



---

# Einführung in die Informatik I

Winter 2005/2006

## 3 Rechnerarchitektur



## 3 Rechnerarchitektur ...

---

### Motivation: Ziele dieses Abschnittes

- Grundbegriffe der *von-Neumann-Rechner*s (insb. heutige PCs)
- grundlegende Komponenten und deren Zusammenwirken
- Datenverarbeitung auf unterer Ebene

### Herausforderungen:

- (potentiell) viele neue Begriffe
- vieles wird nur grob angerissen (Vertiefung z.B. in Vorlesung ***Rechnerarchitektur***)

### Literatur: Diverse Abschnitte in [Claus] plus

- [Ernst] Abschnitt 1.3, tw. Kapitel 5 und 6
- Lehrmaterial der Universität Hamburg, Prof. Zhang:  
`tams-www.informatik.uni-hamburg.de/applets/baukasten/DA/Inhalt.html`

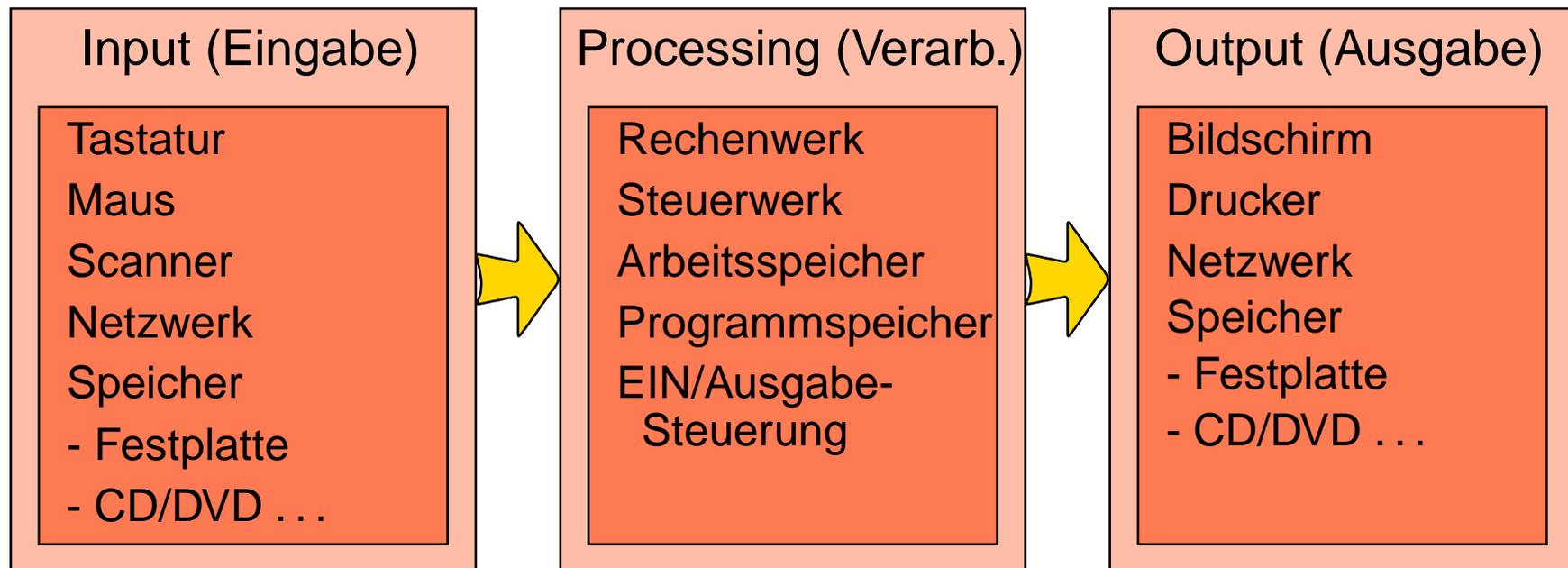
## 3.1 Abstraktes IPO-Prinzip

**IPO:** Abstraktes Verarbeitungsschema für beliebige Prozesse

Input → Processing (Verarbeitung) → Output

**Input & Output** sind Mensch-Maschine Schnittstellen

**Konkret** für einen Rechner



## 3.2 Von-Neumann Rechnerarchitektur

---

**John von Neumann (1903-1975):** Physiker, Konzept des modernen *General-Purpose-Computers*

- Grundprinzip der heutigen PCs und Server
- Ausnahmen z.B. Parallelrechner

**Grundprinzipien:** (nur die Wichtigsten)

- Problemunabhängigkeit → **Programm** steuert konkreten Ablauf
- Keine Trennung von Daten und Programm → **Speicher**

