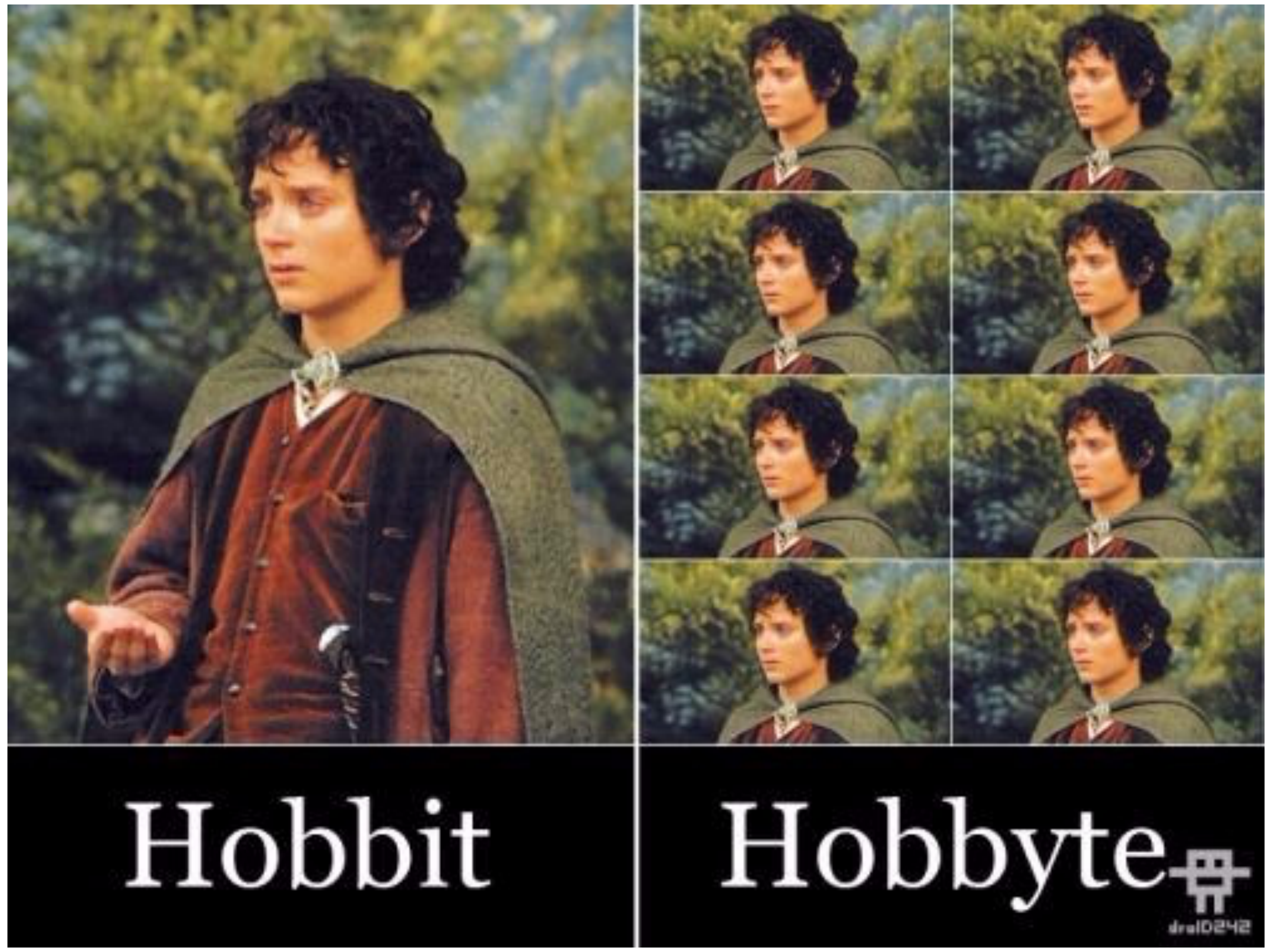
Zahlenrepräsentationen

Informationseinheiten

- **BIT** (binary digit) - Kurzform für Binärziffer ("0" oder "1")

- **BYTE** – 8 zusammen betrachtete Binärziffern

- **WORD** - Eine Folge von Zeichen, die in einem bestimmten Zusammenhang als eine Einheit betrachtet wird. (Meist 16 Bit)



- Wenn ein WORD einen Befehl enthält, so spricht man von einem Speicherwort.

