Dr.-Ing. Oliver Knodel

Architekturkonzepte und Rechnerarchitekturen

Von-Neumann, Harvard, Princeton

Dresden // April, 2024

Architekturkonzepte — Hauptbestandteile

Rechner sind dem Prinzip nach nichts anderes als hochkomplexe, oft unüberschaubare, endliche Automaten, also elektrische Schaltwerke.

Hauptbestandteile von Schaltwerken sind:

- Ein-, Ausgangsklemmen,
- Verknüpfungsglieder,
- Schaltnetze,
- Speicherglieder,
- Verbindungsnetzwerk.

Die Anzahl der innere Zustände eines Schaltwerkes ergibt sich direkt aus der Anzahl der vorhandenen Speicherglieder.

Gegenüberstellung: Schaltwerk — Rechner

	„einfache“ Schaltwerke	Rechner
Ein-, Ausgänge	endliche Anzahl	nahezu unendliche Vielfalt
Schaltwerke	einfache Logik	komplex, verteilt
Speicherglieder	konzentriert angeordnet	über das System verteilt
innere Zustände	überschaubar, gering	unüberschaubar, extrem
Vernetzung	einfache Rückführungen	hochkomplex, lokal, global
Darstellung	Logische Schaltung	hochkomplexe Systeme
Beschreibung	Boolesche Gleichungen	Automaten-, Architekturmodelle

Rechner können aufgrund ihrer Komplexität nicht wie Schaltwerke dargestellt und beschrieben werden. Die Darstellung und Beschreibung von Rechnern erfolgt daher auf einem relativ hohem Abstraktionsniveau (Architekturebene).

Abgrenzung: Rechner- und Prozessorarchitektur

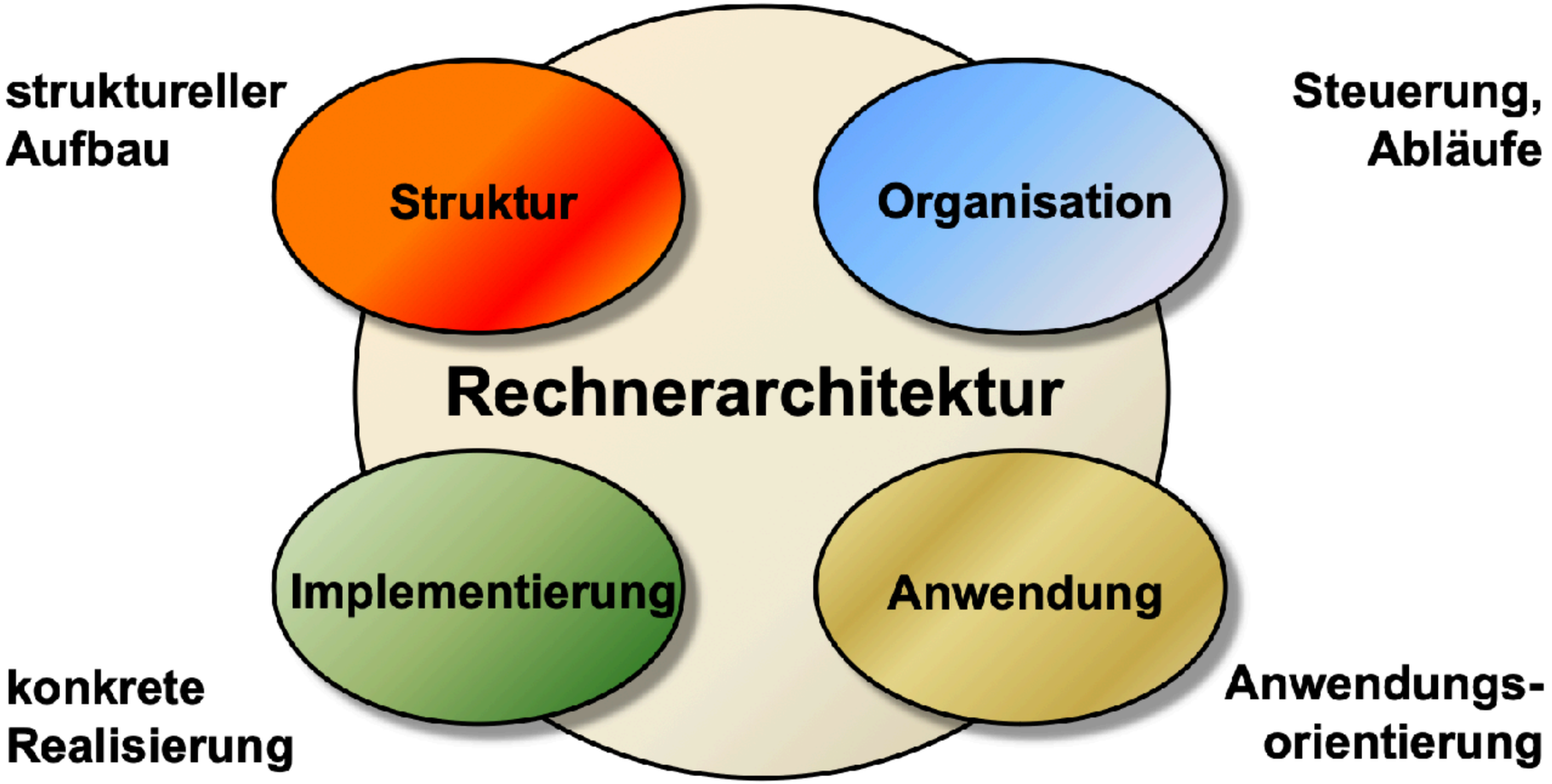
Rechnerarchitektur

- Analyse und Synthese von Rechnerstrukturen und deren Organisationen
- Aufbau und Eigenschaften von Rechnern als einheitliches System
- Untersuchung des äußeren Erscheinungsbildes eines Rechners
- Weitestgehende Abstraktion von inneren Vorgängen des Rechners
- Klassifizieren, vergleichen, bewerten und entwerfen von Rechnern
- Komponenten und deren Verbindungen in der Struktur von Rechnern
- Kommunikations- und Verarbeitungsstruktur und deren Organisation
- Implementierung und Verhältnis von Hard- und Softwarerealisierung

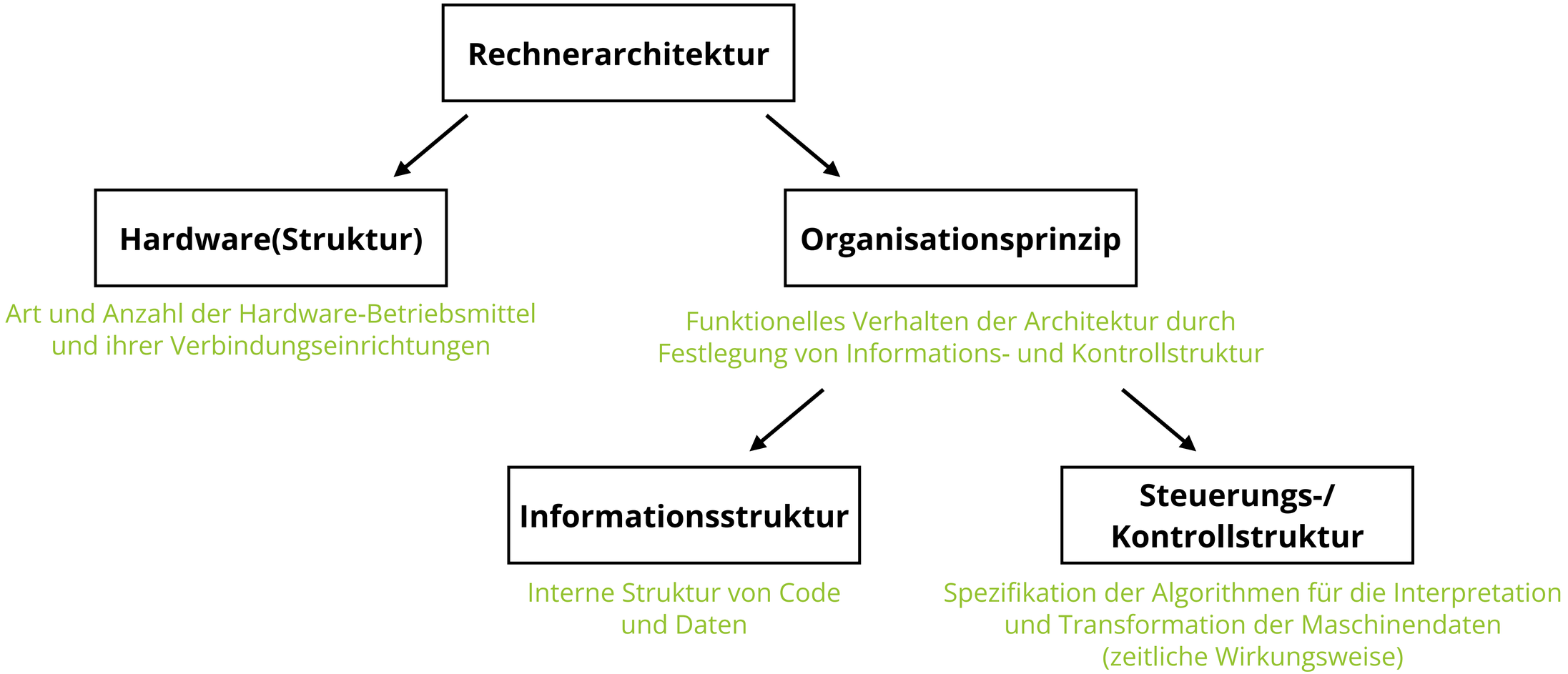
Prozessorachitektur

Beschränkung auf den Prozessor (auch Mikroprozessor, Mikrocontroller) als integrierte Verarbeitungseinheit und seine Verbindungen nach außen.

Bereiche der Rechnerarchitektur



Aufteilung der Rechnerarchitektur in Komponenten



Begriffsdefinition

Allgemeine Architektur-Definition nach Tanenbaum

Die auf jeder Ebene verfügbaren Datentypen, Operationen und Merkmale nennt man **Architektur** (Architecture). Die Architektur betrifft die Aspekte, die für den Benutzer der jeweiligen Ebene sichtbar sind. …"

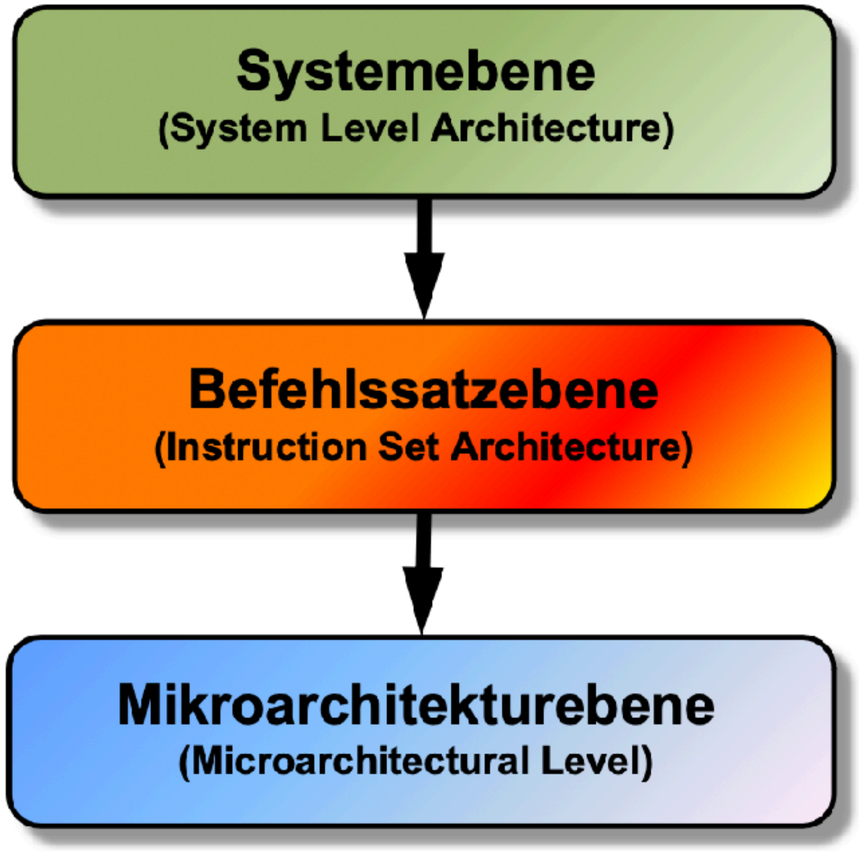
"… **Implementierungsaspekte**, etwa welche Chiptechnik zur Speicherimplementierung verwendet wird, sind **nicht** Teil der Architektur. …"

"… Im praktischen Umfeld bedeuten Computer-Architektur und Computer-Organisation das gleiche."

Darstellungs- und Beschreibungsebenen

Ebene	Darstellung / Beschreibung
Architekturebene:	System- und Verhaltensdarstellung Hardware-, Architekturbeschreibungssprachen
Register-Transfer-Ebene:	Vernetzung von Komponenten, Module, Busse Hardwarebeschreibungssprachen
Logikebene:	Vernetzung von Verknüpfungs- und Speichergliedern Netzlisten, Logikplan
Schaltungsebene:	Bauelemente, Verbindungen, Technologie elektrische Netzwerke, Strom, Spannung, Ladung

Unterteilung nach Ebenen innerhalb der Rechnerarchitektur

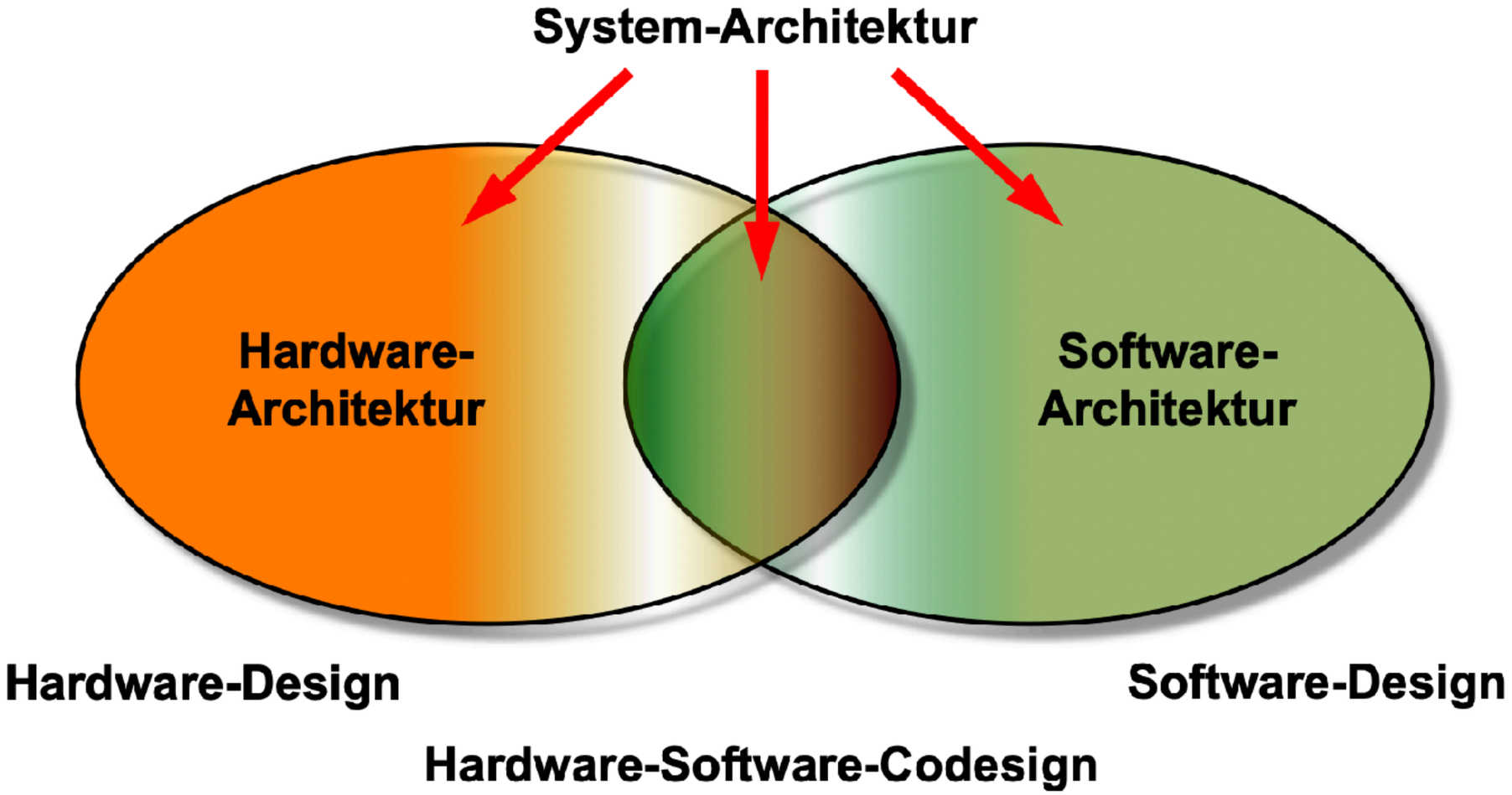


- Komponenten, Konfiguration
- Verbindungstechnik
- Speicher, Peripherie

- Befehlssatz, Befehlsformat
- Adressierung, Datenformat
- Schnittstelle HW-SW

- Mikroprogramm, Steuerung
- Implementierung, Parallelität
- Schnittstellen, Rechnernetz

Architektursichten



Einfache (theoretische) Modelle für Rechner

Für die einfache Beschreibung und Modellierung von Rechnern sind spezielle, erweiterte Automatenmodelle erforderlich.

Turing-Maschine

Einfaches theoretisches Automatenmodell, das vor allem für theoretische Untersuchungen des Rechners gut geeignet ist: Berechenbarkeit – Entscheidbarkeit – Akzeptierbarkeit.

