TECHNISCHE UNIVERSITÄT DRESDEN

VLSI-EDA

Fakultät Informatik, Institut für Technische Informatik, Professur VLSI-Entwurfssysteme, Diagnostik und Architektur

Rechnerstrukturen und -organisation

Von Neumann Architektur

Rainer G. Spallek

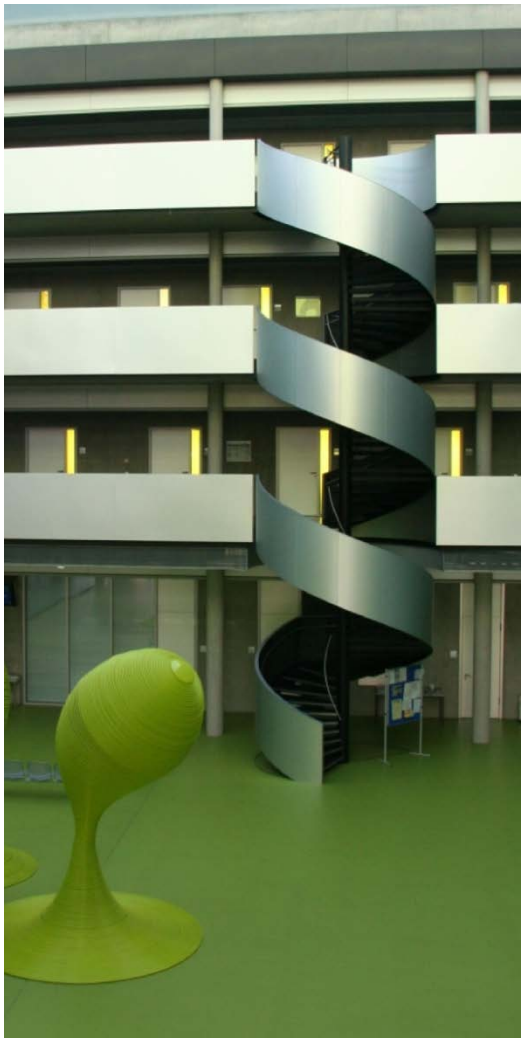
TU Dresden, 06.01.2021

Rainer.Spallek@tu-dresden.de

http://vlsi-eda.inf.tu-dresden.de

Gliederung

1 Zielstellung
2 Von Neumann Rechnerkonzept
3 Struktur des von Neumann Rechners
4 Komponenten des von Neumann Rechners
5 Befehlszyklus des von Neumann Rechners
6 Beispiel zum von Neumann Rechner
7 Engpässe (Flaschenhälse)
8 Alternative Architekturen
9 Zusammenfassung



1 Zielstellung

- Erlangung eines Grundverständnisses für die von Neumann Architektur.
- Realisierung einer Hardware-minimalen Rechnerarchitektur.
- Struktur und Organisation der von Neumann Rechnerarchitektur.
- Einführung des Befehlszyklus und der Abarbeitungsphasen.
- Erwerb von Grundlagen moderner Rechnerarchitekturen.
- Verständnis im Umgang mit der Komponenten eines Rechners.
- Vor- und Nachteile der von Neumann Architektur → mögliche Auswege.
- Alternative Rechnerkonzepte.

2 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 modernen praktisch verwendeten Rechner.

Das von Neumann Rechnerkonzept stellt ein „Hardware-minimales“ Konzept mit einem Optimum an Funktionalität und Leistungsfähigkeit dar.

Kriterien des von Neumann Rechners

1. - Hauptkomponenten des Rechners sind Rechenwerk, Steuerwerk, Speicher, Ein-/Ausgabeeinheit und Verbindungen (Datenwege, Busse).
   - Steuerwerk und Rechenwerk bilden zusammen die zentrale Verarbeitungseinheit (Prozessor, CPU).

2. - Die intern verwendete Signalmenge ist binär codiert.
   - Es werden Worte fester Länge parallel verarbeitet.
   - Die Verarbeitung erfolgt taktgesteuert.
   - Der Rechner arbeitet nach seinem Start automatisch.

