



La codifica binaria

Prof. Alberto Borghese
Dipartimento di Scienze dell'Informazione
borgnese@dsi.unimi.it

Università degli Studi di Milano



Sommario

Sistema di numerazione binario

Conversione in e da un numero binario

Operazioni elementari su numeri binari (somma, sottrazione e moltiplicazione intera).

I numeri decimali

Codifica IEEE754 dei numeri reali anche in forma denormalizzata.

Rappresentazione binaria dell'Informazione



Sistemi di numerazione



Sistema di numerazione mediante simboli (numerazione romana: I, V, X, L, C, M) il cui valore non dipende dalla posizione: e.g. XXXI = 31, XI = 11....

Sistema di numerazione posizionale (decimale): **cifra + peso**.
Il peso è la base elevata alla posizione della cifra.

1 ha un valore diverso nelle due scritture:

100

1000



Numerazione Binaria



Alfabeto della numerazione:

{0, 1, 2, 3, 4, 5, 6, 7, 8, 9} numerazione araba decimale.

{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F} numerazione esadecimale.

{0, 1} numerazione binaria.

Sistemi di numerazione binario, ottale ed esadecimale.

Conversioni decimale -> binario e viceversa.



Codifica posizionale di un numero



Fondata sul concetto di **base**: $B = [b_0, b_1, b_2, b_3, \dots]$.

Ciasun elemento, E , può essere rappresentato come combinazione lineare degli elementi della base: $E = \sum_k c_k b_k$

Esempi:

- $12,21_{10} = 1 \times 10^1 + 2 \times 10^0 + 2 \times 10^{-1} + 1 \times 10^{-2} = 12,21$
- $100,11_2 = 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = 4,75$



Osservazioni sulla numerazione binaria



Il linguaggio di un elaboratore elettronico è fatto di due segnali: **on** e **off**, rappresentati dai simboli **1** e **0** (**alfabeto binario**).

- Sia le istruzioni che i dati sono rappresentati da *parole* di numeri binari.
- Un alfabeto binario non limita le funzionalità di un elaboratore a patto di avere parole di lunghezza sufficiente.
- 1000110010100000 rappresenta un'istruzione di addizione in MIPS.



Tassonomia ed unità di misura



Hertz - numero di ciclo al secondo nei moti periodici (clock).

- MIPS - Milioni di istruzioni per secondo.
- MFLOPS - Milioni di istruzioni in virgola mobile (FLOating point) al secondo.

Prefissi:

k - chilo (mille: 10^3).

M - mega (un milione: 10^6).

G - giga (un miliardo: 10^9).

T - tera (1000 miliardi: 10^{12})

m - milli (un millesimo: 10^{-3})

μ - micro (un milionesimo: 10^{-6})

n - nano (un miliardesimo: 10^{-9})

p - pico (un millesimo di miliardo: 10^{-12})



Terminologia



Bit = binary digit.

- 1 byte = 8 bit.
- 1kbyte = 2^{10} byte = 1,024 byte
- 1Mbyte = 2^{20} byte = 1,048,576 byte.
- 1Gbyte = 2^{30} byte = 1,073,741,824 byte.
- 1Tbyte = 2^{40} byte = 1,099,511,627,776 byte.

- Parola (word) numero di bit trattati come un unicum dall'elaboratore.
- Le parole oggi arrivano facilmente a 64bit (Itanium).



Sommario



Sistema di numerazione binario

Conversione in e da un numero binario

Operazioni elementari su numeri binari (somma, sottrazione e moltiplicazione intera).

I numeri decimali

Codifica IEEE754 dei numeri reali anche in forma denormalizzata.

