

RAPPRESENTAZIONE DELL'INFORMAZIONE

- Internamente a un elaboratore, ogni informazione è rappresentata tramite *sequenze di bit (cifre binarie)*
- Una sequenza di bit *non dice “che cosa” essa rappresenta*

Ad esempio, 01000001 può rappresentare:

- l'intero 65, il carattere 'A', il booleano 'vero', ...
- ... il valore di un segnale musicale,
- ... il colore di un puntino sullo schermo...

1

INFORMAZIONI NUMERICHE

- La rappresentazione delle *informazioni numeriche* è di particolare rilevanza
- In questa sede ci limiteremo ai *numeri naturali (interi senza segno)*

Dominio: $N = \{ 0,1,2,3, \dots \}$

2

NUMERI NATURALI (interi senza segno)

Dominio: $N = \{ 0,1,2,3, \dots \}$

Rappresentabili con diverse notazioni

◆ **non posizionali**

- ad esempio la notazione romana:
I, II, III, IV, V, IX, X, XI...
- Risulta difficile l'utilizzo di regole generali per il calcolo

◆ **posizionale**

- 1, 2, .. 10, 11, ... 200, ...
- Risulta semplice l'individuazione di regole generali per il calcolo

3

NOTAZIONE POSIZIONALE

- Concetto di **base di rappresentazione B**
- Rappresentazione del numero come **sequenza di simboli (cifre)** appartenenti a un **alfabeto** di B simboli distinti
- **ogni simbolo rappresenta un valore compreso fra 0 e $B-1$**

Esempio di rappresentazione su N cifre:

$$d_{n-1} \dots d_2 d_1 d_0$$

4

NOTAZIONE POSIZIONALE

Il valore di un numero espresso in questa notazione è ricavabile

- ◆ a partire dal valore rappresentato da ogni simbolo
- ◆ pesandolo in base alla posizione che occupa nella sequenza



5

NOTAZIONE POSIZIONALE

In formula:

dove
$$v = \sum_{k=0}^{n-1} d_k B^k$$

- ◆ $B = \text{base}$
- ◆ ogni cifra d_k rappresenta un valore fra 0 e $B-1$

Esempio (base B=4):

$$\begin{array}{cccc} 1 & 2 & 1 & 3 \\ d_3 & d_2 & d_1 & d_0 \end{array}$$

$$\text{Valore} = 1 * B^3 + 2 * B^2 + 1 * B^1 + 3 * B^0 = \text{centotre}$$

6

NOTAZIONE POSIZIONALE

Quindi, ***una sequenza di cifre non è interpretabile*** se non si precisa la base in cui è espressa

Esempi:

Stringa	Base	Alfabeto	Calcolo valore	Valore
"12"	<i>quattro</i>	{0,1,2,3}	$4 * 1 + 2$	<i>sei</i>
"12"	<i>otto</i>	{0,1,...,7}	$8 * 1 + 2$	<i>dieci</i>
"12"	<i>dieci</i>	{0,1,...,9}	$10 * 1 + 2$	<i>dodici</i>
"12"	<i>sedici</i>	{0,..,9, A,., F}	$16 * 1 + 2$	<i>diciotto</i>

7

NOTAZIONE POSIZIONALE

Inversamente, ogni numero può essere espresso, ***in modo univoco, come sequenza di cifre in una qualunque base***

Esempi:

Numero	Base	Alfabeto	Rappresentazione
<i>venti</i>	<i>due</i>	{0,1}	"10100"
<i>venti</i>	<i>otto</i>	{0,1,...,7}	"24"
<i>venti</i>	<i>dieci</i>	{0,1,...,9}	"20"
<i>venti</i>	<i>sedici</i>	{0,..,9, A,., F}	"14"

8

CONCLUSIONE

Quindi:

- *i numeri sono concetti, che esistono in quanto tali*
- *la loro **rappresentazione** può invece variare a seconda delle convenzioni adottate*

Non bisogna confondere un numero con una sua RAPPRESENTAZIONE!

