TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Macroarea di Ingegneria
Dipartimento di Ingegneria Civile e Ingegneria Informatica

Rappresentazione dei Numeri in Virgola Mobile

Valeria Cardellini

Corso di Calcolatori Elettronici
A.A. 2019/20

Argomenti

- Rappresentazione di numeri reali
- Rappresentazione in virgola mobile
- Standard IEEE 754
- Confronto e addizione in virgola mobile

Rappresentazione di numeri reali

- Con un numero finito di cifre è possibile rappresentare solo un numero razionale che *approssima* con un certo *errore* il numero reale dato
- Si usano due notazioni
  - Notazione **in virgola fissa**
    - ✓ Dedica parte delle cifre alla parte intera e le rimanenti alla parte frazionaria (la posizione della virgola è fissata):
    - ✓ Esempio: XXX .YY
  - Notazione **in virgola mobile**
    - ✓ Dedica alcune cifre a rappresentare un esponente della base, che indica l'ordine di grandezza del numero rappresentato
    - ✓ Altre cifre sono usate per rappresentare la mantissa
    - ✓ Esempio notazione scientifica $\pm$ X.XX·$10^{\pm YY}$

Limitazioni della rappresentazione in virgola fissa

- Fissato il numero di cifre e la posizione della virgola:
  1. Non rappresenta bene numeri frazionari molto grandi
  2. Non rappresenta bene numeri (frazioni) molto piccoli
- Esempio: come rappresentare in virgola fissa
  - 500000000000000.00003
  - 0.00000000000000000008
  - Fissato il numero di cifre per rappresentare parte intera e frazionaria

Perché la rappresentazione in virgola mobile

- La rappresentazione in virgola mobile estende l'intervallo di numeri rappresentati a parità di cifre, rispetto alla notazione in *virgola fissa*
- *Notazione scientifica*
  - Si esprime 432 000 000 000 come 4.32×10^{11}
  - Le 11 posizioni dopo il 4 vengono espresse dall'esponente
- Principio della rappresentazione in virgola mobile (detta anche *floating point*)
  - Si fa scorrere la virgola decimale fino ad una posizione conveniente, tenendo conto di ogni spostamento con l'esponente

Rappresentazione in virgola mobile

- E' utile perché
  - Permette di rappresentare in maniera compatta numeri molto grandi, ma anche molto piccoli, sia positivi sia negativi
- Numeri reali rappresentati tramite una coppia di numeri <m, e>

$$n = \pm m \cdot b^{\pm e}$$

  - m: ***mantissa*** (detto anche *significante*), normalizzata tra due potenze successive della *base* b

$$b^{i-1} \leq |m| < b^i$$

  - e: ***esponente*** intero (detto anche *caratteristica*)
- Sia m che e hanno un numero fissato di cifre:
  - Intervalli limitati
  - Errori di arrotondamento

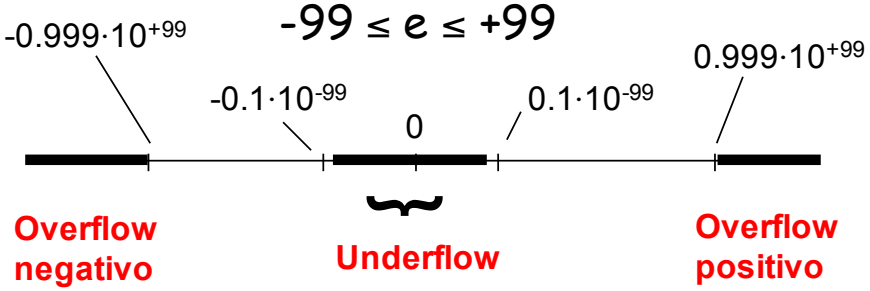
Esempio in base 10

- Numerali a 5 cifre $\pm$ .XXX $\pm$ EE
- Mantissa: 3 cifre con segno

$$0.1 \leq |m| < 1$$

- Esponente: 2 cifre con segno

$$-99 \leq e \leq +99$$



- Notare che con lo stesso numero di cifre in notazione a virgola fissa $\pm$ XXX .YY
  - L'intervallo scende [-999.99, 999.99]
  - Ma si hanno 5 cifre significative invece di 3