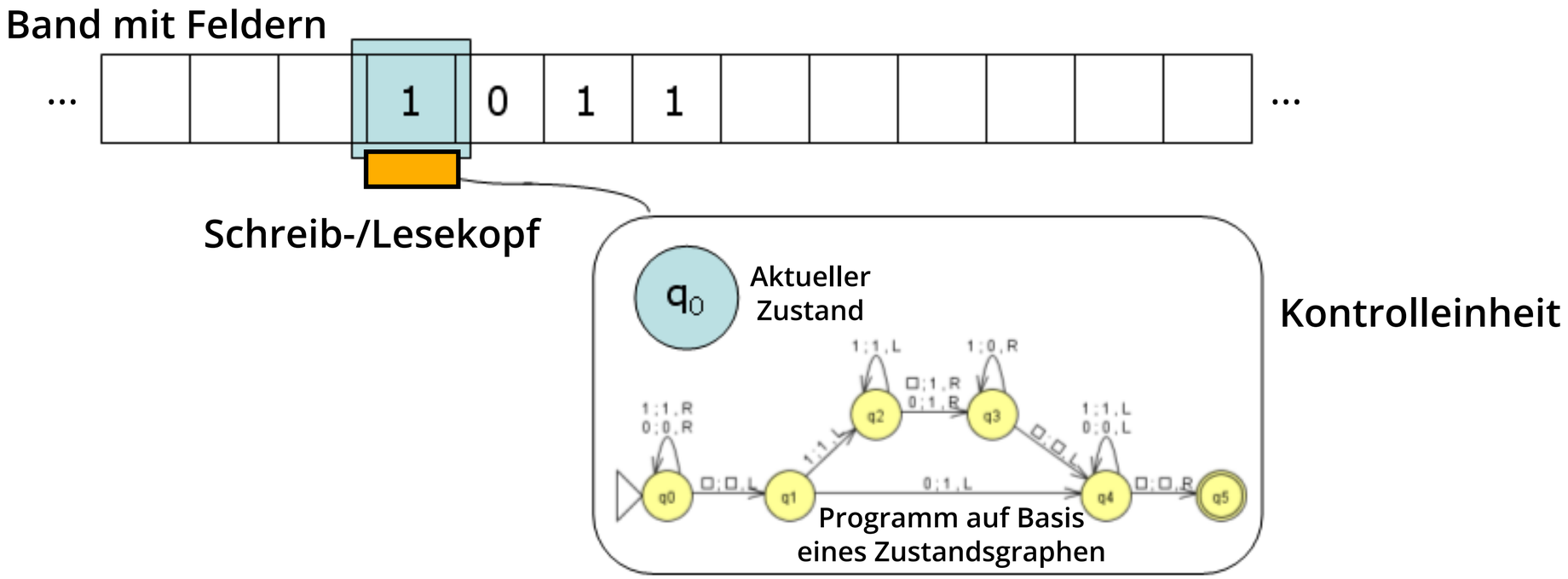
Zur Beschreibung praktischer, realer Rechner nicht geeignet.

von-Neumann-Rechner

Erweitertes Automatenmodell zur Beschreibung voll programmgesteuerter Rechner. Mit diversen Erweiterungen zur Beschreibung praktischer, realer Rechner geeignet.

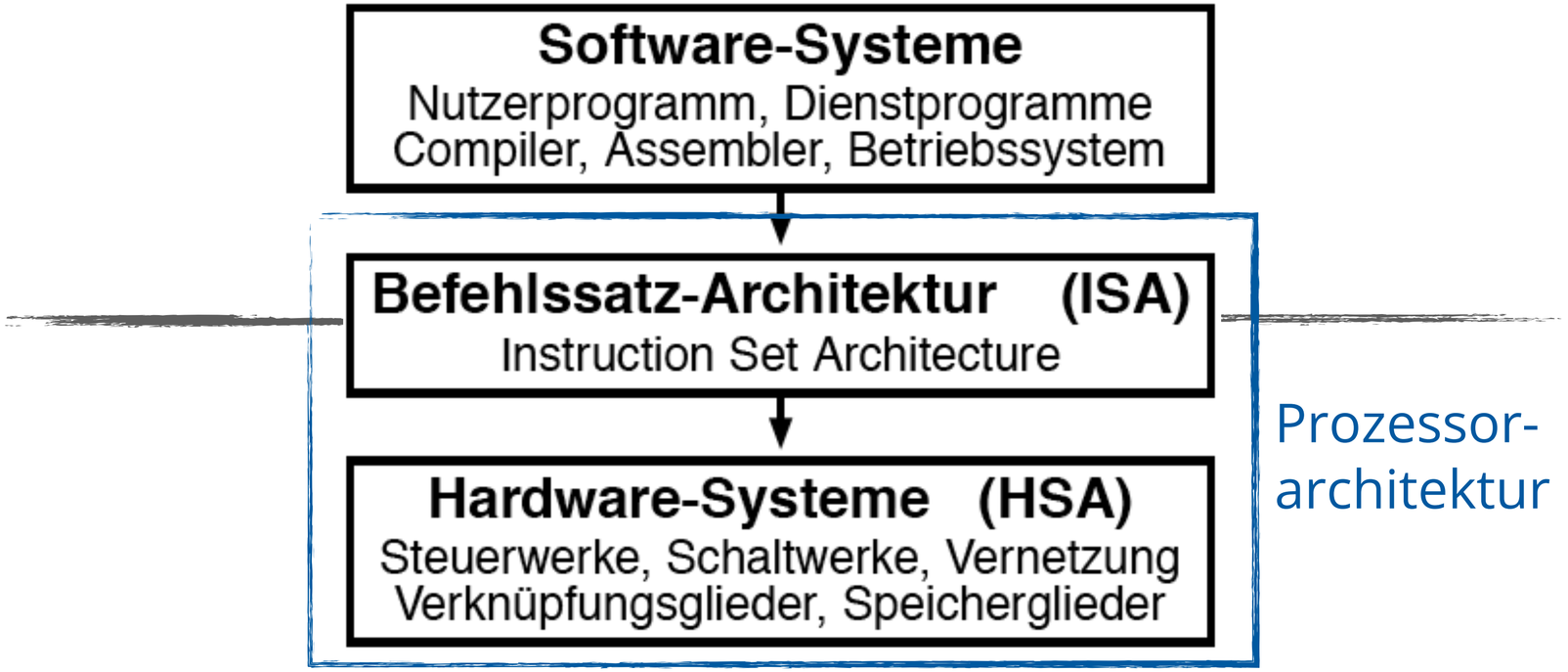
Die Turing-Maschine

- Einfaches theoretisches Modell zur Berechenbarkeit von Funktionen.
- Basis ist ein unendliches Band mit einzelnen Feldern (Bits), ein Schreib-/Lesekopf und eine Kontrolleinheit.
- Die Kontrolleinheit beinhaltet ein Programm und einen internen Zustand



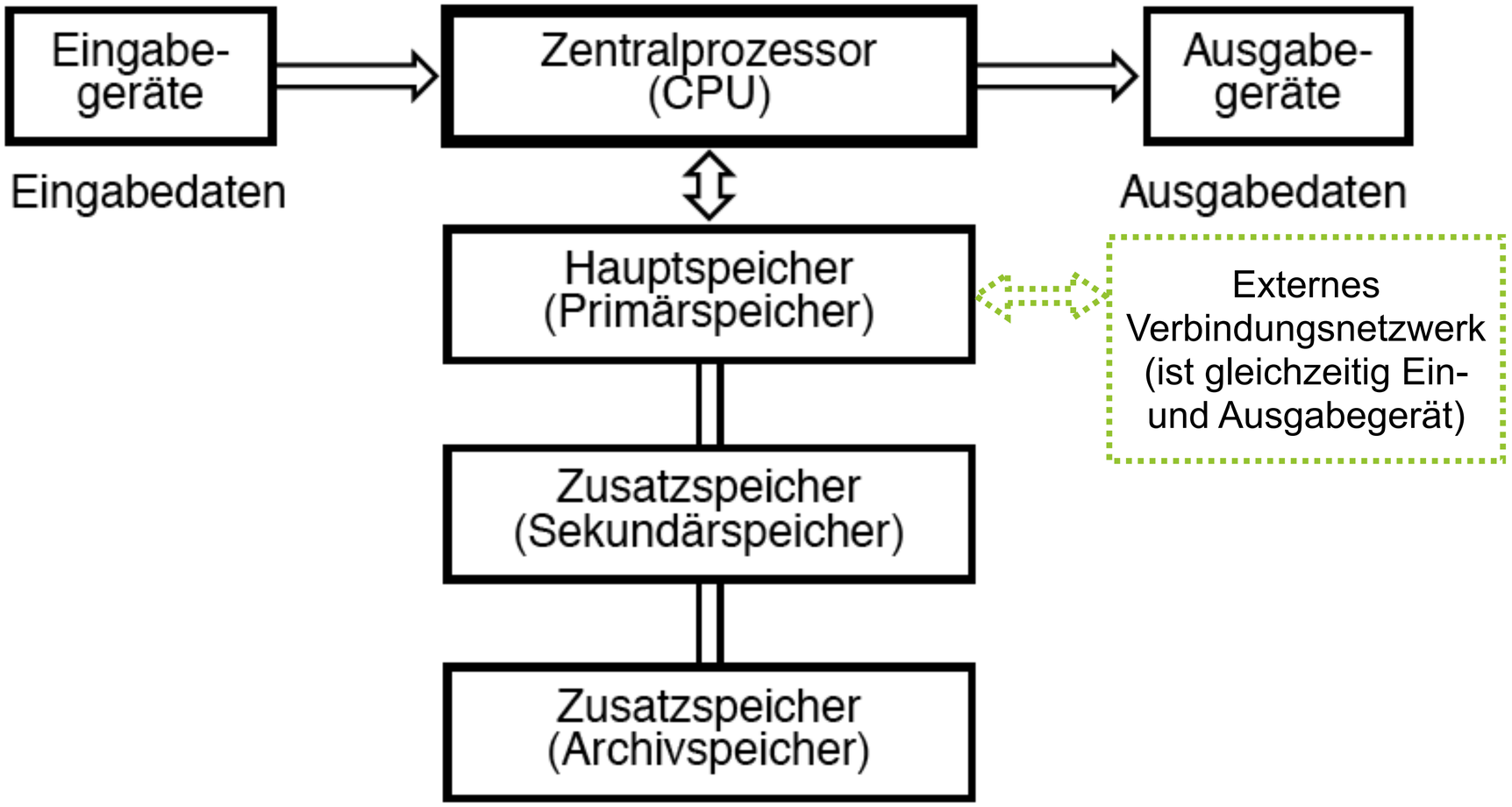
Schichtenmodell eines Rechners

Vereinfachtes Schichtenmodell



Der Befehlssatz als Architekturmerkmal eines Rechners, **Instruction Set Architecture (ISA)**, definiert die Software – Hardware – Schnittstelle, **Software ⇒ Hardware-System-Architecture (HSA)**.

Hauptkomponenten eines Rechners



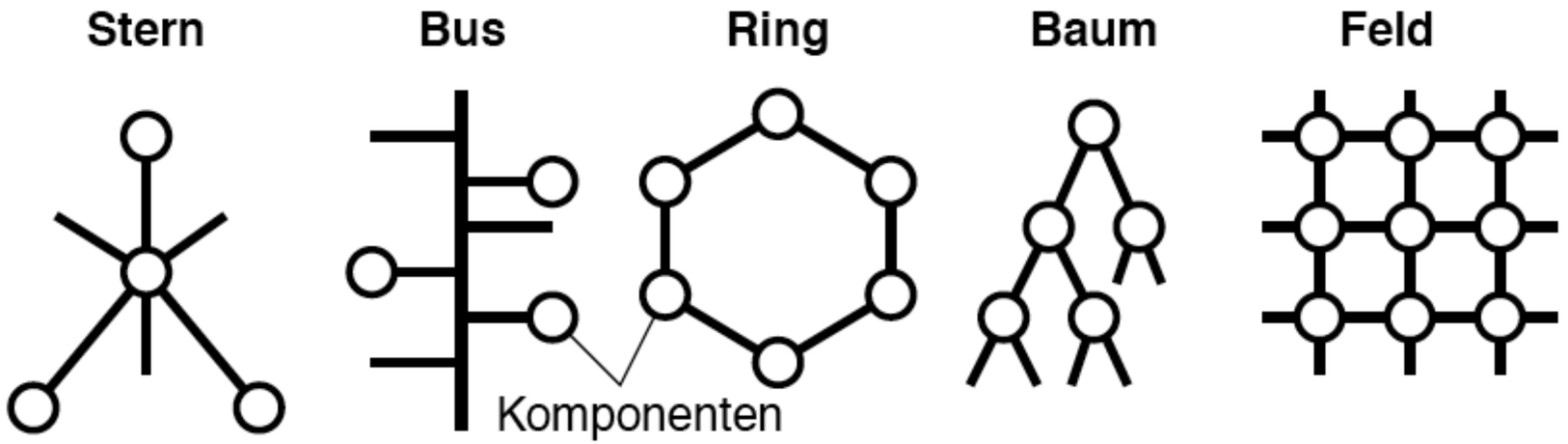
Hauptkomponenten eines Rechners

CPU	Zentrale Verarbeitungseinheit mit Rechenwerk und Steuerwerk (Mikroprozessor)
Primärspeicher	Hauptspeicher (DRAM, SRAM)
Sekundärspeicher	Zusatzspeicher mit direktem Zugriff (Festplatte)
Archivspeicher	Massenspeicher mit verzögertem Zugriff (CD, DVD, Tape)
Systembus	verbindet alle Komponenten des Computers miteinander (PCI-Bus)
Ein-/Ausgabegeräte	realisieren die Kommunikation des Computers nach aussen (Monitor, Keyboard, Maus, Drucker, Netzkarte, Modem, Sound, . . .)

Verbindungsstrukturen

Die Verbindungsstruktur eines Rechners stellte ein Architekturmerkmal bzgl. der Verbindung der einzelnen Rechnerkomponenten untereinander bzw. auch zwichen den Komponenten und der Umwelt dar (Ein-/Ausgabekomponenten).

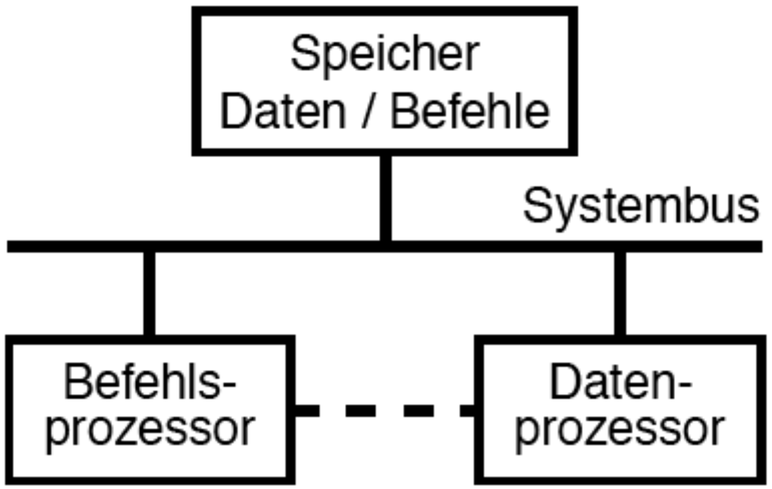
Die Verbindungen können als Stern, Bus, Ring, Baum, Feld, Crossbar (steuerbarer Kreuzschienenverteiler), . . . ausgeführt sein.



Busstrukturen

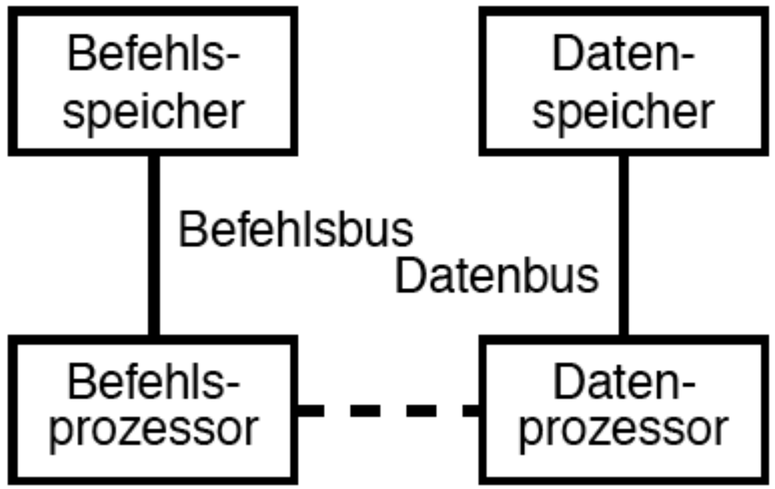
Princeton Architektur

(J.von Neumann)



Harvard Architektur

(H. Aiken)



Die Busse können unterteilt sein: Datenbus, Adressbus, Steuerbus, Befehlsbus.
Weitere Komponenten und weitere abgesetzte Busse sind möglich.

