64-040 Modul InfB-RSB
Rechnerstrukturen und Betriebssysteme

https://tams.informatik.uni-hamburg.de/lectures/2023ws/vorlesung/rsb

– Kapitel 2 –

Andreas Mäder

Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik
Technische Aspekte Multimodaler Systeme

Wintersemester 2023/2024

Definition: Digitalrechner

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 auzuführen 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**

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

$\Rightarrow$ extrem niedriges Abstraktionsniveau

- natürliche Sprache immer mit Kontextwissen
  Beispiel: „Vereinbaren Sie einen Besprechungstermin“ $\Rightarrow$ Wo? Wer? Thema?

- **Semantic gap**:
  Diskrepanz zu einfachen elementaren Anweisungen
- Vermittlung zwischen Mensch und Computer erfordert zusätzliche Abstraktionsebenen und Software

Rechnerarchitektur bzw. -organisation

- 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

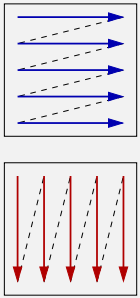
Rückwirkung von unteren Ebenen: Arithmetik


```
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 );             // 0.3000000000000004
    printDouble( (0.3) - (0.1+0.1+0.1) );       // -5.5E-17
  }
}
```

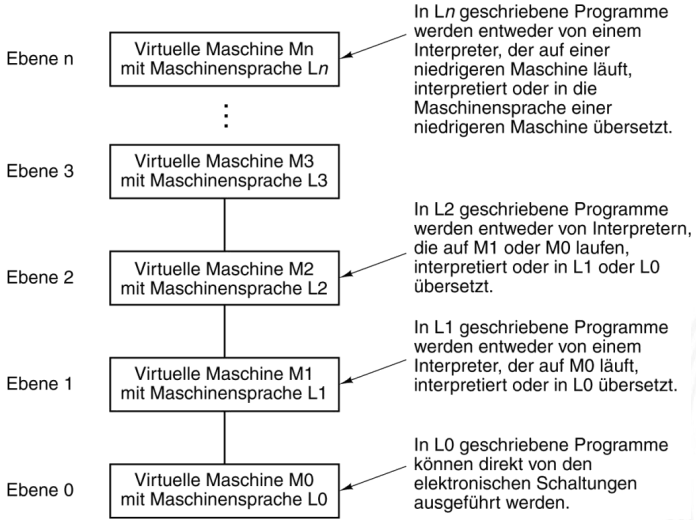
Rückwirkung von unteren Ebenen: Performanz


```
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]

Abstraktionsebenen und Sprachen

- jede Ebene definiert eine neue (mächtigere) Sprache
- Abstraktionsebene $\Longleftrightarrow$ Sprache
- L0 $<$ L1 $<$ L2 $<$ L3 $<$ ...

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

Virtuelle Maschine

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

$\Rightarrow$ dies ist gleichwertig mit einer:

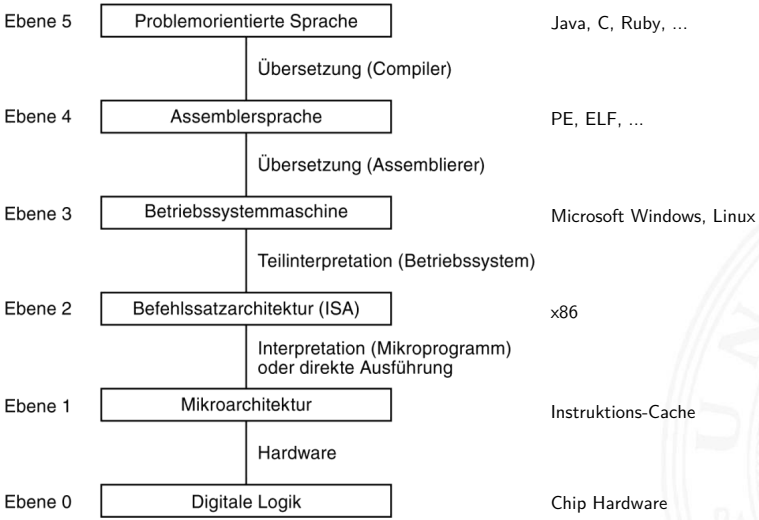
Virtuellen Maschine M1 für die Sprache L1

- 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, C, Ruby, ...)
Assemblerebene	low-level Anwendungsprogrammierung
Betriebssystemebene	Betriebssystem, Systemprogrammierung
Rechnerarchitektur	Schnittstelle zwischen SW und HW: Befehlssatz, Datentypen
Mikroarchitektur	Steuerwerk, Operationswerk: Register, ALU, Speicher, Busse ...
Logikebene	Grundschaltungen: Gatter, Flipflops ...
Transistorebene	Elektrotechnik: Transistoren, Widerstände, Kapazitäten ...
Physikalische Ebene	Chip-Layout: Geometrien für die IC-Fertigung

Beispiel: Sechs Ebenen



Hinweis: Ebenen vs. Vorlesungen im BSc-Studiengang

Anwendungsebene:	SE1+SE2, AD ...
Assemblerebene:	RSB
Betriebssystemebene:	RSB, MB
Rechnerarchitektur:	RSB
Mikroarchitektur:	RSB
Logikebene:	RSB
Device-Level:	–

