Grundlagen der Informatik

Zahlensysteme

Prof. Dr. Peter Jüttner

Inhalte

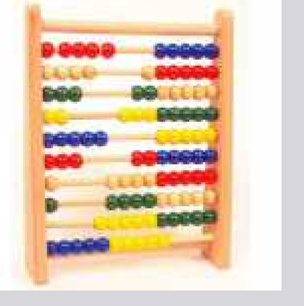
- Einleitung
- Information & Nachricht
- Zahlensysteme
- Codierungen
- Logik
- Rechnerarchitekture
- Endliche Automaten
- …

Darstellung von Zahlen

Motivation

- Mechanische Rechner als Vorfahren der Computer
- Computer als nach wie vor „Rechenmaschine“
- Darstellung von Zahlen im Computer „computergerecht“ nicht “menschengerecht“

$\Rightarrow$ Zahlen sind eine der bzw. ggf. die wichtigsten Informationen, die von einem Rechner zu verarbeiten sind



Darstellung von Zahlen

Darstellung einer ganzen Zahl zu einer Basis 10 (Dezimalsystem)

Beispiele: 10, 123, 5144, 1000456

Eine Dezimalzahl d läßt sich auch anders darstellen in der Form

$$\mathbf{d = d_{n-1} * 10^{n-1} + d_{n-2} * 10^{n-2} + \ldots + d_1 * 10 + d_0}$$

wobei $d_i \in \{0, 1, 2, \ldots, 9\}$

Darstellung von Zahlen

Darstellung einer ganzen Zahl zu einer Basis 10 (Dezimalsystem)

Beispiele:

$10 = 1*10 + 0$

$123 = 1*10^2+2*10+3$

$5144 = 5*10^3+1*10^2+4*10+4$

$1000456 = 1*10^6+4*10^2*5*10+6$

Darstellung von Zahlen

Darstellung einer ganzen Zahl zu einer Basis b

Gegeben seien natürliche Zahlen b mit $b \geq 2$ (die „Basis") und n (die „Länge"). Dann kann jede ganze Zahl z mit $0 \leq z \leq b^n - 1$ eindeutig in der Form

$$\mathbf{z = z_{n-1} \cdot b^{n-1} + z_{n-2} \cdot b^{n-2} + \ldots + z^1 \cdot b + z_0}$$

mit $z_k \in \{0,1, \ldots, b-1\}$ dargestellt werden.

Dabei heißt z_k die k-te ***Ziffer*** (zur Basis b) von z und b^k heißt die ***Wertigkeit*** der Ziffer z_k.

Darstellung von Zahlen

Beweis durch vollständige Induktion über z

1.) $z = 0 \Rightarrow z = z_0$ für jede beliebige Basis $b \geq 2$

2.) Sei z darstellbar durch

$z = z_{n-1} \cdot b^{n-1} + z_{n-2} \cdot b^{n-2} + \ldots + z_1 \cdot b + z_0$

$\Rightarrow z+1 = (z_{n-1} \cdot b^{n-1} + z_{n-2} \cdot b^{n-2} + \ldots + z_1 \cdot b + z_0) + 1$

1. Fall: $z_0 < b-1$

$\Rightarrow z+1 = z_{n-1} \cdot b^{n-1} + z_{n-2} \cdot b^{n-2} + \ldots + z_1 \cdot b + (z_0+1)$

Darstellung von Zahlen

Beweis durch vollständige Induktion

2. Fall: Es gibt ein i: $0 \leq i < n-1$ und $z_i = b-1$ und $z_{i+1} < b-1$

$\Rightarrow z+1 = z_{n-1} \cdot b^{n-1} + z_{n-2} \cdot b^{n-2} + \ldots + (z_{i+1}+1) \cdot b^{i+1}$

3. Fall: $z_i = b-1$ für alle i: $0 \leq i \leq n-1$

$\Rightarrow z+1 = 1 * b^n$

qed.

Darstellung von Zahlen

Praktisch relevante Basen:

- b = 10 (**Dezimaldarstellung**) — im Alltag übliche Darstellung mit den Ziffern 0, . . . ,9: 407_{10}

- b = 2 (**Binär-/Dualdarstellung**) — in Computern dominierende Repräsentation mit den Ziffern 0 und 1: 110010111_2

Darstellung
der Basis

Darstellung von Zahlen

Praktisch relevante Basen:

- b = 16 (**Hexadezimaldarstellung**) — leichter lesbare Alternative zur Binärdarstellung mit den Ziffern 0, . . . ,9,A(= 10),B,C,D,E,F(= 15): $151F_{16}$

- b = 8 (**Oktaldarstellung**) — Alternative zur Binärdarstellung mit den Ziffern 0, . . . ,7: 12437_8

Darstellung von Zahlen

Praktisch relevante Basen:

- b = 1 (**Unäre Darstellung)**
  Darstellung mit der Ziffer 1:
  11111111_1
- Heute nur noch selten verwendet

Darstellung von Zahlen

Darstellung von Binärzahlen als Bitsequenzen

(zunächst nur Zahlen ≥ 0)

1	0	0	1	1	0	0	0

Binärzahl der Länge 8 Bit (1 Byte)

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

Binärzahl der Länge 16 Bit (2 Byte)

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

Binärzahl der Länge 32 Bit (4 Byte)

Darstellung von Zahlen

Rechnen mit Binärzahlen (zunächst nur Zahlen ≥ 0)

Addition „wie gewohnt“:

0+0 = 0

0+1 = 1

1+1 = 10

```
  100101
 +101001
  1   1
 -------
 1001110
```

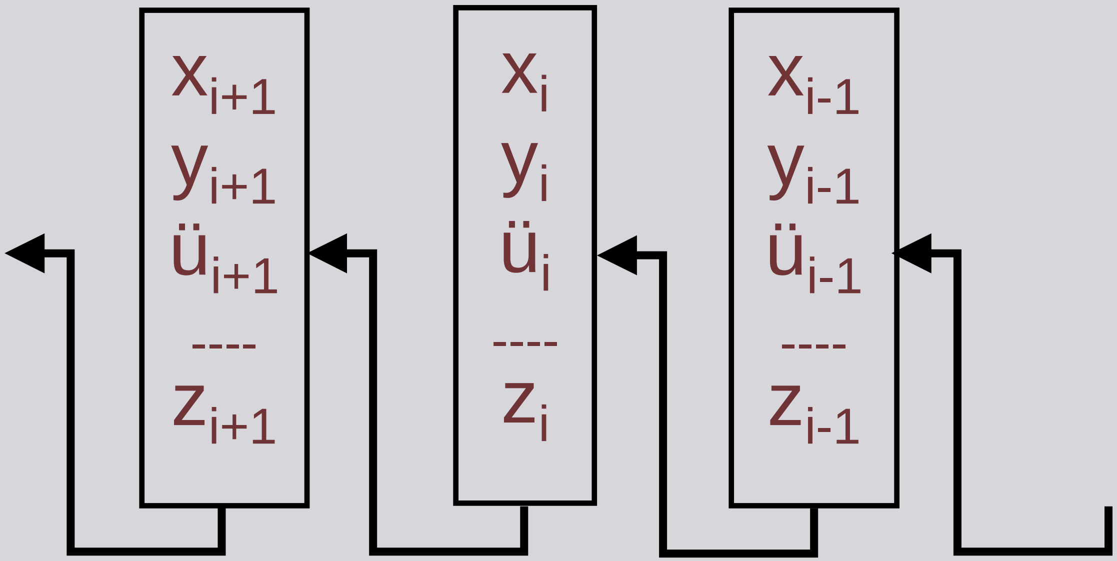
Darstellung von Zahlen

Rechnen mit Binärzahlen (zunächst nur Zahlen ≥ 0)

Addition allgemein:

$$
\begin{array}{l}
\;\; x_n x_{n-1} x_{n-2} \ldots x_1 x_0 \\
+ y_n y_{n-1} y_{n-2} \ldots y_1 y_0 \\
\text{--------------------} \\
\;\; z_n z_{n-1} z_{n-2} \ldots z_1 z_0
\end{array}
$$

Für jede Stelle i gilt



Stellenbilanz: $x_i + y_i + u_i = z_i + 2 * ü_{i+1}$

Darstellung von Zahlen

Rechnen mit Binärzahlen (zunächst nur Zahlen ≥ 0)

Übertragbehandlung bei Addition

x_i	y_i	$ü_i$
0	0	0
0	0	1
0	1	0
1	0	0
0	1	1
1	0	1
1	1	0
1	1	1

z_i	$ü_{i+1}$
0	0
1	0
1	0
1	0
0	1
0	1
0	1
1	1

Stellenbilanz: $x_i+y_i+u_i = z_i+2*ü_{i+1}$

Darstellung von Zahlen

Rechnen mit Binärzahlen (zunächst nur Zahlen ≥ 0)

Subtraktion „wie gewohnt“:

0-0 = 0

1-0 = 1

1-1 = 0

0-1 = 1 (Übertrag)

```
 1001110
-  101001
  1      1
---------
  100101
```

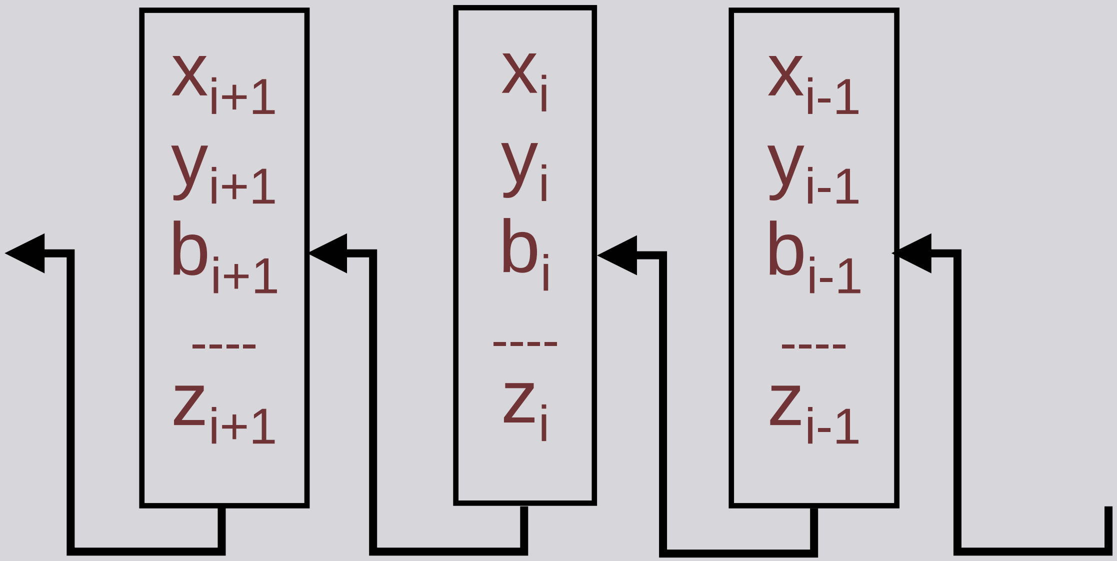
Darstellung von Zahlen

Rechnen mit Binärzahlen (zunächst nur Zahlen ≥ 0)

Subtraktion allgemein:

$$\begin{array}{l} \;\; x_n x_{n-1} x_{n-2} \ldots x_1 x_0 \\ -\; y_n y_{n-1} y_{n-2} \ldots y_1 y_0 \\ \text{--------------------} \\ \;\; z_n z_{n-1} z_{n-2} \ldots z_1 z_0 \end{array}$$

Für jede Stelle i gilt



Stellenbilanz: $x_i - y_i - b_i = z_i - 2 * b_{i+1}$

Darstellung von Zahlen

Rechnen mit Binärzahlen (zunächst nur Zahlen ≥ 0)

Übertragbehandlung bei Subtraktion

x_i	y_i	b_i
0	0	0
0	0	1
0	1	0
1	0	0
0	1	1
1	0	1
1	1	0
1	1	1

z_i	b_{i+1}
0	0
1	1
1	1
1	0
0	1
0	0
0	0
1	1

Stellenbilanz: $x_i - y_i - b_i = z_i - 2*b_{i+1}$

Darstellung von Zahlen

Umwandlung Darstellung mit Basis b_1 in Darstellung mit Basis b_2:

Sei z eine ganze Zahl mit $z \geq 0$ und b1 und b2 ganze Zahlen > 0

Dann lässt sich z darstellen als

$$z = z_{n-1} \cdot b_1^{n-1} + z_{n-2} \cdot b_1^{n-2} + \ldots + z_1 \cdot b_1 + z_0$$

und

$$z = z'_{m-1} \cdot b_2^{m-1} + z'_{m-2} \cdot b_2^{n-2} + \ldots + z'_1 \cdot b_2 + z'_0$$

Darstellung von Zahlen

Umwandlung Dezimal $\Rightarrow$ Hexadezimaldarstellung:

Sei z eine ganze Zahl > 0 in Dezimaldarstellung.

