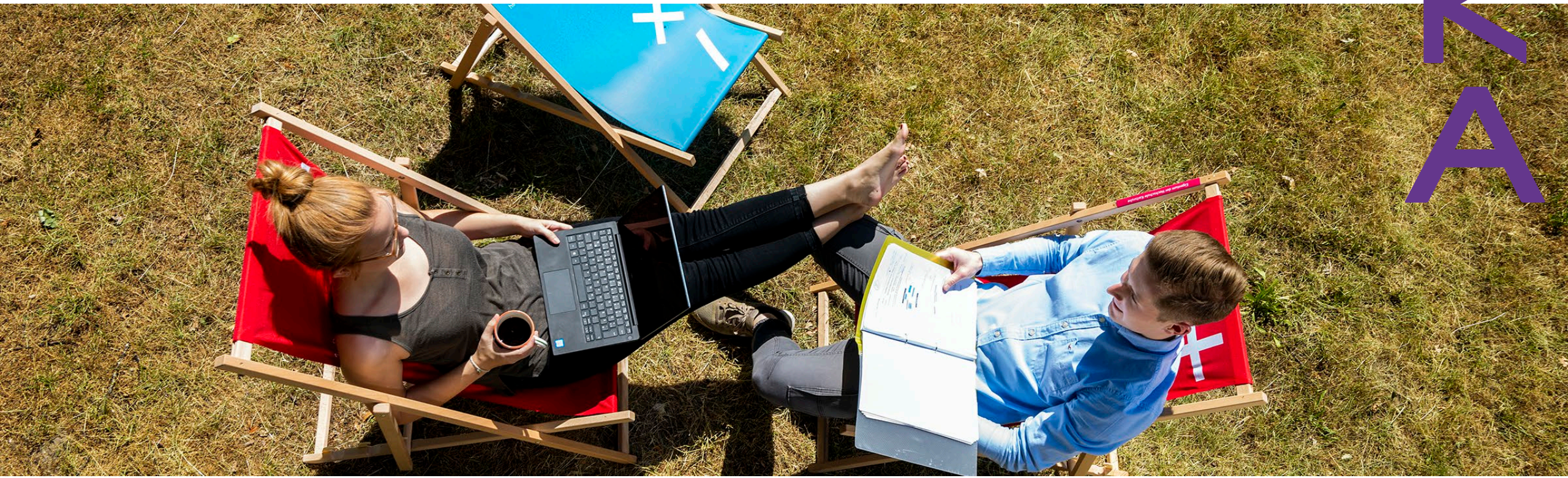
Hochschule Karlsruhe
University of
Applied Sciences

Fakultät für
Informatik und
Wirtschaftsinformatik

DSCB130 | 1.2 Zahlensysteme und binäre Arithmetik



Matthias Mruzek-Vering M.Sc.

Foto: Quirin Leppert

Themenblock 1 Informationsverarbeitung

Lehrinhalte

1.2 Zahlensysteme und binäre Arithmetik

- Begriffsklärung
- Darstellung von ganzen und negativen Zahlen
- Umwandlung von Zahlen in verschiedene Darstellungssysteme
- Binäre Arithmetik
- Größeneinheiten

1.2 Zahlensysteme und binäre Arithmetik

Begriffsklärung

Dezimalsystem

+ Zehnersystem, dekadisches Ziffernsystem
+ Eine natürliche Zahl z wird als Summe von Potenzen zur Basis $\boldsymbol{b = 10}$ dargestellt

$$z = a_n 10^n + a_{n-1} 10^{n-1} + \cdots + a_2 10^2 + a_1 10^1 + a_0 10^0$$

+ wobei die Koeffizienten $a_0, a_1, \ldots, a_n$ aus der Menge der Grundziffern $G = \{0,1,2,3,4,5,6,7,8,9\}$ zu wählen sind.
+ Wird dieses Konzept um negative Exponenten erweitert, so lassen sich auch Dezimalbrüche, d. h. rationale Zahlen r darstellen, die auch als Näherungen für reelle Zahlen dienen:

$$r = a_n 10^n + \cdots + a_1 10^1 + a_0 10^0 + a_{-1} 10^{-1} + a_{-2} 10^{-2} + \cdots + a_{-m} 10^{-m}$$

+ Eine rationale Zahl hat in obiger Notation also $n + 1$ Vorkommastellen und m Nachkommastellen.
+ Beispiel

$$123{,}76_{\text{dez}} = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 + 7 \cdot 10^{-1} + 6 \cdot 10^{-2}$$

1.2 Zahlensysteme und binäre Arithmetik

Begriffsklärung

Dualsystem

+ Ausgangspunkt: Repräsentation von Daten in Datenverarbeitungsanlagen durch die beiden Zustände `1` und `0`
+ Zweiersystem oder Binärsystem
+ Basis $\boldsymbol{b = 2}$ und den beiden Grundziffern $G = \{0,1\}$
+ Besondere Einfachheit der Arithmetik in diesem System, insbesondere der Subtraktion

+ Grundlegende Einheiten
  - Kleinste digitale Informationseinheit **Bit**
  - Bekannteste und meist verwendete Informationseinheit **Byte**
  - 1 Byte = 8 Bit
  - Größter Zahlenwert $2^8 = 256_{dez}$

1.2 Zahlensysteme und binäre Arithmetik

Begriffsklärung

Dualsystem

+ Wertebereich **Byte**: 0..255 bzw. -127..128 (mit Vorzeichenbit)
+ Höherwertigstes Bit links, niederwertigstes Bit rechts, analog zu unserer dezimalen Zahlendarstellung
+ Stellenwert des Bits an Stelle i ist 2^i

Stelle	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Wert	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Wert	128	64	32	16	8	4	2	1

1.2 Zahlensysteme und binäre Arithmetik

Begriffsklärung

Dualsystem

+ Beispiel A: $177_{dez} = 1011\ 0001_{bin}$

Stelle	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Wert	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Wert	128	64	32	16	8	4	2	1
Beispiel	1	0	1	1	0	0	0	1
Beispiel	$1x2^7$	$+ 0x2^6$	$+ 1x2^5$	$+ 1x2^4$	$+ 0x2^3$	$+ 0x2^2$	$+ 0x2^1$	$+ 1x2^0$
177 =	128	+ 0	+ 32	+ 16	+ 0	+ 0	+ 0	+ 1

1.2 Zahlensysteme und binäre Arithmetik

Begriffsklärung

Dualsystem

+ Beispiel B: $13_{dez} = 0000\ 1101_{bin}$

Stelle	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Wert	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Wert	128	64	32	16	8	4	2	1
Beispiel	0	0	0	0	1	1	0	1
Beispiel	$0x2^7$	$+ 0x2^6$	$+ 0x2^5$	$+ 0x2^4$	$+ 1x2^3$	$+ 1x2^2$	$+ 0x2^1$	$+ 1x2^0$
13 =	0	+ 0	+ 0	+ 0	+ 8	+ 4	+ 0	+ 1

