

# Daten und Datenrepräsentation

Prof. Dr. Christian Becker

Universität Stuttgart, Institut für Parallele und Verteilte Systeme

15. Mai 2025



# Übersicht

- ▶ Daten vs. Informationen
- ▶ Bits und Bytes
- ▶ Darstellung von Zahlen
- ▶ Darstellung von anderen Daten

noch mehr?



# Daten vs. Informationen

## Daten

- ▶ Daten sind rohe Fakten, wie Worte oder Zahlen.

## Informationen

- ▶ Informationen sind Daten mit Kontext oder Bedeutung
- ▶ Verständlich für Menschen
- ▶ Ergebnis von Datenverarbeitung

## Beispiel:

- ▶ Daten: 10260000
- ▶ Information: 102.600,00 € (Der Preis eines neuen Elektro-Porsche Taycan)

# Übersicht

So stellen wir diesen Datenpunkt dar:



**10260000**

Aber wie macht es ein Computer?



**???**

# Bits und Bytes

- ▶ Computer speichern Daten mit Bits
- ▶ Ein Bit ist die kleinste Einheit der Datenspeicherung
- ▶ Ein Bit ist die Menge an Informationen die man braucht um zwei Zustände voneinander zu unterscheiden (0 und 1)
- ▶ Technische Darstellung der Zustände

Spannung    1: Positiv,                0: Negativ

CD-ROM     1: land,                0: pit

Kassetten    1: magnetisiert      0: nicht magnetisiert

Computer    1: Strom                0: kein Strom



# Bits und Bytes

- ▶ Bits werden gruppiert um mehr als 2 Zustände zu speichern
  - ▶ 2 Bits:  $2^2 = 4$  Zustände (00, 01, 10, 11)
  - ▶ 3 Bits:  $2^3 = 8$  Zustände (000, 001, 010, 011, 100, 101, 110, 111)
  - ▶ ...
  - ▶ 8 Bits:  $2^8 = 256$  Zustände
  - ▶ ...
  - ▶ n Bits:  $2^n$  Zustände
- ▶ Eine Gruppe von 8 Bits ist ein **Byte**

# Darstellung von Zahlen

- ▶ Wenn Computer nur zwischen 0 und 1 unterscheiden können, wie stellen wir dann „normale Zahlen“ im Computer dar?
- ▶ Was sind eigentlich „normale“ Zahlen?
- ▶ Es gibt jede Menge Möglichkeiten, Zahlen darzustellen

# Babylonisches Zahlensystem

- ▶ Haben Sie sich schon mal gefragt wieso eine Stunde in 60 Minuten unterteilt wird?
- ▶ Überreste der babylonischen Mathematik!
- ▶ Das Babylonische Zahlensystem basierte auf 60 Zeichen pro Stelle

↷ 1	↷↷ 11	↷↷↷ 21	↷↷↷↷ 31	↷↷↷↷↷ 41	↷↷↷↷↷↷ 51
↷↷ 2	↷↷↷ 12	↷↷↷↷ 22	↷↷↷↷↷ 32	↷↷↷↷↷↷ 42	↷↷↷↷↷↷↷ 52
↷↷↷ 3	↷↷↷↷ 13	↷↷↷↷↷ 23	↷↷↷↷↷↷ 33	↷↷↷↷↷↷↷ 43	↷↷↷↷↷↷↷↷ 53
↷↷↷↷ 4	↷↷↷↷↷ 14	↷↷↷↷↷↷ 24	↷↷↷↷↷↷↷ 34	↷↷↷↷↷↷↷↷ 44	↷↷↷↷↷↷↷↷↷ 54
↷↷↷↷↷ 5	↷↷↷↷↷↷ 15	↷↷↷↷↷↷↷ 25	↷↷↷↷↷↷↷↷ 35	↷↷↷↷↷↷↷↷↷ 45	↷↷↷↷↷↷↷↷↷↷ 55
↷↷↷↷↷↷ 6	↷↷↷↷↷↷↷ 16	↷↷↷↷↷↷↷↷ 26	↷↷↷↷↷↷↷↷↷ 36	↷↷↷↷↷↷↷↷↷↷ 46	↷↷↷↷↷↷↷↷↷↷↷ 56
↷↷↷↷↷↷↷ 7	↷↷↷↷↷↷↷↷ 17	↷↷↷↷↷↷↷↷↷ 27	↷↷↷↷↷↷↷↷↷↷ 37	↷↷↷↷↷↷↷↷↷↷↷ 47	↷↷↷↷↷↷↷↷↷↷↷↷ 57
↷↷↷↷↷↷↷↷ 8	↷↷↷↷↷↷↷↷↷ 18	↷↷↷↷↷↷↷↷↷↷ 28	↷↷↷↷↷↷↷↷↷↷↷ 38	↷↷↷↷↷↷↷↷↷↷↷↷ 48	↷↷↷↷↷↷↷↷↷↷↷↷↷ 58
↷↷↷↷↷↷↷↷↷ 9	↷↷↷↷↷↷↷↷↷↷ 19	↷↷↷↷↷↷↷↷↷↷↷ 29	↷↷↷↷↷↷↷↷↷↷↷↷ 39	↷↷↷↷↷↷↷↷↷↷↷↷↷ 49	↷↷↷↷↷↷↷↷↷↷↷↷↷↷ 59
↶ 10	↶↶ 20	↶↶↶ 30	↶↶↶↶ 40	↶↶↶↶↶ 50	

