Corso di Calcolatori Elettronici

Rappresentazione dei numeri reali in un calcolatore

Prof. Roberto Canonico

Università degli Studi di Napoli Federico II
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione

v. 2023

Rappresentazione di numeri reali

- Con un numero finito di cifre non è possibile rappresentare esattamente un qualsiasi numero reale
- Con un numero finito di cifre è possibile rappresentare solo un sottoinsieme finito di numeri razionali
- *Dato un numero reale arbitrario r, con un numero finito di cifre è possibile rappresentare un numero razionale r' che approssima con un certo errore* il numero reale r
- Nelle macchine digitali vengono usate due notazioni:

A) <u>Notazione in virgola fissa</u>

Dedica parte delle cifre alla parte intera e le altre alla parte frazionaria

± XXX .YY

B) <u>Notazione in virgola mobile</u>

Dedica alcune cifre a rappresentare un esponente della base che indica l'ordine di grandezza del numero rappresentato

Numeri reali: rappresentazione in virgola fissa

- Quando di un numero frazionario si rappresentano separatamente la parte intera e la parte frazionaria si parla di rappresentazione in *virgola fissa*
- La rappresentazione dei due contributi può essere realizzata secondo una delle tecniche viste in precedenza
- La parte frazionaria è rappresentata con un numero finito m di cifre binarie, scalata di un fattore 2^m che la rende intera
- La posizione della virgola è fissa e resta sottintesa

Numeri reali in virgola fissa

- La stringa

$$,b_{-1}b_{-2}..b_{-m}$$

  si interpreta come

$$b_{-1}2^{-1}+b_{-2}2^{-2}+\ \dots\ +b_{-m}2^{-m}$$

- Esempio: .1011

  si interpreta come

$$2^{-1}+2^{-3}+2^{-4}= 1/2 + 1/8 + 1/16 = 0,5 + 0,125 + 0,0625 = 0,6875$$

  ovvero come

$$11 / 16 = 0,6875$$

- La stringa 1011 è rappresentativa dell’intero $(11)_{10}$ che va scalato del fattore 2^{-4}

Numeri reali: rappresentazione in virgola mobile

- Un numero reale x può essere rappresentato dalla tripla

$$(s,m,e)$$

tale che:

$$x = (-1)^s \cdot m \cdot b^e$$

  - *s è il segno (s=0 positivo, s=1 negativo)*
  - *m* è detta *mantissa*
  - *e* è detto *esponente*
  - *b* è la base di numerazione adottata
- In macchina sia **m** che **e** hanno un numero prefissato di cifre
  - intervalli limitati ed errori di arrotondamento

Vantaggi della rappresentazione in virgola mobile

- La notazione a virgola mobile permette di rappresentare un ampio intervallo di valori con un numero di cifre prefissato, grazie alla **flessibilità della posizione della virgola**,
  che dipende dal valore dell'esponente
  - Esempi (in base 10):
    considerando 3 cifre per la mantissa e 2 per l'esponente
    - PI Greco: 0.314×10^{1}
    - Massa di un Elettrone (kg): 0.911×10^{-30}
    - Massa della Via Lattea (kg): 0.136×10^{43}

Rappresentazione finita e discreta dei numeri reali

- In un intervallo, comunque piccolo, esistono infiniti numeri reali
  - i numeri reali formano un continuo
- I numeri rappresentabili con un numero finito di cifre costituiscono invece un sottoinsieme finito
- Ciascun elemento di questo sottoinsieme sarà chiamato a rappresentare i numeri reali che si trovano "nell'intorno"
- In altri termini, diviso l'insieme dei numeri reali in intervalli di fissata dimensione, si ha che ogni $x \in [X_i, X_{i+1}[$ viene rappresentato con il valore $X = X_i$
- Nota: gli intervalli $[X_i, X_{i+1}[$ non hanno tutti la stessa ampiezza a causa della finitezza del numero di cifre della mantissa

