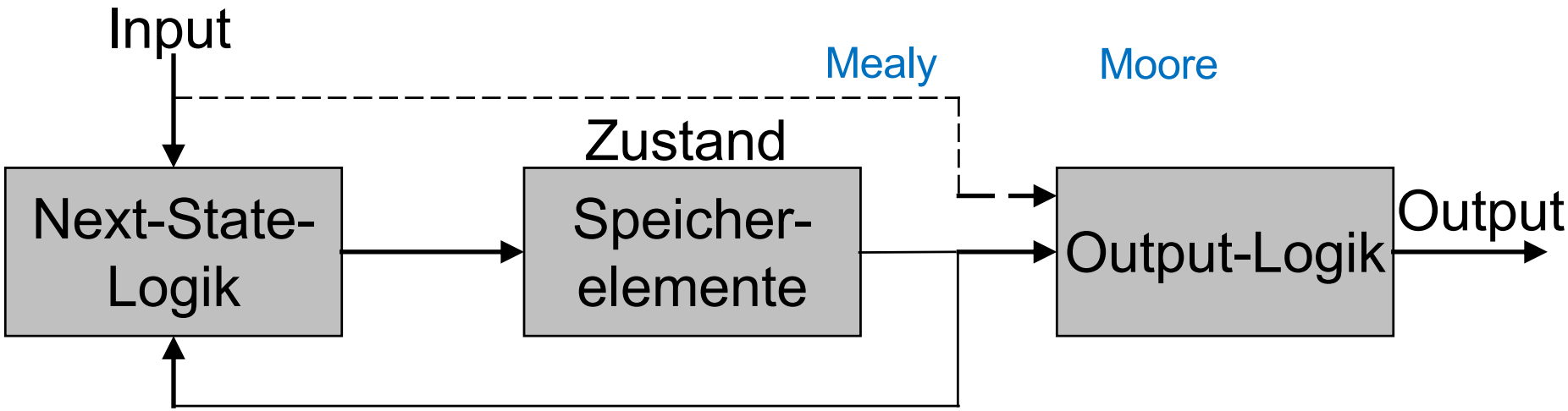
2. Von Neumann Architektur

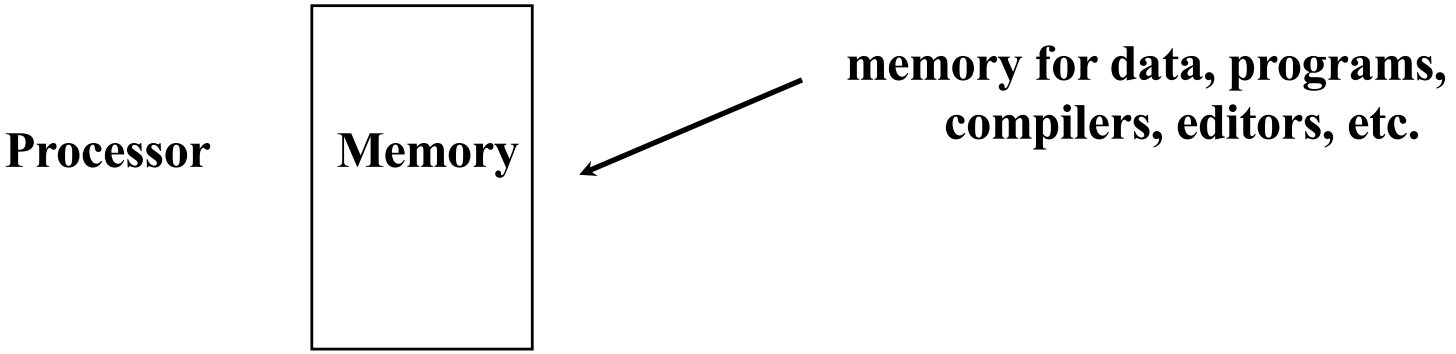
Sequenzieller Rechner



- Beschreibung durch endlichen Automaten (finite-state machine, FSM)
- ungeeignet zur Beschreibung von realen Rechnern. Probleme:
  - Verarbeitung / Speicherung grosser Datenmengen und Transport dieser Datenmengen zwischen Rechnermodulen kann so nicht abgebildet werden.
  - Ein Rechner soll sein Verhalten ändern können.
    Die Funktionsweise dieses Automaten wäre aber vorgegeben.

