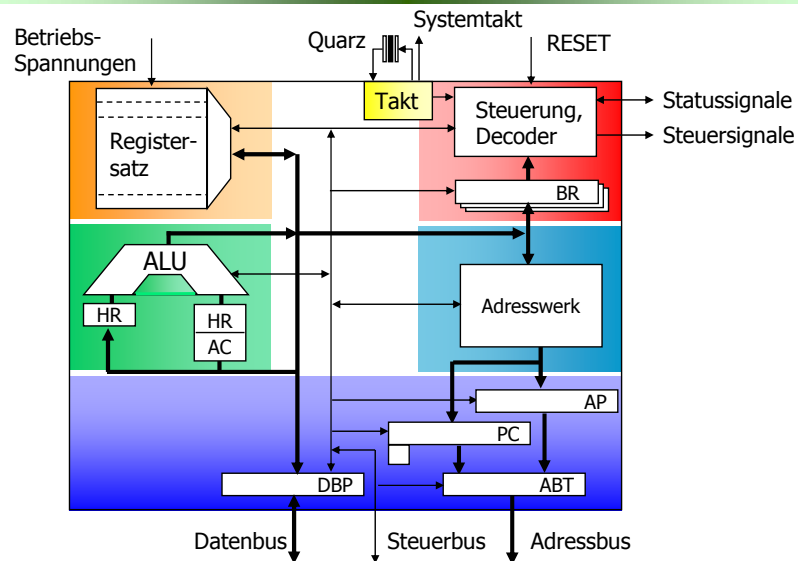


## 1.6 Interner Aufbau eines einfachen $\mu P$



## Interner Aufbau eines einfachen $\mu P$

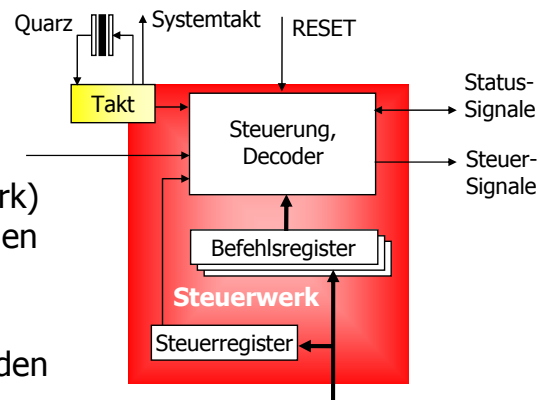
- **Steuerwerk**
- **Rechenwerk**
- **Registersatz**
- **Adresswerk**
- **Systembusschnittstelle**
- **Interne Busse**



## Steuerwerk

Steuert die Systemkomponenten

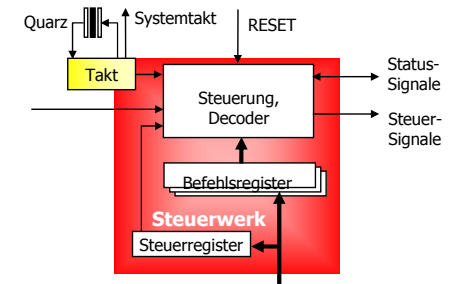
- **Befehlsregister**  
enthält den gerade ausgeführten Befehl
- **Dekoder** (mikro-programmiertes Schaltwerk)  
wird von den Statussignalen beeinflusst und erzeugt die Steuersignale
- **Taktgenerator** erzeugt den vom externen Quarz festgelegten Systemtakt



## 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.

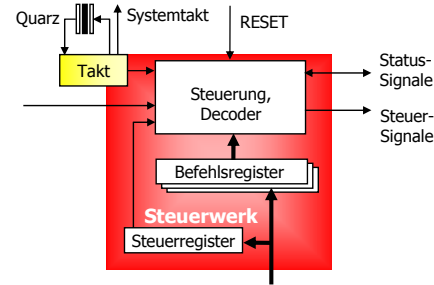
**Taktgenerator** ist bei modernen Mikroprozessoren on Chip (mit externem Quarz verbunden)



