

Rechnerarchitektur und -organisation

- Zahlensysteme, Zahlendarstellung, Arithmetik und Numerik - - Teil 2 -

Prof. Dr.-Ing. Peter Gregorius

Beuth Hochschule für Technik Berlin
FB VI Technische Informatik - Embedded Systems

23. November 2019

Übersicht zur Lehrveranstaltung I

Termin	Inhalt
1	Einführung in Rechnerarchitekturen
2	Architekturprinzipien, Modularisierung und Abstraktion
3	Funktion einer zentralen Recheneinheit
4	ISA, Assembler und Betriebssystemebene
5	Zahlensysteme, Zahlendarstellung, Arithmetik und Numerik - Teil 1
6	Zahlensysteme, Zahlendarstellung, Arithmetik und Numerik - Teil 2
7	Rechnerarithmetik - Addition/Subtraktion
8	Rechnerarithmetik - Multiplikation/Division
9	FPU, CORDIC, DSP-Einheiten und GPU
10	SOB, SOC, HPC, Embedded CPU/MCU, etc.
11	Speicherorganisation und Speichertechnologien
12	Datenschnittstellen in Rechnersystemen
13	Rechnerarchitekturen und Multimedia
Selbststudium	Grundlagen der Technischen Informatik / Wiederholung

Inhalt

Einführung

Motivation

Konvertierung von Zahlenformaten

Höhere Funktionen

Numerik

Ausblick

Worum geht es?

- ▶ Rechnerarithmetik
 - ▶ Addition
 - ▶ Subtraktion
 - ▶ Multiplikation
 - ▶ Division

⇒ Arithmetik Schaltungen, **A**rithmetic and **L**ogic **U**nit (ALU)

Wie werden Arithmetikoperation in Hardware abgebildet?

Inhalte der Lehreinheit:

- ▶ Konvertierung in verschiedene Zahlenformate für Rechner
 - ▶ BCD nach INT / INT nach BCD usw.
 - ▶ Höherer Funktionen
 - ▶ CORDIC

Inhalt

Einführung

Motivation

Konvertierung von Zahlenformaten

Höhere Funktionen

Numerik

Ausblick

Motivation I

```
C:\Octave\OCTAVE-1.1\bin\octave-gui.exe

There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.

Octave was configured for "x86_64-w64-mingw32".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-involved.html

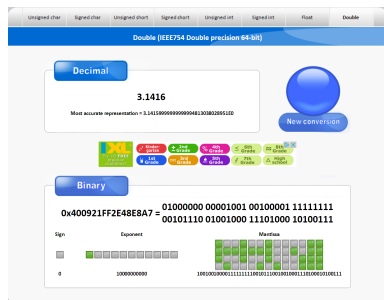
Read http://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

octave-1> pi
ans = 3.1416
octave-2> whos ans
Variables in the current scope:

  Attr Name      Size      Bytes  Class
  ----
  ans          1x1         8      double

Total is 1 element using 8 bytes

octave-3>
```



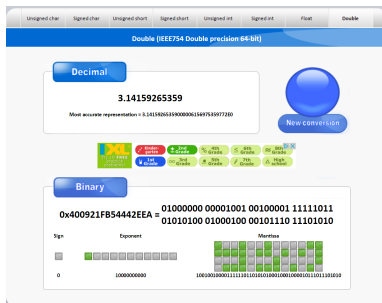
<http://www.binaryconvert.com/>

$$\pi_{\text{Octave, GUI}} = \{3, 146\}_{10} = \{400921FF2E48E8A7\}_{16}$$

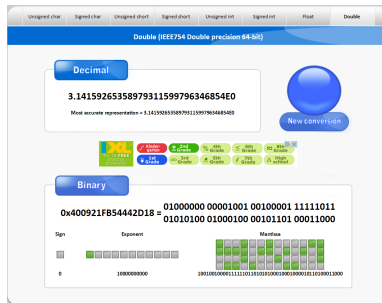
$$\begin{aligned}\pi_{\text{Octave, Docu}} &= \{3, 14159265358979311599796346854\ E0\}_{10} \\ &= \{400921FF2E48E8A7\}_{16}\end{aligned}$$

Motivation II

► π mit 11 Nachkommastellen



► π mit 29 Nachkommastellen



- IEEE Double \Rightarrow Ist das immer sinnvoll?
- Was mache ich in der Hardware?

Motivation III

Näherungen für die Zahl π :

► Leibniz-Reihe

$$\frac{\pi}{4} \approx \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} = 1 - \frac{1}{3} + \frac{1}{5} - \dots$$

► Näherungsbrüche

$$\pi \approx \frac{22}{7} = 3,1428\dots$$

Genauigkeit: 2 Nachkommastellen

$$\pi \approx \frac{355}{113} = 3,1415929\dots$$

Genauigkeit: 6 Nachkommastellen

$$\pi \approx \frac{103993}{33102} = 3,14159265301\dots$$

Genauigkeit: 9 Nachkommastellen

Inhalt

Einführung

Motivation

Konvertierung von Zahlenformaten

Höhere Funktionen

Numerik

Ausblick

Konvertierung von Zahlenformaten - Einführung I

C++ Datentypen für Ganzzahlen

Schreibweise	Typ	Anzahl Bits nach <i>data model</i>				
		C++ standard	LP32	ILP32	LLP64	LP64
<code>signed char</code>	<code>signed char</code>	mindestens 8	8	8	8	8
<code>unsigned char</code>	<code>unsigned char</code>	8	8	8	8	8
<code>short</code>	<code>short</code>	mindestens 16	16	16	16	16
<code>short int</code>						
<code>signed short</code>						
<code>signed short int</code>						
<code>unsigned short</code>						
<code>unsigned short int</code>	<code>unsigned short</code>					
<code>int</code>	<code>int</code>	mindestens 16	16	32	32	32
<code>signed</code>						
<code>signed int</code>						
<code>unsigned</code>	<code>unsigned</code>					
<code>unsigned int</code>						
<code>long</code>	<code>long</code>	mindestens 32	32	32	32	64
<code>long int</code>						
<code>signed long</code>						
<code>signed long int</code>						
<code>unsigned long</code>						
<code>unsigned long int</code>	<code>unsigned long</code>					
<code>long long</code>	<code>long long</code>	mindestens 64	64	64	64	64
<code>long long int</code>						
<code>signed long long</code>						
<code>signed long long int</code>						
<code>unsigned long long</code>						
<code>unsigned long long int</code>	<code>unsigned long long</code>					

Konvertierung von Zahlenformaten - Einführung II

C++ Datentypen für Gleitkommazahlen

Typ	Speicherplatz	Wertebereich	kleinste positive Zahl	Genauigkeit
float	4 Byte	$\pm 3,4 \cdot 10^{38}$	$1,2 \cdot 10^{-38}$	6 Stellen
double	8 Byte	$\pm 1,7 \cdot 10^{308}$	$2,3 \cdot 10^{-308}$	12 Stellen
long double	10 Byte	$\pm 1,1 \cdot 10^{4932}$	$3,4 \cdot 10^{-4932}$	18 Stellen

Quelle: [https:](https://scc.ustc.edu.cn/zlsc/czxt/200910/W020100308601263456982.pdf)

[//scc.ustc.edu.cn/zlsc/czxt/200910/W020100308601263456982.pdf](https://scc.ustc.edu.cn/zlsc/czxt/200910/W020100308601263456982.pdf)

Anmerkung: Zum Einsatz kommen verschiedene Datentypen (*data models*):
LP64 ILP64 LLP64 ILP32 LP32 (L: larger long longs, P: pointers)

Konvertierung von Zahlenformaten - Einführung - Einschub *Data Types* I

► 32-Bit und 64-Bit Datentypen

Data model	sizeof(int)	sizeof(long)	sizeof(long long)	sizeof(void*)	example
ILP32	32b	32b	64b	32b	Win32, i386
LP64	32b	64b	64b	64b	OSX, Linux
LLP64	32b	32b	64b	64b	x86-64 OSX, Linux Win64

	ILP32	LP64	LLP64	ILP64
char	8	8	8	8
short	16	16	16	16
int	32	32	32	64
long	32	64	32	64
long long	64	64	64	64
pointer	32	64	64	64

Anmerkung: Gegenstand der Informatikvorlesung

Konvertierung von Zahlenformaten - Einführung I

ASCII-Zeichentabelle, **hexadezimale** Nummerierung

Code	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0...	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1...	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2...	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5...	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6...	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7...	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

		R	e	c	h	n	e	r
HEX		52	65	63	68	6E	65	72

► siehe Termin 5

Konvertierung von Zahlenformaten - Einführung II

ASCII-Zeichentabelle, **hexadezimale** Nummerierung

Code	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0...	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1...	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2...	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5...	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6...	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7...	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

		0	0	0	1	9	6	7
HEX		30	30	30	31	39	36	37

31

H = 0011 0001

9

H = 0011 1001

6

H = 0011 0110

7

H = 0011 0111

Konvertierung von Zahlenformaten - Einführung III

ASCII-Zeichentabelle, **hexadezimale** Nummerierung

Code	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0...	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1...	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2...	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5...	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6...	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7...	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

3

31H =

0011 0001
MSHB LSHB

9

39H =

0011 1001

6

36H =

0011 0110

7

37H =

0011 0111

3963

=

0001 1001 0110 0111
Digit 3 Digit 2 Digit 1 Digit 0
MSByte LSByte

Konvertierung von Zahlenformaten - Einführung IV

31H	=	0011 0001
		<i>MSHB LSHB</i>
39H	=	0011 1001
36H	=	0011 0110
37H	=	0011 0111
3963	=	$\underbrace{\underbrace{0001 \cdot (10^3)}_{\text{Digit 3}} \underbrace{1001 \cdot (10^2)}_{\text{Digit 2}}}_{\text{MSByte}} \underbrace{\underbrace{0110 \cdot (10^1)}_{\text{Digit 1}} \underbrace{0111 \cdot (10^0)}_{\text{Digit 0}}}_{\text{LSByte}}$
Byte 1	=	$\underbrace{\underbrace{0001 \cdot (10^3)}_{\text{Digit 3}} \underbrace{1001 \cdot (10^2)}_{\text{Digit 2}}}_{\text{MSByte}}$
Byte 0	=	$\underbrace{\underbrace{0110 \cdot (10^1)}_{\text{Digit 1}} \underbrace{0111 \cdot (10^0)}_{\text{Digit 0}}}_{\text{LSByte}}$

