Rechnerarchitektur I (RAI)

Von Neumann Architektur

Prof. Dr. Akash Kumar

Chair for Processor Design

Inhalt

- Von Neumann Rechnerkonzept
- Struktur des von Neumann Rechners
- Komponenten des von Neumann Rechners
- Befehlzyklus des von Neumann Rechners
- Beispiel zum von Neumann Rechner
- Engpässe (Flaschenhälse) des von Neumann Rechners
- Alternatuve Architekturen zum von Neumann Rechner
- Zusammenfassung von Neumann Rechner

© Akash Kumar

Von Neumann Rechnerkonzept

- Erweitertes Modell zur Beschreibung voll programmgesteuerter Rechner (verallgemeinertes Automatenmodell)
- Realisiert nur einen einzigen Kontrollfluß (Kontrollpfad) und nur einen Datenfluß (Datenpfad) ⇒ **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).

© Akash Kumar

Kriterien des von Neumann Rechners

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. 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 (Von Neumann Variable ⇒ Adresse + Inhalt).

© Akash Kumar

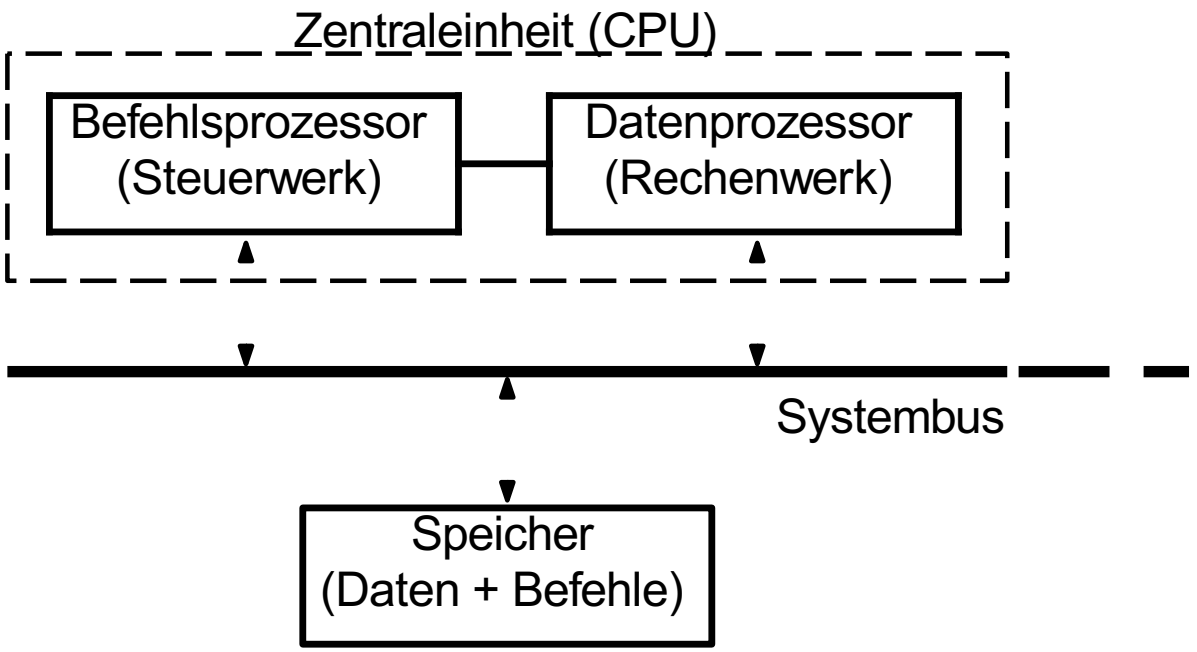
Kriterien des von Neumann Rechners

4. Der Rechner verarbeitet extern eingegebene Programme und Daten, die intern im Speicher gespeichert werden, sequentiell.
   Die natürliche Verarbeitung erfolgt in der Abspeicherungsreihenfolge der Programmbefehle.
   Die sequentielle geradlinige Abarbeitung kann durch Sprungbefehle (jump instr.) oder datenbedingte Verzweigungen unterbrochen werden.
6. Die Architektur und Organisation des Rechners ist unabhängig von der zu bearbeitenden Aufgabe (Universalrechner).
   Die Steuerung des Rechners erfolgt über ein Programm (programmgesteuert), dass an die jeweilige zu bearbeitende Aufgabe angepasst ist ($\Rightarrow$ Programmablaufplan: Program flow chart).
   Die zu bearbeitende Aufgabe wird ausschließlich durch ein Programm und die dazugehörigen Daten repräsentiert.

