TU WIEN Informatics

03 – Numerik

Grundzüge digitaler Systeme (192.134)

Vortrag von: Stefan Neumann

Fakultät für Informatik
TU Wien
E-Mail: gds@list.tuwien.ac.at

Numerik

- Methoden zur Lösung mathematischer Problemstellungen auf Computern
- Hauptfelder
  - Effektive und effiziente Berechnung
  - Fehlerabschätzung
  - Aufwandsabschätzung
  - Stabilitätsanalyse
- Anwendungsgebiete in
  - Statistik und Machine Learning
  - Ingenieur-, Natur-, Wirtschafts- und Sozialwissenschaften

Zahlendarstellung im Computer

- In Mathematik unendliche Zahlenmengen
  - $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$
- Am Computer endliche Zahlenmengen
  - Stellenwertsystem zur Basis 2 mit n Stellen
  - Unterschiedliche „Zahlen“
    - $\mathbb{N}$
    - $\mathbb{Z}$: Negative Zahlen erfordern Codierung
      $\hookrightarrow$ VZ + Betrag, Einer- / Zweierkomplement, Exzessdarstellung
    - $\mathbb{Q}$, $\mathbb{R}$: Nachkommastellen!

Numerik – Übersicht

1. Festpunkt-Darstellung
2. Gleitkomma-Darstellung
   - Struktur von Gleitkomma-Zahlensystemen
   - Gleitkomma-Zahlensysteme nach IEEE 754
3. Arithmetik auf Gleitkomma-Zahlensystemen
   - Runden
   - Rundungsfehler
   - Beispiele für Arithmetik mit Gleitkomma-Zahlen

Festpunkt-Darstellung

- Gesamtlänge Bit
  - g Vorkommastellen
  - n Nachkommastellen
  - 1 Vorzeichen (0 $\Rightarrow$ positiv, 1 $\Rightarrow$ negativ)
- $\Rightarrow$ $N = n + g + 1$ Bits insgesamt und Position des Nachkommateils fixiert

$$\overbrace{\boxed{v}}^{\text{VZ}}\ \overbrace{\boxed{d_{n+g-1}}\boxed{d_{n+g-2}}\boxed{\cdots}\boxed{d_n}}^{\text{Vorkommateil } (g)}\ \overbrace{\boxed{d_{n-1}}\boxed{\cdots}\boxed{d_1}\boxed{d_0}}^{\text{Nachkommateil } (n)}$$

- Beispiel für Zahlendarstellung ($g = 12$, $n = 3$):

VZ	Vorkommateil (g)	Nachkommateil (n)
1	000 0000 0000 1	011

$$\begin{aligned}
&\hat{=} -(1.011)_2 \\
&= (-1)^1 \cdot (2^0 + 2^{-2} + 2^{-3}) \\
&= (-1)^1 \cdot 2^{-3} \cdot (2^3 + 2^1 + 2^0) \\
&= -(1.375)_{10}
\end{aligned}$$

Festpunkt-Darstellung

- Gesamtlänge Bit
  - g Vorkommastellen
  - n Nachkommastellen
  - 1 Vorzeichen ($0 \Rightarrow$ positiv, $1 \Rightarrow$ negativ)

  $\Rightarrow$ $N = n + g + 1$ Bits insgesamt und Position des Nachkommateils fixiert

$$\overbrace{\boxed{v}}^{\text{VZ}}\,\overbrace{\boxed{d_{n+g-1}}\boxed{d_{n+g-2}}\boxed{\cdots}\boxed{d_n}}^{\text{Vorkommateil } (g)}\,\overbrace{\boxed{d_{n-1}}\boxed{\cdots}\boxed{d_1}\boxed{d_0}}^{\text{Nachkommateil } (n)}$$

- Entspricht Skalierung der ganzen Zahl um Faktor 2^{-n}
- Bitfolge interpretiert als vorzeichenbehaftete Binärzahl mit Nachkommastellen

$$\textbf{Codierung:}\; v\, d_{N-2}\, d_{N-3} \ldots d_1\, d_0$$

$$\mathrel{\hat{=}} (-1)^v \cdot 2^{-n} \sum_{j=0}^{N-2} d_j \cdot 2^j$$

$$\textbf{Festpunktzahl:}\; (-1)^v \cdot d_{N-2} \ldots d_n . d_{n-1} \ldots d_1 d_0$$

Festpunkt-Darstellung – Beispiel

- Die Zahl $(-10.375)_{10}$ ist in das folgende (binäre) Festpunktformat umzurechnen
- Format: $N = 12$ Bit Breite und $n = 3$ Nachkommastellen
  $\Rightarrow$ 1 Bit Vorzeichen, 8 Bit Vorkommateil, 3 Bit Nachkommateil

$$(-10.375)_{10} = (-1010.011)_2 = 1\mathbf{0000}1010011$$

$$\begin{array}{r|l} & \cdot 2 \\ 0.375 & 0 \\ 0.75 & 1 \\ 0.5 & 1 \end{array} \qquad \begin{array}{r|l} & :2 \\ 10 & 0 \\ 5 & 1 \\ 2 & 0 \\ 1 & 1 \end{array}$$

Festpunkt-Darstellung

- Welche Zahlenmenge können wir so darstellen?
- Für das Festpunkt-Zahlensystem mit $N = 16$ Bit Breite und $n = 3$ Nachkommastellen ist die Zahlenmenge beschrieben durch:

$$v\, d_{14}\, d_{13} \ldots d_1\, d_0 \triangleq (-1)^v \cdot 2^{-3} \sum_{j=0}^{14} d_j \cdot 2^j$$

- Kleinste in diesem Zahlensystem ($N = 16$, $n = 3$) darstellbare Zahl

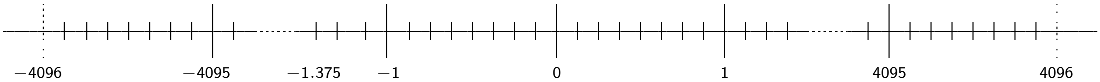
$$\begin{array}{cccl} \text{VZ} & \text{Vorkommateil} & \text{Nkt.} & \\ 1 & 111\ 1111\ 1111\ 1 & 111 & \triangleq -(1111\ 1111\ 1111.111)_2 \\ & & & = (-1)^1 \cdot 2^{-3} \cdot (2^{14} + 2^{13} + \cdots + 2^1 + 2^0) \\ & & & = -(4095.875)_{10} \end{array}$$

Festpunkt-Darstellung

- Differenz zwischen zwei aufeinander folgenden Zahlen für $N = 16$, $n = 3$:

VZ	Vorkommateil	Nkt.	
0	000 0000 0000 0	001	$\hat{=} (0.001)_2 = (-1)^0 \cdot 2^{-3} \cdot 2^0 = (0.125)_{10}$

- Zahlenbereich $[-4095.875, +4095.875]$



Festpunkt-Darstellung – Eigenschaften

- Jede Festpunktzahl ist rational ($\mathbb{Q}$)
  - D.h. irrationale Zahlen können nicht exakt dargestellt werden
- Manche einfache rationale Zahlen können nicht genau dargestellt werden
  - Beispiel: $(^1/_3)_{10}$ dezimal dargestellt
  - Beispiel: $(^1/_{10})_{10}$ binär dargestellt
- Wir müssen uns damit abfinden, dass reelle Zahlen im Rechner nur mit einer gewissen Genauigkeit dargestellt werden können

  $\Rightarrow$ Es muss gerundet werden
- Ergebnis einer Rechnung von zwei darstellbaren Zahlen muss nicht unbedingt darstellbar sein

  $\Rightarrow$ Es muss gerundet werden

Numerik – Übersicht

1. Festpunkt-Darstellung
2. Gleitkomma-Darstellung
   - Struktur von Gleitkomma-Zahlensystemen
   - Gleitkomma-Zahlensysteme nach IEEE 754
3. Arithmetik auf Gleitkomma-Zahlensystemen
   - Runden
   - Rundungsfehler
   - Beispiele für Arithmetik mit Gleitkomma-Zahlen

Festpunkt-Darstellung → Gleitkommazahlen

Gewünschte Eigenschaften

- In vielen Anwendungen wird eine große Zahlendynamik benötigt
  - Sehr kleine und sehr große Zahlen sollen einheitlich dargestellt werden
- Große Anzahl an Nachkommastellen in der Umgebung von 0

  ⇒ Betragsmäßig sehr kleine Zahlen darstellbar
- Große Anzahl an Vorkommastellen

  ⇒ Für Zahlen mit großem Absolutbetrag
- Anzahl der Nachkommastellen kann mit steigendem Absolutbetrag abnehmen, da auch ihre Bedeutung abnimmt

⇒ Man benötigt ein Zahlensystem, bei dem die Anzahl der Nachkommastellen und damit die Position des Binärpunktes abhängig vom Absolutbetrag variieren (gleiten) kann:

Gleitkommazahlen (auch Gleitpunktzahlen, floating point number)

Exponentialschreibweise

Darstellung

- Grundlage von Gleitkommazahlen: Exponentialschreibweise
- $x = m \cdot b^e$ z.B.: $0.0488 = 4.88 \cdot 10^{-2}$
  - m ... Mantisse
  - e ... Exponent
  - b ... Basis
  - $\Rightarrow$ Im Gegensatz zu bisher ist die Mantisse nun selbst eine rationale Zahl (nicht unbedingt eine Ganzzahl)
- **Problem**: Exponential-Darstellung ist mehrdeutig
  - Beispiel: $0.00123 = 123 \cdot 10^{-5} = 12.3 \cdot 10^{-4} = \dots$
- **Lösung**: Normalisierung, d.h. genau eine Stelle vor dem Komma $\neq 0$
  - Beispiel: 0.00123 wird normalisiert dargestellt als $1.23 \cdot 10^{-3}$
- Wir gehen davon aus, dass
  - Exponent ganzzahlig ist und
  - Basis ganzzahlig ist mit $b > 1$