Konvertierung von Zahlenformaten - Einführung V

ASCII-Zeichentabelle, **hexadezimale** Nummerierung

Code	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0...	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1...	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2...	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5...	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6...	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7...	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

		()	*	+	,	-	.	/
HEX		28	29	2A	2B	2C	2D	2E	2F
				:	;	<	=	>	?
HEX				3A	3B	3C	3D	3E	3F

Konvertierung von Zahlenformaten - (u)BCD \rightarrow INT I

► BCD ohne VZ

12-Bit pBCD $N_{BCD} \in \{0 \dots 999\}_{10}$

Digit 2				Digit 1				Digit 0			
d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0

3 Digits BCD $\begin{array}{c} 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \\ \boxed{d_3 \quad d_2 \quad d_1 \quad d_0} \end{array} (\cdot 10^2) \quad \begin{array}{c} 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \\ \boxed{d_3 \quad d_2 \quad d_1 \quad d_0} \end{array} (\cdot 10^1) \quad \begin{array}{c} 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \\ \boxed{d_3 \quad d_2 \quad d_1 \quad d_0} \end{array} (\cdot 10^0)$

10-Bit INT $N_{INT} \in \{0 \dots 1023\}_{10}$

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0

$$n\text{-Bits INT} \quad n_{INT} = \lceil \log_2 (N_{max,BCD}) \rceil$$

Konvertierung von Zahlenformaten - (u)BCD \rightarrow INT II

► BCD ohne VZ

12-Bit packed BCD

Digit 2				Digit 1				Digit 0			
d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
1	0	0	1	0	1	1	0	0	0	1	1

3 Digits BCD

2^3	2^2	2^1	2^0	$(\cdot 10^2)$	2^3	2^2	2^1	2^0	$(\cdot 10^1)$	2^3	2^2	2^1	2^0	$(\cdot 10^0)$
1	0	0	1		0	1	1	0		0	0	1	1	

10-Bit INT $N_{INT} \in \{0 \dots 1023\}_{10}$

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	1	1	0	0	0	0	1	1

$$N_{10} = 2^9 + 2^8 + 2^7 + 2^6 + 2^1 + 2^0 = 963$$

Konvertierung von Zahlenformaten - (u)BCD \rightarrow INT III

► BCD ohne VZ

12-Bit packed BCD

Digit 2				Digit 1				Digit 0			
d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
1	0	0	1	0	1	1	0	0	0	1	1

3 Digits BCD

$$\begin{array}{|c|c|c|c|} \hline 2^3 & 2^2 & 2^1 & 2^0 \\ \hline 1 & 0 & 0 & 1 \\ \hline \end{array} (\cdot 10^2) \begin{array}{|c|c|c|c|} \hline 2^3 & 2^2 & 2^1 & 2^0 \\ \hline 0 & 1 & 1 & 0 \\ \hline \end{array} (\cdot 10^1) \begin{array}{|c|c|c|c|} \hline 2^3 & 2^2 & 2^1 & 2^0 \\ \hline 0 & 0 & 1 & 1 \\ \hline \end{array} (\cdot 10^0)$$

$$N_{INT} = (Digit\ 2) \cdot 10^2 + (Digit\ 1) \cdot 10^1 + (Digit\ 0) \cdot 10^0$$

$$N_{INT} = (Digit\ 2) \cdot (1100100)_2 + (Digit\ 1) \cdot (1010)_2 + (Digit\ 0) \cdot 1_2$$

$$= \underbrace{(1001)_2 \cdot (1100100)_2}_{N'''} + \underbrace{(0110)_2 \cdot (1010)_2}_{N''} + \underbrace{(0011)_2 \cdot 1_2}_{N'}$$

Konvertierung von Zahlenformaten - (u)BCD \rightarrow INT IV

$$N_{INT} = \underbrace{(1001)_2 \cdot (1100100)_2}_{N'''} + \underbrace{(0110)_2 \cdot (1010)_2}_{N''} + \underbrace{(0011)_2 \cdot 1_2}_{N'}$$

	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	
$N_{0,D2}$							1	0	0	1	
							0	0	0	0	0 LSB
						0	0	0	0		0
					0	0	0	0			0
				1	0	0	1				1
			0	0	0	0					0
		0	0	0	0						0
	1	1	0	0	1						1
	1	0	0	1							1 MSB
N'''	1	1	1	0	0	0	0	1	0	0	$= 900_{10}$

Konvertierung von Zahlenformaten - (u)BCD \rightarrow INT V

$$N_{INT} = \underbrace{(1001)_2 \cdot (1100100)_2}_{N'''} + \underbrace{(0110)_2 \cdot (1010)_2}_{N''} + \underbrace{(0011)_2 \cdot 1_2}_{N'}$$

	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	
$N_{0,D1}$					0	1	1	0	
					0	0	0	0	0 LSB
				0	1	1	0		1
			0	0	0	0			0
		0	1	1	0				1 MSB
N''			1	1	1	1	0	0	$= 60_{10}$

Konvertierung von Zahlenformaten - (u)BCD \rightarrow INT VI

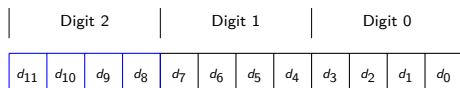
$$N_{INT} = \underbrace{(1001)_2 \cdot (1100100)_2}_{N'''} + \underbrace{(0110)_2 \cdot (1010)_2}_{N''} + \underbrace{(0011)_2 \cdot 1_2}_{N'}$$

	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	
$N_{0,D1}$					0	0	1	1	
					0	0	1	1	1
N'					0	0	1	1	$= 60_{10}$

Konvertierung von Zahlenformaten - (s)BCD \rightarrow INT I

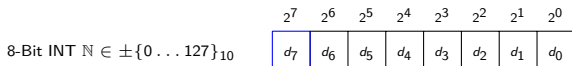
► BCD mit VZ

12-Bit pBCD $N \in \pm\{0 \dots 99\}$



3 Digits BCD

$$\begin{array}{c} 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \\ \boxed{d_3} \quad \boxed{d_2} \quad \boxed{d_1} \quad \boxed{d_0} \end{array} \left(\cdot (-1)^{d_{3(11)}} \right) \begin{array}{c} 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \\ \boxed{d_3} \quad \boxed{d_2} \quad \boxed{d_1} \quad \boxed{d_0} \end{array} \left(\cdot 10^1 \right) \begin{array}{c} 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \\ \boxed{d_3} \quad \boxed{d_2} \quad \boxed{d_1} \quad \boxed{d_0} \end{array} \left(\cdot 10^0 \right)$$











► Vorzeichen-Betrags-Darstellung

$$N_b = (-1)^{d_{n-1}} \cdot \sum_{i=0}^{n-2} d_i \cdot b^i = (-1)^{d_{n-1}} \cdot |N_b| \quad (\text{siehe Termin 5})$$

Konvertierung von Zahlenformaten - (s)BCD \rightarrow INT II

► 32-Bit packed BCD mit Vorzeichen (MS Tetrade)

d_{31}	d_{30}	d_{29}	d_{28}	d_{27}	d_{26}	d_{25}	d_{24}	d_{23}	d_{22}	d_{21}	d_{20}	d_{19}	d_{18}	d_{17}	d_{16}
															
1000: -VZ / 0000: + VZ				10^6 -Ziffern				10^5 -Ziffern				10^4 -Ziffern			
				D_6 (MSD)				D_5				D_4			
Byte 3								Byte 2							
d_{15}	d_{14}	d_{13}	d_{12}	d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
															
10^3 -Ziffern				10^2 -Ziffern				10^1 -Ziffern				10^0 -Ziffern			
D_3				D_2				D_1				D_0 (LSD)			
Byte 1								Byte 0							

► Zahlenbereich: $-9.999.999 \dots + 9.999.999$

Konvertierung von Zahlenformaten - (s)BCD \rightarrow INT III

► BCD mit VZ

12-Bit pBCD $N \in \pm\{0 \dots 99\}$

d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
0	0	0	0	0	1	1	0	0	0	1	1

Sign + 2 D.