1.2 Zahlensysteme und binäre Arithmetik

Begriffsklärung

Oktalsystem

+ Geschriebene Zahlen im Dualsystem können sehr lang und daher schwer zu merken sein
+ Achtersystem fasst drei binäre Stellen zu einer Oktalstelle zusammen
+ Basis $\boldsymbol{b = 2^3 = 8}$ und die Menge der acht Grundziffern $G = \{0,1,2,3,4,5,6,7\}$
+ Bildung der Zahlenwerte
  - Dezimalzahl zuerst in Dualzahl umwandeln
  - Mit der niederwertigsten Stelle der Binärzahl
  - Jeweils drei Binärziffern zu einer Oktalziffer vereinigen

1.2 Zahlensysteme und binäre Arithmetik

Begriffsklärung

Oktalsystem

+ Beispiel: $53_{dez} = 0011\ 0101_{bin} = 65_{okt}$

Stelle	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Wert	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Wert	128	64	32	16	8	4	2	1
Beispiel	0	0	1	1	0	1	0	1
Beispiel	$0x2^7$	$+ 0x2^6$	$+ 1x2^5$	$+ 1x2^4$	$+ 0x2^3$	$+ 1x2^2$	$+ 0x2^1$	$+ 1x2^0$
53_{dez} =	0	+ 0	+ 32	+ 16	+ 0	+ 4	+ 0	+ 1
	$+ 0x2^1$	$+ 0x2^0$	$+ 1x2^2$	$+ 1x2^1$	$+ 0x2^0$	$+ 1x2^2$	$+ 0x2^1$	$+ 1x2^0$
65_{okt} =	0	+ 0	+ 4	+ 2	+ 0	+ 4	+ 0	+ 1

1.2 Zahlensysteme und binäre Arithmetik

Begriffsklärung

Hexadezimalsystem

+ Vereinfachte und deshalb besser lesbare Darstellung binärer Werte
+ Zusammenfassung von vier binären Stellen zu einer Hexadezimalziffer
+ Basis $\boldsymbol{b = 2^4 = 16}$ und die Menge der acht Grundziffern $G = \{0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, F\}$
+ Wortlänge entscheidet bei der Umwandlung in Dezimalwert, hier 32 Bit (Double Word)

Stelle	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	1
39911_{dez} =	$1x2^3$			$+1x2^0$	$1x2^3$		$1x2^1$	$+1x2^0$	$1x2^3$	$+1x2^2$	$+1x2^1$			$+1x2^2$	$+1x2^1$	$+1x2^0$
$9BE7_{hex}$ =	9				B				E					7		

1.2 Zahlensysteme und binäre Arithmetik

Darstellung von ganzen und negativen Zahlen

- Vorzeichen-Bit zeigt an, ob Zahl positiv oder negativ ist
- Darstellung als Komplement (hier *Zweierkomplement*)
  - Negiere alle Bits und addiere 1
  - Separates Vorzeichen-Bit kann eingespart werden
  - Addition funktioniert ohne komplexe Schaltungen durch Überlauf zu 0
- **Wichtig**: die Größe der Zahl (Anzahl Bytes) entsprechend der Wortlänge im Auge behalten

Beispiel mit Wortlänge Byte

+4	=	0000 0100	
-4 (-5+1)	=	1111 1011 + 1	= 1111 1100 = -4
-4	=	1111 1100	
+4	+	0000 0100	
		(1) 0000 0000	
	=	0000 0000 = 0	

1.2 Zahlensysteme und binäre Arithmetik

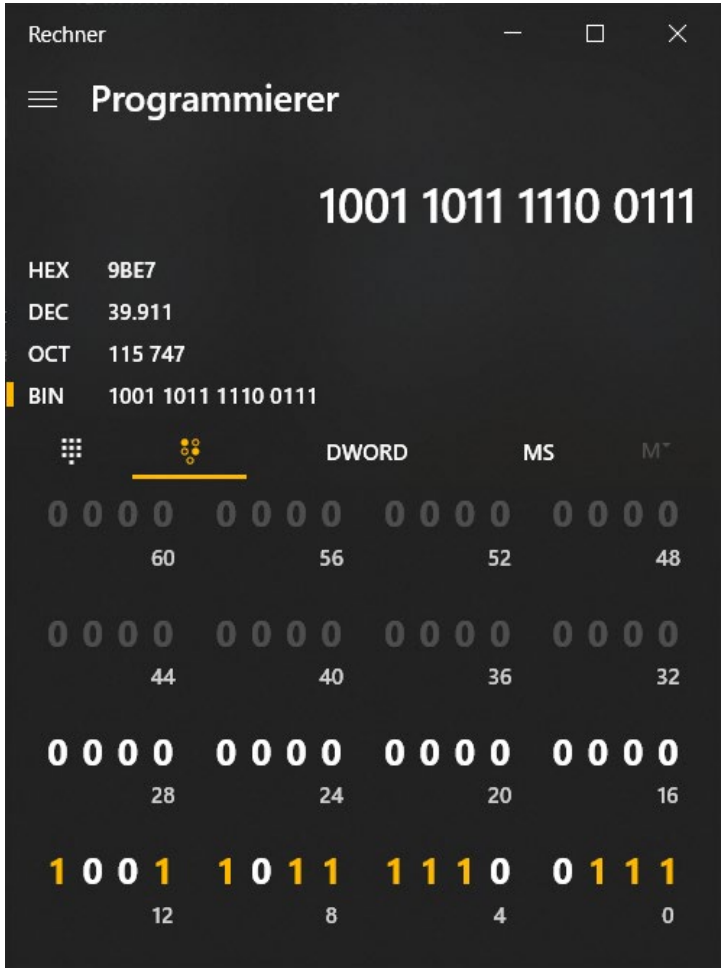
Umwandlung von Zahlen in verschiedene Darstellungssysteme

Beispiel

+ 39.911_{dez}
+ $1001\ 1011\ 1110\ 0111_{bin}$
+ $9BE7_{hex}$

Die <u>Wortlänge entscheidet</u> bei der Umwandlung

+ Datenwort, oder einfach Wort, bestimmte Datenmenge, die ein Computer in der arithmetisch-logischen Einheit des Prozessors in einem Schritt verarbeiten kann.
+ 4 Bit = ½ Byte (Nibble)
+ 8 Bit = 1 Byte
+ 16 Bit = 2 Byte (Word)
+ 32 Bit = 4 Byte (Double Word)
+ 64 Bit = 8 Byte (Quadruple Word)



Eigene Darstellung

1.2 Zahlensysteme und binäre Arithmetik

Umwandlung von Zahlen in verschiedene Darstellungssysteme - Wertetabelle

Dezimal	Dual	Oktal	Hexadezimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

1.2 Zahlensysteme und binäre Arithmetik

Umwandlung von Zahlen in verschiedene Darstellungssysteme

Direkte Methode und Zusammenfassen von Binärstellen

+ Umwandlung von Binärzahlen in das mit dem Dualsystem eng verwandte oktale oder hexadezimale Ziffernsystem
  - Zusammenfassung von jeweils drei binären Stellen in eine oktale Stelle
  - Zusammenfassung von jeweils vier binären Stellen in eine Hexadezimalstelle

+ Vorgehensweise binär in hexadezimal
  - Beginn mit dem niederwertigsten Bit
  - Ist die Anzahl der Binärziffern nicht durch vier teilbar, dann führende Nullen an die Binärzahl entspr. ergänzen
  - Das Ergänzen mit Nullen ändert den Zahlenwert nicht