© Akash Kumar

Struktur des von Neumann Rechners

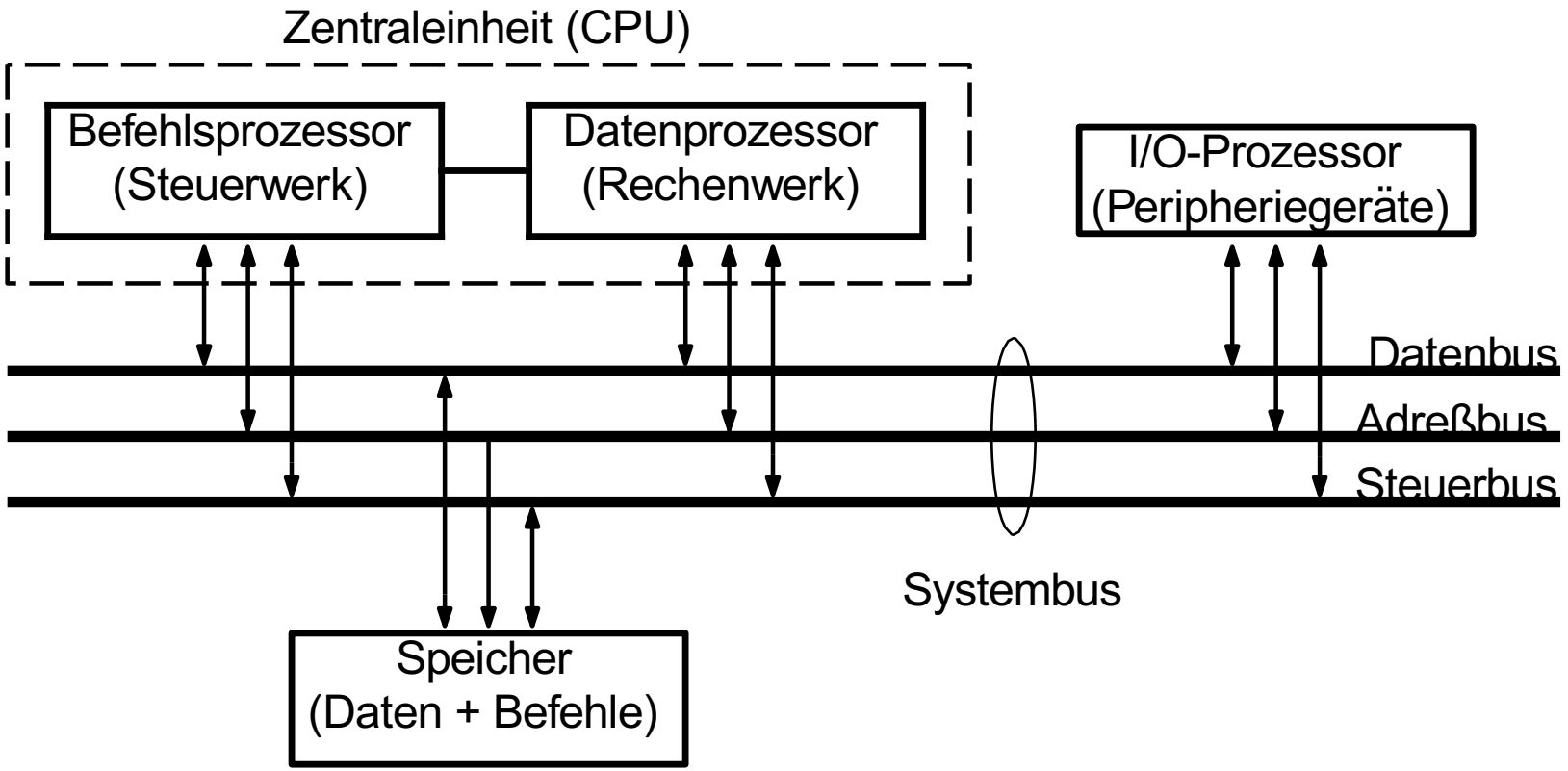
Vereinfachte Princeton Architektur (Von Neumann Architektur)



Prozessorarchitektur: ohne I/O-Prozessor und Rechnerumfeld.

© Akash Kumar

Struktur mit I/O-Prozessor und Busaufteilung



Unterteilung des Systembuses in Adress-, Daten- und Steuerbus (⇒ Multiplex).

© Akash Kumar

Befehlsformat des Von Neumann Rechners

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.

