde un unico residuo
- ad un residuo
  - corrispondono in generale due numeri, uno positivo ed uno negativo
  - corrisponde però un unico numero nell'intervallo $[-b^n/2,\ b^n/2)$

$\Longrightarrow$ **definizione semplificata di residuo**:

$$|X|_{b^n} = \begin{cases} X, & \text{se } 0 \leq X < b^n/2 \\ b^n - |X|, & \text{se } -b^n/2 \leq X < 0 \end{cases}$$

Il residuo viene utilizzato per la rappresentazione in complemento alla base.

Rappresentazione in complemento alla base (b con n cifre)

- è la rappresentazione di interi relativi nell'intervallo $[-b^n/2,\ b^n/2)$ tramite il residuo modulo b^n
  - se X è nell'intervallo rappresentabile e $X \geq 0$: RCB di X è compresa in $[0,\ b^n/2)$
  - se X è nell'intervallo rappresentabile e $X < 0$: RCB di X è compresa in $[b^n/2,\ b^n)$
- lo 0 ha una sola rappresentazione
- se $b = 2$ $\Longrightarrow$ **rappresentazione in complemento a 2**
  - positivi: cifra più significativa è 0 (rappresentati nella parte inferiore dell'intervallo)
  - negativi: cifra più significativa è 1 (rappresentati nella parte superiore dell'intervallo)

Esempio: $b = 2, \quad n = 6, \quad X = -9$

$$b^n - |X| = 64_{10} - 9_{10} = 55_{10} = 110111_2$$

La rappresentazione in complemento a due di -9 con 6 cifre è quindi 110111.

Osservazione:

$$\begin{array}{rcll} 2^n - |X| & = & 2^n - |X| + 1 - 1 & \\ & = & 2^n - 1 & n \text{ uni} \\ & & -|X| & \text{inverto i bit di } |X| \\ & & +1 & \text{sommo } 1 \end{array}$$

Esempio: $b = 2, \quad n = 6, \quad X = -9$

$$\begin{array}{rl} (9)_{10} = (001001)_2 & \\ \text{inverto i bit:} & 110110 \\ \text{sommo } 1\text{:} & 110111 \end{array}$$

Metodo ancora più rapido:

1. si rappresenta $|X|$ con n bit
2. a partire da destra si lasciano inalterate tutte le cifre fino al primo 1 compreso
3. si invertono le rimanenti cifre

Per ottenere un numero X (rappresentandolo in decimale) a partire dalla sua rappresentazione R in complemento a 2, procediamo come segue:

- se R inizia per 0 (cioè 0 è la cifra più significativa), allora X è positivo e si ottiene applicando la formula di conversione da binario a decimale vista per i numeri naturali;
- se invece R inizia per 1 (cioè 1 è la cifra più significativa), allora X è negativo e quindi vale $b^n - |X| = R$, dove n indica il numero di cifre in R.

Esempio: $R = 1101$, quindi $n = 4$ ed il numero X è negativo

$$2^4 - |X| = (1101)_2 \rightarrow |X| = 2^4 - 13 = 3.$$

si conclude quindi che $X = -3$

Operazioni su interi relativi in complemento a 2

Somma di due numeri

- si effettua **bit a bit**
- non è necessario preocuparsi dei segni
- il risultato sarà corretto (in complemento a 2 se negativo)

- può verificarsi **trabocco** (overflow) $\implies$ il risultato non è corretto

  Si verifica quando il numero di bit a disposizione non è sufficiente a rappresentare il risultato.

Esempio: $n = 5$, $\quad \pm 9 \pm 3$, $\quad \pm 9 \pm 8$

intervallo di rappresentazione: da -2^4 a $2^4 - 1$ (ovvero, da -16 a 15)

riporto		00011		11001		00111		11111
	$+9$	01001	$+9$	01001	-9	10111	-9	10111
	$+3$	00011	-3	11101	$+3$	00011	-3	11101
	$+12$	01100	$+6$	00110	-6	11010	-12	10100

In questo caso non si ha trabocco.

riporto		01000		10000
	$+9$	01001	-9	10111
	$+8$	01000	-8	11000
	-15	10001	$+15$	01111
	(e non $+17$)		(e non -17)	

In questo caso si ha trabocco.

Si ha **trabocco** quando il riporto sul bit di segno è diverso dal riporto dal bit di segno verso l'esterno.

Differenza tra due numeri: sommo al primo il complemento del secondo