

Einführung in TI-1 für die Informationswirte

Einführung in die Technische Informatik I für Studierende der Informationswirtschaft

- Mittwoch 20. April 2005 (morgen)
- 17:30 – 19:00 Uhr
- HS -102 in Informatikgebäude am Fasanengarten



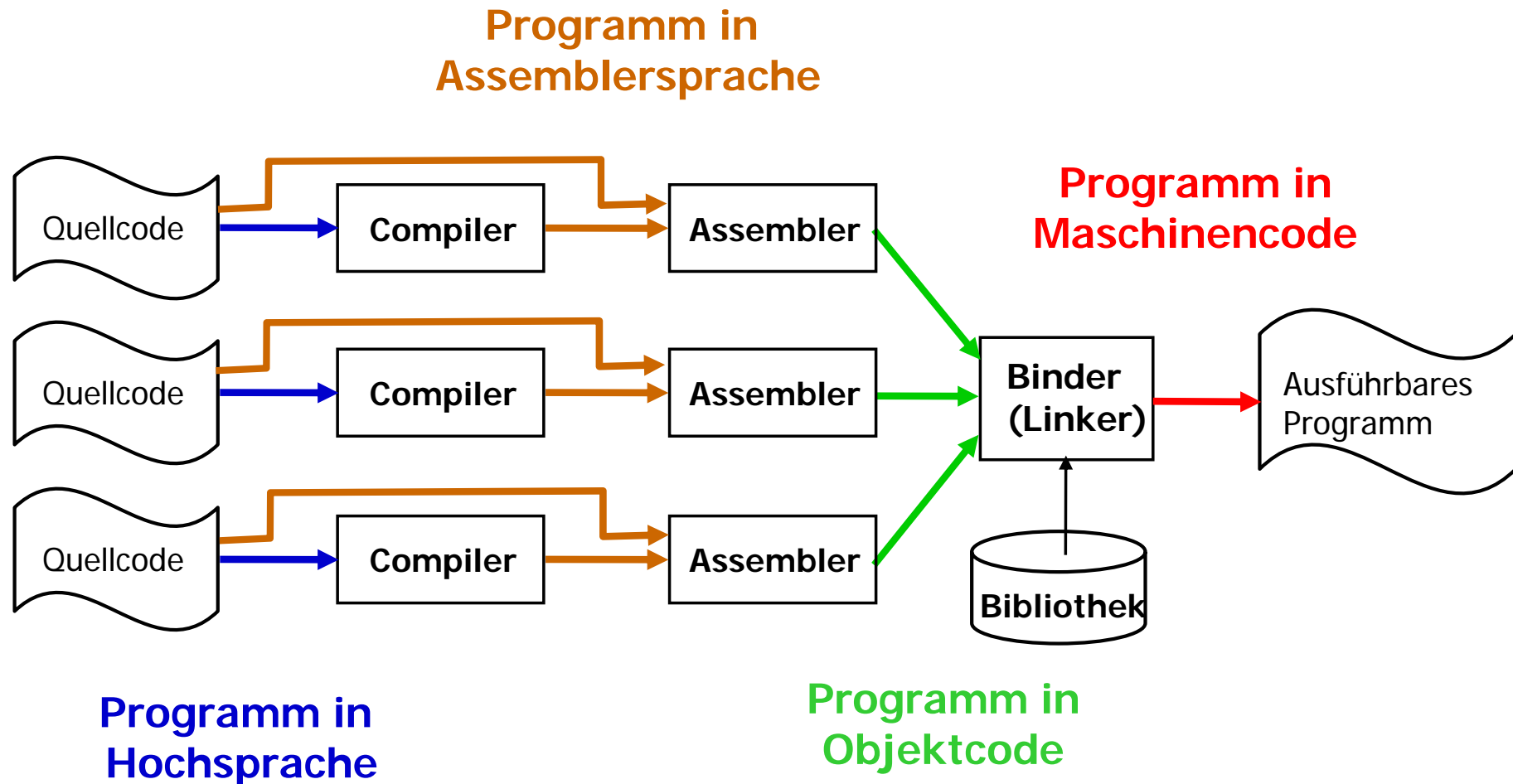
Kapitel 2

Anforderungen höherer Programmiersprachen: Die Programmiersprache c

- Vom Quellcode zum ausführbaren Programm
- Die Entwicklungsgeschichte von C
- Grundlagen: Datentypen, Operatoren, Ausdrücke
- Kontrollstrukturen
- Funktionen und Programmstruktur
- Zeiger und Vektoren
- ...



2.1 Vom Quellcode zum ausführbaren Programm



2.6 Zeiger und Vektoren

- ❑ Mächtigstes Werkzeug der Sprache C zur maschinennahen Programmierung
- ❑ Effiziente Programmierung durch Vermeidung von Kopieroperationen
- ❑ Pointerarithmetik → Direkter Zugriff auf Elemente strukturierter Daten
- ❑ Fortgeschrittene Verwendung von Pointern kann zu kryptischem Code und Unleserlichkeit führen ☹
- ❑ Macht das Programmieren mit C fehleranfällig.