Rappresentazione in virgola mobile dei numeri binari

- Lo stesso approccio della virgola mobile può essere seguito per rappresentare i numeri binari nella forma $\pm m \cdot b^{\pm e}$

$$\pm 1.xxxxxxxxx_2 \cdot 2^{\pm yyyy_2}$$

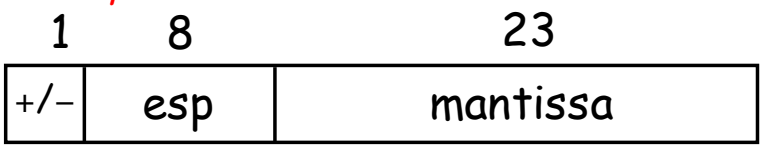
- Da notare che, di solito, la base b è implicita e quindi
  - È sufficiente memorizzare *segno*, *mantissa* (o significante) ed *esponente* (o caratteristica)
- Si usa un certo numero di bit (almeno 32)
  - Si riserva spazio per segno, mantissa ed esponente

Standard per la rappresentazione

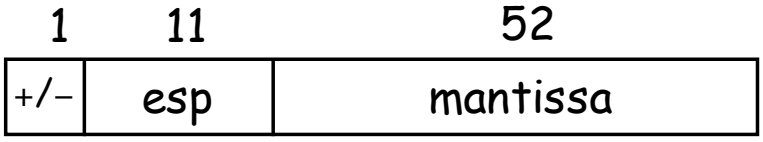
- Importanza di definire uno standard per la rappresentazione dei numeri in virgola mobile
  - Per definire la semantica delle istruzioni in virgola mobile
- International Standard Organization (ISO)
  - Vi appartengono circa 100 organizzazioni che si occupano di creare standard
- IEEE Computer Society (Institute of Electrical and Electronics Engineers) definisce lo "IEEE standard for binary floating arithmetic" (riferito come **IEEE 754**) nel 1985
  - Specifica il formato, le operazioni, le conversioni tra i diversi formati floating point e quelle tra i diversi sistemi di numerazione, la gestione delle eccezioni
- Nel 1989 IEEE 754 diventa uno standard internazionale (IEC 559)

Standard IEEE 754

- Formato non proprietario, ossia non dipendente dall'architettura del calcolatore
- Precisione *semplice* a 32 bit:



- Precisione *doppia* a 64 bit:



- Notazione in modulo e segno
- Alcune configurazioni dell'esponente sono riservate
- Segno (1 bit):
  - 0 per positivo, 1 per negativo

IEEE 754 a 32 bit: esponente

- Esponente (**8 bit**)
  - Rappresentato in **eccesso 127** (*polarizzazione* o *bias*)
  - L'intervallo di rappresentazione è [-127, 128]
  - Le due configurazioni estreme sono riservate, quindi
    $-126 \leq e \leq 127$
  - Se gli 8 bit dell'esponente contengono $10100011 = 163_{10}$
    - ✓ L' esponente vale 163-127=36
  - Se gli 8 bit dell'esponente contengono $00100111 = 39_{10}$
    - ✓ L' esponente vale 39-127=-88

- Perché la polarizzazione?
  - Il numero più grande che può essere rappresentato è 11…11
  - Il numero più piccolo che può essere rappresentato è 00…00
  - Quindi, quando si confrontano due interi polarizzati, per determinare il minore basta considerarli come interi senza segno

Richiamo su rappresentazione in eccesso 2^{n-1}

- Caso particolare della rappresentazione in eccesso k, con $k = 2^{n-1}$
  - Con n bit si rappresenta l'eccesso 2^{n-1}
  - Il numero m viene rappresentato in eccesso 2^{n-1} come $m + (2^{n-1}-1)$
- Intervallo di rappresentazione uguale a CP2: $[-2^{n-1}, 2^{n-1}-1]$
- Regola pratica
  - I numerali in eccesso 2^n-1 si ottengono da quelli in CP2 complementando il bit più significativo
- Esempio
  - n=4 bit: eccesso $2^{4-1}=8$, intervallo [-8, 7]
  - $(-3)_{10}$ -3+8=5 ➔ 0101
  - $(+4)_{10}$ +4+8=12➔ 1100

