Rechnerarchitektur

Dr. Andreas Müller
TU Chemnitz
Fakultät für Informatik
Fakultätsrechen- und Informationszentrum
anmu@informatik.tu-chemnitz.de

Rechnerarchitektur

Dr. Andreas Müller
TU Chemnitz
Fakultät für Informatik
Fakultätsrechen- und Informationszentrum
anmu@informatik.tu-chemnitz.de

http://www.tu-chemnitz.de/informatik/friz/Grundl-Inf/Rechnerarchitektur/

Rechnerarchitektur (2)

- Einführung
- Geschichte
- v.-Neumann-Architekturen
- Befehle
  - Operationen und Operanden der Rechnerhardware
  - Logische Operationen
  - Entscheidungsbefehle
  - Compiler (Beispiel: C-Programm)

Rechnerarchitektur (3)

- Rechnerarithmetik
  - Addition, Subtraktion, Multiplikation, Division
  - Gleitkommaarithmetik
- Der Prozessor: Datenpfad und Steuerwerk
  - Entwurf von Logikschaltungen
  - Implementierungsmethoden
  - Steuerwerksentwurf
- Speicherorganisation
  - Speicherhierarchie
  - Cache
  - Speicherhierarchien bei P4/Opteron

Literatur

- Patterson, Hennesy: Rechnerorganistion und -entwurf, Spektrum, 2005
- Materialien zu dieser LV, insbesondere LC-1

  http://www.tu-chemnitz.de/informatik/friz/Grundl-Inf/Rechnerarchitektur/

- **<u>Quellen zur Vorbereitung der LV:</u>**

  - Script von Prof. Hardt, TU Chemnitz
  - Script von Dr. Naumann, TU Chemnitz

Was ist Rechnerarchitektur?

Amdahl, Brooks und Blaauw definieren 1967:

„Rechnerarchitektur definiert sich aus den Attributen und dem Verhalten eines Computers, wie dieser von einem Maschinensprachenprogrammierer gesehen wird. Diese Definition umfasst den Befehlssatz, die Befehlsformate, die OP-Codes, die Adressierungsarten und alle Register und Speicher, die direkt durch einen Maschinensprachenprogrammierer verändert werden können. Die Implementation ist durch den aktuellen Hardwareaufbau, das logische Design und die Organisation der Datenpfade einer bestimmten Ausführung der Architektur definiert.“

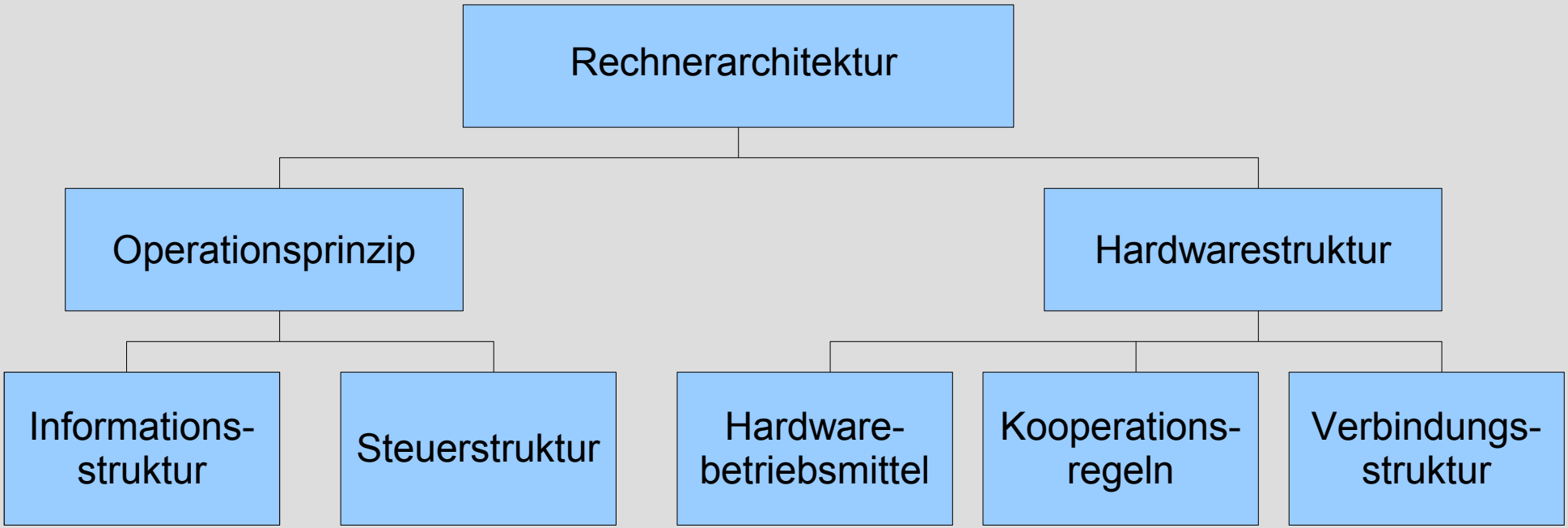
Was ist Rechnerarchitektur? (2)

