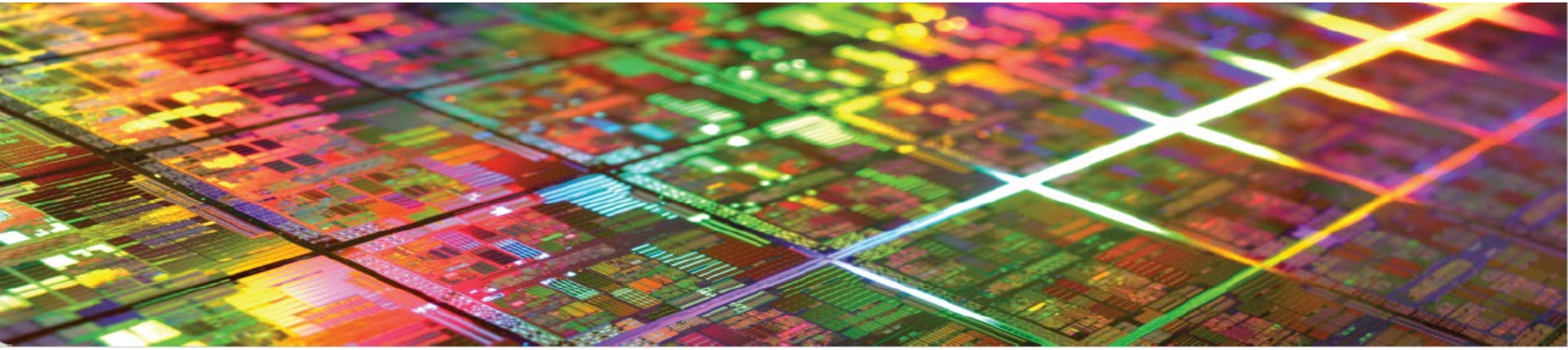
Karlsruher Institut für Technologie

Rechnerorganisation

Prof. Dr. Wolfgang Karl

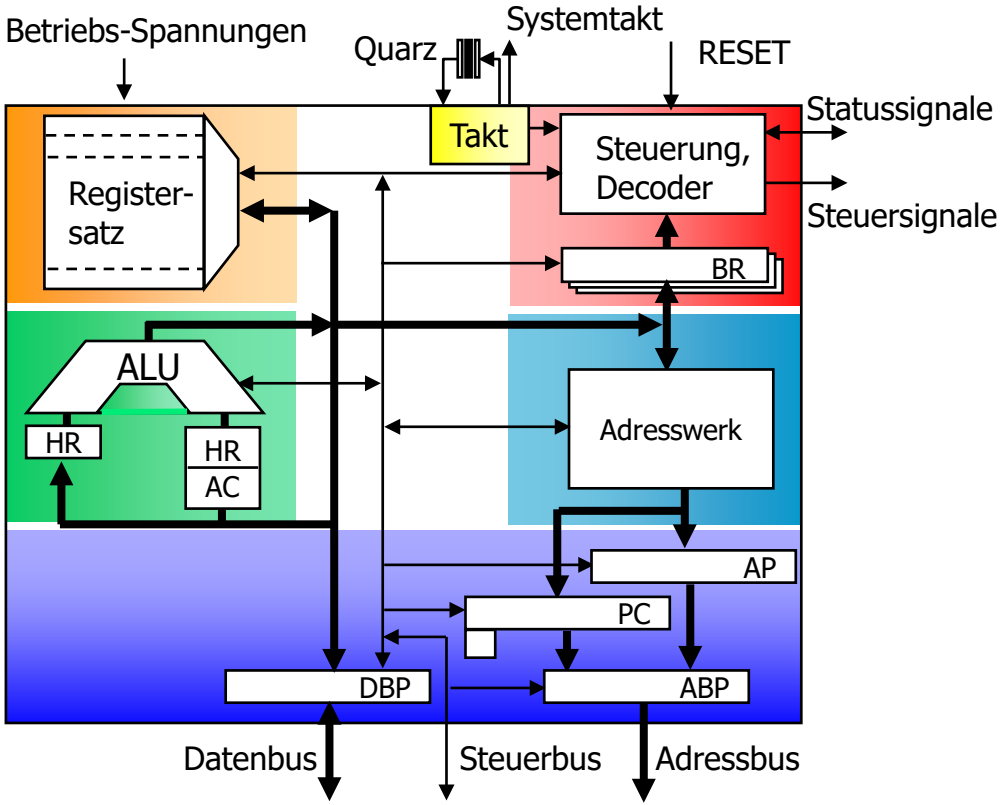
Vorlesung im Wintersemester 2025/2026 – Foliensatz: RO25-FS03



Kapitel 3
Aufbau eines einfachen Mikroprozessors

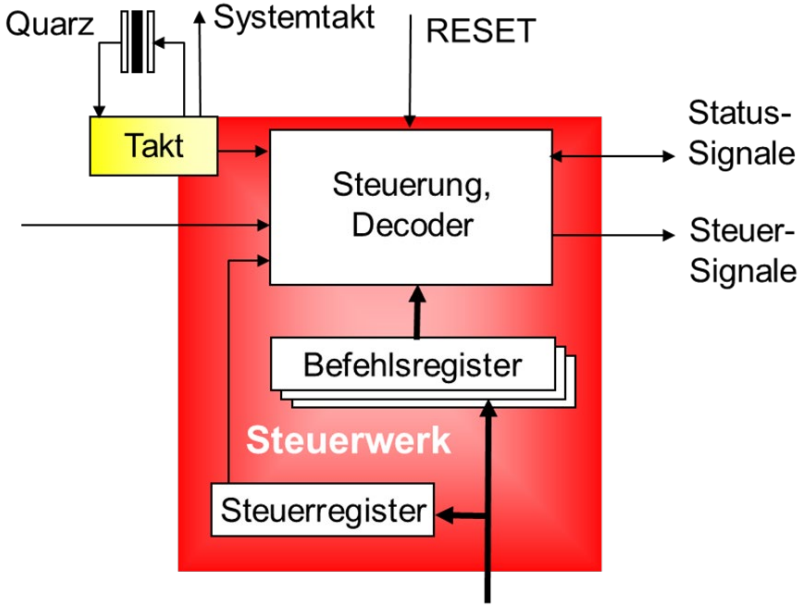
- Steuerwerk (Leitwerk)
- Rechenwerk
- Registersatz
- Adresswerk
- Busschnittstelle
- Internes Verbindungsnetzwerk

Aufbau eines einfachen Mikroprozessors



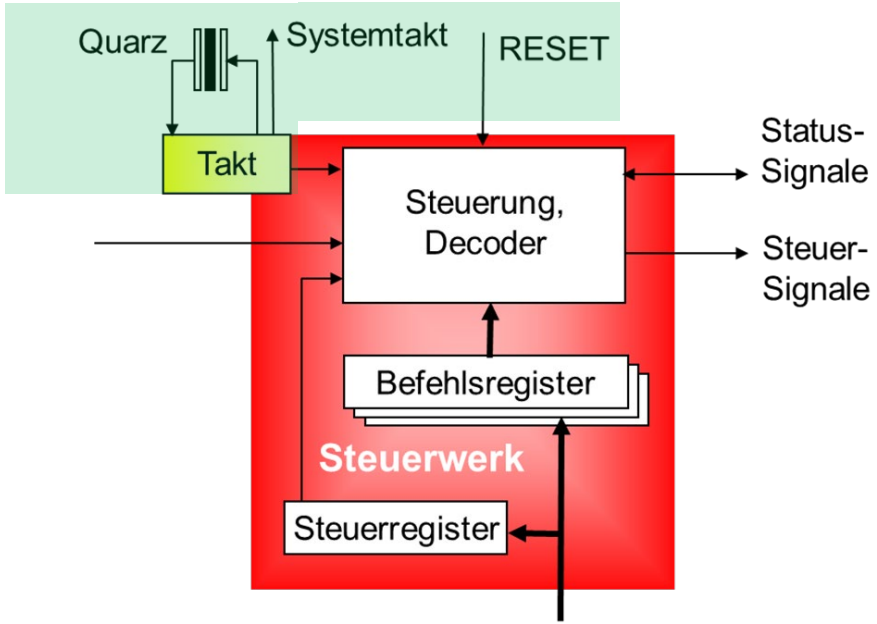
Aufbau eines einfachen Mikroprozessors

- **Steuerwerk**
  - Generierung der für die Steuerung der verschiedenen Werke im Rechner notwendigen Signale;



Aufbau eines einfachen Mikroprozessors

- **Steuerwerk: Taktgenerator**



Aufbau eines einfachen Mikroprozessors

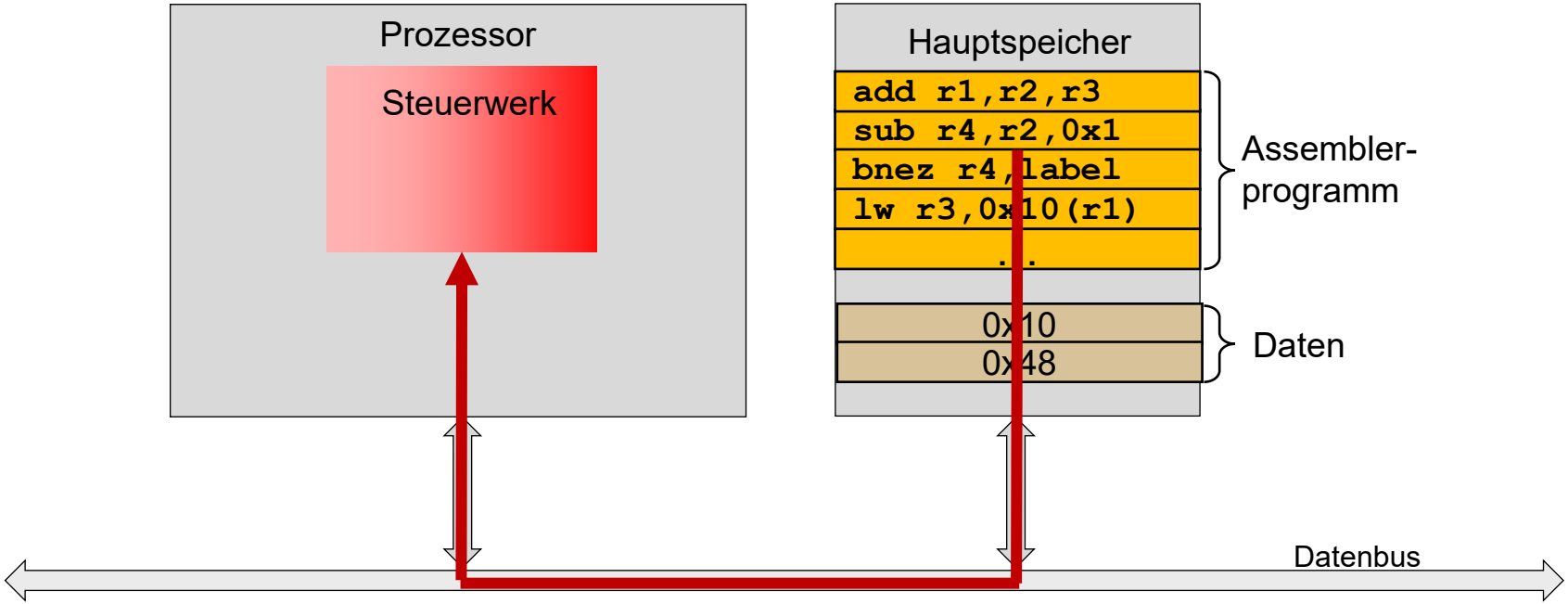
- **Steuerwerk: Taktgenerator**
  - Schaltung zur Erzeugung eines Taktsignals mit Hilfe eines Quarzoszillators;
    - Taktsignal ist durch die Frequenz (in Hertz) oder die Taktdauer (in Sekunden) charakterisiert.
    - Frequenz, Taktrate: Arbeitsgeschwindigkeit des Prozessors;
  - Taktgenerator ist bei modernen Mikroprozessoren auf dem Chip (mit externem Quarz verbunden).
    - Auf einem Prozessorchip können Frequenzen zur Steuerung der elektrischen Leistung dynamisch angepasst werden.
  - Erzeugen eines mit dem Prozessortakt synchronisierten Rücksetzsignals (RESET)
    - Zurücksetzen / Initialisierung des Prozessors
    - Erzeugen eines definierten Anfangszustands

Aufbau eines einfachen Mikroprozessors

- **Steuerwerk**
  - Holt die Befehle eines Programms aus dem Speicher;
  - Dekodiert die Befehle;
  - Steuert ihre Ausführung in der durch die Programmordnung vorgegebene Reihenfolge mithilfe von Steuer- und Status-Signalen;

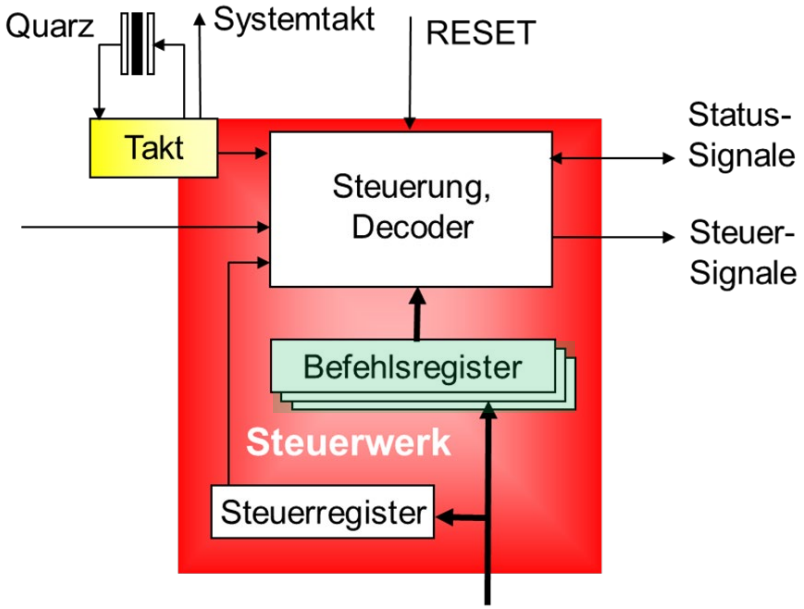
Aufbau eines einfachen Mikroprozessors

- **Steuerwerk**
  - Holt die Befehle eines Programms aus dem Speicher;



Aufbau eines einfachen Mikroprozessors

- **Steuerwerk: Befehlsregister**

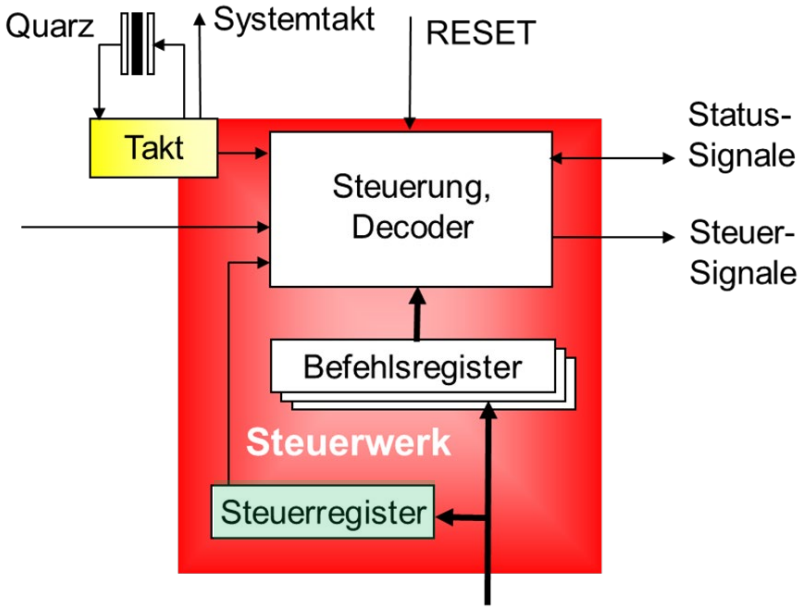


Aufbau eines einfachen Mikroprozessors

- **Steuerwerk: Befehlsregister**
  - Enthält den gerade auszuführenden Befehl;
  - Kann aus mehreren Registern bestehen, insbesondere bei einem variablen Befehlsformat.
    - Variables Befehlsformat: die Befehle sind unterschiedlich lang, im Allgemeinen ein Vielfaches von einem Byte;

Aufbau eines einfachen Mikroprozessors

- **Steuerwerk: Steuerregister (control register)**

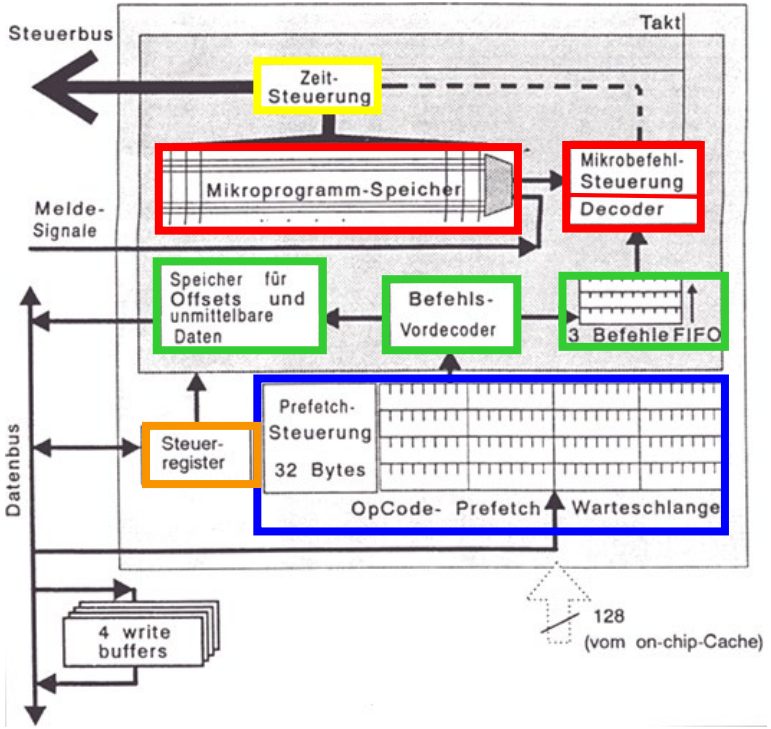


Aufbau eines einfachen Mikroprozessors

- **Steuerwerk: Steuerregister (control register)**
  - Mit Hilfe des Steuerregisters kann die aktuelle Arbeitsweise des Steuerwerks beeinflusst werden.
    - Es können bestimmte Arbeitsmodi für den Prozessor festgelegt werden.
    - Beispiel: User- oder System Modus (einstellbar über U/S-Bit)
      - Im **User-Modus** ist der Zugriff auf die Ressourcen des Prozessors / Rechners eingeschränkt.
        - Nur auf einen Teil der Speicherressourcen, Register kann zugegriffen werden.
        - Nur ein Teil des Befehlssatzes kann benutzt werden.
      - Im **System-Modus** sind alle Ressourcen des Rechners zugreifbar.
        - Auf alle Speicherressourcen / Register kann zugegriffen werden.
        - Alle Befehle sind verfügbar.

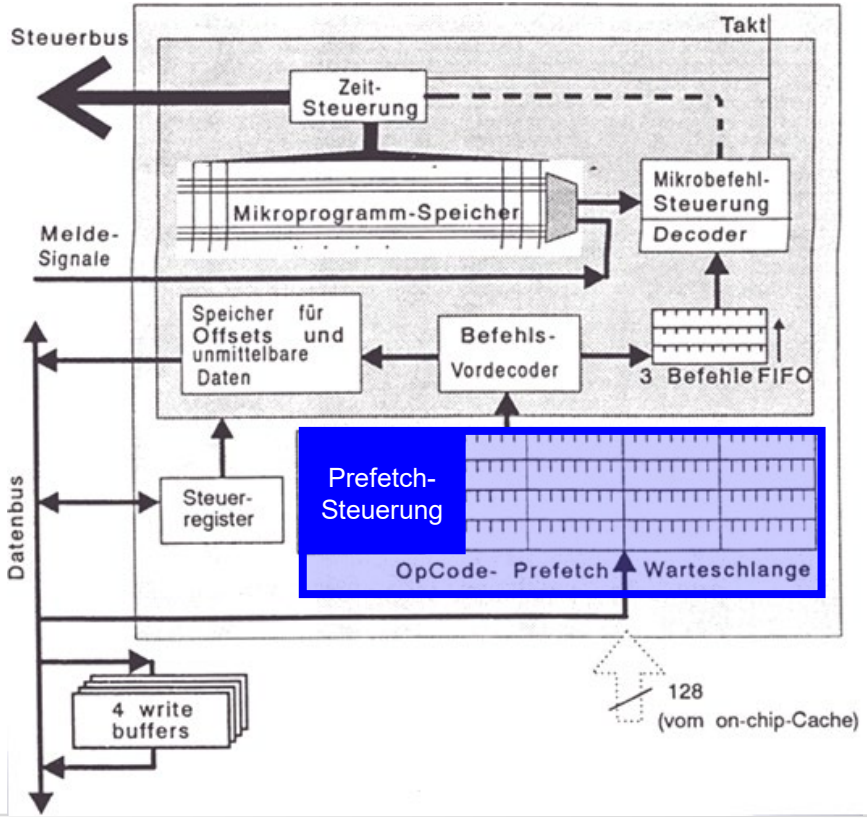
Aufbau eines einfachen Mikroprozessors

- **Fallstudie MIMA: mikroprogrammierbare Minimalmaschine**
  - **Steuerwerk**



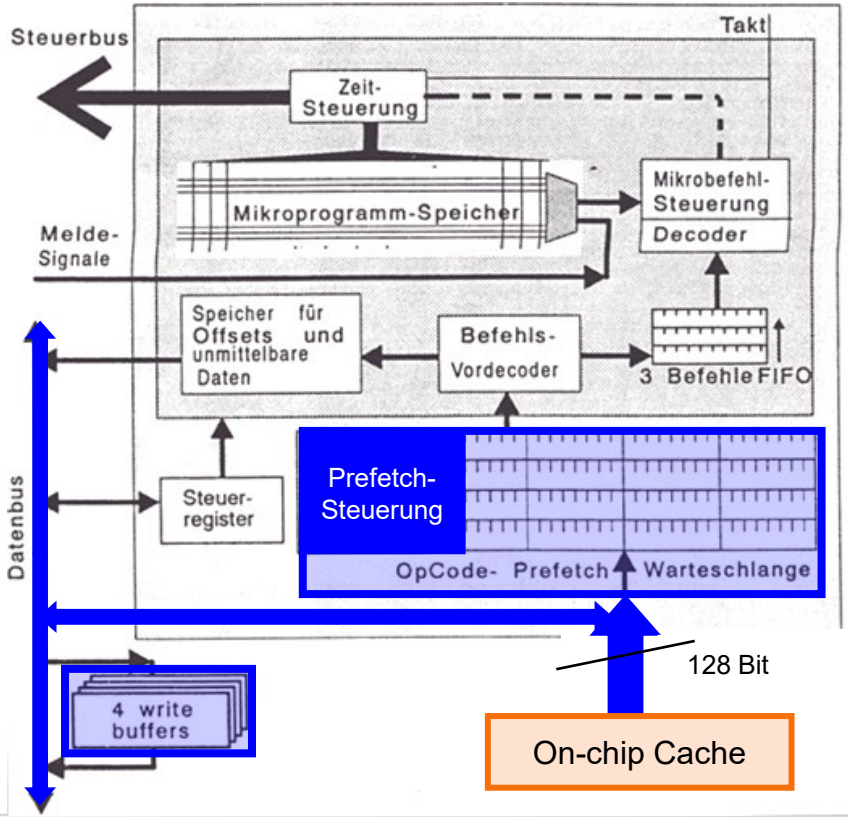
Fallstudie MIMA

- **Steuerwerk**
  - **Opcode Prefetch Warteschlange**
    - FIFO 32 Bytes
    - Speichert die nächsten zu verarbeitenden Befehle
  - **Prefetch-Steuerung**
    - Holt Befehle im voraus.



