Fließkommaarithmetik auf Z80-Rechnern

Jens Müller, 18.04.2026

korrigierte Version vom 22.04.2026

Motivation

- Auf 8-Bit-Rechnern werden Fließkommaberechnungen meistens in einem BASIC-Interpreter ausgeführt und sind für viele Anwender eine undurchsichtige Black Box.
- Fließkommaberechnungen gelten als langsam.
- Implementierung einer 32-Bit-Fließkommaarithmetik im BASIC-Compiler von JKCEMU 0.9.9

Abbildung binärer Kommazahlen

Bit 2	Bit 1	Bit 0	Komma	Bit -1	Bit -2	Bit -3
4	2	1		0,5	0,25	0,125
2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}

Beispiele:

3,25 dezimal	011,010 binär
0,125 dezimal	000,001 binär

Approximierte Zahlen (1)

- Alle binären Kommazahlen, deren Nachkommaanteil nicht aus der Summe von Teilungen durch 2 gebildet werden kann,
  sind nicht exakt abbildbar (z.B. 0,1 0,3 0,7)
- Solche Dezimalzahlen werden mit Binärzahlen abgebildet, deren jeweiliger numerischer Wert nur angenähert der Dezimalzahl entspricht → Approximation
- **Dezimale Kommazahlen sind als Binärzahlen meistens nicht exakt numerisch abbildbar!**

Approximierte Zahlen (2)

- Durch die Approximation entstehen Rundungsfehler, die mit jedem Rechenschritt größer werden**!**

  Beispiel in BASIC:

```
10 A=0
20 FOR I=1 TO 1000
30 A=A+0.1
40 NEXT I
50 PRINT A
```

Fließkommazahlen (1)

- Wenn das Komma fließend ist (nach links und rechts verschiebbar), lassen sich bei gleicher Genauigkeit sehr kleine und sehr große Zahlen abbilden.
- Abbildung mit Mantisse (Basiszahl) und Exponent (Position des Kommas)
- Größe der Mantisse bestimmt die Genauigkeit.
- Größe des Exponenten bestimmt den Wertebereich.

Fließkommazahlen (2)

Mantisse und Exponent am Beispiel von Dezimalzahlen:

Mantisse	Exponent	Wert
12345	0	$12345 * 10^{0} = 12345$
123	-4	$123 * 10^{-4} = 0{,}0123$
9876	2	$9876 * 10^{2} = 987600$

Fließkommazahlen auf Z80-Rechnern

- ZX81 BASIC, ZX Spectrum BASIC und Locomotive BASIC (Amstrad CPC und damit auch KC compact) arbeiten mit 40-Bit-Fließkommazahlen (4 Byte Mantisse, 1 Byte Exponent).
- Microsoft BASIC für i8080 von 1978 arbeitet mit 32-Bit-Fließkommazahlen (3 Byte Mantisse, 1 Byte Exponent).
- Die meisten der in der DDR verwendeten BASIC-Interpreter basieren auf Microsoft BASIC und arbeiten deshalb auch mit 32-Bit-Fließkommazahlen.

32-Bit-Fließkommazahlen

- 8 Bit Exponent → Wertebereich ca. $\pm 10^{-38}$ bis $\pm 10^{38}$
- 24 Bit Mantisse → Genauigkeit ca. 7,2 Dezimalstellen
  (0FFFFFFh = 16777215)
- Wegen Fehleranteil durch Approximationen wird bei der Ausgabe häufig auf 6 Dezimalstellen gerundet.
- 1 Bit Vorzeichen → **das sind zusammen 33 Bit**!
  → siehe Normalisierung (später)

Sonderfall Null

- Numerisch 0 muss oft separat betrachtet werden (z.B. bei Division)
- Deshalb Festlegung:
  Wert 0 = alle Bytes Null, bei 8-Bit-Rechner nur Test auf Exponent = 0
- Da Exponent = 0 für numerisch 0 steht, muss der „normale“
  Exponent 0 anders abgebildet werden → Offset (Bias)
- Offset (bei 32 Bit) historisch meistens 80h,
  in IEEE 754 mit 7Fh festgelegt

Normalisierung

- Exponent = 0 → Zahl hat Wert 0
  Exponent <> 0 → Zahl hat Wert ungleich 0
- Ist der Exponent <> 0, ist der Wert der ganzen Zahl <> 0 und damit ist auch die Mantisse <> 0 und muss mindestens ein 1-Bit haben. Das höchste gesetzte Bit wird ganz nach links geschoben (und der Exponent entsprechend verkleinert) → Bei einer Nicht-Null-Zahl ist das oberste Bit der Mantisse immer gesetzt und braucht deshalb nicht gespeichert zu werden → An dieser Stelle steht das Vorzeichen.

IEEE 754

- Mit Aufkommen von mathematischen Co-Prozessoren Anfang der 80er Jahre (i8087 erschien 1980) wurde eine Standardisierung notwendig.
- IEEE 754 wurde 1985 verabschiedet und
  definiert verschiedene Fließkommaformate, u.a.
  32 Bit (einfache Genauigkeit, float, single)
  64 Bit (doppelte Genauigkeit, double)
- Das oberste Bit ist das Vorzeichen, danach kommt der Exponent
  → auf 8-Bit-Rechnern so nicht performant abbildbar,
  da der Exponent um 1 Bit zur Byte-Grenze verschoben ist

Addition und Subtraktion

- Exponenten beider Zahlen müssen gleich sein
  → Operand mit kleinerem Exponent wird angepasst:
  Mantisse wird solange nach rechts geschoben und Exponent entsprechend erhöht, bis die Exponenten gleich sind
- Sind die Exponenten gleich, können die Mantissen einfach addiert bzw. subtrahiert werden.
- Bei einem Überlauf wird die Mantisse um ein Bit verschoben und der Exponent angepasst.
- Zum Schluss wird das Ergebnis normalisiert.

Multiplikation und Division

- Exponenten werden addiert (Multiplikation) bzw. subtrahiert (Division).
- Mantissen werden multipliziert bzw. dividiert.
- Ergebnis wird wieder normalisiert.

Quadratwurzel

- Quadratwurzel wird meistens mit dem iterativen Heron-Verfahren berechnet: $x_{n+1} = (x_n + y_n) / 2$
- Startwert: Eingangswert mit halbiertem (um ein Bit nach rechts verschobenem) Exponenten
  → Startwert liegt damit schon in der Nähe des Ergebnisses.
  → für 32 Bit Genauigkeit damit nur 6 Iterationsschritte notwendig

Sinus, Cosinus, Tangens, Arkustangens

- Sinus: Taylorreihe mit vorberechneten Koeffizienten:
  $\sin(x) = x - x^3/3! + x^5/5! - x^7/7! + \ldots$
  Koeffiziententabelle: -1/3!, +1/5!, -1/7!, +1/9!, -1/11!, ...
- Für 32 Bit Genauigkeit reichen 6 Iterationsschritte
- $\cos(x) = \sin(x + 90°)$
- $\tan(x) = \sin(x) / \cos(x)$
- $\arctan(x) = x - x^3/3 + x^5/5 - x^7/7 + \ldots$
  9 Iterationsschritte für 32 Bit Genauigkeit notwendig

Natürlicher Logarithmus (1)

- Der Logarithmus wird mit einer Taylorreihe berechnet, die jedoch nur im Bereich 0,75 bis 1,0 schnell konvergiert.
- Zuerst wird x durch n-fache Teilung durch 2 oder n-fache Multiplikation mit 2 (bei Binärzahlen sehr schnell) in den Bereich 0.5 bis 1 gebracht und eine Differenz berechnet,
  die dann später zu addieren ist:

  $$\ln(x * n / 2) = \ln(x) + (n * \ln(0.5))$$
  $$\ln(x * n * 2) = \ln(x) + (n * \ln(2))$$

Natürlicher Logarithmus (2)

- Bei $x < 0.75$ wird x mit 1.5 multipliziert und o.g. Differenz um ln(1.5) reduziert, da sonst die Taylorreihe zu langsam konvergiert.
- Berechnung mit folgender Reihe:

$$\ln(x) = \ln((1+a)/(1-a))$$
$$= 2a + 2/3a^3 + 2/5a^5 + \ldots$$
$$= 2 * (a + (a^3)/3 + (a^5)/5 + \ldots)$$

- Alternativ kann auch folgende, allgemein bekannte Taylorreihe verwendet werden (konvergiert aber langsamer):

$$\ln(1+x) = x - x^2/2 + x^3/3 - x^4/4 + x^5/5 - \ldots$$

Exponentialfunktionen

- Die Exponentialfunktion e^x wird so zerlegt, dass die eigentliche Berechnung mit Hilfe der Taylorreihe mit einem Wert kleiner 1 erfolgen kann, da nur in diesem Bereich die Reihe ausreichend schnell konvergiert.

  $e^x = e^{(k*\ln(2))+r} = e^{k*\ln(2)} * e^r = 2^k * e^r$ (Verschiebung von e^r um k Binärstellen)

  $x = k*\ln(2) + r$

  $r = x - (k*\ln(2))$

- Taylorreihe: $e^x = 1 + x + x^2/2! + x^3/3! + \ldots$ mit $x = r$
- Power-Funktion: $a^b = e^{b*\ln(a)}$

Cordic

- Cordic (Coordinate Rotation Digital Computer) ist ein effizienter Algorithmus, mit dem sich viele mathematische Funktionen implementieren lassen.
- 1959 in den USA für Navigationsrechner für Bomber entwickelt
- 1971 zur heute üblichen Form erweitert
- Arbeitet nur mit Addition, Subtraktion, Schiebebefehlen und einer Tabelle mit Arkustangens-Werten (für sin, cos)
  → Dadurch gut in Hardware implementierbar (Co-Prozessoren)
- Aufgrund Speicherplatzbedarf und vieler Iterationsschritte
  selten auf Z80-Rechnern implementiert

Speicherplatzbedarf

- Die vier Grundrechenarten inklusive Ein- und Ausgabe (Umwandlung Dezimalzahl <> Binärzahl) lassen sich in weniger als 2 kByte Programmcode implementieren.
- Beispielprogramm im JKCEMU BASIC Compiler:

```
dim a as single, b as single
input "a=";a
input "b=";b
print "a+b=";a+b
print "a-b=";a-b
print "a*b=";a*b
print "a/b=";a/b
```

Performance

- Performance von binären 32-Bit Fließkommaberechnungen auf Z80-Rechnern:
  - Addition/Subtraktion: ca. 400 bis 1000 Taktzyklen,
    abhängig davon, wie weit Exponenten auseinander liegen
  - Multiplikation/Division: ca. 3500 bis 4500 Taktzyklen
  - Quadratwurzel: ca. 20000-25000 Taktzyklen
  - Sinus: ca. 50000 Taktzyklen
  - Logarithmus und Exponential-Funktion: ca. 35000-70000 Taktzyklen

Fließkommaarithmetik mit BCD

- BCD (Binary Coded Decimal) speichert die Zahlen im Dezimalsystem
  → keine Rundungsfehler
  → geeignet für Finanzberechnungen
- Performance von 6-Byte-BCD-Arithmetik auf Z80-Rechnern:
  Multiplikation/Division: ca. 40000 Taktzyklen
- BCD-Arithmetik ist wesentlich langsamer als binäre Fließkommaarithmetik.

Zusammenfassung

- 32-Bit binäre Fließkommaarithmetik ist auf Z80-Rechnern ein guter Kompromiss aus:
  - Rechengenauigkeit (6 angezeigte Dezimalstellen)
  - Performance (nur wenig langsamer als 32-Bit-Integer-Berechnungen)
  - Implementierungsaufwand (weniger als 2 kByte für Grundrechenarten und Ein-/Ausgabe)
- Schlechte Performance im BASIC-Interpreter liegt mehr am Interpreter als an der binäre Fließkommaarithmetik

Zum Schluss

Vielen Dank für die Aufmerksamkeit!

Gibt es Fragen?