Stahlknecht und Hasenkamp verwenden folgende Definition:
“Unter dem Begriff Rechnerarchitektur versteht man:

•die interne Struktur des Rechners, d. h. seinen Aufbau aus verschiedenen Komponenten, und
•die Organisation der Arbeitsabläufe im Rechner.“

Was ist Rechnerarchitektur? (3)

- Taxonomie nach Giloi



Was ist Rechnerarchitektur? (4)

- Ebenen eines Rechnersystems
  - Anwendungsebene (Anwendungssoftware)
  - Assemblerebene (Beschreibung von Algorithmen, Link und Bind)
  - Betriebssystem (Speichermanagement, Prozesskommunikation)
  - Instruction Set Architecture
  - Microarchitektur (RISC, CISC)
  - Logische Ebene (Register, Schieberegister, Latches)
  - Transistorebene (Transistoren, MOS)

Was ist Rechnerarchitektur? (3)

- Klassifikation
  - Nach Rechenprinzip
    - Von Neumann (Steuerfluss)
    - Harvard-Architektur
    - Datenfluss (Zündregel, Petrinetze)
    - Reduktion (Funktionsaufruf)
    - Objektorientiert (Methodenaufruf)
  - Nach Architekturgrundkonzept
    - Vektorrechner (Pipeline)
    - Array-Computer (Data-Array)
    - Assoziativ-Rechner (Assoziativspeicher)

Was ist Rechnerarchitektur?(5)

- Rechnerarithmetik
  - Addition, Subtraktion, Multiplikation, Division
  - Gleitkommaarithmetik
- Der Prozessor: Datenpfad und Steuerwerk
  - Entwurf von Logikschaltungen
  - Implementierungsmethoden
  - Steuerwerksentwurf
- Speicherorganisation
  - Speicherhierarchie
  - Cache
  - Speicherhierarchien bei P4/Opteron

Was ist Rechnerarchitektur?(5)

- Rechnerarithmetik — LV *Digitaltechnik* (Prof. Stopje)
  - Addition, Subtraktion, Multiplikation, Division
  - Gleitkommaarithmetik
- Der Prozessor: Datenpfad und Steuerwerk
  - Entwurf von Logikschaltungen
  - Implementierungsmethoden
  - Steuerwerksentwurf
- Speicherorganisation — 2. Teil
  - Speicherhierarchie
  - Cache
  - Speicherhierarchien bei P4/Opteron

Einführung

- Grundbegriffe eines Rechnersystems
  - Rechenwerke
  - Steuerwerke
  - Interruptverarbeitung
- Befehlsstruktur und Befehlssatz
- Einführung in aktuelle Rechnerarchitekturen
  - Peripherie-Bausteine
  - Interne Busstrukturen eines Mikrorechnersystems
- Hilfsmittel zum Arbeiten mit Mikrorechnern

v.-Neumann-Prinzipien

1. der Rechner besteht aus 5 Einheiten: Steuerwerk, Rechenwerk, Speicher, Ein- und Ausgabewerk
2. Struktur des Rechners ist unabhänig vom zu lösenden Problem
3. Programm und Daten werden im gleichen Speicher abgelegt
4. Speicher sind in gleichgroße Zellen unterteilt, die fortlaufend nummeriert sind