Exponentialschreibweise

Rechnen

- Bei Addition oder Subtraktion weniger handlich
- Bei Multiplikation und Division praktisch
  - Exponenten werden addiert oder subtrahiert
    - $2.1 \cdot 10^{-4} \cdot 3 \cdot 10^{-5} = (2.1 \cdot 3) \cdot 10^{-4+(-5)} = 6.3 \cdot 10^{-9}$
    - $9.6 \cdot 10^{3} : (2 \cdot 10^{-2}) = (9.6 : 2) \cdot 10^{3-(-2)} = 4.8 \cdot 10^{5}$
    - $(3 \cdot 10^{7})^{3} = 3^{3} \cdot 10^{7 \cdot 3} = 27 \cdot 10^{21} = 2.7 \cdot 10^{22}$
    - $\log(200000) = \log(2 \cdot 10^{5}) = \log(2) + 5$
  - Allgemein:
    - $(m_1 \cdot 10^{e_1}) \cdot (m_2 \cdot 10^{e_2}) = (m_1 \cdot m_2) \cdot 10^{e_1+e_2}$
    - $(m_1 \cdot 10^{e_1}) : (m_2 \cdot 10^{e_2}) = (m_1 : m_2) \cdot 10^{e_1-e_2}$
    - $(m \cdot 10^{e})^{p} = m^{p} \cdot 10^{e \cdot p}$
    - $\log(m \cdot 10^{e}) = e + \log(m)$

Gleitkomma-Darstellung

Format

- Gesamtlänge $N = 1 + n + p$ Bits
  - Vorzeichenbit v ($0 \Rightarrow$ positiv, $1 \Rightarrow$ negativ)
  - n Stellen Exponent, bezeichnet als $E_{n-1}, \dots, E_0$
  - p Stellen Mantisse, bezeichnet als $m_0, \dots, m_{p-1}$

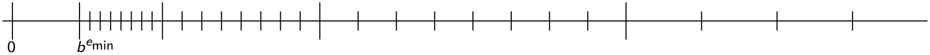
VZ	Exponent				Mantisse					
v	E_{n-1}	$\cdots$		E_0	m_0	$\cdots$			m_{p-1}	Indizierung
v	d_{N-2}	d_{N-3}	$\cdots$	$\cdots$	d_p	d_{p-1}	$\cdots$	$\cdots$	d_1	d_0	Bitnummer
msb										lsb	

- Implementierung der Normalisierung wie folgt:
  - Normalisierungsbedingungen:
    - Erste Stelle der Mantisse $m_0 \neq 0$
    - Genau eine Stelle vor dem Komma
    - $\Rightarrow$ Mantisse wird dargestellt als Festpunktzahl mit genau einer Ziffer im Vorkommateil
  - Beispiel: 0.00123 wird normalisiert dargestellt als $1.23 \cdot 10^{-3}$

Gleitkomma-Darstellung

Normalisierung

- Kleinste positive normalisierte Zahl: $m_{\min} \cdot b^{e_{\min}} = 1.0 \cdot b^{e_{\min}}$
  - $m_{\min}$ ... minimale Mantisse
  - $e_{\min}$ ... minimaler Exponent
- Normalisierte Zahlen auf der positiven Zahlengerade

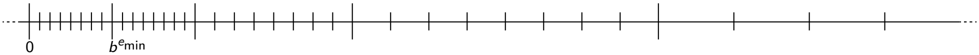


- **Problem:** Lücke um 0
  - Lücke tritt auf, da $m_0 \neq 0$
  - 0 so nicht darstellbar
  - Wir führen Sonderdarstellung für 0 und Zahlen „nahe“ 0 ein, sogenannte Denormalisierung

Gleitkomma-Zahlensystem

Denormalisierte Zahlen

- Wegen $m_0 \neq 0$ fallen einige Zahlen weg
- Dieses Problem löst man durch sogenannte Denormalisierung:
  - Man hebt die Normalisierungsbedingung in Spezialfällen auf: Für den minimalen Exponenten $e = e_{\min}$ erlauben wir $m_0 = 0$
  - $\Rightarrow$ Schließt die „Lücke“ um 0
  - Denormalisierte Zahlen (auch „subnormale“ Zahlen genannt)
  - Im Binärsystem mögliche m: (0.0...01) bis (0.1...11)
  - 0 gilt als normalisiert
- Denormalisierte Zahlen auf der positiven Zahlengerade



Struktur von Gleitkomma-Zahlensystemen

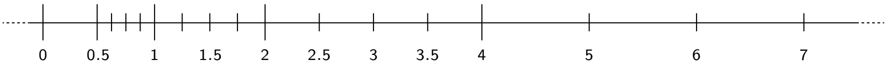
$\mathbb{F}(b, p, e_{\text{min}}, e_{\text{max}}, \textit{denorm})$ ... Gleitkomma-Zahlensystem

- b ... Basis (base, radix) ($b \geq 2$)
- p ... Mantissenlänge (precision) ($p \geq 2$)
- e_{min} ... kleinster Exponent
- e_{max} ... größter Exponent
- *denorm* ... Normalisierungsindikator
  - true $\Rightarrow$ enthält denormalisierte Zahlen
  - false $\Rightarrow$ enthält keine denormalisierten Zahlen

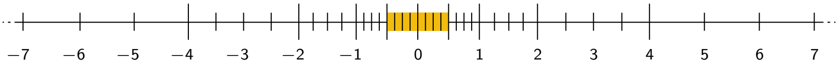
Gleitkomma-Darstellung

Bsp.: Normalisierte und denormalisierte Gleitkommazahlen

- Gleitkomma-Zahlensystem mit $b = 2$, $p = 3$, $e_{\text{min}} = -1$, $e_{\text{max}} = 2$
- Normalisierte Gleitkommazahlen (positiver Teil dargestellt)



- Denormalisierte Gleitkommazahlen



  - Skalierungsfaktor für denormalisierte Zahlen: $b^{e_{\text{min}}}$
  - In IEEE-754 durch Sonderwert im Exponenten codiert: $e_{\text{min}} - 1$ (mehr dazu später)

Struktur von Gleitkomma-Zahlensystemen

- **Frage:** Wie viele verschiedene Zahlen können wir darstellen?
- **Antwort:** IEC/IEEE Gleitkomma-Zahlensystem $\mathbb{F}(2, 24, -126, 127, \text{true})$:

  $\Rightarrow$ Basis $= 2$, 24 Bits in der Mantisse, Exponenten $e_{\text{min}} = -126$ bis $e_{\text{max}} = 127$, mit Denormalisierung

  - Normalisierte Zahlen: $2 + 2^{24} \cdot 254 = 4\,261\,412\,866 \approx 4.26 \cdot 10^9$
  - Denormalisierte Zahlen: $2 \cdot (2^{23} - 1) = 16\,777\,214$

- **Allgemein:** Gleitkomma-Zahlensystem $\mathbb{F}(b, p, e_{\text{min}}, e_{\text{max}}, denorm)$

  - Normalisierten Zahlen: $\underbrace{2}_{\pm 0} + \overbrace{2}^{\text{Vorzeichen}} \cdot \underbrace{\overbrace{(b-1)}^{m_0 \neq 0} \cdot b^{p-1}}_{\text{Anzahl der möglichen normalisierten Mantissen}} \cdot \overbrace{(e_{\text{max}} - e_{\text{min}} + 1)}^{\text{Anzahl der möglichen Exponenten}}$

  - Denormalisierten Zahlen: $\underbrace{2}_{\text{VZ}} \cdot \underbrace{1}_{m_0 = 0} \cdot (b^{p-1} \underbrace{-1}_{\text{Mantisse } 0 \cdots 0})$

Struktur von Gleitkomma-Zahlensystemen

- **Frage:** Was ist der größte Wert, den wir darstellen können?
- **Antwort:** Größte Gleitkommazahl eines Gleitkomma-Zahlensystems ist

$$x_{max} = M_{max} \cdot b^{e_{max}} = b \cdot (1 - b^{-p}) \cdot b^{e_{max}}$$

mit der Mantisse $M_{max} = (\delta.\delta\delta \ldots \delta\delta)_b$, wobei $\delta = b - 1$

  - Für Binärsystem $x_{max} = 2 \cdot (1 - 2^{-p}) \cdot 2^{e_{max}}$

    Herleitung:

    - Maximaler Wert der Mantisse:
    $$\begin{array}{rr} & \overbrace{\delta \cdots \delta}^{p \text{ Stellen}} \\ + & 1 \\ \hline & 10 \cdots 0 = b^p \end{array}$$
    - $(b^p - 1) \cdot \underbrace{b^{-(p-1)}}_{\text{Skalierung}} = (b^p - 1) \cdot b \cdot b^{-p} = b \cdot (1 - b^{-p})$
    - $x_{max} = M_{max} \cdot b^{e_{max}} = b \cdot (1 - b^{-p}) \cdot b^{e_{max}}$

Struktur von Gleitkomma-Zahlensystemen

- **Frage:** Was ist der kleinste positive Wert, den wir darstellen können?
- **Antworten:**
  - Kleinste positive normalisierte Gleitkommazahl ($denorm = \text{false}$)

    $$x_{\text{min}} = M_{\text{min}} \cdot b^{e_{\text{min}}} = 1 \cdot b^{e_{\text{min}}} = b^{e_{\text{min}}}$$

