

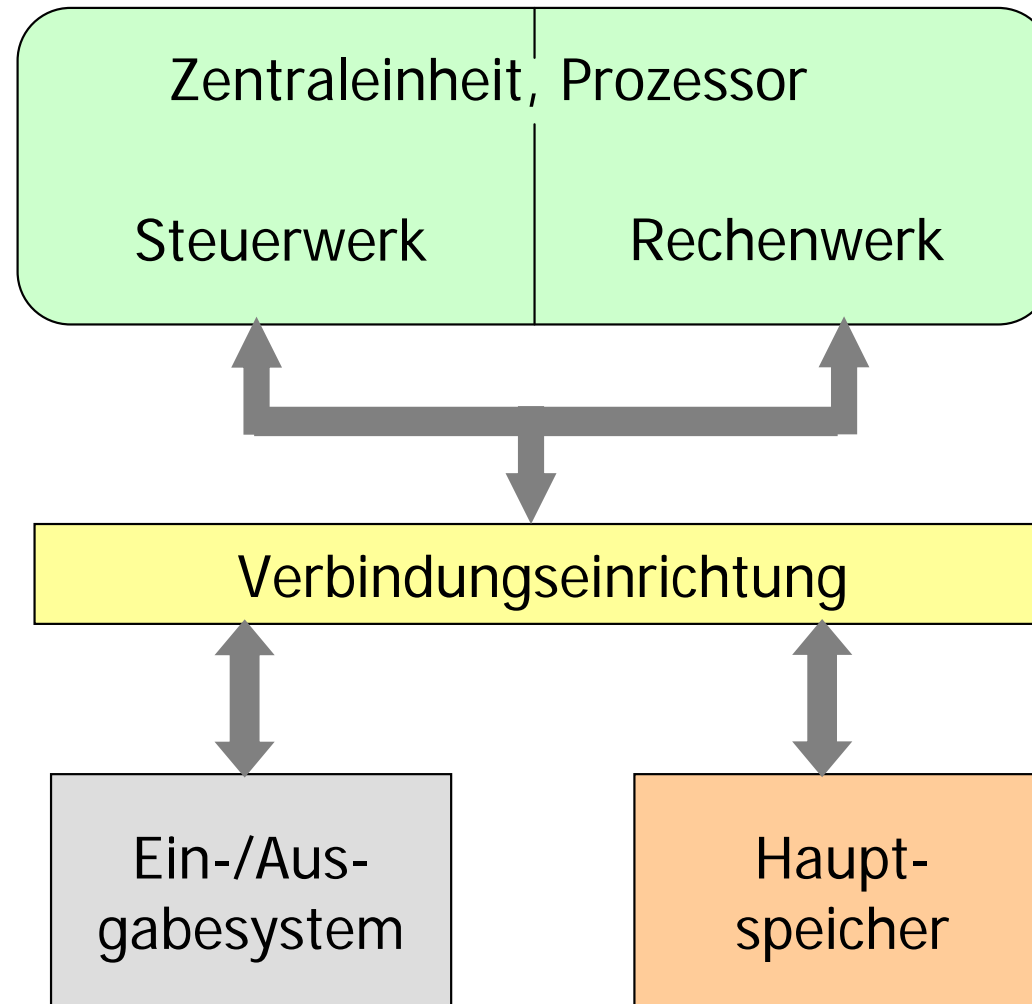
# Kapitel 3

---

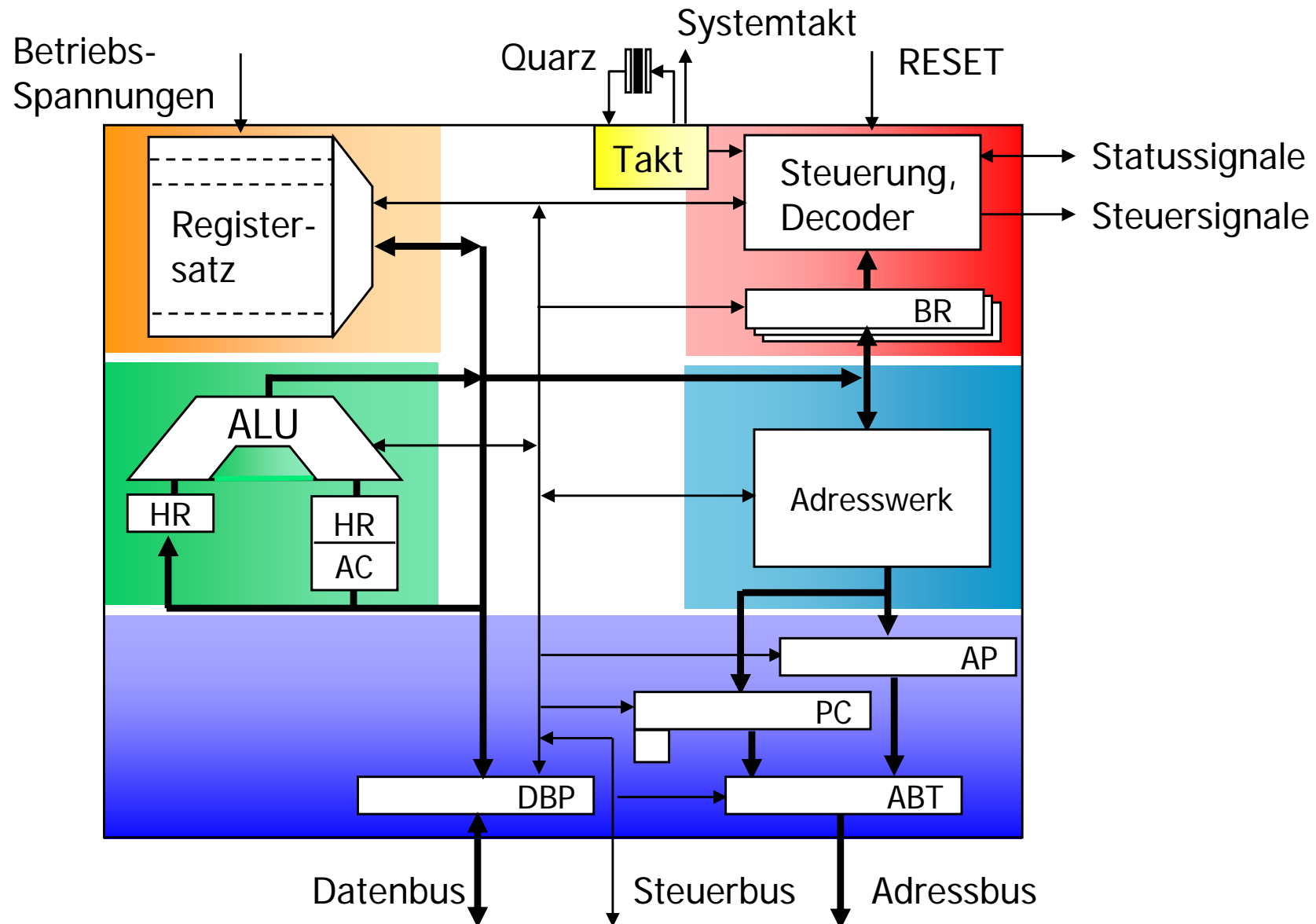
## Ein grundlegendes Rechnermodell

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

# Wdh. Von Neumann Rechner



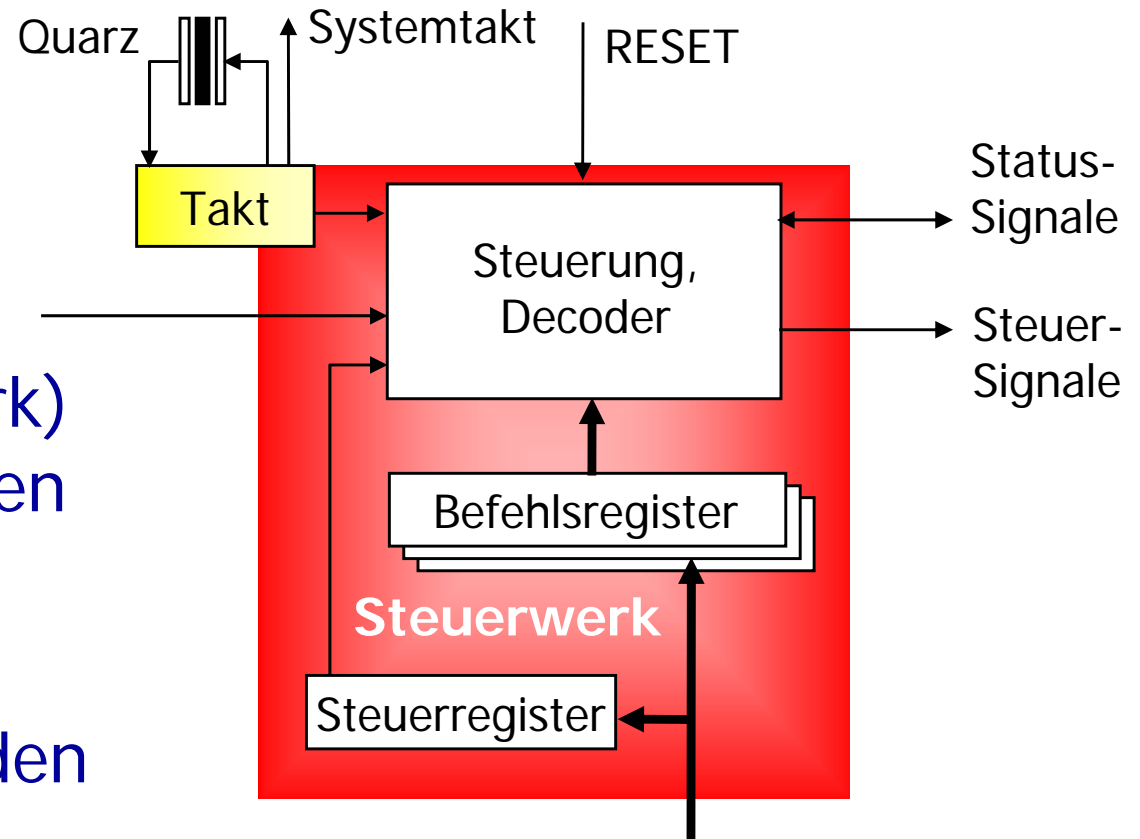
## 3.2 Aufbau eines einfachen $\mu P$



# 3.2.1 Steuerwerk

Steuert die Systemkomponenten

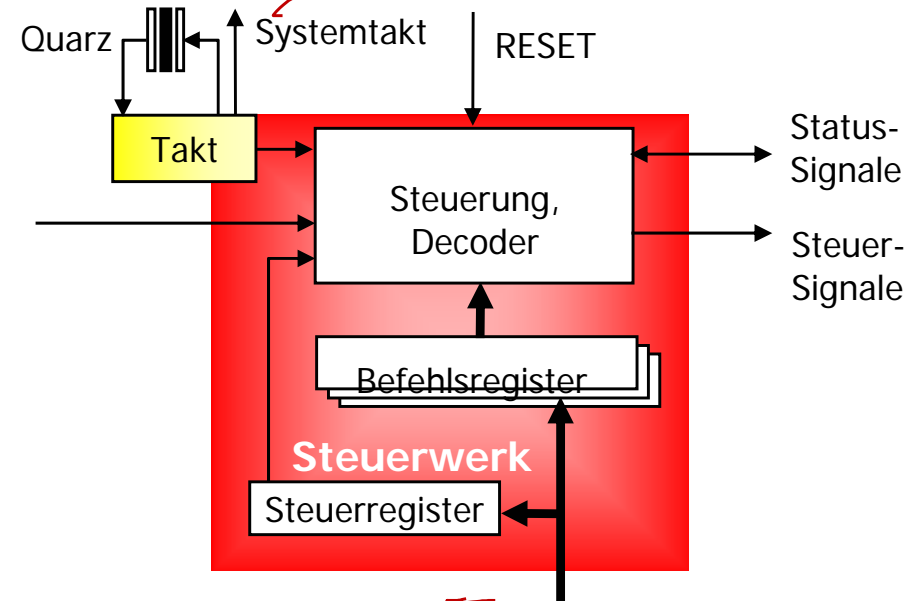
- **Befehlsregister**  
enthält den gerade ausgeführten Befehl
- **Dekoder** (mikro-programmiertes Schaltwerk)  
wird von den Statussignalen beeinflusst und erzeugt die Steuersignale