Rappresentazione binaria dell'Informazione



Conversione da base n a base 10



Un numero $x=[c_0, c_1, c_2, c_3, \dots]$ in base n , $[b_0, b_1, b_2, b_3, \dots]$ si trasforma in base 10, E , facendo riferimento alla formula:

$$E = \sum_k c_k b_k$$

- ciascuna cifra k -esima viene moltiplicata per la base corrispondente: $b_k = n^k$.
- i valori così ottenuti sono sommati per ottenere il numero in notazione decimale.

$$\begin{aligned} \mathbf{101\ 1101\ 0100}_{\text{due}} &= 1x2^{10} + 0x2^9 + 1x2^8 + 1x2^7 + 1x2^6 + 0x2^5 + \\ &1x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 0x2^0 = \\ &1024 + 256 + 128 + 64 + 16 + 4 = 1492 \end{aligned}$$



“Spelling” di un numero



Vogliamo rappresentare 1492_{dieci}

Unità $1492 = 10 \times 149 + 2$ ← Cifra meno significativa

Decine **(10x)** $149 = 10 \times 14 + 9$

Centinaia **(100x)** $14 = 10 \times 1 + 4$

Migliaia **(1000x)** $1 = 10 \times 0 + 1$ ← Cifra più significativa

$$1492 = 2 + 9 \times 10 + 4 \times 100 + 1 \times 1000$$



Conversione base 10 -> base 2



Vogliamo rappresentare 1492_{dieci} in binario: 10111010100_{due}

$1492 = 2 \times 746 + 0$ ← Bit meno significativo

$746 = 2 \times 373 + 0$

$373 = 2 \times 186 + 1$

$186 = 2 \times 93 + 0$

$93 = 2 \times 46 + 1$

$46 = 2 \times 23 + 0$

$23 = 2 \times 11 + 1$

$11 = 2 \times 5 + 1$

$5 = 2 \times 2 + 1$

$2 = 2 \times 1 + 0$

$1 = 2 \times 0 + 1$ ← Bit più significativo



Perché funziona?



Prendiamo il numero E_0 e dividiamo per 2.

Se E_0 è pari il resto, R_0 , sarà 0, altrimenti sarà 1. Infatti:
 $\text{int}(E_0 / 2) * 2 + R_0 = E_0$.

Esempio:

$$E_0 = 5_{10} = ?_2$$

$$R_0 = \text{resto}(5/2) = 1$$

Chiamiamo $E_1 = \text{int}(E_0 / 2)$ e procediamo.

$$E_1 = \text{quoz}(5/2) = 2$$

Prendiamo E_1 e dividiamo per 2. Chiamiamo $E_2 = \text{int}(E_1 / 2)$. Se E_1 è pari il resto, R_1 , sarà 0, altrimenti sarà 1. $E_1 = \text{int}(E_1 / 2) * 2 + R_1$

$$R_1 = \text{resto}(2/2) = 0$$

Chiamiamo $E_2 = \text{int}(E_1 / 2)$ e procediamo.

$$E_2 = \text{quoz}(2/2) = 1$$

Prendiamo E_2 e dividiamo per 2. Se E_2 è pari il resto, R_2 , sarà 0, altrimenti sarà 1. $E_2 = \text{int}(E_2 / 2) * 2 + R_2$.

$$R_2 = \text{resto}(1/2) = 0$$

Si procede fino a quando E_i non è < 2 ($=1$).

$$E_0 = R_0 + R_1 * 2 + R_2 * 2^2 \\ = 1 0 1$$

$$E_0 = \text{int}[E_0 / 2] * 2 + R_0.$$

$$E_0 = \text{int}[E_1] * 2 + R_0.$$

$$E_0 = \text{int}[\text{int}(E_1 / 2) * 2 + R_1] * 2 + R_0.$$

$$E_0 = \text{int}[\text{int}(E_2) * 2 + R_1] * 2 + R_0.$$

$$E_0 = \text{int}[\text{int}[\text{int}(E_2 / 2) * 2 + R_2] * 2 + R_1] * 2 + R_0. \quad \text{int}(E_2 / 2) = 0$$

<http://homes.dsi.unimi.it/~borgnese>



Conversione base 10 -> base n: algoritmo



Un numero x in base 10 si trasforma in base n usando il seguente procedimento:

