Einführung in die Technische Informatik

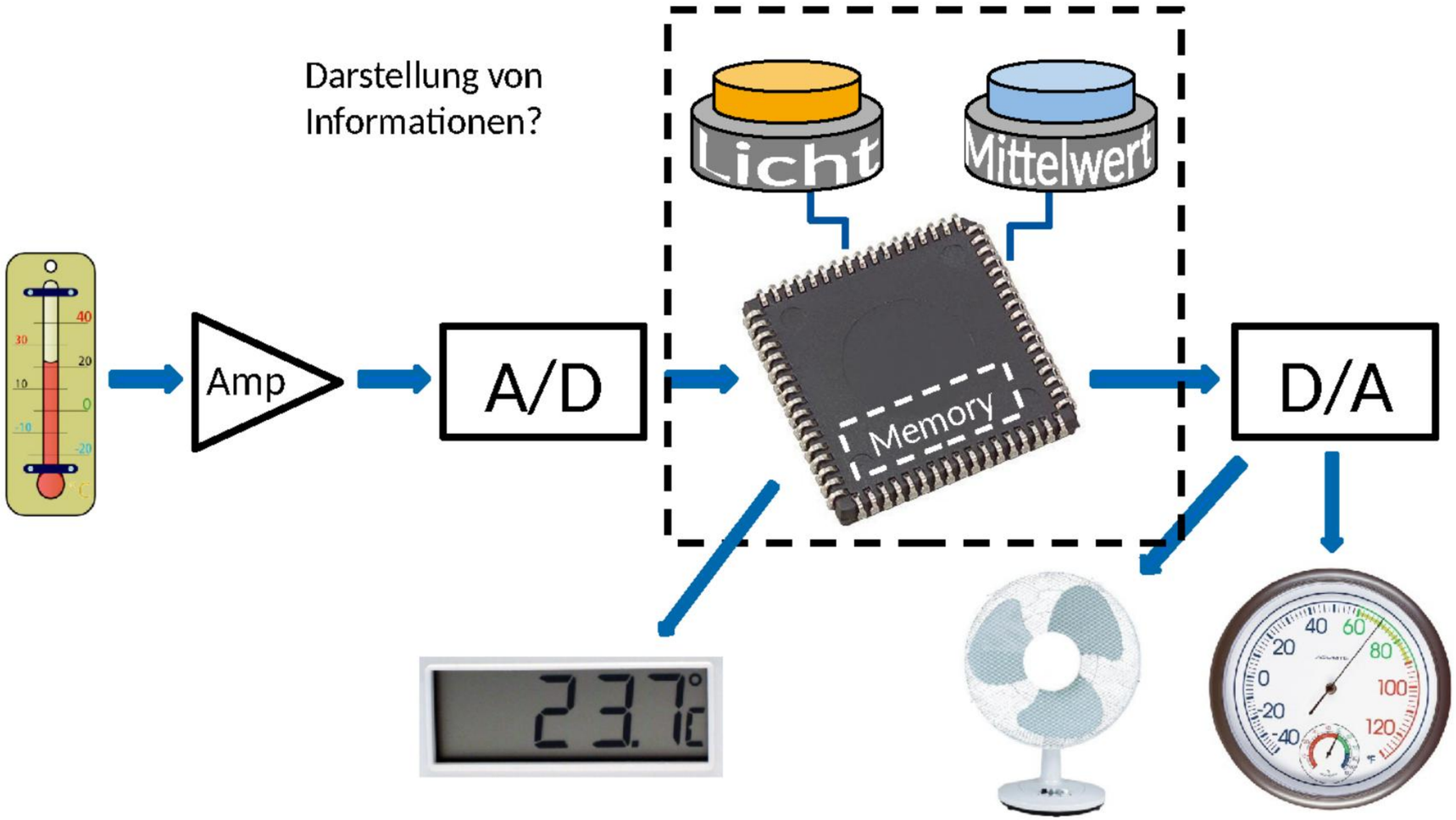
Prof. Jian-Jia Chen | Dr.-Ing. Christian Hakert | Dr. rer. nat. Marcus Völker

RWTH Aachen University

Kapitel 1: Zahlendarstellung

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Anwendungsbeispiel



Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Abschnitt 1.1

- Zahlendarstellungen
- b-adische Darstellung natürlicher Zahlen

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Zahlendarstellungen

Endliches Alphabet:

- z.B. im Dezimalsystem: $\sum_{10} = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9$
- **allgemein:** $\sum_{b} = 0, 1, \dots, b-1$, wobei b **Basis** genannt wird

b-adische Zahlen mit **endlicher Wortlänge** *n*: $\sum_{b}^{n}$

- **Beispiel:** $00456 \in \sum_{10}^{5}$

Wichtige Zahlensysteme in der Informatik:

- **Dual-/Binärsystem:** $\sum_{2} = 0, 1$
- **Oktalsystem:** $\sum_{8} = 0, 1, 2, 3, 4, 5, 6, 7$
- **Hexadezimalsystem:** $\sum_{16} = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F$

Wahl der Basis

- Warum keine Basis 10?
  - Natürliche Darstellung für Finanztransaktionen, etwa € 1.20
  - Verbreitet in Wissenschaft, z.B. $6.6206 \bullet 10^{34}$
- Aber elektronische Bearbeitung schwierig
  - Speicherung anspruchsvoll, der ENIAC benutzte 10 Röhren pro Ziffer
  - Übertragung von 10 Signalniveaus auf einer Leitung benötigt hohe Präzision
  - Implementierung von Addition & Co ist schwierig

b-adische Darstellung natürlicher Zahlen

- Sei $b \in N$ mit $b > 1$. Dann ist jede natürliche Zahl z mit $0 \leq z \leq b^n - 1$
- (und $n \in N$) eindeutig als Wort der Länge n über $\sum_b^n$ darstellbar durch

$$z = \sum_{i=0}^{n-1} z_i b^i$$

mit $z_i \in \sum_b = \{0, 1, \dots, b-1\}$ für $i = 0, 1, \dots, n-1$.

- Als vereinfachende Schreibweise ist dabei die folgende
- **Ziffernschreibweise** üblich:

$$z = (z_{n-1} z_{n-2} \dots z_1 z_0)_b$$

Wichtiger Spezialfall: $b = 2$ („**Binärdarstellung**“ natürlicher Zahlen)

Beispiel: Binäre Zahlendarstellung

- Betrachte $z = (0110)_2$
- Als natürliche Zahl interpretiert als:

$$z = \sum_{i=0}^{3} 2^i \cdot z_i = 2^0 \cdot 0 + 2^1 \cdot 1 + 2^2 \cdot 1 + 2^3 \cdot 0 = 6$$

- Größte darstellbare Zahl mit n Bits:

$$z_{max} = \sum_{i=0}^{n-1} 2^i = 2^n - 1$$

- Beispiel: z ist 3-stellig $\Rightarrow z_{max} = (111)_2 = (7)_{10} = 2^3 - 1$

Hexadezimalsystem

- Basis $b = 16$, d.h. wir benötigen **16 verschiedene Ziffern**
- Die Ziffern 0 bis 9 reichen nur für die Wertigkeiten 0 bis 9
- Ab der Wertigkeit 10 wird mit **Buchstaben** weitergezählt:

Ziffer	**0**	**1**	**...**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
Wert	0	1	...	8	9	10	11	12	13	14	15

- Wert einer Hexadezimalzahl: $z = \sum_{i=0}^{n-1} z_i \cdot 16^i$
- Eine Hexadezimalziffer entspricht genau **4 Bit**, da $16 = 2^4$
  - Kompakte Schreibweise für Binärzahlen, z.B. $(1111)_2 = (\mathrm{F})_{16}$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Horner-Schema

- Schema zur Umrechnung zwischen Zahlensystemen mit unterschiedlichen Basen

$$w = \sum_{i=0}^{n-1} z_i b^i = z_0 + b \cdot (z_1 + b \cdot (z_2 + \ldots (b \cdot (z_{n-3} + b \cdot (z_{n-2} + b \cdot z_{n-1})) \ldots))$$

- Ermittle Sequenz durch Division mit Rest:

$\frac{w}{b} = z_1 + b \cdot (z_2 + \ldots (b \cdot (z_{n-3} + b \cdot (z_{n-2} + b \cdot z_{n-1})) \ldots))$ Rest z_0
$\frac{w_1}{b} = z_2 + \ldots (b \cdot (z_{n-3} + b \cdot (z_{n-2} + b \cdot z_{n-1})) \ldots))$ Rest z_1
$\frac{w_{n-2}}{b} = z_{n-1}$ Rest z_{n-2}
$\frac{w_{n-1}}{b} = 0$ Rest z_{n-1}

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Beispiel: Horner-Schema

Beispiel: $(2595)_{10}$ nach Basis 11

$$\frac{(2595)_{10}}{11} = (235)_{10} \qquad \text{Rest: } (10)_{10} = (A)_{11}$$

$$\frac{(235)_{10}}{11} = (21)_{10} \qquad \text{Rest: } (4)_{10} = (4)_{11}$$

$$\frac{(21)_{10}}{11} = (1)_{10} \qquad \text{Rest: } (10)_{10} = (A)_{11}$$

$$\frac{(1)_{10}}{11} = (0)_{10} \qquad \text{Rest: } (1)_{10} = (1)_{11}$$

$$(2595)_{10} = (1A4A)_{11}$$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Divisionsmethode: Algorithmus

- Umrechnung einer natürlichen Zahl z (dezimal) in eine Darstellung zur Basis b
- Idee: Der Rest der Division durch b liefert jeweils die nächste Ziffer **von rechts**

```
Eingabe:  natürliche Zahl z, Basis b > 1
Ausgabe:  Ziffern z_(n-1) ... z_1 z_0 zur Basis b

i := 0
wiederhole
    z_i := z mod b        // Rest = nächste Ziffer
    z   := z div b        // ganzzahliger Quotient
    i   := i + 1
solange z > 0

Ergebnis: (z_(i-1) ... z_1 z_0)_b
```

- Die Reste werden also **von unten nach oben** gelesen

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Beispiel: Umrechnung Dezimal- zu Binärsystem

- 47 : 2 = 23 Rest 1
- 23 : 2 = 11 Rest 1
- 11 : 2 = 5 Rest 1
- 5 : 2 = 2 Rest 1
- 2 : 2 = 1 Rest 0
- 1 : 2 = 0 Rest 1
- $(47)_{10} = (?)_2$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Beispiel: Hexadezimalsystem

- **Hexadezimal → Dezimal:** Betrachte $z = (2\mathrm{AF})_{16}$

$$z = 2 \cdot 16^2 + 10 \cdot 16^1 + 15 \cdot 16^0 = 512 + 160 + 15 = (687)_{10}$$

- **Dezimal → Hexadezimal** mit der Divisionsmethode:
  - 687 : 16 = 42 Rest 15 → F
  - 42 : 16 = 2 Rest 10 → A
  - 2 : 16 = 0 Rest 2 → 2
  - Reste von unten nach oben gelesen: $(687)_{10} = (2\mathrm{AF})_{16}$
- **Hexadezimal ↔ Binär:** jede Ziffer einzeln in 4 Bit umwandeln
  - $(2\mathrm{AF})_{16} = (0010\ 1010\ 1111)_2$

Addition

- mod-Operation („modulo“): $amodb$ ist der Rest der Division von a durch b
- Resultat von $x + y$ ist definiert für n Bits als $(x + y)\, mod 2^n$
- Beispiel:

$$\begin{array}{rr} & 0011 \\ + & 0010 \\ \hline & 0101 \end{array}$$

$$\begin{array}{rcr} 3 & = & 2^0 + 2^1 \\ 2 & = & 2^1 \\ \hline 5 & = & 2^0 + 2^2 \end{array}$$

- Überlauf eines jeden Bits ergibt Übertrag

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Überlauf bei Addition

Beispiel: Was geschieht bei $15 + 2$?

$$\begin{array}{rr} & 1111 \\ + & 0010 \\ \hline & 0001 \end{array}$$

$$\begin{array}{rcr} 15 & = & 2^0 + 2^1 + 2^2 + 2^3 \\ 2 & = & 2^1 \\ \hline 1 & = & 2^1 \end{array}$$

$$1 = (15 + 2)\, mod 2^4$$

- Überläufe entstehen durch endliche Präzision. Die implementierte Addition ist daher „modular“, und das Resultat von $x + y$ bei n Bits ist definiert als $(x + y)\, mod 2^n$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Multiplikation

Schulmethode:
Sei x der Multiplikand, $y = (y_{n-1}, \dots, y_0)$ der Multiplikator, dann ist

$$x \cdot y = x \cdot y_0 + x \cdot y_1 \cdot 2 + x \cdot y_2 \cdot 2^2 + \dots + x \cdot y_{n-1} \cdot 2^{n-1}$$

$$= \sum_{i=0}^{n-1} x \cdot y_i \cdot 2^i$$

- In der Praxis ist es sinnvoll, jeden Term der Form $x \cdot y_i \cdot 2^i$
- zu addieren, sobald er generiert wurde:

$$x \cdot y = \left(\dots\left(\left(x \cdot y_0 + x \cdot y_1 \cdot 2\right) + x \cdot y_2 \cdot 2^2\right) + \dots\right) + x \cdot y_{n-1} \cdot 2^{n-1}$$

Abschnitt 1.2

Ganze Zahlen

- Darstellung ganzer Zahlen im Rechner
- Alternative Darstellungen ganzer Zahlen
- BCD-Code

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Darstellung ganzer Zahlen im Rechner

- Bisher wurden nur natürliche Zahlen betrachtet. Wie sollte man ganze Zahlen darstellen?
- Der konzeptionell einfachste Ansatz:
- Dualdarstellung wie bisher plus Vorzeichen-Bit
- 0 = +, 1 = -, Bsp. +5 = 0101, -5 = 1101
- **Nachteile:** 0 hat zwei Darstellungen und man braucht sowohl Addier- als auch Subtrahierwerk:

Nachteile der vorzeichenlosen Version Verwendung von K1(x) und K2(x)

1.	+x, +y	x+y	Add.
2.	-x,-y	-(x+y)	Add.
3.	+x, -y (x ≥ y)	x-y	Subtr.
	-x, +y (y ≥ x)	y-x	Subtr.
4.	+x, -y (x < y)	-(y-x)	Subtr.
	-x, +y (y < x)	-(x-y)	Subtr.