1.2 Zahlensysteme und binäre Arithmetik

Umwandlung von Zahlen in verschiedene Darstellungssysteme

Direkte Methode und Zusammenfassen von Binärstellen

+ In Beispiel a) wurden zwei und in Beispiel b) drei Nullen ergänzt.
+ Bei Nachkommastellen müssen ggf. am Ende der Zahl Nullen angehängt werden.
+ In Beispiel c) wurde eine Null angehängt.

$$\text{a) } 53_{\text{dez}} = \underbrace{0011}_{3}\,\underbrace{0101}_{5}{}_{\text{bin}} = 35_{\text{hex}}$$

$$\text{b) } 430_{\text{dez}} = \underbrace{0001}_{1}\,\underbrace{1010}_{\text{A}}\,\underbrace{1110}_{\text{E}}{}_{\text{bin}} = 1\text{AE}_{\text{hex}}$$

$$\text{c) } 11{,}625_{\text{dez}} = 1011{,}101_{\text{bin}} = \text{B,A}_{\text{hex}}$$

1.2 Zahlensysteme und binäre Arithmetik

Umwandlung von Zahlen in verschiedene Darstellungssysteme

Direkte Methode und Zusammenfassen von Binärstellen

+ Beispiel: Umwandlung einer Hexadezimalzahl in die zugehörige Binärzahl
  - Ersetzen der einzelnen Hexadezimalziffern durch die entsprechenden vierstelligen Binärzahlen
  - In Wertetabelle nachschauen
  - Führende Nullen werden dabei unterdrückt

$$2E4_{hex} = \underbrace{0010}_{2}\,\underbrace{1110}_{E}\,\underbrace{0100}_{4}{}_{bin} = 1011100100_{bin}$$

Die Dezimaldarstellung findet man durch Aufsummieren der den Binärstellen entsprechenden Potenzen von 2, jeweils multipliziert mit dem Stellenwert 0 oder 1:

$$\begin{aligned} 1011100100_{bin} &= 1\cdot 2^9 + 0\cdot 2^8 + 1\cdot 2^7 + 1\cdot 2^6 + 1\cdot 2^5 + 0\cdot 2^4 + 0\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 0\cdot 2^0 \\ &= 512 + 0 + 128 + 64 + 32 + 0 + 0 + 4 + 0 + 0 = 740_{dez} \end{aligned}$$

1.2 Zahlensysteme und binäre Arithmetik

Umwandlung von Zahlen in verschiedene Darstellungssysteme

Einfache Divisionsmethode zur Umwandlung von Dezimalzahlen in Dualzahlen

+ Verfahren
  - Dividiere die umzuwandelnde Dezimalzahl durch die größte Potenz von 2, die kleiner ist als diese Dezimalzahl und notiere als erste (höchstwertige) Binärstelle eine 1.
  - Das Ergebnis wird nun durch die nächst kleinere Potenz von 2 dividiert; das Resultat (0 oder 1), gibt die nächste Binärstelle an.
  - Auf diese Weise verfährt man weiter, bis schließlich nach der Division durch $2^0 = 1$ das Verfahren abbricht.

+ Das Verfahren kann analog in irgendeinem Zahlensystem für eine beliebige andere Basis angewandt werden.

+ Die Dezimalzahl 116_{dez} in dualer und hexadezimaler Schreibweise

$$
\begin{array}{rl}
116 : 64 & = 1 \\
\underline{-64} & \\
52 : 32 & = 1 \\
\underline{-32} & \\
20 : 16 & = 1 \\
\underline{-16} & \\
4 : 8 & = 0 \\
\underline{-0} & \\
4 : 4 & = 1 \\
\underline{-4} & \\
0 : 2 & = 0 \\
\underline{-0} & \\
0 : 1 & = 0
\end{array}
\qquad \text{Ergebnis: } 1110100_{bin} = 74_{hex}
$$

1.2 Zahlensysteme und binäre Arithmetik

Umwandlung von Zahlen in verschiedene Darstellungssysteme

Horner-Schema und Restwertmethode

+ Eine in einem Zahlensystem zur Basis B dargestellte Zahl z

$$z = a_n B^n + a_{n-1} B^{n-1} + \cdots + a_2 B^2 + a_1 B^1 + a_0 B^0$$

+ Kann durch vollständiges Ausklammern der Basis B in die Horner-Schreibweise gebracht werden:

$$z = ((\ldots(a_n B + a_{n-1})B + \cdots + a_2)B + a_1)B + a_0$$

+ Folge: Fortgesetzte Division einer Dezimalzahl durch B als Divisionsrest liefert die Koeffizienten a_0 bis a_n für die Darstellung dieser Zahl zur Basis B
+ Fazit: Wesentlich eleganter als Divisionsmethode

1.2 Zahlensysteme und binäre Arithmetik

Umwandlung von Zahlen in verschiedene Darstellungssysteme

Die Restwertmethode für Nachkommastellen

+ Verfahren
  - Zunächst den ganzzahligen Anteil, also die Vorkommastellen, umwandeln
  - Anschließend die Nachkommastellen
+ Das Horner-Schema für die Nachkommastellen sieht wie folgt aus:

$$z = \frac{1}{B} \cdot \left(a_{-1} + \frac{1}{B} \cdot \left(a_{-2} + \frac{1}{B} \cdot \left(a_{-3} + \cdots + \frac{1}{B} \cdot \left(a_{-m+1} + \frac{1}{B} \cdot a_{-m} \right) \ldots \right) \right) \right)$$

+ Bei fortgesetzter Multiplikation mit der gewünschten Basis als den ganzzahligen Teil d. h. die Vorkommastellen des Multiplikationsergebnisses erhält man die Koeffizienten (= Ziffern) a_{-1}, a_{-2} …
+ Im darauffolgenden Schritt werden die die Vorkommastellen weggelassen
+ Das Verfahren wird fortgesetzt, bis die Multiplikation eine ganze Zahl ergibt oder bis im Falle eines nicht abbrechenden Dezimalbruchs die gewünschte Genauigkeit erreicht ist.

1.2 Zahlensysteme und binäre Arithmetik

Umwandlung von Zahlen in verschiedene Darstellungssysteme

Beispiel: Zahl 10172_{dez} mit der Restwertmethode in duale und hexadezimale Schreibweise umwandeln

Die Umwandlung in eine Dualzahl folgt aus der fortgesetzten Division durch 2:

10172 : 2 = 5086 Rest 0	317 : 2 = 158 Rest 1	9 : 2 = 4 Rest 1
5086 : 2 = 2543 Rest 0	158 : 2 = 79 Rest 0	4 : 2 = 2 Rest 0
2543 : 2 = 1271 Rest 1	79 : 2 = 39 Rest 1	2 : 2 = 1 Rest 0
1271 : 2 = 635 Rest 1	39 : 2 = 19 Rest 1	1 : 2 = 0 Rest 1
635 : 2 = 317 Rest 1	19 : 2 = 9 Rest 1	

Ergebnis: $10172_{dez} = 10011110111100_{bin}$.

Fortgesetzte Division durch 16 liefert für die Umwandlung in eine Hexadezimalzahl:

10172 : 16 = 635 Rest 12 (= C)
635 : 16 = 39 Rest 11 (= B)
39 : 16 = 2 Rest 7
2 : 16 = 0 Rest 2

Ergebnis: $10172_{dez} = 27BC_{hex}$.

