Fondamenti di Informatica

a.a. 2007/2008

Demis Ballis <demis@dimi.uniud.it>
Dipartimento di Matematica e Informatica
Università degli Studi di Udine

Obiettivi

- conoscere il calcolatore (parte HWe SW)
- apprendere le basi della programmazione (imparare a formalizzare e risolvere problemi con l'ausilio del calcolatore)

L'informazione

- Diverse proposte in letteratura
  - Bateson, "Ricevere informazioni vuol dire necessariamente ricevere notizie di differenza" (Es.: Luce accesa/spenta, 1/0,...)
  - Shannon, visione probabilistica. Un evento porta molta informazione se e' poco probabile. ("Al polo nord fa freddo").

Codifica dell'informazione

- Gli algoritmi sono costituiti da istruzioni (blocchi sequenziali, condizionali, iterativi) che operano su dati.
- Per trasformare un programma in una descrizione eseguibile da un calcolatore è necessario rappresentare istruzioni e dati in un formato memorizzabile e facilmente manipolabile.

Codifica dell'informazione

- Rappresenteremo l'informazione numerica e non numerica mediante sequenze di simboli scelti da un insieme chiamato alfabeto.

- Ad ogni alfabeto è associato un insieme di regole di composizione che consentono di costruire le successioni di simboli. Successioni che sono costruite mediante le regole di composizione si dicono "ben formate".

Esempio

Rappresentazione usuale dei numeri frazionari in base 10

Alfabeto={"0","1","2","3","4","5","6","7","8","9", ","}

103,56 successione ben formata
12,45,67 successione non ben formata

Codifica dell'informazione

- Oltre a dare delle regole di composizione per rappresentare l'informazione (parte sintattica), è necessario assegnare all'informazione un significato mediante dei codici (parte semantica).

Esempio

103,56 significa $1*10^{2}+0*10^{1}+3*10^{0}+5*10^{-1}+6*10^{-2}$

Codifica dell'informazione

- Dato un alfabeto di n simboli, si possono generare n^k successioni di k simboli.
- Generalmente non tutte le successioni codificano dell'informazione (solo le successioni ben formate)

Esempio

Con un alfabeto di 10 simboli si possono ottenere 10^4=10.000 sequenze di lunghezza 4.

Codifica binaria

- Un alfabeto piuttosto semplice e allo stesso tempo molto comodo è l'alfabeto binario, il quale è composto da soli due simboli.
- $A_2=\{0,1\}$
- I calcolatori utilizzano per la memorizzazione dell'informazione dei dispositivi bistabili (bit - Binary digIT), in grado di assumere due sole configurazioni.
- Nell'ipotesi che ogni successione di bit sia ben formata, utilizzando l'alfabeto A_2 e k bit si possono ottenere 2^k successioni diverse.

Codifica binaria

- 8 bit = 1 Byte
- 2^{10} Byte = 1 KByte
- 2^{10} KByte = 2^{20} Byte = 1 MByte
- 2^{10} MByte = 2^{20} KByte = 2^{30} Byte = 1 GByte

Dati non numerici

- Dati non numerici esempi: caratteri alfanumerici ('a','b','=',...), sequenze di caratteri ("pippo", "mamma", ...).
- Problema: come codificare mediante un codice binario dei dati non numerici?

Dati non numerici

- Dato un insieme di dati non numerici S costituito da #S elementi, qual è il numero minimo di bit per codificare tutti gli elementi mediante una codifica binaria che sia univoca?

- Sappiamo che con k bit possiamo codificare $n=2^k$ elementi. Se n=#S allora k è uguale al tetto di $\log_2$ #S, dove il tetto di un numero p è il minimo intero superiore a p.