Exzessdarstellung

- Idee: Zu jeder Zahl z wird ein fester Wert q addiert (**Exzess**, auch **Bias**). Gespeichert wird die vorzeichenlose Binärzahl $z + q$.
- Rückrechnung: $z = (x_{n-1} \dots x_0)_2 - q$
- Üblich bei n Bit: $q = 2^{n-1} \Rightarrow$ Wertebereich $-2^{n-1} \leq z \leq 2^{n-1} - 1$
- **Beispiel:** $n = 4$, $q = 8$

z	$z + 8$	**Bitfolge**
-8	0	0000
-3	5	0101
0	8	1000
$+5$	13	1101
$+7$	15	1111

- Die Reihenfolge der Bitfolgen entspricht der Reihenfolge der Zahlen: Vergleich wie bei vorzeichenlosen Zahlen
- Anwendung: Exponent von Gleitkommazahlen (IEEE 754, dort mit $q = 2^{k-1} - 1$, siehe Abschnitt 1.4)

Darstellung ganzer Zahlen im Rechner

- Wir arbeiten hauptsächlich auf der Menge $\sum_2 = 0, 1.$
- $\sum_2^n$ beschreibt dann die Menge der n-stelligen Binärzahlen.

- Sei $x = (x_{n-1}...x_0)_2 \in \sum_2^n$ eine n-stellige Binärzahl.
- $K_1(x) := (\overline{x_{n-1}},...,\overline{x_0})_2$ heißt **Einer-Komplement** von x
- $K_2(x) := (\overline{x_{n-1}},...,\overline{x_0})_2 + 1 = K_1(x) + 1\ (modulo)\ 2^n$ heißt **Zweier-Komplement** von x

Beispiel: x = 10110010: $K_1(x) = 01001101$ $K_2(x) = 01001110$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Umrechnungsregeln

- Einerkomplement: Invertierung

Beispiel: $x = 10110010$

$K_1(x) = 01001101$

- Zweierkomplement: Invertierung $+1$

Beispiel: $x = 10110010$

$K_1(x) = 01001101$

$K_2(x) = 01001110$

Beispiel: Vorzeichenbehaftete Zahlen mit 4 Bit

- Das MSB legt das Vorzeichen fest
- $(0???)_2$
- $(1???)_2$
  - Wie interpretieren wir die Bits bei den Fragezeichen?

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Einerkomplement

- $(0???)_2$: wird interpretiert wie im vorzeichenlosen Fall
- $(1???)_2$: Bilde Einerkomplement K_1
- Beispiel:

$$(0010)_2 = +(010)_2 = (2)_{10}$$

$$(1010)_2 = -K_1(1010)_2 = -(0101)_2 = -(5)_10$$

- $K_1(K_1(x)) = x$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Addition im Einerkomplement

Beispiel: $(3)_{10} + (-6)_{10} = (-3)_{10}$

$(3)_{10} = (0011)_2$

$-6_{10} = K_1(0110)_2 = (1001)_2$

$$\begin{array}{rr} & 0011 \\ + & 1001 \\ \hline & 1100 \end{array}$$

$$(1100)_2 = -K_1(1100)_2 = -(0011)_2 = -(3)_{10}$$

Zweierkomplement

- $(0???)_2$: wird interpretiert wie im vorzeichenlosen Fall
- $(1???)_2$: Bilde Zweierkomplement K_2
- Beispiel:
- $(0010)_2 = +(010)_2 = (2)_{10}$
- $(1010)_2 = -K_2(1010)_2 = -(0101 + 1)_2 = -(6)_{10}$
- $K_2(K_2(x)) = x$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Addition im Zweierkomplement

- **Beispiel:** $(3)_{10} + (-6)_{10} = -(3)_{10}$
- $(3)_{10} = (0011)_2$
- $-(6)_{10} = K_2(0110)_2 = (1001 + 1)_2 = (1010)_2$

$$\begin{array}{rr} & 0011 \\ + & 1010 \\ \hline & 1101 \end{array}$$

- $(1101)_2 = -K_2(1101)_2 = -(0010 + 1)_2$
- $= -(3)_{10}$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Darstellungsbereiche bei 4 Bit

- Vorzeichenlos:
- $(0)_{10} \cdots (15)_{10}$
- Einerkomplement:
- $-(7)_{10} \cdots (7)_{10}$
- Zweierkomplement:
- $-(8)_{10} \cdots (7)_{10}$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Darstellungsbereiche

- Sei $x = (x_{n-1} \dots x_0)_2 \in B^n$ eine n-stellige Binärzahl.
- Im Einerkomplement: $-\left(2^{n-1} - 1\right) \leq x \leq 2^{n-1} - 1$
- Im Zweierkomplement: $-2^{n-1} \leq x \leq 2^{n-1} - 1$

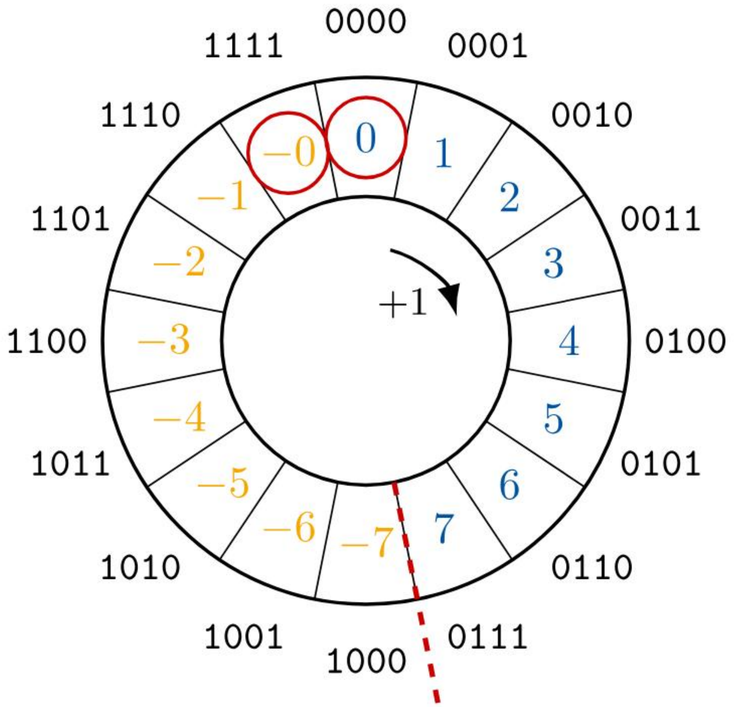
- Wegen der doppelten Darstellung der 0 im Einerkomplement lässt sich im Zweierkomplement eine Zahl mehr darstellen
- Einfache (technische) Umsetzung arithmetischer Operationen
- Addition bspw. genauso wie vorzeichenlos

