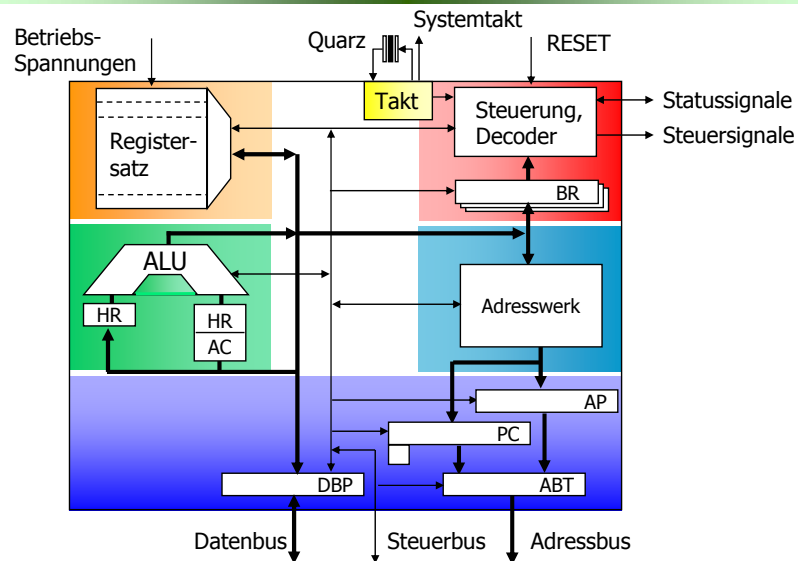
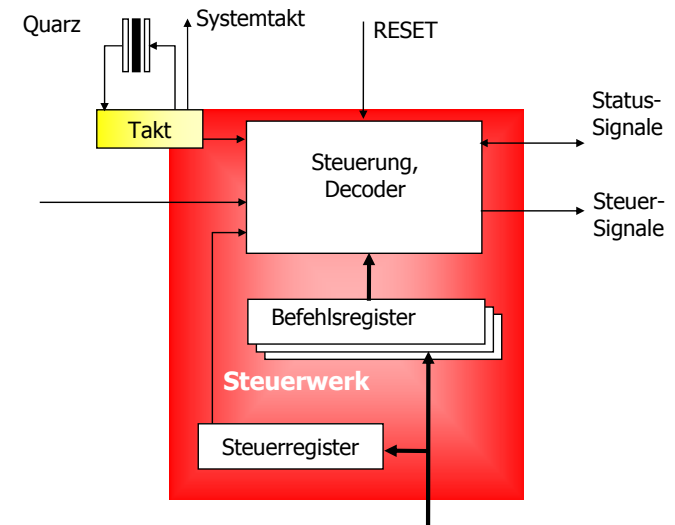


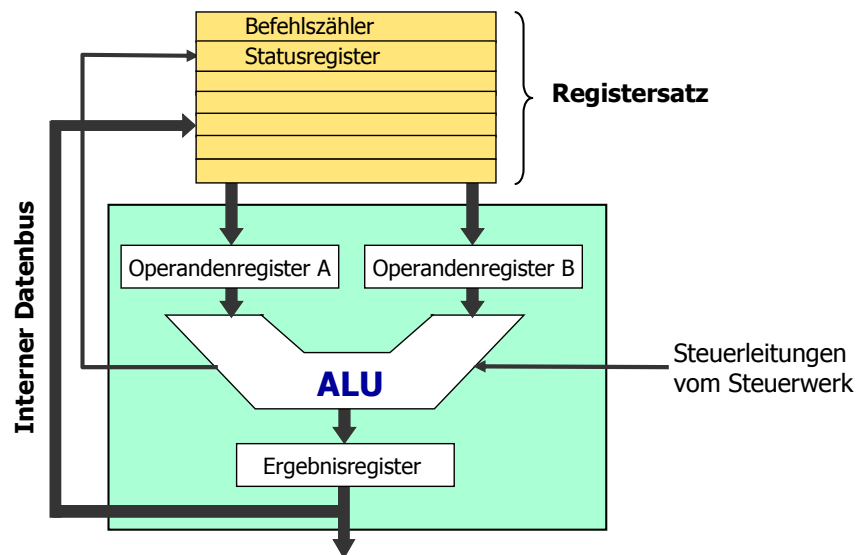
## Interner Aufbau eines einfachen $\mu P$



