Vorlesung 3

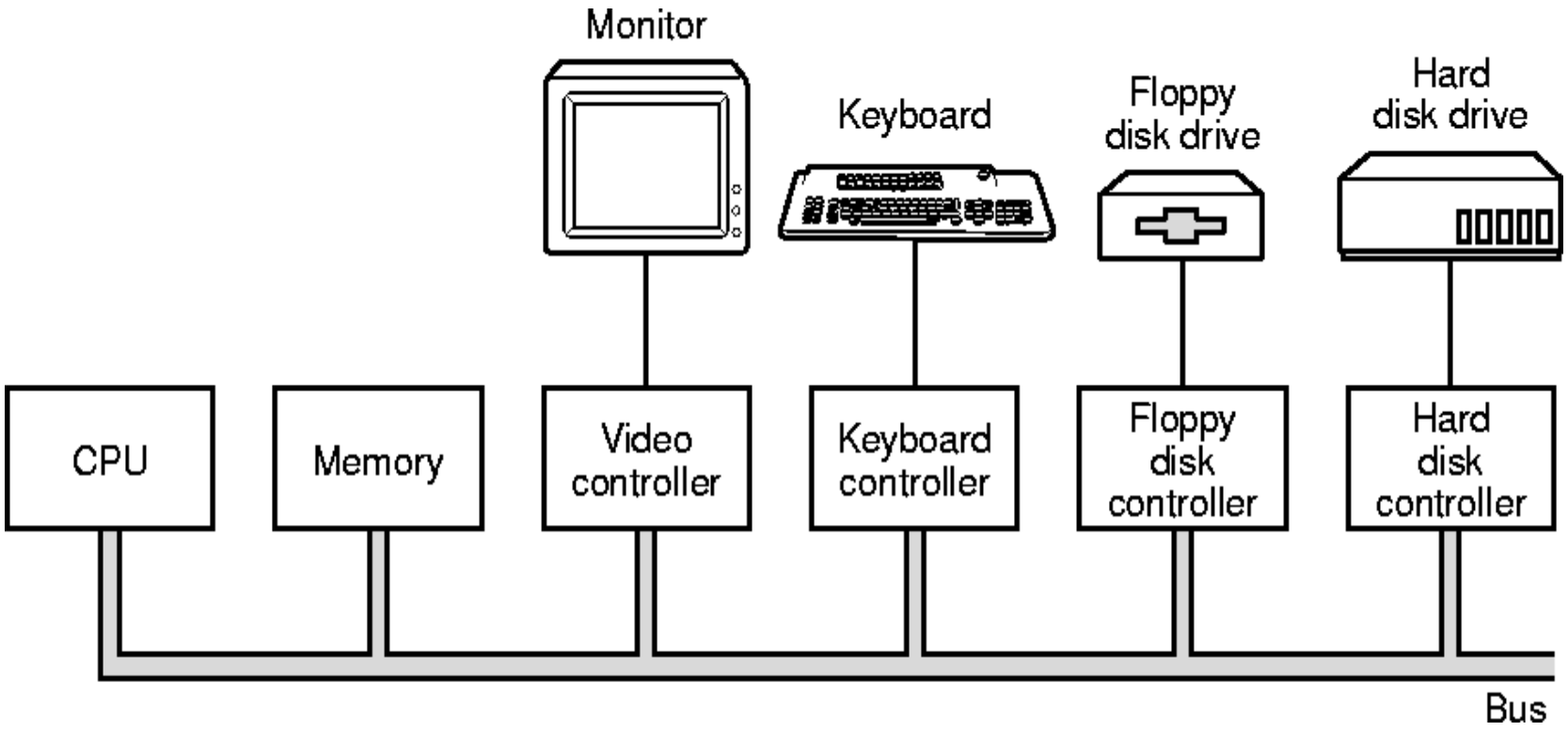
Verschiedenes

Peter B. Ladkin
ladkin@rvs.uni-bielefeld.de

Inhalt

