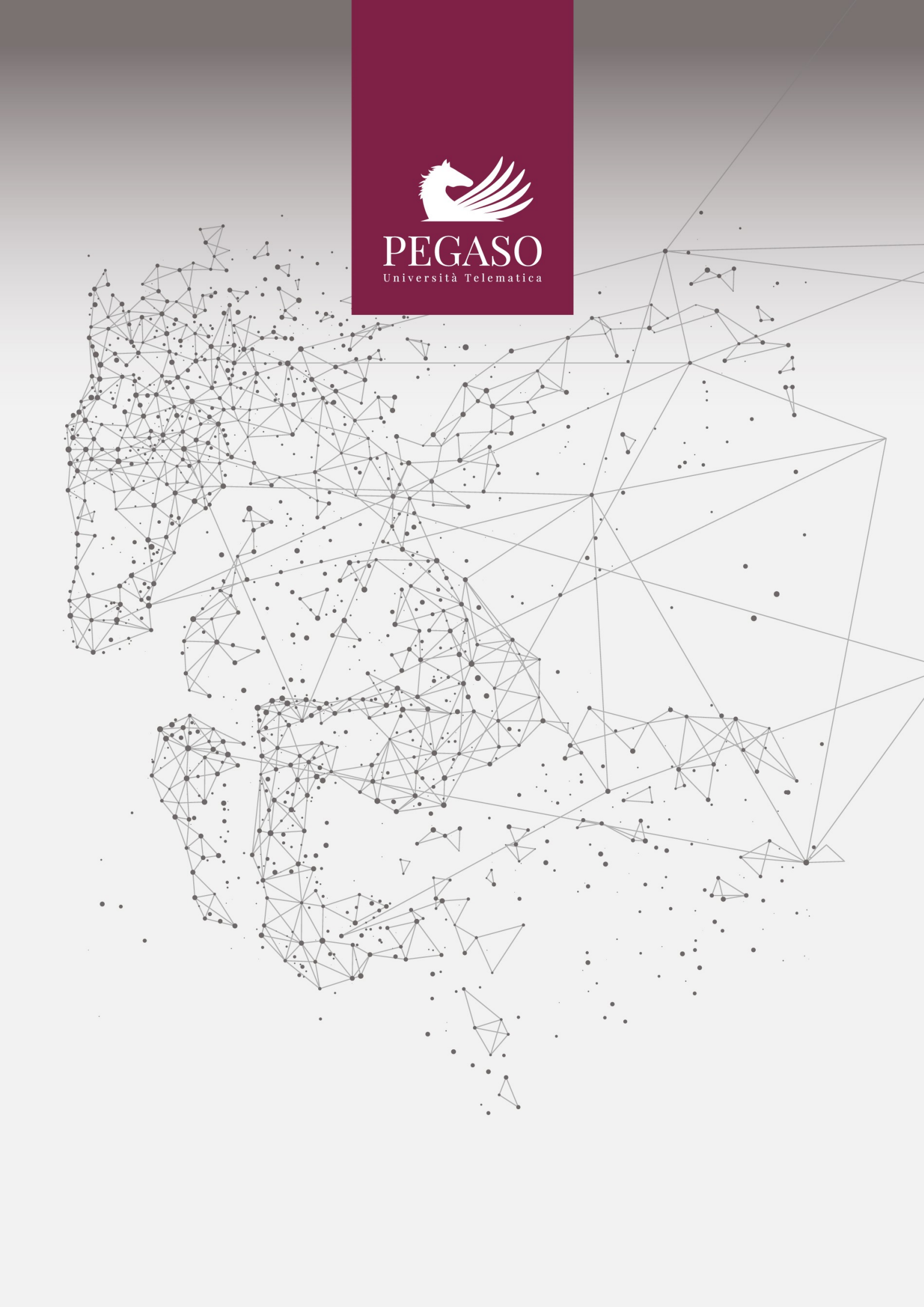

PEGASO
Università Telematica

Indice

1. Lo standard 754 IEEE

Lo standard IEEE 754, introdotto nel 1985, ha rivoluzionato la rappresentazione dei numeri reali nei calcolatori, fornendo un formato uniforme per la virgola mobile. Prima di questo standard, i produttori utilizzavano implementazioni proprietarie che portavano a incoerenze nei calcoli.

Lo standard definisce diverse precisioni:

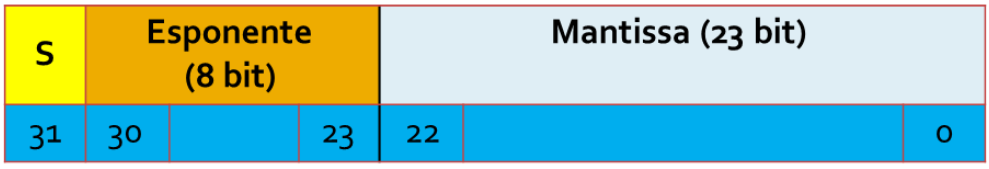
- singola precisione o precisione semplice (32 bit)
- doppia precisione (64 bit)
- precisione estesa (80 bit)

Prima dell'introduzione dello standard IEEE 754, le implementazioni della virgola mobile variavano tra i produttori, portando a risultati incoerenti nei calcoli.

