Architettura degli Elaboratori

Rappresentazione in virgola mobile

Barbara Masucci
UNIVERSITA DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA
DIPARTIMENTO DI ECCELLENZA

Punto della situazione

Abbiamo visto le rappresentazioni dei numeri:

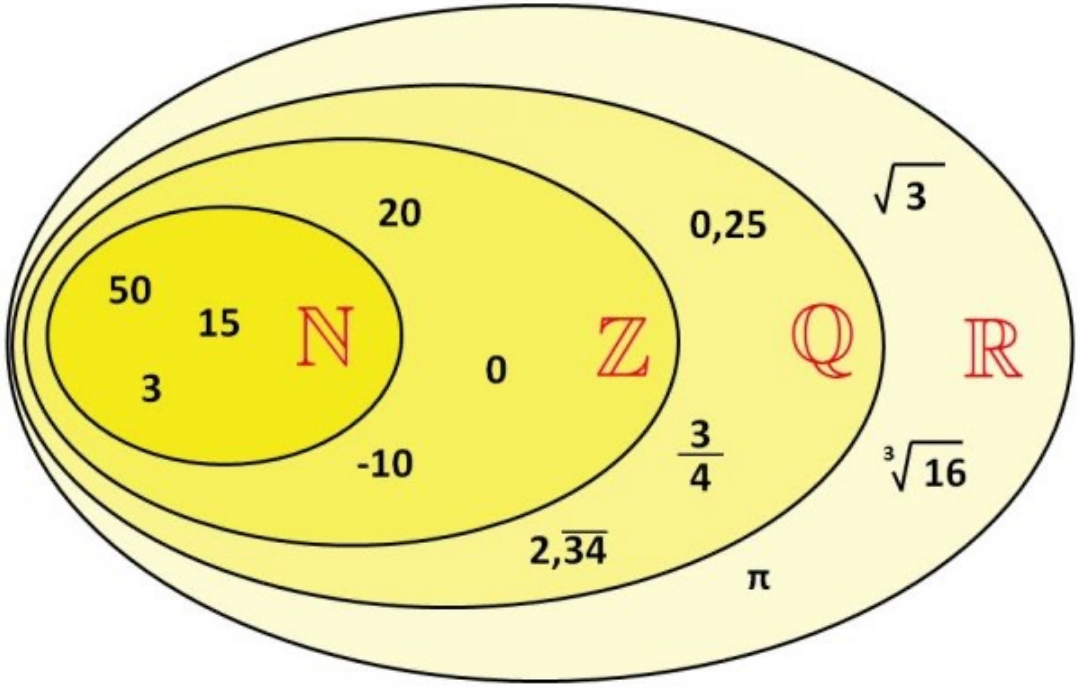
- Sistema posizionale pesato per
  - Interi positivi (nelle varie basi)
  - Numeri con la virgola positivi (frazioni proprie)
- Modulo e segno per interi col segno
- Complemento a 2 per interi col segno

Ora vedremo la rappresentazione in
virgola mobile per i numeri reali
(con virgola e segno)

Numeri Reali

L'insieme dei numeri reali (R) comprende

- I numeri naturali (N)
- I numeri interi (Z)
- I numeri razionali (Q)



Numeri Reali

- Alcuni esempi di numeri reali
  - $\pi = 3{,}14159265\ldots_{10}$
  - $e = 2{,}71828\ldots_{10}$ (numero di Nepero)
  - $0{,}000000001_{10}$ (numero di secondi in un nanosecondo)
  - 3155760000_{10} (numero di secondi in un secolo)
- Per numeri molto piccoli o molto grandi si utilizza la notazione scientifica in base 10
  - E' un modo conciso di esprimere i numeri, utilizzando potenze intere (positive o negative) di 10
  - Il numero viene scritto con una sola cifra prima della virgola
    - $1{,}0_{10} \times 10^{-9}$
    - $3{,}15576_{10} \times 10^{9}$

Notazione scientifica

- Vantaggi
  - Permette di rappresentare in maniera compatta numeri molto grandi o molto piccoli
  - Fornisce un'idea immediata sull'ordine di grandezza del numero

- Notazione scientifica normalizzata
  - Notazione scientifica in cui non ci sono zeri a sinistra della virgola
  - Esempi:
    - $1,0_{10} \times 10^{-9}$ è in notazione normalizzata
    - $0,1_{10} \times 10^{-8}$ non è in notazione normalizzata