2^3	2^2	2^1	2^0
0	0	0	0

 $\left(\cdot (-1)^{d_{3(11)}} \right)$

2^3	2^2	2^1	2^0
0	1	1	0

 $\left(\cdot 10^1 \right)$

2^3	2^2	2^1	2^0
0	0	1	1

 $\left(\cdot 10^0 \right)$

8-Bit INT $N \in \pm\{0 \dots 127\}_{10}$

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	0	0	0	1	1

$$N_{10} = (-1)^0 \cdot (2^6 + 2^2 + 2^1) = 67$$

Konvertierung von Zahlenformaten - (s)BCD \rightarrow INT IV

► BCD mit VZ

12-Bit pBCD $N \in \pm\{0 \dots 99\}$

d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
1	0	0	0	0	1	1	0	0	0	1	1

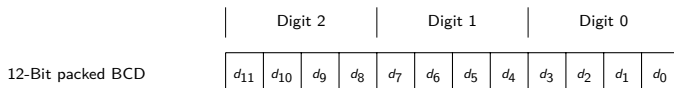
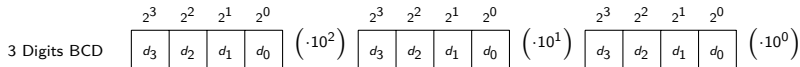
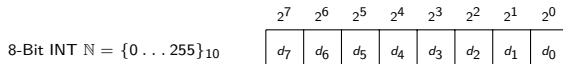
Sign + 2 D. $\begin{array}{c} 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \\ \boxed{1} \quad \boxed{0} \quad \boxed{0} \quad \boxed{0} \end{array} \left(\cdot (-1)^{d_{3(11)}} \right) \begin{array}{c} 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \\ \boxed{0} \quad \boxed{1} \quad \boxed{1} \quad \boxed{0} \end{array} \left(\cdot 10^1 \right) \begin{array}{c} 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \\ \boxed{0} \quad \boxed{0} \quad \boxed{1} \quad \boxed{1} \end{array} \left(\cdot 10^0 \right)$

8-Bit INT $N \in \pm\{0 \dots 127\}_{10}$ $\begin{array}{c} 2^7 \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \\ \boxed{1} \quad \boxed{1} \quad \boxed{0} \quad \boxed{0} \quad \boxed{0} \quad \boxed{0} \quad \boxed{1} \quad \boxed{1} \end{array}$

$$N_{10} = (-1)^1 \cdot (2^6 + 2^2 + 2^1) = -67$$

Konvertierung von Zahlenformaten - INT \rightarrow BCD I

► INT ohne VZ



$n =$

Konvertierung von Zahlenformaten - INT \rightarrow BCD II

► INT ohne VZ

10-Bit INT $\mathbb{N}_{INT} \in \{0 \dots 1023\}_{10}$

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	1	1	0	0	0	0	1	1

3 Digits BCD

2^3	2^2	2^1	2^0
1	0	0	1

 $(\cdot 10^2)$

2^3	2^2	2^1	2^0
0	1	1	0

 $(\cdot 10^1)$

2^3	2^2	2^1	2^0
0	0	1	1

 $(\cdot 10^0)$

12-Bit packed BCD

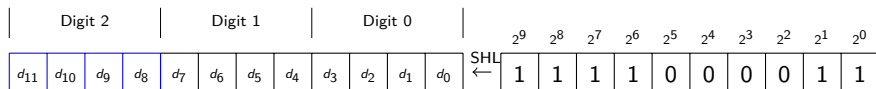
d_{11}	Digit 2 d_{10}	d_9	d_8	d_7	Digit 1 d_6	d_5	d_4	d_3	Digit 0 d_2	d_1	d_0
1	0	0	1	0	1	1	0	0	0	1	1

► Rechenweg: *shift-and-add-3 algorithm*

https://en.wikipedia.org/wiki/Double_dabble

Konvertierung von Zahlenformaten - INT \rightarrow BCD III

► Rechenweg: *shift-and-add-3 algorithm*



1. $SHL(1)$ = schiebe die n -stellige Binärzahl um eine Position nach links.
2. Wenn die 4-stellige Binärzahl innerhalb eines Digits $N_{Digit,x} > \{4\}_{10} = \{0100\}_2$, addiere zu dem Digit, x binär $\{3\}_{10} (= 0011)$ hinzu.
 - 2.1 Falls für mehrere Digits $N_{Digit,x} > \{4\}_{10} = \{0100\}_2$ gilt, addiere binär $\{3\}_{10} (= 0011)$ nacheinander hinzu, beginnend mit dem höchstwertigen Digit.
3. Wenn die Anzahl m der Schiebeoperation $SHL < n$ ist, gehe zu Schritt 1, andernfalls stoppe die Berechnung.

Konvertierung von Zahlenformaten - INT \rightarrow BCD IV

► Beispiel: $N_{INT} = \{11\ 1100\ 0011\}_2 = \{963\}_{10}$

d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0		d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	
0	0	0	0	0	0	0	0	0	0	0	0	INIT 1 SHL(1) 2 SHL(1) 3 SHL(1) ADD 3_{10}	1	1	1	1	0	0	0	0	1	1	
0	0	0	0	0	0	0	0	0	0	0	1		1	1	1	0	0	0	0	1	1		
0	0	0	0	0	0	0	0	0	0	0	1		1	1	1	0	0	0	0	1	1		
0	0	0	0	0	0	0	0	0	1	1	1		1	1	0	0	0	0	1	1			
								0	0	1	1												
0	0	0	0	0	0	0	0	1	0	1	0	4 SHL(1) ADD 3_{10}	1	0	0	0	0	1	1				
0	0	0	0	0	0	0	1	0	1	0	1		0	0	0	0	1	1					
								0	0	1	1												
0	0	0	0	0	0	0	1	1	0	0	0	5 SHL(1) 6 SHL(1) ADD 3_{10}	0	0	0	0	1	1					
0	0	0	0	0	0	1	1	0	0	0	0		0	0	0	1	1						
0	0	0	0	0	0	1	1	0	0	0	0		0	0	1	1							
								0	0	1	1												
0	0	0	0	0	1	0	0	0	0	0	0	7 SHL(1) 8 SHL(1) 9 SHL(1) ADD 3_{10} 10 SHL(1)	0	0	1	1							
0	0	0	1	0	0	1	0	0	0	0	0		0	1	1								
0	1	0	0	0	1	0	0	0	0	0	1		1										
								0	0	1	1												
0	1	0	0	0	1	0	1	0	0	0	1	1											
1	0	0	1	0	1	1	0	0	0	1	1												
9				6				3															

► Nachteil: Anzahl der Register!

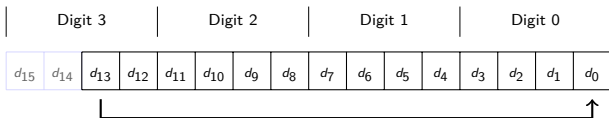
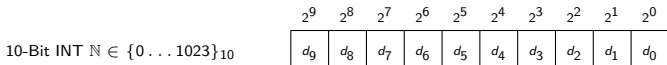
Konvertierung von Zahlenformaten - INT \rightarrow BCD V

► Rechenweg: Modifizierter *shift-and-add-3 algorithm*

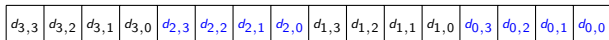
Digit 3 (10^3)				Digit 2 (10^2)				Digit 1 (10^1)				Digit 0 (10^0)				
d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	d_3	d_2	d_1	d_0	
		1	1	1	1	0	0	0	0	1	1	x	0	0	0	INIT
		1	1	1	0	0	0	0	1	1	x	0	0	0	1	1 SHL(1)
		1	1	0	0	0	0	1	1	x	0	0	0	1	1	2 SHL(1)
		1	0	0	0	0	1	1	x	0	0	0	1	1	1	3 SHL(1)
												0	0	1	1	ADD (3)
		1	0	0	0	0	1	1	x	0	0	1	0	1	0	
		0	0	0	0	1	1	x	0	0	1	0	1	0	1	4 SHL(1)
												0	0	1	1	ADD (3)
		0	0	0	0	1	1	x	0	0	1	1	0	0	0	
		0	0	0	1	1	x	0	0	1	1	0	0	0	0	5 SHL(1)
		0	0	1	1	x	0	0	1	1	0	0	0	0	0	6 SHL(1)
												0	0	1	1	ADD (3)
		0	0	1	1	x	0	1	0	0	1	0	0	0	0	
		0	1	1	x	0	1	0	0	1	0	0	0	0	0	7 SHL(1)
		1	1	x	0	1	0	0	1	0	0	0	0	0	0	8 SHL(1)
		1	x	0	1	0	0	1	0	0	0	0	0	0	1	9 SHL(1)
												0	0	1	1	ADD (3)
		1	x	0	1	0	0	1	0	1	1	0	0	0	1	
		x	0	1	0	0	1	0	1	1	0	0	0	1	1	10 SHL(1)
0	0	0	0	1	0	0	1	0	1	1	0	0	0	1	1	10 SHL(1)
+				9				6				3				

Konvertierung von Zahlenformaten - INT \rightarrow BCD VI

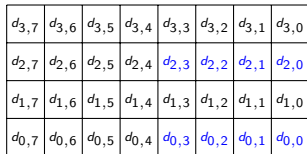
► Struktur: Modifizierter *shift-and-add-3 algorithm*



16-Bit pBCD $N \in \{0000 \dots 1023\}_{10}$



4 \times 8-Bit BCD



Konvertierung von Zahlenformaten - INT \rightarrow BCD VII

► Rechenweg: Modifizierter *shift-and-add-3 algorithm*

Digit 3 (10^3)				Digit 2 (10^2)				Digit 1 (10^1)				Digit 0 (10^0)				
d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	d_3	d_2	d_1	d_0	
		1	1	1	1	0	0	0	0	1	1	x	0	0	0	INIT
		1	1	1	0	0	0	0	1	1	x	0	0	0	0	1 ROL(1)
		1	1	0	0	0	0	1	1	x	0	0	0	1	1	2 ROL(1)
		1	0	0	0	0	1	1	x	0	0	0	1	1	1	3 ROL(1)
												0	0	1	1	ADD (3)
		1	0	0	0	0	1	1	x	0	0	1	0	1	0	4 ROL(1)
		0	0	0	0	1	1	x	0	0	1	0	1	0	1	ADD (3)
		0	0	0	0	1	1	x	0	0	1	1	0	0	0	5 ROL(1)
		0	0	0	1	1	x	0	0	1	1	0	0	0	0	6 ROL(1)
		0	0	1	1	x	0	0	1	1	0	0	0	0	0	ADD (3)
		0	0	1	1	x	0	1	0	0	1	0	0	0	0	7 ROL(1)
		0	1	1	x	0	1	0	1	0	0	0	0	0	0	8 ROL(1)
		1	1	x	0	1	0	1	0	0	0	0	0	0	0	9 ROL(1)
		1	x	0	1	0	0	1	0	0	1	0	0	0	1	ADD (3)
		1	x	0	1	0	0	1	0	1	1	0	0	0	1	10 ROL(1)
0	0	0	0	1	0	0	1	0	1	1	0	0	0	1	1	10 ROL(1)
+				9				6				3				