## Wiederholung: Steuerwerk

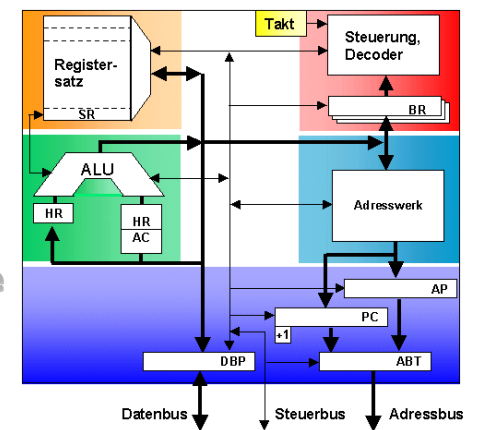


## Wiederholung: Rechenwerk



## Interner Aufbau eines einfachen $\mu P$

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



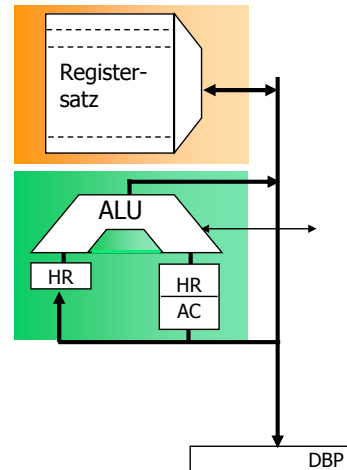
# Registersatz

**Register:** Speicherzellen mit Zugriffszeit unter einer bis zu wenigen Nanosekunden

- Sie sind auf dem Chip untergebracht  
→ schneller Zugriff als auf den Hauptspeicher
- Ihre Auswahl erfolgt durch individuelle Steuerleitungen  
→ Zeit zur Adressdekodierung entfällt

**Registersatz:** Erweiterung des Rechenwerks

- Häufig benutzte Operanden können dort zwischen-gespeichert werden



# Registersatz

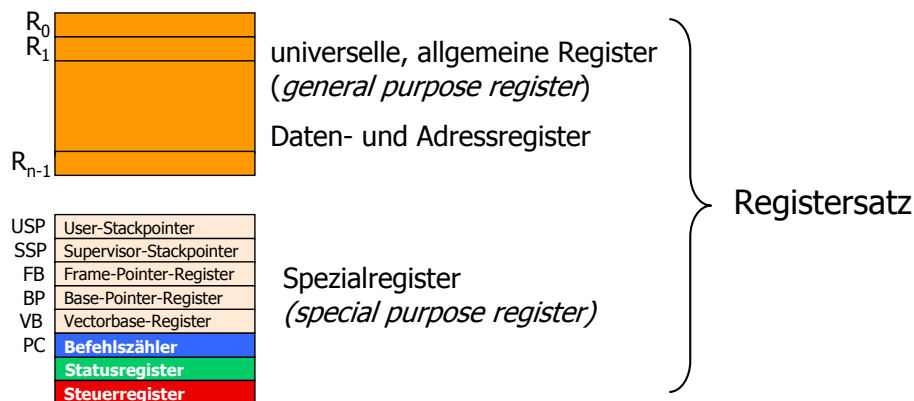
- Getrennte Ein-/Ausgänge → Dual Port Speicher, zwischen Eingangs- und Ausgangsbuss gehängt → gleichzeitiges Schreiben eines Registers und Lesen eines anderen Registers möglich

**Heutige superskalar-Prozessoren: Gleichzeitiges Schreiben und Lesen mehrerer allgemeiner Register in einem Taktzyklus**

- Oft dynamische Speicherzellen → Refresh erforderlich
- „Größere“ Registersätze (*Register File*) werden mit zusätzlichem Adressdekodierer realisiert
- Register mit Zusatzfunktionen: Inkrementieren, Dekrementieren, Inhalt auf Null setzen, Inhalt verschieben



# Registersatz



Die benutzerzugänglichen Register werden als **Programmiermodell des Prozessors** bezeichnet. Sie fassen alle für den Assembler-programmierer wesentlichen Eigenschaften der Prozessor-Hardware



