

Aufbau und Funktionsweise eines Computers

Thomas Röfer

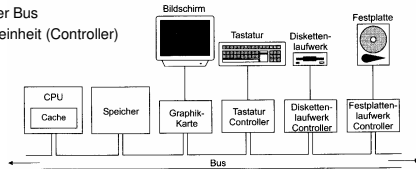
Hardware und Software
von Neumann Architektur
Schichtenmodell der Software
Zahlsysteme
Repräsentation von Daten im Computer

Tutorien und Praktika

	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
08		Tutorium	Tutorium		
09		Tutorium	Tutorium		
10	Vorlesung		Praktikum 54	Praktikum 18	Praktikum 29
11			Praktikum 42	Praktikum 18	Praktikum 29
12					
13					
14	Tutorium (Nebenfächler)	FAQ	Praktikum 18	Praktikum 35	Praktikum 21
15	Tutorium	Tutorium	Praktikum 18	Praktikum 35	Praktikum 21
16	Tutorium	Tutorium	Praktikum 18	Praktikum 35	Praktikum 21
17	Tutorium	Tutorium	Praktikum 19	Praktikum 13	
18	Tutorium	Tutorium	Tutorium	Praktikum 13	
19					

Hardware

- Prozessor (CPU)
- Hauptspeicher
- Peripherie
 - Festplatte, Monitor, Tastatur, Maus, CD/DVD Laufwerk/Brenner, Diskettenlaufwerk, Netzwerkkarten
 - Anbindung über Bus
 - Eigene Steuereinheit (Controller)



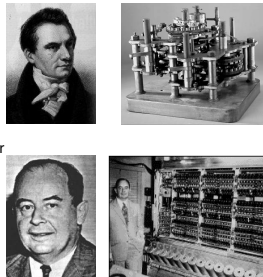
Software

- Betriebssystem
 - Isoliert Anwendersoftware von der Hardware
 - Verwaltet Ressourcen des Rechners
 - Firmware/BIOS
 - MSDOS, MS Windows, Linux, MacOS, Solaris, ...
 - PalmOS, MS Windows Mobile, SymbianOS, ...
- Systemsoftware
 - Lösung von Aufgaben im Rechner
 - z.B. Entwicklungswerkzeuge: Compiler, Linker, Debugger
- Anwendersoftware
 - Lösung von Problemen der externen Welt der Anwender
 - z.B. Textverarbeitung, Tabellenkalkulation, ..., Spiele
- Zuordnung zunehmend schwierig

Anwendersoftware
Betriebssystem
Hardware

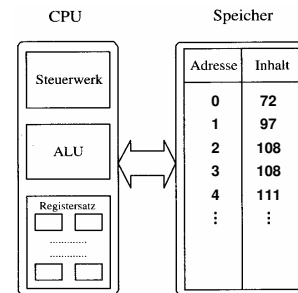
Grundlagen

- Analytical Engine
 - Charles Babbage (1834)
 - Prozessoreinheit
 - Speichereinheit
 - Programmsteuerung
- Programme
 - sind Sequenzen von Befehlen,
 - werden nacheinander vom Prozessor verarbeitet,
 - werden auf Daten im Speicher angewendet
- von Neumann Architektur
 - 1950
 - elektronischer Rechner



von Neumann Architektur

- Prozessor (CPU)
 - Steuernwerk
 - Arithmetisch-logische Einheit (ALU)
 - Registersatz
- Speicher
 - Wahlfreier Zugriff (RAM)
 - Jede Speicherzelle hat
 - eine Adresse
 - einen Inhalt
 - Beispiel: ein PC mit 512 MB hat 536.870.912 Speicherzellen
- Programm und Daten im selben Speicher



