

# Zahlendarstellung

## IEEE-P 754-FLOATING-POINT-STANDARD

Benjamin Tröster

Hochschule für Technik und Wirtschaft Berlin

15. Dezember 2021

# Fahrplan

IEEE-P 754

Beispiele

Rundung

# IEEE-P 754

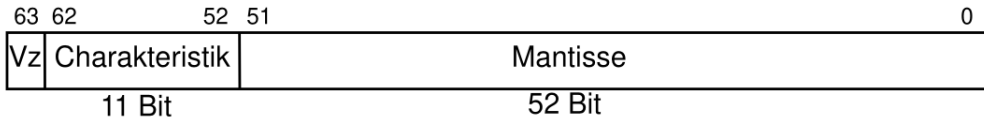
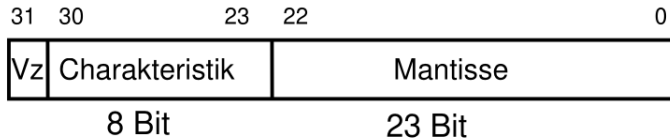
- ▶ Standardisierung von Gleitkommazahlen via IEEE Standard
- ▶ Original: [85]
- ▶ Aktuell: [Cor09]
- ▶ Schönes Paper: [Gol91]

# Normierung (IEEE-Standard)

- ▶ In vielen Programmiersprachen lassen sich Gleitkomma-Zahlen mit verschiedener Genauigkeit darstellen
  - ▶ C:
    - ▶ float
    - ▶ double
    - ▶ long double
  - ▶ Java
    - ▶ float
    - ▶ double
- ▶ Der IEEE-Standard definiert mehrere Darstellungsformen
  - ▶ IEEE single: 32 Bit
  - ▶ IEEE double: 64 Bit
  - ▶ IEEE extended: 80 Bit

# IEEE-P 754-Floating-Point-Standard

## Maschinenformate des IEEE-Standards



# Eigenschaften des IEEE-P 754

- ▶ Die Basis  $q$  ist gleich 2
- ▶ Das erste Bit der Mantisse wird implizit zu 1 angenommen, wenn die Charakteristik nicht nur Nullen enthält
- ▶ **Normalisierung:** das erste Bit der Mantisse (die implizite 1) steht vor dem Komma
- ▶ Ist die Charakteristik gleich 0, entspricht dies dem gleichen Exponenten wie die Charakteristik 1
  - ▶ Für die Darstellung der Subnormals und der Null
  - ▶ Normalisierte Zahlen kommen nur bis Minreal: Werte kleiner darstellbar, aber nicht normalisiert
- ▶ Das erste Bit der Mantisse wird aber dann explizit dargestellt
- ▶ **Auch die Null ist darstellbar**

# Eigenschaften des IEEE-P 754

- ▶ Sind alle Bits der Charakteristik gleich 1, signalisiert dies eine Ausnahmesituation
- ▶ Wenn zusätzlich die Mantisse gleich Null ist, wird die Situation „overflow“ (bzw. die „Zahl“  $\pm\infty$ ) kodiert
- ▶ Dies erlaubt es dem Prozessor, eine Fehlerbehandlung einzuleiten
- ▶ Intern arbeiten Rechner nach dem IEEE-Standard mit 80 Bit, um Rundungsfehler unwahrscheinlicher zu machen
- ▶ Charakteristik gleich 1 und Mantisse ungleich 0: NaN

# Zusammenfassung der Parameter des IEEE-P 754

Parameter	Single	Double
Bits Gesamt	32	64
Bits Mantisse	23(+1)	52(+1)
Bits Charakteristik	8	11
Exponent Bias	+127	+1023
$E_{max}$	+127	+1023
$E_{min}$	-126	-1022