  - Kleinste positive denormalisierte Zahl ($denorm = \text{true}$):

    $$\overline{x}_{\text{min}} = b^{-p+1} \cdot b^{e_{\text{min}}} = b^{e_{\text{min}}-p+1}$$

- IEC/IEEE Gleitkomma-Zahlensystem $\mathbb{F}(2, 24, -126, 127, \text{true})$:

  $$x_{\text{min}} = 2^{-126} \approx 1.18 \cdot 10^{-38}$$

  $$x_{\text{max}} = 2 \cdot (1 - 2^{-24}) \cdot 2^{127} \approx 3.40 \cdot 10^{38}$$

Struktur von Gleitkomma-Zahlensystemen

Absolute Abstände der Gleitkommazahlen

- **Frage:** Was ist der Abstand von zwei Zahlen, die wir darstellen können?
- Für eine normalisierte Gleitkommazahl besteht die kleinste und die größte Mantisse aus den Ziffern
  - $m_0 = 1,\ m_1 = \cdots = m_{p-1} = 0$ bzw.
  - $m_0 = m_1 = \cdots = m_{p-1} = \delta = b - 1$
- Die Mantisse durchläuft somit Werte zwischen
  - $M_{\text{min}} = (1.00 \cdots 00)_b$ und
  - $M_{\text{max}} = (\delta.\delta\delta \cdots \delta\delta)_b$

  mit einer konstanten Schrittweite von
  - $\text{ulp} = (0.00 \cdots 01)_b = b^{-p+1}$
  - Grundinkrement der Mantisse: ulp (**u**nit of **l**east **p**recision)
- **Antwort:** Benachbarte Zahlen aus $\mathbb{F}$ haben im Intervall $[b^e, b^{e+1})$ (also bei fixiertem Exponenten e) den konstanten Abstand

$$\Delta x = 1\,\text{ulp} \cdot b^e = b^{e-p+1}$$

Struktur von Gleitkomma-Zahlensystemen

Positive Zahlen aus dem Gleitkomma-Zahlensystem $\mathbb{F}(2, 3, -1, 2, \text{true})$

M	e	$(\text{Wert})_2$	$(\text{Wert})_{10}$	Intervall	Δx	denormalisiert
1.11 1.10 1.01 1.00	2	$(111)_2$ $(110)_2$ $(101)_2$ $(100)_2$	$(7)_{10}$ $(6)_{10}$ $(5)_{10}$ $(4)_{10}$	$[2^2, 2^3)$	$(1.0)_2$	nein
1.11 1.10 1.01 1.00	1	$(11.1)_2$ $(11.0)_2$ $(10.1)_2$ $(10.0)_2$	$(3.5)_{10}$ $(3.0)_{10}$ $(2.5)_{10}$ $(2.0)_{10}$	$[2^1, 2^2)$	$(0.1)_2$	nein
1.11 1.10 1.01 1.00	0	$(1.11)_2$ $(1.10)_2$ $(1.01)_2$ $(1.00)_2$	$(1.75)_{10}$ $(1.50)_{10}$ $(1.25)_{10}$ $(1.00)_{10}$	$[2^0, 2^1)$	$(0.01)_2$	nein
1.11 1.10 1.01 1.00	-1	$(0.111)_2$ $(0.110)_2$ $(0.101)_2$ $(0.100)_2$	$(0.875)_{10}$ $(0.750)_{10}$ $(0.625)_{10}$ $(0.500)_{10}$	$[2^{-1}, 2^0)$	$(0.001)_2$	nein
0.11 0.10 0.01	-2	$(0.011)_2$ $(0.010)_2$ $(0.001)_2$	$(0.375)_{10}$ $(0.250)_{10}$ $(0.125)_{10}$	$[0, 2^{-1})$	$(0.001)_2$	ja
1.00	-2	0	0	—	—	nein

Gleitkomma-Zahlensysteme nach IEEE 754

- Jetzt: Gleitkomma-Zahlensysteme nach IEEE 754
- International anerkannter Standard, wie man Gleitzahlen darstellt
- IEEE = Institute of Electrical and Electronics Engineers
  - Gesprochen (auf Englisch): „I triple E“
  - Berufsverband für Informatik, Elektrotechnik, etc.
  - Definiert diverse Standards für elektronische Systeme (ähnlich DIN-Normen)
  - Organisiert auch akademische Konferenzen
  - Weltweit vertreten (auch in Wien)

Gleitkomma-Zahlensysteme nach IEEE 754

- IEEE 754 Single Precision Format mit 32 Bit floats: $\mathbb{F}(2, 24, -126, +127, \text{true})$
- IEEE 754 Double Precision Format mit 64 Bit floats: $\mathbb{F}(2, 53, -1022, +1023, \text{true})$
- Exponent ist in **Exzessdarstellung**
  - Single Precision: Warum gehen Exponenten nur von $e_{\text{min}} = -126$ bis $e_{\text{max}} = 127$, Exzessdarstellung würde uns -127 bis 128 erlauben?

  ⇒ Die Werte $e = e_{\text{min}} - 1 = -127$ und $e = e_{\text{max}} + 1 = 128$ hält man sich für Spezifälle frei (siehe einige Slides weiter, double precision analog)

	Format			
Parameter	Single	Single Ext.	Double	Double Ext.
b	2	2	2	2
p	24	≥ 32	53	≥ 64
e_{min}	-126	≤ -1022	-1022	≤ -16382
e_{max}	$+127$	$\geq +1023$	$+1023$	$\geq +16383$
denorm	true	true	true	true
Exzess des Exponenten	$+127$	unspez.	$+1023$	unspez.
Bitbreite des Exponenten	8	≥ 11	11	≥ 15
Bitbreite des Formats	32	≥ 43	64	≥ 79

Gleitkomma-Zahlensysteme nach IEEE 754

Codierung nach IEEE 754 Single Precision Format (1 von 3)

1 Bit VZ	8 Bit Exponent					23 Bit Mantisse					
v	E_7	$\cdots$			E_0	m_1	$\cdots$			m_{23}	Indizierung
d_{31}	d_{30}	d_{29}	$\cdots$	$\cdots$	d_{23}	d_{22}	$\cdots$	$\cdots$	d_1	d_0	Bitnummer
msb										lsb	

- Darstellung von normalisierten Gleitkommazahlen:
  - Normalisierungsbedingung: $m_0 \neq 0$, daher immer $m_0 = 1$
    $\Rightarrow$ Vorkommastelle m_0 wird weggelassen: „Implizites erstes Bit“
  - Exponenten in Exzessdarstellung

- Darstellung von denormalisierten Gleitkommazahlen:
  - Spezielle **Codierung** im Exponenten: $e = e_{\min} - 1$
    - **Zeigt** implizites erstes Bit **an**: $m_0 = 0$
    - Als **Wert** des Exponenten wird gesetzt $e_{\min}$

Gleitkomma-Zahlensysteme nach IEEE 754

Codierung nach IEEE 754 Single Precision Format (2 von 3)

- Wie wird die Zahl Null dargestellt?
  - $e = e_{\text{min}} - 1 = -127$, Mantisse $= 0$
  - Zwei Möglichkeiten:

	31	30 ⋯ 23	22 ⋯ 0
$+0$	0	$0 \cdots 0$	$0 \cdots 0$
-0	1	$0 \cdots 0$	$0 \cdots 0$

- Wie wird Not a Number (NaN) dargestellt?
  - Sonderwert für „Ergebnisse“ nicht möglicher Berechnungen, wie
    - $\frac{0}{0}$
    - $\sqrt{-1}$
  - $e = e_{\text{max}} + 1 = 128$, Mantisse > 0
  - Binärdarstellung z.B.:

31	30 ⋯ 23	22 ⋯ 0
0	$1 \cdots 1$	$0010 \cdots 0$

Gleitkomma-Zahlensysteme nach IEEE 754

Codierung nach IEEE 754 Single Precision Format (3 von 3)

- Wie wird der Wert unendlich dargestellt?
- Überlauf: Ergebnis ist betragsmäßig zu groß, um im betreffenden Format dargestellt werden zu können
  - Es wird auf $+\infty$ bzw. $-\infty$ als Rückgabewert zurückgegriffen
  - $e = e_{\text{max}} + 1 = 128$
  - Binärdarstellung:

	31	30 ⋯ 23	22 ⋯ 0
$+\infty$	0	$1 \cdots 1$	$0 \cdots 0$
$-\infty$	1	$1 \cdots 1$	$0 \cdots 0$

    - $\sqrt{+\infty} = +\infty$
    - $\frac{1}{+\infty} = +0$ bzw. $\frac{1}{-\infty} = -0$
    - $4 - \infty = -\infty$
    - $\frac{1}{+0} = +\infty$ bzw. $\frac{1}{-0} = -\infty$

Gleitkomma-Zahlensysteme nach IEEE 754

Codierung nach IEEE 754 Exzessdarstellung des Exponenten

- Gleitkomma-Zahlensystem: $\mathbb{F}(b, p, e_{\min}, e_{\max}, \mathit{denorm})$ mit denorm $=$ true
- Wir müssen die Werte $[e_{\min} - 1, e_{\max} + 1]$ abbilden
  - Wertebereich des Exponenten $[e_{\min}, e_{\max}]$
  - $e_{\min} - 1$ als Sonderwert für denormalisierte Zahlen
  - $e_{\max} + 1$ als Sonderwert für NaN, $\pm\infty$
- Exzess ergibt sich aus der kleinsten Zahl: $e = -e_{\min} + 1$
  - $-e_{\min}$ da $e_{\min}$ negativ ist
- Bsp: Single Precision Format
  - $e_{\min} = -126$, $e_{\max} = 127$
  - Exzess $e = 127$
  - Wertebereich der Exzesscodierung $[0, 255]$ (8 bit)