- Busarchitektur
- Virtuelle Maschine

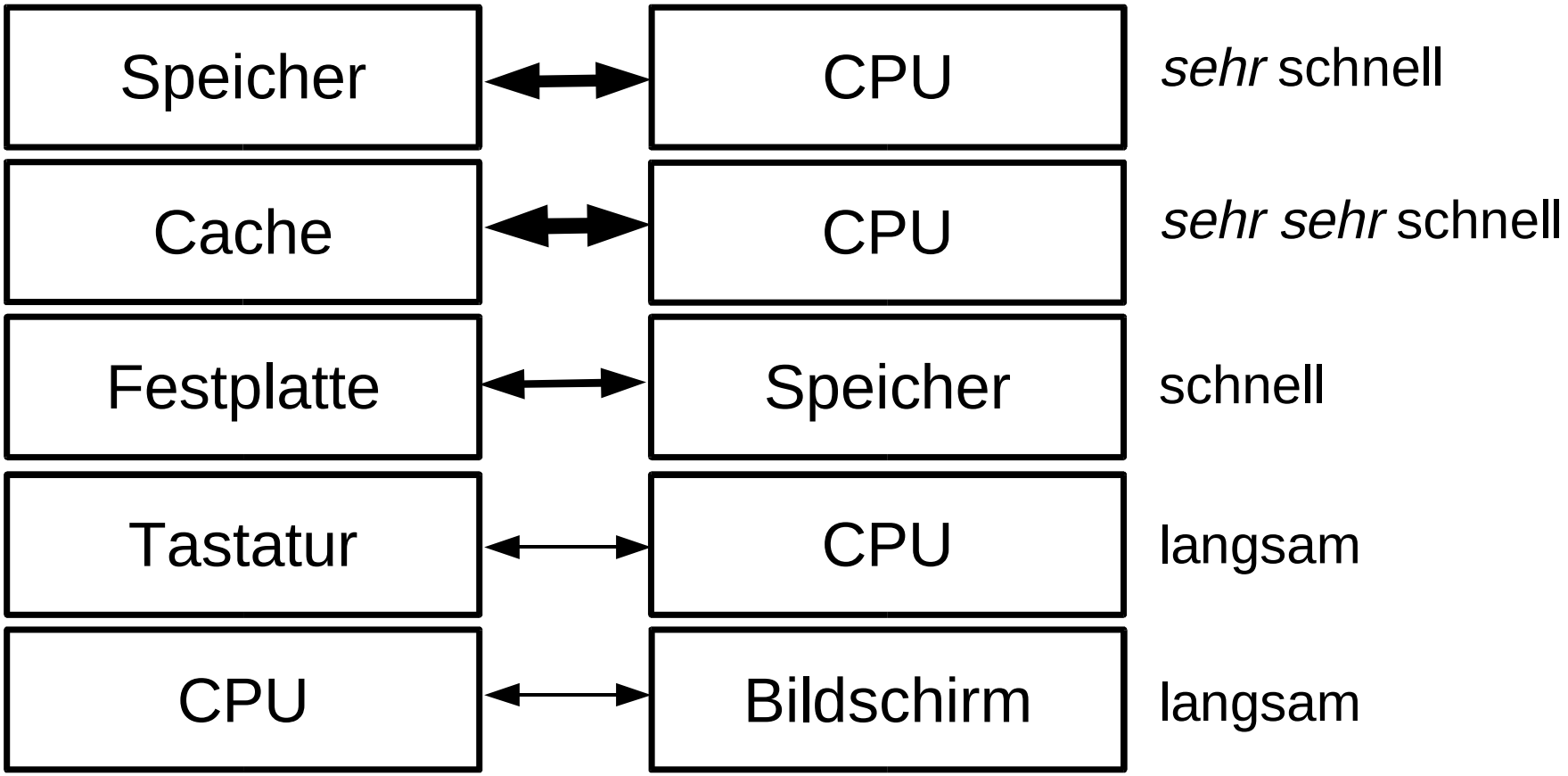
Busarchitektur - das Ideal



Busarchitektur

