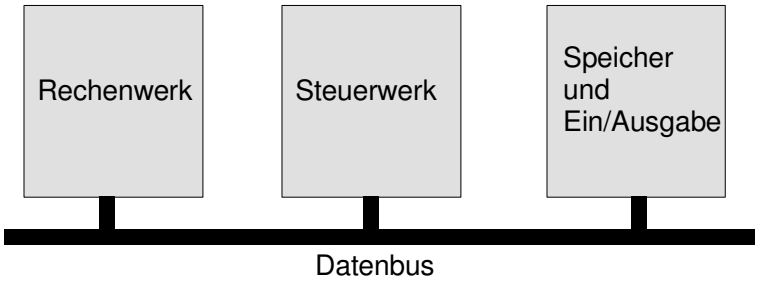
7. Prozessoren

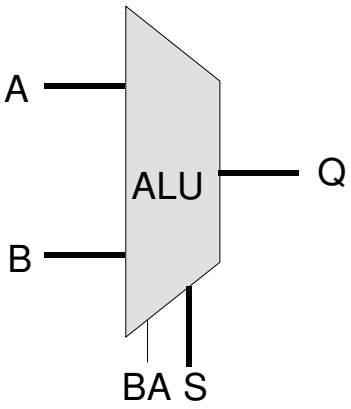
7.1. Grundaufbau

Die Grundbestandteile eines Rechners sind Rechenwerk, Steuerwerk, Speicher und Ein/Ausgabe-Einheiten, die über einen Datenbus miteinander gekoppelt sind.



7.2. ALU und Rechenwerk

ALU:



Q = A ¤ B (¤: ALU-Operator, abh. von S, BA)

Steuerung: (s_3,s_2,s_1,s_0) S. Steuervektor

Operation: $q_i = (g_i \vee \overline{p_i}) \equiv (BA \vee c_{i-1})$ BA: Betriebsart

$g_i = s_3\, a_i\, b_i \vee s_2\, a_i\, \overline{b_i}$

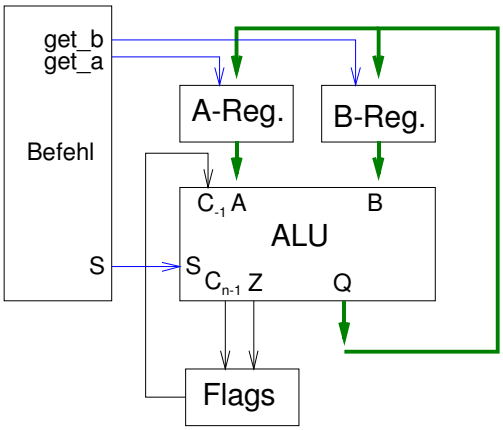
$p_i = s_1\, \overline{a}\, b \vee s_0\, a\, \overline{b}$

Übertrag: $c_i = g_i \vee p_i\, c_{i-1}$

Die sequentielle Arbeitsweise der ALU mit verschiedenen Operanden und Operationen erfordert:

- temporäre Register für A, B
- Zwischenspeicher für das Ergebnis Q (z.B. Akkumulatorregister A)
- Datentransport zwischen Ergebnis und Operand (Busstruktur)
- temporäre Speicherung der Steuersignale S (Befehlsspeicher)

Diese Komponenten bilden das Rechenwerk:



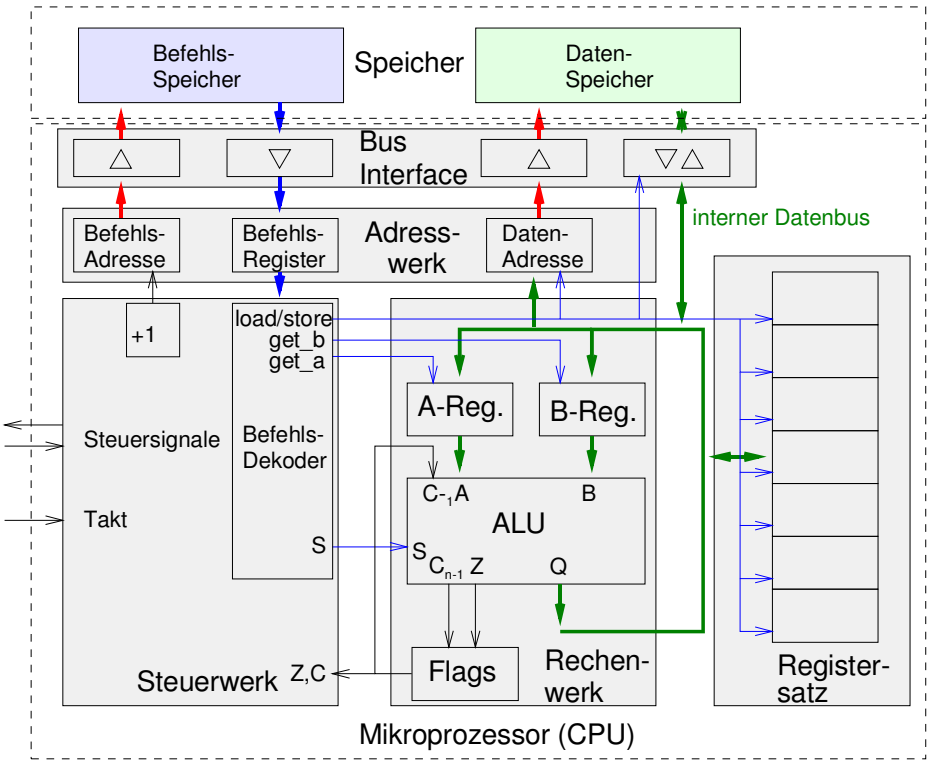
Beispiel einer 4-bit-ALU:
A=0010 (2)
B=1110 (14)
C_{-1}=0
S=[1,0,0,1] (Addition mit Übertrag)

liefert nach der Verzögerungszeit der ALU:
Q=0000
Z=1
C_{n-1}=1

Der Wert von Q wird durch ein get_a Signal in das Ergebnisregister A (taktflankengesteuerte Flipflops) eingeschrieben.

Zur Realisierung einer algorithmischen Abarbeitung sind erforderlich:

- Speicherstrukturen für Programme und Daten
- Adressgenerierung für den Programmspeicher (Befehlszähler, Befehls-Adressregister)
- Adressgenerierung für den Datenspeicher (Daten-Adreßregister)
- Speicherung des Übertrages C (Carry-Flag) und des Null-Ergebnisses Z (Zero-Flag)
- Auswertung der Flags für Verzweigungen (Steuerwerk)
- Steuerung externer Ein/Ausgabekanäle (Steuersignale)



Die Grundbestandteile eines Rechners sind Rechenwerk, Steuerwerk, Registersatz, Speicher und (hier noch nicht dargestellt) Ein/Ausgabe-Einheiten. Das Rechenwerk beinhaltet zumindest eine ALU und Operanden- bzw. Ergebnisregister, die über einen Datenbus mit dem Registersatz, Datenspeicher oder Ein/Ausgabebaugruppen kommunizieren. Das Steuerwerk bestimmt Funktion und Betriebsart der ALU und organisiert die Transportfunktionen über dem Datenbus. Das Befehlsregister IR (Instruction Register) speichert den aktuell ausgeführten Befehl, der über eine Befehlsadresse aus dem Speicher gelesen wird. Aufeinanderfolgende Befehle werden in aufeinanderfolgenden Speicherplätzen abgelegt, daher wird die Befehlsadresse durch einen inkrementierenden Befehlszählers PC (Program Counter) realisiert .

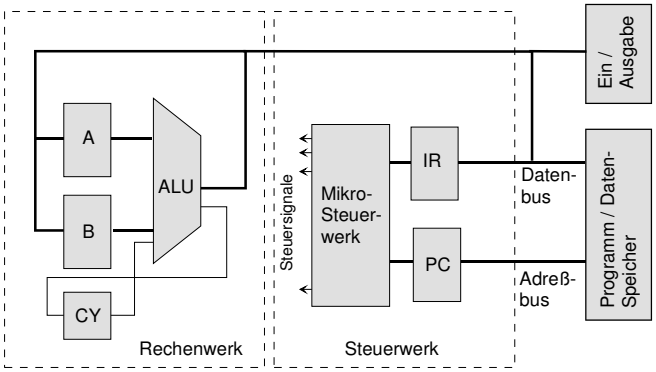
weitere Baugruppen zur Erhöhung der Leistungsfähigkeit