# 12-er System

- ▶ Basis 12 war früher beliebt
- ▶ 12 Pence ergeben 1 Shilling
- ▶ Viele Sprachen haben noch immer einen Begriff für eine Einheit von 12 (ein Duzend, a dozen)



# Unärsystem



- ▶ Zahl ist die Summe der erhobenen Finger

# Zahlensysteme

- ▶ Wir stellen Zahlen üblicherweise im Dezimalsystem dar
- ▶ Wir haben Zehn verschiedene Zeichen für jede Stelle (0-9)
- ▶ Der Wert setzt sich zusammen aus dem Zeichen und der Stelle an der es steht
- ▶ Eine 1 kann 0.1, 1, 10, 100, ... bedeuten, je nachdem wo sie steht

# Binärsystem

- Idee: Jeder Finger hat zwei Zustände: Erhoben und nicht erhoben
- Damit zählt nicht nur die Anzahl Finger sondern auch die Position

Unär



1

1

1

2

Binär



1

2

3

4

# Darstellung von Zahlen im Allgemeinen

- ▶ Es lassen sich beliebige Zahlensysteme nach diesem Schema bilden
- ▶ Der Wert einer Ziffer setzt sich aus dem Zeichen und der Stelle zusammen
- ▶ Die Basis entscheidet die Anzahl Zeichen pro Stelle und damit auch den Wert einer Stelle

Der Wert der Zahl  $a_n a_{n-1} \dots a_0$  in der Basis  $B$  ist

$$a_n \cdot B^n + a_{n-1} \cdot B^{n-1} + \dots + a_0 = \sum_{i=0}^n a_i \cdot B^i$$

wobei die  $a_i \in \{0, 1, \dots, B - 1\}$  sind.

# Dezimalsystem

- ▶ Zehn verschiedene Zeichen für jede Stelle (0-9)
- ▶ Der Wert setzt sich zusammen aus dem Zeichen und der Stelle an der es steht

1

5

9

Hunderter

Zehner

Einer

Stelle 2

1 0

Wert  $1 \cdot 10^2$

$5 \cdot 10^1$   $9 \cdot 10^0$

# Zahlensysteme

- ▶ Alle Berechnungen im Rechner werden im Binärsystem durchgeführt
- ▶ Frage: Wie konvertieren wir vom Binärsystem ins Dezimalsystem?
- ▶ Schauen wir uns die Binärzahl 111 an:

1	1	1
Vierer	Zweier	Einer
Stelle	2	1
Wert	$1 \cdot 2^2$	+
	$1 \cdot 2^1$	+
	$1 \cdot 2^0$	= 7

# Konvertierung Dezimal zu Binär

- Wie konvertiert man eine Dezimalzahl ins Binärsystem?

$$19 \quad \div \quad 2 \quad = \quad 9 \quad \text{Rest } 1$$

1



# Konvertierung Dezimal zu Binär

- Wie konvertiert man eine Dezimalzahl ins Binärsystem?

$$\begin{array}{r} 19 \\ \div 2 = 9 \quad \text{Rest } 1 \end{array}$$

$$\begin{array}{r} 9 \\ \div 2 = 4 \quad \text{Rest } 1 \end{array}$$

1 1

# Konvertierung Dezimal zu Binär

- Wie konvertiert man eine Dezimalzahl ins Binärsystem?