Speicherzugriff

- Der Speicher speichert die einzelnen Bits ab
- Der Zugriff erfolgt immer pro **Byte**
- Die Bytes bekommen eine Speicheradresse, über welche sie angesprochen werden können
- Anweisungen in der CPU können mit einer definierten Anzahl an Bytes rechnen (CPU-spezifisch) - z.B. 32 Bit oder 64 Bit (<- aktuelle CPUs)
- Die Anbindung zum Speicher erfolgt auch nach dieser definierten Anzahl und daher werden z.B. immer 64 Bit aus dem Speicher geholt

Arbeitsspeicher

- Speicherzugriff bei einer 64 Bit Architektur:
  - Zugriff auf Adresse 0x0
  - Rückgabe von 64 Bit ab Adresse 0x0
- Es kann auch auf einzelne Byte zugegriffen werden (in Software umgesetzt)

Adresse	Inhalt
0x0000000000000000	00000001
0x0000000000000001	10101010
0x0000000000000002	00000001
0x0000000000000003	10101010
0x0000000000000004	10111111
0x0000000000000005	00000001
0x0000000000000006	10101010
0x0000000000000007	00000000
0x0000000000000008	10111111
0x0000000000000009	00000001
0x000000000000000A	10101010
0x000000000000000B	00000000
0x000000000000000C	10111111
0x000000000000000D	00000000
0x000000000000000E	00000000
0x000000000000000F	00000001
0x0000000000000010	10101010
0x0000000000000011	00000000
0x0000000000000012	00000000
0x0000000000000013	10111111
0x0000000000000014	00000000

Stellenwertsysteme

- Eine Folge von Bits kann als natürliche Zahl interpretiert werden
- Dualsystem oder auch Binärsystem (im Gegensatz zum Dezimalsystem)
- Zur Erinnerung: Zahlen im Dezimalsystem enthalten Ziffern von 0-9, deren Wertigkeit von rechts nach links um eine Zehnerpotenz zunimmt:
  - $125 = 1*10^2 + 2*10^1 + 5*10^0$
- Im Dualsystem steigt die Wertigkeit von Ziffern demnach von rechts nach links um eine Zweierpotenz
  - $101 = 1*2^2 + 0*2^1 + 1*2^0 = 5_{dezimal}$

Stellenwertsystem

- Dezimal: 0,1,2,3,4,5,6,7,8,9

Dezimal		Dezimal
0		
1		
...		
9		
1 0	$= \mathbf{1} \times 10^1 + \mathbf{0} \times 10^0$	$= 10 + 0 =$ 10
1 1	$= \mathbf{1} \times 10^1 + \mathbf{1} \times 10^0$	$= 10 + 1 =$ 11
1 2	$= \mathbf{1} \times 10^1 + \mathbf{2} \times 10^0$	$= 10 + 2 =$ 12
...		
2 8	$= \mathbf{2} \times 10^1 + \mathbf{8} \times 10^0$	$= 20 + 8 =$ 28
...		
1 3 4	$= \mathbf{1} \times 10^2 + \mathbf{3} \times 10^1 + \mathbf{4} \times 10^0$	$= 100 + 30 + 4 =$ 134
...		

Stellenwertsystem

- Hexadezimal: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Hexadezimal		Dezimal
0		
...		
9		
A		= 10
B		= 11
...		
F		= 15
1 0	$= \mathbf{1} \times 16^1 + \mathbf{0} \times 16^0$	$= 16 + 0 = 16$
1 1	$= \mathbf{1} \times 16^1 + \mathbf{1} \times 16^0$	$= 16 + 1 = 17$
...		
1 C 4	$= \mathbf{1} \times 16^2 + \mathbf{12} \times 16^1 + \mathbf{4} \times 16^0$	$= 256 + 192 + 4 = 452$
...		

