Binärdarstellung von Zahlen

Theorie

Inhaltsverzeichnis

1 Etwas Mathematik für die Informatik

1.1 Potenzen, Wurzeln und Logarithmen

Zweierpotenzen

$$\cdots \underset{:2}{\overset{\cdot 2}{\rightleftarrows}} \tfrac{1}{16} \underset{:2}{\overset{\cdot 2}{\rightleftarrows}} \tfrac{1}{8} \underset{:2}{\overset{\cdot 2}{\rightleftarrows}} \tfrac{1}{4} \underset{:2}{\overset{\cdot 2}{\rightleftarrows}} \tfrac{1}{2} \underset{:2}{\overset{\cdot 2}{\rightleftarrows}} 1 \underset{:2}{\overset{\cdot 2}{\rightleftarrows}} 2 \underset{:2}{\overset{\cdot 2}{\rightleftarrows}} 4 \underset{:2}{\overset{\cdot 2}{\rightleftarrows}} 8 \underset{:2}{\overset{\cdot 2}{\rightleftarrows}} 16 \underset{:2}{\overset{\cdot 2}{\rightleftarrows}} \cdots$$

Auswendig lernen: 2^{-10}, 2^{-9}, …, 2^{-1}, 2^0, 2^1, … 2^9, 2^{10}

Rechenoperationen dritter Stufe

- Beim *Potenzieren* wird die Potenz gesucht:

  $2^3 = 8 \quad \text{Basis}^{\text{Exponent}} = \text{Potenz}$

- Beim *Radizieren* wird die Basis gesucht:

  $\sqrt[3]{8} = 2 \quad \sqrt[\text{Wurzelexponent}]{\text{Potenz}} = \text{Basis}$

- Beim *Logarithmieren* wird der Exponent gesucht:

  $\log_2(8) = 3 \quad \log_{\text{Basis}}(\text{Numerus}) = \text{Logarithmus}$

  [sprich: *Der Logarithmus von 8 zur Basis 2 ist 3*]

Beispiele

(a) $2^8 =$

(b) $2^{-3} =$

(c) $\sqrt[3]{125} =$

(d) $\sqrt[10]{1024} =$

(e) $\log_7(49) =$

(f) $\log_2(16) =$

(g) $\log_5(125) =$

(h) $\log_{10}(100) =$

(i) $\log_{10}(10) =$

(j) $\log_{10}(1) =$

(k) $\log_{10}(0.1) =$

(l) $\log_{10}(0.01) =$

(m) $\log_2(\frac{1}{32}) =$

(n) $\log_1(1) =$

1.2 Auf- und Abrundefunktion (*floor* und *ceiling*)

- $\lfloor x \rfloor$ ist die grösste ganze Zahl, die kleiner oder gleich x ist.
- $\lceil x \rceil$ ist die kleinste ganze Zahl, die grösser oder gleich x ist.

Beispiele:

(a) $\lfloor 1.7 \rfloor =$

(b) $\lceil 1.7 \rceil =$

(c) $\lfloor -1.7 \rfloor =$

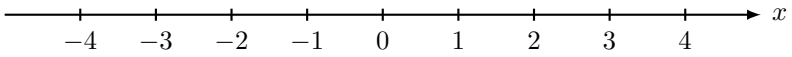
(d) $\lceil -1.7 \rceil =$

(e) $\lfloor 3 \rfloor =$

(f) $\lceil 3 \rceil =$

(g) $\lfloor -3 \rfloor =$

(h) $\lceil -3 \rceil =$



1.3 Der Divisionsrest

Sind a und b natürliche Zahlen und $b \neq 0$, so definieren wir den *Divisionsrest* $a \bmod b$ [sprich: „a modulo b“] wie folgt:

$$a \bmod b = a - \left\lfloor \frac{a}{b} \right\rfloor \cdot b$$

Beispiele

(a) $7 \bmod 3 =$

(b) $14 \bmod 5 =$

(c) $9 \bmod 13 =$

(d) $18 \bmod 6 =$

(d) $0 \bmod 4 =$

1.4 Formale Sprachen

Alphabete

Ein *Alphabet* Σ ist eine endliche, nicht leere Menge von Symbolen. *Beispiele:*

- $\Sigma = \{\mathrm{a}, \mathrm{b}, \ldots, \mathrm{z}\}$ (lateinische Kleinbuchstaben)
- $\Sigma = \{0, 1, 2, \ldots, 9\}$ (Menge der Ziffern im Dezimalsystem)
- $\Sigma = \{0, 1\}$ (binäres Alphabet)