1. Fall: $z \leq 15$: Hexadezimalziffer mit dem Wert von z

2. Fall: $z > 15$:

1. Nimm den Rest der Division z:16 als letzte Ziffer der Hexadezimaldarstellung von z.
2. Ziehe den Rest der Division z:16 von z ab.
   Teile das Ergebnis durch 16 und wandle diese Zahl in Hexadezimaldarstellung um.
   Füge die aus 1. erhaltene Ziffer an das Ergebnis hinten an.

Darstellung von Zahlen

Umwandlung Dezimal $\Rightarrow$ Hexadezimaldarstellung:

Beispiel: Wandle 123_{10} in Hexadezimaldarstellung:

Rest von 123_{10}:16 ist $11_{10} = B_{16} \Rightarrow$ letzte Ziffer ist B_{16}
$(123_{10}-11_{10}) = 112_{10}$
$112_{10}:16_{10} = 7_{10}$
$7_{10} = 7_{16}$; $7<16 \Rightarrow$ erste Ziffer der Hex.-Darstellung ist 7

$\Rightarrow 123_{10} = 7B_{16}$

Darstellung von Zahlen

Umwandlung Hexadezimaldarstellung ⇒ Dezimal:

Beispiel: Wandle $z = AB2_{16}$ in Dezimaldarstellung um:

$$z = \underbrace{\underbrace{10*16^2}_{A} + \underbrace{11*16}_{B} + 2 = 10*256 + 176 + 2}_{\text{Darstellung im Dezimalsystem}} = 2738_{10}$$

Darstellung von Zahlen

Umwandlung Hexadezimaldarstellung ⇒ Dezimal:

Sei z eine ganze Zahl > 0 in Hexadezimaldarstellung.

$\Rightarrow z = z_n*16^n + z_{n-1}*16^{n-1} + ... + z1*16+z_0$

$\Rightarrow z = (...((z_n*16) + z_{n-1})*16) + z_{n-1})*16 ... + z1)*16+z_0$

Ermittle die Dezimaldarstellung von z nach einer der obigen Darstellungen.

Darstellung von Zahlen

Umwandlung Dezimal $\Rightarrow$ Binär:

Sei z eine ganze Zahl > 0 (in Dezimaldarstellung).

1\. Fall: $z \leq 1$: Binärziffer mit dem Wert von z

2\. Fall: $z > 1$:

1. Nimm den Rest der Division z:2 als letzte Ziffer der Binärdarstellung von z.
2. Ziehe den Rest der Division z:2 von z ab.
   Teile das Ergebnis durch 2 und wandle diese Zahl in Binärdarstellung um.
   Füge die aus 1. erhaltene Ziffer an das Ergebnis hinten an.

Darstellung von Zahlen

Umwandlung Dezimal $\Rightarrow$ Binärdarstellung:

Beispiel: Wandle 123_{10} in Binärdarstellung:

Rest von 123_{10}:2 ist 1_{10}	$\Rightarrow$ letzte Ziffer ist 1
$(123_{10}-1_{10}) = 122_{10}$	$122_{10}:2_{10} = 61_{10}$
Rest von $61_{10}:2_{10} = 1_{16}$	$\Rightarrow$ letzte Ziffer ist 1
$(61_{10}-1_{10}) = 60_{10}$	$60_{10}:2_{10} = 30_{10}$
Rest von $30_{10}:2_{10} = 0_{16}$	$\Rightarrow$ letzte Ziffer ist 0
$30_{10}:2_{10} = 15_{10}$	
Rest von $15_{10}:2_{10} = 1_{16}$	$\Rightarrow$ letzte Ziffer ist 1
$(15_{10}-1_{10}) = 14_{10}$	$14_{10}:2_{10} = 7_{10}$
Rest von $7_{10}:2_{10} = 1_{16}$	$\Rightarrow$ letzte Ziffer ist 1
$(7_{10}-1_{10}) = 6_{10}$	$6_{10}:2_{10} = 3_{10}$
Rest von $3_{10}:2_{10} = 1_{16}$	$\Rightarrow$ letzte Ziffer ist 1
$1 \leq 1$	$\Rightarrow$ letzte Ziffer ist 1 (fertig!)

$$\Rightarrow 123_{10} = 1111011_2$$

Darstellung von Zahlen

Umwandlung Binärdarstellung $\Rightarrow$ Dezimal:

Beispiel: Wandle $z = 1111011_2$ in Dezimaldarstellung um:

$$z = \underbrace{1*2^6+1*2^5+1*2^4+1*2^3+0*2^2+1*2^1+1}_{\text{Darstellung im Dezimalsystem}} = 123_{10}$$

Darstellung von Zahlen

Umwandlung Binärdarstellung $\Rightarrow$ Dezimal:

Sei z eine ganze Zahl > 0 in Binärdarstellung.

$\Rightarrow z = z_n*2^n + z_{n-1}*2^{n-1} + ... + z1*2+z_0$

$\Rightarrow z = (...((z_n*2) + z_{n-1})*2) + z_{n-1})*2 ... + z1)*2+z_0$

Ermittle die Dezimaldarstellung von z nach einer der obigen Darstellungen.

Darstellung von Zahlen

Umwandlung Binär- $\Leftrightarrow$ Hexadezimaldarstellung:

jeweils 4 Binärziffern zu einer Hexadezimalziffer zusammenfassen bzw. jede Hexadezimalziffer in 4 Binärziffern umwandeln:

$$1\ 0101\ 0001\ 1111_2$$

$$\updownarrow \quad \updownarrow \quad \updownarrow \quad \updownarrow$$

$$1 \quad 5 \quad 1 \quad F \qquad = 151F_{16}$$

(ggf. in der Binärdarstellung führende Nullen ergänzen.)

