

4 Zahlensysteme und Rechnerarithmetik

Stellenwertsysteme, Ganzzahlen und Gleitkommazahlen

Fabian Schwarzenberger

Inhaltsverzeichnis

4.1 Stellenwertsysteme	2
(Un-)Eindeutigkeit der Zahlendarstellung	2
Endliche und unendliche Nachkommastellen	2
4.2 Umrechnen zwischen Zahlensystemen	4
4.3 Binärdarstellung natürlicher Zahlen	7
4.4 Zweierkomplement	9
4.5 Gleitkommadarstellung	13

Eine Zahl – mehrere Schreibweisen

Beispiel. Die Zahl siebenundvierzig kann unter anderem so geschrieben werden:

- $(47)_{10}$ im Dezimalsystem,
- $(101111)_2$ im Binärsystem,
- $(57)_8$ im Oktalsystem,
- $(2F)_{16}$ im Hexadezimalsystem.

Die Schreibweisen sind verschieden, der dargestellte Zahlenwert ist derselbe. Erst bei einer festen Anzahl von Speicherstellen kommen neue Fragen hinzu: Welche Werte passen in das Format, und was geschieht beim Runden oder bei einem Überlauf?

4.1 Stellenwertsysteme

Definition 4.1 (Stellenwertdarstellung). Seien $b \in \mathbb{N}$ mit $b \geq 2$ und $x_j \in \{0, 1, \dots, b - 1\}$. Dann bezeichnet

$$(x_p \dots x_1 x_0, x_{-1} \dots x_{-q})_b = \sum_{j=-q}^p x_j b^j$$

eine **Stellenwertdarstellung zur Basis b** . Die x_j heißen Ziffern; die Stelle j besitzt das Gewicht b^j .

Wichtige Stellenwertsysteme

Beispiel.

- $b = 2$: Binär- oder Dualsystem mit den Ziffern 0, 1,
- $b = 8$: Oktalsystem mit den Ziffern 0, ..., 7,
- $b = 10$: Dezimalsystem mit den Ziffern 0, ..., 9,
- $b = 16$: Hexadezimalsystem mit 0, ..., 9, A, B, C, D, E, F für die Werte 0, ..., 15.

Zum Beispiel gilt

$$(47)_{10} = (101111)_2 = (57)_8 = (2F)_{16}.$$

(Un-)Eindeutigkeit der Zahlendarstellung

Stellenwertdarstellungen sind nicht immer eindeutig. Beispielsweise gilt

$$(0, \overline{9})_{10} = (1)_{10}$$

Im Allgemeinen gilt sogar $(0, \overline{b-1})_b = (1)_b$. Um Eindeutigkeit zu erhalten, schließt man Darstellungen aus, die ab einer Stelle nur noch die Ziffer $b - 1$ verwenden.

Endliche und unendliche Nachkommastellen

Beispiel. Es gilt:

$$\frac{1}{8} = (0,125)_{10} = (0,001)_2$$

$$\frac{1}{10} = (0,1)_{10} = (0,000\overline{11})_2 = (0,0001100110011\dots)_2$$

Satz 4.1 (Endliche Nachkommendarstellung). Sei $\frac{r}{s}$ eine vollständig gekürzte rationale Zahl und $b \in \mathbb{N} \setminus \{1\}$. Die folgenden Aussagen sind äquivalent:

- Die Zahl $\frac{r}{s}$ besitzt eine endliche Darstellung zur Basis b .
- $\exists k \in \mathbb{N} : s \mid b^k$.
- Jeder Primteiler von s teilt auch die Basis b .

Binärzahl mit Nachkommastellen

Bestimmen Sie den Dezimalwert von $(101101,011)_2$.