Normalizzazione

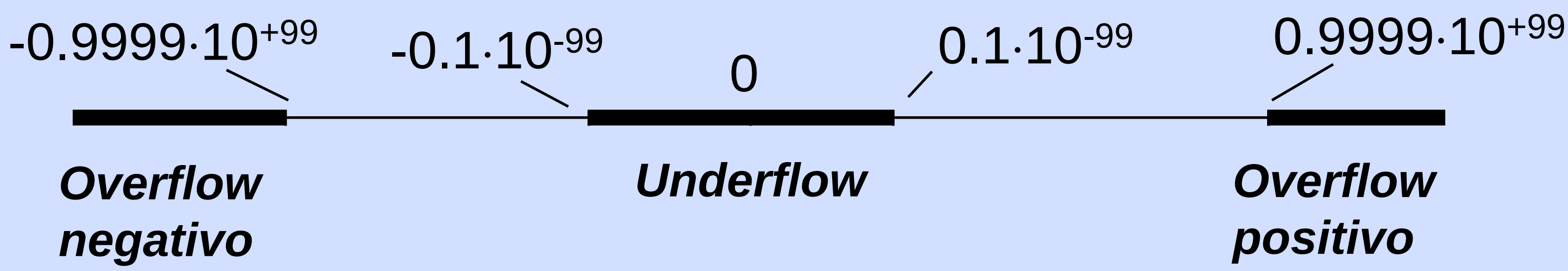
- Per ciascun numero esistono infinite coppie mantissa-esponente che lo rappresentano
- Esempio (b=10):
    - 346.09801 è rappresentato da
        - » m= 346.09801, e=0 oppure
        - » m= 346098.01, e= - 3 oppure
        - » m= 0.034609801, e= 4 ecc…
- Per **rappresentazione normalizzata** del numero si intende convenzionalmente quella in cui la mantissa ha la prima cifra a destra della virgola diversa da zero
- ovvero: $1/b \leq m < 1$
- Esempio:

m=0.34609801, e=3

Esempio: intervallo di rappresentazione

- Con b=10, usando 4 cifre per *m* e 2 per *e* (più due bit per i relativi segni), l'insieme rappresentabile (utilizzando solo rappresentazioni normalizzate) è:

$$[-0.9999\times10^{\mathbf{99}},-0.1000\times10^{\mathbf{-99}}]\cup\{0\}\cup[+0.1000\times10^{\mathbf{-99}},+0.9999\times10^{\mathbf{99}}]$$

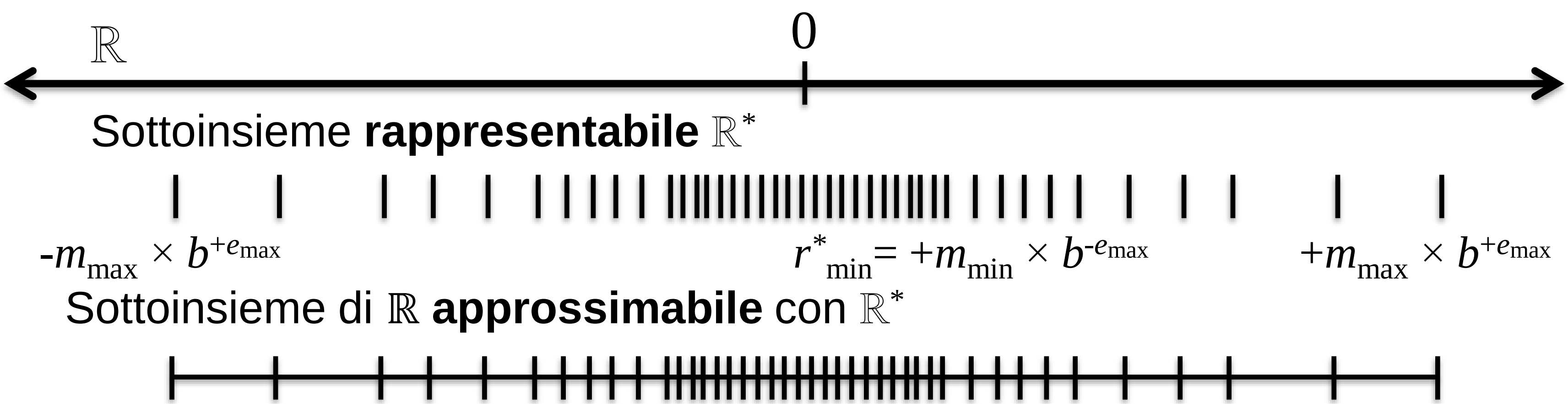


Con le stesse **6=4+2** cifre in virgola fissa ± **XXXX .YY** :