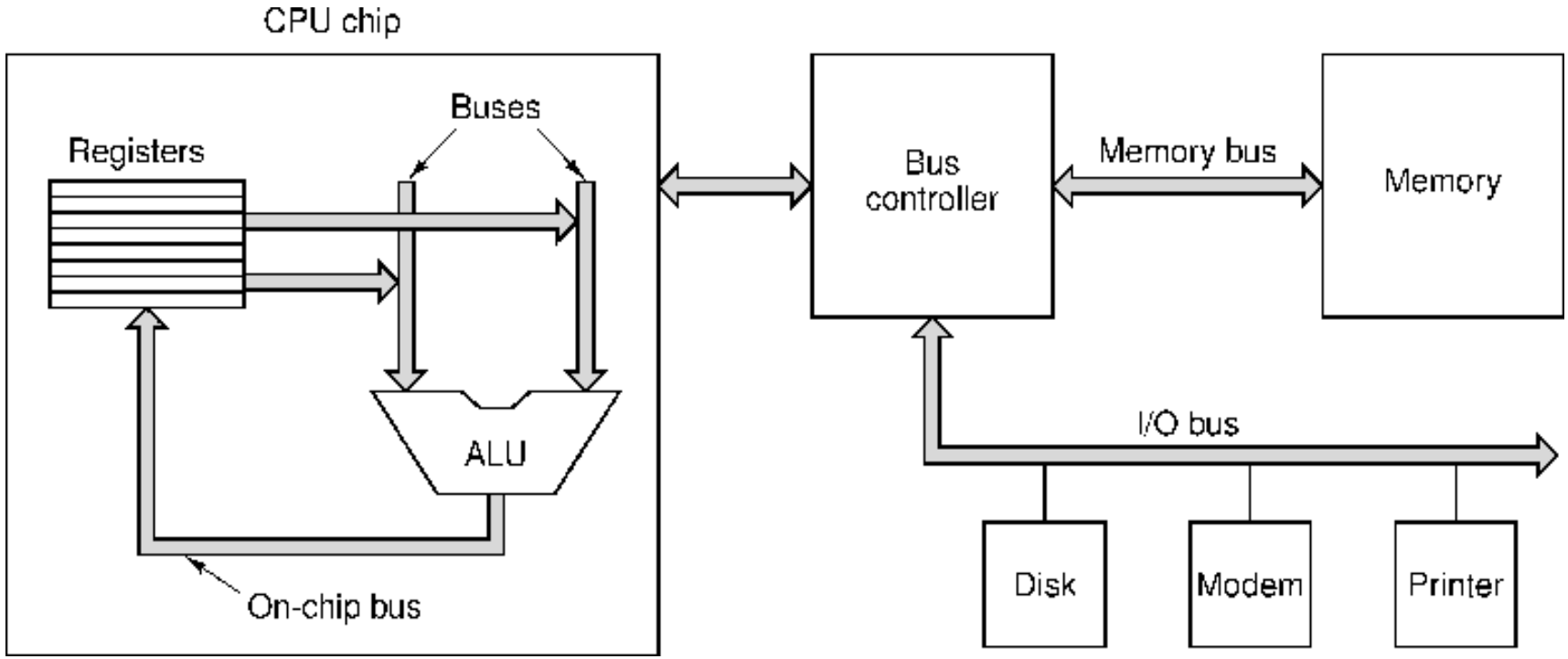
- Alles sitzt auf dem gleichen Kabel
- Das Gerät wird nur durch die Adresse unterschieden