Alternative Darstellung ganzer Zahlen

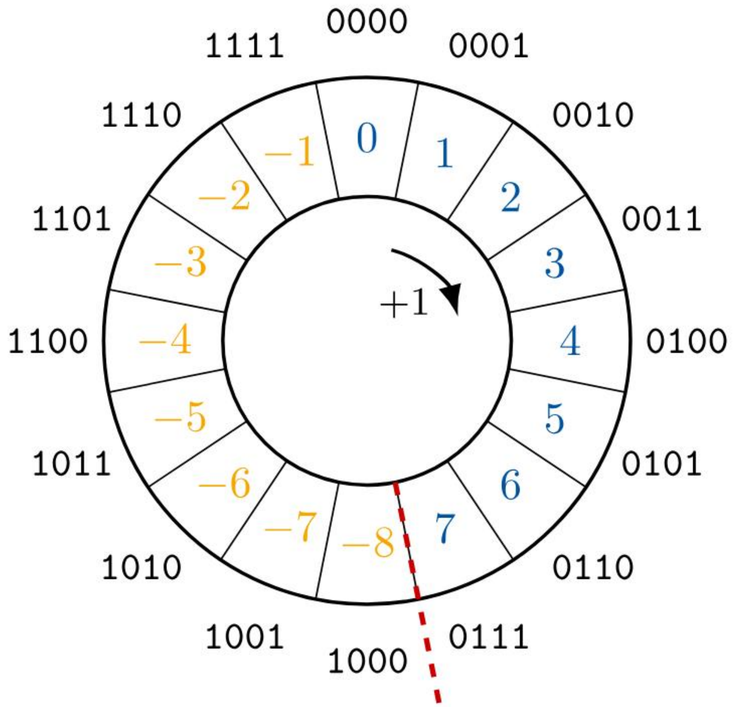
Bitfolge	Darstellung in Dezimalnotation		
	Vorz./Betrag	K1	K2
0000	+0	+0	+0
0001	+1	+1	+1
0010	+2	+2	+2
0011	+3	+3	+3
0100	+4	+4	+4
0101	+5	+5	+5
0110	+6	+6	+6
0111	+7	+7	+7
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Zahlenrad: Einer- vs. Zweierkomplement



Einerkomplement



Zweierkomplement

- $+1$ entspricht einem Schritt im Uhrzeigersinn, der **Überlauf** liegt zwischen 0111 und 1000
- **Einerkomplement:** zwei Darstellungen der 0 (0000 und 1111), beim Schritt von -0 nach 0 geht ein Schritt verloren
- **Zweierkomplement:** jede Bitfolge steht für genau eine Zahl, z.B. $-1 + 1 = 0$ ohne Korrektur

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Übersicht: Darstellung ganzer Zahlen

	kleinste Zahl	größte Zahl	$n = 4$	Vorteile	Nachteile
Vorzeichen/ Betrag	$-(2^{n-1}-1)$	$2^{n-1}-1$	$-7 \dots 7$	intuitiv; Negation durch Kippen des Vorzeichenbits	zwei Darstellungen der 0; Addier- *und* Subtrahierwerk nötig
Einer-komplement	$-(2^{n-1}-1)$	$2^{n-1}-1$	$-7 \dots 7$	Negation durch Invertieren; Subtraktion als Addition	zwei Darstellungen der 0; Übertrag muss bei der Addition zurückgeführt werden
Zweier-komplement	-2^{n-1}	$2^{n-1}-1$	$-8 \dots 7$	eindeutige 0; Addition wie vorzeichenlos; Standard in Rechnern	asymmetrisch (-2^{n-1} hat kein positives Gegenstück); Negation: Invertieren $+1$
Exzess-2^{n-1}	-2^{n-1}	$2^{n-1}-1$	$-8 \dots 7$	eindeutige 0; Vergleich wie vorzeichenlos	Addition braucht Korrektur um q; 0 ist nicht $00 \dots 0$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

BCD-Code

Binary **C**oded **D**ecimal

- Ziffern werden einzeln mit 4 Bits als vorzeichenlose Binärzahl
- dargestellt.

Beispiel: 4739 = 0100 0111 0011 1001

- 6 Bitmuster ungenutzt, erlaubt Darstellung der Vorzeichen:
- z.B. + = 1010 und - = 1011

Nachteil: Erschwerte Addition

Beispiel: 4739 + 1287 = 6026

0100	0111	0011	1001
0001	0010	1000	0111
0101	1001	1100	0000

Z.B. Fehler an der letzten Stelle wegen des Übertrags.

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

BCD-Code

- Korrektur: Bei jedem Übertrag und bei jeder ungültigen
- BCD-Darstellung 6 aufaddieren:

0101	1001	1100	0000
			0110
0101	1001	1100	0110
		0110	
0101	1010	0010	0110
	0110		
0110	0000	0010	0110
6	0	2	6

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Exkurs: Anwendungen BCD Code

- Vorteil von BCD: Einfache Konvertierung in Dezimalsystem, daher BCD Code für Anzeigen sinnvoll
- Konkrete Anwendungsbeispiele
  - DCF Funkuhr Signal
  - Bis heute existieren Hardwarebausteine (IC), welche u.a. BCD Codes für 7 Segment Displays dekodieren
  - Taschenrechner (u.a. von Texas Instruments) nutzen die freien Kombinationen für weitere Symbole (z.B. unendlich)

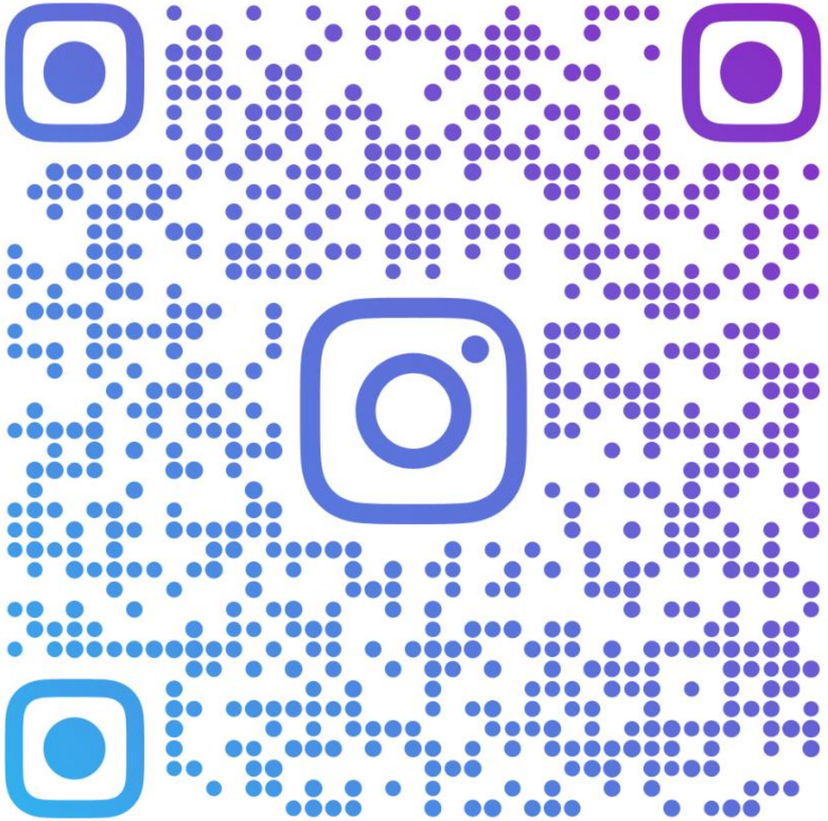
Abschnitt 1.3

Festkommazahlen

Festkomma-Darstellung

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Instagram



TI_RWTH

Motivation: Festkommazahlen

- Fest vorgegebene Anzahl an Vor- und Nachkommastellen
- Getrennt voneinander binär dargestellt
- Häufig aus Performancegründen eingesetzt