- L'intervallo scende **[-9999.99,+9999.99]**
- Ma si hanno **6** cifre significative invece di **4**

La discretizzazione di $\mathbb{R}$

- Essendo l'insieme reale **denso**, per ogni coppia di elementi distinti di $\mathbb{R}$ vi è sempre un elemento compreso tra i due
  - I valori rappresentabili di $\mathbb{R}^*$ sono un sottoinsieme che contiene un numero finito di valori reali



I valori rappresentabili **<u>NON</u> sono equidistanti** nell'intervallo di rappresentazione!

Ogni elemento in $\mathbb{R}^*$ approssima un intervallo di valori del continuo

Approssimazione

- Errore assoluto: $e_A = | r - r' |$
- Errore relativo: $e_R = | r - r' | / | r |$
- Nella rappresentazione in virgola mobile l’errore assoluto non è costante
- L'errore di approssimazione è piccolo in prossimità dello zero e va aumentando progressivamente a mano a mano che il numero aumenta (in valore assoluto)
- Ad esempio:
  - in prossimità dello zero l’errore massimo che può essere commesso è $0.1001*10^{-99} - 0.1000*10^{-99} = \mathbf{0.0001*10^{-99}}$
  - in prossimità dell’estremo superiore dell’intervallo di rappresentazione, invece, l’errore massimo che si può commettere è $0.9999*10^{99} - 0.9998*10^{99} = \mathbf{0.0001*10^{99}}$
- Si commettono quindi “errori piccoli” su “numeri piccoli” ed “errori grandi” su “numeri grandi”
- Quello che resta inalterato è invece l’errore relativo, costante su tutto l’asse di rappresentabilità

Overflow e Underflow

- L’errore relativo dipende dal numero di cifre della mantissa
- Gli estremi dell’intervallo di rappresentazione dipendono dal numero di cifre dell’esponente

- Nel caso precedente di 2 cifre per l’esponente, si ha overflow per numeri maggiori (in modulo) di 10^{99} e si ha underflow per numeri minori (in modulo) di 10^{-99}

Standard IEEE 754 (1985)

- Formato standard indipendente dall'architettura
- Precisione semplice a 32 bit:

1	8	23
SEGNO	ESPONENTE	MANTISSA

- Precisione doppia a 64 bit

1	11	52
SEGNO	ESPONENTE	MANTISSA

Argomento	Precisione singola 32 Bit	Precisione doppia 64 bit
Bit del segno	1	1
Bit per l'esponente	8	11
Bit per la mantissa	23	52
Cifre decimali mantissa	Circa 7 (23/3.3)	Circa 15 (52/3.3)
Esponente (rappresentazione)	base 2 ad eccesso 127	base 2 ad eccesso 1023
Esponente (valori)	[-126, 127]	[-1022, 1023]

IEEE 754 a 32 bit

1	8	23
SEGNO	ESPONENTE	MANTISSA

$$\mathbf{x = (-1)^{S} \times 1.fraction \times 2^{exponent-bias}}$$

- **ESPONENTE**
  - Rappresentato in *eccesso 127* (bias = 127)
  - I valori di esponente ammessi sono nell'intervallo **[-126, +127]**
  - **I valori -127 e +128 sono riservati per rappresentazioni speciali**
- **MANTISSA**
  - Se ne rappresenta *solo la parte frazionaria*

$$\begin{cases} N = (-1)^{S} \times 1.fraction \times 2^{exponent-127}, & 1 \leq exponent \leq 254 \\ N = (-1)^{S} \times 0.fraction \times 2^{exponent-126}, & exponent = 0 \end{cases}$$

