Technische Informatik 1

Prof. Dr. Rolf Drechsler

Christina Plump

Überblick

Teil 1: Der Rechneraufbau (Kapitel 2-5)

- **Rechner im Überblick**
  - **Rechnersichten**
  - **Rechnerorganisation: Aufbau und Funktionsweise**
  - **Assembler**
- RISC – CISC, Pipelining
- Speicherhierarchie
- Parallelverarbeitung

Teil 2: Der Funktionalitätsaufbau (Kapitel 6-12)

- Kodierung von Zeichen und Zahlen
- Grundbegriffe, Boolesche Funktionen
- Darstellungsmöglichkeiten
- Schaltkreise, Synthese, spezielle Schaltkreise

Kapitel 2: Rechner im Überblick

Rechnersichten

Rechnerorganisation: Aufbau und Funktionsweise

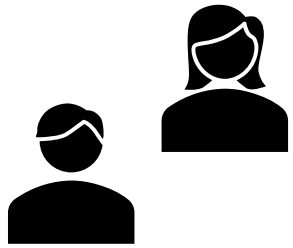
Assembler

Lernziele

- Die Modellierung eines Rechners als Zusammenspiel zwischen unterschiedlichen Ebenen verstehen
  - Unterschiedliche Rechnerebenen kennen
  - Unterschiede der RISC- und CISC-Modellierung kennen und verstehen
- Folgende Begriffe im Kontext von unterschiedlichen Rechnersichten kennen und voneinander abgrenzen können
  - Compiler, Vollcompiler
  - Interpreter

Höhere Sicht eines Rechners

Sicht eines/einer Programmierer*in



Höhere Sicht eines Rechners

Sicht eines/einer Programmierer*in

Rechner ist ein schwarzer Kasten mit der Eigenschaft:

Höhere Sicht eines Rechners

Sicht eines/einer Programmierer*in

Rechner ist ein schwarzer Kasten mit der Eigenschaft:

Bei Eingabe eines korrekten 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 RISC-Rechners

RISC steht für **R**educed Instruction **S**et **C**omputer

Modellierung eines RISC-Rechners

RISC steht für **R**educed Instruction **S**et **C**omputer

Höhere Programmiersprachen	C++, Java, Fortran
Digitale Ebene	Die eigentliche Hardware

Modellierung eines RISC-Rechners

RISC steht für **R**educed Instruction **S**et **C**omputer

Höhere Programmiersprachen	C++, Java, Fortran
Assembler	Symbolische Notation der bisher vorhandenen Befehle
Digitale Ebene	Die eigentliche Hardware

Modellierung eines RISC-Rechners

RISC steht für **R**educed Instruction **S**et **C**omputer

Höhere Programmiersprachen	C++, Java, Fortran
Assembler	Symbolische Notation der bisher vorhandenen Befehle
Betriebssystemebene	Zusätzliche Dienste (z.B. Speicherorganisation, Dateiverwaltung
Digitale Ebene	Die eigentliche Hardware

Modellierung eines RISC-Rechners

RISC steht für **R**educed Instruction **S**et **C**omputer

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

Mit Ausnahme der untersten Ebene gehört zu jeder Ebene eine Sprache

Modellierung eines RISC-Rechners

RISC steht für **R**educed Instruction **S**et **C**omputer

Heute: über 99% Marktanteil

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

Mit Ausnahme der untersten Ebene gehört zu jeder Ebene eine Sprache

Modellierung eines CISC-Rechners

CISC steht für **C**omplex **I**nstruction **S**et **C**omputer

Modellierung eines CISC-Rechners

CISC steht für **C**omplex Instruction **S**et **C**omputer

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

Modellierung eines CISC-Rechners

CISC steht für **C**omplex Instruction **S**et **C**omputer

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

Modellierung eines CISC-Rechners

CISC steht für **C**omplex Instruction **S**et **C**omputer

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 CISC-Rechners

CISC steht für **C**omplex Instruction **S**et **C**omputer

Bekanntester Vertreter: x86

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

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**.

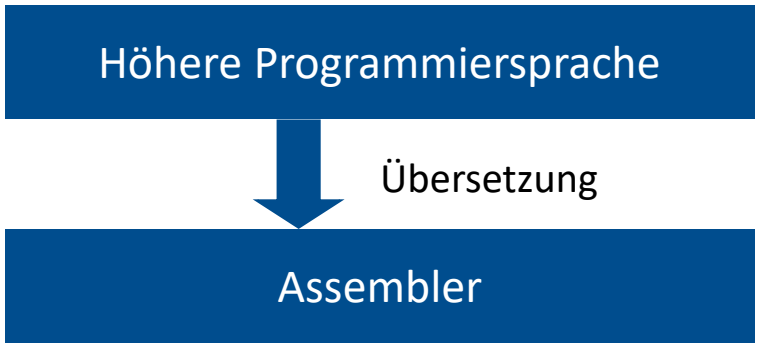
- Die virtuelle Maschine M_i „gaukelt dem Benutzer vor“, seine in L_i geschriebenen Programme würden direkt auf Hardware ausgeführt.