Eigene Lösung / Rechnung:

4.2 Umrechnen zwischen Zahlensystemen

Satz 4.2 (Umrechnung aus dem Dezimalsystem). Gegeben seien eine Basis $b \in \mathbb{N}$ mit $b \geq 2$ und eine reelle Zahl $x \geq 0$. Schreibe $x = N + r$ mit $N = \lfloor x \rfloor$ und $0 \leq r < 1$. Dann gilt

$$x = (a_m \dots a_0, a_{-1} a_{-2} \dots)_b, \quad a_j \in \{0, 1, \dots, b-1\},$$

wobei die Ziffern durch folgende Verfahren bestimmt werden:

- **Vorkommaziffern:** Setze $N_0 = N$. Für $k \geq 0$ sind $N_{k+1} = \left\lfloor \frac{N_k}{b} \right\rfloor$ die Quotienten und $a_k = N_k - bN_{k+1}$ die Reste. Stoppe beim ersten $N_{m+1} = 0$; gelesen wird $a_m \dots a_0$.
- **Nachkommaziffern:** Setze $r_0 = r$. Für $k \geq 1$ sind $a_{-k} = \lfloor br_{k-1} \rfloor$ die ganzzahligen Anteile und $r_k = br_{k-1} - a_{-k}$ die neuen Reste. Ein Rest $r_k = 0$ liefert ab dann nur noch Nullen. Wiederholt sich ein Rest, wiederholen sich die nachfolgenden Ziffern periodisch.

Dezimalzahl in das Hexadezimalsystem

Beispiel. Gesucht ist die Hexadezimaldarstellung von $(156,625)_{10}$.

Für den ganzzahligen Anteil:

$$\begin{aligned} 156 &= 9 \cdot 16 + 12 \\ 9 &= 0 \cdot 16 + 9. \end{aligned}$$

Die Reste werden von unten nach oben gelesen: 9 und $12 = C$, also $(156)_{10} = (9C)_{16}$.

Für den gebrochenen Anteil:

$$0,625 \cdot 16 = 10 + 0.$$

Der ganzzahlige Anteil ist $10 = A$; der Rest ist bereits 0.

Damit gilt

$$(156,625)_{10} = (9C, A)_{16}.$$

Dezimal nach Oktal: Vorkommateil

Beispiel: Dezimal nach Oktal mit Periode

Gesucht ist die Oktaldarstellung von $(345,2)_{10}$. Bestimmen Sie zunächst die Vorkommaziffern durch wiederholte Division durch 8.

Eigene Lösung / Rechnung:

Dezimal nach Oktal: Nachkommateil

Beispiel (Fortsetzung): Nachkommateil und Periode

Bestimmen Sie die Nachkommaziffern von $(345,2)_{10}$ und die Periode.

Eigene Lösung / Rechnung:

Von einer beliebigen Basis ins Dezimalsystem

Beispiel. Für $(1E2, B8)_{16}$ gilt $(E)_{16} = 14$ und $(B)_{16} = 11$. Daher

$$\begin{aligned}(1E2, B8)_{16} &= 1 \cdot 16^2 + 14 \cdot 16 + 2 \\ &\quad + 11 \cdot 16^{-1} + 8 \cdot 16^{-2} \\ &= 256 + 224 + 2 + \frac{11}{16} + \frac{8}{256} \\ &= (482,71875)_{10}.\end{aligned}$$

Zwischen Potenzen einer Basis umrechnen

Satz 4.3 (Gruppierung bei Potenzen der Basis). Seien $b, k \in \mathbb{N}$ mit $b \geq 2, k \geq 1$ und $B = b^k$. Dann bleibt der Wert einer Zahl unverändert, wenn man in ihrer Darstellung zur Basis B jede Ziffer durch ihre Darstellung zur Basis b mit genau k Ziffern ersetzt. Dabei ergänzt man jede Gruppe bei Bedarf links mit Nullen.

Umgekehrt werden die Ziffern zur Basis b **vom Komma aus** in Gruppen zu je k Stellen zusammengefasst und durch je eine Ziffer zur Basis B ersetzt. Unvollständige Gruppen ergänzt man im Vorkommateil links und im Nachkommateil rechts mit Nullen.

Beispiel: Umrechnen mit Vierergruppen ($16 = 2^4$)

$$(A8C, B2)_{16} = (1010\ 1000\ 1100, 1011\ 0010)_2.$$

$$\begin{aligned} (110\ 1110, 101)_2 &= (0110\ 1110, 1010)_2 \\ &= (6E, A)_{16}. \end{aligned}$$

Umrechnen mit Kontrolle

Rechnen Sie $(2026)_{10}$ in das Hexadezimal- und anschließend direkt in das Binärsystem um. Kontrollieren Sie das Ergebnis durch Auswerten.