$$\begin{array}{r} 19 \\ \div 2 = 9 \quad \text{Rest } 1 \end{array}$$

$$\begin{array}{r} 9 \\ \div 2 = 4 \quad \text{Rest } 1 \end{array}$$

$$\begin{array}{r} 4 \\ \div 2 = 2 \quad \text{Rest } 0 \end{array}$$

0 1 1

# Konvertierung Dezimal zu Binär

- Wie konvertiert man eine Dezimalzahl ins Binärsystem?

$$\begin{array}{r} 19 \\ \div 2 = 9 \quad \text{Rest } 1 \end{array}$$

$$\begin{array}{r} 9 \\ \div 2 = 4 \quad \text{Rest } 1 \end{array}$$

$$\begin{array}{r} 4 \\ \div 2 = 2 \quad \text{Rest } 0 \end{array}$$

$$\begin{array}{r} 2 \\ \div 2 = 1 \quad \text{Rest } 0 \end{array}$$

0 0 1 1

# Konvertierung Dezimal zu Binär

- Wie konvertiert man eine Dezimalzahl ins Binärsystem?

$$\begin{array}{r} 19 \\ \div 2 = 9 \quad \text{Rest } 1 \end{array}$$

$$\begin{array}{r} 9 \\ \div 2 = 4 \quad \text{Rest } 1 \end{array}$$

$$\begin{array}{r} 4 \\ \div 2 = 2 \quad \text{Rest } 0 \end{array}$$

$$\begin{array}{r} 2 \\ \div 2 = 1 \quad \text{Rest } 0 \end{array}$$

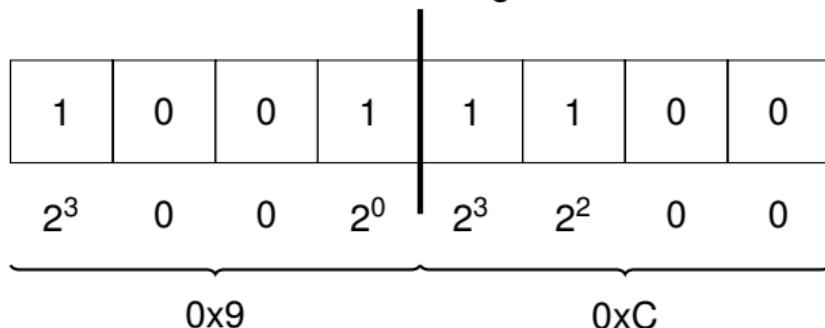
$$\begin{array}{r} 1 \\ \div 2 = 0 \quad \text{Rest } 1 \end{array}$$

1 0 0 1 1

# Hexadezimalzahlen

- ▶ Hexadezimalzahlen verwendet man manchmal als abkürzende Schreibweise für Binärzahlen, da Binärzahlen sehr schnell lang werden
- ▶ Es gibt 16 mögliche Zeichen für jede Ziffer: 0-9A-F
- ▶ Wir schreiben üblicherweise 0x vor eine Hexadezimalzahl um Verwechslungen zu vermeiden

Umwandlung Binär – Hexadezimal kann immer in 4-Bit Blöcken erfolgen:



Zeichen	Dezimalwert
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

# Darstellung von Zahlen

- ▶ Wo fangen Zahlen an, wo hören sie auf?
- ▶ Wir legen vorher eine feste Größe fest:
- ▶ Zum Beispiel 8, 16, 32, 64 Bit pro Zahl
- ▶ Abhängig von der Anzahl Bits unterscheidet sich der Wertebereich

# Addition von Binärzahlen

- ▶ Addition von Binärzahlen funktioniert wie schriftliche Addtion (aber  $1 + 1 = 10$ )
- ▶ Subtraktion kommt später genauer (funktioniert genau gleich)

$$\begin{array}{r} 1\ 0\ 1\ 1 \\ + \quad 0\ 0\ 0\ 1 \\ \hline = \quad 1\ 1\ 0\ 0 \end{array}$$

# Darstellung von negativen Zahlen

- ▶ Wie stelle ich  $-3$  im Binärsystem dar?
- ▶ Erste Idee: Ein Bit für Vorzeichen

$+3$	0	0	1	1
$-3$	1	0	1	1

- ▶ Problem: Zwei verschiedene Darstellungen für die 0 ( $+0, -0$ )
- ▶ Problem: Rechnen mit Vorzeichenbit ist kompliziert