2.6 Zeiger und Vektoren

- Ein **Zeiger** „**pointer**“ enthält eine Adresse, die auf Daten verweist.

```
int *p;          /* *ptr ist vom Typ „int“  
                ptr ist vom Typ „Zeiger auf int“ */  
  
int *q;  
int a = 3;  
int b;  
  
p = &a;         /* der unäre Operator „&“ liefert die  
                Adresse von */  
  
b = *p + 1;     /* der unäre Operator „*“ liefert den  
                Wert, auf den p zeigt */  
  
q = (int*) 0x8010 /* Typumwandlung einer Adresse in  
                Hexadezimalschreibweise auf  
                „Zeiger auf int“ */
```



2.6 Zeiger und Vektoren

```
int *p;  
int *q;  
int a = 3;  
int b;  
  
p = &a;  
b = *p + 1;  
q = (int*) 0x8010
```

	Adresse	Inhalt
p	...	0x8004
a	0x8004	3
b	...	4
q	...	0x8010
	0x8010	



2.6 Zeiger und Vektoren

```
int n[] = {1, 2, 3};  
char c[] = {'a', 'b', 'c'};  
char s[] = "de";  
s[1] = 'f';
```

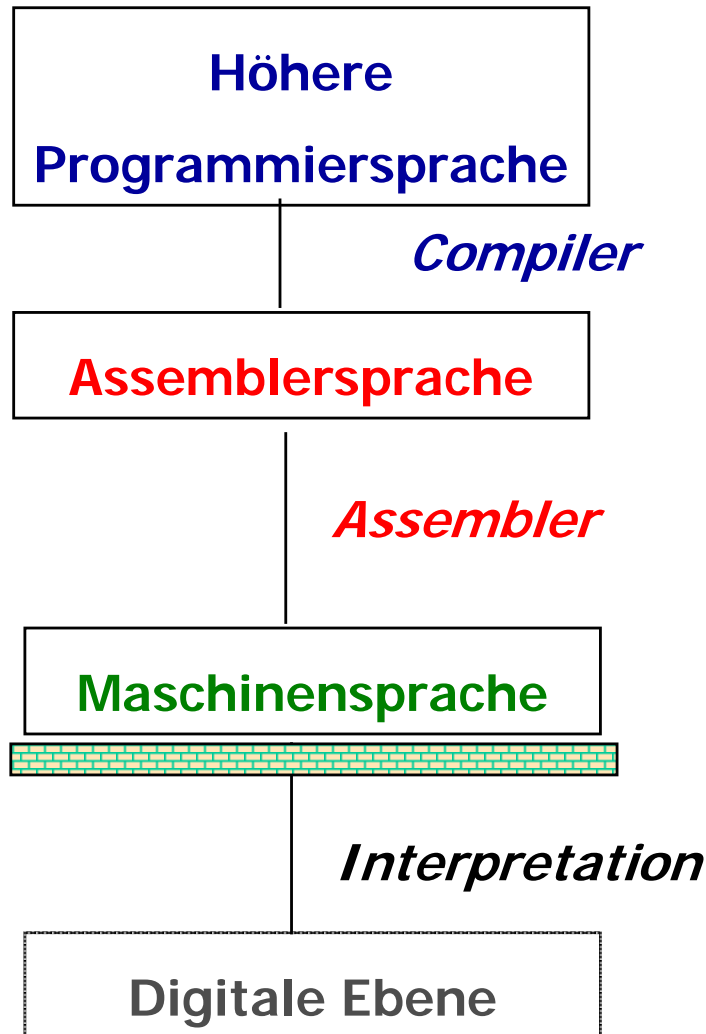
	Adresse	Inhalt
n[0]	0x8004	1
n[1]	0x8008	2
n[2]	0x800C	3

	Adresse	Inhalt
c[0]	0x8014	'a'
c[1]	0x8018	'b'
c[2]	0x801C	'c'

	Adresse	Inhalt
s[0]	0x8024	'd'
s[1]	0x8028	'f'
s[2]	0x802C	'\0'



Hierarchie



```
temp = v[k];  
v[k] = v[k+1];  
v[k+1] = temp;
```

```
lw $15, 0($2)  
lw $16, 4($2)  
sw $16, 0($2)  
sw $15, 4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000  
1010 1111 0101 1000 0000 1001 1100 0110  
1100 0110 1010 1111 0101 1000 0000 1001  
0101 1000 0000 1001 1100 0110 1010 1111
```

```
ALUOP[0:3] ← InstReg[9:11] & MASK
```