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

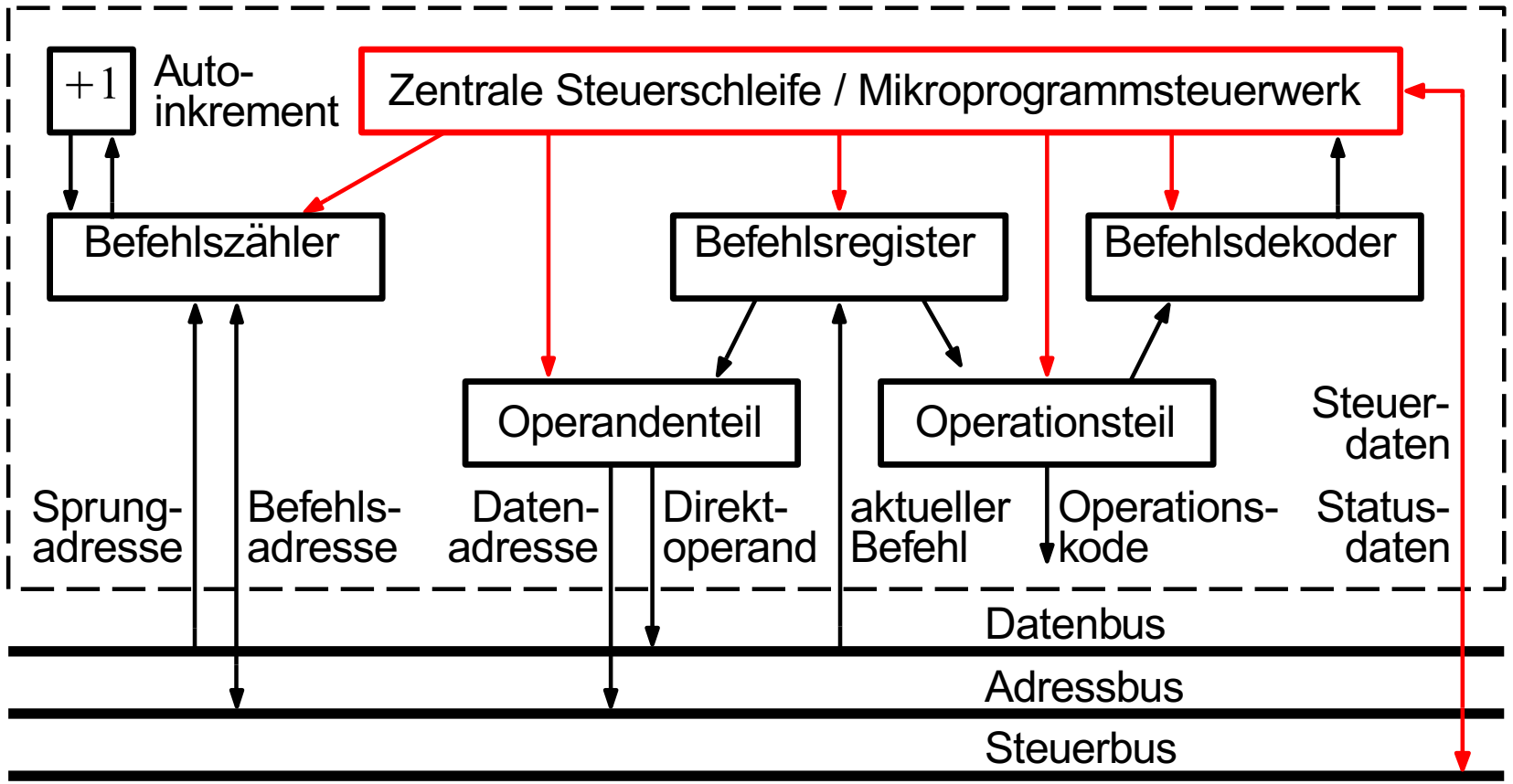
© Akash Kumar

Komponenten des von Neumann Rechners

- Steuerwerk (Befehlsprozessor)
  ⇒ **Kontrollfluß – controlflow**
- Rechenwerk (Datenprozessor)
  ⇒ **Datenfluß – dataflow**
- Speicher (Daten- und Befehlsspeicher)
- Input/Output (I/O-Prozessor)
- Systembus (Adress-, Daten- und Steuerbus)
- Steuerwerk und Rechenwerk bilden zusammen die Zentrale Verarbeitungseinheit (CPU – Central Processing Unit)

© Akash Kumar

Steuerwerk, Control Unit (Befehlsprozessor)



© Akash Kumar

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):

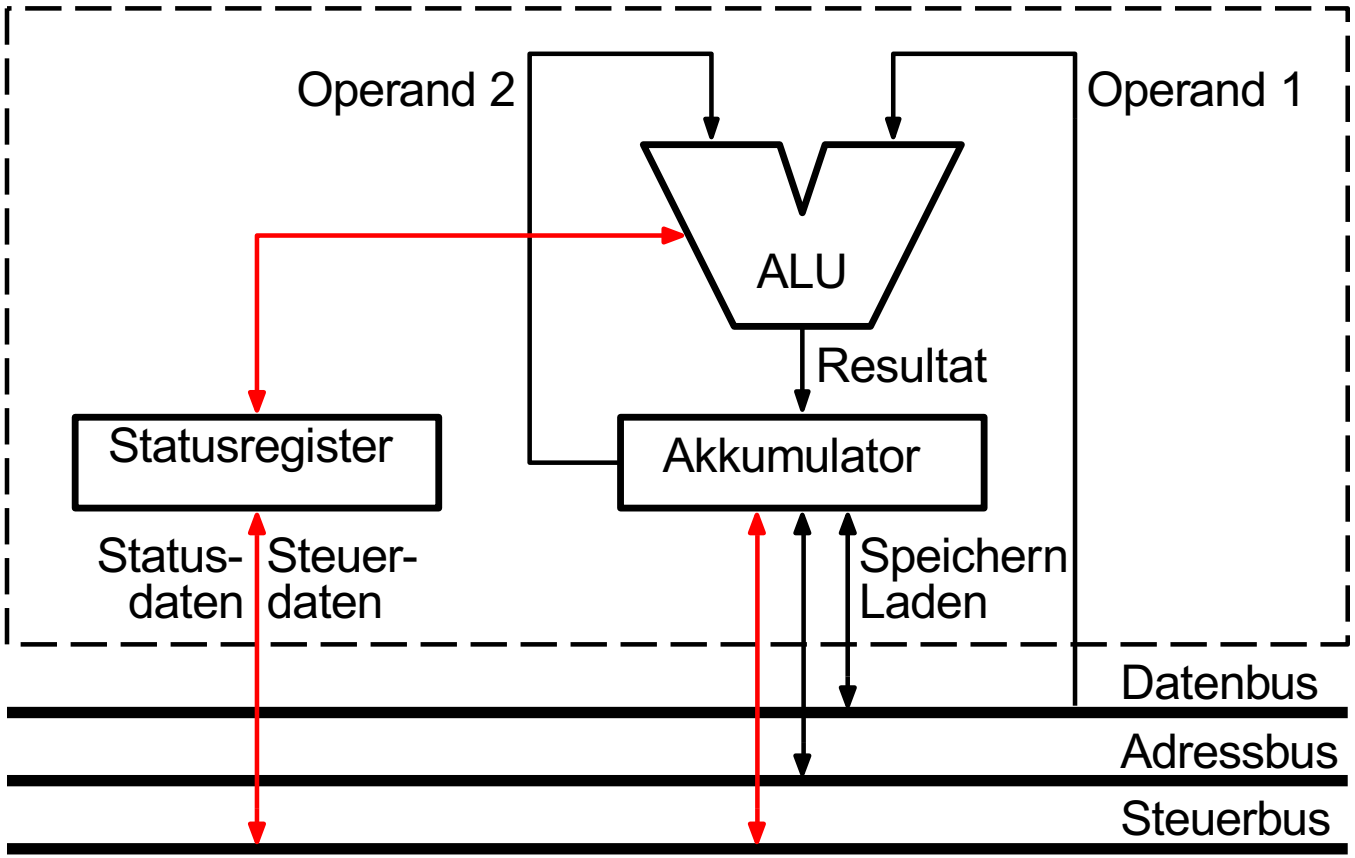
Dekodierung des Operationsteils (Opcode) des aktuellen Befehls und Initialisierung des Zentralen Steuerschleife (⇒ Mikroprogrammsteuerwerk).