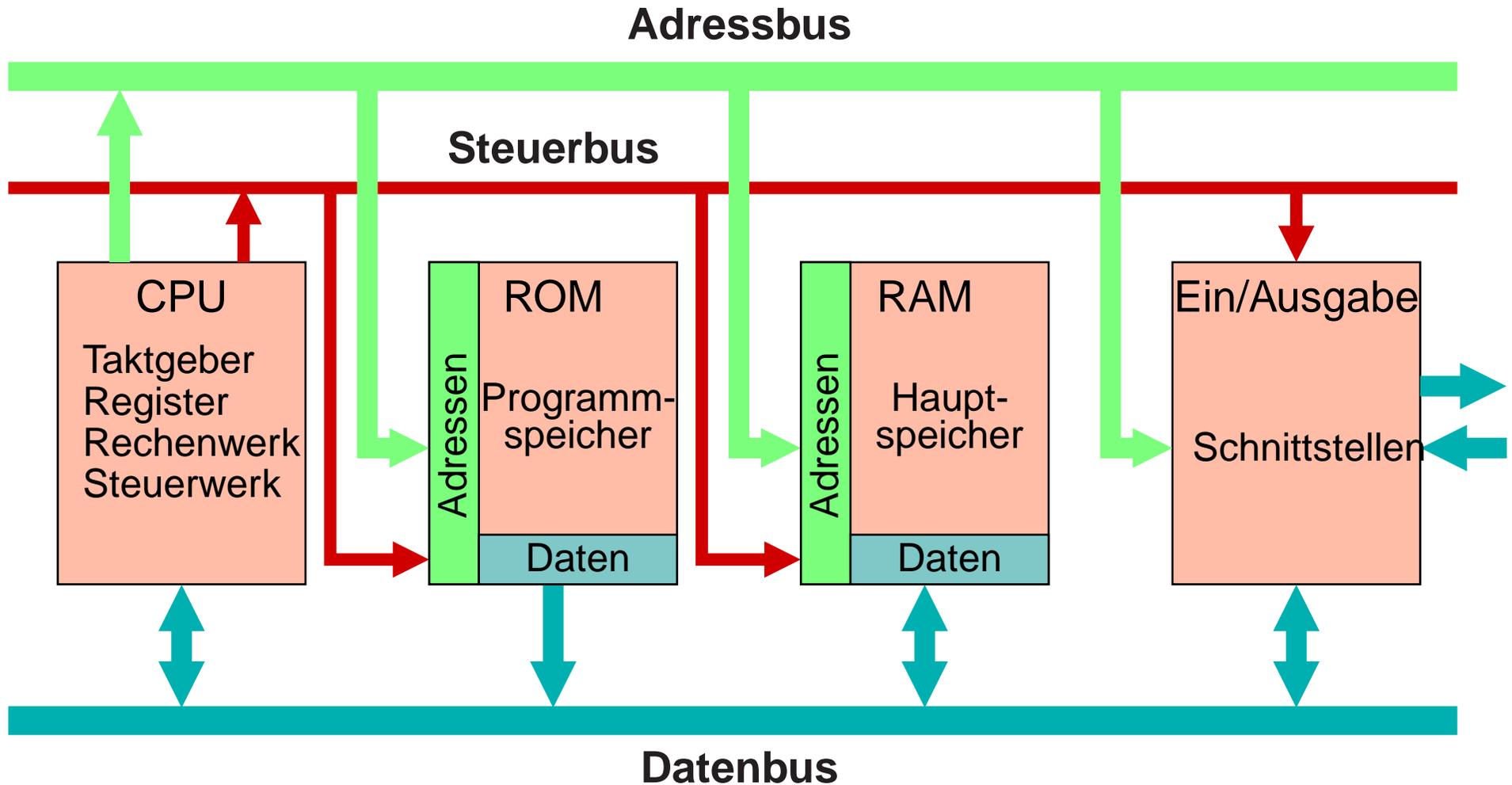
### Komponenten eines Von-Neumann-Rechners