- Dividere il numero x per n
- Il resto della divisione è la cifra di posto 0 in base n
- Il quoziente della divisione è a sua volta diviso per n
- Il resto ottenuto a questo passo è la cifra di posto 1 in base n
- Si prosegue con le divisioni dei quozienti ottenuti al passo precedente fino a che l'ultimo quoziente è 0.
- l'ultimo resto è la cifra più significativa in base n



Esercizi



Dati i numeri decimali 23456, 89765, 67489, 121331, 2453, 111010101

- si trasformino in base 3
- si trasformino in base 7
- si trasformino in base 2

- Dati i numeri 23456_7 , 121331_5 , 2453_8 , 111010101_2
- convertire ciascuno in decimale e in binario



Codifica esadecimale



Il codice esadecimale viene utilizzato come forma compatta per rappresentare numeri binari:

- 16 simboli: 0,1,...,9,A,B,...,F

- Diverse notazioni equivalenti:
0x9F
 $9F_{16}$
9Fhex

$$0x9F = 9 \times 16^1 + 15 \times 16^0 = 159_{10}$$



Conversione esadecimale -> binario



Vogliamo rappresentare 9Fhex in binario. E' semplice.

- Ogni simbolo viene convertito in un numero binario di 4 cifre:

9hex --> 1001_{due}

Fhex --> 1111_{due}

9Fhex --> 10011111_{due}

- È sufficiente ricordarsi come si rappresentano in binario i numeri decimali da 0 a 15 (o derivarli)



Conversione binario -> esadecimale



Da binario ad esadecimale si procede in modo analogo:

- Ogni gruppo di 4 cifre viene tradotto nel simbolo corrispondente:

Esempio: convertire 1101011_{due} in esadecimale:

1011_{due} --> B_{hex}

110_{due} --> 6_{hex}

1101011_{due} --> 6B_{hex}

Viene aggiunto un "leading" 0



Sommario



Sistema di numerazione binario

Conversione in e da un numero binario

Operazioni elementari su numeri binari (somma, sottrazione e moltiplicazione intera).

I numeri decimali

Codifica IEEE754 dei numeri reali anche in forma denormalizzata.

Rappresentazione binaria dell'Informazione



Somma



$$\begin{array}{r} 111 \\ 1011 + \\ 110 = \\ \hline 10001 \end{array}$$

← Riporto



Numeri negativi



I numeri negativi sono complementari ai numeri positivi: $a + (-a) = 0$

Codifica in complemento a 1: il numero negativo si ottiene cambiando 0 con 1 e viceversa.

Problema:

00	0
01	1
10	-1
11	0

 Doppia codifica per lo 0.

MATHEMATICS AND COMPUTERS IN SIMULATION

Review form

Prof. Tullio Ciurlo, Dipartimento di Matematica, Università di Palermo, Italy
the La. Rubens nature reserve of Palermo

Authors: Antonio Caselles, Pantaleon David Romero, Jean-Marie Jacques and Eduardo Peris

(Detailed comments in reply to these questions should be given on page 2.)

1. Is the paper interesting? Yes No

2. Is the paper technically sound? No

3. Does it make adequate references to the literature? No

4. How about length? too **about right**

5. On the scale below, please check your evaluation of this paper.

1 2 3 4 5 6 7 8 9 10

(1: excellent → 10: very bad)

Your recommendation:

Publish as it is, or with minor revisions (suggested on page 2)

Publish only after major revisions (suggested on page 2)

Reviewer's Comments to the Editor

This report is for the internal report but not a research paper. I hardly see any suggestion to the authors to carry out research in depth in the area of OCS and its applications.

Reviewer: Dr Dayou Li

Reviewer's Signature:

Date: 23/03/2005

Codifica in complemento a 2: il numero negativo si ottiene cambiando 0 con 1 e viceversa, e sommando 1.

00	0
01	1
10	-2
11	-1

10 + 1 = 11

NB La prima cifra è il bit di segno.