Stellenwertsystem

- Binär: 0,1

Binär		Dezimal
0		
1		
1 0	$= \mathbf{1} \times 2^1 + \mathbf{0} \times 2^0$	$= 2 + 0 = \quad 2$
1 1	$= \mathbf{1} \times 2^1 + \mathbf{1} \times 2^0$	$= 2 + 1 = \quad 3$
1 0 0	$= \mathbf{1} \times 2^2 + \mathbf{0} \times 2^1 + \mathbf{0} \times 2^0$	$= 4 + 0 + 0 = \quad 4$
1 0 1	$= \mathbf{1} \times 2^2 + \mathbf{0} \times 2^1 + \mathbf{0} \times 2^0$	$= 4 + 0 + 1 = \quad 5$
1 1 0	$= \mathbf{1} \times 2^2 + \mathbf{0} \times 2^1 + \mathbf{0} \times 2^0$	$= 4 + 2 + 0 = \quad 6$
1 1 1	$= \mathbf{1} \times 2^2 + \mathbf{0} \times 2^1 + \mathbf{0} \times 2^0$	$= 4 + 2 + 1 = \quad 7$
1 0 0 0	$= \mathbf{1} \times 2^3 + \mathbf{0} \times 2^2 + \mathbf{0} \times 2^1 + \mathbf{0} \times 2^0$	$= 8 + 0 + 0 + 0 = \quad 8$
1 0 0 1	$= \mathbf{1} \times 2^3 + \mathbf{0} \times 2^2 + \mathbf{0} \times 2^1 + \mathbf{1} \times 2^0$	$= 8 + 0 + 0 + 1 = \quad 9$
...		
1 1 0 1 1 0	$= \mathbf{1} \times 2^5 + \mathbf{1} \times 2^4 + \mathbf{0} \times 2^3 + \mathbf{1} \times 2^2 + \mathbf{1} \times 2^1 + \mathbf{0} \times 2^0$	$= 32 + 16 + 0 + 4 + 2 + 0 = 54$
...		

Stellenwertsysteme

- Der Wert einer Zahl (=Ziffernfolge) ist abhängig von der Basis. Ohne Angabe der Basis ist der Wert nicht eindeutig definiert.

Basis b	Bezeichnung	Ziffernbereich
2	Binär, Dual	0,1
8	Oktal	0,1,…,7
10	Dezimal	0,1,…,9
16	Hexadezimal	0,1,…,9,A,B,C,D,E,F

Umrechnung nach Dezimal

- $101{,}11_{(2)} = 1 * \mathbf{2^2} + 0 * 2^1 + 1 * \mathbf{2^0} + 1 * \mathbf{2^{-1}} + 1 * \mathbf{2^{-2}}$

  $= \mathbf{4} + 0 + \mathbf{1} + \mathbf{0{,}5} + \mathbf{0{,}25} = 5{,}75$

$$\boxed{2^{-2} = \frac{1}{2^2}}$$

- $5{,}5_{(16)} = 5 * 16^0 + 5 * \mathbf{16^{-1}} =$

  $5 + 5 * \mathbf{0{,}0625} = 5{,}3125$

Beispiel

z = 29	z div 2	z mod 2
29	14	1
14	7	0
7	3	1
3	1	1
1	0	1

Rest von unten nach oben gelesen
ergibt das Ergebnis

=> $(1\ 1\ 1\ 0\ 1)_2$

von Dezimal zu Binär

- $1{,}8125_{(10)}$ zu Binär:

0,8125	*2	=	**1**,625
0,625	*2	=	**1**,25
0,25	*2	=	**0**,5
0,5	*2	=	**1**,0

$= 1\ ,\ \mathbf{1101}_{(2)}$

OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG
INFORMATIK UND
MATHEMATIK

Beispiel 2

z = 105	z / 2	z mod 2 (Rest)
105	52	1
52	26	0
26	13	0
13	6	1
6	3	0
3	1	1
1	0	1

=> $1101001_{(2)}$

Beispiel 2

- $0{,}2525_{(10)}$ zu Binär:

0,2525	*2	=	**0**,505
0,505	*2	=	**1**,01
0,01	*2	=	**0**,02
0,02	*2	=	**0**,04
0,04	*2	=	**0**,08
0,08	*2	=	**0**,16
0,16	*2	=	**0**,32
0,32	*2	=	**0**,64
0,64	*2	=	**1**,28