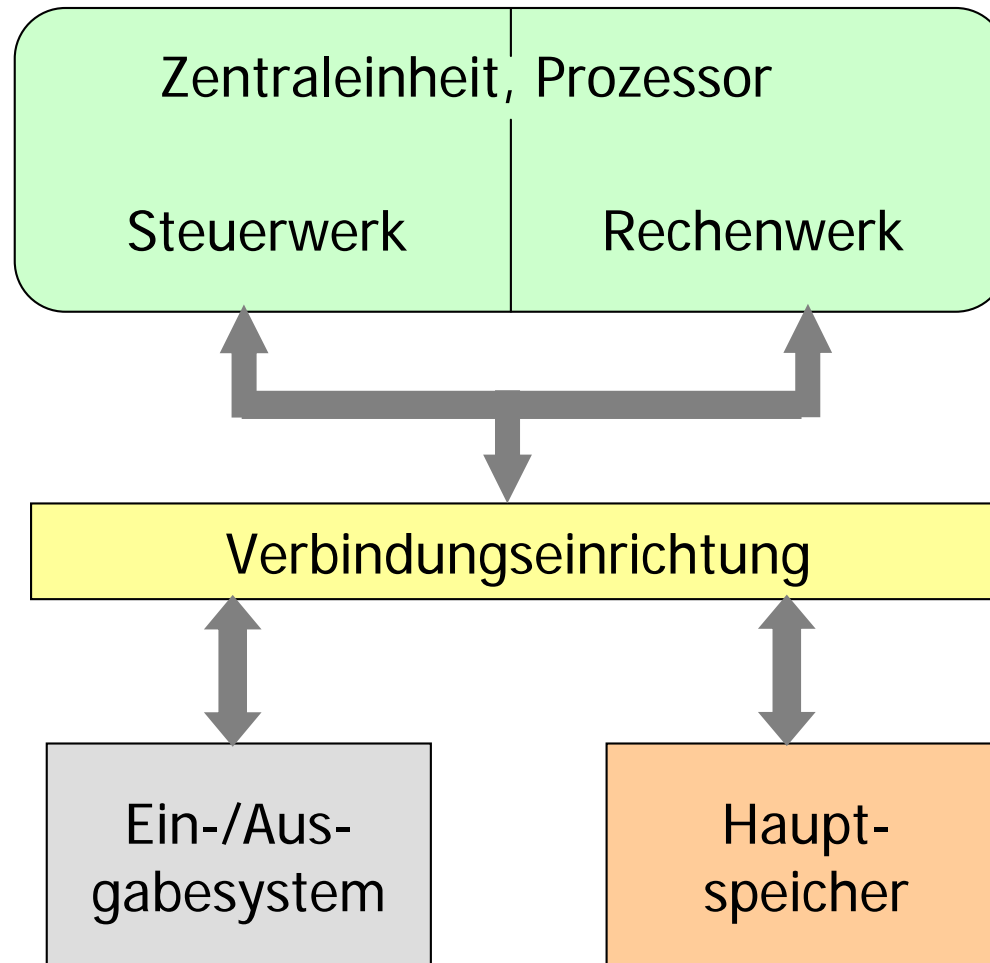
Kapitel 3

Ein grundlegendes Rechnermodell

- Organisationsprinzip des von Neumann Rechners
- Aufbau eines einfachen Mikroprozessors
 - Steuerwerk (Leitwerk)
 - Rechenwerk
 - Speicherwerk
 - Ein-Ausgabewerk
 - Verbindungsstrukturen
- Maschinenbefehlszyklus



3.1 Organisationsprinzip des von Neumann Rechners



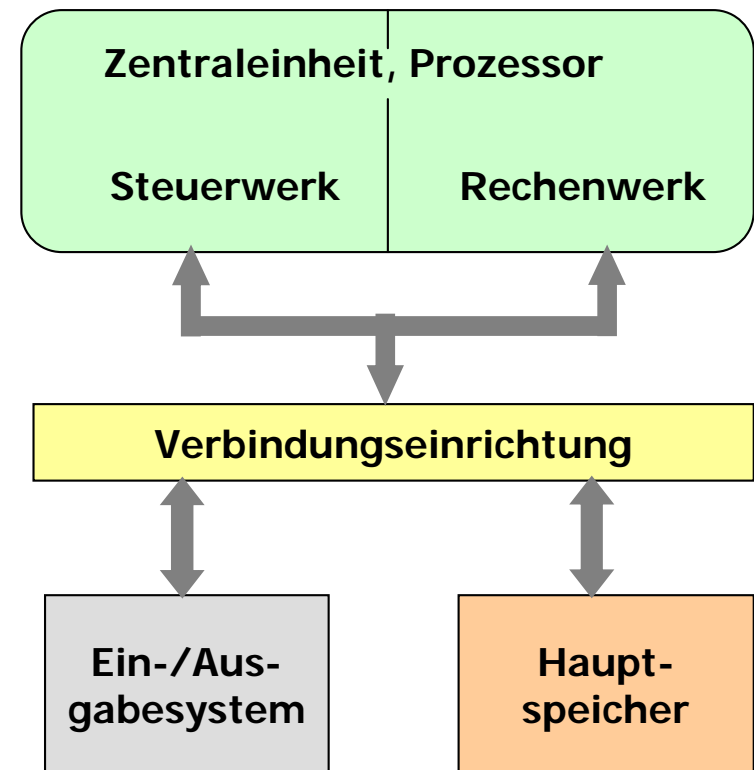
Komponenten des von Neumann Rechners

□ Zentraleinheit

(central processing unit, CPU, Prozessor)

Verarbeitet Daten gemäß eines Programms. Sie besteht aus Leitwerk und Rechenwerk:

- **Leitwerk** (Steuerwerk, control unit, CU)
 - Holt die Befehle eines Programms aus dem Speicher
 - entschlüsselt sie und
 - steuert ihre Ausführung in der verlangten Reihenfolge durch Steuer- und Synchronisier-Signale.