Numeri normalizzati

- Un numerale è in rappresentazione normalizzata quando **e≠00000000**
- In questa rappresentazione, la mantissa è normalizzata tra 1 e 2: **1 ≤ m < 2**
- Quindi, la mantissa è sempre nella forma:

  1.XXXXXXXXX…X
- Si usano tutti i 23 bit per rappresentare la sola parte frazionaria (1 prima della virgola è implicito)
- Gli intervalli di numeri rappresentati sono pertanto:

  $(-2^{128},-2^{-126}]$ $[2^{-126},2^{128})$
  - Gli estremi sono esclusi perché il massimo valore assoluto di m è molto vicino a 2, ma è comunque inferiore

Numeri denormalizzati

- Problema: come rappresentare un numero molto piccolo?
- Un numerale è in rappresentazione denormalizzata quando **e=00000000**
- L'esponente assume il valore *convenzionale* -126
- La mantissa è tra 0 e 1: **0 < m < 1**
- Quindi, la mantissa è sempre nella forma:

  0.XXXXXXXXX…X
- Si usano tutti i 23 bit per rappresentare la sola parte frazionaria
- La più piccola mantissa vale 2^{-23}
- Gli intervalli rappresentati sono:

  $(-2^{-126},-2^{-149}]$ $[2^{-149},2^{-126})$

Esempi: conversione da virgola mobile

- Quale numero in singola precisione rappresentano i seguenti 32 bit

  1 10000001 01000000000000000000000

  - ✓ Segno negativo (-)
  - ✓ Esponente e = $2^7 + 2^0 - 127 = 129 - 127 = 2$
  - ✓ Mantissa m = $1 + 2^{-2} = 1.25$
  - ✓ Quindi il numero rappresentato è $-1.25 \cdot 2^2 = -5$

  0 10000011 00110000000000000000000

  - ✓ Segno positivo (+)
  - ✓ Esponente e = $2^7 + 2^1 + 2^0 - 127 = 131 - 127 = 4$
  - ✓ Mantissa m = $1 + 2^{-1} + 2^{-4} + 2^{-5} = 1.59375$
  - ✓ Quindi il numero rappresentato è $1.59375 \cdot 2^4 = 25.5$

Esempi: conversione in virgola mobile

- Quale è la rappresentazione a singola precisione del numero

  8.5

  - ✓ Segno positivo (0)
  - ✓ 8.5 in binario è $1000.1 \cdot 2^0 = 1.0001 \cdot 2^3$
  - ✓ Esponente e: 3 + 127 = 130 = 10000010
  - ✓ Mantissa m: 00010000000000000000000
  - ✓ Quindi 0 10000010 00010000000000000000000

  -13.75

  - ✓ Segno negativo (1)
  - ✓ 13.75 in binario è $1101.11 \cdot 2^0 = 1.10111 \cdot 2^3$
  - ✓ Esponente e: 3 + 127 = 130 = 10000010
  - ✓ Mantissa m: 10111000000000000000000
  - ✓ Quindi 1 10000010 10111000000000000000000