Aus der Horner-Schreibweise von 10172 für die Basis 10 bzw. 16 lassen sich die durchgeführten Rechenschritte nachvollziehen:

$$10172 = (((1 \cdot \mathbf{10} + 0) \cdot \mathbf{10} + 1) \cdot \mathbf{10} + 7) \cdot \mathbf{10} + 2 = ((2 \cdot \mathbf{16} + 7) \cdot \mathbf{16} + 11) \cdot \mathbf{16} + 12.$$

1.2 Zahlensysteme und binäre Arithmetik

Umwandlung von Zahlen in verschiedene Darstellungssysteme

Beispiel: Zahl $39{,}6875_{dez}$ als Dualzahl darstellen

1. Umwandlung des ganzzahligen Anteils

39 : 2 = 19 Rest 1	4 : 2 = 2 Rest 0
19 : 2 = 9 Rest 1	2 : 2 = 1 Rest 0
9 : 2 = 4 Rest 1	1 : 2 = 0 Rest 1

Ergebnis: $39_{dez} = 100111_{bin}$.

2. Umwandlung der Nachkommastellen

$0{,}6875 \cdot 2 = 1{,}375$	1 abspalten
$0{,}3750 \cdot 2 = 0{,}750$	0 abspalten
$0{,}7500 \cdot 2 = 1{,}500$	1 abspalten
$0{,}5000 \cdot 2 = 1{,}000$	1 abspalten (fertig, Ergebnis ganzzahlig)

Ergebnis für die Nachkommastellen: $0{,}6875_{dez} = 0{,}1011_{bin}$.

Insgesamt hat man also: $39{,}6875_{dez} = 100111{,}1011_{bin} = 27{,}B_{hex}$.

Im Hexadezimalsystem ist die Rechnung viel kürzer in nur einem Schritt durchführbar:

$0{,}6875 \cdot 16 = 11{,}000$	11=B abspalten

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Rechenregeln für Binärzahlen analog zu den Rechenregeln für Dezimalzahlen

Ausführung von Algorithmen führt grundsätzlich zu

+ einer Unterteilung des Problems in Teilaufgaben,
+ die unter Verwendung der vier Grundrechenarten
+ und der logischen Operationen
+ gelöst werden können.

Es genügt daher, sich auf die binäre **Addition, Subtraktion, Multiplikation, Division** und die logischen Operationen zu beschränken.

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Logische Operationen

+ In Computersystemen grundsätzlich Bit für Bit Durchführung

+ Zweistellige Operationen
  - Logisches UND (AND, Symbol: $\wedge$)
  - Logisches ODER (OR, Symbol: $\vee$)
  - Exklusives ODER (exclusive OR, XOR), Definition: $a\ XOR\ b\ =\ (a \wedge \neg b) \vee (\neg a \wedge b)$

+ Einstellige Operation
  - Inversion bzw. Negation (NOT, Symbol: $\neg$)

+ Alle anderen logischen Operationen können durch Verknüpfung der Grundfunktionen abgeleitet werden.
  - Siehe hierzu Thema Boolesche Algebra

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Logische Operationen

+ Grundfunktionen sind durch ihre Wahrheitstafeln definiert

OR	$1 \vee 1 = 1$	$0 \vee 1 = 1$	$1 \vee 0 = 1$	$0 \vee 0 = 0$
AND	$1 \wedge 1 = 1$	$0 \wedge 1 = 0$	$1 \wedge 0 = 0$	$0 \wedge 0 = 0$
NOT	$\neg 1 = 0$	$\neg 0 = 1$		
XOR	1 XOR 1 = 0	0 XOR 1 = 1	1 XOR 0 = 1	0 XOR 0 = 0

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Logische Operationen

+ Binärzahlen mit mehr als einer Ziffer werden oft durch logische Operationen verknüpft
+ Verknüpfung erfolgt stellenweise

Beispiel

$$\begin{array}{r} 10011 \\ \vee\ 10101 \\ \hline = 10111 \end{array} \qquad \begin{array}{r} 10011 \\ \wedge\ 10101 \\ \hline = 10001 \end{array} \qquad \begin{array}{r} \\ \neg\ 10101 \\ \hline = 01010 \end{array}$$

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Binäre Addition

+ Rechenregeln
  0+0 = 0
  0+1 = 1
  1+0 = 1
  1+1 = 0 Übertrag 1
+ Regeln sind mit denen des logischen XOR identisch, es kommt lediglich der Übertrag hinzu
+ Anwendung auch auf Binärbrüche

Beispiel

Die Aufgabe $11 + 14 = 25$ soll in binärer Arithmetik gelöst werden:

```
    1011
+   1110
  111        Übertrag
-------------------------
= 11001      Ergebnis
```

Die Aufgabe $151{,}875 + 27{,}625 = 179{,}5$ soll in binärer Arithmetik gelöst werden:

```
   10010111.111
+     11011.101
    111111 11      Übertrag
-------------------------------
= 10110011.100     Ergebnis
```

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Direkte Binäre Subtraktion

+ Rechenregeln
  0-0 = 0
  1-1 = 0
  1-0 = 1
  0-1 = 1 Übertrag -1

Beispiel

Die Aufgabe $13 - 11 = 2$ soll in binärer Arithmetik gelöst werden:

$$\begin{array}{lll} & 1101 & \\ - & 1011 & \\ & -1 & \text{Übertrag} \\ \hline = & 0010 & \text{Ergebnis} \end{array}$$

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Zweierkomplement und Subtraktion

+ Methode zur Subtraktion, die sich einfach mit Hardware realisieren lässt
+ Zahlen werden in Computern als Bitmuster dargestellt
+ Vorzeichen einer Zahl wird durch ein Bit codiert, meist das Bit mit dem höchsten Stellenwert
  - Most Significant Bit, MSB
+ Vorausgesetzt ist eine feste Stellenzahl n
  - In der Regel 8 Bit oder ein Vielfaches davon
  - Der Zahlenbereich das MSB wird nicht mit umfasst

1.2 Zahlensysteme und binäre Arithmetik

Zweierkomplement und Subtraktion

Fall 1:

+ Codierung des MSB einer Zahl auf den Wert 1, wenn diese Zahl negativ ist
  und den Wert 0, wenn die Zahl Null oder positiv ist.

+ Bei $n = 8$ für die Zahlen $+5$ und -5 ergäbe $5_{dez} = 00000101_{bin}$ und $-5_{dez} = 10000101_{bin}$

+ Folge: Rechnen wäre wegen Sonderbehandlung der Vorzeichenbits aufwendig und Regeln der direkten Subtraktion müssten angewandt werden.