Zentrale Steuerschleife – ZSS (CL – Control Loop):

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

© Akash Kumar

Rechenwerk, Arithmetic Logical Unit (Datenprozessor)



© Akash Kumar

Komponenten des Rechenwerkes

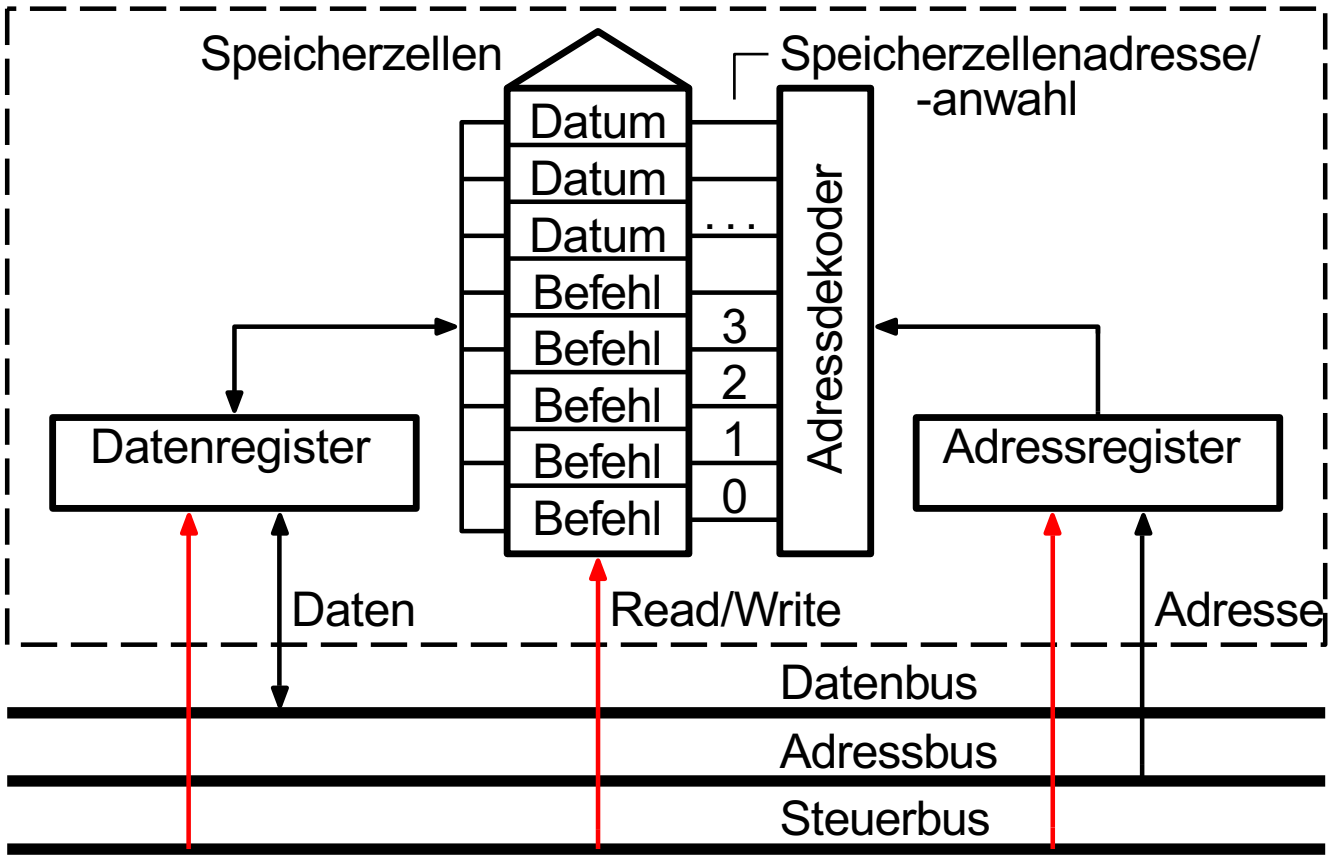
Verarbeitungseinheit – ALU (ALU – Arithmetic Logical Unit):
Realisiert arithmetische -, logische -, Vergleichs- und Verschiebeoperationen von Operanden 1 und Operend 2 (Akkumulator) zum Resultat (Akkumulator). Daten zur Operation und zum Ergebnis werden im Statusregister gepeichert.

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).

© Akash Kumar

Speicher, Main Memory (Hauptspeicher)



© Akash Kumar

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).

