



# La codifica binaria

Prof. Alberto Borghese  
Dipartimento di Scienze dell'Informazione  
[borgnese@dsi.unimi.it](mailto:borgnese@dsi.unimi.it)

Università degli Studi di Milano



## Sommario

### Rappresentazione binaria dell'Informazione

Conversione in e da un numero binario

Operazioni elementari su numeri binari

I numeri reali

Codifica IEEE754 dei numeri reali



# Rappresentazione dell'informazione



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.

.....



0	32	64	96	128	160	192	224
1	33	65	97	129	161	193	225
2	34	66	98	130	162	194	226
3	35	67	99	131	163	195	227
4	36	68	100	132	164	196	228
5	37	69	101	133	165	197	229
6	38	70	102	134	166	198	230
7	39	71	103	135	167	199	231
8	40	72	104	136	168	200	232
9	41	73	105	137	169	201	233
10	42	74	106	138	170	202	234
11	43	75	107	139	171	203	235
12	44	76	108	140	172	204	236
13	45	77	109	141	173	205	237
14	46	78	110	142	174	206	238
15	47	79	111	143	175	207	239
16	48	80	112	144	176	208	240
17	49	81	113	145	177	209	241
18	50	82	114	146	178	210	242
19	51	83	115	147	179	211	243
20	52	84	116	148	180	212	244
21	53	85	117	149	181	213	245
22	54	86	118	150	182	214	246
23	55	87	119	151	183	215	247
24	56	88	120	152	184	216	248
25	57	89	121	153	185	217	249
26	58	90	122	154	186	218	250
27	59	91	123	155	187	219	251
28	60	92	124	156	188	220	252
29	61	93	125	157	189	221	253
30	62	94	126	158	190	222	254
31	63	95	127	159	191	223	255



## Il codice ASCII la rappresentazione dell'informazione alfanumerica

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



## Numerazione



Sistema di conteggio.

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.



## 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 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.

- Aggiungendo un bit a sinistra di ciascuna parola (0 o 1)?

Confrontate con la codifica decimale sullo stesso numero di cifre!



## Richieste per la codifica dei caratteri alfanumerici



Quanti bit devono avere le parole binarie usate per identificare 21 oggetti diversi (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?  $\text{ceil}[\log_2 100]$



## 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:  $10^9$ ).

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

$\mu$  - micro (un milionesimo:  $10^{-6}$ )

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



## Terminologia



Bit = binary digit.

- 1 byte = 8 bit.
  - 1kbyte =  $2^{10}$ byte = 1,024
  - 1Mbyte =  $2^{20}$ byte = 1,048,576.
  - 1Gbyte =  $2^{30}$ byte = 1,073,741,824.
  - 1Tbyte =  $2^{40}$ byte = 1,099,511,627,776.
- 
- Parola (word) numero di bit trattati come un unicum dall'elaboratore.
  - Le parole oggi arrivano facilmente a 64bit (Itanium).



## Sommario



Rappresentazione binaria dell'Informazione

Conversione in e da un numero binario

Operazioni elementari su numeri binari

I numeri reali

Codifica IEEE754 dei numeri reali



## Conversione base 10 -> base 2



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

$$1492 = 2 \times 746 + 0$$

← 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



## Conversione base 10 → base 2



Perchè funziona?

Esempio:

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

Prendiamo il numero E e dividiamo per 2.

Se E è pari il resto, R0, sarà 0, altrimenti sarà 1.

$$R0 = \text{resto}(5/2) = 1$$

$$E1 = \text{quoz}(5/2) = 2$$

Infatti:  $\text{int}(E / 2) * 2 + R0 = E$ .

$$R1 = \text{resto}(2/2) = 0$$

$$E2 = \text{quoz}(2/2) = 1$$

Chiamiamo  $E1 = \text{int}(E / 2)$  e procediamo.

Prendiamo E1 e dividiamo per 2.

Se E1 è pari il resto, R1, sarà 0, altrimenti sarà 1.

$$R2 = \text{resto}(1/2) = 1$$

$$E = R0 + R1 * 2 + R2 * 2^2 \\ = 101$$

$$E1 = \text{int}(E1 / 2) * 2 + R1$$

$$E = \text{int} [ \text{int} ( \text{int} (E1 / 2) * 2 + R1) * 2 + R0 ]$$





## “Spelling” di un numero



Vogliamo rappresentare  $1492_{\text{dieci}}$

$$1492 = 10 \times 149 + 2$$

← Cifra meno significativa

$$149 = 10 \times 14 + 9$$

$$14 = 10 \times 1 + 4$$

$$1 = 10 \times 0 + 1$$

← Cifra più significativa



## 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  $n$



## Conversione da base n a base 10



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

$$E = \sum_k c_k b_k$$

- ciascuna cifra  $n$ -aria è moltiplicata per la base elevata ad  $n_i$  dove  $i$  è la posizione della cifra nel numero.
- i valori così ottenuti sono sommati per ottenere il numero in decimale.