## Taktgenerator

### Aufgaben des Taktgenerators

- Taktfrequenz herstellen
- Erzeugung eines mit dem Prozessortakt synchronisierten Rücksetzsignals



Beim Rücksetzen durchläuft das Steuerwerk eine Initialisierungsroutine

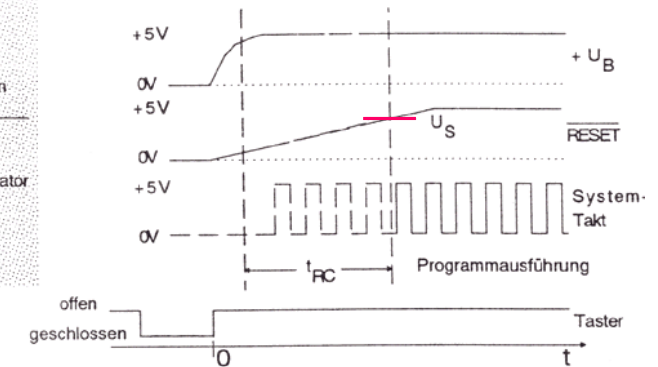
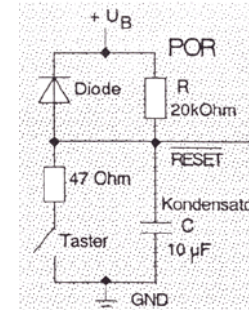
Diese wird bei vielen Mikroprozessoren ausgeführt, während das Rücksetzsignal aktiv ist

Deshalb muss das Rücksetzsignal genauen zeitlichen Spezifikationen genügen.



## Rücksetzen eines Prozessors

- manuelles Rücksetzen über Taster
- automatisches Rücksetzen nach Einschalten der Betriebsspannung



## Rücksetzen eines Prozessors

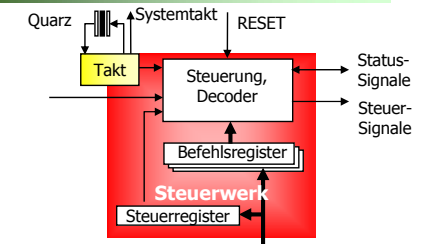
- Nach dem Einschalten der Versorgungsspannung wird der Kondensator C langsam über den Widerstand R aufgeladen
- Zeitkonstante:  $\tau = C * R$  ( $10 \mu F * 20K\Omega = 200 ms$ )
- Nach Überschreiten von  $U_s$  am Reset-Eingang beginnt die Programmausführung mit dem Laden des ersten Befehls
- Durch Druck auf die Taste kann C über einen Strombegrenzungs-Widerstand wieder entladen werden  
➔ manueller Reset
- Die Diode dient zum schnellen Entladen von C nach Abschaltung der Versorgungsspannung



## Steuerwerk

### Mikroprogrammsteuerwerk

Im Festwertspeicher liegt für jeden Befehl ein Mikroprogramm



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

### Aufbau eines Mikrobefehls:

Folge-Adresse	Register	Rechenwerk	Systembus Schnittstelle	Adresswerk	externe Steuersignale
		ALU	AC, HR		

Einzelne Bits eines Mikrobefehls  $\equiv$  Mikrooperationen  $\equiv$  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



## 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



## 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)



### 1. Systembus-Zuteilung:

meist müssen mehrer Komponenten eines Mikroprozessor-Systems aktiv auf den Systembus zugreifen → Systembuszuteilung (Bus Arbitration) ist erforderlich

### Quittierungsbetrieb mit 3-Leitungs-Handshake:

- **Busanforderung**

(*bus request HOLD, BR oder BREQ*)

Dieses Signal teilt dem Prozessor mit, dass eine andere Systemkomponente den Systembus belegen will



- **Busfreigabe** (*bus grant*)

**HOLDA** (Hold acknowledge), BG oder BGRT)

Mit diesem Signal teilt der Prozessor mit, dass der Systembus nun zur Verfügung steht. Alle Prozessorausgänge werden vorher auf hochohmig geschaltet

