I METODI DI NUMERAZIONE

I numeri naturali

... sono rappresentati ricorrendo a simboli che sintetizzano il concetto di numerosità.

Il numero dei simboli usati per valutare la numerosità costituisce la ***base*** della "numerazione".

Gli Egizi sin dal 3000 a.C. usavano una numerazione che prevedeva i simboli 1, 10, 100, 1000 e 10000.

I Babilonesi avevano due numerazioni: una in base 60 per scopi scientifici ed una in base 10 per scopi commerciali.

I Maya, circa 2000 anni fa, usavano una numerazione in base 18.

Nell'America del nord venivano usate numerazioni in base 3, 4, 5, 6, 8 e 20.

I Romani usavano una numerazione che prevedeva, al crescere del numero, l'introduzione di nuovi simboli.

IL SISTEMA DI NUMERAZIONE

Perché una numerazione costituisca un ***sistema di numerazione***, i numeri devono:

- essere rappresentabili tramite un insieme finito e non nullo di simboli (***alfabeto numerico***);
- possedere la proprietà della ***posizionalità dei simboli***, in quanto ogni simbolo deve possedere un significato legato alla posizione che esso occupa nel numero
- prevedere le ***operazioni aritmetiche***.

Un numero ad ***n*** cifre nel sistema di numerazione in base ***b***

$$a_{n-1} \cdot b^{n-1} + a_{n-2} \cdot b^{n-2} + \ldots + a_2 \cdot b^2 + a_1 \cdot b^1 + a_0 \cdot b^0 = \sum_{i=0,n-1} a_i \cdot b^i$$

ove gli $0 \leq \mathbf{a_i} \leq (b - 1)$ sono simboli dell'alfabeto numerico

IL SISTEMA DI NUMERAZIONE ARABO O DECIMALE

... è una evoluzione del sistema usato in India nel 3° secolo a.C., che non prevedeva il simbolo "0" e che non era effettivamente posizionale.

- l'***alfabeto numerico*** {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} ➔ base 10
- la ***posizionalità*** 354 ➔ $3 \cdot 10^2 + 5 \cdot 10^1 + 4 \cdot 10^0$
- le ***operazioni aritmetiche*** ➔ **+ - . :**

LA CODIFICA BINARIA DELL'INFORMAZIONE

In un calcolatore **tutta l'informazione** (numeri, testi, immagini, colori, ecc.) **è *codificata* in forma binaria**, facendo ricorso cioè al sistema di numerazione binario, basato sui simboli **0** **e** **1**.

D'altra parte, come si vedrà successivamente, anche le operazioni che devono essere eseguite sull'informazione sono codificate in forma binaria.

10111001000100011111001100011000101

... guardando all'interno della memoria ...

A seconda del tipo di dispositivo, i valori 0 e 1 sono ottenuti:

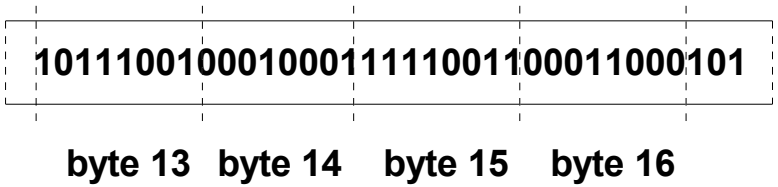
- ✓ con un differente livello di tensione elettrica
- ✓ con un differente valore del campo magnetico indotto
- ✓ con un differente stato di polarizzazione magnetica
- ✓ con l'alternanza fra buio e luce
- ✓ con la presenza o l'assenza di un raggio riflesso
- ✓ .....

L'UNITÀ D'INFORMAZIONE

il **BIT** (BInary digiT), che corrisponde ad uno dei 2 possibili stati di un dispositivo fisico e che viene interpretato come 0 oppure 1.

Ogni informazione complessa è trasformata, all'interno del calcolatore, in una sequenza opportuna di bit.

Nella memoria di un calcolatore si fa riferimento non al singolo bit, ma ad un "ottetto" di bit, il **BYTE**.