- Steuerwerk: Kontrolliert Ablauf im Gesamtsystem „Rechner“
- Rechenwerk (**Arithmetic & Logic Unit, ALU**): Arithmetische & logische Berechnungen
- Arbeitsspeicher: Enthält maschinenlesbare Programme und Daten
- Ein-/Ausgabeeinheit: Bereitstellen von und Speichern auf externen Geräten

## 3.2 Von-Neumann Rechnerarchitektur ...



### Prinzipaufbau moderner Rechner





## 3.2.1 Central Processing Unit (CPU)

---

**Central Processing Unit (CPU)** = Zentrale Verarbeitungseinheit

**Aufgabe:** ○ Befehle interpretieren und ausführen

- zu jedem Zeitpunkt wird genau 1 Befehl ausgeführt, der genau 1 Datenwert berechnet

**Speicherplatz auf CPU:** Register und Caches

- eingeschränkter Speicherplatz mit schnellem Zugriff
- Speicherung von Ergebnissen & Zuständen, z.B. Befehlsregister, Befehlszähler, Akkumulator

**Steuerwerk:** Teil des von-Neumann-Konzeptes

- Register: Speicherplatz für einzelne Daten/Befehle
- Befehlsdekodierer (Umsetzung von Maschinenbefehlen)

**Arithmetic Logical Unit, ALU:** Teil des von-Neumann-Konzeptes

- Ausführung von Operationen (+, −, \*, /, ∧, ∨, ...)
- zwei Eingangsregister (**Operanden**), ein Ausgangsregister
- weitere Arbeitsregister