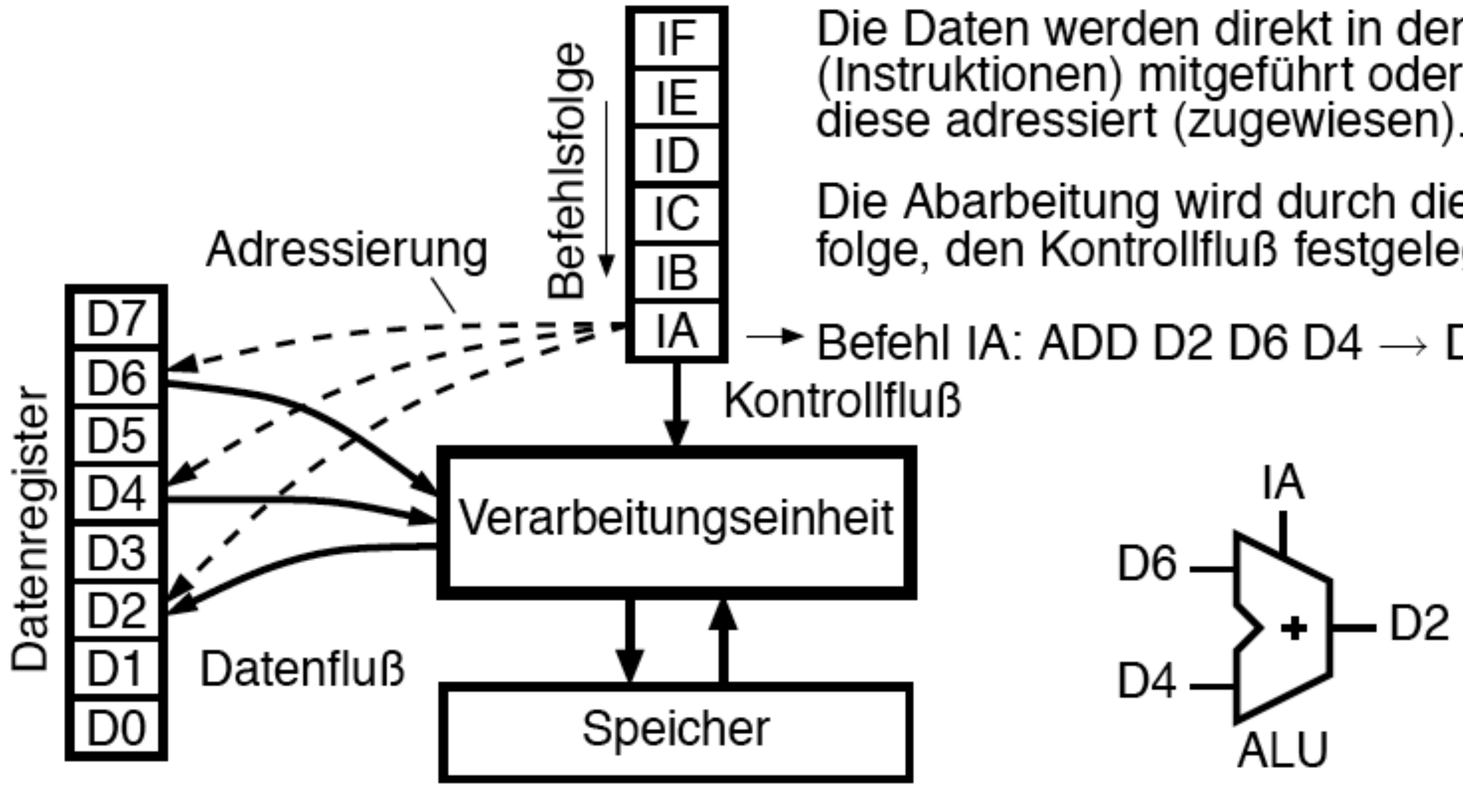
Klassifikation von Rechnerarchitekturen

- Ordnungsstruktur und -prinzip von Rechnerarchitekturen
- Anordnung, Organisation und Verbindungen der Rechnerkomponenten
- Operationsprinzip, Verarbeitungsprinzip, Kommunikationprinzip des Rechners
- Befehls- und Datenflüsse und deren Organisation
- Parallelität auf verschiedenen Ebenen
- Prinzip der Datenspeicherung und -konistenz des Rechners

Eine eindeutige Klassifikation von Rechnerstrukturen ist nicht möglich. Vielmehr werden durch die Klasseneinteilung nur Grundprinzipien hervorgehoben. Die einzelnen Klassen können sich dabei weitgehend überdecken.

Die Zuordnung zu den einzelnen Klassen erfolgt jeweils nach der hervorzuhebenden Haupteigenschaft der Rechnerarchitektur.
Eine Architektur kann somit auch mehreren Klassen zugeordnet werden.

Kontrollflußarchitektur



Die Daten werden direkt in den Befehlen (Instruktionen) mitgeführt oder durch diese adressiert (zugewiesen).

Die Abarbeitung wird durch die Befehlsfolge, den Kontrollfluß festgelegt.

Befehl IA: ADD D2 D6 D4 → D2:=D6+D4

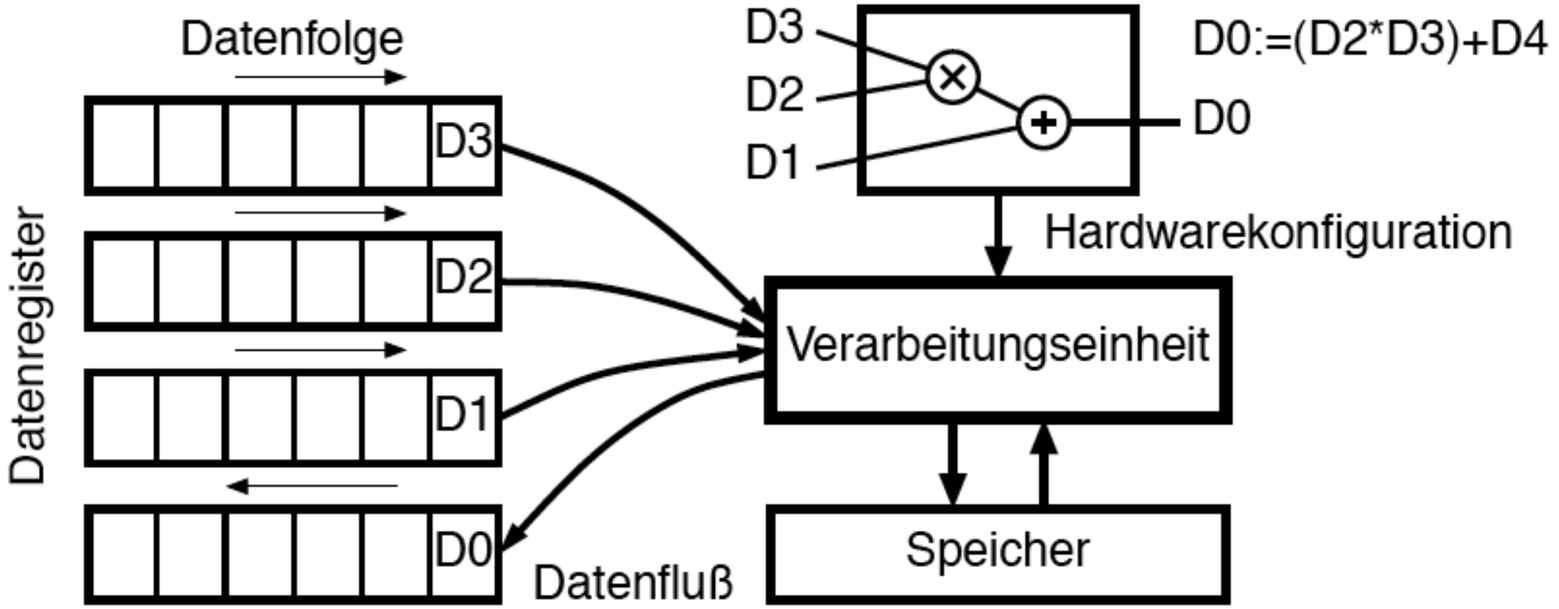
Kontroll- und Datenfluß sind orthogonal. Der Kontrollfluß steuert den Datenfluß.

Der moderne Rechner auf Basis des Von-Neumann Rechners ist eine typische Kontrollflußarchitektur

Datenflußarchitektur

Die Befehle werden direkt im Datenfluß mitgeführt oder in der Verarbeitungseinheit durch den Datenfluß konfiguriert.

Die Abarbeitung wird durch den Datenfluß und die Hardwarekonfiguration bestimmt.



Der Datenfluß enthält implizit den Kontrollfluß, er bestimmt die Abarbeitung.

Datenflussarchitekturen sind als Anwendungsspezifische Spezialarchitekturen in vielen Anwendungen vertreten

Unterschiedung Kontrolle - und Datenflußarchitekturen

Kontrollflußarchitektur

von-Neumann	RISC,CISC - Universalprozessor (GPP)
Harvard	ASIP - Applikationsspezifischer Befehlssatzprozessor
Superscalar	DSP - Digitaler Signalprozessor
VLIW	ASSP Applikationsspezifischer Signalprozessor
	RPA - Rekonfigurierbare Computerarchitektur
	CCM - Custom Computing Machine
	Systolische Feld-Architekturen
	Xputer, Datenflußrechner

Datenflußarchitektur

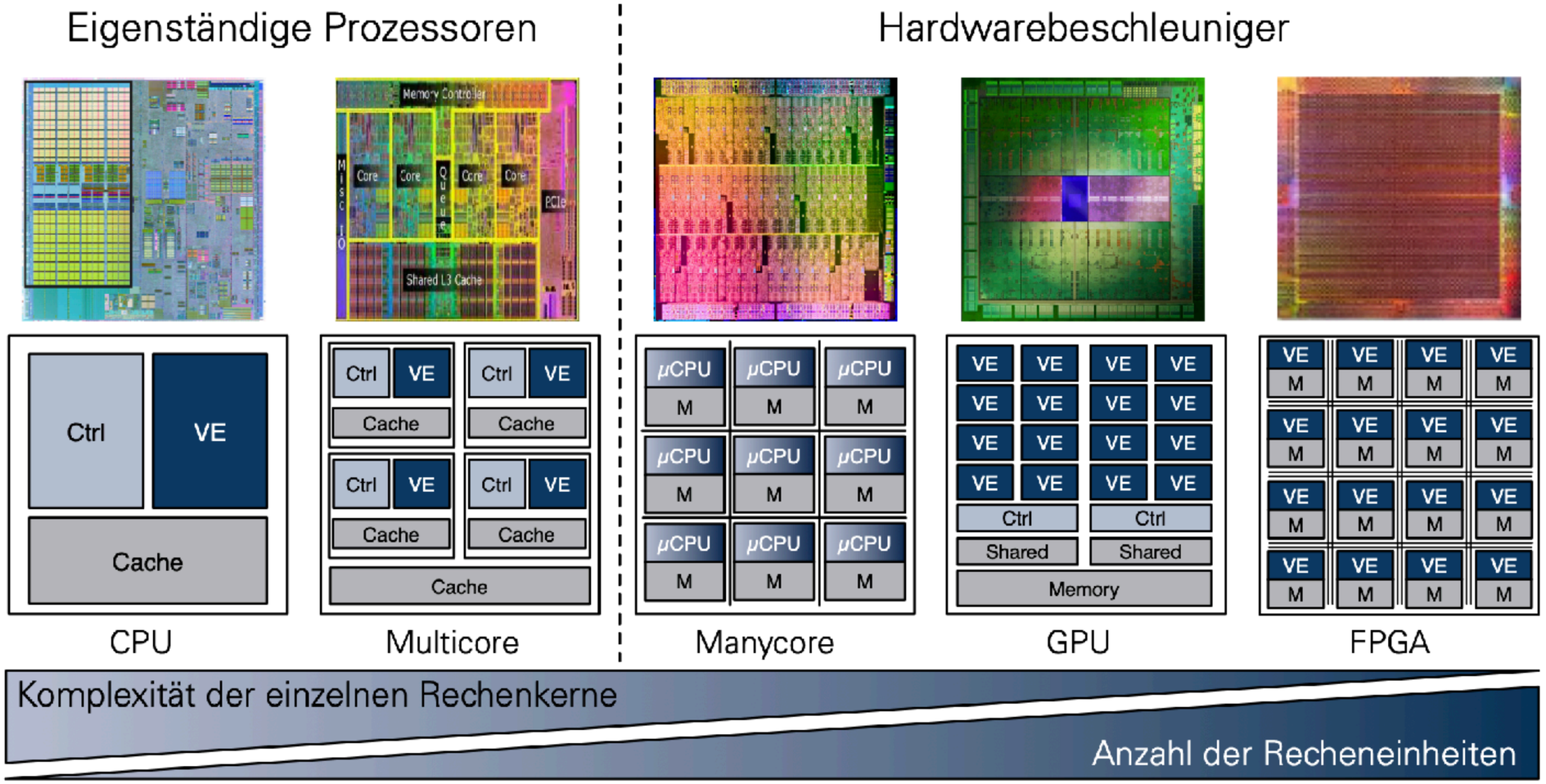
Breites Spektrum von Architekturen zwichen Kontroll- und Datenflußarchitektur.

Parallelitätsebenen in Rechnerarchitekturen

Bitebene: (Bit Level Parallelism)	BLP	Die Bitstellen eines Datums oder mehrere Daten werden parallel verarbeitet.
Befehlsebene: (Instruction Level Parallelism)	ILP	Mehrere Befehle eines Kontrollflusses werden parallel ausgeführt.
Kontrollflußebene: (Thread Level Parallelism)	MT	Parallele Ausführung mehrerer Teile (Threads) eines Kontrollflusses (Multithreaded).
Programmebene: (Application Level Parallelism)	MP	Parallele Ausführung mehrere Programme, Prozesse (Multiprocessing).
Datenflußebene: (Data Flow Processing)	DFP	Die Datenflußverarbeitung ist vom Konzept her hochparallel.

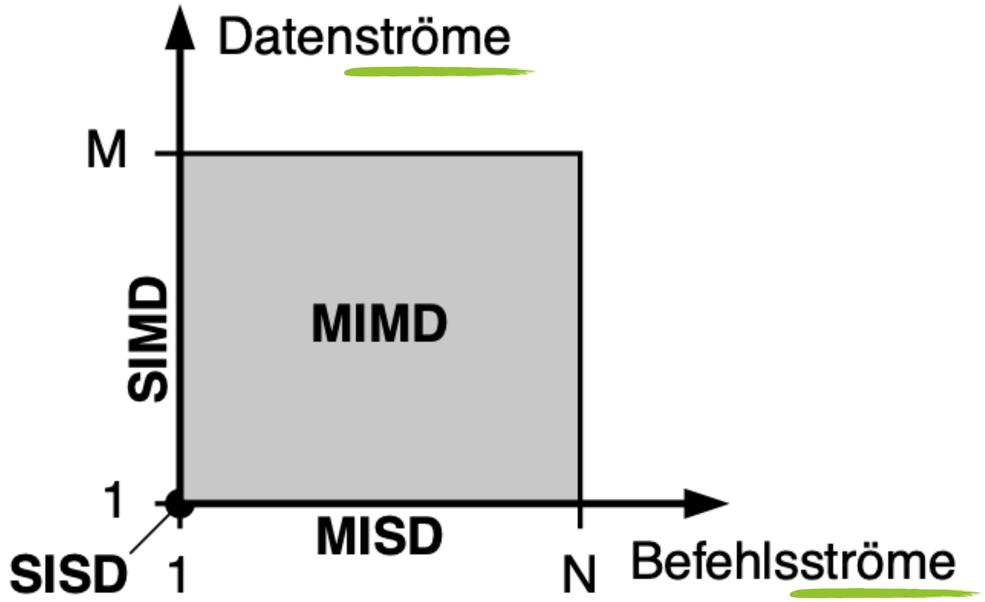
Bit- und Befehlsebenenparallelität werden feinkörnig (fine grained) bezeichnet, Kontrollfluß- und Prozeßparallelität dagegen grobkörnig (coarse grained).

Spezielle Prozessoren und Hardwarebeschleuniger



Klasssifikation von Parallelarchitekturen nach Flynn (1972)

Zweidimensionale Klasseneinteilung der Rechnerarchitekturen nach der Anzahl der Befehls- und Datenströme.



Befehlsströme	Datenströme: Single Data	Datenströme: Multiple Data
Single Instruction	**SISD**	**SIMD**
Multiple Instruction	**MISD**	**MIMD**

Klasseneinteilung nach Flynn

SISD Single Instruction – Single Data (von-Neumann Rechnerkonzept).

MISD Multiple Instruction – Single Data, Grafikkarten (GPUs), Erweiterung zu **SPMD** — Single Program Multiple Data

SIMD Single Instruction – Multiple Data

Feldrechner: Parallele Anordnung von gleichartigen Verarbeitungseinheiten, die alle gleichzeitig den selben Befehl ausführen.

Vektorrechner: Anordnung der Verarbeitungseinheiten als Pipeline zur zeitlich überlappten Befehlsabarbeitung (Vektor-, Matrixoperationen).

MIMD Multiple Instruction – Multiple Data, Multiprocessor und Multicomputer.

SMT: Simultaneous Multithreaded

MTA: Multithreaded Architecture

MPP: Masively Parallel Multiprocessing

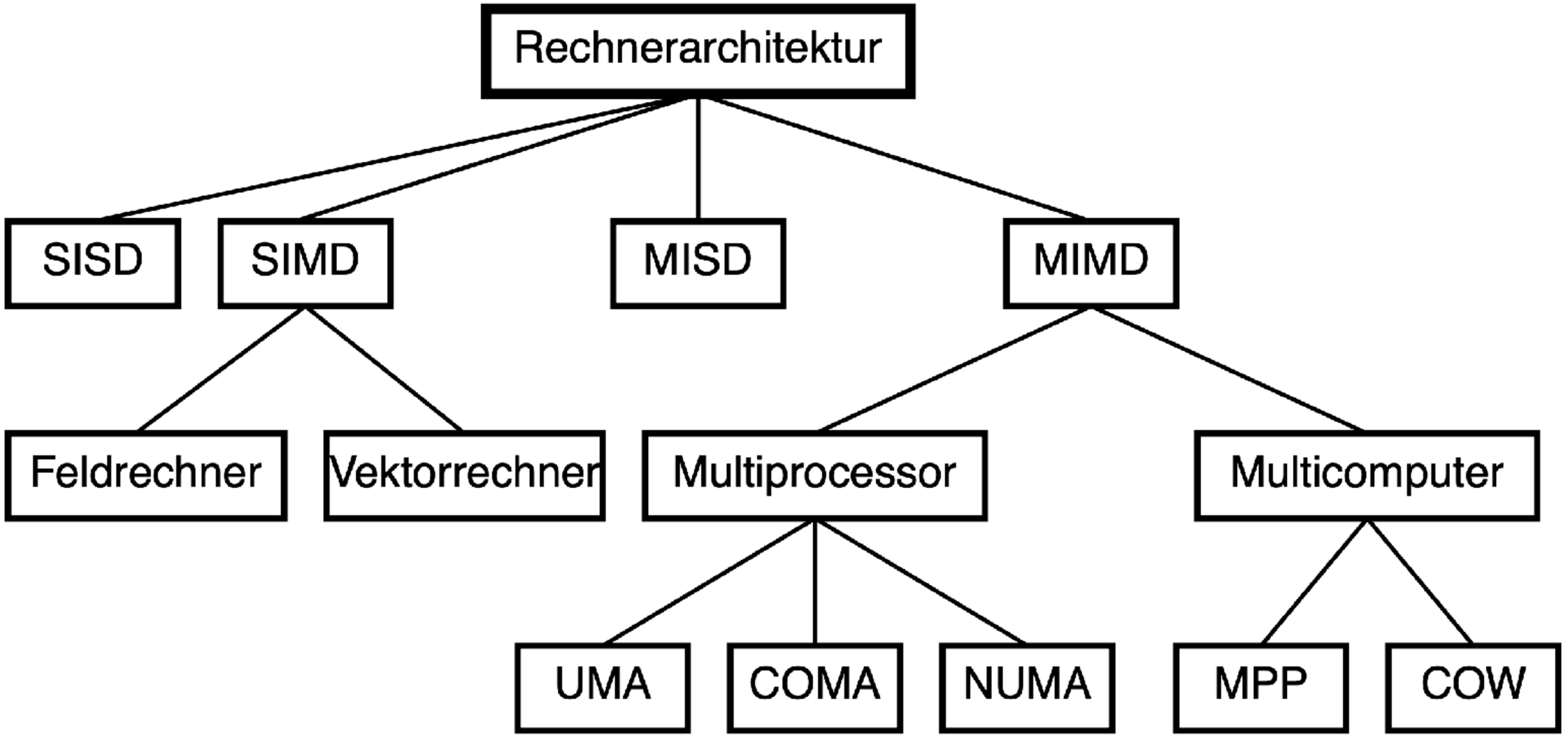
SMP: Symmetric Multiprocessing

COW: Cluster Of Workstations

Speicheranbindung bei MIMD-Systemen

UMA	Uniform Memory Access, globaler gemeinsamer Adressraum.
NUMA	Non Uniform Memory Access, verteilter gemeinsamer Adressraum.
NORMA	No Remote Memory Access, physikalisch und programmtechnisch verteilter getrennter Speicher (z.B. Cluster).
COMA	Cache-Only Memory Access.
CC-NUMA	Cache Coherent NUMA.
NCC-NUMA	Non Cache Coherent NUMA.
UCA	Uniform Communication Architecture.
NUCA	Non Uniform Communication Architecture.
DSM	Distributed Shared Memory.
VSM	Virtual Shared Memory.

