(Grundlagen der) Betriebssysteme | C.1

Franz J. Hauck | Institut für Verteilte Systeme, Univ. Ulm

C | Aufbau eines Rechnersystems

(Grundlagen der) Betriebssysteme

Franz J. Hauck | Institut für Verteilte Systeme, Univ. Ulm

Überblick

Überblick der Themenabschnitte

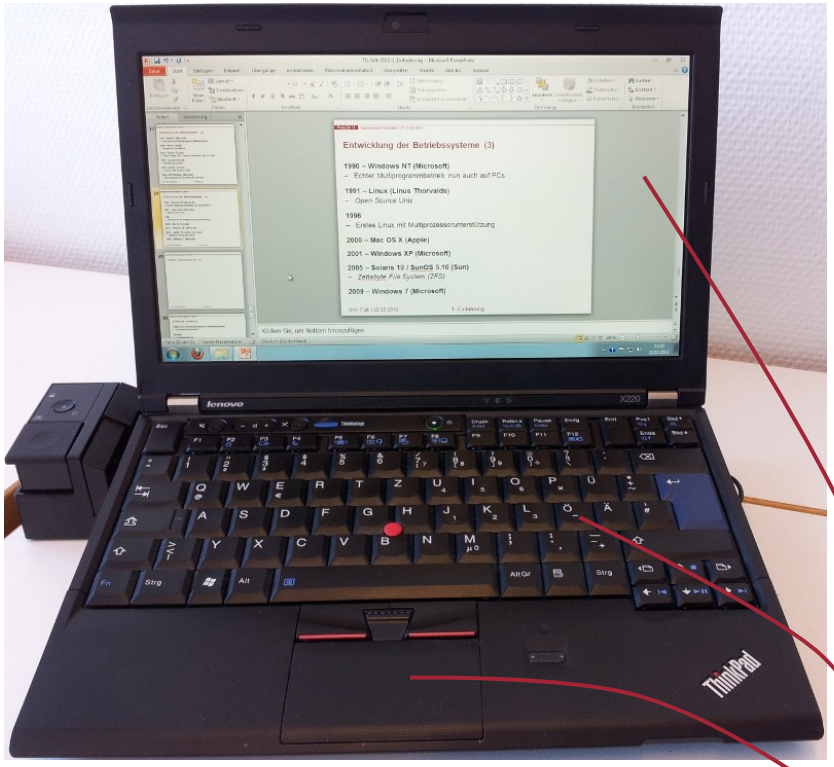
- A – Organisatorisches
- B – Zahlendarstellung und Rechnerarithmetik GdBS
- C – Aufbau eines Rechnersystems
- D – Einführung in Betriebssysteme
- E – Prozessverwaltung und Nebenläufigkeit
- F – Dateiverwaltung
- G – Speicherverwaltung
- H – Ein-, Ausgabe und Geräteverwaltung
- I – Virtualisierung BS
- J – Verklemmungen BS
- K – Rechteverwaltung

Inhaltsüberblick

Aufbau von Rechnersystemen

- Heutige Rechner
- typischer Hardware-Aufbau
  - Speicher
  - Prozessor
- Befehlsbearbeitung
  - Befehle
  - Reset
- Programmausführung