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



## 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 -->  $1001_{\text{due}}$   
Fhex -->  $1111_{\text{due}}$   
9Fhex -->  $10011111_{\text{due}}$

- 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  $1101011_{\text{due}}$  in esadecimale:

$1011_{\text{due}} \rightarrow B_{\text{hex}}$

$110_{\text{due}} \rightarrow 6_{\text{hex}}$

$1101011_{\text{due}} \rightarrow 6B_{\text{hex}}$

Viene aggiunto un "leading" 0



## Sommario



Rappresentazione binaria dell'Informazione

Conversione in e da un numero binario

**Operazioni elementari su numeri binari**

I numeri reali

Codifica IEEE754 dei numeri reali



## 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	Doppia codifica per lo 0.
	01	1	
	10	-1	
	11	0	

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

00	0	
01	1	$10 + 1 = 11$
10	-2	
11	-1	

**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



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

$$100010 \quad \Rightarrow \quad 00100010_2 = 32_{10}$$

$$10011_2 \quad \Rightarrow \quad 11110011_2 = -13_{10}$$

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

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

Fuori dalla  
Capacità  
di 8 bit

$$\begin{array}{r}
(1)11000100 \\
00100010 + \\
11110011 = \\
\text{-----} \\
00010101 \rightarrow 21_{10}
\end{array}$$



## Moltiplicazione binaria



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

$$\begin{array}{r}
 11011 \times \\
 111 = \\
 \hline
 11111 \\
 11011+ \\
 11011- \\
 \hline
 1 \\
 1010001+ \\
 11011- - \\
 \hline
 10111101
 \end{array}$$



## 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:

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

$$213_{10} = (2 \times 10^2 + 1 \times 10^1 + 3 \times 10^0) / 10^1 =$$

$$(2 \times 10^2 + 1 \times 10^1 + 3 \times 10^0) \times 10^{-1} =$$

$$(2 \times 10^2 \times 10^{-1} + 1 \times 10^1 \times 10^{-1} + 3 \times 10^0 \times 10^{-1}) =$$

$$(2 \times 10^1 + 1 \times 10^0 + 3 \times 10^{-1}) = 21.3 \text{ cvd.}$$

Esempio:

$$23 / 4 = 5,75 \Rightarrow 10111 / 100 =$$

$$(1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0) \times 2^{-2} =$$

$$(1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}) = 5,25 \text{ cvd.}$$



## Sommario



Rappresentazione binaria dell'Informazione

Conversione in e da un numero binario

Operazioni elementari su numeri binari

**I numeri reali**

Codifica IEEE754 dei numeri reali



## 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 rappresentazione.**



## 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,

$$0,75 - 1x2^{-1} \Rightarrow 1$$

$$0,25 - 1x2^{-2} \Rightarrow 1$$

##### 11

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

$$0,76 - 1x2^{-1} \Rightarrow 1$$

$$0,26 - 1x2^{-2} \Rightarrow 1$$

$$0,01 - 1x2^{-3} \Rightarrow 0$$

$$0,01 - 1x2^{-4} \Rightarrow 0$$

$$0,01 - 1x2^{-5} \Rightarrow 0$$

$$0,01 - 1x2^{-6} \Rightarrow 0$$

$$0,01 - 1x2^{-7} \Rightarrow 1$$

0,0021875



## Sommario



Rappresentazione binaria dell'Informazione

Conversione in e da un numero binario

Operazioni elementari su numeri binari

I numeri reali

**Codifica IEEE754 dei numeri reali**



# Standard IEEE 754 (1980)

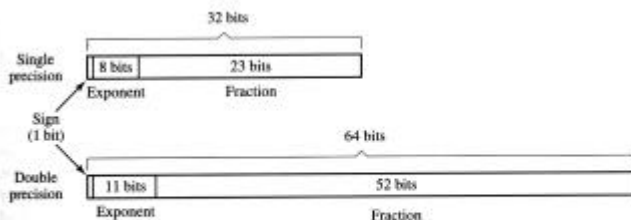


Figure 2-10 Single-precision and double-precision IEEE 754 floating point formats.

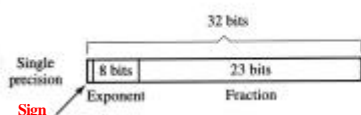
Rappresentazione polarizzata dell'esponente:

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:  $\pm 1,xxxxxx \times 2^e$        $-1,01011 \times 2^3$
- 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

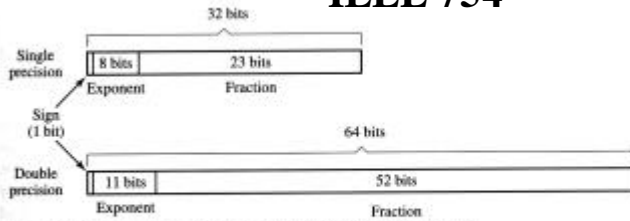


Figure 2-10 Single-precision and double-precision IEEE 754 floating point formats.

Configurazioni notevoli:

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

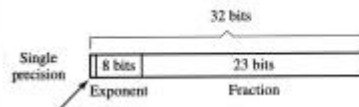
Range degli esponenti (8 bit):  $1--254 \Rightarrow -126 \leq \text{exp} \leq +127$ .

Numeri float:  $1.0 \times 2^{-126} = 1.175494351 \times 10^{-38} \div 3.402823466 \times 10^{38} = 1.1...11 \times 2^{127}$

Min denormalizzato:  $2^{-149}$



# Capacità dello Standard IEEE 754



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

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

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

Capacità float:

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

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

Risoluzione sul numero più piccolo:  $2^{-(23-126)} = 1.401298464324817 \times 10^{-45}$

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

Soluzione: Numeri denormalizzati. Min denormalizzato:  $2^{-149}$

Numero denormalizzato  $= 1.401298464324817 \times 10^{-45}$   
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:

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

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

.....



## Sommario



Rappresentazione binaria dell'Informazione

Conversione in e da un numero binario

Operazioni elementari su numeri binari

I numeri reali

Codifica IEEE754 dei numeri reali