Busarchitektur - Der Plan



→ **unterschiedliche Geschwindigkeiten**

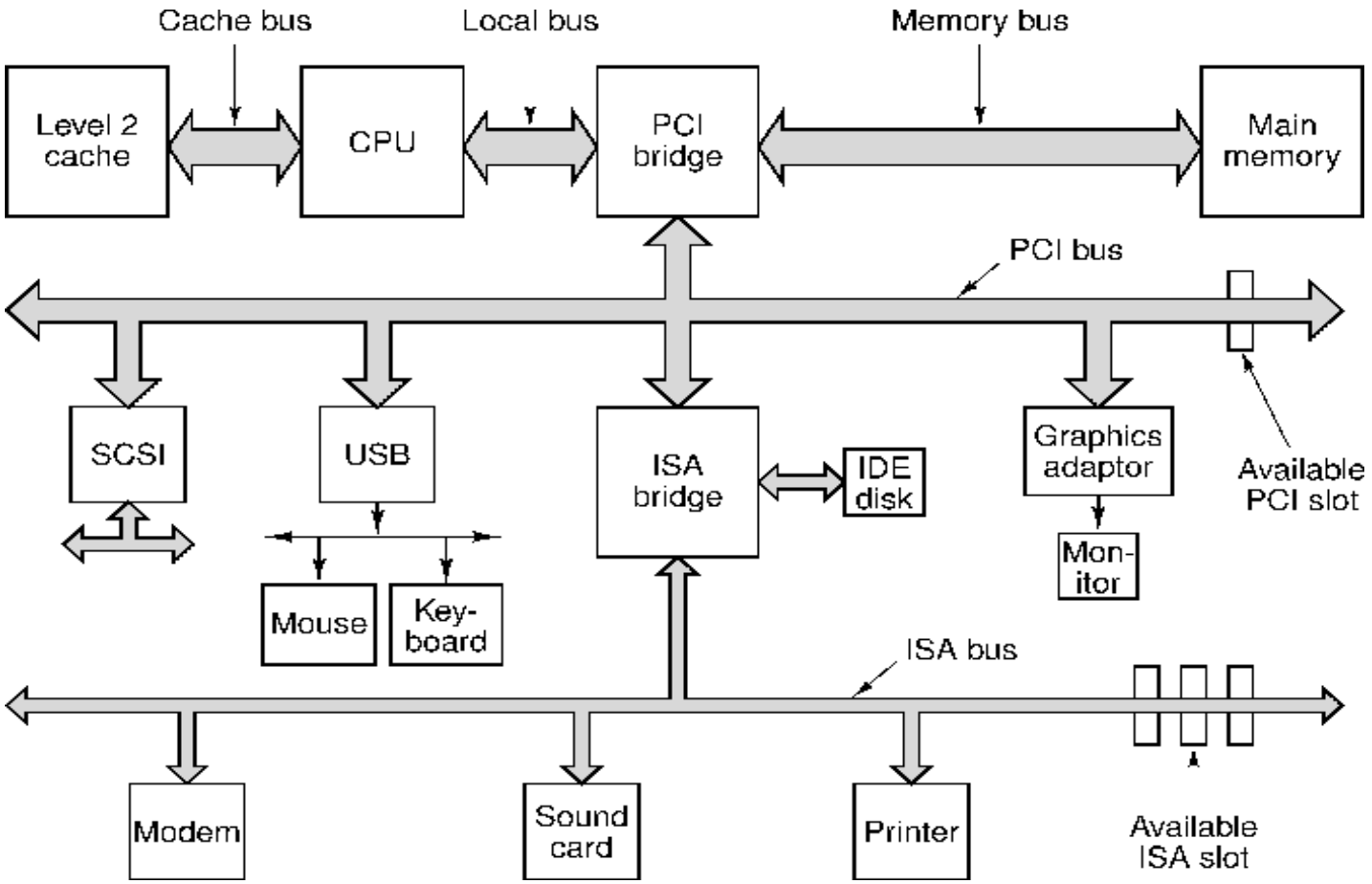
Busarchitektur - Der Plan



Busarchitektur - Die Realität

- Unterschiedliche Geräte von unterschiedlichen Herstellern sind billiger/teurer, schneller/langsamer, früher/später, ...
- Unterschiedliche Geräte funktionieren nur mit einer Architektur
- Computer sind komponentenweise aufgebaut

Busarchitektur - Die Realität



Busarchitektur - Die Realität

- Memory-Bus
- Cache-Bus
- Local-Bus
- PCI-BUs
- ISA-Bus
- .....