- **Taktgenerator** erzeugt den vom externen Quarz festgelegten Systemtakt



## 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



→ **Mindest-Refresh-Taktfrequenz ist erforderlich**

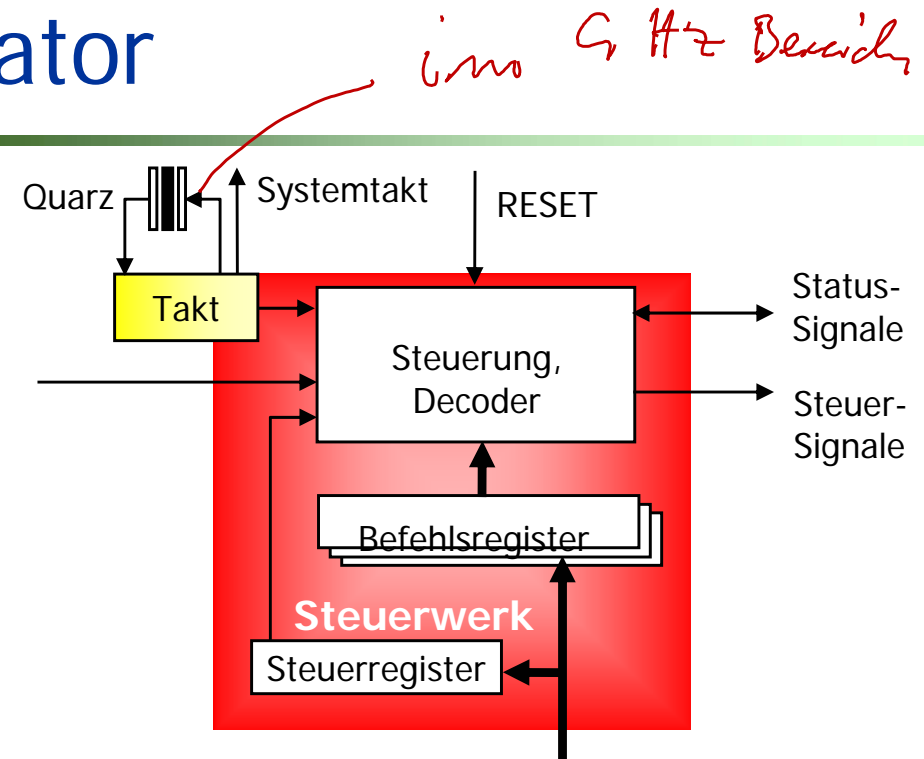
Unterhalb dieser Refresh-Taktfrequenz gehen die Inhalte durch Leckströme bereits vor dem nächsten Taktzyklus verloren.

**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

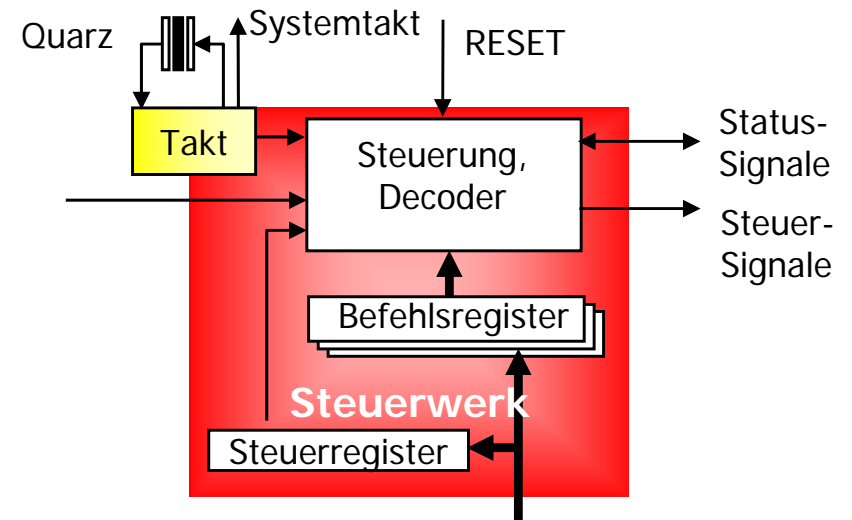
Diese wird bei vielen Mikroprozessoren ausgeführt, während das Rücksetzsignal aktiv ist

Deshalb muss das Rücksetzsignal genauen zeitlichen Spezifikationen genügen.