- **Busübernahme-Bestätigung** (*Bus grant acknowledge BGA*)

Mit diesem Signal bestätigt eine Komponente dem Prozessor, dass sie nun den Bus übernommen hat (neuer Bus-Master)

Bewerben sich mehrer Komponenten gleichzeitig um den Bus, so muss der Bus-Arbitrator eine hiervon auswählen



### 2. Unterbrechungsanforderungen:

Unterbrechungen (Interrupts) sind erforderlich, um auf besondere äußere Ereignisse reagieren zu können

Das laufende Programm wird unterbrochen, eine Unterbrechungsbehandlungsroutine (Interrupt Service Routine) angestoßen und nach deren Abarbeitung zum laufenden Programm zurückgekehrt



### Interrupt-Eingänge:

- **IRQ** (Interrupt request), INT oder INTR:

löst eine maskierbare Unterbrechungsbehandlung aus

- **NMI** (Non maskable interrupt):

löst eine nicht maskierbare Unterbrechungsbehandlung aus

- **HALT-Leitung:**

Aufforderung zum Stoppen des Prozessors, durch Interrupt oder RESET wieder aufweckbar, meist bei schweren Systemfehlern eingesetzt



## Ein-/Ausgabesignale des Steuerwerks

### 3. Fehlermeldungen:

- Fehlermeldung **BERR** (Bus error) kann von Systemkomponenten erzeugt werden, die gerade den Systembus benutzen
- Das Signal **BERR** zeigt dem Prozessor an, das bei der auf dem Systembus gerade laufenden Aktion ein Fehler aufgetreten ist, z. B.
  - ausbleibendes Quittungssignal,
  - Zugriff auf einen geschützten Speicherbereich,
  - Datenübertragungsfehler auf dem Bus, ... usw.

Das Signal BERR löst in der Regel eine Unterbrechungsbehandlung aus.



## Ein-/Ausgabesignale des Steuerwerks

### 4. Statusinformationen:

Durch eine Reihe von Statussignalen, wie

$FC_i, \dots, FC_0$	(Function code)
$ST_i, \dots, ST_0$	(Status)

kann der Prozessor angeschlossenen Komponenten seinen Zustand und den Typ der gerade ausgeführten Operation mitteilen, z. B. Benutzerprogramm/Systemprogramm; Halt-Zustand; Zugriff auf Arbeitsspeicher oder Peripherie, ... usw.



## Ein-/Ausgabesignale des Steuerwerks

### 5. Kommunikation mit Coprozessor:

Eine Reihe von Signalen ist hierfür vorgesehen (später)

### 6. Systembus-Steuersignale:

Bestimmen im wesentlichen:

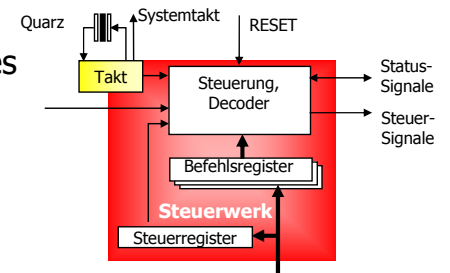
- Art und Richtung des Datentransports (lesen/schreiben, Byte/Word/Double-Word-Zugriff, ...)
- Art der selektierten Komponente (Arbeitsspeicher/Peripherie)