## 3.2.1 Central Processing Unit (CPU) ...



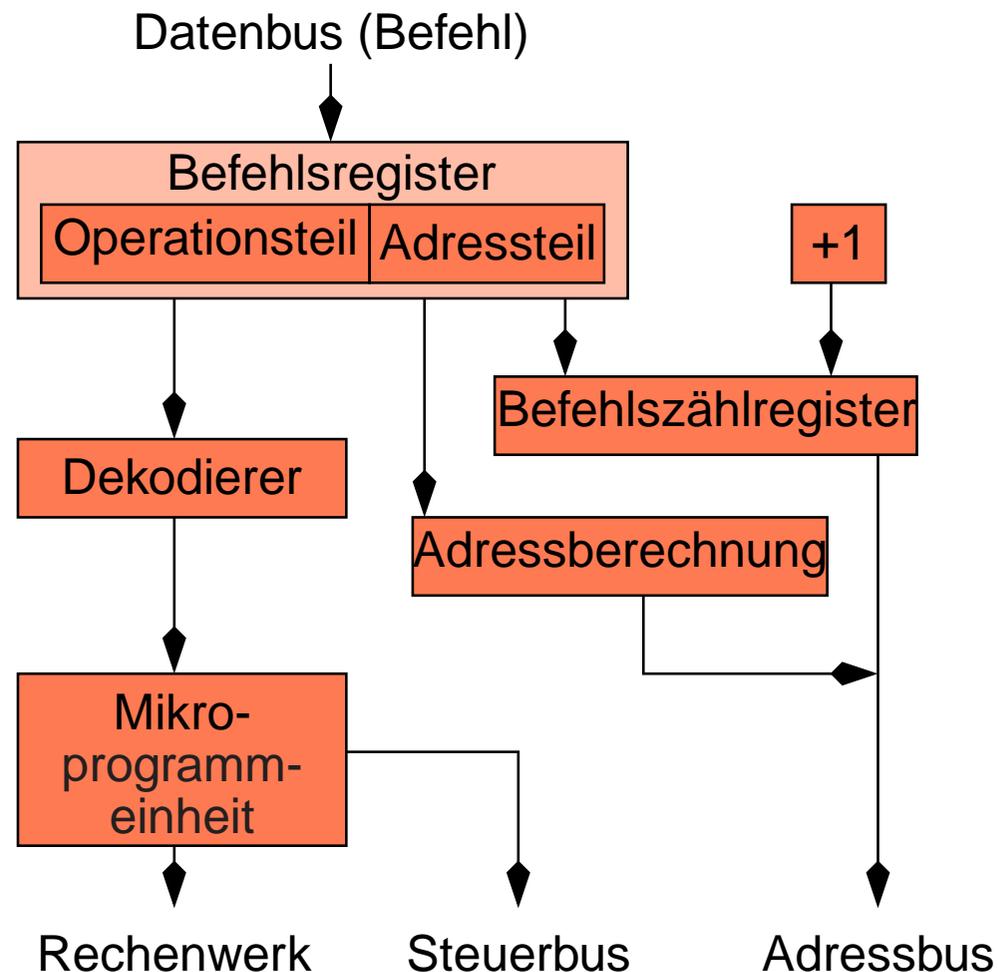
### Das Steuerwerk

- Laden von Programmbefehlen, z.B.

```
LDA XX0001
```

(lade Daten aus Speicherzelle XX0001 in Akkumulator)

- Befehlsdekodierung: Zerlegung in **Elementaroperationen**
- Steuerung der an Befehlsausführung beteiligten Einheiten, insb. Auswahl der Operationen für ALU (Rechenwerk)



## 3.2.1 Central Processing Unit (CPU) ...



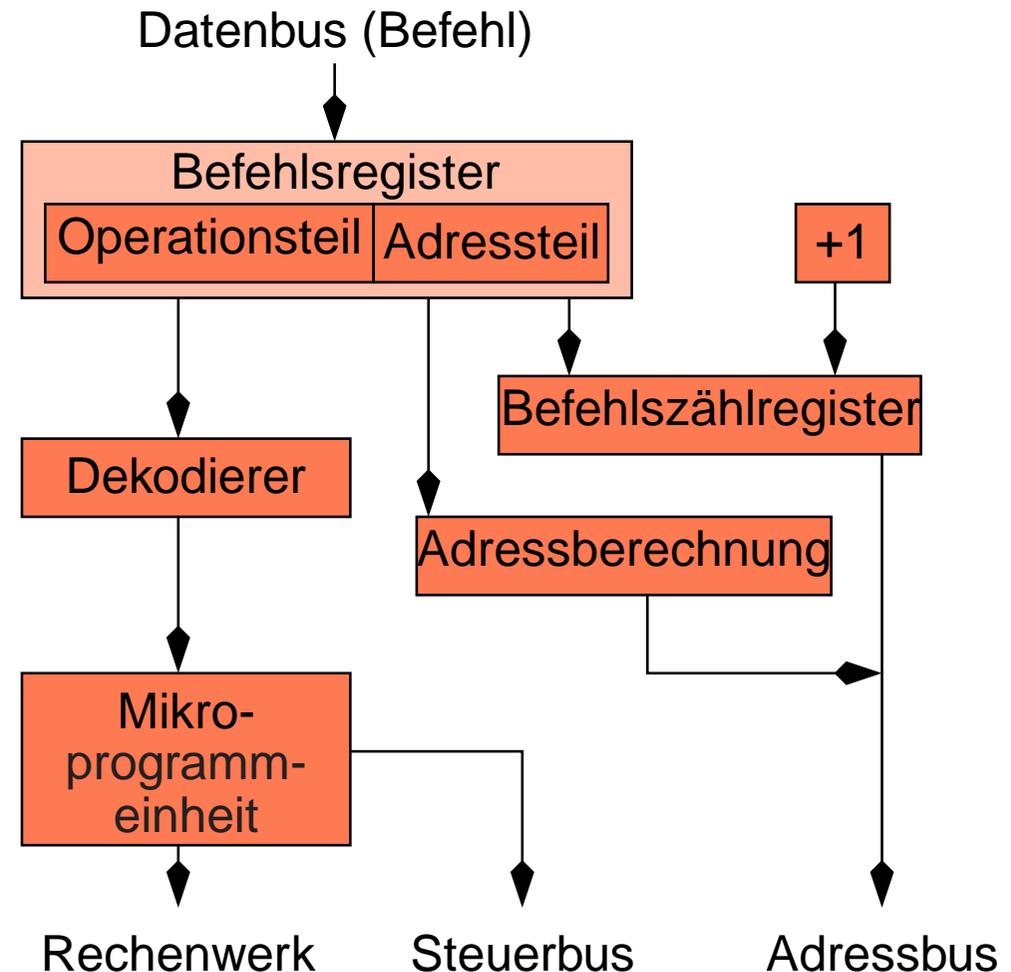
### Das Steuerwerk – Register

- Befehlsregister hält auszuführenden Befehl

Zerlegung Operand & Adresse

Adresswert i.A. relativ (→ Adressberechnung)