Wörter

Ist Σ ein Alphabet und k eine natürliche Zahl, dann ist ein *Wort* w der Länge k eine Folge von k Symbolen aus Σ. *Beispiele:*

- 011010 (Wort der Länge 6 aus dem binären Alphabet)
- abcdda (Wort der Länge 6 aus dem Kleinbuchstaben-Alphabet)

Das spezielle Wort, das aus keinem Symbol besteht (also die Länge $k = 0$ hat), wird *das leere Wort* genannt und mit ε bezeichnet.

Formale Sprachen

Eine *formale Sprache* L über einem Alphabet Σ ist eine Teilmenge aller Wörter, die aus den Zeichen von Σ gebildet werden können.

Beispiele formaler Sprachen

(a) Die Menge aller Wörter der Länge 4 aus $\Sigma = \{0, 1\}$ mit einer geraden Anzahl Einsen.

(b) Die Menge aller zweistelligen Zahlen mit Ziffern aus $\Sigma = \{0, 1, 2, \ldots, 9\}$.

(c) Die Menge aller Wörter aus den Zeichen von $\Sigma = \{\mathrm{I}, \mathrm{V}, \mathrm{X}, \mathrm{L}, \mathrm{C}, \mathrm{D}, \mathrm{M}\}$, die eine römische Zahl darstellen.

(c) Die Menge der Morsezeichen mit $\Sigma = \{\cdot, -\}$.

Anzahl der Wörter der Länge k

Wie viele Wörter der Länge k lassen sich aus einem Alphabet Σ mit n Zeichen bilden?

- Für das 1-te Element gibt es n Auswahlmöglichkeiten
- Für das 2-te Element gibt es n Auswahlmöglichkeiten
- ...
- Für das k-te Element gibt es n Auswahlmöglichkeiten

insgesamt:

$\underbrace{n \cdot n \cdot n \cdot \ldots \cdot n}_{k \text{ Faktoren}} = n^k$ Möglichkeiten

Beispiele

Wie viele Wörter der Länge k mit Zeichen aus Σ gibt es?

(a) $\Sigma = \{0, 1, 2, \ldots, 9\}$, $k = 5$

(b) $\Sigma = \{A, B, C, \ldots, Z\}$, $k = 2$

(c) $\Sigma = \{0, 1\}$, $k = 8$

2 Bits und Bytes

Codes

Ein *Code* ist eine Abbildung, die jedem Wort einer Sprache L_1 eindeutig ein Wort einer Sprache L_2 zuordnet. Dabei darf auch $L_1 = L_2$ gelten (siehe Beispiel 2.2).

Beispiel 2.1

Der Morsecode ordnet den 26 lateinischen Buchstaben, den 10 Ziffern und einigen Sonderzeichen jeweils eine eindeutige Folge von Punkten und Strichen zu.

A	·−
B	−···
⋮	⋮
Z	−−··

0	−−−−−
1	·−−−−
⋮	⋮
9	−−−−·

.	·−·−·−
,	−−··−−
⋮	⋮
@	·−−·−·

Beispiel 2.2

Bei einer monoalphabetischen Substitutionschiffre werden Wörter der Länge 1 (Zeichen) durch andere Wörter der Länge 1 ersetzt.

```
Klartextalphabet:    ABCDEFGHIJKLMNOPQRSTUVWXYZ
Geheimtextalphabet:  YPATCEDJLBFZKOSUNGHXIQMVRW
```

```
DASISTEINEGEHEIMEBOTSCHAFT
```

```
TYHLHXCLOCDCJCLKCPSXHAJYEX
```

Ist dieser Code sicher?