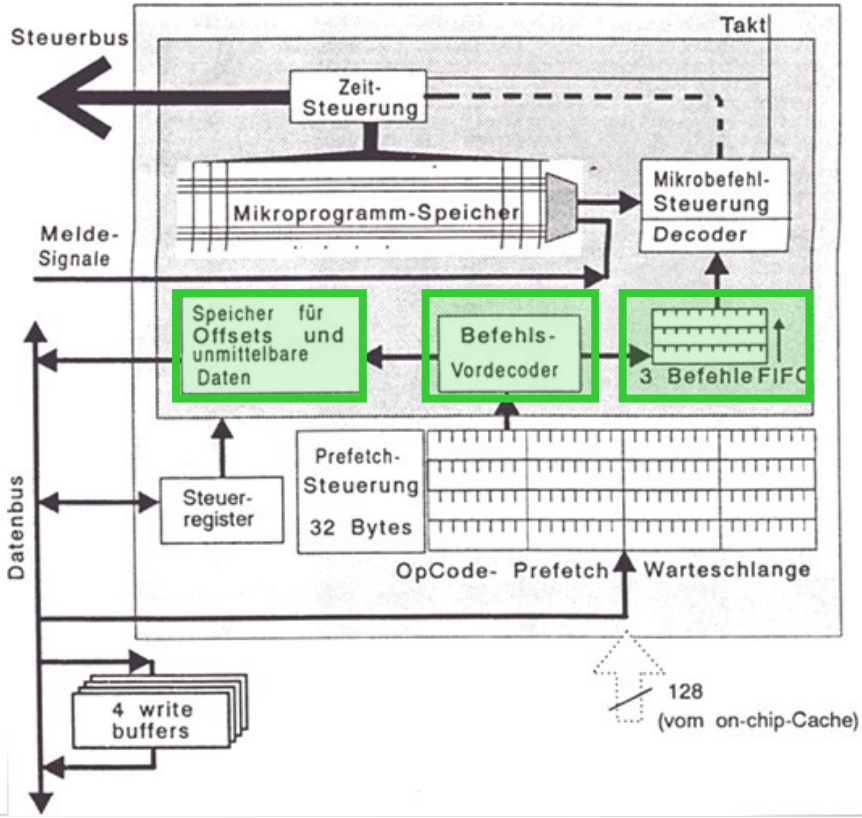
Fallstudie MIMA

- **Steuerwerk**
  - **Prefetch-Steuerung**
    - Die Prefetch-Warteschlange kann aus dem On-chip Cache über einen 128 Bit breiten Datenpfad geladen werden (Cache Hit);
    - Bei einem Cache-Fehlzugriff (Cache Miss) werden die Befehle aus dem Arbeitsspeicher geholt. Diese Zugriffe haben höchste Priorität.
    - Eventuell gleichzeitig über den Datenbus auszugebende Daten werden in **Schreibpuffern (write buffers)** zwischengespeichert.



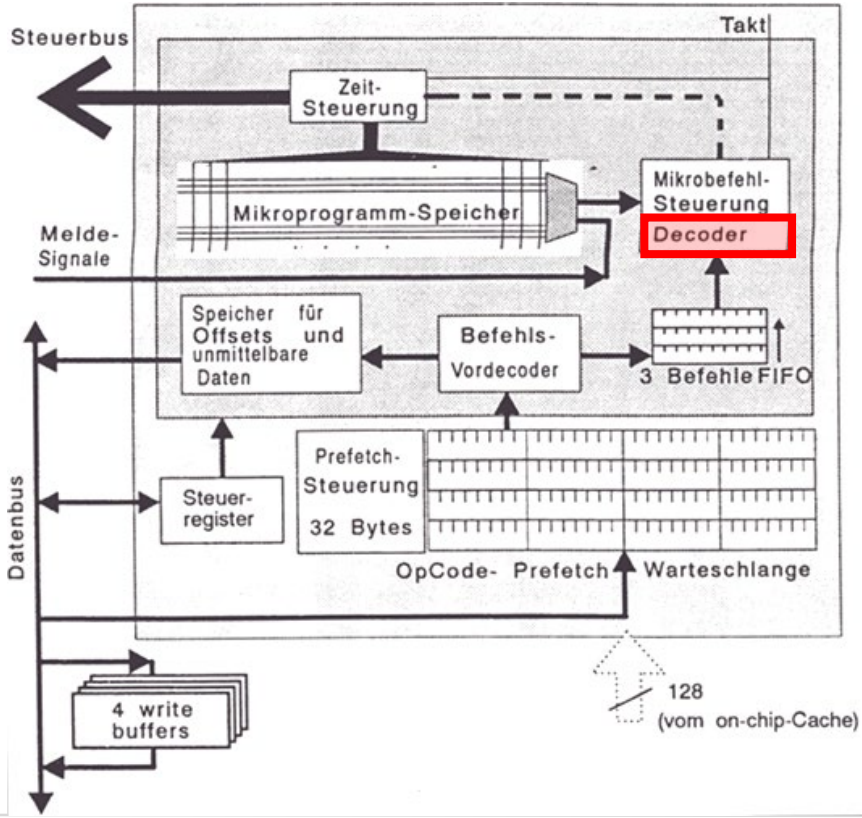
Fallstudie MIMA

- **Steuerwerk**
  - **Befehls-Vordecoder:**
    - Aus der Prefetch Warteschlange gelangen die Befehle in den Befehls-Vordecoder.
    - Vorbearbeitung der Befehle;
    - Direkt im Befehl angegebene Konstanten (unmittelbare Daten) sowie Adressdistanzen (Offsets) werden abgezweigt und separat gespeichert.
  - **Befehls-FIFO:**
    - Vordekodierte Befehle werden in den Befehls-FIFO geladen.
    - Platz für 3 Befehle



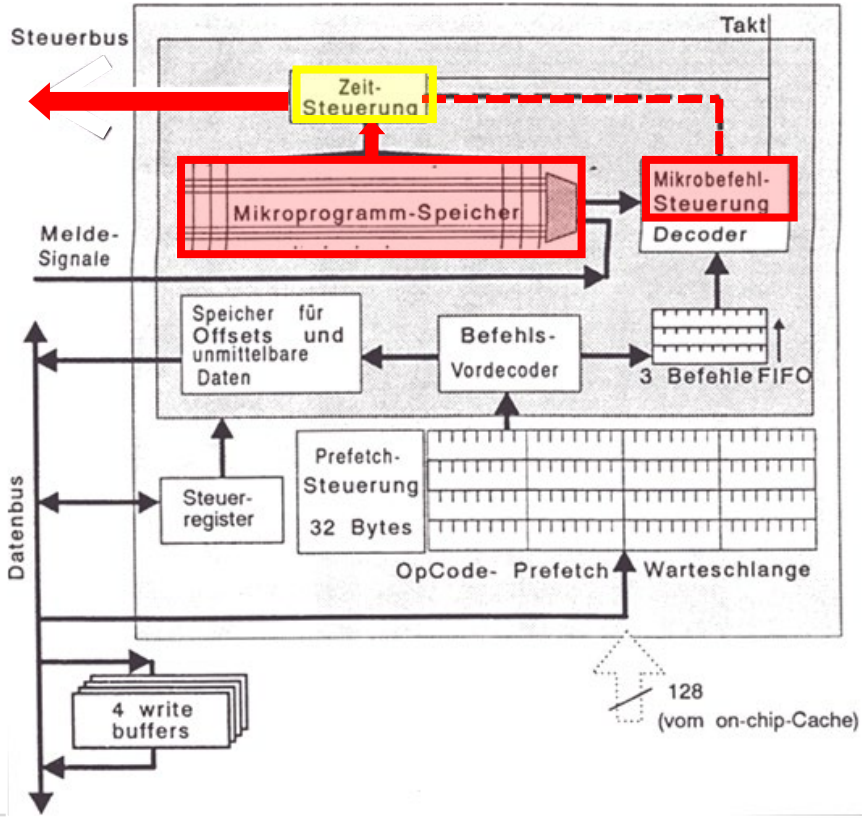
Fallstudie MIMA

- **Steuerwerk**
  - **Decoder:**
    - Holt einen vordecodierten Befehl aus dem obersten Element des Befehls-FIFOs
    - Ermittelt aus dem Opcode die Startadresse des zugehörigen Mikroprogramms;



Fallstudie MIMA

- **Steuerwerk**
  - **Mikroprogramm-Speicher:**
    - Enthält für jeden Maschinenbefehl ein Mikroprogramm;
  - **Mikrobefehls-Steuerung**
    - Übernimmt die Ausführung eines Mikroprogramms;
    - Erzeugt die Signale zur Ansteuerung der Komponenten im Rechner;
  - **Zeit-Steuerung:**
    - Synchronisiert die erzeugten Steuersignale mit dem Systemtakt;



Aufbau eines einfachen Mikroprozessors

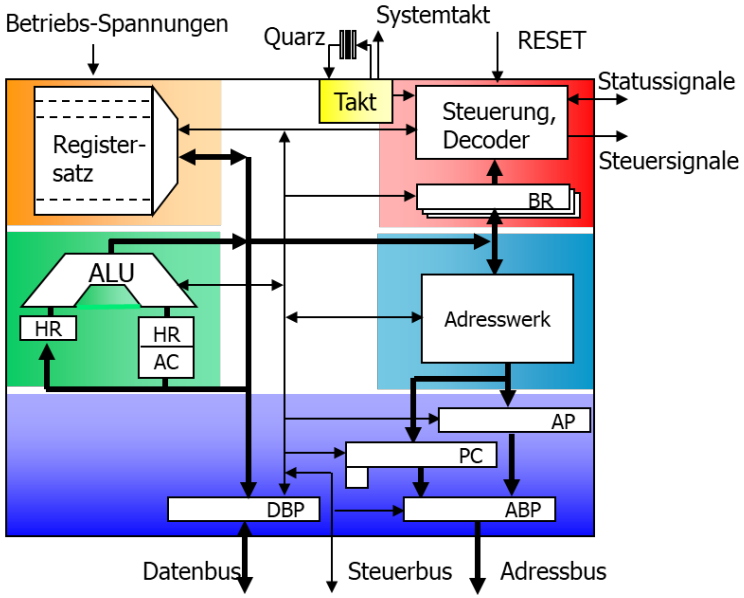
- **Mikroprogrammiertes Steuerwerk**
  - **Mikroprogrammierte Implementierung des Maschinenbefehlszyklus**
    - **Holphase**
      - Holen des Maschinenbefehls aus dem Speicher (On-chip Cache) in das Befehlsregister
    - **Decodierphase**
      - Einsprung in Mikroprogrammspeicher
      - Adresse wird aus Opcode des Maschinenbefehls generiert
    - **Ausführungsphase**
      - Abarbeiten des Mikroprogramms

Aufbau eines einfachen Mikroprozessors

- **Grundlagen Mikroprogrammierung**
  - Jeder **Maschinenbefehl** wird durch ein Mikroprogramm implementiert.
  - Ein **Mikroprogramm** besteht aus einer Folge von Mikrobefehlen.
    - Jedes Mikroprogramm ist im **Mikroprogrammspeicher** abgelegt.
  - Jeder **Mikrobefehl** fasst die **Mikrooperationen** (Steuersignale) zusammen, die in einem Taktzyklus angestoßen werden.

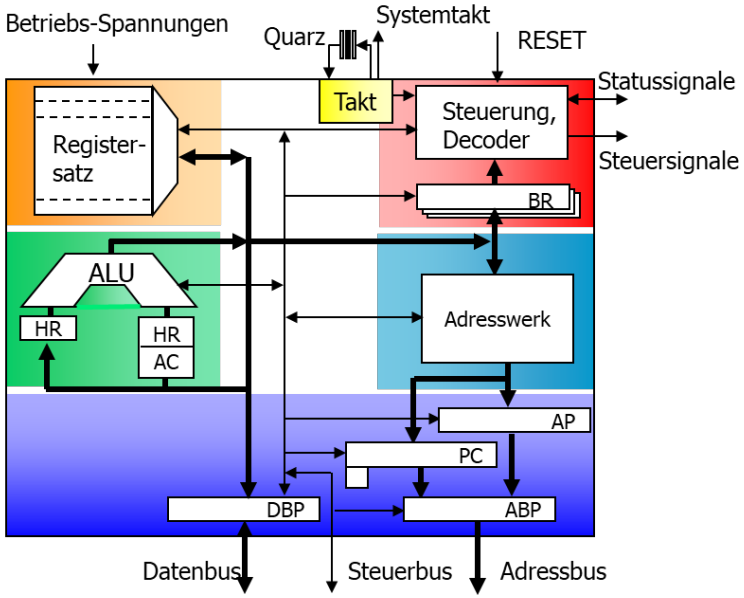
Aufbau eines einfachen Mikroprozessors

- **Grundlagen Mikroprogrammierung**
  - **Mikrooperationen (Steuersignale)**
    - Ansteuerung der Komponenten im Prozessor / Rechner
    - Beispiel:
    - ALU
      - Auszuführende Operationen:
        - add, subr, subs, or, and, not, xor, nand, …
    - Registersatz
      - Registeradressen
    - Adresswerk



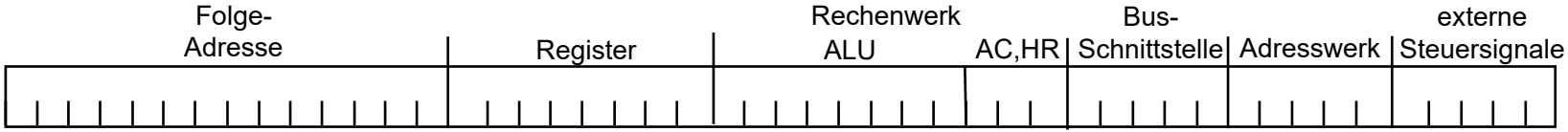
Aufbau eines einfachen Mikroprozessors

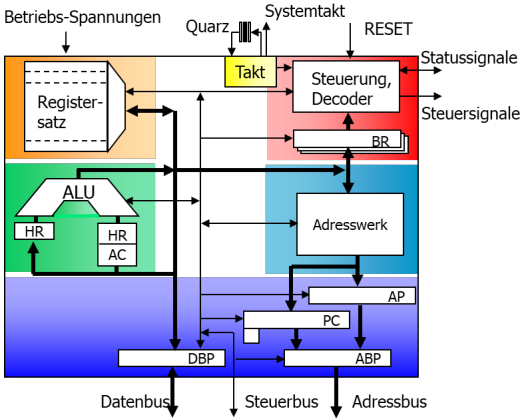
- **Grundlagen Mikroprogrammierung**
  - **Mikrooperationen (Steuersignale)**
    - Ansteuerung der Komponenten im Prozessor / Rechner
    - Beispiel:
    - Prozessorschnittstelle
      - Buszyklus
        - Lese-/Schreibsignal
        - Adresse auf Bus (ABP)
        - Datenübernahme, -übergabe (DBP)
      - Weitere Aktivitäten
        - Inkrementiere PC
        - Lade PC
        - Lade AP



Aufbau eines einfachen Mikroprozessors

- **Grundlagen Mikroprogrammierung**
  - Die Anordnung der Mikrooperationen ist im **Mikrobefehlsformat** festgelegt.
    - Beispiel:



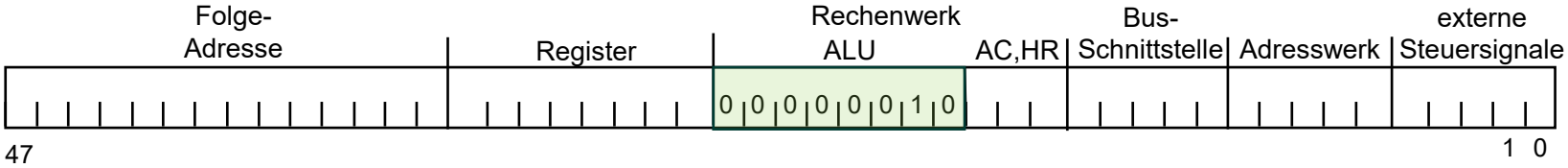


Aufbau eines einfachen Mikroprozessors

- **Grundlagen Mikroprogrammierung**
  - **Mikrobefehlsformat**
    - **Horizontale Mikroprogrammierung**
      - Jedes Bit im Mikrobefehl entspricht einer Mikrooperation (Steueroperation)

Aufbau eines einfachen Mikroprozessors

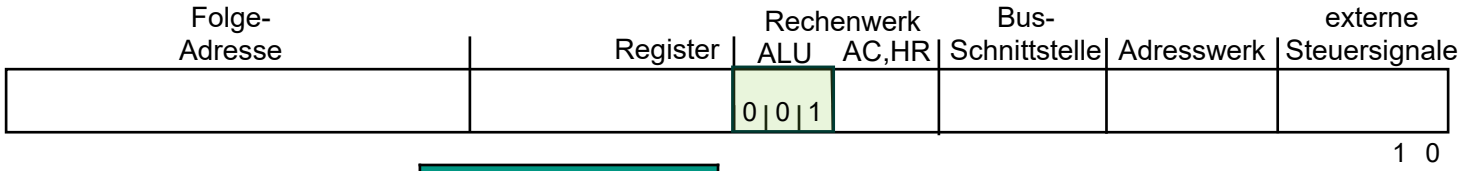
- **Grundlagen Mikroprogrammierung**
  - **Mikrobefehlsformat**
    - **Horizontale Mikroprogrammierung**
      - Jedes Bit im Mikrobefehl entspricht einer Mikrooperation (Steueroperstion)



8 ALU-Operationen								
0	0	0	0	0	0	0	1	add
0	0	0	0	0	0	1	0	subr
0	0	0	0	0	1	0	0	subs
0	0	0	0	1	0	0	0	or
0	0	0	1	0	0	0	0	and
…								

Aufbau eines einfachen Mikroprozessors

- **Grundlagen Mikroprogrammierung**
  - **Mikrobefehlsformat**
    - **Quasi-horizontale / vertikale Mikroprogrammierung**
      - Kodierung von Mikrooperationen



8 ALU-Operationen			
0	0	0	add
0	0	1	subr
0	1	0	subs
0	1	1	or
1	0	0	and
…			

Aufbau eines einfachen Mikroprozessors

- **Grundlagen Mikroprogrammierung**
  - **Mikroprogrammspeicher**
    - Jeder Maschinenbefehl wird durch ein Mikroprogramm implementiert.
    - Alle Mikroprogramme sind im Mikroprogrammspeicher (auf dem Prozessor) abgelegt.

