



# 64-040 Modul InfB-RSB

## Rechnerstrukturen und Betriebssysteme

[https://tams.informatik.uni-hamburg.de/  
lectures/2022ws/vorlesung/rsb](https://tams.informatik.uni-hamburg.de/lectures/2022ws/vorlesung/rsb)

– Kapitel 2 –

Andreas Mäder



Universität Hamburg  
Fakultät für Mathematik, Informatik und Naturwissenschaften  
Fachbereich Informatik

Technische Aspekte Multimodaler Systeme

Wintersemester 2022/2023



## Informationsverarbeitung

Semantic Gap

Abstraktionsebenen

Beispiel: HelloWorld

Definitionen und Begriffe

Informationsübertragung

Zeichen

Literatur



## Tanenbaum, Austin: *Rechnerarchitektur* [TA14]

*Ein Computer oder Digitalrechner ist eine Maschine, die Probleme für den Menschen lösen kann, indem sie die ihr gegebenen Befehle ausführt. Eine Befehlssequenz, die beschreibt, wie eine bestimmte Aufgabe auszuführen ist, nennt man **Programm**.*

*Die elektronischen Schaltungen eines Computers verstehen eine begrenzte Menge einfacher Befehle, in die alle Programme konvertiert werden müssen, bevor sie sich ausführen lassen. . . .*

- ▶ Probleme lösen: durch Abarbeiten einfacher **Befehle**
- ▶ Abfolge solcher Befehle ist ein **Programm**
- ▶ Maschine versteht nur ihre eigene **Maschinensprache**

# Befehlssatz und Semantic Gap

... verstehen eine begrenzte Menge einfacher Befehle ...

Typische Beispiele für solche Befehle:

- ▶ addiere die zwei Zahlen in Register R1 und R2
  - ▶ überprüfe, ob das Resultat Null ist
  - ▶ kopiere ein Datenwort von Adresse 13 ins Register R4
- ⇒ extrem niedriges Abstraktionsniveau
- ▶ natürliche Sprache immer mit Kontextwissen  
Beispiel: „vereinbaren Sie einen Termin für die Prüfung“
  - ▶ **Semantic gap:**  
Diskrepanz zu einfachen elementaren Anweisungen
  - ▶ Vermittlung zwischen Mensch und Computer erfordert zusätzliche Abstraktionsebenen und Software

- ▶ Definition solcher Abstraktionsebenen bzw. Schichten
- ▶ mit möglichst einfachen und sauberen Schnittstellen
- ▶ jede Ebene definiert eine neue (mächtigere) **Sprache**
  
- ▶ diverse Optimierungs-Kriterien/Möglichkeiten:
  - ▶ Performanz, Größe, Leistungsaufnahme ...
  - ▶ Kosten: Hardware, Software, Entwicklung ...
  - ▶ Zuverlässigkeit, Wartungsfreundlichkeit, Sicherheit ...

Achtung / Vorsicht:

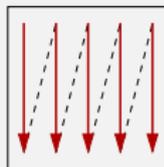
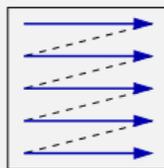
- ▶ Gesamtverständnis erfordert Kenntnisse auf allen Ebenen
- ▶ häufig Rückwirkung von unteren auf obere Ebenen

# Rückwirkung von unteren Ebenen: Arithmetik

```
public class Overflow {
    ...
    public static void main( String[] args ) {
        printInt( 0 );           // 0
        printInt( 1 );           // 1
        printInt( -1 );          // -1
        printInt( 2+(3*4) );     // 14
        printInt( 100*200*300 ); // 6000000
        printInt( 100*200*300*400 ); // -1894967296 (!)
        printDouble( 1.0 );      // 1.0
        printDouble( 0.3 );      // 0.3
        printDouble( 0.1 + 0.1 + 0.1 ); // 0.30000000000000004 (!)
        printDouble( (0.3) - (0.1+0.1+0.1) ); // -5.5E-17 (!)
    }
}
```

# Rückwirkung von unteren Ebenen: Performanz

```
public static double sumRowCol( double[][] matrix ) {  
    int rows = matrix.length;  
    int cols = matrix[0].length;  
    double sum = 0.0;  
    for( int r = 0; r < rows; r++ ) {  
        for( int c = 0; c < cols; c++ ) {  
            sum += matrix[r][c];  
        }  
    }  
    return sum;  
}
```



Matrix creation (5000×5000)