Lo standard ha risolto questo problema, fornendo:

- Un formato comune per rappresentare numeri in virgola mobile.
- Requisiti per la gestione delle operazioni matematiche, inclusi casi speciali come divisioni per zero o numeri non rappresentabili.

Singola precisione (32 bit):



$$(-1)^{b_{31}} \times 2^{b_{30}b_{29}\ldots b_{24}b_{23}} \times 1.\, b_{22}b_{21}\ldots b_1 b_0$$

Figura 1 – IEEE 754 a singola precisione (32 bit).

- Numero di bit per il segno: **1**
- Numero di bit per l'esponente: **8**
- Rappresentazione esponente: **Base 2 per eccesso 127**
- Valori esponente: **[-126, 127]**
    - le parole codice 0_{10} (-127_{10}) e 255_{10} (128_{10}) sono riservate per funzioni speciali
- Numero di bit per la mantissa: **23**
- Rappresentazione (mantissa-1): **Binaria**
- Cifre decimali mantissa: **≅7 (23 / 3.3**)

I valori assunti dall'esponente ***e*** e dalla mantissa ***m*** determinano l'appartenenza del numero ad una di queste categorie:

- zeri
- numeri in forma normale
- numeri in forma denormalizzata
- infiniti
- NaN (not a number)

Categoria	Esp.	Mantissa
Zeri	0	0
Numeri denormalizzati	0	non zero
Numeri normalizzati	1-254	qualunque
Infiniti	255	0
NaN (Not a Number)	255	non zero

Quindi l'esponente distingue i numeri in modo primario, mentre la mantissa in modo secondario.

Doppia precisione (64 bit):

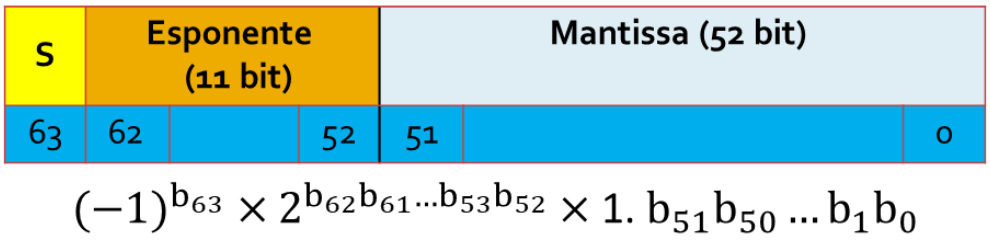


Figura 2 – IEEE 754 a doppia precisione (64 bit).

- Numero di bit per il segno: **1**
- Numero di bit per l'esponente: **11**
- Rappresentazione esponente: **Base 2 per eccesso 1023**
- Valori esponente: **[-1022, 1023]**
    - le parole codice 0_{10} (-1023_{10}) e 255_{10} (1024_{10}) sono riservate per funzioni speciali
- Numero di bit per la mantissa: **52**
- Rappresentazione (mantissa-1): **Binaria**
- Cifre decimali mantissa: **≅15 (52 / 3.3**)

2. Da virgola mobile a decimale mobile

Dato un numero in virgola mobile (standard IEEE 754 a singola precisione), si vuol capire come convertirlo in formato decimale.

Ad esempio, consideriamo la seguente stringa di bit che rappresenta un numero in formato virgola mobile:

1 10000001 01000000000000000000000

L'obiettivo è rispondere al seguente quesito:

$(-1)^s * M * 2^E = ?$

Innanzitutto, il primo bit (in verde) definisce il segno del numero:

1 10000001 01000000000000000000000

s = 1 → segno negativo

$(-1)^s * M * 2^E = -$ .....

I bit in rosso rappresentano la mantissa:

1 10000001 01000000000000000000000

Per calcolare la mantissa (in rosso), bisogna aggiungere il bit implicito (1.) e i restanti bit, essendo che dopo dopo "01" son tutti zeri, quest'ultimi si omettono.

$M = (1.01)_2 = (1*2^0+0*2^{-1}+1*2^{-2}) = 1,25$

$(-1)^s * M * 2^E = -1{,}25$ …

Per calcolare esponente (in blu)

1 10000001 01000000000000000000000

$E = (10000001)_{2 \text{ eccesso } \mathbf{127}} = (1*2^7+1*2^0) - \mathbf{127} = 2$

N.B. per definizione l'esponente è in eccesso 127, quindi dopo aver convertito l'esponente bisogna sottrarre 127.