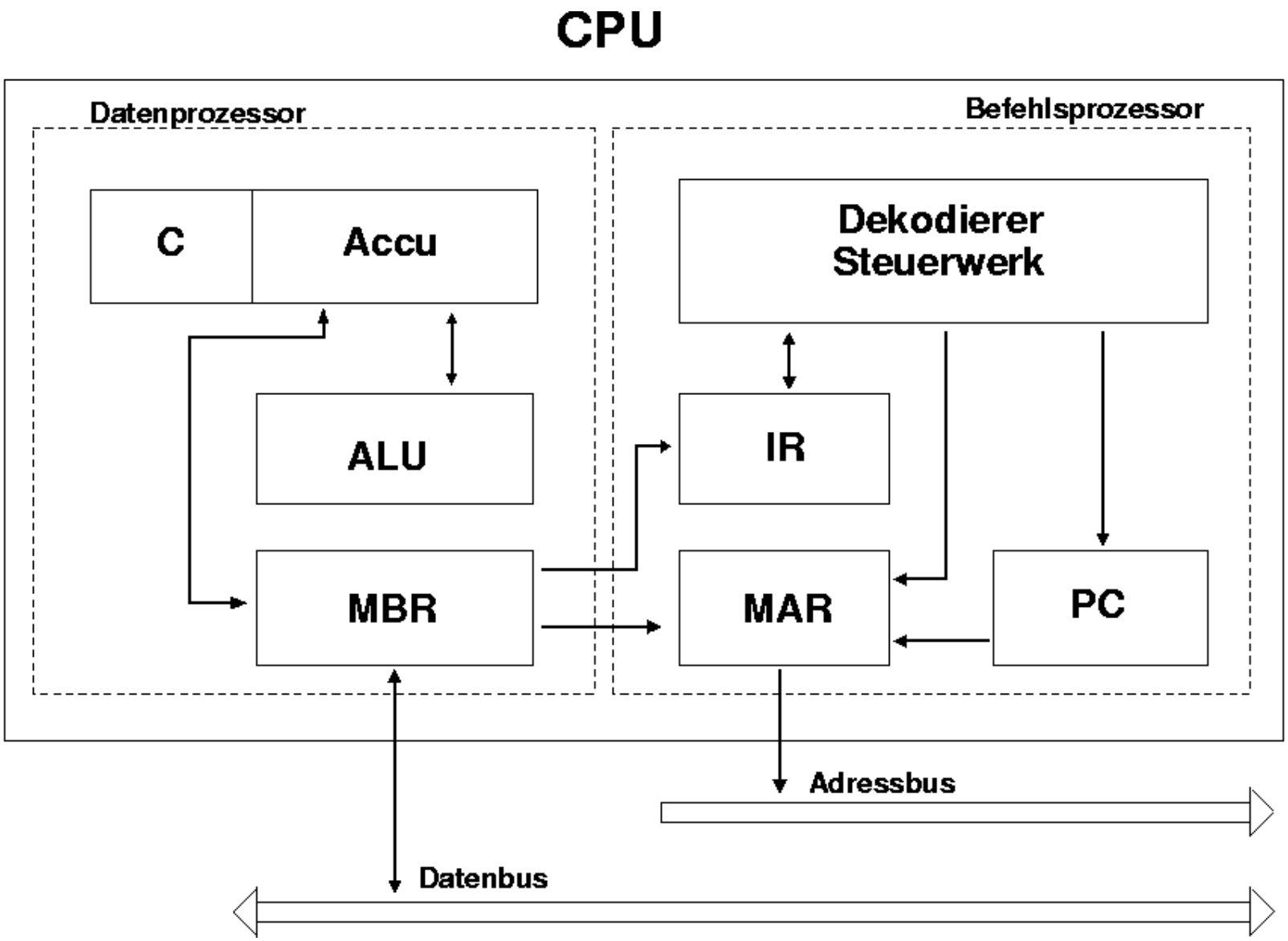
Busarchitektur - Die Realität

- Anzahl der Komponenten ist größer als Anzahl der Kommunikationswege
- Es gibt eine Menge von beiden
- Idee von Busarchitektur einfach
- Die Realität hat mit Geschichte, Marketing und Business zu tun
- Alles ist komplizierter als es sein müsste

Virtuelle Maschine

- Ein-Baustein-Idee für die ganze Informatik
- Überall gefunden

VM – Beispiel


CPU
Datenprozessor
Befehlsprozessor
C
Accu
ALU
MBR
Dekodierer
Steuerwerk
IR
MAR
PC
Adressbus
Datenbus