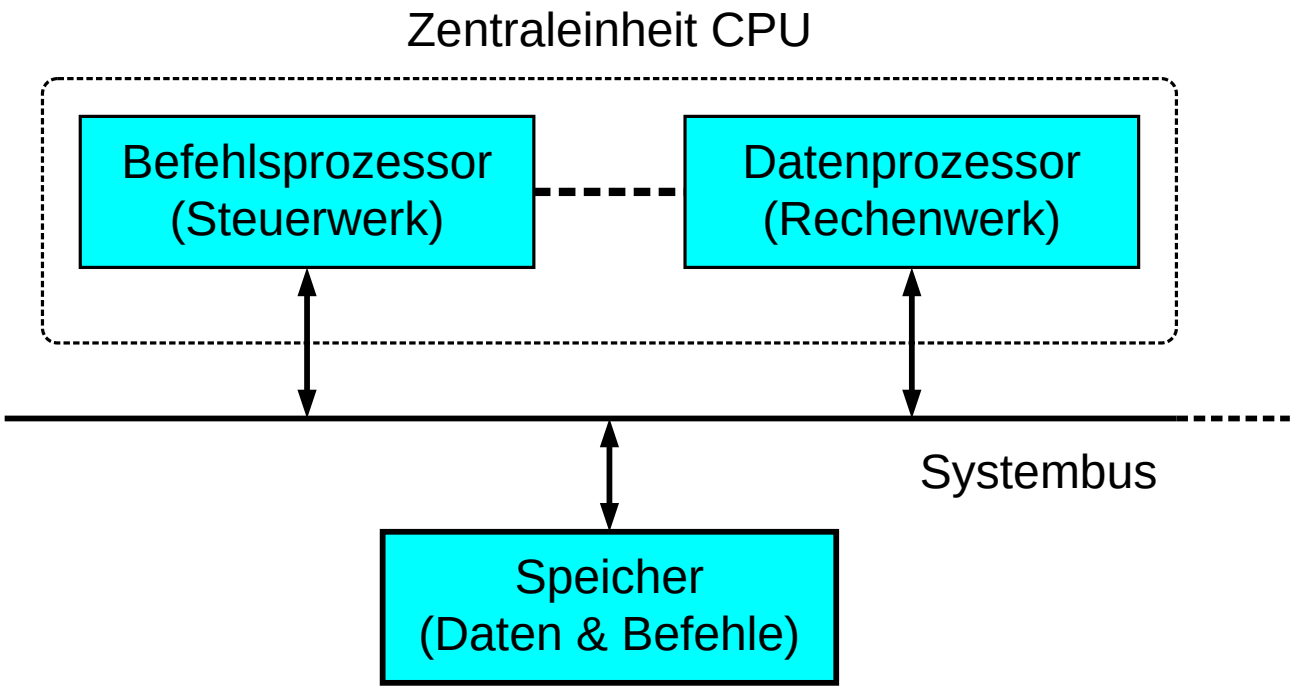
3. - Programmbefehle und Daten werden im einheitlichen Speicher ohne Kennzeichnung gespeichert.
   - Der Speicher besteht aus fortlaufend adressierten Speicherworten deren Inhalt nur über die Adresse angesprochen werden kann (von Neumann Variable → Adresse → Inhalt).

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 oder datenbedingte Verzweigungen unterbrochen werden.

5. - Die Architektur und Organisation des Rechners ist unabhängig von der zu bearbeitenden Aufgabe (Universalrechner, Software-Steuerung).
   - Die Steuerung des Rechners erfolgt über ein Programm (programmgesteuert), dass an die jeweilige zu bearbeitende Aufgabe angepasst ist (→ Programmablaufplan).
   - Die zu bearbeitende Aufgabe wird ausschließlich durch ein Programm und die dazugehörigen Daten repräsentiert.

