Binärformate reeller Zahlen

1. Vorbemerkung

Zum leichteren Erkennen von Zahlen werden verschiedene Zahlenarten im nachfolgenden Text durch unterschiedlichen Satz hervorgehoben.
Alle nichtdezimalen Zahlen sind *kursiv* gesetzt. Die Basis steht jeweils tiefgestellt am Ende der Zahl und ist immer im Dezimalsystem. Die rechnerinternen Zahlendarstellungen stehen in `Schreibmaschinenschrift`. Manche Rechner speichern Teile von Zahlen nicht explizit ab. Solche Teile sind in eckige Klammern [...] eingeschlossen. Manche Rechner benutzen bestimmte Bitgruppen in einer Maschinenzahl nicht. Solche unbenutzten Bitgruppen werden durch `O`, `L` oder `X` markiert, je nachdem, ob eine bedeutungslose Null, eine bedeutungslose Eins oder ein bedeutungsloses Zufallmuster in einer solchen Bitgruppe gespeichert wird.[1]

Zahlenart	Satz	Beispiel
Dezimalzahlen	normal	3.14159
Nichtdezimale Zahlen	*kursiv*	*3.11037 55242...*$_8$
Rechnerinterne Zahlen	`Schreibmaschinenschrift`	`0 10000 LLOOXXXX`[`1.`]`100100100`

Alle Bitmuster werden grundsätzlich als nichtnegative, ganze Zahlen betrachtet. Werte einer 10 Bit breiten Mantisse m werden also als $m = 167 = A7_{16}$ = `0010100111` geschrieben. Oft wird eine solche Mantisse als Dualbruch interpretiert. Die dann verwendeten Kurzschreibweisen $0.m = \frac{m}{2^{10}}$ und $1.m = 1 + \frac{m}{2^{10}}$ sind suggestiv und sollten unmittelbar verständlich sein.

Damit die folgenden Rechnungen leichter nachvollziehbar werden, folgt eine Darstellung der Zahl π in verschiedenen Zahlensystemen:

$$\begin{aligned}\pi &= 3.14159\,26535\,89793\,23846\,26433\,83279\,50288\,41971\ldots\\ &= \textit{3.11037 55242 10264 30215 14230 63050 56006 70163 21122...}_8\\ &= \textit{3.243F6 A8885 A308D 31319 8A2E0 37073 44A40...}_{16}\\ &= \textit{11.001 001 000 011 111 101 101 010 100 010 001 000 010 110 100}\\ &\qquad \textit{011 000 010 001 101 001 100 010 011 000 110 011 000 101 000}\\ &\qquad \textit{101 110 000 000 110 111 000 001 110 011 010 001 001 010 010...}_2\end{aligned}$$

Weitere wichtige Zahlen finden Sie in der Tabelle am Schluss des Textes.

2. Übersicht über reelle Zahlenformate

Die folgende Tabelle enthält eine Übersicht über verschiedene Rechner- und Softwareformate für reelle Zahlen und deren Speicherbedarf. Die hier eingeführten Kurzbezeichnungen werden anschließend verwendet. Das IEEE-Format[2] ist genormt und somit portabel zwischen Anlagen, die diese Norm unterstützen. Da nur zwei IEEE-Formate in der Norm definiert sind, verwenden einige Hersteller nach demselben Verfahren weitere Formate, die als IEEE-analoge Formate aufgeführt sind. Die Interpretation solcher IEEE-analoger Zahlen ist allerdings nicht mehr einheitlich. Die Norm legt lediglich die Interpretation der Bits, aber nicht ihre Anordnung im Speicher fest, so daß gelegentlich bitweise Umordnungen durchgeführt werden müssen, wenn IEEE-Binärzahlen auf eine andere Rechenanlage transportiert werden. Alle übrigen Formate werden in der Regel jeweils nur von einem Hersteller verwendet.

---

[1] Natürlich ist dieses Vorgehen nicht unbedingt sinnvoll, da damit in jeder reellen Zahl Speicherplatz verschenkt wird.

[2] sprich: ai'tripel'ie

Formatname	4 Byte	6 Byte	8 Byte	10 Byte	16 Byte	24 Byte
IEEE-Format	IEEE4		IEEE8			
(bei DEC)	S_Float		T_Float			
IEEE-analog (Sil. Gr.)					IEEE16a	
8087-Format				8087		
Cray-Format			Cray8		Cray16	Cray24
DEC-Format	F_Float		G_Float		H_Float	
			D_Float			
Comparex-Format	IBM4		IBM8			
Turbo-Pascal-Format		TP6				

3. Beschreibung der reellen Zahlformate

Jedes reelle Zahlenformat definiert eine endliche Teilmenge der überabzählbar großen Menge $\mathbb{R}$ der reellen Zahlen. Diese Teilmenge kann für die meisten Zwecke hinreichend genau durch drei Schlüsselwerte beschrieben werden. Damit ist z.B. eine Beurteilung der Lösbarkeit eines bestimmten numerischen Problems möglich. Diese Werte sind der Umfang des Zahlenbereiches (`minposreal` und `maxreal`) und die relative Genauigkeit (ε) einer Zahl.

Wert	Beschreibung
`maxreal`	größte gerade noch darstellbare reelle Zahl
`minposreal`	kleinste positive reelle Zahl (kleinste reelle Zahl, die gerade größer als Null ist)
ε	kleinste positive reelle Zahl mit der Bedingung $1 + \varepsilon > 1$ (in Programmen `epsilon`)

Etwas präziser ist die Beschreibung eines Zahlformates durch die fünf Werte r, p, *emin*, *emax* und *denorm* nach der ISO-Norm *Language Independent Arithmetic*. Dabei ist

Wert	Beschreibung
$r \in \mathbb{Z}$	Wurzel des Zahlformates
$p \in \mathbb{Z}$	Stellenanzahl in der Mantisse
$emin \in \mathbb{Z}$	Kleinster Wert des Exponenten
$emax \in \mathbb{Z}$	Größter Wert des Exponenten
$denorm \in$ boolean	Existenz unnormalisierter Zahlen

Mit diesen Werten werden die endlichen Mengen F_N, F_D und F der verfügbaren normalisierten Zahlen, der nichtnormalisierten Zahlen und aller Zahlen definiert:

$$F_N = \{0, \pm i \cdot r^{e-p} | i, e \in \mathbb{Z}, r^{p-1} \leq i \leq r^p - 1, emin \leq e \leq emax\}$$

$$F_D = \{\pm i \cdot r^{e-p} | i, e \in \mathbb{Z}, 1 \leq i \leq r^{p-1} - 1, e = emin\}$$

$$F = \begin{cases} F_N \cup F_D & \text{falls } denorm = \text{true} \\ F_N & \text{falls } denorm = \text{false} \end{cases}$$

Durch diese acht Werte lassen sich die Zahlenformate grob und exakt charakterisieren.

Format	Breite		Grobcharakterisierung			Feincharakterisierung				
	Bit	Byte	minposreal	maxreal	ε	r	p	*emin*	*emax*	*denorm*
IEEE4	32	4	$1,175 \cdot 10^{-38}$	$3,403 \cdot 10^{38}$	$1,192 \cdot 10^{-7}$	2	24	-125	128	true
F_Float	32	4	$2,939 \cdot 10^{-39}$	$1,701 \cdot 10^{38}$	$1,192 \cdot 10^{-7}$	2	24	-127	127	false
IBM4	32	4	$8,636 \cdot 10^{-78}$	$7,237 \cdot 10^{75}$	$5,960 \cdot 10^{-8}$	16	6	-64	63	false
TP6	48	6	$2,939 \cdot 10^{-39}$	$1,701 \cdot 10^{38}$	$1,819 \cdot 10^{-12}$	2	40	-127	127	false
IEEE8	64	8	$2,225 \cdot 10^{-308}$	$1,798 \cdot 10^{308}$	$2,220 \cdot 10^{-16}$	2	53	-1021	1024	true
Cray8	64	8	$3,667 \cdot 10^{-2466}$	$2,727 \cdot 10^{2465}$	$7,105 \cdot 10^{-15}$	2	48	-8189	8190	true
G_Float	64	8	$5,563 \cdot 10^{-309}$	$8,988 \cdot 10^{307}$	$2,220 \cdot 10^{-16}$	2	53	-1023	1023	false
D_Float	64	8	$2,939 \cdot 10^{-39}$	$1,701 \cdot 10^{38}$	$2,776 \cdot 10^{-17}$	2	56	-127	127	false
IBM8	64	8	$8,636 \cdot 10^{-78}$	$7,237 \cdot 10^{75}$	$1,388 \cdot 10^{-17}$	16	14	-64	63	false
8087	80	10	$1,681 \cdot 10^{-4932}$	$1,190 \cdot 10^{4932}$	$1,084 \cdot 10^{-19}$	2	64	-16382	16384	true
IEEE16a	128	16	$3,362 \cdot 10^{-4932}$	$1,190 \cdot 10^{4932}$	$1,926 \cdot 10^{-34}$	2	113	-16381	16384	true
Cray16	128	16	$3,667 \cdot 10^{-2466}$	$2,727 \cdot 10^{2465}$	$2,524 \cdot 10^{-29}$	2	96	-8189	8190	true
H_Float	128	16	$8,405 \cdot 10^{-4933}$	$5,949 \cdot 10^{4931}$	$1,926 \cdot 10^{-34}$	2	113	-16383	16383	false
Cray24	192	24	$3,667 \cdot 10^{-2466}$	$2,727 \cdot 10^{2465}$	$8,968 \cdot 10^{-44}$	2	144	-8189	8190	true

4. Verfügbarkeit in Fortran

In der folgenden Tabelle finden Sie Angaben, welcher Compiler unter welchem Namen welches Format unterstützt. Um Platz zu sparen, werden die Fortran-Bezeichungen REAL, DOUBLE PRECISION, REAL*4, REAL*8 u.a. mit R, DP, R*4, R*8 usw. abgekürzt.