…

= 0 , **010000001…**$_{(2)}$

Spezialfälle

- Konvertierungen zwischen **hexadezimal – binär** und **oktal** sind sehr einfach durchzuführen:

$$(A \quad B \quad C)_{16}$$

$$(\overbrace{1010}\,\overbrace{1011}\,\overbrace{1100})_2 = (\underbrace{101}\,\underbrace{010}\,\underbrace{111}\,\underbrace{100})_2$$

$$(5 \quad 2 \quad 7 \quad 4)_8$$

Spezialfälle Beispiele

7F5A7,A Hex zu Bin: 1111111010110100111,101

7	F	5	A	7	,	A
0111	1111	0101	1010	0111	,	1010

1111100111011111011,11 Bin zu Hex: 7CEFB,C

0111	1100	1110	1111	1011	,	11**00**
7	C	E	F	B	,	C

Darstellung von Zahlen

- Computer kennt nur 0 und 1
- Wo fängt die Zahl an?
- Wo ist sie zu Ende?

- Feste Länge pro Zahl definieren:
- z.B. Integer: 32 Bit, Long: 64 Bit (Beispielhaft, je nach Architektur und Sprache verschieden)

Darstellung von Zahlen

$468.749.823_{(10)}$

0001 1011 1111 0000 1000 1101 1111 1111 32 Bit

Kürzere Schreibweise in Hex: 0x1BF08DFF

Darstellung von Zahlen

- **Achtung:** Zahlen im Speicher sind entweder 1, 2, 4 oder 8 Byte groß und damit in der Anzahl der Ziffern begrenzt!
- Was passiert bei folgender Rechnung, wenn die Zahl in einem Byte gespeichert wird?
- 11111110 + 00000010 = ????????
- Bei der Wahl der Datentypen aufpassen und vorher die **maximale Größe** von Werten abschätzen

Ganze Zahlen

- Die Nutzung des 2er-Komplement ermöglicht die Darstellung von negativen und positiven ganzen Zahlen
- Vorzeichenbit repräsentiert Zahlenbereich
- Subtraktion durch Umwandlung in Addition mit negativen Zahlen

$$z = -a_n * 2^n + a_{n-1} * 2^{n-1} + .. + a_1 * 2^1 + a_0 * 2^0$$

2er Komplement

- Aus der positiven Zahl:
- $16.608_{(10)}$ => $0100000011100000_{(2)}$
- die negative Zahl:
- $-16.608_{(10)}$ => $?_{(2)}$

