<u>Beispiele</u> für die Gleitkommadarstellung (mit Basis **b = 2**):

0,5	**= 0,5**	$\cdot 2^{0}$
- 17,0	**= - 0,53125**	$\cdot 2^{5}$
1,024	**= 0,512**	$\cdot 2^{1}$
- 0,001	**= - 0,512**	$\cdot 2^{-9}$
3,141592...	**= 0,785398...**	$\cdot 2^{2}$
	Codierung in **m**	Codierung in **e**

Zur Erinnerung:
$\mathbf{n = +/- m \cdot 2^e}$
mit $0{,}5 \leq m < 1$

Bemerkung: Es ist **nicht** möglich, **periodische** Zahlen **genau** abzubilden.

Wieviele Bits werden für die Gleitkommadarstellung benötigt?

In der Praxis werden im Allgemeinen **drei** Fälle unterschieden:

- **Einfache** Genauigkeit (single precision) **32 Bits**
- **Doppelte** Genauigkeit (double precision) **64 Bits**
- **Erweiterte** Genauigkeit (extended precision) **80 Bits**

Aufteilung der **32 Bits** bei **einfacher** Genauigkeit:

- **1 Bit** für das **Vorzeichen**
- **8 Bits** für den **Exponenten e**
- **23 Bits** für die **Mantisse m**

Zur Erinnerung:
$\mathbf{n = +/- m \cdot b^e}$
mit $1/b \leq m < 1$

	1 Bit	8 Bits	23 Bits
Vorzeichen →		**Exponent e**	**Mantisse m**

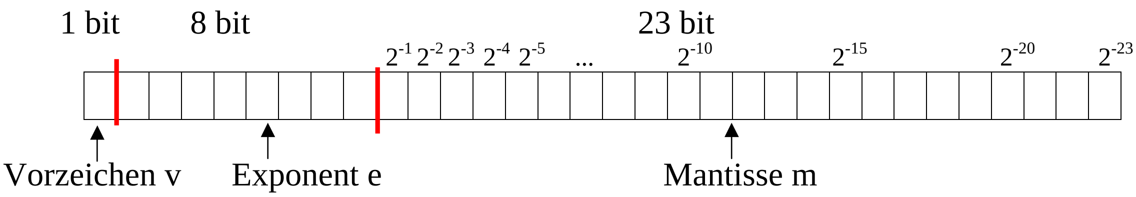
Bei **doppelter** Genauigkeit ist die Aufteilung **1** zu **11** zu **52 Bits**,
bei **erweiterter** Genauigkeit ist die Aufteilung **1** zu **15** zu **64 Bits**.

Informationsdarstellung 2.2

Darstellung einer 32-Bit-Gleitkommazahl **n**:

$$\mathbf{n = (-1)^v \cdot m \cdot b^e}$$

mit **Vorzeichenbit v**, **Basis b**, **Mantisse m** ($1/b \le m < 1$), **Exponent e**



0,5 = 2^{-1}	0,001953125 = 2^{-9}	0,00007629394553125 = 2^{-17}
0,25 = 2^{-2}	0,0009765625 = 2^{-10}	0,000003814697265625 = 2^{-18}
0,125 = 2^{-3}	0,00048828125 = 2^{-11}	0,0000019073486328125 = 2^{-19}
0,0625 = 2^{-4}	0,000244140625 = 2^{-12}	0,00000095367431640625 = 2^{-20}
0,03125 = 2^{-5}	0,0001220703125 = 2^{-13}	0,000000476837158203125 = 2^{-21}
0,015625 = 2^{-6}	0,00006103515625 = 2^{-14}	0,0000002384185791015625 = 2^{-22}
0,0078125 = 2^{-7}	0,000030517578125 = 2^{-15}	0,00000011920928955078125 = 2^{-23}
0,00390625 = 2^{-8}	0,0000152587890625 = 2^{-16}	

Informationsdarstellung 2.2

Beispiel: (1)

Zur Erinnerung:

$\mathbf{n = (-1)^v \cdot m \cdot 2^e}$

mit $0{,}5 \le m < 1$

Codierung des Dezimalbruchs 0,5:

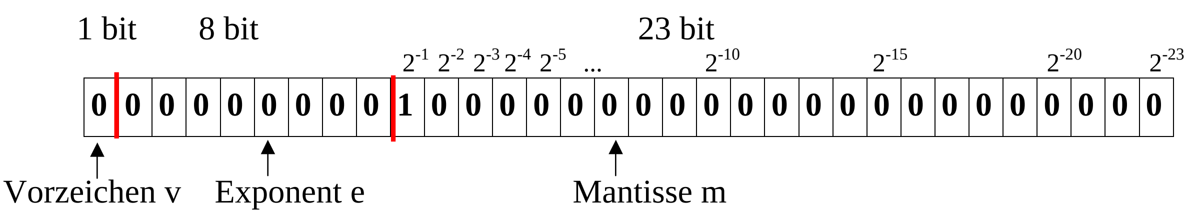
$\mathbf{0{,}5 = 0{,}5 \cdot 2^0}$

$\boxed{\mathbf{+\ 0{,}5 = +\ 0{,}5 \cdot 2^0}}$

Codierung von **e=0** als Binärzahl **00000000**

Codierung von „ + “ in **v** als Binärzahl **0**

Codierung von **m**: $\mathbf{0{,}5 = 2^{-1}}$



Beispiel: (2)

Codierung des Dezimalbruchs –17,0:

$-17{,}0 = -\,0{,}53125 \cdot 2^5$

$$-17{,}0 = -\,0{,}53125 \cdot 2^5$$

Codierung von **e=5** als Binärzahl **00000101**

Codierung von „ - “ in **v** als Binärzahl **1**

Codierung von **m:**
$0{,}53125 = 0{,}5 + 0{,}03125 = 2^{-1} + 2^{-5}$

1 bit | 8 bit | 23 bit

2^{-1} 2^{-2} 2^{-3} 2^{-4} 2^{-5} ... 2^{-10} 2^{-15} 2^{-20} 2^{-23}

1	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Vorzeichen   Exponent e   Mantisse m

Beispiel (3) - Codierung des Dezimalbruchs **1,024**

Zur Erinnerung:
$n = (-1)^v \cdot m \cdot 2^e$
mit $0{,}5 \le m < 1$

$$+1{,}024 = +\,0{,}512 \cdot 2^1$$

Codierung von **e=1** als Binärzahl **00000001**

Codierung von „ + “ in **v** als Binärzahl **0**

Codierung von **m:**
$0{,}512 = 2^{-1} + 2^{-7} + 2^{-8} + 2^{-12} + 2^{-15} + 2^{-18} + 2^{-19} + 2^{-21} + 2^{-22} + 2^{-23} + \ldots$

1 bit | 8 bit | 23 bit

2^{-1} 2^{-2} 2^{-3} 2^{-4} 2^{-5} ... 2^{-10} 2^{-15} 2^{-20} 2^{-23}

0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	1	0	0	1	0	0	1	1	0	1	1	1

Vorzeichen   Exponent e =1   Mantisse m = 0,51199996471405029296875

Bemerkung: Die Mantisse **0,512** kann im dualen System **nicht exakt** repräsentiert werden. Bei einer Genauigkeit von 32 Bit bleibt ein **Rest** von **$0{,}000000035285949707031 25_{10}$**

Beispiel: (4a) - Codierung des Dezimalbruchs 0,001

Es gilt: $\mathbf{0{,}001 = 0{,}512 \cdot 2^{-9}}$

> **Zur Erinnerung:**
> $\mathbf{n = (-1)^v \cdot m \cdot 2^e}$
> mit $0{,}5 \leq m < 1$

➔ Problem: „*Wie stelle ich den **negativen** Exponenten **e = -9** dar?*“

Möglichkeit 1:

„Abschneiden“ eines Bits $\mathbf{v_e}$ für das Vorzeichen des Exponenten:

1 bit | 1+7 bit | 23 bit

2^{-1} 2^{-2} 2^{-3} 2^{-4} 2^{-5} ... 2^{-10} 2^{-15} 2^{-20} 2^{-23}

0	1	0	0	0	1	0	0	1	1	0	0	0	0	0	1	1	0	0	0	1	0	0	1	0	0	1	1	0	1	1	1

v $\quad$ $\mathbf{v_e}$ $\quad$ e=9 $\quad$ m = 0,51199996471405029296875

Beispiel: (4b) - Codierung des Dezimalbruchs 0,001

Möglichkeit 2: Einführung einer **Charakteristik**

> **Zur Erinnerung:**
> $\mathbf{n = (-1)^v \cdot m \cdot 2^{e'-c}}$
> mit $0.5 \leq m < 1$

Speicherung von **e’ := e + c** mit **e’ ≥ 0** anstelle von **e**.
c heißt **Charakteristik**, zum Beispiel **c := 100**.

Es gilt: $\mathbf{0{,}001 = 0{,}512 \cdot 2^{-9}}$

➔ **e = -9, also e’ = e + c = -9 + 100 = 91**

1 bit | 8 bit | 23 bit

2^{-1} 2^{-2} 2^{-3} 2^{-4} 2^{-5} ... 2^{-10} 2^{-15} 2^{-20} 2^{-23}

0	0	1	0	1	1	0	1	1	1	0	0	0	0	0	1	1	0	0	0	1	0	0	1	0	0	1	1	0	1	1	1

v $\quad$ e’ = 91 $\quad$ m

Zusammenfassung:

Wie gehe ich bei der binären Codierung von Dezimalbrüchen vor?

(1) **Normierung** der Dezimalzahl $\mathbf{n = (-1)^v \cdot m \cdot 2^e}$ **mit** $0{,}5 \le m < 1$

(2) Eventuell Berechnung von **e'** bezüglich der **Charakteristik c**

(3) **Codierung** von **e** bzw. **e'** als Summe von positiven 2er-Potenzen

(4) **Codierung** von **m** als Summe von negativen 2er-Potenzen

Beispiel: Darstellung der Zahl –24 mit einer Charakteristik c=127

(1) $\mathbf{n = (-1)^1 \cdot 0{,}75 \cdot 2^5}$

(2) e'= 127+5=132

(3) und (4)

$2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}\ 2^{-5} \ldots\ 2^{-10}\ \ldots\ 2^{-15}\ \ldots\ 2^{-20}\ \ldots\ 2^{-23}$

1	1	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

In der **Praxis** wird häufig der sog. **IEEE-754-Standard** verwendet.

Geg.: reelle Zahl n

$$\boxed{\mathbf{n = (-1)^v \cdot 1{,}m' \cdot 2^{e'-127}}}$$

mit - **Vorzeichen v** $\in$ **{0,1}**

- **Mantisse m = 1,m'**
- **Exponent e' = c + e** mit **Charakteristik c = 127**

Beispiel: Darstellung von $\mathbf{0{,}001 = 1{,}024 \cdot 2^{-8}}$

➔ m = 1,m' also **m' = 0,024** und **e'** = 127 – 8 = **119**

1 bit 8 bit 23 bit

$2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}\ 2^{-5} \ldots\ 2^{-10}\ \ldots\ 2^{-15}\ \ldots\ 2^{-20}\ \ldots\ 2^{-23}$

0	0	1	1	1	0	1	1	1	0	0	0	0	0	1	1	0	0	0	1	0	0	1	0	0	1	1	0	1	1	1	0

Vorzeichen v Exponent e' Mantisse m'

Durch diese spezielle Codierung wird ein weiteres Bit eingespart.

Codierung von Textdaten

Ein **Textstück** (z.B. „x“ oder „Hallo Welt!“) wird dargestellt, indem man es in **Anführungszeichen** (**“** oder **’**) setzt.