## 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

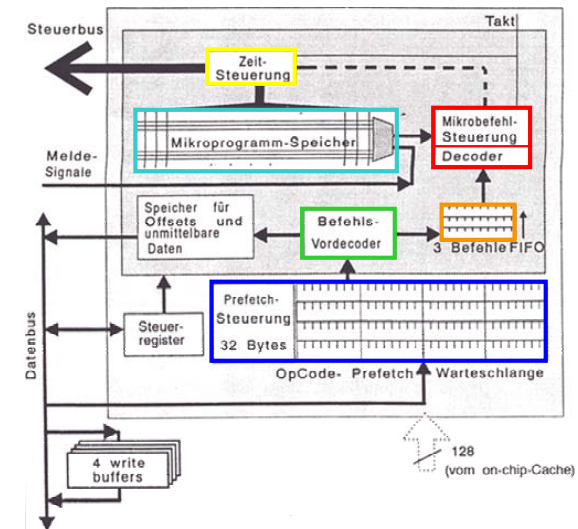
erlaubt Befehlsabarbeitung im Einzelschritt (Single Step Mode), d. h. nach jeder Befehlsausführung wird eine Unterbrechungsroutine gestartet → Debugging

## ➤ Decimal Bit

entscheidet, ob Dual oder BCD gerechnet wird



# Steuerwerk des Intel 80486



# Steuerwerk des Intel 80486

## 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 des Intel 80486

## 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 des Intel 80486

## Befehls-FIFO:

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

## Befehls-Decoder:

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



# Steuerwerk des Intel 80486

## Mikrobefehls-Steuerung, Mikrobefehls-Speicher:

synchrones mikroprogrammiertes Schaltwerk, erzeugt die Steuersignale, interpretiert die Meldesignale

## Zeit-Steuerung:

synchronisiert die erzeugten Steuersignale mit dem Systemtakt



# Steuerwerk: Zusammenfassung

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

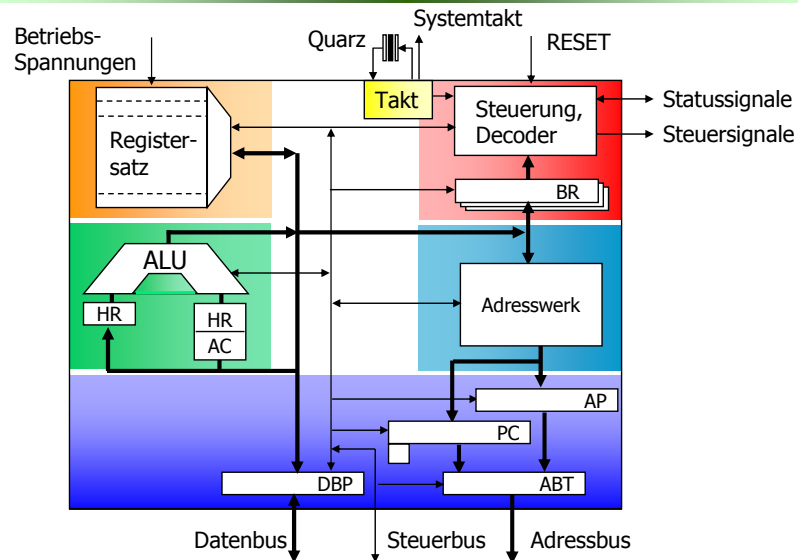


# Interner Aufbau eines einfachen $\mu P$

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

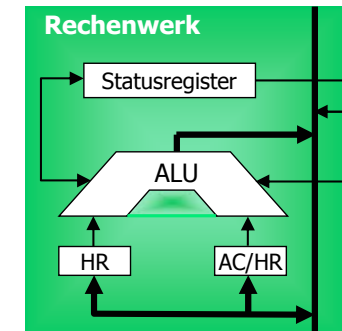


# Interner Aufbau eines einfachen µP



# 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



## Rechenwerk

- 2 Eingangsbusse ( 2 Operanden) und 1 Ausgangsbuss (Ergebnis). Sie entsprechen in der Breite dem internen Prozessor-Bus und sind mit diesem verbunden
- Die ALU selbst ist ein reines Schaltnetz
- Vor die ALU sind Hilfsregister zur Zwischenspeicherung von Operanden geschaltet
- Die 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