2105 ms

Matrix row-col summation

75 ms

Matrix col-row summation

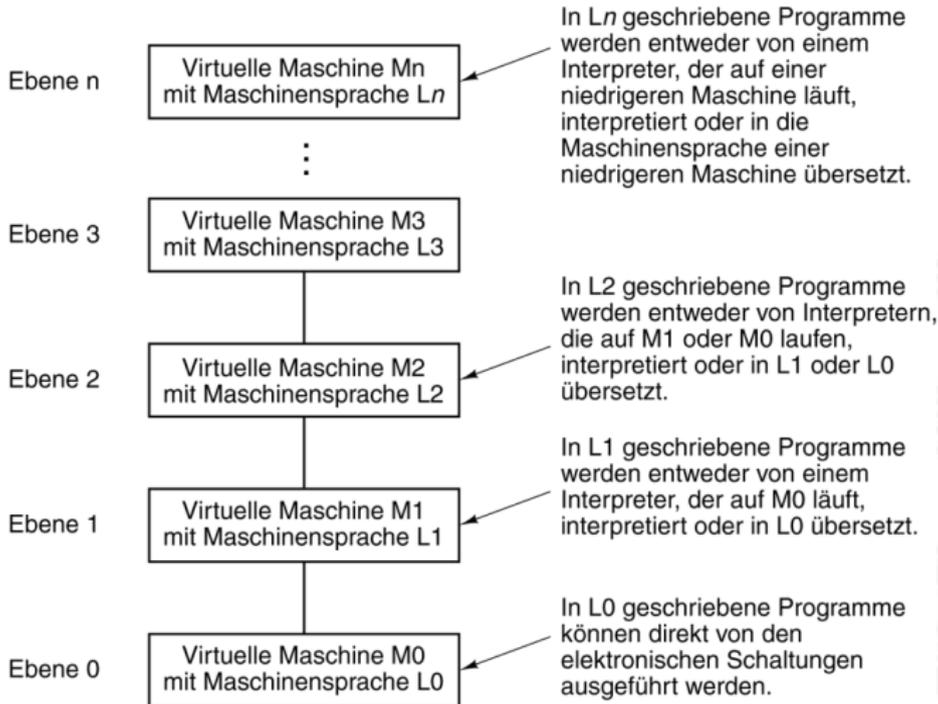
383 ms

⇒ 5 × langsamer

Sum = 600,8473695346258 / 600,8473695342268

⇒ andere Werte

# Maschine mit mehreren Ebenen



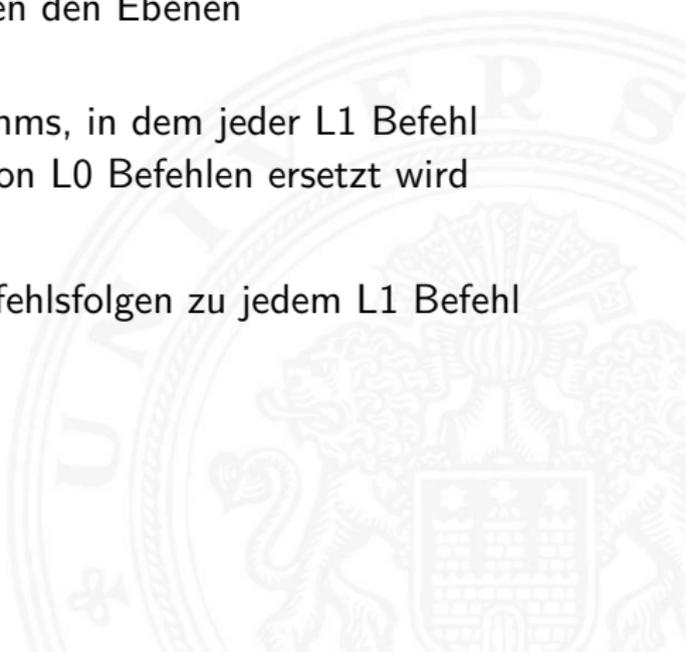
Tanenbaum: *Structured Computer Organization* [TA14]



- ▶ jede Ebene definiert eine neue (mächtigere) Sprache
- ▶ Abstraktionsebene  $\iff$  Sprache
- ▶  $L_0 < L_1 < L_2 < L_3 < \dots$

Software zur Übersetzung zwischen den Ebenen

- ▶ **Compiler:**  
Erzeugen eines neuen Programms, in dem jeder L1 Befehl durch eine zugehörige Folge von L0 Befehlen ersetzt wird
- ▶ **Interpreter:**  
direkte Ausführung der L0 Befehlsfolgen zu jedem L1 Befehl



