Technische Universität München
Institut für Informatik
Prof. Dr. Hans-Joachim Bungartz
Hendrik Möller

WiSe 2022/23

Numerisches Programmieren, Übungen

1. Trainingsblatt: Zahlendarstellung

1) IEEE Umwandlung Schritt für Schritt

Wir haben in den Tutorien das IEEE Zahlenformat kennengelernt. Es nutzt 32 Bits: Eins für ein Vorzeichen, 8 für den Exponenten und die restlichen 23 für die sogenannte Mantisse. Die folgende Aufgabe soll die schrittweise Umwandlung einer dezimalen Zahl zum IEEE Format vertiefen.

a) Wandeln Sie 14 in das binäre System um.

b) Überlegen sie aus (a), welcher (dezimale) Exponent benötigt wird, damit eine Mantisse normiert ist.

c) Leiten Sie nun nacheinander aus (b) erst das Vorzeichenbit, dann die Exponentenbits und dann die Mantisse her.

d) Wandeln Sie nun $-\frac{1}{6}$ in das IEEE Format um.

a) □ 8 4 2 1
16 8 4 2 1

b) $1110_2 = 1.11 \times 2^3$
Exponent ist 3

c) Vorzeichen: 0
Exp: 10000011 $3+127$?
8T $= 130$
Mantisse: 11 00 ....

(d) $6 = 110$

$$\begin{array}{r} 0.0010101 \\ 110 \overline{)1000} \\ 110 \\ \hline 1000 \\ 110 \\ \hline 1000 \end{array}$$

$0.0010101\ldots$

$= 1.010101\ldots \times 10^{-3}$

1 Bit | 8 Bit | 23 Bit
1 | | 0101...01

$-3+127 = 124$

Auf der nächsten Seite folgt die Lösung...

Lösung:

a) 14 lässt sich binär herunterbrechen auf $8 + 4 + 2 = 14$. Somit ist die binäre Zahl 1110.

b) Die binäre Zahl ist 1110. Eine Mantisse ist genau dann normiert, wenn diese eine führende Eins hat und sofort ein Komma folgt. Wir können also mit folgender Rechnung herleiten:

$$1110 = 111.0 \cdot 2 = 111.0 \cdot 2^1 = 11.10 \cdot 2^2 = 1.110 \cdot 2^3$$

Der Exponent ist also eine Kommaverschiebung nach links. Ein negativer Exponent wäre entsprechend eine Verschiebung nach rechts. Der benötigte Exponent für die Normierung ist hier also 3, da hierbei die Mantisse 1.110 entspricht. Eine führende Eins und direkt danach ein Komma.

c) Vorzeichenbit ist das einfachste: Die Zahl 14 ist positiv, also ist das Vorzeichenbit 0.

Der dezimale Exponent ist 3, wie wir aus (b) wissen. Auf diesen müssen wir den IEEE Offset von 127 addieren. Wir erhalten 130. Diese Zahl müssen wir jetzt nur noch in Binär darstellen. Das ist entsprechend 10000010. 128+2

Die Mantisse können wir direkt aus (b) auslesen, sie ist 1.110.

Wenn wir jetzt alles zusammensetzen, müssen wir nur aufpassen, dass wir von der Mantisse nicht die führende Eins oder das Komma mitnehmen, da diese bei IEEE nicht abgespeichert wird. Wir erhalten (mit Reihenfolge Vorzeichen|Exponent|Mantisse):

$$0|10000010|11000000000000000000000$$

d) Erstmal in binär umwandeln. Das ist nicht trivial, also muss man schriftliche Division im binären Bereich anwenden. $-\frac{1}{6} = -001_2 : 110_2$ $-\frac{1}{6}$ ist also $0.0\overline{01}_2$. Der benötigte Exponent

$$\begin{array}{llllll}
1 & & & & : 110_2 & = \\
1 & 0 & 0 & 0 & & = 0.00 \\
- & 1 & 1 & 0 & & = 0.001 \\
\hline
 & 0 & 1 & 0 & 0 & \\
 & & 1 & 0 & 0 & 0 \quad = 0.0010 \\
 & - & 0 & 1 & 1 & 0 \quad = 0.00101 \\
\hline
 & & & & 1 & 0 \\
 & & & & \multicolumn{2}{l}{\ldots}
\end{array}$$

ist -3, da dann die Mantisse $1.\overline{01}_2$ und damit normiert ist. Jetzt noch in IEEE Format umwandeln.

Vorzeichenbit ist 1, da die Zahl negativ ist.

Exponent ist $-3 + 127 = 124$ in binär, also 01111100.

Die Mantisse können wir ablesen, die Periode müssen wir solange wiederholen, bis uns die Bits ausgehen. Also 01010101010101010101010|101. Wir müssen aber alles nach den 23 Bits (hier mit einem | markiert) runden. Hier haben wir den eindeutigen Fall, bei dem aufgerundet wird.

Vollständig haben wir also: 1|01111100|01010101010101010101011.

2) Gleitkomma Arithmetik

Nun gucken wir uns das Thema der Maschinenzahlen etwas allgemeiner an.

a) Was ist der Nachteil eines Vorzeichenbits? Warum entsteht dieser Nachteil?

b) Nehmen Sie eine binäre Darstellung mit 3 signifikanten Stellen an. Formulieren Sie eine Rechnung zwischen 2 Zahlen ihrer Wahl, bei dem das Problem der Absorption auftreten wird. Berechnen Sie von Ihrer Rechnung den absoluten sowie relativen Fehler.

c) Berechnen Sie $101.11 - 110.011$ bei einer binären Darstellung mit 3 signifikanten Stellen. Runden Sie immer wie in der Übung besprochen. Was passiert hier? Berechnen Sie den relativen Fehler dieser Rechnung.

a) Bit - Verschwendung
da "0" hat zwei Binärdarstellung
nämlich ein positive null und
ein negative null

b) $1.00 + 0.0001 = 1.001 \overset{\text{Runden}}{=} 1.0001$
exakt: 1.001
absolut: $|x - rd(x)| = |0.001| = 0.001$
relativ: $\frac{0.001}{1.001}$

c) exakt: $|01.11 - 110.011|$

Auf der nächsten Seite folgt die Lösung...

Lösung:

a) Das Vorzeichenbit bringt meistens den Nachteil der uneindeutigen Null. Es gibt eine bestimmte Kombination aus Exponent und Mantisse (meist nur Nullen), die für den Wert der 0 steht. Dadurch kann man sowohl eine $+0$ als auch eine -0 darstellen, obwohl diese vom Wert her identisch sind.

b) Hier kann man viele Beispiele bringen. Am einfachsten ist es zwei Zahlen zu wählen, die mehr als die drei Stellen auseinander liegen. Also ein Beispiel:

$$100 + 0.011 = 100.011$$

Hier würde 100 die 0.011 absorbieren, das eigentlich exakte Ergebnis 100.011 (4.375 in dezimal) muss gerundet werden, da es mehr als 3 signifikante Stellen hat. Hier wird abgerundet, da die erste nicht darstellbare Ziffer eine 0 ist. Das gerundete Ergebnis ist 100. Damit ist die Addition also völlig wertlos gewesen!

Der absolute Fehler berechnet sich durch $|x - \mathrm{rd}(x)|$. In dem Fall also $100.011 - 100 = 0.011$. Im dezimalen Bereich entspräche das also $\frac{3}{8}$.

Um den relativen Fehler zu erhalten müssen wir den absoluten nochmal durch das exakte Ergebnis teilen, in dem Fall also $\frac{3/8}{4.375} = 0.0857$, also etwa 8.6% Abweichung.

c) Vor der Berechnung müssen wir beide Zahlen runden, da sie mehr als 3 signifikante Stellen haben. 101.11 rundet sich auf zu 110 und 110.011 rundet man ab auf 110. Das Ergebnis der Subtraktion ist trivialerweise 0. Der exakte Wert wäre aber $101.11 - 110.011 = -0.101$ (-0.625 in dezimal). Der relative Fehler ist logischerweise immens! Dieses Phänomen nennt man Auslöschung.

3) Fremde Zahlenformate

In dieser Aufgabe werden fiktive Zahlendarstellungen vorgestellt und dazu Verständnisfragen gestellt. Es handelt sich also um Transferaufgaben.

a) Ein Zahlenformat, hier mal mit NDR abgekürzt, verwendet ein Vorzeichenbit, 3 Bits für einen Exponenten, wobei dieser vorzeichenbehaftet ist (das erste dieser 3 Bits steht also für das Vorzeichen des Exponentens) und sonst einer Ganzzahl entspricht. Für die Mantisse stehen 4 Bits zur Verfügung. Diese beginnt immer mit einer führenden Eins und ist wie bei IEEE normiert. Allerdings muss die führende Eins bei NDR abgespeichert werden.

Hinweis: 0|101|1000 im NDR entspräche $2^{-1} \cdot 1.000 = 0.1 = \frac{1}{2}$

i) Was ist die größte und kleinste darstellbare Zahl mit dem NDR Format?

ii) Bestimmen Sie die Maschinengenauigkeit von NDR.

iii) Berechnen Sie die betragsmäßig kleinste Zahl < 0 des NDR Formates.

b) Das CNF (crazy numbering format) ist folgendermaßen aufgebaut: Die Mantisse besteht aus 3 Bits. Allerdings muss diese nicht normiert sein, sondern wird direkt als binäre Zahl interpretiert. Der Exponent besteht auch aus 3 Bits und wird im 2-Komplement Format dargestellt. Es gibt kein Vorzeichenbit.

Hinweis: 001|011 in CNF entspräche $2^1 \cdot 011 = 110 = 6$.

i) Ist das CNF Format eindeutig? Machen Sie ein Beispiel, dass Ihre These bekräftigt.

ii) Geben Sie die kleinste Zahl > 0 an, die mit dem CNF gerade nicht mehr darstellbar ist.

iii) Wieviele verschiedene Zahlen kann das CNF wirklich repräsentieren? (Hard)

Auf der nächsten Seite folgt die Lösung...

Lösung:

a) i) Die kleinste darstellbare Zahl erhalten wir, wenn wir das Vorzeichen auf 1 setzen, die Zahl also negativ wird und ansonsten probieren die betragsmäßig größte Zahl zu erhalten. Das wäre bei 1|011|1111 der Fall. Wichtig hier ist, dass das erste Bit des Exponenten keine Eins ist, sonst wäre der Exponent negativ und würde unsere Zahl betragsmäßig klein machen. Obige Darstellung entspräche übersetzt $-1 \cdot 2^3 \cdot 1.111 = -1111$. Dezimal wäre das eine -15.

Die größte darstellbare Zahl machen wir analog. 0|011|1111 ist $2^3 \cdot 1.111 = 1111$ und damit eine $+15$.

ii) Die Maschinengenauigkeit beschreibt die größte positive Zahl, die mit Eins addiert gerundet wieder Eins ergibt (also genau weg absorbiert wird). Also müssen wir uns überlegen, wie klein diese Zahl sein muss. Das ist bei NDR genau dann der Fall, wenn unsere Mantisse nicht mehr ausreicht, um die Zahl darzustellen. Schon mit der addierten Eins eingerechnet muss unsere Mantisse also 1.0001 entsprechen, die gesuchte Zahl ergo 0.0001. Das ist äquivalent zu $1 \cdot 2^{-4} = \frac{1}{16}$.

Wir können auch überprüfen, dass das stimmt, in dem wir den Weg rückwärts gehen. Wir legen die Maschinengenauigkeit als 0.0001 fest. Also muss $0.0001 + 1.0$ gerundet im NDR Format Eins ergeben. Wir probieren 1.0001 in NDR darzustellen. Das wäre 0|000|10001. Die Mantisse muss gerundet werden, und hier haben wir den Fall, bei dem abgerundet wird. Also erhalten wir 0|000|1000, das im dezimalen der Eins entspricht. Die Maschinengenauigkeit kann auch nicht größer sein, da sobald wir unsere Zahl etwas erhöhen, z.B. 000100000001, diese bei der Rechnung aufgerundet wird und somit eine Zahl größer Eins herauskommen würde.

iii) Wir suchen eine Zahl < 0, also ist das Vorzeichenbit eine Eins, die gesuchte Zahl ist negativ.

Betragsmäßig kleinste Zahl, also muss der Exponent so gering wie möglich sein. Das ist bei 111 der Fall, da das einer -3 entspricht.

Die Mantisse muss eine führende Eins haben, aber weitere Einsen würden die Zahl betragsmäßig nur größer machen, also setzen wir sie auf 1000.