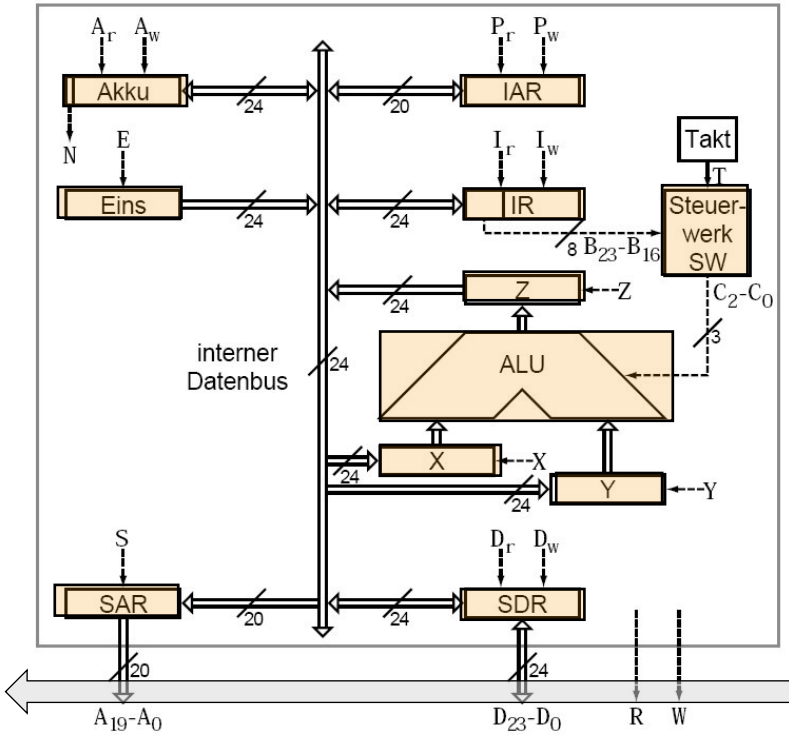
    - **Festwertspeicher (ROM)**
      - Mikroprogramme können nicht verändert werden
    - **Schreib-/Lesespeicher (RAM)**
      - Mikroprogramme können verändert werden.
      - Mit neuen Mikroprogrammen kann der Maschinenbefehlssatz erweitert werden.
      - **Mikroprogrammierbarer Rechner**

Aufbau eines einfachen Mikroprozessors

- **Alternative: Festverdrahtetes Steuerwerk**
  - Steuersignale werden mit Hilfe eines Schaltnetzes generiert.
  - Beispiel: RISC-V Prozessor

Fallstudie MIMA

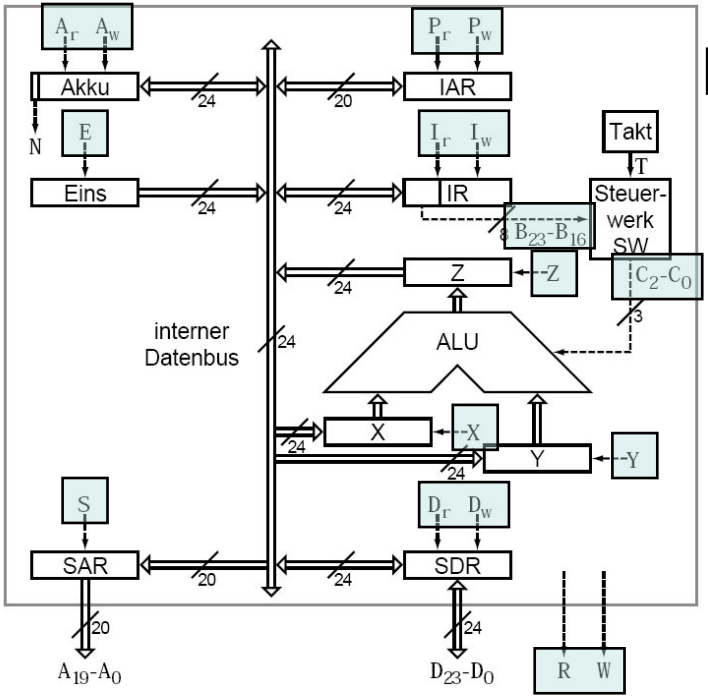
Mikroprogrammierbare Minimalmaschine



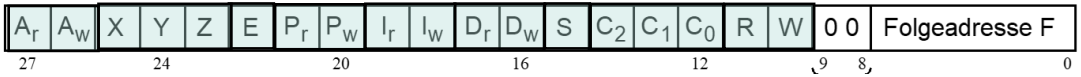
Steuerwerk
ALU
X, Y, Z: Hilfsregister
IAR: Instruction Adress Register, Befehlszähler
IR: Instruction Register: Befehlsregister
Akku: Akkumulator
Eins: Register mit Konstante 1 festverdrahtet
SAR: Speicheradressregister
SDR: Speicherdatenregister

Fallstudie MIMA

Mikroprogrammsteuerung



Mikrobefehlsformat



Mikrooperationen:

R, W: Lese- / Schreibsignal für Speicher

C_2, C_1, C_0, ALU-Operationen

S: Lade SAR Register

D_R, D_W: Lese- /Schreibsignal für SDR Register

I_R, I_W: Lese- /Schreibsignal für IR Register

P_R, P_W: Lese- /Schreibsignal für IAR Register

E: Enable Eins-Register

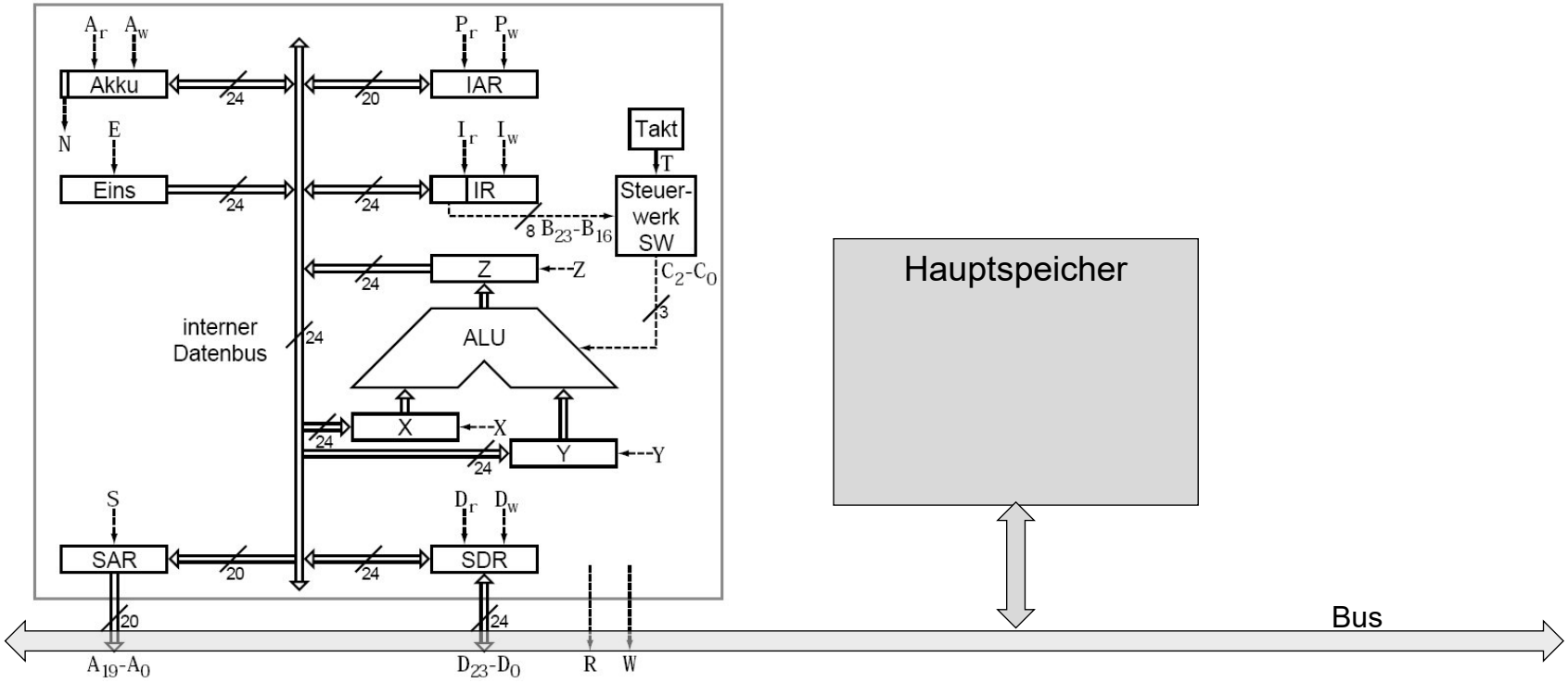
X,Y,Z: Steuersignale für Hilfsregister

A_R, A_W: Lese- /Schreibsignal für Akku

B_{23}-B_{16}: Opcode des Maschinenbefehl

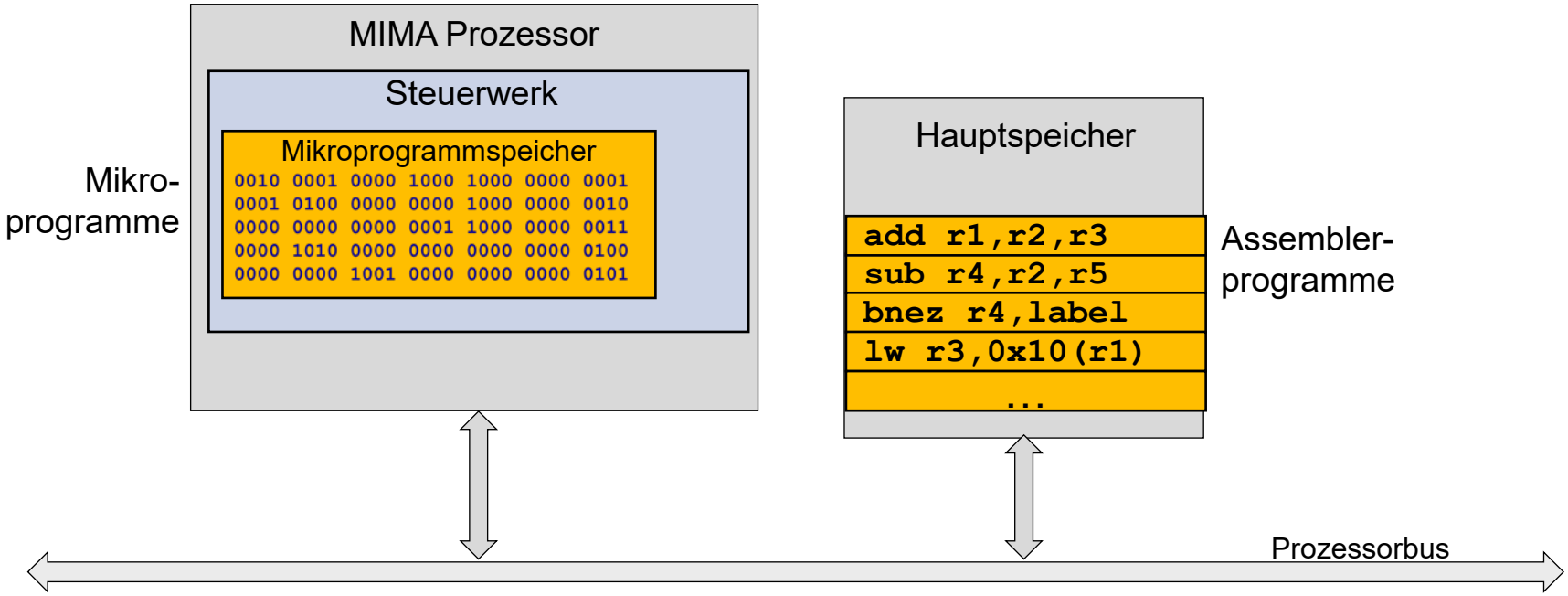
Fallstudie MIMA

- **Mikroprogrammsteuerung**



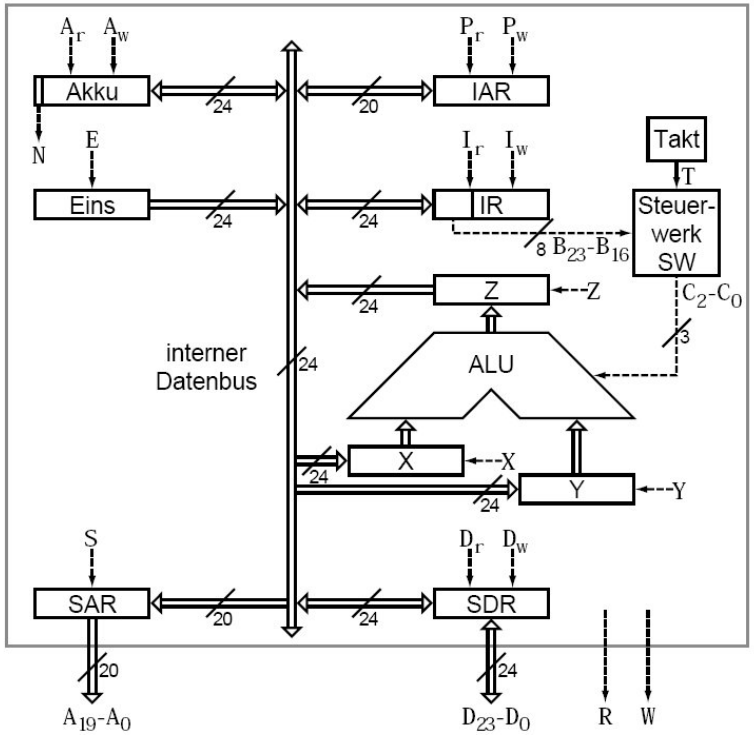
Fallstudie MIMA

Mikroprogrammsteuerung



Fallstudie MIMA

Mikroprogrammsteuerung: Befehlszyklus



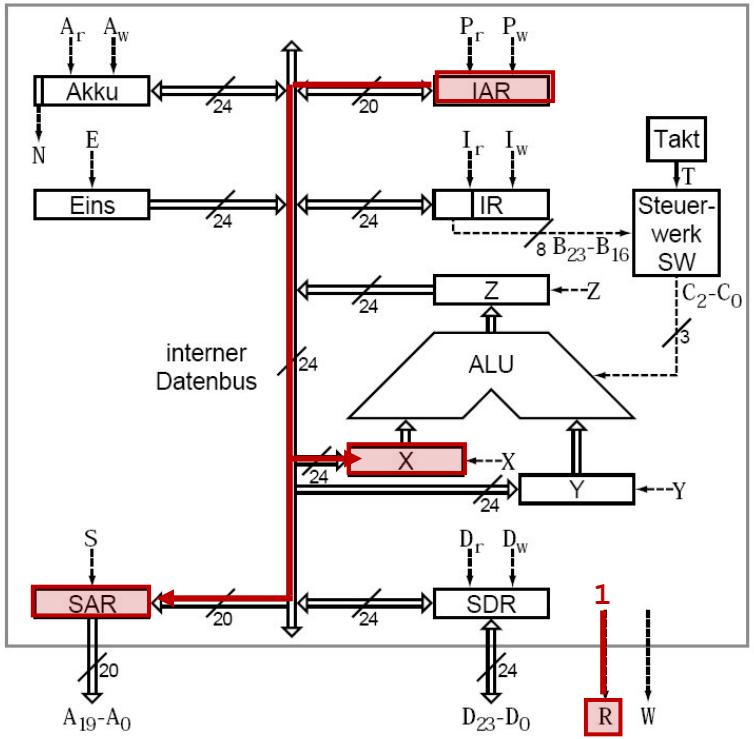
Holphase:
Holen des Maschinenbefehls aus dem Speicher in das Befehlsregister (IR)

Decodierphase
Einsprung in Mikroprogrammspeicher
Adresse wird aus Opcode des Maschinenbefehls generiert

Ausführungsphase
Abarbeiten des Mikroprogramms

Fallstudie MIMA

Mikroprogrammsteuerung: Befehlszyklus



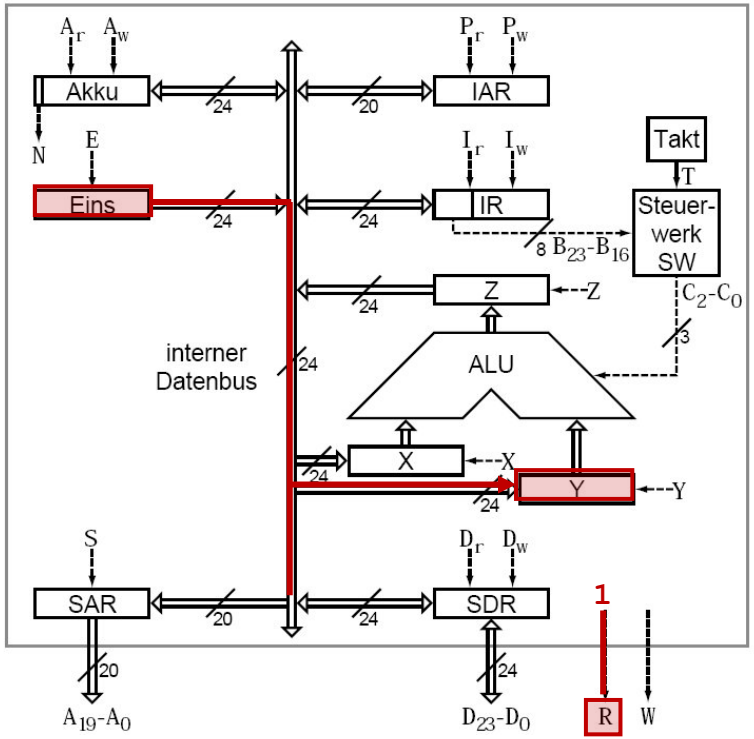
Holphase:

Mikroprogramm in Pseudo-Notation

```
1.Takt: IAR→SAR; IAR→X; R=1
```

Fallstudie MIMA

Mikroprogrammsteuerung: Befehlszyklus



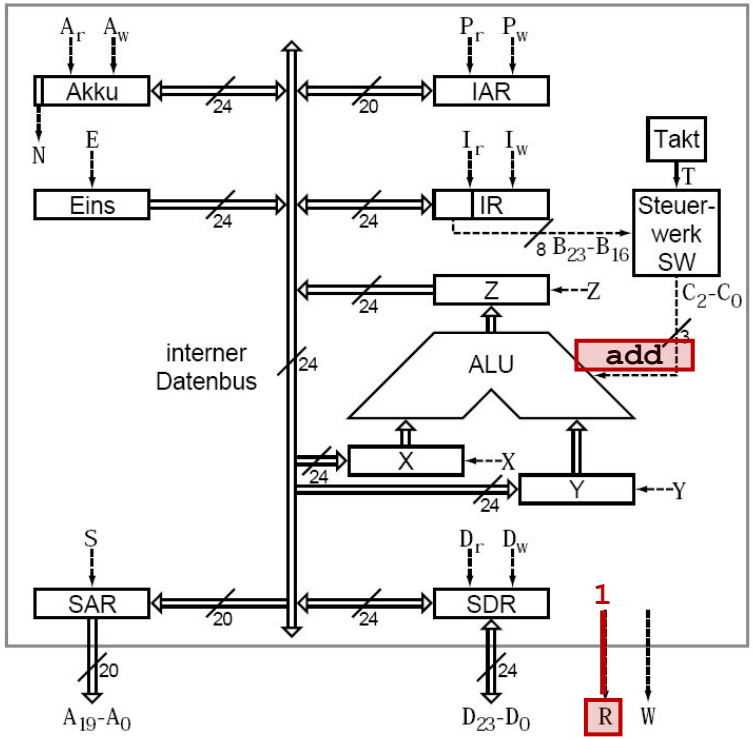
Holphase:

Mikroprogramm in Pseudo-Notation

```
1.Takt: IAR→SAR; IAR→X; R=1
2.Takt: Eins→Y; R=1
```

Fallstudie MIMA

Mikroprogrammsteuerung: Befehlszyklus

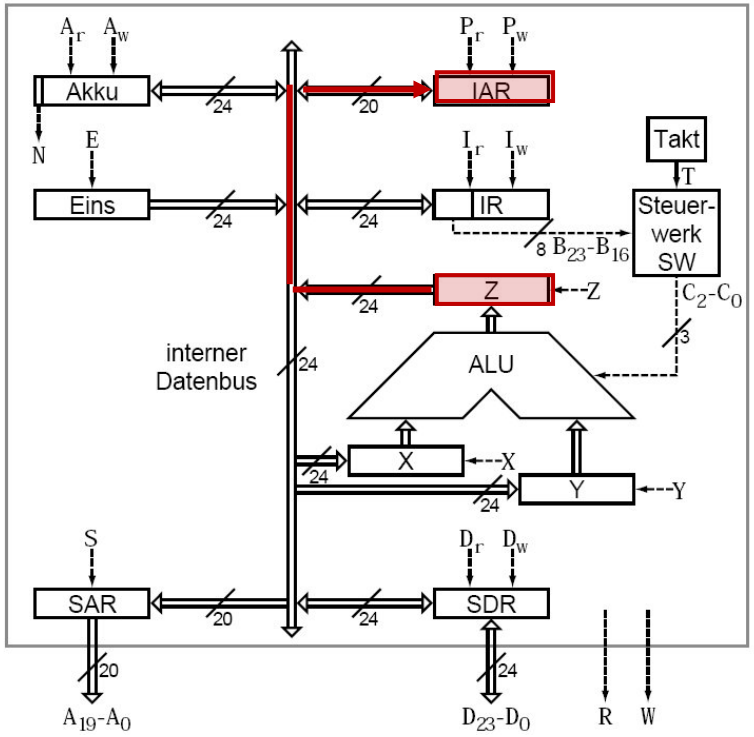


Holphase:
Mikroprogramm in Pseudo-Notation

```
1.Takt: IAR→SAR; IAR→X; R=1
2.Takt: Eins→Y; R=1
3.Takt: ALU add; R=1;
```