- ▶ für einen Interpreter sind L1 Befehle einfach nur Daten
- ▶ die dann in die zugehörigen L0 Befehle umgesetzt werden

⇒ dies ist gleichwertig mit einer:

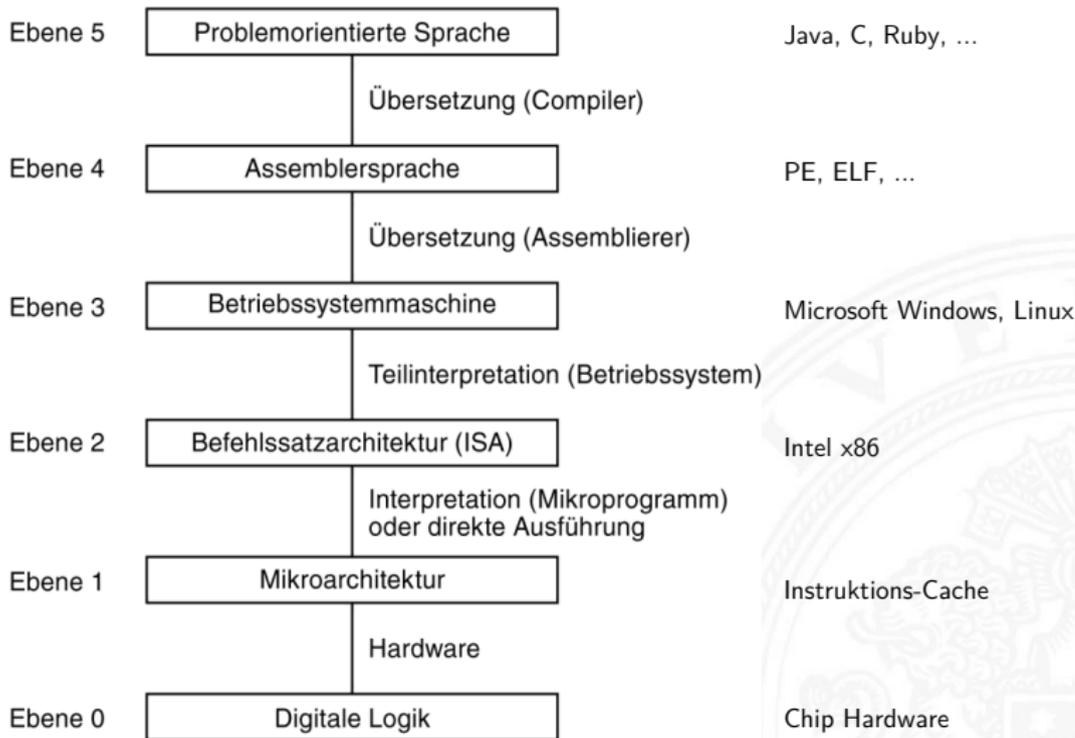
## **Virtuellen Maschine M1 für die Sprache L1**

- ▶ ein Interpreter erlaubt es, jede beliebige Maschine zu simulieren
- ▶ und zwar auf jeder beliebigen (einfacheren) Maschine M0
- ▶ Programmierer muss sich nicht um untere Schichten kümmern
- ▶ Nachteil: die virtuelle Maschine ist meistens langsamer als die echte Maschine M1
- ▶ Maschine M0 kann wiederum eine virtuelle Maschine sein!
- ▶ unterste Schicht ist jeweils die Hardware

# Übliche Einteilung der Ebenen

Anwendungsebene	Hochsprachen (Java, Smalltalk ...)
Assemblerebene	low-level Anwendungsprogrammierung
Betriebssystemebene	Betriebssystem, Systemprogrammierung
Rechnerarchitektur	Schnittstelle zwischen SW und HW, Befehlssatz, Datentypen
Mikroarchitektur	Steuerwerk und Operationswerk: Register, ALU, Speicher ...
Logikebene	Grundsaltungen: Gatter, Flipflops ...
Transistorebene	Elektrotechnik, Transistoren, Chip-Layout
Physikalische Ebene	Geometrien, Chip-Fertigung

# Beispiel: Sechs Ebenen



Anwendungsebene: SE1+SE2, AD ...