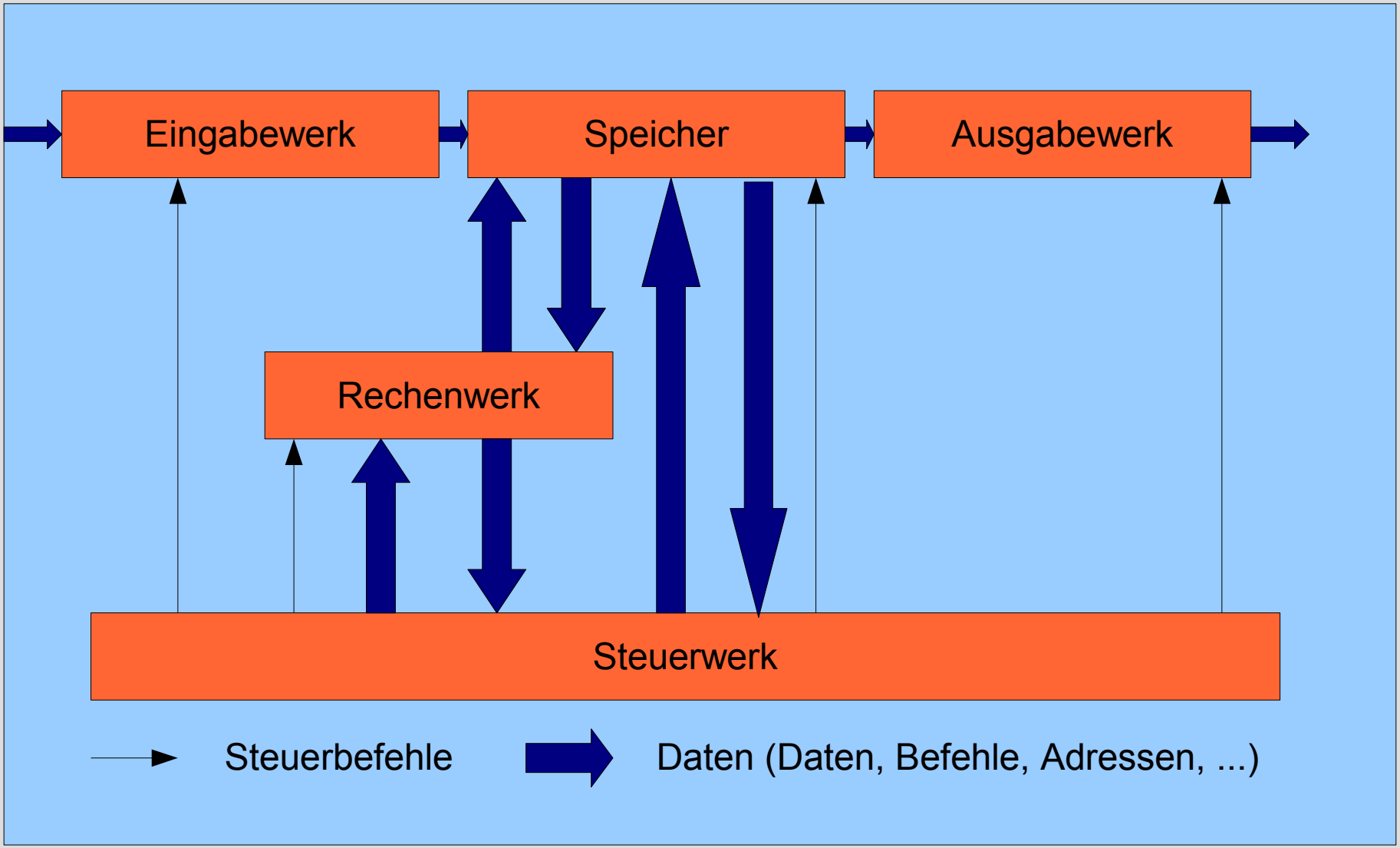
v.-Neumann-Prinzipien (2)

5. aufeinander folgende Befehle eines Programms werden in aufeinander folgenden Speicherzellen abgelegt
6. durch Sprungbefehle kann von der sequentiellen Bearbeitung abgewichen werden

v.-Neumann-Prinzipien (3)

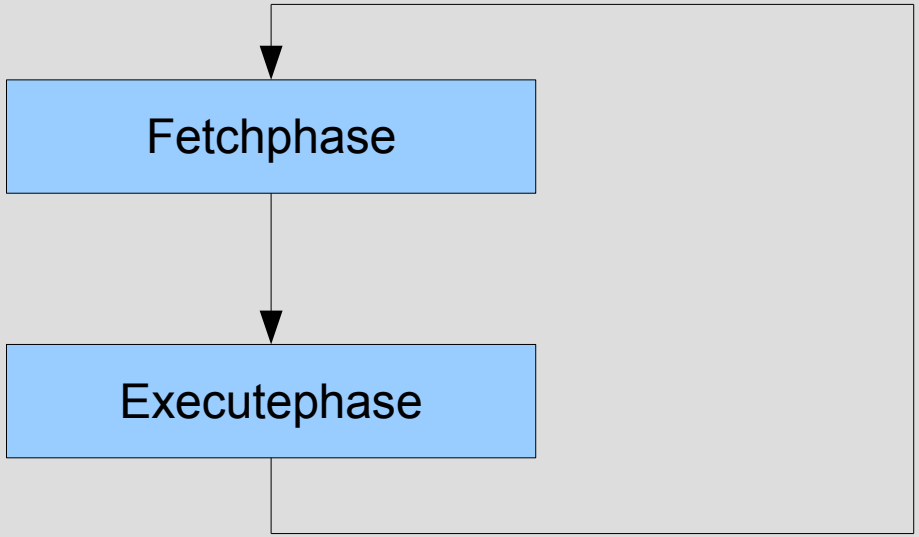
7. Es existiert ein Befehlssatz
   - arithmetische Befehle
   - logische Befehle
   - Transportbefehle
   - Verzweigungsbefehle
   - sonstige Befehle
8. alle Daten werden binär kodiert