Sottrazione



Sommo i seguenti 2 numeri $11 + (-13)$:

$$01011_2 = 11_{10}$$

$$10011_{10} = -13_{10}$$

E' equivalente ad effettuare la differenza: $11 - 13$.

$$\begin{array}{r}
 00110 \\
 01011 + \\
 10011 = \\
 \text{-----} \\
 11110 \rightarrow -2_{10}
 \end{array}$$



Estensione del segno



Somma i seguenti 2 numeri $34 + (-13)$, su 8 bit:

$$\begin{array}{lcl}
 100010 & \Rightarrow & 00100010_2 = 32_{10} \\
 10011_2 & \Rightarrow & 11110011_2 = -13_{10}
 \end{array}$$

NB: l'estensione verso sinistra di un numero copia il bit piú significativo.

E' equivalente ad effettuare la differenza: $34 - 13$.

$$\begin{array}{r}
 \text{Fuori dalla} \\
 \text{Capacità} \\
 \text{di 8 bit} \nearrow \\
 (1)11000100 \\
 00100010 + \\
 11110011 = \\
 \text{-----} \\
 00010101 \rightarrow 21_{10}
 \end{array}$$



Moltiplicazione binaria



$$\begin{array}{rcl}
 \text{Moltiplicatore} & \longrightarrow & 11011x \\
 \text{Moltiplicando} & \longrightarrow & 111 =
 \end{array}$$

$$\begin{array}{r}
 11011x \quad 27_{10} \\
 111 = \quad 7_{10} \\
 \text{-----} \\
 111111 \\
 11011+ \\
 11011- \\
 11011- - \\
 \text{-----} \\
 10111101 \quad 189_{10}
 \end{array}$$

$$\begin{array}{r}
 \text{-----} \\
 11111 \\
 11011+ \\
 11011- \\
 \text{-----} \\
 1 \\
 1010001+ \\
 11011- - \\
 \text{-----} \\
 \text{prodotto} \longrightarrow 10111101
 \end{array}$$

prodotti parziali



Moltiplicazione mediante shifting



Lo shift di un numero a dx, di k cifre, corrisponde ad una divisione per la base elevata alla k-esima potenza.

Lo shift di un numero a sx, di k cifre, corrisponde ad una moltiplicazione per la base elevata alla k-esima potenza.

Esempio:

$$213_{10} / 10 = 21.3_{10}$$

$$\begin{aligned} 213_{10} &= (2 \times 10^2 + 1 \times 10^1 + 3 \times 10^0) / 10^1 = \\ &= (2 \times 10^2 + 1 \times 10^1 + 3 \times 10^0) \times 10^{-1} = \\ &= (2 \times 10^2 \times 10^{-1} + 1 \times 10^1 \times 10^{-1} + 3 \times 10^0 \times 10^{-1}) = \\ &= (2 \times 10^1 + 1 \times 10^0 + 3 \times 10^{-1}) = 21.3 \text{ cvd.} \end{aligned}$$

Esempio:

$$\begin{aligned} 23 / 4 = 5,75 \Rightarrow 10111 / 100 = \\ (1x2^4 + 0x2^3 + 1x2^2 + 1x2^1 + 1x2^0) \times 2^{-2} = \\ (1x2^2 + 0x2^1 + 1x2^0 + 1x2^{-1} + 1x2^{-2}) = 5,25 \text{ cvd.} \end{aligned}$$



Sommario



Sistema di numerazione binario

Conversione in e da un numero binario

Operazioni elementari su numeri binari (somma, sottrazione e moltiplicazione intera).

I numeri decimali

Codifica IEEE754 dei numeri reali anche in forma denormalizzata.

Rappresentazione binaria dell'Informazione



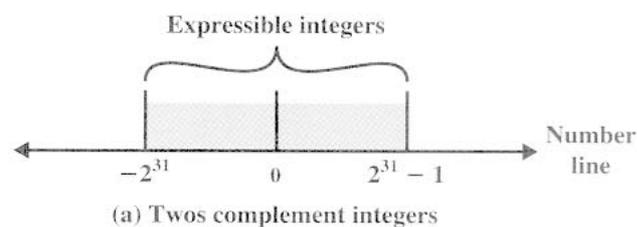
Capacità di rappresentazione: Numeri Interi



Interi con segno su N bit. Range: $-2^{N-1} \leq c \leq 2^{N-1} - 1$.