# 3.2.1 Steuerwerk

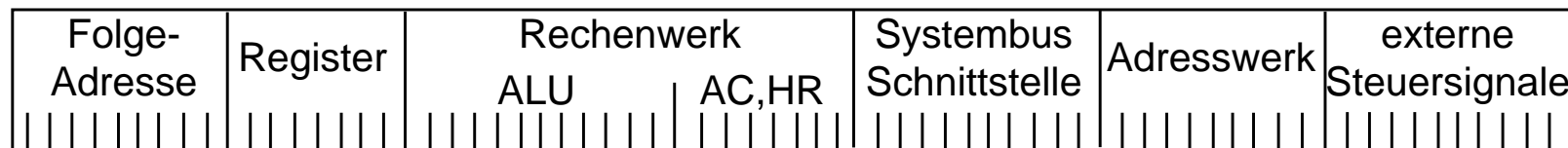
## Mikroprogrammsteuerwerk

Im Festwertspeicher liegt für jeden Befehl ein Mikroprogramm



**Mikroprogramm  $\equiv$  Folge von Mikrobefehlen**

**Aufbau eines Mikrobefehls:**



Einzelne Bits eines Mikrobefehls  $\equiv$  Mikrooperationen  $\equiv$  Auswahl- und Freigabesignale für die benötigten Komponenten

*Viele Mikro-befehle  $\Rightarrow$  Maschinen-befehle*

# Phasen der Befehlsausführung

Phase A  
B  
C  
↓  
A

0  
B  
A

B  
A



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

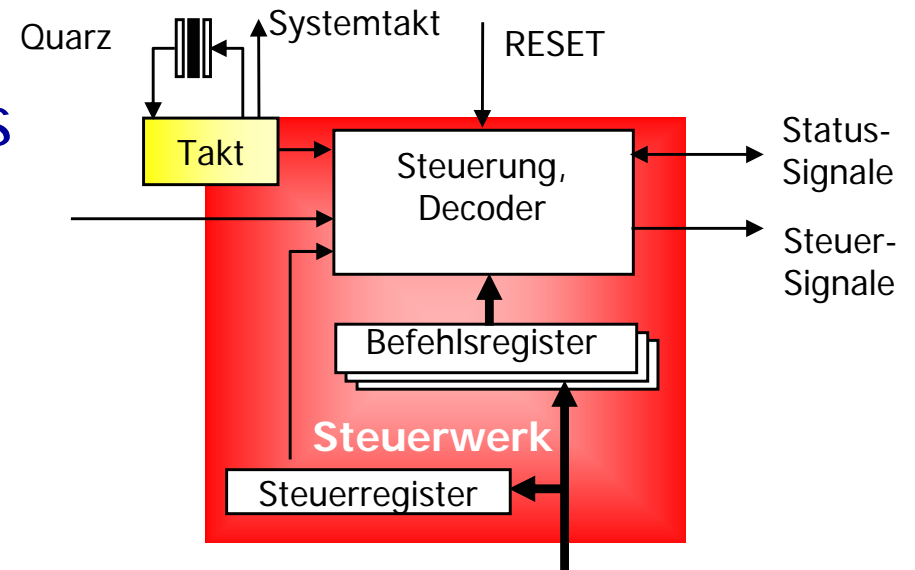
---

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

# Das Steuerregister

Mit Hilfe des **Steuerregisters** kann die aktuelle Arbeitsweise des Steuerwerks beeinflußt werden

Die Bedeutung der Bits des Steuerregisters hängen vom jeweiligen Prozessor ab



Beispiele für die Bedeutung von Bits des Steuerregisters:

- **Interrupt enable Bit**  
bestimmt, ob auf eine Unterbrechungsanforderung am INT-Eingang reagiert wird

# Das Steuerregister

## ➤ User/System Bit

bestimmt ob der Prozessor im User-Modus (nur beschränkter Teil des Befehlsvorrats nutzbar) oder im Systemmodus (alle Befehle verfügbar, i. A. für das Betriebssystem reserviert) arbeitet

## ➤ Trace Bit

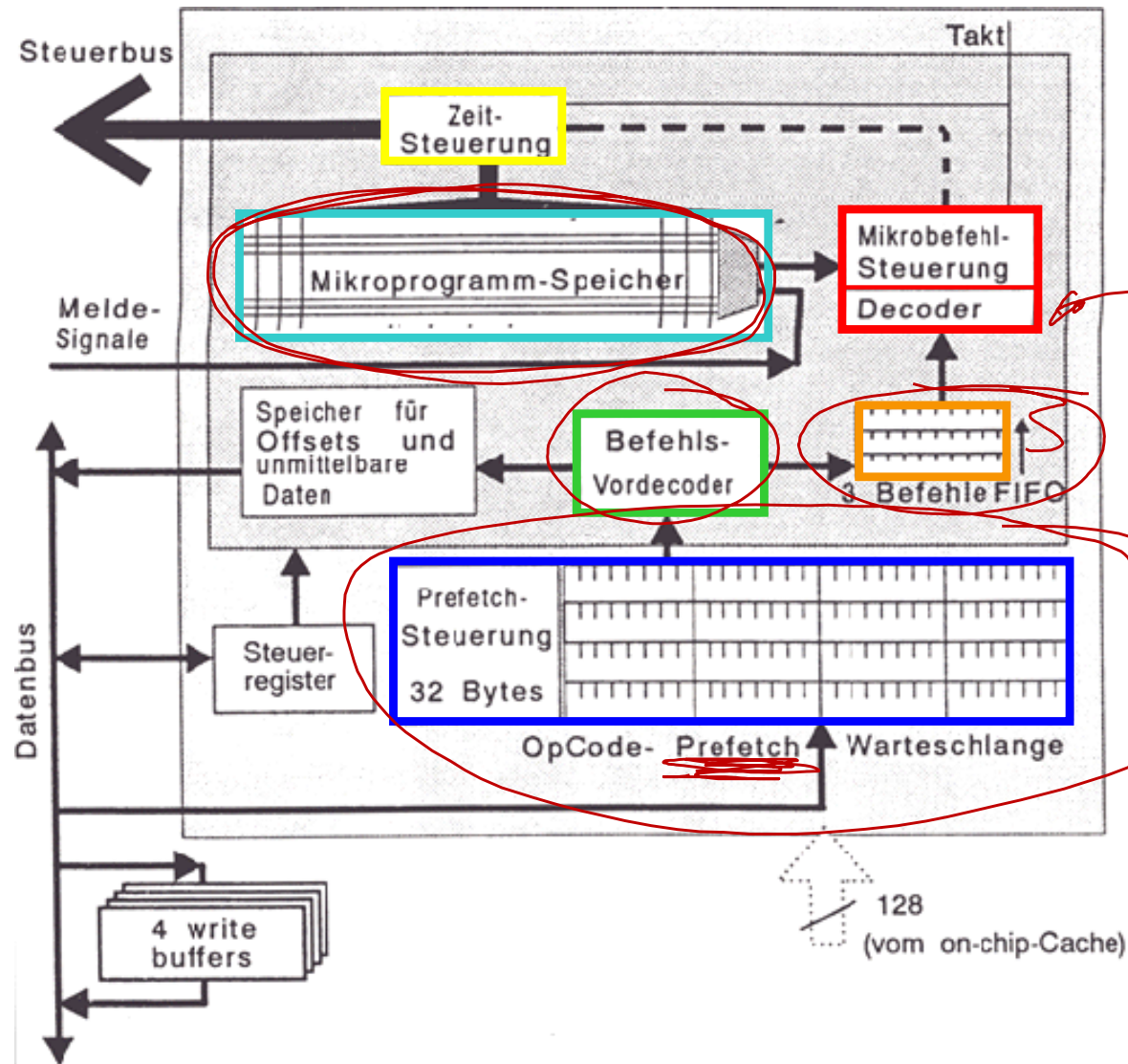
erlaubt Befehlsabarbeitung im Einzelschritt (Single Step Mode), d. h. nach jeder Befehlsausführung wird eine Unterbrechungsroutine gestartet ➡ Debugging