Darstellung von Zahlen

Anmerkung

In kaufmännischen Anwendungen wird oft dezimal gerechnet. Deshalb wird hier oft die **BCD-Darstellung** (**binary coded decimals**) verwendet. D. h. jede Dezimalziffer wird mit 4 Binärziffern dargestellt, z.B.

$407_{10} = 0100\ 0000\ 0111_2$

Darstellung von Zahlen

Binärdarstellung negativer Zahlen

- bisher nur Zahlen ≥ 0 betrachtet
- **Problem:** Vorzeichen muss irgendwie dargestellt werden.
- **Lösungsmöglichkeit: Explizite Speicherung des Vorzeichens:** Vorzeichen muss mit einer zusätzlichen Binärziffer dargestellt werden, z. B. „+“ durch 0, „-“ durch 1 (oder umgekehrt),

Darstellung von Zahlen

Binärdarstellung negativer Zahlen

Vorzeichenbit

- 1. Bit steht für das Vorzeichen, die anderen Bits für den Betrag der Zahl

- Beispiel mit 8 Bit:
  $26_{10} = 00011001_2$
  $-26_{10} = 10011001_2$

Darstellung von Zahlen

Binärdarstellung negativer Zahlen

Vorzeichenbit

$\Rightarrow$ Leicht zu verstehen

$\Rightarrow$ Zwei Darstellungen für die Null:
„+0“= 000. . .0 („positive Null“) und
„−0“ = 100. . .0 („negative Null“)

$\Rightarrow$ Mit n Bit darstellbare Zahlen: $-2^{n-1}-1$, . . . , $+2^{n-1}-1$

Darstellung von Zahlen

Binärdarstellung negativer Zahlen

<u>Vorzeichenbit</u>

$\Rightarrow$ Arithmetik mit negativen Zahlen wird aufwändig (Fallunterscheidung „positiv“ / „negativ“)

$\Rightarrow$ nicht gebräuchlich!

Darstellung von Zahlen

Binärdarstellung negativer Zahlen

Alternativen zum „expliziten Vorzeichenbit“:

- Einerkomplement
- Zweierkomplement

Darstellung von Zahlen

Binärdarstellung negativer Zahlen

Einerkomplementdarstellung:

Die Binärdarstellung einer negativen ganzen Zahl wird gebildet durch Invertieren*) aller Bits der entsprechenden positiven Zahl.

Beispiel**): $26_{10} = 00011010_2$

$-26_{10} = 11100101_2$

*) Invertieren eines Bits: $0 \rightarrow 1$, $1 \rightarrow 0$

**) Hier dargestellt als Binärzahl der Länge 8

Darstellung von Zahlen

Binärdarstellung negativer Zahlen

Einerkomplementdarstellung:

- 1. Bit ist wieder Vorzeichenbit
- Zwei Darstellungen für die Null:
  „+0“= 000. . .0 und „−0“ b= 111. . .1
- Mit n Bit darstellbare Zahlen: $-2^{n-1}-1, \ldots, +2^{n-1}-1$
- relativ einfache, aber „ungenaue“ Arithmetik mit negativen Zahlen

Darstellung von Zahlen

Binärdarstellung negativer Zahlen

Einerkomplementdarstellung:

- 26 + (-0) (negative Null) entspricht in Binärdarstellung

```
 00011010₂
+11111111₂
--------------
100011001₂ = 25
```

nur 8 Stellen betrachtet!

Darstellung von Zahlen

Binärdarstellung negativer Zahlen

<u>Zweierkomplementdarstellung</u>:

Die Binärdarstellung einer negativen ganzen Zahl wird gebildet durch Invertieren jedes Bits und anschließende Addition von 1

Beispiel**): $26_{10} = 00011001_2$

$-26_{10} = 11100110_2 + 1 = 11100111_2$

26 als Binärzahl invertiert

1 dazu addiert

-26 in binärer Zweierkomplement-darstellung

Darstellung von Zahlen

Binärdarstellung negativer Zahlen

Zweierkomplementdarstellung:

- 1. Bit ist wieder Vorzeichenbit
- Eine Darstellungen für die Null:
  „0“= 000. . .0
- Mit n Bit darstellbare Zahlen: $-2^{n-1}, \ldots, +2^{n-1}-1$
- einfache, „genaue“ Arithmetik mit negativen Zahlen (Addition / Subtraktion „wie gewohnt“)

Darstellung von Zahlen

Binärdarstellung negativer Zahlen

<u>Zweierkomplementdarstellung</u>:

Definition: Der Wert $zk_n(z)$ einer Zahl z
mit $-2^{n-1} \leq z \leq 2^{n-1}-1$
wird definiert als nicht negative n-stellige Binärzahl

$$zk_n(z) := \begin{cases} z, \text{ falls } z \geq 0, \\ 2^n + z, \text{ falls } z < 0 \end{cases}$$

Darstellung von Zahlen

Binärdarstellung negativer Zahlen

Zweierkomplementdarstellung:

Beispiele:

$zk_4(-3) = 2^4-3 = 16-3 = 13$
$13_{10} = 1101_2 = -3$ (interpretiert im Zweierkomplement)