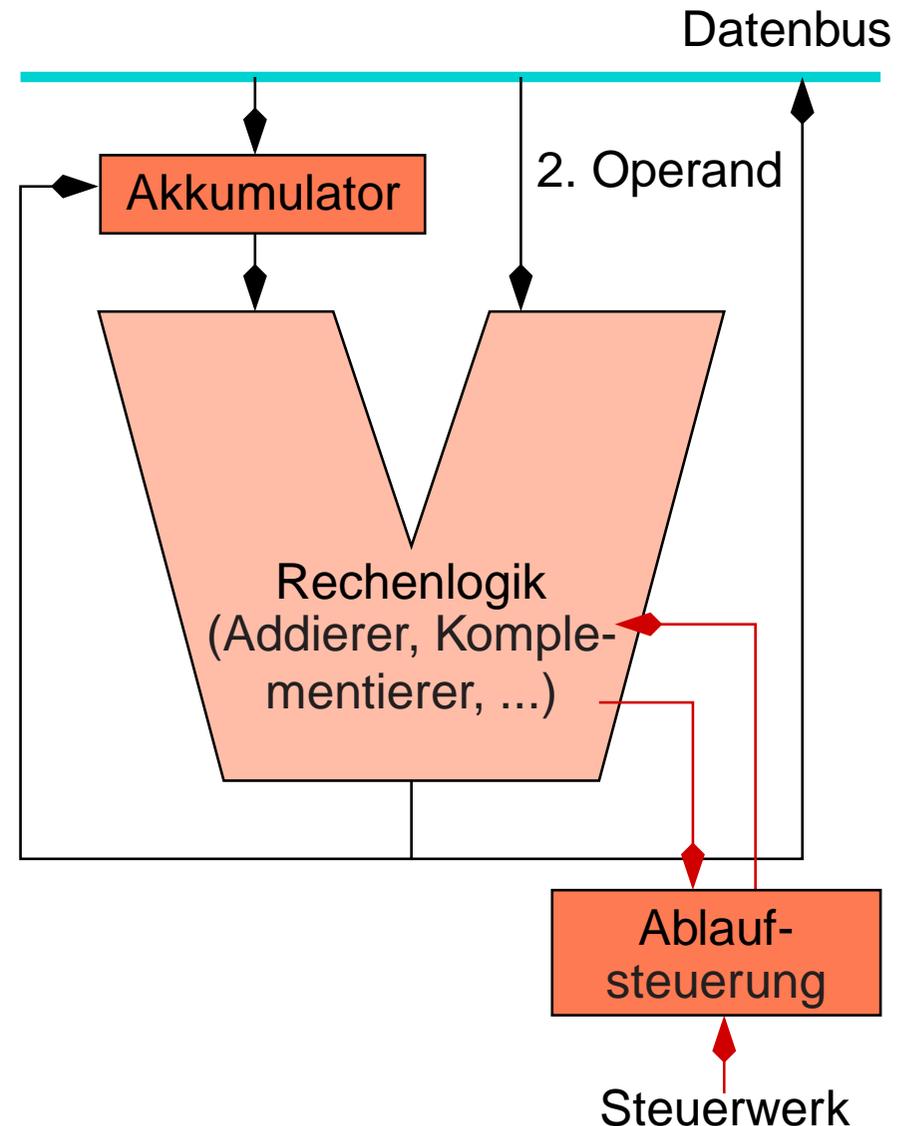
- Befehlszählregister hält Adresse des nächsten Befehles
  - Fortschreiten („+1“)
  - Springen (Befehl JMP XX0024)
- weitere Zustandsregister



### Das Rechenwerk

- Akkumulator:
  - erster Operand für Berechnung
  - hält Ergebnis nach Berechnung
- Zweiter Operand direkt vom Datenbus
- Rechenlogik: I.W. addieren & Komplementbildung
- Ablaufsteuerung: Folge von **Elementaroperationen** (**Mikroprogramm**)

z.B.: Multiplikation → Additionen





### Befehlsausführungszyklus

**Programm** = Folge von Befehlen

**Befehlszyklus:** Ausführung eines einzelnen Befehls

#### **Konkreter Ablauf** in einem Befehlszyklus

1. Holen der aktuellen Befehlsadresse
2. Laden des Befehls in den Befehlsspeicher
3. Befehlsregister um 1 erhöhen bzw. bei Sprungsbefehl mit neuer Adresse (Operand) laden
4. Befehlsdekodierung (Umsetzung in Elementaroperationen)
5. Operandenbereitstellung
6. Ausführung der Elementar-Operationen
7. Ergebnisspeicherung im Akkumulator