2er Komplement

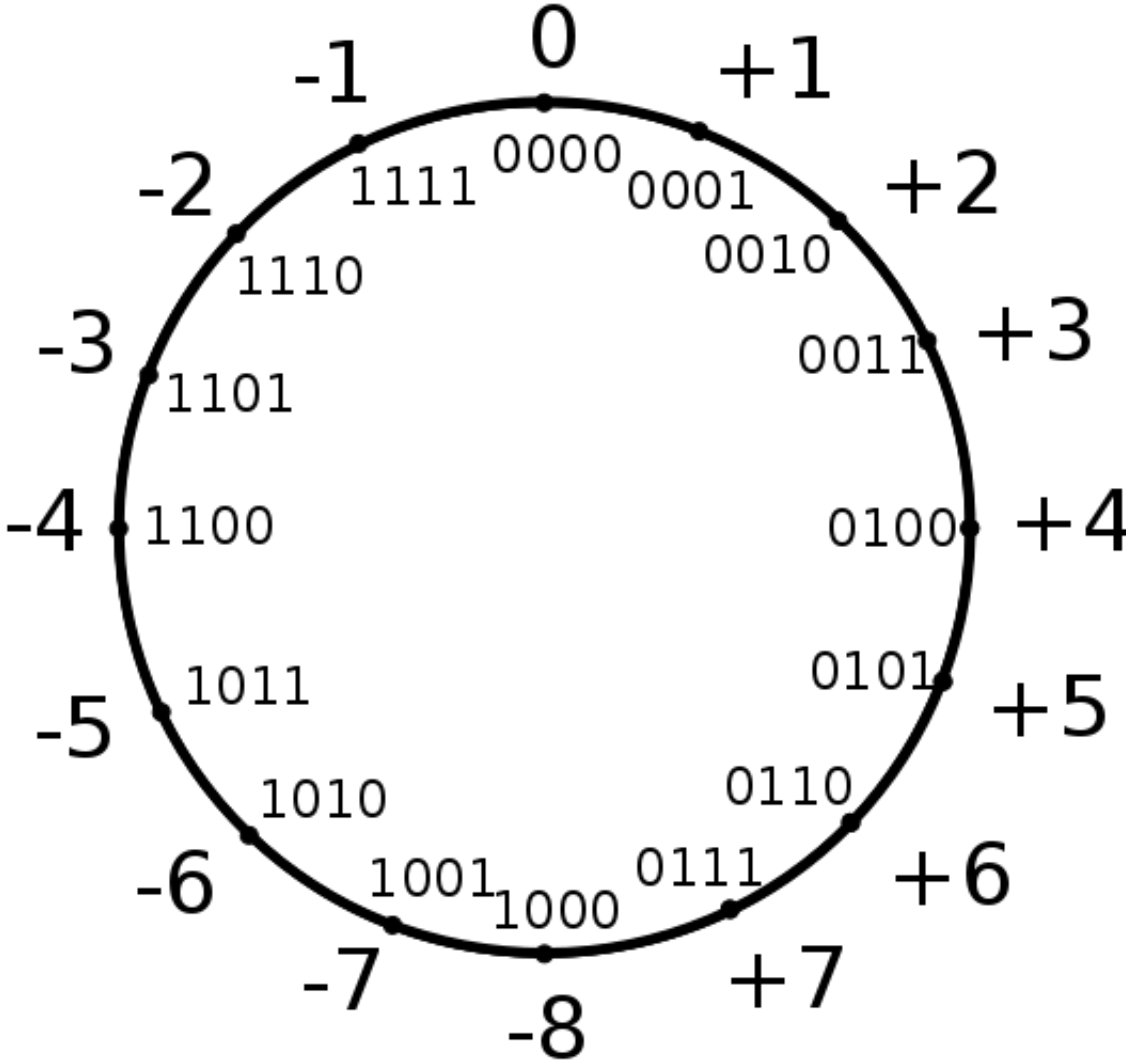
- Von rechts bis zur ersten 1 abschreiben (incl. erste 1) und dann die restlichen Stellen umdrehen (aus 1 wird 0 und aus 0 wird 1):

- 0100000011 100000 <= von rechts beginnen

- Umdrehen, abschreiben

- 1011111100 100000

2er Komplement



2er Komplement

- Interpretation:
  - das höchstwertige Bit hat eine negative Wertigkeit

Wertigkeit	-128	64	32	16	8	4	2	1	Dezimal
Bitfolge	0	0	0	1	1	0	1	0	= 26
Bitfolge	1	1	1	0	0	1	1	0	= -26

- $00011010_{(2)} = 16 + 8 + 2 = 26$
- $11100110_{(2)} = -128 + 64 + 32 + 4 + 2 = -26$

2er Komplement

- Keine negative 0
- Alle Rechenoperationen können ohne Anpassungen verwendet werden
- Der Registerüberlauf muss ignoriert werden
- Immer die Anzahl der Stellen beachten!
- Beim kopieren in eine andere Größe: Sign beachten (1011 zu 1111 1011)

Festkommazahlen

- Darstellung durch Kommazahlen mit fester Anzahl n Stellen vor dem Komma und m Stellen nach dem Komma (Festkommadarstellung, fixed point)

$$z = (a_n a_{n-1} \ldots a_1 a_0 , b_1 b_2 \ldots b_{m-1} b_m)_2$$
$$= (a_n * 2^n + a_{n-1} * 2^{n-1} + \ldots + a_0 , b_1 * 2^{-1} + \ldots + b_m * 2^{-m})_2$$

- Behandelte Rechenverfahren können direkt übernommen werden, evtl. Anpassungen

- Stellenkorrektur

```
0000 1010  * 0000 1100  = 0111 1000
0000 1,010 * 0000 1,100 = 0000 1,111
```