Fallstudie MIMA

Mikroprogrammsteuerung: Befehlszyklus



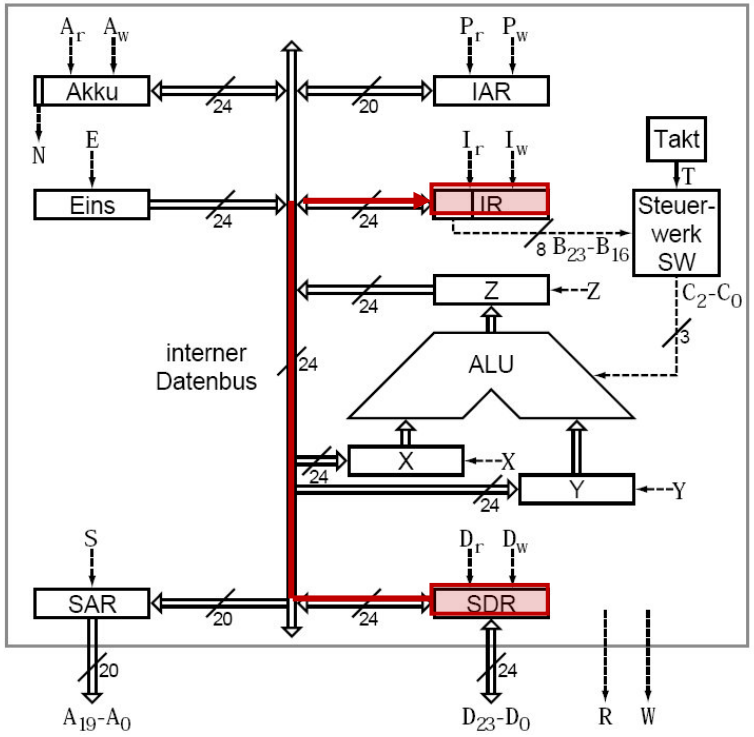
Holphase:

Mikroprogramm in Pseudo-Notation

```
1.Takt: IAR→SAR; IAR→X; R=1
2.Takt: Eins→Y; R=1
3.Takt: ALU add; R=1;
4.Takt: Z→IAR;
```

Fallstudie MIMA

Mikroprogrammsteuerung: Befehlszyklus



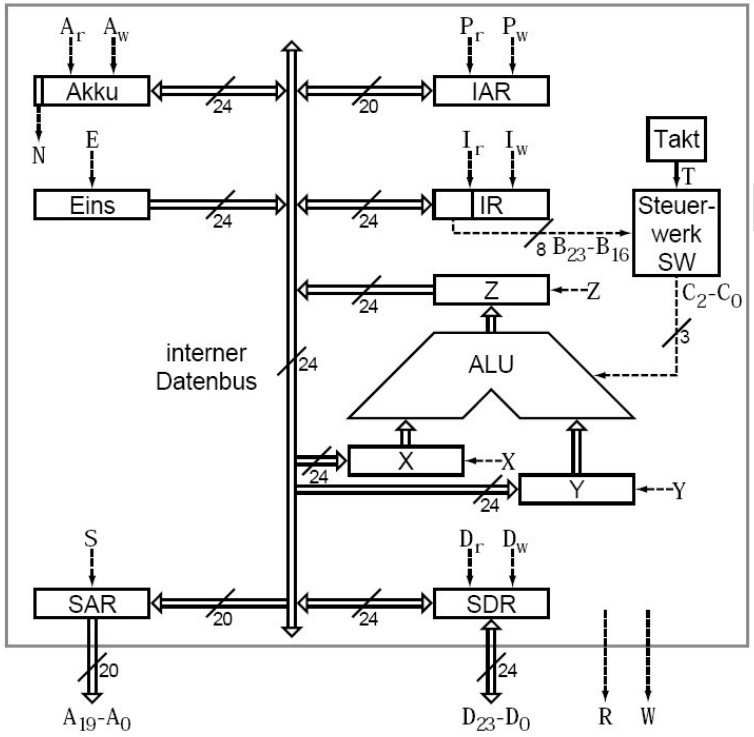
Holphase:

Mikroprogramm in Pseudo-Notation

```
1.Takt: IAR→SAR; IAR→X; R=1
2.Takt: Eins→Y; R=1
3.Takt: ALU add; R=1;
4.Takt: Z→IAR;
5.Takt: SDR→IR;
```

Fallstudie MIMA

Mikroprogrammsteuerung: Befehlszyklus



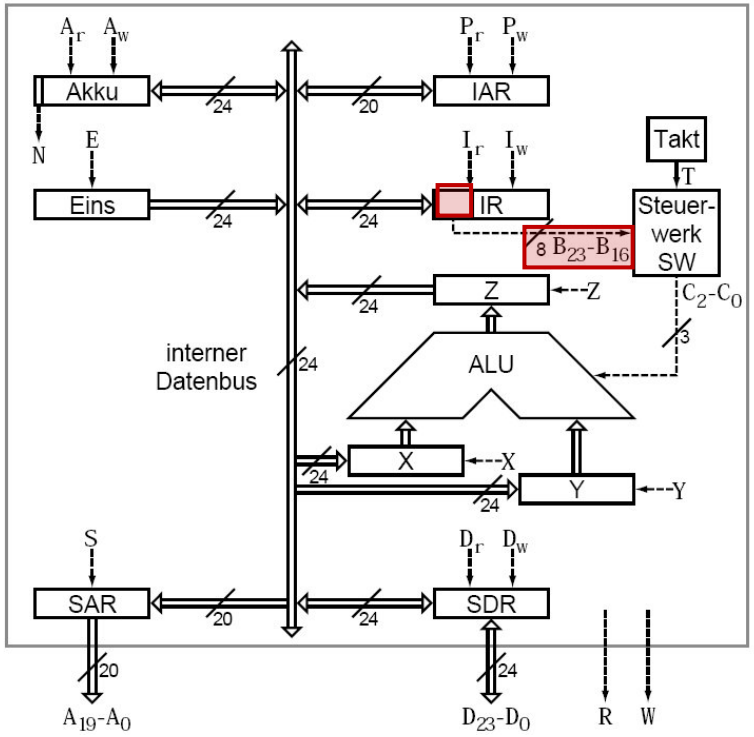
Holphase:
Mikroprogramm als Binärcode

Mikrobefehlsformat

A_r	A_w	X	Y	Z	E	P_r	P_w	I_r	I_w	D_r	D_w	S	C_2	C_1	C_0	R	W	0 0	Folgeadresse F
27			24				20				16				12			9 8	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	00	00000001
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	00	00000010
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	00	00000011
0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	00	00000100
0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	00	00000101

Fallstudie MIMA

Mikroprogrammsteuerung: Befehlszyklus



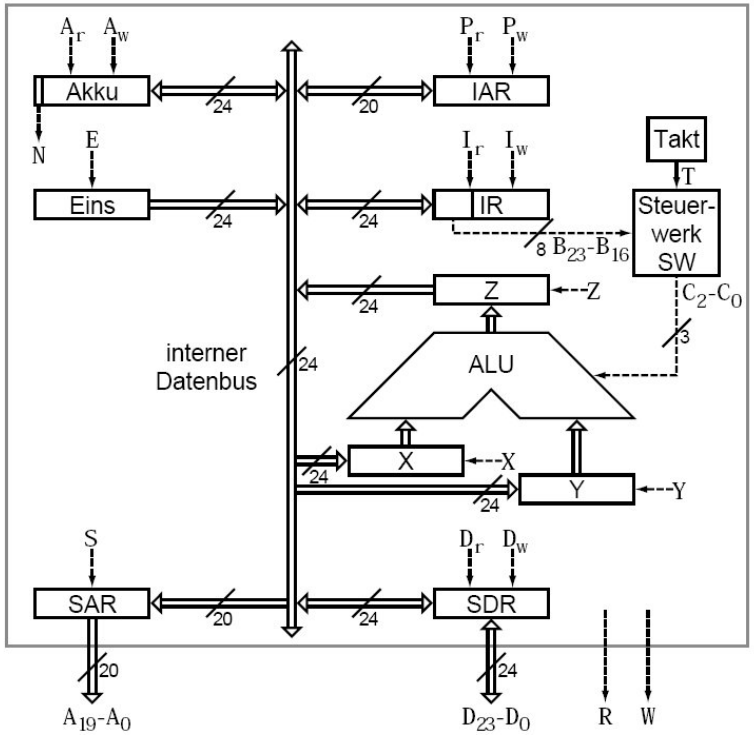
Dekodierphase

Einsprung in Mikroprogrammspeicher

Adresse wird aus Opcode (Bit $B_{23} – B_{16}$) des Maschinenbefehls generiert

Fallstudie MIMA

Mikroprogrammsteuerung: Befehlszyklus

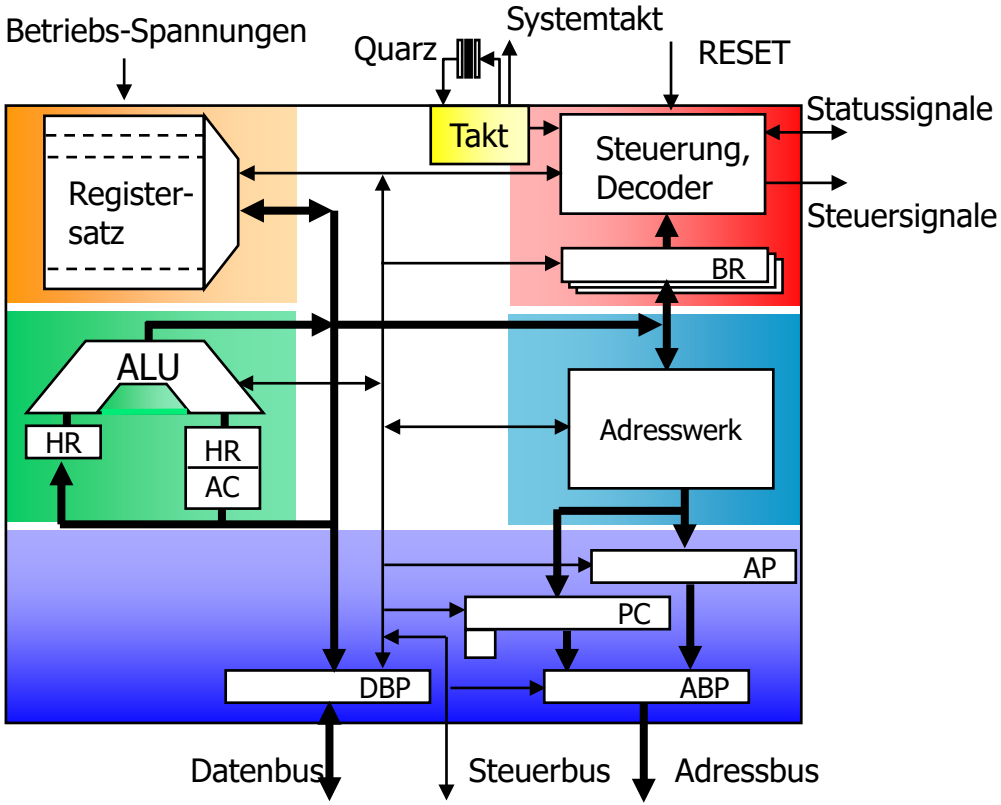


Ausführungsphase

Abarbeitung des ausgewählten Mikroproprogramms

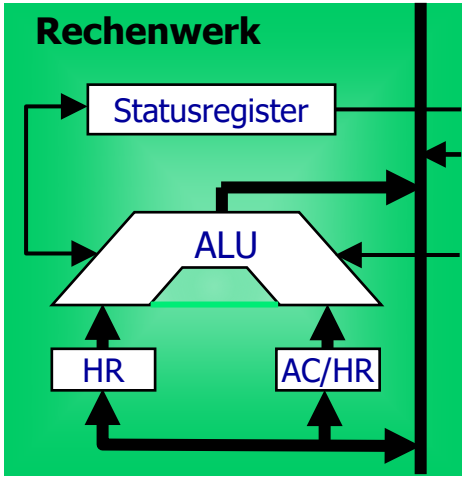
Nach Abarbeitung des ausgewählten Mikroprogramms wird das Mikroprogramm zum Holen Des nächsten Maschinenbefehls angesprungen.

Aufbau eines einfachen Mikroprozessors



Aufbau eines einfachen Mikroprozessors

- **Rechenwerk**
  - Ausführung von Berechnungen

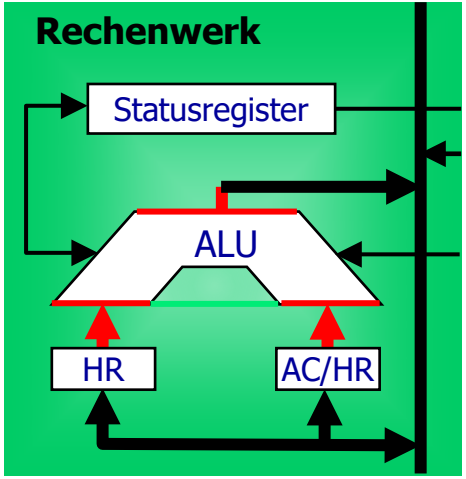


Aufbau eines einfachen Mikroprozessors

- **Rechenwerk**
  - **ALU (Arithmetische und logische Einheit)**
    - führt die vom Steuerwerk angestoßenen logischen und arithmetischen Operationen aus
  - **Hilfregister (HR) vor den ALU -Eingängen**
    - Zwischenspeichern von Operanden
  - **Akkumulator (AC)**
    - ausgezeichnetes Register
    - Speichern von Operanden / Ergebnissen
  - **Statusregister**
    - Festhalten der Statusinformationen von ALU -Berechnungen (z.B. Carry, Overflow, Zero, Sign, ...)
    - Statusinformationen dienen der Auswertung von Bedingungen.

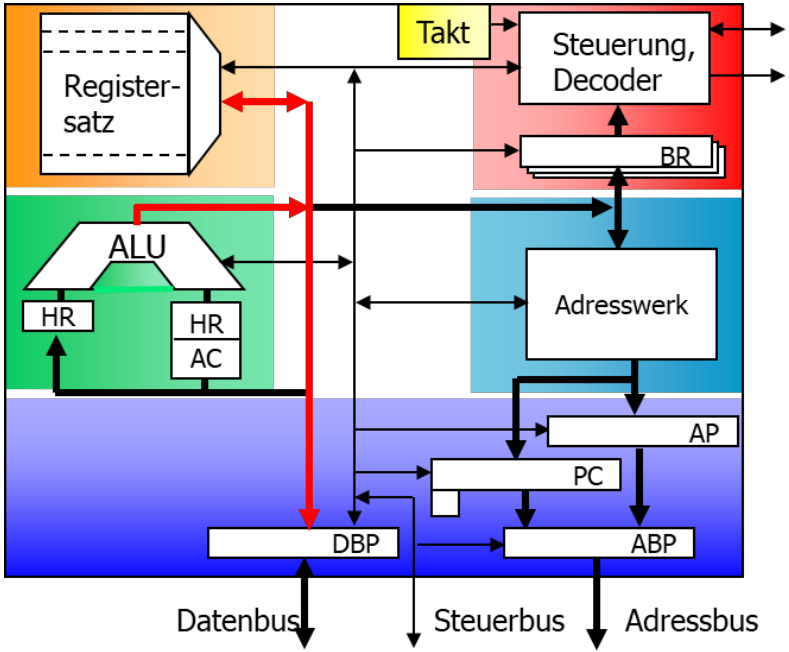
Aufbau eines einfachen Mikroprozessors

- **Rechenwerk:**
  - **ALU (Arithmetische und logische Einheit)**
    - 2 ALU-Eingänge (Operanden) und 1 ALU-Ausgang (Ergebnis).



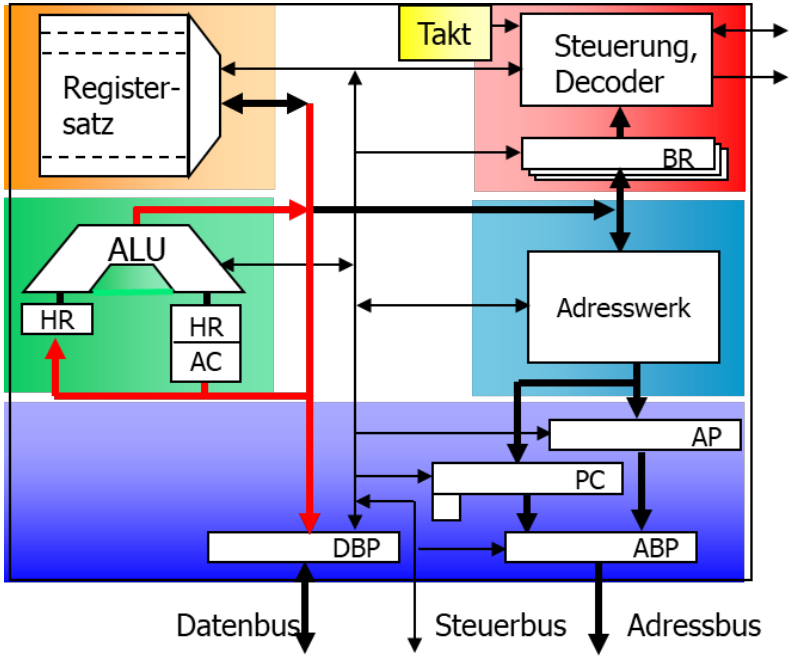
Aufbau eines einfachen Mikroprozessors

- **Rechenwerk**
  - **ALU (Arithmetische und logische Einheit)**
    - Ergebnisse:
      - werden entweder in Registern der Registerdatei gespeichert,
      - auf die Hilfsregister vor den ALU-Eingängen zurückgeführt oder
      - über den externen Datenbus an andere Systemkomponenten übertragen



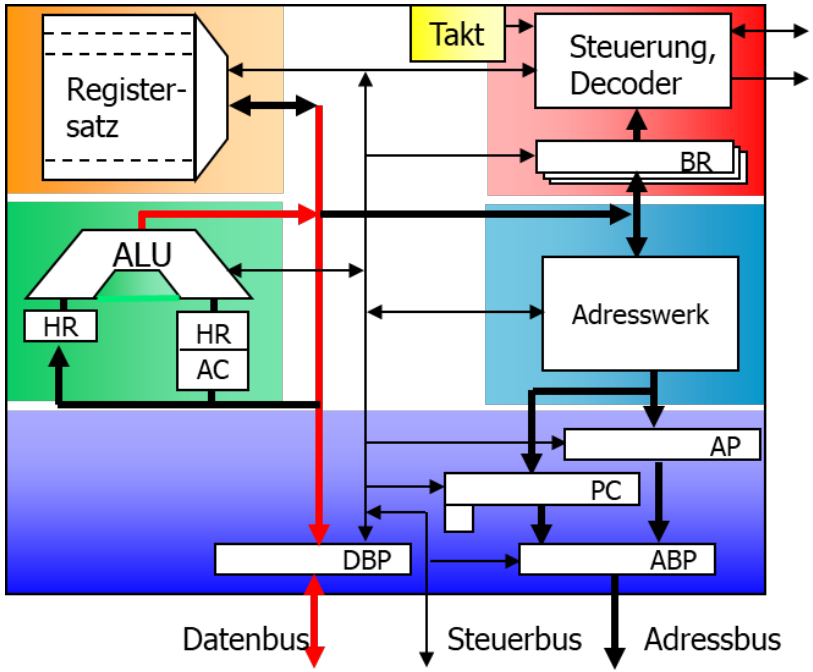
Aufbau eines einfachen Mikroprozessors

- **Rechenwerk**
  - **ALU (Arithmetische und logische Einheit)**
    - Ergebnisse:
      - werden entweder in Registern der Registerdatei gespeichert,
      - auf die Hilfsregister vor den ALU-Eingängen zurückgeführt oder
      - über den externen Datenbus an andere Systemkomponenten übertragen



Aufbau eines einfachen Mikroprozessors

- **Rechenwerk**
  - **ALU (Arithmetische und logische Einheit)**
    - Ergebnisse:
      - werden entweder in Registern der Registerdatei gespeichert,
      - auf die Hilfsregister vor den ALU-Eingängen zurückgeführt oder
      - über den externen Datenbus an andere Systemkomponenten übertragen



Aufbau eines einfachen Mikroprozessors

- **Rechenwerk: ALU (Arithmetische und logische Einheit)**
  - **Operationen**
    - **Arithmetische Operationen:**
      - Addieren ohne/mit Übertrag
      - Subtrahieren ohne/mit Übertrag
      - Inkrementieren/Dekrementieren
      - Multiplizieren ohne/mit Vorzeichen
      - Dividieren ohne/mit Vorzeichen
      - Komplementieren (Zweierkomplement)

Aufbau eines einfachen Mikroprozessors

- **Rechenwerk: ALU (Arithmetische und logische Einheit)**
  - **Operationen**
    - **Logische bitweise Verknüpfungen:**
      - Negation
      - UND
      - ODER
      - Antivalenz
      - …

Aufbau eines einfachen Mikroprozessors

- **Rechenwerk: ALU (Arithmetische und logische Einheit)**
  - **Operationen**
    - **Schiebe- und Rotations-Operationen:**
      - Links-Verschieben
      - Rechts-Verschieben
      - Links-Rotieren ohne Übertragsbit
      - Links-Rotieren durch das Übertragsbit
      - Rechts-Rotieren ohne Übertragsbit
      - Rechts-Rotieren durch das Übertragsbit

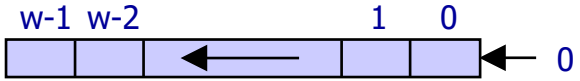
Aufbau eines einfachen Mikroprozessors

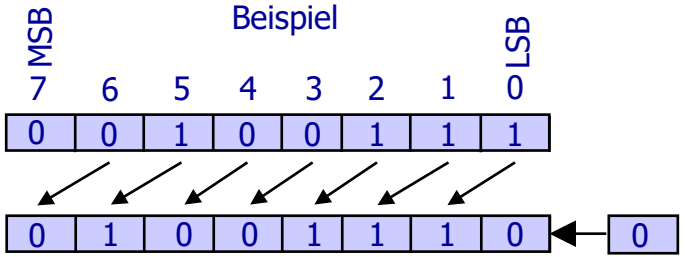
- **Rechenwerk: ALU (Arithmetische und logische Einheit)**
  - **Operationen**
    - **Schiebe- und Rotations-Operationen:**
      - **Logisches Verschieben**
        - Unabhängig von Schieberichtung und Vorzeichen werden die am jeweiligen Ende hinausgeschobenen Bits verworfen und Nullen nachgezogen.
      - **Arithmetisches Verschieben**
        - Das höchstwertige Bit ist das Vorzeichenbit (Zweierkomplementdarstellung).
        - Hinausgeschobene Bits gehen verloren.
        - Verschiebung nach links: am rechten Ende werden Nullen nachgezogen.
        - Verschiebung nach rechts: Kopien des Vorzeichenbits werden an der Vorzeichenstelle eingeschoben (*sign propagation*);

Aufbau eines einfachen Mikroprozessors

- **Rechenwerk: ALU (Arithmetische und logische Einheit)**
  - **Operationen**
    - **Schiebe-Operationen:**
      - **Linksschieben** entspricht der Multiplikation mit 2

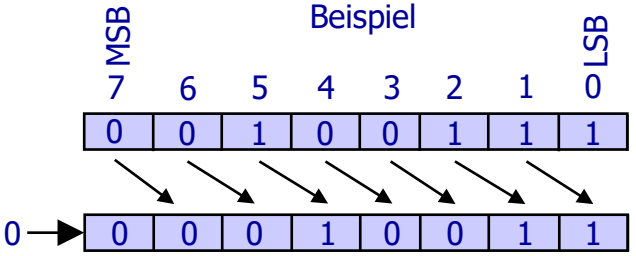
logisches und arithmetisches Verschieben nach links





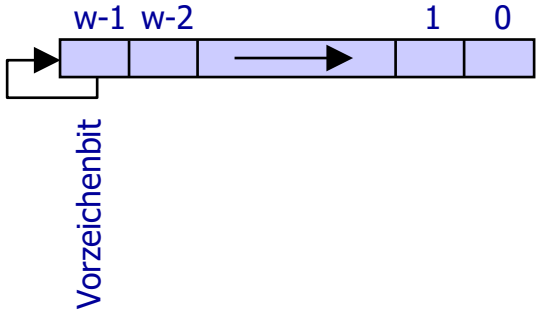
Aufbau eines einfachen Mikroprozessors

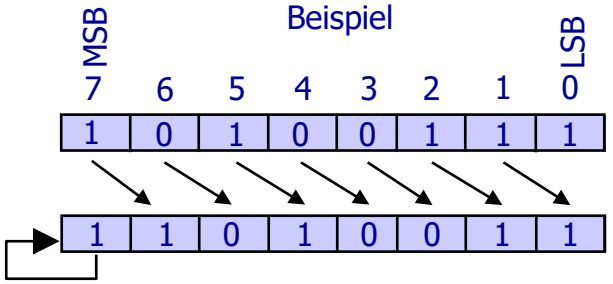
- **Rechenwerk: ALU (Arithmetische und logische Einheit)**
  - **Operationen**
    - **Schiebe-Operationen:**
      - **Logischen Rechtschieben** entspricht die Division durch 2
        - Gilt für positive Zahlen.