Heutige Rechner



Anwendungen

Betriebssystem

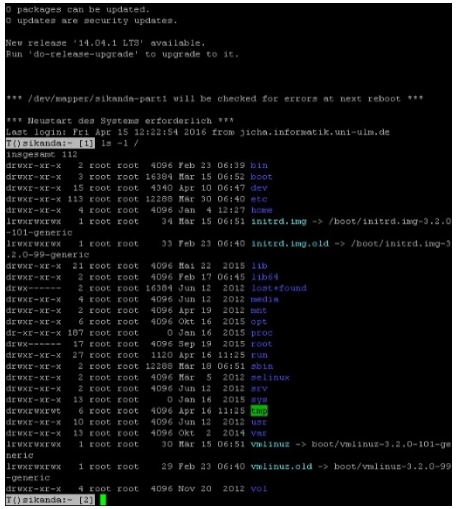
Hardware

Ein-, Ausgabe

Laptop

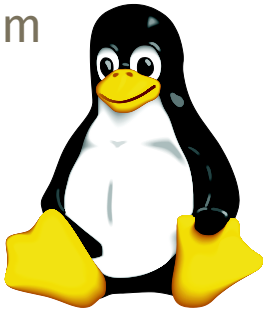
Heutige Rechner (2)

Anwendungen



Hardware

Betriebssystem



Ein-, Ausgabe

Server

Heutige Rechner (3)

Anwendungen

Betriebssystem

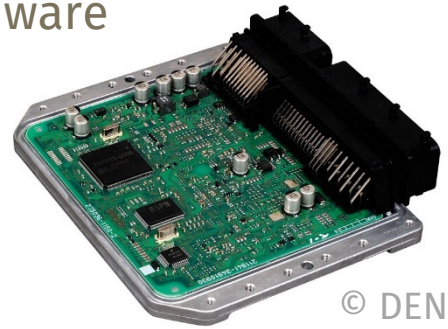
Ein-, Ausgabe

Hardware

Mobiles Gerät

Heutige Rechner (4)

