Teil 1
Kapitel 2

Rechner im Überblick

2.1 Rechnersichten

2.2 Rechnerorganisation: Aufbau und Funktionsweise

2.1 Rechnersichten

Modellierung eines Rechners

„Zusammenspiel zwischen verschiedenen Ebenen"

Neue Begriffe

Compiler, Vollcompiler
Interpreter
Hardware/Software Schnittstelle

Höhere Sicht eines Rechners

Sicht eines C-Programmierers:

Rechner ist ein schwarzer Kasten mit der Eigenschaft:

Bei Eingabe eines korrekten C-Programms P und dazugehörigen Parametern reagiert der Rechner „wie gewünscht".

Hierarchie von Ebenen

Es gibt verschiedene Sichten eines Rechners.

Sichten sind gemäß ihrer Abstraktion hierarchisch angeordnet
(beginnend bei Ebene 1, der digitalen Ebene)

Rechner = Hierarchie von Ebenen

Modellierung eines CISC-Rechners

(CISC = Complex Instruction Set Computer)

Verschiedene Ebenen und Sprachen

Höhere Programmiersprachen
C++, Java, Fortran, ...

Assembler
symbolische Notation der bisher vorhandenen Befehle

Betriebssystemebene
zusätzliche Dienste (z.B Speicherorganisation, Dateiverwaltung)

Maschinensprache
unterste frei zugängliche Sprache; Befehle sind Folgen über 0 und 1

Mikroprogrammebene
Instruktionen zum Setzen von Steuersignalen

digitale Ebene
die eigentliche Hardware

Mit Ausnahme der untersten Ebene gehört zu jeder Ebene eine Sprache

Modellierung eines RISC-Rechners

(RISC = Reduced Instruction Set Computer)

Verschiedene Ebenen und Sprachen

Höhere Programmiersprachen
C++, Java, Fortran, ...

Assembler
symbolische Notation der bisher vorhandenen Befehle

Betriebssystemebene
zusätzliche Dienste (z.B Speicherorganisation, Dateiverwaltung)

Maschinensprache
unterste frei zugängliche Sprache; Befehle sind Folgen über 0 und 1

digitale Ebene
die eigentliche Hardware

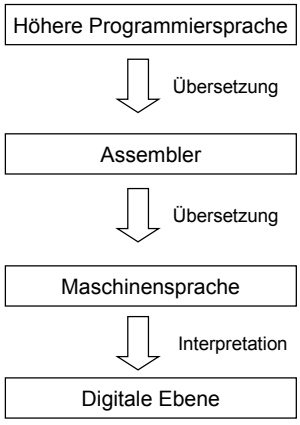
Virtueller Rechner

Jede Ebene (mit Ausnahme der untersten) ist ein **"virtueller Rechner"** M_i, zu der eine **Sprache** L_i gehört.

Programme der Sprache L_i werden
in Programme der Sprache L_j ($j<i$) **übersetzt** oder
in die Sprache L_j ($j<i$) **interpretiert**.

Die virtuelle Maschine M_i gaukelt dem Benutzer vor, seine in L_i geschriebenen Programme würden direkt auf Hardware ausgeführt.

Hierarchie virtueller Rechner: Illustration




```
a = b + 3;
```

```
LOAD ACC 14      // b in den Akku laden
ADDI 3           // Konstante 3 addieren
STORE ACC 13     // Ergebnis aus Akku in a
                 // abspeichern
```

```
1001 0001 0000 0000 0000 0000 0000 1110
1101 0000 0000 0000 0000 0000 0000 0011
1000 0001 0000 0000 0000 0000 0000 1101
```

Übersetzung versus Interpretation

Übersetzung

Ein **L_i-Compiler** ist eine Software, die ein in der Sprache L_i geschriebenes Programm einliest und ein äquivalentes Programm der Sprache L_j ($j < i$) ausgibt

Ist L_j die Maschinensprache, so spricht man von einem **L_i-Vollcompiler.**

Ausführung eines L_i-Programms

Transformation in ein äquivalentes L_j-Programm ($j < i$)

Ausführung des L_j-Programms

Interpretation

cmd := erste Instruktion des L_i-Programms;

repeat

transformiere *cmd* in eine Befehlssequenz *cmd2* der Sprache L_j;

führe *cmd2* auf Ebene j aus;

cmd := nächste auszuführende Instruktion des L_i-Programms;

until L_i-Programm abgearbeitet;

Übersetzung versus Interpretation ff

Interpreter sind einfacher zu implementieren als Compiler

- geeignet für Prototyp-Implementierungen von neuen Sprachen

Compilation ist nicht immer möglich

- Ausführung von Programmen in Maschinensprache nur durch Interpretation möglich

Übersetzte Programme sind schneller als interpretierte Programme

- Codeoptimierung bei Übersetzung möglich

(vollständige) Compilation nicht immer erwünscht

- Systemunabhängigkeit (Portabilität)

Performanzmaße eines Rechners