Rechner/Compiler	4 Byte		8 Byte		16 Byte	
PC/Prospero	R	(IEEE4)	DP	(IEEE8)		
PC/WATCOM	R, R*4	(IEEE4)	DP, R*8	(IEEE8)		
SunSparc/Unix/f77	R, R*4	(IEEE4)	DP, R*8	(IEEE8)	R*16	(IEEE16a)
Comparex/VM	R	(IBM4)	DP	(IBM8)		
Indigo, Indy, Challenge	R, R*4	(IEEE4)	DP, R*8, R*16	(IEEE8)		
Cray/Unicos			R	(Cray8)	DP	(Cray16)

5. Verfügbarkeit in C

In der folgenden Tabelle finden Sie Angaben, welcher Compiler unter welchem Namen welches Format unterstützt. Um Platz zu sparen, werden die C-Bezeichnungen *float*, *double* und *long double* mit fl, db und ld abgekürzt.

Rechner/Compiler	4 Byte		8 Byte		10 Byte		16 Byte	
PC/Borland	fl	IEEE4	db	IEEE8	ld	8087		
PC/Watcom	fl	IEEE4	db	IEEE8				
PC/Symantec	fl	IEEE4	db, ld	IEEE8				
Alpha/Unix/cc	fl	IEEE4	db, ld	IEEE8				
SunSparc/Unix/cc	fl	IEEE4	db	IEEE8				
SunSparc/Unix/acc	fl	IEEE4	db	IEEE8			ld	(IEEE16a)
SunSparc/Unix/CC	fl	IEEE4	db, ld	IEEE8				
Comparex/VM	fl	IBM4	db	IBM8				
Indy, Indigo, Challenge	fl	IEEE4	db, ld	IEEE8				
Cray/Unicos			fl, db	Cray8			ld	Cray16

6. Verfügbarkeit in Pascal

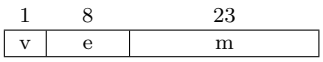
In der folgenden Tabelle finden Sie Angaben, welcher Compiler unter welchem Namen welches Format unterstützt.

Rechner/Compiler	4 Byte	6 Byte	8 Byte	10 Byte
PC/Turbo 3.0		real (TP6)		
PC/Turbo-87 3.0			real (IEEE8)	
PC/Turbo ab 4.0	single (IEEE4)	real (TP6)	double (IEEE8)	extended (8087)
PC/Turbo für Windows	single (IEEE4)	real (TP6)	double (IEEE8)	extended (8087)
PC/Prospero	real (IEEE4)		longreal (IEEE8)	
SunSparc/Unix	real (-xl) shortreal (IEEE4) single		real (kein-xl) longreal (IEEE8) double	
Comparex/VM	shortreal (IBM4)		real (IBM8)	
Indy, Indigo, Challenge	real (IEEE4)		double (IEEE8)	
Cray/Unicos			real (Cray8)	

7. Das IEEE-Standardformat mit 4 Bytes

Das IEEE[3]-Format ist genormt und somit portabel zwischen Rechnern, die dieses Format verwenden. Derzeit sind das bei uns die PCs, die Unix-Rechner Sun und Indy und Alpha-Chip-Rechner.

7.1. Bitverteilung



7.2. Interpretation der Bits

$$
\begin{aligned}
0 < e < 255 &\implies x = (-1)^v \cdot 1.m \cdot 2^{e-127} && \text{Normale reelle Zahl}\\
e = 0, m = 0 &\implies x = 0 && \text{Null}\\
e = 0, m \neq 0 &\implies x = (-1)^v \cdot 0.m \cdot 2^{1-127} && \text{Nichtnormalisierte Zahl}\\
e = 255, m = 0 &\implies x = (-1)^v \cdot \infty && \text{Unendlich}\\
e = 255, m \neq 0 &\implies x = \text{NAN} && \text{„Nicht-Zahl“, not a number}
\end{aligned}
$$

7.3. Schlüsselwerte

$$\text{minposreal} = 1,175 \cdot 10^{-38} \quad \text{maxreal} = 3,403 \cdot 10^{38} \quad \varepsilon = 1,192 \cdot 10^{-7}$$

[3] sprich: ai'tripel'ie

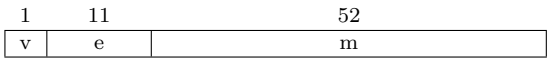
7.4. Beispielzahlen

$$
\begin{aligned}
0 &= \texttt{0 00000000 [ ]00000000000000000000000}\\
\texttt{maxreal} &= \texttt{0 11111110 [1.]11111111111111111111111}\\
&= +\,\textit{1.11111111111111111111111}_2 \cdot 2^{254-127}\\
&= +\,(2.0 - 2^{-23}) \cdot 2^{127}\\
&\approx +\,2.0 \cdot 2^{127}\\
&= 2^{128}\\
&= 3.402823\ldots \cdot 10^{38}\\
\texttt{minposreal} &= \texttt{0 00000001 [1.]00000000000000000000000}\\
&= +\,\textit{1.00000000000000000000000}_2 \cdot 2^{1-127}\\
&= +\,1.0 \cdot 2^{-126}\\
&= 2^{-126}\\
&= 1.175494\ldots \cdot 10^{-38}\\
1 &= \texttt{0 01111111 [1.]00000000000000000000000}\\
1+\varepsilon &= \texttt{0 01111111 [1.]00000000000000000000001}\\
\varepsilon &= +\,\textit{0.00000000000000000000001}_2\\
&= 2^{-23}\\
&= 1.192092 \cdot 10^{-7}\\
\pi &= \textit{11.00100 10000 11111 10110 10101}\ldots_2\\
&= \textit{1.10010010000111111011010101}\ldots_2 \cdot 2^1\\
&\approx \textit{1.10010010000111111011011}_2 \cdot 2^{128-127}\\
&= \texttt{0 10000000 [1.]10010010000111111011011}
\end{aligned}
$$

8. Das IEEE-Standardformat mit 8 Bytes

Das IEEE-Format ist genormt und somit portabel zwischen Rechnern, die dieses Format verwenden. Derzeit sind das bei uns die PCs, die Unix-Rechner Sun und Indy und Alpha-Chip-Rechner.

8.1. Bitverteilung



8.2. Interpretation der Bits

$0 < e < 2047$	$\Longrightarrow$	$x = (-1)^v \cdot 1.m \cdot 2^{e-1023}$	Normale reelle Zahl
$e = 0, m = 0$	$\Longrightarrow$	$x = 0$	Null
$e = 0, m \neq 0$	$\Longrightarrow$	$x = (-1)^v \cdot 0.m \cdot 2^{1-1023}$	Nichtnormalisierte Zahl
$e = 2047, m = 0$	$\Longrightarrow$	$x = (-1)^v \cdot \infty$	Unendlich
$e = 2047, m \neq 0$	$\Longrightarrow$	$x = \mathrm{NAN}$	„Nicht-Zahl“, not a number

Die Zahlen mit eingeschränkter Genauigkeit bedeuten – wie schon der Name sagt – einen Genauigkeitsverlust und sollten daher für normale numerische Berechnungen vermieden werden.