Bits und Bytes

- 1 Bit kann 2 Zustände einnehmen: 0 oder 1
- Mehrere Bits: 2^{Anzahl} Zustände
- Bei n Bits
 - Bit 0 = least significant bit (LSB)
 - Bit n-1 = most significant bit (MSB)
- 1 Byte = 8 Bits
 - 2^8 Zustände = 256 Zustände = 0 ... 255
 - Speicherzellen sind 1 Byte groß
- 1 Nibble = 4 Bits
- 1 Wort
 - Anzahl Bits ergibt sich aus der Breite des Datenbusses
 - Aktuell: IA32: 32 Bit, IA64: 64 Bit
 - Adresse eines Wortes ist immer die Adresse des ersten Bytes des Wortes
 - Das LSB kann Teil des ersten Bytes eines Wortes sein (little endian) oder Teil des letzten Bytes (big endian)

Bit 2	Bit 1	Bit 0	
0	0	0	Zustand 0
0	0	1	Zustand 1
0	1	0	Zustand 2
0	1	1	Zustand 3
1	0	0	Zustand 4
1	0	1	Zustand 5
1	1	0	Zustand 6
1	1	1	Zustand 7

Adresse	Little end.	Big endian
n	Bits 0-7	Bits 24-31
n+1	Bits 8-15	Bits 16-23
n+2	Bits 16-23	Bits 8-15
n+3	Bits 24-31	Bits 0-7

Bits und Bytes

- 1 Bit kann 2 Zustände einnehmen: 0 oder 1
- Mehrere Bits: 2^{Anzahl} Zustände
- Bei n Bits
 - Bit 0 = least significant bit (LSB)
 - Bit n-1 = most significant bit (MSB)
- 1 Byte = 8 Bits
 - 2^8 Zustände = 256 Zustände = 0 ... 255
 - Speicherzellen sind 1 Byte groß
- 1 Wort
 - Anzahl Bits ergibt sich aus der Breite des Datenbusses
 - Aktuell: IA32: 32 Bit, IA64: 64 Bit
 - Adresse eines Wortes ist immer die Adresse des ersten Bytes des Wortes
 - Das LSB kann Teil des ersten Bytes eines Wortes sein (little endian) oder Teil des letzten Bytes (big endian)

Beispiel: „3“ als Wort
 Little endian: 03 00 00 00
 Big endian: 00 00 00 03

Bit 2	Bit 1	Bit 0	
0	0	0	Zustand 0
0	0	1	Zustand 1
0	1	0	Zustand 2
0	1	1	Zustand 3
1	0	0	Zustand 4
1	0	1	Zustand 5
1	1	0	Zustand 6
1	1	1	Zustand 7

Adresse	Little end.	Big endian
n	Bits 0-7	Bits 24-31
n+1	Bits 8-15	Bits 16-23
n+2	Bits 16-23	Bits 8-15
n+3	Bits 24-31	Bits 0-7

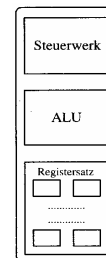
Zustände und ihre Interpretationen

- Bits im Speicher repräsentieren Zustände
- Ihre Bedeutung hängt davon ab, wie sie interpretiert werden
 - 4 Bytes (z.B. 65 117 116 111) können z.B.
 - als ganze Zahl gedeutet werden (integer, z.B. 1869903169)
 - als kurze Gleitkommazahl (float, z.B. 7.56561e+028)
 - als Folge von vier Zeichen (string, z.B. „Auto“)
 - als Folge von Befehlen für den Prozessor


```
INC    CX
JNZ   +116
DB    111
```
 - als Farbe (vom Grafikchip, z.B. ████████)
- In Programmiersprachen wird die Interpretation durch Datentypen festgelegt

Instruktionszyklus einer CPU

- Programmausführung
 - Programme sind in Maschinsprache kodiert
 - Prozessortyp-abhängig (z.B. PowerPC vs. IA32)
 - Spezielles Register: Befehlszähler
- Fundamentaler Instruktionszyklus einer CPU
 - Fetch:** Hole den Befehl, dessen Adresse im Befehlszähler steht, aus dem Speicher in das Instruktionsregister.
 - Increment:** Inkrementiere den Befehlszähler, damit er auf die nächste auszuführende Instruktion verweist.
 - Decode:** Dekodiere die Instruktion, damit klar wird, was zu tun ist.
 - Fetch Operands:** Falls nötig, hole die Operanden aus den im Befehl bezeichneten Stellen im Speicher.
 - Execute:** Führe die Instruktion aus, ggf. durch die ALU. (Bei einem Sprung wird hier ein neuer Wert in den Befehlszähler geschrieben.)
 - Loop:** Gehe zu Schritt 1 □