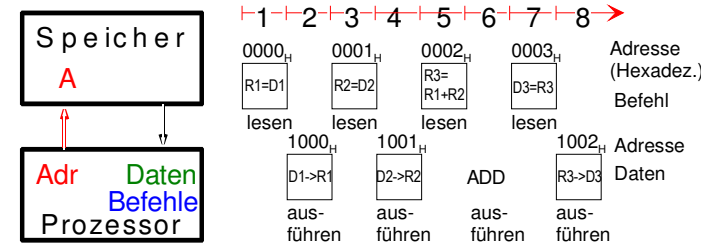
- mehrere interne Befehlsregister (Prefetch queue) als FIFO-Speicher
- Adresswerk mit komplexen Adressierungsmechanismen
- interner Stapelspeicher (Stack) bzw. Adressgenerator für Stapelspeicher (Stack Pointer)
- Parallelisierung von Baugruppen (z.B. Rechenwerke, Register, Speicher)
- schnelle interne Daten- und Programmspeicher (Cache)
- Komponenten für eine Gleitkommaarithmetik: FPU (Floating Point Unit), Gleitkomma-Register

Von-Neumann-Struktur

Konzept zur Gestaltung eines universellen Rechners:

- Struktur ist unabhängig von der Problemlösung
- Rechenwerk, Steuerwerk, Speicher, Ein- und Ausgabewerk
- in einem Speicher werden Programme und Daten binär codiert gehalten
- Speicherzellen werden durchnummeriert, erhalten eine eindeutige Adresse
- Aufeinanderfolgende Befehle werden in fortlaufenden Speicherstellen abgelegt.
- Eine Änderung der Abarbeitungsfolge ist durch Sprungbefehle möglich
- Verarbeitungsschritte erfolgen durch Transportbefehle, Arithmetik- und Logikbefehle
- Die Steuerung erfolgt durch binäre Schaltwerke





von-Neumann-Struktur

Dx: Speicherplätze z.B. D1=(1000_H)
Rx: Prozessor-Register

Parallelisieren von Rechnerstrukturen:

Im einfachsten Fall können zwei oder mehr Rechenwerke vorgesehen werden, die beispielsweise über den Datenbus gekoppelt werden, oder die nacheinander arbeiten.

Q = A ¤ B ¤ C ALU1 : Q1 = A ¤ B , ALU2: Q = Q1 ¤ C

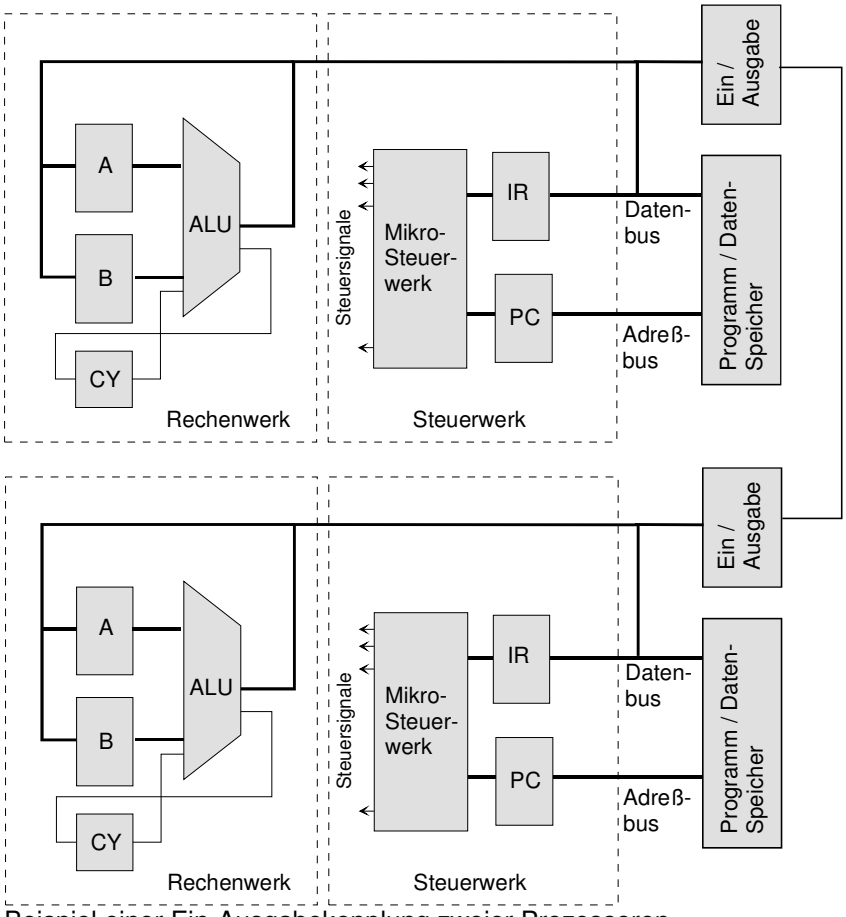
Sehr oft wird diese Struktur in Signalprozessoren angewendet, die typischerweise eine Multipliziereinheit anstelle von ALU2 besitzen und damit eine MAC Operation (Multiply ACcumulate) durchführen können:

Q = Q + A * X

Diese Operation, mehrfach durchgeführt, eignet sich zur effektiven Berechnung gewichteter Summen:

$$Q = \sum_i A_i X_i$$

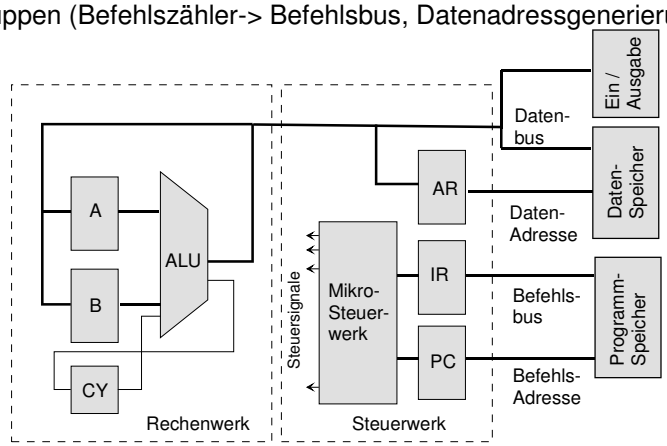
Ebenso können natürlich zwei oder mehrere Von-Neumann Rechnerstrukturen über den Datenbus, über Ein-Ausgabeleitungen oder über den Speicher (z.B. Dual-Port-Speicher) gekoppelt werden.

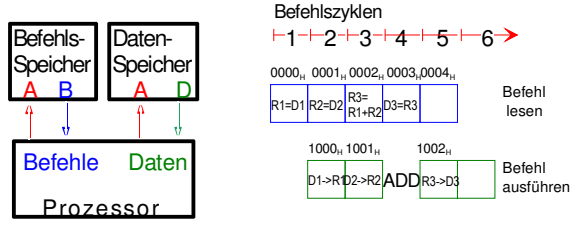


Beispiel einer Ein-Ausgabekopplung zweier Prozessoren

Harvard-Struktur

getrennte Daten- und Programmspeicher ermöglicht die zeitlich parallele Nutzung vorhandener Baugruppen (Befehlszähler-> Befehlsbus, Datenadressgenerierung AR -> Datenbus)