Assemblerebene: RSB

Betriebssystemebene: RSB, MB

Rechnerarchitektur: RSB

Mikroarchitektur: RSB

Logikebene: RSB

Device-Level: -



```
/* HelloWorld.c - print a welcome message */  
  
#include <stdio.h>  
  
int main( int argc, char ** argv ) {  
    printf( "Hello, world!\n" );  
    return 0;  
}
```

## Übersetzung

```
gcc -S HelloWorld.c  
gcc -c HelloWorld.c  
gcc -o HelloWorld.exe HelloWorld.c
```

```
.file "HelloWorld.c"
.text
.section .rodata
.LC0:
.string "Hello, world!"
.text
.globl main
.type main, @function
main:
.LFB0:
.cfi_startproc
endbr64
pushq %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq %rsp, %rbp
.cfi_def_cfa_register 6
subq $16, %rsp
movl %edi, -4(%rbp)
movq %rsi, -16(%rbp)
leaq .LC0(%rip), %rax
movq %rax, %rdi
call puts@PLT
movl $0, %eax
leave
.cfi_def_cfa 7, 8
ret
.cfi_endproc
.LFE0:
.size main, .-main
.ident "GCC: (Ubuntu 11.2.0-19ubuntu1) 11.2.0"
.section .note.GNU-stack,"",@progbits
...
```

```
00000000 457f 464c 0102 0001 0000 0000 0000 0000
00000020 0001 003e 0001 0000 0000 0000 0000 0000
00000040 0000 0000 0000 0000 0260 0000 0000 0000
00000060 0000 0000 0040 0000 0000 0040 000e 000d
00000100 0ff3 fa1e 4855 e589 8348 10ec 7d89 48fc
00000120 7589 48f0 058d 0000 0000 8948 e8c7 0000
00000140 0000 00b8 0000 c900 48c3 6c65 6f6c 202c
00000160 6f77 6c72 2164 0000 4347 3a43 2820 6255
00000200 6e75 7574 3120 2e31 2e32 2d30 3931 6275
00000220 6e75 7574 2931 3120 2e31 2e32 0030 0000
00000240 0004 0000 0010 0000 0005 0000 4e47 0055
00000260 0002 c000 0004 0000 0003 0000 0000 0000
00000300 0014 0000 0000 0000 7a01 0052 7801 0110
00000320 0c1b 0807 0190 0000 001c 0000 001c 0000
00000340 0000 0000 0029 0000 4500 100e 0286 0d43
00000360 6006 070c 0008 0000 0000 0000 0000 0000
00000400 0000 0000 0000 0000 0000 0000 0000 0000
00000420 0001 0000 0004 fff1 0000 0000 0000 0000
```

...

```
HelloWorld.o:      file format elf64-x86-64
```

```
Disassembly of section .text:
```

```
0000000000000000 <main>:  
  0:   f3 0f 1e fa          endbr64  
  4:   55                   push   %rbp  
  5:   48 89 e5            mov    %rsp,%rbp  
  8:   48 83 ec 10         sub    $0x10,%rsp  
  c:   89 7d fc            mov    %edi,-0x4(%rbp)  
  f:   48 89 75 f0         mov    %rsi,-0x10(%rbp)  
 13:  48 8d 05 00 00 00 00 lea   0x0(%rip),%rax  
# 1a <main+0x1a>  
 1a:  48 89 c7            mov    %rax,%rdi  
 1d:  e8 00 00 00 00     call   22 <main+0x22>  
 22:  b8 00 00 00 00     mov    $0x0,%eax  
 27:  c9                 leave  
 28:  c3                 ret
```

```
00000000 457f 464c 0102 0001 0000 0000 0000 0000
00000020 0003 003e 0001 0000 1060 0000 0000 0000
00000040 0040 0000 0000 0000 3698 0000 0000 0000
00000060 0000 0000 0040 0038 000d 0040 001f 001e
00000100 0006 0000 0004 0000 0040 0000 0000 0000
00000120 0040 0000 0000 0000 0040 0000 0000 0000
00000140 02d8 0000 0000 0000 02d8 0000 0000 0000
00000160 0008 0000 0000 0000 0003 0000 0004 0000
00000200 0318 0000 0000 0000 0318 0000 0000 0000
00000220 0318 0000 0000 0000 001c 0000 0000 0000
00000240 001c 0000 0000 0000 0001 0000 0000 0000
00000260 0001 0000 0004 0000 0000 0000 0000 0000
00000300 0000 0000 0000 0000 0000 0000 0000 0000
00000320 0628 0000 0000 0000 0628 0000 0000 0000
00000340 1000 0000 0000 0000 0001 0000 0005 0000
00000360 1000 0000 0000 0000 1000 0000 0000 0000
00000400 1000 0000 0000 0000 0181 0000 0000 0000
00000420 0181 0000 0000 0000 1000 0000 0000 0000
```

...