© Akash Kumar

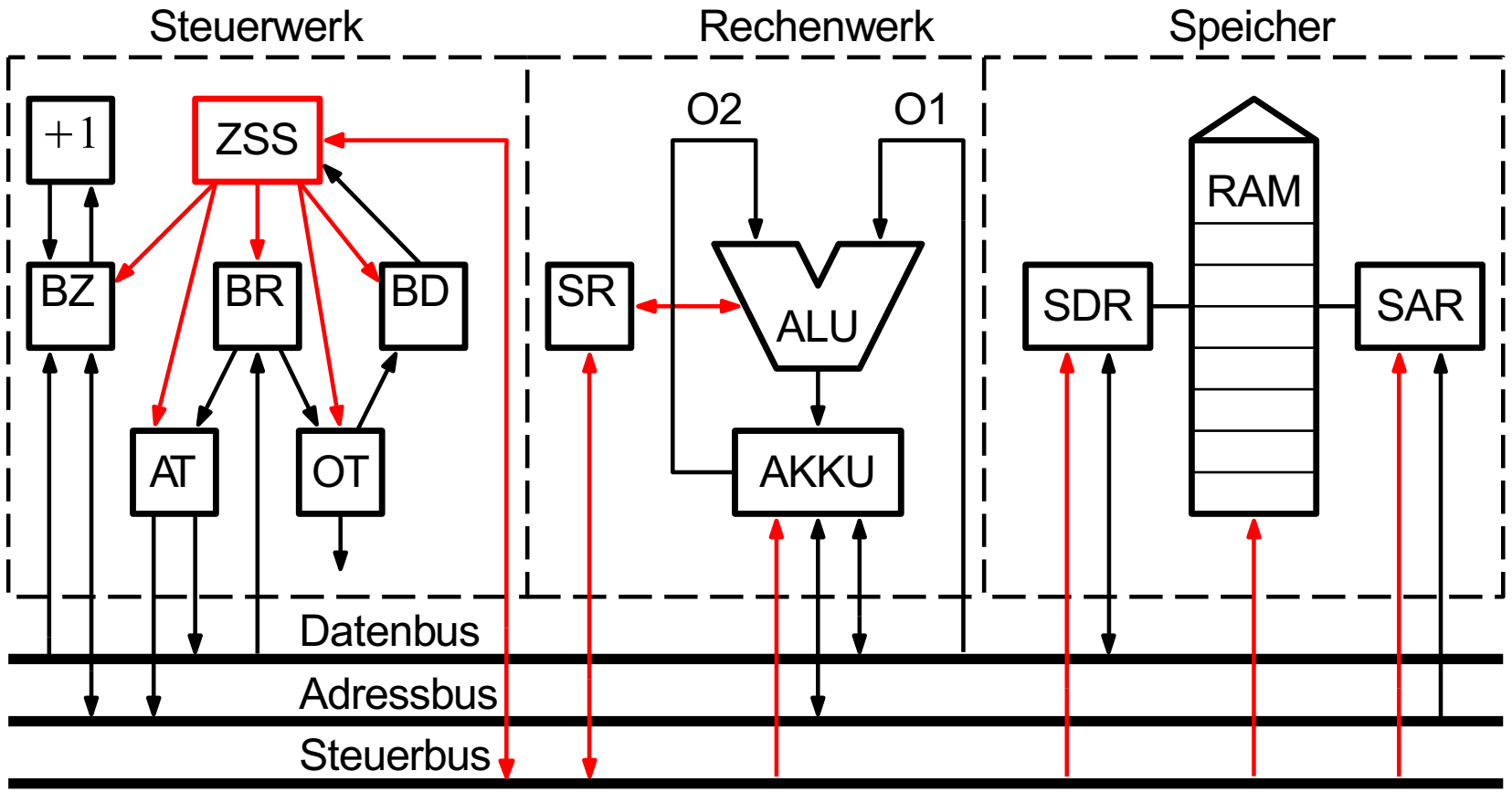
Input/Outputprozessor, I/O-Processor

Der Input/Outputprozessor 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 (Bedienersteuerung mit Keyboard, Mouse, Monitor . . .).

© Akash Kumar

Zusammenfassung der Komponenten



© Akash Kumar

Befehlszyklus des von Neumann Rechners

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.

© Akash Kumar

Phase des Befehlszyklus – Operand holen und Operation ausführen

Operand holen (OF – Operand Fetch):

