Kapitel 2

Zahlendarstellung und Kodierung

- Zahlensysteme
- Darstellung negativer Zahlen
- Darstellung Fest- und Fließkommazahlen
- Kodierungen zur Zahlen- und Zeichendarstellung
- Fehlerkorrigierende Kodes

Wdh. Festkommazahlen

Vereinbarung:

- Das Komma sitzt innerhalb des Maschinenwortes, das eine Dualzahl enthalten soll, an einer festen Stelle
- Meist setzt man das Komma hinter die letzte Stelle
- Andere Zahlen können durch entsprechende Maßstabsfaktoren in die gewählte Darstellungsform überführt werden
- **Negative Zahlen:** meist Zweierkomplement-Darstellung

Wdh. Gleitkomma-Maschinenformat

$$X = \pm \mathit{Mantisse} \cdot b^{\mathit{Exponent}}$$

$\overset{31}{y}$	$\overset{30}{y-1}$ ... $\overset{23}{x}$	$\overset{22}{x-1}$ ... 0
Vz	Charakteristik	Mantisse

$$X = (-1)^{Vz} * (0{,}Mantisse) * b^{Exponent}$$

$$Exponent \;= Charakteristik - \underbrace{b^{(\overset{30}{y}-1)-\overset{23}{x}}}_{\substack{7 \\ 128}}$$

Wdh. Gleitkomma-Darstellung

- Bei der **Mantisse** ist die Lage des Kommas wieder durch Vereinbarung festgelegt (meist links vom MSB).
- Der **Exponent** ist eine ganze Zahl, die in Form ihrer Charakteristik dargestellt wird.
- Sowohl für die Charakteristik als auch für die Mantisse wird im Rechner ein **feste Anzahl von Speicherstellen** festgelegt.
- Die Länge der Charakteristik y-x bestimmt die Größe des Zahlenbereichs, die Länge der Mantisse x die Genauigkeit der Darstellung.

Wdh. Normalisierung (1)

Eine Gleitkommazahl heisst **normalisiert**, wenn für den Wert der Mantisse gilt:

$$\frac{1}{b} \leq 0{,}Mantisse \;<\; 1$$

→ **In dualer Darstellung ist die erste Stelle nach dem Komma gleich 1.**

Ausnahme:

Bei der Zahl 0 sind alle Stellen der Mantisse gleich Null

Wdh. Normalisierung (2)

- Legt man für die Zahl 0 ein spezielles Bitmuster fest, ist die erste Stelle der Mantisse in normalisierter Form immer gleich 1 (d.h. 0,1 . . .)
- Die erste Stelle der Mantisse braucht im Maschinenformat gar nicht erst dargestellt zu werden:

  → **Man spart ein Bit bei der Speicherung oder gewinnt bei gleichem Speicherbedarf ein Bit an Genauigkeit.**
- Bei arithmetischen Operationen und bei der Konversion in andere Darstellungen darf diese Stelle natürlich nicht vergessen werden.

Warum Normalisierung?

- Beispiel: Periodische Zahlen

  $0{,}1_{10} = (0{,}000\ 1100\ 1100\ \ldots)_2$
- Darstellung im Rechner mit endlicher Stellenanzahl
- Man behält möglichst viele signifikante Stellen, wenn man die führenden Nullen „abschneidet“

  → $0{,}1_{10} = 0{,}11001100 \ldots \times 2^{-3}$

Beispiel (1)

Drei verschiedene **32**-Bit-Zahlenformate mit Basis $b = 2$

Die Zahl 7135_{10} wird in jedem dieser Formate dargestellt.

7135_{10} = $0001\ 1011\ 1101\ 1111_2$

a) Festkommadarstellung mit Zweierkomplement:

31	30 ... 0
0	000 0000 0000 0000 0001 1011 1101 1111

↑ Vz

$0\ 000\ 0000\ 0000\ 0000\ 0001\ 1011\ 1101\ 1111_2 = 0000\ 1BDF_{16}$

Beispiel (2)

b) Gleitkommadarstellung, normalisiert:

$7135_{10} = 0\ 000\ 0000\ 0000\ 0000\ 0001\ 1011\ 1101\ 1111_2$

$= 0{,}1\ 1011\ 1101\ 1111_2 \cdot 2^{13}$

- Komma um 13 Stellen nach links verschieben
- Entspricht Multiplikation mit 2^{-13}
- Also kompensation durch Multiplikation mit 2^{13}
- Darstellung mit Charakteristik

$$2^{13} = 2^{\overset{\uparrow}{141} - 128}$$

Charakteristik

Beispiel (3)

- $141_{10} = 128 + 8 + 4 + 1 = 2^7 + 2^3 + 2^2 + 2^0$

  $= 1000\ 1101_2$
- Vorzeichen: 0

31	30 ... 23	22 ... 0
0	100 0110 1	110 1111 0111 1100 0000 0000
Vz	Charakteristik	Mantisse