IEEE 754: forma normalizzata

- La mantissa binaria normalizzata deve presentare un 1 a sinistra della virgola binaria. L'esponente deve essere aggiustato di conseguenza
- Essendo sempre presente tale cifra non è informativa così come la virgola binaria; esse vengono considerate implicitamente presenti e non vengono memorizzate
- Per evitare confusione con una frazione tradizionale la combinazione dell'1 implicito della virgola binaria e delle 23/52 cifre significative vengono chiamate ***significando*** (invece che frazione o mantissa)
  - Tutti i numeri normalizzati hanno un esponente e>0
  - Tutti i numeri normalizzati hanno un significando s=1.f tra 1≤s<2
  - I numeri normalizzati non possono avere un esponente composto da soli 1. Tale configurazione serve per modellare il valore infinito (∞)

IEEE 754: forma denormalizzata

- La mantissa binaria denormalizzata può assumere qualsiasi configurazione. Questa rappresentazione viene utilizzata per rappresentare valori inferiori a 2^{-126}
  - Tutti i bit dell'esponente sono posti a 0 (questa configurazione indica l'utilizzo della forma denormalizzata)
  - <u>Il bit della mantissa a sinistra della virgola binaria è posto implicitamente a 0 per i numeri informa denormalizzata</u>
  - Il numero più piccolo rappresentabile in questa configurazione è composto da una mantissa con tutti 0 a eccezione del bit più a destra

Esempio

$$-\left(6+\frac{5}{8}\right)=-\left(4+2+\frac{4}{8}+\frac{1}{8}\right)=-\left(4+2+\frac{1}{2}+\frac{1}{8}\right)$$

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

$$=-\left(110.101_{2}\right)=-\left(1.10101_{2}\times 2^{2}\right)$$

Esponente:

$$exponent-127=2\Rightarrow exponent=129$$

Rappresentazioni speciali

Esponente 255 = 11111111_2 indica un valore speciale

Esponente 255 = 11111111_2 ed $f = 0 \rightarrow$ valore rappresentato $\pm\infty$

Esponente 255 = 11111111_2 ed $f \neq 0 \rightarrow$ valore rappresentato
Not a Number (NaN)

Not a Number (NaN) indica un valore indefinito
Esso è impiegato per rappresentare il risultato di operazioni di calcolo non definite, come la divisione 0 / 0
o la radice quadrata di un numero negativo

Rappresentazione di zero

- esponente tutti 0
- mantissa tutti zero
- Segno: sia + che -

+0: 0 00000000 00000000000000000000000

-0: 1 00000000 00000000000000000000000

Da decimale a floating point

- Caso semplice:
  numero razionale con denominatore potenza di 2
- Esempio: - ¾ = -0.75_{10} =- 0.11_2
- Forma normalizzata: $- 1.10_2 \cdot 2^{-1}$
- Rappresentazione IEEE 754 a singola precisione:

$$\mathbf{x = (-1)^S \times 1.fraction \times 2^{exponent-bias}}$$

s = 1

exponent-bias = -1 $\rightarrow$ exponent=bias-1=127-1=126 = 01111110_2

fraction = 1000…..00 (23 bit)

1	**0111 1110**	**1000 0000 0000 0000 0000 000**

Da decimale a floating point

- Esempio: $+ 1/3 = +0.3333\ldots_{10} =+ 0.01010101\ldots._{2}$
- Forma normalizzata: $+ 1.010101\ldots._{2} \cdot 2^{-2}$
- Rappresentazione IEEE 754 a singola precisione:

$$\mathbf{x = (-1)^{S} \times 1.fraction \times 2^{exponent-bias}}$$

s = 0

exponent-bias = -2 → exponent=bias-2=127-2=125 = 01111101_2

fraction = 0101 0101 0101 0101 0101 010 (23 bit)

0	**0111 1101**	**0101 0101 0101 0101 0101 010**