<u>Beachte:</u> Eine **Zahl in Anführungszeichen** (z.B. “7“) wird als **Textzeichen** interpretiert, d.h. es können damit **keine** Rechenoperationen durchgeführt werden

Ein weit verbreiteter **7-Bit-Code** zur Darstellung von **Ziffern**, **Buchstaben** und sogenannte **Sonderzeichen** ist die **ASCII**-Darstellung.
(**ASCII** = **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange)

Das im Byteformat **freie achte Bit** dient meist als **Paritätsbit** (Prüfbit) oder wird zur **Erweiterung** des darstellbaren **Zeichensatzes** verwendet.

Die folgende Tabelle stellt den ASCII-Zeichensatz **dar:**

NUL	*0*	DLE	*16*	␣	*32*	0	*48*	@	*64*	P	*80*	@	*96*	p	*112*
SOH	*1*	DC1	*17*	!	*33*	1	*49*	A	*65*	Q	*81*	a	*97*	q	*113*
STX	*2*	DC2	*18*	"	*34*	2	*50*	B	*66*	R	*82*	b	*98*	r	*114*
ETX	*3*	DC3	*19*	#	*35*	3	*51*	C	*67*	S	*83*	c	*99*	s	*115*
EOT	*4*	DC4	*20*	$	*36*	4	*52*	D	*68*	T	*84*	d	*100*	t	*116*
ENQ	*5*	NAK	*21*	%	*37*	5	*53*	E	*69*	U	*85*	e	*101*	u	*117*
ACK	*6*	SYN	*22*	&	*38*	6	*64*	F	*70*	V	*86*	f	*102*	v	*118*
BEL	*7*	ETB	*23*	'	*39*	7	*55*	G	*71*	W	*87*	g	*103*	w	*119*
BS	*8*	CAN	*24*	(	*40*	8	*56*	H	*72*	X	*88*	h	*104*	x	*120*
HT	*9*	EM	*25*	)	*41*	9	*57*	I	*73*	Y	*89*	i	*105*	y	*121*
LF	*10*	SUB	*26*	*	*42*	:	*58*	J	*74*	Z	*90*	j	*106*	z	*122*
VT	*11*	ESC	*27*	+	*43*	;	*59*	K	*75*	[	*91*	k	*107*	{	*123*
FF	*12*	FS	*28*	,	*44*	<	*60*	L	*76*	\	*92*	l	*108*	|	*124*
CR	*13*	QS	*29*	–	*45*	=	*61*	M	*77*	]	*93*	m	*109*	}	*125*
SO	*14*	RS	*30*	.	*46*	>	*62*	N	*78*	^	*94*	n	*110*	~	*126*
SI	*15*	US	*31*	/	*47*	?	*63*	O	*79*	_	*95*	o	*111*	DEL	*127*

Steuerzeichen Schriftzeichen

eingeschränkter ASCII-Zeichensatz

Beispiel: ***Gegeben sei folgender 7 Bit-ASCII-Code:***

69 – 108 – 118 – 105 – 115 – 32 – 108 – 101 – 98 – 116

Gemäß der ASCII-Tabelle ergibt sich folgender Text:

Was unterscheidet 75 von “75“?

- Zahl 75

... hat die folgende interne Darstellung

0	**1**	**0**	**0**	**1**	**1**	**1**	**1**

Repräsentation der **Zahl** 75

- Zeichen(kette) “75“

... hat die folgende interne Darstellung

0	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**1**

ASCII-Repräsentation für das **Zeichen** “7“ = 55

ASCII-Repräsentation für das **Zeichen** “5“ = 53

Stellt eine** gegebene Bitfolge **nun eine** Zahl **oder ein** Zeichen **dar?

Bitfolge (binäre Darstellung)

0	**1**	**0**	**0**	**1**	**0**	**1**	**1**

kann interpretiert werden als:

natürliche Zahl:
$2^6 + 2^3 + 2^1 + 2^0 =$ **75**

ASCII-Zeichen mit Nr. 75:
(vgl. Tabelle) ➔ **“K“**

Um eine gegebene Bitfolge **korrekt** interpretieren **zu können, muss man also wissen, mit welchem** Codierungsverfahren **die binäre Darstellung erzeugt wurde (➔Aufgabe des Betriebssystems).**

Literatur: Helmut Bähring: Mikrorechner-Systeme, Springer 1994, u.a.