Aufbau eines einfachen Mikroprozessors

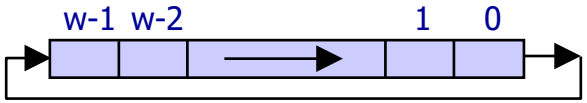
- **Rechenwerk: ALU (Arithmetische und logische Einheit)**
  - **Operationen**
    - **Schiebe-Operationen:**
      - **Arithmetisches Rechtschieben**
        - Negativen Zahlen (Zweierkomplementdarstellung): zur Vorzeichenerhaltung wird das höchstwertigste Bit (Vorzeichenbit) in sich selbst zurückgeführt.

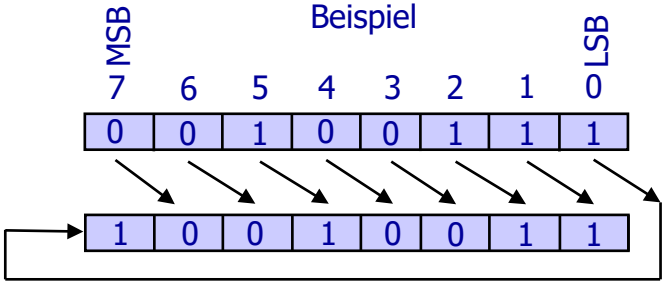




Aufbau eines einfachen Mikroprozessors

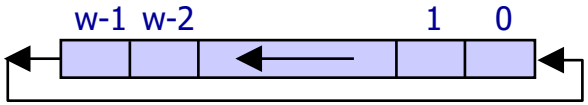
- **Rechenwerk: ALU (Arithmetische und logische Einheit)**
  - **Operationen**
    - **Rotations-Operationen:**
      - Register wird als geschlossene Bitkette betrachtet
      - **Rotation nach rechts:**

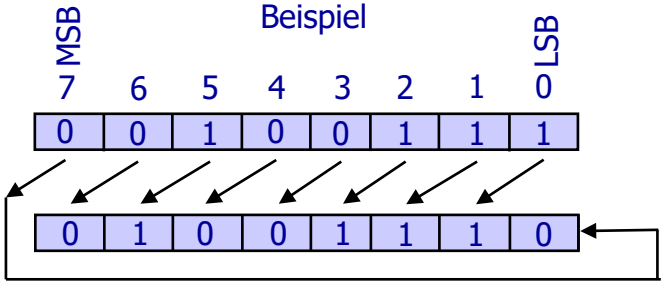




Aufbau eines einfachen Mikroprozessors

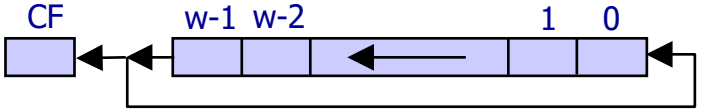
- **Rechenwerk: ALU (Arithmetische und logische Einheit)**
  - **Operationen**
    - **Rotations-Operationen:**
      - Register wird als geschlossene Bitkette betrachtet
      - **Rotation nach links:**

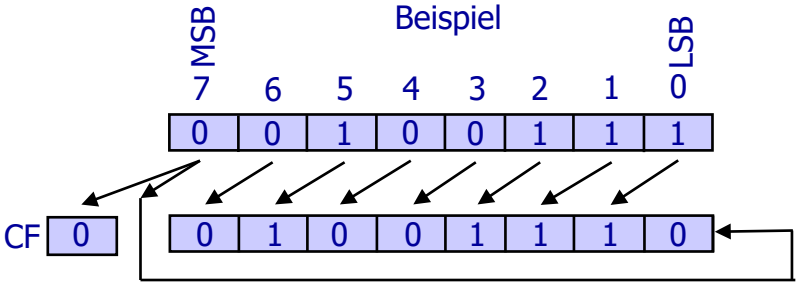




Aufbau eines einfachen Mikroprozessors

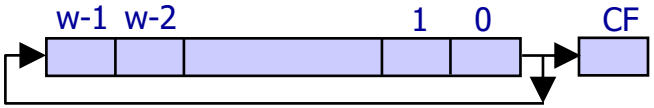
- **Rechenwerk: ALU (Arithmetische und logische Einheit)**
  - **Operationen**
    - **Rotations-Operationen:**
      - Carry-Flag kann wahlweise mitbenutzt oder als zusätzliches Bit mit einbezogen werden
      - **Rotation nach links mit Carry-Flag:**

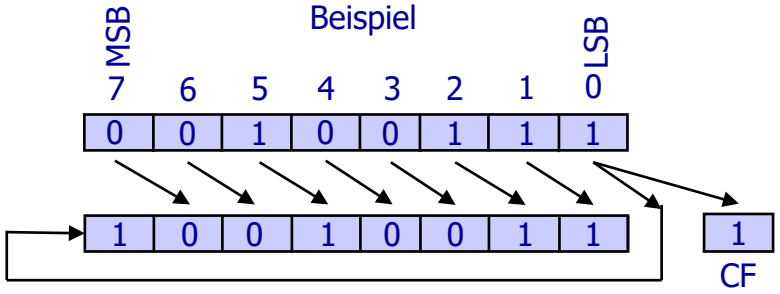




Aufbau eines einfachen Mikroprozessors

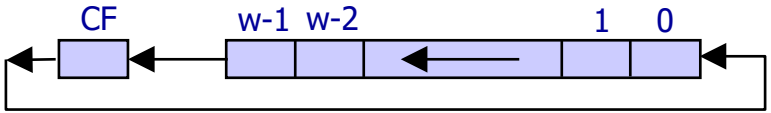
- **Rechenwerk: ALU (Arithmetische und logische Einheit)**
  - **Operationen**
    - **Rotations-Operationen:**
      - Carry-Flag kann wahlweise mitbenutzt oder als zusätzliches Bit mit einbezogen werden
      - **Rotation nach rechts mit Carry-Flag:**

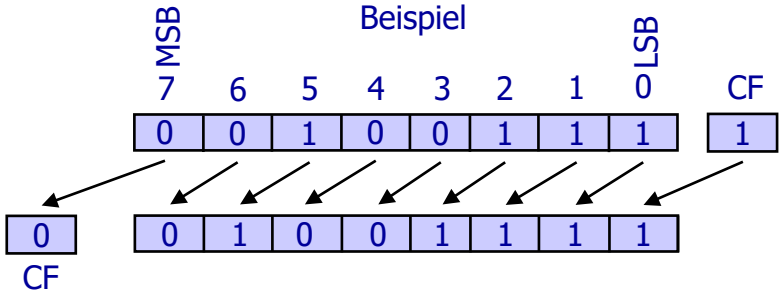




Aufbau eines einfachen Mikroprozessors

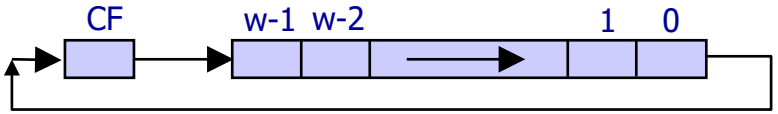
- **Rechenwerk: ALU (Arithmetische und logische Einheit)**
  - **Operationen**
    - **Rotations-Operationen:**
      - Carry-Flag kann wahlweise mitbenutzt oder als zusätzliches Bit mit einbezogen werden
      - **Rotation nach links mit Carry-Flag als zusätzliches Bit:**

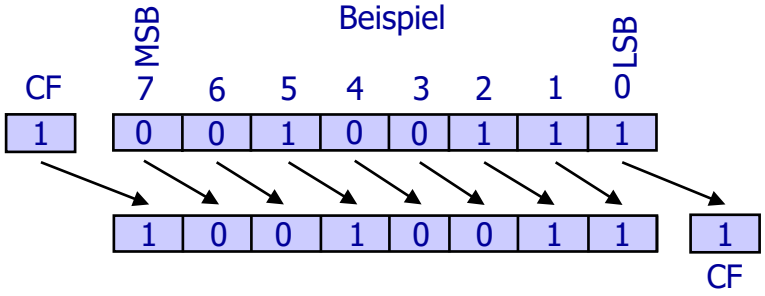




Aufbau eines einfachen Mikroprozessors

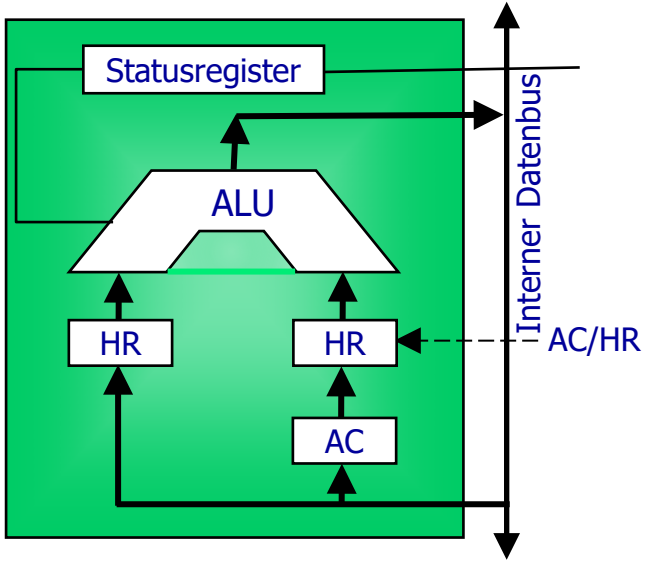
- **Rechenwerk: ALU (Arithmetische und logische Einheit)**
  - **Operationen**
    - **Rotations-Operationen:**
      - Carry-Flag kann wahlweise mitbenutzt oder als zusätzliches Bit mit einbezogen werden
      - **Rotation nach rechts mit Carry-Flag als zusätzliches Bit:**





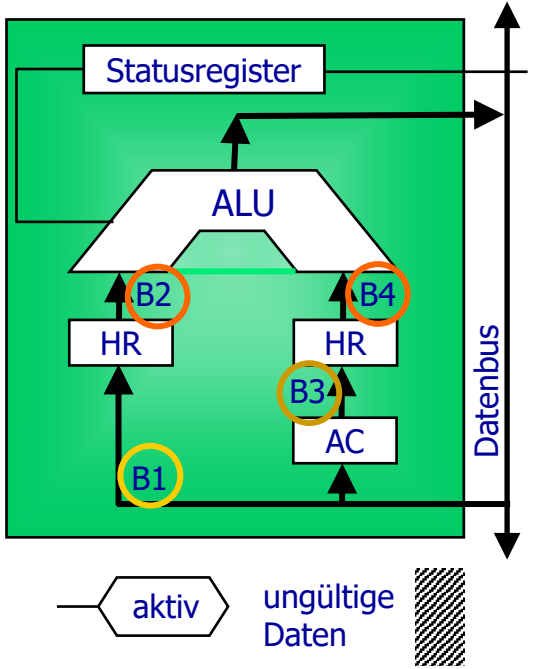
Aufbau eines einfachen Mikroprozessors

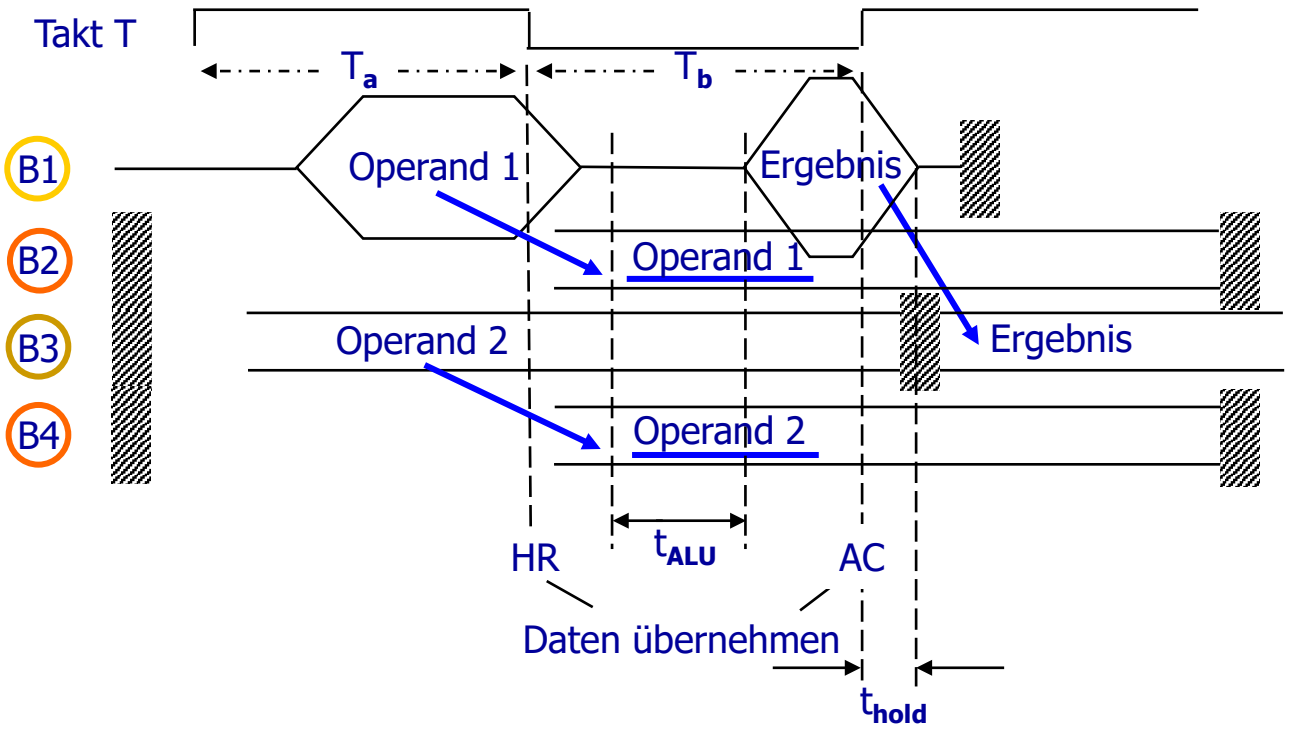
- **Rechenwerk**
  - **Hilfsregister / Akkumulator**
    - **Variante 1:**
      - Akkumulator ist zweigeteilt:
        - eigentliches Akkumulator-Register AC
        - nachgeschaltetes Hilfsregister HR (Latch)



Aufbau eines einfachen Mikroprozessors

- **Rechenwerk**
  - **Zeitverhalten**



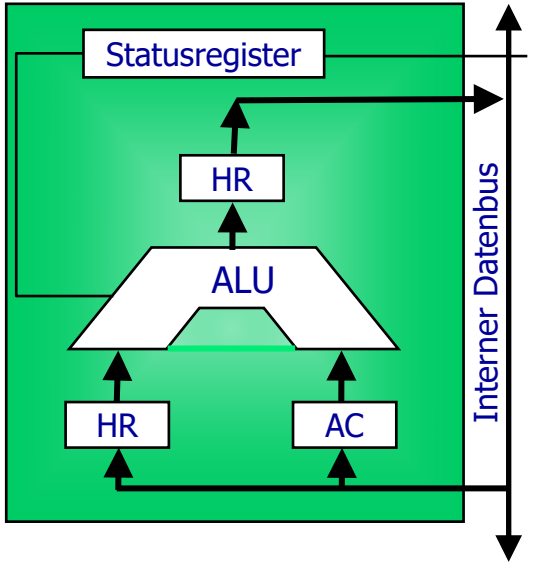


Aufbau eines einfachen Mikroprozessors

- **Rechenwerk**
  - **Zeitverhalten**
    - Während der positiven Takthälfte von T liegen auf B1 und B3 die Operanden.
    - Diese werden mit der negativen Taktflanke in die Hilfsregister übernommen
    - Danach stehen diese auf B2 und B4 stabil zur Verfügung, die ALU berechnet in einer Ausführungszeit t_{ALU} 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 ALU-Ausgang (Hasards, Wettläufe) sofort wieder auf den ALU-Eingang auswirken.
      - Es wäre ein asynchrones Schaltwerk mit allen bekannten Problemen und Fehl-Ergebnissen entstehen.

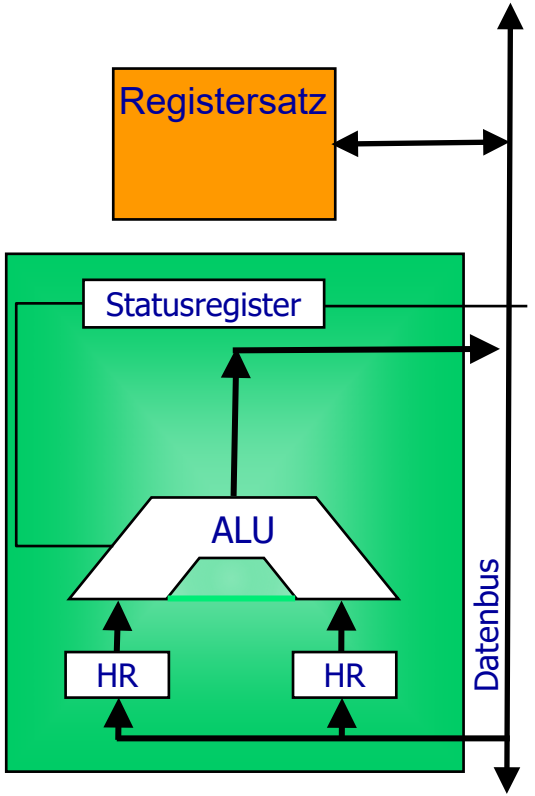
Aufbau eines einfachen Mikroprozessors

- **Rechenwerk**
  - **Hilfsregister / Akkumulator**
    - **Variante 2:**
      - Hilfsregister des Akkumulators wird an den ALU-Ausgang verlegt;
      - Vorteil: ALU-Operationen können ausgeführt werden, ohne dass der Inhalt des Akkumulators sich ändert.
      - Hauptsächlich in älteren 8-Bit-Prozessoren, bei denen auch die Adressberechnung in der ALU des Rechenwerkes durchgeführt wird.



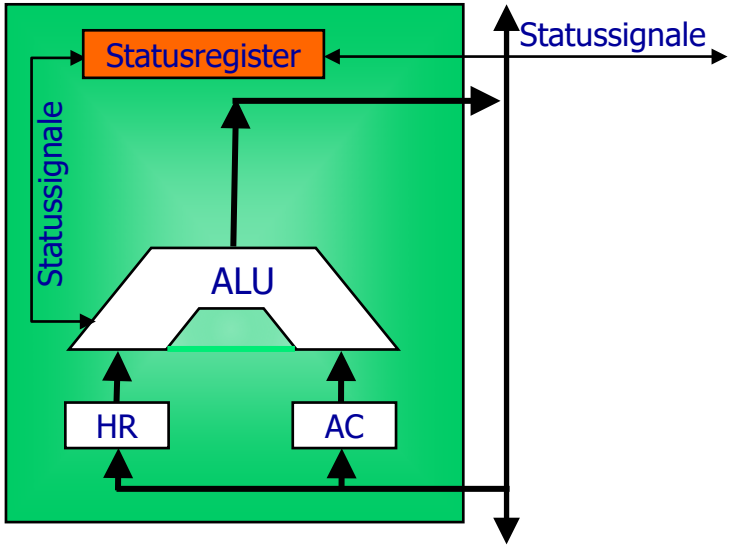
Aufbau eines einfachen Mikroprozessors

- **Rechenwerk**
  - **Hilfsregister / Akkumulator**
    - **Variante 3:**
      - Rechenwerk ohne Akkumulator, d. h. kein ausgezeichnetes Register mehr;
      - Jedes Register der Registerdatei kann verwendet werden.
      - Üblich bei Prozessoren mit Registerdateien



Aufbau eines einfachen Mikroprozessors

- **Rechenwerk**
  - **Statusregister**
    - Speichert Statusinformationen, die bei ALU-Berechnungen generiert werden.