Gleitkomma-Zahlensysteme nach IEEE 754

Die Grundformate einfacher und doppelter Genauigkeit

	Exponent		NKSt. f	Wert
Format	allgemein	dezimal	der Mant.	der Gleitkommazahl
single	$e_{max}+1$	128	$f \neq 0$	NaN
	$e_{max}+1$	128	$f = 0$	$(-1)^v \cdot \infty$
	$e_{min} \leq e \leq e_{max}$	$-126 \leq e \leq 127$	beliebig	$(-1)^v \cdot 1.f \cdot 2^e$
	$e_{min}-1$	-127	$f \neq 0$	$(-1)^v \cdot 0.f \cdot 2^{-126}$
	$e_{min}-1$	-127	$f = 0$	$(-1)^v \cdot 0$
double	$e_{max}+1$	1024	$f \neq 0$	NaN
	$e_{max}+1$	1024	$f = 0$	$(-1)^v \cdot \infty$
	$e_{min} \leq e \leq e_{max}$	$-1022 \leq e \leq 1023$	beliebig	$(-1)^v \cdot 1.f \cdot 2^e$
	$e_{min}-1$	-1023	$f \neq 0$	$(-1)^v \cdot 0.f \cdot 2^{-1022}$
	$e_{min}-1$	-1023	$f = 0$	$(-1)^v \cdot 0$

v ... VZ-Bit der Mantisse

NKSt. ... Nachkommastellen

Gleitkomma-Zahlensysteme nach IEEE 754

Beispiel: Codierung einer Dezimalzahl in das IEEE 754-Format (1 von 2)

- Wandeln Sie die Zahl $(-172.625)_{10}$ in das IEEE 754 Single Precision Format um.
- **Schritt 1:** Umwandeln ins Binärsystem

$$(-172.625)_{10} = (-10101100.101)_2$$

$$\begin{array}{r|l} & \cdot 2 \\ 0.625 & 1 \\ 0.25 & 0 \\ 0.5 & 1 \end{array} \downarrow \qquad \begin{array}{r|l} & :2 \\ 172 & 0 \\ 86 & 0 \\ 43 & 1 \\ 21 & 1 \\ 10 & 0 \\ 5 & 1 \\ 2 & 0 \\ 1 & 1 \end{array} \uparrow$$

Gleitkomma-Zahlensysteme nach IEEE 754

Beispiel: Codierung einer Dezimalzahl in das IEEE 754-Format (2 von 2)

- **Schritt 2:** Normalisierung

$$(10101100.101)_2 \cdot 2^0 = (1.0101100101)_2 \cdot 2^7$$

- **Schritt 3:** Exzessdarstellung des Exponenten berechnen

$$\begin{array}{rcccccccclll}
 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & = (127)_{10} & & \text{Exzess} \\
+ & {}_1 0 & {}_1 0 & {}_1 0 & {}_1 0 & {}_1 0 & {}_1 1 & {}_1 1 & {}_1 1 & = (7)_{10} & & \text{Exponent d. normalisierten Darstellung} \\
\hline
 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & = (134)_{10} & & \text{Exponent in Exzessdarstellung}
\end{array}$$

- **Schritt 4:** Vorzeichenbit setzen
  - Negative Zahl $\Rightarrow v = 1$

1 Bit VZ	8 Bit Exponent	23 Bit Mantisse
1	10000110	01011001010000000000000

Implizites 1. Bit

Gleitkomma-Zahlensysteme nach IEEE 754

Beispiel: Ablesen von Zahlen im IEEE 754-Format (1 von 2)

- Gegeben folgendes Codewort im IEEE 754 Single Precision Format:

1 Bit VZ	8 Bit Exponent	23 Bit Mantisse
1	10000111	10110000000000000000000

Implizites 1. Bit

  - Exponent ist $(1000\ 0111)_2 - e = (1000\ 0111)_2 - (0111\ 1111)_2 = (0000\ 1000)_2 = (8)_{10}$
  - Mantisse (inklusive implizitem 1. Bit) ist $(1.1011)_2 = 1.6875$
  - Dargestellte Zahl ist $-1.6875 \cdot 2^8 = -432$

Gleitkomma-Zahlensysteme nach IEEE 754

Beispiel: Ablesen von Zahlen im IEEE 754-Format (2 von 2)

- Gegeben folgendes Codewort im IEEE 754 Single Precision Format – normalisierte Zahl:

0	00000001	00000000010000000000000

  - Exponent ist $e_{\text{min}} = -126$
  - Mantisse (inklusive implizitem 1. Bit) ist $(1.0000000001)_2 = 1 + 2^{-10}$
  - Dargestellte Zahl ist $(1 + 2^{-10}) \cdot 2^{-126}$

- Gegeben folgendes Codewort im IEEE 754 Single Precision Format – denormalisierte Zahl:

0	00000000	00000000010000000000000

  - Exponent ist eigentlich -127, aber der IEEE-Standard legt fest, dass wir in dem Fall denormalisierte Zahlen mit dem Exponenten $e_{\text{min}} = -126$ verwenden
  - Mantisse (inklusive implizitem 1. Bit) ist $(0.0000000001)_2 = 2^{-10}$
  - Dargestellte Zahl ist $1 \cdot 2^{-136}$

    $\Rightarrow$ Viel kleiner als die normalisierte Zahl oben

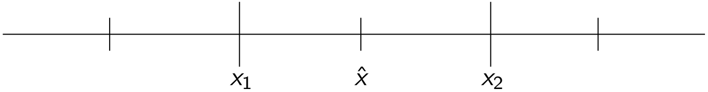
Numerik – Übersicht

1. Festpunkt-Darstellung
2. Gleitkomma-Darstellung
   - Struktur von Gleitkomma-Zahlensystemen
   - Gleitkomma-Zahlensysteme nach IEEE 754
3. Arithmetik auf Gleitkomma-Zahlensystemen
   - Runden
   - Rundungsfehler
   - Beispiele für Arithmetik mit Gleitkomma-Zahlen

Arithmetik auf Gleitkomma-Zahlensystemen

Runden (1 von 5)

- Unendlich viele reelle Zahlen $\mathbb{R}$ im Widerspruch zu:
  - Endlich viele in einem Computer darstellbare Gleitkommazahlen $\mathbb{F}$
  - Mittels Rundungsfunktion $\square$ reelle Zahl auf Gleitkommazahl abbilden
  - Abbildung $\square : \mathbb{R} \to \mathbb{F}$,
    die jeder reellen Zahl $x \in \mathbb{R}$
    eine bestimmte „benachbarte“ Zahl $\square x \in \mathbb{F}$ zuordnet



  - $x_1, x_2 \in \mathbb{F}$, $\hat{x}$ . . . Grenzwert
  - **Rundungsfunktion $\square$ bestimmt** als Ergebnis der Rundung $\square x$ **einen der beiden Werte** x_1 oder x_2
    - Beispiel: $\square$ = Runden auf nächste Ganzzahl
    - $x = 13.7$, $x_1 = 13$, $x_2 = 14$, $\hat{x} = 13.5$
    - $\square x = 14$

Arithmetik auf Gleitkomma-Zahlensystemen

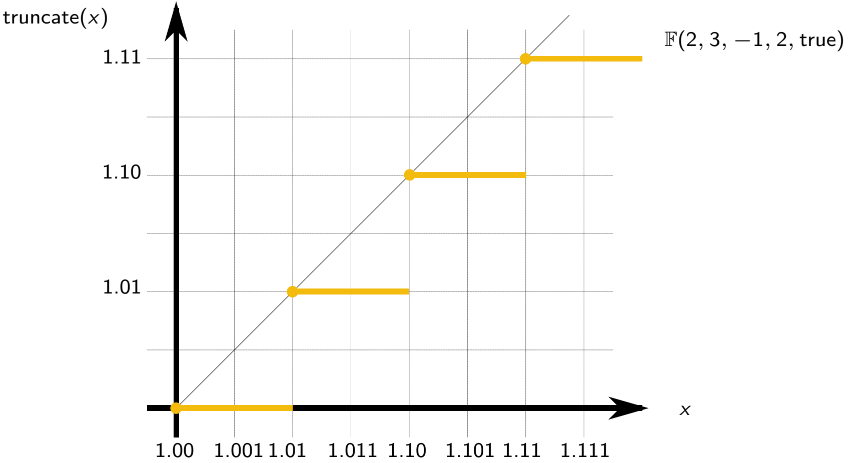
Runden (2 von 5)

- Berechnungen mit Zahlen aus $\mathbb{F}$: Ergebnis meist keine Zahl aus $\mathbb{F}$
  - Beispiel:
    - $\mathbb{F} =$ Dezimalzahlen mit 2 Nachkommastellen
    - $7.11 \cdot 1.38 = 9.8118 \notin \mathbb{F}$