- ▶ eine virtuelle Maschine führt L1 Software aus
  - ▶ und wird mit Software oder Hardware realisiert
- ⇒ Software und Hardware sind logisch äquivalent  
**„Hardware is just petrified Software“**  
– jedenfalls in Bezug auf L1 Programmausführung

Karen Panetta Lentz

Entscheidung für Software- oder Hardwarerealisierung?

- ▶ abhängig von vielen Faktoren, u.a.
- ▶ Kosten, Performanz, Zuverlässigkeit
- ▶ Anzahl der (vermuteten) Änderungen und Updates
- ▶ Sicherheit gegen Kopieren ...
- ▶ Beispiele: Virtualisierung im RZ = Software  
(High-Level Synthese+IP+) FPGAs = Hardware



- ▶ **Information**  $\sim$  abstrakter Gehalt einer Aussage
- ▶ Die Aussage selbst, mit der die Information dargestellt bzw. übertragen wird, ist eine **Repräsentation** der Information
- ▶ im Kontext der Informationsverarbeitung / -übertragung:  
**Nachricht**
- ▶ Das Ermitteln der Information aus einer Repräsentation heißt **Interpretation**
- ▶ Das Verbinden einer Information mit ihrer Bedeutung in der realen Welt heißt **Verstehen**

Beispiel: Mit der Information „25“ sei die abstrakte Zahl gemeint, die sich aber nur durch eine Repräsentation angeben lässt:

- ▶ Text deutsch:                    fünfundzwanzig
- ▶ Text englisch:                    twentyfive
- ...
- ▶ Zahl römisch:                    XXV
- ▶ Zahl dezimal:                    25
- ▶ Zahl binär:                    11001
- ▶ Zahl Dreiersystem:              221
- ...
- ▶ Morse-Code:                    •• --- •••••

# Interpretation: Information vs. Repräsentation

- ▶ Wo auch immer Repräsentationen auftreten, meinen wir eigentlich die Information, z.B.:

$$5 \cdot (2 + 3) = 25$$

- ▶ Die Information selbst kann man überhaupt nicht notieren (!)
- ▶ Es muss immer Absprachen geben über die verwendete Repräsentation. Im obigen Beispiel ist implizit die Dezimaldarstellung gemeint, man muss also die Dezimalziffern und das Stellenwertsystem kennen.
- ▶ Repräsentation ist häufig mehrstufig, z.B.

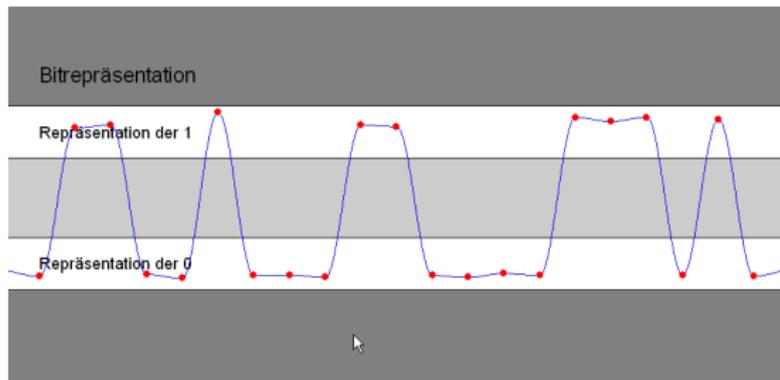
Zahl:	Dezimalzahl	347
Ziffer:	4-bit binär	0011 0100 0111 (BCD)
Bit:	elektrische Spannung	0,1V 0,1V 2,5V 2,5V ...



In jeder (Abstraktions-) Ebene gibt es beliebig viele Alternativen der Repräsentation

- ▶ Auswahl der jeweils effizientesten Repräsentation
- ▶ unterschiedliche Repräsentationen je nach Ebene
- ▶ Beispiel: Repräsentation der Zahl  $\pi = 3,1415 \dots$  im
  - ▶ x86 Prozessor      80-bit Binärdaten, Spannungen
  - ▶ Hauptspeicher    64-bit Binärdaten, Spannungen
  - ▶ Festplatte          codierte Zahl, magnetische Bereiche
  - ▶ CD-ROM            codierte Zahl, Land/Pits-Bereiche
  - ▶ Papier              Text, „3,14159265...“
  - ▶ ...