```c
/* 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
```

HelloWorld: Assemblerebene `cat HelloWorld.s`


```
        .file   "HelloWorld.c"
        .text
        .section        .rodata
.LC0:
        .string "Hello, world!"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        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: (Debian 12.2.0-14) 12.2.0"
        .section        .note.GNU-stack,"",@progbits
```

HelloWorld: Objectcode

`od -x HelloWorld.o`

```
0000000 457f 464c 0102 0001 0000 0000 0000 0000
0000020 0001 003e 0001 0000 0000 0000 0000 0000
0000040 0000 0000 0000 0000 0228 0000 0000 0000
0000060 0000 0000 0040 0000 0000 0040 000d 000c
0000100 4855 e589 8348 10ec 7d89 48fc 7589 48f0
0000120 058d 0000 0000 8948 e8c7 0000 0000 00b8
0000140 0000 c900 48c3 6c65 6f6c 202c 6f77 6c72
0000160 2164 0000 4347 3a43 2820 6544 6962 6e61
0000200 3120 2e32 2e32 2d30 3431 2029 3231 322e
0000220 302e 0000 0000 0000 0014 0000 0000 0000
0000240 7a01 0052 7801 0110 0c1b 0807 0190 0000
0000260 001c 0000 001c 0000 0000 0000 0025 0000
0000300 4100 100e 0286 0d43 6006 070c 0008 0000
0000320 0000 0000 0000 0000 0000 0000 0000 0000
0000340 0000 0000 0000 0000 0001 0000 0004 fff1
0000360 0000 0000 0000 0000 0000 0000 0000 0000
0000400 0000 0000 0003 0001 0000 0000 0000 0000
...
```

HelloWorld: Disassemblieren `objdump -d HelloWorld.o`

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


Disassembly of section .text:

0000000000000000 <main>:
   0:   55                      push   %rbp
   1:   48 89 e5                mov    %rsp,%rbp
   4:   48 83 ec 10             sub    $0x10,%rsp
   8:   89 7d fc                mov    %edi,-0x4(%rbp)
   b:   48 89 75 f0             mov    %rsi,-0x10(%rbp)
   f:   48 8d 05 00 00 00 00    lea    0x0(%rip),%rax    # 16 <main+0x16>
  16:   48 89 c7                mov    %rax,%rdi
  19:   e8 00 00 00 00          call   1e <main+0x1e>
  1e:   b8 00 00 00 00          mov    $0x0,%eax
  23:   c9                      leave
  24:   c3                      ret
```

HelloWorld: Maschinencode `od -x HelloWorld.exe`

```
0000000 457f 464c 0102 0001 0000 0000 0000 0000
0000020 0003 003e 0001 0000 1050 0000 0000 0000
0000040 0040 0000 0000 0000 3698 0000 0000 0000
0000060 0000 0000 0040 0038 000d 0040 001f 001e
0000100 0006 0000 0004 0000 0040 0000 0000 0000
0000120 0040 0000 0000 0000 0040 0000 0000 0000
0000140 02d8 0000 0000 0000 02d8 0000 0000 0000
0000160 0008 0000 0000 0000 0003 0000 0004 0000
0000200 0318 0000 0000 0000 0318 0000 0000 0000
0000220 0318 0000 0000 0000 001c 0000 0000 0000
0000240 001c 0000 0000 0000 0001 0000 0000 0000
0000260 0001 0000 0004 0000 0000 0000 0000 0000
0000300 0000 0000 0000 0000 0000 0000 0000 0000
0000320 0618 0000 0000 0000 0618 0000 0000 0000
0000340 1000 0000 0000 0000 0001 0000 0005 0000
0000360 1000 0000 0000 0000 1000 0000 0000 0000
0000400 1000 0000 0000 0000 0169 0000 0000 0000
...
```