Spostare la virgola

Il numero $12345,6789012345_{10}$ può essere scritto come:

$$12345,6789012345_{10} \times 10^0$$
$$1234,56789012345_{10} \times 10^1$$
$$123,456789012345_{10} \times 10^2$$
$$12,3456789012345_{10} \times 10^3$$
$$1,23456789012345_{10} \times 10^4$$

Spostare la virgola a sinistra di n cifre decimali
corrisponde ad incrementare l'esponente del 10 di n

Spostare la virgola

Il numero 12345,6789012345 può essere scritto come:

$$12345,6789012345_{10} \times 10^{0}$$
$$123456,789012345_{10} \times 10^{-1}$$
$$1234567,89012345_{10} \times 10^{-2}$$
$$12345678,9012345_{10} \times 10^{-3}$$
$$123456789,012345_{10} \times 10^{-4}$$

Spostare la virgola a destra di n cifre decimali
corrisponde ad decrementare l'esponente del 10 di n

Notazione scientifica

- Anche i numeri binari possono essere espressi in notazione scientifica normalizzata

- Esempio:
  - $1{,}0_2 \times 2^{-1}$ è in notazione normalizzata
  - $0{,}1_2 \times 2^{2}$ non è in notazione normalizzata

- Forma generale di un numero binario in notazione scientifica normalizzata

$$1{,}xxx_2 \times 2^{yyy}$$

Spostare la virgola

Il numero $10111,1101011101_2$ può essere scritto come:

$$10111,1101011101_2 \times 2^0$$
$$1011,11101011101_2 \times 2^1$$
$$101,111101011101_2 \times 2^2$$
$$10,1111101011101_2 \times 2^3$$
$$1,01111101011101_2 \times 2^4$$

Spostare la virgola a sinistra di n cifre binarie
corrisponde ad incrementare l'esponente del 2 di n

Spostare la virgola

Il numero $10111,1101011101_2$ può essere scritto come:

$$10111,1101011101_2 \times 2^0$$
$$101111,101011101_2 \times 2^{-1}$$
$$1011111,01011101_2 \times 2^{-2}$$
$$10111110,1011101_2 \times 2^{-3}$$
$$101111101,011101_2 \times 2^{-4}$$

Spostare la virgola a destra di n cifre binarie
corrisponde ad decrementare l'esponente del 2 di n

Notazione a virgola fissa

- La rappresentazione vista la volta scorsa per i numeri con la virgola in binario viene detta "a virgola fissa"
  - Utilizza un numero finito (n+s) di cifre, di cui
    - n dedicate alla parte intera
    - s dedicate alla parte frazionaria
  - Fornisce una *rappresentazione approssimata* del numero dato
    - Ha dei limiti quando si devono rappresentare numeri molto grandi o molto piccoli

Rappresentazione FP

- La rappresentazione in virgola mobile, anche detta FP (Floating Point)
  - Estende l'intervallo dei numeri rappresentabili a parità di cifre
  - Permette di rappresentare in maniera compatta numeri molto grandi, ma anche molto piccoli (sia positivi che negativi)
  - Utilizza la notazione scientifica normalizzata, specificando
    - Segno (+,-)
    - Mantissa (parte frazionaria)
    - Esponente (intero con segno)

Rappresentazione FP

- I numeri binari in forma normalizzata sono rappresentati da una tripla <s, M, E>, dove
  - s = segno (0 per i positivi, 1 per i negativi)
  - M = mantissa
  - E = esponente

- Il numero corrispondente è

$$N = (-1)^s \times (1+M) \times 2^E$$

Rappresentazione FP

Fissato il numero totale di bit per la rappresentazione, bisogna stabilire:

- Quanti bit assegnare per la mantissa M?
  - Maggiore è il numero di bit, maggiore è la precisione
- Quanti bit assegnare per l'esponente E?
  - Maggiore è il numero di bit, più ampio è l'intervallo di rappresentabilità

Standard IEEE 754

- E' importante definire uno standard per la rappresentazione dei numeri FP
- Lo standard IEEE 754 (IEEE standard for binary floating arithmetic), proposto nel 1985
  - Specifica il formato, le operazioni aritmetiche e di confronto per numeri FP
  - E' utilizzato dal MIPS che studieremo in seguito
  - Propone due formati:
    - Precisione Singola (32 bit: 1 parola macchina)
    - Precisione Doppia (64 bit: 2 parole macchina)