- Esempio: S= {Lun,Mar,Mer,Gio,Ven,Sab,Dom}, #S=7. Quindi $\log_2$ #S

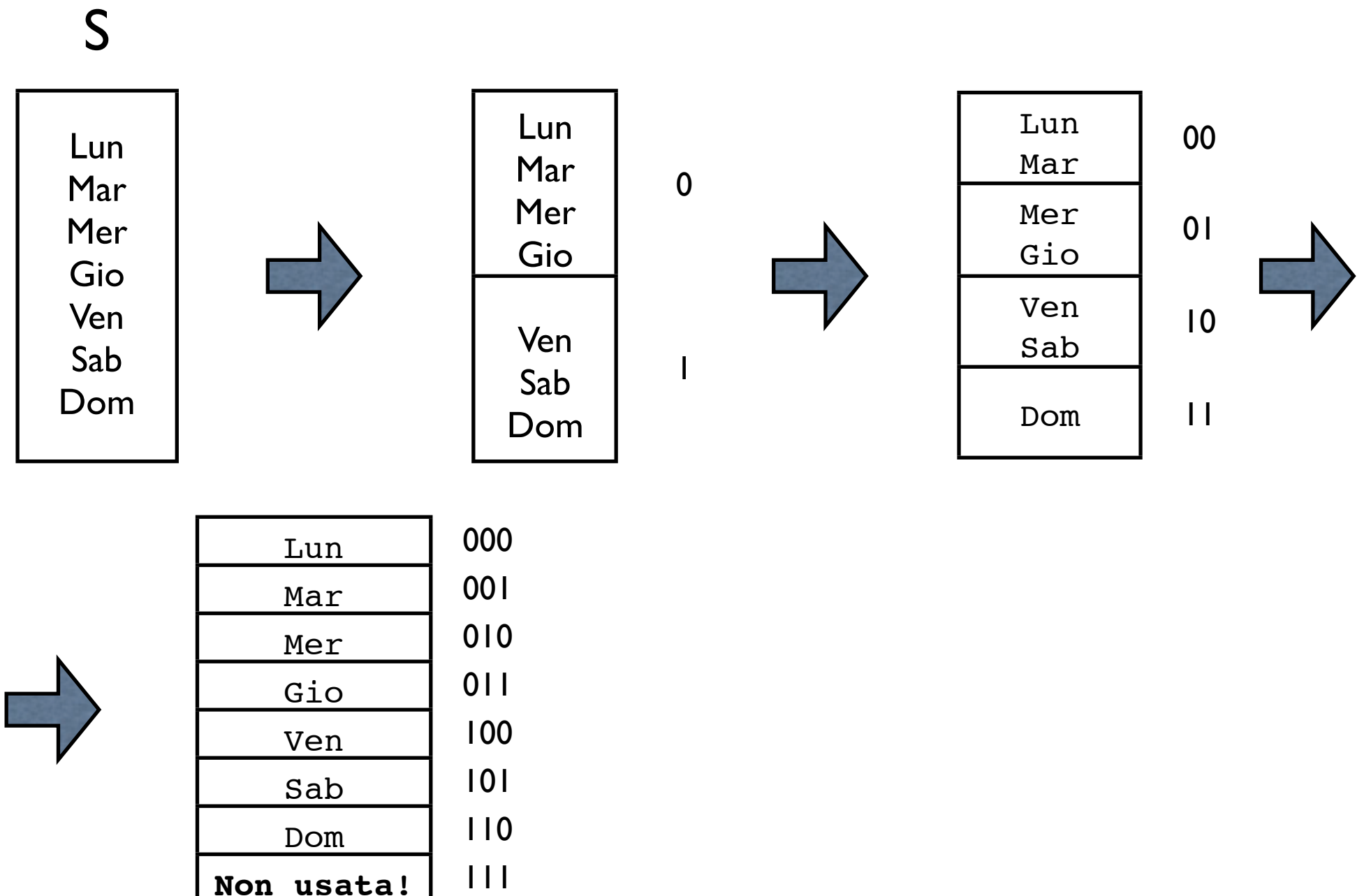
$$\text{tetto}(\log_2 \#S)=\text{tetto}(\log_2 7)= 3$$

Algoritmo per la codifica dei dati non numerici

Problema: dato un insieme S di dati non numerici, determinare una codifica binaria univoca per gli elementi di S.

Soluzione: Dato un insieme S da codificare, un singolo bit permette di distinguere i dati in due sottoinsiemi. Iterando tale procedimento sui sottoinsiemi identificati, siamo in grado di generare delle successioni di bit che identificano in maniera non ambigua i singoli elementi di S.