Von Neumann Konzept

- Anweisungen sind Bitfolgen
- Programme werden im Speicher gehalten
  — gelesen und geschrieben wie *normale* Daten



- Fetch & Execute Cycle
  - Anweisungen werden aus dem Speicher geholt (fetch) und in ein **Spezialregister** abgelegt
  - Bits im Register **kontrollieren** die folgenden Aktionen (execute)
  - Hole (fetch) die nächste Anweisung und fahre fort…

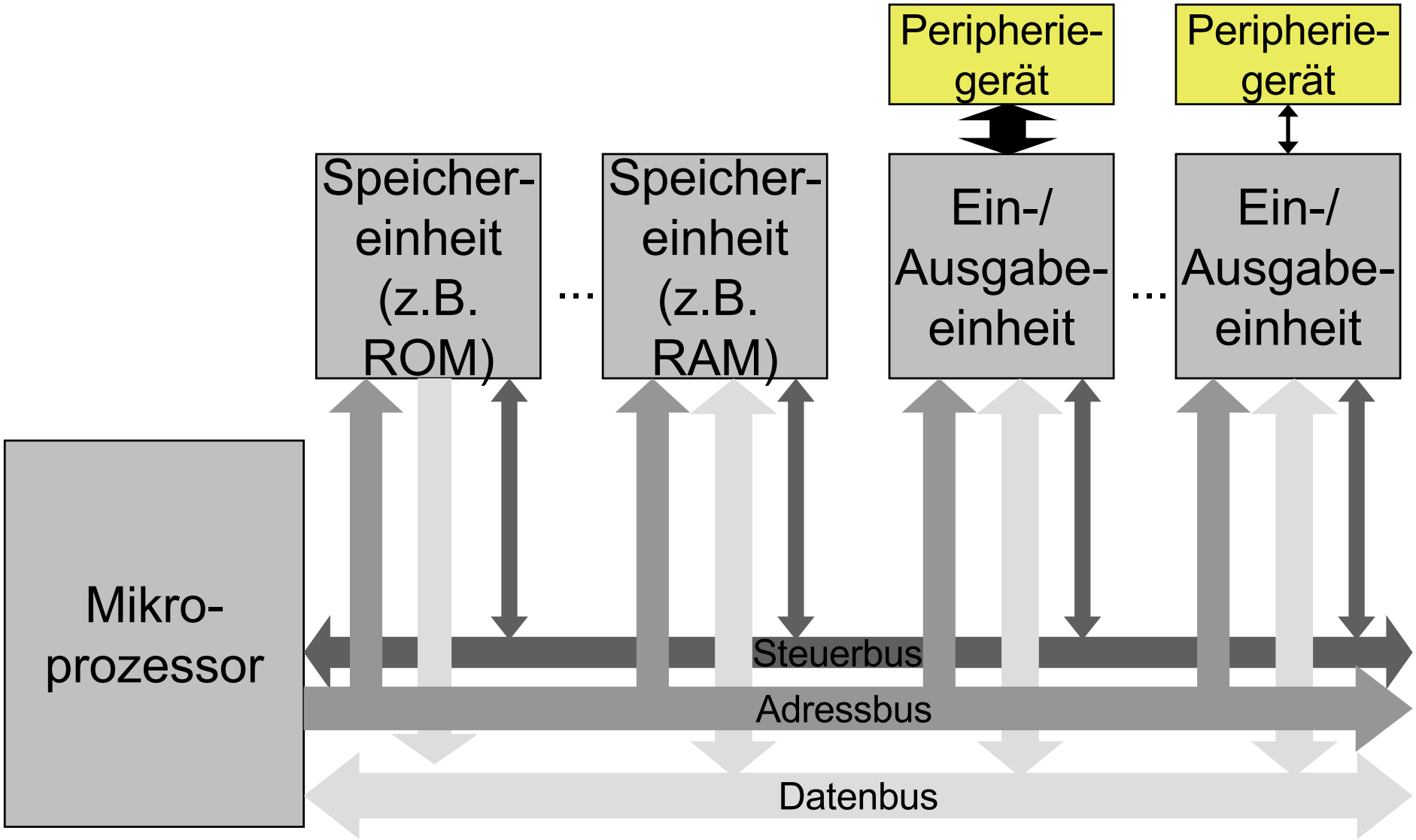
Modell eines Rechners

- Grundbestandteile eines Rechners
  - Zentraleinheit (Central Processing Unit, CPU)
  - Speicher
  - Ein-/Ausgabeeinheiten
- Problem-unabhängige Rechnerstruktur
  - Für jedes neue Problem wird ein eigenes Programm im Speicher abgelegt.
  - Programm-gesteuerter Universalrechner
- Speicher für Programme und Daten
  - besteht aus Plätzen fester Wortlänge
  - Ansprechen über Adressen

Kennzeichen eines von-Neumann-Rechners

- Bearbeitung eines speziellen Problems erfolgt durch ein Programm (Befehlsfolge).
- Befehl: Binärzahl mit festem Format
- Daten und Programme werden nicht in getrennten Speichern untergebracht, ohne weitere Massnahmen besteht kein Schutz vor inkorrektem Zugriff.
- Alle Speicherworte können als Daten, Befehle oder Adressen verwendet werden.
- Zu jedem Zeitpunkt führt die CPU genau einen Befehl aus, welcher höchstens einen Datenwert bearbeiten kann.
  Klassifikation nach Flynn: Single Instruction Single Data, SISD.