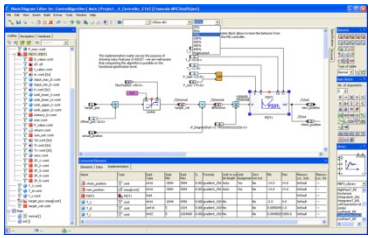
Hardware



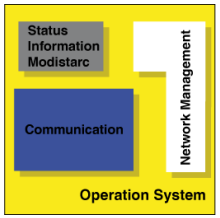
© DENSO

© Bosch

Anwendungen



Betriebssystem

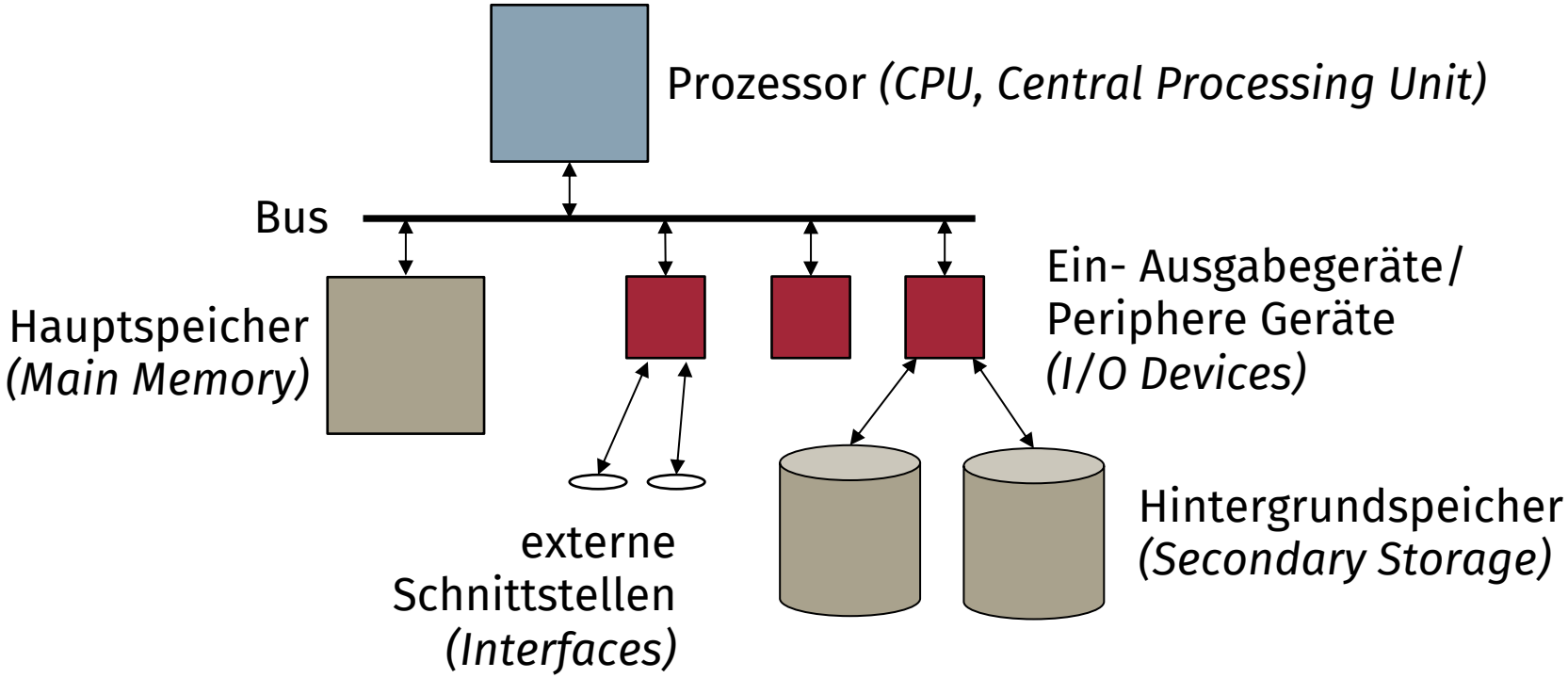


OSEK/VDK

Ein-, Ausgabe

Eingebettetes System

Hardware-Aufbau



Hauptspeicher

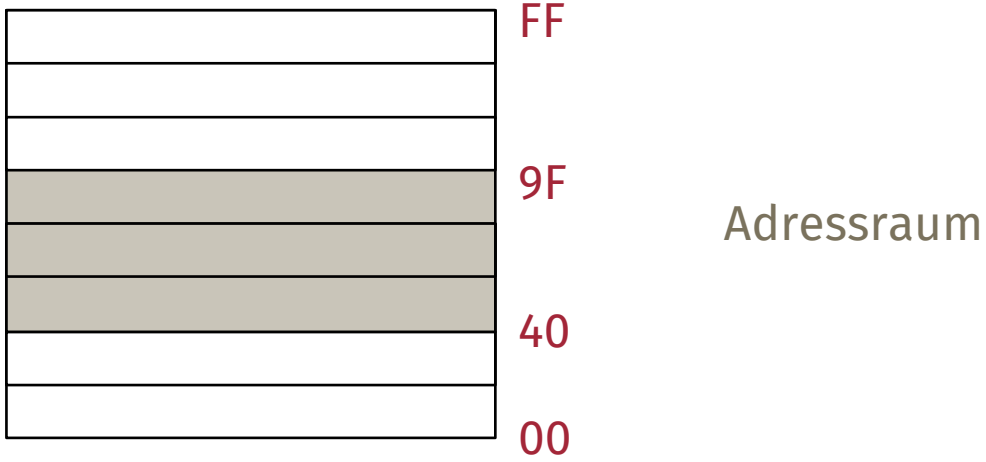
Menge von Speicherzellen

- Adressen zur Auswahl einer Zelle
  - typisch: positive ganze Zahl
  - z.B. $4A8218FE_{16}$
- feste Bitbreite für Adressen
  - Adressraum des Systems
  - z.B. 64 Bit, d.h. 18.446.744.073.709.551.616 verschiedene Adressen
  - nicht an allen Adressen tatsächlich Speicher verfügbar
- Zellengröße meist ein Byte
  - d.h. 8 Bit
  - aber: Speicherzugriff häufig wortweise, d.h. auf mehre Bytes gleichzeitig

Hauptspeicher (2)