# Repräsentation: digitale und analoge Welt



Beispiel: Binärwerte in  
2,5V CMOS-Technologie

K. von der Heide [Hei05]  
Interaktives Skript T1, demobitrep

- ▶ Spannungsverlauf des Signals ist kontinuierlich
- ▶ Abtastung zu bestimmten Zeitpunkten
- ▶ Quantisierung über abgegrenzte Wertebereiche:
  - ▶  $0,0V \leq v(t) \leq 0,7V$ : Interpretation als 0
  - ▶  $1,7V \leq v(t) \leq 2,5V$ : Interpretation als 1
  - ▶ außerhalb und innerhalb: ungültige Werte



▶ Aussagen

N1 Er besucht General Motors

N2 Unwetter am Alpenostrand

N3 Sie nimmt ihren Hut

▶ Alle Aussagen sind aber doppel/mehrdeutig:

N1 Firma? Militär?

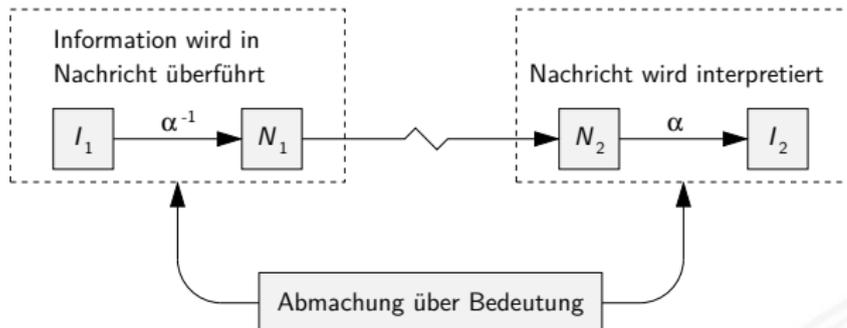
N2 Alpen-Ostrand? Alpeno-Strand?

N3 tatsächlich oder im übertragenen Sinn?

⇒ **Interpretation:** Es handelt sich um drei **Nachrichten**, die jeweils zwei verschiedene **Informationen** enthalten



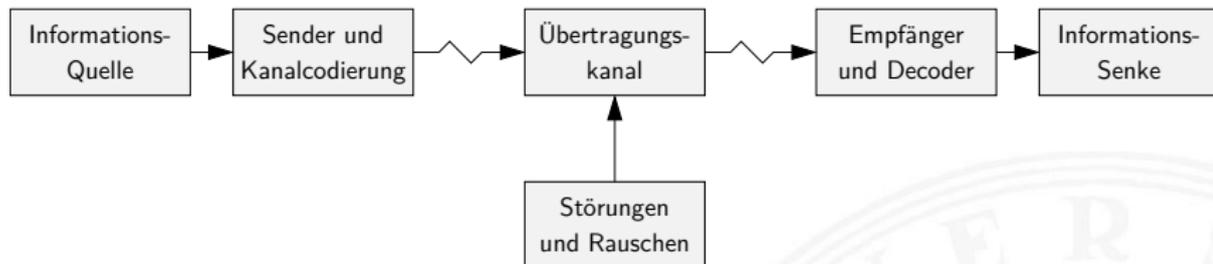
- ▶ **Information:** Wissen um oder Kenntnis über Sachverhalte und Vorgänge – als Begriff nicht informationstheoretisch abgestützt, sondern an umgangssprachlicher Bedeutung orientiert
- ▶ **Nachricht:** Zeichen oder Funktionen, die Informationen zum Zweck der Weitergabe aufgrund bekannter oder unterstellter Abmachungen darstellen (DIN 44 300)
- ▶ Beispiel für eine Nachricht:  
Temperaturangabe in Grad Celsius oder Fahrenheit
- ▶ Die Nachricht ist also eine Darstellung von Informationen und nicht der Übermittlungsvorgang



Beschreibung der **Informationsübermittlung**:

- ▶ Abbildung  $\alpha^{-1}$  erzeugt Nachricht  $N_1$  aus Information  $I_1$
- ▶ Übertragung der Nachricht an den Zielort
- ▶ Interpretation  $\alpha$  der Nachricht  $N_2$  liefert die Information  $I_2$