**Random Access Memory (RAM):** Speicher mit Lese- & Schreibzugriff

- speichert Daten und Programme
- Identifikation der Lese- bzw. Schreibposition über Adressen
- keine persistente Speicherung von Daten bzw. Programmen

**Read-Only Memory (ROM):** Festwertspeicher mit Lesezugriff

- Speicherung der **Mikroprogramme** für das Rechenwerk
- Identifikation der Lese- bzw. Schreibposition über Adressen

**Kodierung:** von Daten im Speicher → später

## 3.2.2 Speicher ...

### Größeneinheiten

- 1 bit ist ein binäres Datum (zwei Zustände: 0, 1)
- 1 byte = 8 bit  $\Rightarrow 2^8 = 16$  unterscheidbare Werte
- Häufig verwendete, aber falsche Benennungen:

Bez.	Abk.	Anzahl Byte	korrekt
Kilo-Byte	KB	$2^{10} = 1.024$	1 KB = $10^3 = 1.000$ Byte
Mega-Byte	MB	$2^{20} = 1.048.576$	1 MB = $10^6$ Byte
Giga-Byte	GB	$2^{30} = 1.073.741.824$	1 GB = $10^9$ Byte
Tera-Byte	TB	$2^{40} = 1.099.511.627.776$	1 TB = $10^{12}$ Byte