Konvertierung von Zahlenformaten - INT \rightarrow BCD VIII

- ▶ Modifizierter *shift-and-add-3 algorithm*

10-Bit INT $N_{INT} \in \{0 \dots 1023\}_{10}$

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	1	1	0	0	0	0	1	1

		Digit 3				Digit 2				Digit 1				Digit 0			
d_{15}	d_{14}	1	1	1	1	0	0	0	0	0	1	1	0	0	0	0	

16-Bit pBCD $N \in \{0000 \dots 1023\}_{10}$

0	0	0	0	1	0	0	1	0	1	1	0	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

4 x 8-Bit BCD





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

4 × 8-Bit ASCII

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

Konvertierung von Zahlenformaten - BCD \rightarrow FIXED POINT I

► BCD mit zwei Nachkommastellen

Vorkommastellen								Nachkommastellen									
d_{15}	d_{14}	d_{13}	d_{12}	d_{11}	d_{10}	d_9	d_8			d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0
																	
10^1 -Ziffer				10^0 -Ziffer						10^{-1} -Ziffer				10^{-2} -Ziffer			
D_3 (MSD)				D_2						D_1				D_0 (LSD)			
Byte 3								Byte 2									

- Vorkommastellen mittels „*shift-and-add-3 algorithm*“ umwandeln
- Nachkommastellen durch Division mit einer Festkomma-Konstanten \Rightarrow Genauigkeit
- Anmerkung: ASCII 78,69 $\rightarrow \{38\ 37\ 2C\ 36\ 39\}_{16}$ oder $\{2E\}_{16}$ für den ''

Konvertierung von Zahlenformaten - BCD \rightarrow FIXED POINT II

$$\begin{aligned} N_{FIX,BCD} &= (Digit\ 3) \cdot 10^1 + (Digit\ 2) \cdot 10^0 + (Digit\ 1) \cdot 10^{-1} + (Digit\ 0) \cdot 10^{-2} \\ &= (Digit\ 3) \cdot (1010)_2 + (Digit\ 2) \cdot (0001)_2 \\ &\quad + (Digit\ 1) \div (1010)_2 + (Digit\ 0) \div (110\ 0100)_2 \end{aligned}$$

$$N_2 = \underbrace{\sum_{i=0}^{n-1} d_i \cdot b^i}_{\text{Vorkommastellen}} + \underbrace{\sum_{j=1}^m d_j \cdot b^{-j}}_{\text{Nachkommastellen}}$$

- **Nachkommastellen:** Division mit einer Fixed Point Konstanten

Konvertierung von Zahlenformaten - BCD \rightarrow FIXED POINT III

Nachkommastellen

b^{-j}	0.x ₁₀	Summe $\sum_{j=1}^m d_j \cdot b^{-j}$
2^{-1}	0,5	0,5
2^{-2}	0,25	0,75
2^{-3}	0,125	0,875
2^{-4}	0,0625	0,9375
2^{-5}	0,03125	0,96875
2^{-6}	0,015625	0,984375
2^{-7}	0,0078125	0,9921875
2^{-8}	0,00390625	0,99609375
2^{-9}	0,001953125	0,998046875

Konvertierung von Zahlenformaten - BCD \rightarrow FIXED POINT IV

8 3 . 6 5

Dividend: $N_x = (0110)_2 = (6)_{10}$

Divisor: $N_y = (1010)_2 = (10)_{10}$

Quotient: $N_q = ?$

Nachkommastellen: 6

x_3	x_2	x_1	x_0	$\cdot x_{-1}$	x_{-2}	x_{-3}	x_{-4}	x_{-5}	x_{-6}		y_3	y_2	y_1	y_0		q_0	$\cdot q_{-1}$	q_{-2}	$\cdot q_{-3}$	q_{-4}	$\cdot q_{-5}$	q_{-6}
0	1	1	0	.0	0	0	0	0	0	\div	1	0	1	0	=	0	.1	0	0	1	1	0
-	1	0	1	0																		
				1	.0	0	0	0	0	0												
				-	1	0	1	0														
					1	1	0	0	0													
				-	1	0	1	0														
						1	0	0														
						1	0	1	0													
							1	0	0													

$$N_{BCD/10} = 0.6 \rightarrow N_{FIX} = (0.1001)_2 = 0.59375$$

Konvertierung von Zahlenformaten - BCD \rightarrow FIXED POINT V

8 4 . 6 5

Dividend $N_x = (0101)_2 = (5)_{10}$

Divisor $N_y = (110\ 0100)_2 = (100)_{10}$

Quotient $N_q = ?$

Nachkommastellen 6

$$N_{BCD/100} = 0.05 \rightarrow N_{FIX} = (0.000010)_2 = 0.03125$$

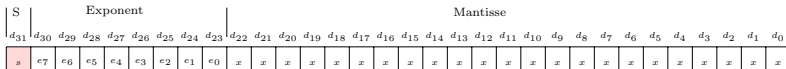
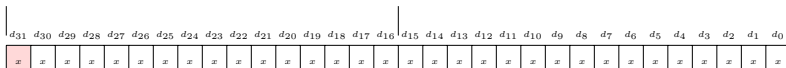
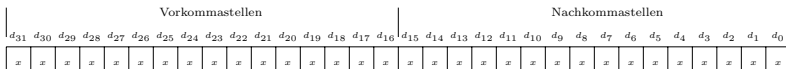
$$N_{BCD/10} = 0.60 \rightarrow N_{FIX} = (0.100110)_2 = 0.59375$$

$$N_{BCD} = 0.65 \rightarrow N_{FIX} = (0.101000)_2 = 0.625$$

$$\text{UQ7.6 } N_{BCD} = 87.65 \rightarrow N_{FIX} = (1010111.101000)_2 = 87.625$$

Konvertierung von Zahlenformaten - FIXPOINT \rightarrow FLOAT I

- Annahme: 32-Bit Repräsentation
- Q16.16 (vorzeichenbehaftet)
- UQ16.16
- IEEE754 Single Precision Floating Point



Konvertierung von Zahlenformaten - FIXPOINT → FLOAT II

$$\text{UQ16.16: } N_2 = \underbrace{\sum_{i=0}^{16-1} d_i \cdot b^i}_{\text{Vorkommastellen}} + \underbrace{\sum_{j=1}^{16} d_j \cdot b^{-j}}_{\text{Nachkommastellen}}$$

d_i : ganzzahliger Koeffizient (Ziffer) Vorkommastellen, $d_i \in \{0, 1\}$

d_j : ganzzahliger Koeffizient (Ziffer) Nachkommastellen, $d_j \in \{0, 1\}$

b : ganzzahlige Basis, $b = 2$

n : Stellenzahl

m : Nachkommastelle

N_2 : Wert der n-stelligen Zahl zur Basis $b=2$

- ▶ Wertebereich: $0, \dots, 65.535, 9999847412109375$
- ▶ 4.2950e+009 Zahlenwerte

Konvertierung von Zahlenformaten - FIXPOINT → FLOAT III

Vorkommastellen			Nachkommastellen		
b^i	\times_{10}	$\sum_{i=1}^{n-1} d_i \cdot b^{-i}$	b^{-j}	$0.\times_{10}$	$\sum_{j=1}^m d_j \cdot b^{-j}$
2^0	1	1	2^{-1}	0,5	0,5
2^1	2	3	2^{-2}	0,25	0,75
2^2	4	7	2^{-3}	0,125	0,875
2^3	8	15	2^{-4}	0,0625	0,9375
2^4	16	31	2^{-5}	0,03125	0,96875
2^5	32	63	2^{-6}	0,015625	0,984375
2^6	64	127	2^{-7}	0,0078125	0,9921875
2^7	128	255	2^{-8}	0,00390625	0,99609375
2^8	256	511	2^{-9}	0,001953125	0,998046875
2^9	512	1023	2^{-10}	0,0009765625	0,9990234375
2^{10}	1024	2047	2^{-11}	0,00048828125	0,99951171875
2^{11}	2048	4095	2^{-12}	0,000244140625	0,999755859375
2^{12}	4096	8191	2^{-13}	0,0001220703125	0,9998779296875
2^{13}	8192	16383	2^{-14}	0,00006103515625	0,99993896484375
2^{14}	16384	32767	2^{-15}	0,000030517578125	0,999969482421875
2^{15}	32768	65535	2^{-16}	0,0000152587890625	0,9999847412109375

Konvertierung von Zahlenformaten - FIXPOINT \rightarrow FLOAT IV

$$\text{Q16.16: } N_2 = \underbrace{-d_{15} \cdot b^{15} + \sum_{i=0}^{16-2} d_i \cdot b^i}_{\text{Vorkommastellen}} + \underbrace{\sum_{j=1}^{16} d_j \cdot b^{-j}}_{\text{Nachkommastellen}}$$

d_i : ganzzahliger Koeffizient (Ziffer) Vorkommastellen, $d_i \in \{0, 1\}$

d_j : ganzzahliger Koeffizient (Ziffer) Nachkommastellen, $d_j \in \{0, 1\}$

b : ganzzahlige Basis, $b = 2$

n : Stellenzahl

m : Nachkommastelle

N_2 : Wert der n-stelligen Zahl zur Basis $b=2$

- ▶ Wertebereich: $-32.768, \dots, 32.767, 9999847412109375$
- ▶ 4.2950e+009 Zahlenwerte