Zweierkomplement und Subtraktion

Fall 2:

+ Darstellung negativer Zahlen
+ Rückführung der Subtraktion auf die Addition

+ Lösung:
  - Maschinell mit Invertern sehr einfach zu realisierender Vorgang der bitweisen Invertierung (Stellenkomplement)
  - Positive Zahlen direkt codieren und für negative Zahlen die Stellen invertieren
  - Bei einer Vorzeichenänderung nur eine Inversion nötig
  - Vorzeichenbit wird automatisch mit umfasst

+ Bei $n = 8$ für die Zahlen $+5$ und -5 ergäbe $5_{dez} = 00000101_{bin}$ und $-5_{dez} = 11110101_{bin}$

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Binäre Multiplikation

+ Rechenregeln entsprechen der logischen UND-Verknüpfung zweier Binärziffern:

$$0 \cdot 0 = 0, \qquad 0 \cdot 1 = 0, \qquad 1 \cdot 0 = 0, \qquad 1 \cdot 1 = 1$$

+ Multiplikation mehrstelliger Zahlen wird auf die Multiplikation des Multiplikanden mit den einzelnen Stellen des Multiplikators und **stellenrichtige Addition der Zwischenergebnisse** zurückgeführt.
  Wie im Zehnersystem gewohnt.

+ Beispiele zeigen, dass die Multiplikation durch fortgesetzte Addition ersetzt wird, da die Multiplikation mit den Grundziffern 0 und 1 keinen Aufwand erfordern.

Beispiel

a) Die Aufgabe $10 \cdot 13 = 130$ ist in binärer Arithmetik zu lösen.

$$\begin{array}{l} 1010 \cdot 1101 \\ 1010 \\ \quad 1010 \\ \quad\quad 0000 \\ \quad\quad\quad 1010 \\ \hline 10000010 \qquad \text{Ergebnis} \end{array}$$

b) Die Erweiterung auf Brüche ist nach denselben Regeln ohne weiteres möglich: Die Aufgabe $17{,}375 \cdot 9{,}75 = 169{,}40625$ ist in binärer Arithmetik zu lösen.

$$\begin{array}{l} 10001{,}011 \cdot 1001{,}11 \\ 10001011 \\ \quad 10001011 \\ \quad\quad 10001011 \\ \quad\quad\quad 10001011 \\ \hline 1010100101101 \qquad \text{Zwischenergebnis} \end{array}$$

Nach stellenrichtigem Einfügen des Kommas erhält man das Ergebnis:

$17{,}375_{\text{dez}} \cdot 9{,}75_{\text{dez}} = 10101001{,}01101_{\text{bin}} = 169{,}40625_{\text{dez}}$.

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Binäre Division

+ Wird in Analogie zu dem im Zehnersystem gewohnten Verfahren durchgeführt.

Beispiel

+ Die Aufgabe 20 : 6 = 3,333… soll in binärer Arithmetik gelöst werden.

$$
\begin{array}{l}
10100 : 110 = 11{,}0101\ldots \\
\underline{-110} \\
\quad 1000 \\
\quad \underline{-110} \\
\quad\quad 1000 \\
\quad\quad \underline{-110} \\
\quad\quad\quad \ldots
\end{array}
$$

Man erhält also auch in der Binärdarstellung einen unendlichen, periodischen Bruch.

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Verschieben

+ Multiplikation und Division in digitalen Rechenanlagen wird durch Kombination von Verschieben (*shift*) und Addieren bzw. Subtrahieren ausgeführt.
+ Wird eine Binärzahl mit einer Zweierpotenz 2^k multipliziert, so entspricht dies in Analogie zur Multiplikation mit einer Potenz von 10 im Zehnersystem, lediglich einer Verschiebung dieser Zahl um k Stellen nach links.
+ Jede Multiplikation kann durch eine Kombination von Verschiebungen und Additionen ausgeführt werden.

Beispiel Multiplikation

+ Dezimale Multiplikation $13 \cdot 4 = 52$ lautet im Dualsystem: $1101 \cdot 100 = 110100$
+ Verschiebung der Zahl 1101 um zwei Stellen nach links
+ Anhängen von **zwei Nullen** an der rechten Seite

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Verschieben

+ Wird eine Binärzahl mit einer Zweierpotenz 2^k dividiert, so entspricht dies in Analogie zur Division mit einer Potenz von 10 im Zehnersystem, lediglich einer Verschiebung dieser Zahl um k Stellen nach rechts.

+ Informationsverlust kann auftreten, wenn bei der Division ein Rest (*Carry*) verbleibt, der nicht entsprechend berücksichtigt wird.

Beispiel Division

+ Die Divisionsaufgabe $26 : 4 = 6\ (Rest\ 2)$ lautet im Dualsystem: $11010 : 100 = 110\ (Rest\ 10)$

1.2 Zahlensysteme und binäre Arithmetik

Verschieben

+ Bei der maschinellen Ausführung wird die zu verschiebende Zahl in einem dem Rechenwerk direkt zugeordneten Speicherplatz (*Register, Akkumulator*) abgelegt.
+ Solche Register haben meist ein Übertrags-Bit (*Carry*), in dem das jeweils aus dem Register hinausgeschobene Bit gespeichert wird.
+ Vorgang wird auch als Setzen eines *Flags* bezeichnet

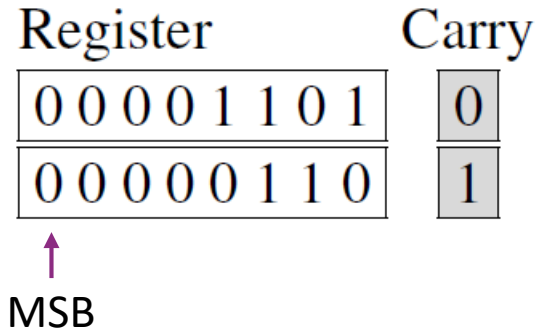
1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Verschieben

Beispiel

+ Verschieben der Binärzahl 1101 um eine Stelle nach rechts.
+ Der obere Bildteil zeigt die Ausgangssituation, der untere Bildteil das Ergebnis nach der Schiebeoperation.
+ Das Übertrags-Bit (*Carry*) wurde in diesem Fall von 0 auf 1 gesetzt.
+ Das am linken Ende des Registers frei werdende MSB wurde mit 0 besetzt
  - Vorgehen wird **logisches Verschieben** bezeichnet.
  - Alternativ bei der Verschiebung nach **rechts** das **MSB reproduzieren**, so dass eine **0** bzw. eine **1** erhalten bleibt. Dieses arithmetische Verschieben ist sinnvoll, wenn das **Vorzeichen** bei einer Verschiebeoperation **erhalten** bleiben soll.



1.2 Zahlensysteme und binäre Arithmetik

Gleitkommazahlen

Ausgangssituation

+ Festkommazahlen mit einer vorab festgelegten Anzahl Vor- und Nachkommastellen lassen wegen der Beschränkung auf eine feste Stellenzahl die Darstellung sehr kleiner oder sehr großer Zahlen nicht zu.

Lösungsansatz

+ Seit Zuse üblich sind *Gleitkommazahlen* bzw. auch *Gleitpunktzahlen* wegen der im englischen Sprachraum üblichen Verwendung eines **Punktes statt des Kommas** in **halblogarithmischer Darstellung**.