Festkomma-Darstellung

- Komma **rechts** von der Stelle mit dem niedrigsten Wert:
- ein n-Bit Wort $(z_{n-1} \dots z_0)_2$ stellt dann die Zahl
- $z = \sum_{i=0}^{n-1} z_i \cdot 2^i$ dar, z.B. 110101.0
- Komma **links** von der Stelle mit dem höchsten Wert:
- ein n-Bit Wort $(x_1 \dots x_n)_2$ stellt dann die Zahl
- $z = \sum_{i=1}^{n} z_i \cdot 2^{-i}$ dar, z.B. 0.110101
- Allgemein stellt eine Bitfolge $(x_{n-1}, \dots, x_1, x_0, x_{-1}, \dots, x_{-m})_2$,
- falls das Komma zwischen x_0 und x_{-1} angenommen wird, die Zahl
- $z = \sum_{i=-m}^{n-1} z_i \cdot 2^i$ dar, z.B. 1101.01

Beispiel für Festkomma-Arithmethik

Verwende 4 Bits für Vor- und 3 Bits für Nachkommateil

$3.5 + 2.5 = (0011)\,.\,(100) + (0010)\,.\,(100)$

$= (0110)\,.\,(000)$

$= 6.0$

Beispiel

$$
\begin{array}{llll}
0{,}6875 & \cdot 2 = & 1 & ,375 \\
0{,}375 & \cdot 2 = & 0 & ,75 \\
0{,}75 & \cdot 2 = & 1 & ,5 \\
0{,}5 & \cdot 2 = & 1 & ,0 \\
0 & & &
\end{array}
$$

- $(0,6875)_{10} = 0,(?)_2$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Abschnitt 1.4

Gleitkommazahlen

- Gleitkomma-Darstellung
- Regeln für das Rechnen mit Gleitkommazahlen
- Rechnerinterne Darstellung von Gleitkommazahlen
- IEEE 754

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Motivation: Gleitkommazahlen

- Alternative zu Festkomma: approximative Darstellung reeller bzw. rationaler Zahlen
- Bei gleicher Anzahl an Bits wie bei Festkommadarstellung wird viel größerer Zahlenbereich abgedeckt
- Aber: Nicht jede Zahl in diesem Bereich kann exakt dargestellt werden
- Bis in die 80er Jahre hinein gab es viele verschiedene Gleitkomma-Darstellungen.
- Das Institute of Electrical and Electronics Engineers (IEEE) gab einen Standard für 32-, 64- und 80-Bit Gleitkommazahlen heraus.
- Probleme durch die Behandlung von Über- bzw. Unterläufen und anderen Ausnahmen.

Gleitkomma-Darstellung

- Jede Zahl z wird in der Form $z = \pm m \cdot b^{\pm e}$ dargestellt mit m **Mantisse**, e **Exponent**, b **Basis** für den Exponenten.
- Die Basis ist für alle auftretenden Exponenten die gleiche; daher rechnerinterne Darstellung einer Gleitkomma-Zahl:

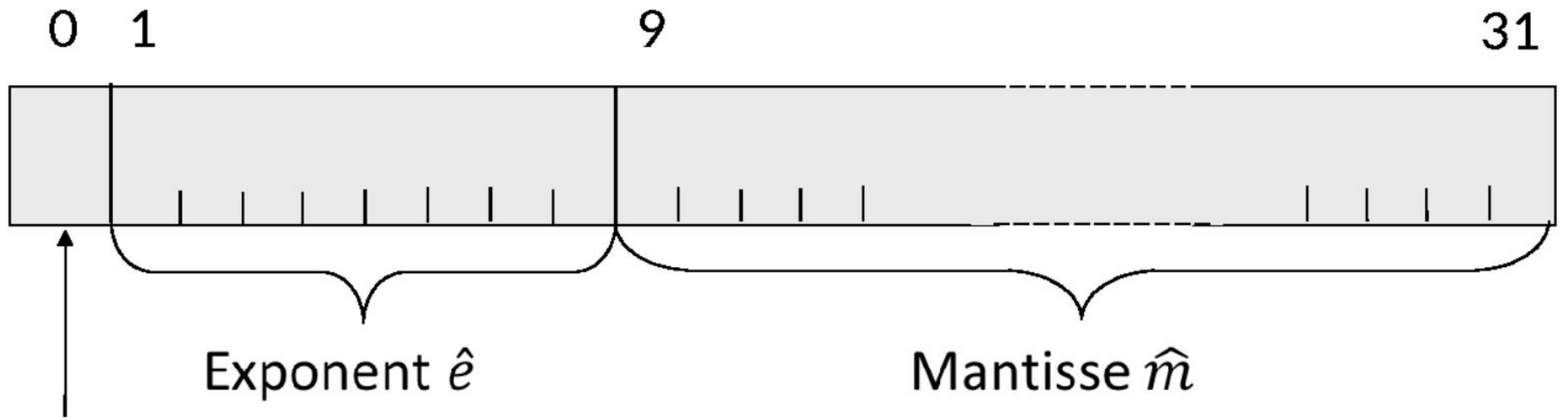
$$(\pm m, \pm e)$$

- Beispiel: $e = 1.6 * 10^{-19} C$

Bit-weise Darstellung

- Speicherung einer Gleitkomma-Zahl in drei Feldern
  - Vorzeichen $\hat{s}$
  - Exponent $\hat{e} = (e_{k-1} \dots e_0)$
  - Mantisse $\hat{m} = (m_{n-1} \dots m_0)$
- Bei 32-Bit haben wir $k = 8$ und $n = 23$
- Bei 64-Bit haben wir $k = 11$ und $n = 52$
- Drei Fälle: normalisiert, denormalisiert und Sonderzahlen

IEEE 754 32-bit („single“)



- 0 bei positiven Zahlen
- 1 bei negativen Zahlen

Vorzeichen $\hat{s}$ der Mantisse

- Mit dieser Aufteilung darstellbare Beträge:

$1.18 \cdot 10^{-38}$ bis $3.40 \cdot 10^{38}$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Normalisierte Darstellung

- $\hat{e} \neq (0 \dots 0)$ und $\hat{e} \neq (1 \dots 1)$
- Exponent $e = \hat{e} - bias$ mit $bias = 2^{k-1} - 1$
  - Für 32-Bit also $bias = 127$ da $k = 8$
- m wird interpretiert als $m = 1 + (0, \hat{m}) = 1, \hat{m}$
  - Darstellungstrick für ein weiteres Bit an Präzision
- Also gilt $1 \leq m < 2$