v.-Neumann-Architektur

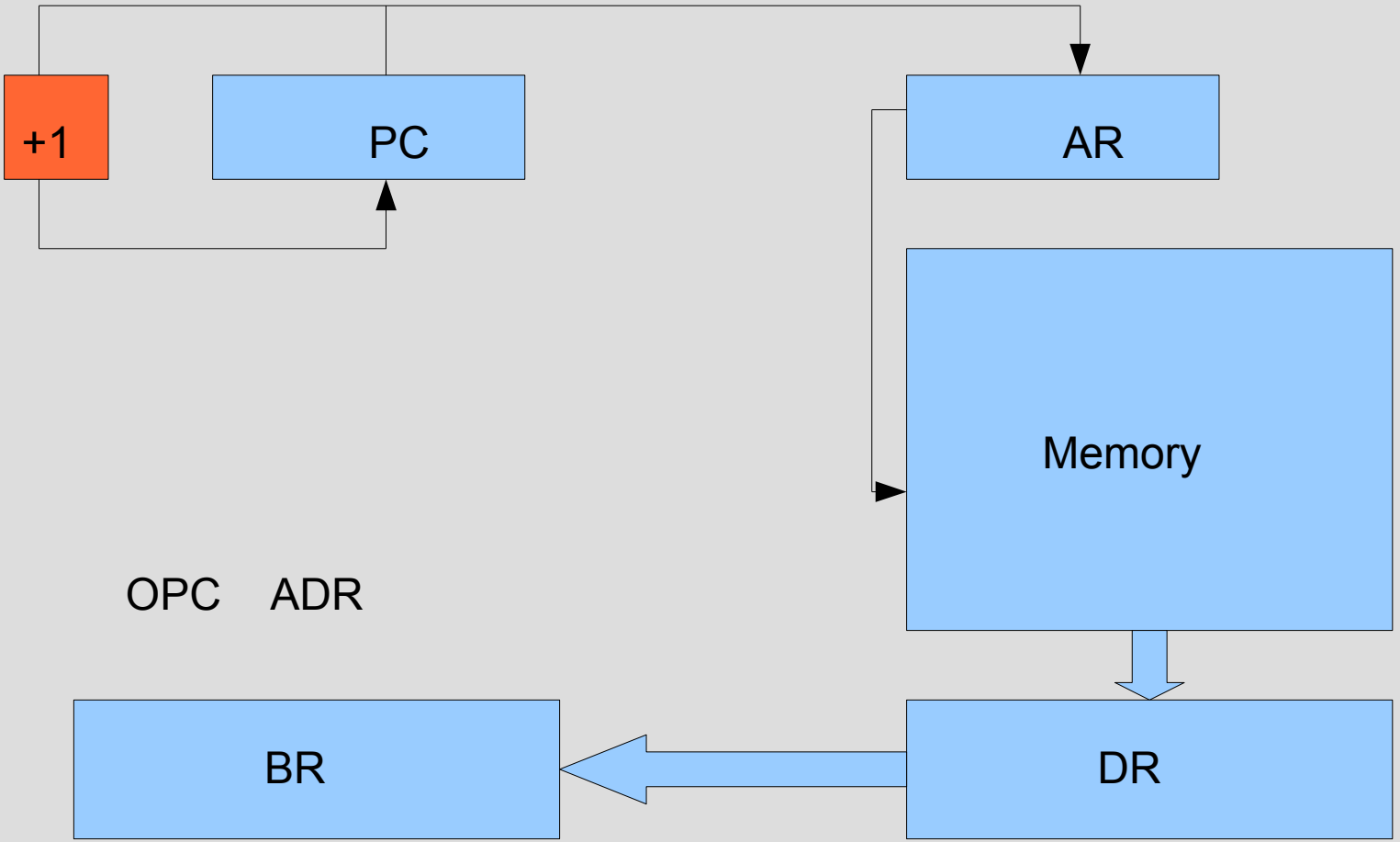


Arbeitsweise des Steuerwerkes

- Programm besteht aus Befehlen
- Abarbeitung des Programmes erfolgt durch sequentielle Abarbeitung der Befehle
- Ein Befehl wird in zwei Phasen ausgeführt:



Arbeitsweise des Steuerwerkes (2)



Arbeitsweise des Steuerwerkes (3)

Beispiel: LC1

Auswertung des Operationscodes
„Befehlsdekodierung“ (PASCAL-ähnliche Notation)

Ausführung des erkannten Befehls

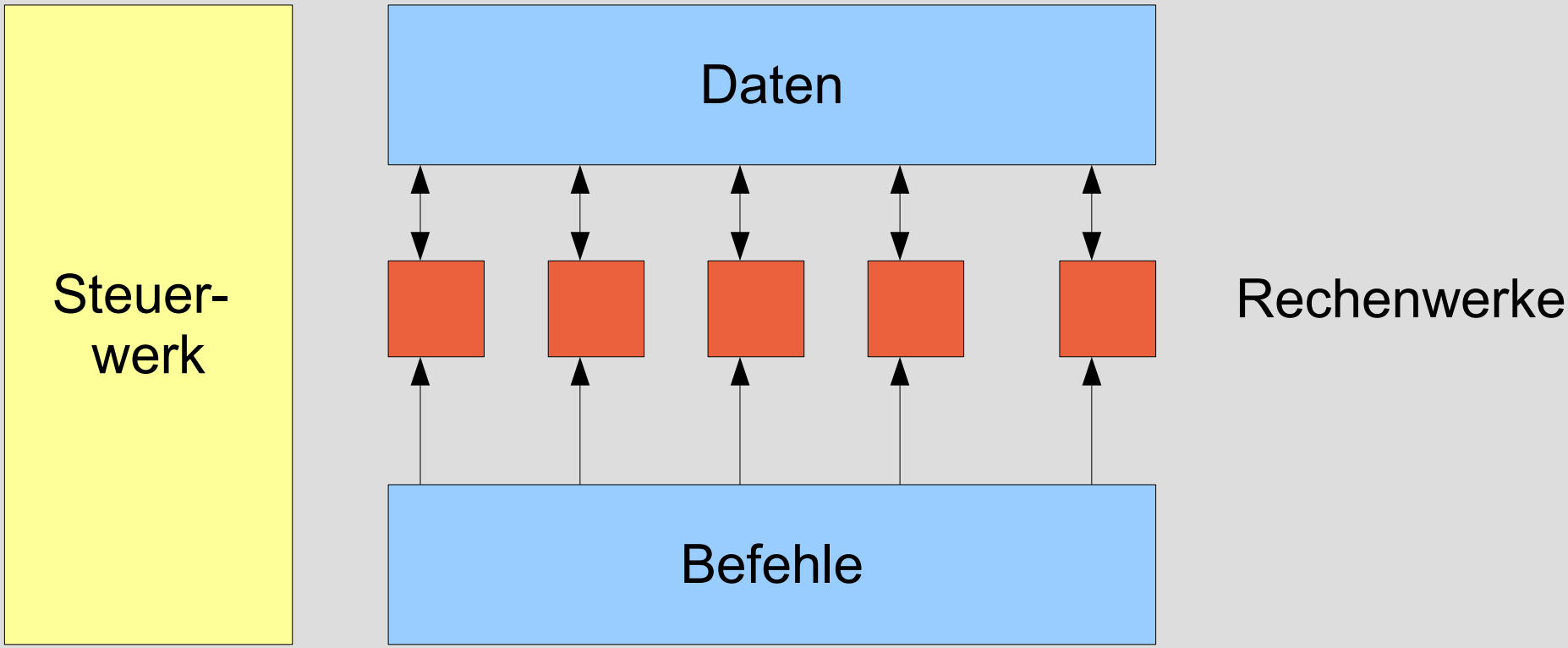
```
case OPC of

  '0000': A:= Memory(ADR);          {load Register A}
          S:= A(9);
  '0001': B:=Memory(ADR);           {load Register B}
  '0010':Memory(ADR):=A;            {move Register A}
  .....
  '1000': PC:=ADR                   {JMP}
  .....
end;
```

Harvard-Architektur

- Befehlsspeicher ist physisch vom Datenspeicher getrennt
  - Vorteil
    - gleichzeitiges Laden und Speichern von Befehlen **und** Daten
    - bei Softwarefehlern kann kein Programmkode überschrieben werden
    - Datenwortbreite kann sich von Befehlswortbreite unterscheiden
  - Nachteil
    - komplexer als v.-Neumann-Architektur
    - Datenspeicher kann nicht als Programmspeicher verwendet werden und umgekehrt.