## ➤ Decimal Bit

entscheidet, ob Dual oder BCD gerechnet wird

*Binary Coded Decimal*

# Steuerwerk: Fallstudie (1)



*eigentlich  
Decoder*

# Steuerwerk: Fallstudie (2)

---

## **Befehlsregister-Block: Prefetch-Queue**

- FIFO-Speicher mit 32 Byte
- Prefetch Steuerung sorgt für weitestmögliche Füllung
- Füllung kann meist aus dem on-chip Cache über einen 128 Bit breiten Datenpfad erfolgen
- Muss die Füllung jedoch aus dem Arbeitsspeicher erfolgen (Cache Miss), haben diese Zugriffe höchste Priorität
- Eventuelle gleichzeitig auf den Bus auszugebende Daten werden in Schreibpuffern (write buffers) zwischengespeichert

# Steuerwerk: Fallstudie (3)

---

## Befehls-Vordecoder:

- Aus der Prefetch-Queue gelangen die Befehle in den Befehls-Vordecoder
- Vorbereitung der Befehle
- direkt im Befehl angegebene Operanden (unmittelbare Daten) sowie Adressdistanzen (Offsets) werden abgezweigt und separat gespeichert

# Steuerwerk: Fallstudie (4)

---

## Befehls-FIFO:

- vordekodierte Befehle gelangen in den Befehls-FIFO
- Platz für 3 Befehle

## Befehls-Decoder:

- entnimmt den obersten vordekodierten Befehl aus dem Befehls-FIFO
- ermittelt die Startadresse des zugehörigen Mikroprogramms

# Steuerwerk: Fallstudie (4)

---

## **Mikrobefehls-Steuerung, Mikrobefehls-Speicher:**

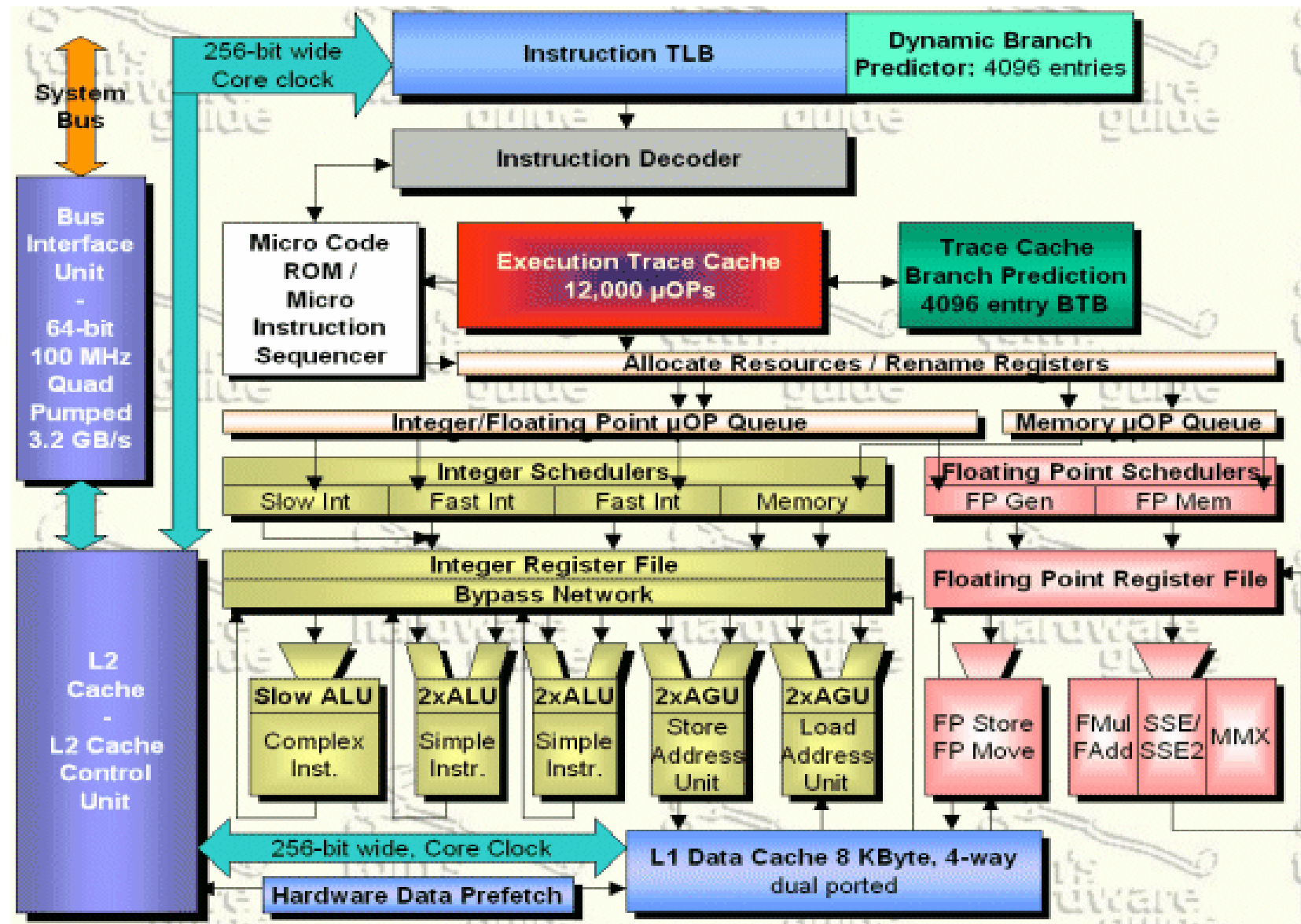
synchrones mikroprogrammiertes Schaltwerk, erzeugt die Steuersignale, interpretiert die Meldesignale