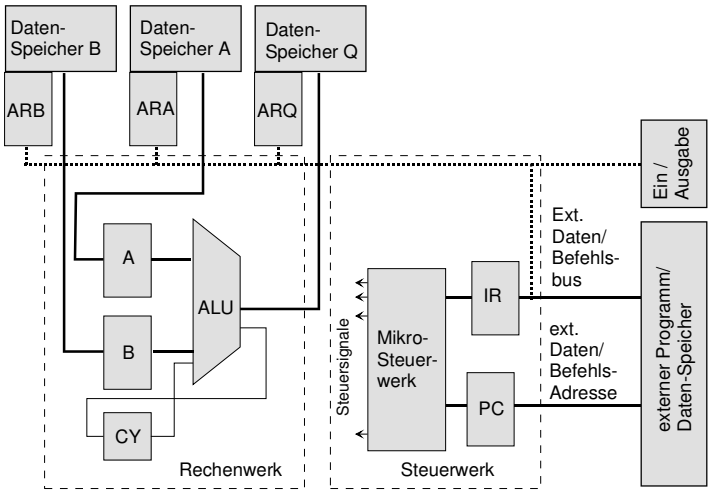


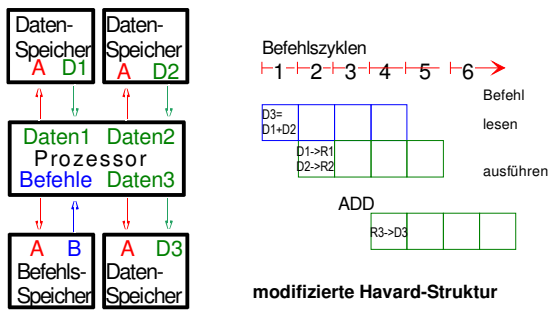


Harvard-Struktur

Modifizierte Harvard-Struktur

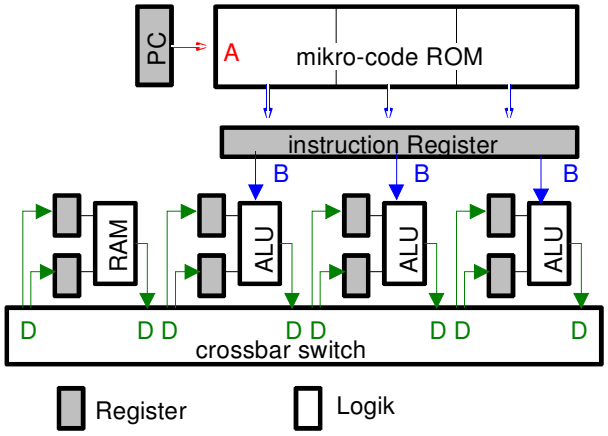
getrennte Datenspeicher für die Operanden A, B und Ergebnisse Q sowie ein Programmspeicher ermöglichen die zeitlich parallele Nutzung vorhandener Baugruppen (Befehlszähler-> Befehlsbus, Datenadressgenerierung ARA, ARB, ARQ -> getrennte Datenbusse, die über einen externen Datenbus mit einem externen Speicher kommunizieren.)





modifizierte Havard-Struktur

VLIW-Struktur
Viele Rechenwerkskomponenten (ALU, Speicher) werden parallel betrieben und programmiert.



VLIW Architektur(very large instruction word)
(Super-skalar-Architektur)

7.3. Befehle und ihre Speicherung

ALU: erfordert Festlegung, welche Operation mit welchen Daten durchgeführt werden muß
→ **Befehl**

Befehlsbestandteile:
1. Art der Operation (Op-Code)
2. erster Operand A
3. zweiter Operand B
4. Ergebnis Q

1	Op.-code
2	Operand A
3	Operand B
4	Ergebnis Q

Programmspeicher

Speicherorganisation:
Mehrere unterschiedliche Befehle sollen auf die gleichen Daten angewendet werden
-> Daten werden in einem eigenen Speicher gehalten und durch eine Adresse angewählt:

Programmspeicher

Programmspeicher Adresse	
1	Op.-code
2	10 (Adresse A)
3	11 (Adresse B)
4	12 (Adresse Q)

Datenspeicher

Datenspeicher Adresse	
10	Operand A
11	Operand B
12	Ergebnis Q
13	

technische Realisierung:
- Eine Speicheradresse bestimmt immer den Zugriff auf genau ein Byte.
- Werden für Programm und Daten unterschiedliche Adressen vereinbart, kann ein einziger physicher Speicher genutzt werden.
- Wenn mehr als 28 Speicherplätze existieren, müssen die Adressen auf mehreren aufeinanderfolgenden Plätzen im Programmspeicher abgelegt werden. (z.B. 16 bit Adressen auf 2 Byte, 32-bit-Adressen auf 4 Byte)
- Es bedarf einer Vereinbarung, ob zuerst der Low-Teil oder der High-Teil der 16 bit Adresse auf aufeinanderfolgenden Speicherzellen abgelegt wird.
- Die Datenspeicherung bei größeren Bitbreiten (16, 32, 64 bit) erfolgt ebenso:

Programmspeicher Adresse	bit: 7 6 5 4 3 2 1 0
2	01 Adr. A (low)
3	10 Adr. A (high)
4	02 Adr. B (low)
5	10 Adr. B (high)

Datenspeicher Adresse (hexadez.)	bit: 7 6 5 4 3 2 1 0
1000	Operand A (low)
1001	Operand A (high)
1002	Operand B (low)
1003	Operand B (high)

1 Byte

Bsp.: Low-High Reihenfolge (sog. "little endian" Format, typisch für Intel-Prozessoren)

Adressierungskonzept für Mikroprozessoren

Abbildung der Befehlsbestandteile auf Speicherplätze des Programmspeichers:

1. vier aufeinanderfolgende Speicherplätze (je 8 bit) -> drei-Adress-Maschine

Op.-code	Adresse A	Adresse B	Adresse Q
1	2	3	4

2. drei aufeinanderfolgende Speicherplätze (je 8 bit) -> zwei-Adress-Maschine

Op.-code	Adresse A und Q	Adresse B
1	2	3

3. zwei aufeinanderfolgende Speicherplätze (je 8 bit) -> ein-Adress-Maschine

Op.-code +Adr. A, Q	Adresse B
1	2

bei 16 bit Adresse B (65536 Speicherplätze):