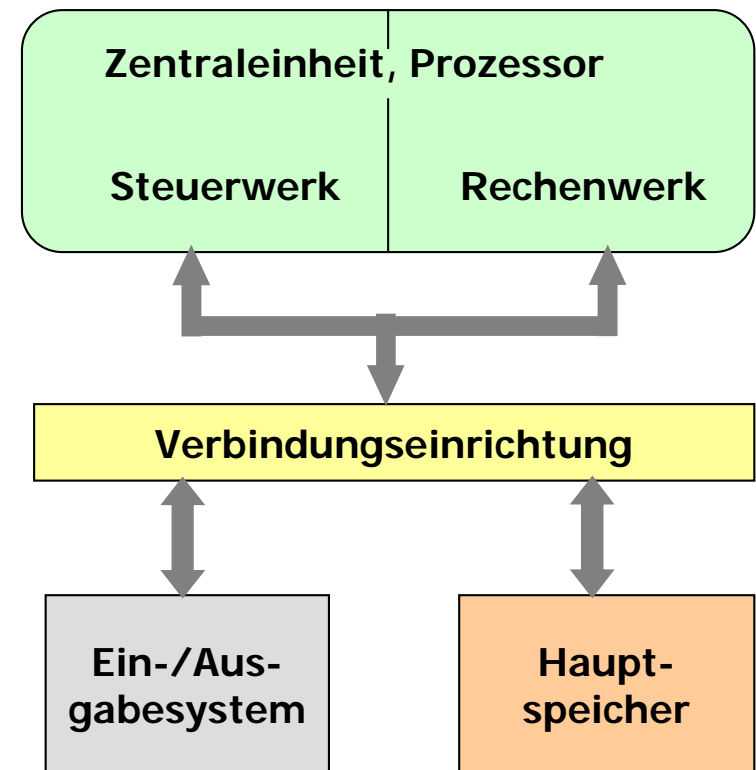


Komponenten des von Neumann Rechners

□ Zentraleinheit

Verarbeitet Daten gemäß eines Programms. Sie besteht aus Leitwerk und Rechenwerk:

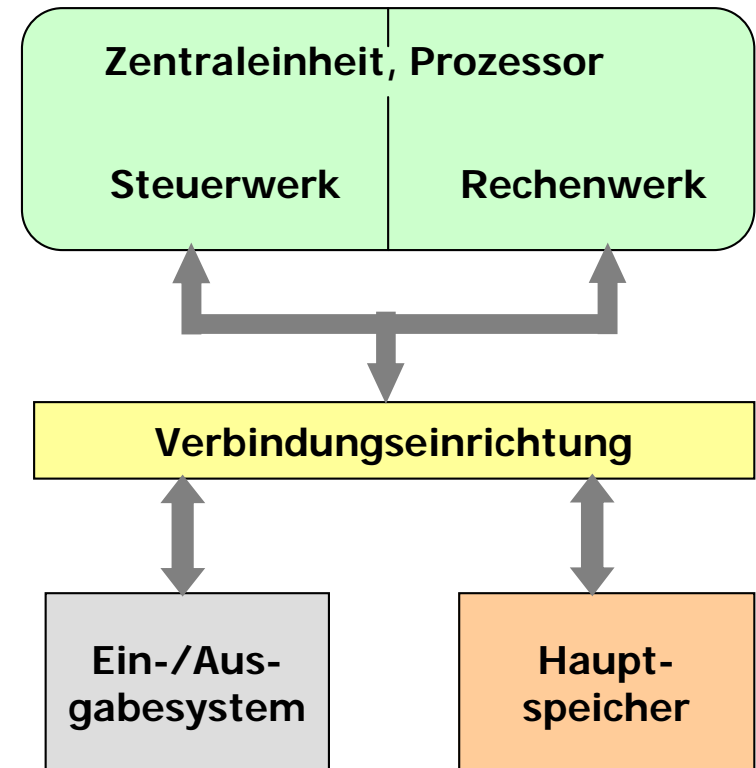
- **Rechenwerk (Operationswerk, Ausführungseinheit, ALU)**
 - Führt arithmetisch/logische Operationen aus.
 - Wird durch Steuersignale des Leitwerks beeinflusst und
 - liefert seinerseits Meldesignale an das Leitwerk zurück.



Komponenten des von Neumann Rechners

□ Hauptspeicher

- Jede Speicherzelle ist eindeutig durch ihre Nummer (Adresse) identifizierbar.
- Dort werden Programme und Daten aufbewahrt (von-Neumann-Konzept).
- Den einzelnen Speicherzellen ist nicht anzusehen, welchen Typ von Information sie enthält
- Alternativ: **Harvard-Architektur** mit getrenntem Programm- und Datenspeicher.
- Inhalt nach Abschaltung des Rechners flüchtig.