Normalisierte Darstellung

- Zum Beispiel wird die Zahl $z = -1.5$ dargestellt durch

$$10111111110000000000000000000000$$

- d.h.

$$\hat{s} = 1$$

$$\begin{aligned} \hat{e} &= 01111111 \\ \hat{m} &= 10000000000000000000000 \end{aligned}$$

$$s = 1$$

$$e = \hat{e} - 127 = 0$$

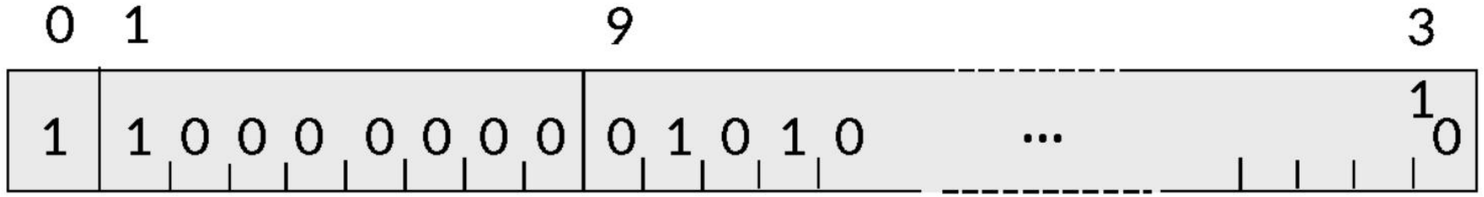
$$m = 1.\hat{m} = 1.5$$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Normalisierte Darstellung

- Stelle -2.625 als IEEE 754 Gleitkommazahl dar
- Vorzeichen: s = 1 $\rightarrow$ ŝ = 1
- Mantisse: $(2,625)_{10} = (10,101)_2$

Normalisieren: $(10,101)_2 = (1,0101)_2 \cdot (2)^1$
m̂ = 01010... - Exponent: e = ê -127 $\leftrightarrow$ ê = e + 127
$\hat{e} = 1 + 127 = 128 = (10000000)_2$
...



Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Normalisierte Darstellung

- Welcher Dezimalzahl entspricht die folgende Binärzahl?

$0\ 10110011\ 010100...0000$

- Vorzeichen: ŝ = 0 → s = 0 → pos. Zahl
- Exponenten: $\hat{e} = (10110011)_2 = (179)_{10}$
  e = ê -127 = 52
- Mantisse: m = 1,$\hat{m} = (1,0101)_2 = (1,3125)_{10}$
- **Ergebnis:** $z = +1,3125 \cdot 2^{52}$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Beispiel: Dezimal $\rightarrow$ IEEE 754 (32 Bit)

Stelle $z = 12{,}375$ als IEEE 754 Gleitkommazahl dar:

1. **Vorzeichen:** $z > 0 \Rightarrow \hat{s} = 0$
2. **Binär umwandeln:** $12 = (1100)_2$, Nachkommateil durch fortgesetztes Multiplizieren mit 2:

$$\begin{aligned} 0{,}375 \cdot 2 &= 0{,}75 &&\rightarrow 0 \\ 0{,}75 \cdot 2 &= 1{,}5 &&\rightarrow 1 \qquad \Rightarrow \quad 12{,}375 = (1100{,}011)_2 \\ 0{,}5 \cdot 2 &= 1{,}0 &&\rightarrow 1 \end{aligned}$$

3. **Normalisieren:** $(1100{,}011)_2 = (1{,}100011)_2 \cdot 2^3 \Rightarrow e = 3,\ m = (1{,}100011)_2$
4. **Exponent mit Bias:** $\hat{e} = e + 127 = 130 = (10000010)_2$
5. **Mantisse:** führende 1 weglassen, auf 23 Bit auffüllen: $\hat{m} = 10001100000000000000000$

$$\underbrace{0}_{\hat{s}}\ \underbrace{10000010}_{\hat{e}}\ \underbrace{10001100000000000000000}_{\hat{m}} = (41460000)_{16}$$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Beispiel: IEEE 754 (32 Bit) $\rightarrow$ Dezimal

Welche Zahl wird durch $(\mathrm{C1A40000})_{16}$ dargestellt?

1. **Bitfolge aufteilen:** $(\mathrm{C1A40000})_{16} = \underbrace{1}_{\hat{s}}\ \underbrace{10000011}_{\hat{e}}\ \underbrace{01001000000000000000000}_{\hat{m}}$
2. **Vorzeichen:** $\hat{s} = 1 \Rightarrow$ negative Zahl
3. **Exponent:** $\hat{e} = (10000011)_2 = 131 \Rightarrow e = \hat{e} - 127 = 4$
4. **Mantisse:** führende 1 ergänzen: $m = (1{,}01001)_2 = 1 + \frac{1}{4} + \frac{1}{32} = 1{,}28125$
5. **Zusammensetzen:** $z = -m \cdot 2^e = -1{,}28125 \cdot 2^4 = -(10100{,}1)_2 = -20{,}5$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Denormalisierte Darstellung

- Normalisiert lässt sich die 0 nicht darstellen wegen $1 \leq m$
- Falls $\hat{e} = (0 \dots 0)$, dann denormalisiert
  - Exponent $e = 1 - bias$
  - Mantisse $m = \hat{m}$ ohne führende $1z = 0, \hat{m}$
- Genaue Darstellung für 0 und Werte nahe der 0
- Durch $1 - bias$ statt $0 - bias$ sanfter Übergang zwischen denormalisiert und normalisiert

Denormalisierte Darstellung

- Durch $1 - bias$ statt $0 - bias$ sanfter Übergang zwischen denormalisiert und normalisiert
- Kleinste normalisierte Zahl (Betrag)
- Größte denormalisierte Zahl (Betrag)

$$\begin{aligned} \hat{m} &= (0 \ldots 0) \\ \hat{e} &= (0 \ldots 1) \end{aligned}$$

$$\Rightarrow$$

$$1.0 \cdot 2^{-126}$$

$$\begin{aligned} \hat{m} &= (1 \ldots 1) \\ \hat{e} &= (0 \ldots 0) \end{aligned}$$

$$(0.11111111111111111111)_2 \cdot 2^{-126}$$

$$= (0.99999880790710449218 75)_{10} \cdot 2^{-126}$$

Darstellung der Null

- Kurioserweise gibt es zwei Darstellungen für die Zahl 0:

$$\left.\begin{pmatrix} \hat{s} = 0 \\ \hat{m} = (0 \dots 0) \\ \hat{e} = (0 \dots 0) \end{pmatrix}\right\} \Rightarrow +0$$

$$\left.\begin{pmatrix} \hat{s} = 1 \\ \hat{m} = (0 \dots 0) \\ \hat{e} = (0 \dots 0) \end{pmatrix}\right\} \Rightarrow -0$$

- Nach IEEE 754 manchmal gleich interpretiert (bei Vergleich z.B.), manchmal unterschiedlich