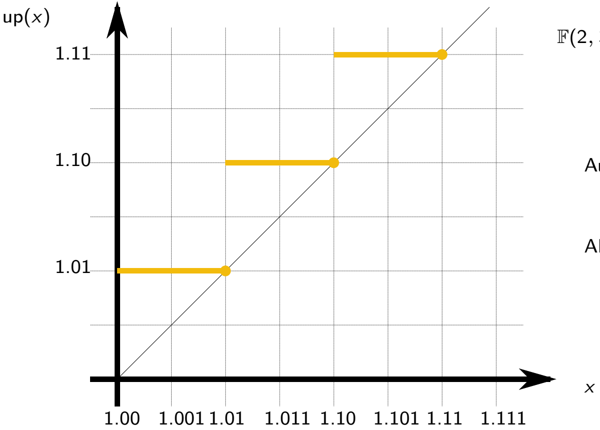
  $\Rightarrow$ Runden des Ergebnisses auf eine Zahl aus $\mathbb{F}$ notwendig
- Eigenschaften einer Rundungsoperation $\square : \mathbb{R} \to \mathbb{F}$
  - $\square x = x$ für $x \in \mathbb{F}$
    Eine Gleitkommazahl wird auf sich selbst gerundet *(Projektivität)*
  - $x \leq y \Rightarrow \square x \leq \square y$
    Die Relation $x \leq y$ bleibt auch nach der Rundung erhalten *(Monotonie)*

Arithmetik auf Gleitkomma-Zahlensystemen

- Abschneiden (truncate)



- Gerichtetes Runden (directed rounding)

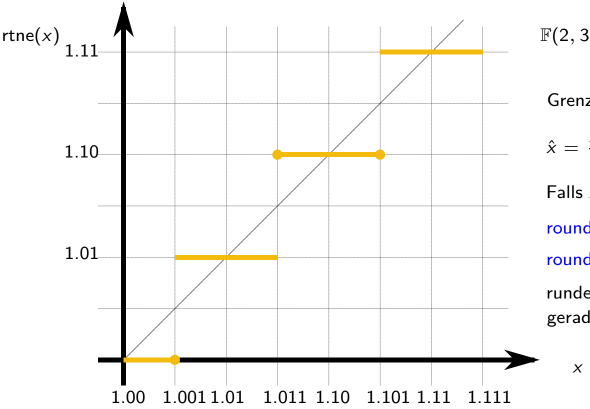


$\mathbb{F}(2, 3, -1, 2, \text{true})$

Aufrunden: $\Box x = \max(x_1, x_2)$

Abrunden: $\Box x = \min(x_1, x_2)$

- Optimale Rundung (round to nearest)



$\mathbb{F}(2, 3, -1, 2, \text{true})$

Grenzpunkt liegt genau in der Mitte

$\hat{x} = \frac{x_1 + x_2}{2}$

Falls $x = \hat{x}$, zwei Möglichkeiten

round away from zero

round to even (auf den Nachbarn runden, dessen letzte Mantissenstelle gerade ist)

Arithmetik auf Gleitkomma-Zahlensystemen

Runden – Beispiel 1

Bsp.: $(-1.626)_{10}$ auf zwei (dezimale) Nachkommastellen runden

- Abschneiden (truncate)
  - $\square x = -1.62$
- Gerichtetes Runden (directed rounding)
  - Aufrunden $\square x = -1.62$
  - Abrunden $\square x = -1.63$
- Optimale Rundung (round to nearest)
  - $x_1 = -1.63$, $x_2 = -1.62$, $\hat{x} = -1.625$, $\square x = -1.63$

Arithmetik auf Gleitkomma-Zahlensystemen

Runden – Beispiel 2

Bsp.: $(1.101)_2 = (1.625)_{10}$ auf zwei (binäre) Nachkommastellen runden

- Abschneiden (truncate)
  - $\Box x = (1.10)_2 = (1.5)_{10}$
- Gerichtetes Runden (directed rounding)
  - Aufrunden $\Box x = (1.11)_2 = (1.75)_{10}$
  - Abrunden $\Box x = (1.10)_2 = (1.5)_{10}$
- Optimale Rundung (round to nearest)
  - $\hat{x} = (1.101)_2,\ x_1 = (1.10)_2,\ x_2 = (1.11)_2$
  - Zahl liegt genau am Grenzpunkt, daher weitere Rundungsregel notwendig
    - *Round away from zero:* $\Box x = (1.11)_2 = (1.75)_{10}$
    - *Round to even:* $\Box x = (1.10)_2 = (1.5)_{10}$

Arithmetik auf Gleitkomma-Zahlensystemen

- Wenn wir programmieren, wollen wir oft prüfen, ob Werte gleich 0 sind
  - `if (x == 0) { /* do something */ }`
  - Da verschiedene reelle Zahlen bei der Rundung nach $\mathbb{F}$ in dieselbe Gleitkommazahl übergehen können, ist es im Allgemeinen keine gute Idee, Gleitkommazahlen auf $= 0$ abzufragen
  - Um festzustellen, ob der exakte Wert eines arithmetischen Ausdrucks positiv ist, muss man verlangen, dass seine Auswertung in $\mathbb{F}$ weit genug von Null entfernt ist:

    $$\text{Ausdruck} \geq \alpha > 0$$

  - Analoge Probleme beim Prüfen, ob zwei Gleitkomma-Zahlen identisch sind

Arithmetik auf Gleitkomma-Zahlensystemen

Rundungsfehler – Beispiel (1 von 2)

- Zu jeder zweistelligen arithmetischen Operation $\circ : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ definiert man die gerundete Operation $\boxed{\circ} : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ mit $x \boxed{\circ} y = \Box(x \circ y)$

- Beispiel: $x = a + b + c$
- Demonstration anhand von $\mathbb{F}(10, 3, -9, 10, \text{false})$
- $a = 1.05 \times 10^3$, $b = c = 4.55 \times 10^0$, optimale Rundung
- Auswertungsreihenfolge von links nach rechts:

$$\begin{aligned}
(a \boxed{+} b) \boxed{+} c &= \Box(a + b) \boxed{+} c \\
&= \Box(\Box(a + b) + c) \\
&= \Box(\Box(1.05 \times 10^3 + 4.55 \times 10^0) + 4.55 \times 10^0) \\
&= \Box(\Box(1.05455 \times 10^3) + 4.55 \times 10^0) \\
&= \Box(1.05 \times 10^3 + 4.55 \times 10^0) \\
&= \Box(1.05455 \times 10^3) \\
&= 1.05 \times 10^3
\end{aligned}$$

Arithmetik auf Gleitkomma-Zahlensystemen

Rundungsfehler – Beispiel (2 von 2)

- Auswertungsreihenfolge von rechts nach links:

$$\begin{aligned}
a \boxed{+} (b \boxed{+} c) &= a \boxed{+} \Box(b + c) \\
&= \Box(a + \Box(b + c)) \\
&= \Box(1.05 \times 10^3 + (\Box(4.55 \times 10^0 + 4.55 \times 10^0))) \\
&= \Box(1.05 \times 10^3 + (\Box(9.10 \times 10^0))) \\
&= \Box(1.05 \times 10^3 + 9.10 \times 10^0) \\
&= \Box(1.0591 \times 10^3) \\
&= 1.06 \times 10^3
\end{aligned}$$

$\Rightarrow$ Auswertungsreihenfolge hat Einfluss auf das Ergebnis!

Arithmetik auf Gleitkomma-Zahlensystemen

Pseudo-Arithmetik

- Keine Gültigkeit der Assoziativität

$$a \boxed{+} (b \boxed{+} c) \neq (a \boxed{+} b) \boxed{+} c$$

$$a \boxed{\times} (b \boxed{\times} c) \neq (a \boxed{\times} b) \boxed{\times} c$$

- Keine Gültigkeit der Distributivität

$$a \boxed{\times} (b \boxed{+} c) \neq (a \boxed{\times} b) \boxed{+} (a \boxed{\times} c)$$

- Aber wegen

$$a \boxed{+} b = \Box(a + b) = \Box(b + a) = b \boxed{+} a$$

und

$$a \boxed{\times} b = \Box(a \cdot b) = \Box(b \cdot a) = b \boxed{\times} a$$

$\Rightarrow$ Kommutativität

Arithmetik auf Gleitkomma-Zahlensystemen

Iterative Summation

- Berechnung einer Näherung der unendlichen Summe

$$\sum_{i \geq 1} \frac{1}{i^2}$$

- Indem man die Summanden für $i = 1, 2, 3, \ldots$ aufaddiert
- Die Zwischensummen werden immer größer und die Summanden immer kleiner
- Man gelangt zu einem bestimmten N, ab dem

$$\sum_{i=1}^{N} \frac{1}{i^2} \boxed{+} \frac{1}{(N+1)^2} = \sum_{i=1}^{N} \frac{1}{i^2}$$

  d.h., dass die Summe ihren Wert nicht mehr ändert.
- Beginnt man mit $i = N, N-1, N-2, \ldots$, so erhält man sogar einen genaueren Näherungswert $\left(\frac{\pi^2}{6}\right)$

Genauigkeitsbetrachtungen

- Subtraktion zweier betragsmäßig annähernd gleich großer Zahlen:
  - Auslöschung
  - Die vorderen übereinstimmenden Mantissenstellen der beiden Operanden heben einander auf
  - Damit werden Ungenauigkeiten an hinteren (weniger wichtigen) Stellen relevanter
- Rechnen mit exakten Werten ($x \in \mathbb{R}$):
  - Die nach der Auslöschung im Ergebnis verbleibenden hinteren Stellen sind unverfälscht
- Rechnen mit gerundeten Operanden ($x \in \mathbb{F}$)
  - Die nach der Auslöschung im Ergebnis verbleibenden hinteren Stellen sind verfälscht
  - Können einen großen relativen Fehler haben