Harvard-Architektur (2)

Schematische Darstellung



Harvard-Architektur

- Beispiele:
  - Produkte der Firma Microchip Technology
    - PICmicro (Umsatz 2007: 1,04 Mrd. US$)
  - Microcontroller der AVR-Reihe (Umsatz 1,676 Mrd. US$ 2005)
  - Analog Devices (DSP, Umsatz 2,6 Mrd. US$)
    - SHARC-Technologie
  - moderne Mikroprozessoren verwenden Mischform aus v.-Neumann und Harvard-Architektur (getrennte Daten- und Befehlsbereiche, getrennte Caches und MMU's, getrennte Busse, extern gemeinsamer Speicher)

n-Adress-Maschinen

Ein Befehl mit dem Aufbau

heißt 1 Adress-Maschine
Es kann aber auch sinnvoll sein,
5-Adress, 4-Adress-, 3-Adress-
2-Adress-Maschinen zu bauen

Wieviele Adressen sind sinnvoll?

- v.-Neumann Prinzip 7: *jeder Rechner besitzt Befehlssatz*
- Adresszahl ----> Mächtigkeit eines Befehls
- 1. Herangehensweise:
  Befehlssatz höhere Programmiersprache entspricht Maschinenbefehl (Beispiel: Symbolics Inc., Hersteller von LISP-Maschinen)
  Die semantische Lücke ist gleich 0

Wieviele Adressen sind sinnvoll? (2)

- 2. Herangehensweise:
  - Analyse der Algorithmen
  - Suchen der atomaren Komponenten
  - daraus Ableiten eines Befehlssatzes mit geringer Mächtigkeit
  - Sematische Lücke ist sehr groß: Compiler
  - dieser Weg soll als der gegenwärtige Königsweg weiterverfolgt werden

Wieviele Adressen sind sinnvoll (3)

- Atomare Operationen (C-Darstellung)

```
c = a OPC b;
if (bed)
    goto f1;
 else
    goto f2;
```

- daraus folgt: Rechner muss atomare Operationen
  - zweistellige **Verknüpfungen**
  - **Verzweigungen**

  umfassen

Wieviele Adressen sind sinnvoll (4)

- 5-Adress-Maschine
  OPC ADR1 ADR2 ADR3 ADR4 ADR5
- Funktion
  <ADR3> := <ADR1> op(OPC) <ADR2>;
  if cond(OPC) = true then
    goto ADR4
  else
    goto ADR5;
- m op-Teile aus OPC und n cond-Teile aus OPC
  ---> m * n verschiedene Befehle
- v-Neumann-Prinzip 5 braucht nicht eingehalten werden
- kein PC erforderlich

Wieviele Adressen sind sinnvoll (5)

- 4-Adress-Maschine
  OPC ADR1 ADR2 ADR3 ADR4
- Funktion
  PC := PC+1;
  <ADR3> := <ADR1> op(OPC) <ADR2>;
  if cond(OPC) = true then
  goto ADR4
- v-Neumann-Prinzip 5 muss eingehalten werden
- PC erforderlich

Wieviele Adressen sind sinnvoll (6)

- 3-Adress-Maschine
  OPC1 ADR1 ADR2 ADR3
  OPC2 ADR4
- Funktion
  PC := PC+1;
  <ADR3> := <ADR1> op(OPC) <ADR2>;
  Flags := gewisse Eigenschaften des Ergebnisses
  PC := PC + 1;
  if cond(OPC, Flags) = true then
  goto ADR4
- m op-Teile aus OPC und n cond-Teile aus OPC
  ---> m + n verschiedene Befehle
- 2 Klassen von Befehlen notwendig (verknüpfende und verzweigende)
- PC erforderlich

Wieviele Adressen sind sinnvoll (7)

- 2-Adress-Maschine
  OPC1 ADR1 ADR2
  OPC2 ADR3
- Funktion
  PC := PC+1;
  <ADR1> := <ADR1> op(OPC) <ADR2>;
  Flags := gewisse Eigenschaften des Ergebnisses
  PC := PC + 1;
  if cond(OPC, Flags) = true then
  goto ADR3
- der Operand ADR1 wird durch das Resultat überschrieben

Wieviele Adressen sind sinnvoll (7)

- 1-Adress-Maschine
  OPC1 ADR1
  OPC2 ADR2
- Funktion
  PC := PC+1;
  <ACC> := <ACC> op(OPC1) <ADR1>;
  Flags := gewisse Eigenschaften des Ergebnisses
  PC := PC + 1;
  if cond(OPC2, Flags) = true then
  goto ADR1
- Akkumulator erforderlich, dieser enthält zunächst einen Operanden und wird dann mit dem Ergebnis überschrieben