# 1er-Komplement

- ▶ Nächste Idee: Um negative Zahl darzustellen, invertiere alle Bits der positiven Zahl
- ▶ Vorderstes Bit gibt an, ob die Zahl negativ ist

+3	0	0	1	1
-3	1	1	0	0

- ▶ Problem: Zwei verschiedene Darstellungen für die 0 (+0, -0)
- ▶ Problem: Rechnen ist weiterhin kompliziert

# 2er-Komplement

- Nächste Idee: Um negative Zahl darzustellen, invertiere alle Bits der positiven Zahl **und addiere 1**

+3	0	0	1	1
-3	1	1	0	1

- Rechnen mit negativen Zahlen funktioniert gleich wie mit positiven Zahlen!
- Problem: Kleinste negative Zahl lässt sich nicht negieren (halb so wild)

# Rechnen im 2er-Komplement

- ▶ Berechne  $3 + (-3)$  mit 4 Bits:

$$\begin{array}{r} 0 \ 0 \ 1 \ 1 \\ + \ 1 \ 1 \ 0 \ 1 \\ \hline = \ 0 \ 0 \ 0 \ 0 \end{array}$$

- ▶ Übertrag der über die verfügbaren Bits hinaus geht wird verworfen
- ▶ Rechnen mit negativen Zahlen funktioniert gleich wie mit positiven Zahlen!

# Integer Overflow

- ▶ Sie haben 4 Bits zur Verfügung.
- ▶ Was ist die größte Zahl, die Sie darstellen können?  
Was passiert, wenn Sie +1 rechnen?
- ▶ Übertrag der über die verfügbaren Bits hinaus geht wird verworfen

$$\begin{array}{r} 1 \quad 1 \quad 1 \quad 1 \\ + \quad 0 \quad 0 \quad 0 \quad 1 \\ \hline = \quad 0 \quad 0 \quad 0 \quad 1 \end{array}$$

Overflows erzeugen überraschende Ergebnisse!

```
1 int a = 1;
2 int b = 2147483647;
3
4 int c = a + b;
```

Java

Hier ist `c = -2147483648`

# Kommazahlen

## Festkommazahlen

12.345

- ▶ Feste Zahl Stellen sowohl vor als auch nach dem Komma

## Fließkommazahlen

$1.xxx \cdot 2^n$

- ▶ Möglichst großes Intervall an darstellbaren Zahlen ermöglichen
- ▶ Tradeoff zwischen Großen Zahlen oder vielen Nachkommastellen
- ▶ Meistens wird nicht beides gleichzeitig benötigt
- ▶ Daher: Feste Zahl Stellen, aber skalierbar mit einem Faktor
- ▶ Problem: Rundungsfehler!!

Genauere Details zu IEE754, Darstellung etc.

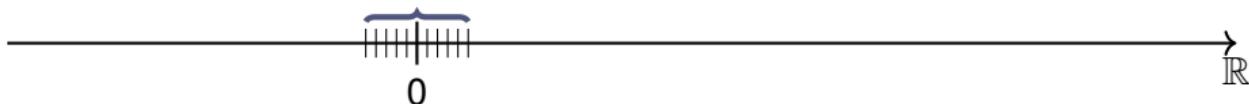


# Fließkommazahlen

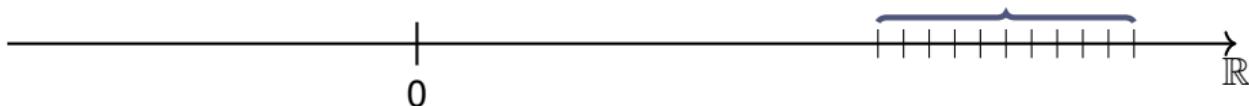
- ▶ Mit dem Exponenten  $n$  lässt sich der darstellbare Bereich der Zahlen auf dem Zahlenstrahl verschieben
- ▶ Je weiter man sich von der 0 entfernt, desto größer wird die Auflösung