Esempio


S
Lun
Mar
Mer
Gio
Ven
Sab
Dom
Lun
Mar
Mer
Gio
Ven
Sab
Dom
0
1
Lun
Mar
Mer
Gio
Ven
Sab
Dom
00
01
10
11
Lun
Mar
Mer
Gio
Ven
Sab
Dom
Non usata!
000
001
010
011
100
101
110
111

Dati non numerici

- Volendo realizzare una codifica binaria per documenti scritti in linguaggio naturale bisogna prima stabilire il numero di simboli necessari. (Es. 26 lettere minuscole, 26 lettere maiuscole, segni di interpunzione, 10 cifre decimali,caratteri speciali...)
- Sono circa 120 simboli, per la cui codifica sono necessari almeno 7 bit.

Dati non numerici

- Codifica ASCII (128 caratteri, 7 bit)
- Codifica ASCII estesa (256 caratteri, 8 bit)
- codifica UNICODE (65535 caratteri, 16 bit)
  - i primi 128 caratteri della codifica UNICODE corrispondono al codice ASCII
- Anche UNICODE non è sufficiente :-(

Istruzioni

- I programmi sono codificati mediante un linguaggio di basso livello chiamato linguaggio macchina.
- Anche le istruzioni del linguaggio macchina sono codificate mediante un codice binario.
  - assegnare un codice univoco ad ogni singola istruzione (opcode)
  - oltre all'opcode è necessario codificare gli eventuali riferimenti agli operandi necessari all'esecuzione dell'istruzione.
  - il numero di operandi dipende dal tipo di istruzione. (ES: l'istruzione ADD (somma) ha due operandi, l'istruzione HALT ha 0 operandi).

Codifica dei dati numerici

- Codifica dei numeri senza segno
  - rappresentzione dei numeri senza segno in base B, B>1.
  - conversioni di base:
    - Da base 10 a base B, B>1
    - Da base B, B>1 a base 10
    - Da base B, B>1 a base B', B'>1
    - Conversioni rapide:
      - da base 8 a base 2 e viceversa
      - da base 16 a base 2 e viceversa

Notazione

Una base B di dimensione n e' un insieme di n cifre {0,1,... n-1}

Esempi:

{0,1} base 2
{0,1,2} base 3
{0,1,2,3,4,5,6,7} base 8
{0,1,2,3,4,5,6,7,8,9} base 10
{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F} base 16

$(c_{n-1}\ c_{n-2}\ \ldots\ c_{n-1}\ c_1\ c_0)_B$ rappresenta un numero senza segno di n cifre in base B.

Conversioni di base

Problema: convertire un numero senza segno da base B a base 10.

Soluzione:

dato il numero senza segno in base B $(c_{n-1}\ c_{n-2}\ \ldots\ c_{n-1}\ c_1\ c_0)_B$ il corrispondente valore in base 10 si calcola con la seguente formula:

$$c_{n-1}*2^{n-1} + c_{n-2}*2^{n-2} + \ldots + c_1*2^1 + c_0*2^0$$

Esempio: $(101011)_2$ corrisponde a $(43)_{10}$, infatti

$$1*2^5+0*2^4+1*2^3+0*2^2+1*2^1+1*2^0 = 32+0+8+0+2+1 = 43$$

Conversioni di base

Problema: convertire un numero senza segno da base 10 a base B.

Soluzione:

dato un numero senza segno X in base 10, si deve calcolare il corrispondente valore in base B $(c_{n-1}\ c_{n-2}\ \ldots\ c_{n-1}\ c_1\ c_0)_B$.

Dividendo X per la base B, si ottiene come resto la cifra c_0 e un certo quoziente Y1.
Dividendo Y1 per la base B, si ottiene come resto la cifra c_1 e un certo quoziente Y2.
Iterando il procedimento fino a raggiungere il quoziente 0,
si ottiene il numero in base B cercato.

Conversioni di base

Esempio: $(12)_{10}$ corrisponde a $(112)_3$, infatti

14:3=4	4:3=1	1:3=0
2	1	1
c_0	c_1	c_2

Esempio: $(12)_{10}$ corrisponde a $(1100)_2$, infatti

12:2=6	6:2=3	3:2=1	1:2=0
0	0	1	1
c_0	c_1	c_2	c_3

Conversioni di base

Problema: convertire un numero senza segno da base B1 a base B2.