Aufbau eines einfachen Mikroprozessors

- **Rechenwerk**
  - **Statusregister**

    - **AC: Hilfsübertragsbit (Auxiliary Carry):**
      - Übertrag von Bit 3 in Bit 4 des Ergebnisses (BCD-Arithmetik);
    - **CF: Übertragsbit (Carry Flag):**
      - Übertrag aus dem höchstwertigen Bit bei einer Addition oder einer Subtraktion (Borrow);
    - **Z: Nullbit (Zero Flag):**
      - Zeigt an, dass das Ergebnis der ausgeführten ALU-Operation gleich 0 ist;
    - **EF: Even Flag:**
      - Zeigt an, dass das Ergebnis der ausgeführten ALU-Operation eine gerade Zahl ist.

Aufbau eines einfachen Mikroprozessors

- **Rechenwerk**
  - **Statusregister**

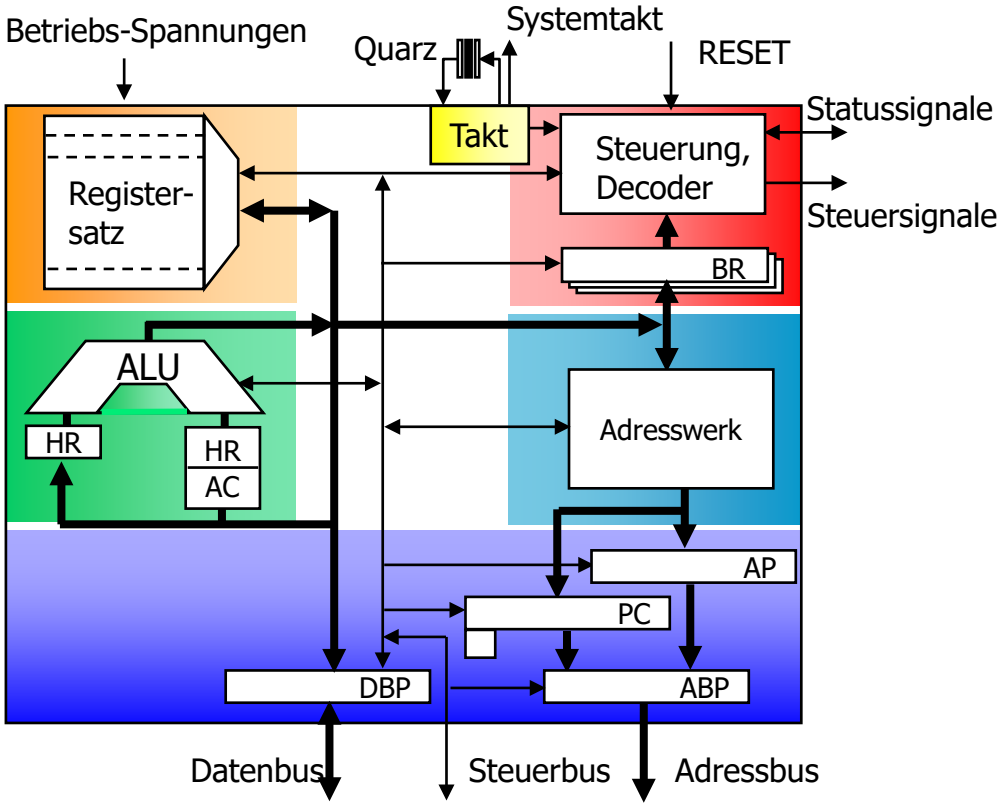
    - **S: Vorzeichenbit (Sign Flag):**
      - Zeigt an, dass das Ergebnis der ausgeführten ALU-Operation negativ ist.
      - Ergibt sich aus dem höchstwertigen Bit (MSB) eines Operanden.
    - **OF: Überlaufbit (Overflow Flag):**
      - Überschreitung des Zahlenbereichs im Zweierkomplement
    - **P: Paritätsbit (Parity Flag):**
      - Zeigt die ungerade Parität des Ergebnisses an;

Aufbau eines einfachen Mikroprozessors

- **Rechenwerk**
  - **Statusregister**
    - Die Werte der Statusbits können Einfluss auf den Kontrollfluss eines Programms haben.
    - **Beispiel: IA-32 / Intel ® 64 Befehlssatz**
      - Verzweigungen (Control Transfer Instructions)
        - `JZ, JNZ:` Jump if zero, Jump if not zero
        - `JC, JNC:` Jump if carry, Jump if not carry
        - `JO, JNO:` Jump if overflow, Jump if not overflow
        - `JS, JNS:` Jump if sign (negative), JNS Jump if not sign (non-negative).
        - …

Aufbau eines einfachen Mikroprozessors

- **Register**



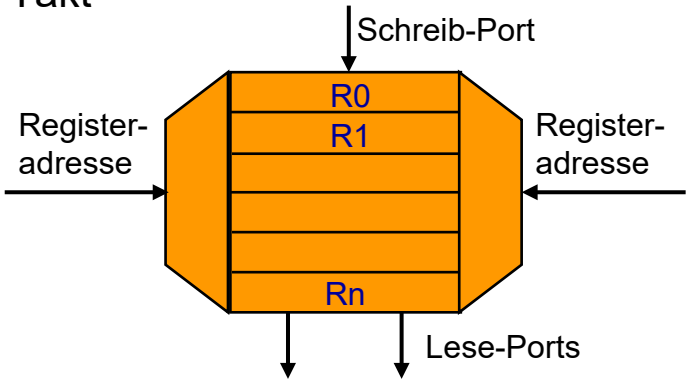
Aufbau eines einfachen Mikroprozessors

- **Register**
  - Speicherkomponenten, auf die der Prozessor besonders schnell zugreifen kann;
  - Enthalten häufig benutzten Informationen.

  - Die Anzahl der Register in einem Prozessor kann von wenigen bis mehrere Hundert variieren.
  - Die Breite der einzelnen Register ist üblicherweise eine Zweierpotenz: 8, 16, 32, …, 512.
    - Bei Spezialprozessoren kann es auch andere Registerbreiten geben.

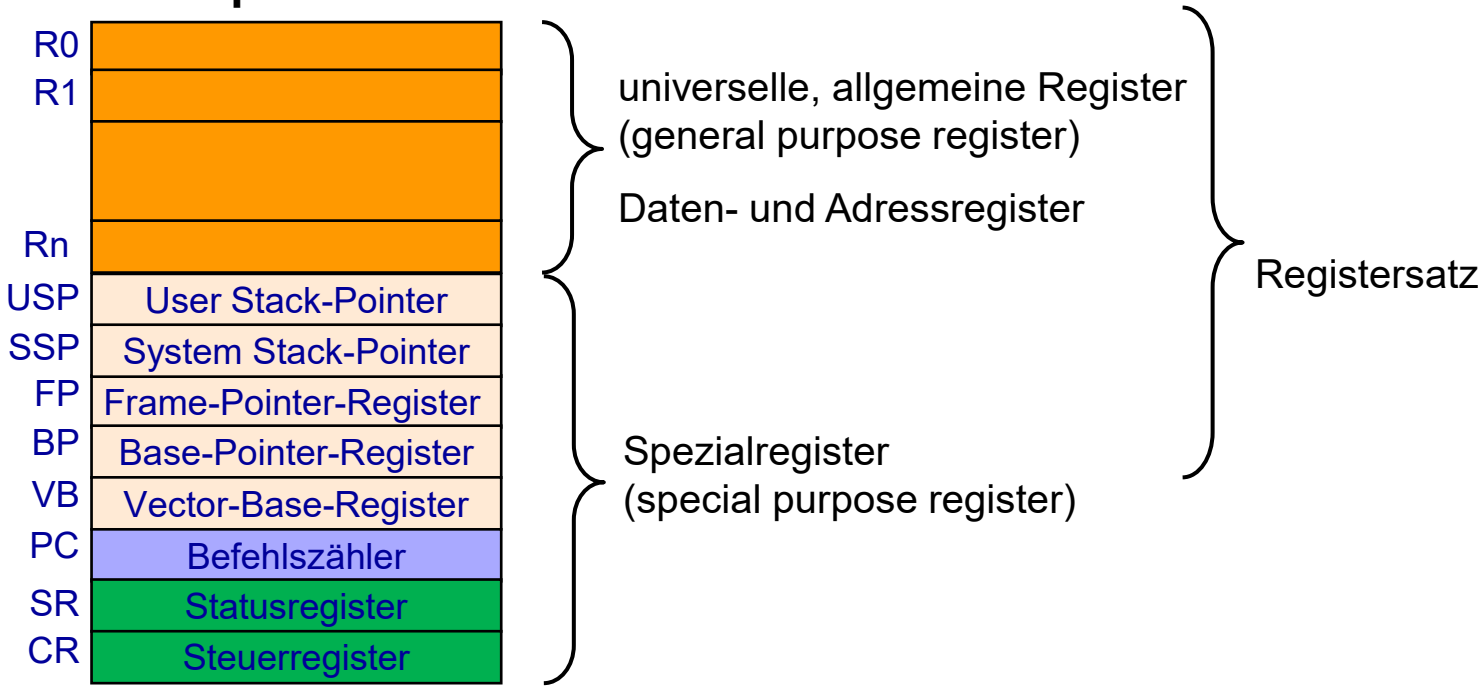
Aufbau eines einfachen Mikroprozessors

- **Register**
  - **Organisation**
    - Die Register eines Prozessors können zu einem **Registersatz (Registerdatei)** zusammengefasst werden.
    - Die Register werden über eine eindeutige Nummer (**Registeradresse** Ri) adressiert.
    - Die Registerdatei verfügt über mehrere Schreib- und Lese-Ports, über die Werte in die Register geschrieben oder aus den Registern gelesen werden.
      - Lesen und Schreiben mehrerer Register in einem Takt



Aufbau eines einfachen Mikroprozessors

- **Register**
  - **Organisation: Beispiel**

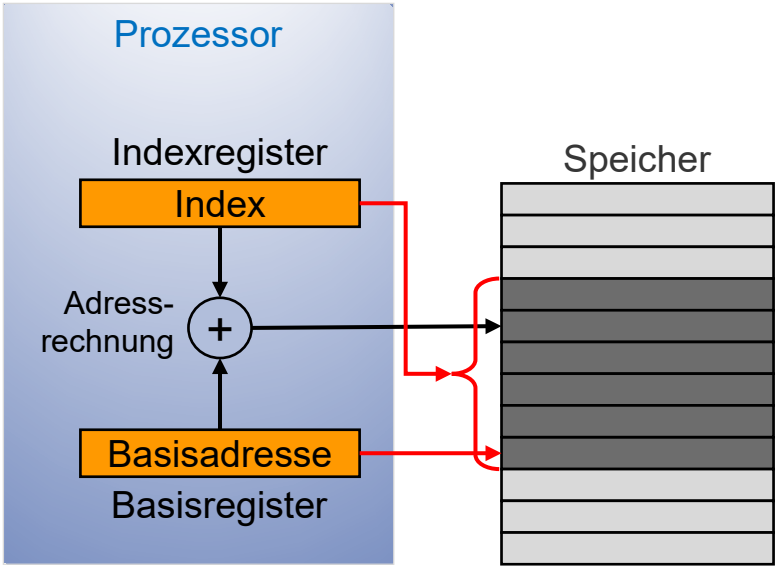


Aufbau eines einfachen Mikroprozessors

- **Register**
  - **Universelle, allgemeine Register, Allzweckregister (general purpose register)**
    - **Datenregister:**
      - Zwischenspeichern von Operanden / Ergebnissen
      - Unterscheidung zwischen
        - Datenregister für skalare Werte (Integer-Werte) vor der Arithmetischen und Logischen Einheit und
        - Datenregister für Gleitkomma-Werte vor einer Gleitkomma-Einheit
    - **Adressregister:**
      - Speichern von Adressinformationen von Befehlen oder Daten im Hauptspeicher
      - Adressrechnung
    - Viele Prozessoren haben keine explizite Unterscheidung.

Aufbau eines einfachen Mikroprozessors

- **Register**
  - **Adressregister:**
    - **Basisregister**
      - enthält die Anfangsadresse eines Bereichs im Speicher
      - Adresse bleibt während der Bearbeitung des Speicherbereichs unverändert
    - **Indexregister**
      - enthält eine Distanz (Offset, Displacement) zu einer Basisadresse
      - dient zur Auswahl eines bestimmten Datums des Speicherbereichs



Aufbau eines einfachen Mikroprozessors

- **Register**
  - **Spezialregister:**
    - **Befehlszähler (program counter, PC)**
    - **Befehlsregister (instruction register, IR, BR)**
    - **Steuerregister (control register, CR) und Statusregister (status Register, SR)**
    - **Vektorbasisregister (vector-base-register, VB)**
      - Enthält die Basisadresse der Vektortabelle
      - Vektortabelle enthält die Unterbrechungsvektoren
      - Unterbrechungsvektor ist die Adresse der Unterbrechungsbehandlungsroutine, die bei einer Ausnahmesituation (z. B. Alarm, Interrupt) angesprungen wird.

Aufbau eines einfachen Mikroprozessors

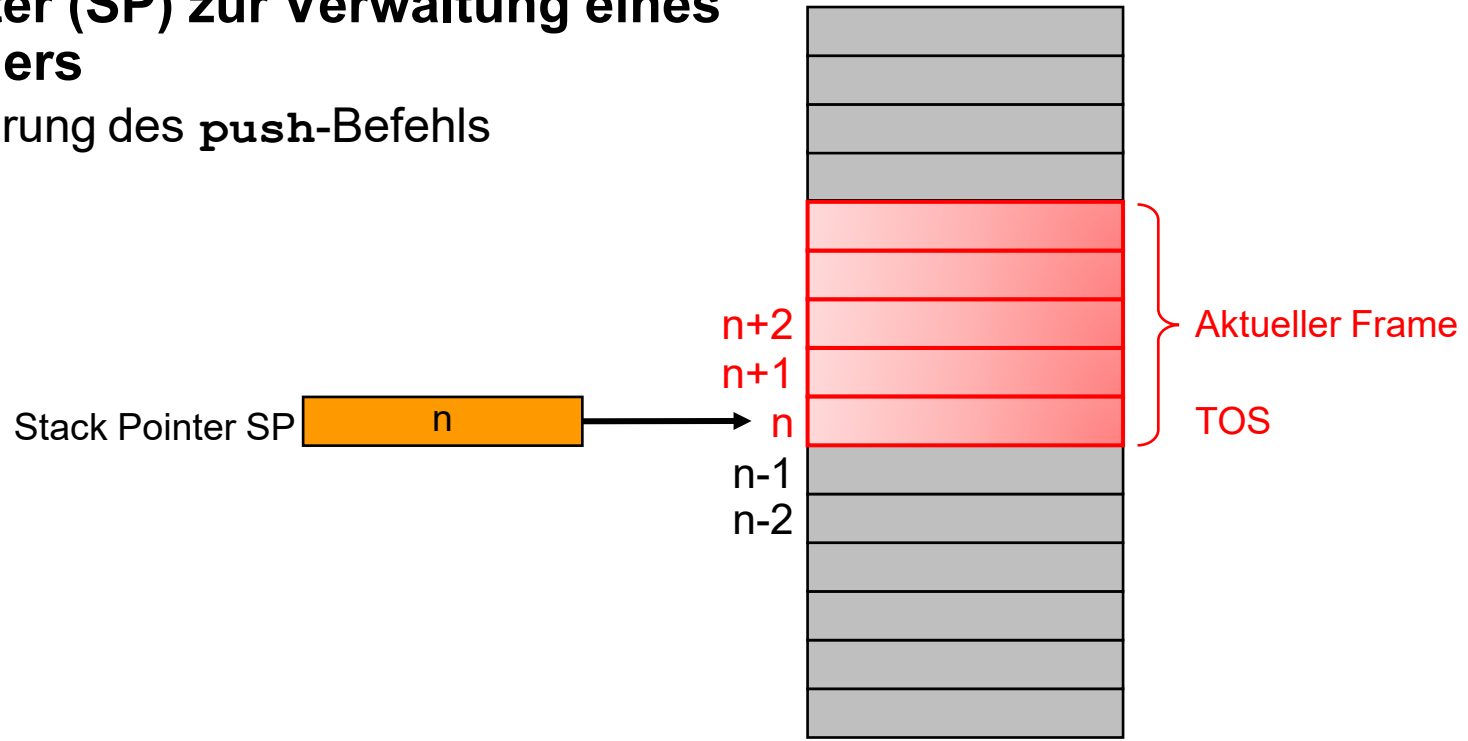
- **Register**
  - **Spezialregister:**
    - **Stackregister (Kellerzeiger, Stapelzeiger, Stack Pointer SP):**
      - Enthält die Adresse des obersten Kellerelements im Speicher (**top of stack, TOS**);
      - Dient zur Verwaltung eines **Laufzeitkellers, Stacks**
      - **User Stack-Pointer (USP)** für den **User-Stack (Anwender-Stack)**
        - Wird von Anwenderprogrammen für Unterprogrammaufrufe verwendet.
        - Ablage des Inhalts des Befehlszählers sowie der Übergabe-, Rückgabe-Parameter und der lokalen Variablen eines Unterprogramms
      - **System Stack-Pointer (SSP)** wird vom Betriebssystem für den **System-Stack** verwendet:
        - Sicherung des Prozessor-Status
        - Prozesswechsel

Aufbau eines einfachen Mikroprozessors

- **Register**
  - **Spezialregister:**
    - **Stackregister (Kellerzeiger, Stapelzeiger, Stack Pointer SP):**
      - Spezielle Befehle zur Datenübertragung in den bzw. aus dem Keller
      - **`PUSH:`** Push onto stack;
        - Inhalt des Stackregisters wird dekrementiert
        - Inhalt des Quelloperanden wird auf das oberste Kellerelement übertragen.
      - **`POP:`** Pop off the stack;
        - Inhalt des obersten Kellerelements wird in den Zieloperanden geladen.
        - Inhalt des Stackregisters wird inkrementiert
      - Alternativ: **Pre-Dekrement / Post-Inkrement Technik** ein
        - Vor bzw. nach dem Zugriff auf das Stackregister wird dieses dekrementiert bzw. inkrementiert.