# 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:

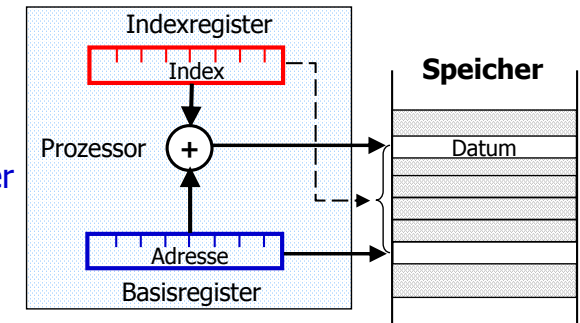
- ❑ Befehlszähler (instruction pointer)
- ❑ Steuerregister
- ❑ Statusregister
- ❑ Register für den Start von Interrupt-Behandlungen (interrupt vector base register)
- ❑ **Stackregister** (user und supervisor Stackpointer)



## Funktion von Basis- und Indexregister

Basisregister enthält die Anfangsadresse eines Speicherbereichs.

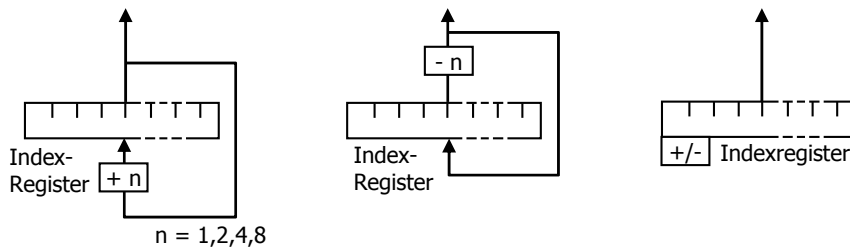
Diese bleibt während der Bearbeitung des Speicherbereichs unverändert.



Indexregister enthält eine Distanz (Offset, Displacement) zu einer Basisadresse und dient zur Auswahl eines bestimmten Datums des Speicherbereichs.



## Automatische Modifikation von Indexregistern



### a) Post-Inkrement:

automatische Erhöhung des Registerwerts um  $+n$  nach Adressierung einer Speicherzelle

### b) Pre-Dekrement:

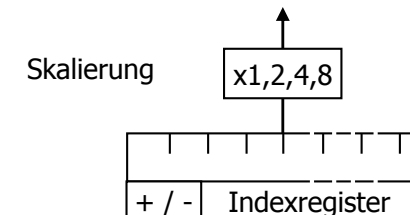
automatische Erniedrigung des Registerwerts um  $-n$  vor Adressierung einer Speicherzelle

### c) Auto-Inkrement / Auto-Dekrement Register



## Register mit Skalierung

Das Indexregister wird vor der Auswertung je nach aktueller Datenlänge (1 Byte, 2 Byte, 4 Byte, 8 Byte) mit dem Faktor 1, 2, 4 oder 8 multipliziert.



### Vorteil:

bessere Ausnutzung der Registerbreite, da das Register selbst nur noch um 1 inkrementiert bzw. dekrementiert werden muss.



## Der (Laufzeit-) Stack "Kellerspeicher"

**Kellerspeicher:** Speicherbereich, der normalerweise im Arbeitsspeicher angelegt (software stack) und nach dem Kellerprinzip (*LIFO Last-in-first-out*) organisiert ist

### Funktion:

- Abspeichern des Prozessorstatus und des Befehlszählers beim Unterprogrammaufruf und Aufruf von Unterbrechungs-Routinen
- Parameterübergabe
- kurzzeitige Lagerung von Daten bei der Ausführung

Bei modernen Prozessoren häufig mehrere getrennte Stackspeicher: System Stack, User Stack, Data Stack