Soluzione: convertire prima il numero da base B1 a base 10 e in seguito convertire il valore risultante da base 10 a base B2.

Esempio: $(101011)_2$ corrisponde a $(1121)_3$, infatti

$(101011)_2 = (43)_{10} = (1121)_3$

Conversioni rapide

In alcuni casi passare da una base B1 a una base B2 può essere molto semplice. È questo il caso delle conversioni da base 8 a base 2, da base 2 a base 8, da base 16 a base 2 e da base 2 a base 16

Problema: convertire un numero senza segno da base 8 a base 2.

Soluzione: convertire indipendentemente ogni singola cifra del numero in base 8 in tre bit.

Problema: convertire un numero senza segno da base 16 a base 2.

Soluzione: convertire indipendentemente ogni singola cifra del numero in base 16 in 4 bit.

Conversioni rapide

Esempio: $(766401)_8$ corrisponde a

$(111\ 110\ 110\ 100\ 000\ 001)_2$
$7\ \ 6\ \ \ 6\ \ \ 4\ \ \ \ 0\ \ \ \ 1$

Esempio: $(A01F)_{16}$ corrisponde a

$(1010\ 0000\ 0001\ 1111)_2$
$A\ \ \ \ 0\ \ \ \ \ 1\ \ \ \ F$

Conversioni rapide

Problema: convertire un numero senza segno da base 2 a base 8.

Soluzione: raggruppare, da destra verso sinistra, i bit del numero in base 2 in gruppi di 3 bit e convertire indipendentemente ogni singolo gruppo di bit in una cifra ottale.

Problema: convertire un numero senza segno da base 2 a base 16.

Soluzione: raggruppare, da destra verso sinistra, i bit del numero in base 2 in gruppi di 4 bit e convertire indipendentemente ogni singolo gruppo di bit in una cifra esadecimale.

Conversioni rapide

Esempio: $(10101011)_2$ corrisponde a

$(\ 2 \qquad 5 \qquad 3)_8$
010 101 011

Esempio: $(1110101011)_2$ corrisponde a

$(\ 3 \qquad A \qquad B\ )_{16}$
0011 1010 1011

Codifica dei dati numerici

- Codifica dei numeri interi (bisogna codificare numero e segno)

  0, ±1, ±2,±3,±4,...

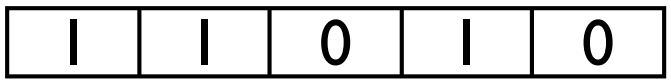
- Due metodi:
  - codifica con modulo e segno
  - codifica in complemento a due

Modulo e segno

- Codifica del segno seguita dalla codifica del valore assoluto
  - Si utilizza il primo bit (bit più significativo) per codificare il segno: 0 codifica "+", 1 codifica "-"
  - I restanti bit si utilizzano per codificare il valore assoluto (i.e. numero senza segno)

Modulo e segno

-10 (su 5 bit) ⇨



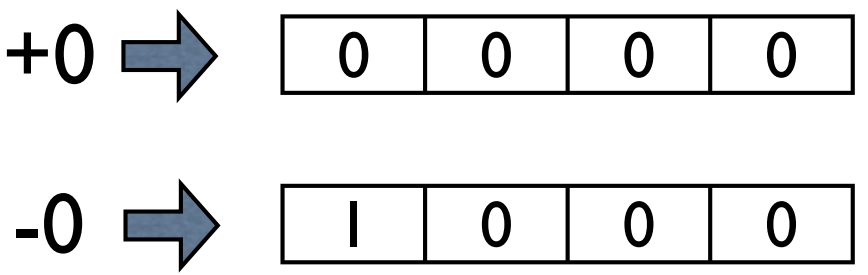
-10 (su 7 bit) ⇨

- Considerando n bit, questa codifica permette di rappresentare i numeri interi

$$\text{da } -(2^{n-1}-1) \text{ a } (2^{n-1}-1)$$

Modulo e segno

- Problema: doppia rappresentazione dello 0 (+0 e -0!)
  - spreco di spazio (si sprecano due successioni di bit per codificare lo stesso valore)
  - gestione delle operazioni di controllo e aritmetiche più complessa