Der Bias ist  $2^{n-1} - 1$  anstatt  $2^{n-1}$



# Zusammenfassung des 64-Bit-IEEE-Formats

Charakter.	Zahlenwert	Bemerkung
0	$(-1)^{V_z} 0, \textit{Mantisse} \cdot 2^{-1022}$	Subnormalisiert
1	$(-1)^{V_z} 1, \textit{Mantisse} \cdot 2^{-1022}$	Normalisiert
...	$(-1)^{V_z} 1, \textit{Mantisse} \cdot 2^{-1023}$	Normalisiert
2046	$(-1)^{V_z} 1, \textit{Mantisse} \cdot 2^{1023}$	Normalisiert
2047	$\textit{Mantisse} = 0 : (-1)^{V_z} \infty$	Overflow
2047	$\textit{Mantisse} \neq 0 :$	<i>NaN</i>

## Beispiel: $4, 4_{10}$

Übersicht: Umrechnung  $4, 4_{10} \rightarrow X_2$

- ▶ Vorgehen via Horner-Schema:
  - ▶ Ganzzahliger Teil:  $4_{10} = 100_2$
  - ▶ Nachkommateil:  $0,4 = 0, \overline{0110}_2$
- ▶ Normalisierung:  $100, \overline{0110}_2 \cdot 2^0 = 1, 000\overline{0110}_2 \cdot 2^2$
- ▶ Verrechnen mit Bias: 2 Bits Shifted:  $127 + 4 = 129$
- ▶ Charakteristik:  $129_{10} = 010000001_2$
- ▶  $010000001000\overline{0110}_2 = 01000000100011001100110011001100_2 \approx 4,3999996185302734375$

# Umrechnung: Ganzzahliger & Nachkommateil

## Ganzzahliger Anteil

	Div	Mod (Remainder)
$4 : 2$	2	0
$2 : 2$	1	0
$1 : 2$	0	1

## Nachkommabereich

		Carry
$0,4 \cdot 2$	0,8	0
$0,8 \cdot 2$	1,6	1
$0,6 \cdot 2$	1,2	1
$0,2 \cdot 2$	0,4	0
$0,4 \cdot 2$	...	

# Zusammensetzen und Verrechnung mit Bias

- ▶ Normalisierung:  $100, \overline{0110}_2 \cdot 2^0 = 1, \overline{000110}_2 \cdot 2^2$ 
  - ▶ Verschiebung um zwei Bits
  - ▶ 1 vor dem Komma redundant
- ▶ Bias: 8 Bit  $2^7 - 1$ ,  $B = 01111111_2 = 127_{10} - 127 \rightarrow$  d.h. Zahlen beginnen nicht bei 0, sonder bei -127
  - ▶ Um die Zahlen ohne Zweierkomplement darstellen zu können
- ▶ Einrechnen des Offsets: Daher  $127 + 2 = 129$  ist der codierte Exponent
- ▶  $129_{10} = 10000001_2$  ist das Exponent
- ▶  $10000001_2 \ 1, \overline{00011001100110011001100}_2$
- ▶ Mantisse 1 vor dem Komma kann weg
- ▶  $0100000010001100110011001100110_2 = 408cccc_{16}$

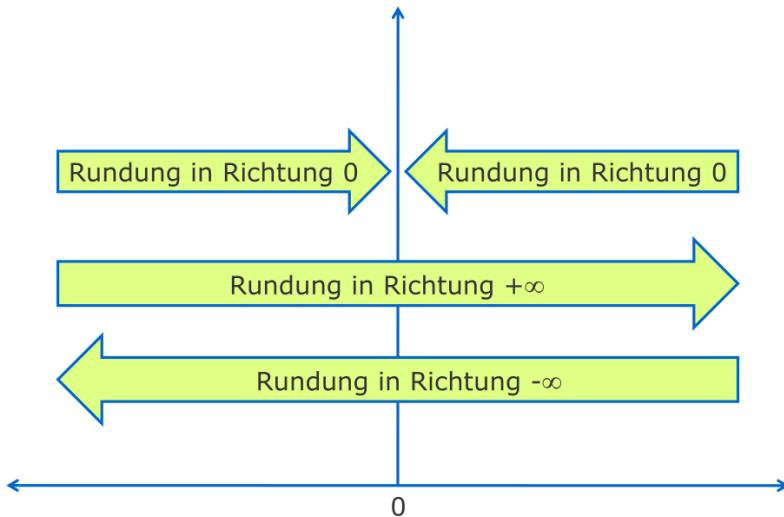
## Float (32 Bit) Minreal, Maxreal, Smallreal

