Kurzwiederholung

Informatik I, Teil I: Grundkonzepte

Foliensatz von Marc-Oliver Pahl, überarbeitet von W. Küchlin
zu Informatik I, Tübingen 2003/04

Neue Version vom 18.2.2004

- 1 **Bit** (Zustand 0 oder 1)
- 8 Bit = 1 **Byte**
- 2^10 Byte = 1024 Byte = 1**kB** (KiloByte)
- 2^10 * 2^10 Byte = 1024 * 1024 Byte = 1**MB** (MegaByte)
- 2^10 * 2^10 * 2^10 Byte = 1 **GB** (GigaByte)

- **Binär rechnen** (Einerkomplement, Zweierkomplement, ...)
- **Zahlkonversion** (Dual -> Hexadezimal u.s.w.)
- **Bitorder** (little Endian (Intel)/ big Endian (Mac))
- **ASCII**-Zeichensatz

- 1 **Halbwort** (short) = 2 Byte = 16 Bit
- 1 **Wort** (word) = 4 Byte = 32 Bit
- 1 **Doppelwort** (double, long) = 8 Byte = 64 Bit

- **Speicherzugriff**
  - **Aligned** (nur an Wortgrenzen)
  - Not aligned (frei)

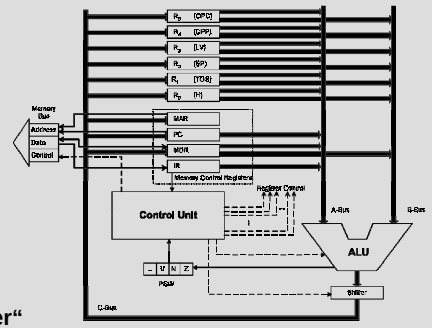
- Es gibt **feste Datentypen**, z.B.
  - char, int, float, double, ...
- IEEE-754

	Sign	Exponent	Fraction	Bias
Single Precision	1 [31]	8 [30-23]	23 [22-00]	127
Double Precision	1 [63]	11 [62-52]	52 [51-00]	1023

  - Beispiel: 0.75 dezimal ergibt in IEEE-754-single precision:

0	01111110	10000000000000000000000

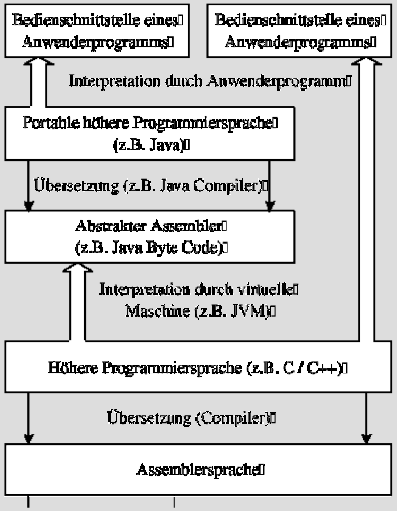
Hardware



„Assembler“

```
Instruction format // code layout (bits)

LOAD R ADDR      [8 | 8 | 16] = 32
STOR R ADDR      [8 | 8 | 16] = 32
BLOD R ADDR      [8 | 8 | 16] = 32
BSTR R ADDR      [8 | 8 | 16] = 32
OP Rc Ra Rb      [8 | 8 | 8 | 8] = 32 // Rc = Ra OP Rb ;
OPI Rc Ra V8     [8 | 8 | 8 | 8] = 32 // Rc = Ra OP V8, "immediate" ;
LODI R V16       [8 | 8 | 16] = 32
```



- Boole‘sche Algebra (a ^ b, a v b, ...)

$$\begin{aligned} \neg\neg a &\simeq a \\ \neg(a \wedge b) &\simeq \neg a \vee \neg b \\ \neg(a \vee b) &\simeq \neg a \wedge \neg b \\ a \Rightarrow b &\simeq \neg a \vee b \end{aligned}$$

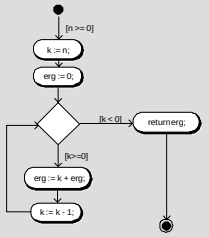
  – *Unterschiedliche Bindung durch Operatoren (Präzedenz)*

- Algorithmus
  - Spezifikation
    - Eingabe-
    - Ausgabe-
  - Durchführbarkeit
    - endlich
    - effektiv
    - determiniert
  - Korrektheit
    - partielle Korrektheit
    - Terminierung

- Pseudocode

```
sum_rek(n){
  if (n == 0) then return 0; fi
  return ( n + summe_rek( n-1 ) );
}
```

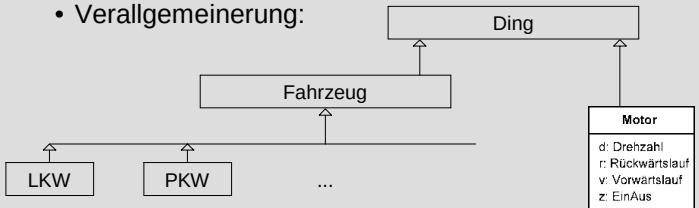
- UML Activity charts/ Flussdiagramme



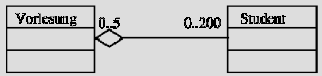
- Verifikation
  - Rekursive Programme meist Induktion
  - Iterative Programme z.B. Floyd

- Softwareentwicklung
  - Analyse
    - Problem in der Welt -> abstraktes Modell
  - Entwurf/ Design
    - Abstraktes Modell -> z.B. Java-Klassendiagramm
  - Implementierung
    - z.B. Java-Klassendiagramm -> Java-Code

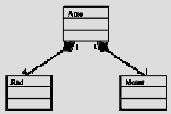
- Modellierung
  - UML
    - Verallgemeinerung:



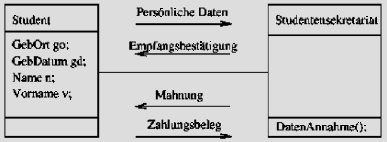
    - Aggregation:



    - Komposition:



- Modellierung
  - UML
    - Kollaboration:



    - Alle Typen sind in einem Diagramm kombinierbar