## Nachrichtentechnisches Modell: **Störungen** bei der Übertragung

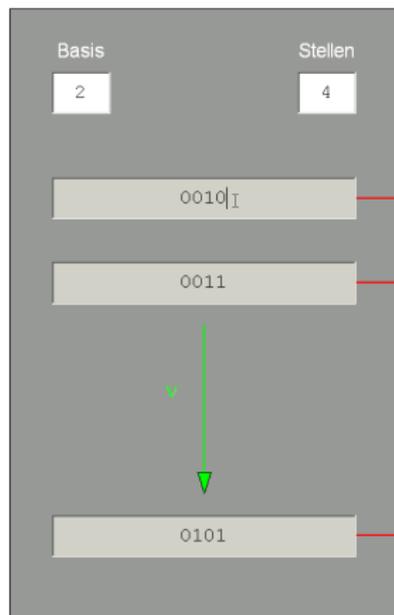


### Beispiele

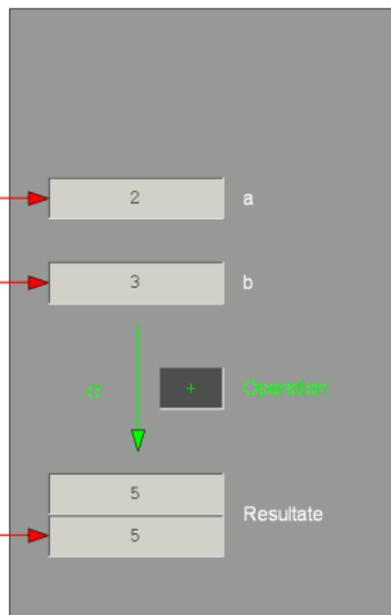
- ▶ Bitfehler beim Speichern
- ▶ Störungen beim Funkverkehr
- ▶ Schmutz oder Kratzer auf einer CD/DVD
- ▶ usw.

# Verarbeitung von Information

## Repräsentation



## Information



Repräsentation natürlicher Zahlen durch Stellenwertsysteme

K. von der Heide [Hei05]  
Interaktives Skript T1,  
infoprepres

Ergibt  $\alpha$  gefolgt von  $\sigma$  dasselbe wie  $\nu$  gefolgt von  $\alpha'$ ,  
dann heißt  $\nu$  **informationstreu**  $\sigma(\alpha(r)) = \alpha'(\nu(r))$

- ▶  $\alpha'$  ist die Interpretation des Resultats der Operation  $\nu$   
häufig sind  $\alpha$  und  $\alpha'$  gleich, aber nicht immer
- ▶ ist  $\sigma$  injektiv, so nennen wir  $\nu$  eine **Umschlüsselung**  
durch die Verarbeitung  $\sigma$  geht keine Information verloren
- ▶ ist  $\nu$  injektiv, so nennen wir  $\nu$  eine **Umcodierung**
- ▶ wenn  $\sigma$  innere Verknüpfung der Menge  $\mathcal{J}$  und  $\nu$  innere  
Verknüpfung der Menge  $\mathcal{R}$ , dann ist  $\alpha$  ein **Homomorphismus**  
der algebraischen Strukturen  $(\mathcal{J}, \sigma)$  und  $(\mathcal{R}, \nu)$
- ▶ ist  $\sigma$  bijektiv, liegt ein **Isomorphismus** vor



Welche mathematischen Eigenschaften gelten bei der Informationsverarbeitung, in der gewählten Repräsentation?

Beispiele

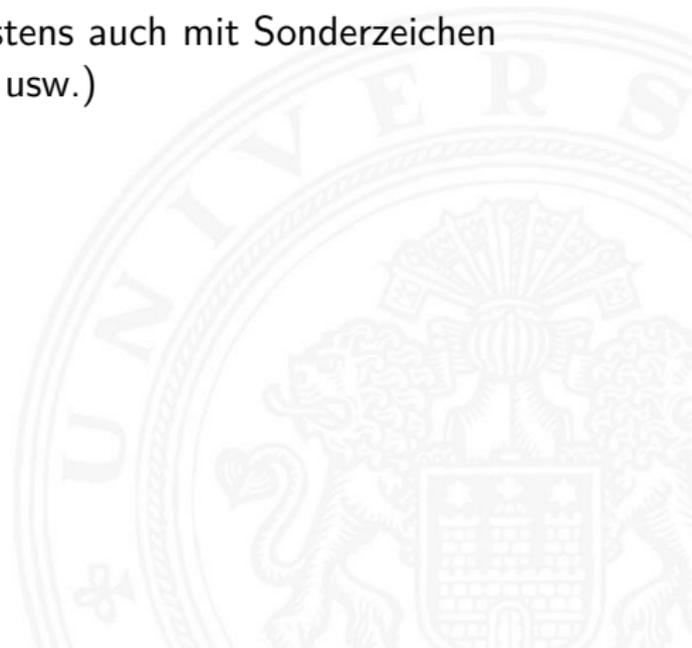
- ▶ Gilt  $x^2 \geq 0$ ?
  - ▶ float: ja
  - ▶ signed integer: nein
  