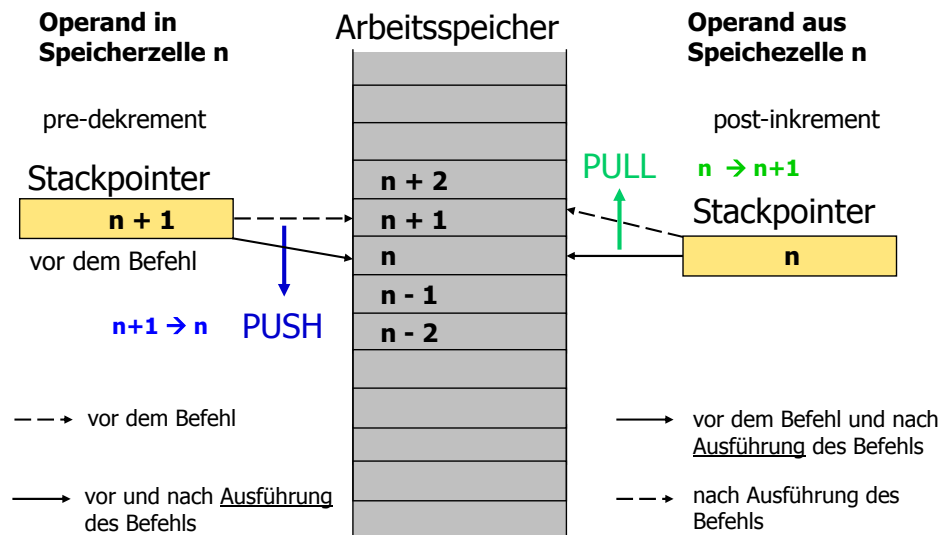


## Hardware-Unterstützung des Stacks

- ❑ Stackregister (Stapelzeiger, Stack Pointer SP): enthält die Adresse des zuletzt in den Stack eingetragenen Datums
- ❑ Spezielle Befehle zur Datenübertragung in den bzw. aus dem Stack
  - **PUSH:**  
Inhalt eines Registers wird in den Stack übertragen
  - **POP (PULL):**  
Inhalt eines Registers wird vom Stack geladen

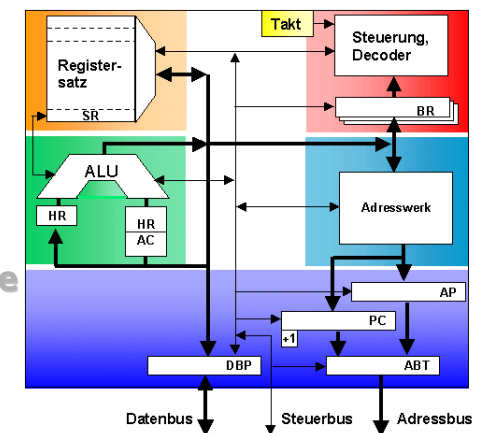


## Verwaltung des Stackregisters



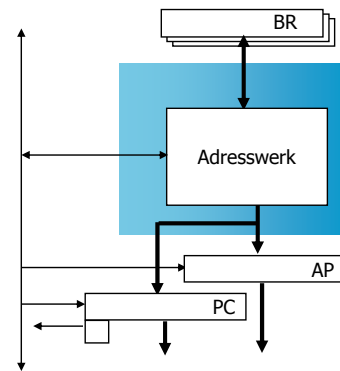
## Interner Aufbau eines einfachen $\mu P$

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

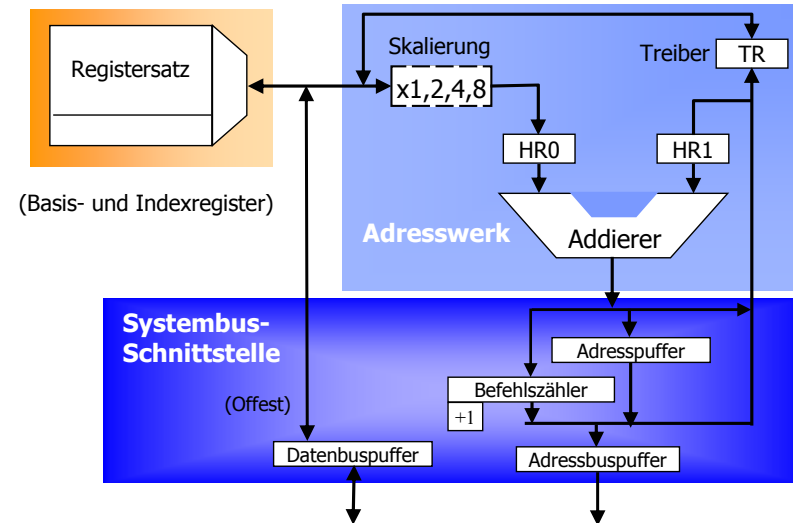


# Adresswerk

- Berechnet nach den Vorschriften des Steuerwerks die Adresse eines Befehls oder eines Operanden
- **Früher:** Häufig Bestandteil des Rechenwerks
- **Heute:** Sehr komplex (viele verschiedene komplexe Adressierungsarten) und deshalb eigenständig