Festkommazahlen - Hinweise

- Genauigkeitsverlust bei kleinen Beträgen
  - 00123,456 : 100 = 00001,234
    - d.h. zwei signifikante Ziffern gehen verloren
- Überlauf bei hohen Beträgen
  - 00123,456 · 1000 = (1)23456,000
    - zu viele Stellen für vorgesehene Darstellung
- Daher **Gleitkommadarstellung (floating point)**
  - Idee: signifikanten Ziffern und ihrer Position getrennt dargestellt (Exponentialschreibweise)

Gleitkommazahlen

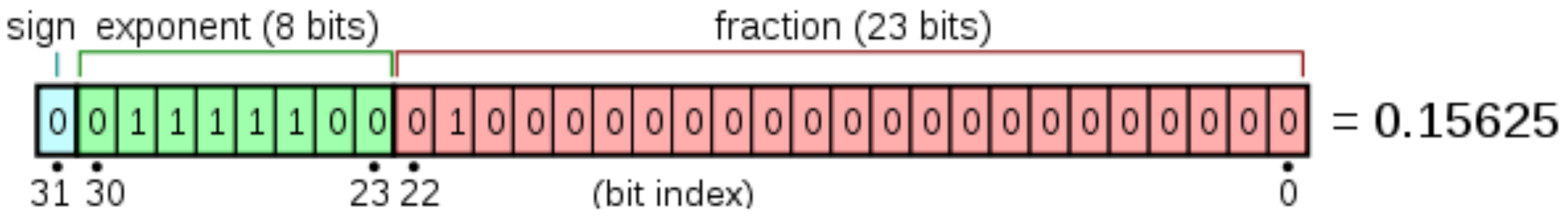
- Nutzung von Exponent und Mantisse
  - Exponent zeigt Nachkommastelle bzw. Position der Mantisse (in Bezug zu einer Basis)
  - Mantisse zeigt darstellbare Stellen

$$123{,}456 = 0{,}\underbrace{123456}_{\text{Mantisse (m)}} \times \underbrace{10}_{\text{Basis (b)}}^{\overbrace{3}^{\text{Exponent (e)}}}$$

- Darstellung nicht eindeutig
- Deshalb „Normalisierung“ erforderlich

IEEE floating point (IEEE 754)

Name	Common name	Base	Digits	Decimal digits	Exponent bits	Decimal E max	Exponent bias[6]	E min	E max
binary16	Half precision	2	11	3.31	5	4.51	$2^{4}-1 = 15$	−14	+15
binary32	Single precision	2	24	7.22	8	38.23	$2^{7}-1 = 127$	−126	+127
binary64	Double precision	2	53	15.95	11	307.95	$2^{10}-1 = 1023$	−1022	+1023
binary128	Quadruple precision	2	113	34.02	15	4931.77	$2^{14}-1 = 16383$	−16382	+16383
binary256	Octuple precision	2	237	71.34	19	78913.2	$2^{18}-1 = 262143$	−262142	+262143
decimal32		10	7	7	7.58	96	101	−95	+96
decimal64		10	16	16	9.58	384	398	−383	+384
decimal128		10	34	34	13.58	6144	6176	−6143	+6144



Gleitkommazahlen

- Hauptstandard: IEEE-Format als Normalisierung
- Single/Double Precision

Größe	32 Bit	64 Bit
Vorzeichen	1	1
Exponent	8	11
Mantisse	23	52

- Vorzeichen: 0 positiv, 1 negativ
- Exponent: Wertebereich -126 bis +127
- Mantisse: normalisiert auf 1,…
  - 0 nicht normalisierbar -> minimale Mantisse/Exponent
  - Sonderfälle: NaN und „Unendlich“

Gleitkommazahlen

- $12{,}25_{(10)}$ in Binäre Gleitkommazahl umwandeln:
  - Vorzeichen: 1 Bit (0: positiv, 1: negativ)
  - Länge des Exponenten: 5 Bit
  - Länge der Mantisse: 6 Bit
  - Normalisierung auf 1,…

Gleitkommazahlen

- $12{,}25_{(10)}$ in Binäre Gleitkommazahl umwandeln:
  - Vorzeichen:
    - positiv -> 0
  - Umwandeln:
    - $12_{(10)} = 1100_{(2)}$
    - $0{,}25_{(10)} = 0{,}01_{(2)}$
  - Normalisieren:
    - $1100{,}01 = 1{,}10001 * 2^3$