Übersicht Klassifikation nach Flynn



Klassifikation von Verbindungsnetzwerken

Verbindungsnetzwerk:
statisch oder dynamisch konfigurierbar, rekonfigurierbar
Stern, Bus, Ring, Baum, Feld, Crossbar, . . .

Verbindungstopologie:
regulär, irregulär oder hierarchisch

Verbindungsaufbau:
zentral oder verteilt

Blockierungsverhalten:
blockierend, blockierungsfrei, oder rearrangierbar blockierungsfrei

Betriebsverhalten:
synchron oder asynchron

Vermittlungsart:
Leitungsvermittlung, Paketvermittlung oder Cut-Through-Routing

Klassifikation nach Giloi (1991)

Rechnerarchitektur: Element im kartesichen Produkt von Operationsprinzip und Struktur

Das Operationsprinzip definiert das funktionelle Verhalten der Architektur durch Festlegung einer Informationsstruktur und einer Kontrollstruktur. Die Informationsstruktur läßt sich als eine Menge von abstrakten Datentypen spezifizieren.

Die Struktur einer Rechnerarchitektur ist gegeben durch Art und Anzahl der Hardware-Betriebsmittel sowie die sie verbindenden Kommunikationseinrichtungen.

Operationsprinzipien:

- von Neuman Operationsprinzip,
- Operationsprinzipien der Programmparallelität,
- Operationsprinzipien der Datenparallelität.

Operationsprinzip nach Giloi

Implizite Parallelität:

- Operationsebene,
- Anweisungsebene,
- Prozeßebene,
- Programmebene.

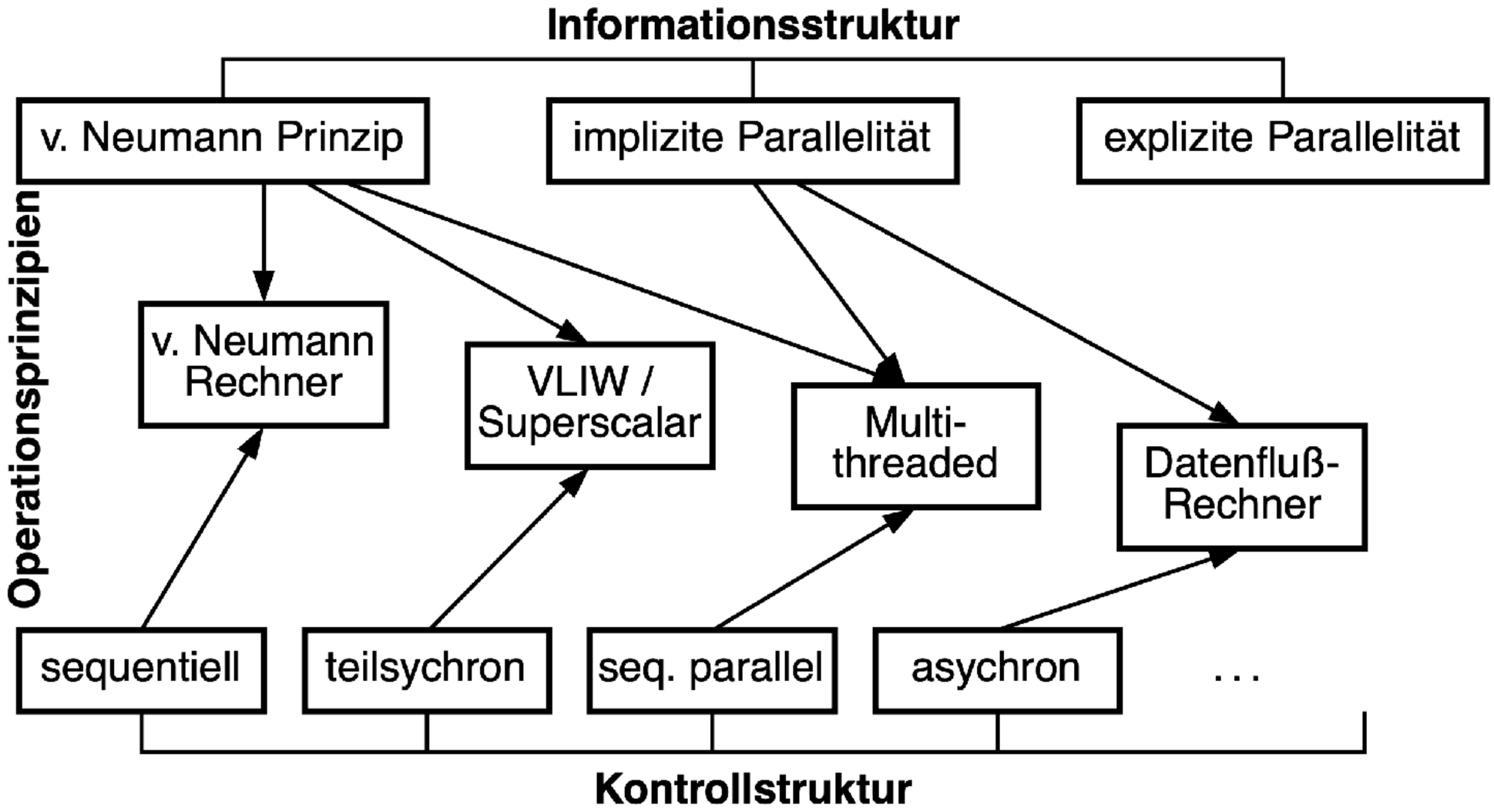
Explizite Parallelität:

- standardisierten Programmstruktur,
- standardisierten Datenstrukturen, Datenparallelität,
- selbst-beschreibenden Informationseinheiten,
- selbst-identifizierenden Daten.

Streng sequentieller Kontrollfluß des v. Neumann Rechners:

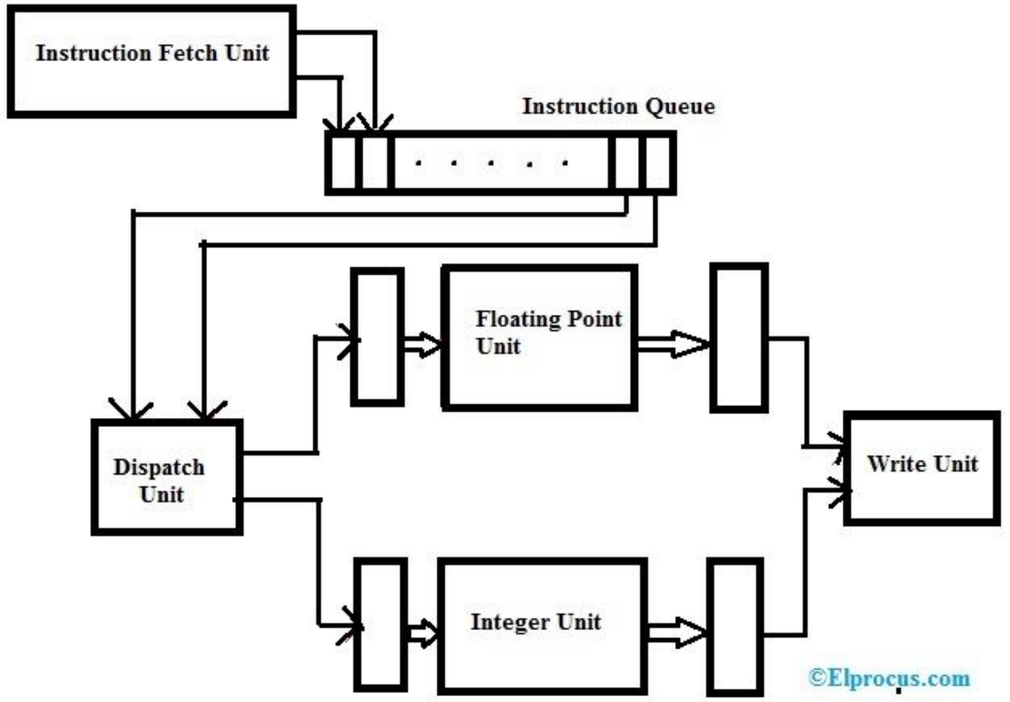
- konventioneller sequentieller Kontrollfluß für das Programm und seine Daten,
- sequentieller Kontrollfluß aber gleichzeitig parallele Ausführung mehrerer Operationen in jedem Rechenschritt,
- sequentieller Programmkontrollfluß bei assoziativem Zugriff auf die Daten,
- viele sequentielle Kontrollflüsse gleichzeitig.

Operationsprinzip nach Giloi (vereinfacht)



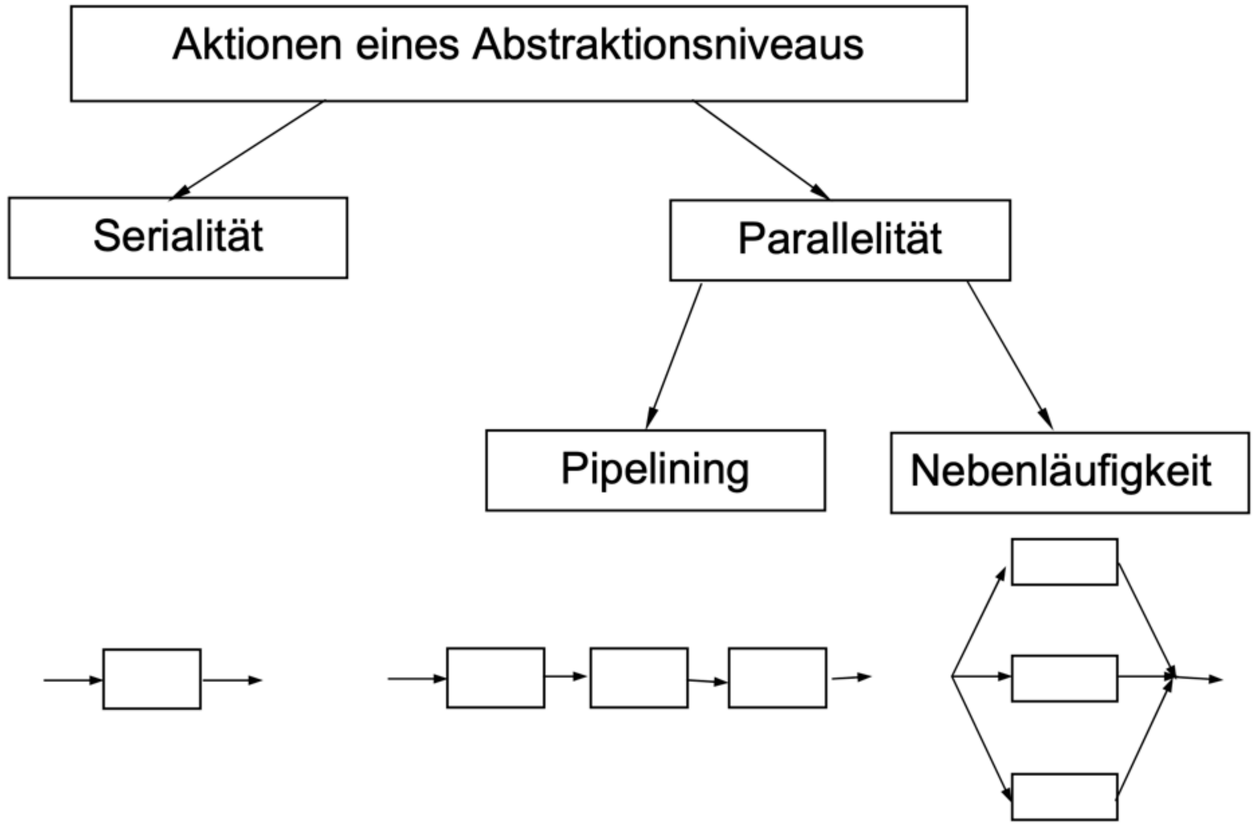
Superskalarität

- Ein **Skalarprozessor** verarbeitet eine Information pro Taktzyklus. Dies wird meist durch eine Pipeline-Architektur ermöglicht.
- **Superskalarität** ist Fähigkeit eines Prozessors, zwei oder mehr skalare Befehle eines Befehlsstroms gleichzeitig mit Hilfe von mehreren parallel arbeitenden Funktionseinheiten auszuführen.
- Eine Nebenläufigkeit bei der Ausführung einzelner Maschinenanweisungen.
- Parallel abgearbeite Befehle müssen voneinander unabhängig sein und der Prozessor muss mindestens zwei parallel betreibbare Ausführungseinheiten besitzen.
- Da (implizite) Superskalartechnik nicht den Befehlssatz der Architektur und nicht die Semantik verändert, wird auch von einer Mikroarchitektur gesprochen.



Erlangen Classification System (ECS)

- Unterscheidung nach Art und Ebenen der Parallelität: **Pipelining** und **Nebenläufigkeit**
- Ebenen der Parallelität: **Befehlsebene**, **Verarbeitungsebene** (Verarbeitungsbreite, Pipelinestufen)



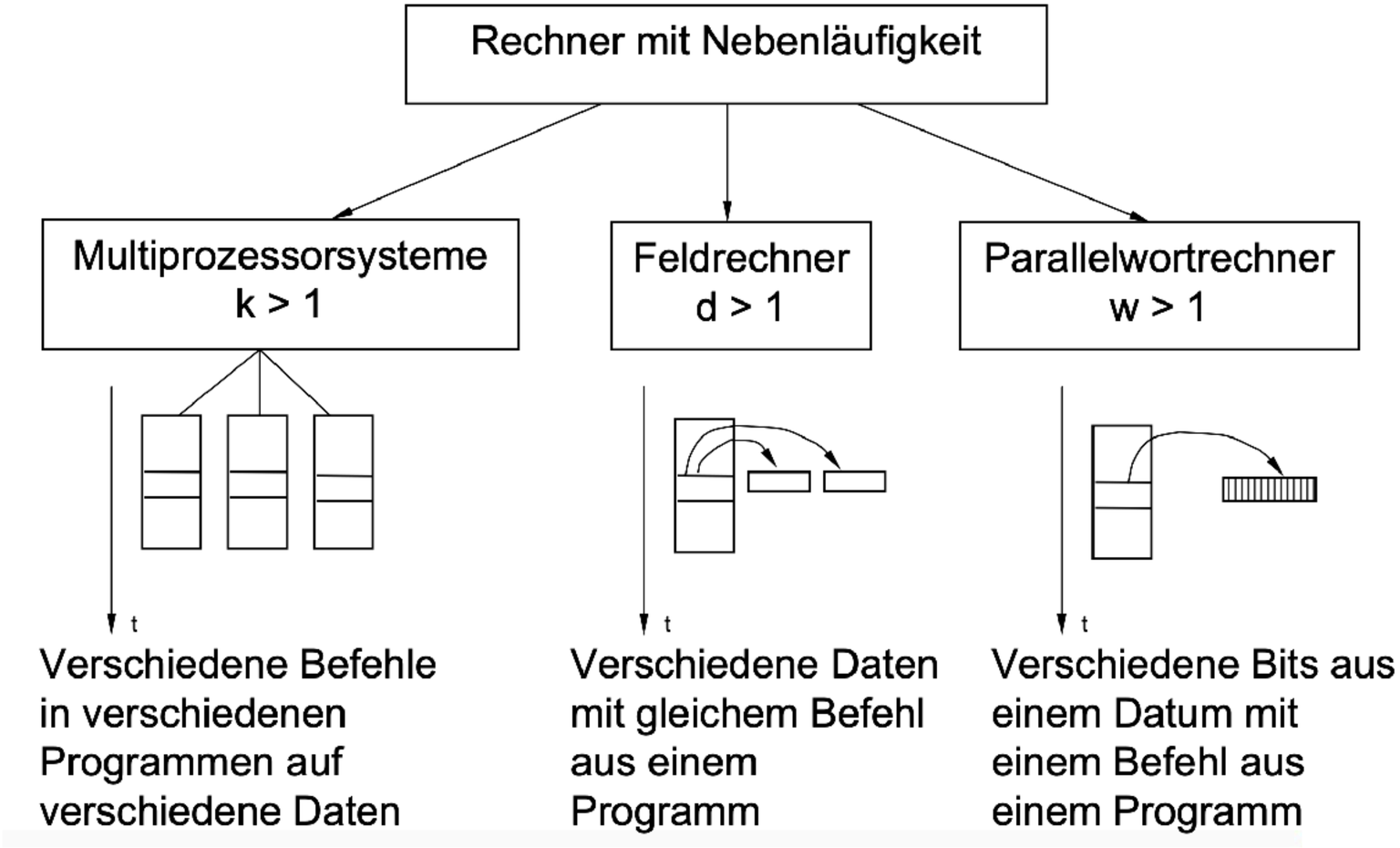
ECS Beschreibung

Die Rechnertyp-Beschreibung erfolgt nach dem ECS-Tripel aus Art der Parallelität und Ebenen der Parallelität:

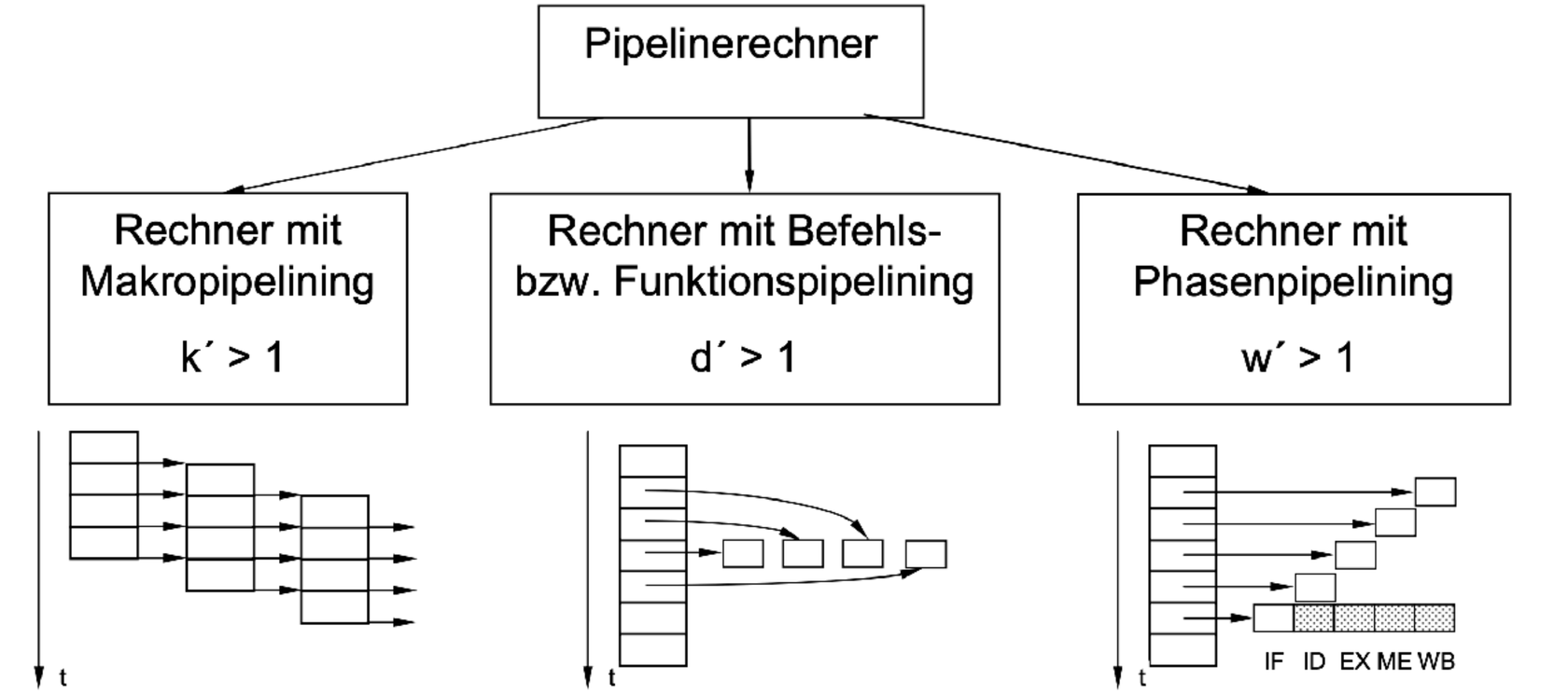
$$t_{Rechnertyp} = (k[*k´], d[*d´], w[*w´])$$

k	Anzahl der nebenläufigen Steuerwerke (Leitwerke)
k´	Anzahl der spezialisierten Steuerwerke für Programm- bzw. Prozessor-Pipelining
d	Anzahl der nebenläufigen Rechenwerke (je Steuereinheit)
d´	Anzahl der Pipelining-Rechenwerke (je Steuerwerk)
w	Anzahl der parallelen Bitstellen im Rechenwerk (Arithmetisch-Logische Einheit)
w´	Anzahl der elementaren Teilwerke

ECS Beschreibung — Nebenläufigkeit



ECS Beschreibung — Pipelining



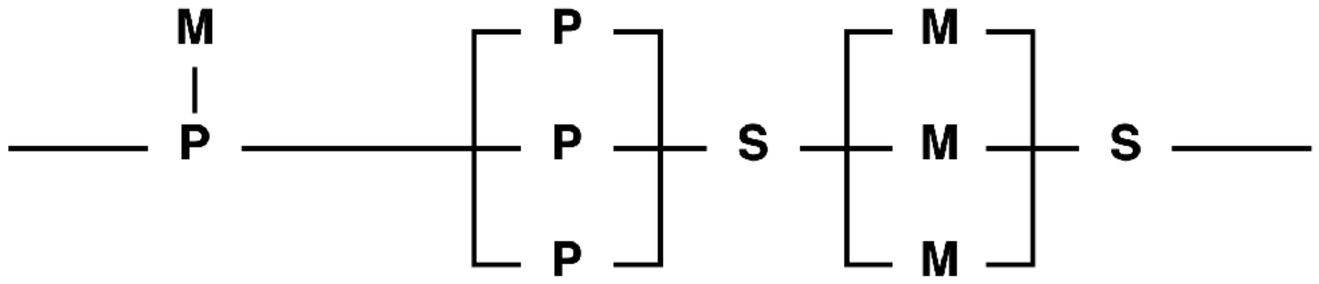
PMS Klassifikation (Processor, Memory, Switch)

Hardware-Beschreibungssprache für Rechnerarchitekturen

PMS-Symbole:

Symbol	Bezeichnung	Bemerkung
C	Computer	vollständige Rechnerbeschreibung
P	Processor	Prozessorkomponente
M	Memory	aufteilbarer adressierbarer Speicher
S	Switch	steuerbare Verbindungskomponente (Bus, Crossbar, Mux)
K	Control	aktive Komponente die andere PMS-Komponente steuert
D	Data	Komponente zur Verarbeitung von Daten (Rechenwerk)
T	Transducer	Übertragungskomponente zur Kommunikation mit der Aussenwelt

PMS-Darstellung auch als Strukturdiagramm:



Aufgaben — Architekturkonzepte in der Rechnerarchitektur I

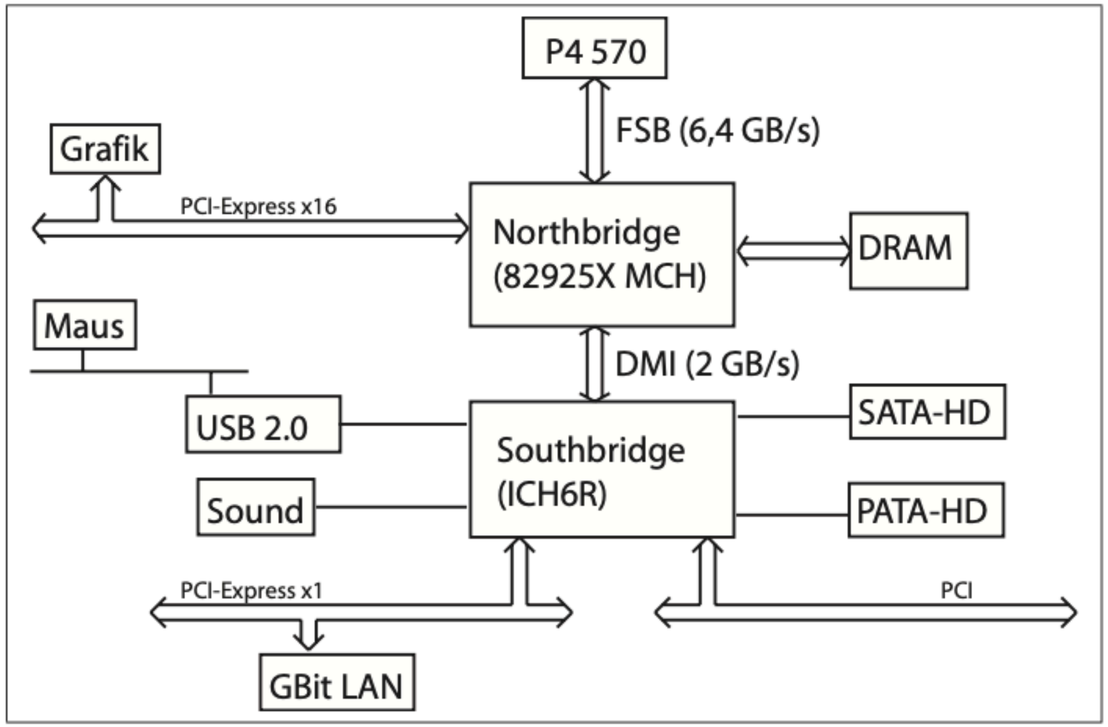
1. Was ist das Besondere an der Harvard Architektur und wie unterscheidet sie sich von der Princeton Architektur?
2. Wie unterscheiden sich Kontroll- und Datenflußarchitekturen voneinander?
3. Wie sind moderne Prozessoren in die unterschiedlichen Klassifikationen (Flynn, Giloi, Erlangen) und Architekturen (Princeton, Harvard) einzuordnen?
4. Ein Multiprozessorsystem mit 4 Prozessoren, pro Prozessor ein Rechenwerk mit 32 Bit Verarbeitungsbreite und 5-fachem Phasenpipelining ist in ECS zu beschreiben.
5. Beschreiben Sie ein (mögliches) Multiprozessorsystem mittels PMS.

Aufgaben — Architekturkonzepte in der Rechnerarchitektur II

6. Stellen Sie die Prozessorarchitektur eines Pentium-4 Prozessors in PMS dar. Die Architektur kann an folgendem vereinfachten Prinzip zusammengefasst werden:
   - Die Bus Interface Unit stellt die Verbindung von Level-2-Cache und dem externen FrontSideBus dar.
   - Eine Instruction-Fetch-Unit führt die Befehle aus dem L2-Cache der Prozessorpipeline zu.
   - Für die Operanden und Ergebnisse steht der Level-1-Cache zur Verfügung, der seine Daten ebenfalls dem L2-Cache entnimmt.
   - Zusätzlich besitzt der Prozessor einen Trace Execution Cache.

Aufgaben — Architekturkonzepte in der Rechnerarchitektur II

7. Beschreiben Sie das externe Pentium-4-System mittels der grafischen PMS-Darstellung. Zum Einsatz kommt ein Intel Pentium 4 570 (3,8 GHz) auf einem Mainboard mit Intel 82925X Chipsatz. Verwenden Sie das abgebildete Blockdiagramm als Grundlage für Ihre Beschreibung.



Von Neumann Architektur

Das Von Neumann Rechnerkonzept (Universalrechner)

- Erweitertes Modell zur Beschreibung voll programmgesteuerter Rechner (verallgemeinertes Automatenmodell)
- Realisiert nur einen einzigen Kontrollfluß (Kontrollpfad) und nur einen Datenfluß (Datenpfad) $\Rightarrow$ **Kontrollflußarchitektur**
- Beschreibung auf Register-Transfer-Ebene (Funktionsblöcke mit Vernetzung)
- Nutzung von Busstrukturen für die Datenvernetzung im Rechner
- Grundlegendes Operationsprinzip von Rechnern (ca. seit 1946)
- Grundprinzip, mit Abwandlungen, fast aller praktisch verwendeten Rechner

Das Von Neumann Rechnerkonzept stellt ein „Hardware-minimales“ Konzept mit einem Optimum an Funktionalität und Leistungsfähigkeit dar (hohe Effektivität).

Kriterien

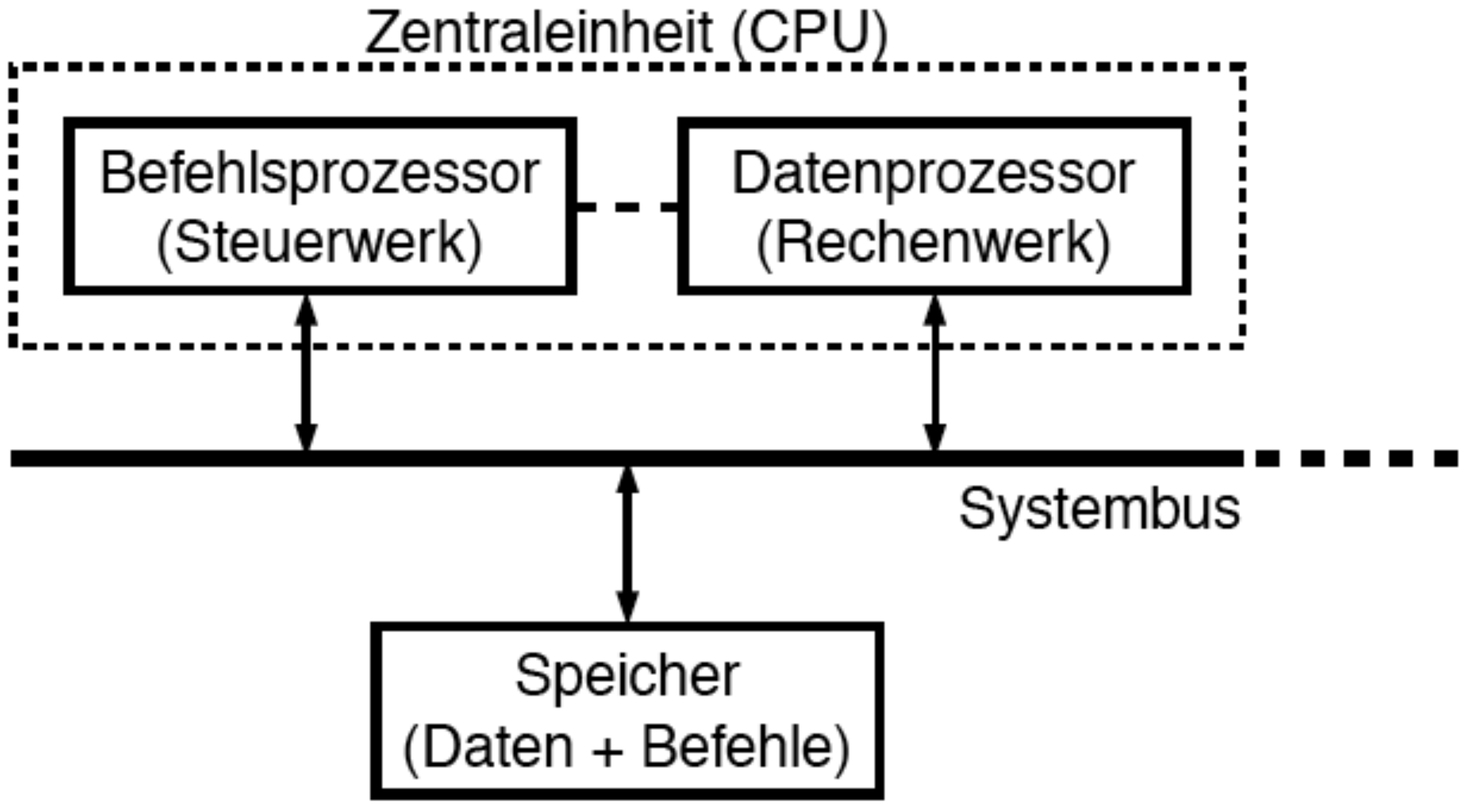
1. Hauptkomponenten des Rechners sind **Rechenwerk**, **Steuerwerk**, **Speicher, Ein-/ Ausgabeeinheit** und die Verbindungen (Datenwege, Busse). Steuerwerk und Rechenwerk bilden zusammen die zentrale Verarbeitungseinheit (Prozessor, CPU).
2. Die intern verwendete Signalmenge ist binär kodiert. Es werden Worte fester Länge parallel verarbeitet. Die Verarbeitung erfolgt taktgesteuert. Der Rechner arbeitet nach einem Start automatisch.
3. Die Architektur und Organisation des Rechners ist unabhängig von der zu bearbeitenden Aufgabe (**Universalrechner**). Die Steuerung des Rechners erfolgt über ein Programm (programmgesteuert). Die zu bearbeitende Aufgabe wird durch Programm und Daten repräsentiert.

Kriterien

4. Programmbefehle und Daten werden im einheitlichen Speicher ohne Kennzeichnung gespeichert. Der Speicher besteht aus fortlaufend adressierten Speicherworten deren Inhalt nur über die Adresse angesprochen werden kann. Entsprechend wird hiebrau auch von einer **Von Neumann Variable** (= Adresse + Inhalt) gesprochen.
5. Der Rechner verarbeitet extern eingegebene Programme und Daten. Die natürliche Verarbeitung erfolgt in der Reihenfolge in der die Befehle abgespeichert sind. Die Abarbeitung kann durch Sprungbefehle verzweigt werden.