8.3. Schlüsselwert

$$\mathrm{minposreal} = 2,225 \cdot 10^{-308} \quad \mathrm{maxreal} = 1,798 \cdot 10^{308} \quad \varepsilon = 2,220 \cdot 10^{-16}$$

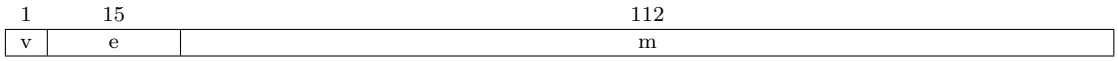
8.4. Beispielzahlen

$$
\begin{aligned}
0 &= \texttt{0 00000000000 [ ]0000000000000000000000000000000000000000000000000000} \\
\texttt{maxreal} &= \texttt{0 11111111110 [1.]1111111111111111111111111111111111111111111111111111} \\
&= +\,\textit{1.1111111111111111111111111111111111111111111111111111}_2 \cdot 2^{2046-1023} \\
&= +\,(2.0 - 2^{-52}) \cdot 2^{1023} \\
&\approx +\,2.0 \cdot 2^{1023} \\
&= 2^{1024} \\
&= 1.79769\,31348\,62315\,9\ldots \cdot 10^{308} \\
\texttt{minposreal} &= \texttt{0 00000000001 [1.]0000000000000000000000000000000000000000000000000000} \\
&= +\,\textit{1.0000000000000000000000000000000000000000000000000000}_2 \cdot 2^{1-1023} \\
&= +\,1.0 \cdot 2^{-1022} \\
&= 2^{-1022} \\
&= 2.22507\,38585\,07201\ldots \cdot 10^{-308} \\
1 &= \texttt{0 01111111111 [1.]0000000000000000000000000000000000000000000000000000} \\
1+\varepsilon &= \texttt{0 01111111111 [1.]0000000000000000000000000000000000000000000000000001} \\
\varepsilon &= +\,\textit{0.0000000000000000000000000000000000000000000000000001}_{\ 2} \\
&= 2^{-52} \\
&= 2.22044\,60492\,50313\ldots \cdot 10^{-16} \\
\pi &= \textit{11.00100 10000 11111 10110 10101 00010 00100 00101 10100 01100 00100}\ldots_2 \\
&= \textit{1.00100100001111110110101010001000100001011010001100001000}\ldots_2 \cdot 2^1 \\
&\approx \textit{1.00100100001111110110101010001000100001011010001100}_2 \cdot 2^{1024-1023} \\
&= \texttt{0 10000000000 [1.]1001001000011111101101010100010001000010110100011000}
\end{aligned}
$$

9. Das IEEE-analoge Format mit 16 Bytes

Das Format ist nicht genormt, sondern nur analog zur IEEE-Norm definiert. Verschiedene Definitionen unterschiedlicher Hersteller müssen deshalb nicht übereinstimmen.

9.1. Bitverteilung



9.2. Interpretation der Bits

$0 < e < 32767$	$\Longrightarrow$	$x = (-1)^v \cdot 1.m \cdot 2^{e-16383}$	Normale reelle Zahl
$e = 0, m = 0$	$\Longrightarrow$	$x = 0$	Null
$e = 0, m \neq 0$	$\Longrightarrow$	$x = (-1)^v \cdot 0.m \cdot 2^{1-16383}$	Reelle Zahl mit eingeschränkter Genauigkeit
$e = 32767, m = 0$	$\Longrightarrow$	$x = (-1)^v \cdot \infty$	Unendlich
$e = 32767, m \neq 0$	$\Longrightarrow$	$x = \mathrm{NAN}$	„Nicht-Zahl“, not a number

Die Zahlen mit eingeschränkter Genauigkeit bedeuten – wie schon der Name sagt – einen Genauigkeitsverlust und sollten daher für normale numerische Berechnungen vermieden werden.

9.3. Schlüsselwerte

$$\mathrm{minposreal} = 3,362 \cdot 10^{-4932} \quad \mathrm{maxreal} = 1,190 \cdot 10^{4932} \quad \varepsilon = 1,926 \cdot 10^{-34}$$

9.4. Beispielzahlen

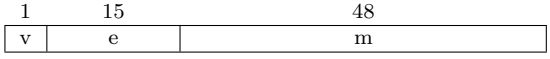
Wegen der Länge werden die Bitmuster hier nicht mehr ausgeschrieben.

$$
\begin{aligned}
0 &= \texttt{0 000000000000000 [ ]0... 0} \\
\texttt{maxreal} &= \texttt{0 111111111111110 [1.]1... 1} \\
&= +\, \mathit{1.1...1}_2 \cdot 2^{32766-16383} \\
&= +\, (2.0 - 2^{-112}) \cdot 2^{16383} \\
&\approx +\, 2.0 \cdot 2^{16383} \\
&= 2^{16384} \\
&= 1.18973\,14953\,57231\,76508\,57593\,26628\,007\ldots \cdot 10^{4932} \\
\texttt{minposreal} &= \texttt{0 000000000000001 [1.]0... 0} \\
&= +\, \mathit{1.0...0}_2 \cdot 2^{1-16383} \\
&= +\, 1.0 \cdot 2^{-16382} \\
&= 2^{-16382} \\
&= 3.36210\,31431\,12093\,50626\,26778\,17321\,7526\ldots \cdot 10^{-4932} \\
1 &= \texttt{0 011111111111111 [1.]0...0} \\
1+\varepsilon &= \texttt{0 011111111111111 [1.]0...01} \\
\varepsilon &= +\, \mathit{0.0...01}_2 \\
&= 2^{-112} \\
&= 1.92592\,99443\,87235\,85305\,59779\,42584\,927\ldots \cdot 10^{-34} \\
\pi &= \mathit{11.00100\ 10000\ldots}_2 \\
&= \mathit{1.10010010000\ldots}_2 \cdot 2^1 \\
&\approx \mathit{1.10010010000}_2 \cdot 2^{1024-1023} \\
&= \texttt{0 100000000000000 [1.]} \\
&\quad \texttt{10010010000111111011010101000100010000101101000110000100} \\
&\quad \texttt{01101001100010011000110011000101000101110000000110111000}
\end{aligned}
$$

10. Das Cray-Format mit 8 Bytes

Dieses Format hat einen ausreichenden Wertebereich für die meisten Anwendungen. Der theoretisch mögliche Exponentenbereich wird – aus mir nicht bekannten Gründen – nicht vollständig ausgenutzt, so daß die maximal mögliche Anzahl reeller Zahlen bei diesem Format nicht benutzt wird.

10.1. Bitverteilung



10.2. Interpretation der Bits

$$
\left.\begin{array}{l} \mathit{2000340000000000000001}_8 \le (e,m) \\ (e,m) \le \mathit{577767777777777777776}_8 \end{array}\right\} \quad \Longrightarrow \quad x = (-1)^v \cdot 0.m \cdot 2^{e-16384} \qquad \text{Normale reelle Zahl}
$$

$$
v = 0, e = 0, m = 0 \quad \Longrightarrow \quad x = 0 \qquad \text{Null}
$$

Alle anderen Bitmuster sind verboten und führen zu Hardware- oder Numerikfehlern oder falschen Resultaten.
Die Bedingung für normale reelle Zahlen kann auch für Mantisse und Exponent getrennt geschrieben werden:

$8196 \le e \le 24573, m \ge 2^{47}$	$\Longrightarrow$	$x = (-1)^v \cdot 0.m \cdot 2^{e-16384}$	Reelle Z. im mittleren Bereich
$e = 8195, m \ge 2^{47} + 1$	$\Longrightarrow$	$x = (-1)^v \cdot 0.m \cdot 2^{e-16384}$	Reelle Z. im unteren Grenzb.
$e = 24574, 2^{47} \le m \le 2^{48} - 2$	$\Longrightarrow$	$x = (-1)^v \cdot 0.m \cdot 2^{e-16384}$	Reele Z. im oberen Grenzb.
$v = 0, e = 0, m = 0$	$\Longrightarrow$	$x = 0$	Null

10.3. Schlüsselwerte

$$\text{minposreal} = 3,667 \cdot 10^{-2466} \quad \text{maxreal} = 2,727 \cdot 10^{2465} \quad \varepsilon = 7,105 \cdot 10^{-15}$$

10.4. Beispielzahlen

$$
\begin{aligned}
0 =& \texttt{0 000000000000000 0000000000000000000000000000000000000000000000000000}\\
\texttt{maxreal} =& \texttt{0 101111111111110 1111111111111111111111111111111111111111111111111110}\\
=& + \textit{0.1111111111111111111111111111111111111111111111111110}_{\,2} \cdot 2^{24574-16384}\\
=& + (1.0 - 2^{-47}) \cdot 2^{8190}\\
\approx& + 1.0 \cdot 2^{8190}\\
=& 2^{8190}\\
=& 2,726870339048539\ldots \cdot 10^{2465}\\
\texttt{minposreal} =& \texttt{0 010000000000011 1000000000000000000000000000000000000000000000000001}\\
=& + \textit{0.1000000000000000000000000000000000000000000000000001}_{\,2} \cdot 2^{8195-16384}\\
=& + (2^{-1} + 2^{-48}) \cdot 2^{-8189}\\
\approx& + 2^{-1} \cdot 2^{-8189}\\
=& 2^{-8190}\\
=& 3,667207735109694\ldots \cdot 10^{-2466}\\
1 =& \texttt{0 100000000000001 1000000000000000000000000000000000000000000000000000}\\
1 + \varepsilon =& \texttt{0 100000000000001 1000000000000000000000000000000000000000000000000001}\\
\varepsilon =& \texttt{0 100000000000001 0000000000000000000000000000000000000000000000000001}\\
=& \textit{0.0000000000000000000000000000000000000000000000000001}_{\,2} \cdot 2^{1}\\
=& 2^{-48} \cdot 2^{1}\\
=& 2^{-47}\\
=& 7,105427357601001 \cdot 10^{-15}\\
\pi =& \textit{11.00100 10000 11111 10110 10101 00010 00100 00101 10100 01100}\ldots_{\,2}\\
=& \textit{0.11001 00100 00111 11101 10101 01000 10001 00001 01101 00011 00}\ldots_{\,2} \cdot 2^{2}\\
\approx& \textit{0.11001 00100 00111 11101 10101 01000 10001 00001 01101 001}\ldots_{\,2} \cdot 2^{2}\\
=& \textit{0.11001 00100 00111 11101 10101 01000 10001 00001 01101 001}\ldots_{\,2} \cdot 2^{16386-16384}\\
=& \texttt{0 100000000000010 1100100100001111110110101010001000100000101101001}
\end{aligned}
$$