Hardware: *„Versteinerte Software“*

- eine virtuelle Maschine führt L1 Software aus
- und wird mit Software oder Hardware realisiert

$\Rightarrow$ 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 $\Rightarrow$ Software
  (High-Level Synthese+IP+) FPGAs $\Rightarrow$ Hardware

Information

- **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**
- **Interpretation** ist das Ermitteln der Information aus einer Repräsentation
- **Verstehen** ist das Verbinden einer Information mit ihrer Bedeutung in der realen Welt

Repräsentation: Beispiele

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 $2 \cdot 10^1 + 5 \cdot 10^0 = 25$
- Zahl binär: 11001 $1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^0 = 25$
- Zahl Dreiersystem: 221 $2 \cdot 3^2 + 2 \cdot 3^1 + 1 \cdot 3^0 = 25$

...

siehe: *3.2 Stellenwertsystem*

- Morse-Code: ●●−−− ●●●●●

Interpretation: Information vs. Repräsentation

- Wo auch immer Repräsentationen auftreten, ist eigentlich die Information gemeint, beispielsweise:

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

- Die Information selbst kann man überhaupt nicht notieren!
- Es muss immer Absprachen über die verwendete Repräsentation geben.

  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:	BCD, 4-bit binär	0011 0100 0111
Bit:	elektrische Spannung	0,1V 0,1V 2,5V 2,5V ...

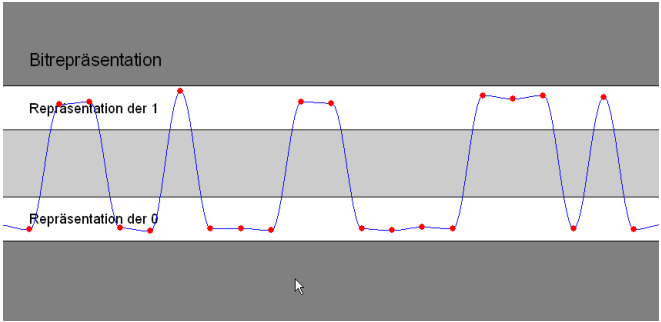
Repräsentation: Ebenen

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 $\pi = 3{,}1415\ldots$ 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 2,5 V CMOS-Technologie

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

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

Information vs. Nachricht

- 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?

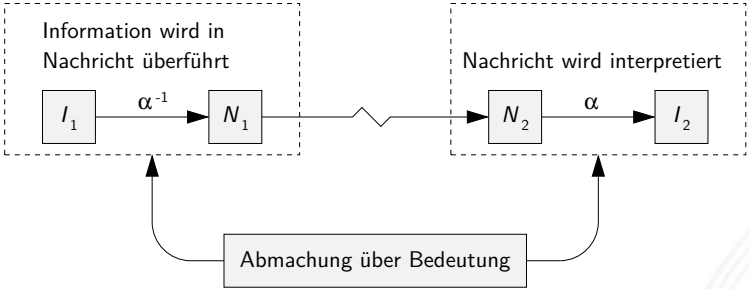
⇒ **Interpretation**: Es handelt sich um drei **Nachrichten**, die jeweils zwei verschiedene **Informationen** enthalten

Information vs. Nachricht (cont.)

- **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: 21° Temperaturangabe in Grad Celsius oder Fahrenheit
- Die Nachricht ist also eine Darstellung von Informationen und nicht der Übermittlungsvorgang

Modell der Informationsübertragung

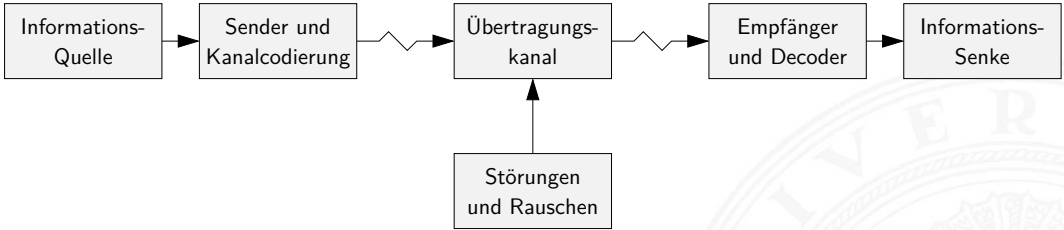


Beschreibung der **Informationsübermittlung**:

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

Modell der Informationsübertragung (cont.)