Genauigkeitsbetrachtungen

Rundungsfehler

- Es gibt mehrere Möglichkeiten, Rundungsfehler zu analysieren
- Absoluter Rundungsfehler $\varepsilon(x) = |\Box\, x - x|$
- Relativer Rundungsfehler $\rho(x) = \frac{|\Box x - x|}{|x|} = \frac{\varepsilon(x)}{|x|}$

Bsp.: $(1.101)_2 = (1.625)_{10}$ auf zwei (binäre) Nachkommastellen

- *Round away from zero:* $\Box x = (1.11)_2 = (1.75)_{10}$
  - $\varepsilon(x) = |(1.75)_{10} - (1.625)_{10}| = (0.125)_{10}$
- *Round to even:* $\Box x = (1.10)_2 = (1.5)_{10}$
  - $\varepsilon(x) = |(1.5_{10} - (1.635)_{10}| = |(-0.125)_{10}| = (0.125)_{10}$

Genauigkeitsbetrachtungen

- Bsp.: $x^2 - y^2$ und $(x-y)\times(x+y)$
  $x = 10.1, y = 9.99$, optimales Runden auf 3 Stellen genau
- Exakt: $x^2 - y^2 = 102.01 - 99.8001 = 2.2099$
- Gutartige Auslöschung

$$(x \boxed{-} y) \boxed{\times} (x \boxed{+} y) = (\square(0.11)) \boxed{\times} (\square(20.09))$$
$$= 0.11 \boxed{\times} 20.1 = \square(2.211) = 2.21$$

  - Relativer Rundungsfehler $\rho(2.21) = \frac{2.21-2.2099}{2.2099} \approx 4 \times 10^{-5}$
- Katastrophale Auslöschung

$$(x^{\boxed{2}} \boxed{-} y^{\boxed{2}}) = (\square(102.01)) \boxed{-} (\square(99.8001)) = 102 \boxed{-} 99.8 = 2.2$$

  - Relativer Rundungsfehler $\rho(2.2) = \frac{2.2-2.2099}{2.2099} \approx 4 \times 10^{-3} \gg 4 \times 10^{-5}$

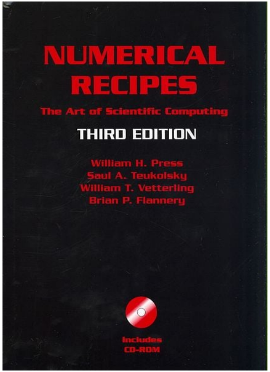
Genauigkeitsbetrachtungen

- Aufgrund von Rundungsfehlern Abweichung zwischen im Computer implementierten arithmetischen Operationen von zu Grunde liegenden mathematisch exakten Operationen

$\Rightarrow$ Jedes Zwischenergebnis einer numerischen Berechnung kann vom exakten Ergebnis abweichen

- Zwischenergebnisse sind Operanden für nachfolgende Rechenschritte
$\Rightarrow$ Nachfolgende Rechenschritte mit verfälschten Argumenten

$\Rightarrow$ **Fehlerfortpflanzung**



Arithmetik auf Gleitkomma-Zahlensystemen

- Wie implementiert man Addition, Multiplikation, etc. um Rundungsfehler klein zu halten?
- Ziel ist es, die Grundrechnungsarten (in Hardware) so zu implementieren, dass gilt:

$$a \boxed{\circ} b = \square(a \circ b)$$

  - **Das ist nicht selbstverständlich!**
  - Auf der rechten Seite der Gleichung steht die Operation $\circ$, die für reelle Zahlen definiert ist
  - Der Computer hat aber nur Zahlen $\in \mathbb{F}$ und Operationen über $\mathbb{F}$ zur Verfügung
  - Er muss es also schaffen, dass obige Gleichung gilt, obwohl er nur in $\mathbb{F}$ rechnen kann
- Man benötigt dazu **zusätzliche** Mantissenstellen
  - Das Beispiel auf den folgenden Seiten zeigt, dass das so ist und wieviele Stellen man dafür braucht

Arithmetik auf Gleitkomma-Zahlensystemen

1. Exponenten angleichen:
   - Größeren Exponenten bestimmen
   - Kleineren Exponenten an den größeren anpassen
   - Entsprechende Mantisse verschieben
2. Mantissen addieren
3. Normalisieren
4. Runden

Arithmetik auf Gleitkomma-Zahlensystemen

Bsp. 1: $2.15 \times 10^{12} - 1.25 \times 10^{-5}$

- Rechnen in $\mathbb{R}$ (Exponentialschreibweise, Zehnersystem)
  1. Exponenten angleichen: $1.25 \times 10^{-5} = 0.00000000000000000125 \times 10^{12}$
  2. Mantissen addieren

$$
\begin{array}{rlcl}
 & 2.15 & \times & 10^{12} \\
- & 0.0000\,0000\,0000\,0000\,125 & \times & 10^{12} \\
\hline
 & 2.1499\,9999\,9999\,9999\,875 & \times & 10^{12}
\end{array}
$$

  3. Normalisieren entfällt
  4. Runden (3 Stellen Mantisse): 2.15×10^{12}

- Rechnen in $\mathbb{F}$ ($b = 10$, $p = 3$)
  - Vor Berechnung abschneiden, sodass Mantisse $p = 3$ Stellen hat?

$$
\begin{array}{rlcl}
 & 2.15 & \times & 10^{12} \\
- & 0.00 & \times & 10^{12} \\
\hline
 & 2.15 & \times & 10^{12}
\end{array}
$$

☺?

Runden:
round to nearest mit
round to even

Arithmetik auf Gleitkomma-Zahlensystemen

Bsp. 2: $10.1 - 9.93$

- Rechnen in $\mathbb{R}$ (Exponentialschreibweise, Zehnersystem)
  - $10.1 \times 10^0 = 1.01 \times 10^1$
  - $9.93 \times 10^0 = 0.993 \times 10^1$

$$\begin{array}{rlcl} & 1.01 & \times & 10^1 \\ - & 0.993 & \times & 10^1 \\ \hline & \mathbf{0.017} & \times & \mathbf{10^1} \end{array} \quad = \mathbf{1.7 \times 10^{-1}}$$

- Rechnen in $\mathbb{F}$ ($b = 10$, $p = 3$) vor Berechnung abschneiden?

$$\begin{array}{rlcl} & 1.01 & \times & 10^1 \\ - & 0.99 & \times & 10^1 \\ \hline & \mathbf{0.02} & \times & \mathbf{10^1} \end{array} \quad = \mathbf{2.0 \times 10^{-1}}$$ ☹

$\Rightarrow$ Zusätzliche Stellen notwendig! Wie viele?

Runden: round to nearest mit round to even

Arithmetik auf Gleitkomma-Zahlensystemen

- Eine zusätzliche Stelle verwenden: Guard Digit (g)
- Bsp.: $1.01 \times 10^1 - 0.993 \times 10^1$
  - Ergebnis in $\mathbb{R}$ berechnen, dann runden (3 Stellen Mantisse):

$$\begin{array}{rrcl} & 1.01 & \times & 10^1 \\ - & 0.993 & \times & 10^1 \\ \hline & 0.017 & \times & 10^1 \end{array} \quad = \mathbf{0.17 \times 10^0}$$

  - In $\mathbb{F}$: 3 Stellen Mantisse + Guard Digit, Restliches abschneiden, dann Ergebnis berechnen
    - Wir rechnen jetzt mit 0.993, benutzen also $p + 1$ Stellen für die Mantisse
    - Die zusätzliche Stelle ist die Guard Digit

$$\begin{array}{rrcl} & 1.01 & \times & 10^1 \\ - & 0.993 & \times & 10^1 \\ \hline & 0.017 & \times & 10^1 \end{array} \quad = \mathbf{0.17 \times 10^0}$$ ☺?

Runden:
round to nearest mit
round to even

Arithmetik auf Gleitkomma-Zahlensystemen

- Bsp.: $1.01 \times 10^2 - 3.76 \times 10^0$
  - Ergebnis in $\mathbb{R}$ berechnen, dann runden (3 Stellen Mantisse):

$$\begin{array}{rlcl} & 1.01 & \times & 10^2 \\ - & 0.0376 & \times & 10^2 \\ \hline & 0.9724 & \times & 10^2 \end{array} \quad \approx \mathbf{9.72 \times 10^1}$$

  - In $\mathbb{F}$: 3 Stellen Mantisse + Guard Digit, Restliches abschneiden, dann Ergebnis berechnen

$$\begin{array}{rlcl} & 1.01 & \times & 10^2 \\ - & 0.037 & \times & 10^2 \\ \hline & 0.973 & \times & 10^2 \end{array} \quad \approx \mathbf{9.73 \times 10^1}$$ ☹ ☹

  - Oben: Haben 9.724 abgerundet auf 9.72, unten: uns fehlt eine Nachkommastelle um korrekt runden zu können

$\Rightarrow$ Eine zusätzliche Stelle reicht nicht ...

Runden:
round to nearest mit
round to even

Arithmetik auf Gleitkomma-Zahlensystemen

- Wir verwenden noch eine zusätzliche Stelle: Round Digit

- Bsp.: $1.01 \times 10^2 - 3.76 \times 10^0$
  - Ergebnis in $\mathbb{R}$ berechnen, dann runden (3 Stellen Mantisse):

$$\begin{array}{rlcl} & 1.01 & \times & 10^2 \\ - & 0.0376 & \times & 10^2 \\ \hline & 0.9724 & \times & 10^2 \end{array} \quad \approx \mathbf{9.72} \times \mathbf{10^1}$$

  - In $\mathbb{F}$: 3 Stellen Mantisse + Guard Digit + Round Digit, Restliches abschneiden, dann Ergebnis berechnen

$$\begin{array}{rlcl} & 1.01 & \times & 10^2 \\ - & 0.0376 & \times & 10^2 \\ \hline & 0.9724 & \times & 10^2 \end{array} \quad \approx \mathbf{9.72} \times \mathbf{10^1}$$ ☺ ☺??