Eigene Lösung / Rechnung:

4.3 Binärdarstellung natürlicher Zahlen

Definition 4.2 (Vorzeichenlose Binärdarstellung). Ein Bitmuster $x_{n-1} \dots x_1 x_0$ mit $x_j \in \{0, 1\}$ stellt den Wert

$$x = \sum_{j=0}^{n-1} x_j \cdot 2^j$$

dar. x_{n-1} heißt **Most Significant Bit (MSB)**, x_0 heißt **Least Significant Bit (LSB)**.

Mit $n \geq 1$ Bits lassen sich vorzeichenlos genau die Zahlen

$$0, 1, \dots, 2^n - 1$$

darstellen.

Das kleinste Muster ist 00 ... 0, das größte ist 11 ... 1 mit

$$\sum_{j=0}^{n-1} 2^j = 2^n - 1.$$

Ein **Byte** besteht aus acht Bits. Es kann vorzeichenlos die Zahlen von 0 bis $2^8 - 1 = 255$ darstellen.

Ein Byte lesen

Beispiel. Die Zahl 101 besitzt als 8-Bit-Wert die Darstellung

$$(101)_{10} = (01100101)_2.$$

Bitposition	7	6	5	4	3	2	1	0
Bit	0	1	1	0	0	1	0	1
Gewicht	128	64	32	16	8	4	2	1

Das MSB ist 0, das LSB ist 1. Die führende Null gehört zur festen Wortbreite, ändert den Zahlenwert aber nicht.

Addition natürlicher Zahlen

Beispiel: Addition ohne Überlauf

Binäre Addition folgt denselben Stellenregeln wie Dezimaladdition. Dabei gilt $1 + 1 = (10)_2$: Wir schreiben 0 und übertragen 1 in die nächste Stelle.

$$\begin{array}{r} 00101101 \\ +00110111 \\ \hline 01100100 \end{array}$$

Also ist $(45)_{10} + (55)_{10} = (100)_{10}$. Die Summe liegt im mit acht Bits darstellbaren Bereich von 0 bis 255.

Addition mit Überlauf

Beispiel: Überlauf bei acht Bits

Die natürliche Zahl 255 lässt sich mit acht Bits als 11111111 darstellen. Bei der Addition von 1 entsteht

$$\begin{array}{r} 11111111 \\ +00000001 \\ \hline 1\,00000000 \end{array}$$

Mathematisch gilt $255 + 1 = 256$. Dafür sind neun Bits nötig. Wenn nur acht Bits gespeichert werden, bleibt 00000000 übrig: Der Übertrag links vom MSB geht verloren. Das ist ein **Überlauf**.

Speicherwort untersuchen

Ein Gerät liefert das 8-Bit-Muster 11101001.

- (1) Welchen Dezimalwert besitzt es?
- (2) Was geschieht bei der Addition von 00011000?

Eigene Lösung / Rechnung:

4.4 Zweierkomplement

Definition 4.3 (Zweierkomplementdarstellung). Das n -Bit-Muster $x_{n-1} \dots x_0$ besitzt im Zweierkomplement den Wert

$$(x_{n-1} \dots x_0)_{2K} = -x_{n-1}2^{n-1} + \sum_{j=0}^{n-2} x_j 2^j.$$

Der Index $2K$ kennzeichnet die Deutung der Bitfolge im Zweierkomplement.

Das MSB hat damit das negative Gewicht -2^{n-1} ; die übrigen Bits behalten ihre positiven Gewichte.