Struktur eines von-Neumann-Rechners



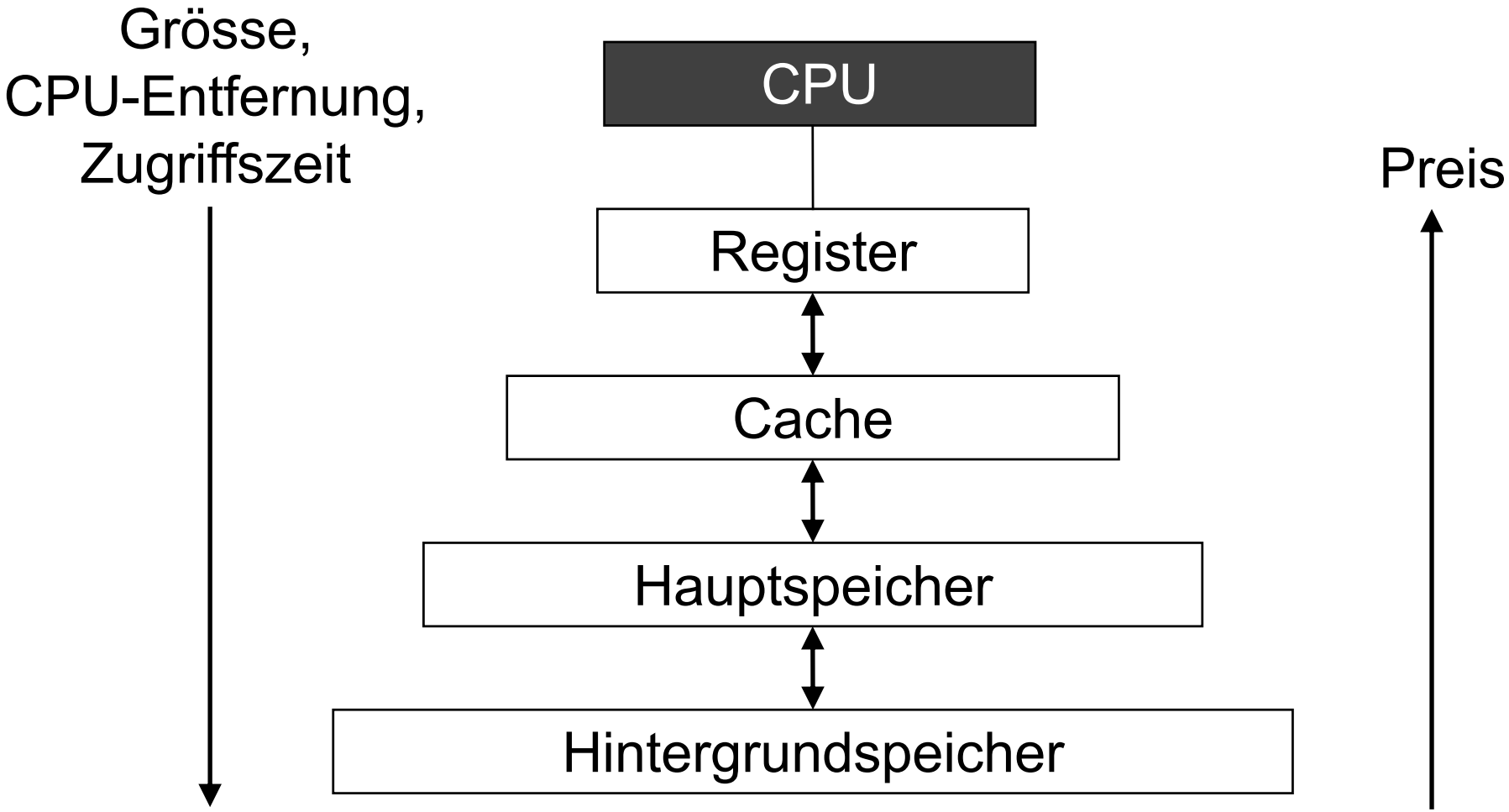
Rechnerkomponenten

- CPU (Mikroprozessor)
  - Verarbeiten von Daten durch ein Programm
  - Steuerwerk/Leitwerk
    - Lesen / Interpretieren von Befehlen und Operanden
    - Ablaufsteuerung
    - Ausführung von Befehlen
    - Ansteuerung der Ein-/Ausgabeeinheiten und des Hauptspeichers
  - Rechenwerk
    - Zwischenspeicherung
    - logische u. arithmetische Operationen
- (Haupt-/Arbeits-)Speicher
  - Speichern von Daten und Programmen
- Ein-/Ausgabeeinheiten
  - Schnittstelle zwischen Mikroprozessor und Peripheriegeräten
  - Einlesen/Ausgabe von Daten von/an Peripheriegeräte
  - Anpassung der Formate und Geschwindigkeiten bei Datenübertragung
  - passiver Interface-Baustein/ Prozessor, Register

Busse