Beispiel 2.3

Polizeicodes (USA): `https://en.wikipedia.org/wiki/Police_code`

Das Bit

Ein *Bit* (von engl. binary digit) ist die Informationsmenge, die durch eine Binärziffer dargestellt werden kann. Beispiele:

- die Position eines Ein-Aus-Schalters
- die Antwort auf eine Ja-Nein-Frage
- die Information, ob eine Aussage wahr oder falsch ist

Informationsmenge

Anzahl Bit	Zustände	Anzahl Zustände
1		
2		
3		
...		
n		

Falls a keine Zweierpotenz ist muss das Resultat aufgerundet werden, da keine halben Bits möglich sind. Dann gilt:

$$n = \lceil \log_2 a \rceil$$

Beispiel 2.4

Die 6 Augenzahlen eines Spielwürfels sollen binär codiert werden. Wie viele Bits sind dafür mindestens nötig?

Augenzahl	Binärcode
⚀	
⚁	
⚂	
⚃	
⚄	
⚅	

Bezeichnungen

8 Bits = 1 Byte

SI-Präfixe

10^3 Byte =

10^6 Byte =

10^9 Byte =

10^{12} Byte =

10^{15} Byte =

10^{18} Byte =

10^{21} Byte =

10^{24} Byte =

Das SI (Système international d'unités) ist das am weitesten verbreitete Einheitensystem für physikalische Grössen.

IEC-Präfixe

1024 Byte =

1024 KiB =

1024 MiB =

1024 GiB =

1024 TiB =

1024 PiB =

1024 EiB =

1024 ZiB =

Die IEC (International Electrotechnical Commission) ist eine internationale Normungsorganisation für Normen im Bereich der Elektrotechnik und Elektronik mit Sitz in Genf.

Faustformel für die Umrechung

$2^{10} = 1024 \approx 1000 = 10^3$

Aus der Gebrauchsanweisung einer USB-Harddisk

Ein Gigabyte (GB) bedeutet $10^9 = 1\,000\,000\,000$ Byte und ein Terabyte (TB) bedeutet $10^{12} = 1\,000\,000\,000\,000$ Byte unter Verwendung von Zehnerpotenzen. Das Computerbetriebssystem zeigt die Speicherkapazität jedoch in der Form von "2 hoch" als

1 GiB $= 2^{30} = 1\,073\,741\,824$ Byte und

1 TiB $= 2^{40} = 1\,099\,511\,627\,776$ Byte

an, was zu einem geringeren Wert führt. Die verfügbare Speicherkapazität variiert je nach Dateigrösse, Formatierung, Einstellungen, Software und Betriebssystem sowie anderen Faktoren.

3 Zahlensysteme

3.1 Das Dezimalsystem

Das Dezimalsystem ist ein Stellenwertsystem (Positionssystem) zur Basis 10. Das bedeutet, dass eine Ziffer neben ihrem eigenen Wert noch einen Wert erhält, der durch ihre Position innerhalb der Zahl gegeben ist.

Beispiel 3.1

Wie bildet man eine Zahl aus ihren Ziffern?

Dezimalziffer	6	9	3
Stellenwert			

Beispiel 3.2

Wie gewinnt man die Ziffern aus einer Zahl?

693 : 10 = Rest
: 10 = Rest
: 10 = Rest

3.2 Das Binärsystem

Das Binärsystem ist ein Stellenwertsystem zur Basis 2.

Bekanntlich werden Zahlen im Dezimalsystem aus den zehn Ziffern 0 bis 9 gebildet. Entsprechend werden Zahlen im Binärsystem aus den zwei Ziffern 0 und 1 gebildet.

Wenn nicht aus dem Kontext hervorgeht, welche Basis einer Zahl zu grunde liegt, kennzeichnet man sie mit einem entsprechenden Index.

Beispiele: 101_{10}, 101_2 oder 235_6.