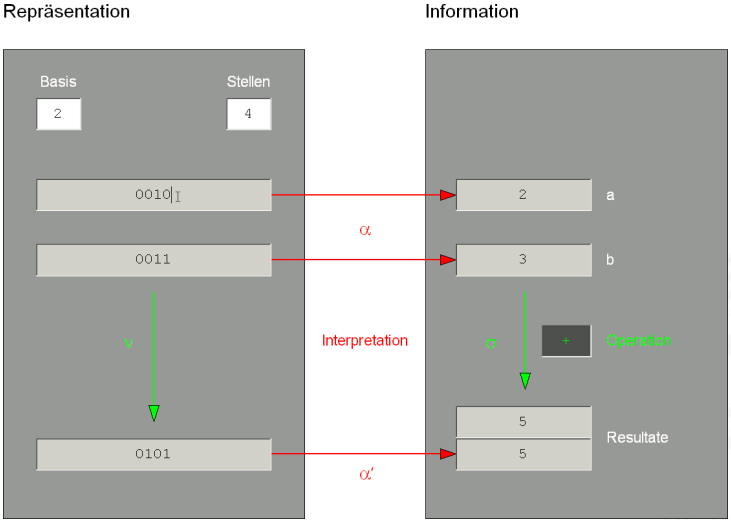
Nachrichtentechnisches Modell: **Störungen** bei der Übertragung



Beispiele

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

Verarbeitung von Information



Repräsentation natürlicher Zahlen durch Stellenwertsysteme

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

Informationstreue

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

- α' ist die Interpretation des Resultats der Operation ν
  häufig sind α und α' gleich, aber nicht immer

- ist σ injektiv, so nennen wir ν eine **Umschlüsselung**
  durch die Verarbeitung σ geht keine Information verloren
- ist ν injektiv, so nennen wir ν eine **Umcodierung**
- wenn σ innere Verknüpfung der Menge $\mathcal{J}$ und ν innere Verknüpfung der Menge $\mathcal{R}$,
  dann ist α ein **Homomorphismus** der algebraischen Strukturen $(\mathcal{J},\sigma)$ und $(\mathcal{R},\nu)$
- ist σ bijektiv, liegt ein **Isomorphismus** vor

Informationstreue (cont.)

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

Beschreibung von Information durch Zeichen

- **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
  - $\mathcal{Z}_1 = \{0, 1\}$
  - $\mathcal{Z}_2 = \{0, 1, 2, \ldots, 9, A, B, C, D, E, F\}$
  - $\mathcal{Z}_3 = \{\alpha, \beta, \gamma, \ldots, \omega\}$
  - $\mathcal{Z}_4 = \{CR, LF\}$
- **Numerischer Zeichensatz**: Zeichenvorrat aus Ziffern und/oder Sonderzeichen zur Darstellung von Zahlen
- **Alphanumerischer Zeichensatz**: Zeichensatz aus (mindestens) den Dezimalziffern und den Buchstaben des Alphabets, meistens auch mit Sonderzeichen (Leerzeichen, Punkt, Komma usw.)

Binärzeichen / Alphabet

- **Binärzeichen**: engl. *binary element, binary digit, bit*
  Jedes der Zeichen aus einem Vorrat / aus einer Menge von zwei Symbolen
  Beispiele
  - $\mathcal{Z}_1 = \{0, 1\}$
  - $\mathcal{Z}_2 = \{\text{high}, \text{low}\}$
  - $\mathcal{Z}_3 = \{\text{rot}, \text{grün}\}$
  - $\mathcal{Z}_4 = \{+, -\}$

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

Zeichenketten

- **Zeichenkette**: Eine Folge von Zeichen — engl. *string*
- **Wort**: — engl. *word*
  Zeichenkette, die in einem gegebenen Zusammenhang als Einheit bezeichnet wird
- Worte aus 8 Binärzeichen (8 bit) werden als **Byte** bezeichnet
- **Stelle**: Die Position eines Zeichens in einer Zeichenkette — engl. *position*

Beispiele

- $s_1 =$ `Hello, world!`
- $s_2 =$ `das sind vier Worte`
- $s_3 =$ `die Zeichenkette hat 32 Stellen!`

Zahlen- und Zeichendarstellung in RSB

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)

Literatur

[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, Vorlesungsskript. `tams.informatik.uni-hamburg.de/lectures/2004ws/vorlesung/t1`