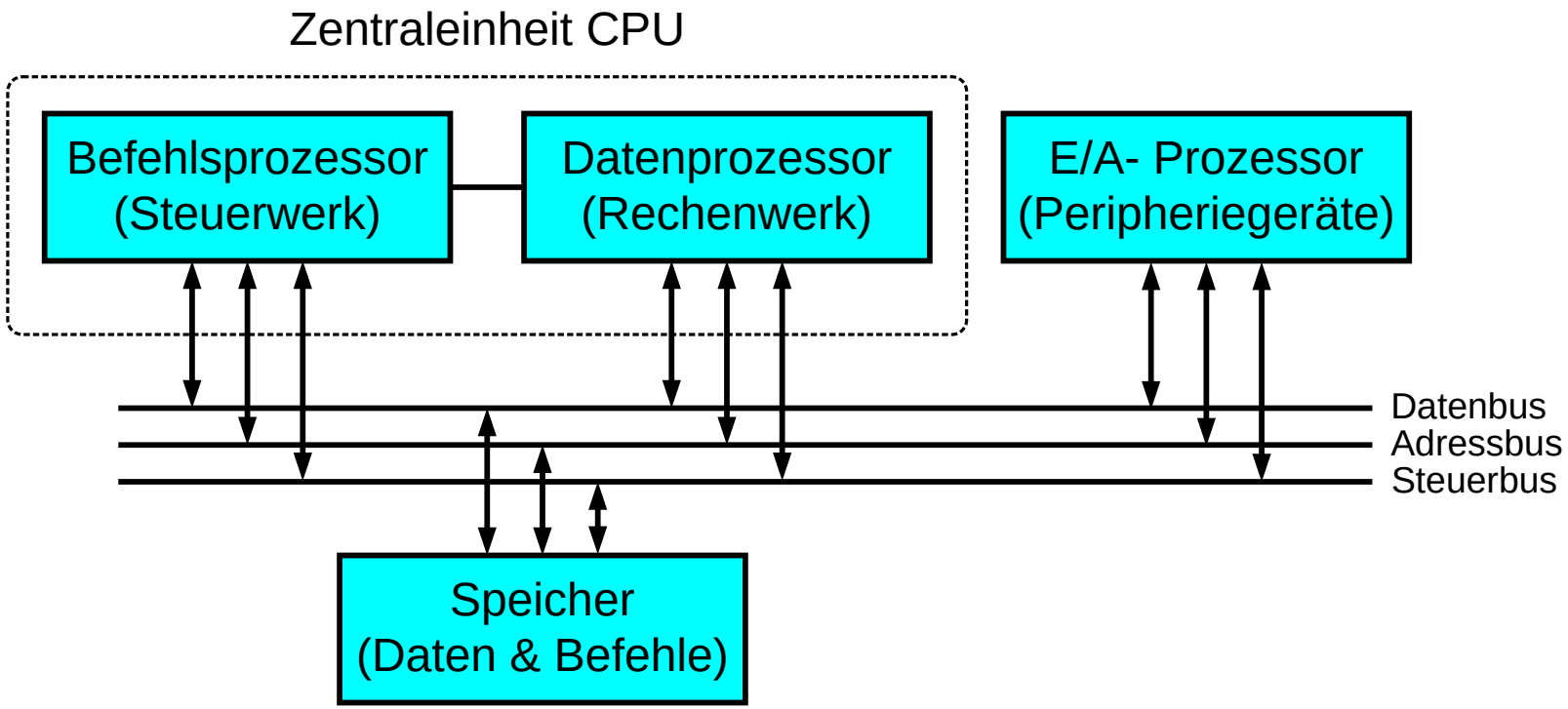
3 Struktur des von Neumann Rechners

Vereinfachte Princeton Architektur (Von Neumann Architektur)



Prozessorarchitektur: ohne E/A-Prozessoren und Rechnerumfeld

Struktur mit E/A-Prozessor und Busaufteilung



Unterteilung des Systembusses in Adress-, Daten- und Steuerbus (=> Multiplex)

Befehlswort des von Neumann Rechners

Einfacher Befehlsaufbau → Trennung in Operationsteil und Operandenteil

Struktur, Format des binärcodierten Befehls:

Operationsteil	**Operandenteil (Adressteil)**
binär codierte Operation	binäre Adresse des Operanden
oder Operationscode	oder Direktoperand
kurz Opcode	oder Verzweigungsadresse

Im Operationsteil (OT) wird die durchzuführende Operation, Anweisung binär codiert.

