Lezione 4

Rappresentazione in virgola mobile

Ricordiamo che si rappresenta il numero con la virgola con la potenza di 10

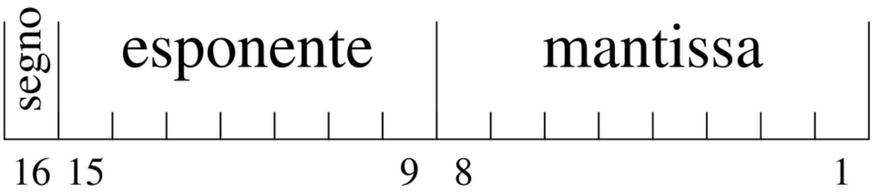
es. 287,452 → 0,287452 * 10^3

Mentre per lo standard IEEE 754 si adegua avendo un numero davanati alla virgola e adeguando la potenza

es. 287,452 → 2,87452 * 10^2

Con la virgola mobile abbiamo bisogno di:

- Segno = {0 → positivo, 1 → negativo}
- Esponente
- Mantissa (parte dopo la virgola)



Operazioni con numeri con la virgola

> ❗ La normalizzazione avviene solo nel momento in cui il risultato dell'operazione ha mantenuto le sue posizioni o se il risultato del primo bit significativo sia 0

Format = < segno ; esponente ; mantissa >

Quando

Somma e sottrazione

Per fare le operazioni bisogna:

1. Portare allo stesso esponente (di solito al più grande)
2. Eseguire l'operazione dopo aver stabilito l'ordine degli operandi valutando magnitudo (grandezza) e segno.
3. Decidere il segno del risultato
4. Normalizzazione del risultato
    a. Scorrimento mantissa
    b. Adeguamento dell'esponente

Moltiplicazione e divisione

1. Segno → {+ = concordi, - = discordi}

2. Esponente somma o differenza
3. Mantissa eseguento l'operazione
4. Normalizzazione

Esempio Moltiplicazione

base = 10

a = < 0 , 2 , 327 >

b = < 1 , 3, 294 >

1) segno → 1 quindi negativo

2) esponente = 2 + 3 = 5

3) 327 * 294 = 96138

> ⚠️ Quando vediamo i valori dobbiamo vederli come in questo caso:
> 327 = 0,327 * 10^5
> 249 = 0,249 * 10^5
> Per questo quando andiamo a normalizzare viene 0,096138, perchè shiftiamo a sinistra e decrementiamo di 1 l'esponente, in questo caso normalizzeremo il risultato perchè non ha mantenuto le sue posizioni, 5 cifre invece che 3.

4) normalizzazione → 0,096138

- scorrimento a sinistra della mantissa di 1 posizione
- decremento di 1 dell'esponente

R) < 1 ; 4 ; 961~~38~~ (rispetto le cifre della forma iniziale)>

> L'overflow in virgola mobile è dato dalla non rappresentabilità dell'esponente

Esempio sottrazione

base = 4

a = < 0 ; 2 ; 3312 >

b = < 0 ; 3 ; 2123 >

1) Esponenti uguali:

(se scorri verso destra aumenti l'esponente e viceversa)

a = < 0 ; 3 ; 0331 >

2) A - B

Però A < B quindi → B - A e segno meno al risultato

$2123_4 - 0331_4 = 1132_4$

R) < 1 ; 3 ; 1132 >

Standard IEEE 754

Standard utilizzato per rappresentare i numeri in virgola mobile che può essere adottata in qualunche calcolatore

Lo standard definisce come sono suddivisi i bit (esponente e mantissa) in base ai bit massimi:

format = | bit segno | bit esponente | bit mantissa |

- 16 BIT (Half Precision) | 1 | 5 | 10 |
- 32 BIT (Single Precision) | 4 | 8 | 23 |
- 64 BIT (Double Precision) | 1 | 11 | 52 |

Esponente (con e bit)

Consideriamo CA2 il range è $[-2^{e-1}; 2^{e-1} - 1]$

- Elimino i due valori più piccoli → $[-2^{e-1} + 2; 2^{e-1} - 1]$
- Sommo il BIAS $2^{e-1} - 1$ → $[1, 2^e - 2]$

Esempio esponente

e = 5 → [-16,15]

Elimino i due valori più piccoli → [-14,15]

Sommo il BIAS $2^{e-1} - 1 = 2^4 - 1 = 15$ → [1,30]

> ⚠️ Facendo ciò si lasciano fuori il numero 0 e il 31 ovvero → 00000 e 11111

In questo caso il primo valore 1 è 00001 e il 30 è 11110

Queste rappresentazioni/configurazioni sono lasciate fuori appositamente perchè serviranno a rendere il circuito più veloce ed efficiente data la loro facile rappresentabilità (tutti zero o tutti uno).

> 💡 Il BIAS serve proprio per fare questo slittamento e rendere più facile il confronto tra due valori, lascio da parte il segno avendo così ho un valore assoluto e posso controllare la magnitudo confrontando questa stringa composta da esponente e mantissa ed ottenere il confronto.