Standard IEEE 754 a 32 bit: estremi degli intervalli

- Più grande normalizzato: $\sim \pm 2^{128}$

  X 11111110 11111111111111111111111

  $\pm$ 2^{127} ~ 2

- Più piccolo normalizzato: $\pm 2^{-126}$

  X 00000001 00000000000000000000000

  $\pm$ 2^{-126} 1

- Più grande denormalizzato: $\sim \pm 2^{-126}$

  X 00000000 11111111111111111111111

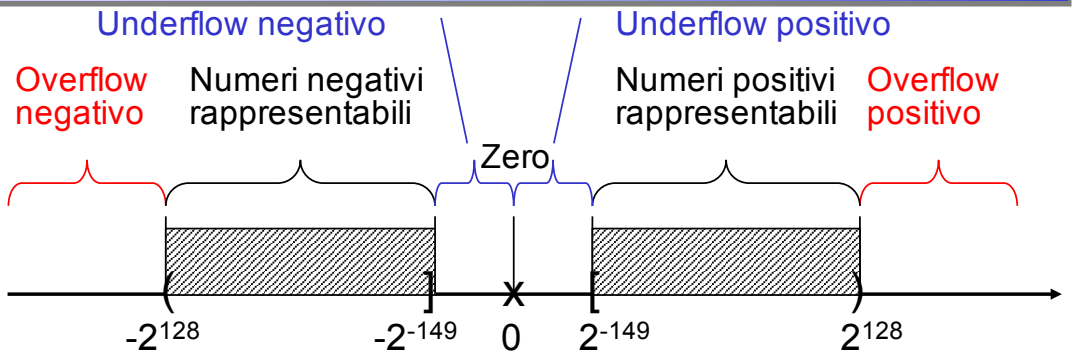
  $\pm$ 2^{-126} $(0.11\ldots1)_2 \approx 1$

- Più piccolo denormalizzato: $\pm 2^{-149}$

  X 00000000 00000000000000000000001

  $\pm$ 2^{-126} $(0.00\ldots01)_2 = 2^{-23}$

Intervallo di rappresentazione



- L'overflow può essere positivo
  - Quando si devono rappresentare numeri positivi maggiori di 2^{128}
- L'overflow può essere negativo
  - Quando si devono rappresentare numeri negativi minori di -2^{128}
- L'underflow può essere positivo
  - Quando si devono rappresentare numeri positivi minori di 2^{-149}
- L'underflow può essere negativo
  - Quando si devono rappresentare numeri negativi maggiori di -2^{-149}

Confronto tra numeri in virgola mobile

- Per stabilire quale di due numeri in virgola mobile sia il maggiore
- Se sono di segno *discorde*, allora il numero positivo è maggiore
- Se sono di segno *concorde*
  - Se sono *positivi*
    - ✓ Il numero con l'esponente *più grande* è il maggiore; a parità di esponente, il numero con mantissa *più grande* è maggiore
  - Se sono *negativi*
    - ✓ Il numero con l'esponente *più piccolo* è il maggiore; a parità di esponente, il numero con mantissa *più piccola* è maggiore

Confronto tra numeri in virgola mobile (2)

- Per due numeri positivi (negativi)
  - Siano a e b due numeri positivi (negativi) rappresentati in virgola mobile da $a_{31}a_{30}...a_0$ e $b_{31}b_{30}...b_0$
  - Notare che $a_{31} = b_{31} = 0$ (1)
- Per verificare quale dei due sia il maggiore, non occorre nessuna conversione
  - È sufficiente confrontarli come se fossero interi senza segno
  - Basta scorrere i bit, ed al primo bit diverso si individua il maggiore
    - ✓ Il numero con l'i-esimo bit a 1 (0)
    - ✓ Il numero con esponente/mantissa più grande (più piccolo) è il maggiore

Configurazioni particolari

- Lo standard IEEE 754 attribuisce valori convenzionali a particolari configurazioni di *e* ed *m*
  - *e* ed *m* tutti 0: rappresentano il valore 0 (*altrimenti non rappresentabile*)
  - *m* tutti 0 ed *e* tutti 1: rappresentano l'*infinito* ($\pm\infty$)
  - $m \neq 0$ ed *e* tutti 1: rappresentano la situazione *Not A Number (NAN)*, cioè un valore indefinito (ad es. il risultato di una divisione per 0 o la radice quadrata di un numero negativo)

Normalizzato	±	0<esp<Max	Qualsiasi stringa di bit
Denormalizzato	±	0	Qualsiasi stringa di bit diversa da zero
Zero	±	0	0
Infinito	±	111…1	0
NaN	±	111…1	Qualsiasi stringa di bit diversa da zero