$zk_5(-30) = 2^6-30 = 64-30 = 34$
$34_{10} = 100010_2 = -30$

Darstellung von Zahlen

Binärdarstellung negativer Zahlen

Zweierkomplementdarstellung – Rechenbeispiele

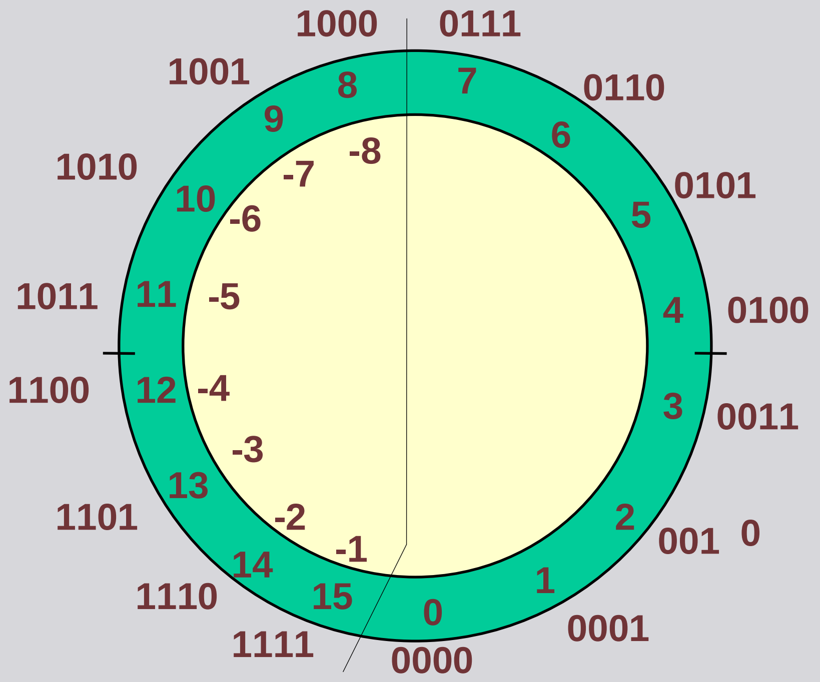
(-5) + (-5)	(-5) + 5	(-5) + 12
entspricht	entspricht	entspricht
11111011	11111011	11111011
+ 11111011	+ 00000101	+ 00001100
1 | 11110110	1 | 00000000	1 | 00000111
entspricht -10	entspricht 0	entspricht 7

Darstellung von Zahlen

Binärdarstellung negativer Zahlen

<u>Zweierkomplementdarstellung Zahlenkreis</u>:
(Bitlänge 4)

negative Zahlen



positive Zahlen

Darstellung von Zahlen

Binärdarstellung negativer Zahlen

Zweierkomplementdarstellung Überlaufproblem bei fester Bitlänge (hier Bitlänge 4):

$7_{10} + 1_{10}$

entspricht in Binärdarstellung

0111 + 0001 = 1000

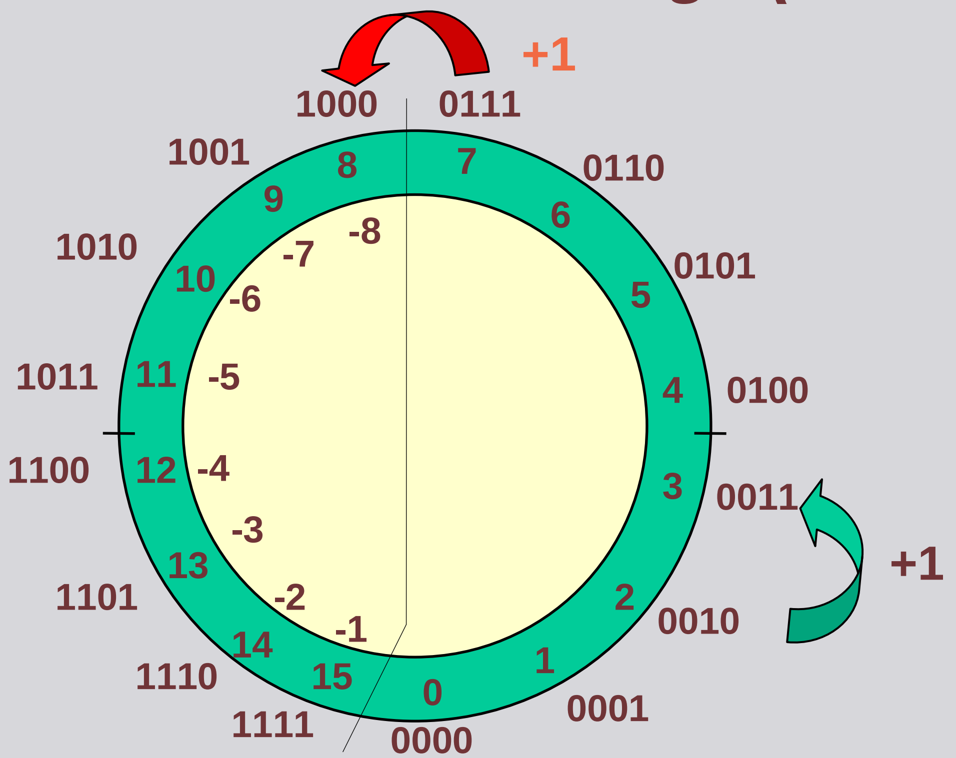
aber $1000_2 = -8_{10}$

Darstellung von Zahlen

Binärdarstellung negativer Zahlen

<u>Zweierkomplementdarstellung Zahlenkreis</u>
<u>Überlaufproblem bei fester Bitlänge (hier Bitlänge 4)</u>:
:

negative Zahlen



positive Zahlen

Darstellung von Zahlen

Binärdarstellung

Wenn eine Berechnung den Bereich darstellbarer ganzer Zahlen „verlässt“, wird dies als ***<u>Überlauf</u>*** bezeichnet.

Bei ganzen Zahlen führt ein Überlauf meist (z. B. im Zweierkomplement) zu einem sog. wrap-around, d. h. bei n-stelliger Darstellung ist das Resultat um 2^n zu groß oder zu klein.

Darstellung von Zahlen

Rechnen mit Binärzahlen

Es gilt folgender Satz:

Tritt bei einer „normalen“ Addition von zwei ganzen Zahlen in n-stelliger Zweierkomplementdarstellung kein Überlauf auf, so ist das Ergebnis im Zweierkomplement dargestellt korrekt. Dabei ist eine evtl. durch Übertrag neu auftretende Eins an Position n zu vernachlässigen.

Folgerung: Mit Zahlen in Zweierkomplementdarstellung kann wie gewohnt addiert oder subtrahiert werden.

Darstellung von Zahlen

Rechnen mit Binärzahlen

Beweis:

Sei $z = x+y$, und, da kein Überlauf gilt $-2^{n-1} \leq x, y, z \leq 2^{n-1}-1$
d.h. x, y, z lassen sich im Zweierkomplement mit n Stellen darstellen.

1. Fall: $x, y \geq 0$, dann wird wie gewohnt gerechnet

2. Fall: $x < 0$, $y \geq 0$, $z \geq 0$, d.h. $zk_n(y) = y$, $zk_n(x) = 2^n+x$, $zk_n(z) = z$
   Addition der Darstellungen: $zk_n(x) + zk_n(y) = 2^n+x+y = 2^n+ z$ (mit $z \geq 0$)
   $\Rightarrow$ es ist ein Übertrag eine führende 1 aufgetreten, die ignoriert wird.