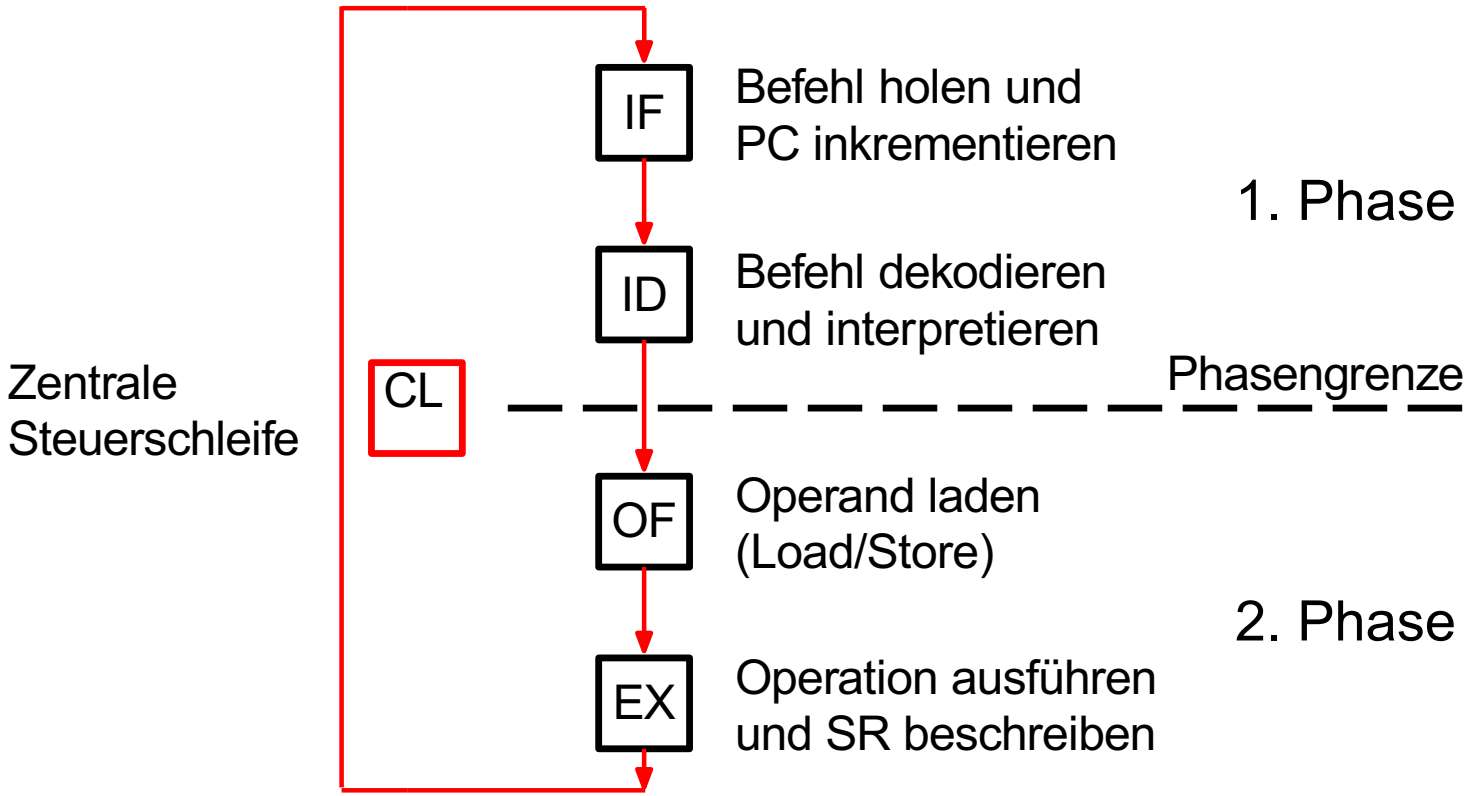
Der Operand wird entsprechend der Adresse aus dem Adressteil des Befehlsdekoders aus dem Speicher geladen und im Rechenwerk im Akkumulatorregister ablegt (load). Er kann auch als Direktoperand direkt vom Adressteil des Befehlsdekoders in das Rechenwerk geladen werden.
Wird kein Operand aus dem Speicher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

© Akash Kumar

Befehlszyklus, IC – Instruction Cycle (Zentrale Steuerschleife)



© Akash Kumar

Instruction Cycle

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 - *branch*) bzw. abgebrochen (Programmausnahmen - *exception*) 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.

© Akash Kumar

Beispiel zum von Neumann Rechner

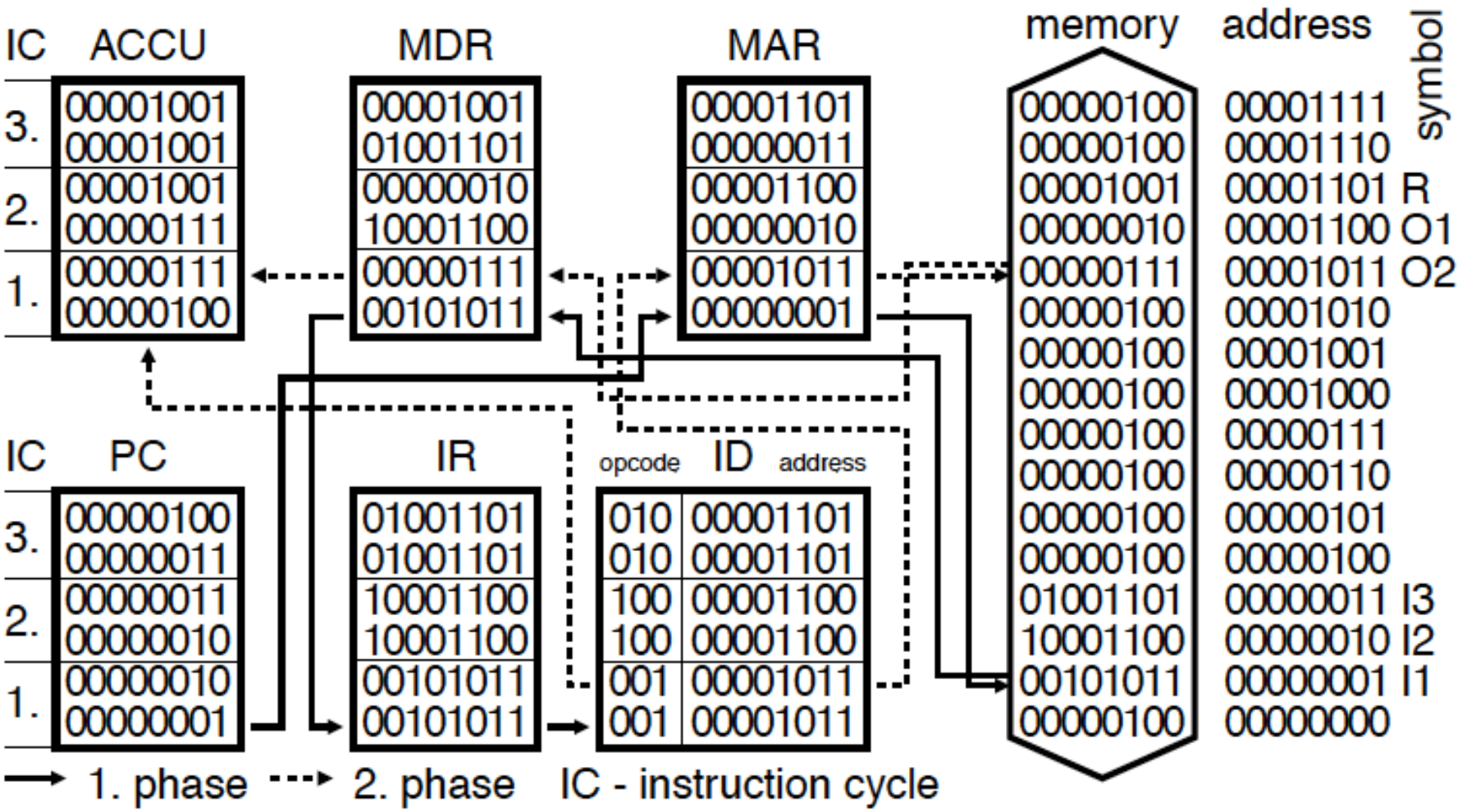
Befehlsformat:	8-Bit	3-Bit Opcode, 5-Bit Adreßteil
Datenformat:	8-Bit	binärkodierte Dualzahlen, Wertebereich: 0 bis 255
Befehlsvorrat:	3-Bit	maximal 8 verschiedene Befehle kodierbar
Adressraum:	8-Bit	maximal 256 Speicherplätze adressierbar
Direktoperand:	5-Bit	Operanden direkt im Befehl, Wertebereich: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