Op.-code und	Adresse A und Q	Adresse B (low)	Adresse B (high)
1	2	3	4

4. zwei aufeinanderfolgende Speicherplätze (je 16 bit) -> ein-Adress-Maschine, jedoch mit Direkt-Operand für B (16-Bit-Zahl)

Op.-code und Adresse bzw. Register	Operand B (low)	Operand B (high)
1	2	3	4

Spezielle Speicher: **Register**

- Adresse A und Q werden nur mit wenigen Bits codiert
  -> wenige Quell- und Zielspeicher, als spezielle integrierte Registerspeicher realisiert.
- Registerspeicher sind in beliebiger Datenbreite realisierbar, also auch als 12, 16, 24, 32, 64 bit Register.
- Op-code und Registercodierung werden je nach Befehlsanzahl und Registeranzahl ausgelegt -> Befehlssatz, Registerstruktur. (Behandlung in "Technische Informatik 2")
- Register können Quell- oder auch Zieladresse sein.
  z.B. 8bit: Opcode und R1, R2 oder 16 bit: Op.-code und R1, R2

Spezielle Adressierungsarten:

5. indirekte Adressierung: der Datenspeicher enthält die Adressen der Operanden

Programmspeicher

Programm-speicher Adresse	
1	Op.-code
2	10 (Adresse A,Q)
3	11 (Adresse B)
4	

Registerspeicher

Daten-speicher Adresse	
10	12 (Adresse A, Q)
11	13 (Adresse B)
12	Op. A, Erg. Q
13	Operand B

6. registerindirekte Adressierung: Register enthalten die Adressen der Operanden

Registerspeicher

Register-speicher Adresse	
R1	12 (Adresse A, Q)
R2	13 (Adresse B)
R3	
R4	

Registerspeicher

Daten-speicher Adresse	
10	
11	
12	Op. A, Erg. Q
13	Operand B

7. indizierte Adressierung: der Datenspeicher oder ein Register enthält die Anfangs-Adresse (Offset) des Operanden, zu der eine Abstand (Index) addiert wird. Der Index kann Direktoperand sein, oder im Register oder Datenspeicher stehen. Es sind viele Vartiationen, auch Mehrfach-Indizierungen möglich.

8. absolute Adressierung des Befehlszählers

Programmspeicher Adresse	
1	Op.-code1
2	5 (Adresse P)
3	
4	
5	Op.-code2

Op.-code1: Sprungbefehl auf die Adresse (5), die im folgenden Programmspeicher liegt
Optionen:
- bedingter Sprung,
  z.B. durch Auswertung der Flag-Register C, Z
- Unterprogrammaufruf,
Programm wird nach Verlassen des Unterprogrammes bei Adresse 3 fortgesetzt

9. befehlszählerrelative Adressierung

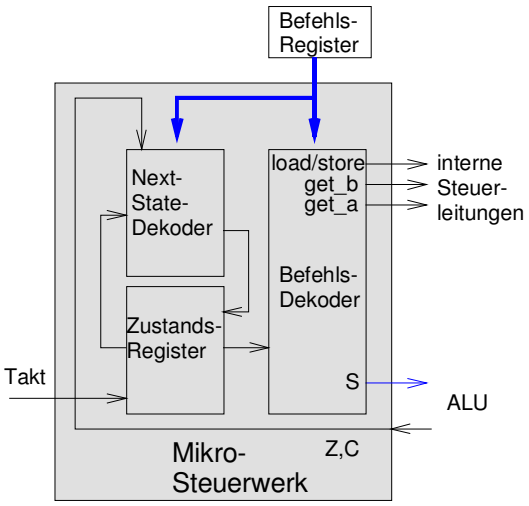
Programmspeicher Adresse	
1	Op.-code1
2	2 (Entfernung)
3	
4	
5	Op.-code2

Op.-code1: Sprungbefehl auf die Adresse, die um eine Entfernung (2) vom nächsten Programmzähler (3) entfernt ist -> Adresse 5
Optionen:
bedingter Sprung, relativer Unterprogrammaufruf