Gleitkommazahlen

- $12{,}25_{(10)}$ in Binäre Gleitkommazahl umwandeln:
  - Normalisieren:
    - $1100{,}01 = 1{,}10001 * 2^3$
  - Mantisse:
    - 10001
  - Exponent:
    - $3_{(10)} = 11_{(2)}$
    -

	VZ	Exponent					Mantisse					
12,25	0	0	0	0	1	1	1	0	0	0	1	0

Gleitkommazahlen

- $12{,}25_{(10)}$ in Binäre Gleitkommazahl umwandeln:
  - Normalisieren:
    - $1100{,}01 = 1{,}10001 * 2^3$
  - Mantisse:
    - 10001
  - Exponent:
    - ~~$3_{(10)} = 11_{(2)}$~~
    -

	VZ	Exponent					Mantisse					
12,25	0	0	0	0	1	1	1	0	0	0	1	0

Gleitkommazahlen IEEE 754

- $x = s * m * b^e$
- Exponent:
  - fester Biaswert B wird addiert: $E = e + B$
  - $B = 2^{r-1} - 1$
  - r = Anzahl der Stellen des Exponenten

Gleitkommazahlen

- $12{,}25_{(10)}$ in Binäre Gleitkommazahl umwandeln:
    - Normalisieren:
        - $1100{,}01 = 1{,}10001 * 2^3$
    - Mantisse:
        - 10001
    - Exponent:
        - $3 + 2^{r-1} - 1 = 3 + 2^{5-1} - 1 = 3 + 16 - 1 = 18_{(10)} = 10010_{(2)}$
        -

	VZ	Exponent					Mantisse					
12,25	0	1	0	0	1	0	1	0	0	0	1	0

Gleitkommazahlen - Konvertierung

- Umrechnung zwischen dezimal – binär durch Horner-Schema (erweiterte Form)
  - Vorkommazahl umrechnen
  - Nachkommazahl umrechnen
  - Normalisierung
  - Exponent berechnen
  - Vorzeichen bestimmen
  - Ergebnis darstellen

Gleitkommazahlen - Probleme

- **Anzahl der Bits für die Speicherung des Zahlenwertes sind begrenzt à Genauigkeit ist begrenzt!**
- Grund:
  - Anzahl der Stellen der Mantisse und des Exponenten
  - Biaswert
- **VORSICHT** bei Gleichheit von Gleitkomma-Zahlen, besser Intervall testen und Gleichungen ggf. umstellen!

Konvertierungen

- Hexadezimal- / Binärtabelle

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
DEC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BIN	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Darstellung von Zeichen

- ASCII: American Standard Code for Information Interchange
- Die Zeichen werden nummeriert 0 … 127
- Zur Darstellung benötigt man also 7 Bit, dies passt in ein Byte mit 8 Bit.
- ‘A‘ = 65 (0x41)
- ‘B‘ = 66 (0x42) …
- …
- ‘a‘ = 97 (0x61) …
- …
- ’0’ = 48 (0x30) Ziffer 0
- ‘1‘ = 49 (0x31) Ziffer 1 usw.
- …

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Zeichenketten

- Zeichenketten sind Folgen von Zeichen
  - “ABC“ entspricht einer Abfolge von 3 Bytes gemäß ASCII-Code im Speicher
- Ihre Darstellung hängt von der Programmiersprache ab, es gibt keine einheitliche Verarbeitung durch CPUs wie z. B. bei ganzen Zahlen
- In C schreibt man eine Zeichenkette in der Form:
  - "Hello"
- ABER: C kennt keine Befehle zur Verarbeitung
- Die Bibliotheken zur Sprache C enthalten Verfahren zur Bearbeitung von Zeichenketten

Zusammenfassung

- Soll das Programm Zahlendarstellungen benutzen, so ist zu wählen
  - Ganze Zahlen
    - Welche Wortbreite (8, 16, 32 oder 64 Bit)
  - Reelle Zahlen
    - Welche Genauigkeit (entspricht der Wortbreite)
- Programmiersprachen stellen hierfür unterschiedliche sogenannte Datentypen bereit
- Achtung: Ungenauigkeiten bei Gleitkomma-Darstellungen und Übertragfehlern bei ganzen Zahlen