Spielbeispiel

- Adressraum 8 Bit breit
  - d.h. 256 Speicherzellen à 1 Byte



  - aktueller Speicher von Adresse 40 bis 9F
    - 96 Byte

Prozessor

Aktive Einheit im System

- mehrere Register
  - z.B. 32 Stück
  - speichern Worte, z.B. 64 Bit
- Registernutzung
  - Zwischenergebnisse bei Rechnungen
  - Adressen, die in den Speicher verweisen
- Programmzähler *(Program Counter)*
  - Adresse in den Speicher
    für nächsten Maschinenbefehl *(Instruction)*

Prozessor (2)

Spielbeispiel

- vier Register

  - speichern jeweils 1 Byte
  - PC enthält Adresse einer Speicherzelle

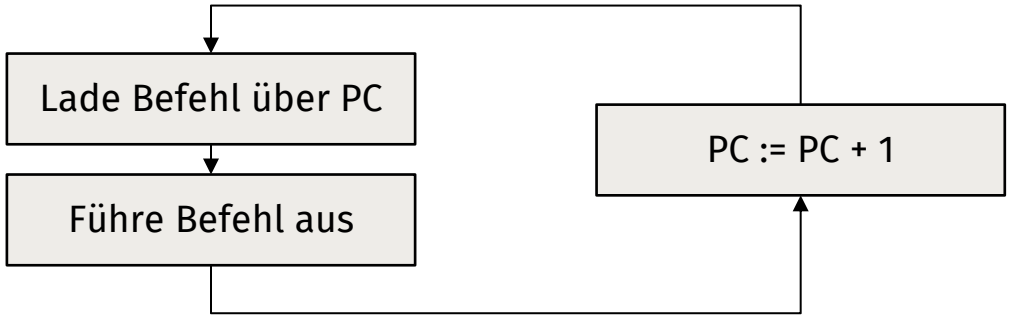
Befehlsbearbeitung

Abarbeitung von Maschinenbefehlen

- Laden des Inhalts der Speicherzelle des PC
  - meist internes zusätzliches Register für Inhalt
    - IR = *Instruction Register*
- Interpretation des Werts
  - Inhalt ist codierter Befehl
  - z.B. 8C entspricht: „Addiere R0 zu R1 und speichere Ergebnis in R0“
- Ausführen des Befehls
- Inkrementieren des PC
  - d.h. PC = PC + 1
  - PC zeigt nun auf nächsten Befehl im Speicher
- Beginn von vorne

Befehlsbearbeitung (2)

Schematische Darstellung



Befehlsbearbeitung (3)

Maschinensprache, Maschinen-Code

- codierte Werte der Befehle
  - z.B. `8C`

Assembler, Assemblerbefehl

- lesbare Notation für Befehle
  - z.B. `ADD R0, R1`
  - meist noch etwas mehr „Syntactic Sugar“ dabei

Befehlsbearbeitung (4)

Mögliche Speicherbefehle (Spielbeispiel)

- Laden von Werten aus dem Speicher
  - z.B. 5C entspricht „Lade eine Speicherzelle in Register R0“
    - in Assembler: `MOV <addr>, R0`
  - Adresse in der nächsten Speicherzelle
  - d.h. PC = PC + 1 und weiterer Speicherzugriff zum Holen der Adresse

- Befehl benötigt u.U. mehrere Bytes/Worte

Befehlsbearbeitung (5)

Mögliche Speicherbefehle (Spielbeispiel)

- Speichern von Werten in den Speicher
  - z.B. 4C entspricht „Speichere R0 in eine Speicherzelle“
    - in Assembler: `MOV R0, <addr>`
  - Adresse in der nächsten Speicherzelle
- Laden von Werten in ein Register
  - z.B. 6C entspricht „Lade eine Konstante in das Register R0“
    - in Assembler: `MOV #<value>, R0`
  - Wert in der nächsten Speicherzelle

Befehlsbearbeitung (6)

Mögliche Rechenbefehle (Spielbeispiel)

- Grundrechenarten
  - z.B. `ADD, SUB, MUL, DIV, MOD`
- bitweise Verknüpfung mit Booleschen Funktionen
  - z.B. `AND, OR, XOR, NOT`

- Bisher nur sequentielle Abarbeitung möglich!

