1 Fließkommazahlen

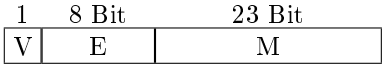
Fließkommazahlen erlauben, einen viel größeren Zahlenbereich darzustellen, allerdings mit einer begrenzten Genauigkeit ("Anzahl der gültigen Ziffern"). Die Zahl wird dargestellt durch einen Exponenten E, eine Mantisse M ($0 \leq M < 1$) und ein Vorzeichenbit V:

$$x = (-1)^V \times 2^E \times M$$

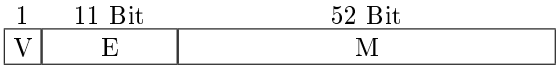
Die genauen Regeln für die Darstellung, Rundungsregeln, Fehlerbehandlung etc. in der Fließkommaarithmetik sind im amerikanischen Standard IEEE 754 aus dem Jahr 1985 festgelegt, der als IEC 559 im Jahr 1989 auch internationaler Standard wurde.

Die Intel-32Bit-Architektur kennt drei Formate:

Einfach langes Format: 32 Bit



Doppelt langes Format: 64 Bit



Erweitertes Format: 80 Bit

Gnu Fortran (`gfortran`) kennt diese Typen als `real, real(kind=8)` und `real(kind=10)`.

Intern arbeitet die Intel-CPU anstelle des doppelt langen immer mit dem erweiterten Format. Dies kann z.B. zu leicht unterschiedlichen Verhalten zwischen mit und ohne Optimierung kompilierten Programmversionen führen: Ohne Optimierung wird ein Zwischenresultat eventuell im RAM zwischengespeichert und dabei auf `double`-Format gekürzt, während es in der optimierten Version in einem (80 Bit langen) Prozessorregister gehalten wird und damit genauer ist.

	einfach	doppelt	erweitert
Exponentenlänge E (Bits)	8	11	15
kleinster Exponent e_{min}	-125	-1021	-16382
größter Exponent e_{max}	128	1024	16384
Mantissenlänge	24	53	64
betragsmäßig kleinste normalisierte Zahl größer Null	1.18×10^{-38}	2.23×10^{-308}	3.36×10^{-4932}
größte darstellbare Zahl	3.40×10^{38}	1.79×10^{308}	1.19×10^{4932}
kleinste Zahl ϵ, für die $1 + \epsilon \neq 1$	1.19×10^{-7}	2.22×10^{-16}	1.08×10^{-19}

- Die Größe ϵ begründet die Sprechweise "Einfach genaue Zahlen haben eine Genauigkeit von etwa 7 Dezimalstellen, doppelt genaue Zahlen etwa 16 gültige Dezimalstellen."
- Der Exponent wird nicht im Zweierkomplement o.ä. kodiert, sondern als vorzeichenlose Zahl, von der eine feste Zahl (bias) abgezogen wird.
- Die Exponenten überdecken nicht den vollen Wertebereich von E Bits. Das Bitmuster "111...11" ($e_{max} + 1$) im Exponentenfeld ist reserviert, um die speziellen Werte `NaN` ("Not a Number", Mantisse ungleich 0) und $\pm$`Unendlich` (Mantisse=0, V=0 oder 1) zu kodieren. Bei einfach- und doppeltgenauen Zahlen ist das Bitmuster "000...00" ($e_{min} - 1$) reserviert für denormalisierte Zahlen.
- Normalisierte und denormalisierte Zahlen: Die Aufteilung in Exponent und Mantisse ist nicht eindeutig. So kann (der Einfachheit halber im Dezimalsystem, die Mantisse sei dreistellig) 0.1 geschrieben werden als $.100 \times 10^0$, $.010 \times 10^1$ und $.001 \times 10^2$. Die erste Form, bei der die erste Stelle der Mantisse ungleich Null ist, ist die normalisierte Form. Nun ist die Mantisse binär kodiert, also ist die erste Stelle bei normalisierten Zahlen immer 1. Daher kann man auf die Speicherung der ersten Stelle der

Mantisse verzichten (implizites erstes Bit) und dies wird bei einfach- und doppeltgenauen Zahlen auch so gemacht. Dies erklärt die Differenz zwischen den Angaben für die Mantissenlänge in der Tabelle (24 bzw. 53) und der vorhergehenden Angabe über die Zahl der Bits, die zum Speichern der Mantisse zur Verfügung stehen (23 bzw. 52). Im erweiterten Format wird das erste Bit mit kodiert.

- Denormalisierte Zahlen werden nur dann verwendet, wenn sie betragsmäßig so klein sind, daß eine normalisierte Darstellung nicht mehr möglich ist. Die kleinste normalisierte einfach genaue Zahl hat die Mantisse .1000...0 binär ($= 2^{-1} = 0.5$) und den Exponenten $e_{min} = -125$, ist also gleich $2^{-125} \times 2^{-1} = 2^{-126} \approx 1.18 \times 10^{-38}$. Wird diese Zahl nun z.B. durch 8 geteilt, so ist das Ergebnis nur noch denormalisiert mit der Mantisse .00010...0 binär darstellbar. Aufgrund des impliziten ersten Bits muß man denormalisierte Zahlen durch einen speziellen Wert im Exponentenfeld kennzeichnen. (Interpretiert werden sie natürlich mit dem Exponenten $e = e_{min}$.)
- Das Erreichen denormalisierter Zahlen ist natürlich mit einem Genauigkeitsverlust begleitet. Trotzdem hat es sich in der Praxis als vorteilhaft herausgestellt, daß dadurch die "Lücke um die Null" kleiner und ein "gradual underflow" möglich ist. Es gibt auch Computersysteme (z.B. Cray, IBM/390), die keine denormalisierten Zahlen zulassen.

2 Zeichensätze

ASCII

Der American Standard Code for Information Interchange (ASCII) ist ein Zeichensatz, der nur 7 von den 8 Bits eines Bytes benutzt und demzufolge 128 Zeichen umfaßt.

Die ersten 32 Zeichen 0x00...0x1F (in hexadezimaler Notation) sind nicht druckbare Zeichen und enthalten z.B. das Nullbyte 0x00 (in C als `\0` darstellbar und als Endmarkierung von Zeichenketten verwendet), den Zeilentrenner 0x0A (`\n`) und das Tabulatorzeichen 0x09 (`\t`).

0x20 kodiert das Leerzeichen. Auf Position 0x30...0x39 stehen die Ziffern 0...9, auf Position 0x41 bis 0x5A die Großbuchstaben A...Z und auf Position 0x61...0x7A die Kleinbuchstaben.

Weitere druckbare Zeichen (neben 0-9, A-Z, a-z und Leerzeichen) sind

```
! " # $ % & ' ( ) * + , - . / : ;  < = > ? @ [ ] \  ^ _ ‘ { } |  ~
```

ISO 8859

standardisiert eine Reihe weiterer 8-Bit-Zeichensätze durch Auffüllung der freien Hälfte des ASCII-Zeichensatzes.

ISO 8859-1 (Latin1) enthält die Zusatzzeichen der meisten westeuropäischen Sprachen, wie ä (0xE4), é (0xE9), ñ (0xF1) oder æ (0xE6).

ISO 8859-15 (Latin9) ist eine kleine Modifikation von Latin1, bei der u.a. das Eurosymbol € auf Position 0xA4 eingeführt wurde.

Unicode

wird als einheitliche Kodierung aller Schriften der Welt immer wichtiger. Es können bis zu ca. 1,1 Millionen Zeichen kodiert werden, davon wird jedoch erst ein kleiner Teil, ca. 100 000 Zeichen genutzt.

Unicode umfaßt bereits alle wichtigen Schriftsprachen der Gegenwart (z.B. über 70 000 chinesisch/japanisch/koreanische Schriftzeichen, Arabisch, Hebräisch, Tamil,...); zahlreiche tote Sprachen (Runen, Keilschrift, Linear B) und Spezialzeichen aus verschiedenen Gebieten (mathematische Operatoren, Notenschrift, Pfeile,...) und wird immer noch erweitert.

Unicode selbst ist keine Kodierung vergleichbar mit ISO 8859, sondern es gibt verschiedene "Unicode transformation formats" (UTF), die einem Unicode-Zeichen eine eindeutige Sequenz von Bytes zuordnen. Die wichtigsten sind UTF-8, UTF-16 und UTF-32.

UTF-8 enthält den ASCII-Zeichensatz als Untermenge. Zeichen, die nicht zu ASCII gehören, werden durch 2 bis 4 Bytes kodiert. Damit ist dies eine Kodierung mit variabler Länge (variable length encoding) im Unterschied UTF-32 (fixed length encoding, immer 4 Byte pro Zeichen). Es ist an den ersten Bits eines jeden Bytes erkennbar, ob es Anfang oder Teil eines Mehrbyte-Zeichens ist.