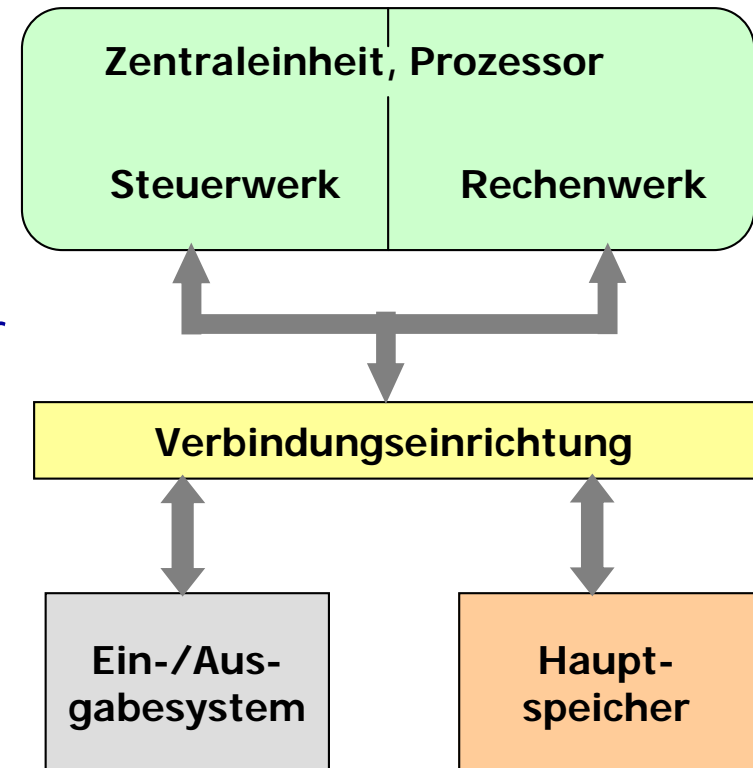
Komponenten des von Neumann Rechners

□ Verbindungsstruktur (BUS)

- **Adreßleitungen:** Leitungen, auf denen die Adressinformation transportiert wird (unidirektional).
- **Datenleitungen:** Transportieren Daten und Befehle von/zum Prozessor (bidirektional).
- **Steuerleitungen:** Geben Steuerinformationen von/zum Prozessor (uni- oder bidirektional).

□ Bus (Sammelschiene)

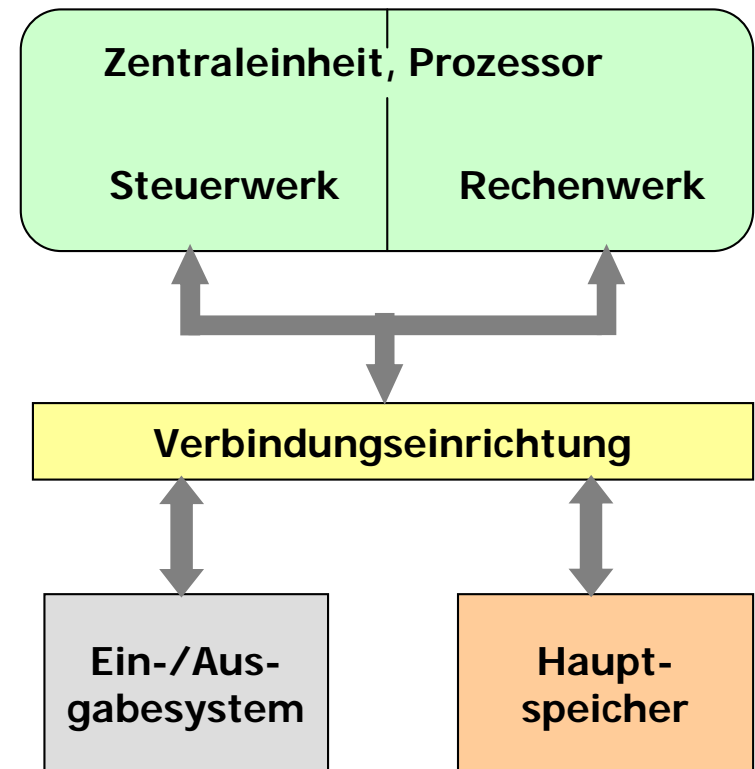
Systembus = Adreßbus + Datenbus + Steuerbus