3.2.1 Umrechnung vom Binär- ins Dezimalsystem

Beispiel 3.3

Analog zum Beispiel 3.1 erhalten wir:

1011_2

Beispiel 3.4

Rechne die Binärzahl 10000101_2 ins Dezimalsystem um.

$10000101_2 =$

3.2.2 Umrechnung vom Dezimal- ins Binärsystem

Beispiel 3.5

Die Umrechnung erfolgt analog zum Beispiel 3.2.

293	:	2	=	Rest
	:	2	=	Rest
	:	2	=	Rest
	:	2	=	Rest
	:	2	=	Rest
	:	2	=	Rest
	:	2	=	Rest
	:	2	=	Rest
	:	2	=	Rest
	:	2	=	Rest

Beispiel 3.6

Welche Darstellung hat die Dezimalzahl 47 im Binärsystem?

:	2	=	Rest
:	2	=	Rest
:	2	=	Rest
:	2	=	Rest
:	2	=	Rest
:	2	=	Rest

Beispiel 3.7

Welche Darstellung hat die Dezimalzahl 148 im Binärsystem?

	: 2	=	Rest
	: 2	=	Rest
	: 2	=	Rest
	: 2	=	Rest
	: 2	=	Rest
	: 2	=	Rest
	: 2	=	Rest
	: 2	=	Rest

3.3 Das Hexadezimalsystem

Für das (Sechzehnersystem) benötigen wir sechzehn Ziffern. Da wir im Dezimalsystem aber nur zehn Ziffern zur Verfügung haben, verwenden wir für die fehlenden Ziffern die ersten sechs Buchstaben unseres Alphabets.

10er-System	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16er-System	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Gross- und Kleinschreibung wird nicht unterschieden.

In der Informatik werden Hexadezimalzahlen zur Kennzeichnung das Präfix `0x` oder dass Suffix h beigefügt. ($\texttt{1A53}_{16} = \texttt{0x1A53} = \texttt{1A53h}$)

3.3.1 Umrechnung vom Hexadezimal- ins Dezimalsystem

Beispiel 3.8

Hexadezimalzahl	1	5	E
Stellenwert			

3.3.2 Umrechnung vom Dezimal- ins Hexadezimalsystem

Beispiel 3.9

1610	:	16	=	Rest
	:	16	=	Rest
	:	16	=	Rest

3.3.3 Umrechnungen vom Hexadezimal- ins Binärsystem

Da die Zahl 16 eine Potenz von 2 ist, sind die Umrechnungen zwischen dem Hexadezimal- und dem Binärsystem besonders einfach. Da jede Hexadezimalzahl durch genau vier Binärziffern dargestellt werden kann, teilen wir jede Binärzahl von rechts nach links in Vierergruppen ein. Fehlende Stellen links werden durch Nullen aufgefüllt. Dann wandelt man jede Vierergruppe in die entsprechende Hexadezimalzahl um.

Beispiel 3.10

Wandle die Binärzahl 1111001001_2 in eine Hexadezimalzahl um:

Beispiel 3.11

Stelle die Hexadezimalzahl F4E7 als Binärzahl dar:

3.4 Das Oktalsystem

Das Oktalsystem ist das Zahlensystem zur Basis 8, das aus den Ziffern 0, 1, ..., 6, 7 besteht.

Oktalzahlen werden in der Informatik manchmal durch eine vorangestellte Null oder ein nachgestelltes kleines „o“ gekennzeichnet, was jedoch leicht zu Verwechslungen führen kann.

Beispiel: $371_8 = 0371 = 371o$

3.4.1 Umrechnung vom Oktal- ins Dezimalsystem

Beispiel 3.12

Oktalziffer	1	0	2	7
Stellenwert				

3.4.2 Umrechnung vom Dezimal- ins Oktalsystem

Beispiel 3.13

843	:	8	=	Rest
	:	8	=	Rest
	:	8	=	Rest
	:	8	=	Rest

Beispiel 3.14

Rechne 473_8 ins Dezimalsystem um:

Beispiel 3.15

Rechne 1217_{10} ins Oktalsystem um:

:	8	=	Rest
:	8	=	Rest
:	8	=	Rest
:	8	=	Rest

3.4.3 Umrechnungen zwischen Oktal- und Binärsystem

Da die Basis 8 des Oktalsystems eine Potenz der Basis 2 des Binärsystems ist, sind die beiden Systeme in einer gewissen Weise miteinander verwandt. Das erleichtert das Umrechnen zwischen diesen Systemen.