Esempio: Visual C++. Intero è su 4byte (word di 32 bit):

$$-2^{31} = -2.147.483.650 \leq c \leq 2.147.483.649 = 2^{31} - 1$$



Numeri decimali (float)



Numeri reali per il computer non sono i numeri reali per la matematica!!
E' meglio chiamarli float (numeri decimali), sono in numero finito.

Dato un certo numero di bit (stringa) per codificare il numero float, esistono due tipi di codifiche possibili:

Rappresentazione in fixed point.

La virgola è in posizione fissa all'interno della stringa.

Supponiamo di avere una stringa di 8 cifre, con virgola in 3a posizione:

$$27,35 = + | 27,35000$$

$$-18,7 = - | 18,70000$$

$$0,001456 = + | 00,00145(6)$$



Numeri decimali: rappresentazione floating point



Rappresentazione come mantissa + esponente.

Esempio di **rappresentazioni equivalenti**:

$$127,35 = 12.735 \times 10^1 = 1,2735 \times 10^2 = \mathbf{0,12735 \times 10^3} = 10^3 \times (1 \times 10^{-1} + 2 \times 10^{-2} + 7 \times 10^{-3} + 3 \times 10^{-4} + 5 \times 10^{-5}) = 10^3 \times 0,12735$$

In grassetto viene evidenziata la rappresentazione normalizzata.

Vengono rappresentati numeri molto grandi e molto piccoli.

Supponiamo di avere una stringa di 8 cifre. 5 per la mantissa, 3 per l'esponente.

$$0,001456 = + | 1456 | - | 02.$$



Conversione base 10 -> base n: algoritmo



Un numero $x.y$ in base 10 si trasforma in base n usando il seguente procedimento.

Per la parte intera, x :

- Dividere il numero x per n
- Il resto della divisione è la cifra di posto 0 in base n
- Il quoziente della divisione è a sua volta diviso per n
- Il resto ottenuto a questo passo è la cifra di posto 1 in base n

- Si prosegue con le divisioni dei quozienti ottenuti al passo precedente fino a che l'ultimo quoziente è 0.

- l'ultimo resto è la cifra più significativa in base n



Conversione base 10 -> base n: algoritmo per la parte frazionaria



Un numero x,y in base 10 si trasforma in base n usando il seguente procedimento.

Per la parte frazionaria, y :

- Moltiplicare il numero y per n
- La prima cifra del risultato coincide con la cifra di posto 1 dopo la virgola.
- Si elimina la parte intera ottenuta e si considera la nuova parte frazionaria.
- La parte frazionaria ottenuta viene moltiplicata per la base n .
- La prima cifra del risultato coincide con la cifra di posto 2 dopo la virgola.
- Si prosegue con le moltiplicazioni della parte frazionaria fino a quando non diventa 0 o non si esaurisce la capacità di

representazione.

A./

borgnese



Errori di approssimazione



Esempio: $10,75_{10} = 1010,11_2$

Esempio: $10,76_{10} = 1010,1100001..._2$

$$10 : 2 \Rightarrow 0$$

$$5 : 2 \Rightarrow 1$$

$$2 : 2 \Rightarrow 0$$

$$1 : 2 \Rightarrow 1$$

1010,

$$0,75 - 1 \times 2^{-1} \Rightarrow 1$$

$$0,25 - 1 \times 2^{-2} \Rightarrow 1$$

11

**Errori di approssimazione:
arrotondamento e troncamento.**

$$0,76 - 1 \times 2^{-1} \Rightarrow 1$$

$$0,26 - 1 \times 2^{-2} \Rightarrow 1$$

$$0,01 - 1 \times 2^{-3} \Rightarrow 0$$

$$0,01 - 1 \times 2^{-4} \Rightarrow 0$$

$$0,01 - 1 \times 2^{-5} \Rightarrow 0$$

$$0,01 - 1 \times 2^{-6} \Rightarrow 0$$

$$0,01 - 1 \times 2^{-7} \Rightarrow 1$$

$$0,0021875$$

$$(2^{-7} = 0,0078125)$$



Sommario



Sistema di numerazione binario

Conversione in e da un numero binario

Operazioni elementari su numeri binari (somma, sottrazione e moltiplicazione intera).