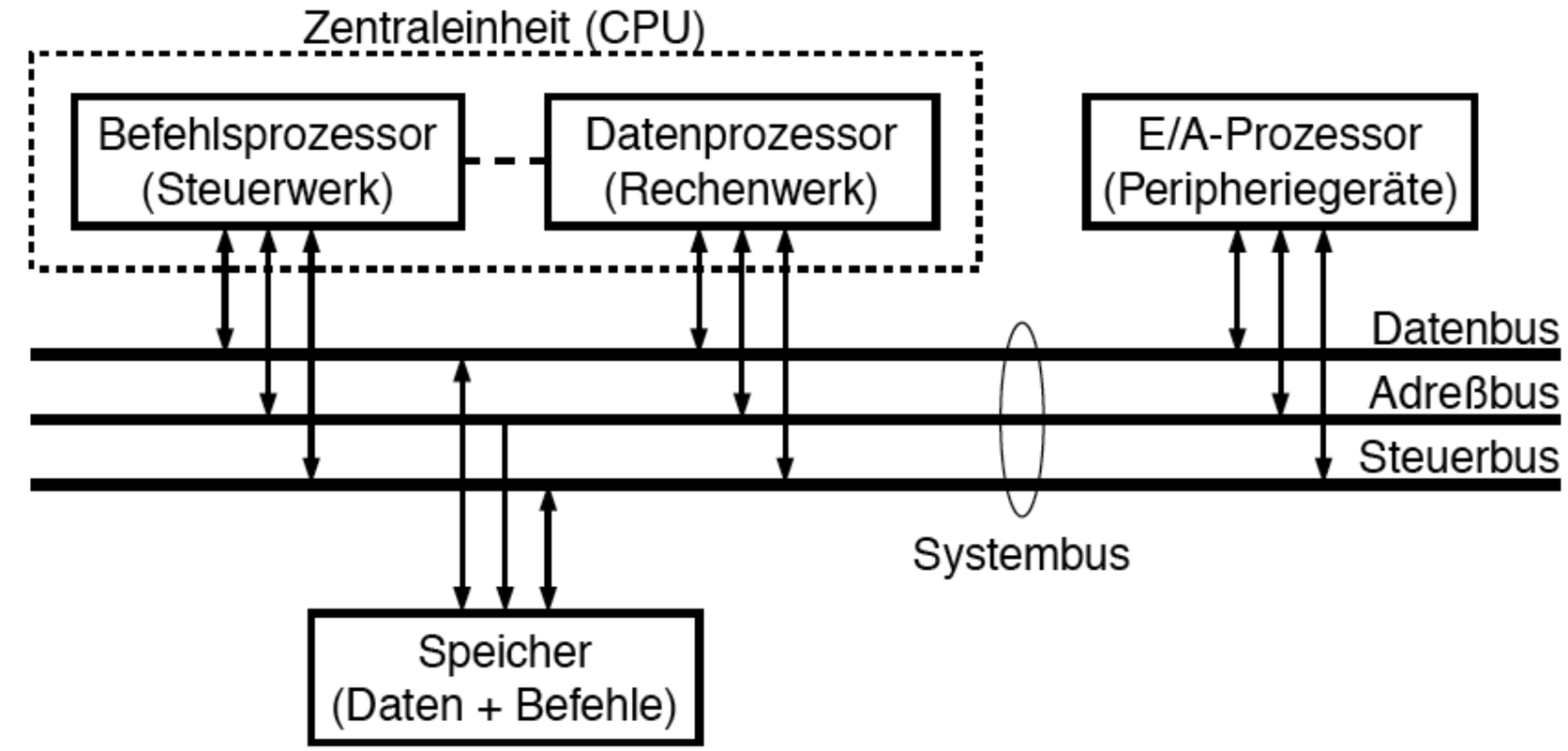
➔ Program Counter

Vereinfachte von-Neumann Architektur



Prozessorarchitektur: ohne E/A-Prozessor und Rechnerumfeld

Busaufteilung

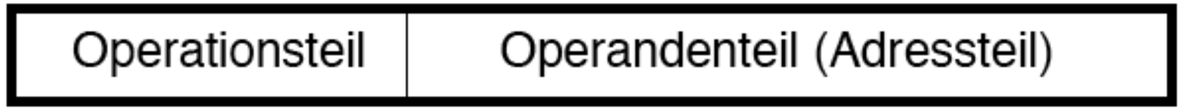


Unterteilung des Systembuses in Adress-, Daten- und Steuerbus ($\Rightarrow$ Multiplex).

Aufbau eines Befehlswortes

Einfacher Befehlsaufbau – Trennung in Operationsteil und Operandenteil

Struktur, Format des binärkodierten Befehls



kodierte Operation (Operationskode) (kurz Opcode)	Adresse des Operanden Direktoperand, Datum Verzweigungsadresse

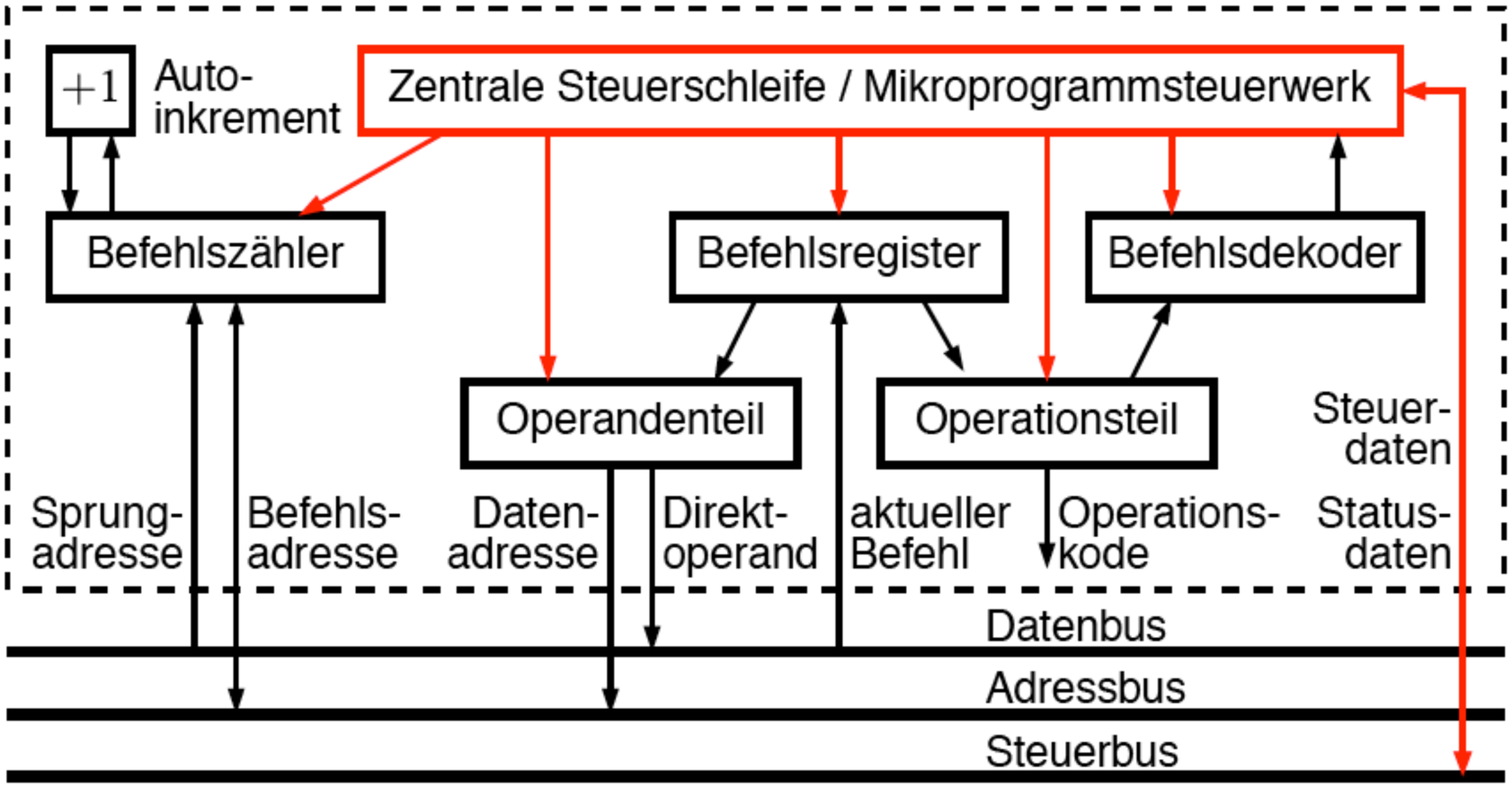
Im Operationsteil (OT) wird die durchzuführende Operation, Anweisung kodiert.
Die Operationen (1-aus-N Kode) werden im Befehl dicht binär kodiert.

Der Operandenteil, Adressteil (AT) umfasst die Opreranden bzw. deren Adressen oder auch Verzweigungsadressen, auf die die Operation angewendet wird.

Komponenten des Von Neumann Rechners

- Steuerwerk (Befehlsprozessor)
  $\Rightarrow$ **Kontrollfluß**
- Rechenwerk (Datenprozessor)
  $\Rightarrow$ **Datenfluß**
- Speicher (Daten- und Befehlsspeicher)
- Ein-/Ausgabe (E/A-Prozessor)
- Systembus (Adress-, Daten- und Steuerbus)
- Steuerwerk und Rechenwerk bilden zusammen die Zentrale Verarbeitungseinheit (CPU – Central Processing Unit)

Steuerwerk, Control Unit (Befehlsprozessor)



Komponenten des Steuerwerkes

Befehlszähler – BZ (PC – Program Counter):

Enthält die Speicheradresse des nächsten auszuführenden Befehls.
Automatische Inkrementierung (Erhöhung um 1) beim Lesen des Befehls.
Programmstart und Programmverzweigung durch Initialisierung (Voreinstellen, Laden) des Befehlszählers.

Befehlsregister – BR (IR – Instruction Register):

Enthält den aktuell abzuarbeitenden Befehl.

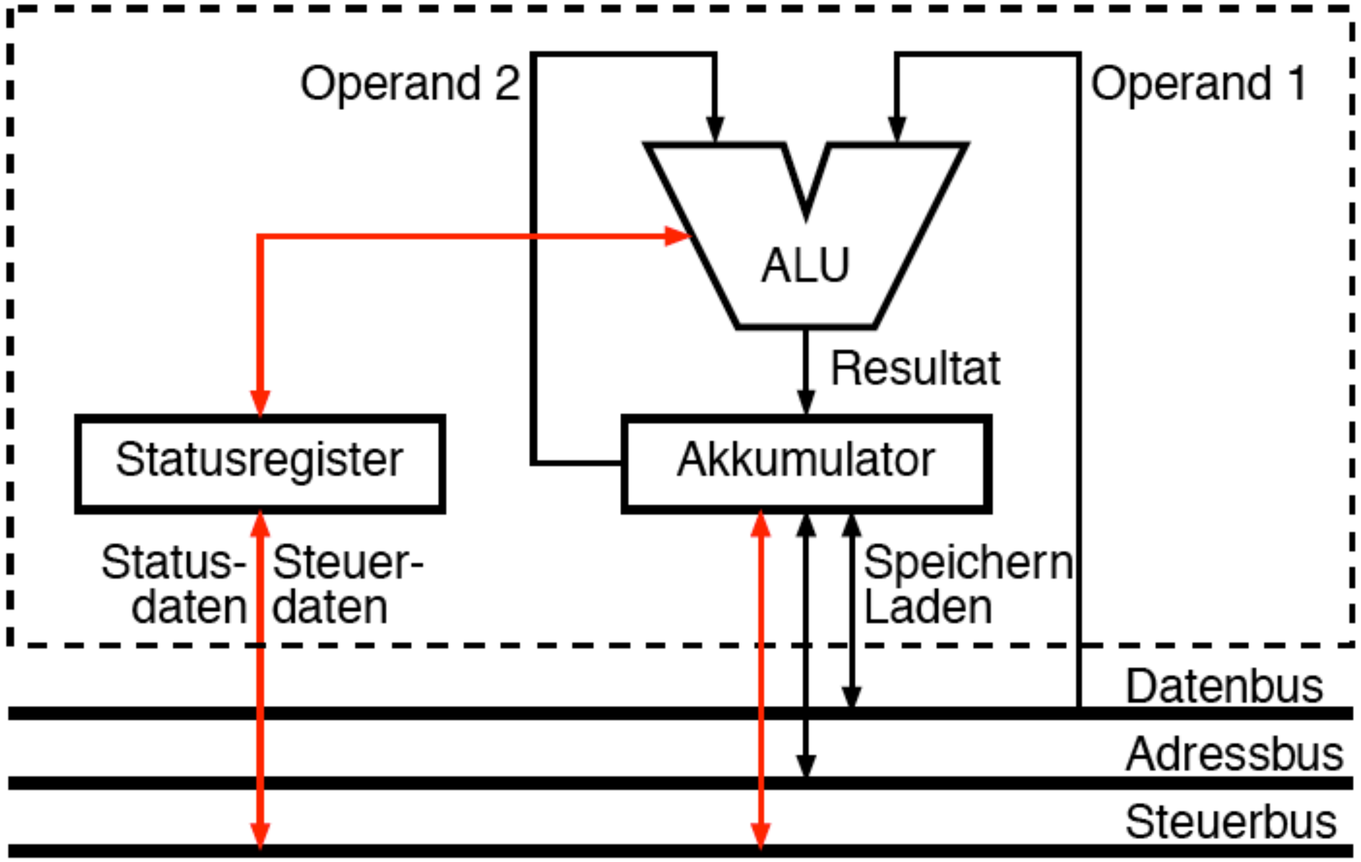
Befehlsdekoder – BD (ID – Instruction Decode):

Dekodierung des Operationsteils (Opcode) des aktuellen Befehls und Initialisierung des Zentralen Steuerschleife ($\Rightarrow$ Mikroprogrammsteuerwerk).

Zentrale Steuerschleife – ZSS (CL – Control Loop):

Realisiert den Befehlszyklus des Rechners im Steuerwerk ($\Rightarrow$ Kontrollfluß).
Koordiniert die Steuerung aller Komponenten des Rechners ($\Rightarrow$ Datenfluß).

Aufbau des Rechenwerkes (Arithmetic Logical Unit (ALU))



Komponenten des Rechenwerkes

Verarbeitungseinheit – ALU (ALU – Arithmetic Logical Unit):

Realisiert arithmetische -, logische -, Vergleichs- und Verschiebeoperationen von Operanden 1 und Operend 2 (Akkumulator) zum Resultat (Akkumulator). Daten zur Operation und zum Ergebnis werden im Statusregister gepeichert.

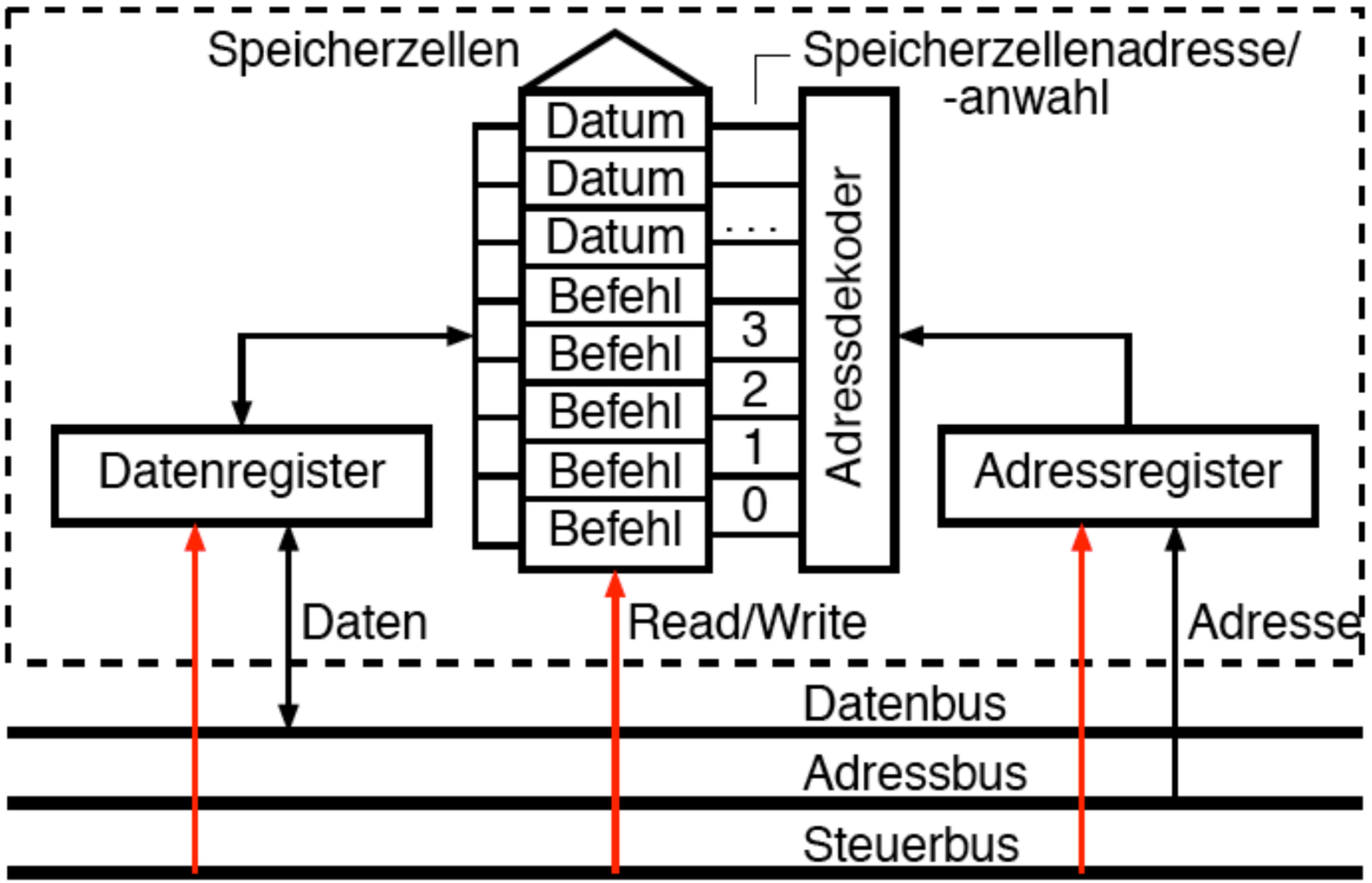
Akkumulator Register – AKKU (ACCU – Accumulator Register):

Universalregister zur Abspeicherung der Ergebnisdaten (Resultat). Gleichzeitig Hilfsregister für Operand 2 und für den Speichertransfer, zum Laden von Operand 2 bzw. zum Speichern des Resultates (Load/Store).

Statusregister – SR (SR – Status Register):

Dient der Zwischenspeicherung der Steuerdaten des Steuerwerkes und der Statusdaten des Rechenwerkes (Bedingungskode, Fehler, . . .).
Realisierung Daten-bedingter Programmverzweigungen durch Abfrage der Statusdaten, bedingte Sprünge ($\Rightarrow$ Mikroprogrammsteuerung).

Speicher, Main Memory (Hauptspeicher)



Komponenten des Speichers

Speicherzellen – RAM (RAM – Random Access Memory):

Lese-/Schreibspeicher (RWM) zur Datenspeicherung (Byte-orientiert). Speicherzellen können nur über ihre Adresse (mit 0 beginnend fortlaufend adressiert) einzeln angesprochen werden. Daten, Befehle, Adressen werden gleichermaßen, ungekennzeichnet und binär kodiert abgespeichert.

Adressdekodierer – AD (DEC – Decode):

Dekodiert die binär kodierte Adresse in einen 1-Aus-N Kode und wählt die so adressierte Speicherzelle aus.

Speicheradressregister – SAR (MAR – Memory Adress Register):

Hilfsregister zur Zwichenspeicherung der binär kodierten Speicheradresse (kann auch im Steuerwerk lokalisiert sein).