Wollen wir eine Binärzahl *direkt* ins Oktalsystem umwandeln, teilen wir ihre Ziffern von rechts nach links in Dreiergruppen ein. Fehlende Stellen links füllen wir mit Nullen auf. Da $001_2 = 1_8$, $010_2 = 2_8$, ..., $111_2 = 7_8$, kann man jede Dreiergruppe aus Binärzahlen in die entsprechende Oktalzahl umwandeln.

Beispiel 3.16

Wandle die Binärzahl 010110011 ohne Umrechnung ins Dezimalsystem in eine Oktalzahl um.

0	1	0	1	1	0	0	1	1

Beispiel 3.17

Die Umrechnung in die andere Richtung ist ebenso einfach. Rechne als Beispiel die Oktalzahl 3756_8 in eine Binärzahl um.

3			7			5			6		

Beispiel 3.18

Stelle 11110101010_2 im Oktalsystem dar:

Beispiel 3.19

Stelle 1037_8 im Binärsystem dar:

4 Ganze Zahlen in Binärdarstellung

4.1 Bitwertigkeit

Um Missverständnisse bei der Darstellung binärer Zahlen zu vermeiden, vereinbaren wir folgende Begriffe:

- Das Bit, das die kleinste Zweierpotenz repräsentiert, wird *least significant bit* (LSB) genannt.
- Das Bit, das die grösste Zweierpotenz repräsentiert, wird *most significant bit* (MSB) genannt.

Da in unserer Zahlendarstellung die Stellenwerte von rechts nach links aufsteigen, vereinbaren wir, dass das LSB jeweils ganz rechts steht. Dies wird schematisch so dargestellt:

7							0
0	1	0	0	1	0	1	1

4.2 Addition

Allgemein gilt:

$0 + 0 =$
$0 + 1 =$
$1 + 0 =$
$1 + 1 =$

Der Ablauf ist wie im Dezimalsystem: von rechts nach links, mit Übertrag. Führende Leerstellen werden wie Nullen behandelt.

Beispiel 4.1

	=	7
+	=	13
=	=	

4.3 Negative ganze Zahlen

Ansatz: Stelle der Binärdarstellung ein zusätzliches Bit voran, wobei beispielsweise `1` eine negative und `0` eine positive Zahl bedeutet.

$0011_2 = 3_{10}$
$1011_2 = -3_{10}$

Wie wir aus der 7. Klasse wissen, muss man beim Rechnen mit Vorzeichen „mühsame“ Fallunterscheidungen beachten.

Deshalb verwendet man eine andere Darstellung für negative Zahlen, mit der Computer einfacher und schneller rechnen können.

Diese Darstellungsweise soll nun vorgestellt werden.

Stelle die ganzen Zahlen von 0 bis 7 im Binärsystem mit führenden Nullen dar. Setze anschliessend die Tabelle in „natürlicher“ Weise in den Bereich der negativen Zahlen fort.

7				
6				
5				
4				
3				
2				
1				
0				
-1				
-2				
-3				
-4				
-5				
-6				
-7				
-8				

Das Einerkomplement

Das *Einerkomplement* $\overline{A}$ einer Binärzahl A erhält man durch „Flippen“ jedes Bits von A.

Eigenschaften

Bestimme die Dezimalzahl $\overline{A}$, die zum Einerkomplement der Zahl A gehört.