Standard IEEE 754
(precisione singola)

Suddivide i 32 bit in:

- 1 bit per il segno s
- 8 bit per rappresentare l'esponente E
- 23 bit per rappresentare la mantissa M

Il numero rappresentato è

$$N = (-1)^s \times (1+M) \times 2^E$$

Standard IEEE 754
(precisione singola)

- Il *segno* è dato da $(-1)^s$
- Analogamente alla rappresentazione in complemento a 2:
  - Il bit $s=0$ rappresenta numeri positivi
  - Il bit $s=1$ rappresenta numeri negativi

Standard IEEE 754
(precisione singola)

- Come usare gli 8 bit per rappresentare esponenti negativi e positivi?
- Se usiamo la rappresentazione in complemento a 2, l'intervallo è [-128,+127]
  - Il confronto tra numeri però non risulta naturale
  - Quindi questa rappresentazione non viene utilizzata

Standard IEEE 754 (precisione singola)

- Come usare gli 8 bit per rappresentare esponenti negativi e positivi?
- La scelta dello standard è quella di usare la rappresentazione in binario puro
  - L'intervallo rappresentabile è [0,+255], di cui
    - 0=00000000 è riservato allo zero
    - 255=11111111 è riservato a ∞ e NaN (Not a Number), usato per definire il risultato di operazioni non valide
  - I numeri nell'intervallo restante ([1,+254]) sono messi in corrispondenza con i 254 appartenenti all'intervallo [-126,127]
    - Questa corrispondenza viene definita "polarizzazione"

Standard IEEE 754
(precisione singola)

- L'esponente E sarà ottenuto come E=e-127, dove e è il contenuto del campo esponente
  - 00000000 = 0 RISERVATO (per lo 0)
  - 00000001 = 1 rappresenta 1-127 =-126
  - 00000010 = 2 rappresenta 2-127 =-125
  - 00000011 = 3 rappresenta 3-127=-124
  - .....
  - 11111101 = +253 rappresenta 253-127=+126
  - 11111110 = +254 rappresenta 254-127=+127
  - 11111111 = +255 RISERVATO (ad infinito e NaN)

Standard IEEE 754
(precisione singola)

- Il valore di un numero in notazione polarizzata è quindi

$$(-1)^{s} \times (1+M) \times 2^{(e-127)}$$

Standard IEEE 754
Esempio

- Scrivere in notazione FP IEEE 754 (precisione singola) il numero $-0,25_{10}$
  - Innanzitutto calcoliamo la frazione binaria equivalente, senza considerare il segno: $0,25_{10} = 0,01_2$
  - Poi la esprimiamo in notazione scientifica normalizzata: $0,01_2 = 1,0_2 \times 2^{-2}$
  - Aggiungiamo il segno ed esplicitiamo la mantissa: $(-1)^1 \times (1+0,0) \times 2^{-2}$
  - Infine, l'esponente polarizzato $-2 = e-127 \rightarrow e=125$
  - Quindi $s = 1$, $e = 125 = 01111101_2$, $M = 00\ldots00_2$

1	01111101	00000000000000000000000

Standard IEEE 754
Esempio

- Quale numero decimale è rappresentato dalla seguente sequenza di bit nello standard IEEE 754 (precisione singola)?

1	00001011	01000000000000000000000

- $s = 1$,
- $e = 00001011_2 = 11_{10}$,
- $M = 0{,}01_2 = 0{,}25_{10}$
- Il numero rappresentato è

$$(-1)^1 \times (1+0{,}25) \times 2^{(11-127)} = -1{,}25_{10} \times 2^{(-116)}$$

Standard IEEE 754
(precisione singola)

- L'intervallo di rappresentabilità con 32 bit è:
  - Numeri positivi:
    - minimo $\approx 2_{10} \times 10^{-38}$
    - Massimo $\approx 2_{10} \times 10^{+38}$
  - Numeri negativi:
    - minimo $\approx -2_{10} \times 10^{-38}$
    - Massimo $\approx -2_{10} \times 10^{+38}$

Standard IEEE 754
(precisione doppia)

Suddivide i 64 bit in:

- 1 bit per il segno s
- 11 bit per rappresentare l'esponente E
- 52 bit per rappresentare la mantissa M