## Rechenwerk

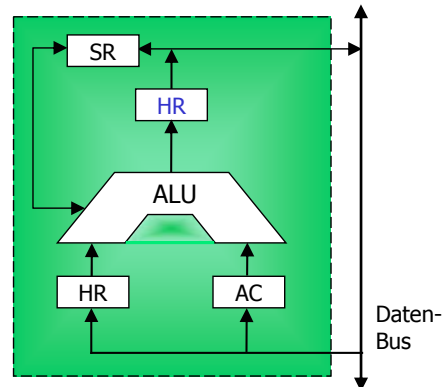
- Melde-Ausgänge für den Status der abgelaufenen Operation (Carry, Overflow, Zero, Sign, ..., usw.)  
Status wird im Status-Register zwischengespeichert
- **Akkumulator (spezielles Register)**  
Bei 8 Bit Mikroprozessoren war das einzige interne Register, das direkt Ergebnisse der ALU aufnehmen kann
  - ➔ 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



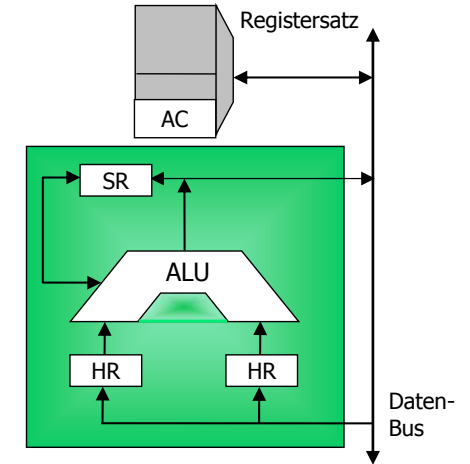
Hauptsächlich in älteren 8-Bit-Prozessoren, bei denen auch die Adressberechnung in der ALU des Rechenwerkes durchgeführt wird



# Rechenwerksvarianten

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

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



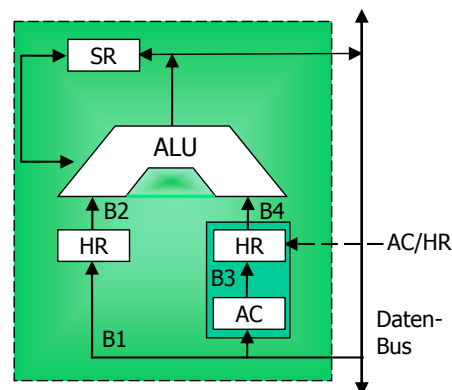
Alle modernen 16/32-Bit-Prozessoren nutzen dieses Konzept



