

# Codifica dell'informazione numerica



Nicola Basilico, [nicola.basilico@unimi.it](mailto:nicola.basilico@unimi.it)  
Gabriella Trucco, [gabriella.trucco@unimi.it](mailto:gabriella.trucco@unimi.it)  
Matteo Re, [matteo.re@unimi.it](mailto:matteo.re@unimi.it)

## Numeri frazionari

- Notazione posizionale e metodo polinomiale si estendono facilmente:

$$\begin{aligned}(I.c_{-1}c_{-2}\dots c_{-m})_B &= \\ I + c_{-1} \cdot B^{-1} + c_{-2} \cdot B^{-2} + \dots + c_{-m} \cdot B^{-m} &= \\ I + \sum_{i=-m}^{-1} c_i \cdot B^i &\end{aligned}$$

- Esempio:  $(0.587)_{10} = 5 \cdot 10^{-1} + 8 \cdot 10^{-2} + 7 \cdot 10^{-3}$
- Conversione da base 2 a base 10:  
 $(0.1011)_2 = 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} = (0.6875)_{10}$

# Rappresentazione dei numeri frazionari

- Come convertire la parte frazionaria da base 10 a base 2? Metodo **iterativo** (moltiplicazioni)

## Procedimento

Abbiamo un numero  $(I.F)_{10}$  da convertire nella base  $B = 2$ :

1. la parte intera  $I$  si converte come abbiamo visto in precedenza;
  2. moltiplicare  $.F$  per 2;
  3. la parte intera del risultato diventa la cifra **più** significativa (la prima che resta da calcolare) del numero in base 2;
  4. tornare a **2** considerando la parte frazionaria del risultato ottenuto al posto di  $.F$
- Quando si finisce?
    - La parte frazionaria ottenuta è 0 (solo i numeri scrivibili come  $N(2^{-z})$ , con  $N$  e  $z$  interi, possono essere rappresentati su un numero finito di bit)
    - Il numero di bit ottenuti costituisce un'approssimazione che si ritiene sufficiente

## Rappresentazione dei numeri frazionari

Esempio:  $(0.587)_{10} = (?)_2$

$0.587 \times 2 = 1.174$	parte intera = 1	MSD
$0.174 \times 2 = 0.348$	parte intera = 0	
$0.348 \times 2 = 0.696$	parte intera = 0	
$0.696 \times 2 = 1.392$	parte intera = 1	
$0.392 \times 2 = 0.784$	parte intera = 0	
$0.784 \times 2 = 1.568$	parte intera = 1	
...		

$$(0.587)_{10} = (0.100101\dots)_2$$

# Rappresentazione approssimata dei numeri reali

## Rappresentazione in **virgola fissa**

- La parte intera è rappresentata sempre su  $n$  bit mentre la parte frazionaria è rappresentata sempre su  $m$  bit
- La virgola può cadere in un'unica posizione, essendo implicita può essere omessa

Esempi: assumiamo base  $B = 2$ ,  $m = n = 4$  e notazione in  $C2$

$$(+4.25)_{10} = (0100.0100)_{C2} \quad (-4.25)_{10} = (1100.0100)_{C2}$$

$$(+3.35)_{10} = (0011.0101)_{C2} \quad (-3.35)_{10} = (1101.0101)_{C2}$$

# Rappresentazione approssimata dei numeri reali

## Rappresentazione in **virgola mobile**

- Un numero razionale  $(N)_B$  è espresso come:

$$N = (-1)^s \cdot m \cdot B^e$$

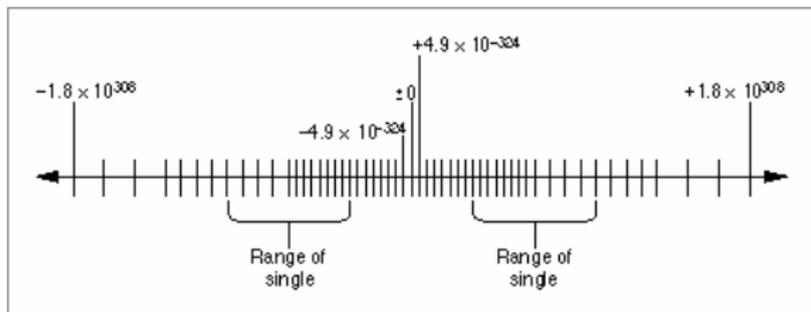
- $s$  è il **segno**, ( $0 \rightarrow +$ ,  $1 \rightarrow -$ )
- $m$  è la **mantissa**, numero frazionario su  $p$  bit in base  $B$
- $e$  è l'**esponente**, numero intero
- Forma **normalizzata**: la parte intera della mantissa ha una cifra significativa, es.  $5.79 \cdot 10^2$
- In base 2:  $\pm 1.b_{-1}b_{-2} \dots b_{-p} \cdot 2^e$ , rappresentiamo:
  - Segno: 0, 1
  - Mantissa: la parte frazionaria  $b_{-1}b_{-2} \dots b_{-p}$  (la parte intera è implicita)
  - Esponente:  $e$

## Approssimare i numeri reali

Il numero:  $\pm 1.b_{-1}b_{-2} \dots b_{-p} \cdot 2^e$  è associato al valore:

$$\pm(1 + b_{-1} \cdot 2^{-1} + b_{-2} \cdot 2^{-2} \dots b_{-p} \cdot 2^{-p}) \cdot 2^e$$

Risoluzione variabile: il contributo del bit meno significativo ( $b_{-p}$ ) dipende da  $e$ .



# Virgola fissa e virgola mobile

Come confrontare le due diverse **rappresentazioni** ( $R_{N,VF}$  e  $R_{N,VM}$ )?

- Minimo e massimo numero rappresentabile  $V_{min}$  e  $V_{max}$

Dato un numero  $x$  che si vuole rappresentare:

- **Errore assoluto**  $\epsilon_A(x)$ : è la differenza tra il numero  $x$  e la sua più vicina rappresentazione
- **Errore relativo**  $\epsilon_R(x)$ : è la differenza tra il numero  $x$  e la sua più vicina rappresentazione in percentuale del valore  $x$ :  $\frac{\epsilon_A(x)}{x}$

Virgola fissa	Virgola mobile
$R_{N,VF} = iii.fff$	$R_{N,VM} = i.fff \cdot 10^{ee}$
$V_{min} = 000.000, V_{max} = 999.999 \cong 10^3$	$V_{min} = 0, V_{max} = 9.999 \cdot 10^{99} \cong 10^{100}$
$\epsilon_A \leq 10^{-3}$	$\epsilon_A \leq 10^{-3} \cdot 10^{ee} \leq 10^{ee-3}$
$\epsilon_R \in \left[ \frac{10^{-3}}{10^3} = 10^{-6}, \frac{0.001}{0} \right)$	$\epsilon_R = \frac{10^{ee-3}}{m \cdot 10^{ee}} \cong 10^{-3}$

- A parità di cifre utilizzate (nell'esempio sono 6), la rappresentazione in virgola mobile è in grado di rappresentare un numero più elevato di valori
- In virgola fissa: precisione assoluta costante e relativa variabile
- In virgola mobile: precisione assoluta variabile con  $N$  e relativa approssimativamente costante

## Numeri reali: lo standard IEEE 754

- IEEE Standard for Floating Point Arithmetic (IEEE 754), formato a precisione *singola*: 32 bit



Quando rappresentiamo un **numero normalizzato**:

- i bit da 0 a 22 sono la mantissa i.e., la parte frazionaria del numero (la parte intera si intende implicitamente pari a 1)
- i bit da 23 a 30 sono l'esponente: un intero tra -126 e 127 memorizzato su 8 bit in **ecceso 127** (significa che memorizziamo l'intero positivo  $E = e + 127$  anzichè  $e$ )
- il bit 31 è il segno: 0 corrisponde a +, 1 corrisponde a -

$$R_{N,VM} = \pm I.F \cdot B^E$$

## Esponente: rappresentazione in eccesso 127

- Per rappresentare il numero  $e_{(10)}$  si somma ad esso  $2^{8-1} - 1 = 127_{(10)}$
- Si rappresenta il valore risultante  $E_{(10)} = e_{(10)} + 127$
- Intervallo rappresentato (numeri normalizzati):  
 $-126 \leq e_{(10)} \leq 127 \rightarrow 0 < E_{(10)} < 255$