Osservazioni sulla precisione singola

- In modo assolutamente indipendente dalla rappresentazione usata, con 32 bit è possibile rappresentare "soltanto" 2^{32} valori diversi
- I numeri rappresentati in virgola mobile hanno una densità maggiore vicino allo zero
- Diversi compromessi nella scelta del formato
  - Ad es. incrementando la dimensione dell'esponente
    - ✓ Si diminuisce la dimensione della mantissa
    - ✓ Si espande l'intervallo di rappresentazione (esponenti maggiori) ma si perdono cifre significative (precisione)

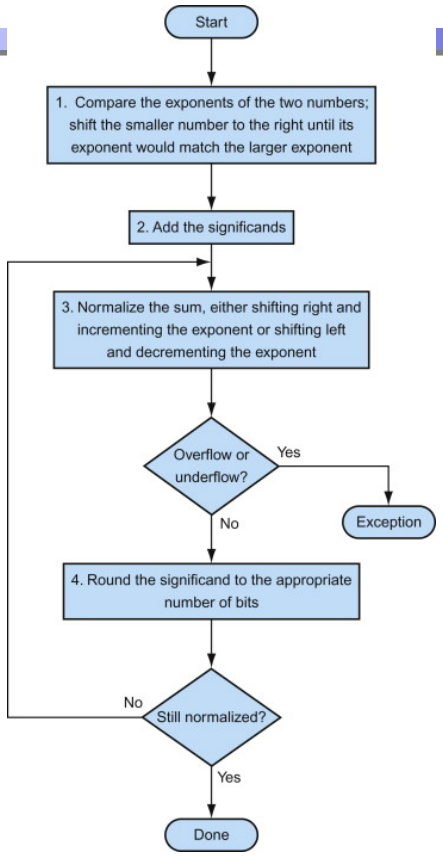
IEEE 754 a 64 bit

- Segno (1 bit)
- Esponente (11 bit)
  - Rappresentato in eccesso 1023
  - L'intervallo di rappresentazione è [-1023, 1024]
- Mantissa (52 bit)
  - Normalizzata come nella singola precisione
- Configurazione riservate come nella singola precisione per la rappresentazione di
  - 0
  - Numeri denormalizzati (positivi e negativi)
  - Infinito (positivo e negativo)
  - NaN (Not a Number)
- Esercizio
  - Qual è l'intervallo di rappresentazione dei numeri a doppia precisione?

Errore assoluto ed errore relativo

- Rappresentando un numero reale n nella notazione floating point si commette un errore di approssimazione
- In realtà viene rappresentato un numero razionale n' con un numero limitato di cifre significative
- *Errore assoluto*: $e_A = n-n'$
- *Errore relativo*: $e_R = e_A/n = (n-n')/n$
- Se la mantissa è normalizzata, l'errore relativo *massimo* è costante su tutto l'intervallo rappresentato, ed è pari ad un'unità sull'ultima cifra rappresentata
  - Esempio: 23 cifre frazionarie $e_R = 2^{-23}$
- Nella notazione non normalizzata, l'errore relativo massimo non è costante

Addizione in virgola mobile



Addizione in virgola mobile (2)

- Per addizionare e sottrarre occorre scalare le mantisse per eguagliare gli esponenti
- Esempio (*notazione IEEE 754*)

  $n_1 + n_2$

  n_1 : 0 10011001 00010111011100101100111

  n_2 : 0 10101010 11001100111000111000100

  $e_1 = (26)_{10}$, $e_2 = (43)_{10}$: Bit implicito della mantissa

  - occorre scalare m_1 di 17 posti

  n'_1 : 0 10101010 00000000000000001000101 +

  n_2 : 0 10101010 11001100111000111000100

  ---

  0 10101010 11001100111001000001001 Somma delle mantisse

  0 10101010 11001100111001000001010 Round

- Notare che l'addendo più piccolo perde cifre significative

Unità aritmetica per addizione in virgola mobile