## **Zeit-Steuerung:**

synchronisiert die erzeugten Steuersignale mit dem Systemtakt



# Pentium 4



# Steuerwerk: Zusammenfassung

---

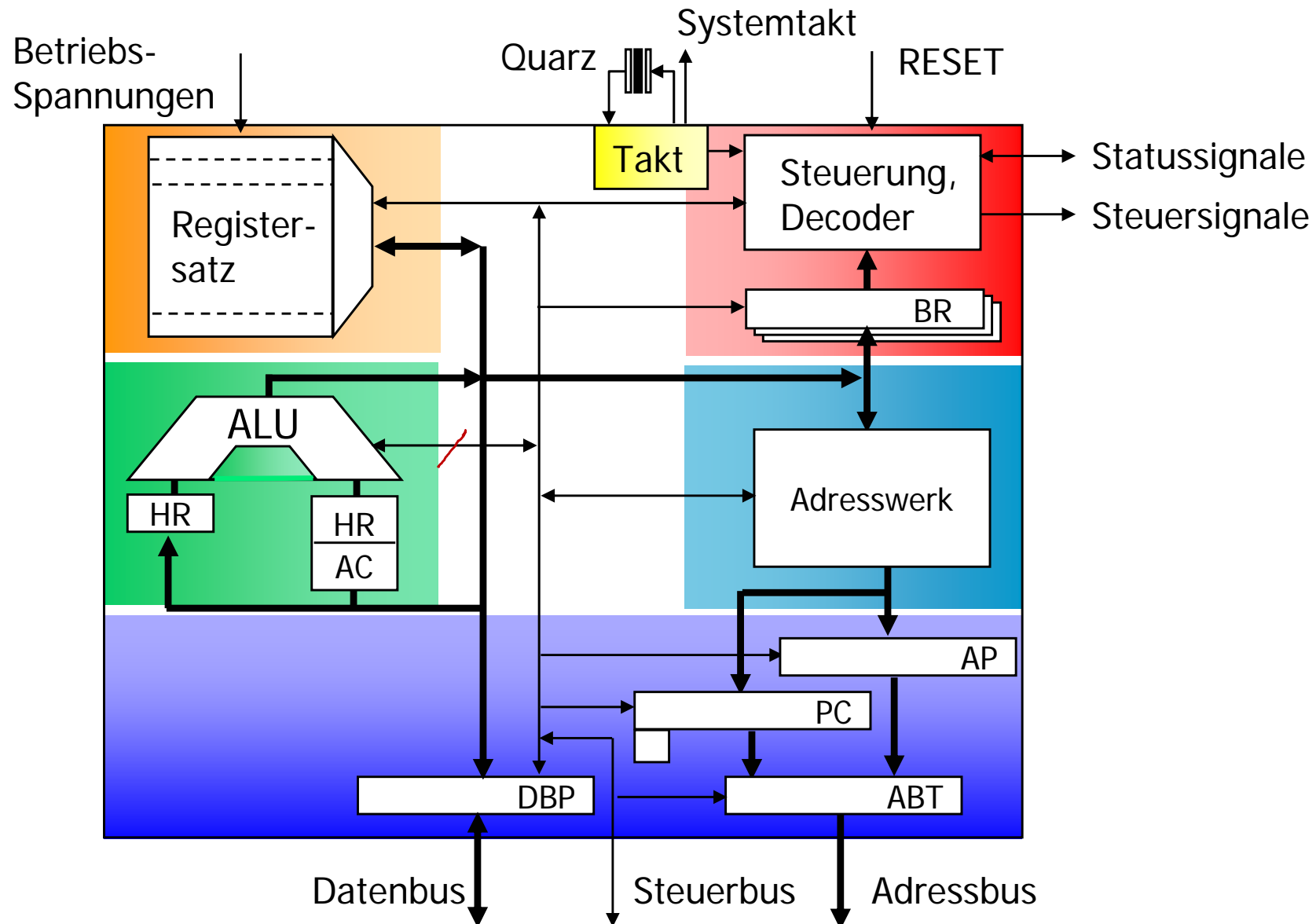
- ❑ Das Steuerwerk interpretiert die Maschinenbefehle und setzt sie unter Berücksichtigung der Statusinformation in Steuerkommandos für andere Komponenten um.
- ❑ Das Befehlsregister enthält den als nächstes auszuführenden Maschinenbefehl, sofern der vergangene Befehl kein Sprungbefehl ist.
- ❑ Das Steuerregister beeinflusst die aktuelle Arbeitsweise des Steuerwerks

# Aufbau eines einfachen $\mu P$

---

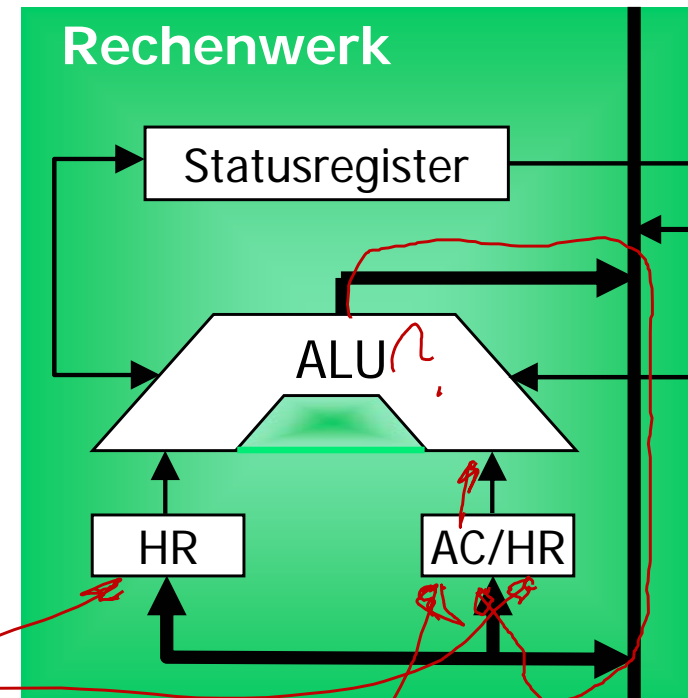
- ❑ Steuerwerk
- ❑ **Rechenwerk**
- ❑ Registersatz
- ❑ Adresswerk
- ❑ Systembusschnittstelle
- ❑ Interne Busse

## 3.2 Aufbau eines einfachen $\mu P$



## 3.2.2 Rechenwerk

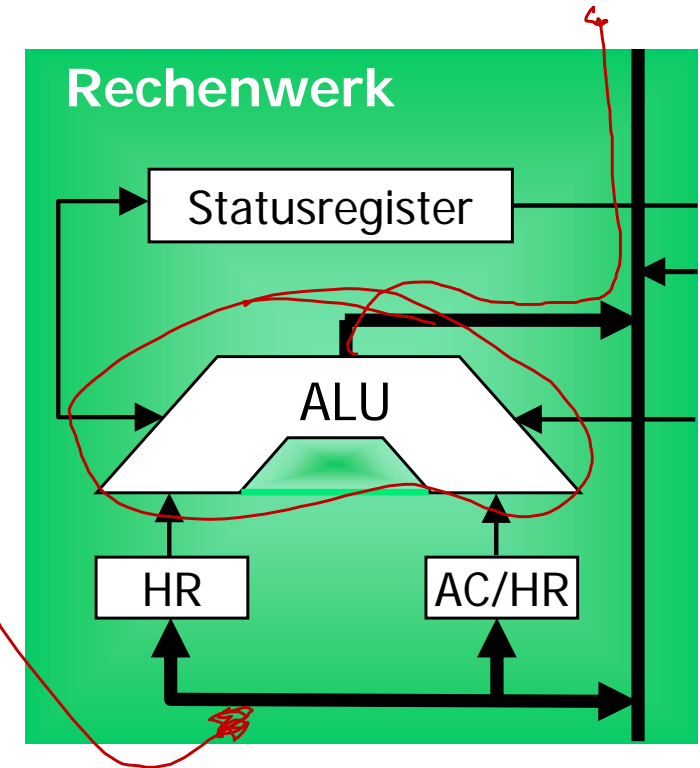
- Führt die vom Steuerwerk verlangten logischen und arithmetischen Operationen aus
- **Statusregister** informiert das Steuerwerk über den Ablauf des Ergebnisses (z. B. Carry, Overflow, Zero, Sign, ...)
- Zum Zwischenspeichern von Operanden und Ergebnissen sind **Hilfregister** und **Akkumulatoren** vorhanden



## 3.2.2 Rechenwerk



- 2 Eingangsbusse (2 Operanden) und 1 Ausgangsbuss (Ergebnis).
- Die ALU ist ein reines Schaltnetz
- Hilfsregister vor der ALU sind zur Zwischenspeicherung von Operanden
- Ergebnisse werden entweder in Prozessor-Registern gespeichert, auf die Eingangsregister der ALU zurückgeführt oder über den externen Datenbus an andere Systemkomponenten übertragen
- Eingänge zur Steuerung der ALU-Operationen



## 3.2.2 Rechenwerk

---

- Melde-Ausgänge für den Status der abgelaufenen Operation (Carry, Overflow, Zero, Sign, ..., usw.)  
Status wird im Status-Register zwischengespeichert