$$1.xxx \cdot 2^n$$

$n = 0$ : Darstellbarer Bereich



$n = 6$ : Darstellbarer Bereich



# ASCII

## American Standard Code for Information Interchange

						...				...		
20	00100000	(Space)				41	01000001	A	61	01100001	a	
21	00100001	!				42	01000010	B	62	01100010	b	
22	00100010	"		...		43	01000011	C	63	01100011	c	
						44	01000100	D	64	01100100	d	
						45	01000101	E	65	01100101	e	
						46	01000110	F	66	01100110	f	
						47	01000111	G	67	01100111	g	
						48	01001000	H	68	01101000	h	
23	00100011	#	30	00110000	0	49	01001001	I	69	01101001	i	
24	00100100	\$	31	00110001	1	4A	01001010	J	6A	01101010	j	
25	00100101	%	32	00110010	2	4B	01001011	K	6B	01101011	k	
26	00100110	&	33	00110011	3	4C	01001100	L	6C	01101100	l	
27	00100111	,	34	00110100	4	4D	01001101	M	6D	01101101	m	
28	00101000	(	35	00110101	5	4E	01001110	N	6E	01101110	n	
29	00101001	)	36	00110110	6	4F	01001111	O	6F	01101111	o	
2A	00101010	*	37	00110111	7	50	01010000	P	70	01110000	p	
2B	00101011	+	38	00111000	8	51	01010001	Q	71	01110001	q	
2C	00101100	,	39	00111001	9	52	01010010	R	72	01110010	r	
2D	00101101	-		...		53	01010011	S	73	01110011	s	
2E	00101110	.				54	01010100	T	74	01110100	t	
2F	00101111	/				55	01010101	U	75	01110101	u	
				...		56	01010110	V	76	01110110	v	
						57	01010111	W	77	01110111	w	
						58	01011000	X	78	01111000	x	
						59	01011001	Y	79	01111001	y	
						5A	01011010	Z	7A	01111010	z	
							...			...		



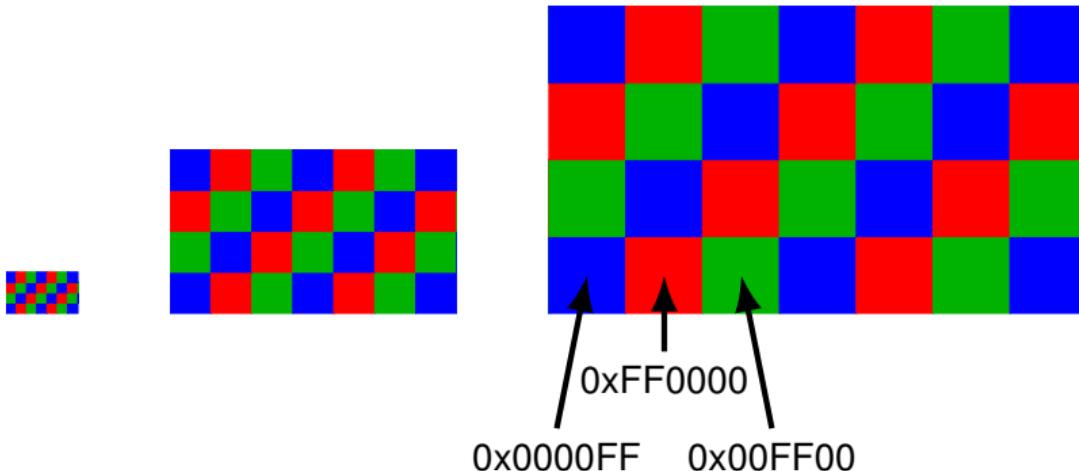
# Unicode

- ▶ Problem an ASCII: Unterstützt keine Sonderzeichen oder Buchstaben anderer Alphabete (z.B. ä,ö,ü,ß)
- ▶ Unicode (Universal Character Encoding) stellt verschiedene Transformationsformate (UTF: Unicode Transformation Format) zur Übersetzung von Buchstaben und Symbolen zur Verfügung
- ▶ Geläufige Transformationen sind UTF-8 (1 Byte pro Zeichen), UTF-16 (2 Byte pro Zeichen), etc.
- ▶ Zeichencode für Emojis ist UTF-32



# Bilder, Video

- ▶ Bilder bestehen aus Pixeln, die sich aus drei Grundfarben zusammensetzen
- ▶ Monitore stellen Pixel durch Kombination von RGB dar



- ▶ In der Praxis werden häufig 1 Byte pro Farbe verwendet (s.o.)
- ▶ Auch möglich: 4 Byte pro Pixel, 2 Byte für Grün; Rot und Blau jeweils 1 Byte (das Auge ist empfindlicher für Grün)