Wieviele Adressen sind sinnvoll (8)

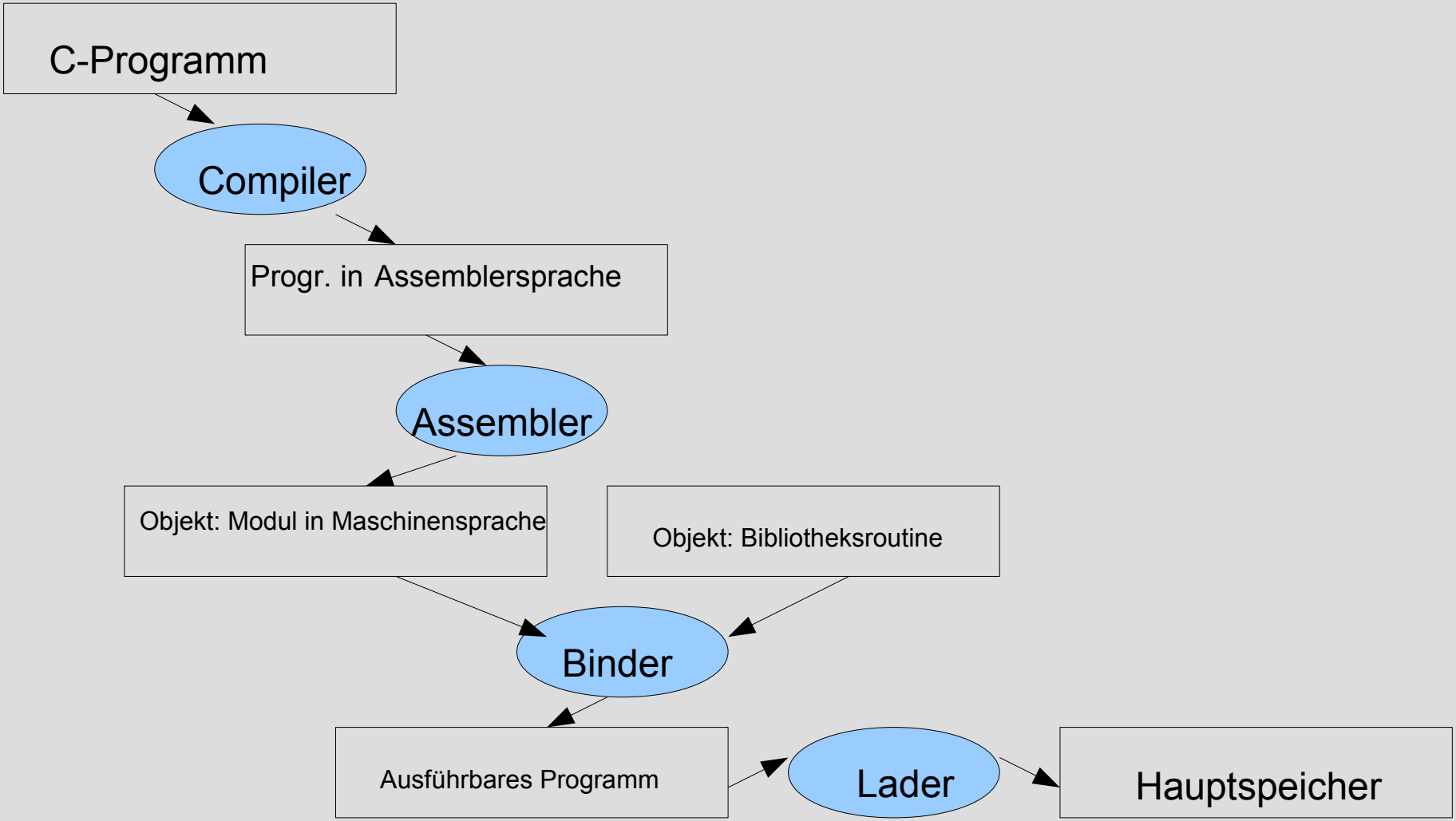
- 0-Adress-Maschine

```
PUSH a      {Übertragung Adresse 1 in Arithmetikkeller}
PUSH b      {Übertragung Adresse 2 in Arithmetikkeller}
ADD
POP  c      {Übertragung Ergebnis an Adresse c}
```

- sogenannte polnische oder reverse polnische Notation
- Taschenrechner von HP, Großrechner von Borroughs
- Intel Gleitkommaprozessor
- abstrakte Java-Maschine (Java-Bytekodeinterpreter)
- Taschenrechner Elektronika BE 34