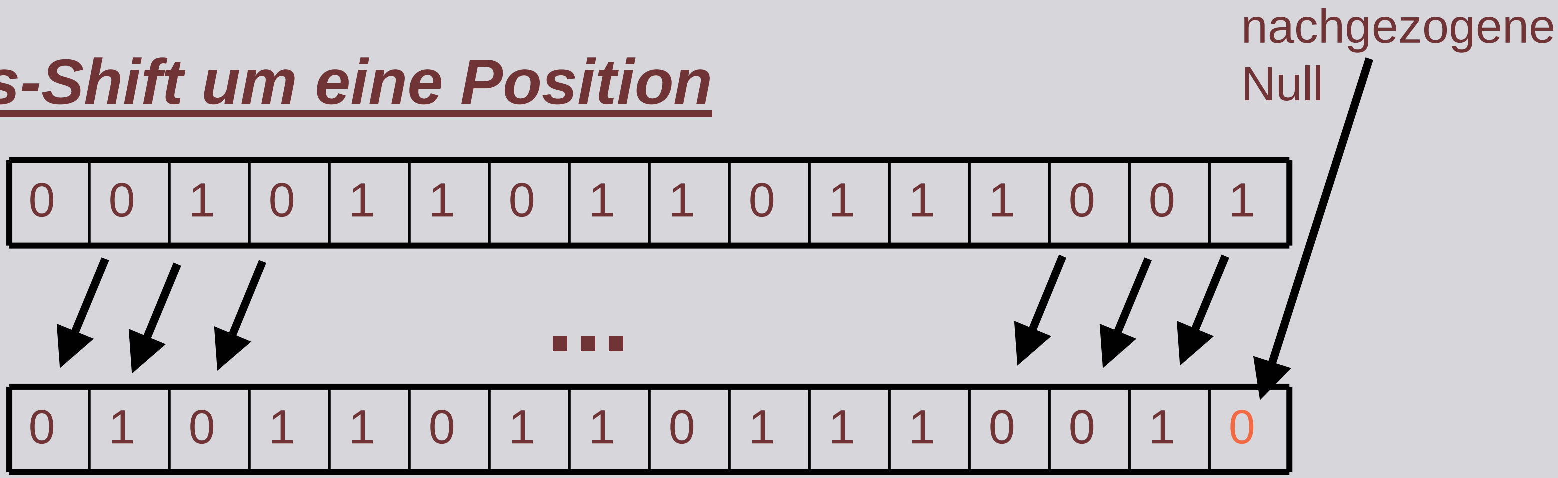
3. Fall: $x < 0$, $y \geq 0$, $z < 0$, d.h. $zk_n(y) = y$, $zk_n(x) = 2^n+x$, $zk(z)_n = 2^n+z$
   Addition der Darstellungen: $zk_n(x) + zk_n(y) = 2^n+x+y = 2^n+ z$ (mit $z < 0$)
   $\Rightarrow$ Darstellung von z im Zweierkomplement.

Darstellung von Zahlen

Ganzzahlarithmetik und Shifts

Definition: Das Verschieben der Bits einer ganzen Zahl in Binärdarstellung um p Positionen nach links (rechts) mit Auffüllen durch Nullen heißt ***bitweiser (Links-/Rechts-) Shift um p Positionen***.

Links-Shift um eine Position



Darstellung von Zahlen

Ganzzahlarithmetik und Shifts

Beispiele:

- Rechtsshift von 00111100 um 2 Positionen ergibt 00001111
- Rechtsshift von 00110001 um 3 Positionen ergibt 00000110
- Linksshift von 00110011 um 8 Positionen ergibt 00000000

Darstellung von Zahlen

Ganzzahlarithmetik und Shifts

Mit Hilfe eines bitweisen Linksshifts um p Positionen einer ganzen Zahl im Zweierkomplement kann eine Multiplikation der Zahl mit 2^p realisiert werden.

$$\underbrace{z_{n-1}z_{n-2}\ \dots\ z_{n-p}}_{p}\underbrace{z_{n-p-1}z_{n-p-2}\ \dots\ z_1z_0}_{n-p} \rightarrow \underbrace{z_{n-p-1}z_{n-p-2}\ \dots\ z_1z_0}_{n-p}\underbrace{000\dots\ 0}_{p}$$

Darstellung von Zahlen

Ganzzahlarithmetik und Shifts

Es gilt folgender Satz:

Tritt kein Überlauf ein, so entspricht die Multiplikation einer im Zweierkomplement dargestellten Zahl z mit einer Zweierpotenz 2^p einem Links-Shift der Ziffern von $zk_n(z)$ um p Positionen.

Überlauf tritt genau dann ein, wenn die Ziffern $z_{n-1}, \ldots, z_{n-p-1}$ (wegfallende Ziffern und erste verbleibende Ziffer) nicht alle gleich sind.

Darstellung von Zahlen

Ganzzahlarithmetik und Shifts

Beweis:
Sei $zk_n(z) = z_{n-1}z_{n-2} \ldots z_{n-p}z_{n-p-1}z_{n-p-2} \ldots z_1z_0$

1. Fall: $z \geq 0$, d.h. $z_{n-1} = 0$
Tritt Überlauf ein, so ist $2^p * z \geq 2^{n-1}$, also $z \geq 2^{n-p-1}$.
Also ist eine der Ziffern $z_{n-1}, \ldots, z_{n-p-1}$ gleich 1, aber $z_{n-1} = 0$.

Tritt kein Überlauf auf, so sind alle diese Ziffern 0 und es gilt
$zk_n(2^p * z) = z_{n-p-1}z_{n-p-2} \ldots z_1z_0 0 \ldots 0$

(entspricht einem Linksshift um p Positionen)

Darstellung von Zahlen

Ganzzahlarithmetik und Shifts

Beweis:

Sei $zk_n(z) = z_{n-1}z_{n-2} \dots z_{n-p}z_{n-p-1}z_{n-p-2} \dots z_1z_0$

2. Fall: $z < 0$, d.h. $zk_n(z) = 2^n+z$ und $z_{n-1} = 1$

Tritt Überlauf ein, so ist $2^p * z < -2^{n-1}$, also $z < -2^{n-p-1}$.