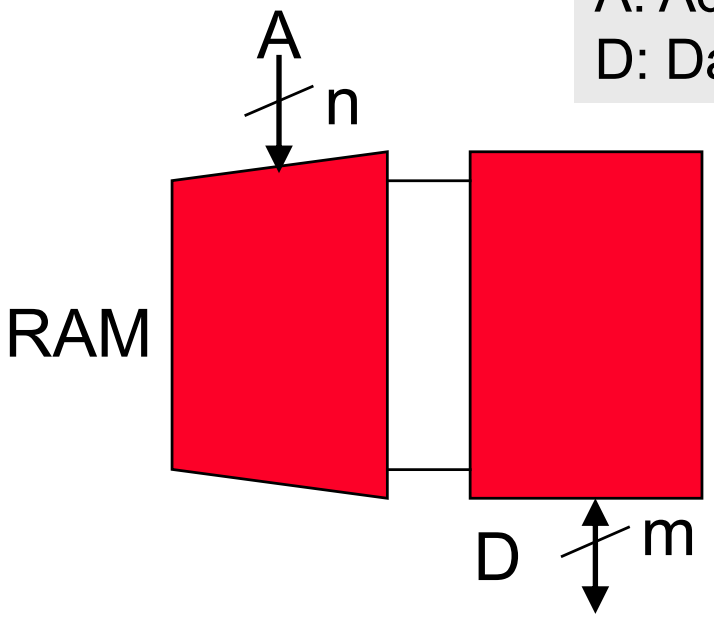
- Bus = Verbindungsweg zwischen Systemkomponenten
- Bündel von funktional zusammengehörenden Signalleitungen, auf denen fest formatierte Bitfolgen transportiert werden müssen
  - seriell (1-Bit-Leitung, billige Lösung)
  - parallel (parallele Übertragung mehrerer Bits)
- Informationsarten
  - Daten
  - Adressen
  - Steuersignale
- Zusammenfassung von Daten-, Adress- und Steuerbus zu Systembus
- Busse können mehrere Rechnerkomponenten verbinden → Synchronisation erforderlich
- Mikroprozessor (CPU) als aktive Komponente (Master) steuert Bussystem.
- Speicher und E/A-Einheiten sind in der Regel passiv (Slaves).
- Zwei Systemkomponenten werden gleichzeitig auf den Bus geschaltet (je ein Sender und Empfänger).
- synchrone oder asynchrone Arbeitsweise