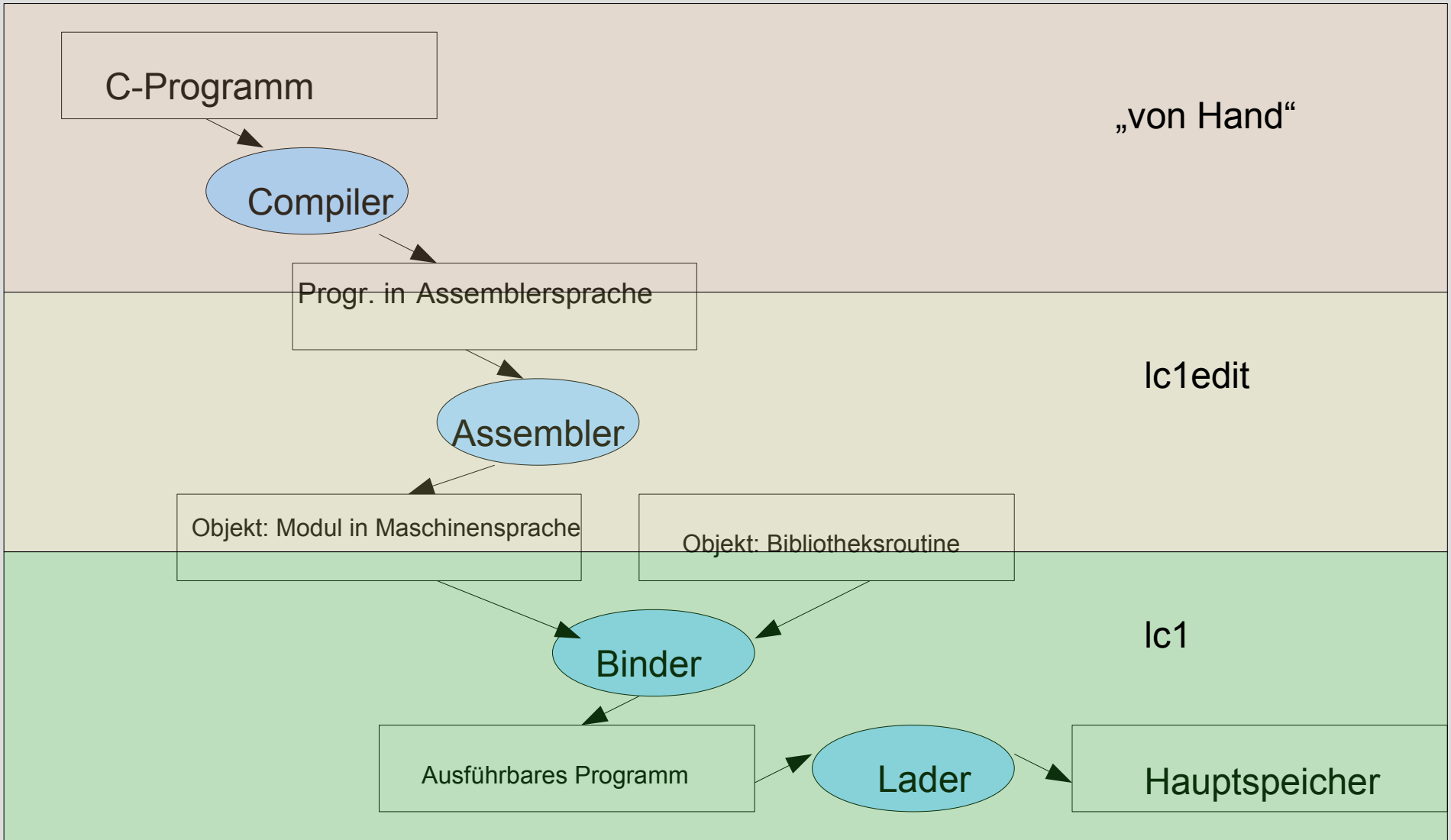
Compiler

Übersetzungshierarchie für C



Compiler

Übersetzungshierarchie für C



LC 1

- 1- Adress-Maschine
- von-Neumann-Architektur
- studentische Belegarbeit (1990 Dipl.Inf. Rico Müller und Dipl.Inf. Sören Schulze)
- Download über

  www.tu-chemnitz.de/informatik/friz/Grundl-Inf/Rechnerarchitektur/
- Beispielarchitektur der LV

LC 1

Hinweis:

Abarbeitung unter Windows 7 / Vista

- DOSBOX (www.dosbox.com)
- XP-Modus (http://www.microsoft.com/windows/virtual-pc/)