Der Operandenteil, Adressteil (AT) umfasst den Operanden bzw. dessen Adresse oder auch die Verzweigungsadresse, auf die die Operation angewendet wird.

4 Komponenten des von Neumann Rechners

- Steuerwerk (Befehlsprozessor)

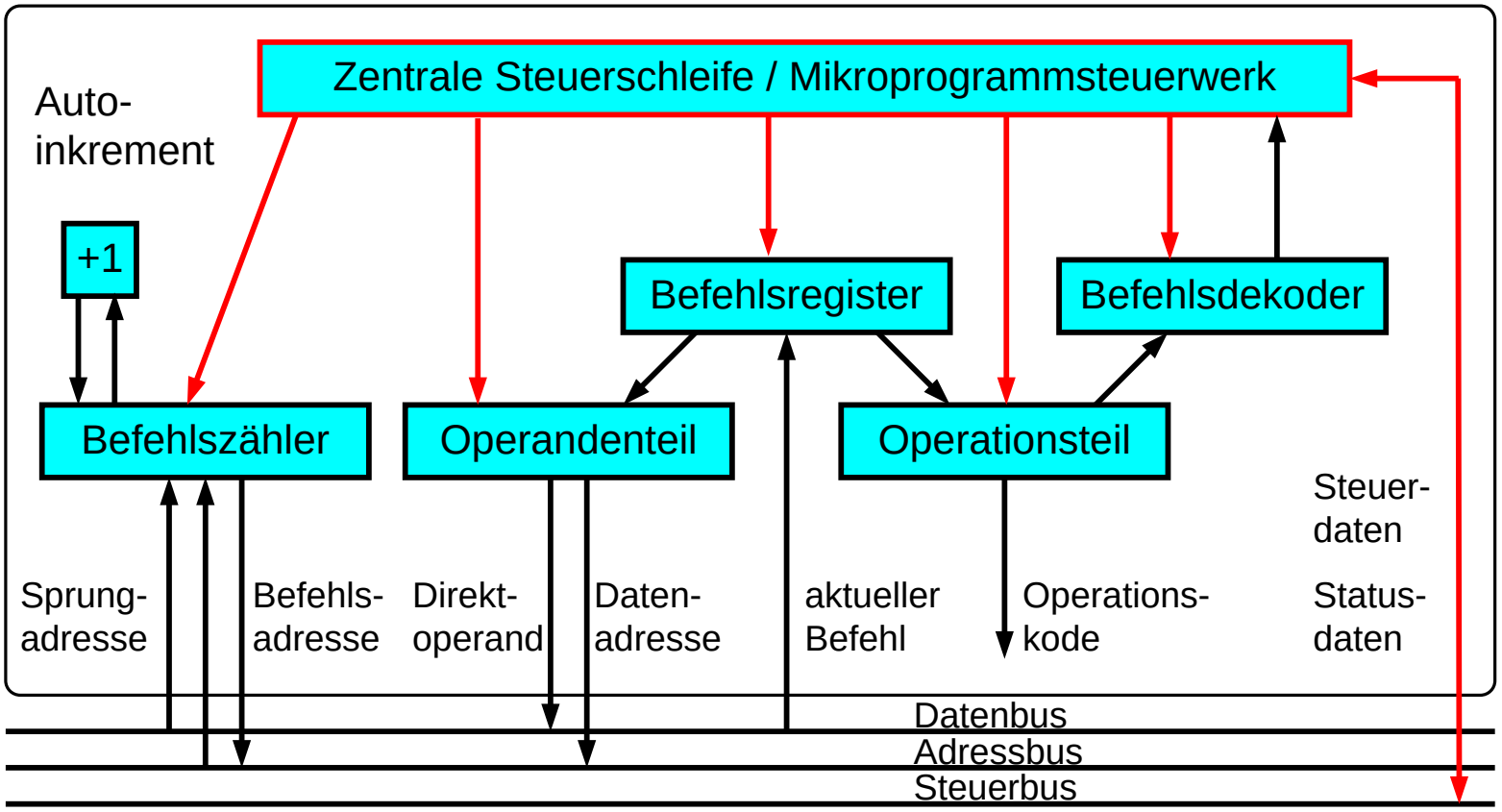
  → **Kontrollfluss.**

- Rechenwerk (Datenprozessor)

  → **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.