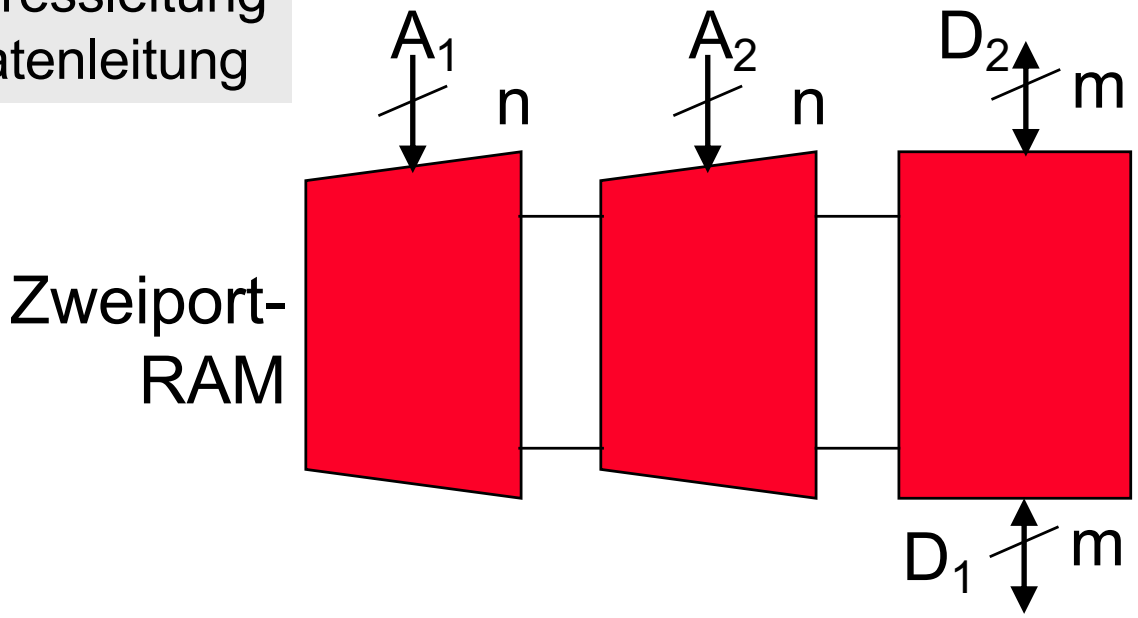
Speicherhierarchie

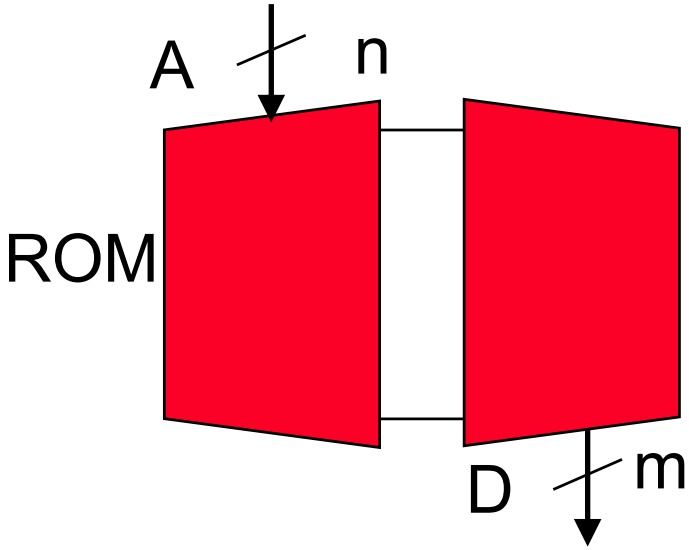


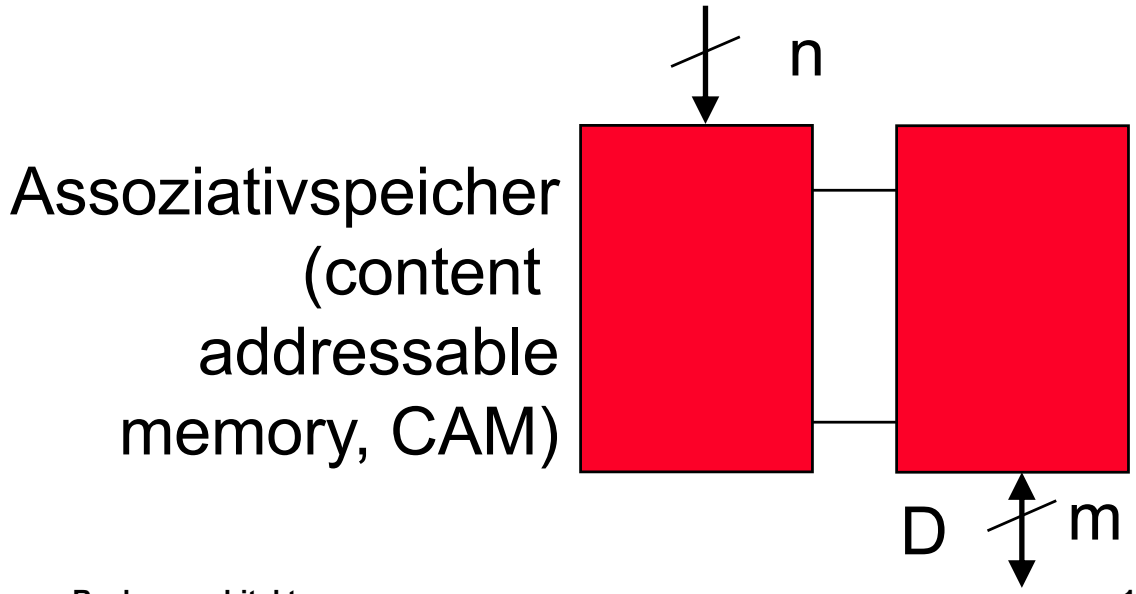
Speichersymbole

A: Adressleitung
D: Datenleitung

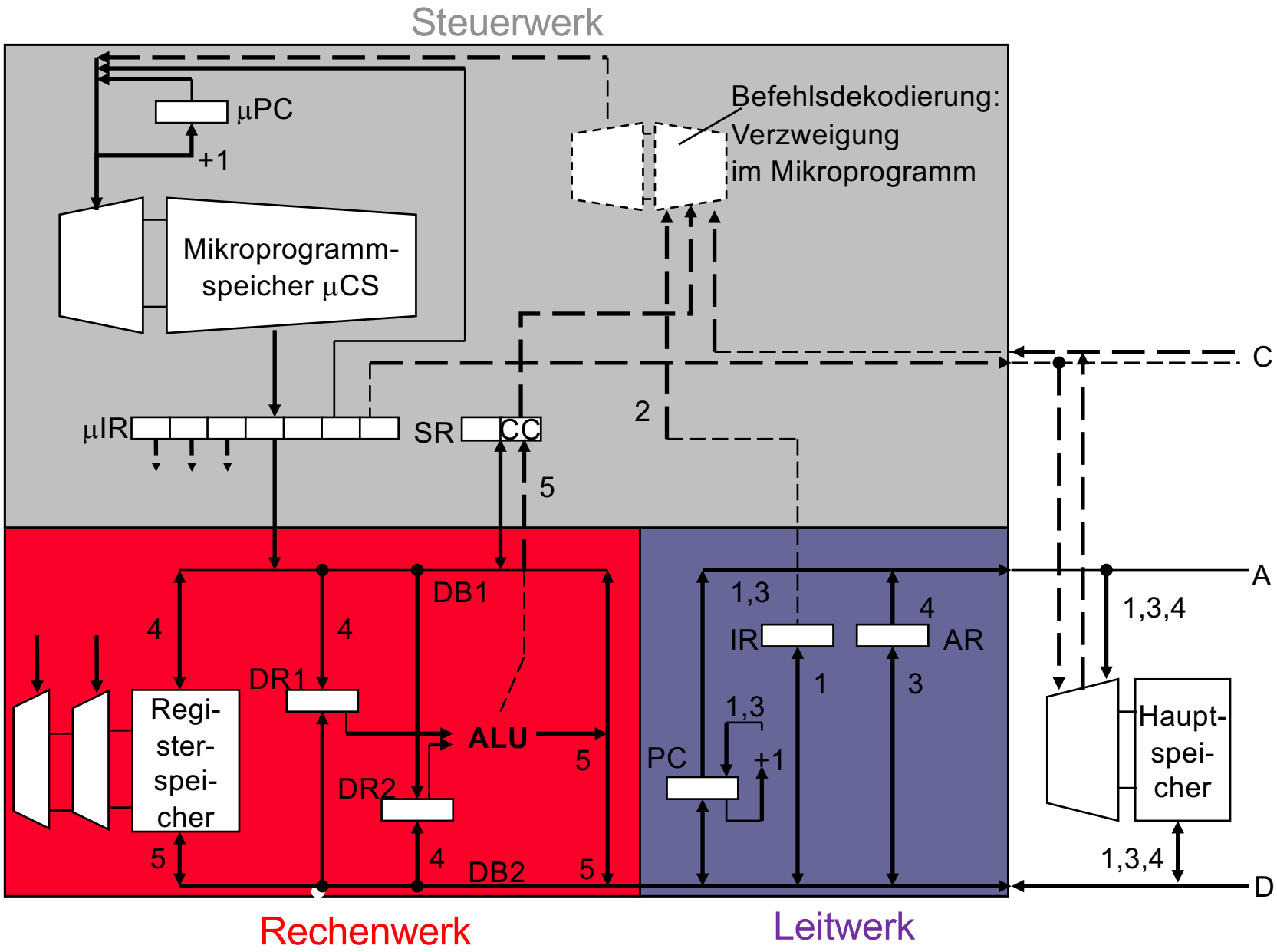








Prozessorstruktur



Mikroprozessor

- Operationswerk/Rechenwerk
  - Aufgabe: Ausführen von Berechnungen
  - Registerspeicher (2-Port-Speicher, für Zwischenergebnisse)
  - Operandenregister DR1, DR2
  - 2 unabhängige interne Datenbusse DB1, DB2 (mit D verbunden)
  - Prozessorstatusregister SR (Overflow, Carry)
  - Arithmetic and Logical Unit (ALU)