Esempio 1:  $(-34)_{(10)} = (?)_{8bit, e127}$

$$e_{(10)} = -34, E_{(10)} = -34 + 127 = +93_{(10)} = 01011101_{(2)}$$

Esempio 2:  $(11100101)_{8bit, e127} = (?)_{(10)}$

$$E_{(10)} = +229, e_{(10)} = 229 - 127 = 102_{(10)}$$

# Mantissa

- La mantissa  $M$  viene rappresentata nella forma  $1.c_{-1}c_{-2}\dots c_{-23}$
- Il bit  $c_0$  corrisponde al peso  $2^0 = 1$ : è, per convenzione, sempre uguale a 1 e non si rappresenta
- Il punto decimale segue sempre il bit  $c_0$  e, per convenzione, non si rappresenta; questo corrisponde alla **rappresentazione normalizzata** (il punto decimale è posto dopo l'unica cifra significativa della parte intera)
- i 23 bit di  $M$  rappresentano quindi l'intervallo  $[1, 2)$

## Numeri reali: lo standard IEEE 754

Esempio 1: convertiamo  $(17.375)_{10}$ , dobbiamo determinare  $s$ ,  $m$ ,  $E$

- Il numero è positivo  $\rightarrow s = 0$
- Converto la parte intera in binario:  $(17)_{10} = (10001)_2$
- Converto la parte frazionaria  $(.375)_{10} (= \frac{3}{2^3})$

$0.375 \cdot 2 = 0.75$	parte intera = 0	MSD	$(.375)_{10} = (.011)_2$
$0.75 \cdot 2 = 1.5$	parte intera = 1		
$0.5 \cdot 2 = 1.0$	parte intera = 1		

- Unisco i risultati:  $(10001.011)_2$
- Normalizzazione:  $(1.0001011)_2 \cdot (10)_2^4$
- Mantissa:  $m = 0001\ 0110\ 0000\ 0000\ 0000\ 000$
- Esponente:  $E = e + 127 = (4)_{10} + (127)_{10} = (131)_{10} = (10000011)_2$

## Numeri reali: lo standard IEEE 754

Esempio 2: convertiamo  $(-0.8)_{10}$ , dobbiamo determinare  $s$ ,  $m$ ,  $E$

- Il numero è negativo  $\rightarrow s = 1$
- Converto la parte intera in binario:  $(0)_{10} = (0)_2$
- Converto la parte frazionaria  $(.8)_{10}$

$0.8 \cdot 2 = 1.6$	parte intera = 1	MSD
$0.6 \cdot 2 = 1.2$	parte intera = 1	
$0.2 \cdot 2 = 0.4$	parte intera = 0	
$0.4 \cdot 2 = 0.8$	parte intera = 0	
$0.8 \cdot 2 = \dots$	la sequenza 1100 si ripete	

- Unisco i risultati:  $(0.8)_{10} = (0.\overline{1100})_2$
- Normalizzazione:  $(0.\overline{1100})_2 = (0.1100 \overline{1100})_2 = (1.100 \overline{1100})_2 \cdot (10)_2^{-1}$
- Mantissa:  $m = 100 \ 1100 \ 1100 \ 1100 \ 1100 \ 1100$
- Esponente:  
 $E = e + 127 = (-1)_{10} + (127)_{10} = (126)_{10} = (1111110)_2 \rightarrow (01111110)_2$

## Numeri reali: lo standard IEEE 754

Certi valori di  $m$  e di  $e$  sono utilizzati secondo diverse convenzioni, definite dallo standard ( $E = e + 127$ )

- Se  $0 < E < 255$  ( $e \in [-126, 127]$ )  $\rightarrow$  numero normalizzato
- Se  $m = 0$ ,  $E = 0 \rightarrow \pm 0$  (a seconda di  $s$ )
- Se  $m = 0$ ,  $E = 255 \rightarrow \pm \infty$  (a seconda di  $s$ )
- Se  $m \neq 0$ ,  $E = 255 \rightarrow \text{NaN}$  (Not a Number)
- Se  $m \neq 0$ ,  $E = 0 \rightarrow$  numero subnormalizzato

## Standard IEEE 754: numeri subnormalizzati

Quale è il numero positivo normalizzato più piccolo che possiamo rappresentare?