# Aufbau eines einfachen Adresswerks



## Funktionsweise

### Adress-Addierer mit 2 Eingängen

#### Eingang (HR0):

- Registersatz (Adresse aus Basis- oder Indexregister)
- Datenbuspuffer (absolute Adresse im Befehl oder absolute Adressdistanz zu einem Basisregister)

#### Eingang (HR1):

- Befehlszähler (Adresse des aktuellen Befehls)
- Adresspuffer (Adresse des aktuellen Operanden)

Der Ausgang kann über den Adresspuffer oder den Befehlszähler auf Eingang **HR1** rückgekoppelt werden

➔ Adresspuffer und Befehlszähler als Akkumulatoren



## Skalierung

### Skalierung:

- Operanden mit 1,2,4,8,... multiplizieren,
- Adreßberechnung für sukzessive Zugriff auf Bytes, Worte, Doppelworte, Vierfachworte,...

### Multiplikation mit positiver Zahl:

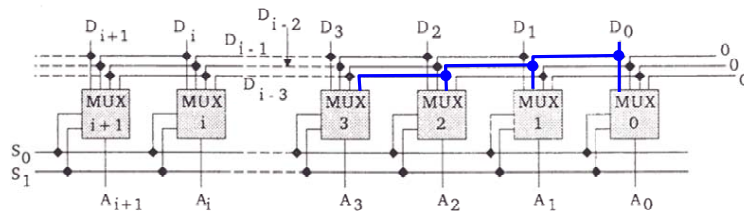
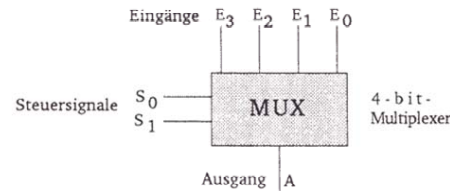
Verschieben nach links mit nachrückenden Nullen

Realisierung durch Schaltnetz: **Barrel-Shifter**

Ein Barrel-Shifter erlaubt die Verschiebung um **n** Bits parallel (d. h. in einem einzigen Taktzyklus)



## Aufbau eines 4 Bit Barrel-Shifter



Auf jede Ausgangsleitung  $A_i$  wird je nach Wert der Steuerleitungen die Eingangsleitung  $D_i$ ,  $D_{i-1}$ ,  $D_{i-2}$  oder  $D_{i-3}$  geschaltet



## Das Adresswerk

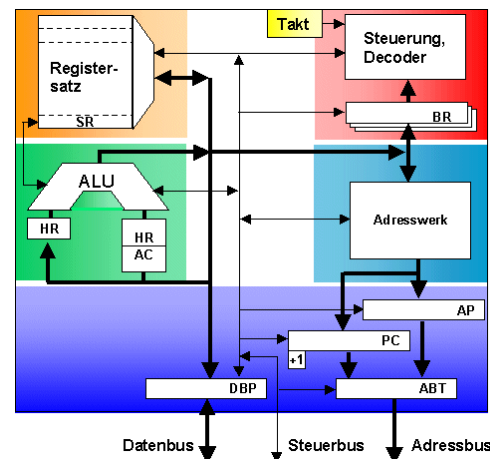
**Heute:** Eigene Adresswerke.

- Adressberechnung findet parallel (im Pipelining-Verfahren) zu den Aktivitäten des Rechenwerks statt!
- Führt neben einfacher Adressrechnung auch die Adressrechnung zur virtuellen Speicherverwaltung durch
- Früher oft separater Baustein, heute in den Prozessor integriert.
- Komplexe Adresswerke zur virtuellen Speicherverwaltung (später!)