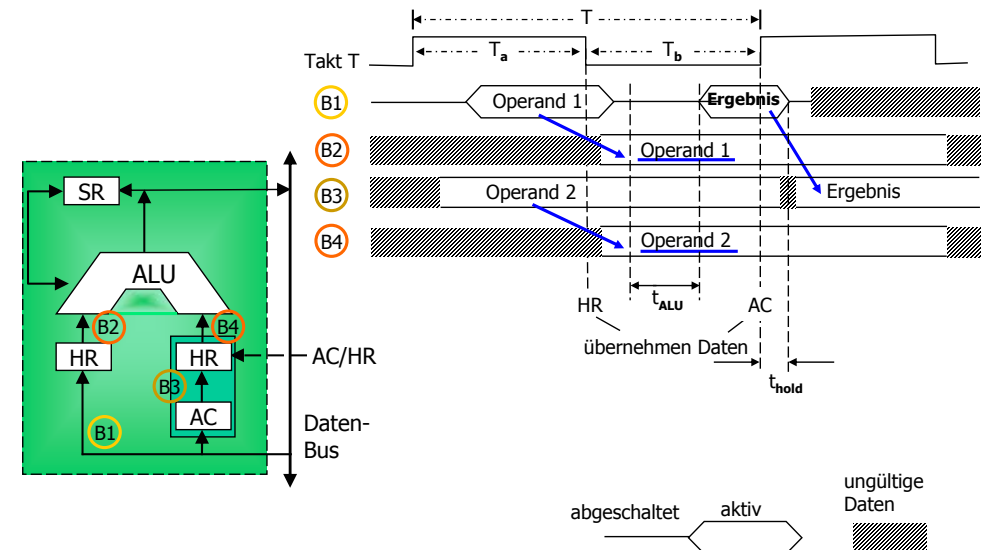
## Aufbau des Rechenwerks

### 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

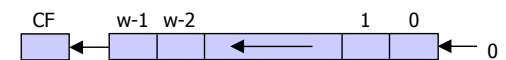
- **Schiebe- und Rotations-Operationen:**
  - Links-Verschieben
  - Rechts-Verschieben
  - Links-Rotieren ohne Übertragsbit
  - Links-Rotieren durchs Übertragsbit
  - Rechts-Rotieren ohne Übertragsbit
  - Rechts-Rotieren durchs Übertragsbit
- **Transport-Operationen:**
  - Transferieren



## 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.

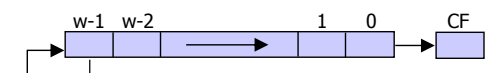
a) logisches und arithmetisches Verschieben nach links



logisches Verschieben nach rechts



b) arithmetisches Verschieben nach rechts



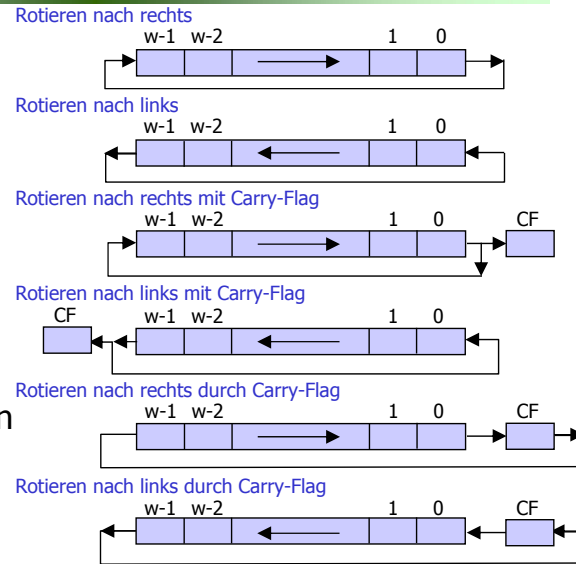
➔ Unterschied logisches/arithmetisches Rechtsschieben



## Rotationsoperationen

- Register wird als geschlossene Bitkette betrachtet

- Carry-Flag kann wahlweise mitbenutzt oder als zusätzliches Bit einbezogen werden



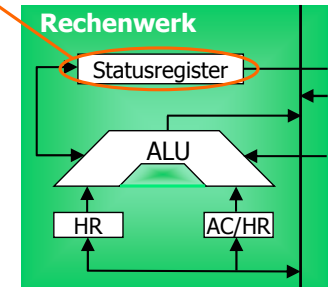
## Statusregister (Zustandsregister, Condition Code Register CCR)

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

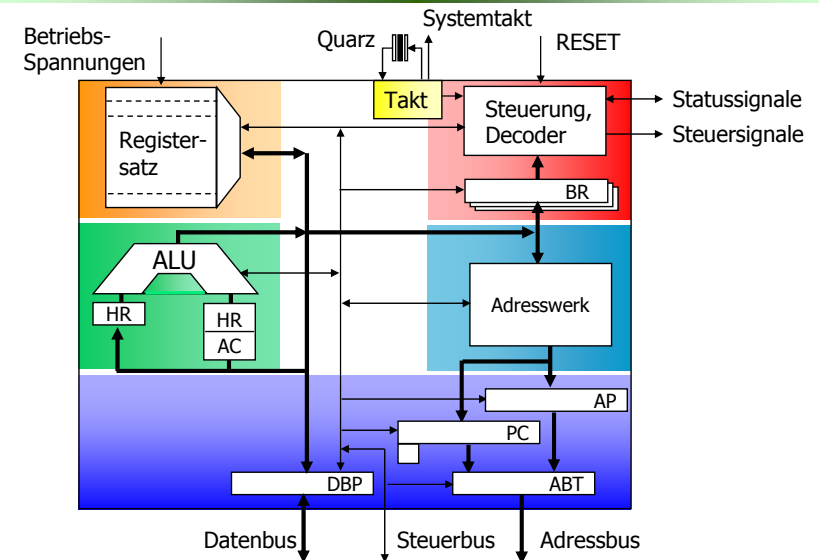
Statusregister und Steuerregister werden häufig zur Erleichterung der Adressierung zusammengefaßt betrachtet und manipuliert, und meist **Prozessorstatuswort (PSW)** genannt