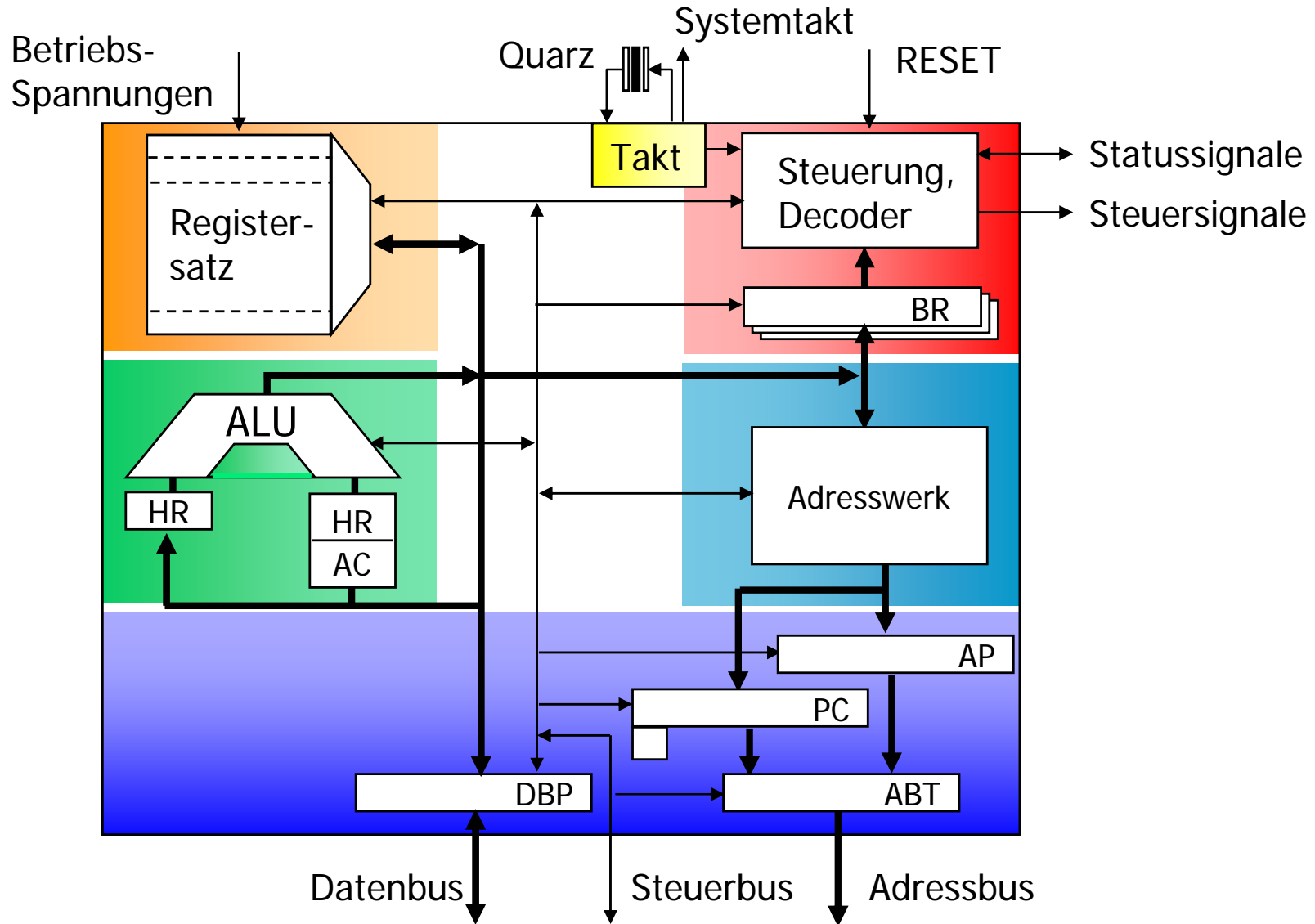
Komponenten des von Neumann Rechners

□ Ein-/Ausgabesystem (Peripheriegeräte)

- Geräte zur Eingabe von Daten und Programmen und zur Ausgabe der verarbeiteten Daten (Bildschirme, Drucker, Terminals, ...)
- Diese Geräte sind über Ein-/Ausgabe-Schnittstellen mit dem Rechner verbunden.
- Die Verbindung der Schnittstellen mit dem Prozessor (und zu den Peripheriegeräten) geschieht durch Adreß-, Daten- und Steuerleitungen.



3.2 Aufbau eines einfachen μP



Aufbau eines einfachen μP

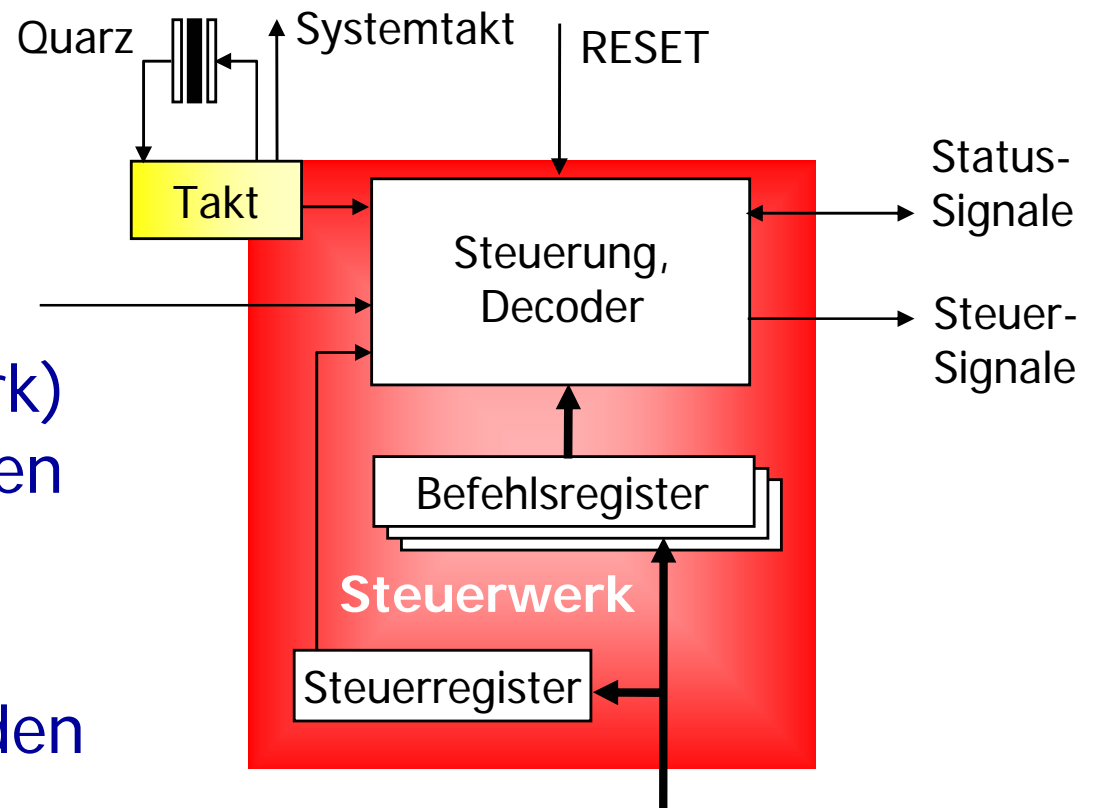
- ❑ **Steuerwerk**
- ❑ Rechenwerk
- ❑ Registersatz
- ❑ Adresswerk
- ❑ Systembusschnittstelle
- ❑ Interne Busse