Konvertierung von Zahlenformaten - FIXPOINT → FLOAT V

Vorkommastellen			Nachkommastellen		
b^i	\times_{10}	$\sum_{i=1}^{n-1} d_i \cdot b^{-i}$	b^{-j}	$0.\times_{10}$	$\sum_{j=1}^m d_j \cdot b^{-j}$
2^0	1	1	2^{-1}	0, 5	0, 5
2^1	2	3	2^{-2}	0, 25	0, 75
2^2	4	7	2^{-3}	0, 125	0, 875
2^3	8	15	2^{-4}	0, 0625	0, 9375
2^4	16	31	2^{-5}	0, 03125	0, 96875
2^5	32	63	2^{-6}	0, 015625	0, 984375
2^6	64	127	2^{-7}	0, 0078125	0, 9921875
2^7	128	255	2^{-8}	0, 00390625	0, 99609375
2^8	256	511	2^{-9}	0, 001953125	0, 998046875
2^9	512	1023	2^{-10}	0, 0009765625	0, 9990234375
2^{10}	1024	2047	2^{-11}	0, 00048828125	0, 99951171875
2^{11}	2048	4095	2^{-12}	0, 000244140625	0, 999755859375
2^{12}	4096	8191	2^{-13}	0, 0001220703125	0, 9998779296875
2^{13}	8192	16383	2^{-14}	0, 00006103515625	0, 99993896484375
2^{14}	16384	32767	2^{-15}	0, 000030517578125	0, 999969482421875
-2^{15}	-32768	-1	2^{-16}	0, 0000152587890625	0, 9999847412109375

Konvertierung von Zahlenformaten - FIXPOINT → FLOAT VI

$$\text{SP: } N_{IEEE\ SP,10} = (-1)^{\text{sign}} \cdot 2^{(E-127)} \cdot \left(1 + \sum_{i=1}^{23} d_{(23-i)} \cdot 2^{-i} \right)$$

d_i : ganzzahliger Koeffizient (Ziffer) Vorkommastellen, $d_i \in \{0, 1\}$

d_j : ganzzahliger Koeffizient (Ziffer) Nachkommastellen, $d_j \in \{0, 1\}$

b : ganzzahlige Basis, $b = 2$

n : Stellenzahl

m : Nachkommastelle

N_2 : Wert der n-stelligen Zahl zur Basis $b=2$

- ▶ Wertebereich: $-32.768, \dots, 32.767, 9999847412109375$
- ▶ 4.2950e+009 Zahlenwerte

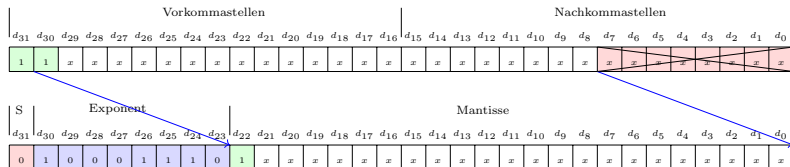
Konvertierung von Zahlenformaten - FIXPOINT \rightarrow FLOAT VII

$$N_{IEEE\ SP,10} = (-1)^{\text{sign}} \cdot 2^{(E-127)} \cdot \left(1 + \sum_{i=1}^{23} b_{(23-i)} \cdot 2^{-i} \right)$$

$$N_{IEEE\ DP,10} = (-1)^{\text{sign}} \cdot 2^{(E-1023)} \cdot \left(1 + \sum_{i=1}^{52} b_{(52-i)} \cdot 2^{-i} \right)$$

Konvertierung von Zahlenformaten - FIXPOINT \rightarrow FLOAT - UQ16.16 I

► UQ16.16 \rightarrow FLOAT SP



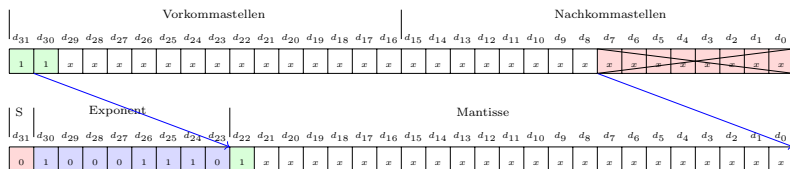
- Bit d_{31} wird als **hidden one** interpretiert.
- Bits $d_{30} \dots d_0$ werden um 15 Stellen nach rechts verschoben $\{\text{SHR}(15)\}$
- UQ16.16 Bits $d_{30} \dots d_8$ bilden die 23-Bit Mantisse
- **Truncation** UQ16.16 Bits $d_7 \dots d_0$
- FLOAT Exponent = Bias + 15 = 142 = $\{1000\ 1110\}_2$

$$N = 1 \cdot 2^{15} + 1 \cdot 2^{14} = 49.152 = 1,5 \cdot 2^{15}$$

Konvertierung von Zahlenformaten - FIXPOINT \rightarrow FLOAT - UQ16.16 II

N_2 : 1100 0000 0000 0000, 0000 0000 $\cdot 2^{(01111111-01111111)}$
1100 0000 0000 000, 0 0000 0000 $\cdot 2^{(10000000-01111111)}$
1100 0000 0000 00, 00 0000 0000 $\cdot 2^{(10000001-01111111)}$
1100 0000 0000 0, 000 0000 0000 $\cdot 2^{(10000010-01111111)}$
1100 0000 0000 , 0000 0000 0000 $\cdot 2^{(10000011-01111111)}$
1100 0000 000, 0 0000 0000 0000 $\cdot 2^{(10000100-01111111)}$
1100 0000 00, 00 0000 0000 0000 $\cdot 2^{(10000101-01111111)}$
1100 0000 0, 000 0000 0000 0000 $\cdot 2^{(10000110-01111111)}$
1100 0000 , 0000 0000 0000 0000 $\cdot 2^{(10000111-01111111)}$
1100 000, 0 0000 0000 0000 0000 $\cdot 2^{(10001000-01111111)}$
1100 00, 00 0000 0000 0000 0000 $\cdot 2^{(10001001-01111111)}$
1100 0, 000 0000 0000 0000 0000 $\cdot 2^{(10001010-01111111)}$
1100 , 0000 0000 0000 0000 0000 $\cdot 2^{(10001011-01111111)}$
110, 0 0000 0000 0000 0000 0000 $\cdot 2^{(10001100-01111111)}$
11, 00 0000 0000 0000 0000 0000 $\cdot 2^{(10001101-01111111)}$
1, 100 0000 0000 0000 0000 0000 $\cdot 2^{(10001110-01111111)}$

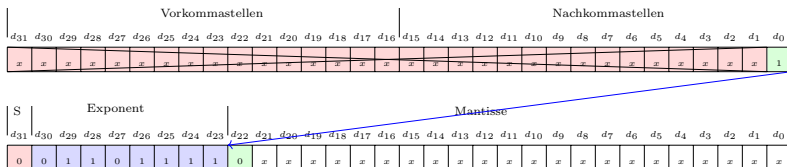
Konvertierung von Zahlenformaten - FIXPOINT → FLOAT - UQ16.16 III



$$1, \underbrace{100\ 0000\ 0000\ 0000\ 0000\ 0000}_{\text{Mantisse}} \dots \cdot 2^{\underbrace{(10001110 - 01111111)}_{\text{Exponent}}} \quad \text{Bias} = \{127\}_{10}$$

Konvertierung von Zahlenformaten - FIXPOINT → FLOAT - UQ16.16 IV

► UQ16.16 → FLOAT SP



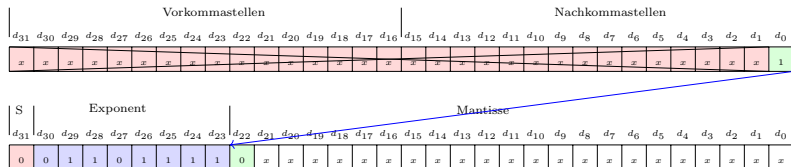
- Bit d_0 wird als **hidden one** interpretiert.
- Bit d_0 wird um 16 Stellen nach links verschoben $\{\text{SHL}(16)\}$
- Die 23-Bit Mantisse wird mit '0' aufgefüllt
- $\text{FLOAT Exponent} = \text{Bias} - 16 = 111 = \{01101111\}_2$

$$N = 1 \cdot 2^{-16} = 1,5259E - 5$$

Konvertierung von Zahlenformaten - FIXPOINT \rightarrow FLOAT - UQ16.16 V

$$\begin{aligned}
N_2 : \quad & \dots 0000, 0000 0000 0000 0001 \dots \cdot 2^{(01111111-01111111)} \\
& \dots 0000 0, 000 0000 0000 0001 \dots \cdot 2^{(01111110-01111111)} \\
& \dots 0000 00, 00 0000 0000 0001 \dots \cdot 2^{(01111101-01111111)} \\
& \dots 0000 000, 0 0000 0000 0001 \dots \cdot 2^{(01111100-01111111)} \\
& \dots 0000 0000, 0000 0000 0001 \dots \cdot 2^{(01111011-01111111)} \\
& \dots 0000 0000 0, 000 0000 0001 \dots \cdot 2^{(01111010-01111111)} \\
& \dots 0000 0000 00, 00 0000 0001 \dots \cdot 2^{(01111001-01111111)} \\
& \dots 0000 0000 000, 0 0000 0001 \dots \cdot 2^{(01111000-01111111)} \\
& \dots 0000 0000 0000, 0000 0001 \dots \cdot 2^{(01111011-01111111)} \\
& \dots 0000 0000 0000 0, 000 0001 \dots \cdot 2^{(01111010-01111111)} \\
& \qquad \qquad \qquad \vdots \\
& \dots 0000 0000 0000 0000 00, 01 \dots \cdot 2^{(01110001-01111111)} \\
& \dots 0000 0000 0000 0000 000, 1 \dots \cdot 2^{(01110000-01111111)} \\
& \dots 0000 0000 0000 0000 0001, \dots \cdot 2^{(01101111-01111111)}
\end{aligned}$$