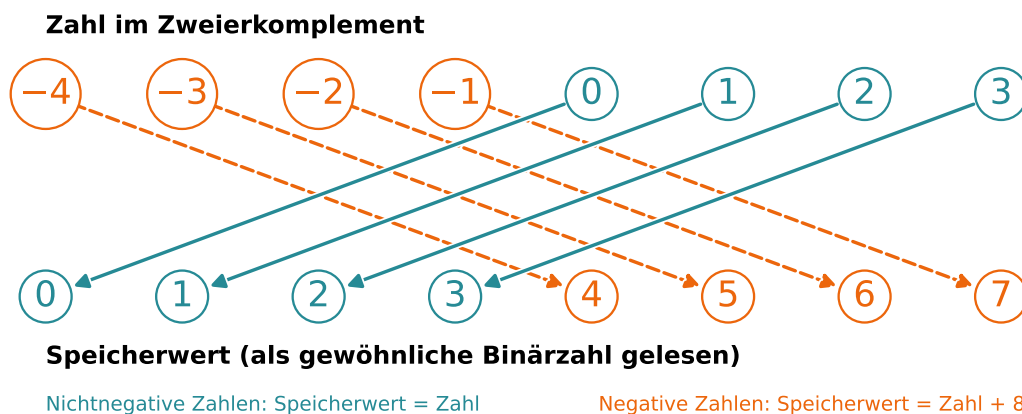
Mit n Bits sind genau die ganzen Zahlen

$$-2^{n-1}, \dots, -1, 0, 1, \dots, 2^{n-1} - 1$$

darstellbar.

Für $n = 8$ reicht der Bereich daher von -128 bis 127 . Bitmuster mit MSB 0 sind nichtnegativ, Bitmuster mit MSB 1 sind negativ.

Beispiel: Kodierung mit drei Bits



Beispielsweise wird -3 durch den Speicherwert 5 codiert: $(5)_{10} = (101)_2$, also $(-3)_{10} = (101)_{2K}$.

Zwei Wege zum Zweierkomplement

Für $-a < 0$ mit $1 \leq a \leq 2^{n-1}$ gibt es zwei Methoden. Beide liefern dieselbe Darstellung mit n Bits:

1. **Über den Speicherwert:** Wir berechnen $2^n - a$ und schreiben das Ergebnis mit genau n Binärziffern. Diese Bitfolge stellt $-a$ im Zweierkomplement dar.
2. **Mit der Bitregel:** Wir schreiben zunächst a mit n Bits. Von rechts übernehmen wir alle Bits bis einschließlich der ersten 1 unverändert. Alle Bits links davon werden invertiert: Aus 0 wird 1 und aus 1 wird 0.

Weg 1: Über den Speicherwert

Beispiel: -100 über den Speicherwert codieren

Stellen Sie -100 mit acht Bits im Zweierkomplement dar. Verwenden Sie die Methode über den Speicherwert.

Eigene Lösung / Rechnung:

Weg 2: Bits von rechts bearbeiten

Beispiel: -12 mit der Bitregel codieren

Stellen Sie -12 mit acht Bits im Zweierkomplement dar. Verwenden Sie die Bitregel.

Eigene Lösung / Rechnung:

Den Dezimalwert bestimmen

Bei MSB 0 hat die Bitfolge denselben Wert wie eine gewöhnliche Binärzahl. Bei MSB 1 lesen wir zunächst ihren gewöhnlichen Binärwert und ziehen anschließend 2^n ab.

Beispiel: Ein negatives Bitmuster lesen

Welchen Wert besitzt $(1010110)_{2K}$ bei sieben Bits?

Das MSB ist 1. Als gewöhnliche Binärzahl gelesen ergibt die Folge

$$(1010110)_2 = 2^6 + 2^4 + 2^2 + 2^1 = 86.$$

Im Zweierkomplement ziehen wir $2^7 = 128$ ab:

$$(1010110)_{2K} = 86 - 128 = (-42)_{10}.$$

Subtraktion durch Addition

Beispiel. Mit acht Bits berechnen wir $64 - 100 = 64 + (-100)$. Die Summanden sind $(64)_{10} = (01000000)_{2K}$ und $(-100)_{10} = (10011100)_{2K}$:

$$\begin{array}{r} 01000000 \\ +10011100 \\ \hline 11011100 \end{array}$$

Das Ergebnis ist negativ. Negieren von 11011100 liefert 00100100 , also 36. Somit gilt $(11011100)_{2K} = (-36)_{10}$.

Bemerkung: Überlauferkennung bei der Addition

Bei der Addition zweier n -Bit-Zahlen im Zweierkomplement liegt genau dann ein Überlauf vor, wenn beide Summanden dasselbe Vorzeichen besitzen, das gespeicherte Ergebnis aber das entgegengesetzte Vorzeichen hat.