... i byte della memoria ...

Multipli del byte:

Potenze in base 2 con esponente multiplo intero di 10

1 **Kilobyte** = 2^{10} byte = 1024 byte = ~10^3 byte = **1KB**

1 **Megabyte** = 2^{20} byte = 1.048.576 byte = ~10^6 byte = **1MB**

1 **Gigabyte** = 2^{30} byte = 1.073.741.824 byte ~10^9 byte = **1GB**

1 **Terabyte** = 2^{40} byte = 1.099.511.627.776 byte ~10^{12} byte = **1TB**

LA CODIFICA BINARIA DEI NUMERI NATURALI

... un numero ad ***n*** cifre nel sistema di numerazione in base ***2***

$$a_{n-1}\bullet 2^{n-1} + a_{n-2}\bullet 2^{n-2} + \ldots + a_2\bullet 2^2 + a_1\bullet 2^1 + a_0\bullet 2^0 = \sum_{i=0,n-1} a_i\bullet 2^i$$ *ove* $\mathbf{a_i}$ *è uno dei simboli* {***0, 1***}

esempio

$$1000101 = 1\bullet 2^6 + 0\bullet 2^5 + 0\bullet 2^4 + 0\bullet 2^3 + 1\bullet 2^2 + 0\bullet 2^1 + 1\bullet 2^0 = 69$$

LA CODIFICA OTTALE DEI NUMERI NATURALI

... un numero ad ***n*** cifre nel sistema di numerazione in base ***8***

$$a_{n-1}\bullet 8^{n-1} + a_{n-2}\bullet 8^{n-2} + \ldots + a_2\bullet 8^2 + a_1\bullet 8^1 + a_0\bullet 8^0 = \sum_{i=0,n-1} a_i\bullet 8^i$$ *ove* $\mathbf{a_i}$ *è uno dei simboli* {***0, 1, 2, 3, 4, 5, 6, 7***}

LA CODIFICA ESADECIMALE DEI NUMERI NATURALI

... un numero ad ***n*** cifre nel sistema di numerazione in base ***16***

$$a_{n-1}\bullet 16^{n-1} + a_{n-2}\bullet 16^{n-2} + \ldots + a_2\bullet 16^2 + a_1\bullet 16^1 + a_0\bullet 16^0 = \sum_{i=0,n-1} a_i\bullet 16^i$$

ove $\mathbf{a_i}$ *è uno dei simboli* {***0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F***}

RAPPRESENTAZIONE DEI NUMERI NELLE DIVERSE BASI

Decimale	Binario	Ottale	Esadecimale
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

LE CONVERSIONI DI BASE DEI NUMERI NATURALI

DA DECIMALE A BINARIO: ***metodo delle divisioni successive***

esempi

DA BINARIO A DECIMALE:

esempi

DA BINARIO A OTTALE: ***metodo del raggruppamento dei bit a gruppi di 3***

esempi

DA BINARIO A ESADECIMALE: ***metodo del raggruppamento dei bit a gruppi di 4***

esempi

I NUMERI INTERI RELATIVI

La rappresentazione in modulo e segno

Se si dispone di ***n*** cifre per rappresentare un numero intero relativo **X** nella base **b**, la prima cifra viene usata per indicare il segno.

Con ***n*** cifre si possono rappresentare i numeri

$$-(b^n/2 - 1) \leq X \leq +(b^n/2 - 1)$$

Lo "0" positivo e lo "0" negativo.

RAPPRESENTAZIONE BINARIA IN MODULO E SEGNO (*b* = 2).

Primo bit = 0 numero positivo
Primo bit = 1 numero negativo

Con ***n*** bit si rappresentano i numeri

$$-(2^{n-1} - 1) \leq X \leq +(2^{n-1} - 1)$$

I NUMERI INTERI RELATIVI

La rappresentazione in complemento alla base

Se $\mathbf{a_{n-1}\ a_{n-2}\ ....\ a_2\ a_1\ a_0}$ è la rappresentazione in base ***b*** con ***n*** cifre di un numero ***X*** intero non negativo, si dice ***complemento alla base*** di X il numero

$$\mathbf{x = b^n - X}$$

che si ottiene facendo il complemento a *b-1* delle singole cifre a_{n-1}, a_{n-2}, ... a_2, a_1 e poi il complemento a *b* di a_0.