11. Das Cray-Format mit 16 Bytes

Dieses Format enthält in einem Nachfolgewort eine 48 Bit lange Erweiterung der Mantisse. Die führenden 16 Bit dieses zweiten Wortes sind unbenutzt und müssen Nullen enthalten. Das Format wird durch eine reine Softwarearithmetik implementiert. Es steht in C (long double) und Fortran (DOUBLE PRECISION) direkt zur Verfügung. Es kann weiter mit Hilfe der Prozeduren DASS, DASV, DAVS, DAVV, DDSS, DDSV, DDVS, DDVV, DMSS, DMSV, DMVS, DMVV, DSSS, DSSV, DSVS und DSVV aus der *math library* benutzt werden.
Es ist nicht vektorisierbar (?), kann aber vorteilhaft z.B. bei der Berechnung von Skalarprodukten verwendet werden.

11.1. Bitverteilung

1	15	48	1	15	48
v	e	m1	0	0	m2

Die beiden Mantissenfelder $m1$ und $m2$ werden einfach aneinandergehängt ($m = 0.m1m2 = \frac{m1}{2^{48}} + \frac{m2}{2^{96}}$). Alle Regeln des Cray-Formates mit 8-Byte bleiben erhalten.

11.2. Schlüsselwerte

$$\begin{aligned}
\text{minposreal} &= 3,667 \cdot 10^{-2466} = 3,66720\,77351\,09694\,33124\,28247\,84096 \cdot 10^{-2466} \\
\text{maxreal} &= 2,727 \cdot 10^{2465} = 2,72687\,03390\,48539\,82365\,74606\,11834 \cdot 10^{2465} \\
\varepsilon = 2^{-95} &= 2,524 \cdot 10^{-29} = 2,52435\,48967\,07237\,77731\,75314\,08904\,9 \cdot 10^{-29}
\end{aligned}$$

12. Das Cray-Format mit 24 Bytes (Tripleformat)

Dieses Format enthält in zwei Nachfolgeworten eine 96 Bit lange Erweiterung der Mantisse. Die führenden 16 Bit dieser beiden Worte sind unbenutzt und müssen Nullen enthalten. Das Format wird durch eine reine Softwarearithmetik implementiert. Es steht nur mit den Prozeduren TADD, TASS, TDIV, TDSS, TMLT, TMSS, TSUB und TSSS aus der *math library* zur Verfügung.

12.1. Bitverteilung

1	15	48
v	e	m1
1	15	48
0	0	m2
1	15	48
0	0	m3

Die Mantissenfelder $m1$, $m2$ und $m3$ werden einfach aneinandergehängt ($m = 0.m1m2m3 = \frac{m1}{2^{48}} + \frac{m2}{2^{96}} + \frac{m3}{2^{144}}$). Alle Regeln des Cray-Formates mit 8-Byte bleiben erhalten.

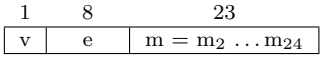
12.2. Schlüsselwerte

$$\begin{aligned}
\text{minposreal} &= 3,667 \cdot 10^{-2466} = 3,66720\,77351\,09694\,33124\,28247\,84096\,96633\,19127\,29942 \cdot 10^{-2466} \\
\text{maxreal} &= 2,727 \cdot 10^{2465} = 2,72687\,03390\,48539\,82365\,74606\,11834\,45715\,61206\,60404 \cdot 10^{2465} \\
\varepsilon = 2^{-143} &= 8,968 \cdot 10^{-44} = 8,96831\,01716\,78829\,25391\,18693\,33055\,46324\,01936\,76428 \cdot 10^{-44}
\end{aligned}$$

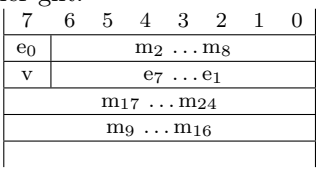
13. Das DEC-Format F_Float

13.1. Bitverteilung

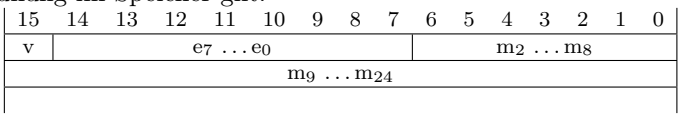
Das Zahlenformat ist dem IEEE-Format ähnlich:



Bei byteweiser Anordnung im Speicher gilt:



Bei wortweiser Anordnung im Speicher gilt:



13.2. Interpretation der Bits

$0 < e \leq 255$	$\Longrightarrow$	$x = (-1)^v \cdot 0.1m \cdot 2^{e-128}$	Normale reelle Zahl
$v = 0, e = 0$	$\Longrightarrow$	$x = 0$	Null
$v = 1, e = 0$	$\Longrightarrow$	$x =$ reservierte Zahl	„Nicht-Zahl“, not a number

13.3. Schlüsselwerte

$$\text{minposreal} = 2,939 \cdot 10^{-39} \quad \text{maxreal} = 1,701 \cdot 10^{38} \quad \varepsilon = 1,192 \cdot 10^{-7}$$

13.4. Beispielzahlen

$$\begin{aligned}
0 =& \texttt{0 00000000 XXXXXXXXXXXXXXXXXXXXXXX}\\
\texttt{maxreal} =& \texttt{0 11111111 [0.1]11111111111111111111111}\\
=& + \textit{0.11111111111111111111111}_2 \cdot 2^{255-128}\\
=& + (1.0 - 2^{-24}) \cdot 2^{127}\\
\approx& + 1.0 \cdot 2^{127}\\
=& 2^{127}\\
=& 1,70141\,18\ldots \cdot 10^{38}\\
\texttt{minposreal} =& \texttt{0 00000001 [0.1]00000000000000000000000}\\
=& + \textit{0.10000000000000000000000}_2 \cdot 2^{1-128}\\
=& + 0.1_2 \cdot 2^{-127}\\
=& 2^{-128}\\
=& 2,93873\,58\ldots \cdot 10^{-39}\\
1 =& \texttt{0 10000001 [0.1]00000000000000000000000}\\
1 + \varepsilon =& \texttt{0 10000001 [0.1]00000000000000000000001}\\
\varepsilon =& + \textit{0.00000000000000000000001}_2 \cdot 2^{1}\\
=& 2^{-24} \cdot 2^{1}\\
=& 2^{-23}\\
=& 1,19209\,28 \cdot 10^{-7}\\
\pi =& \textit{11.00100 10000 11111 10110 10101}\ldots_2\\
=& \textit{0.11001001000011111011010101}\ldots_2 \cdot 2^{2}\\
\approx& \textit{0.11001001000011111011011}_2 \cdot 2^{130-128}\\
=& \texttt{0 10000010 [1.]10010010000111111011011}
\end{aligned}$$

14. Das DEC-Format G_Float

14.1. Bitverteilung

Das Zahlenformat ist dem IEEE-Format ähnlich:

1	11	52
v	e	$m = m_2 \ldots m_{53}$

Bei byteweiser Anordnung im Speicher gilt:

7 6 5 4	3 2 1 0
$e_3 \ldots e_0$	$m_2 \ldots m_5$
v	$e_1 0 \ldots e_4$
$m_{14} \ldots m_{21}$	
$m_6 \ldots m_{13}$	
$m_{30} \ldots m_{37}$	
$m_{22} \ldots m_{29}$	
$m_{46} \ldots m_{53}$	
$m_{38} \ldots m_{45}$	

Bei wortweiser Anordnung im Speicher gilt:

15 14 13 12 11 10 9 8 7 6 5 4	3 2 1 0
v	$e_{10} \ldots e_0$
$m_6 \ldots m_{21}$	
$m_{22} \ldots m_{37}$	
$m_{38} \ldots m_{53}$	

14.2. Interpretation der Bits

$0 < e \leq 2047$	$\Longrightarrow$	$x = (-1)^v \cdot 0.1m \cdot 2^{e-1024}$	Normale reelle Zahl
$v = 0, e = 0$	$\Longrightarrow$	$x = 0$	Null
$v = 1, e = 0$	$\Longrightarrow$	$x =$ reservierte Zahl	„Nicht-Zahl“, not a number

14.3. Schlüsselwerte

$$\text{minposreal} = 5,563 \cdot 10^{-309} \quad \text{maxreal} = 8,988 \cdot 10^{307} \quad \varepsilon = 2,220 \cdot 10^{-16}$$