Speicherdatenregister – SDR (MDR – Memory Data Register):

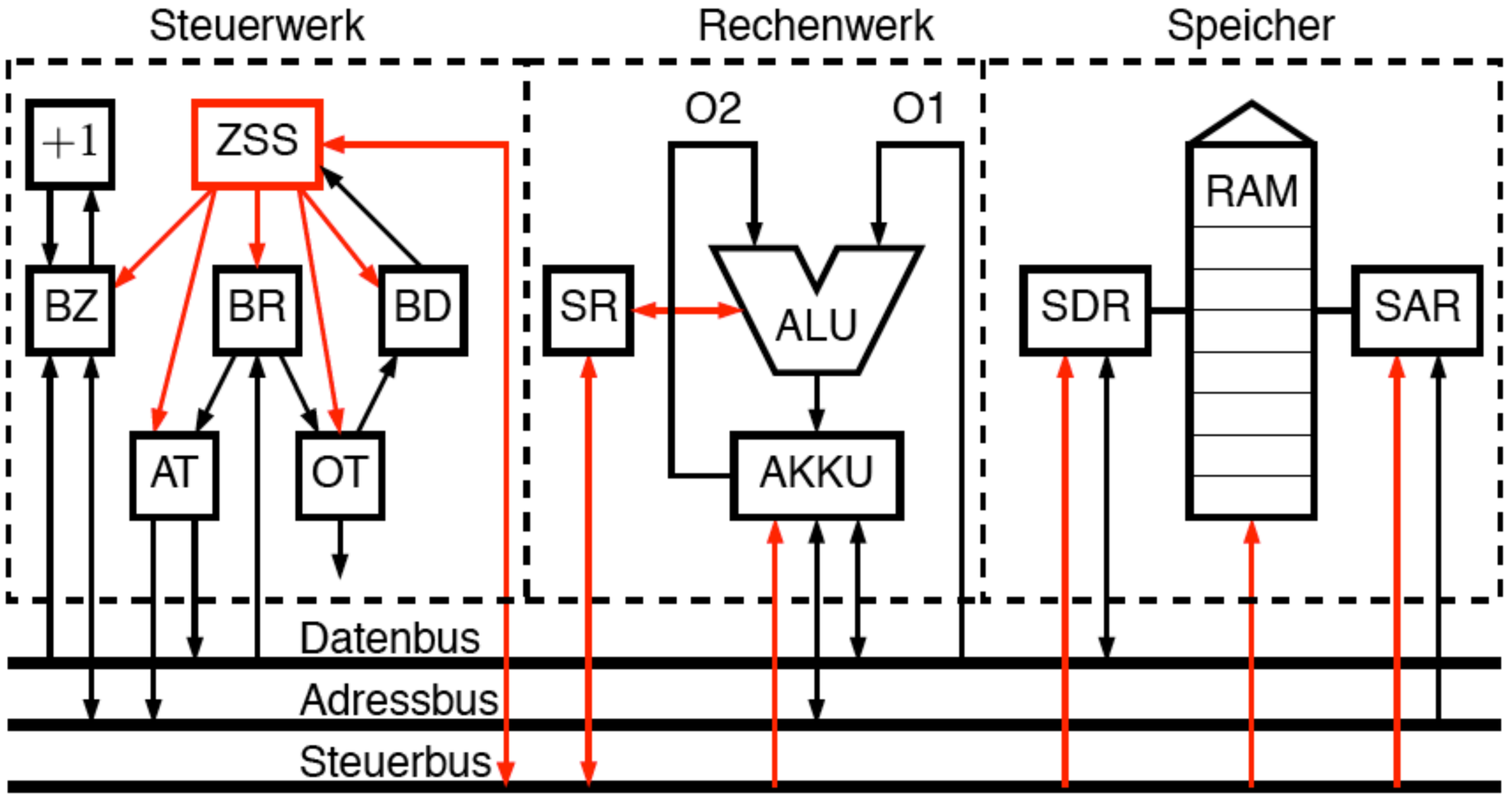
Hilfsregister zur Zwichenspeicherung der zu speichernden oder gelesenen Daten und Befehle (kann auch im Rechenwerk/Steuerwerk lokalisiert sein).

Ein-/Ausgabeprozessor (I/O-Processor)

Der Ein-/Ausgabeprozessor realisiert die Kommunikation der CPU nach außen, zu den externen Geräten und zum Nutzer:

- Realisierung spezieller Ein-/Ausgabe-Bussysteme, abgesetzte Busse zum Datentransfer (seriell, parallel, SCSI, USB, . . .).
- Steuerung der Ein-/Ausgaben von externen Geräten (Interrupt-, Abfragemethode, . . .).
- Anschluß externer Speichermedien (Archievspeicher, Austauschmedien, Backup, . . .)
- Realisierung der Netzwerkkommunikation (LAN, WAN, Firewire, . . .).
- Abfrage bzw. Steuerung von Sensoren bzw. Aktuatoren (Automatisierungstechnik, Anlagensteuerung, . . .).
- Realisierung des Benutzerinterfaces

Zusammenfassung der Komponenten



Befehlszyklus des Von Neumann Rechners (Inst. Fetch and Decode)

Die Trennung des Befehlszyklus in zwei zeitlich voneinander getrennte Phasen ist aufgrund des zweimalig notwendigen Zugriffs auf den Speicher (Befehle und Daten) zwingend erforderlich (nur ein Speicher, nur ein Bus ⇒ Engpaß).

1. Phase des Befehlszyklus – Befehl holen und dekodieren

Befehl holen (IF – Instruction Fetch):

Der Befehl wird entsprechend der Adresse aus dem Befehlszähler aus dem Speicher geladen und im Steuerwerk im Befehlsregister ablegt.
Der Befehlszähler wird automatisch inkrementiert (um 1 erhöht).

Befehl dekodieren (ID – Instruction Decode):

Der Befehlsdekoder dekodiert den Operationsteil des Befehls aus dem Befehlsregister. Der dekodierte Operationsteil wird durch das Mikroprogrammsteuerwerk in der Zentralen Steuerschleife interpretiert. Es übernimmt die weitere Steuerung der Abarbeitung .

Befehlszyklus des Von Neumann Rechners (Op. Fetch and Execute)

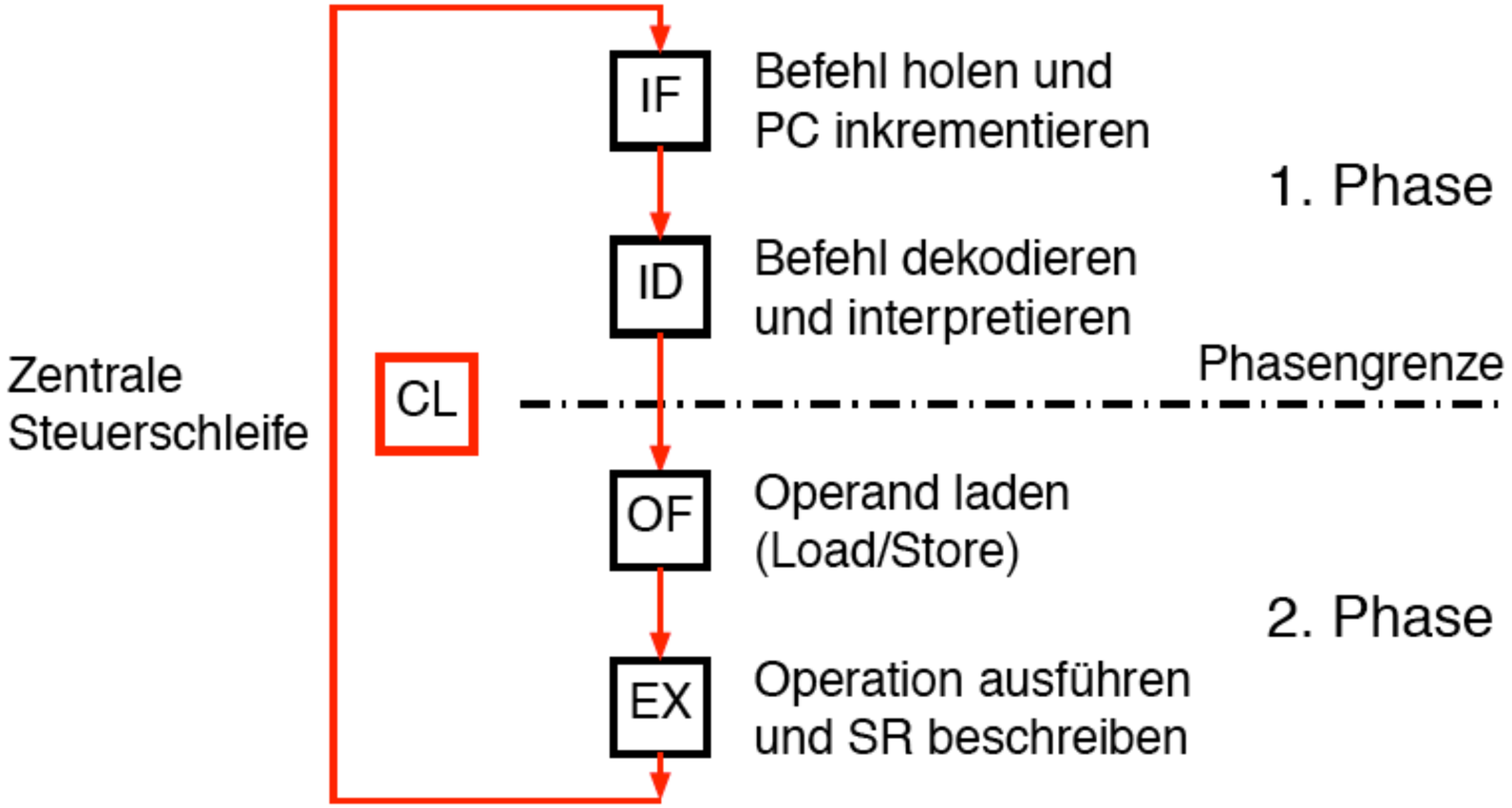
Operand holen (OF – Operand Fetch):

Der Operand wird entsprechend der Adresse aus dem Adressteil des Befehlsdekoders aus dem Speicher geladen und im Rechenwerk im Akkumulatorregister ablegt (load). Er kann auch als Direktoperand direkt vom Adressteil des Befehlsdekoders in das Rechenwerk geladen werden. Wird kein Operand aus dem Speicher geladen, so kann hier auch der Inhalt des Akkumulatorregisters in den Speicher geschrieben werden (store).

Operation ausführen (EX – Execute):

Die im Operationsteil kodierte Operation wird durch das Mikroprogramm gesteuert in der ALU oder im Akkumulatorregister ausgeführt. Die Statusinformationen der Operation werden an das Steuerwerk über das Statusregister übertragen. Der Befehlszähler des Steuerwerkes kann ebenfalls durch das Rechenwerk beschrieben und gelesen werden.

Befehlszyklus und Instruction Cycle (Zentrale Steuerschleife)



Die getrennten Phasen im Befehlszyklus

Diese beiden zeitlich getrennten Phasen der Befehlsabarbeitung wechseln sich zyklisch ab (Befehlszyklus, IC – Instruction Cycle).
Ursache ist der getrennte Speicherzugriff für Befehle und Daten.

Der Programmstart erfolgt durch Initialisierung einer Startadresse im Befehlszähler (Adresse des ersten auszuführenden Befehls).

Die automatische Programmabarbeitung (Befehl für Befehl) kann nur durch spezielle Befehle (Laden/Speichern des Befehlszählers) unterbrochen (Programmverzweigungen) bzw. abgebrochen (Programmausnahmen) werden.

Die Phasen des Befehlszyklus sind praktisch oft aufwendiger durch z.B.:

- die Möglichkeit mehrstufiger Adressberechnungen,
- das Lesen und Dekodieren unterschiedlichster Befehlsformate,
- Mehrfachzugriffe auf den Speicher (Bearbeitung größerer Datenformate),
- die Ausführung eines komplexen Mikroprogramms in der Ausführungsphase.

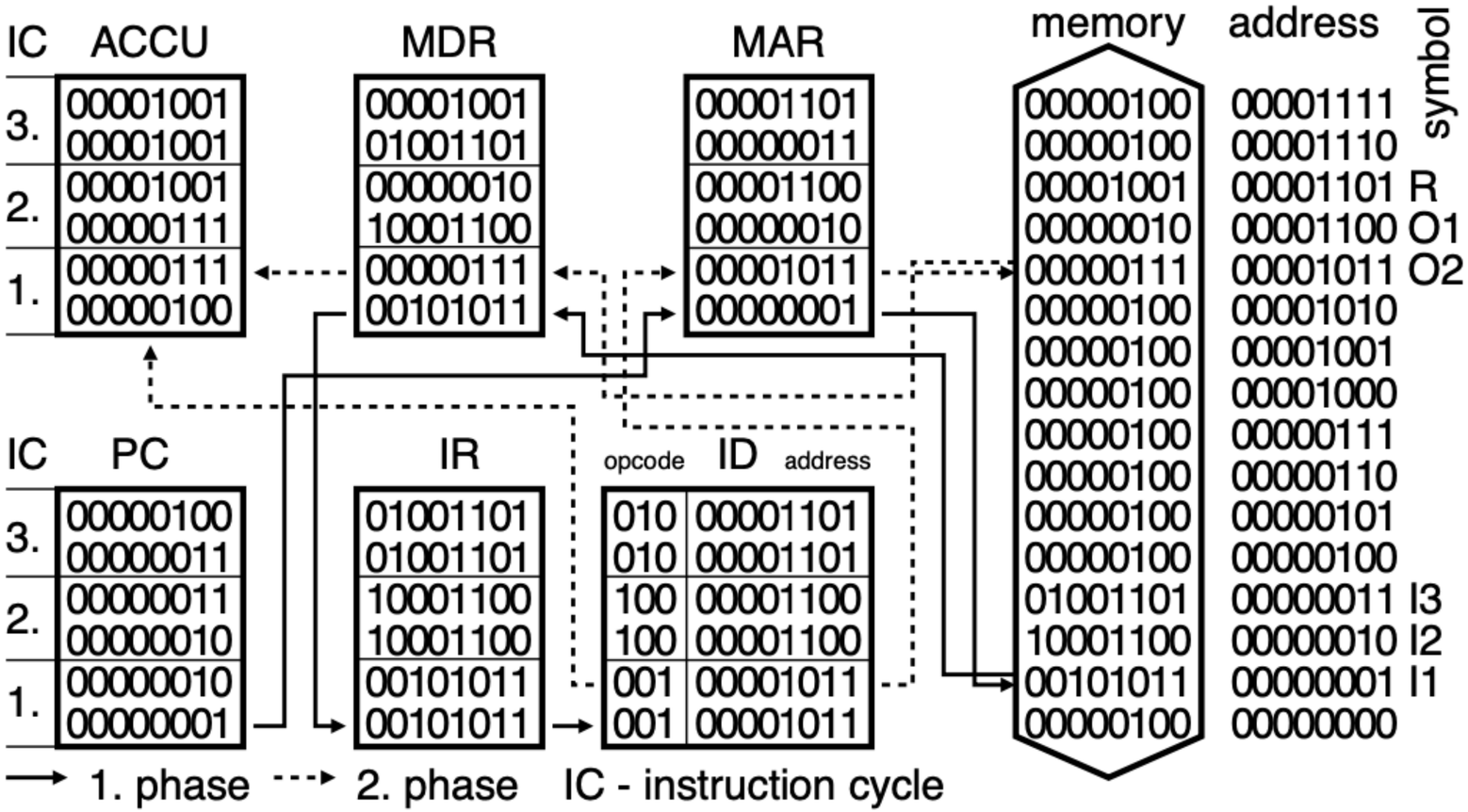
Beispielhafter Von Neumann Rechner I

Befehlsformat:	8-Bit	3-Bit Opcode, 5-Bit Adreßteil
Datenformat:	8-Bit	binärkodierte Dualzahlen, Wertebereich: 0 bis 255
Befehlsvorrat:	3-Bit	maximal 8 verschiedene Befehle kodierbar
Adressraum:	8-Bit	maximal 256 Speicherplätze adressierbar
Direktoperand:	5-Bit	Operanden direkt im Befehl, Wertebereich: 0 bis 31
Direktadresse:	5-Bit	maximal 32 Speicherplätze direkt adressierbar

Beispielprogramm zur Berechnung von: $R := O2 + O1$

N.	I-adress	instruction		opcode	D-adress	Bemerkung
I1	00000001	LOAD	O2	001	00001011	lade O2 in ACCU
I2	00000010	ADD	O1	100	00001100	addiere O1 zum ACCU
I3	00000011	STORE	R	010	00001101	speichere ACCU auf R

Beispielhafter Von Neumann Rechner II



Beispielprogramm zur Berechnung von: $R := O2 + O1$

N.	I-adress	instruction	opcode	D-adress	Bemerkung
I1	00000001	LOAD O2	001	00001011	lade O2 in ACCU
I2	00000010	ADD O1	100	00001100	addiere O1 zum ACCU
I3	00000011	STORE R	010	00001101	speichere ACCU auf R

Engpässe (Flaschenhälse) des Von Neumann Rechners I

1. **Speicherinterface:**
   Befehle und Daten stehen gleichzeitig nebeneinander und ungekennzeichnet im einheitlichen Speicher und werden über den gleichen Bus gelesen bzw. geschrieben (Speicherinterface). Befehle und Daten behindern sich somit gegenseitig ($\longrightarrow$ Trennung des Befehlszyklus in zwei Phasen).

   Ausweg: Getrennte Speicher und Busse (Speicherinterfaces) für Befehle und Daten (oder zumindest getrennte Caches) $\longrightarrow$ Harvard – Architektur.
2. **Sequentielle Abarbeitung:**
   Sequentielle Abarbeitung der einzelnen Programmschritte (Befehle) in jeweils zwei getrennten Phasen. Keine Möglichkeit der gleichzeitigen, parallelen Ausführung mehrerer Befehle oder eines Befehls mit mehreren Daten.