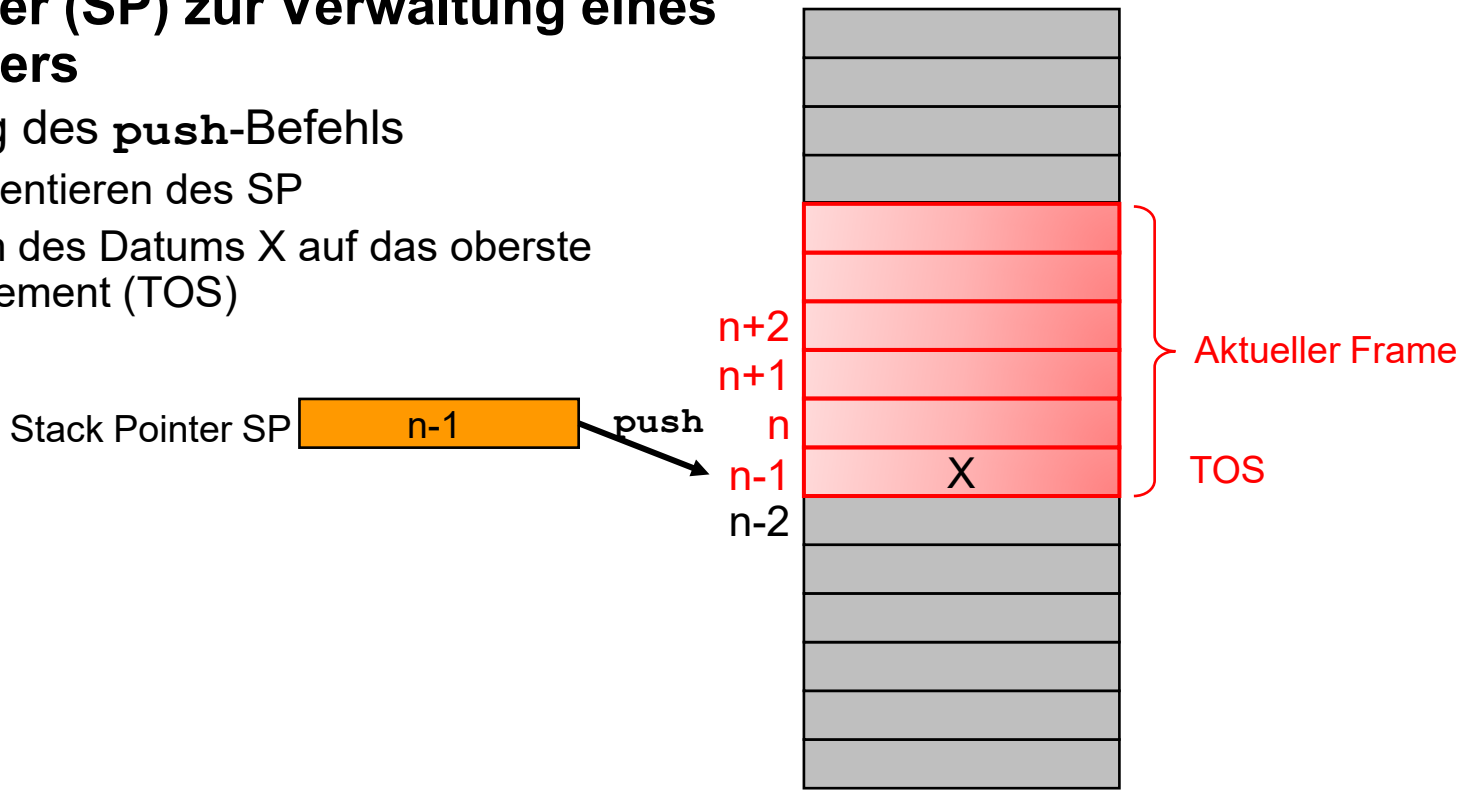
Aufbau eines einfachen Mikroprozessors

- **Stack Pointer (SP) zur Verwaltung eines Laufzeitkellers**
  - Vor Ausführung des **`push`**-Befehls



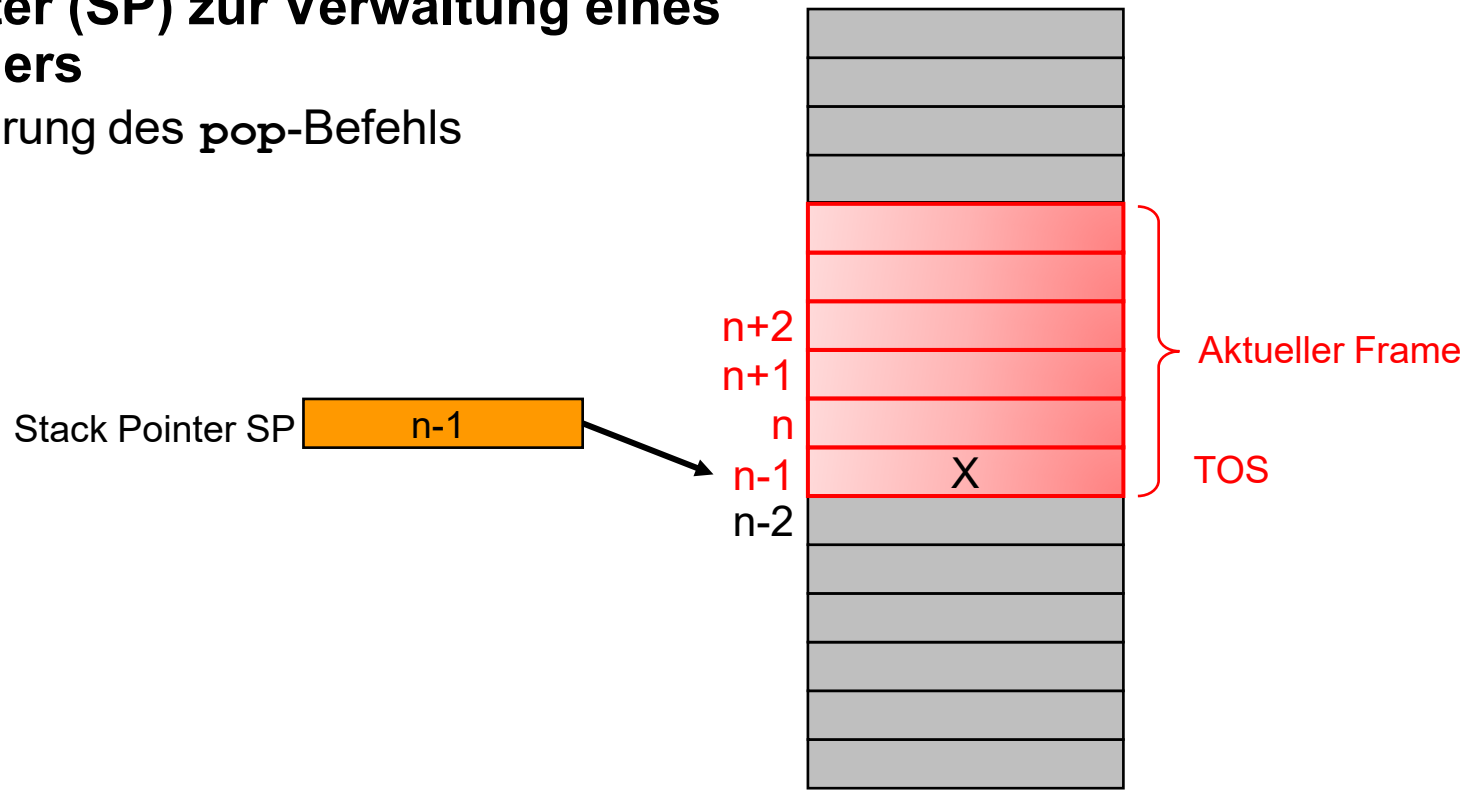
Aufbau eines einfachen Mikroprozessors

- **Stack Pointer (SP) zur Verwaltung eines Laufzeitkellers**
  - Ausführung des **`push`**-Befehls
    1. Dekrementieren des SP
    2. Ablegen des Datums X auf das oberste Kellerelement (TOS)



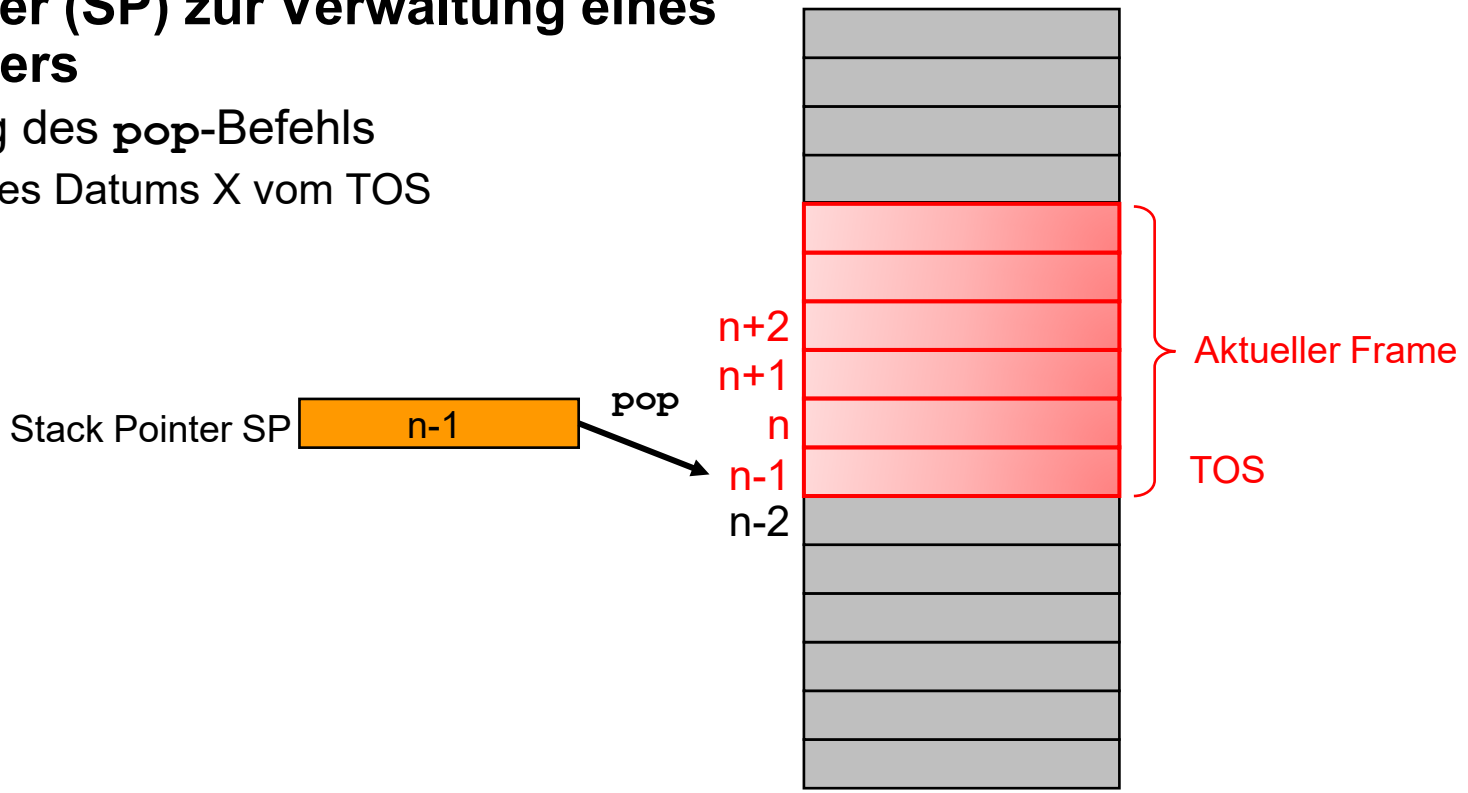
Aufbau eines einfachen Mikroprozessors

- **Stack Pointer (SP) zur Verwaltung eines Laufzeitkellers**
  - Vor Ausführung des `pop`-Befehls



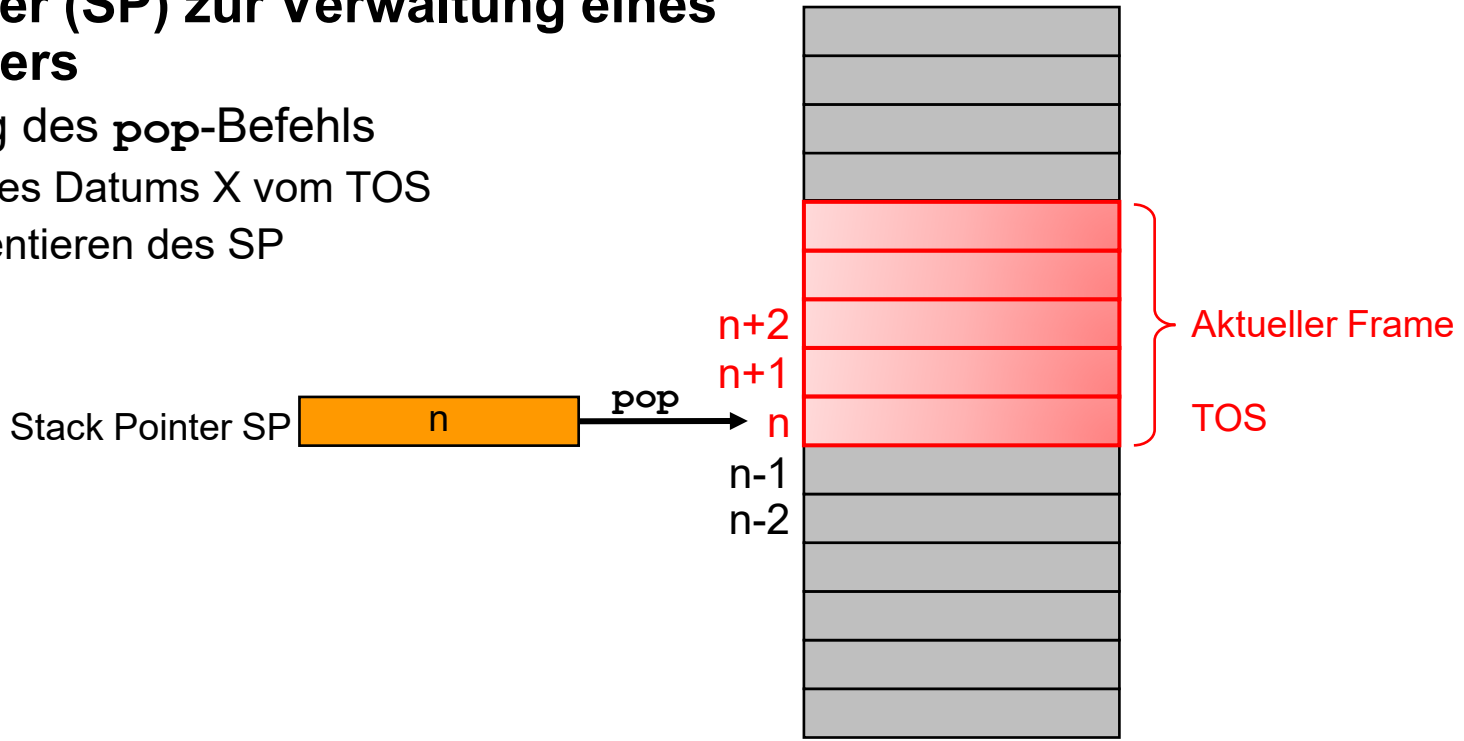
Aufbau eines einfachen Mikroprozessors

- **Stack Pointer (SP) zur Verwaltung eines Laufzeitkellers**
  - Ausführung des `pop`-Befehls
    1. Holen des Datums X vom TOS



Aufbau eines einfachen Mikroprozessors

- **Stack Pointer (SP) zur Verwaltung eines Laufzeitkellers**
  - Ausführung des `pop`-Befehls
    1. Holen des Datums X vom TOS
    2. Inkrementieren des SP

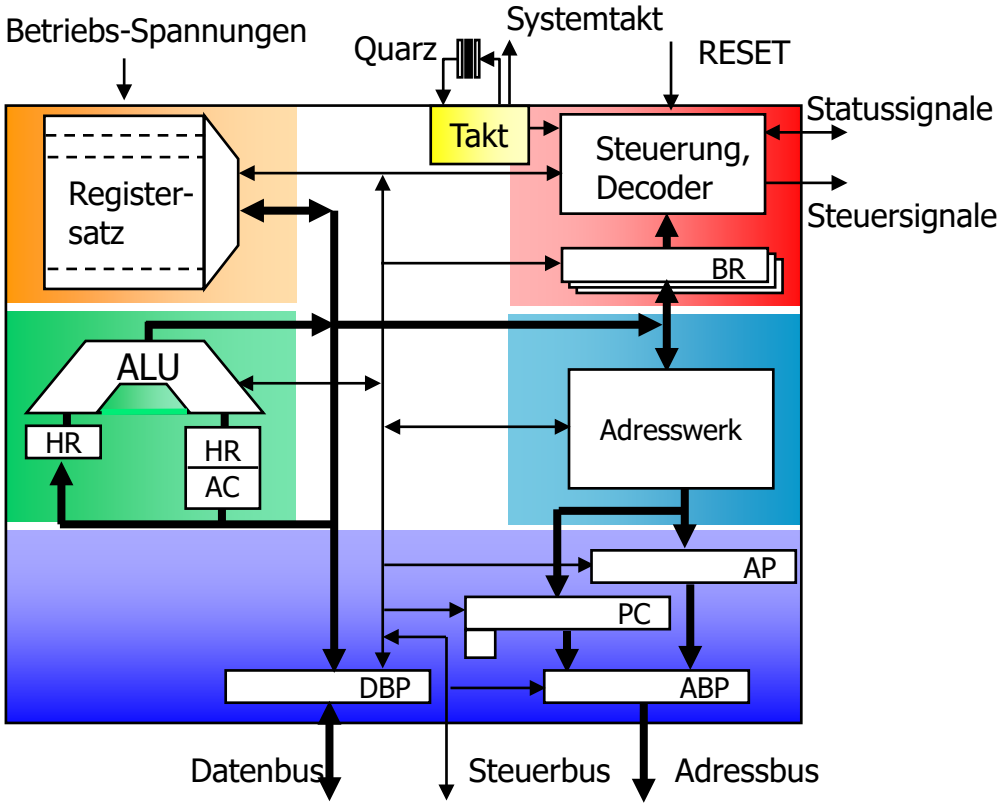


Aufbau eines einfachen Mikroprozessors

- **Rechenwerk:**
  - **Register mit Zusatzfunktionen:**
    - Inkrementieren / Dekrementieren
    - auf Null setzen
    - Inhalt verschieben (Schieberegister)

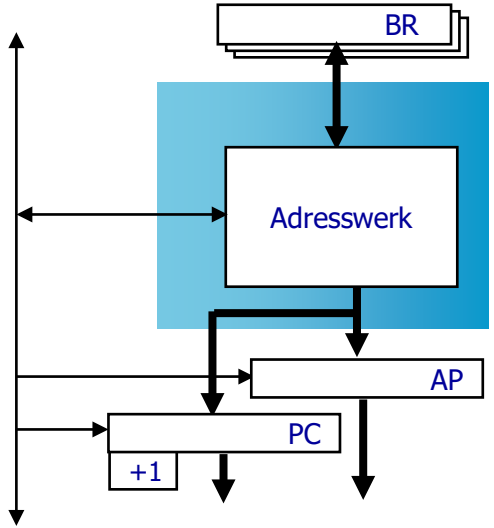
Aufbau eines einfachen Mikroprozessors

- **Adresswerk**



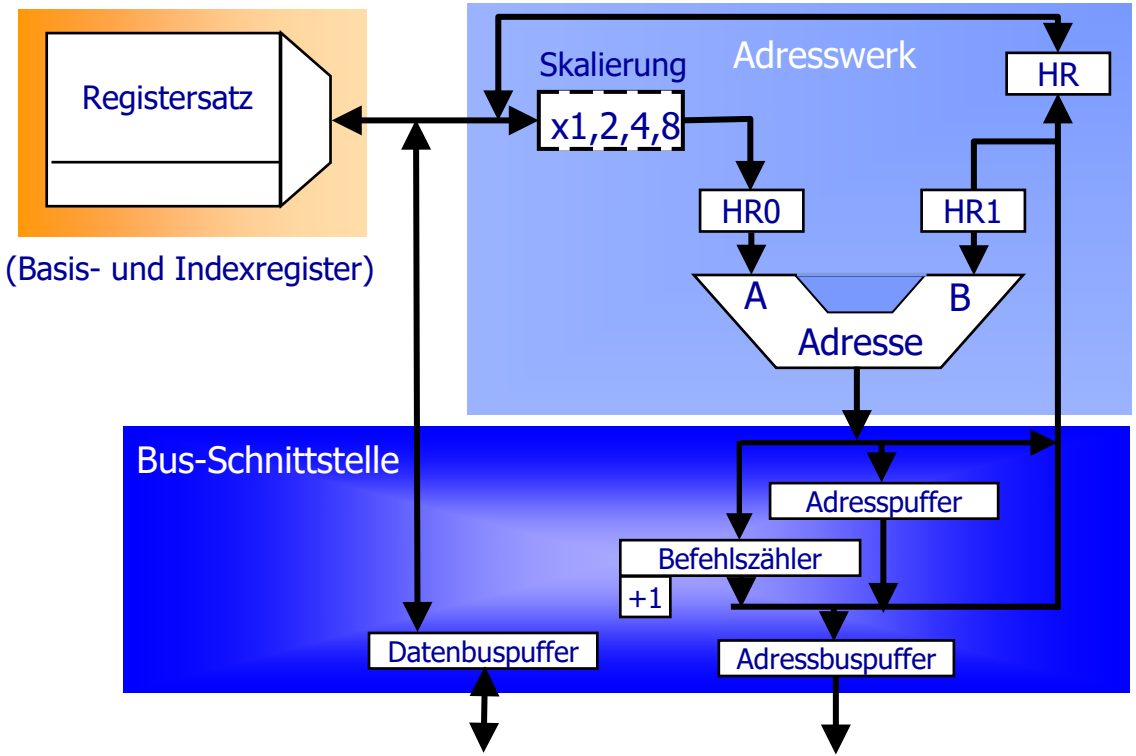
Aufbau eines einfachen Mikroprozessors

- **Adresswerk**
  - Führt die vom Steuerwerk angestoßenen Adressberechnungen für Befehle und Daten durch
  - Eigenständiges Adresswerk ermöglicht unabhängig vom Rechenwerk Adressberechnungen durchzuführen
    - Operationen im Rechenwerk und Adressberechnung können parallel und ohne Konflikte durchgeführt werden



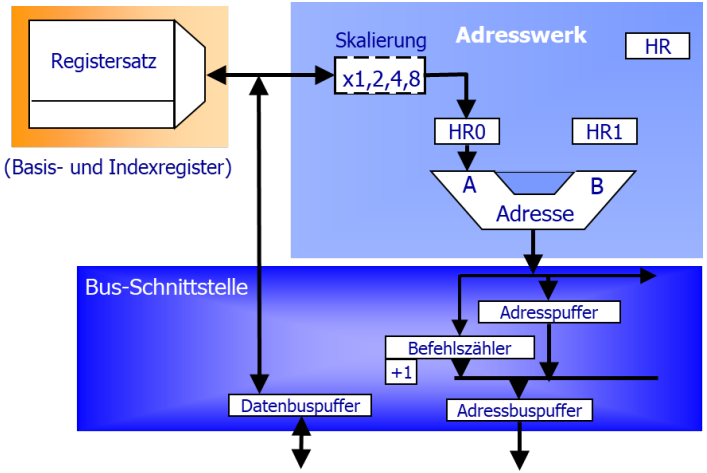
Aufbau eines einfachen Mikroprozessors

- **Adresswerk**
  - **Aufbau**



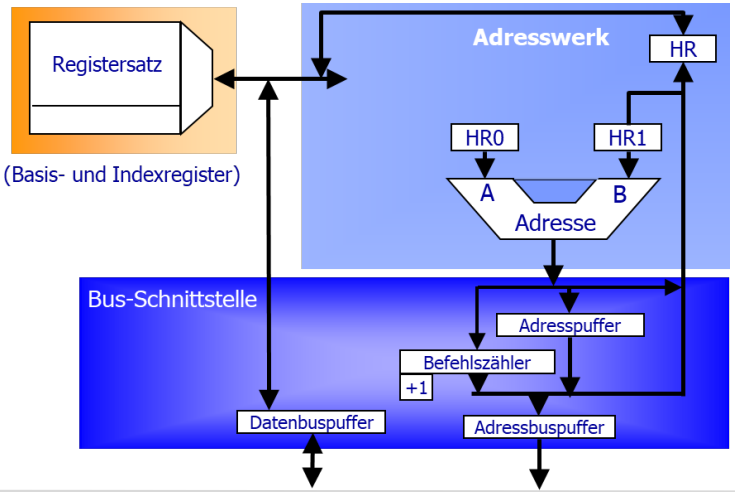
Aufbau eines einfachen Mikroprozessors

- **Adresswerk: Aufbau**
  - Addierer mit 2 Eingängen und einem Ausgang
    - **Eingang A:**
      - Es können die Inhalte von Registern des Registersatzes anliegen.
        - Beispiel: Adresse aus Basis- oder Indexregister
      - Es kann der Inhalt des Datenbuspuffers anliegen.
        - Beispiele: absolute Adresse in einem Befehl
        - absolute Adressdistanz zu einem Basisregister