Runden:
round to nearest mit
round to even

Arithmetik auf Gleitkomma-Zahlensystemen

- Bsp.: $4.5674 \times 10^0 + 2.5003 \times 10^{-4}$
  - Ergebnis in $\mathbb{R}$ berechnen, dann runden (5 Stellen Mantisse):

$$\begin{array}{rll} & 4.5674 & \times\ 10^0 \\ + & 0.00025003 & \times\ 10^0 \\ \hline & 4.56765003 & \times\ 10^0 \end{array} \quad \approx \mathbf{4.5677 \times 10^0}$$

  - In $\mathbb{F}$: 5 Stellen Mantisse + Guard Digit + Round Digit, Restliches abschneiden, dann Ergebnis berechnen

$$\begin{array}{rll} & 4.5674 & \times\ 10^0 \\ + & 0.000250 & \times\ 10^0 \\ \hline & 4.567650 & \times\ 10^0 \end{array} \quad \approx \mathbf{4.5676 \times 10^0}$$ ☹ ☹

  - Wir runden falsch wegen:
    - „Round to even"-Regel
    - Weil wir denken, dass wir den Grenzfall .5 haben, da wir vergessen haben, dass es eigentlich 0.5003 war

⇒ Zwei zusätzliche Stellen reichen nicht ...

Runden:
round to nearest mit
round to even

Arithmetik auf Gleitkomma-Zahlensystemen

- Zwei zusätzliche Stellen reichen nicht, **aber drei schon**, damit man korrekt runden kann!
  - Dritte zusätzliche Stelle: Sticky Bit
  - **Verändert sich nicht mehr, sobald es einmal den Wert 1 angenommen hat!**
  - Kommen rechts vom Round Digit noch Stellen $\neq 0$, die wir abgeschnitten haben?
- So wird das Sticky Bit gesetzt:
  - true . . . es gibt rechts vom Round Digit noch Stellen $\neq 0$
  - false . . . es gibt rechts vom Round Digit **keine** Stellen $\neq 0$
  - Wird mit 1 (true) bzw. 0 (false) codiert
- Bei arithmetischen Berechnungen:
  - grs werden in Berechnungen einbezogen
    - grs = guard round sticky
  - D.h. Addition und Subtraktion beginnen beim Sticky Bit

Arithmetik auf Gleitkomma-Zahlensystemen

1. Angleichung der Exponenten
2. Mantissen addieren/subtrahieren
   - Vorzeichen gleich: Addition
   - Vorzeichen ungleich: Subtraktion
3. Ergebnis normalisieren
4. Runden (Guard Digit, Round Digit, Sticky Bit)

- Regeln für optimale Rundung:

G	R	S	Ergebnis / Mantisse
0	x	x	Unverändert (abrunden)
1	1	x	Ergebnis += 1 (aufrunden)
1	0	0	**Weitere** Rundungsregel für Grenzfall nötig!
1	0	1	Ergebnis += 1 (aufrunden)

Arithmetik auf Gleitkomma-Zahlensystemen

- Spezialfälle für das Runden mit grs $= 100$
- Optimale Rundung / round to even

G	R	S	Ergebnis / Mantisse
1	0	0	Wenn $lsb = 0 \rightarrow$ unverändert Wenn $lsb = 1 \rightarrow$ += 1

- Optimale Rundung / round away from zero

G	R	S	Ergebnis / Mantisse
1	0	0	+= 1

Arithmetik-Beispiele nach (modifiziertem) IEEE 754

Bsp. Addition und Subtraktion:

- $A = (5.58)_{10}$ und $B = (62.27)_{10}$
- Umrechnung ins Gleitkommaformat nach modifiziertem (da Format gekürzt) IEEE 754-Standard:
  - 5 Bit Exponent
  - Mantisse: implizites erstes Bit + 10 Bit
  - Exzess $= (15)_{10} = (01111)_2$
  - Runden durch Abschneiden
- $A + B, A - B$ mit optimaler Rundung mit „round to even“

Arithmetik-Beispiele nach (modifiziertem) IEEE 754

Umrechnung ins Gleitkommaformat:

- Zahl $A = (5.58)_{10}$
  - Umwandeln ins Binärsystem

    $$(5.58)_{10} = (101.1001010)_2$$

  - Normalisieren

    $$(101.1001010)_2 \times 2^0 = (1.011001010)_2 \times 2^2$$

  - Exponenten berechnen

    $$\begin{array}{cccccccll} & 0 & 1 & 1 & 1 & 1 & = & (15)_{10} & \text{Exzess} \\ + & 0 & 0 & 0 & 1 & 0 & = & (2)_{10} & \text{Exponent der normalisierten Darstellung} \\ \hline & 1 & 0 & 0 & 0 & 1 & = & (17)_{10} & \text{Exponent in Exzessdarstellung} \end{array}$$

  - Vorzeichenbit: 0

Arithmetik-Beispiele nach (modifiziertem) IEEE 754

- Zahl $B = (62.27)_{10}$
  - Umwandeln ins Binärsystem

    $$(62.27)_{10} = (111110.0100)_2$$

  - Normalisieren

    $$(111110.0100)_2 \times 2^0 = (1.111100100)_2 \times 2^5$$

  - Exponenten berechnen

$$\begin{array}{cccccccll} & 0 & 1 & 1 & 1 & 1 & = & (15)_{10} & \text{Exzess} \\ + & 0 & 0 & 1 & 0 & 1 & = & (5)_{10} & \text{Exponent der normalisierten Darstellung} \\ \hline & 1 & 0 & 1 & 0 & 0 & = & (20)_{10} & \text{Exponent in Exzessdarstellung} \end{array}$$

  - Vorzeichenbit: 0

	VZ	Exponent					Mantisse									
A	0	1	0	0	0	1	0	1	1	0	0	1	0	1	0	0
B	0	1	0	1	0	0	1	1	1	1	0	0	1	0	0	0

Implizites 1. Bit!

Arithmetik-Beispiele nach (modifiziertem) IEEE 754

Addition $A + B$

1. Angleichung der Exponenten
   - $e_A < e_B \Rightarrow$ Exponent von A an Exponent von B anpassen
   - „Hinausgeschobene“ Bits füllen Guard/Round/Sticky auf

	VZ	Exponent					/	Mantisse									
A	0	1	0	0	0	1	1	0	1	1	0	0	1	0	1	0	0
B	0	1	0	1	0	0	1	1	1	1	1	0	0	1	0	0	0

Implizites 1. Bit (/) ist nur gedacht und wird nicht gespeichert

Exp. $+3$ um 3 Bit „nach hinten geschoben“

	VZ	Exponent					/	Mantisse										G	R	S
A	0	1	0	**1**	**0**	**0**	**0**	**0**	**0**	**1**	0	1	1	0	0	1	0	1	0	0
B	0	1	0	1	0	0	1	1	1	1	1	0	0	1	0	0	0	0	0	0

Durch das Schieben wird das implizite Bit nun explizit

Arithmetik-Beispiele nach (modifiziertem) IEEE 754

Addition $A + B$

2. Mantissen addieren

	VZ	Exponent					/	Mantisse										G	R	S
A	0	1	0	**1**	**0**	**0**	**0**	**0**	**0**	**1**	0	1	1	0	0	1	0	1	0	0
B	0	1	0	1	0	0	1	1	1	1	1	0	0	1	0	0	0	0	0	0
	0	1	0	1	0	0	10	0	0	0	1	1	1	1	0	1	0	1	0	0

3. Ergebnis normalisieren

VZ	Exponent					/	Mantisse										G	R	S
0	1	0	1	0	0	10	0	0	0	1	1	1	1	0	1	0	1	0	0

VZ	Exponent					/	Mantisse										G	R	S
0	1	0	1	0	1	1	0	0	0	0	1	1	1	1	0	1	0	1	0

Exp. $+1$ um 1 Bit „nach hinten geschoben“

Arithmetik-Beispiele nach (modifiziertem) IEEE 754

Addition $A + B$

4. Runden

VZ	Exponent					/	Mantisse										G	R	S
0	1	0	1	0	1	1	0	0	0	0	1	1	1	1	0	1	0	1	0

$\downarrow\downarrow$ Runden

G	R	S	Ergebnis / Mantisse
0	x	x	unverändert
1	1	x	Ergebnis += 1
1	0	0	**weitere** Rundungsregel für Grenzfall nötig!
1	0	1	Ergebnis += 1

VZ	Exponent					/	Mantisse									
0	1	0	1	0	1	1	0	0	0	0	1	1	1	1	0	1

Ergebnis:

- In der (modifizierten) IEEE-Schreibweise (ohne implizites 1. Bit) 0 10101 0000111101
- Entspricht der Zahl $(1.0000111101)_2 \times 2^6$

Arithmetik-Beispiele nach (modifiziertem) IEEE 754

Subtraktion $A - B$

1. Angleichung der Exponenten
   - $e_A < e_B \rightarrow$ Exponent von A an Exponent von B anpassen
   - „Hinausgeschobene“ Bits füllen Guard/Round/Sticky auf

	VZ	Exponent					/	Mantissa									
A	0	1	0	0	0	1	1	0	1	1	0	0	1	0	1	0	0
B	0	1	0	1	0	0	1	1	1	1	1	0	0	1	0	0	0