- **Akkumulator (spezielles Register)**

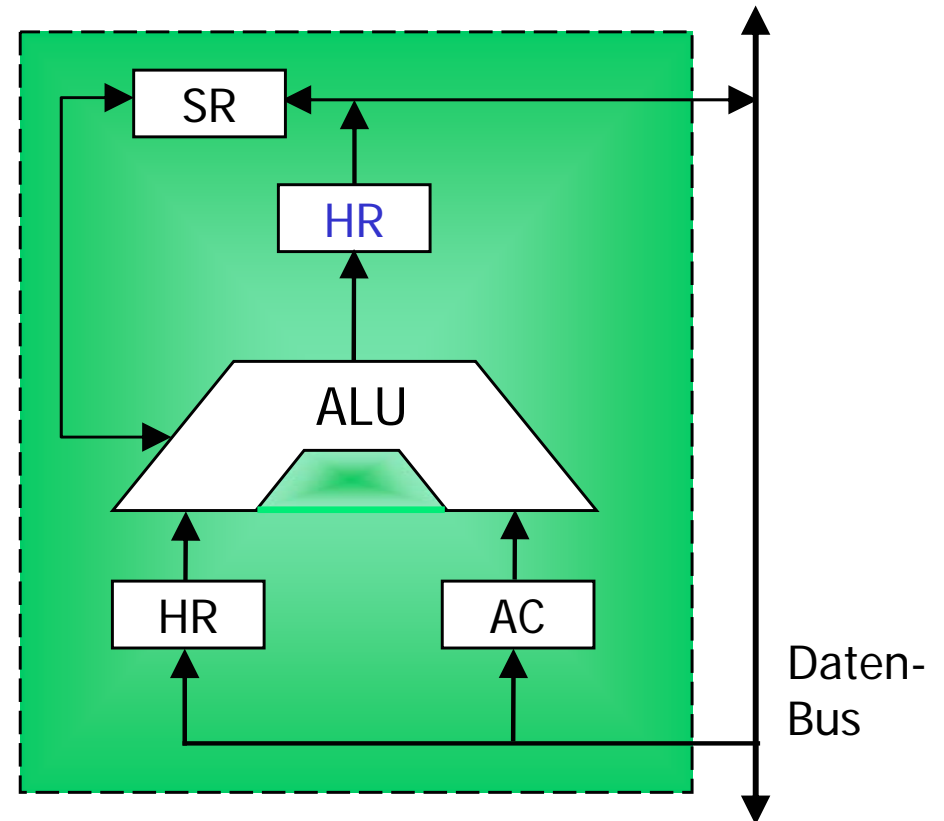
Bei 8 Bit Mikroprozessoren war das einzige interne Register, das direkt Ergebnisse der ALU aufnehmen kann

- ➡ alle ALU Ergebnisse werden dort abgelegt
- ➡ Akkumulator "sammelt" die Ergebnisse auf

# Rechenwerksvarianten

**Variante A:** Hilfsregister des Akkumulators wird hinter die ALU verlegt

**Vorteil:** ALU-Operationen ohne Veränderung des Akkumulators sind möglich

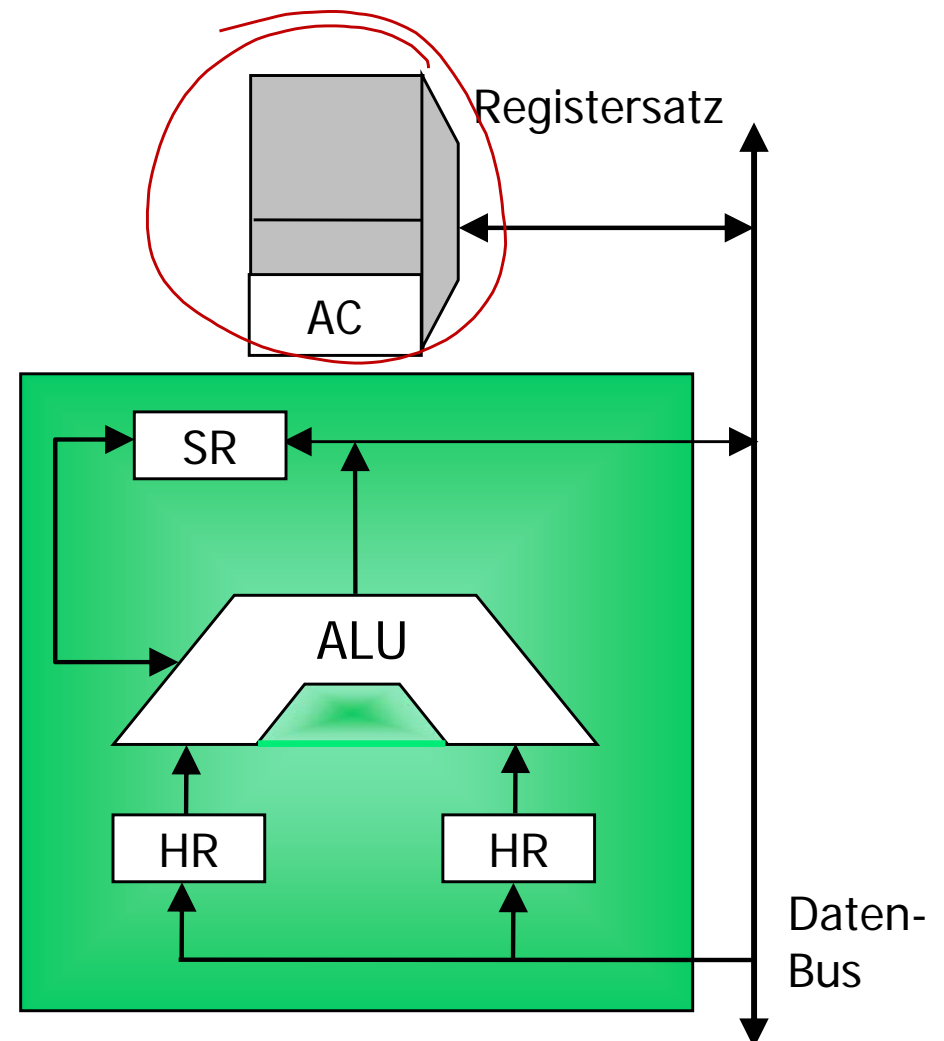


Haupsächlich in älteren 8-Bit-Prozessoren, bei denen auch die Adressberechnung in der ALU des Rechenwerkes durchgeführt wird