- ▶ Maxreal:  $3.402823 \cdot 10^{38} = 34028230000000000000000000000000$ 
  - ▶  $0111111101111111111111111111111101_2$
  - ▶ Exponent:  $2^{127}$  mit Bias:  $254_{10} = 111111101_2$
- ▶ Minreal:  $1.175494 \cdot 10^{-38} = 0.0000000000000000000000000000117549393043$ 
  - ▶  $0000000001111111111111111111111101_2$
  - ▶ Exponent:  $2^{-126}$  mit Bias: 0, Mantisse voll besetzt
- ▶ Smallreal:  $1.0000001$  als Addition von  $1.0 + 1.0_2 \cdot 2^{-22}$ 
  - ▶ Binär:  $1_{10} = 00111111100000000000000000000000_2$
  - ▶ Kleinster Wert der mit +1 darstellbar ist:  
 $1 \cdot 10^{-7} = 00110011110101101011111110010101_2$
  - ▶ Addition ergibt aufgrund des Exponenten der 1:  
 $001111111000000000000000000000001$
  - ▶ Gegenbeispiel:  $1.00000001_{10} = 00111111100000000000000000000000$
  - ▶ Fehlerrate:  $-1E-8 = -1 \cdot 10^{-8}$

# Rundung

- ▶ IEEE Standard Forderung:
  - ▶ Das Ergebnis, das man durch eine arithmetische Operation mit dem Rechner erhält, soll dasselbe sein, als wenn man exakt rechnet und anschließend entsprechend eines geeigneten Modus rundet
- ▶ IEEE Standard definiert vier Rundungsmodi:
  - ▶ Rundung zum nächstliegenden Gleitkommawert:
    - ▶ Falls der Abstand zu zwei Gleitkommawerten gleich ist, wird zu jenem Wert gerundet, dessen niederwertigste Stelle eine gerade Ziffer ist („round-to-even“-Regel)
  - ▶ Rundung zum nächsten Gleitkommawert in Richtung 0
  - ▶ Rundung zum nächsten Gleitkommawert in Richtung  $+\infty$
  - ▶ Rundung zum nächsten Gleitkommawert in Richtung  $-\infty$

# Rundungen



# Rundungen

- ▶ Am schwierigsten zu implementierende Rundung:
  - ▶ Rundung zum nächstliegenden Gleitkommawert
- ▶ Eine Möglichkeit: Summe exakt berechnen und anschließend runden
  - ▶ sehr lange Register, sehr aufwändig
- ▶ Auch mit weniger Hardware-Aufwand möglich?
- ▶ Es gibt zwei Fälle für eine Rundung bei Addition und Subtraktion:
  - ▶ auftretender Übertrag
  - ▶ Exponentenanpassung



# Beispiel a: Übertrag bei Addition

Basis 10, drei signifikante Stellen (d.h. Mantisse hat maximal drei Stellen)

$$\begin{array}{r} 2,34 \cdot 10^2 \\ + 8,51 \cdot 10^2 \\ \hline 10,85 \cdot 10^2 \end{array}$$

wird gerundet zu

$$1,08 \cdot 10^3$$

## Beispiel b: ungleiche Exponenten

Basis 10, drei signifikante Stellen

$$\begin{array}{r} 2,34 \cdot 10^2 \\ + 2,56 \cdot 10^0 \end{array}$$

wird gerundet zu

$$\begin{array}{r} 2,34 \cdot 10^2 \\ 0,0256 \cdot 10^2 \\ \hline 2,3656 \cdot 10^2 \end{array}$$