    - Verknüpft die zu Ausführungsbeginn in DR1 und DR2 geladenen Operanden
    - Erzeugt Resultat u. Statusinformation (Condition Code für Programmverzweigungen)
    - Quelle und Ziel von Datentransporten: Haupt- oder Registerspeicher
- Steuerwerk
  - Festwertspeicher für Befehlsdekodierung (Erzeugen von Mikroprogramm-Startadressen)
  - Mikroprogramm-speicher (μCS)
  - Mikrobefehlszähler (μPC)
  - Mikrobefehls-register (μIR)
- Leitwerk
  - Befehlszähler (PC)
  - Instruktionszähler (IR)
  - Adressregister (AR)

Mikroprozessorbefehlssatz

- Programm (im Hauptspeicher) legt die Funktionsweise eines Mikroprozessorsystems fest.
  - Folge von Binärzahlen nach festem Format (Maschinencode), schwer lesbar
  - Benutzerfreundliche Darstellung: Assemblersprachen mit speziellem Mnemocode für jeden Befehl
- Befehle für
  - Datentransport
  - Arithmetische und logische Verknüpfungen
  - Änderung der Abarbeitungsreihenfolge
- Befehlssatz legt Art der möglichen Befehle fest.
  - Complex Instruction Set Computer (**CISC**)
    - Beispiele: Intel i386, Motorola MC680x0, MIPS R3000, Intel Pentium
  - Reduced Instruction Set Computer (**RISC**)
    - Beispiele: Sun (Ultra)SPARC, DEC Alpha, Motorola PowerPC, MIPS R10000