$0\ 100\ 0110\ 1\ 110\ 1111\ 0111\ 1100\ 0000\ 0000_2 = 46EF\ 7C00_{16}$

Beispiel (4)

c) Gleitkommadarstellung, normalisiert, erste "1" implizit

31	30 ... 23	22 ... 0
0	100 0110 1	110 1111 0111 1100 0000 0000

⇩

31	30 ... 23	22 ... 0
0	100 0110 1	101 1110 1111 1000 0000 0000
Vz	Charakteristik	Rest-Mantisse

$0\ 100\ 0110\ 1\ 101\ 1110\ 1111\ 1000\ 0000\ 0000_2 = 46DE\ F800_{16}$

Beispiel: Darstellbarer Zahlenbereich (1)

Anzahl darstellbarer Zahlen (Bitkombinationen) in allen drei Fällen gleich (2^{32})

aber

Bereich und damit Dichte der darstellbaren Zahlen auf dem Zahlenstrahl ist sehr unterschiedlich

Beispiel: Darstellbarer Zahlenbereich (2)

- Format a:

31	30 ... 0

- Größte positive Zahl

$$Z = \overset{31}{0}\underbrace{\overset{30}{1}\overset{29}{1}1 \ldots \overset{1}{1}\overset{0}{1}_2}_{31\ \text{"1"}} = 2^{30} + 2^{29} + \ldots + 2^1 + 2^0$$

$$2Z = \quad = 2^{31} + 2^{30} + 2^{29} + \ldots + 2^1$$

$$Z = 2Z - Z = 2^{31} - 2^0 = 2^{31} - 1$$

Beispiel: Darstellbarer Zahlenbereich (3)

- kleinste negative Zahl

$$Z = \overset{31}{1}\underbrace{\overset{30}{0}\overset{29}{0}0 \ldots \overset{1}{0}\overset{0}{0}_2}_{31\ \text{"0"}}$$

$$= -1 \cdot 2^{31} + 0 \cdot 2^{30} + 0 \cdot 2^{29} + \ldots + 0 \cdot 2^1 + 0 \cdot 2^0$$

$$= -2^{31}$$

Zahlen zwischen -2^{31} und $2^{31}-1$

Beispiel: Darstellbarer Zahlenbereich (4)

- Format b:

31	30 ... 23	22 ... 0
Vz	Charakteristik	Mantisse

- Größte positive Zahl: Mantisse $M = 0{,}\underbrace{\overset{22}{1}\overset{21}{1} \ldots \overset{1}{1}\overset{0}{1}}_{23\ \text{"1"}}{}_2 = 2^{-1} + 2^{-2} + \ldots + 2^{-23}$

$$2M = 1 + 2^{-1} + 2^{-2} + \ldots + 2^{-22}$$

$$M = \quad 2^{-1} + 2^{-2} + \ldots + 2^{-22} + 2^{-23}$$

$$M = 2M - M = 1 - 2^{-23}$$

Exponent: $E = 255 - 128 = 127$

$$M \cdot 2^E = (1 - 2^{-23})\, 2^{127}$$

Beispiel: Darstellbarer Zahlenbereich (5)

- Kleinste positive Zahl:
  - Mantisse $M = 0{,}\overset{22}{1}\overset{21}{0} \ldots \overset{1}{0}\overset{0}{0}_2 = 2^{-1} = 0{,}5_{10}$
  - Exponent $E = 0 - 128 = -128$

    $M \cdot 2^E = 0{,}5 \cdot 2^{-128}$

Negative Zahlen:	$-(1-2^{-23}) \cdot 2^{127}$ ...	$-0{,}5 \cdot 2^{-128}$,
Positive Zahlen:	$0{,}5 \cdot 2^{-128}$ ...	$(1-2^{-23}) \cdot 2^{127}$,
und	***Null***	

Beispiel: Darstellbarer Zahlenbereich (6)

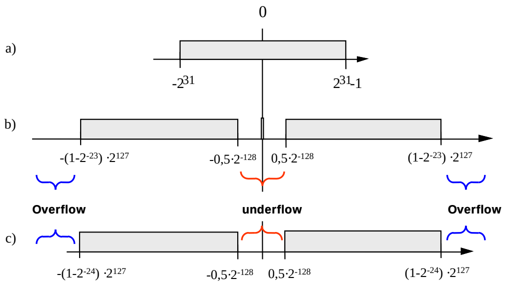
- Format c:

31	30 … 23	22 … 0
Vz	Charakteristik	Rest-Mantisse

negative Zahlen: $-(1-2^{-24})\cdot 2^{127}$ … $-0{,}5\cdot 2^{-128}$
positive Zahlen $0{,}5\cdot 2^{-128}$ … $(1-2^{-24})\cdot 2^{127}$