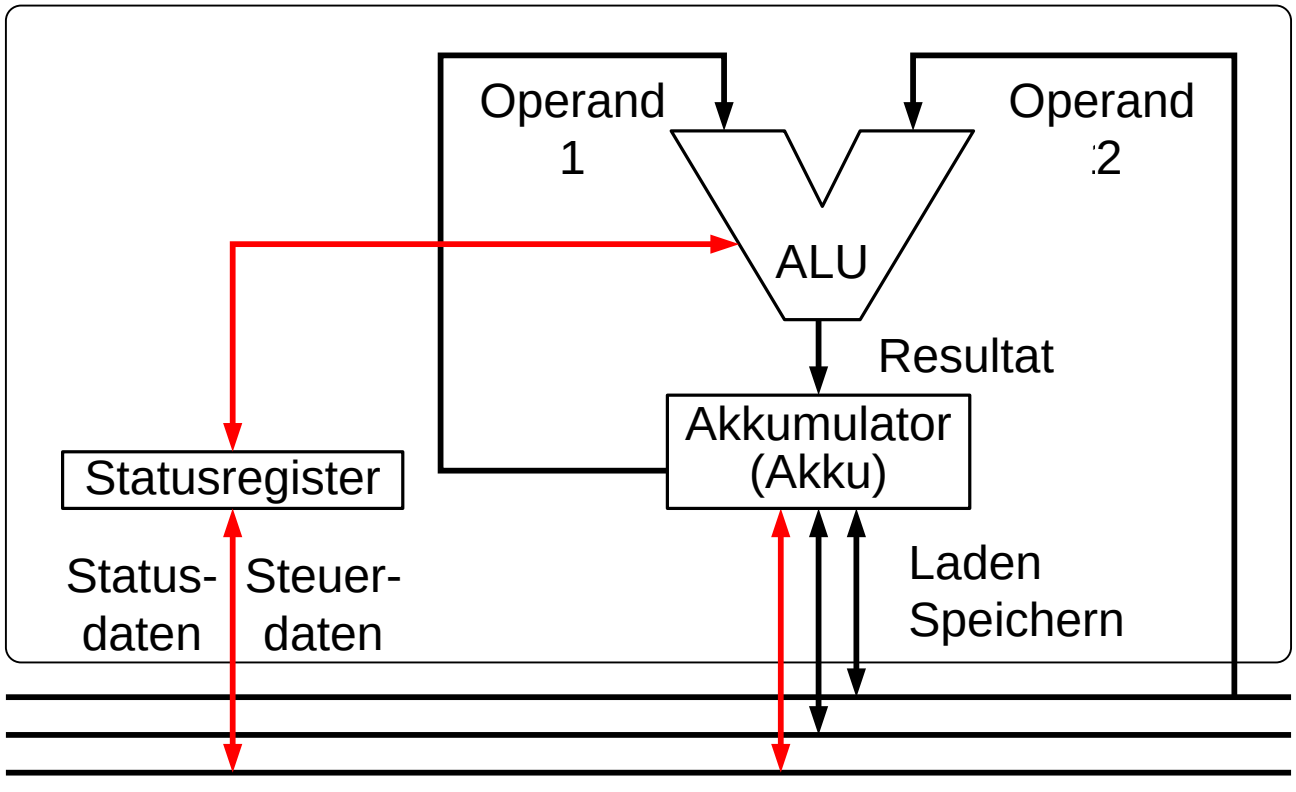
Befehlsdecoder – BD (ID – Instruction Decode):

Decodierung 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).