Insgesamt haben wir also 1|111|1000. Das entspricht $-1 \cdot 2^{-3} \cdot 1.000 = -0.001$ und das wiederum der $-\frac{1}{8}$.

b) i) Nein, das CNF ist keinesfalls eindeutig. Man betrachte die Zahlen 001|010 und 010|001 an. Übersetzt erhält man $2^1 \cdot 10 = 100 = 2^2 \cdot 001 = 010|001$. Beide repräsentieren die Zahl 4.

ii) Fangen wir mit der kleinsten Zahl > 0 an, die darstellbar ist mit CNF. Das wäre 100|001. Das entspricht nämlich $2^{-4} \cdot 1 = 2^{-4}$. Jetzt suchen wir nur noch die nächstkleinere Zahl, die eben nicht mehr darstellbar ist. Der Twist ist, dass wir jetzt nicht als Mantisse 0001 wählen können, da diese ja direkt als binäre Zahl interpretiert wird. Nur das Verringern des Exponenten kann die Zahl noch kleiner machen. Die nächstkleinere Zahl, die also nicht mehr darstellbar wäre, ist 2^{-5}.

iii) Rein theoretisch würde man die Anzahl der verschiedenen Bit-Kombinationen zählen. In diesem Fall haben wir 6 Bits, es gibt also $2^6 = 64$ verschiedene Kombinationen, wie die Bits gesetzt werden können. Leider stimmt das aber nicht, da wir durch die fehlende Normierung sehr viele Kombinationen haben, die zu demselben Wert führen, siehe dazu (i). Wir können auch nicht einfach die kleinste und größte

Zahl anschauen und davon auf die Anzahl Zahlen schließen, da es einige Lücken gibt. Zum Beispiel können wir die Zahl 9 nicht darstellen, jedoch aber die 8 und die 10. Es gibt aber immer nur 8 Möglichkeiten für die Mantisse und diese repräsentieren die Werte von 0 bis 7 in ganzen Schritten. Das in Kombination mit den verschiedenen Exponenten hilft uns weiter.

Nehmen wir einen Exponent von 000 an, also 2^0. Das gibt uns also die Zahlen von 0 bis 7 (mit den verschiedenen Kombinationen der Mantisse). Der Exponent 001 gibt mit den verschiedenen Mantissen die Zahlen $\{0, 2, 4, 6, 8, 10, 12, 14\}$. Wenn wir alle durchgehen:

Exponent	Faktor	Mögliche Zahlen	Neu hinzugekommen
000	1	$\{0, 1, 2, 3, 4, 5, 6, 7\}$	$\{0, 1, 2, 3, 4, 5, 6, 7\}$
001	2	$\{0, 2, 4, 6, 8, 10, 12, 14\}$	$\{8, 10, 12, 14\}$
010	4	$\{0, 4, 8, 12, 16, 20, 24, 28\}$	$\{16, 20, 24, 28\}$
011	8	$\{0, 8, 16, 24, 32, 40, 48, 56\}$	$\{32, 40, 48, 56\}$

Spätestens hier kann man jetzt schon ein Muster erkennen. Jeder Exponent fügt 4 neue mögliche Zahlen hinzu, da alle anderen schon durch den vorherigen Exponenten darstellbar sind. Das können wir auch mit der Mantisse aus 3 Bits erklären. Aus den 8 Möglichkeiten der Mantisse gibt es 4, die man redundant mit dem Exponenten ändern kann. Das wäre einmal 000, da Null mal einen Faktor immer noch Null ergibt. Die anderen sind 001, die man durch einen Exponentenänderung äquivalent zu der Mantisse 010 machen kann, wie in (i). 010 kann man zu 100 shiften und 011 zu 110.

So wird das also für die negativen Exponenten genauso laufen. Wir können also jetzt schon berechnen, wieviele verschiedene Zahlen das CNF darstellen kann. Zur Verständlichkeit halber liste ich hier trotzdem mal ein Beispiel auf:

Exponent	Faktor	Mögliche Zahlen	Neu hinzugekommen
111	$1/2$	$\{0, \frac{1}{2}, 1, \frac{3}{2}, 2, \frac{5}{2}, 3, \frac{7}{2}\}$	$\{\frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \frac{7}{2}\}$
110	$1/4$	$\{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, \frac{5}{4}, \frac{3}{2}, \frac{7}{4}\}$	...
101	$1/8$	...	
100	$1/16$		

Insgesamt haben wir also die Start 8 Zahlen (von 0 bis 7) plus sieben verschiedene mögliche Exponenten, die alle genau 4 neue Zahlen hinzufügen.

$8 + 7 \cdot 4 = 36$. CNF kann genau 36 unterschiedliche Zahlen darstellen, obwohl 64 unterschiedliche Kombinationen der Bits existieren. Man kann also auf $64 - 36 = 28$ redundante Zahlendarstellungen schließen!