# Rechenwerksvarianten

**Variante B:** Rechenwerk ohne Akkumulator. Der Akkumulator wird in den Registersatz des Prozessors verlegt

**Vorteil:** Mehrere Register des Registersatzes können Akkumulatorfunktion übernehmen  
➔ mehrere Akkumulatoren

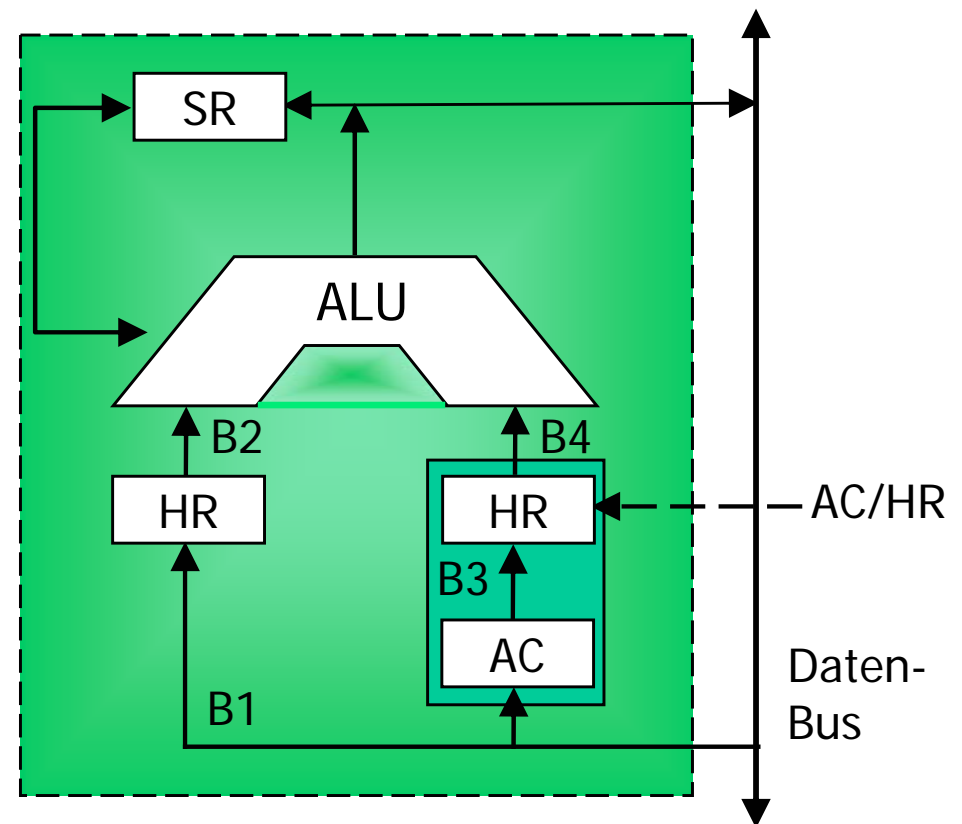


Alle modernen 16/32-Bit-Prozessoren nutzen dieses Konzept

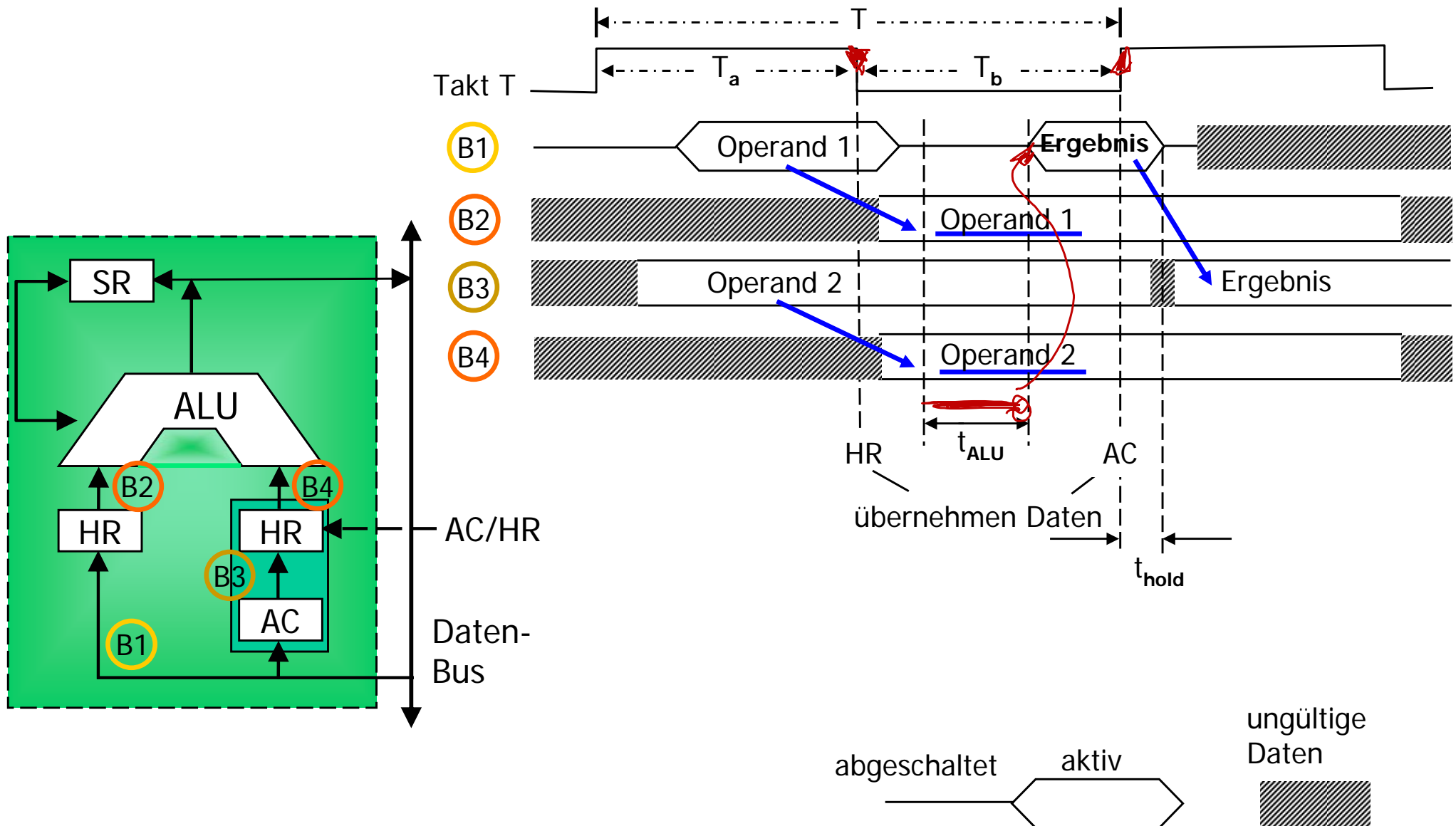
# Aufbau des Rechenwerks

## Akkumulator

meist zweigeteilt, bestehend aus dem eigentlichen Akkumulator-Register **AC** und einem nachgeschalteten Hilfsregister **HR** (Latch)



# Zeitverhalten des Rechenwerks



# Zeitverhalten des Rechenwerks

---

- Während der positiven Takthälfte von **T** liegen auf **B<sub>1</sub>** und **B<sub>3</sub>** die Operanden
- Diese werden mit der negativen Taktflanke in die Hilfsregister übernommen
- Danach stehen diese auf **B<sub>2</sub>** und **B<sub>4</sub>** stabil zur Verfügung, die ALU berechnet in einer Ausführungszeit **t<sub>ALU</sub>** das Ergebnis
- Dieses Ergebnis wird mit der positiven Taktflanke in den Akkumulator übernommen
- Ohne Hilfsregister würden sich die während der ALU-Rechenzeit ergebenden Schwankungen am Ausgang (Hasards, Wettläufe) sofort wieder auf den ALU-Eingang auswirken ➡ ein asynchrones Schaltwerk mit allen bekannten Problemen und Fehl-Ergebnissen würde entstehen.

# Operationsvorrat der ALU

---

## □ Arithmetische Operationen:

- Addieren ohne/mit Übertrag
- Subtrahieren ohne/mit Übertrag
- Inkrementieren/Dekrementieren
- Multiplizieren ohne/mit Vorzeichen
- Dividieren ohne/mit Vorzeichen
- Komplementieren (Zweierkomplement)

## □ Logische bitweise Verknüpfungen:

- Negation
- UND
- ODER
- Antivalenz

# Operationsvorrat der ALU

---

## ❑ Schiebe- und Rotations-Operationen:

- Links-Verschieben
- Rechts-Verschieben
- Links-Rotieren ohne Übertragsbit
- Links-Rotieren durchs Übertragsbit
- Rechts-Rotieren ohne Übertragsbit
- Rechts-Rotieren durchs Übertragsbit

## ❑ Transport-Operationen:

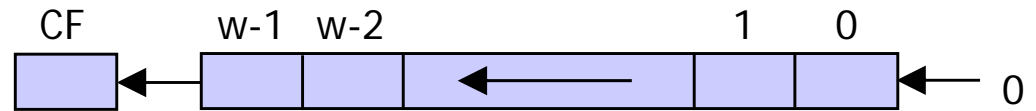
- Transferieren



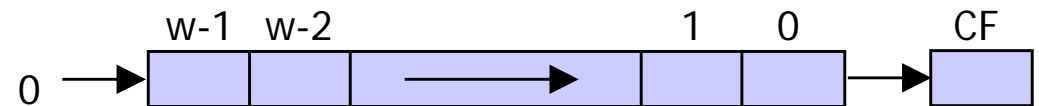
# Schiebeoperationen

- Dem logischen Linksschieben entspricht die Multiplikation mit 2
- Dem logischen Rechtschieben entspricht die Division durch 2
- Dies gilt jedoch nur für positive Zahlen. Bei negativen Zahlen im Zweierkomplement muss zur Vorzeichenerhaltung das höchstwertigste Bit in sich selbst zurückgeführt werden.