MSB(a)	MSB(b)	MSB($a + b$)	Bewertung
0	0	1	Überlauf
1	1	0	Überlauf

Acht-Bit-Rechnung beurteilen

Berechnen Sie im 8-Bit-Zweierkomplement:

(1) $100 + 50$

(2) $-100 - 50$

Entscheiden Sie jeweils, ob das Bitmuster ein gültiges Ergebnis darstellt.

Eigene Lösung / Rechnung:

4.5 Gleitkommadarstellung

Definition 4.4 (Normalisierte Gleitkommazahl). Gegeben seien eine Basis $b \in \mathbb{N}$ mit $b > 1$, eine Mantissenlänge $p \in \mathbb{N}$ mit $p \geq 1$ und Exponentengrenzen $e_{\min}, e_{\max} \in \mathbb{Z}$ mit $e_{\min} \leq e_{\max}$. Eine von null verschiedene normalisierte Gleitkommazahl besitzt die Form

$$x = (-1)^V m b^e,$$

Dabei ist $V \in \{0, 1\}$ das Vorzeichenbit und $e \in \mathbb{Z}$ mit $e_{\min} \leq e \leq e_{\max}$ der Exponent. Die Mantisse lautet

$$m = (m_0, m_1 \dots m_{p-1})_b.$$

Für die Ziffern gilt $m_0 \in \{1, \dots, b-1\}$ und $m_j \in \{0, \dots, b-1\}$ für $j = 1, \dots, p-1$. Damit ist $1 \leq m < b$.

Darstellbare Zahlen bilden ein Raster

Satz 4.4 (Abstand benachbarter Gleitkommazahlen). Seien $b \in \mathbb{N}$ mit $b > 1$, $p \in \mathbb{N}$ mit $p \geq 1$ und $e \in \mathbb{Z}$ mit $e_{\min} \leq e \leq e_{\max}$ fest gewählt. Dann haben benachbarte positive normalisierte Gleitkommazahlen mit diesem Exponenten den Abstand

$$b^{e-p+1}.$$

Wächst der Exponent um 1, wächst auch der Abstand um den Faktor b . Gleitkommazahlen liegen deshalb nicht gleichmäßig auf der Zahlengeraden.

Darstellbare Zahlen auf dem Zahlenstrahl

Beispiel: Darstellbare Zahlen auf dem Zahlenstrahl

Wir wählen $b = 10$, $p = 1$ und $e \in \{-1, 0, 1\}$. Die positive Mantisse ist eine der Ziffern $1, \dots, 9$.

Überblick: 0 bis 90



Vergrößerung: 0 bis 10



Exponent -1: Abstand 0,1 Exponent 0: Abstand 1 Exponent 1: Abstand 10

Kleine Beträge sind absolut genauer darstellbar: Die Abstände sind 0,1, 1 beziehungsweise 10. Für negative Zahlen ist das Raster spiegelbildlich.

Definition 4.5 (Absoluter und relativer Rundungsfehler). Wird eine Zahl $x \in \mathbb{R} \setminus \{0\}$ durch die endliche Maschinenzahl $\text{fl}(x) \in \mathbb{R}$ ersetzt, heißen

$$|\text{fl}(x) - x| \quad \text{und} \quad \frac{|\text{fl}(x) - x|}{|x|}$$

absoluter beziehungsweise relativer Rundungsfehler.

Beispiel: Warum 0,1 im Binärsystem problematisch ist

Die Dezimalzahl 0,1 besitzt die unendliche Binärdarstellung

$$(0,0001100110011 \dots)_2.$$

Ein binäres Gleitkommaformat muss diese Folge abschneiden und runden. Daher ist der gespeicherte Wert im Allgemeinen nur eine sehr gute Näherung an 0,1. Das erklärt, warum wiederholtes Addieren von 0,1 nicht immer exakt beim erwarteten Dezimalwert endet.