VM – Beispiel

- JUMP <Sp-Adr>
  - Dekodiere JUMP / <Sp-Adr>
  - <Sp-Adr> -> MAR; PC <- <Sp-Adr> + 1
  - Datum -> MBR
  - MBR -> IR
  - IR -> DSW

VM – Beispiel

- ADD <Sp-Adr>
  - Dekodiere ADD / <Sp-Adr>
  - ADD -> ALU; PC <- PC + 1; <Sp-Adr> -> MAR
  - Daten -> MBR; PC -> MAR
  - MBR -> ALU; ACC -> ALU; Daten -> MBR
  - ALU -> ACC; MBR -> IR
  - IR -> DSW

VM – Beispiel

- JUMP, ADD, SUBTRACT, MULTIPLY, DIVIDE, LOAD, STORE
- "Higher Level"

VM – Beispiel

- "Lower Level"
  - PC: +1, Load
  - MAR: Load (PC, DSW), Put
  - MBR: Load, Put (IR, ALU)
  - IR: Load, Put
  - DSW: Dekodiere, Put(PC, MAR, ALU)
  - ALU, Load(ACC, MBR), Put, Store
  - .....

VM – Beispiel

- "Higher-Level" Operationen werden als "Programme" von "Lower-Level" Operationen definiert
- "Higher-Level" Datenstrukturen werden als strukturierte Kombinationen von "Lower-Level" Datenstrukturen definiert

VM – Allgemein

- "Higher-Level" DS und Ops werden als Strukturen bzw Programme von "Lower-Level" DS und Ops definiert
- Eine V-Maschine wird über ihre DS und Ops festgelegt
- Also definiert man VM1 von Rechner 0; VM2 von VM 1; VM3 von VM2; ...usw

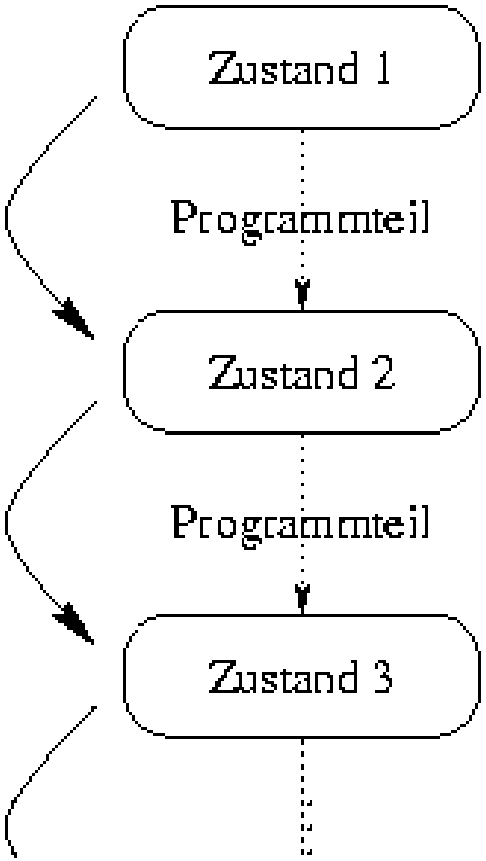
Allgemeine Ontologie

- Was gibt es für Objekte?
- Objekte haben Zustände (wechselbare Eigenschaften)
- Was gibt es für Operationen?
- Operationen bedeuten Wechsel (Change) der Zustände der Objekte

VM – Allgemein

- Ein Programm besteht aus Definitionen der Wechsel der Objekte
- Die Sammlung der Zustände (Eigenschaften) der Programm-Objekte ist der Zustand des Programms
- Jede Programm-Operation bedeutet ein Wechsel der Zustände der Objekte