Die Null kann nicht dargestellt werden

Beispiel: Darstellbarer Zahlenbereich (7)



Charakteristische Zahlen

Um verschiedene Gleitkomma-Darstellungen miteinander vergleichen zu können, definiert man drei charakteristische Zahlen:

- **maxreal** ist die größte darstellbare normalisierte positive Zahl
- **minreal** ist die kleinste darstellbare normalisierte positive Zahl
- **smallreal** ist die kleinste Zahl, die man zu 1 addieren kann, um einen von 1 verschiedenen Wert zu erhalten

Beispiel

In Format b) im letzten Beispiel

31	30 … 23	22 … 0
Vz	Charakteristik	Mantisse

maxreal $= (1-2^{-23})\cdot 2^{127}$ **minreal** $= 0{,}5\cdot 2^{-128}$

Die Zahl 1 wird normalisiert als $0{,}5\cdot 2^1$ dargestellt
$= 0{,}1_2\ 2^1$

Die nächstgrößere darstellbare Zahl hat in der Mantisse zusätzlich zur 1 in Bit 22 eine 1 in Bit 0

➔ **smallreal** $= 0{,}00000000000000000000001_2\cdot 2^1$, also
smallreal $= 2^{-23}\cdot 2^1 = 2^{-22}$

Ungenauigkeiten

- Differenz zwischen zwei aufeinanderfolgenden Zahlen wächst bei Gleitkommazahlen exponentiell mit der Größe der Zahlen, während sie bei Festkommazahlen konstant ist
- Bei der Darstellung großer Zahlen ergibt sich eine hohe Ungenauigkeit

➔ Die Gesetzmäßigkeiten, die für reelle Zahlen gelten, werden für Maschinendarstellungen verletzt. (auch wenn diese Zahlen in einer höheren Programmiersprache **real** heißen)

Beispiel

Das Assoziativgesetz $(x + y) + z = x + (y + z)$ gilt selbst dann nicht unbedingt, wenn kein *overflow* oder *underflow* auftritt.

Beispiel: $x = 1;$ $y = z = smallreal/2$

$$(x + y) + z = (1 + smallreal/2) + smallreal/2 = 1 + smallreal/2 = 1$$

$$x + (y + z) = 1 + (smallreal/2 + smallreal/2) = 1 + smallreal \neq 1$$

Problematik unterschiedlicher Definitionen

- Es existieren beliebig viele Möglichkeiten, selbst mit einer festen Wortbreite unterschiedliche Gleitkommaformate zu definieren:
  - unterschiedliche Basis b
  - Darstellung der Null
  - Anzahl der Stellen für Charakteristik und Mantisse
- Es existierten (bis Mitte der 80er Jahre) viele verschiedene, herstellerabhängige Formate
- Man konnte mit dem gleichen Programm auf unterschiedlichen Rechnern sehr unterschiedliche Ergebnisse erhalten

➔ **Normierung erforderlich**

Normierung (IEEE-Standard)

IEEE-P 754-Floating-Point-Standard

In vielen Programmiersprachen lassen sich Gleitkomma-zahlen mit verschiedener Genauigkeit darstellen

z. B. in C: float
double
long double

➔ **Der IEEE Standard definiert mehrere Darstellungsformen**

IEEE single: 32 Bit
IEEE double: 64 Bit
IEEE extended: 80 Bit

IEEE-P 754-Floating-Point-Standard

32-Bit Maschinenformate des IEEE-Standards:

31	30 … 23	22 … 0
Vz	Charakteristik	Mantisse
	8 Bit	23 Bit

64-Bit Maschinenformate des IEEE-Standards:

63	62 … 52	51 … 0
Vz	Charakteristik	Mantisse
	11 Bit	52 Bit

Eigenschaften des IEEE-P 754

- Basis b ist gleich 2
- Erstes Bit der Mantisse wird implizit zu 1 angenommen, wenn die Charakteristik nicht nur Nullen enthält
- **Normalisierung:** Erstes Bit der Mantisse (die implizite 1) steht **vor** dem Komma.
- Exponent gleich bei Charakteristik 0 und 1
  - Charakteristik 0: Erstes Bit der Mantisse explizit dargestellt → auch die Null ist darstellbar

Eigenschaften des IEEE-P 754

- Sind alle Bits der Charakteristik gleich 1, signalisiert dies eine Ausnahmesituation

  Wenn zusätzlich die Mantisse gleich Null ist, wird die Situation "overflow" (bzw. die "Zahl" $\pm\infty$) kodiert. Dies erlaubt es dem Prozessor, eine Fehlerbehandlung einzuleiten.