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Gleitkommazahlen

Zahlendarstellung

$$m \cdot b^e$$

Mantisse m ist eine Festkommazahl

+ Genau eine Stelle vor dem Dezimalpunkt und eine von Null verschiedene Ziffer
+ Bezeichnung als die ***Normalform***
+ Genauigkeit der Zahldarstellung hängt von der Stellenzahl der Mantisse ab

b^e ist ein Faktor mit Basis b und Exponent e

+ Darstellbarer Zahlenbereich ist abhängig von der Basis und vor allem vom Exponenten

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Gleitkommazahlen

Beispiele

+ Dezimale Gleitkommazahlen mit der Basis $b = 10$:

$$0{,}000112 = 1{,}12 \cdot 10^{-4}$$

$$7123458 = 7{,}123458 \cdot 10^{6}$$

$$-24{,}317 = -2{,}4317 \cdot 10^{1}$$

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Binäre Gleitkommazahlen

Standard IEEE 754-2008 (PDF Download im HKA Intranet)

+ Definiert sowohl binäre Gleitkommazahlen (Basis 2) als auch dezimale Gleitkommazahlen (Basis 10)

$$z = (-1)^s \cdot 1, f \cdot 2^e$$

+ Binäre Gleitkommazahl z

+ Vorzeichenbit s
  - Höchstwertiges Bit (MSB) der binären Gleitkommazahl
  - $s = 0$ wegen $(-1)^0 = 1$ für positives
  - $s = 1$ wegen $(-1)^1 = -1$ für negatives Vorzeichen von z

Binäre Gleitkommazahlen

Standard IEEE 754-2008

+ Definiert sowohl binäre Gleitkommazahlen (Basis 2) als auch dezimale Gleitkommazahlen (Basis 10)

$$z = (-1)^s \cdot 1,f \cdot 2^e$$

+ Exponent e
  - Sog. Charakteristik $c = e + B$
  - B ist eine Verschiebung (*Bias*), die so gewählt ist, dass der Nullpunkt für e in die Mitte des zur Verfügung stehenden Wertebereichs $[0,2B + 1]$ verschoben wird.
  - Darstellung der Exponenten $e = -B$ (entsprechend $c = 0$) und $e = B + 1$ (entsprechend $c = 2B + 1$)

1.2 Zahlensysteme und binäre Arithmetik

Binäre Gleitkommazahlen

Standard IEEE 754-2008

+ Definiert sowohl binäre Gleitkommazahlen (Basis 2) als auch dezimale Gleitkommazahlen (Basis 10)

$$z = (-1)^s \cdot 1, f \cdot 2^e$$

+ Binäre Mantisse $1, f$
  - In Normalform
  - Binäre Nachkommastellen f der Mantisse
  - Führende 1 muss nicht gespeichert werden (verborgene Eins, *Hidden Bit*), ist nach Definition konstant

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Binäre Gleitkommazahlen

+ Kurze Gleitkommazahl verwendet 32 Bit
  - 8 Bit Charakteristik $c = e + 127$
  - Wertebereich $[0, 255]$
  - Bias $B = 127$

Binäre Darstellung vorzeichenbehaftete Zahlen

+ Binärcodierung basiert auf einer Wertebereichsverschiebung: Die Addition einer Verschiebung$B$
+ **Exzesscode** oder auch **Überschuss-Code** (engl. Offset binary)
+ Hier speziell mit $B = 127$ wird bezeichnet als **127-Exzesscode**

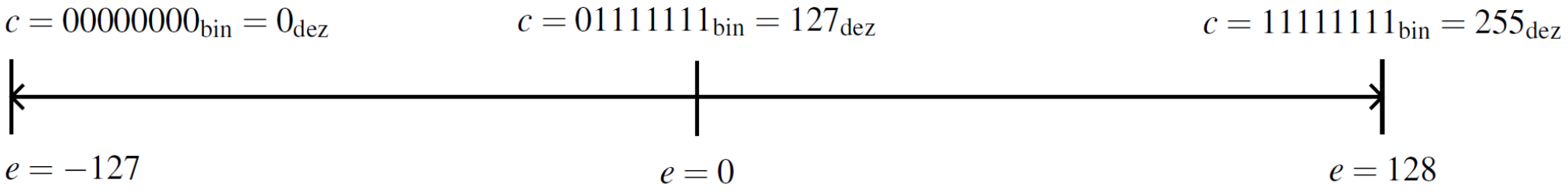


Abb. 1.13 Zur Darstellung des Exponenten e und der Charakteristik c im 127-Exzess-Code

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Binäre Gleitkommazahlen

+ Kurze Gleitkommazahl verwendet 32 Bit

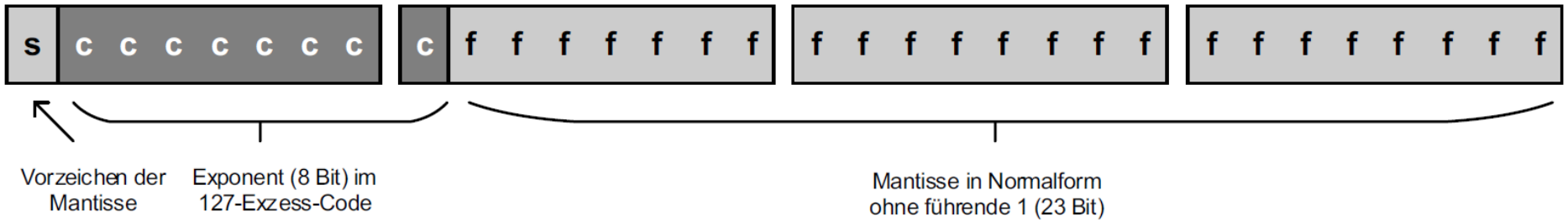


Abb. 1.14 Aufbau einer kurzen Gleitkommazahl nach dem IEEE 754 Standard

+ Mantisse einer kurzen Gleitkommazahl mit 23 Bit für die Nachkommastellen lautet:

$$m = 1, f_0 f_1 \dots f_{22}$$

+ Daraus resultiert eine Genauigkeit von 2^{-24} mit entsprechend 7 signifikanten Dezimalstellen.

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Binäre Gleitkommazahlen

\+ Kurze Gleitkommazahl verwendet 32 Bit

Beispiel: Umwandlung von $148{,}625_{dez}$ in ein binäre Gleitkommazahl

1. Schritt: $148{,}625_{\text{dez}} = 10010100{,}101_{\text{bin}}$
2. Schritt: $10010100{,}101 = 1{,}0010100101 \cdot 2^7$,
   Normalform ist erreicht, der Exponent ist $e = 7$
3. Schritt: Das Vorzeichen der Mantisse ist positiv, das MSB lautet also $s = 0$
4. Schritt: Charakteristik: $c = e + 127 = 134_{\text{dez}} = 10000110_{\text{bin}}$
5. Schritt: Ergebnis: $0\mathbf{1000011}\,\mathbf{0}0010100\;10100000\;00000000_{\text{bin}} = 43\,14\,A0\,00_{\text{hex}}$
   Byte 1 Byte 2 Byte 3 Byte 4

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Binäre Gleitkommazahlen

+ Gleitkommazahlen sind nicht gleichmäßig verteilt
+ Abstand zwischen je zwei benachbarten Gleitkommazahlen wird mit steigendem Betrag immer größer
+ Im gleichen Verhältnis wachsen auch die Präzisionsprobleme.