Il caso dello "0", che ha come complemento alla base se stesso.

L'operazione di complementazione alla base:

- ☞ consente di rappresentare i numeri

$$\mathbf{- b^n/2 \le X \le + (b^n/2 - 1)}$$

- ☞ suddivide le b^n configurazioni delle *n* cifre in due parti:
  - ✓ *le configurazioni che rappresentano numeri negativi*

$$\mathbf{- b^n/2 \le X < 0}$$

  - ✓ *le configurazioni che rappresentano numeri non negativi*

$$\mathbf{0 \le X \le + (b^n/2 - 1)}$$

LA CODIFICA BINARIA DEGLI INTERI RELATIVI

RAPPRESENTAZIONE BINARIA IN COMPLEMENTO A 2.

Se n sono i bit a disposizione ...

... la rappresentazione in complemento a 2 (che indicheremo con x) di un numero X non negativo coincide con X.

$$x = X \qquad \forall\ X \geq 0$$

esempi

... la rappresentazione in complemento a 2 di un numero X negativo sarà:

$$x = 2^n - X \qquad \forall\ X < 0$$

esempi

Primo bit = 0 → numero positivo

Primo bit = 1 → numero negativo

LE OPERAZIONI TRA NUMERI INTERI RELATIVI

Addizione $X + Y$

Nel caso $X, Y \geq 0$

$$x + y = X + Y$$

la condizione di overflow (trabocco) per $X + Y \geq 2^{n-1}$

Nel caso $X \geq 0;\ Y < 0$

$$x + y = X + 2^n - Y = 2^n + (X - Y) =$$

$$= \begin{cases} X - Y & \text{se } X - Y > 0 \text{ e si trascura il trabocco} \\ 2^n - (-(X - Y)) & \text{se } X - Y < 0 \end{cases}$$

Nel caso $X < 0;\ Y \geq 0$

analogamente

Nel caso $X < 0;\ Y < 0$

$$x + y = 2^n - X + 2^n - Y = 2^n + 2^n - (X + Y)$$

la condizione di underflow (trabocco) per $-(X + Y) > 2^{n-1}$

LE OPERAZIONI TRA NUMERI INTERI RELATIVI

Sottrazione X - Y

X - Y = X + (- Y)

⬇

vedi *Addizione*

Moltiplicazione X * Y

Addizione di numeri spostati di un bit

Divisione X / Y

L'algoritmo ordinario
si basa sul confronto di bit e sulla ***Sottrazione***

LA CODIFICA BINARIA DEI NUMERI FRAZIONARI

$$a_{-1}\bullet 2^{-1} + a_{-2}\bullet 2^{-2} + ... + a_{-(n-2)}\bullet 2^{-(n-2)} + a_{-(n-1)}\bullet 2^{-(n-1)} + a_{-n}\bullet 2^{-n} = \sum_{i=-1,-n} a_i \bullet 2^i$$

La conversione di numeri frazionari

Da decimale a binario:

metodo delle moltiplicazioni successive

esempi

I NUMERI REALI (o in virgola mobile)

La **rappresentazione normalizzata**

$$X = m * b^c$$

b	→	*base del sistema di numerazione*
$1/b \leq m < 1$	→	*mantissa*
c	→	*caratteristica*

La rappresentazione con 32 bit

S	caratteristica	mantissa

S = segno del numero 1 bit
caratteristica 7 bit in complemento a 2
mantissa 24 bit

esempi
Usando la numerazione decimale con base 10 il numero 0.587 è interpretato nel modo seguente:
$0.587=5 *10^{-1}+ 8 *10^{-2}+ 7 *10^{-3}$
La stessa formula consente di convertire un numero da una base ad una diversa: ad esempio il numero 0.1011 in base 2 può essere espresso in base 10 applicando la formula precedente
$(0.1011)_2 = (1 * 2^{-1} + 1 * 2^{-3} + 1 * 2^{-4})_{10} = (0.6875)_{10}$

OSSERVAZIONI

I numeri reali rappresentati in un calcolatore sono in realtà un sottoinsieme dei numeri reali.