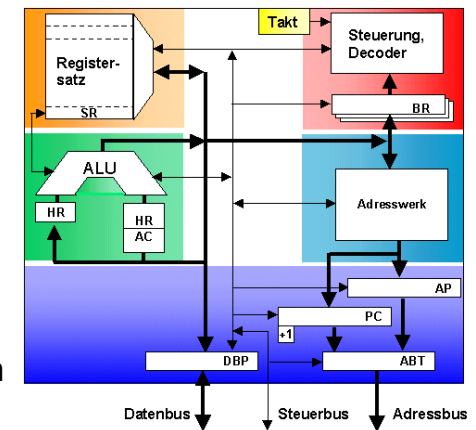
## Interner Aufbau eines einfachen $\mu P$

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

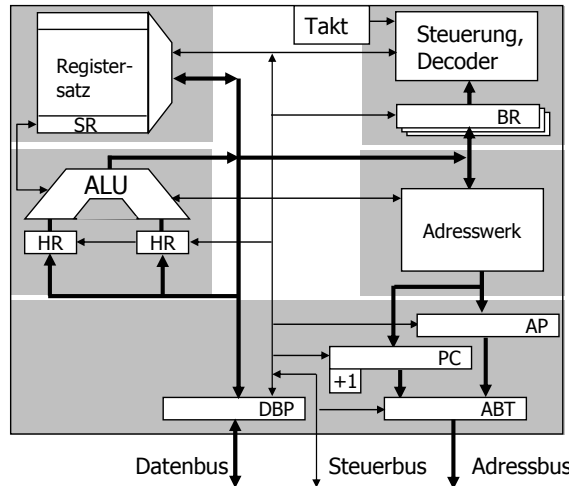


## Das interne Bussystem

- Dient zur Verbindung der Prozessor-Komponenten untereinander
- Keine eindeutige Unterscheidung Daten- und Adressbus, da einmal Operanden und einmal Adressen übertragen werden



# Das interne Bussystem



Legende:

ABT	Adressbustreiber
ALU	Arithmetisch/logische Einheit
AP	Adresspuffer
BR	Befehlsregister
DBP	Datenbuspuffer
HR	Hilfsregister
PC	Befehlszähler
SR	Statusregister



# Das interne Bussystem

## Steuerbus:

- Heterogene Einzelleitungen, funktional nicht zusammenhängend, werden der Übersichtlichkeit wegen jedoch als Bus dargestellt

## Adressbus:

- Ursprung im Adresswerk (AU)
- Zwischenspeichern von Adressen im Befehlszähler (PC, Befehlsadressen) und Adresspuffer (AP, Operandenadressen)
- Über Adressbustreiber mit externem Adressbus verbunden

## Datenbus:

- verbindet alle übrigen Komponenten (Rechenwerk, Adresswerk, Steuerwerk, Registersatz) zwecks Datentransports
- Über Datenbuspuffer mit externem Datenbus verbunden



# Das interne Bussystem

## Problem bei dieser Datenbus-Architektur:

Höchstens ein Datentransfer gleichzeitig zwischen zwei bestimmten Prozessor-Komponenten möglich

➔ **geringe Parallelität** bei der Befehlsbearbeitung

## Lösung:

Aufspalten des internen Datenbus in mehrere Teilbusse  
➔ höherer Parallelität, höhere Geschwindigkeit

## Aber:

Busse erfordern ein erhebliches Maß an Chipfläche  
➔ Ausbau des internen Bussystems nicht beliebig möglich

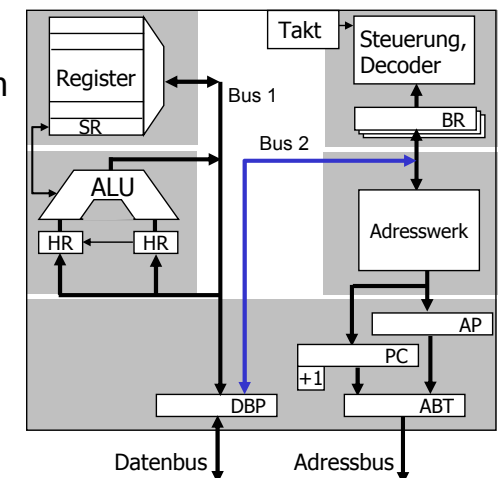


# Varianten des internen Bussystems

## 1. Prefetch Bus

Hinzufügen eines zweiten, getrennten Datenbusses zum Adress- und Steuerwerk

- ➔ Parallel zu Aktionen im Rechenwerk/Registersatz kann das Befehlsregister mit dem nächsten Befehl geladen werden
- ➔ Opcode Prefetch in eine Prefetch-Queue wird möglich