- $s = 0, e = (-126)_{10} \rightarrow E = (0000\ 0001)_2, m = 0000 \dots 000$
- $(1.0)_2 \cdot (10)_2^{-126} = 2^{-126} \approx 1.17 \cdot 10^{-38}$

Il successivo?

- $s = 0, e = (-126)_{10} \rightarrow (0000\ 0001)_2, m = 0000 \dots 001$
- $(1.0 \dots 01)_2 \cdot (10)_2^{-126} = 2^{-126} + 2^{-126-23} \approx 1.17 \cdot 10^{-38} + 1.4 \cdot 10^{-45}$

In prossimità dello zero non abbiamo risoluzione costante!

Possiamo riempire questa regione in modo più “furbo” (precisione maggiore e localmente costante)? → I numeri subnormali hanno un ordine di grandezza minore di quello del più piccolo numero normalizzato

## Standard IEEE 754: numeri subnormalizzati

- Ad  $E$  viene assegnato il codice 0000 0000 che si interpreta come esponente pari a  $-126$  (non  $-127$  !)
- la mantissa è un numero **diverso** da 0 e la parte intera, diversamente dai normalizzati, è implicitamente assunta essere 0
- Quindi un numero subnormalizzato è sempre fatto così :

$$(-1)^s \cdot (0.m) \cdot (10)_2^{-126}$$

Quale è il numero positivo subnormalizzato più **piccolo** che possiamo rappresentare?

- $s = 0$ ,  $e = (-126)_{10}$  subnormalizzato  $\rightarrow E = (0000\ 0000)_2$ ,  
 $m = 0000 \dots 001$
- $(0.0 \dots 01)_2 \cdot (10)_2^{-126} = 2^{-23} \cdot 2^{-126} = 2^{-149} = 1.4012985 \cdot 10^{-45}$
- il successivo sarà ovviamente  $2 \cdot 2^{-149} = 2.8025969 \cdot 10^{-45}$

In prossimità dello 0 abbiamo una precisione maggiore e localmente costante

## Standard IEEE 754: numeri subnormalizzato

Quale è il numero positivo subnormalizzato più **grande** che possiamo rappresentare?

- $s = 0$ ,  $e = (-126)_{10}$  subnormalizzato  $\rightarrow E = (0000\ 0000)_2$ ,  
 $m = 1111 \dots 111$
- $(0.1 \dots 11)_2 \cdot (10)_2^{-126} = (2^{-23} + 2^{-22} + \dots + 2^{-1}) \cdot 2^{-126} =$   
 $(1 - 2^{-23}) \cdot 2^{-126} = 1.1754942 \cdot 10^{-38}$
- il precedente sarà ovviamente  $(1 - 2^{-22}) \cdot 2^{-126} = 1.1754941 \cdot 10^{-38}$

Anche qui abbiamo una precisione localmente costante

## Standard IEEE 754: numeri subnormalizzato

Esempio 1: rappresentare in formato IEEE 754 il numero  
 $-(0.125)_{10} \cdot 2^{-125}$

- Il numero è negativo  $\rightarrow s = 1$
- Il numero è esprimibile come un subnormalizzato? Per esserlo devo poterlo scrivere come  $(0.m) \cdot 2^{-126}$  ( $m$  ha 23 bit)
- $0.125 \cdot 2^{-125} = 0.125 \cdot 2^1 \cdot 2^{-126} = 0.25 \cdot 2^{-126}$ , quindi posso scriverlo come subnormalizzato
- converto  $0.25 \left(\frac{1}{2^2}\right)$  in base 2:  $(0.25)_{10} = (0.01)_2$
- Mantissa  $m = 010\dots 0$
- Esponente  $E = 0000\ 0000$