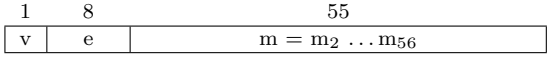
14.4. Beispielzahlen

$$
\begin{aligned}
0 &= \texttt{0 00000000000 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX} \\
\texttt{maxreal} &= \texttt{0 11111111111 [0.1]11111111111111111111111111111111111111111111111111} \\
&= +\,\textit{0.1111111111111111111111111111111111111111111111111111}_2 \cdot 2^{2047-1024} \\
&= +\,(1.0 - 2^{-53}) \cdot 2^{1023} \\
&\approx +\,1.0 \cdot 2^{1023} \\
&= 2^{1023} \\
&= 8,\,98846\,56743\,11579\,5\ldots \cdot 10^{307} \\
\texttt{minposreal} &= \texttt{0 00000000001 [0.1]0000000000000000000000000000000000000000000000000000} \\
&= +\,\textit{0.1000000000000000000000000000000000000000000000000000}_2 \cdot 2^{1-1024} \\
&= +\,0.1_2 \cdot 2^{-1023} \\
&= 2^{-1024} \\
&= 5,\,56268\,46462\,68003\ldots \cdot 10^{-309} \\
1 &= \texttt{0 10000000001 [0.1]0000000000000000000000000000000000000000000000000000} \\
1+\varepsilon &= \texttt{0 10000000001 [0.1]0000000000000000000000000000000000000000000000000001} \\
\varepsilon &= +\,\textit{0.0000000000000000000000000000000000000000000000000001}_2 \cdot 2^{1} \\
&= 2^{-53} \cdot 2^{1} \\
&= 2^{-52} \\
&= 2,\,22044\,60492\,50313\ldots \cdot 10^{-16} \\
\pi &= \textit{11.00100 10000 11111 10110 10101 00010 00100 00101 10100 01100 00100}\ldots_2 \\
&= \textit{0.11001001000011111101101010100010001000010110100011000100}\ldots_2 \cdot 2^{2} \\
&\approx \textit{0.11001001000011111101101010100010001000010110100011000}_2 \cdot 2^{1026-1024} \\
&= \texttt{0 10000000010 [0.1]10010010000111111011010101000100010000101101000110000}
\end{aligned}
$$

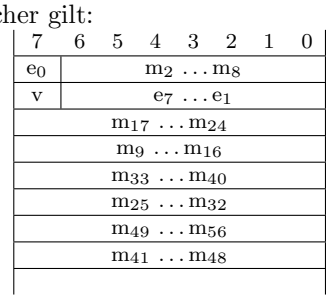
15. Das DEC-Format D_Float

15.1. Bitverteilung

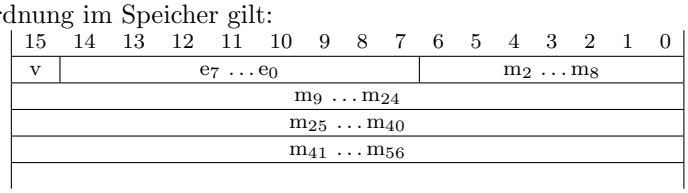
Das Zahlenformat hat gleichen Wertebereich wie F_Float und größere Genauigkeit als G_Float.



Bei byteweiser Anordnung im Speicher gilt:



Bei wortweiser Anordnung im Speicher gilt:



15.2. Interpretation der Bits

$0 < e \leq 255$	$\Longrightarrow$	$x = (-1)^v \cdot 0.1m \cdot 2^{e-128}$	Normale reelle Zahl
$v = 0, e = 0$	$\Longrightarrow$	$x = 0$	Null
$v = 1, e = 0$	$\Longrightarrow$	$x =$ reservierte Zahl	„Nicht-Zahl“, not a number

15.3. Schlüsselwerte

$$\text{minposreal} = 2,939 \cdot 10^{-39} \quad \text{maxreal} = 1,701 \cdot 10^{38} \quad \varepsilon = 2,776 \cdot 10^{-17}$$

15.4. Beispielzahlen

$$
\begin{aligned}
0 =& \texttt{0 00000000 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX} \\
\texttt{maxreal} =& \texttt{0 11111111 [0.1]11111111111111111111111111111111111111111111111111111111} \\
=& + \textit{0.11111111111111111111111111111111111111111111111111111111}_2 \cdot 2^{255-128} \\
=& + (1.0 - 2^{-56}) \cdot 2^{127} \\
\approx& + 1.0 \cdot 2^{127} \\
=& 2^{127} \\
=& 1,70141\,18346\,04692\,3\ldots \cdot 10^{38} \\
\texttt{minposreal} =& \texttt{0 00000001 [0.1]00000000000000000000000000000000000000000000000000000000} \\
=& + \textit{0.10000000000000000000000000000000000000000000000000000000}_2 \cdot 2^{1-128} \\
=& + 0.1_2 \cdot 2^{-127} \\
=& 2^{-128} \\
=& 2,93873\,58770\,55718\,77\ldots \cdot 10^{-39} \\
1 =& \texttt{0 10000001 [0.1]00000000000000000000000000000000000000000000000000000000} \\
1 + \varepsilon =& \texttt{0 10000001 [0.1]00000000000000000000000000000000000000000000000000000001} \\
\varepsilon =& + \textit{0.00000000000000000000000000000000000000000000000000000001}_2 \cdot 2^1 \\
=& 2^{-56} \cdot 2^1 \\
=& 2^{-55} \\
=& 2,77555\,75615\,62891\,35\ldots \cdot 10^{-17} \\
\pi =& \textit{11.00100 10000 11111 10110 10101 00010 00100 00101 10100 01100 00100 01}\ldots_2 \\
=& \textit{0.11001001000011111101101010100010001000010110100011000010001}\ldots_2 \cdot 2^2 \\
\approx& \textit{0.11001001000011111101101010100010001000010110100011000010}_2 \cdot 2^{130-128} \\
=& \texttt{0 10000010 [0.1]10010010000111111011010101000100010000101101000110000010}
\end{aligned}
$$

16. Das DEC-Format H_Float

16.1. Bitverteilung

1	15	112
v	e	$m = m_2 \ldots m_{113}$

Bei byteweiser Anordnung im Speicher gilt:

7	6	5	4	3	2	1	0
$e_7 \ldots e_0$							
v	$e_{14} \ldots e_8$						
$m_{10} \ldots m_{17}$							
$m_2 \ldots m_9$							
$m_{26} \ldots m_{33}$							
$m_{18} \ldots m_{25}$							
$m_{42} \ldots m_{49}$							
$m_{34} \ldots m_{41}$							
$m_{58} \ldots m_{65}$							
$m_{50} \ldots m_{57}$							
$m_{74} \ldots m_{81}$							
$m_{66} \ldots m_{73}$							
$m_{90} \ldots m_{97}$							
$m_{82} \ldots m_{89}$							
$m_{106} \ldots m_{113}$							
$m_{98} \ldots m_{105}$							

Bei wortweiser Anordnung im Speicher gilt:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
v	$e_{14} \ldots e_0$														
$m_2 \ldots m_{17}$															
$m_{18} \ldots m_{33}$															
$m_{34} \ldots m_{49}$															
$m_{50} \ldots m_{65}$															
$m_{66} \ldots m_{81}$															
$m_{82} \ldots m_{97}$															
$m_{98} \ldots m_{113}$															

16.2. Interpretation der Bits

$$
\begin{array}{rcll}
0 < e \leq 32767 & \Longrightarrow & x = (-1)^v \cdot 0.1m \cdot 2^{e-16384} & \text{Normale reelle Zahl} \\
v = 0, e = 0 & \Longrightarrow & x = 0 & \text{Null} \\
v = 1, e = 0 & \Longrightarrow & x = \text{reservierte Zahl} & \text{„Nicht-Zahl“, not a number}
\end{array}
$$

16.3. Schlüsselwerte

$$\text{minposreal} = 8,405 \cdot 10^{-4933} \quad \text{maxreal} = 5,949 \cdot 10^{4931} \quad \varepsilon = 1,926 \cdot 10^{-34}$$