Befehlsbearbeitung (7)

Sprungbefehle (Spielbeispiel)

- Ausbruch aus der sequentiellen Bearbeitung
- Sprung an eine neue Stelle
  - z.B. 5F entspricht „Springe an neue Adresse“
    - in Assembler: `JMP <addr>`
  - Adresse in der nächsten Speicherzelle
  - lädt Programmzähler mit neuer Adresse

- Endlosschleifen möglich, aber noch keine Alternativen!

Befehlsbearbeitung (8)

Erzeugen von Bedingungen

- Grundlage: Vergleich zweier Registerinhalte
  - kleiner, gleich größer bzw. Kombinationen davon
- Berechnungsgrundlage: Subtraktion der Werte
  - bei 0: gleich
  - kleiner 0: kleiner
  - größer 0: größer
- Spezielles Register *(Condition-Code Register, CCR)*
  - enthält verschiedene Bits *(Flags)*
  - Flags durch verschiedene Befehle gesetzt insbes. SUB
  - Flags signalisieren kleiner, gleich, größer

Befehlsbearbeitung (9)

Bedingte Sprungbefehle (Spielbeispiel)

- Springen bei Gleichheit
  - z.B. 70 entspricht „Springe bei Gleichheit“
    - in Assembler: `JEQ <addr>`
  - Adresse in der nächsten Speicherzelle
  - Auswertung des CCR
  - bei Ungleichheit der letzten Subtraktion sequentielle Abarbeitung
    - Grundlage für Verzweigungen!

Befehlsbearbeitung (10)

Bedingte Sprungbefehle (Spielbeispiel)

- Springen bei Ungleichheit
  - z.B. 71 entspricht „Springe bei Ungleichheit“
    - in Assembler: `JNE <addr>`
- Springen falls kleiner
  - z.B. 72 entspricht „Springe bei kleiner“
    - in Assembler: `JLT <addr>`
- Springen falls größer gleich
  - z.B. 73 entspricht „Springe bei größer gleich“
    - in Assembler: `JGE <addr>`