7.4. Steuerwerk

1. Mikrosteuerlogik (Random-Logik)

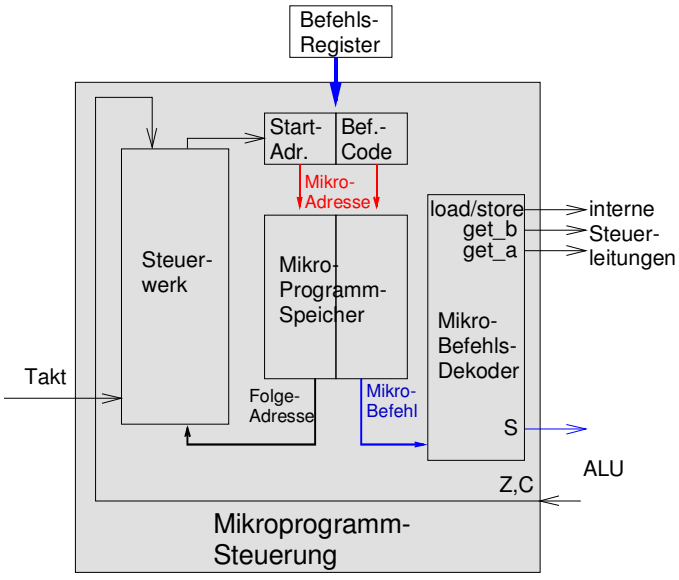
- Mikrosteuerwerk: sehr schneller digitaler Automat (z.B. Mealy-Automat), oder asynchrone Laufzeitschaltungen (Random-Logik)
- gut optimierbar, wenig Bauelemente, kurze Signallaufzeiten
- günstig für RISC-Rechnerstrukturen (Reduced Instruction Set Computer)
  d.h. für einfache reguläre Befehlssätze geeignet
- schwieriger Entwurf, Änderungen erfordern einen kompletten Neuentwurf



2. Mikroprogrammsteuerung

aus dem Befehlskode wird über den Mikroprogrammspeicher der Mikrobefehl entweder direkt (horizontale Mikroprogrammierung) oder indirekt über einen Mikrobefehls-Dekoder (vertikale Mikroprogrammierung) abgeleitet.

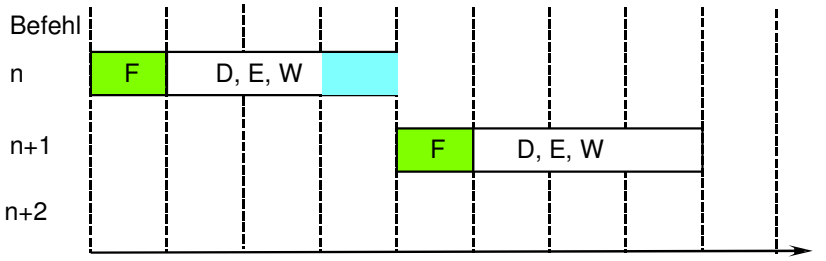
- einfacher Entwurf, leichte Änderung im Entwurfsprozeß möglich
- leichte Implementation nahezu beliebiger, auch komplizierter Befehle
  z.B. CISC-Rechnerstrukturen (Complex Instruction Set Computer)
- großer Platzbedarf, relativ langsame Abarbeitung, schlechte Optimierbarkeit



Konzepte zur zeitlichen Abfolge der Operationen des Steuerwerkes:

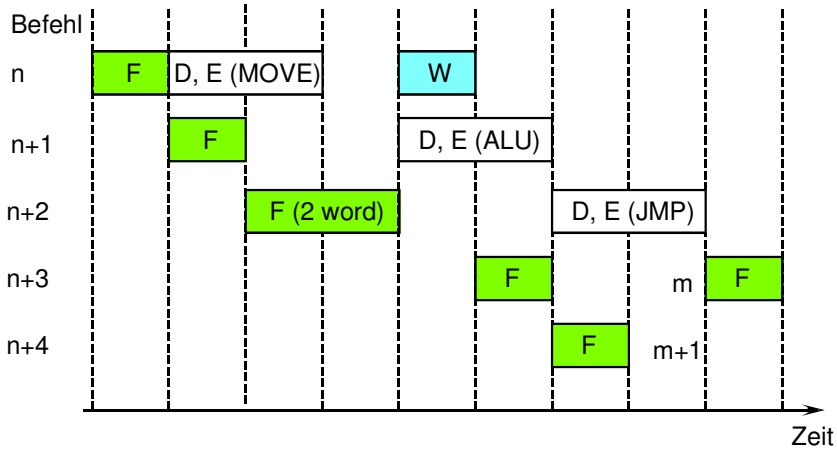
CISC: Complex Instruction Set Computer
Kennzeichen:
- mindestens 2, je nach Befehl auch beliebig viele Takte zur Abarbeitung eines Befehles
- der nächste Befehl wird erst nach vollständiger Abarbeitung des vorigen Befehles geholt.

F: Fetch (Befehl holen) D: Decode (Befehl dekodieren)
E: Execute (Befehl ausführen) W: Write Back (Zurückschreiben)

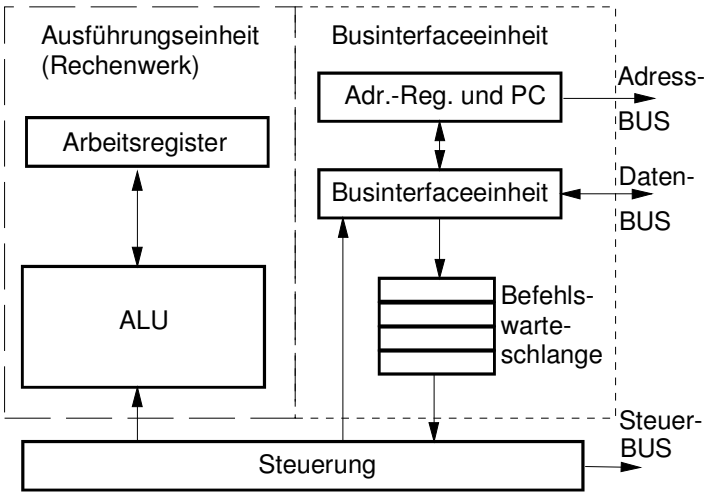


klassische Rechnerarchitektur (CISC)

CISC mit Befehlswarteschlange und Businterfaceeinheit
Kennzeichen:
- mindestens 2, je nach Befehl auch beliebig viele Takte zur Abarbeitung eines Befehles
- der nächste Befehl wird bereits in eine Befehlswarteschlange (FIFO) geschrieben, sobald der Bus nicht genutzt wird (z.B. Ausführungsphase)



CISC Rechnerarchitektur mit Befehlswarteschlange und Businterfaceeinheit

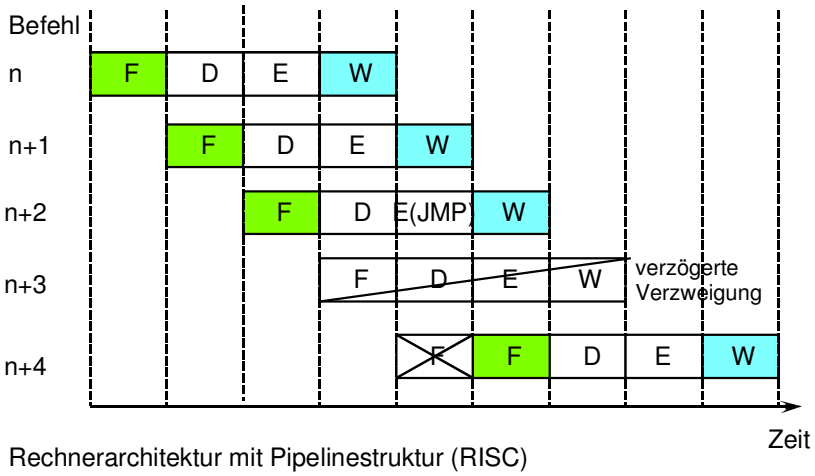


Mikroprozessor mit Businterface-Einheit und Befehlswarteschlange

RISC: Reduced Instruction Set Computer

Kennzeichen:

- genau 4 Takte zur Abarbeitung jedes Befehles
- konsequente Pipeline-Struktur, d.h. alle 4 Zyklen (F, D, E, W) erfolgen parallel
- 4- fache Geschwindigkeit gegenüber der klassischen CISC-Struktur
- eingeschränkter Befehlssatz, keine komplexen Befehle, jeder Befehl in einem Takt ausführbar
- Probleme bei Sprungoperationen, da die neue Befehlsadresse erst mit E Zyklus des JMP Befehls verfügbar ist, sind F, D und E des n+3 Befehles nutzlos.
  → Lösung durch verzögerte Verzweigung, d.h. der nächste Befehl n+3 wird unabhängig vom Sprung immer ausgeführt, nur F des n+4 Befehls wird verworfen.



Rechnerarchitektur mit Pipelinestruktur (RISC)