(Maschinen-)Befehlszyklus

= Ablauf für Befehlszugriff und -ausführung

- Taktgenerator erzeugt Prozessor-(Maschinen-)takt (typische Taktfrequenz: mehrere 100 MHz / 1 GHz)
- Beispiel: zweistellige Operation
  1. Transport des Befehls vom Hauptspeicher in Befehlsregister, Erhöhen des PC
  2. Befehlsdekodierung
  3. Transport des 1. Operanden von Haupt- oder Registerspeicher in Operationswerk
  4. Transport des 2. Operanden in Operationswerk
  5. Operationsausführung (Verknüpfen der Operanden)
  6. Transport des Resultats vom Rechenwerk in Haupt- oder Registerspeicher

Beispiel Befehlszyklus

- ADD SPADR, R5

• PC → Hauptspeicher → IR, PC+1 → PC	• Lesen des 1. Befehlsworts
• Befehlsdekodierung	• Auswerten Op-Code und Adressierungsarten
• PC → Hauptspeicher → AR, PC+1 → PC	• Lesen Adresse 1. Operand
• AR →Hauptspeicher →DR2, Registerspeicher → DR1	• Lesen der Operanden
• DR2 + DR1 → Registerspeicher, Statusinformation → SR	• Addition, Schreiben des Resultats in Registerspeicher u. Status-info nach SR (CC-Bits)

CISC-Mikroprozessor

- in den 70er Jahren: Ausstattung der Prozessoren mit immer mächtigeren Befehlssätzen
  - Ziel: Verringern der semantischen Lücke zwischen höheren Programmiersprachen und einfachen Maschinenbefehlen
- typisch: > 200 Befehle
- grosse Anzahl von Adressierungsarten
- viele Kombinationen von Befehlen und Adressierungsarten
- Mikrocode für jeden Befehl in Steuerwerk
- Mikroprogrammierung des Steuerwerks ist langsamer als feste Verdrahtung.
- Versuch, CPU durch komplexe Instruktionen stärker zu belasten (Speicherbus als Flaschenhals)
- Viele Instruktionen und Adressierungsformen werden sehr selten verwendet.

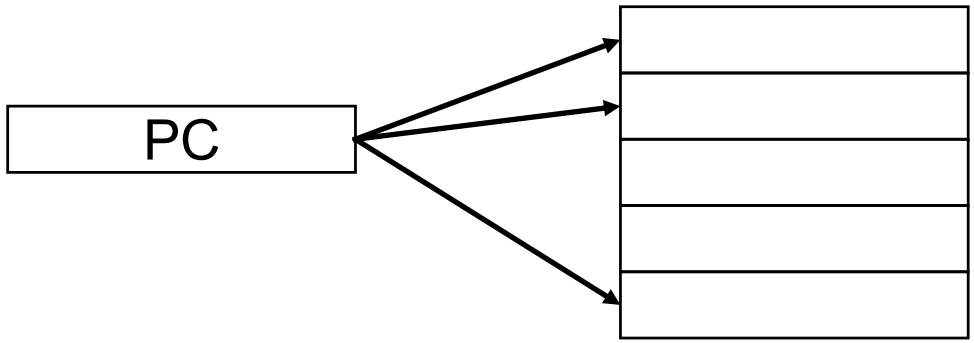
RISC-Mikroprozessor

- Reduced Instruction Set Computer
- Fixe Befehlslänge
- Wenige Befehle
- Wenige Addressierungsarten
- Optimiert, damit Compiler schnellen Code produzieren können

MIPS, Sun SPARC, HP PA-RISC, IBM PowerPC, Intel (Compaq) Alpha, …

Befehlszähler

- Program Counter, PC
- enthält Adresse des nächsten Befehls
- Vielfaches von Bytes oder Halbworten
- Verändern des PC
  - Inkrementieren
  - Überschreiben bei Sprungbefehl



Stackpointerregister

- Adressierung eines Keller-(Stapel-)speichers
- Stapelelemente können nur oben aufgelegt und entfernt werden.
- Inkrementieren/Dekrementieren des Stackpointers
- Benutzung auch bei Unterprogrammsprüngen
- oft: getrennte Benutzer/System-Stacks (USP/SSP)