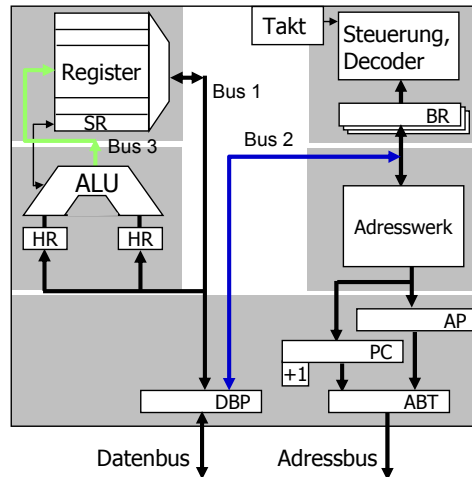
## Varianten des internen Bussystems

### 2. Ergebnis-Bus

Zusätzlicher Bus, der die Ergebnisse des Rechners in den Registersatz transportiert

➔ Parallelisierung im Rechner

Während das Rechner ein Ergebnis in den Registersatz transportiert, können die Hilfsregister mit dem nächsten Operanden geladen werden



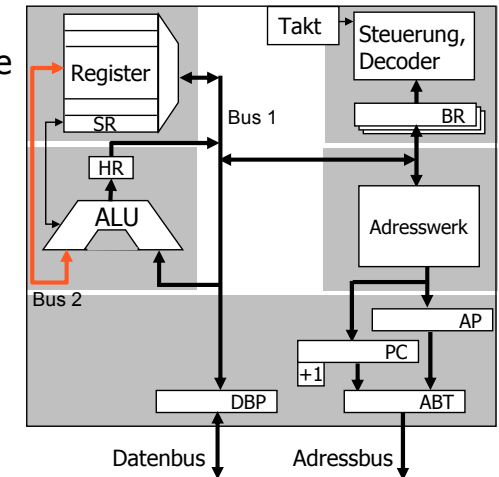
## Varianten des internen Bussystems

### 3. Zweiter Operandenbus

beide Operanden parallel an die ALU führen

➔ alternative Parallelisierung im Rechner

- Operand 2 stammt immer aus dem Registersatz (Bus 2)
- Operand 1 wahlweise aus dem Registersatz oder dem Datenbuspuffer (Bus 1)
- Das Hilfsregister entkoppelt Operand 1 vom Ergebnis



## Varianten des internen Bussystems

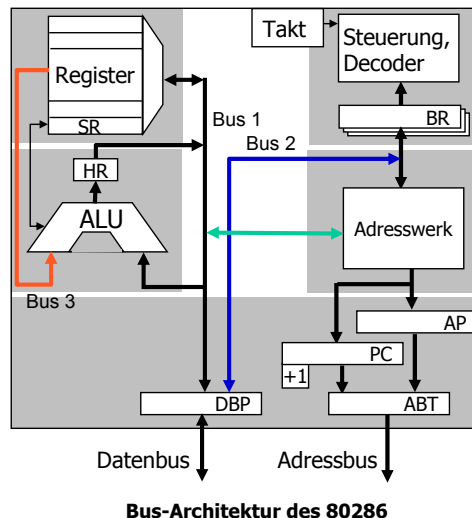
### 4. Zweiter Operandenbus / Prefetch Bus

Kombination der Varianten 1 und Variante 3

Zusätzliche Verbindung des Operanden/ Ergebnis Busses (Bus 1) zum Adresswerk

Prefetch-Bus ➔ Parallele Rechner-Operationen

Direkter Zugriff des Adresswerks auf Adressregister im Registersatz ohne Umweg über Daten- oder Prefetch-Bus



## Varianten des internen Bussystems

### 5. Extrembeispiel

Kombination aus:

- Prefetch Bus (Bus 4)
- zwei Operandenbussen (Bus 1 und 2)
- ein Ergebnisbus (Bus 3)
- Hilfsregister wird nur bei gleichem Operanden- und Ergebnisregister benötigt