Rechenwerk, Arithmetic Logic Unit (ALU)



load O1
add O2
store R
(R):= (O1)+(O2)

O1-Adresse Operand1
O2-Adresse Operend2
R -Adresse Resultat
(O1)-Inhalt von Speicheradresse O1
(O2)-Inhalt von Speicheradresse O2
(R) -Inhalt von Speicheradresse R

Komponenten des Rechenwerkes

Verarbeitungseinheit – ALU (ALU – Arithmetic Logical Unit):

Realisiert arithmetische -, logische -, Vergleichs- und Verschiebeoperationen von Operanden zum Resultat (Operationen auch mit dem Akkumulator). Daten zur Operation und zum Ergebnis werden im Statusregister gespeichert.

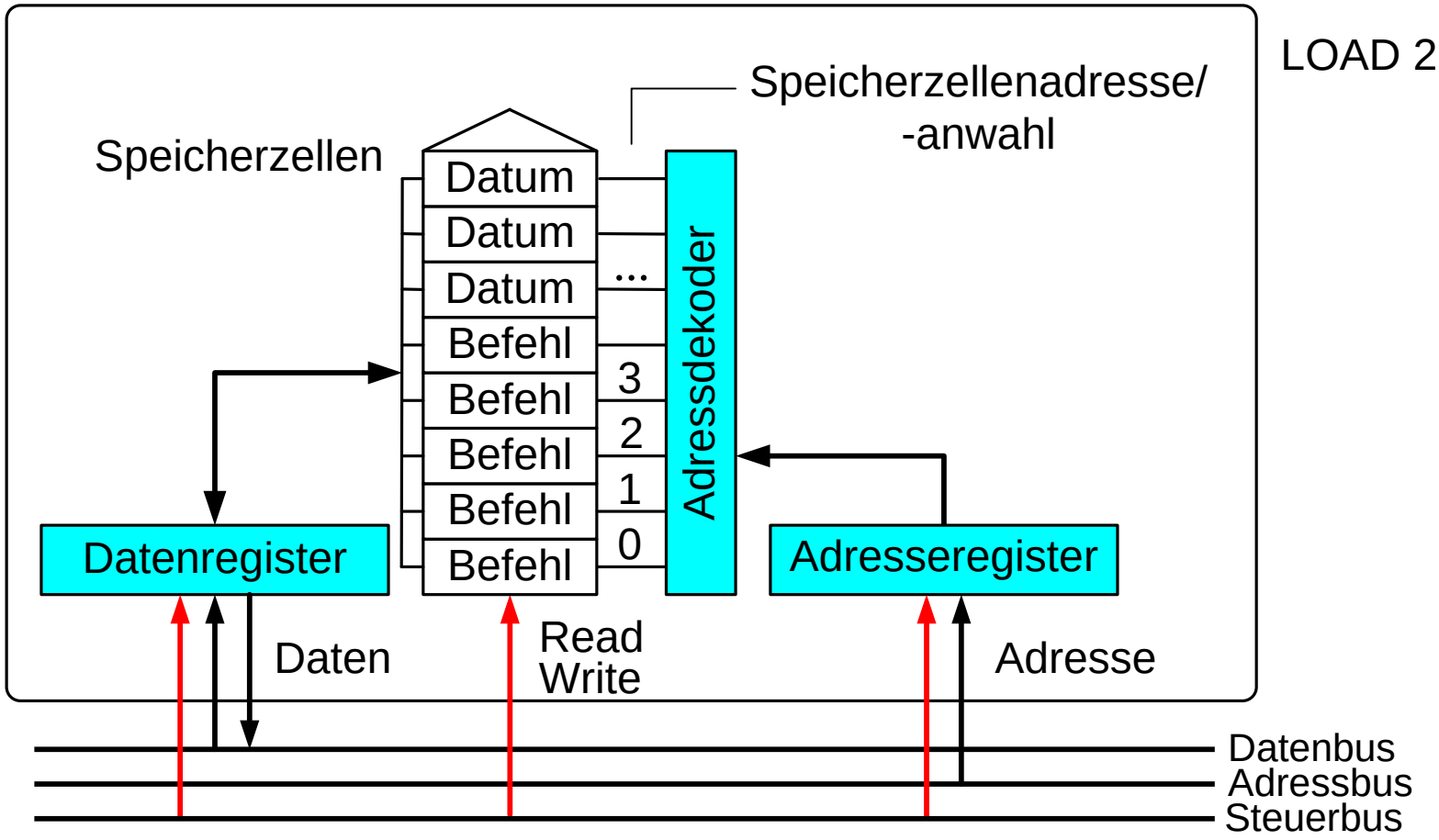
Akkumulator Register – AKKU (ACCU – Accumulator Register):