Es gilt dann $zk_n(z) = 2^n+z < 2^n - 2^{n-p-1} = 2^{n-1}+2^{n-2}+2^{n-p-1}$

Also ist mindestens eine der Ziffern $z_{n-1}, \dots, z_{n-p-1}$ gleich 0, aber $z_{n-1} = 0$.

Tritt kein Überlauf auf, so sind alle diese Ziffern 1 und es gilt

$zk_n(2^p * z) = 2^n+2^p * z = 2^p * 2^{n-p}+z = 2^p * zk_{n-p}(z) =$

$z_{n-p}z_{n-p-1}z_{n-p-2} \dots z_1z_00\dots0$

Darstellung von Zahlen

Ganzzahlarithmetik und Shifts

Mit Hilfe eines bitweisen Rechtsshifts um p Positionen einer ganzen Zahl $z \geq 0$ im Zweierkomplement kann eine ganzzahlige Division der Zahl durch 2^p realisiert werden.

$$\underbrace{z_{n-1}z_{n-2} \ldots z_{p+1}z_p}_{n-p}\underbrace{z_{p-1} \ldots z_1z_0}_{p} \rightarrow \underbrace{000\ldots0}_{p}\underbrace{z_{n-1}z_{n-2} \ldots z_{p+1}z_p}_{n-p}$$

Darstellung von Zahlen

Ganzzahlarithmetik und Shifts

Es gilt folgender Satz:

Die Division einer nichtnegativen Binärzahl z durch eine Zweierpotenz 2^p entspricht dem Verschieben der Ziffern um p Positionen nach rechts und Auffüllen mit p Nullen von links. Die letzten p Ziffern von z ergeben den Rest bei der Division.

Darstellung von Zahlen

Zum Schluss dieses Abschnitts …

Noch Fragen ??

Darstellung von Zahlen

Festkommazahlen (Fixed Point)

Zahl mit einer festen Anzahl von Nachkommastellen.

allgemeine Darstellung einer Festkommazahl Fz mit m Nachkommastellen:

$$Fz = z_{n-1} \cdot b^{n-1} + \ldots + z_1 \cdot b + z_0 \cdot b^0 + z_{-1} \cdot b^{-1} + z_{-2} \cdot b^{-2} + \ldots + z_{-m} b^{-m}$$

Beispiele Dezimalzahl (b=10) mit 2 Nachkomnmastellen:
-0.03, 0.40, 126.44

Darstellung von Zahlen

Festkommazahlen (Fixed Point)

Es gilt:

- Addition und Subtraktion mit Festkommazahlen rechnet korrekt, sofern kein Überlauf stattfindet.
  (Begründung wie bei ganzen Zahlen)

- Multiplikation und Division sind ggf. mit Rundungsfehlern

Darstellung von Zahlen

Festkommazahlen (Fixed Point)

Beispiel für Rundungsfehler bei Dezimalzahlen mit 2 festen Nachkommastellen:

- 0.01 * 0.01 = 0.0001 = 0.00

genaue Rechnung ohne Festkomma

Interpretation als Festkommazahl mit 2 Nachkommastellen

Darstellung von Zahlen

Festkommazahlen (Fixed Point)

- Vorteile: Rechnungen schnell (wie ganze Zahlen)

➔häufige Anwendung in zeitkritischen embedded Systemen

- Nachteil: Rundungsfehler müssen beachtet werden

Darstellung von Zahlen

Zum Schluss dieses Abschnitts …

Noch Fragen ??

Darstellung von Zahlen

Gleitkommazahlen (Floating Point)

allgemeine Darstellung einer Gleitkommazahl Gz:

Gz := (Vorzeichen) Mantisse * Basis $^{\text{(Vorzeichen)Exponent}}$

Beispiele:

$+0{,}030 * 10^3 = 30$ (Das Komma wird um 3 Stellen nach rechts geschoben)

$-0{,}4 * 10^{-2} = -0{,}004$ (Das Komma wird um 2 Stellen nach links geschoben)

Darstellung von Zahlen

Gleitkommazahlen (Floating Point)

allgemeine Darstellung einer Gleitkommazahl Gz (Alternative):

$$Gz := \pm (z_0 + z_{-1}*b^{-1} + z_{-2}*b^{-2} + \ldots z_{1-m} + b^{1-m})*b^e$$
$$= z_0 . z_{-1} z_{-2} \ldots z_{1-m} *b^e$$

mit $z_k \in \{1, \ldots , b-1\}$

$z_0 . z_{-1} z_{-2} \ldots z_{1-m}$ ist die Mantisse (oder auch Signifikand)
b ist die Basis
e der Exponent

Darstellung von Zahlen

Gleitkommazahlen (Floating Point)

Definition:

Eine Gleitkommazahl heißt ***normiert***, wenn Mantisse < 1 und erste Nachkommastelle ungleich 0

Beispiele:

$0{,}030 * 10^3$ ist nicht normiert

$-0{,}4 * 10^{-2}$ ist normiert

Darstellung von Zahlen

Gleitkommazahlen (Floating Point)

Es gelten folgende Aussagen:

- für eine Basis b, maximale Längen von Mantisse und Exponent gibt es nur einen **endliche** Menge von Gleitkommazahlen
  $\Rightarrow$ die meisten reellen und rationalen Zahlen müssen auf eine Gleitkommazahl gerundet werden
- die normierte Gleitkommadarstellung einer Zahl ist eindeutig, sofern sie existiert
- bei Rechenoperationen mit Gleitkommazahlen mit fester Längen von Mantisse und Exponenten treten Rundungsfehler auf

Darstellung von Zahlen

Gleitkommazahlen (Floating Point)

Es gelten folgende Aussagen:

- Im allgemeinen lassen sich mittels Gleitkommazahlen (betragsmäßig) wesentlich größere und kleinere Zahlen darstellen als mittels Fixkommadarstellung.
- Größte darstellbare Zahl (normiert): alle Stellen der Mantisse = b-1, größtmöglicher Exponent, z.b. $0,99999*10^{99}$ für 6-stellige Mantisse und 2-stelligen Exponent zur Basis 10
- Kleinste darstellbar Zahl >0 (normiert): $0.10000*10^{-99}$ für 6-stellige Mantisse und 2-stelligen Exponent zur Basis 10
- Es gibt mehrere (viele) Darstellungen der Zahl 0