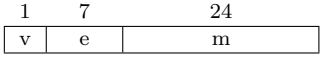
16.4. Beispielzahlen

$$
\begin{aligned}
0 &= \texttt{0 000000000000000 X...X} \\
\texttt{maxreal} &= \texttt{0 111111111111111 [0.1]1...1} \\
&= +\,0.1\ldots1_2 \cdot 2^{32767-16384} \\
&= +\,(1,0-2^{-113}) \cdot 2^{16383} \\
&\approx +\,1.0 \cdot 2^{16383} \\
&= 2^{16383} \\
&= 5,94865\,74767\,86158\,82542\,87966\,33140\,03565\ldots \cdot 10^{4931} \\
\texttt{minposreal} &= \texttt{0 000000000000001 [0.1]0...0} \\
&= +\,0.1_2 \cdot 2^{1-16384} \\
&= +\,0.1_2 \cdot 2^{-16383} \\
&= 2^{-16384} \\
&= 8,40525\,78577\,80233\,76565\,66945\,43304\,38150\ldots \cdot 10^{-4933} \\
1 &= \texttt{0 100000000000001 [0.1]0...00} \\
1+\varepsilon &= \texttt{0 100000000000001 [0.1]0...01} \\
\varepsilon &= +\,0.0\ldots01_2 \cdot 2^{1} \\
&= 2^{-113} \cdot 2^{1} \\
&= 2^{-112} \\
&= 1,92592\,99443\,87235\,85305\,59779\,42584\,92731 \cdot 10^{-34} \\
\pi &= 11.00100\ 10000\ 11111\ 10110\ 10101\ 00010\ 00100\ 00101\ 10100\ldots11100\ 00011\ 10\ldots_2 \\
&= 0.11001001000011111101101010100010001000010110100\ldots111000001110\ldots_2 \cdot 2^2 \\
&\approx 0.11001001000011111101101010100010001000010110100\ldots111000_2 \cdot 2^{16386-16384} \\
&= \texttt{0 100000000000001 [0.1]} \\
&\quad \texttt{10010010000111111011010101000100010000101101000110000100} \\
&\quad \texttt{01101001100010011000110011000101000101110000000110111000}
\end{aligned}
$$

17. Das IBM-Format mit 4 Bytes

Es handelt sich um ein Spezialformat der Firma IBM mit relativ kleinem Wertebereich. Auch die Basis des Zahlensystems (16) fällt aus dem üblichen Rahmen.

17.1. Bitverteilung



17.2. Interpretation der Bits

$m \neq 0$	$\Longrightarrow$	$x = (-1)^v \cdot 0.m \cdot 16^{e-64}$	Normale reelle Zahl
$v=0, e=0, m=0$	$\Longrightarrow$	$x = 0$	Null
$e \neq 0, m=0$	$\Longrightarrow$		undefiniertes Verhalten

Die Prüfung $m \neq 0$ berücksichtigt nur die ersten vier Bits von m.

17.3. Schlüsselwerte

$$\text{minposreal} = 8,636 \cdot 10^{-78} \quad \text{maxreal} = 7,237 \cdot 10^{75} \quad \varepsilon = 5,960 \cdot 10^{-8}$$

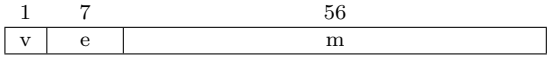
17.4. Beispielzahlen (IBM4)

$$
\begin{aligned}
0 &= \texttt{0 0000000 00000000000000000000000} \\
\texttt{maxreal} &= \texttt{0 1111111 11111111111111111111111} \\
&= +\, \textit{0.111111111111111111111111}_{\,2} \cdot 16^{127-64} \\
&= +\,(1.0 - 2^{-24}) \cdot 16^{63} \\
&\approx +\,1.0 \cdot 16^{63} \\
&= 16^{63} \\
&= 2^{252} \\
&= 7,23700\,5577\ldots \cdot 10^{75} \\
\texttt{minposreal} &= \texttt{0 0000000 00010000000000000000000} \\
&= +\, \textit{0.000100000000000000000000}_{\,2} \cdot 16^{0-63} \\
&= +\,1.0 \cdot 16^{-64} \\
&= 2^{256} \\
&= 8,63616\,8555\ldots \cdot 10^{-78} \\
1 &= \texttt{0 1000001 00010000000000000000000} \\
1+\varepsilon &= \texttt{0 1000001 00010000000000000000001} \\
\varepsilon &= +\, \textit{0.000000000000000000000001}_{\,2} \\
&= 2^{-24} \\
&= 5,96046\,447\ldots \cdot 10^{-8} \\
\pi &= \textit{11.00100 10000 11111 10110 10101 00010}\ldots_2 \\
&= \textit{0.001100100100001111110110101010100010}\ldots_2 \cdot 16^1 \\
&\approx \textit{0.00110010010000111111011}_2 \cdot 2^{65-64} \\
&= \texttt{0 1000001 00110010010000111111011}
\end{aligned}
$$

18. Das IBM-Format mit 8 Bytes

Es handelt sich um ein Spezialformat der Firma IBM mit demselben Wertebereich wie beim IBM-Format mit 4 Bytes. Diese Eigenschaft macht sich bei wissenschaftlichen Berechnungen gelegentlich unangenehm bemerkbar.

18.1. Bitverteilung



18.2. Interpretation der Bits

$$
\begin{aligned}
m \neq 0 &\implies x = (-1)^v \cdot 0.m \cdot 16^{e-64} && \text{Normale reelle Zahl} \\
v = 0, e = 0, m = 0 &\implies x = 0 && \text{Null} \\
e \neq 0, m = 0 &\implies && \text{undefiniertes Verhalten}
\end{aligned}
$$

Die Prüfung $m \neq 0$ berücksichtigt nur die ersten vier Bits von m.

18.3. Schlüsselwerte

$$\text{minposreal} = 8,636 \cdot 10^{-78} \quad \text{maxreal} = 7,237 \cdot 10^{75} \quad \varepsilon = 1,388 \cdot 10^{-17}$$

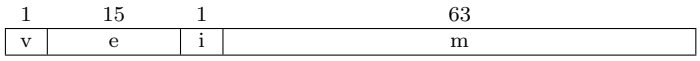
18.4. Beispielzahlen (IBM8)

$$
\begin{aligned}
0 &= \texttt{0 0000000 00000000000000000000000000000000000000000000000000000000}\\
\texttt{maxreal} &= \texttt{0 1111111 11111111111111111111111111111111111111111111111111111111}\\
&= +\,\textit{0.1111111111111111111111111111111111111111111111111111111111111111}_{\,2} \cdot 16^{127-64}\\
&= +\,(1.0 - 2^{-56}) \cdot 16^{63}\\
&\approx +\,1.0 \cdot 16^{63}\\
&= 2^{252}\\
&= 7,\,23700\,55773\,32262\,21\ldots \cdot 10^{75}\\
\texttt{minposreal} &= \texttt{0 0000000 00010000000000000000000000000000000000000000000000000000}\\
&= +\,\textit{0.0001000000000000000000000000000000000000000000000000000000000000}_{\,2} \cdot 16^{0-63}\\
&= +\,1.0 \cdot 16^{-64}\\
&= 2^{256}\\
&= 8,\,63616\,85550\,94444\,6\ldots \cdot 10^{-78}\\
1 &= \texttt{0 1000001 00010000000000000000000000000000000000000000000000000000}\\
1+\varepsilon &= \texttt{0 1000001 00010000000000000000000000000000000000000000000000000001}\\
\varepsilon &= +\,\textit{0.0000000000000000000000000000000000000000000000000000000000000001}_{\,2}\\
&= 2^{-56}\\
&= 1,\,38777\,87807\,81445\,67\ldots \cdot 10^{-17}\\
\pi &= \textit{11.00100 10000 11111 10110 10101 00010 00100 00101 10100 01100 00100 01101}\ldots_2\\
&= \textit{0.0011001001000011111101101010100010001000010110100011000010001101}\ldots_2 \cdot 16^1\\
&\approx \textit{0.0011001001000011111101101010100010001000010110100011000}1_2 \cdot 2^{65-64}\\
&= \texttt{0 1000001 00110010010000111111011010101000100010000101101000110001}
\end{aligned}
$$

19. Das interne Format des Coprozessors 8087

Dieses Format ist das interne Rechenformat des numerischen 80x87-Coprozessors von Intel. Ursprünglich gar nicht zur Verwendung durch den Anwendungsprogrammierer gedacht, sollte es Berechnungen mit beiden IEEE-Formaten vereinheitlichen und durch zusätzliche Genauigkeit bei Arithmetik und elementaren transzendenten Funktionen die Gültigkeit auch der letzten Mantissenbits im IEEE4- und IEEE8-Format sicherstellen. Alle Berechnungen werden vom Coprozessor in diesem Format durchgeführt. Erst wenn die Ergebnisse im Speicher abgelegt werden, werden sie auf die gewünschte (kürzere) Mantissenlänge gerundet. Manche Compiler geben dieses Format entgegen den Intentionen der Konstrukteure an den Anwendungsprogrammierer weiter, der sich darüber im klaren sein sollte, daß z.B. bei transzendenten Funktionen Ungenauigkeiten in den letzten Stellen auftreten können.

19.1. Bitverteilung



19.2. Interpretation der Bits

$0 \le e < 32767$	$\Longrightarrow$	$x = (-1)^v \cdot i.m \cdot 2^{e-16383}$	Normale reelle Zahl
$e = 32767, m = 0$	$\Longrightarrow$	$x = (-1)^v \cdot \infty$	Unendlich
$e = 32767, m \neq 0$	$\Longrightarrow$	$x = NAN$	„Nicht-Zahl“, not a number

Die Zahl 0 wird durch $i = 0, m = 0$ dargestellt und ist hier kein Sonderfall.