Rundungsfehler in R beobachten

Schon bei $0,1 + 0,2$ lässt sich der Unterschied zum erwarteten Dezimalwert 0,3 nachweisen. Mit 17 Ziffern wird er sichtbar:

```
# Ergebnis und Vergleichswert mit 17 Ziffern anzeigen
x <- 0.1 + 0.2
sprintf("%.17g", x)      # "0.30000000000000004"
sprintf("%.17g", 0.3)    # "0.29999999999999999"
x == 0.3                  # FALSE
x - 0.3                   # 5.551115e-17
```

Im HTW-JupyterHub öffnen

R rundet die Ausgabe normalerweise. Der scheinbare Wert 0.3 verbirgt deshalb die kleine Abweichung; ein exakter Vergleich macht sie sichtbar.

Die Reihenfolge der Teilrechnungen ist wichtig

Bemerkung: Reihenfolge der Berechnungen

Bei Berechnungen mit Gleitkommazahlen werden Zwischenergebnisse gerundet. Die dabei entstehenden Rundungsfehler können sich je nach Reihenfolge der Teilrechnungen unterschiedlich auswirken.

Deshalb können mathematisch gleichwertige Rechnungen wie $(a + b) + c$ und $a + (b + c)$ unterschiedliche Ergebnisse liefern. Welche beiden Zahlen zuerst addiert werden, kann also das Endergebnis verändern.

Ein kleiner Summand kann etwa beim Addieren zu einer sehr großen Zahl durch die Rundung verloren gehen. Werden mehrere kleine Summanden zuerst untereinander addiert, kann ihr Beitrag erhalten bleiben.

Die Reihenfolge verändert das Ergebnis

Beispiel: Die Summe $a + b + c$ bestimmen

Ziel ist, $a + b + c$ zu berechnen. Wir rechnen dezimal mit einer dreistelligen Mantisse und vergleichen zwei Klammerungen:

$$a = 3,73 \cdot 10^6, \quad b = -3,71 \cdot 10^6, \quad c = 6,42 \cdot 10^3.$$

Zuerst $a + b$:

$$a + b = 2,00 \cdot 10^4, \quad c \approx 0,64 \cdot 10^4, \quad (a + b) + c = 2,64 \cdot 10^4.$$

Zuerst $b + c$:

$$c \approx 0,01 \cdot 10^6, \quad b + c = -3,70 \cdot 10^6, \quad a + (b + c) = 3,00 \cdot 10^4.$$

Der exakte Wert ist 26420. Die beiden geklammerten Rechnungen liefern wegen der Zwischenrundungen 26400 beziehungsweise 30000. **Die Reihenfolge der Teilrechnungen beeinflusst das Ergebnis: Das Assoziativgesetz gilt hier nicht.**

Kleine Summanden bewahren

Ein System rechnet dezimal mit vier signifikanten Ziffern. Vergleichen Sie

$$(10^4 + 6) + 6 \quad \text{und} \quad 10^4 + (6 + 6).$$

Welches Ergebnis ist jeweils zu erwarten, wenn nach jeder Operation gerundet wird?

Eigene Lösung / Rechnung:

IEEE-754-Darstellung

Definition 4.6 (IEEE 754 binary32). Ein 32-Bit-Gleitkommawort besteht aus

$$\underbrace{V}_{1 \text{ Bit}} \quad \underbrace{E}_{8 \text{ Bit}} \quad \underbrace{M}_{23 \text{ Bit}} .$$

Dabei gilt $V \in \{0, 1\}$ und $E \in \{0, \dots, 255\}$. Das Mantissenfeld M hat 23 Bits mit Ziffern aus $\{0, 1\}$ und einen ganzzahligen Wert aus $\{0, \dots, 2^{23} - 1\}$.

Für $E \in \{1, \dots, 254\}$ ist der Wert einer normalisierten Zahl

$$x = (-1)^V (1, M)_2 2^{E-127}.$$

Das Vorzeichenbit ist V , der gespeicherte Exponent ist E mit Bias 127, und M enthält die Nachkommabits der Mantisse. Die führende 1 der normalisierten Mantisse wird nicht gespeichert.

Runden in binary32

Bemerkung: Zur nächsten Zahl runden, im Gleichstand zur geraden

Die Standardrundung in binary32 wählt die **nächstgelegene darstellbare Zahl**.

Liegt der exakte Wert **genau in der Mitte** zwischen zwei benachbarten darstellbaren Zahlen, wird die Darstellung gewählt, deren **letztes gespeichertes Mantissenbit 0** ist (*round to nearest, ties to even*).