9

NUMERI E LORO RAPPRESENTAZIONE

- Internamente, un elaboratore adotta per i numeri interi (non negativi) una **rappresentazione binaria** (base $B=2$)
- Esternamente, le costanti numeriche che scriviamo nei programmi e i valori che stampiamo a video / leggiamo da tastiera sono invece **sequenze di caratteri ASCII**

Il passaggio dall'una all'altra forma richiede dunque un processo di **conversione**

10

Esempio: RAPPRESENTAZIONE INTERNA/ESTERNA

- Numero: *centoventicinque*
- Rappresentazione interna binaria (16 bit):

00000000 01111101

- Rappresentazione esterna in base 10:

occorre produrre la sequenza di caratteri ASCII '1', '2', '5'

00110001 00110010 00110101

vedi tabella ASCII

Esempio: RAPPRESENTAZIONE INTERNA/ESTERNA

- Rappresentazione esterna in base 10:

È data la sequenza di caratteri ASCII '3', '1', '2', '5', '4'

vedi tabella ASCII

00110011 00110001 00110010 00110101 00110100

- Rappresentazione interna binaria (16 bit):

01111010 00010110

- Numero:
trentunomiladuecentocinquantaquattro

CONVERSIONE STRINGA/NUMERO

Si applica la definizione:

$$v = \sum_{k=0}^{n-1} d_k B^k$$

le cifre d_k sono note,
il valore v va calcolato

$$= d_0 + B * (d_1 + B * (d_2 + B * (d_3 + ...)))$$

Ciò richiede la valutazione di un polinomio

→ **Metodo di Horner**

13

CONVERSIONE NUMERO/STRINGA

- Problema: **dato un numero, determinare la sua rappresentazione in una base data**
- Soluzione (**notazione posizionale**): **manipolare la formula** per dedurre un algoritmo

$$v = \sum_{k=0}^{n-1} d_k B^k$$

v è noto,
le cifre d_k vanno calcolate

$$= d_0 + B * (d_1 + B * (d_2 + B * (d_3 + ...)))$$

14

CONVERSIONE NUMERO/STRINGA

- Per trovare le cifre bisogna *calcolarle una per una*, ossia...
- ... bisogna trovare un modo per *isolarne una dalle altre*

$$v = d_0 + B * (...)$$

Osservazione:

d₀ è la sola cifra non moltiplicata per B

Conseguenza:

d₀ è ricavabile come v modulo B

15

CONVERSIONE NUMERO/STRINGA

Algoritmo delle divisioni successive

- **si divide v per B**
 - *il resto costituisce la cifra meno significativa (d₀)*
 - *il quoziente serve a iterare il procedimento*
- **se tale quoziente è zero, l'algoritmo termina;**
- **se non lo è, lo si assume come nuovo valore v', e si itera il procedimento con il valore v'**

16

CONVERSIONE NUMERO/STRINGA

Esempi

Numero	Base	Calcolo valore	Stringa
<i>quattordici</i>	4	14 / 4 = 3 con resto 2 3 / 4 = 0 con resto 3	“32”
<i>undici</i>	2	11 / 2 = 5 con resto 1 5 / 2 = 2 con resto 1 2 / 2 = 1 con resto 0 1 / 2 = 0 con resto 1	“1011”
<i>sessantatre</i>	10	63 / 10 = 6 con resto 3 6 / 10 = 0 con resto 6	“63”
<i>sessantatre</i>	16	63 / 16 = 3 con resto 15 3 / 16 = 0 con resto 3	“3F”

17

NUMERI NATURALI: valori rappresentabili

- **Con N bit, si possono fare 2^N combinazioni**
- **Si rappresentano così i numeri da 0 a 2^N-1**

Esempi

☐ **con 8 bit, [0 255]**

In C: unsigned char = byte

☐ **con 16 bit, [0 65.535]**

In C: unsigned short int (su alcuni compilatori)

In C: unsigned int (su alcuni compilatori)

☐ **con 32 bit, [0 4.294.967.295]**

In C: unsigned int (su alcuni compilatori)

In C: unsigned long int (su molti compilatori)

18

OPERAZIONI ED ERRORI

La rappresentazione binaria rende possibile fare *addizioni e sottrazioni con le usuali regole algebriche*

Esempio:

5 +	0101
3 =	0011
---	-----
8	1000

19

ERRORI NELLE OPERAZIONI

Esempio (supponendo di avere solo 7 bit per la rappresentazione)

60 +	0111100
75 =	1100011
-----	-----
135	1 0011111

Errore!
Massimo numero rappresentabile:
 2^7-1 cioè 127

- Questo errore si chiama *overflow*
- Può capitare *sommando due numeri dello stesso segno* il cui risultato non sia rappresentabile utilizzando il numero massimo di bit designati

20