I numeri decimali

Codifica IEEE754 dei numeri reali anche in forma denormalizzata.

Rappresentazione binaria dell'Informazione



Standard IEEE 754 (1980)

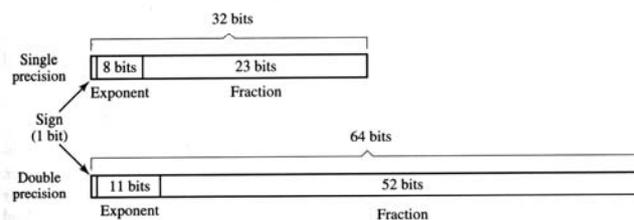


Figure 2-10 Single-precision and double-precision IEEE 754 floating point formats.

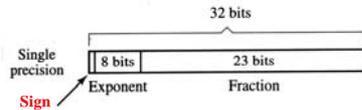
Rappresentazione polarizzata dell'esponente:

Polarizzazione pari a 127 per singola precisione =>
1 viene codificato come 1000 0000.

Polarizzazione pari a 1023 in doppia precisione.
1 viene codificato come 1000 0000 000.



Codifica mediante lo standard IEEE 754



Esempio: $N = -10,75_{10} = -1010,11_2$

- 1) Normalizzazione: $\pm 1,xxxxxx \times 2^e$ Esempio: $-1,01011 \times 2^3$
- 2) Codifica del segno 1 = - 0 = +
- 3) Calcolo dell'esponente, e , in rappresentazione polarizzata:
 $e = 3 + 127 = 130_{10} = 10000010_2$

$N = 1 \mid 1000\ 0010 \mid 0101\ 1000\ 0000\ 0000\ 0000\ 0000$



Configurazioni notevoli nello Standard IEEE 754

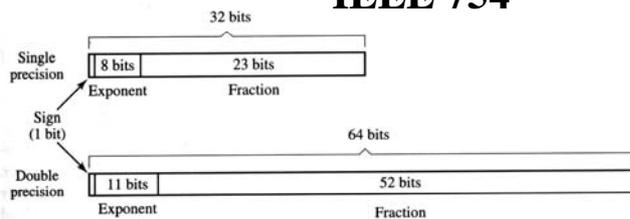


Figure 2-10 Single-precision and double-precision IEEE 754 floating point formats.

Configurazioni notevoli:

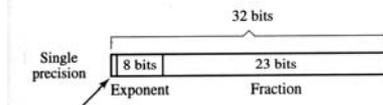
0	Mantissa: 0	Esponente: 00000000
$+\infty$	Mantissa: 0	Esponente: 11111111.
NaN	Mantissa: $\neq 0$.	Esponente: 11111111.

Range degli esponenti (8 bit): $1--254 \Rightarrow -126 \leq \text{exp} \leq +127$.

Numeri float: $1.0 \times 2^{-126} = 1.175494351 \times 10^{-38} \div 3.402823466 \times 10^{38} = 1.1\dots11 \times 2^{127}$



Capacità dello Standard IEEE 754



Range degli esponenti (8 bit): $1-254 \Rightarrow -126 \leq \text{exp} \leq +127$.

Minimo float (in valore assoluto!): 1.0×2^{-126}

Massimo float: $1.1111\ 1111\ 1111\ 1111\ 1111\ 1111 \times 2^{+127}$

Capacità float:

Minimo: $1.175494350822288 \times 10^{-38}$ (1.175494350822288e-038)

Massimo: $3.402823466385289 \times 10^{38}$ (3.402823466385289e+038)

Discontinuità tra Minimo_float e 0 il delta è $1.175494350822288 \times 10^{-38}$

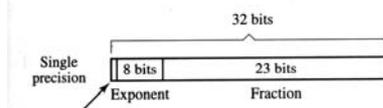
Si può fare di meglio?