Precisamente sono numeri razionali, con parte intera e parte frazionaria che approssimano i numeri reali con una precisione arbitraria.

L'insieme è distribuito in maniera non uniforme: vi sono valori estremamente vicini fra loro nell'intorno dello zero ed estremamente lontani nell'intorno del massimo numero esprimibile.

La rappresentazione, infatti, fa uso di un numero finito di bit per rappresentare sia la mantissa che l'esponente, questo comporta delle approssimazioni che si propagano nelle operazioni causando errori numerici anche rilevanti.

Il calcolo numerico è la disciplina che studia e valuta l'entità degli errori numeri introdotti durante l'esecuzione delle operazioni.

LA CODIFICA BINARIA DEI CARATTERI

Il codice ASCII (*American Standard Code for Information Interchange*)

... ogni carattere è rappresentato tramite 7 (oppure 8) bit, cioè un byte

... consente di codificare 128 (oppure 256) caratteri

... i caratteri sono distinti in *caratteri di comando*, *caratteri alfanumerici* e *segni*

... i ***caratteri di comando*** servono come codici di trasmissione o di controllo della stampa

Line Feed (LF) 00001010 Escape (Esc) 00011011

... i ***caratteri alfanumerici*** servono per rappresentare le lettere dell'alfabeto (maiuscole e minuscole) e le cifre numeriche

A	01000001	B	01000010	C	01000011
a	01100001	b	01100010	c	01100011

... i ***segni*** servono per rappresentare i simboli di punteggiatura e gli operatori aritmetico-logici

;	00111011	"	00100010	[	01011011
*	00101010	/	00101111	>	00111110

... in alcuni casi l'ottavo bit viene usato come ***bit di parità***

Altri codici ... ***BCD, EBCDIC, FIELDDATA***

Unicode È un codice a 16 bit: 65536 simboli

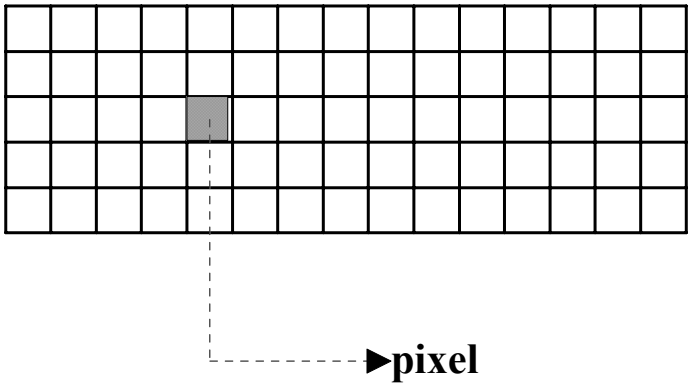
Rappresenta ogni simbolo utilizzando 4 cifre esadecimali

Contiene il codice ASCII (i primi 256 simboli)

Rappresenta i caratteri di tutte le lingue

LA CODIFICA BINARIA DELLE IMMAGINI

L'immagine è suddivisa in areole o punti o picture element (***pixel***).



Di ogni pixel viene rappresentato il colore o livello di grigio, espresso da un codice numerico intero.

L'immagine diviene quindi una lunga sequenza di bit (***digitalizzazione***).

Per ricostruire l'immagine a partire dalla sequenza di bit bisogna conoscere la base e l'altezza (espressa in pixel) dell'immagine stessa.