Universalregister zur Abspeicherung der Ergebnisdaten (Resultat). Gleichzeitig Hilfsregister zum Laden eines Operanden (load) bzw. zum Speichern des Resultates (store) und für den Speichertransfer. Verschiebeoperationen, Setzen / Rücksetzen auch im Akkumulator möglich.

Statusregister – SR (SR – Status Register):

Dient der Zwischenspeicherung der Steuerdaten des Steuerwerkes und der Statusdaten des Rechenwerkes (Bedingungscode, Fehler, . . .). Realisierung Daten-abhängiger Programmverzweigungen durch Abfrage der Statusinformationen (flags) des Statusregisters (bedingte Verzweigungen).

Speicher, Main Memory (RWM)



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 codiert abgespeichert.

Adressdecodierer – AD (DEC – Decoder):

Decodiert die binär codierte Adresse in einen 1-Aus-N Code und wählt die so adressierte Speicherzelle aus.

Speicheradressregister – SAR (MAR – Memory Adress Register):

Hilfsregister zur Zwischenspeicherung der binär codierten Speicheradresse (kann auch im Steuerwerk lokalisiert sein).

Speicherdatenregister – SDR (MDR – Memory Data Register):

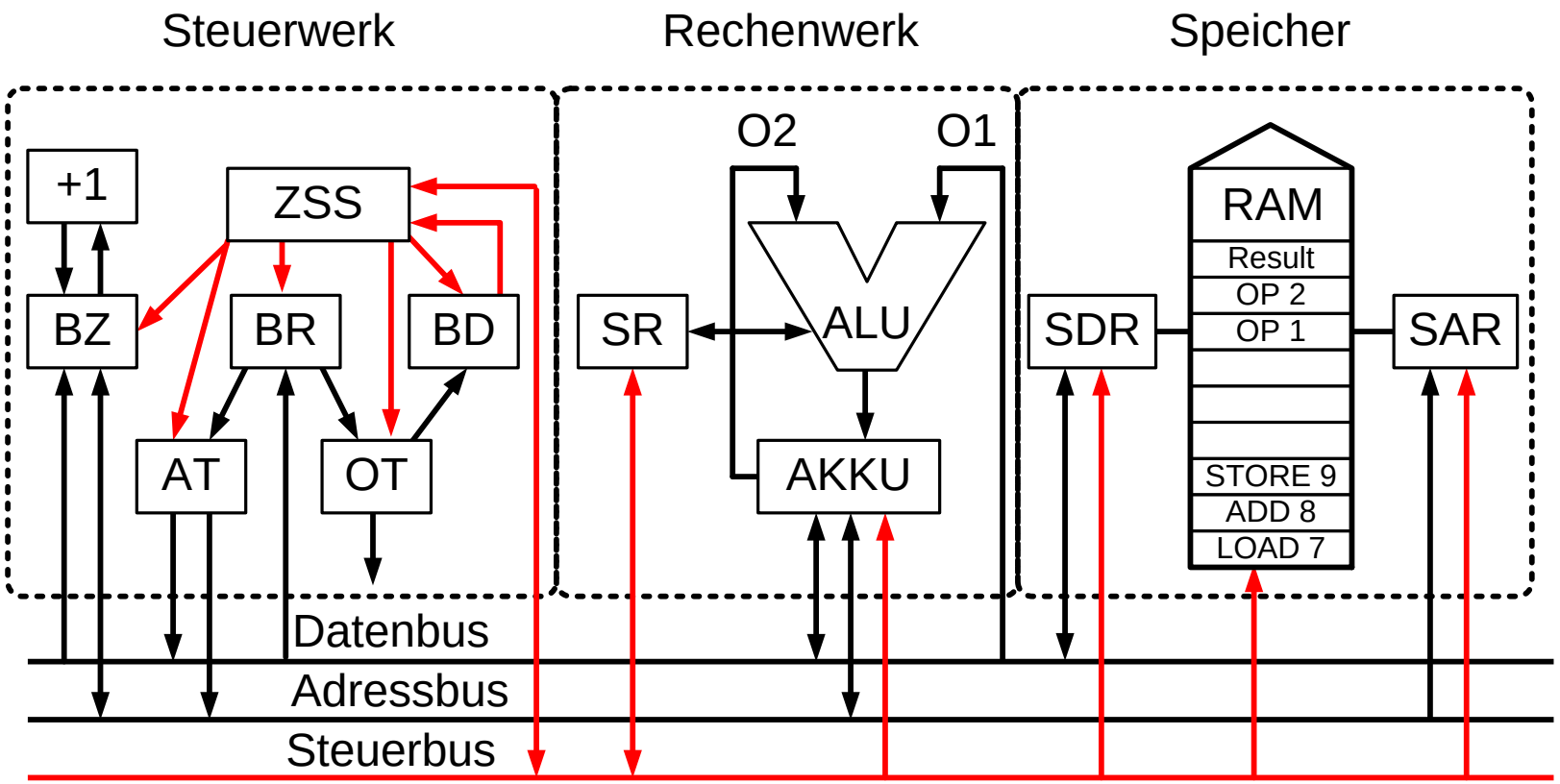
Hilfsregister zur Zwischenspeicherung der zu speichernden oder gelesenen Daten und Befehle (kann auch im Rechenwerk/Steuerwerk lokalisiert sein).

Ein-/Ausgabeprozessor, I/O-Processor

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

- Realisierung des Benutzerinterfaces (Bedienersteuerung mit Keyboard, Mouse, Monitor . . .)