Mantissa

La mantissa può essere presa come la parte dopo la virgola considerando 1 prima della virgola

In binario → 1, ... * 2^e

La mantissa nello Standard IEEE 754 è **questa** ovvero la parte dopo la virgola

> 💡 Il numero 1 davanti la virgola si lascia implicito cosi che possiamo risparmiare un bit sapendo che quello sarà sempre 1.

Esempio mantissa ed esponente

m = -14,25

1) Per prima cosa converto in base 2 → -1110,01

2) Portiamo in standard 754 → -1,11001 * 2^3

Half precision

N = 16 BIT (Half Precision) | 1 | 5 | 10 |

3) Sommiamo il BIAS all'esponente → 3 + 15 = 18 che si rappresenta come 10010

R) In questo momento abbiamo tutto per rappresentare il nostro numero con la virgola

segno = 1 | esponente = 18 | mantissa 11011

→ < 1 ; 10010 ; 1100100000 > (I zero aggiunti al valore non alterano la mantissa, li portiamo fino ad arrivare a 10 bit, ovvero lo standard 16 bit half precision che ha 10 bit disponibili per la mantissa.)

Single Precision

Principalmente la mantisse rimane la stessa bisognerà aggiungere solo tanti zeri finali

Mentre per l'esponente sarà diverso perchè il bias cambierà

Nella Single Precision abbiamo 8 bit disponbili quindi il nostro BIAS sara:

→ $2^{8-1} - 1 = 127 + 3 = 130$

$130_{10} -> 10000010_2$

e quindi sarà < 1 ; 10000010 ; 11001 + 18 zeri>

Rappresentazione esadecimale dell'IEEE 754

semplicemente prendere i blocchi da 4 e trasformarli in esadecimale:

< 1 ; 10010 ; 1100100000 > → CB20

C B 2 0

Tabella rappresentazione (Standard IEEE 754)

Questa tabella riassume tutti le possibili rappresentazioni (valori)

Tipo	Esponente	Column
Zeri	0	0
Numeri denormalizzati	0	≠ 0
Normali	$[1, 2^e - 2]$	Qualunque
Infiniti	$2^e - 1$	0
NaN (Not a number)	$2^e - 1$	≠ 0

> NaN = Non corretti o in overflow, i casi che non rientrano sopra

La tabella va letta in questo modo:

- Se esponente = 0 → Mantissa può valere 0 o ≠ 0
- Se esponente è compreso tra 1 e $2^e - 2$ → Mantissa può valere qualunque valore
- Se esponente è il massimo valore ovvero $2^e - 1$ → Ho la rappresentazione dei due infiniti se la mantissa è 0 o mantissa ≠ 0 se ho NaN (derivato come scritto da errori di calcolo non rappresentabili, errori, etc...)

Esempio Sottrazione IEEE 754 (Half Precision) Completo

Somma tra A = 26,42 e B = -37,68 = D0B5

$26_{10} = 11010_2$

A)

$0,42_{10} = 011010$

0,42 * 2 = 0,84

0,84 * 2 = 1,68

0,68 * 2 = 1,36

0,36 * 2 = 0,72

0,72 * 2 = 1,44

0,44 * 2 = 0,88 (approssimato)

$A_2 = 11010,011010$ → 1,1010011010 * 2^4

Esponente = 4 + 15 (BIAS) = 19 → 10011_2

Risultato A) < 0 ; 10011 ; 1010011010 >

B)

Partiamo da un valore esadecimale, basta trasformarlo e spacchettarlo in bit (secondo la half precision).

D0B5 → 1101 0000 1011 0101

Secondo lo standard quindi diventa → < 1,10100 , 0010110101 >

Risultato B) < 1,10100 , 0010110101 >

Verifichiamo:

1. Segno = 1 → quindi negativo → -37,68
2. Esponente = 10100 = 20
    a. 37 = 100101 che se portiamo in standard ieee 754 diventa 1,00101 * 10^5
    b. se aggiungiamo il BIAS (15) diventa 5 + 15 = 20 quindi anche esponente verifcato
    c. in fine se verifichiamo la mantissa → 0,68 = 0,10101 e se lo aggiungiamo a 37 diventa 0010110101 che è uguale a quello della mantissa nella tripla

Sottrazione

Esponente di A = 10011 = 19

Esponente di B = 10100 = 20

→ Quindi bisogna portare A ad esponente 20.

Porto A esponente 20) < 0 ; 10100; 1101001101~~0~~ >

(1 implicito è stato esplicitato)

quindi aumentando di esponente aggiungo 1 e rimuovo il bit meno significativo

Dato che | B | > | A | → B - A e segno negativo

Sottrazione tra B e A)

1,0010110101

-0,1101001101

0,0101101000

Normalizziamo portando il primo numero prima della virgola a 1)

Per farlo bisogna shiftare di due posizioni e diminuire la potenza di 2.

0,0101101000 → 1,0110100000 * 10^{18}

Risultato) < 1 ; 10010 ; 0110100000 >

Esadeciamale) C9A0