Numero denormalizzato

Mantissa: $\neq 0$ Esponente: 00000000



Denormalizzazione nello Standard IEEE 754



Range degli esponenti (8 bit): $1-254 \Rightarrow -126 \leq \text{exp} \leq +127$.

Minimo float (in valore assoluto!): $1.0 \times 2^{-126} = 1.175494350822288 \times 10^{-38}$

Tuttavia abbiamo anche la mantissa a disposizione. Se troviamo una codifica per cui possiamo scrivere 0,0000 0000 0000 0000 0000 001, otteniamo un numero più piccolo (in valore

assoluto!) pari a : $2^{-(23-126)} = 1.401298464324817 \times 10^{-45}$.

Discontinuità tra Minimo_float e 0 diventa $2^{-149} = 1.401298464324817 \times 10^{-45}$

Configurazioni notevoli:

0

Mantissa: 0

Esponente: 00000000

$+\infty$

Mantissa: 0

Esponente: 11111111.

NaN

Mantissa: $\neq 0$.

Esponente: 11111111.

Numero denormalizzato

Mantissa: $\neq 0$

Esponente: 00000000



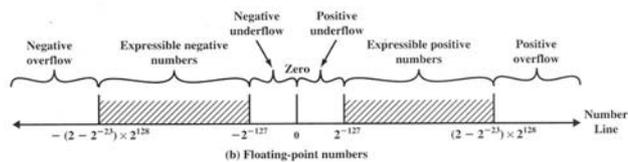
Risoluzione della codifica dei reali

Distanza tra due numeri vicini.



Fixed point: Risoluzione fissa, pari al peso del bit meno significativo.

Esempio su 8 bit: +1111,101 la risoluzione per tutti i numeri sarà: $1 \times 2^{-3} = 0,125$



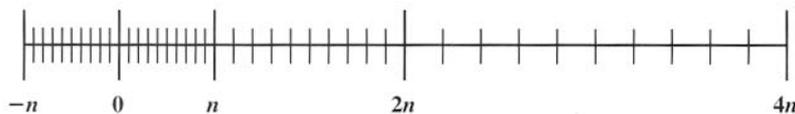
Floating point: Risoluzione *relativa* fissa, pari al peso del bit meno significativo.

Il bit meno significativo è in 23a posizione in singola precisione $\Rightarrow 2^{-23}$, ne consegue che la risoluzione sarà 2^{-23} volte il numero descritto.

Esempi:

$$100, \dots = 1,000 \times 2^2 \Rightarrow \text{La risoluzione sarà } 2^{-23} \times 2^2 = 2^{-21}$$

$$1.0 \times 2^{-126} \Rightarrow \text{La risoluzione sarà } 2^{-23} \times 2^{-126} = 2^{-149}$$



A.A. 2004

borgnese



Sommario



Sistema di numerazione binario

Conversione in e da un numero binario

Operazioni elementari su numeri binari (somma, sottrazione e moltiplicazione intera).

I numeri decimali

Codifica IEEE754 dei numeri reali anche in forma denormalizzata.

Rappresentazione binaria dell'Informazione

A.A. 2004-2005

40/46

<http://homes.dsi.unimi.it/~borgnese>



Rappresentazione dell'informazione



Non solo conteggio, ma anche enumerazione di oggetti....

Noi rappresentiamo gli oggetti tramite parole composte da un alfabeto di simboli: A,B,...,Z,0,1,...,9,...

- Diversi alfabeti possono essere usati per rappresentare gli stessi oggetti.
- I simboli degli alfabeti possono assumere diverse forme.
- Segni su carta, livelli di tensione, fori su carta, segnali di fumo.

.....



Esempio di codifica binaria



- Quanti oggetti diversi possiamo rappresentare con parole binarie di 3 bit?