Il numero rappresentato è

$$N = (-1)^s \times (1+M) \times 2^E$$

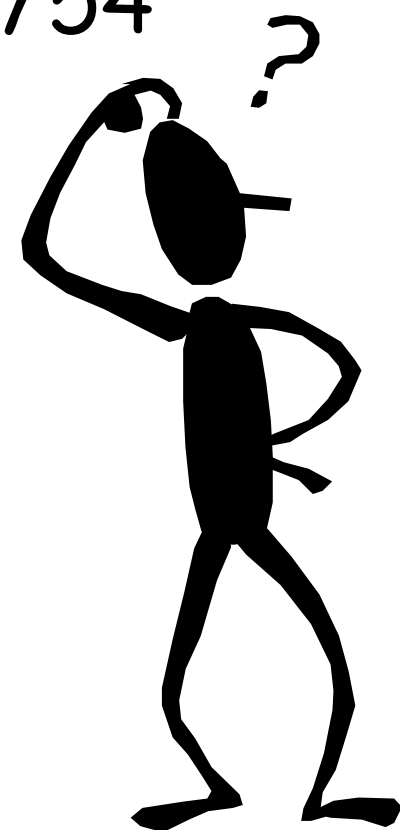
Standard IEEE 754
(precisione doppia)

- L'intervallo di rappresentabilità con 64 bit è:
  - Numeri positivi:
    - minimo $\approx 2_{10} \times 10^{-308}$
    - Massimo $\approx 2_{10} \times 10^{+308}$
  - Numeri negativi:
    - minimo $\approx -2_{10} \times 10^{-308}$
    - Massimo $\approx -2_{10} \times 10^{+308}$

Rispetto alla precisione singola, l'intervallo è molto più ampio, e ciascun numero viene rappresentato con una maggiore accuratezza

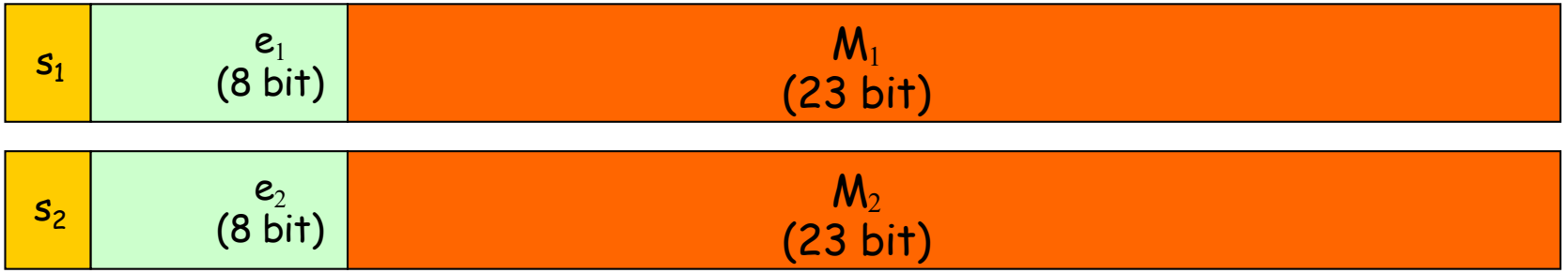
IEEE 754:
Confronto tra due numeri

Come effettuare il confronto tra due numeri in formato IEEE 754



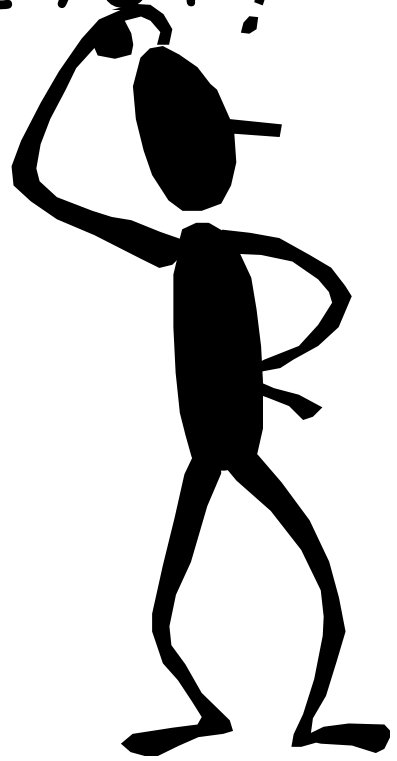
IEEE 754:
Confronto tra due numeri

- Il confronto tra due numeri in formato IEEE 754 è un'operazione facilitata dal formato
  - Innanzitutto si confrontano i segni s_1 ed s_2 dei due numeri
  - Poi si confrontano gli esponenti e_1 ed e_2
    - Numeri con esponenti più grandi sono più grandi, indipendentemente dalle mantisse
  - Infine si confrontano le mantisse M_1 ed M_2



IEEE 754:
Somma di due numeri

Come effettuare la somma di due numeri in formato IEEE 754 ?