Start des Systems

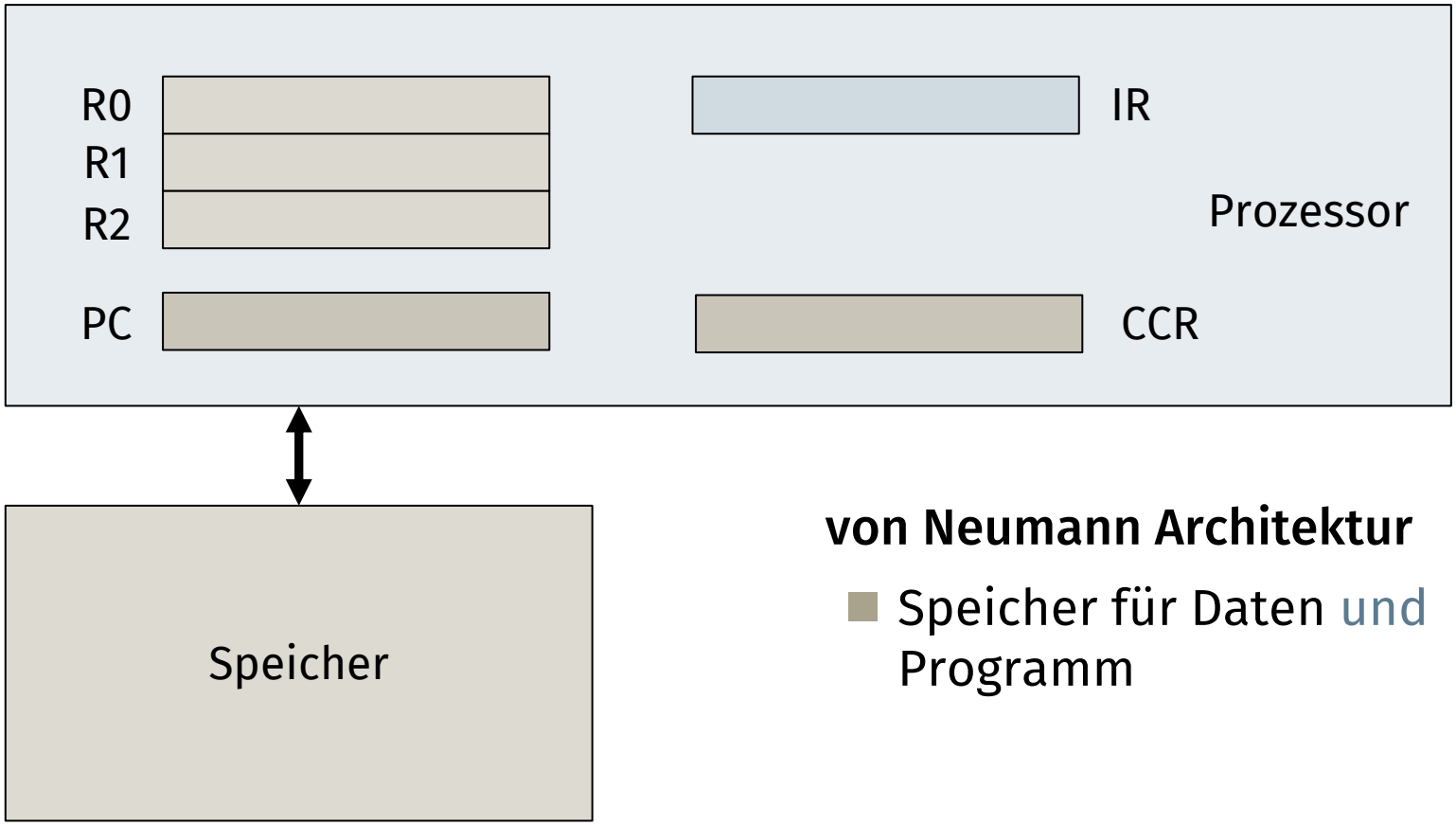
Neustart, Reset

- Initialisieren des PC mit fester Adresse
- Festwertspeicher (ROM) mit initialem Programm
  - BIOS, Basic Input Output System
  - Firmware
- Abarbeitung einer Startsequenz
  - Initialisierung der Hardware
  - Laden des Betriebssystems von einem Hintergrundspeicher
  - Starten des Betriebssystems

Ein-, Ausgabe

Vertagt bis zum Kapitel H

Architektur des Spielbeispiels



von Neumann Architektur

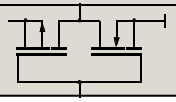
- Speicher für Daten und Programm

Programmausführung (Beispiel Java)

Abstraktionsebenen

Compiler

Assembler

Anwendungsprogramm	**Java:** … `for, while, if …`	GdPI, OOP / EidI
Zwischensprache	**Bytecode:** `iload1, ifne …`	–
Assemblerprogramm	`JNE loop`	
Maschinenprogramm	`01110001 00001011`	GdTI, RA / GdRA
Register-Transfer-Verhalten	`Reg[2] := Reg[3]`	
Register-Transfer-Struktur	**Addierer, Register …**	
Gatter	$f = a \wedge b$	GdTI, E-Technik
Transistoren		Physik

Inhaltsüberblick

Aufbau von Rechnersystemen

- Heutige Rechner
- typischer Hardware-Aufbau
  - Speicher
  - Prozessor
- Befehlsbearbeitung
  - Befehle
  - Reset
- Programmausführung