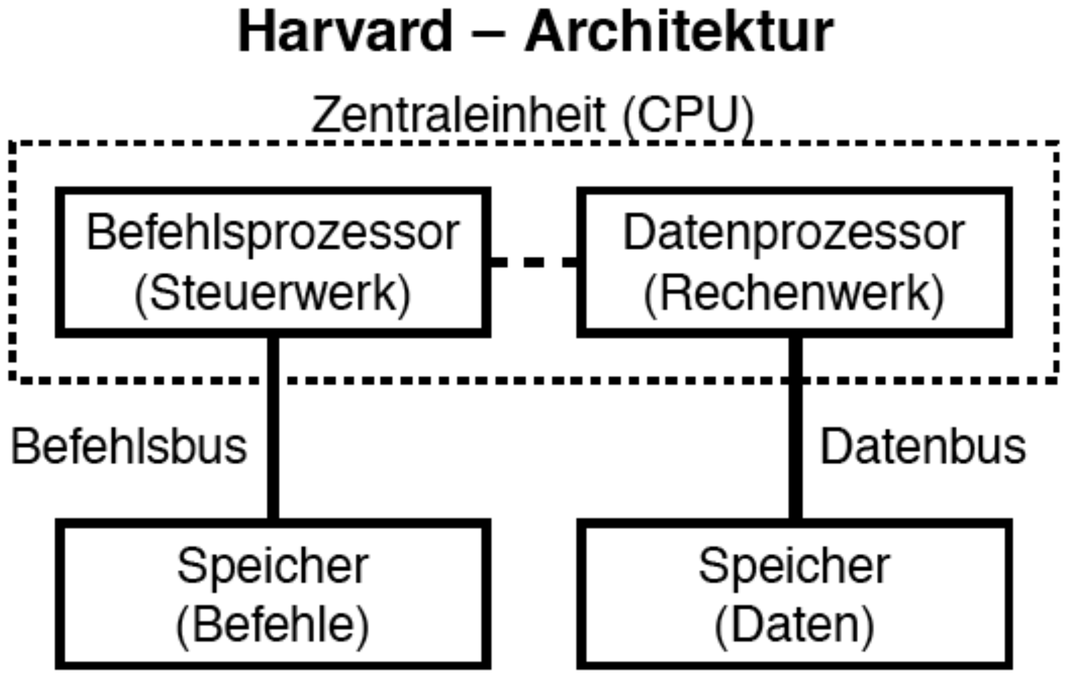
   Ausweg: Nutzung von Parallelität: erweiterte BLP (Bit Level Parallelism), ILP (Intruction Level Parallelism), MT (MultiThreaded), MP (MultiProcessor).

Engpässe (Flaschenhälse) des Von Neumann Rechners II

3. **Kontrollpfad:**
   Nur ein Kontrollfluß mit einem festen Kontrollpfad vorhanden. Feste Befehlsabfolge und Steuerung in nur einer Ebene (nur ein Befehlszähler, Befehlsregister, Akkumulator) führt zu Problemen bei der Unterprogrammtechnik, wie auch der Multitasking bzw. Multiuser Nutzung.

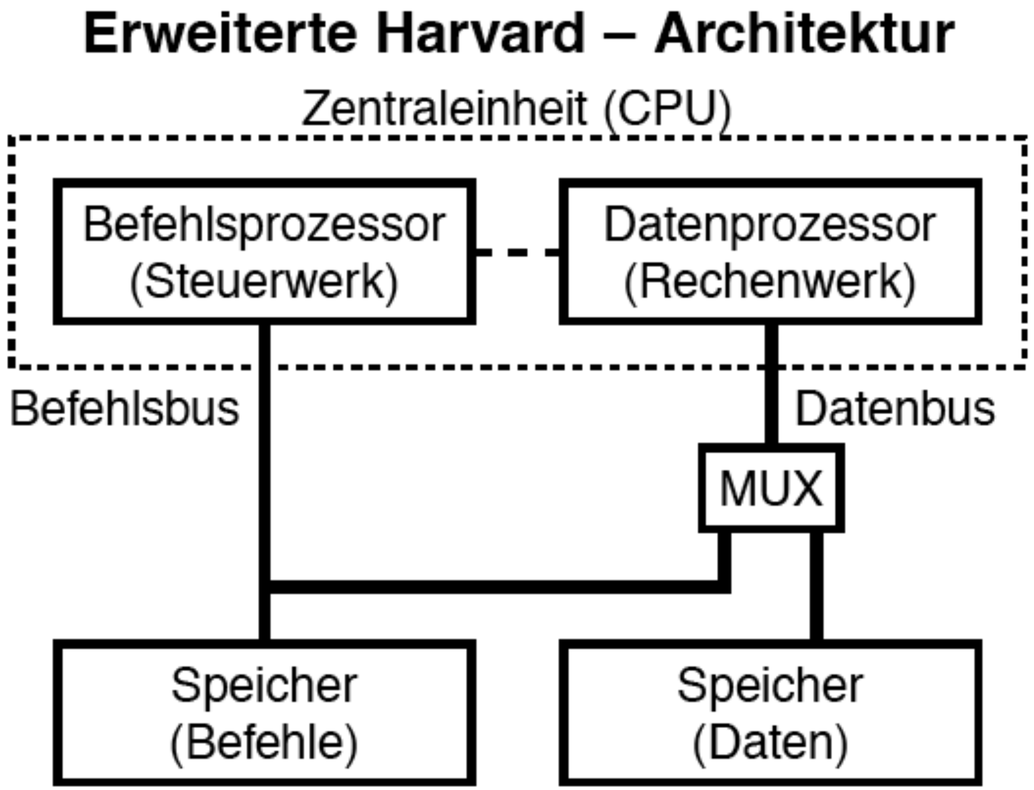
   Ausweg: Einführung eines Universalregistersatzes und vereinfachte Steuerungsabläufe für die Befehlsausführung (→ RISC – Architekturen).

Ausweg — Trennung von Befehl und Daten



Bei der Harvard-Architektur ist es möglich auf Befehle und Daten (Operanden) gleichzeitig, in einem Zyklus parallel zuzugreifen ($\Rightarrow$ getrennte Speicher und Busse für Befehle und Daten).

Kompromiss: Erweiterte Harvard-Architektur

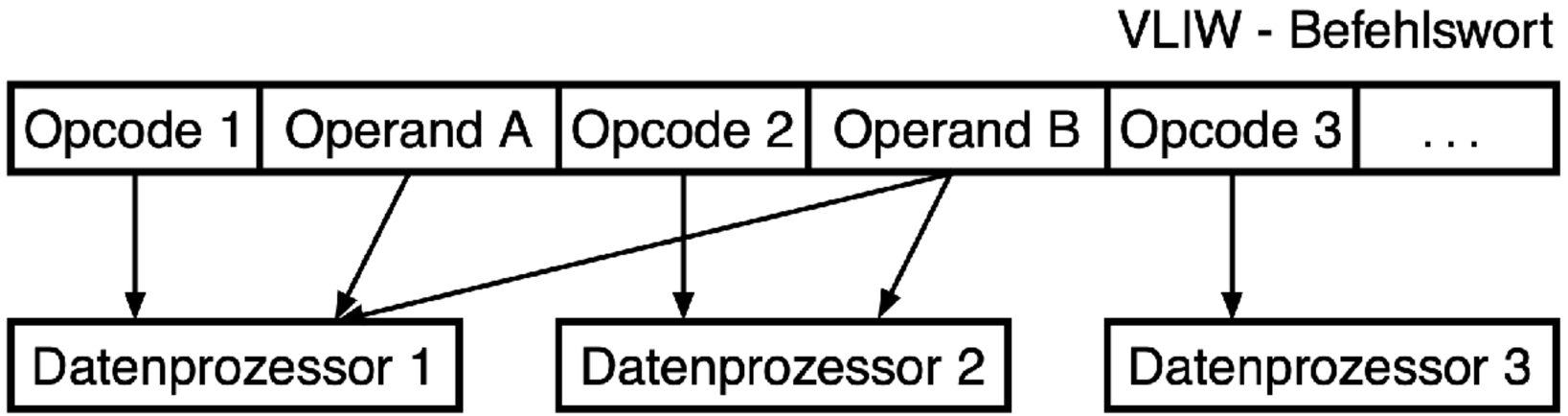


Durch den Multiplexer kann ein Operandenzugriff auch auf den Befehlsspeicher erfolgen ($\Rightarrow$ Zugriff auf beide Speicher möglich).

VLIW (Very Long Instruction Word)

Nutzung von Befehlsebenenparallelität (ILP) durch Zusammenfassen mehrerer sequentieller Befehle zu einem langen Befehlsword.

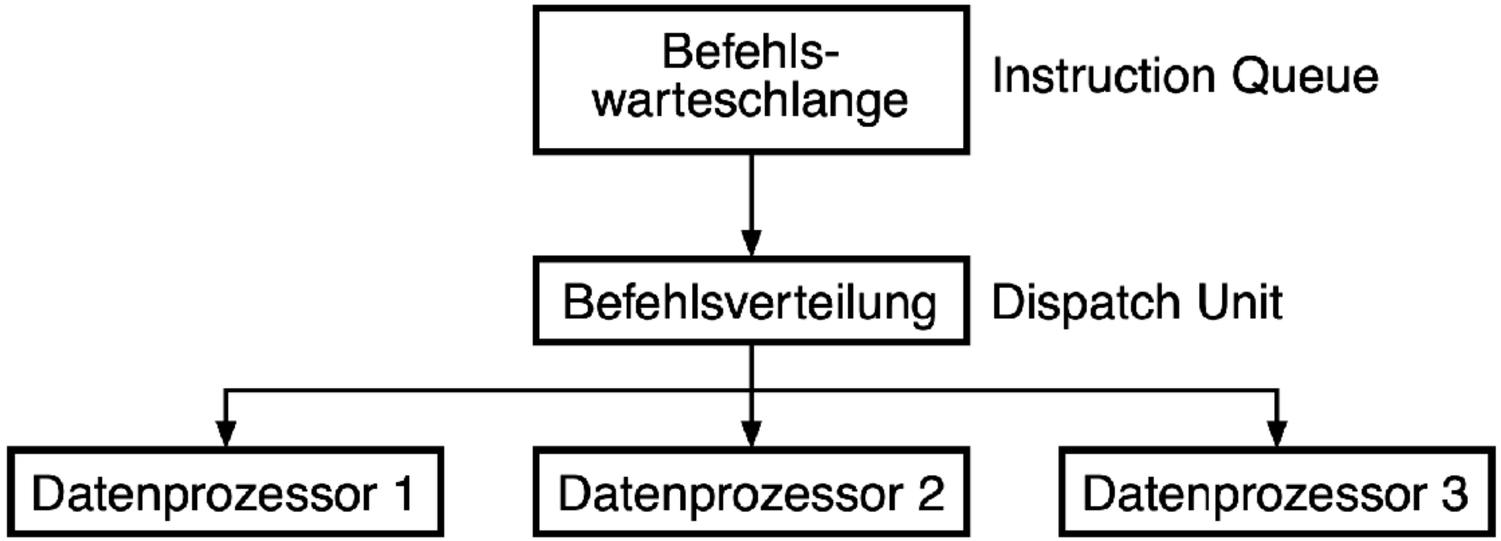
Das Befehlsword hat reservierte Bereiche (Operationsteil und Operandenteil) für jede einzelne Funktionseinheit. Die Operandenteile können sich überschneiden.



Superskalare Architektur

Nutzung von Befehlsebenenparallelität (ILP) durch parallele Ausführung von sequentiellen Befehlen auf verschiedenen parallelen Funktionseinheiten.

Die Zuordnung der Befehle zu den einzelnen Funktionseinheiten erfolgt durch eine Befehlsverteiler (Dispatch Unit). Datenabhängigkeiten sind zu beachten.



Zusammenfassung Von Neumann Rechner

- Einfache abstrakte Beschreibung eines Rechners
- Hardware-optimale, robuste, vollprogrammierbare Rechnerarchitektur
- Vollständig binärkodierte Darstellung, Speicherung und Verarbeitung
- Trennung von Kontroll- und Datenfluß, einfacher Befehlsaufbau
- Grundprinzipien eines einfachen universellen Rechners (Rechnerkonzept)

Engpässe und Einschränkungen

- Optimiertes Speicherkonzept führt zu Engpässen im Befehlszyklus
- Sequentielle Befehlsabarbeitung führt zu Engpässen bei der Abarbeitung
- Aufwändiges Steuerwerk zur Steuerung aller Komponenten
- Grundkonzept aller modernen Universalrechner (z.B. Speicherkonzept)
- Vielfältige Abwandlungen des Grundkonzeptes zur Vermeidung der Nachteile, Engpässe – Realisierung von Parallelität auf den verschiedensten Ebenen
- Vermeidung der Nachteile durch grundsätzlich andere Architekturkonzepte

Aufgaben — Konzepte des Von-Neumann Rechners

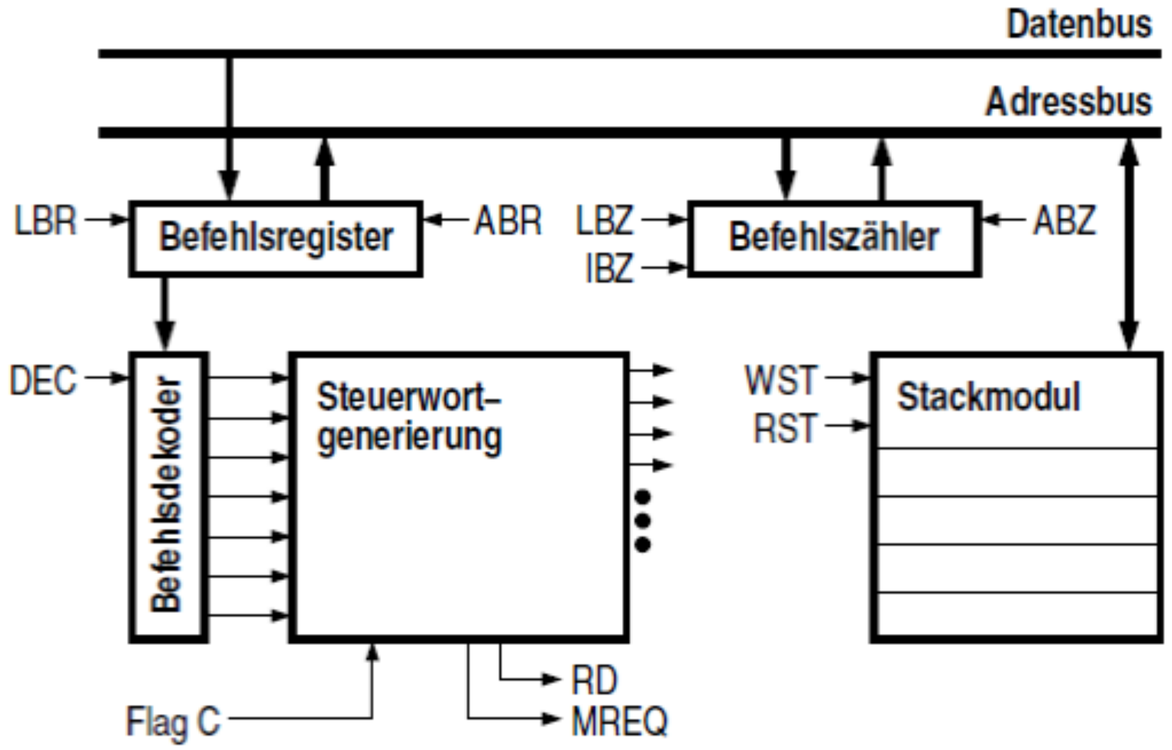
1. Aus welche Komponenten/Struktureinheiten besteht die von-Neumann Architektur? Was sind die wesentlichen Merkmale und Engpässe?
2. Ein Von-Neumann Rechner ist mittels PMS und ECS zu beschreiben.
3. Wie ist ein Befehlswort aufgebaut?
4. Wie sind die Aufgaben der Befehlsabarbeitung zwischen Steuer- und Rechenwerk verteilt?
5. Was ist die Aufgabe der zentralen Steuerschleife?

Aufgaben — Steuerwerk I

1. Zählen Sie die Grundkomponenten eines Steuerwerks auf und benennen Sie kurz den Funktion.
2. Welche drei wesentlichen Phasen durchläuft die Abarbeitung eines Befehles?

Aufgaben — Steuerwerk II

3. Gegeben ist ein Steuerwerk mit folgender Grundstruktur



Der Stapelspeicher zur Verwaltung des Aufrufstacks ist in Hardware direkt im Steuerwerk realisiert.

Aufgaben — Steuerwerk III

Folgende Steuersignale sind relevant:

`LBR`	Befehlsregister vom Datenbus laden
`ABR`	Adressteil des Befehlsregisters auf Adressbus legen
`LBZ`	Befehlszähler vom Adressbus laden
`ABZ`	Befehlszähler auf Adressbus ausgeben
`IBZ`	Befehlszähler inkrementieren
`WST`	Datum vom Adressbus auf den Stack legen
`RST`	Oberstes Stackelement vom Stack nehmen und auf Adressbus ausgeben
`DEC`	Befehl aus dem Befehlsregister dekodieren
`MREQ`	Hauptspeicher anfordern
`RD`	Hauptspeicher lesen

Der Lesezyklus des Hauptspeichers benötigt drei Takte:

1. Adresse anlegen, `MREQ` und `RD` aktivieren.
2. Wartetakt.
3. Daten auf Datenbus.

Entwerfen Sie eine Schaltung zur Ablaufsteuerung der folgenden Befehle:

- unbedingter Sprungbefehl (`JMP`),
- bedingter Sprungbefehl (`JC` für ein einziges Bedingungsflag *C*),
- unbedingter Unterprogrammaufruf (`CALL`), und
- Unterprogrammrückkkehr (`RET`)!

Aufgaben — Programmablauf

1. Gegeben sei das folgende Maschinenprogramm mit Sprüngen (`JMP`), Unterprogrammaufrufen (`CALL`), Rücksprüngen (`RETURN`) und abstrahierten Verarbeitungsbefehlen (`VBx`):

Adresse	Befehl
0x200	VB1
	VB2
	CALL @1
	VB3
	JMP @3
@1:	VB4
	CALL @4
@2:	VB5
	RETURN
@3:	VB6
	JMP @5
@4:	VB7
	JMP @2
@5:	VB8

Das Programm liegt im Speicher ab der Adresse `0x200`. Sprünge und Unterprogrammaufrufe belegen zwei Speicherzellen, alle anderen Befehle eine. Unterprogrammaufrufe hinterlegen die Rücksprungadresse auf einem Stack, von dem diese bei einem Rücksprung wieder in den Befehlszähler geladen wird.

(a) Ordnen Sie jedem Befehl seine Adresse im Speicher zu!

(b) Protokollieren Sie einen Programmdurchlauf! Geben Sie dazu jeweils den Inhalt des Befehlszählers *vor* der Befehlsausführung, die Befehlsmnemonik des ausgeführten Befehls und den Stackinhalt *nach* dessen Ausführung an!


END