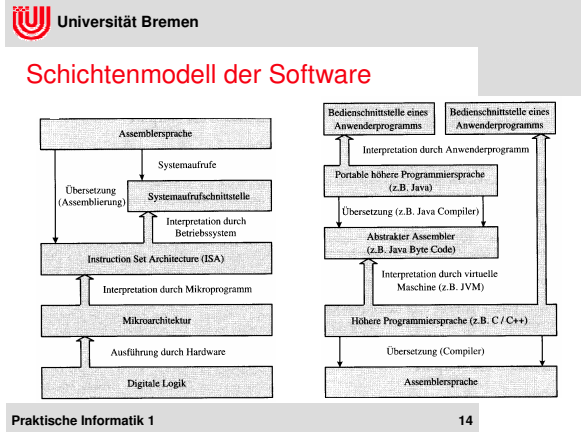
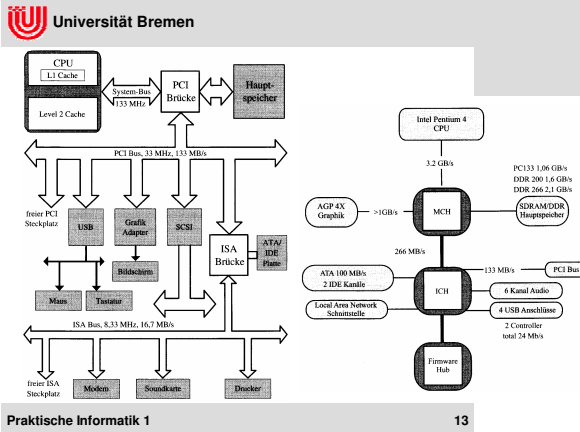


Programmausführung

- Typische Befehle
 - laden (Speicher → Register)
 - schreiben (Register → Speicher)
 - verknüpfen von Registern (z.B. addieren)
 - (bedingter) Sprung
- Takt
 - Befehle werden getaktet ausgeführt
 - Anzahl Takte/Befehl hängt vom Befehl ab
 - Beschleunigung durch Fließbandverarbeitung (Pipelining)
- von Neumannscher Flaschenhals (bottleneck)
 - Prozessor (viel) schneller als Speicher
 - Dadurch muss Prozessor oft auf Speicher warten
 - Gebräuchliche Lösungsansätze
 - viele CPU-Register
 - schnelle Zwischenspeicher (Cache)

Moore's Gesetz

- Gordon Moore (Intel):
 - Die Anzahl der Transistoren pro Chip verdoppelt sich alle 18 Monate
- Folgen
 - Verdopplung der Speichergröße alle 18 Monate
 - Verdopplung der Geschwindigkeit alle 18 Monate
- Beispiel
 - 1981: IBM PC, 4,77 MHz, 64 KB RAM
 - 2003: Intel P4, 3,2 GHz (nur $2^{9,4}$ · 4,77MHz, aber P4 braucht weniger Takte pro Instruktion), 512 MB RAM (2^{13} · 64 KB)



- Betriebssystem**
- Verwaltet Ressourcen eines Rechners
 - Systemaufrufsschnittstellen (oft APIs – Application Programming Interfaces)
 - Verwaltet Prozesse und ihre Ressourcen
 - Prozessorzeit (oft in Zeitscheiben)
 - Speicher
 - Dateien
 - Kommunikationspfade
 - Enthält Gerätetreiber
 - bzw. bietet Schnittstellen für Gerätetreiber
 - Bietet Ein- und Ausgabeströme
 - Dateisystem
 - TCP/IP