Zahl A	Binärdarstellung von A	Einerkomplement von A	$\overline{A}$
5			
2			
-7			
-1			

- $A + \overline{A} = 1111_2 = 10000_2 - 1_2 = 2^4 - 1$
- $\overline{A} + 1 = -A \quad$ für $-7 \leq A \leq 7$

Beispiel 4.2

$$\begin{array}{rcccccccc} A & = & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ \overline{A} & = & & & & & & & & \\ \hline A+\overline{A} & = & & & & & & & & \end{array}$$

Allgemein gilt für eine n-stellige Binärzahl A und ihr Einerkomplement $\overline{A}$:

Das Zweierkomplement

Das *Zweierkomplement* einer Binärzahl A ist der um 1 vergrösserte Wert des Einerkomplements $\overline{A}$.

Beispiel 4.3

$$\begin{array}{rcccccccc} A & = & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ \overline{A} & = & & & & & & & & \\ \overline{A}+1 & = & & & & & & & & \end{array}$$

(Einerkomplement)

(Zweierkomplement)

Allgemein gilt für eine n-stelligen Binärzahl A und ihr Zweierkomplement $\overline{A}+1$:

Zusammenfassung

Beispiel 4.4

Bestimme die Gegenzahl von 3 wenn die Zahlen mit vier Bit dargestellt werden.

$$\begin{array}{rl} A & = \\ \overline{A} & = \\ \overline{A}+1 & = \end{array}$$

Beispiel 4.5

Bestimme die Gegenzahl von -5 wenn die Zahlen mit vier Bit dargestellt werden.

$$\begin{array}{rl} A & = \\ \overline{A} & = \\ \overline{A}+1 & = \end{array}$$

Beispiel 4.6

Bestimme die Gegenzahl von 0 wenn die Zahlen mit vier Bit dargestellt werden.

A	$=$
$\overline{A}$	$=$
$\overline{A}+1$	$=$

4.4 Subtraktion

Da wir jetzt eine Methode kennen, mit der man aus der Binärdarstellung der Zahl k ihre (binäre) Gegenzahl $-k$ bestimmen kann, können wir die Subtraktion einer Zahl k als Addition ihrer Gegenzahl $-k$ darstellen. Formal:

$$m - k = m + (-k)$$

Subtraktion mit positivem Ergebnis

Berechne $6 - 4 = 6 + (-4)$:

	$=$
$+$	$=$
$=$	$=$

So lange das Resultat innerhalb des Darstellungsbereichs (hier von -8 bis $+7$) liegt, darf eine allfällige Übertrags-Eins ganz links ignoriert werden.

Subtraktion mit negativem Ergebnis

Berechne $4 - 7 = 4 + (-7)$:

	$=$
$+$	$=$
$=$	$=$

Addition von zwei negativen Zahlen

Berechne $-2 + (-3)$:

	$=$
$+$	$=$
$=$	$=$

4.5 Multiplikation

Allgemein gilt:
$$\begin{aligned} 0 \cdot 0 &= 0 \\ 0 \cdot 1 &= 0 \\ 1 \cdot 0 &= 0 \\ 1 \cdot 1 &= 1 \end{aligned}$$

Beispiel 4.7

Die Multiplikation zweier Binärzahlen besteht aus einer fortgesetzten Addition unter Stellenverschiebung; hier gezeigt an der Rechnung $13 \cdot 10$.

1 1 0 1 · 1 0 1 0

Beispiel 4.8

1 0 · 1 0 1 1

4.6 Division

Die Division binärer Zahlen wird auf eine fortgesetzte Subtraktion unter Stellenverschiebung zurückgeführt.

Beispiel 4.9

Hier wird 65 : 13 im Binärsystem gerechnet.