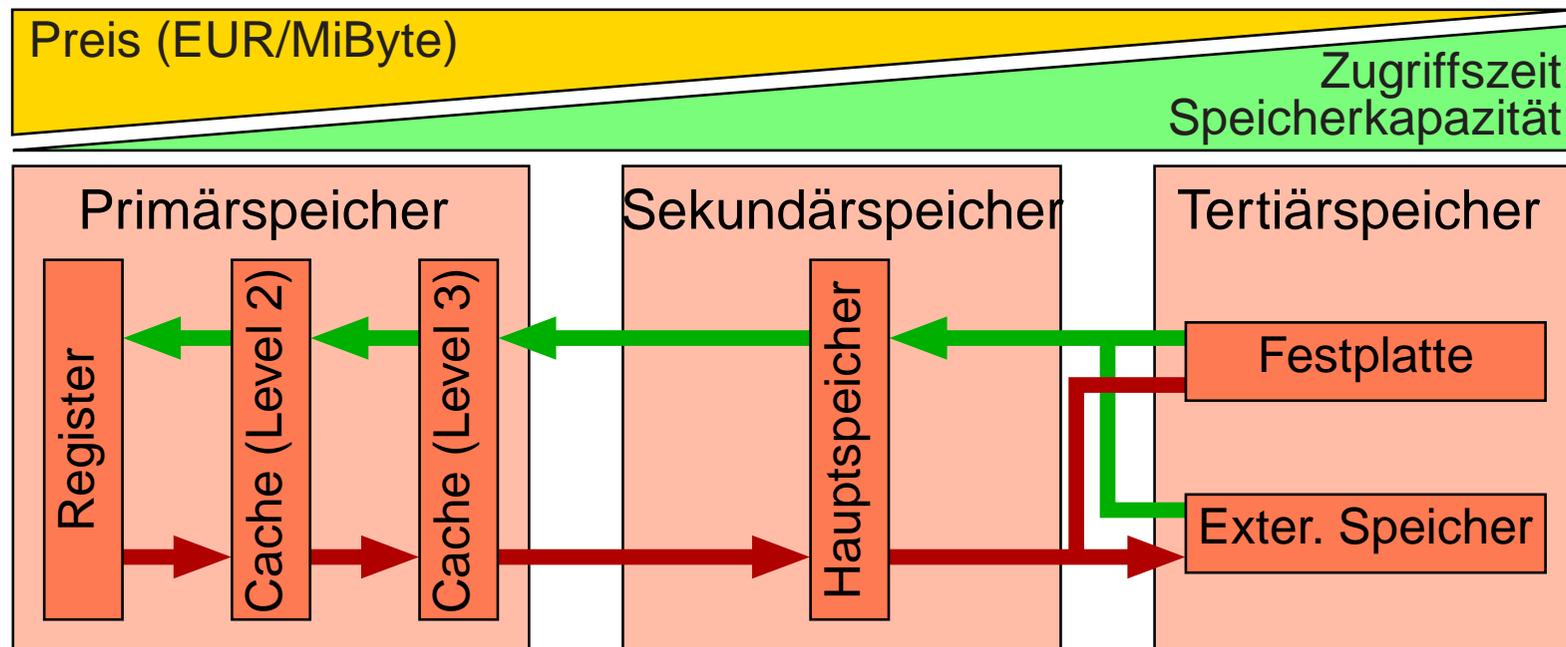
- Benennungen nach IEC-Norm 60027-2

Bez.	Abk.	Anzahl Byte
Kibi-Byte	KiB	$2^{10} = 1.024$
Mebi-Byte	MiB	$2^{20} = 1.048.576$
Gibi-Byte	GiB	$2^{30} = 1.073.741.824$
Tebi-Byte	TiB	$2^{40} = 1.099.511.627.776$

## 3.2.2 Speicher ...

### Speicher-Hierarchie

Drei Speicherebenen für Daten & Programme



Caches speichern

- zuletzt genutzte Daten
- zuletzt berechnete Ergebnisse



**Ziel:** Verbindung von Komponenten zum Austausch von Daten

**Problem:** Direkte Verkabelung „aller mit allen“ extrem platzintensiv

**Ansatz:** Mehrere Geräte nutzen einen Datenübertragungsweg

- zusätzliche Steuerung: Ein Gerät sendet, mind. eines empfängt Daten

**Serieller Bus:** Sequentielle Übertragung von Daten im Binärformat (0-1-Folge)

**Paralleler Bus:** Gleichzeitige Übertragung über n Leitungen (n ist Busbreite)

Typisch: 32-Bit und 64-Bit Busse

**Bandbreite eines Busses:** Übertragungsrates pro Zeiteinheit

**Systembus:** Gesamtheit der Busse im Rechner:

- Datenbus: Überträgt Daten & Befehlen vom/zum Speicher
- Adressbus: Festlegung der Speicheradresse
- Steuerbus: Übertragung von Steuerbefehlen (z.B. „Schreibe“, „Lese“)



### Probleme der Von-Neumann-Architektur

**Von-Neumann Flaschenhals:** Bandbreite des Datenbusses rel. zur CPU-Leistung gering

⇒ Datenbereitstellung bremst u.U. CPU aus

**Beliebiger Datenzugriff:** Einzelner Befehl legt zu verarbeitende Daten fest  
⇒ Datenbereitstellung erst *nach* Befehlsdekodierung

**Lokalitätsprinzip:** Nächster Befehl nutzt höchstwahrscheinlich „benachbarte Daten“ im Speicher/Cache

Konkrete Lage der Daten im Speicher beeinflusst **Performanz:**

- schneller Zugriff auf Daten in Nähe der zuletzt geladenen Daten
- langsamer Zugriff auf Daten entfernt zu zuletzt geladenen Daten



**Datum:** Einzahl von *Daten*

- In der Informatik: Maschinenlesbare und -bearbeitbare Struktur zur Repräsentation von *Information* (Information  $\neq$  Daten)
- Wird in Zeichen kodiert, deren Aufbau strengen Regeln (der *Syntax*) folgt

**Datei:** Daten-Verbund auf einem Speichermedium

- Besitzt einen Namen (z.B. Inhalt.txt)
- Besteht aus inhaltlich zusammengehörigen Daten
- Existiert über die Laufzeit eines Programms hinaus (*Persistenz*)
- Daten können in einer (menschlich-) lesbaren Form (ASCII-Dateien) oder einer binären Form abgelegt sein
- Dateiendung (meist die letzten drei Zeichen hinter einem Punkt) bezeichnet Typ des Inhalts (z.B. txt für einfache Text-Datei)



**Algorithmus:** Handlungsvorschrift zur Lösung eines Problems

- Exakt festgelegter Ablauf mit dessen Hilfe ein bestimmtes **Ziel** erreicht werden kann (auch z.B. Kochrezept)
- **Abstraktes** Gegenstück zu der Umsetzung in Form eines konkreten Programms