19.3. Schlüsselwerte

$$\text{minposreal} = 1,681 \cdot 10^{-4932} \quad \text{maxreal} = 1,190 \cdot 10^{4932} \quad \varepsilon = 1,084 \cdot 10^{-19}$$

19.4. Verbreitung des Formats

Rechner	Betriebssystem	Compiler	Pascal-Bezeichnung
PC	MS-DOS	Turbo ab 4.0	extended
PC	Windows	Turbo für Windows	extended

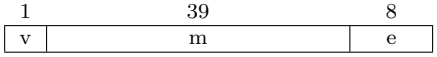
19.5. Beispielzahlen (8087)

$$
\begin{aligned}
0 &= \texttt{0 000000000000000 0 0000000000000000000000000000000000000000000000000000000000000000} \\
\texttt{maxreal} &= \texttt{0 111111111111101 1111111111111111111111111111111111111111111111111111111111111111} \\
&= +\,\textit{1.1111111111111111111111111111111111111111111111111111111111111111}_{\,2} \cdot 2^{32766-16383} \\
&= +\,(2.0 - 2^{-63}) \cdot 2^{16383} \\
&\approx +\,2.0 \cdot 2^{16383} \\
&= 2^{16384} \\
&= 1,18973\,14953\,57231\,765\ldots \cdot 10^{4932} \\
\texttt{minposreal} &= \texttt{0 000000000000000 1 0000000000000000000000000000000000000000000000000000000000000000} \\
&= +\,\textit{1.0000000000000000000000000000000000000000000000000000000000000000}_{\,2} \cdot 2^{0-16383} \\
&= +\,1.0 \cdot 2^{-16383} \\
&= 1,68105\,15715\,56046\,753\ldots \cdot 10^{-4932} \\
1 &= \texttt{0 011111111111111 1 0000000000000000000000000000000000000000000000000000000000000000} \\
1+\varepsilon &= \texttt{0 011111111111111 1 0000000000000000000000000000000000000000000000000000000000000001} \\
\varepsilon &= +\,\textit{0.0000000000000000000000000000000000000000000000000000000000000001}_{\,2} \\
&= 2^{-63} \\
&= 1,08420\,21724\,85504\,434\ldots \cdot 10^{-19} \\
\pi &= \textit{11.00100 10000 11111 10110 10101 00010 00100 00101 10100 01100 00100 01101 00110 00100}\ldots_2 \\
&= \textit{1.00100100001111111011010101000100010000101101000110000100011010011000100}\ldots_2 \cdot 2^1 \\
&\approx \textit{1.00100100001111111011010101000100010000101101000110000100011010}_{\,2} \cdot 2^{16384-16383} \\
&= \texttt{0 100000000000000 1 00100100001111111011010101000100010000101101000110000100011010 1}
\end{aligned}
$$

20. Das Turbo-Pascal Format mit 6 Bytes

Dieses exotische Format wurde von Borland als Kompromiß zwischen ausreichender Genauigkeit (mehr als 4 Bytes) und ausreichender Rechengeschwindigkeit bei Emulation der Arithmetik durch die nichtnumerische CPU entworfen. Es kommt nur bei Borland vor, ist im Grunde durch die Entwicklung der modernen leistungsfähigen Coprozessoren überholt und wird wahrscheinlich auf lange Sicht nicht mehr unterstützt. Trotzdem ist es noch in den Fällen günstig, in denen 8-Byte-Zahlen zu viel Speicher fressen und 4-Byte-Zahlen zu ungenau sind. Die Arithmetik ist wegen notwendiger Bitmanipulationen auch bei Verwendung des Coprozessors erstaunlich langsam.

20.1. Bitverteilung



20.2. Interpretation der Bits

$0 < e \le 255$	$\Longrightarrow$	$x = (-1)^v \cdot 1.m \cdot 2^{e-129}$	Normale reelle Zahl
$e = 0$	$\Longrightarrow$	$x = 0$	Null

20.3. Schlüsselwerte

$$\text{minposreal} = 2,939 \cdot 10^{-39} \quad \text{maxreal} = 1,701 \cdot 10^{38} \quad \varepsilon = 1,819 \cdot 10^{-12}$$

20.4. Beispielzahlen (Turbo)

$$
\begin{aligned}
0 &= \texttt{0 00000000000000000000000000000000 00000000} \\
\texttt{maxreal} &= \texttt{0 11111111111111111111111111111111 11111111} \\
&= +\ \textit{1.11111111111111111111111111111111}_{\,2} \cdot 2^{255-129} \\
&= +\,(2.0 - 2^{-39}) \cdot 2^{126} \\
&\approx +\,2.0 \cdot 2^{126} \\
&= 2^{127} \\
&= 1,70141\,18346\,0\ldots \cdot 10^{38} \\
\texttt{minposreal} &= \texttt{0 00000000000000000000000000000000 00000001} \\
&= +\ \textit{1.00000000000000000000000000000000}\ \cdot 2^{1-129} \\
&= +\,1.0 \cdot 2^{-128} \\
&= 2,93873\,58770\,557\ldots \cdot 10^{-39} \\
1 &= \texttt{0 00000000000000000000000000000000 10000010} \\
1+\varepsilon &= \texttt{0 00000000000000000000000000000001 10000010} \\
\varepsilon &= +\ \textit{0.00000000000000000000000000000001}_{\,2} \\
&= 2^{-39} \\
&= 1,81898\,94035\,458\ldots \cdot 10^{-12} \\
\pi &= \textit{11.00100 10000 11111 10110 10101 00010 00100 00101}\ldots_2 \\
&= \textit{1.10010010000111111011010101000100010000101}\ldots_2 \cdot 2^1 \\
&\approx \textit{1.10010010000111111011010101000100010000}1_{\,2} \cdot 2^{130-129} \\
&= \texttt{0 10010010000111111011010101000100010000 1} \texttt{0000010}
\end{aligned}
$$

21. Weitere Zahlenwerte

$0 =$	$0_{16} =$	$0_8 =$	0_2
$1 =$	$1_{16} =$	$1_8 =$	1_2
$2 =$	$2_{16} =$	$2_8 =$	10_2
$3 =$	$3_{16} =$	$3_8 =$	11_2
$4 =$	$4_{16} =$	$4_8 =$	100_2
$5 =$	$5_{16} =$	$5_8 =$	101_2
$6 =$	$6_{16} =$	$6_8 =$	110_2
$7 =$	$7_{16} =$	$7_8 =$	111_2
$8 =$	$8_{16} =$	$10_8 =$	1000_2
$9 =$	$9_{16} =$	$11_8 =$	1001_2
$10 =$	$A_{16} =$	$12_8 =$	1010_2
$11 =$	$B_{16} =$	$13_8 =$	1011_2
$12 =$	$C_{16} =$	$14_8 =$	1100_2
$13 =$	$D_{16} =$	$15_8 =$	1101_2
$14 =$	$E_{16} =$	$16_8 =$	1110_2
$15 =$	$F_{16} =$	$17_8 =$	1111_2
$16 =$	$10_{16} =$	$20_8 =$	10000_2
$30 =$	$1E_{16} =$	$36_8 =$	11110_2
$31 =$	$1F_{16} =$	$37_8 =$	11111_2
$32 =$	$20_{16} =$	$40_8 =$	100000_2
$33 =$	$21_{16} =$	$41_8 =$	100001_2
$34 =$	$22_{16} =$	$42_8 =$	100010_2
$62 =$	$3E_{16} =$	$76_8 =$	111110_2
$63 =$	$3F_{16} =$	$77_8 =$	111111_2
$64 =$	$40_{16} =$	$100_8 =$	1000000_2
$65 =$	$41_{16} =$	$101_8 =$	1000001_2
$66 =$	$42_{16} =$	$102_8 =$	1000010_2