IEEE 754: Somma di due numeri

- Per effettuare la somma di due numeri
  - Bisogna innanzitutto che i loro esponenti abbiano lo stesso ordine di grandezza
    - Se così non è, per il numero con esponente più piccolo si sposta la virgola a sinistra del numero di posizioni necessario (allineamento delle mantisse)
  - Poi si sommano i valori ottenuti
  - Si normalizza il risultato
  - Si arrotonda al numero di bit richiesto per la mantissa (precisione)
    - Ed eventualmente si normalizza di nuovo

Esempio 1

- Calcoliamo $5_{10} + 3{,}625_{10}$ (con precisione a 4 bit)
  - $5_{10} = 101_2 = 1{,}01_2 \times 2^2$
  - $3{,}625_{10} = 11{,}101_2 = 1{,}1101_2 \times 2^1$
  - Hanno ordini di grandezza differenti!
    - Spostiamo la virgola a sinistra di un posto per il numero con esponente più piccolo: $3{,}625_{10} = 1{,}1101_2 \times 2^1 = 0{,}11101_2 \times 2^2$
  - Effettuiamo la somma dei valori ottenuti:

$$\begin{array}{r} 1{,}01000 + \\ \underline{0{,}11101 =} \\ 10{,}00101 \times 2^2 \end{array}$$

  - Normalizziamo il risultato: $1{,}000101 \times 2^3$
  - Arrotondiamo la mantissa al numero di bit richiesto:

$$1{,}0001 \times 2^3$$

Esempio 1 (cont.)

- Il risultato ottenuto per $5_{10} + 3{,}625_{10}$ è

  $1{,}0001 \times 2^3$

  che corrisponde a $1000{,}1_2 = 8{,}5_{10}$

- Facendo la somma in decimale si ha che

  $5_{10} + 3{,}625_{10} = 8{,}625_{10}$

  che corrisponde a $1000{,}101_2$

Quindi l'aver utilizzato solo 4 bit per la mantissa ha comportato un errore di arrotondamento

Esempio 2

- Calcoliamo $-0{,}4375_{10} + 0{,}5_{10}$ (con precisione a 4 bit)
  - $0{,}4375_{10} = 0{,}0111_2 = 1{,}110_2 \times 2^{-2}$
  - $0{,}5_{10} = 0{,}1_2 = 1{,}000_2 \times 2^{-1}$
  - Hanno ordini di grandezza differenti!
    - Effettuiamo lo spostamento della virgola a sinistra di un posto per il numero con esponente più piccolo:
      $0{,}4375_{10} = 1{,}110_2 \times 2^{-2} = 0{,}111_2 \times 2^{-1}$
  - Effettuiamo la somma dei valori ottenuti:

    1,000−
    0,111=
    0,001 x 2^{-1}

    Si tratta di una sottrazione in binario perchè il secondo valore è negativo
  - Normalizziamo il risultato: $1{,}000 \times 2^{-4}$
  - Il numero di bit della mantissa è già quello richiesto

Esempio 2 (cont.)

- Il risultato ottenuto per $-0,4375_{10} + 0,5_{10}$ è
  $1,000 \times 2^{-4}$
  che corrisponde a $0,0001_2 = 0,0625_{10}$

- Facendo la somma in decimale si ha che
  $-0,4375_{10} + 0,5_{10} = 0,0625_{10}$

I due valori coincidono perché non ci sono state modifiche dovute all'arrotondamento

Per concludere

- Non studieremo moltiplicazione e divisione di numeri FP
- Le istruzioni MIPS che studieremo tratteranno numeri rappresentati in complemento a 2
  - Per esempio: `add`, `sub`,…
- Il MIPS supporta anche il formato IEEE 754 a singola (e doppia) precisione con istruzioni particolari:
  - Per esempio: `add.s`, `sub.s`,…

Riepilogo e riferimenti

- I numeri in virgola mobile e lo standard IEEE 754 a singola e doppia precisione
  - [PH] par. 3.5
- Somma in virgola mobile
  - [PH] par. 3.5 (escluso 'La moltiplicazione in virgola mobile')