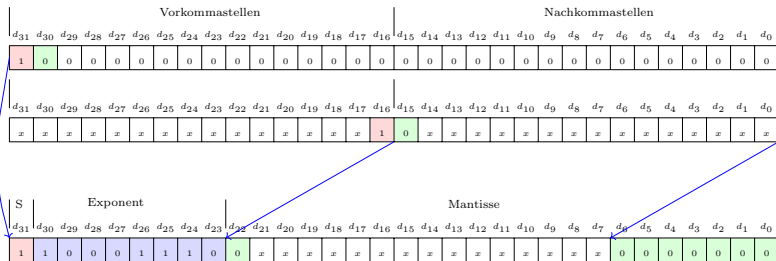
Konvertierung von Zahlenformaten - FIXPOINT \rightarrow FLOAT - UQ16.16 VI



$$1, \underbrace{0000\ 0000\ 0000\ 0000\ 0000\ 0000}_{\text{Mantisse}} \dots \cdot 2^{\underbrace{(01101111 - 01111111)}_{\text{Exponent}}} \quad \text{Bias} = \{127\}_{10}$$

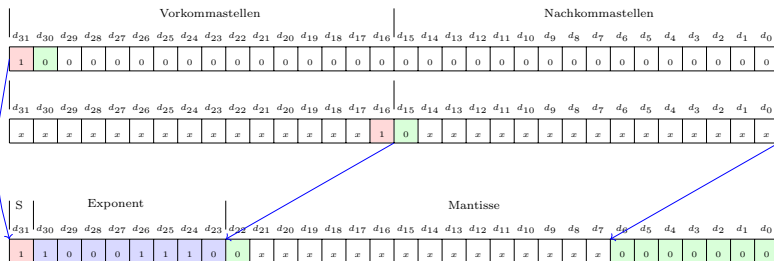
Konvertierung von Zahlenformaten - FIXPOINT \rightarrow FLOAT - Q16.16 I

- Q16.16 \rightarrow FLOAT SP, $N = -2^{15} = -32.768$



- Sonderfall:
 $N = -32.768 = \{1000\ 0000\ 0000\ 0000\ .\ 0000\ 0000\ 0000\ 0000\}$
- Detektion des Stellenwertes $d_{31} \rightarrow$ Vorzeichen
- Detektion der Stellenwerte $d_{30} \dots d_0 \rightarrow$ NULL
- Exponent: Addition $+15 \rightarrow$ Exponent $= 127+15= 142$

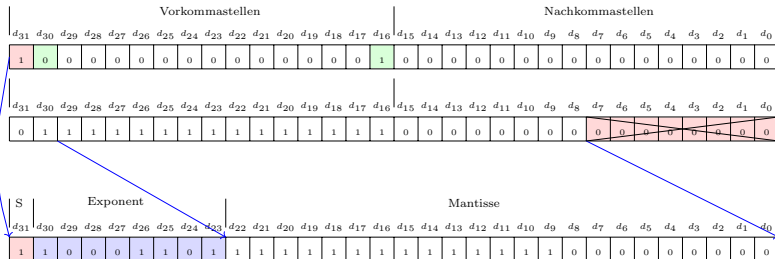
Konvertierung von Zahlenformaten - FIXPOINT → FLOAT - Q16.16 II



$$1, \underbrace{000\ 0000\ 0000\ 0000\ 0000\ 0000}_{\text{Mantisse}} \dots \cdot 2^{\underbrace{(1000\ 1110) - (01111111)}_{\text{Exponent}}} \quad \text{Bias} = \{127\}_{10}$$

Konvertierung von Zahlenformaten - FIXPOINT \rightarrow FLOAT - Q16.16 III

- Q16.16 \rightarrow FLOAT SP, $N = -2^{15} + 2^0 = -32767$



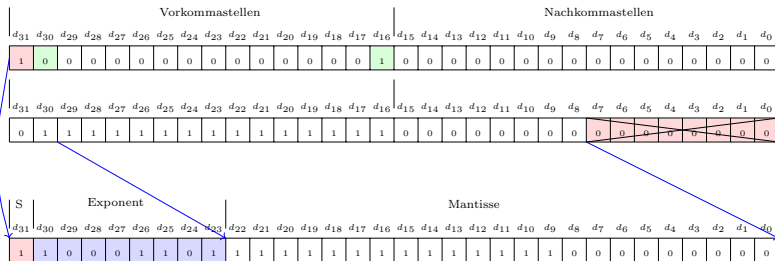
- Detektion der „führenden“ 1 \rightarrow Stellenwert d_{16}
- Stellenwert d_{31} definiert das Vorzeichen
- 2er-Komplement bilden und „führenden“ 1 detektieren

Konvertierung von Zahlenformaten - FIXPOINT \rightarrow FLOAT - Q16.16 IV

$$\begin{aligned} N_2 : & \dots 1000\ 0000\ 0000\ 0001, 0000 \dots \\ N'_2 : & \dots 0111\ 1111\ 1111\ 1111, 0000 \dots \\ N'_2 : & \dots 0111\ 1111\ 1111\ 1111, 0000 \dots \cdot 2^{(01111111 - 01111111)} \\ & \dots 0111\ 1111\ 1111\ 111, 1\ 0000 \dots \cdot 2^{(10000000 - 01111111)} \\ & \dots 0111\ 1111\ 1111\ 11, 11\ 0000 \dots \cdot 2^{(10000001 - 01111111)} \\ & \dots 0111\ 1111\ 1111\ 1, 111\ 0000 \dots \cdot 2^{(10000010 - 01111111)} \\ & \dots 0111\ 1111\ 1111, 1111\ 0000 \dots \cdot 2^{(10000011 - 01111111)} \\ & \dots 0111\ 1111\ 111, 1\ 1111\ 0000 \dots \cdot 2^{(10000100 - 01111111)} \\ & \dots 0111\ 1111\ 11, 11\ 1111\ 0000 \dots \cdot 2^{(10000101 - 01111111)} \\ & \dots 0111\ 1111\ 1, 111\ 1111\ 0000 \dots \cdot 2^{(10000110 - 01111111)} \\ & \vdots \\ & \vdots \\ & \dots 0111, 1111\ 1111\ 1111\ 0000 \dots \cdot 2^{(10001011 - 01111111)} \\ & \dots 011, 1\ 1111\ 1111\ 1111\ 0000 \dots \cdot 2^{(10001100 - 01111111)} \\ & \dots 01, 11\ 1111\ 1111\ 1111\ 0000 \dots \cdot 2^{(10001101 - 01111111)} \end{aligned}$$

Konvertierung von Zahlenformaten - FIXPOINT \rightarrow FLOAT - Q16.16 V

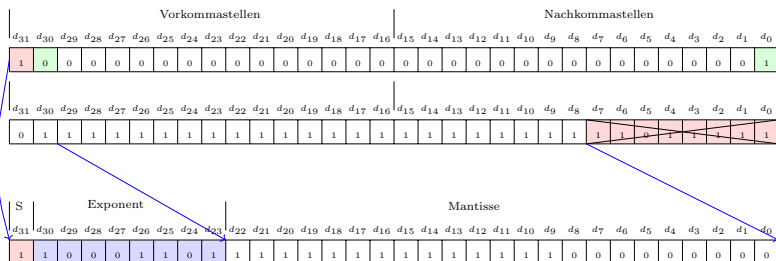
► Q16.16 \rightarrow FLOAT SP, $N = -2^{15} + 2^0 = -32767$



$$1, \underbrace{111\ 1111\ 1111\ 1110\ 0000\ 0000}_{\text{Mantisse}} \dots \cdot 2^{\underbrace{(1000\ 1101)_{\text{Exponent}} - \underbrace{(01111111)_{\text{Bias}=\{127\}_{10}}}}_{\text{Exponent}}}$$

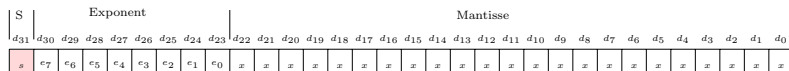
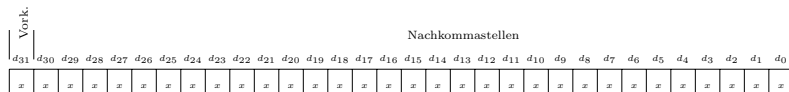
Konvertierung von Zahlenformaten - FIXPOINT \rightarrow FLOAT - Q16.16 VI

► Q16.16 \rightarrow FLOAT SP, $N = -2^{15} + 2^{-16} \approx -32767$



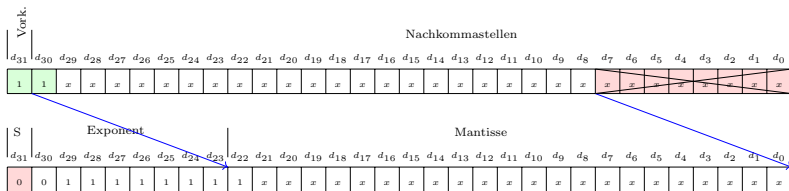
$$\epsilon_{max} = N_{FIX} - N_{FLOAT} = \sum_{j=9}^{16} 2^{-j} = 0,0038910$$

Konvertierung von Zahlenformaten - FIXPOINT \rightarrow FLOAT - UQ1.31 I



Konvertierung von Zahlenformaten - FIXPOINT \rightarrow FLOAT - UQ1.31 II

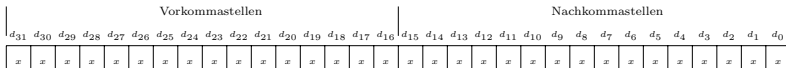
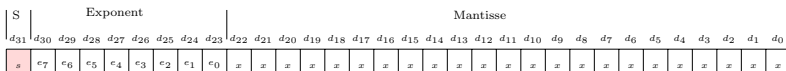
► Q1.31 \rightarrow FLOAT SP, $N = 2^0 + 2^{-1} = 1.5$



$$\epsilon_{max} = N_{FIX} - N_{FLOAT} = \sum_{j=24}^{31} 2^{-j} = 1,898E-7$$

Konvertierung von Zahlenformaten - FLOAT \rightarrow FIXPOINT I

- ▶ Annahme: 32-Bit Repräsentation
- ▶ IEEE754 Single Precision Floating Point
- ▶ Q16.16 (vorzeichenbehaftet)
- ▶ UQ16.16



Inhalt

Einführung

Motivation

Konvertierung von Zahlenformaten

Höhere Funktionen

Numerik

Ausblick

Höhere Funktionen - FLOAT \rightarrow FIXPOINT I

Wie werden höhere Funktionen in einem Rechenwerk realisiert?