100 =	64_{16} =	144_{8} =	1100100_{2}
126 =	$7E_{16}$ =	176_{8} =	1111110_{2}
127 =	$7F_{16}$ =	177_{8} =	1111111_{2}
128 =	80_{16} =	200_{8} =	10000000_{2}
129 =	81_{16} =	201_{8} =	10000001_{2}
130 =	82_{16} =	202_{8} =	10000010_{2}
254 =	FE_{16} =	376_{8} =	11111110_{2}
255 =	FF_{16} =	377_{8} =	11111111_{2}
256 =	100_{16} =	400_{8} =	100000000_{2}
257 =	101_{16} =	401_{8} =	100000001_{2}
258 =	102_{16} =	402_{8} =	100000010_{2}
510 =	$1FE_{16}$ =	776_{8} =	111111110_{2}
511 =	$1FF_{16}$ =	777_{8} =	111111111_{2}
512 =	200_{16} =	1000_{8} =	1000000000_{2}
513 =	201_{16} =	1001_{8} =	1000000001_{2}
514 =	202_{16} =	1002_{8} =	1000000010_{2}
1000 =	$3E8_{16}$ =	1750_{8} =	1111101000_{2}
1022 =	$3FE_{16}$ =	1776_{8} =	1111111110_{2}
1023 =	$3FF_{16}$ =	1777_{8} =	1111111111_{2}
1024 =	400_{16} =	2000_{8} =	10000000000_{2}
1025 =	401_{16} =	2001_{8} =	10000000001_{2}
1026 =	402_{16} =	2002_{8} =	10000000010_{2}
2046 =	$7FE_{16}$ =	3776_{8} =	11111111110_{2}
2047 =	$7FF_{16}$ =	3777_{8} =	11111111111_{2}
2048 =	800_{16} =	4000_{8} =	100000000000_{2}
2049 =	801_{16} =	4001_{8} =	100000000001_{2}
2050 =	802_{16} =	4002_{8} =	100000000010_{2}
4095 =	FFF_{16} =	7777_{8} =	111111111111_{2}
4096 =	1000_{16} =	10000_{8} =	1000000000000_{2}
4097 =	1001_{16} =	10001_{8} =	1000000000001_{2}
8191 =	$1FFF_{16}$ =	17777_{8} =	1111111111111_{2}
8192 =	2000_{16} =	20000_{8} =	10000000000000_{2}
8193 =	2001_{16} =	20001_{8} =	10000000000001_{2}
10000 =	2710_{16} =	23420_{8} =	10011100010000_{2}
16383 =	$3FFF_{16}$ =	37777_{8}	
16384 =	4000_{16} =	40000_{8}	
16385 =	4001_{16} =	40001_{8}	
24572 =	$5FFC_{16}$ =	57774_{8}	
24573 =	$5FFD_{16}$ =	57775_{8}	
24574 =	$5FFE_{16}$ =	57776_{8}	
24575 =	$5FFF_{16}$ =	57777_{8}	
24576 =	6000_{16} =	60000_{8}	
32767 =	$7FFF_{16}$ =	77777_{8}	
32768 =	8000_{16} =	100000_{8}	
32769 =	8001_{16} =	100001_{8}	
65535 =	$FFFF_{16}$ =	177777_{8}	
65536 =	10000_{16} =	200000_{8}	
65537 =	10001_{16} =	200001_{8}	
100000 =	$186A0_{16}$ =	303240_{8}	
131071 =	$1FFFF_{16}$ =	377777_{8}	
131072 =	20000_{16} =	400000_{8}	
131073 =	20001_{16} =	400001_{8}	
262144 =	40000_{16} =	1000000_{8}	
524288 =	80000_{16} =	2000000_{8}	
1000000 =	$F4240_{16}$ =	3641100_{8}	
1048576 =	100000_{16} =	4000000_{8}	
2097152 =	200000_{16} =	10000000_{8}	
4194304 =	400000_{16} =	20000000_{8}	
8388608 =	800000_{16} =	40000000_{8}	
10000000 =	989680_{16} =	46113200_{8}	
16777216 =	1000000_{16} =	100000000_{8}	
33554432 =	2000000_{16} =	200000000_{8}	

$$
\begin{aligned}
67108864 &= 4000000_{16} &&= 400000000_{8}\\
100000000 &= 5F5E100_{16} &&= 575360400_{8}\\
134217728 &= 8000000_{16} &&= 1000000000_{8}\\
268435456 &= 10000000_{16} &&= 2000000000_{8}\\
536870912 &= 20000000_{16} &&= 4000000000_{8}\\
1000000000 &= 3B9ACA00_{16} &&= 7346545000_{8}\\
1073741824 &= 40000000_{16} &&= 10000000000_{8}\\
2147483648 &= 80000000_{16} &&= 20000000000_{8}\\
4294967296 &= 100000000_{16} &&= 40000000000_{8}\\
8589934592 &= 200000000_{16} &&= 100000000000_{8}\\
10000000000 &= 2540BE400_{16} &&= 112402762000_{8}\\
100000000000 &= 174876E800_{16} &&= 1351035564000_{8}
\end{aligned}
$$

$$
\begin{aligned}
2^{-16384} &= 8,40525\,78577\,80233\,76565\,66945\,43304\,38150\,64951\,98362\,11617\,81002\,72068\,07452\,18558\ldots\cdot 10^{-04933}\\
2^{-16383} &= 1,68105\,15715\,56046\,75313\,13389\,08660\,87630\,12990\,39672\,42323\,56200\,54413\,61490\,43712\ldots\cdot 10^{-04932}\\
2^{-16382} &= 3,36210\,31431\,12093\,50626\,26778\,17321\,75260\,25980\,79344\,84647\,12401\,08827\,22980\,87423\ldots\cdot 10^{-04932}\\
2^{-8190} &= 3,66720\,77351\,09694\,33124\,28247\,84096\,96633\,19127\,29942\,71713\,44745\,65277\,90107\,31675\ldots\cdot 10^{-02466}\\
2^{-1024} &= 5,56268\,46462\,68003\,45772\,55817\,93331\,01016\,05480\,39951\,15582\,95763\,83318\,54221\,80110\ldots\cdot 10^{-00309}\\
2^{-1022} &= 2,22507\,38585\,07201\,38309\,02327\,17332\,40406\,42192\,15980\,46233\,18305\,53327\,41688\,72044\ldots\cdot 10^{-00308}\\
2^{-256} &= 8,63616\,85550\,94444\,62538\,63518\,62800\,39957\,11160\,00364\,43628\,13850\,23703\,47016\,85918\ldots\cdot 10^{-00078}\\
2^{-143} &= 8,96831\,01716\,78829\,25391\,18693\,33055\,46324\,01936\,76428\,00970\,09392\,45237\,01689\,46629\ldots\cdot 10^{-00044}\\
2^{-128} &= 2,93873\,58770\,55718\,76992\,18413\,43055\,61419\,45466\,63891\,93021\,88037\,71879\,26569\,60431\ldots\cdot 10^{-00039}\\
2^{-126} &= 1,17549\,43508\,22287\,50796\,87365\,37222\,24567\,78186\,65556\,77208\,75215\,08751\,70627\,84173\ldots\cdot 10^{-00038}\\
2^{-112} &= 1,92592\,99443\,87235\,85305\,59779\,42584\,92731\,85381\,01648\,21538\,81952\,39938\,79556\,65588\ldots\cdot 10^{-00034}\\
2^{-95} &= 2,52435\,48967\,07237\,77731\,75314\,08904\,91593\,49542\,60592\,34887\,36152\,64892\,57812\,5\cdot 10^{-00029}\\
2^{-63} &= 1,08420\,21724\,85504\,43400\,74528\,00869\,94171\,14257\,8125\cdot 10^{-00019}\\
2^{-56} &= 1,38777\,87807\,81445\,67552\,95395\,85113\,52539\,0625\cdot 10^{-00017}\\
2^{-55} &= 2,77555\,75615\,62891\,35105\,90791\,70227\,05078\,125\cdot 10^{-00017}\\
2^{-52} &= 2,22044\,60492\,50313\,08084\,72633\,36181\,64062\,5\cdot 10^{-00016}\\
2^{-47} &= 7,10542\,73576\,01001\,85871\,12426\,75781\,25\cdot 10^{-00015}\\
2^{-39} &= 1,81898\,94035\,45856\,47583\,00781\,25\cdot 10^{-00012}\\
2^{-24} &= 5,96046\,44775\,39062\,5\cdot 10^{-00008}\\
2^{-23} &= 1,19209\,28955\,07812\,5\cdot 10^{-00007}\\
2^{126} &= 8,50705\,91730\,23461\,58658\,43651\,85794\,20528\,64\cdot 10^{+00037}\\
2^{127} &= 1,70141\,18346\,04692\,31731\,68730\,37158\,84105\,728\cdot 10^{+00038}\\
2^{128} &= 3,40282\,36692\,09384\,63463\,37460\,74317\,68211\,456\cdot 10^{+00038}\\
2^{252} &= 7,23700\,55773\,32262\,21397\,31865\,63042\,99424\,08293\,74041\,60253\,52524\,66099\,00049\,45706\ldots\cdot 10^{+00075}\\
2^{1023} &= 8,98846\,56743\,11579\,53864\,65259\,53945\,12366\,80898\,84894\,71153\,28636\,71504\,05788\,66338\ldots\cdot 10^{+00307}\\
2^{1024} &= 1,79769\,31348\,62315\,90772\,93051\,90789\,02473\,36179\,76978\,94230\,65727\,34300\,81157\,73268\ldots\cdot 10^{+00308}\\
2^{8190} &= 2,72687\,03390\,48539\,82365\,74606\,11834\,45715\,61206\,60404\,99058\,17310\,79581\,96547\,43035\ldots\cdot 10^{+02465}\\
2^{16383} &= 5,94865\,74767\,86158\,82542\,87966\,33140\,03565\,38172\,23435\,48255\,11873\,63374\,10616\,63074\ldots\cdot 10^{+04931}\\
2^{16384} &= 1,18973\,14953\,57231\,76508\,57593\,26628\,00713\,07634\,44687\,09651\,02374\,72674\,82123\,32615\ldots\cdot 10^{+04932}
\end{aligned}
$$