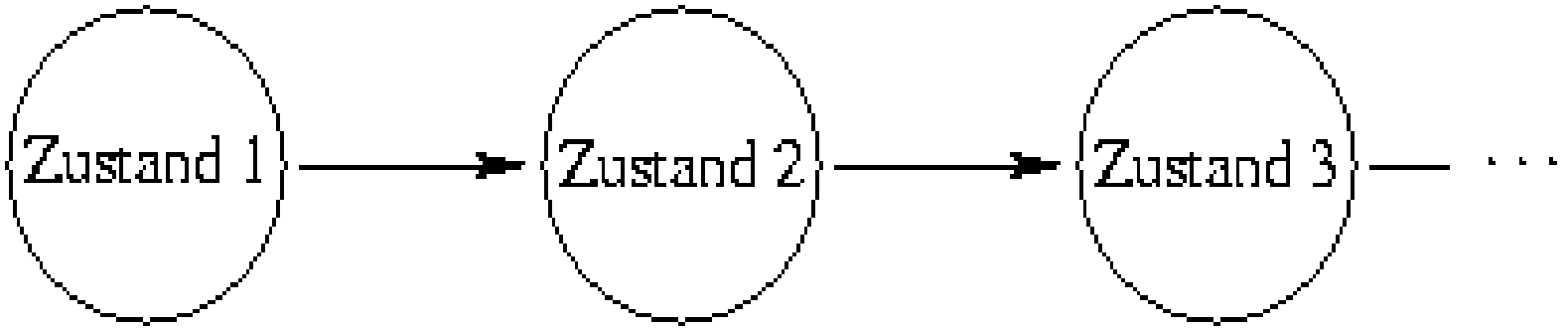
VM – Allgemein



VM – Allgemein

- Dies wird anders bezeichnet

VM – Allgemein



VM – Allgemein

- Die Operationen können komplex oder einfach sein
- Die Zustandsänderungen können komplex oder einfach sein
- Es kommt darauf an, auf welchem "Level" sie definiert sind

VM – Konkret

- "Lower-Level" Objekte
  - PC
  - MAR
  - MBR
  - ACC
  - IR

VM – Konkret

- "Lower-Level" Operationen
  - Load (MBR, MAR, PC, IR, DSW, ALU, ACC)
  - Store (MBR, ...)
  - ADD (MBR, ACC); SUBRACT (MBR, ACC),.....
  - +1 (PC)

VM – Konkret

- "Higher-Level" (Benutzerebene)
  - "Schreiben" / "Speichern" von "Files"
  - "File" = beliebige "Sequenz" von "Buchstaben"
  - "Buchstabe" = "Sequenz" von 8 "Bits"
  - "Lesen"/"Schicken" von "E-Mail"
  - "E-Mail" = "Header" + "File"
  - "Schicken" von "Inhalt" + "Adresse"
  - "Laden" von "WWW-Seite"

VM – Konkret

- "Compilieren" eines "Programms"
- "Ausführen" eines "Programms"
- Und so weiter ...

VM – Aufbau

- "File" = "Sequenz" von "Buchstaben"
- "Buchstabe" = "Byte"
- Also: bestimmte Anzahl von Bytes

VM – Aufbau

- File-Änderung
  - Speicher von <Adr> bis <Adr>+Anzahl kopieren
  - "Buffer"
  - "Cursor" in <Adr>.....<Adr>+Anzahl
  - Änderung = (in Kopie)
    - Byte von "Tastatur" -> ACC
    - Von <adr>+Anz bis "Cursor": <adr> <- <adr> + 1
    - "Cursor" <- ACC
    - "Buffer" in <Adr> bis <Adr>-Anzahl+1 kopieren...wenn.....

Betriebssystem

- Definiert "übliche" High-Level Datenstrukturen
- Definiert "übliche" Operationen
- Macht die ganze Buchhaltung
- Auf Basis der HW-Ebene
  - Assembly-Sprache
  - Speicher, Festplatte, Drucker, Tastatur, Bildschirm...

Betriebssystem

- Was ist "üblich"?
  - Speicher ist eine Sequenz von Buchstaben
  - Files sind Sequenzen von Buchstaben/Zahlen
  - Operationen sind File-Operationen und Variable-Operationen
    - d.h. Operationen an Buchstaben
    - Operationen an Sequenzen von Buchstaben/Zahlen
    - Arithmetik

Betriebssysteme

- "Higher-Level": traditionelle Operationen
- Buchhaltung
- Geräte und Komponenten "versteckt"
- Alles ist ein File (Unix)
- Alle Operationen sind File-Operationen
- Grenze ist eine Grau-Zone

Das nächste Mal

- Speicher und Adressen
- "Virtuelle" Speicher
- Verschiedenes