La ***risoluzione*** di un'immagine è determinata dal numero di punti per pollice (dot per inch o *dpi*) e dal numero di colori o livelli di grigio (fissato dal numero di bit con cui si codifica tale caratteristica). In genere vengono usati 8 bit (un byte) e quindi 256 colori o livelli di grigio.

Le proprietà e le operazioni

Considerato un insieme ***S*** di elementi:

- fra i quali sia definita una *relazione di uguaglianza* che goda delle proprietà *riflessiva, simmetrica e transitiva*
- per i quali sia definita una legge di composizione, detta ***somma logica*** (+), tale che Z = X + Y con X,Y, Z ∈ ***S***
- per i quali sia definita una legge di composizione, detta ***prodotto logico*** (•), tale che Z = X • Y con X,Y, Z ∈ ***S***
- che contenga un *elemento neutro rispetto alla somma* (o zero), indicato dal simbolo ***0***, tale che X + ***0*** = X
- che contenga un *elemento neutro rispetto al prodotto* (o unità), indicato dal simbolo ***1***, tale che X • ***1*** = X
- per il quale valgono le *proprietà commutative delle operazioni* (+ e •), tali che X + Y = Y + X e X • Y = Y • X
- per il quale valgono le *proprietà distributive* della operazione + rispetto alla operazione • e della operazione • rispetto alla operazione +, cioè

  $X + (Y \bullet Z) = (X + Y) \bullet (X + Z)$ $\qquad$ $X \bullet (Y + Z) = (X \bullet Y) + (X \bullet Z)$

- tale che $X \in \boldsymbol{S} \Rightarrow \exists\, (\underline{X} \in \boldsymbol{S})\ \ni'\ (X + \underline{X} = \boldsymbol{1})$ e $(X \bullet \underline{X} = \boldsymbol{0})$ $\underline{X}$ è detto ***complemento*** di X
- nel quale esiste almeno una coppia di elementi X,Y ∈ ***S*** ∋′ X≠Y

Le proprietà suddette definiscono su ***S*** una ***struttura algebrica*** comunemente detta ***Algebra di Boole***.

I teoremi fondamentali

$X + X = X$ $\qquad$ $X + 1 = 1$

$X \bullet X = X$ $\qquad$ $X \bullet 1 = X$

$\overline{(X + Y)} = \overline{X} \bullet \overline{Y}$

$\overline{X \bullet Y} = \overline{X} + \overline{Y}$

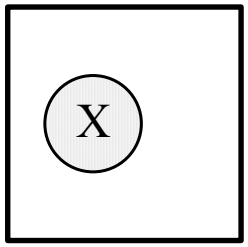
Teoremi di De Morgan

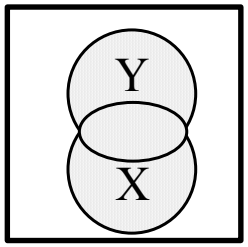
$X + (Y + Z) = (X+Y) + Z$ Proprietà associativa della somma logica

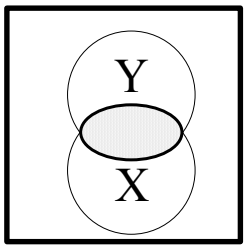
Gli esempi

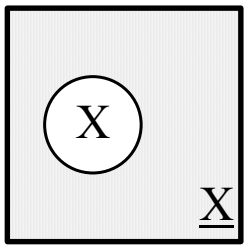
... di insiemi che godono delle proprietà dell'Algebra di Boole