# Interner Aufbau eines einfachen $\mu P$

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



# Interner Aufbau eines einfachen $\mu P$

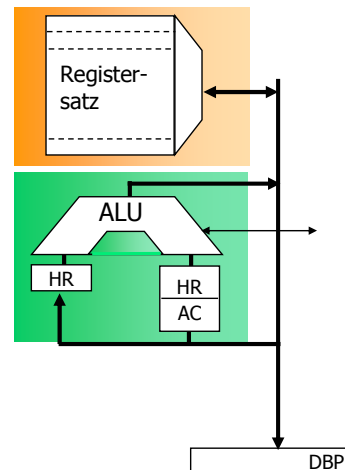


## Registersatz

Erweiterung des Rechenwerks:

Häufig benutzte Operanden können dort zwischengespeichert werden

→ schnellerer Zugriff als auf den Hauptspeicher

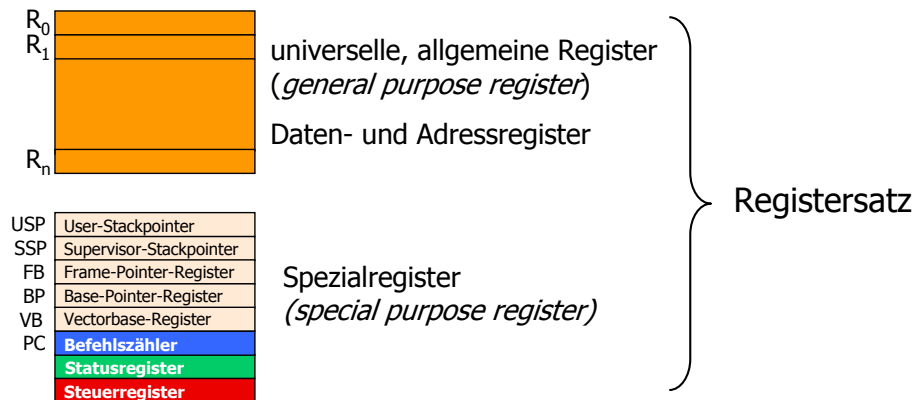


## Registersatz

- ❑ Getrennte Ein-/Ausgänge → Dual Port Speicher, zwischen Eingangs- und Ausgangsbuss gehängt
  - Schreiben eines Registers und gleichzeitiges Lesen eines anderen Registers möglich
- Heutige superskalar-Prozessoren: pro Takt mehrere allgemeine Register schreiben und lesen
- ❑ Oft dynamische Speicherzellen → Refresh erforderlich
- ❑ Register mit Zusatzfunktionen: Inkrementieren/Dekrementieren/auf Null setzen/Inhalt verschieben



# Registersatz



# Daten- und Adressregister

## Datenregister:

- Zwischenspeichern von Operanden
- schneller Zugriff auf häufig benutzte Operanden
- bei modernen Prozessoren sind mehrere Datenregister als Akkumulator nutzbar

## Adressregister:

Speichern von Adressen (oder Teile davon) eines Operanden im Hauptspeicher

- Basisregister
- Indexregister



## Spezialregister (special purpose register)

Register zur speziellen Verwendung:

- ❑ Programmzähler (instruction pointer)
- ❑ Steuerregister
- ❑ Statusregister
- ❑ Register für den Start von Interrupt-Behandlungen (interrupt vector base register)
- ❑ **Stackregister** (user und supervisor Stackpointer)