„Gerade“ bezieht sich hier auf den ganzzahligen Wert des Mantissenfeldes M : Eine Binärzahl ist genau dann gerade, wenn ihr letztes Bit 0 ist. Diese Regel vermeidet eine systematische Bevorzugung des Auf- oder Abrundens.

Aufbau eines binary32-Wortes

Bemerkung: Aufbau und normalisierte Zahlen

Vorzeichen V	Exponent E	Mantissenfeld M
1 Bit	8 Bit	23 Bit

- $V = 0$ bedeutet nichtnegativ, $V = 1$ negativ.
- Der tatsächliche Exponent ist $e = E - 127$.
- Für $1 \leq E \leq 254$ lautet die Mantisse $(1, M)_2$, mit einer führenden 1 vor dem Komma. Diese Zahlen heißen **normalisierte** oder **normale Zahlen**. Die führende 1 wird nicht mitgespeichert.
- Die Reihenfolge im Speicher ist $V \mid E \mid M$.

Sonderwerte speichern

Bemerkung: Null, Unendlich und NaN

Für Sonderwerte sind bestimmte Kombinationen von Exponentenfeld E und Mantissenfeld M reserviert:

- $E = 0$ und $M = 0$: **Null**; das Vorzeichenbit unterscheidet $+0$ und -0 .
- $E = 255$ und $M = 0$: **Unendlich**; $V = 0$ ergibt $+\infty$, $V = 1$ ergibt $-\infty$.
- $E = 255$ und $M \neq 0$: **NaN** (*Not a Number*, kein gültiger Zahlenwert).

Dezimalzahl nach binary32

Beispiel. Wir codieren 13,25.

$$(13,25)_{10} = (1101, 01)_2 = (1, 10101)_2 \cdot 2^3.$$

Damit

$$V = 0, \quad E = 3 + 127 = 130 = (10000010)_2,$$

und $M = 101010000000000000000000$.

Das vollständige Wort lautet

0 10000010 101010000000000000000000

Dasselbe Speicherwort, in Vierergruppen geschrieben, ist 0100 0001 0101 0100 0000 0000 0000 0000.

Vier Bits bilden jeweils eine Hexadezimalziffer. So entsteht 0x41540000; das Präfix 0x kennzeichnet die Hexadezimalschreibweise dieses Speicherwortes.

Dezimalzahl nach binary32 mit Rundung

Beispiel: Ein Mittelpunktsfall in binary32

Codieren Sie $x = 8388609,5$ in binary32 mit der Standardrundung. Bestimmen Sie V , E , M und den tatsächlich gespeicherten Dezimalwert. Hinweis: $2^{23} = 8388608$.

Eigene Lösung / Rechnung:

binary32 nach Dezimal

Bestimmen Sie den Wert des binary32-Wortes

1 10000011 011111000000000000000000.

Eigene Lösung / Rechnung:

Größenbereich von binary32 und binary64

Bemerkung: Speicheraufteilung und Größenbereiche

Format	Vorzeichen	Exponent	Mantissenfeld	Bias
binary32 (32 Bit)	1 Bit	8 Bit	23 Bit	127
binary64 (64 Bit)	1 Bit	11 Bit	52 Bit	1023

Die folgenden Grenzen gelten für positive endliche Zahlen:

Format	Kleinster positiver Wert	Größter endlicher Wert
binary32	$\approx 1,401 \cdot 10^{-45}$	$\approx 3,403 \cdot 10^{38}$
binary64	$\approx 4,941 \cdot 10^{-324}$	$\approx 1,798 \cdot 10^{308}$

Die Untergrenzen schließen sehr kleine **nicht normalisierte (subnormale)** Zahlen ohne führende 1 ein. Negative Werte sind spiegelbildlich darstellbar; auch die Null gehört zum Format.

Welche Information liefern die Felder?

Ein binary32-Wort besitzt $V = 0$, $E = 255$ und ein von null verschiedenes Mantissenfeld. Ein zweites Wort besitzt $V = 1$, $E = 0$ und $M = 0$.

- (1) Welche Werteklassen sind codiert?
- (2) Welche Rolle spielt das Vorzeichenbit jeweils?

Eigene Lösung / Rechnung: