Zusatzinformationen zum Thema "einfache"/"doppelte" Genauigkeit

Auf handelsüblichen PCs und Workstations wird eine einheitliche binäre Gleitkommadarstellung benutzt, die sich an einem Standard[1] des IEEE (*Institute of Electrical and Electronics Engineers*, `http://www.ieee.org`) orientiert. Die normalisierten Maschinenzahlen $M(p,m,n)$ der Vorlesung

$$\mathbb{R} \ni r = f \cdot p^e, \quad f = \begin{cases} \pm(0.b_1 \dots b_m)_p = \pm \sum_{k=1}^m b_k p^{-k} & , \ b_1 \neq 0 \\ 0 & , \ b_1 = 0 \end{cases}, \quad e = \pm(e_1, \cdots, e_n)_p = \sum_{k=1}^n e_k p^{n-k}$$

für $p = 2$ werden in den IEEE–Standardformaten `single` und `double` wie folgt mit $m + n + 1$ Bits realisiert:

- Da Zahlen $r \neq 0$ immer die Eigenschaft $b_1 \neq 0$ besitzen, braucht b_1 nicht gespeichert zu werden und wird implizit auf 1 gesetzt. Im IEEE–Standard steht allerdings der Binärpunkt *hinter* b_1, man arbeitet dort also mit der modifizierten Darstellung
$$r = \pm(b_1.b_2 \dots b_m)_2 \cdot 2^{e-1}$$
und speichert nur $b_2, \dots, b_m$ bzw. $e_1, \dots, e_n$. Dass man damit trotzdem noch die Zahl $r = 0$ realisieren kann, werden wir gleich sehen.

- Das Vorzeichen v_M der Mantisse f wird als höchstes Bit abgespeichert ($0 \leftrightarrow +$, $1 \leftrightarrow -$).

- Die nächsten $n+1$ Bits sind für den Exponenten $e-1 = -1 \pm \sum_{k=1}^n e_k 2^{n-k} \in \{-2^n, \dots, 2^n-2\}$ reserviert. Um das Vorzeichen von $e-1$ nicht behandeln zu müssen, speichert man stattdessen die natürliche Zahl $e^* = e+2^n \in \mathbb{N}$. Die speziellen Werte $e^* \in \{0, 2^{n+1}-1\}$ (nur Nullen bzw. Einsen) sind dabei besonderen Zahlzuständen vorbehalten:

e^*	Erläuterung
0	Falls auch $f = 0$, bedeutet dies $r = \pm 0$, je nach Vorzeichen v_M. Falls $f \neq 0$, so vereinbart man $r = (-1)^{v_M}(0.b_2 \dots b_m)_2 \cdot 2^{-2^n+2}$ (*nicht* normalisiert!).
$1, \dots, 2^{n+1}-2$	$r = (-1)^{v_M}(1.b_2 \dots b_m)_2 \cdot 2^{e^*-2^n+1}$
$2^{n+1}-1$	Hierzu existiert kein gültiger Exponent e. Falls $f = 0$, dann setzt man $r = \pm\infty$, je nach Vorzeichenbit v_M (Resultat von $\pm\frac{1}{0}$). Gilt $f \neq 0$, dann ist $r = \mathrm{NaN}$ (*not a number*, Resultat von $\frac{0}{0}$).

- Die letzten $m-1$ Bits speichern die Mantissenbits $b_2, \dots, b_m$.

- Bemerkung: die $m + n + 1$ Bits sind demnach im Speicher so angeordnet, dass der Computer das Prädikat "$<$" lexikographisch auswerten kann (d.h. durch Lesen von vorne nach hinten).

- Die durch das IEEE–Format darstellbaren normalisierten Zahlen reichen von $\pm 2^{-2^n+1}$ bis $\pm(2-2^{1-m})2^{2^n-1}$, zusätzlich hat man im Fall $e^* = 0$ noch die *nicht* normalisierten Zahlen $\pm 2^{-2^n-m+3}$ bis $\pm(1-2^{1-m})2^{-2^n+2}$. Zum Vergleich: die normalisierte Gleitpunktdarstellung der Vorlesung erlaubt einen Bereich von $\pm 2^{-2^n}$ bis $\pm(1-2^{-m})2^{2^n-1}$, insgesamt also nur ein geringfügiger Unterschied zwischen $M(p,m,n)$ und dem IEEE–Standard.

Konkret wurden in den IEEE–Standardformaten folgende (logische) Bitverteilungen gewählt[2]:

Zahlformat	Speicherbedarf	#Mantissenbits m	#Exponentenbits n
IEEE `single`	32 Bit	24	7
IEEE `double`	64 Bit	53	10

Bei der Implementierung der arithmetischen Grundoperationen $* \in \{+, -, \cdot, :\}$ in einer Gleitkommaeinheit (FPU) wird gemäß des IEEE–Standards z.B. durch eine erhöhte interne Genauigkeit (sogenannte *hidden bits*) zugesichert, dass der relative Fehler ε bei der Berechnung von $x \circledast y = (x * y)(1 + \varepsilon)$ unterhalb der Maschinengenauigkeit liegt, $|\varepsilon| \leq$ `eps`.

[1] ANSI/IEEE Std 754–1985, IEEE Standard for Binary Floating–Point Arithmetic, IEEE Computer Society (1985)

[2] Physikalisch korrekt (und auch in der Literatur oft so angegeben) sind eigentlich die Bitverteilungen 23:8 bei IEEE `single` und 52:11 bei IEEE `double`, da ja von den m Mantissenbits nur $m-1$ Stück wirklich gespeichert werden, andererseits aber zu den n Bits für den Exponenten noch dessen Vorzeichenbit zu zählen ist.