**Programm:** von griech.: próγραμμα = Vorschrift

- Ein Programm ist eine konkrete Umsetzung eines Algorithmus (oder mehrerer Algorithmen)
- Ablauf besteht aus einer Folge von einzelnen **Befehlen**, die vom Prozessor verarbeitet werden
- Sowohl der in einer Programmiersprache geschriebene **Quelltext**, als auch der vom Computer ausführbare **Maschinencode** wird als Programm bezeichnet
- Quelltext und Maschinencode werden in *unterschiedlichen* Dateien auf der Festplatte gesichert. Nur der Maschinencode ist durch den Prozessor ausführbar



### Assembler, Compiler, Interpreter: Übersetzer

- Programme, die ein Programm einer Sprache in eine andere überführen
- Compiler übersetzen **Quelltext**, der in **Hochsprachen** (wie C++ oder Java) geschrieben wurde, in Assembler-Code
- Interpreter lesen den Quelltext direkt ein, analysieren ihn und führen ihn unmittelbar aus
- Assembler übersetzen maschinennahen Assembler-Code in **Maschinencode** (dies ist der einzige Code den Prozessoren ausführen können)



**Betriebssystem:** Programm zum Ausführen von Programmen

- **Software** (Programm), die zum Betreiben eines Computers benötigt wird
- Verwaltet Betriebsmittel wie Speicher, Prozessor und andere Geräte
- Steuert die Ausführung von Programmen
- Besteht aus einem Kern (**Kernel**), der die Hardware steuert und grundlegenden Systemprogrammen
- Schnittstellen zur Betriebssystemnutzung und Konfiguration:
  - Mindestens Kommandozeileninterpreter (auch **Shell** genannt)
  - Graphische Benutzeroberfläche (**GUI** = Graphical User Interface)
- Populäre Beispiele: Unix und dessen Derivate (Linux, BSD,...), Windows, MacOS



**Benutzer:** Unterschiedliche Form der Nutzung eines Rechners:

- **Single-User:** Nur ein Nutzer nutzt Rechner/Programm zur Zeit
- **Multi-User:** *Quasi-gleichzeitige* Nutzung durch mehrere Nutzer
- **Remote-User:** Nutzer arbeitet transparent auf entferntem Rechner

**Dateisystem:** ○ Logische, i.a. hierarchische Verwaltungsstruktur für Dateien

- Rechteverwaltung („Welcher Nutzer darf was?“)
- Datensicherheit z.B. bei Systemabstürzen

**Multi-Tasking:** Verschiedene Programme laufen *quasi-gleichzeitig* ab

- BS teilt Programmen Ressourcen mehrerer Zeitscheiben zu
- BS lagert Programme mit Daten in CPU & Hauptspeicher ein bzw. aus

**Multiprocessing:** Echte Parallelverarbeitung mit mehreren Prozessoren

## 3.4 Betriebssysteme (BS) ...

---

### MS Windows

**1990:** Windows 3.0; erste multitasking-fähige Version

**bis 2001:** Windows 95, Windows NT, Windows 98, Windows 2000

- zunehmende Stabilität und Hardwareunterstützung
- Client-Server-Anwendungen

**Windows XP:** Ansatzweise Multi-User und Remote-User

**Stärken und Schwächen:** (unvollständig)

- geringe Einstiegshürde für Anfänger, stark im Clientbereich
- sehr gute Hardwareunterstützung, gute Systemstabilität
- viel Anwendersoftware, meistens allerdings kostenpflichtig



### Unix/Linux

**1970er:** Entwicklung von Unix als multi-tasking, multi-user BS auf Basis von C

**1980er:** ○ Entwicklung von X-Windows als Graphische Benutzeroberfläche (GUI) für Netzwerk-User

- Integration von Netzwerkdiensten (TCP/IP)

**1991:** Linus Torwald arbeitet an freiem Betriebssystem für PCs (Linux)

**Stärken und Schwächen:** (unvollständig)

- früher höhere, heute auch geringe Einstiegshürde, stark im Serverbereich
- sehr gute Systemstabilität, zeitlich verzögerte Hardwareunterstützung
- viele freie Anwendersoftware
- viele Distributions-Derivate (SuSE, RedHat, Debian, ...)