$(-1)^s * M * 2^E = -1{,}25 \times 2^2$

Riepilogando:

1 10000001 01000000000000000000000

s = 1 → segno negativo

$M = (1.01)_2 = (1*2^0+0*2^{-1}+1*2^{-2}) = 1{,}25$

$E = (10000001)_{2 \text{ eccesso } \mathbf{127}} = (1*2^7+1*2^0) - \mathbf{127} = 2$

Il risultato finale è:

$(-1)^s * M * 2^E = -1{,}25 \times 2^2 = \mathbf{-5}$

Si consideri un'altra stringa bit che rappresenta un numero in formato virgola mobile:

0 10000011 10011000000000000000000

Ragionando come l'esempio precedente:

0 10000011 10011000000000000000000

s = 0 → segno positivo

M = $(1.10011)_2 = (2^0+2^{-1}+2^{-4}+2^{-5}) = 1{,}59375$

E = $(10000011)_{2\ \text{eccesso}\ \mathbf{127}} = (2^7+2^1+2^0) - \mathbf{127} = 4$

$(-1)^s * M * 2^E = +1{,}59375 \times 2^4 = \mathbf{25{,}5}$

3. Da decimale a virgola mobile

Invece, per determinare la rappresentazione in virgola mobile a singola precisione del numero reale, bisogna compiere i seguenti passi.

Consideriamo il numero decimale 8,5 , da convertire in virgola mobile a singola precisione (32 bit).

1. Convertire da decimale in binario 8:

$(8)_{10} = (1000)_2$

2. Convertire da decimale in binario 0,5:

$(0.5)_{10} = (0.1)_2$

3. Somma dei risultati 1 e 2:

$(8,5)_{10} = (1000.1)_2$

$(8,5)_{10} = (1000.1)_2 = 1.0001 * 2^3 = (-1)^s * M * 2^E$

- segno positivo → $s = 0$ → $b_{31} = 0$
  - essendo 8,5 positivo

- esponente: 3
  - calcolato spostando la virgola fino al bit più significato (1.) della rappresentazione binaria del numero decimale di partenza; in questo caso spostato di 3 posizioni verso sinistra.

- Mantissa (M) = $(1.0001)_2$ → $b_{22}b_{21}...b_1b_0 = 0001000...000$
  - si omette 1. («bit implicito»)

Riguardo la codifica dell'esponente, si ricorda che la rappresentazione dell'esponente è in **Base 2 per eccesso 127 (IEEE 754),** quindi:

- Aggiungere l'eccesso all'esponente: $(3)_{10} + 127 = (130)_{10Ecc127}$
- Conversione $(130)_{10Ecc127}$ in binario: $(10000010)_{2Ecc127} \rightarrow b_{30}b_{29}...b_{24}b_{23}$

Quindi, ricapitolando:

- segno positivo → $s = 0 \rightarrow b_{31} = 0$
- Esponente:$(10000010)_{2Ecc127} \rightarrow b_{30}b_{29}...b_{24}b_{23} = 10000010$
- Mantissa (M) = $(1.0001)_2 \rightarrow b_{22}b_{21}...b_1b_0 = 0001000...000$
- $(8,5)_{10} = 0\ 10000010\ 00010000000000000000000$

Come ulteriore esempio, si vuole determinare la rappresentazione in virgola mobile a singola precisione del numero reale -67,25.

1. Convertire da decimale in binario 67:

$(67)_{10} = (1000011)_2$

2. Convertire da decimale in binario 0,25:

$(0.25)_{10} = (0.01)_2$

3. Somma dei risultati 1 e 2:

$(67,5)_{10} = (1000011.01)_2$

$(-67,25)_{10} = (1000011.01)_2 = 1.00001101 * 2^6 = (-1)^s * M * 2^E$

- segno negativo → $s = 1 \rightarrow b_{31} = 1$
  - essendo 67,5 negativo

- esponente: 6
    - calcolato spostando la virgola fino al bit più significato (1.) della rappresentazione binaria del numero decimale di partenza; in questo caso spostato di 6 posizioni verso sinistra.

- mantissa (M) = $(1.00001101)_2 \rightarrow b_{22}b_{21}...b_1b_0 = 00001101...000$
    - si omette 1. («bit implicito»)

Riguardo la codifica dell'esponente, si ricorda che la rappresentazione dell'esponente è in **Base 2 per eccesso 127 (IEEE 754),** quindi:

- Aggiungere l'eccesso all'esponente: $(6)_{10} + 127 = (133)_{10Ecc127}$
- Conversione $(133)_{10Ecc127}$ in binario: $(10000101)_{2Ecc127} \rightarrow b_{30}b_{29}...b_{24}b_{23}$

Quindi, ricapitolando:

- segno negativo → s = 1 → $b_{31} = 1$
- esponente:$(10000101)_{2Ecc127} \rightarrow b_{30}b_{29}...b_{24}b_{23} = 10000101$
- mantissa (M) = $(1.00001101)_2 \rightarrow b_{22}b_{21}...b_1b_0 = 00001101...000$
- $(-67,25)_{10} = 1\ 10000101\ 00001101000000000000000$

Bibliografia

- Andrew S. Tanenbaum e Todd Austin. Architettura dei calcolatori. Un approccio strutturale. 6/ED. Anno 2013. Pearson Italia spa. Appendice A. (Disponibile nella sezione “Biblioteca”)