- ▶ Gilt  $(x + y) + z = x + (y + z)$ ?
  - ▶ integer: ja
  - ▶ float: nein
$$1.0E20 + (-1.0E20 + 3.14) = 0$$
  
- ▶ Details folgen später



- ▶ **Zeichen:** engl. *character*  
Element  $z$  aus einer zur Darstellung von Information vereinbarten, einer Abmachung unterliegenden, endlichen Menge  $Z$  von Elementen
- ▶ Die Menge  $Z$  heißt **Zeichensatz** oder **Zeichenvorrat**  
engl. *character set*
- ▶ Beispiele
  - ▶  $Z_1 = \{0, 1\}$
  - ▶  $Z_2 = \{0, 1, 2, \dots, 9, A, B, C, D, E, F\}$
  - ▶  $Z_3 = \{\alpha, \beta, \gamma, \dots, \omega\}$
  - ▶  $Z_4 = \{CR, LF\}$



- ▶ **Numerischer Zeichensatz:** Zeichenvorrat aus Ziffern und/oder Sonderzeichen zur Darstellung von Zahlen
- ▶ **Alphanumerischer Zeichensatz:** Zeichensatz aus (mindestens) den Dezimalziffern und den Buchstaben des gewöhnlichen Alphabets, meistens auch mit Sonderzeichen (Leerzeichen, Punkt, Komma usw.)





- ▶ **Binärzeichen:** engl. *binary element, binary digit, bit*  
Jedes der Zeichen aus einem Vorrat / aus einer Menge von zwei Symbolen
  
- ▶ Beispiele
  - ▶  $Z_1 = \{0, 1\}$
  - ▶  $Z_2 = \{\text{high, low}\}$
  - ▶  $Z_3 = \{\text{rot, grün}\}$
  - ▶  $Z_4 = \{+, -\}$



- ▶ **Alphabet:** engl. *alphabet*  
Ein in vereinbarter Reihenfolge geordneter Zeichenvorrat  $\mathcal{A} = \mathcal{Z}$
- ▶ Beispiele
  - ▶  $\mathcal{A}_1 = \{0, 1, 2, \dots, 9\}$
  - ▶  $\mathcal{A}_2 = \{\text{Mo, Di, Mi, Do, Fr, Sa, So}\}$
  - ▶  $\mathcal{A}_3 = \{\text{A, B, C, } \dots, \text{Z}\}$



- ▶ **Zeichenkette:** engl. *string*  
Eine Folge von Zeichen
- ▶ **Wort:** engl. *word*  
Eine Folge von Zeichen, die in einem gegebenen Zusammenhang als Einheit bezeichnet wird
- ▶ Worte mit 8 bit werden als **Byte** bezeichnet
- ▶ **Stelle:** engl. *position*  
Die Lage/Position eines Zeichens innerhalb einer Zeichenkette
- ▶ Beispiel
  - ▶ `s = H e l l o , w o r l d !`



- 3. Natürliche Zahlen engl. *integer numbers*
  - Festkommazahlen engl. *fixed point numbers*
  - Gleitkommazahlen engl. *floating point numbers*
  
- 4. Arithmetik
  
- 5. Aspekte der Textcodierung
  - Ad-hoc Codierungen
  - ASCII und ISO-8859-1
  - Unicode
  
- 13. Pointer (Referenzen, Maschinenadressen)



- [TA14] A.S. Tanenbaum, T. Austin: *Rechnerarchitektur – Von der digitalen Logik zum Parallelrechner*. 6. Auflage, Pearson Deutschland GmbH, 2014. ISBN 978–3–8689–4238–5
- [Hei05] K. von der Heide: *Vorlesung: Technische Informatik 1 — interaktives Skript*. Universität Hamburg, FB Informatik, 2005. [tams.informatik.uni-hamburg.de/lectures/2004ws/vorlesung/t1](http://tams.informatik.uni-hamburg.de/lectures/2004ws/vorlesung/t1)