Darstellung von Zahlen

Gleitkommazahlen (Floating Point)

Rechenoperationen

- Addition / Subtraktion:

  1. Angleichen der Exponenten

  2. Mantissen Addieren / Subtrahieren

  3. Normieren

Darstellung von Zahlen

Gleitkommazahlen (Floating Point)

Rechenoperationen

- Addition / Subtraktion Beispiel (1-stelliger Exponent, ohne Rundungsfehler):

$0{,}125 \cdot 10^3 - 0{,}35 \cdot 10^2 = 0{,}125 \cdot 10^3 - 0{,}035 \cdot 10^3 = 0{,}090 \cdot 10^3$

$= 0{,}9 \cdot 10^2$

Angleichen der Exponenten

Mantissen Addieren / Subtrahieren

Normieren

Darstellung von Zahlen

Gleitkommazahlen (Floating Point)

Rechenoperationen

- Addition / Subtraktion Beispiel (1-stelliger Exponent, 4-stellige Mantisse, mit Rundungsfehler):

$0{,}125 \cdot 10^9 - 0{,}35 \cdot 10^2 = 0{,}125 \cdot 10^9 - 0{,}000 \cdot 10^9 = 0{,}125 \cdot 10^9$

$= 0{,}125 \cdot 10^9$

Angleichen der Exponenten

Mantissen Addieren / Subtrahieren

Normieren

Darstellung von Zahlen

Gleitkommazahlen (Floating Point)

Darstellung im Rechner gemäß IEEE Standard

- IEEE = Institute of Electrical and Electronics Engineers = “Ei Trippl I”
- weltweiter Berufsverband für Elektro-, Elektronik- und Software-Ingenieure
- Konferenzen, Zeitschriften und auch Standardisierungen
- IEEE 754 (IEC-60559:1989)
  - Standarddarstellung für binäre Gleitkommazahlen in Computern
  - IEEE Standard for Binary Floating-Point Arithmetic for microprocessor systems

Darstellung von Zahlen

Gleitkommazahlen (Floating Point)

Darstellung im Rechner gemäß IEEE Standard

- Vorzeichen, Mantisse und Exponent getrennt gespeichert
- verschieden Genauigkeiten (Datenlängen) 32Bit (single), 64Bit (double) und Erweiterungen
- Vorzeichen der Gleitkommazahl in einem Bit gespeichert (0 = positiv, 1 = negativ)
- Mantisse m normiert auf $1 \leq m < 2$, d.h. erste Ziffer ist immer 1 (d.h. beim Speichern der Mantisse kann Summand 1 weggelassen werden) $\Rightarrow$ höhere Genauigkeit

Darstellung von Zahlen

Gleitkommazahlen (Floating Point)

Darstellung im Rechner gemäß IEEE Standard

- Exponenten als vorzeichenlose Zahl als „verschobener" (**biased**) Wert, d.h. Exponent $e_{gespeichert} = e_{real} - e_{min} + 1$ (e_{min} kleinstmöglicher Exponent),

  z.B. bei 8Bit Länge und $e_{min} = -126$ entspricht der gespeicherte Exponent 1 dem realen Exponenten -126. Der gespeicherte Exponent 254 entspricht dem realen Exponenten 127

Darstellung von Zahlen

Gleitkommazahlen (Floating Point)

Darstellung im Rechner gemäß IEEE Standard

- Gesamtdarstellung Feldgröße

	single	double
Länge des Vorzeichens	1 Bit	1 Bit
e_{min}	-126	-1022
e_{max}	127	1023
Länge Exponenten	8	11
Länge Mantisse (Signifikand)	24	53
Gesamtlänge (Länge Mantisse -1 beachten)	32	64

Darstellung von Zahlen

Gleitkommazahlen (Floating Point)

Darstellung im Rechner gemäß IEEE Standard

- Gesamtdarstellung Zahlenbereiche

	single	double
Größte darstellbare Zahl	$(1-2^{-24})*2^{128} \approx 3.403*10^{38}$	$(1-2^{-53})*2^{1024} \approx 1.798*10^{308}$
Kleinste normiert darstellbare Zahl > 0	$2^{-126} \approx 1{,}175*10^{-38}$	$2^{-1022} \approx 2{,}225*10^{-308}$
Kleinste nicht normiert darstellbare Zahl > 0	$2^{-149} \approx 1{,}401*10^{-45}$	$2^{-1074} \approx 4{,}491*10^{-324}$
ε (Abstand von 1 zur nächst größeren Zahl, d.h. $\varepsilon = \min \{z : z>1\}$)	$2^{-23} \approx 1{,}192*10^{-7}$	$2^{-52} \approx 2{,}220*10^{-16}$

Darstellung von Zahlen

Gleitkommazahlen (Floating Point)

Darstellung im Rechner gemäß IEEE Standard

als Bitsequenz (single)*):

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
V	E	E	E	E	E	E	E	E	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
	Exponent								Mantisse																						

*) die genaue Darstellung ist rechnerabhängig (1. Bit links oder rechts)

Darstellung von Zahlen

Gleitkommazahlen (Floating Point)

Darstellung im Rechner gemäß IEEE Standard

Sonderfälle:

- Darstellung der Zahl 0: Mantisse 0, Exponent 0 (keine normalisierte Darstellung möglich!)
- Darstellung der Werte +/- ∞: Mantisse 0, größter Exponent+1
- Darstellung eines Rechenergebnisses, das nicht darstellbar ist: Mantisse ≠ 0, größter Exponent+1

Darstellung von Zahlen

Gleitkommazahlen (Floating Point)

Darstellung im Rechner gemäß IEEE Standard

Anwendung der Sonderfälle:

- ∞ „entsteht“, wenn das Ergebnis einer Rechnung größer wird als die maximal darstellbare Zahl
- 0 mit Vorzeichen „entsteht“, wenn das Ergebnis einer Rechnung kleiner ist als die kleinste darstellbare Zahl
- NaN „entsteht“, wenn das Ergebnis einer Rechnung nicht eindeutig darstellbar ist (z.B. Wurzel aus -2).

Motivation

Zum Schluss dieses Abschnitts …

Noch Fragen ??