$$2,37 \cdot 10^2$$

## Beispiel c: Übertrag und ungl. Exponenten

Basis 10, drei signifikante Stellen

beides

$$\begin{array}{r} 9,51 \cdot 10^2 \\ + 0,642 \cdot 10^2 \\ \hline \end{array}$$

wird gerundet zu

$$\begin{array}{r} 10,152 \cdot 10^2 \\ 1,02 \cdot 10^3 \end{array}$$

# Problem

- ▶ Für jeden dieser Fälle muss die Summe mit mehr als drei signifikanten Stellen berechnet werden, um eine korrekte Rundung zu ermöglichen
- ▶ Es gibt auch Fälle, bei denen eine Rechnung mit mehr als drei signifikanten Stellen notwendig ist, obwohl keine Rundung erfolgt
  - ▶ Subtraktion nahe beieinanderliegender Zahlen
  - ▶ Siehe Beispiel d

## Beispiel d: Subtraktion von naheliegenden Zahlen

$$\begin{array}{r} 1,47 \cdot 10^2 \\ -0,876 \cdot 10^2 \\ \hline 0,594 \cdot 10^2 \end{array}$$

- ▶ Bei den bisherigen Beispielen reichte eine zusätzliche Stelle aus
- ▶ Es gibt aber auch Fälle, bei denen dies nicht genügt

## Beispiel e

$$\begin{array}{r} 1,01 \cdot 10^2 \\ -0,0376 \cdot 10^2 \\ \hline 0,9724 \cdot 10^2 \end{array}$$

wird gerundet zu

$$0,972 \cdot 10^2$$

- Wenn die niederwertigste Ziffer 6 von 0,0376 gestrichen würde, wäre das Ergebnis 0,973 anstatt 0,972

# Rundungs- und Prüfstelle

- ▶ Es lässt sich zeigen, dass unter Vernachlässigung der „round-to-even“-Regel zwei weitere Stellen für eine korrekte Rundung stets ausreichend sind
- ▶ Diese beiden Stellen heißen:
  - ▶ die Rundungsstelle  $r$  und
    - ▶ Falls  $r > 0$  ist Runden einfach, da wir nicht in der Mitte sind
    - ▶ Was wenn  $r = 0$ ?
  - ▶ Prüfstelle  $g$ 
    - ▶ Sagt, ob wir  $r$  genauer betrachten müssen, wir könnten in der Mitte ( $g = 5_{10}$ ) sein
- ▶ Aber: Die „round-to-even“-Regel erfordert zusätzlichen Aufwand.

## Beispiel f:

Fünf signifikante Stellen:

$$4,5674 \cdot 10^0$$

$$2,5001 \cdot 10^{-4}$$

$$4,5674$$

$$+ 0,00025001$$

---

$$4,56765001$$

*gr*

wird gerundet zu

$$4,5677$$




- ▶ Rundungsstelle *r* und Prüfstelle *g* genügen nicht
- ▶ Information: sind alle niederwertigeren Stellen hinter der Rundungsstelle gleich Null, dann nur ein **sticky**-Bit



# Sticky-Bit

- ▶ Für eine richtige Rundung ist die Information ausreichend, ob alle niederwertigeren Stellen hinter der Rundungsstelle gleich Null sind
- ▶ Es genügt ein Bit: „sticky“-Bit
- ▶ Wenn eine der Stellen, die durch das Angleichen der Exponenten beider Operanden gestrichen werden, ungleich Null ist, wird das „sticky“-Bit gesetzt
- ▶ Falls das Ergebnis in gleichem Abstand zum oberen und unteren nächstliegenden Fließkommawert liegt, entscheidet das „sticky“-Bit, ob nach oben oder nach unten gerundet wird

# Quellen I

-  Cornea, Marius (2009). „IEEE 754-2008 Decimal Floating-Point for Intel® Architecture Processors“. In: *2009 19th IEEE Symposium on Computer Arithmetic*. IEEE, S. 225–228.
-  Goldberg, David (1991). „What every computer scientist should know about floating-point arithmetic“. In: *ACM computing surveys (CSUR)* 23.1, S. 5–48.
-  „IEEE Standard for Binary Floating-Point Arithmetic“ (1985). In: *ANSI/IEEE Std 754-1985*, S. 1–20. DOI: 10.1109/IEEESTD.1985.82928.