- ▶ \sqrt{x}
- ▶ $\sin x$
- ▶ $\cos x$
- ▶ $\ln x$
- ▶ ...

- ▶ ANSI/IEEE Standard

Wurzelfunktion (Beispiel: $q = \sqrt{95241}$)

Radikand:
$$z = \sum_{i=0}^{n-1} z_i \cdot 10^i = z_{n-1} \cdot 10^{n-1} + \dots + z_0 \cdot 10^0$$

Quadratwurzel:
$$q = \sum_{i=0}^{n-1} q_i \cdot 10^i = q_{n-1} \cdot 10^{n-1} + \dots + q_0 \cdot 10^0$$

Restwert:
$$r = z - q^2 = \sum_{i=0}^{n-1} r_i \cdot 10^i = r_{n-1} \cdot 10^{n-1} + \dots + r_0 \cdot 10^0$$

- ▶ $q = \sqrt{x - r}$
- ▶ $0 \leq r \leq 2q$: Wertebereich für den Restwert

Höhere Funktionen - Wurzelfunktion - Schulmethode II

	q_2	q_1	q_0	$=q$	$q^{(0)} = 0$		
SQRT	(9	5	2	4	1)	$=z$	$q^{(1)} = 3$
	9						
	0	5	2			$(2 \cdot q^{(1)} \cdot 10 + 4) \cdot 4 = 256 \leq 52$	$q^{(2)} = 30$
		0	0				
		5	2	4	1	$(2 \cdot q^{(2)} \cdot 10 + 2 \cdot 4) \cdot 2 \cdot 4 = 4864 \leq 5241$	$q^{(3)} = 308$
		4	8	6	4		
		3	7	7		$r = 377$	$q = 308$

Probe: $z = q^2 + r$

$$= (308)^2 + 377 = 95.241$$

Höhere Funktionen - Wurzelfunktion - Schulmethode III

Die Funktion der Schulmethode lässt sich an der trinomischen Formel erläutern:

$$(a + b + c)^2 = a^2 + b^2 + c^2 + 2ab + 2ac + 2cb$$

$$\sqrt{a^2 + b^2 + c^2 + 2ab + 2ac + 2cb} = (a + b + c) = q$$

SQRT

$$a \cdot a =$$

$$(a^2 + b^2 + c^2 + 2ab + 2ac + 2bc) = a + b + c$$

$$a^2$$

$$\begin{array}{r} 0 \quad b^2 \quad +2ab \\ \hline \end{array}$$

$$(b + 2a) \cdot b =$$

$$b^2 \quad +2ab$$

$$\begin{array}{r} c^2 \quad 0 \quad +2ac \quad +2bc \\ \hline \end{array}$$

$$[2(a + b) + c] \cdot c =$$

$$c^2 \quad 0 \quad +2ac \quad +2bc$$

$$\begin{array}{r} 0 \quad 0 \quad 0 \quad 0 \\ \hline \end{array}$$

Höhere Funktionen - Wurzelfunktion - Schulmethode IV

$$\sqrt{95241} = \underbrace{308}_q + \underbrace{377}_r$$

$$q = a + b + c = 300 + 0 + 8$$

$$z = q^2 + r$$

$$= (a + b + c)^2 + r$$

$$= a^2 + b^2 + c^2 + 2ab + 2ac + 2bc + r$$

$$= 90.000 + 0 + 64 + 0 + 4.800 + 0 + 377$$

$$= 95.241$$

Höhere Funktionen - Wurzelfunktion - Schulmethode V

► Wurzel aus einer binären vorzeichenlose Ganzzahl (unsigned Integer)

	q_3	q_2	q_1	q_0	q		$q^{(0)} = 0$
SQRT	(0 1	1 1	0 1	1 0)	$z = (118)_{10}$	$q_3 = 1$	$q^{(1)} = 1$
	0 1						
	0 0	1 1			$\geq \underline{101}$ nein	$q_2 = 0$	$q^{(2)} = 10$
	0	0 0					
	0	1 1	0 1		$\geq \underline{1001}$ ja	$q_1 = 1$	$q^{(3)} = 101$
		1 0	0 1				
		0 1	0 0	1 0	$\geq \underline{10101}$ nein	$q_0 = 0$	$q^{(4)} = 1010$
		0 0	0 0	0 0			
		0 1	0 0	1 0	$r = (18)_{10}$ nein	$q = (1010)_2$	$= (10)_{10}$

Höhere Funktionen - Wurzelfunktion - Schulmethode VI

- Wurzel aus *unsigned Integer* in DOT-Notation

	q_3	q_2	q_1	q_0	q
SQRT	(• •	• •	• •	• •)	z
	• •				$-q_3 (q^{(0)} 0 \ q_3) 2^6$
	•	• •			$-q_2 (q^{(1)} 0 \ q_2) 2^4$
		• •	• •		$-q_1 (q^{(2)} 0 \ q_1) 2^2$
		•	• •	• •	$-q_0 (q^{(3)} 0 \ q_0) 2^0$
		•	• •	• •	r

Inhalt

Einführung

Motivation

Konvertierung von Zahlenformaten

Höhere Funktionen

Numerik

Ausblick

Was ist **Arithmetik**?

Die Arithmetik („die Zahlenmäßige [Kunst]“) umfasst das Rechnen mit den natürlichen Zahlen, vor allem mit den **Grundrechenarten** wie der Addition, Subtraktion, Multiplikation und Division. Die Arithmetik kann als Teil der Algebra verstanden und leitet zur Zahlentheorie.

Was ist **Numerik**?

Die Numerik genannt, beschäftigt sich als Teilgebiet der Mathematik mit der Konstruktion und Analyse von **Algorithmen** für kontinuierliche mathematische Probleme. Hauptanwendung ist dabei die approximative Berechnung von Lösungen mit Hilfe von Computern.

- ▶ Ein in der Praxis häufig auftretendes Problem ist die Berechnung von Funktionswerten.
 - ▶ Standardfunktionen wie \sqrt{x} , $\log x$, $\cos x$ etc. werden als Algorithmen implementiert
 - ▶ Näherungen für **physikalische Konstante**

Numerik - Beispiel I

Berechnung von $y = e^{-5,5} = 4,086771438 \cdot 10^{-3}$ als GP mit Mantissenlänge $n = 5$

$$y = e^x \approx \sum_{i=0}^k \frac{x^i}{i!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

Numerik - Beispiel II

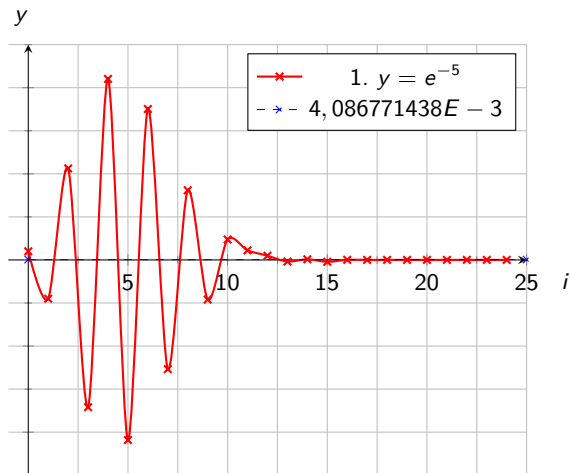
$$y = e^{-5,5} \approx \sum_{i=0}^k \frac{x^i}{i!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

i_{10}	$e^{-5,5}$	$\frac{x^i}{i!}$	$s_k = \sum_{i=0}^k \frac{x^i}{i!}$	
0	=	1,0000	+	1,0000
1	-	5,5000	-	4,5000
2	+	15,125	+	10,625
3	-	27,729	-	17,104
4	+	38,127	+	21,023
5	-	41,940	-	20,917
6	+	38,445	+	17,528
7	-	30,207	-	12,679
8	+	20,767	+	8,088
9	-	12,691	-	4,603
\vdots		\vdots		\vdots

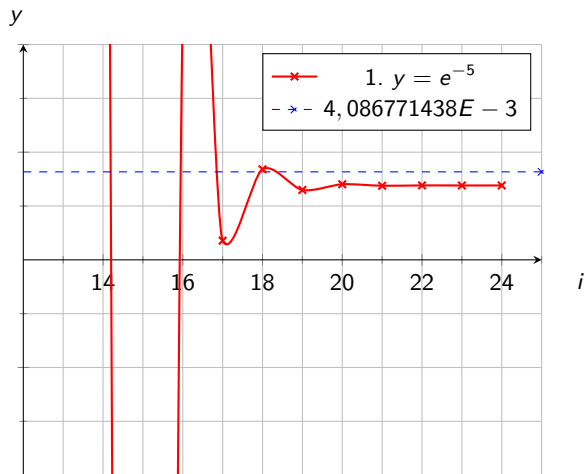
Numerik - Beispiel III

i_{10}	$e^{-5,5}$	$\frac{x^i}{i!}$	$s_k = \sum_{i=0}^k \frac{x^i}{i!}$	
0	=	1,0000	+	1,0000
1	-	5,5000	-	4,5000
2	+	15,125	+	10,625
⋮		⋮		⋮
10	+	6,9801	+	2,3771
11	-	3,4901	-	1,1130
12	+	1,5996	+	0,48660
13	-	0,67676	-	0,19016
⋮		⋮		⋮
20	+	$2,6372 \cdot 10^{-4}$	+	$3,2459 \cdot 10^{-3}$
21	-	$6,9068 \cdot 10^{-5}$	+	$3,5096 \cdot 10^{-3}$
22	+	$1,7267 \cdot 10^{-5}$	+	$3,4578 \cdot 10^{-3}$
23	-	$4,1291 \cdot 10^{-6}$	+	$3,4537 \cdot 10^{-3}$
24	+	$9,4627 \cdot 10^{-7}$	+	$3,4546 \cdot 10^{-3}$
⋮		⋮		⋮