- Intern arbeiten Rechner nach dem IEEE-Standard mit 80 Bit, um Rundungsfehler unwahrscheinlicher zu machen

Zusammenfassung des 32 (64)-Bit-IEEE-Formats

Charakteristik	Zahlenwert
0 (0)	$(-1)^{Vz}\ 0{,}Mantisse\cdot 2^{-126\ (-1022)}$
1 (1)	$(-1)^{Vz}\ 1{,}Mantisse\cdot 2^{-126\ (-1022)}$
...	$(-1)^{Vz}\ 1{,}Mantisse\cdot 2^{Charakteristik-127\ (-1023)}$
254 (2046)	$(-1)^{Vz}\ 1{,}Mantisse\cdot 2^{127\ (1023)}$
255 (2047)	$Mantisse = 0$: $(-1)^{Vz}\ \infty$ overflow
255 (2047)	$Mantisse \neq 0$: NaN (not a number)

Ist Charakteristik 0, so wird Bitfolge als denormalisierte Zahl mit explizter Darstellung aller Bits der Mantisse interpretiert

$$(-1)^{Vz}\cdot \underbrace{Mantisse}_{\text{explizit mit Komma und 1. Stelle}}\ 2^{0-127}$$

Linksverschiebung des Kommas um eine Stelle

$$= (-1)^{Vz}\cdot 0{,}Mantisse\cdot 2^1\, 2^{-127}$$

$$= (-1)^{Vz}\ 0{,}Mantisse\cdot 2^{-126}$$

IEEE-P 754 Standard (32-Bit-Format)

$$X = \pm\, 0{,}Mantisse\cdot b^{Exponent}$$

31	30 … 23	22 … 0
Vz	Charakteristik	Mantisse

$$Dezimalzahl = (-1)^{Vz} * (1{,}Mantisse)_2 * b^{Exponent_{10}}$$

$$Exponent = Charakteristik - b^{(y-1)-x} + 1$$

IEEE 754: ***b = 2***

Literatur

- IEEE Computer Society:
  IEEE Standard for Binary Floating-Point Arithmetic
  ANSI/IEEE Standard 754-1985, SIGPLAN Notices, Vol. 22, No. 2, pp 9-25, 1978
- D. Goldberg:
  What every computer scientist should know about floating point arithmetic
  ACM Computing Suveys, Vol. 13, No. 1, pp. 5-48,1991

2.4 Kodierungen zur Zahlen- und Zeichendarstellung

Umgehen der aufwändigen Umwandlung von Dezimalzahlen in Dualzahlen: Dezimalzahl ziffernweise in eine Binärdarstellung überführen

- Um 10 Ziffern kodieren zu können benötigt man $\lceil \mathbf{ld\ 10} \rceil = \mathbf{4}$ Bit. Eine solche Gruppe von 4 Bit wird **Tetrade** genannt, eine so entstehende Kodierung **Tetradenkodierung**.
- 6 der 16 möglichen Kodierungen stellen keine gültige Dezimalziffer dar, sie heißen daher **Pseudotetraden**.
- Nimmt man zur Kodierung der zehn Dezimalziffern ihr Dualäquivalent, erhält man Zahlen in der **BCD-Kodierung** (engl. Binary coded decimal)

BCD-Kodierung

Beispiel:

Dezimalzahl **8127**

als BCD-Zahl: 1000 0001 0010 0111$_{BCD}$

als Dualzahl: 1111110111111$_2$

Nachteile der BCD-Kodierung:

Suboptimale Speicherplatzausnutzung und Probleme bei der Ausführung arithmetischer Operationen

Gray-Kodierung

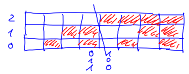
- Aufeinanderfolgende Zahlen sind so durch Binärzeichen dargestellt, dass sich stets nur ein einziges Binärzeichen ändert
- Kodierungen mit dieser Eigenschaft heißen **einschrittige** Kodierungen
- Diese Eigenschaft bietet Vorteile bei der A/D-Wandlung und für mechanische Abtaster
- Die Stellen besetzen bei der Gray-Kodierung keine feste Stellenwertigkeit
  - ➡ Die Ausführung arithmetischer Operationen ist recht schwierig

Gray-Kodierung: Tabelle

Dezimal	Gray-Codierung	Dezimal	Gray-Codierung
0	**0** 0 0 0	8	**1** 1 0 0
1	0 0 0 **1**	9	1 1 0 **1**
2	0 0 **1** 1	10	1 1 **1** 1
3	0 0 1 **0**	11	1 1 1 **0**
4	0 **1** 1 0	12	1 **0** 1 0
5	0 1 1 **1**	13	1 0 1 **1**
6	0 1 **0** 1	14	1 0 **0** 1
7	0 1 0 **0**	15	1 0 0 **0**

Positionsencoder:

- Kodierung mit Dualzahlen



⟹ Problem bei Schrägstellung des Abtasters!

- Kodierung mit Grey-Code