Sonderfall: Über- und Unterlauf

- **Überlauf:** Das Resultat einer Gleitkomma-Operation ist zu groß, um es darzustellen (bzw. zu klein, dann **Unterlauf**)
- Das Ergebnis einer solchen Gleitkomma-Operation wird durch **„infinity“** gekennzeichnet, Symbole $+\infty$ oder $-\infty$
- Beispiel:

$$\frac{1}{0} = +\infty$$

$$\frac{1}{-0} = -\infty$$

- Im IEEE 754 wird dies durch $\hat{m} = (0 \dots 0)$ und einen
- Exponenten $\hat{e} = (1 \dots 1)$ mit $\hat{s} \in 0, 1$ dargestellt

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Ausnahmebedingungen: NaN

- Wenn das Resultat einer Gleitkomma-Operation keine gültige Gleitkommazahl ist, dann wird eine Sonderzahl mit der Bedeutung **„not a number“ (NaN)** generiert.
  - Beispiel:

$$+\infty + (-\infty) = NaN$$

- *NaN* werden durch $\hat{m} \neq (0 \dots 0)$ und einen Exponenten
- $\hat{e} = (1 \dots 1)$ dargestellt

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Approximative Darstellung

- Es können nicht alle Zahlen dargestellt werden
- Beispiel: $(0.1)_{10}$
- Nächstkleinere Zahl

$\hat{s} = 0(1.10011001100110011001101)_2 \cdot 2^{-4}$
$\hat{e} = (01111011) = 0.100000001490116119384765625$

$$\hat{m} = (10011001100110011001101)$$

$$(1.10011001100110011001100)_2 \cdot 2^{-4}$$

$$= 0.099999994039535522460937 5$$

$$\hat{s} = 0$$

$$\hat{e} = (01111011)$$

$$\hat{m} = (10011001100110011001100)$$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Approximative Darstellung

- float a = 0.1; float b = 0.3;
- if (a == 0.1) ergibt true
- Aber if (a+b == 0.4) ergibt false
- Daher Vergleich mit Epsilon-Umgebung nötig

Rechnen mit Gleitkommazahlen

- Seien $x = m_x \cdot 2^{d_x}$
  $y = m_y \cdot 2^{d_y}$
- Falls $d_x \leq d_y$ dann

$$
\begin{aligned}
x + y &= \left(m_x \cdot 2^{d_x - d_y} + m_y\right) \cdot 2^{d_y} \\
x - y &= \left(m_x \cdot 2^{d_x - d_y} - m_y\right) \cdot 2^{d_y}
\end{aligned}
$$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Rechnen mit Gleitkommazahlen

- Seien $x = m_x \cdot 2^{d_x}$
  $y = m_y \cdot 2^{d_y}$
- Falls $d_x \leq d_y$ dann

$$\begin{aligned} x + y &= \left(m_x \cdot 2^{d_x - d_y} + m_y\right) \cdot 2^{d_y} \\ x - y &= \left(m_x \cdot 2^{d_x - d_y} - m_y\right) \cdot 2^{d_y} \end{aligned}$$

- Ebenso $x \cdot y = (m_x \cdot m_y) \cdot 2^{d_x + d_y}$

$$\frac{x}{y} = \frac{m_x}{m_y} \cdot 2^{d_x - d_y}$$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Beispiel: Addition zweier IEEE 754 Zahlen (1/2)

Berechne $x + y$ für

$$\begin{aligned} x &= \underbrace{1}_{\hat{s}}\ \underbrace{10000001}_{\hat{e}}\ \underbrace{10100000000000000000000}_{\hat{m}} \\ y &= \underbrace{1}_{\hat{s}}\ \underbrace{10000000}_{\hat{e}}\ \underbrace{11100000000000000000000}_{\hat{m}} \end{aligned}$$

0. **Zerlegen:** x: $\hat{s} = 1$, $e = 129 - 127 = 2$, $m = (1{,}101)_2 \Rightarrow x = -(1{,}101)_2 \cdot 2^2$
   y: $\hat{s} = 1$, $e = 128 - 127 = 1$, $m = (1{,}111)_2 \Rightarrow y = -(1{,}111)_2 \cdot 2^1$
1. **Exponenten angleichen:** kleineren Exponenten an den größeren anpassen, d.h. Mantisse von y um $d_x - d_y = 1$ Stelle nach rechts schieben:

$$y = -(1{,}111)_2 \cdot 2^1 = -(0{,}1111)_2 \cdot 2^2$$

2. **Vorzeichen prüfen:** beide Vorzeichen negativ $\Rightarrow$ **echte Addition** der Beträge, Ergebnis negativ:

$$x + y = -\left(|x| + |y|\right)$$

   (Bei verschiedenen Vorzeichen: Beträge subtrahieren, Vorzeichen des betragsgrößeren Operanden)

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Beispiel: Addition zweier IEEE 754 Zahlen (2/2)

3. **Mantissen binär addieren:**

$$
\begin{array}{lccccccc}
 & & 1 & , & 1 & 0 & 1 & 0 \\
+ & & 0 & , & 1 & 1 & 1 & 1 \\
\text{Übertrag} & 1 & 1 & & 1 & 1 & & \\
\hline
 & 1 & 0 & , & 1 & 0 & 0 & 1
\end{array}
\qquad \Rightarrow \quad |x| + |y| = (10{,}1001)_2 \cdot 2^2
$$

4. **Normalisieren:** Mantisse muss die Form $1, \ldots$ haben $\Rightarrow$ Komma um eine Stelle nach links, Exponent $+1$:

$$(10{,}1001)_2 \cdot 2^2 = (1{,}01001)_2 \cdot 2^3$$

5. **Ergebnis aufschreiben:** $\hat{s} = 1$, $\hat{e} = 3 + 127 = 130 = (10000010)_2$, $\hat{m} = 01001000000000000000000$

$$x + y = \underbrace{1}_{\hat{s}} \; \underbrace{10000010}_{\hat{e}} \; \underbrace{01001000000000000000000}_{\hat{m}}$$

Probe: $x = -6{,}5$, $y = -3{,}75 \Rightarrow x + y = -10{,}25 = -(1{,}01001)_2 \cdot 2^3$ ✓

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Beispiel: Multiplikation zweier IEEE 754 Zahlen (1/2)

Berechne $x \cdot y$ für

$$\begin{aligned} x &= \underbrace{0}_{\hat{s}}\ \underbrace{10000010}_{\hat{e}}\ \underbrace{10000000000000000000000}_{\hat{m}} \\ y &= \underbrace{1}_{\hat{s}}\ \underbrace{01111101}_{\hat{e}}\ \underbrace{01000000000000000000000}_{\hat{m}} \end{aligned}$$

0. **Zerlegen:** x: $\hat{s} = 0$, $e = 130 - 127 = 3$, $m = (1{,}1)_2 \Rightarrow x = +(1{,}1)_2 \cdot 2^3$
   y: $\hat{s} = 1$, $e = 125 - 127 = -2$, $m = (1{,}01)_2 \Rightarrow y = -(1{,}01)_2 \cdot 2^{-2}$
1. **Vorzeichen:** unabhängig vom Rest, einfach $\hat{s} = \hat{s}_x \oplus \hat{s}_y = 0 \oplus 1 = 1$
   - Keine Fallunterscheidung wie bei der Addition: Mantissen werden immer als Beträge multipliziert
2. **Exponenten addieren:** kein Angleichen nötig!