1 0 0 0 0 0 1 : 1 1 0 1 =

Beispiel 4.10

Binäre Division von 22 : 2:

1 0 1 1 0 : 1 0 =

Beispiel 4.11

Binäre Division von 17 : 2:

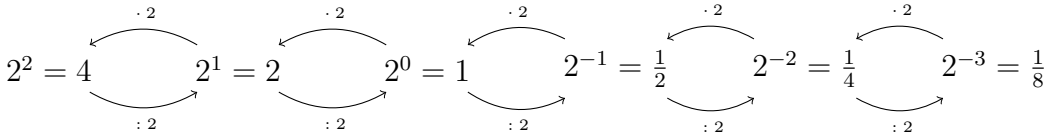
1 0 0 0 1 : 1 0 =

---

5 Binärdarstellung von Dezimalzahlen

5.1 Binärdarstellung von Zahlen zwischen 0 und 1

Potenzdarstellung mit negativen Exponenten



Beispiel 5.1

Bestimme die Binärdarstellung von 0.8125 durch Probieren:
Die Stellenwerte sind hier: $0.5 = 2^{-1}$, $0.25 = 2^{-2}$, $0.125 = 2^{-3}$ ...

n	2^{-n}	Bit	kumuliert

algorithmisch: („C“ steht für *carry*; d. h. Übertrag)

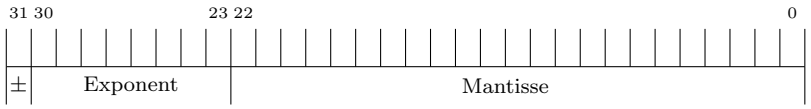
2	·	=	C
2	·	=	C
2	·	=	C
2	·	=	C
2	·	=	C
2	·	=	C

Beispiel 5.2

Bestimme die Binärdarstellung von 0.1:

2	·	=	C
2	·	=	C
2	·	=	C
2	·	=	C
2	·	=	C
2	·	=	C
2	·	=	C
2	·	=	C

5.2 Gleitkommazahlen im IEEE 754-Standard



Vorzeichen (Bit 31)

S (*sign*) ist das Vorzeichenbit.

$S = 0$ markiert eine positive Zahl; $S = 1$ eine negative Zahl.

Für die Null erlaubt der Standard sowohl ein positives als auch ein negatives Vorzeichen.

Exponent (Bits 23–30)

Mit 8 Bits lassen sich $2^8 = 256$ Exponenten darstellen. Jedoch sind 0 und 255 für spezielle Zahlen reserviert und können nicht als Exponenten verwendet werden.

Negative Exponenten werden durch Addition von $B = 127$ (*bias*) positiv gemacht und dann binär dargestellt.

Mantisse (Bits 0–22)

Die Mantisse ist die Ziffernfolge einer Zahl. Die Gleitkommazahlen 0.002357 und 235.7 haben beispielsweise dieselbe Mantisse 2357.

Die Binärzahl wird so lange mit einer Zweierpotenz multipliziert oder dividiert, bis die führende 1 vor dem Dezimalpunkt steht. Diesen Vorgang nennt man *Normalisieren*.

Binärzahl	Normalform	Mantisse
1101.01	$1.10101 \cdot 2^3$	(1)10101
0.00101	$1.01 \cdot 2^{-3}$	(1)01

Indem man die durch das Normalisieren zwangsläufig entstehende führende 1 weglässt, kann man ein Bit Speicher sparen.

5.3 Umrechnung Dezimalzahl ins IEEE 754-Format

Beispiel 5.3

Welche IEEE 754-Darstellung hat die Zahl 5.75?

Vorzeichenbit

Wegen $5.75 > 0$ ist $S =$

Mantisse

Den ganzzahligen und den gebrochenen Anteil binär darstellen:

ganzzahliger Anteil:	:	2	=	R
	:	2	=	R
	:	2	=	R

gebrochener Anteil:	2	·	=	C
	2	·	=	C

Normalisierung:

Exponent

Addiere den Bias ($B = 127$) zum Exponenten und stelle ihn anschliessend binär dar.

Resultat

5.4 Umrechnung einer IEEE 754-Zahl in eine Dezimalzahl

Beispiel 5.4

Welcher Gleitkommazahl entspricht die Binärzahl
1|10000100|01010000000000000000000
im IEEE 754-Format?

Vorzeichen

Exponent

Mantisse

Resultat

5.5 Spezielle Zahlen

Die betragsmässig grösste normalisierte Zahl

Der maximale Exponent beträgt $E = 254 - 127 = 127$, da $11111111_2 = 255$ für andere Zwecke reserviert ist.