Hierarchie virtueller Rechner: Illustration

Höhere Programmiersprache

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

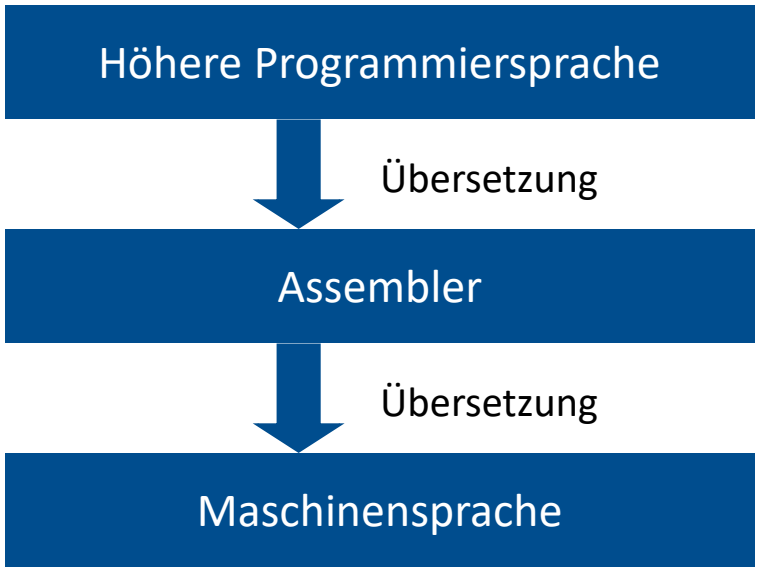
Hierarchie virtueller Rechner: Illustration




```
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
```

Hierarchie virtueller Rechner: Illustration

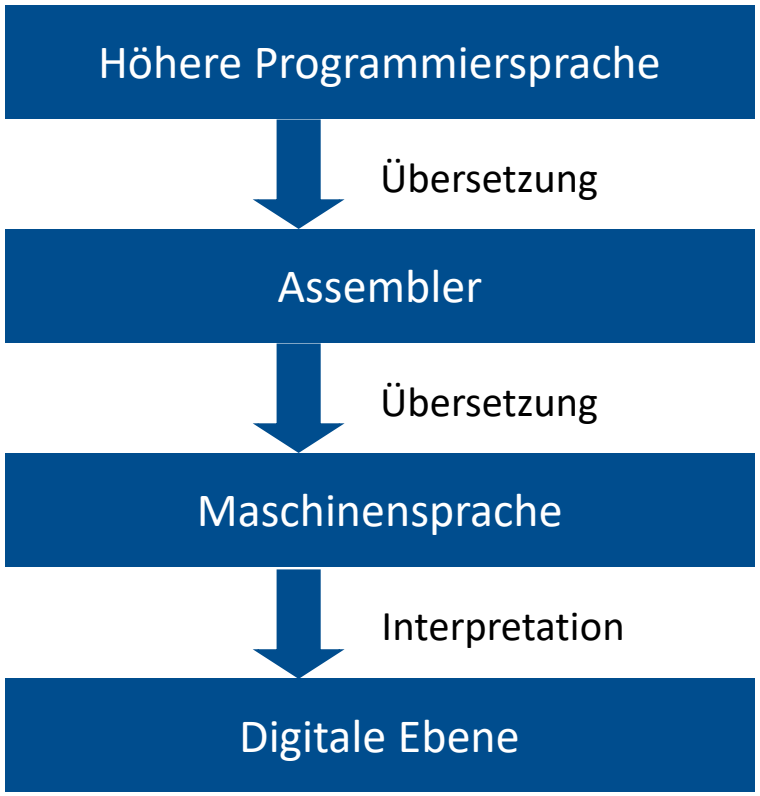



```
a = b + 3;
```

```
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
```

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 (1)

Compiler

- 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

Interpreter

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 (2)

Einfachheit:	Durchführbarkeit	Effizienz	Situationspassung
• Interpreter sind einfacher zu implementieren als Compiler • geeignet für Prototyp-Implementierungen von neuen Sprachen	• Kompilation ist nicht immer möglich • Ausführung von Programmen in Maschinensprache nur durch Interpretation möglich	• Übersetzte Programme sind schneller als interpretierte Programme • Code-Optimierung bei Übersetzung möglich	• (vollständige) Kompilation nicht immer erwünscht • Systemunabhängigkeit (Portabilität)

Performanzmaße eines Rechners

Höhere Programmiersprache	Laufzeit eines Programms Reaktionszeit der Anwendungen
Assembler	
Betriebssystemebene	
Maschinensprache	Millions of instructions per second (MIPS) Millions of floating-point operations per second (MFLOPS)
Mikroprogrammebene	
Digitale Ebene	Taktrate (Länge des Datenpfades, ...)