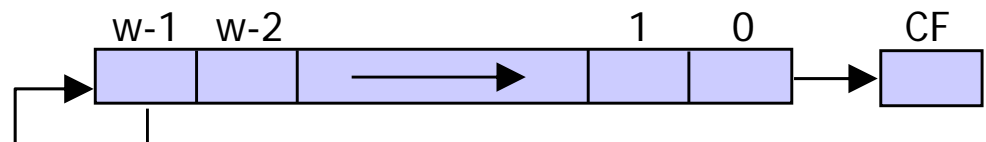
a) logisches und arithmetisches Verschieben nach links



logisches Verschieben nach rechts



b) arithmetisches Verschieben nach rechts

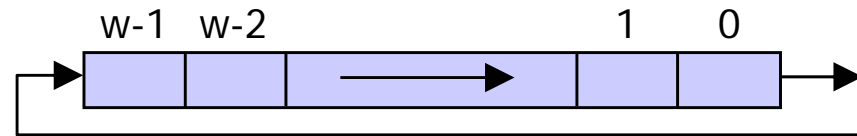


➡ Unterschied logisches/arithmetisches Rechtsschieben

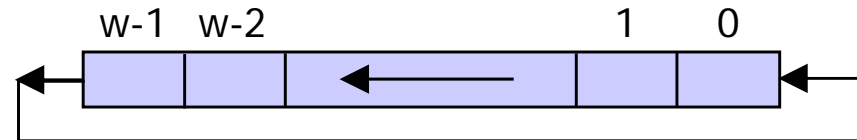
# Rotationsoperationen

- Register wird als geschlossene Bitkette betrachtet
- Carry-Flag kann wahlweise mitbenutzt oder als zusätzliches Bit einbezogen werden

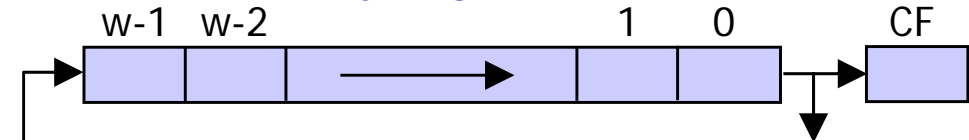
Rotieren nach rechts



Rotieren nach links



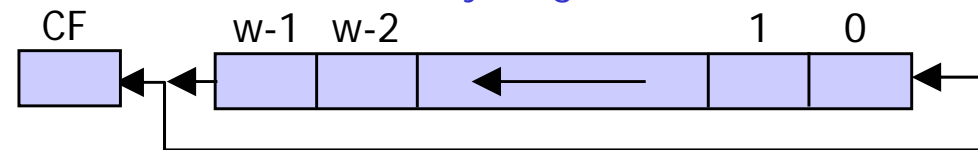
Rotieren nach rechts mit Carry-Flag



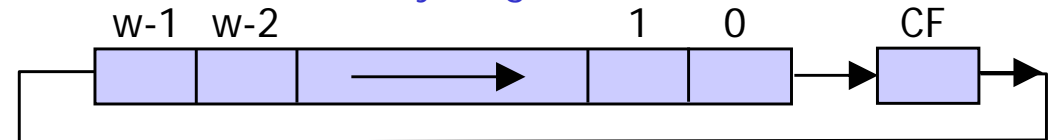
# Rotationsoperationen

- Register wird als geschlossene Bitkette betrachtet
- Carry-Flag kann wahlweise mitbenutzt oder als zusätzliches Bit einbezogen werden

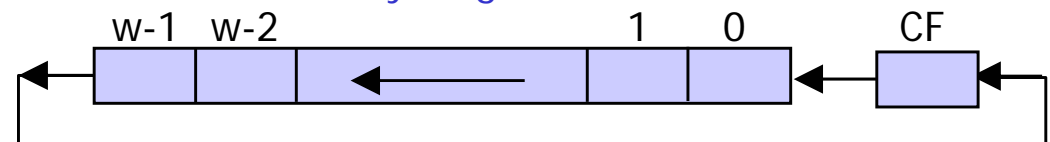
Rotieren nach links mit Carry-Flag



Rotieren nach rechts durch Carry-Flag

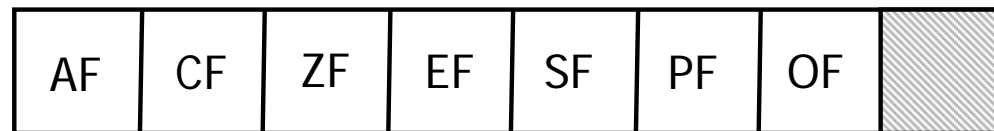


Rotieren nach links durch Carry-Flag



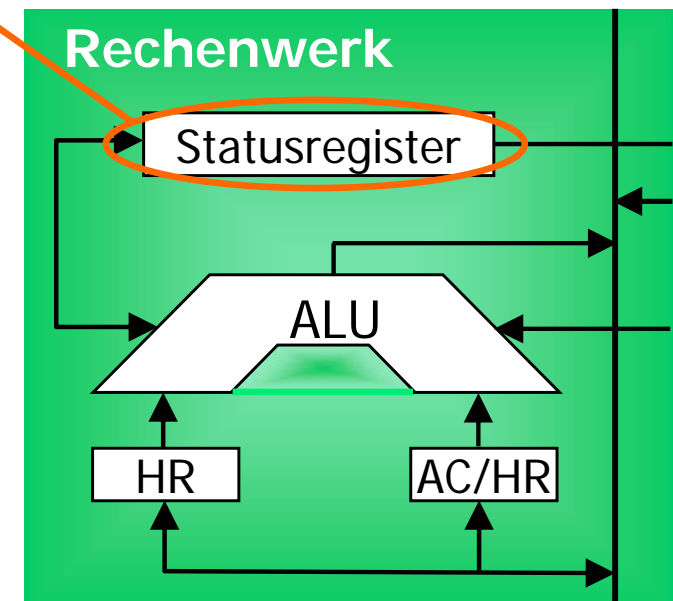
# Statusregister (Zustandsregister, Condition Code Register CCR)

Einzelne Bits, die das Ergebnis einer arithmetischen Operation widerspiegeln werden im Statusregister gespeichert



Das Statusregister enthält meist folgende Bits (Flags):

- Übertragsbit (Carry Flag CF)
- Hilfsübertragsbit (Auxiliary Carry AF)
- Nullbit (Zero Flag ZF)
- Vorzeichenbit (Sign Flag SF)
- Überlaufbit (Overflow Flag OF)
- Even Flag EF
- Paritätsbit (Parity Flag PF)



# Bedeutung der Statusflags

---

- **Carry Flag (CF):** Übertrag aus dem höchstwertigsten Bit bei Addition oder Subtraktion (Borrow) → sequentielle Addition und Subtraktion mit größerer Wortbreite ist möglich.
- **Aux Carry (AF):** Übertrag von Bit 3 in Bit 4 des Ergebnisses (BCD-Arithmetik).
- **Zero Flag (ZF):** Zeigt an, ob das Ergebnis der letzten Operation gleich 0 war (bedingte Programmverzweigungen, Zählschleifen)
- **Sign Flag (SF):** Zeigt an, dass das Ergebnis negativ ist (Most Significant Bit = 1). Spiegelt das höchstwertige Bit eines Operanden wieder (Bedingte Programmverzweigungen )
- **Overflow Flag:** Bereichsüberschreitung im Zweierkomplement
- **Even Flag (EV):** zeigt an, ob das Ergebnis eine gerade Zahl ist
- **Parity Flag (PF):** Signalisiert ungerade Parität des Ergebnisses,

# Statusregister (Zustandsregister, Condition Code Register CCR)

---

Werte von Statusbits können direkt Einfluß auf die Ausführung des Mikroprogramms haben

➔ bedingte Programmverzweigung

Statusregister und Steuerregister werden häufig zur Erleichterung der Adressierung zusammengefaßt betrachtet und manipuliert und meist **Prozessorstatuswort (PSW)** genannt

# Aufbau eines einfachen $\mu P$

---

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