Somme e sottrazioni in base 2

+	**0**	**1**
0	0	1
1	1	0 (riporto 1)

-	**0**	**1**
0	0	1 (prestito 1)
1	1	0

Esempi

```
 10111011 +
 11010011
----------
110001110
```

```
 10111011 +
        1
----------
 10111100
```

```
 10111011 -
 10000101
----------
 00110110
```

```
 10000000 -
        1
----------
 01111111
```

Complemento a 1

Dato un numero binario $(X)_2$, il

complemento a 1 di $(X)_2$ è un numero binario $(Y)_2$ che si ottiene invertendo i bit di $(X)_2$.

Esempio:

il complemento a 1 di $(00111011)_2$ è $(11000100)_2$

Complemento a 2

- Unica rappresentazione dello 0!
- Data una successione di n bit, per codificare un numero intero X, si utilizza il valore binario corrispondente a (2^n+X).
- Es: codifica su 4 bit => $2^4=16$

+6 = 16+6 = +22 => 10110 

0	1	1	0

-6 = 16-6 = +10 => 1010 

1	0	1	0

Complemento a 2

- Fatti:
  - il bit più significativo di un numero positivo in complemento a due è sempre 0
  - il bit più significativo di un numero negativo in complemento a due è sempre 1
  - Considerando n bit, questa codifica permette di rappresentare i numeri interi da -2^{n-1} a $(2^{n-1}-1)$
- Problema: dato un numero intero codificato in complemento a due, determinare il corrispondente valore in base 10

Complemento a 2

Problema: Convertire un numero codificato in complemento a 2 $(X)_2$ nel corrispondente valore decimale.

Soluzione:

Se $(X)_2 >= 0$
    stesso procedimento della codifica modulo e segno
Se $(X)_2 < 0$
  1. Calcolare il complemento a 1 di $(X)_2$ e chiamarlo $(Y)_2$
  2. Sommare 1 a $(Y)_2$, convertire in decimale il risultato, e chiamarlo $(Z)_{10}$
  3. Il valore corrispondente è $-(Z)_{10}$

Esempi

Caso numero negativo

Il valore decimale di $(1011)_2$ è $(-5)_{10}$ infatti...

complemento a 1 $0100 +$

$$\begin{array}{r} 1 \\ \hline 0101 \end{array}$$

e convertendo $(0101)_2$ si ottiene 5.

Caso numero positivo

Il valore decimale di $(0011)_2$ è $(+3)_{10}$ infatti...

convertendo $(011)_2$ si ottiene 3.

Compressione dei dati

- Finora non abbiamo valutato il costo e l'efficienza delle codifiche scelte.
- La compressione dei dati permette di ridurre il numero di bit necessari alla codifica di un insieme di elementi di informazione.
  - costo di memorizzazione minore
  - trasmissione dati più veloce

Esempio

- supponiamo di avere l'alfabeto A={A,C,G,T} e una successione S di 1.000.000 di caratteri scelti in A.
- Possibile codifica binaria dell'alfabeto: A=00, C=01, G=10, T=11 => la codifica di S prevede l'uso di 2.000.000 di bit.
- Altra possibilità: assegnare codici binari (non ambigui!!) di lunghezza variabile agli elementi dell'alfabeto, in base alla loro frequenza relativa rispetto a S (più la frequenza è alta, più il codice sarà corto). Se A si presenta in S il 50% dei casi, C il 25%, G e T il 12.5%, allora, per esempio,

A=0, C=10, G=110, T=111

in questo caso, la codifica di S risulta più corta, infatti

(1*0.5+2*0.25+3*0.125+3*0.125) bit/carattere * 1.000.000 caratteri = 1.750.000 bit

Compressione dei dati

- due tipi di algoritmi di compressione
  - lossless (senza perdita di informazione)
    - tutta l'informazione è importante e non si può perdere!
    - Es: slide precedente, documenti word, programmi eseguibili,...
  - lossy (con perdita di informazione)
    - si è disposti a perdere informazione pur di ottenere un maggior tasso di compressione
    - Es.: formati per immagini (JPEG, GIF,...), formati audio (MP3), formati video (famiglia MPEG)