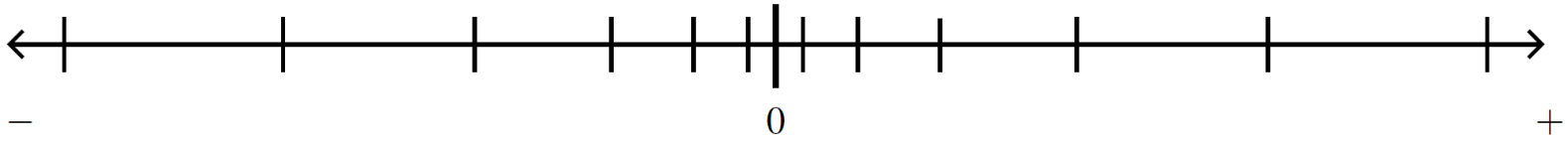


Abb. 1.15 Die Dichte der binären Gleitkommazahlen ist nicht konstant über der Zahlengeraden, sondern es besteht eine Häufung um den Nullpunkt

Vorgehensweise in Python

+ Exakte Vergleiche zweier Gleitkommazahlen sind durch die begrenzte Genauigkeit der Darstellung problematisch.
+ Für einen Vergleich sind nach [4] diese Zahlen vorher zu runden.
  - Modul `math`, Funktion `round(a)` übernimmt kaufmännisches Runden einer Zahl `a`
  - Modul `decimal` kann Zahlen genauer darstellen. Geeigneter für Berechnungen mit Dezimalzahlen

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Binäre Gleitkommazahlen

\+ Lange Gleitkommazahl verwendet 64 Bit

Bit 0 (MSB):	Vorzeichenbit, 0 entspricht positiv, 1 entspricht negativ
Bit 1 bis 11:	11-Bit für die Charakteristik $c = e + 1023$
Bit 12 bis 63:	52 Bit für die Mantisse in Normalform $m = 1, f_0 f_1 \dots f_{53}$
Genauigkeit:	ca. 15 signifikante Dezimalstellen

Denormalisierte Mantissen und weitere Sonderfälle

+ Ausgangssituation
  - Gleitkommazahl mit 32 Nullen hätte den endlichen Wert $z_{min} = 2^{-127}$
  - Folge: Exakte Null ist damit nicht darstellbar. Offene Frage, wann eine Zahl als Unendlich ($\pm\infty$) anzusehen ist.
+ Lösung
  - Für $c = 0$, also $e = -127$ wird die Annahme der normalisierten Mantisse $1, f$ fallen gelassen und durch ***denormalisierte*** Mantissen $0, f$ mit dem Exponenten $e = -126$ ersetzt.
  - Folge: Kleinste positive Gleitkommazahl mit normalisierter Mantisse ist:

$$z_{min} = 2^{-126} \approx 1{,}1754943508 \cdot 10^{-38}$$

  - Denormalisierte Gleitkommazahlen mit $0, f \cdot 2^{-126}$ umfassen den Wertebereich

$$\pm 2^{-149}\ bis \pm (1 - 2^{-23}) \cdot 2^{-126} \approx 1{,}1754942107 \cdot 10^{-38}$$

  - Fazit: $f = 0$ hat den exakten Zahlenwert $z = 0$, wenn alle 32 Bit den Wert 0 annehmen.

1.2 Zahlensysteme und binäre Arithmetik

Denormalisierte Mantissen und weitere Sonderfälle

+ Bei Gleitkommazahlen gibt es, je nach Vorzeichenbit, sowohl eine positive $+0{,}0$ als auch eine negative Null $-0{,}0$
+ Nach IEEE-Norm muss das Rechenwerk dafür sorgen, dass diese trotz unterschiedlicher Binärdarstellung bei Vergleichsoperationen als gleich behandelt werden.
+ Negative Null ergibt sich beispielsweise, wenn Zahlen entstehen, die kleiner als die betragsmäßig kleinste darstellbare denormalisierte Gleitkommazahl sind.
  In diesem Fall, dem sog. Unterlauf (*Underflow*), wird auf Null gerundet, das Vorzeichen bleibt dabei erhalten.
+ War die zu rundende Zahl positiv, entsteht $+0{,}0$
+ War die zu rundende Zahl negativ, entsteht $-0{,}0$

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Denormalisierte Mantissen und weitere Sonderfälle

+ Unendlich
  - Festgelegte Zahl $1{,}0 \cdot 2^{128}$
  - Nachkommastellen der Mantisse sind alle Null und es gilt $c = 255$
  - Größte Zahl lautet damit $z_{max} = (2 - 2^{-23}) \cdot 2^{127} \approx 3{,}4028234664 \cdot 10^{38}$
  - Zwischen $+\infty$ und $-\infty$ wird durch das Vorzeichenbit entschieden.
  - Entstehung von unendlich bspw. bei der Division $\frac{x}{0}$ mit $|x| > 0$

+ Norm zur Kennzeichnung unerlaubter Zahlenbereiche
  - $NaN, Not\ a\ Number$
  - Zahlen der Art $1, f \cdot 2^{128}$ mit $f > 0$
  - Entstehung dieser Zahlen insbesondere bei $\infty \pm \infty$, $\frac{\infty}{\infty}$, $\sqrt{-|x|}$ mit $|x| > 0$ und ähnlichen Operationen

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Rechnen mit Gleitkommazahlen

+ Gleitkommaarithmetik ist nicht *assoziativ* und *distributiv*.
  - Eine (zweistellige) Verknüpfung ist assoziativ, wenn die Reihenfolge der Ausführung keine Rolle spielt.
  - *Distributives* Verhalten zweier (zweistelligen) Verknüpfungen bei der Auflösung von Klammern zueinander, *Ausmultiplizieren*. Ebenso das umgangssprachliche *Ausklammern* oder *Herausheben.*
+ Die Ungültigkeit des Assoziativgesetzes der Addition wird hier anhand von 8-stelligen Dezimalzahlen demonstriert

$$(1{,}1111113 \cdot 10^7 + (-1{,}1111111 \cdot 10^7)) + 7{,}5111111 \cdot 10^0 =$$
$$2{,}0000000 + 7{,}5111111 = 9{,}5111111$$
$$1{,}1111113 \cdot 10^7 + (-1{,}1111111 \cdot 10^7 + 7{,}5111111 \cdot 10^0) =$$
$$1{,}1111113 \cdot 10^7 + (-1{,}1111111 \cdot 10^7 + 0{,}0000008 \cdot 10^7) =$$
$$1{,}1111113 \cdot 10^7 + (-1{,}1111103 \cdot 10^7) = 10{,}000000$$

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Rechnen mit Gleitkommazahlen

+ Addition und Subtraktion
  1. Exponenten angleichen, indem die Mantisse des Operanden mit dem kleineren Exponenten entsprechend verschoben wird. Dabei können Stellen verloren gehen, d. h. es entsteht dann ein Rundungs- oder Abbruchfehler.
  2. Anschließend werden die Mantissen addiert bzw. subtrahiert.

+ Multiplikation
  - Die Mantissen der Operanden werden multipliziert, die Exponenten addiert.

+ Division
  - Die Mantissen der Operanden werden dividiert, der neue Exponent ergibt sich als Differenz des Exponenten des Dividenden und des Divisors.