Die grösste Mantisse beträgt $M = (1)11111111111111111111111$ wenn wir wieder die Eins an die höchstwertige Stelle setzen.

Damit erhalten wir

$$\begin{aligned} & 1.11111111111111111111111 \cdot 2^{127} \\ &= 11111111111111111111111 \cdot 2^{104} \\ &\approx 3.403 \cdot 10^{38} \end{aligned}$$

als betragsmässig grösste normalisierte Zahl

Die betragsmässig kleinste normalisierte Zahl

Der minimale Exponent beträgt $1 - 127 = -126$, da $00000000_2 = 0$ für andere Zwecke reserviert ist.

Die kleinste Mantisse beträgt $M = (1)00000000000000000000000$ wenn wir wieder die Eins an die höchstwertige Stelle setzen.

Damit erhalten wir

$$1 \cdot 2^{-126} \approx 1.175 \cdot 10^{-38}$$

als betragsmässig kleinste normalisierte Zahl

Die Null

Auf der einen Seite gewinnen wir durch die Normalisierung immer eine Binärstelle mehr an Genauigkeit. Andererseits zwingt uns dies mit $m = 1.M$ immer mindestens eine 1 in der Mantisse auf, so dass die Darstellung der 0 auf diese Weise unmöglich wird.

Um die Normalisierung zu „verhindern“ wird der Exponent mit dem Wert $E = 0$ codiert und die Mantisse wird in der Form $m = 0.M$ interpretiert. Dies führt dazu, dass die Null auch als Gleitkommazahl aus lauter Nullen besteht – naja nur fast, denn es gibt auch noch die „negative“ Null, welche der Standard nicht verbietet. Somit hat die Null die folgende(n) Darstellung(en):

$$\begin{aligned} +0 &= 0|00000000|00000000000000000000000 \\ -0 &= 1|00000000|00000000000000000000000 \end{aligned}$$

Subnormale (denormalisierte) Zahlen

Mit $E = 00000000$ und $M = 00000000000000000000000$ wird die Null codiert. Was sollen wir nun aber mit den Mantissen

$$\begin{aligned} & 10000000000000000000000 \\ & \ldots \\ & 11111111111111111111111 \end{aligned}$$

machen, wenn der Exponent $E = 00000000$ ist? Diese Werte lassen sich dazu verwenden, um Zahlen darzustellen, die zwischen Null und der kleinsten normalisierte Zahl sind. Deshalb der Ausdruck *sub*normale (oder: denormalisierte) Zahlen.

Unendlich

Nachdem wir mit dem Exponenten $E = 0$ die Zahl Null und die subnormalen Zahlen gewonnen haben, klären wir noch, was es mit dem maximalen Exponenten $E = 255$ auf sich hat.

Die kleinste mit diesem Exponenten darstellbare Mantisse $M = 0$ wird für Unendlich (`Infinity`) verwendet. Wieder gibt es zwei Formen:

`0|11111111|00000000000000000000000` = `+Infinity`

`1|11111111|00000000000000000000000` = `-Infinity`

Der Wert `+Inf` bzw. `-Inf` repräsentiert Zahlen, deren Betrag zu gross ist, um dargestellt zu werden.

Zahlen, die gar keine sind

Auch hier können wir uns fragen, was wir mit den übrigen Mantissen M zum Exponenten $E = 255$ anfangen sollen. Die Informatiker haben hier eine besondere Lösung gefunden. Mit den Mantissen $M \neq 0$ werden Ereignisse angezeigt, die es bei korrektem Rechnen nicht geben darf.

- Division durch Null
- Quadratwurzel aus einer negativen Zahl
- Logarithmus einer negativen Zahl
- Arcussinus oder Arcuscosinus einer Zahl x mit $|x| > 1$

Diese mit $E = 255$ und $M \neq 0$ codierten Objekte werden als `NaN` (*Not a Number*) bezeichnet. Der IEEE 754-Standard ignoriert dabei das Vorzeichenbit.