0	000	A
1	001	B
2	010	C
3	011	D
4	100	E
5	101	F
6	110	G
7	111	H



Codifica dei caratteri alfanumerici



Quanti bit devono avere le parole binarie usate per identificare i 26 caratteri diversi dell'alfabeto inglese (es: A,B,...,Z)?

$$2^4 < 26 < 2^5$$

Quanti bit devono avere le parole binarie usate per identificare 26+26 oggetti diversi (es: A,B,...,Z, a,b,... z)?

$$2^5 < 52 < 2^6$$

Quanti per 100 oggetti? $\text{ceil}[\log_2 100]$



0		32		64	Q	96	`	128	Ç	160	á	192	L	224	œ
1	☐	33	!	65	A	97	a	129	ü	161	í	193	l	225	ß
2	☐	34	"	66	B	98	b	130	é	162	ó	194	T	226	Γ
3	♥	35	#	67	C	99	c	131	â	163	ú	195	†	227	∏
4	♦	36	\$	68	D	100	d	132	ä	164	ñ	196	—	228	Σ
5	♣	37	%	69	E	101	e	133	à	165	ñ	197	†	229	ϕ
6	♣	38	â	70	F	102	f	134	ã	166	ã	198	†	230	μ
7	+	39	'	71	G	103	g	135	ç	167	ç	199		231	τ
8	☐	40	(72	H	104	h	136	ê	168	ê	200		232	ø
9	◊	41)	73	I	105	i	137	ë	169	ë	201		233	θ
10	☐	42	*	74	J	106	j	138	è	170	ï	202		234	ϖ
11	♠	43	+	75	K	107	k	139	ÿ	171	ÿ	203		235	ð
12	♀	44	,	76	L	108	l	140	î	172	¼	204		236	œ
13	♠	45	-	77	M	109	m	141	ì	173	ï	205	=	237	ϕ
14	♠	46	.	78	N	110	n	142	ñ	174	«	206		238	€
15	♠	47	/	79	O	111	o	143	ñ	175	»	207	±	239	∏
16	▶	48	0	80	P	112	p	144	é	176		208		240	≡
17	◀	49	1	81	Q	113	q	145	»	177		209	T	241	±
18	♠	50	2	82	R	114	r	146	ff	178		210		242	λ
19	!!	51	3	83	S	115	s	147	ô	179		211		243	ς
20		52	4	84	T	116	t	148	ö	180	†	212	L	244	Γ
21	§	53	5	85	U	117	u	149	ò	181	†	213	Γ	245	J
22	-	54	6	86	U	118	u	150	û	182		214		246	÷
23	±	55	7	87	W	119	w	151	ù	183		215		247	≈
24	T	56	8	88	X	120	x	152	ÿ	184	†	216	†	248	°
25	↓	57	9	89	Y	121	y	153	ÿ	185		217	J	249	·
26	→	58	:	90	Z	122	z	154	ÿ	186		218	Γ	250	·
27	+	59	;	91	[123	{	155	ç	187		219		251	J
28	L	60	<	92	\	124		156	ç	188		220		252	
29	+	61	=	93]	125	}	157	ÿ	189		221		253	±
30	▲	62	>	94	^	126	~	158	ÿ	190		222		254	
31	▼	63	?	95	_	127	◊	159	f	191	†	223		255	



**Il codice
ASCII
la rappresentazione
dell'informazione
alfanumerica**

- 8 bit
- 0-31 codici di controllo.
- 128-255 extended ASCII



Codifica binaria



Quanti oggetti diversi possiamo rappresentare con parole binarie di k bit?

- Con una parola di 1 bit rappresentiamo 2 oggetti (1 bit ha due possibili valori).
- Supponiamo di avere parole di $k-1$ bit. Quanti oggetti riescono a rappresentare?

2^{k-1} oggetti.



Sommario



Sistema di numerazione binario

Conversione in e da un numero binario

Operazioni elementari su numeri binari (somma, sottrazione e moltiplicazione intera).

I numeri decimali

Codifica IEEE754 dei numeri reali anche in forma denormalizzata.

Rappresentazione binaria dell'Informazione