© Akash Kumar

Instruction Flow



© Akash Kumar

Engpässe (bottleneck) des von Neumann Rechners

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 (→ Trennung des Befehlszyklus in zwei Phasen).

   Ausweg: Getrennte Speicher und Busse (Speicherinterfaces) für Befehle und Daten (oder zumindest getrennte Caches) → 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).

© Akash Kumar

Engpässe (bottleneck) des von Neumann Rechners

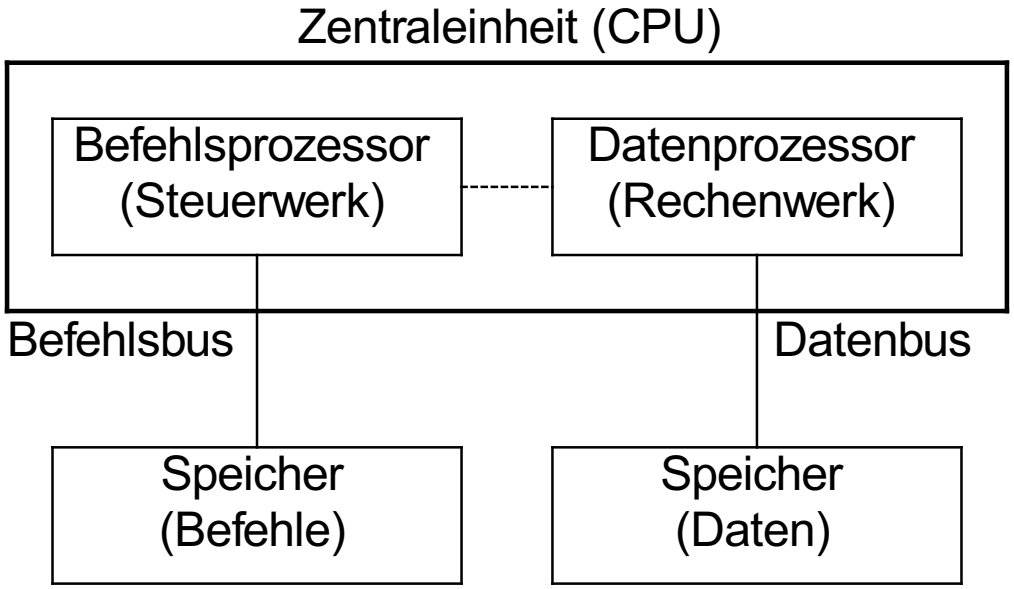
3. **Kontrollpfad:**
   Nur ein Kontrollfluß mit einem festen Kontrollpfad vorhanden. Feste Befehlsabfolge und Steuerung in nur einer Ebene (nur ein Befehlszähler, Befehlsregister, Akkumulator) führt zu Problemen bei der Unterprogrammtechnik, wie auch der Multitasking bzw. Multiuser Nutzung.