Implizites 1. Bit (/) ist nur gedacht und wird nicht gespeichert

Exp. +3 um 3 Bit „nach hinten geschoben“

	VZ	Exponent					/	Mantisse										G	R	S
A	0	1	0	**1**	**0**	**0**	**0**	**0**	**0**	**1**	0	1	1	0	0	1	0	1	0	0
B	0	1	0	1	0	0	1	1	1	1	1	0	0	1	0	0	0	0	0	0

Arithmetik-Beispiele nach (modifiziertem) IEEE 754

Subtraktion $A - B$

2. Mantissen subtrahieren
   - $|B| > |A| \rightarrow$ wir wissen, dass Ergebnis negativ sein wird
   - Trick um uns Rechnen über 0 zu ersparen: berechnen $B - A$ und setzen Ergebnis negativ $\Rightarrow A - B = -(B - A)$

	VZ	Exponent					/	Mantisse										G	R	S
B	0	1	0	1	0	0	1	1	1	1	1	0	0	1	0	0	0	0	0	0
$-A$	0	1	0	1	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	0
	1	1	0	1	0	0	1	1	1	0	0	0	1	0	1	0	1	1	0	0

3. Ergebnis normalisieren
   - Ist bereits normalisiert!

VZ	Exponent					/	Mantisse										G	R	S
1	1	0	1	0	0	1	1	1	0	0	0	1	0	1	0	1	1	0	0

Arithmetik-Beispiele nach (modifiziertem) IEEE 754

4. Runden

VZ	Exponent					/	Mantisse										G	R	S
1	1	0	1	0	0	1	1	1	0	0	0	1	0	1	0	1	1	0	0

⇊ Runden

VZ	Exponent					/	Mantisse									
1	1	0	1	0	0	1	1	1	0	0	0	1	0	1	**1**	**0**

G	R	S	Ergebnis / Mantisse
0	x	x	unverändert
1	1	x	Ergebnis += 1
1	0	0	lsb = 0 → unverändert lsb = 1 → Erg. PlusEins
1	0	1	Ergebnis += 1

Ergebnis:

- In der (modifizierten) IEEE-Schreibweise (**ohne** implizites 1. Bit) 1 10100 1100010110
- Entspricht der Zahl $(-1.1100010110)_2 \times 2^5$

Arithmetik-Beispiele nach (modifiziertem) IEEE 754

Bei Subtraktion prinzipiell zwei Möglichkeiten:

1. Für alle Basen gültig:
   - Sticky Bit wird nur verglichen (übernommen), nicht subtrahiert
   - Subtraktion „beginnt“ bei Round Digit
   - Erfordert Fallunterscheidung bei Rundung:
     - Z.B. Optimale Rundung bei 101 nur bei gleichem VZ: $m += 1$
     - bei Nachnormalisieren: Sticky Bit bleibt, 0 wird nachgeschoben, Runden

2. Im Binärsystem vereinfacht:
   - Sticky Bit wird einfach mitgerechnet
   - Subtraktion „beginnt“ schon bei Sticky Bit
   - Erspart Fallunterscheidung bei Rundung:
     - Z.B. Optimale Rundung bei 101 immer: $m += 1$
     - Bei Nachnormalisieren: Sticky Bit weiterschieben, wird 0, Runden

Arithmetik-Beispiele nach (modifiziertem) IEEE 754

1. Angleichung der Exponenten
   - $e_C > e_D \rightarrow$ Exponent von D an Exponent von C anpassen
   - „Hinausgeschobene“ Bits füllen Guard/Round/Sticky auf

	VZ	Exponent					/	Mantisse									
C	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1
D	1	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1

Implizites 1. Bit (/) ist nur gedacht und wird nicht gespeichert

Exp. +10 um 10 Bit „nach hinten geschoben“

	VZ	Exponent					/	Mantisse										G	R	S
C	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0
D	1	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	1	0	0	1

Arithmetik-Beispiele nach (modifiziertem) IEEE 754

$C + D$ (2 von 3)

2. Mantissen subtrahieren
   - $|C| > |D| \rightarrow$ wir wissen, dass Ergebnis positiv sein wird

	VZ	Exponent					/	Mantisse										G	R	S
C	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0
D	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
	0	1	1	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

3. Ergebnis normalisieren
   - Mantisse eine Stelle nach links verschieben
   - Exponent dekrementieren
   - Sticky Bit auf 0 setzen

VZ	Exponent					/	Mantisse										G	R	S
0	1	1	0	**0**	**1**	**1**	1	1	1	1	1	1	1	1	1	1	1	1	**0**

Arithmetik-Beispiele nach (modifiziertem) IEEE 754

$C + D$ (3 von 3)

4. Runden

VZ Exponent / Mantisse G R S

0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

$\downarrow\downarrow$ Runden

VZ Exponent / Mantisse

0	1	1	0	0	1	10	0	0	0	0	0	0	0	0	0	0

G	R	S	Ergebnis / Mantisse
0	x	x	unverändert
1	1	x	Ergebnis += 1
1	0	0	lsb $= 0 \rightarrow$ unverändert lsb $= 1 \rightarrow$ Erg. += 1
1	0	1	Ergebnis += 1

5. Normalisieren

VZ Exponent / Mantisse

0	1	1	0	**1**	**0**	1	0	0	0	0	0	0	0	0	0	0

Ergebnis:

- In der (modifizierten) IEEE-Schreibweise 0 11010 0000000000

Arithmetik auf Gleitkomma-Zahlensystemen

Multiplikation

1. Multiplikation der Mantissen
2. Summe der Exponenten
3. Normalisieren
4. Runden

Arithmetik-Beispiele nach (modifiziertem) IEEE 754

Multiplikation $A \cdot B$ (1 von 4)

	VZ	Exponent					/	Mantisse									
A	0	1	0	0	0	1	1	0	1	1	0	0	1	0	1	0	0
B	0	1	0	1	0	0	1	1	1	1	1	0	0	1	0	0	0

1. Multiplikation der Mantissen

```
1  0 1 1 0 0 1 0 1 0 0 · 1 1 1 1 1 0 0 1 0 0 0
-------------------------------------------------
1  0 1 1 0 0 1 0 1 0 0 0
   1 0 1 1 0 0 1 0 1 0 0 0
     1 0 1 1 0 0 1 0 1 0 0 0
       1 0 1 1 0 0 1 0 1 0 0 0
         1 0 1 1 0 0 1 0 1 0 0 0
               1 0 1 1 0 0 1 0 1 0 0 0
                   0 0 0 0 0 0 0 0 0 0 0 0
-------------------------------------------------
10 1 0 1 1 0 1 1 0 0 1 1 1 1 0 1 0 0 0 0 0
```

Arithmetik-Beispiele nach (modifiziertem) IEEE 754

Multiplikation $A \cdot B$ (2 von 4)

	VZ	Exponent					/	Mantisse									
A	0	1	0	0	0	1	1	0	1	1	0	0	1	0	1	0	0
B	0	1	0	1	0	0	1	1	1	1	1	0	0	1	0	0	0

2. Summe der Exponenten: $(\mathrm{Exp}(A) + \mathrm{Exp}(B))_e = (\mathrm{Exp}(A))_e + (\mathrm{Exp}(B))_e - e$

$$
\begin{array}{cccccccll}
 & 1 & 0 & 0 & 0 & 1 & = & (17)_{10} & \mathrm{Exp}(A)_e \\
- & 0 & 1 & 1 & 1 & 1 & = & (15)_{10} & e \\
\hline
 & 0 & 0 & 0 & 1 & 0 & = & (2)_{10} & \mathrm{Exp}(A)
\end{array}
$$

$$
\begin{array}{cccccccll}
 & 1 & 0 & 1 & 0 & 0 & = & (20)_{10} & \mathrm{Exp}(B)_e \\
+ & 0 & 0 & 0 & 1 & 0 & = & (2)_{10} & \mathrm{Exp}(A) \\
\hline
 & 1 & 0 & 1 & 1 & 0 & = & (22)_{10} & (\mathrm{Exp}(A) + \mathrm{Exp}(B))_e
\end{array}
$$

Arithmetik-Beispiele nach (modifiziertem) IEEE 754

Multiplikation $A \cdot B$ (3 von 4)

3. Normalisieren

VZ	Exponent					/	Mantisse										G	R	S
0	1	0	1	1	0	10	1	0	1	1	0	1	1	0	0	1	1	1	1

$\downarrow\downarrow$

VZ	Exponent					/	Mantisse										G	R	S
0	1	0	1	1	**1**	1	0	1	0	1	1	0	1	1	0	0	1	1	1

Exp. +1 um 1 Bit „nach hinten geschoben“

Die ersten 14 Stellen der Multiplikation

Arithmetik-Beispiele nach (modifiziertem) IEEE 754

Multiplikation $A \cdot B$ (4 von 4)

4. Runden

VZ	Exponent					/	Mantisse										G	R	S
0	1	0	1	1	**1**	1	0	1	0	1	1	0	1	1	0	0	1	1	1

↓↓ Runden

VZ	Exponent					/	Mantisse									
0	1	0	1	1	**1**	1	0	1	0	1	1	0	1	1	0	**1**

G	R	S	Ergebnis / Mantisse
0	x	x	unverändert
1	1	x	Ergebnis += 1
1	0	0	**weitere** Rundungsregel für Grenzfall nötig!
1	0	1	Ergebnis += 1

Ergebnis:

- In der (modifizierten) IEEE-Schreibweise (ohne implizites 1. Bit) 0 10111 0101101101
- Entspricht der Zahl $(1.0101101101)_2 \times 2^8$