3.2.1 Steuerwerk

Steuert die Systemkomponenten

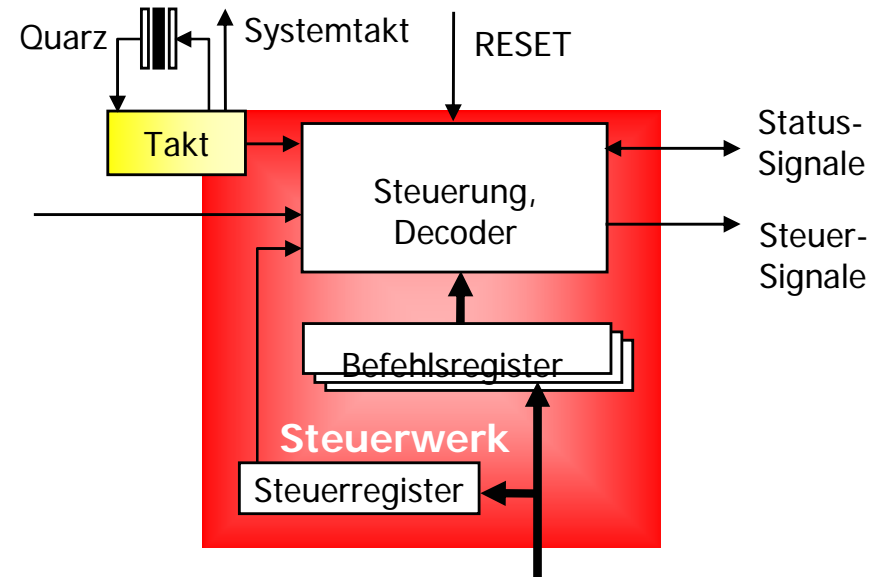
- **Befehlsregister** enthält den gerade ausgeführten Befehl
- **Dekoder** (mikro-programmiertes Schaltwerk) wird von den Statussignalen beeinflusst und erzeugt die Steuersignale
- **Taktgenerator** erzeugt den vom externen Quarz festgelegten Systemtakt



3.2.1 Steuerwerk

Synchrones Schaltwerk

Meist liegt ein sog. **dynamisches Schaltwerk** vor, d. h. die Zustandsinformation ist nicht in Flipflops, sondern in Kondensatoren gespeichert



➔ Mindesttaktfrequenz ist erforderlich

Unterhalb dieser Taktfrequenz gehen die Inhalte durch Leckströme bereits vor dem nächsten Taktzyklus verloren.

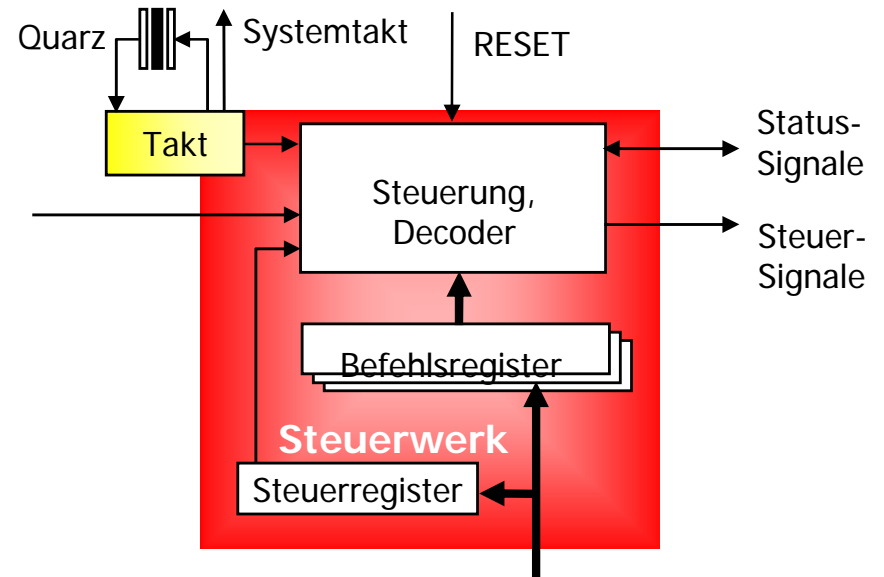
Taktgenerator ist bei modernen Mikroprozessoren on Chip (mit externem Quarz verbunden)



Taktgenerator

Aufgaben des Taktgenerators

- Taktfrequenz herstellen
- Erzeugung eines mit dem Prozessortakt synchronisierten Rücksetzsignals



Beim Rücksetzen durchläuft das Steuerwerk eine Initialisierungsroutine

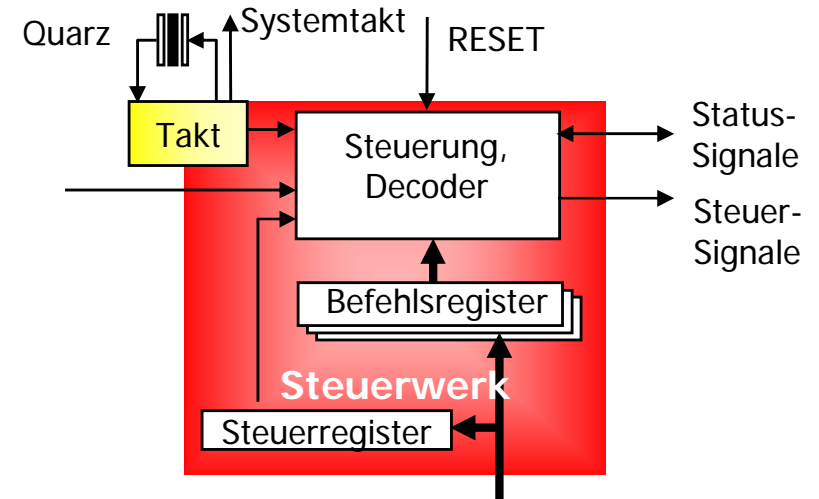
Diese wird bei vielen Mikroprozessoren ausgeführt, während das Rücksetzsignal aktiv ist

Deshalb muss das Rücksetzsignal genauen zeitlichen Spezifikationen genügen.