Numerik - Beispiel IV



Numerik - Beispiel V



Numerik - Beispiel VI

- Vergleich der Ergebnisse:

$$\text{exakt: } y = e^{-5,5} = 4,086771438 \cdot 10^{-3}$$

$$\text{Näherung: } y = e_{GP}^{-5,5} = 3,4546 \cdot 10^{-3}$$

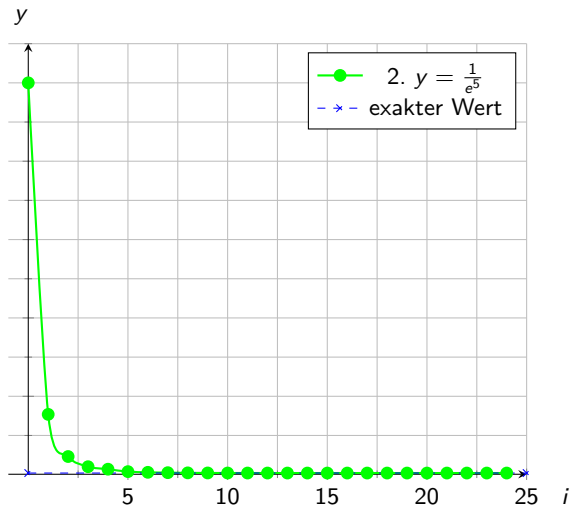
- Approximation und exaktes Ergebnis stimmen in keiner Stelle der Mantisse überein
- Auslöschung
- Alternativer Ansatz: $y = e^{-x} = \frac{1}{e^x} = \frac{1}{\sum_{i=0}^k \frac{x^i}{i!}}$

Numerik - Beispiel VII

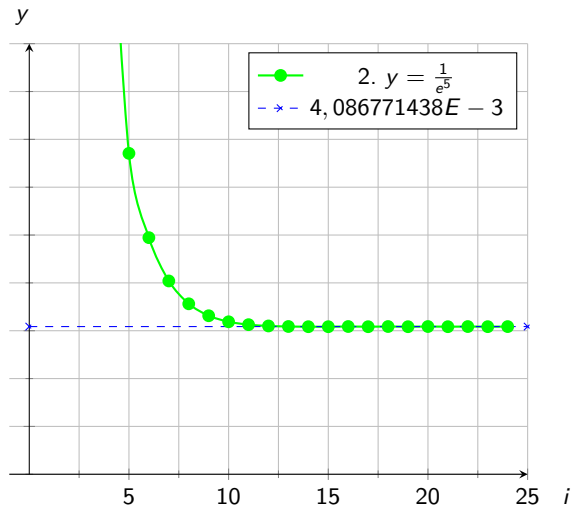
Beispiel: Berechnung von $y = e^{-5,5} = \frac{1}{e^{5,5}} = \frac{1}{244,6919323} = 4,086771438 \cdot 10^{-3}$
als GP mit Mantissenlänge $n = 5$

i_{10}	$\frac{1}{e^{5,5}}$	$\frac{x^i}{i!}$	$s_k = \sum_{i=0}^k \frac{x^i}{i!}$	$\frac{1}{s_k}$	
0	=	1,0000	+	1,0000	1
1	+	5,5000	+	6,5000	$1,5384 \cdot 10^{-1}$
2	+	15,125	+	21,625	$4,6242 \cdot 10^{-2}$
3	+	27,729	+	49,354	$2,0261 \cdot 10^{-2}$
4	+	38,127	+	87,809	$1,3883 \cdot 10^{-2}$
5	+	41,940	+	129,74	$7,7077 \cdot 10^{-3}$
6	+	38,445	+	168,19	$5,9456 \cdot 10^{-3}$
7	+	30,207	+	198,40	$5,0403 \cdot 10^{-3}$
8	+	20,767	+	219,16	$4,5627 \cdot 10^{-3}$
9	+	12,691	+	231,85	$4,3131 \cdot 10^{-3}$
10	+	6,9801	+	238,83	$4,1870 \cdot 10^{-3}$
11	+	3,4901	+	242,32	$4,1267 \cdot 10^{-3}$
12	+	1,5996	+	243,92	$4,0997 \cdot 10^{-3}$
13	+	$6,7676 \cdot 10^{-1}$	+	244,59	$4,0884 \cdot 10^{-3}$
14	+	$2,6587 \cdot 10^{-1}$	+	244,85	$4,0841 \cdot 10^{-3}$
⋮		⋮		⋮	

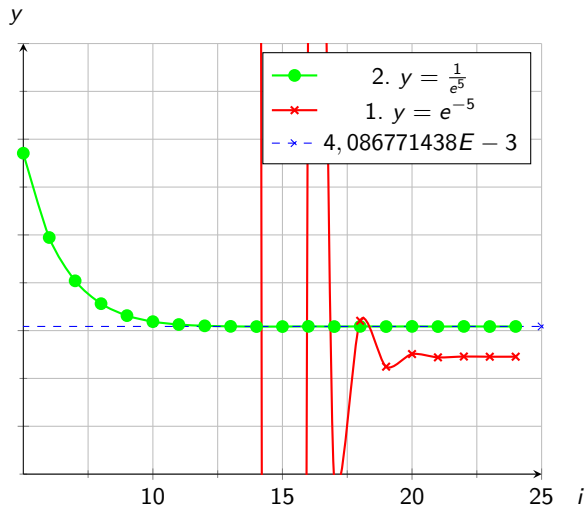
Numerik - Beispiel VIII



Numerik - Beispiel IX



Numerik - Beispiel X



Numerik - Beispiel XI

- ▶ Für die Berechnung von $y = e^{-5,5}$ haben wir gesehen, dass es zu erheblichen Fehlern kommen kann, je nach gewählten Algorithmus.
- ▶ Fehlerfortpflanzung
- ▶ Der Fehler bei zusammengesetzten Algorithmen muss entsprechend abgeschätzt werden.

Numerik - Beispiel XII

- Produkt von Maschinenzahlen $N_1 \odot N_2 \odot N_3 \odot \dots \odot N_k$ mit $N_i \in \mathbb{M}$

$$P_1 = N_1$$

$$P_2 = N_1 \odot N_2 = N_1 \cdot N_2 \cdot (1 + \epsilon_2)$$

$$P_3 = N_1 \odot N_2 \odot N_3 = P_2 \cdot N_3 \cdot (1 + \epsilon_3) = N_1 \cdot N_2 \cdot (1 + \epsilon_2) \cdot N_3 \cdot (1 + \epsilon_3)$$

$$\vdots$$

$$P_k = P_{k-1} \odot N_k = N_1 \cdot N_2 \cdot \dots \cdot N_k \cdot (1 + \epsilon_1) \cdot (1 + \epsilon_2) \cdot \dots \cdot (1 + \epsilon_k)$$

- $|\epsilon_i| \leq \text{eps}$ für $i = 2, \dots, k$

$$(1 - \text{eps})^{(k-1)} \leq (1 + \epsilon_2) \cdot (1 + \epsilon_3) \cdot \dots \cdot (1 + \epsilon_k) \leq (1 + \text{eps})^{(k-1)} \quad (1)$$

- Fehlerabschätzung

$$N_1 \odot N_2 \odot \dots \odot N_k = N_1 \cdot N_2 \cdot \dots \cdot N_k \cdot (1 + F)$$

$$\text{mit } (1 - \textit{eps})^{(k-1)} \leq 1 + F \leq (1 + \textit{eps})^{(k-1)}$$

- Die Betrachtungen zur Fehlerabschätzung sind für alle Operationen durchzuführen!

Numerik - Eulerzahl I

Näherungen für die Eulerzahl e :

► Leibniz-Reihe

$$e \approx \sum_{k=0}^{\infty} \frac{1}{k!}$$

► Näherungsbrüche

$$e \approx \frac{22}{7} = 3,1428 \dots \quad \text{Genauigkeit: 2 Nachkommastellen}$$

$$\pi \approx \frac{355}{113} = 3,1415929 \dots \quad \text{Genauigkeit: 6 Nachkommastellen}$$

$$\pi \approx \frac{103993}{33102} = 3,14159265301 \dots \quad \text{Genauigkeit: 9 Nachkommastellen}$$

Numerik - Kreiszahl I

Näherungen für die Zahl π :

► Leibniz-Reihe

$$\frac{\pi}{4} \approx \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} = 1 - \frac{1}{3} + \frac{1}{5} - \dots$$

► Näherungsbrüche

$$\pi \approx \frac{22}{7} = 3,1428\dots \quad \text{Genauigkeit: 2 Nachkommastellen}$$

$$\pi \approx \frac{355}{113} = 3,1415929\dots \quad \text{Genauigkeit: 6 Nachkommastellen}$$

$$\pi \approx \frac{103993}{33102} = 3,14159265301\dots \quad \text{Genauigkeit: 9 Nachkommastellen}$$

Danke für Ihre Aufmerksamkeit!

Inhalt

Einführung

Motivation

Konvertierung von Zahlenformaten

Höhere Funktionen

Numerik

Ausblick

Ausblick I

In den nächsten Einheiten geht es um

- ▶ Arithmetik in Recheneinheiten