I diagrammi di Venn





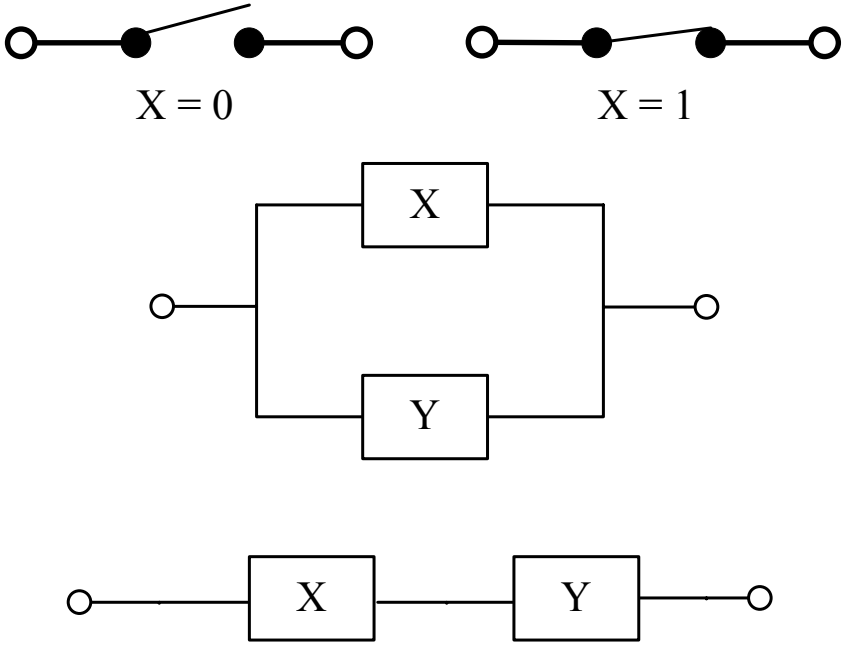




Gli esempi

... di insiemi che godono delle proprietà dell'Algebra di Boole

Le reti di interruttori



Gli esempi

... di insiemi che godono delle proprietà dell'Algebra di Boole

Le variabili logiche a due valori

Sia f una funzione di variabili x_1, x_2, ... , x_n, ciascuna delle quali possa assumere solo due valori (0 e 1, falso e vero, aperto e chiuso, ecc.).

Si dice che f è una funzione di variabili booleane, che può assumere, a sua volta, solo i due valori anzidetti.

Gli operatori logici e le tavole di verità

x_1	x_2	$f = x_1 + x_2$ $f = x_1 \cup x_2$ $f = x_1$ .or. x_2
0	0	0
0	1	1
1	0	1
1	1	1

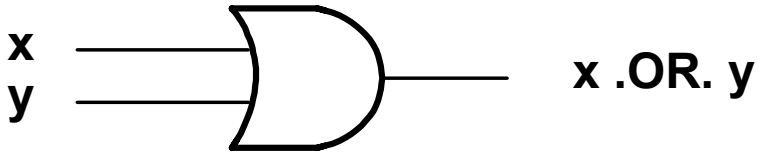
x_1	x_2	$f = x_1 \bullet x_2$ $f = x_1 \cap x_2$ $f = x_1$ .and. x_2
0	0	0
0	1	0
1	0	0
1	1	1

x	$f = \underline{x}$ $f = !\, x$ $f =$ .not. x
0	1
1	0

Gli operatori logici e le porte logiche

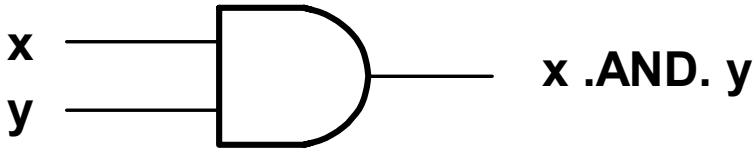
OR

$(x \cup y)$



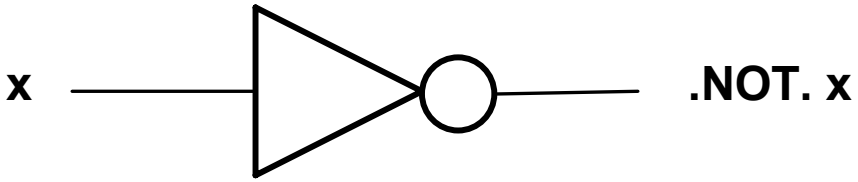
AND

$(x \cap y)$



NOT

$(\underline{x})$



La composizione opportuna di porte logiche consente di realizzare circuiti elettronici che realizzano varie funzioni, in particolare i ***circuiti sommatori***.