   $$e = e_x + e_y = 3 + (-2) = 1$$

   bzw. direkt mit Bias: $\hat{e} = \hat{e}_x + \hat{e}_y - 127 = 130 + 125 - 127 = 128$

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Beispiel: Multiplikation zweier IEEE 754 Zahlen (2/2)

3. **Mantissen multiplizieren:** Komma im Ergebnis nach $2 + 1 = 3$ Nachkommastellen

$$\begin{array}{cccccccc} 1, & 0 & 1 & \cdot & 1, & 1 & & \\ \hline & & & & 1 & 0 & 1 & \\ & & & & & 1 & 0 & 1 \\ \hline & & & & 1, & 1 & 1 & 1 \end{array} \quad \Rightarrow \quad m = (1{,}111)_2$$

4. **Normalisieren:** $1 \leq m < 2 \Rightarrow$ bereits normalisiert (sonst: Komma schieben, Exponent anpassen)
5. **Ergebnis aufschreiben:** $\hat{s} = 1$, $\hat{e} = 1 + 127 = 128 = (10000000)_2$, $\hat{m} = 11100000000000000000000$

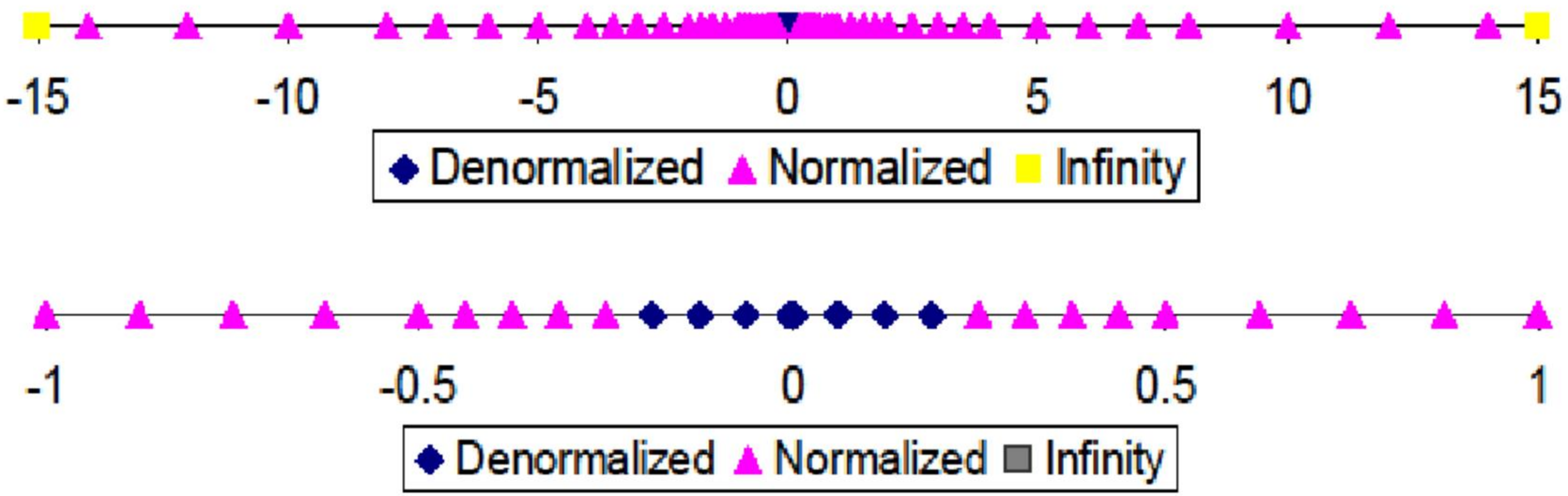
$$x \cdot y = \underbrace{1}_{\hat{s}} \ \underbrace{10000000}_{\hat{e}} \ \underbrace{11100000000000000000000}_{\hat{m}}$$

Probe: $x = 12, y = -0{,}3125 \Rightarrow x \cdot y = -3{,}75 = -(1{,}111)_2 \cdot 2^1$ ✓

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Verteilung von Gleitkommazahlen

- Vorzeichen: 1 Bit
- Exponent: 3 Bits
- Mantisse: 2 Bits
- Weicher Übergang zwischen normalisierten und denormalisierten Werten
- Präzise Verteilung um die 0



Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

IEEE 754 - 64-Bit Zahlen

- Genau wie bei den vorgestellten 32-Bit Gleitkomma-Zahlen, jedoch:
  - Der Exponent besitzt 11 Bits (vorher: 8 Bits).
  - Die Mantisse ist 52 Bits lang (vorher: 23 Bits).
- Außerdem beschreibt IEEE 754 auch 80-Bit Gleitkomma-Zahlen mit 15-Bit Exponenten und 64-Bit Mantisse
- x86-Architekturen verwenden interne 80-Bit Darstellung, Speicherung erfolgt in 32- oder 64-Bit
- 64-Bit Gleitkomma-Zahlen werden auch als **double** bezeichnet.

Achtung: Rechnen mit Gleitkommazahlen

- Mit $x, y \in R$ ist das Ergebnis einer IEEE 754 Operation
- $\circ \in +, -, \cdot,$ / definiert als $round(x \circ y)$
- IEEE 754 definiert nicht, wie exakt gerundet werden muss, sondern liefert Alternativen
  - Round to zero
  - Round down
  - Round up
  - Round to nearest
    - Tie to even
    - Tie away from zero
- D.h. die gleichen 32-Bit FP-Operationen können unterschiedliche Ergebnisse liefern, obwohl korrekt (den Standard befolgend)

Beispiel: Rundung

Modus	1.40	1.60	1.50	2.50	-1.50
Round to nearest (tie to even)	1	2	2	2	-2
Round to nearest (tie away from zero)	1	2	2	3	-2
Round toward zero	1	1	1	2	-1
Round down	1	1	1	2	-2
Round up	2	2	2	3	-1

Abschnitt 1.5

Zusammenfassung

Chair of Cyber-Physical Systems
RWTH AACHEN UNIVERSITY

Achtung: Fehler kosten (viel) Geld

- Ariane 5 stürzte 37 Sekunden nach dem Start ab
- Kosten in Höhe von mehreren Hundert Millionen US-Dollar



Zusammenfassung

- Darstellung von natürlichen und ganzen Zahlen sowie deren wichtigsten Rechenoperationen
- BCD-Kodierung
- Festkommadarstellung und IEEE 754 Fließkommadarstellung
- Wichtig: Operationen verhalten sich nicht immer wie erwartet

Literatur

- W. Kahan: An Interview with the Old Man of Floating-Point (https://people.eecs.berkeley.edu/~wkahan/ieee754status/754story.html)
- D. Goldberg: *What Every Computer Scientist Should Know About Floating-Point Arithmetic*. In: *ACM Computing Surveys*. 23, 1991, S. 5-48 (http://docs.sun.com/source/806-3568/ncg_goldberg.html)
- R. Bryant, D. O'Hallaron: Computer Systems – A Programmer's Perspective (Chapter 2). Prentice Hall