3.2.1 Steuerwerk

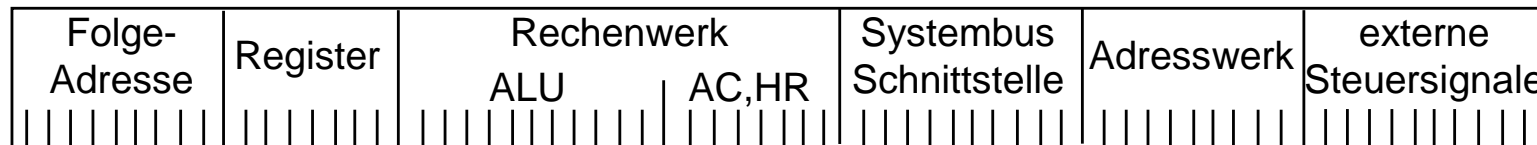
Mikroprogrammsteuerwerk

Im Festwertspeicher liegt für jeden Befehl ein Mikroprogramm



Mikroprogramm ≡ Folge von Mikrobefehlen

Aufbau eines Mikrobefehls:



Einzelne Bits eines Mikrobefehls ≡ Mikrooperationen ≡ Auswahl- und Freigabesignale für die benötigten Komponenten



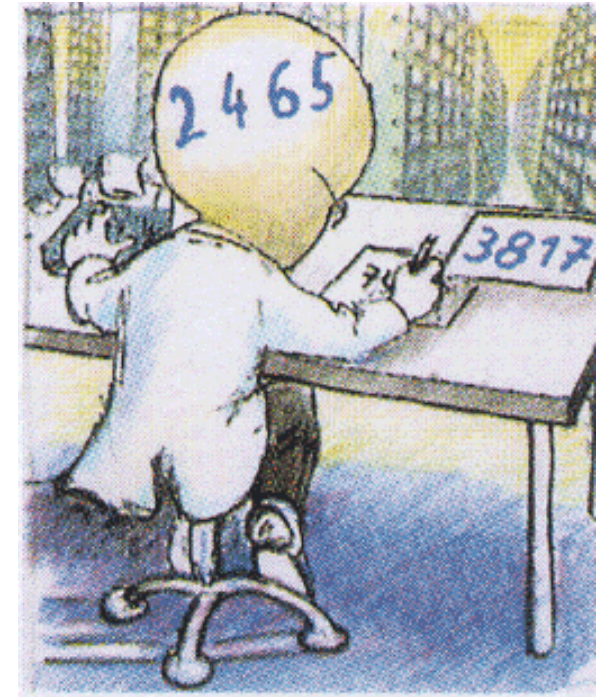
Phasen der Befehlsausführung



Holphase



Decodierphase



Ausführungsphase



Phasen der Befehlsausführung

Holphase:

der nächste Befehl in das Befehlsregister laden

Decodierphase:

der Befehlsdecoder ermittelt die Startadresse des Mikroprogramms, welches den Befehl ausführt

Ausführungsphase:

das Mikroprogramm steuert die Befehlsausführung, indem es entsprechende Signalfolgen an die anderen Prozessorkomponenten übermittelt und Meldesignale auswertet



3.2.1 Steuerwerk

Das Befehlsregister besteht aus mehreren Registern:

Gründe:

- unterschiedlich lange Befehlsformate:
verschiedene Befehle sind unterschiedlich lang
(1-Wort-Befehle, 2-Wort-Befehle, 3-Wort-Befehle, ...)
- Vorabladen von Befehlen (*Opcode-Prefetching*):
zur Steigerung der Verarbeitungsgeschwindigkeit
werden bereits mehrere folgende Befehle in das
Befehlsregister geladen, während der aktuelle Befehl
gerade dekodiert wird

Opcode prefetch queue, Warteschlange, Pipelining



3.2.1 Steuerwerk

- Für jeden Befehl liegt ein Mikroprogramm im Festwertspeicher des Steuerwerks vor
- Mikroprogramme können vom Benutzer nicht verändert werden
 - ➔ Das Steuerwerk ist mikroprogrammiert

Andere Variante:

Steuerwerk als festverdrahtetes Schaltwerk (RISC-Prozessoren)



Ein-/Ausgabesignale des Steuerwerks

- **Systembus-Zuteilung:**
 - Busanforderung
 - Busfreigabe
 - Busübernahme-Bestätigung
- **Unterbrechungsanforderungen**
 - Interrupt-Eingänge IRQ, NMI, HALT
- **Fehlermeldungen (Bus error)**
- **Statusinformationen**
- **Kommunikation mit Coprozessor**
- **Systembus-Steuersignale**

