Zahldarstellung in Digitalrechnern

Die kleinste Informationseinheit in Digitalrechnern ist **1 Bit** (**b**inary digi**t**) mit den Werten 0 oder 1. Üblicherweise werden 8 Bit zu einem **Byte** zusammengefasst und Information nur in Vielfachen davon gespeichert.

Die Zahldarstellung ist auf verschiedenen Rechnern unterschiedlich. Die folgenden Angaben beziehen sich auf die x86-64-Architektur und richten sich nach der C-Notation:

Integer–Zahlen

Mit INTEGER–Zahlen können ganze Zahlen des erlaubten Zahlenbereichs **exakt** dargestellt werden.

Formate

- Char (int8_t): 1 Byte = 8 Bits
- Short Integer (int16_t): 2 Bytes = 16 Bits
- Integer (int32_t): 4 Bytes = 32 Bits
- Long Integer (int32_t): 8 Bytes = 64 Bits

Darstellung

- positive Zahlen

VZ	2^{n-2}						…	2^2	2^1	2^0
0	I	0	I	I	0	I	…	0	I	I
n–1	*n*–2						…	2	1	0

$0 < x \leq 2^{n-1} - 1$

- Null

VZ	2^{n-2}						…	2^2	2^1	2^0
0	0	0	0	0	0	0	…	0	0	0
n–1	*n*–2						…	2	1	0

$x = 0$

- negative Zahlen
  Darstellung im Allgemeinen im Zweierkomplement, d.h. Invertieren der Darstellung von $(-x)$ $(1 \rightarrow 0, 0 \rightarrow 1$, Einserkomplement) und Addition von 1.

VZ	2^{n-2}						…	2^2	2^1	2^0
I	I	0	I	I	0	I	…	0	I	I
n–1	*n*–2						…	2	1	0

$-2^{n-1} \leq x < 0$

Beispiel: Darstellung von $\pm 1234_{10}$ als Short Integer

$$
\begin{array}{rcccccccccccccccc}
1234_{10} = & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & {}_2 \\
\text{Einserkomplement} & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & {}_2 \\
+ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & {}_2 \\
\text{Zweierkomplement} & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & {}_2
\end{array} = -1234_{10}
$$

Gleitpunktzahlen (ANSI/IEEE 754 Standard)

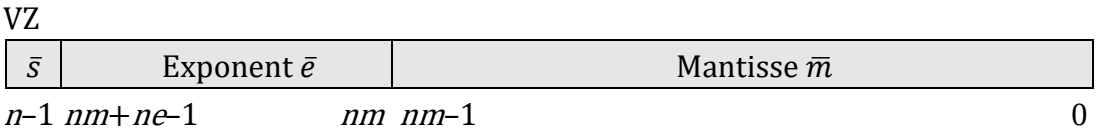
Mit Gleitkommazahlen lassen sich reelle Zahlen im Allgemeinen nur **fehlerbehaftet** darstellen, da sie zu diesem Zweck gerundet werden müssen. Die auf $(nm + 1)$ Stellen gerundete Dualzahl sei wie folgt normiert (Eindeutigkeit der Darstellung):

$$x = \pm \underbrace{1.z_{-1}z_{-2} \dots z_{-nm}}_{m} \cdot 2^e, \quad z_{-i} \in \{0,1\}.$$

Formate

- einfache Genauigkeit (float): 4 Bytes = 32 Bits
- doppelte Genauigkeit (double): 8 Bytes = 64 Bits

Darstellung



Vorzeichen

Das Vorzeichen wird in einem Vorzeichenbit festgehalten. Ist das Vorzeichenbit gesetzt ($\bar{s} = 1$), so ist die Zahl negativ.

Mantisse

Bei der Speicherung der Mantisse wird die Ziffer vor dem Dualpunkt als gesetzt angenommen und nur die Ziffern hinter dem Dualpunkt gespeichert (1 Bit zusätzliche Genauigkeit):

$$\bar{m} = m - 1.$$

Exponent

Der Exponent wird um einen Offset (Bias $= 2^{ne-1} - 1$) verschoben, so dass nur positive Werte auftreten:

$$\bar{e} = e + (2^{ne-1} - 1),$$
$$e = \bar{e} - (2^{ne-1} - 1), \quad -2^{ne-1} + 1 \leq e \leq 2^{ne-1}.$$

Dabei sind die extremen Exponenten allerdings Sonderfällen vorbehalten:

$\bar{e} = 0 \ldots 0_2$:	$\bar{m} = 0$	Darstellung von $x = 0$
	$\bar{m} \neq 0$	Darstellung von nicht–normierten Zahlen
$\bar{e} = 1 \ldots 1_2$:	$\bar{m} = 0$	Darstellung von $x = \pm\infty$
	$\bar{m} \neq 0$	Darstellung einer Nicht–Zahl (NaN) zur Fehlererkennung

Der Wert einer dargestellten Zahl ergibt sich damit im Normalfall zu

$$x = (-1)^{\bar{s}} \cdot 1.\bar{m} \cdot 2^{\bar{e}-(2^{ne-1}-1)}.$$

	float	double
Vorzeichen		
Bitposition	31	63
Exponent		
Anzahl der Bits ne	8	11
Bitpositionen	30 ÷ 23	62 ÷ 52
Bias	127	1023
Mantisse		
Anzahl der Bits nm	23	52
Bitpositionen	22 ÷ 0	51 ÷ 0
Zahlenbereich (normierte Darstellung)		
Betragsmaximum	3.402823 e+38	1.79769 e+308
Betragsminimum	1.175494 e-38	2.22507 e-308
Genauigkeit	5.960464 e-08	1.11022 e-16

Beispiel: Darstellung von $x = -21.3$ in einfacher Genauigkeit (single precision):

$$-21.3_{10} = -1\ 0\ 1\ 0\ 1.0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0 \ldots_2$$

normiert $-1.0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0 \ldots \cdot 2^4$

gerundet $-1.\underbrace{0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0}_{\bar{m}} \cdot 2^4$

$\bar{s} = 1$

$\bar{e} = 4_{10} + 127_{10} = 10000011_2$