- Zahlsysteme**
- Endlich (β) viele Ziffern (z.B. 0 ... 9 beim Dezimalsystem, $\beta = 10$)
 - β wird auch Radix genannt
 - Schreibung einer Zahl: $z_{n-1} \dots z_0, 0 \leq z_i < \beta$
 - Wert der Zahl: $z = \sum_{i=0}^{n-1} z_i \beta^i$
 - Beispiel
 - $42_{10} = 4 \cdot 10_{10}^1 + 2 \cdot 10_{10}^0$
 - $= 100_2 \cdot 1010_2^1 + 10_2 \cdot 1010_2^0$
 - $= 101000_2 + 10_2$
 - $= 101010_2$
 - $101010_2 = 1_2 \cdot 10_2^5 + 0_2 \cdot 10_2^4 + 1_2 \cdot 10_2^3 + 0_2 \cdot 10_2^2 + 1_2 \cdot 10_2^1 + 0_2 \cdot 10_2^0$
 - $= 1_{10} \cdot 2_{10}^5 + 0_{10} \cdot 2_{10}^4 + 1_{10} \cdot 2_{10}^3 + 0_{10} \cdot 2_{10}^2 + 1_{10} \cdot 2_{10}^1 + 0_{10} \cdot 2_{10}^0$
- | Dezimal | Dual |
|---------|---------|
| 0 | ...0000 |
| 1 | ...0001 |
| 2 | ...0010 |
| 3 | ...0011 |
| 4 | ...0100 |
| 5 | ...0101 |
| 6 | ...0110 |
| 7 | ...0111 |
| 8 | ...1000 |
| 9 | ...1001 |

- Dezimal \rightarrow Dualsystem**
- Algorithmus
 - Ist die umzurechnende Zahl gerade, schreibe ein 0 sonst eine 1.
 - Teile die umzurechnende Zahl durch 2 (mit Abrunden).
 - Ist die umzurechnende Zahl nicht Null, weiter mit 1.
 - Die Ziffern 0 und 1 müssen von rechts nach links geschrieben werden.
 - Beispiel
 - $42_{10} = 42_{10} \cdot 1_2$
 - $= 21_{10} \cdot 10_2 + 0_2$
 - $= 10_{10} \cdot 100_2 + 10_2$
 - $= 5_{10} \cdot 1000_2 + 010_2$
 - $= 2_{10} \cdot 10000_2 + 1010_2$
 - $= 1_{10} \cdot 100000_2 + 01010_2$
 - $= 101010_2$

- Negative Zahlen (Möglichkeiten)**
- Vorzeichenbit
 - MSB für Vorzeichen reservieren (0 \rightarrow positiv, 1 \rightarrow negativ)
 - $-3_{10} = -0011_2 \equiv 1011_2$
 - Einerkomplement
 - Alle Bits invertieren
 - $-3_{10} \equiv 1100_2$
 - Zweierkomplement
 - Alle Bits invertieren und eins dazuzählen
 - $-3_{10} \equiv 1101_2$
 - Entspricht: $16_{10} - 3_{10} = 13_{10} = 1101_2$
 - Vorteil: $5_{10} - 3_{10} = 5_{10} + (-3_{10})$
 - $\equiv 0101_2 + 1101_2$
 - $\equiv 10010_2$
 - Grund: $\equiv 5_{10} + (16_{10} - 3_{10}) = 16_{10} + (5_{10} - 3_{10})$
 - Modulo-Arithmetik
 - $(5_{10} - 3_{10}) \bmod 2_{10}^4$
 - $= (5_{10} + (-3_{10})) \bmod 2_{10}^4$
 - $= (0101_2 + 1101_2) \bmod 10000_2$
 - $= 10010_2 \bmod 10000_2$
 - $= 0010_2$

Weitere Zahlssysteme

Hexadezimalsystem

- › Radix $\beta = 16$, Ziffern 0 ... 9, A, B, C, D, E, F
- › Eine Hexadezimalziffer entspricht 4 Bit
- › ein Byte kann durch 2 Ziffern dargestellt werden

Oktalsystem

- › Radix $\beta = 8$, Ziffern 0 ... 7
- › Eine Oktalziffer entspricht 3 Bit
- › Wird eher selten verwendet

Dezimal	Zweierkomplement
+8	nicht darstellbar
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
0	0000
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000