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

© Akash Kumar

Alternative Architekturen zum von Neumann Rechner

Harvard – Architektur

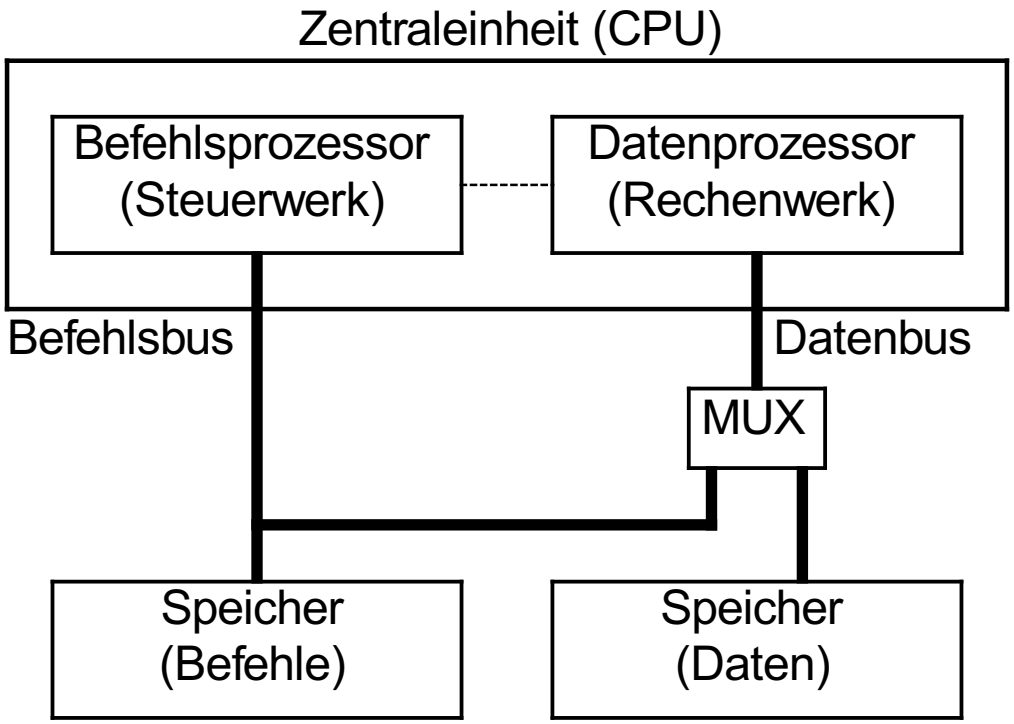


Bei der Harvard-Architektur ist es möglich auf Befehle und Daten (Operanden) gleichzeitig, in einem Zyklus parallel zuzugreifen (⇒ getrennte Speicher und Busse für Befehle und Daten).

© Akash Kumar

Alternative Architekturen zum von Neumann Rechner

Erweiterte Harvard – Architektur



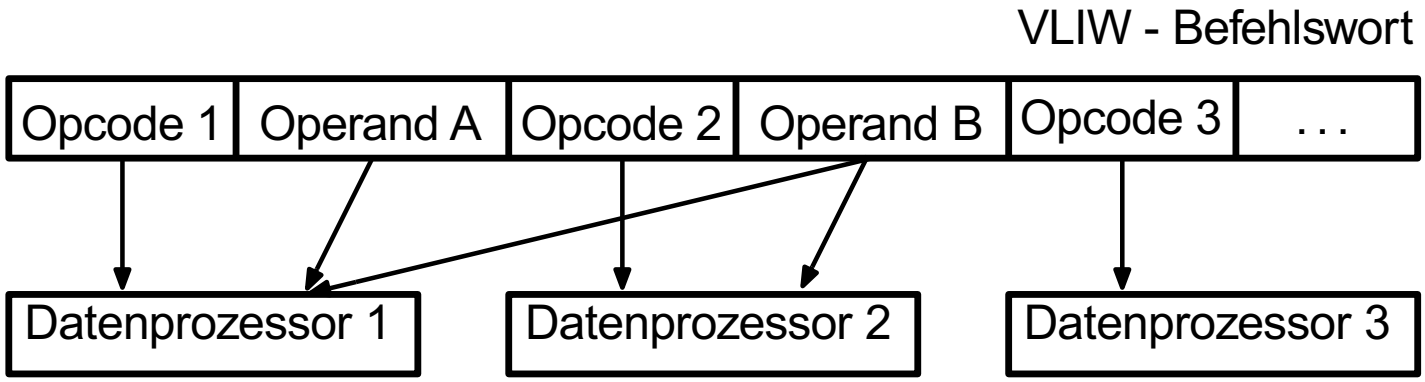
Durch den Multiplexer kann ein Operandenzugriff auch auf den Befehlsspeicher erfolgen (⇒ Zugriff auf beide Speicher möglich).

© Akash Kumar

VLIW – Architektur (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.

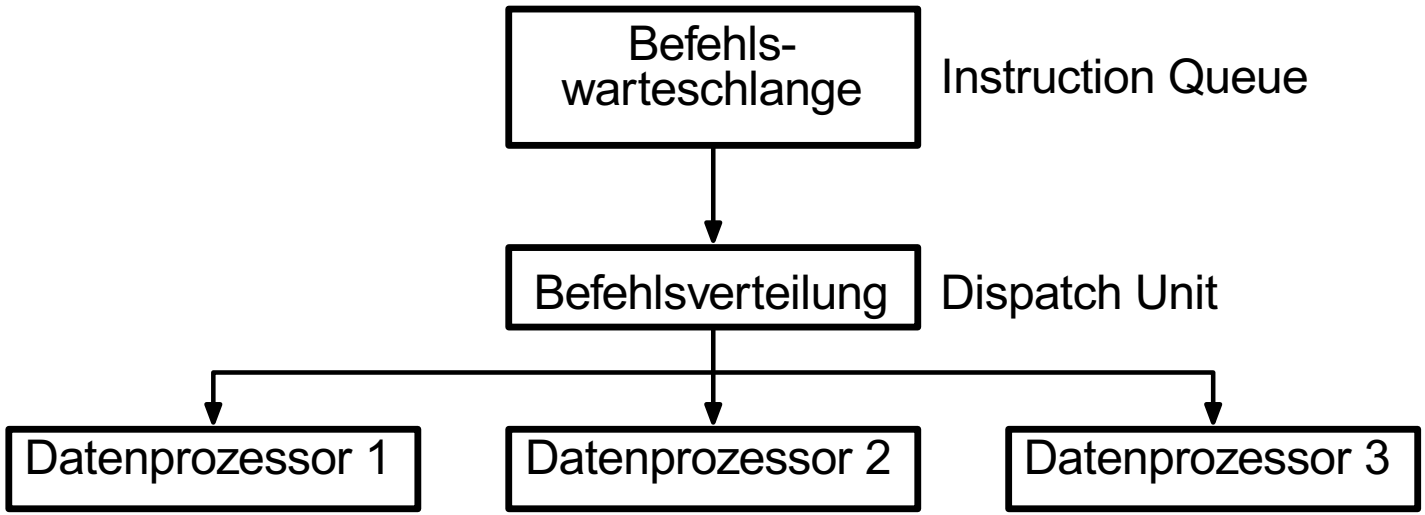


© Akash Kumar

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.



© Akash Kumar

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)
- 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

© Akash Kumar