1.2 Zahlensysteme und binäre Arithmetik

Binäre Arithmetik

Rechnen mit Gleitkommazahlen

- Nach allen Operationen ist zu prüfen, ob die Ergebnisse in der Normalform vorliegen, ggf. ist durch Verschieben wieder zu normalisieren.
- Die angegebenen betragsmäßig kleinste Zahl z_{min} und die betragsmäßig größte Zahl z_{max} sind zu berücksichtigen.
- Resultate arithmetischer Gleitkomma-Operationen sind nicht notwendigerweise wieder Gleitkommazahlen.
- Sie werden daher zu den nächstgelegenen Gleitkommazahlen gerundet.
- Wird dabei z_{max} überschritten, ergibt sich ein Überlauf (*Overflow*).

1.2 Zahlensysteme und binäre Arithmetik

Größeneinheiten

SI Name	Symbol	Größe (dez)	Größe (bin)	Präfix in Byte	Diff. in %	Wert (dez)
Kilo	K	$1000^1 = 10^3$	$1024^1 = 2^{10}$	Kibi	2,4	1.024
Mega	M	$1000^2 = 10^6$	$1024^2 = 2^{20}$	Mebi	4,9	1.048.576
Giga	G	$1000^3 = 10^9$	$1024^3 = 2^{30}$	Gibi	7,4	1.073.741.824
Tera	T	$1000^4 = 10^{12}$	$1024^4 = 2^{40}$	Tebi	10,0	1.099.511.627.776
Peta	P	$1000^5 = 10^{15}$	$1024^5 = 2^{50}$	Pebi	12,6	1.125.899.906.842.624
Exa	E	$1000^6 = 10^{18}$	$1024^6 = 2^{60}$	Exbi	15,3	1.152.921.504.606.846.976
Zetta	Z	$1000^7 = 10^{21}$	$1024^7 = 2^{70}$	Zebi	18,1	1.180.591.620.717.411.303.424
Yotta	Y	$1000^8 = 10^{24}$	$1024^8 = 2^{80}$	Yobi	20,9	1.208.925.819.614.629.174.706.176
Ronna	R	$1000^9 = 10^{27}$	$1024^9 = 2^{90}$	Robi	23,8	1.237.940.039.285.380.274.899.124.224
Quetta	Q	$1000^{10} = 10^{30}$	$1024^{10} = 2^{100}$	Quebi	26,7	1.267.650.600.228.229.401.496.703.205.376

Metric (SI) Prefixes

1.2 Zahlensysteme und binäre Arithmetik

Größeneinheiten

Übungsbeispiel

+ Kapazität einer externen USB-Festplatte: $1\ TB = 2^{40}\ Byte = 2^{40} * 2^3\ Bit = 2^{43}\ Bit$

+ USB 2.0 Übertragungsgeschwindigkeit: $480\ Mbit/s \cong 2^9 * 2^{20} Bit/s = 2^{29} Bit/s$

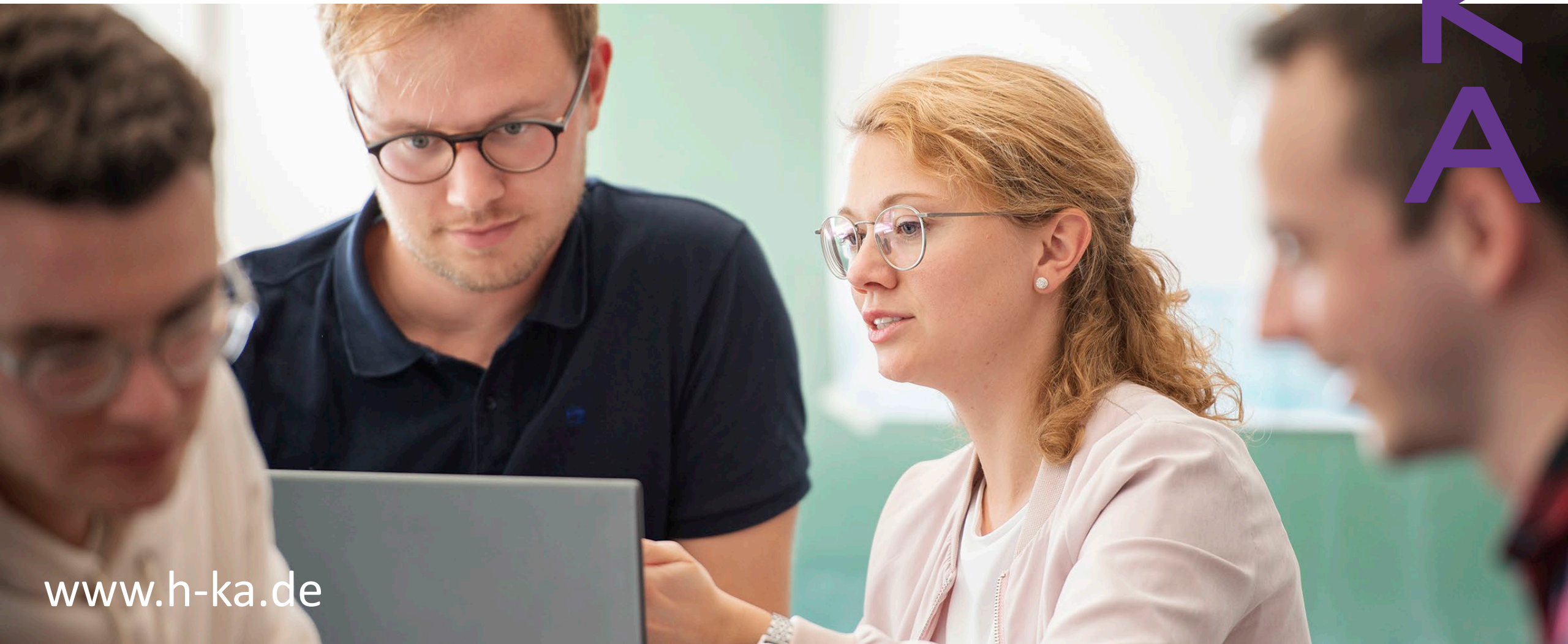
+ Dauer zum Beschreiben der externen Festplatte unter idealen Bedingungen: $\frac{2^{43}}{2^{29}} s = 2^{14}\ s = 16.384\ s \cong 4{,}5\ h$

Legende

+ TB = Terabyte, $Mbit$ = Megabit, s = Sekunde, h = Stunden

Hochschule Karlsruhe
University of
Applied Sciences

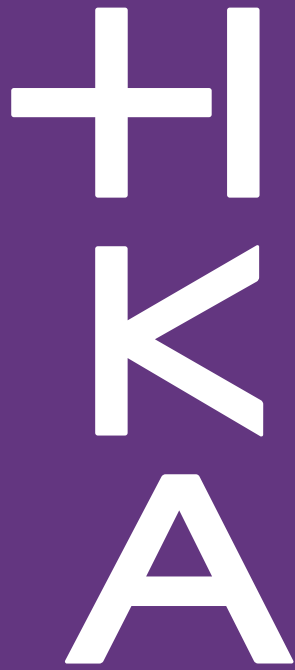
Fakultät für
Informatik und
Wirtschaftsinformatik



www.h-ka.de

Hochschule Karlsruhe
University of
Applied Sciences

Fakultät für
Informatik und
Wirtschaftsinformatik



www.h-ka.de