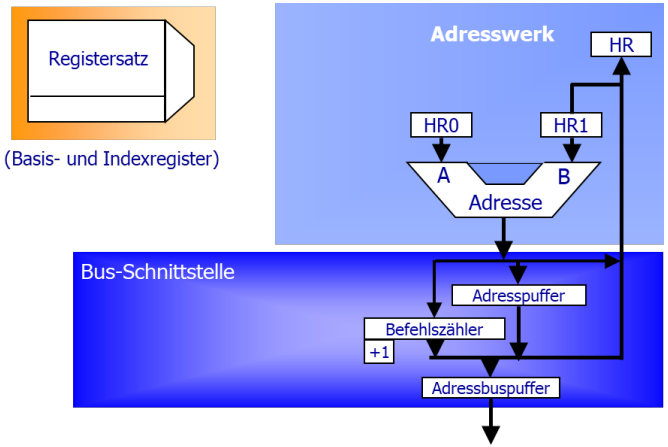
Aufbau eines einfachen Mikroprozessors

- **Adresswerk: Aufbau**
  - Addierer mit 2 Eingängen und einem Ausgang
    - **Eingang B:**
      - Es können die Inhalte von Registern des Registersatzes anliegen.
        - Beispiel: Adresse aus Basis- oder Indexregister
      - Es kann der Inhalt des Befehlszähler anliegen.
        - Adresse des aktuellen Befehls
      - Es kann der Inhalt des Adresspuffers anliegen
        - Adresse des aktuellen Operanden



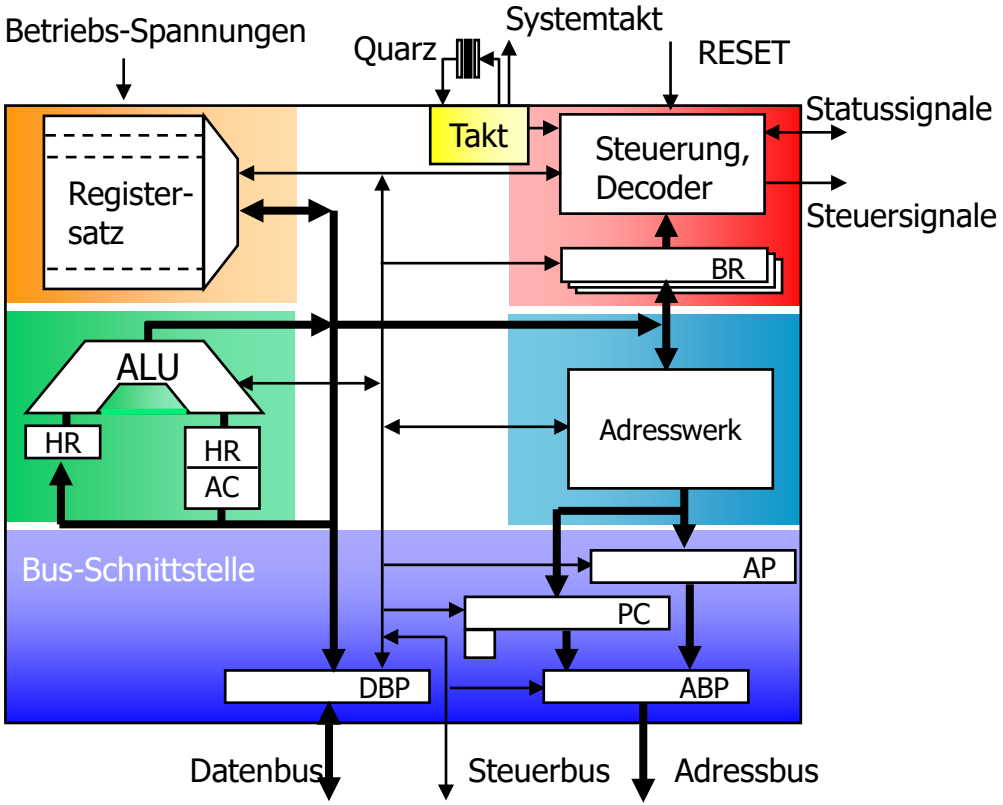
Aufbau eines einfachen Mikroprozessors

- **Adresswerk: Aufbau**
  - Addierer mit 2 Eingängen und einem Ausgang
    - **Ausgang**
      - kann über den Adresspuffer oder den Programmzähler auf Eingang B rückgekoppelt werden;
      - Adresspuffer und Programmzähler können als Akkumulatoren dienen.



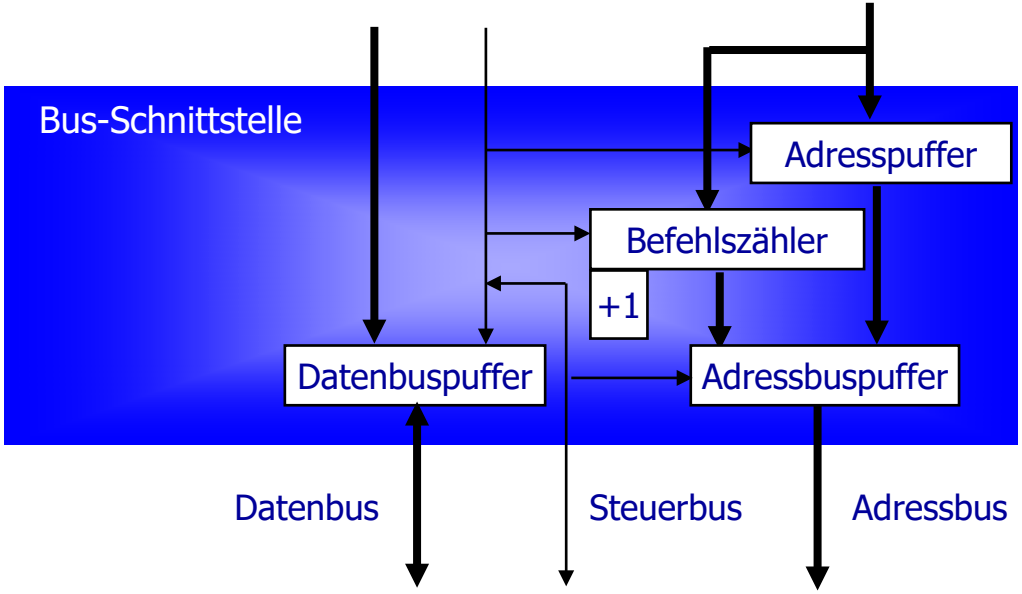
Aufbau eines einfachen Mikroprozessors

- **Bus-Schnittstelle**



Aufbau eines einfachen Mikroprozessors

- **Bus-Schnittstelle**



Aufbau eines einfachen Mikroprozessors

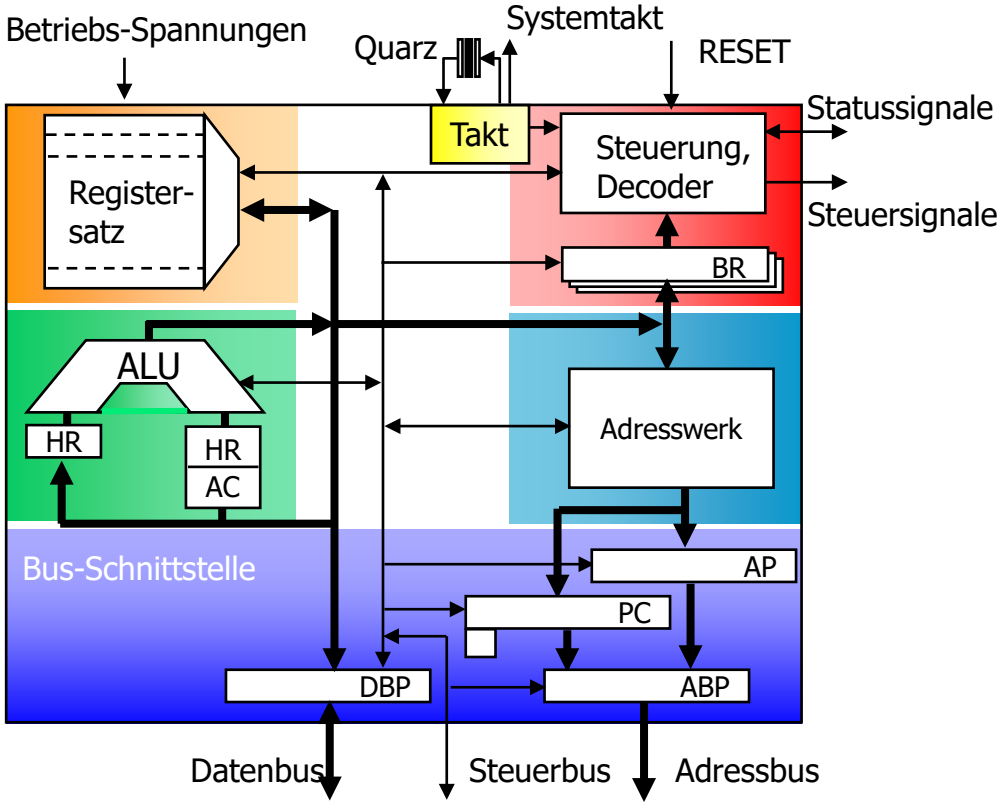
- **Bus-Schnittstelle**
  - Anbindung des Prozessors an den Bus
    - **Adressbus**
      - Menge der Adressleitungen
    - **Datenbus**
      - Menge der Datenleitungen
    - **Steuerbus**
      - Menge der Steuerleitungen
  - Steuerung der Aktionen auf dem Bus (z.B. Lese- / Schreib-Zugriffe)

Aufbau eines einfachen Mikroprozessors

- **Bus-Schnittstelle**
  - Enthält diverse Register (Puffer) zur kurzfristigen Speicherung von Adressen und Daten, z.B.
    - **Befehlszähler**: Adresse des nächsten zu holenden / auszuführenden Befehls (Instruction Pointer, Program Counter)
    - **Adresspuffer:** Akkumulator für Adresswerk; (Zielregister, Quellregister für Adresswerk);
    - **Adressbuspuffer**: Adresse des zu ladenden oder zu speichernden Datums im Hauptspeicher
    - **Datenbuspuffer**: puffert den zu speichernden oder zu ladenden Wert
    - Enthält Aus- und Eingangstreiber (Tristate-Treiber)

Aufbau eines einfachen Mikroprozessors

- **Internes Verbindungsnetzwerk**



Aufbau eines einfachen Mikroprozessors

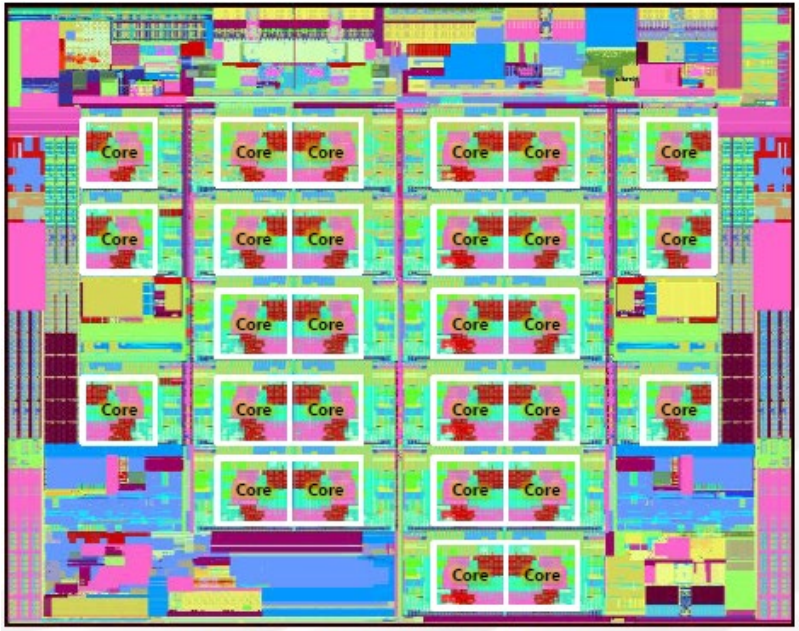
- **Internes Verbindungsnetzwerk**
  - **Menge von Signalleitungen, interne Busse**
    - Verbinden die Komponenten des Prozessors
      - Übertragung von Adressen und Daten
      - Übertragung von Steuer- und Statusinformationen

Allzweck-Mikroprozessoren

- **Weitere Komponenten auf dem Chip:**
  - Rechenwerk
    - Gleitkommaverarbeitung, FPU
    - Multimediaeinheiten
    - …
  - Komponenten zur Unterstützung von Aufgaben des Betriebssystems
    - Speicherverwaltungseinheit (Memory Management Unit, MMU)
    - …
  - Cache-Speicher
  - Schnittstellenbausteine

Allzweck-Mikroprozessoren

- **Fallstudie: Intel Ice Lake SP**

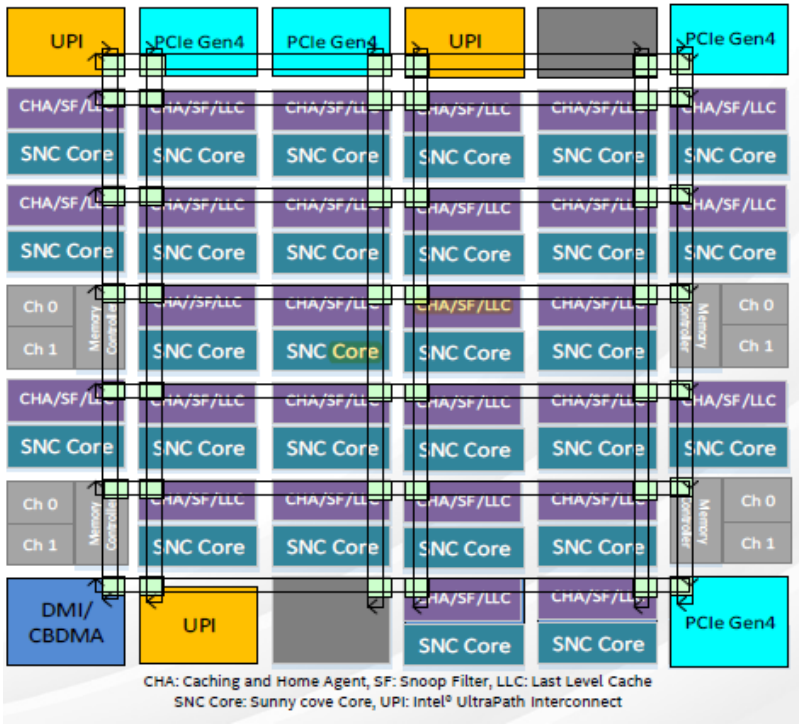


Die picture of a 28C Ice Lake-SP die

https://hc32.hotchips.org/assets/program/conference/day1/HotChips2020_Server_Processors_Intel_Irma_ICX-CPU-final3.pdf

Allzweck-Mikroprozessoren

Fallstudie: Intel Ice Lake SP



SNC Core: Prozessorkerne

CHA, LLC, SF: kohärente Cache-Speicheranbindung

Memory Controller, Ch0, Ch1: Ansteuerung für Off-Chip-Speicher

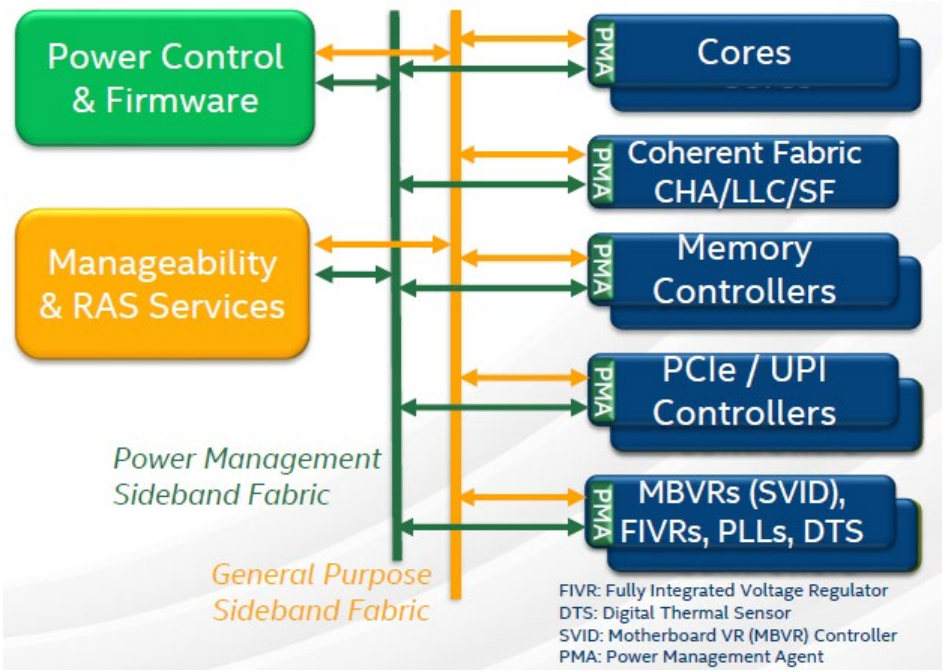
Verbindungsnetz zur internen Kommunikation

PCIe: PCI Express Steuerung: direkten Anbindung von schnellen peripheren Komponenten

UPI : UltraPath Interconnect: Verbindungsstruktur zum Zusammenschluss von mehreren Prozessorchips auf einem Board

Allzweck-Mikroprozessoren

- **Fallstudie: Intel Ice Lake SP**

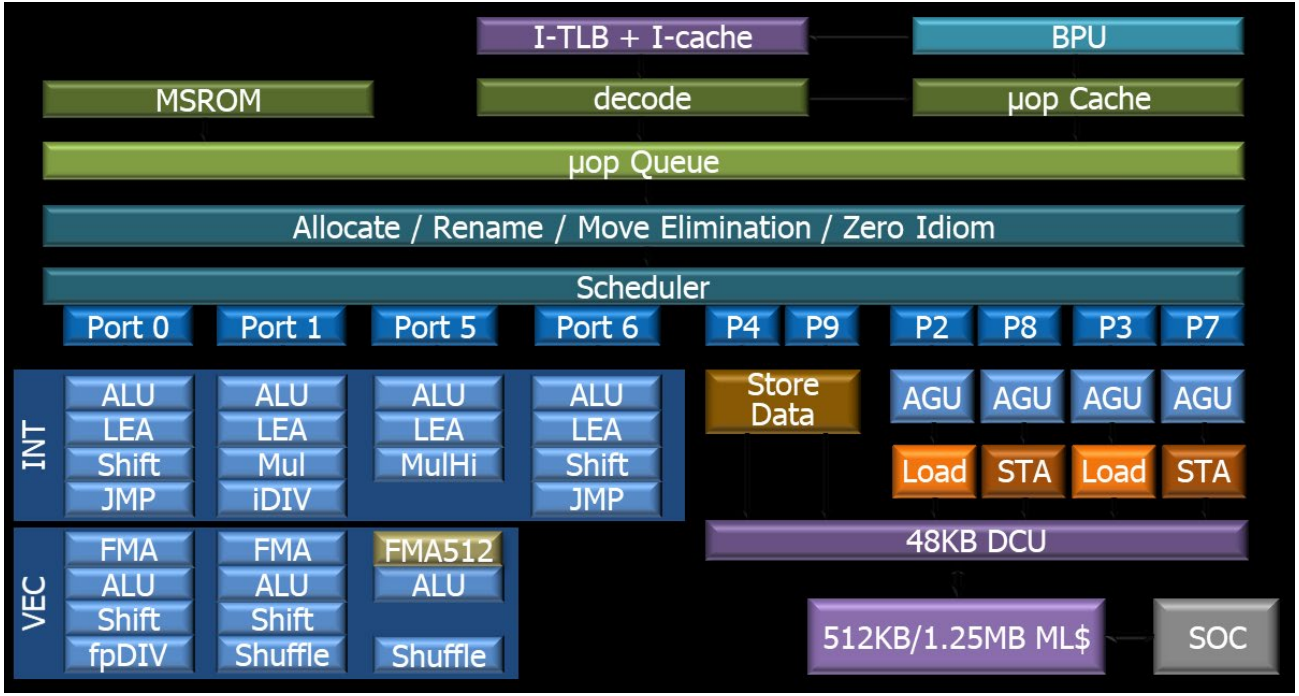


Verwaltungsaufgaben

https://hc32.hotchips.org/assets/program/conference/day1/HotChips2020_Server_Processors_Intel_Irma_ICX-CPU-final3.pdf

Allzweck-Mikroprozessoren

- **Fallstudie: Intel Ice Lake SP Sunny Cove Core Microarchitecture**



https://hc32.hotchips.org/assets/program/conference/day1/HotChips2020_Server_Processors_Intel_Irma_ICX-CPU-final3.pdf