- Abfrage von Sensoren und Steuerung von Aktuatoren (Automatisierungstechnik, Anlagensteuerung, . . .)
- Realisierung der Netzwerkkommunikation (LAN, WAN**,** Firewire, . . .)
- Realisierung spezieller Ein-/Ausgabe-Bussysteme, abgesetzte Busse zum Datentransfer (seriell, parallel, SCSI, USB, Firewire, . . .)
- Steuerung der Ein-/Ausgaben von externen Geräten (Interrupt-, Abfragemethode, . . .)
- Anschluss externer Speichermedien (Archivspeicher, Austauschmedien, Backup, . . .)

Zusammenfassung der Komponenten



5 Befehlszyklus des von Neumann Rechners

Der Befehlszyklus durchläuft aufgrund des gesonderten Zugriffs auf Befehle und Daten im Speicher zwei zeitlich voneinander getrennte Phasen.
Nur ein Speicher für Befehle und Daten und nur ein Bus → Engpass.

1. Phase (Holphase) – Befehl holen und decodieren

Befehl holen (IF – Instruction Fetch):

Der Befehl wird entsprechend der Adresse im Befehlszähler aus dem Speicher geladen und im Steuerwerk im Befehlsregister abgelegt.
Der Befehlszähler wird gleichzeitig inkrementiert.

Befehl decodieren (ID – Instruction Decode):

Der Befehlsdecoder decodiert den Operationsteil des Befehls aus dem Befehlsregister. Der decodierte Operationsteil wird durch das Mikroprogrammsteuerwerk in der Zentralen Steuerschleife interpretiert.
Die zentrale Steuerschleife übernimmt die Steuerung der Abarbeitung.

2. Phase (Ausführungsphase) – Operand holen und Operation ausführen

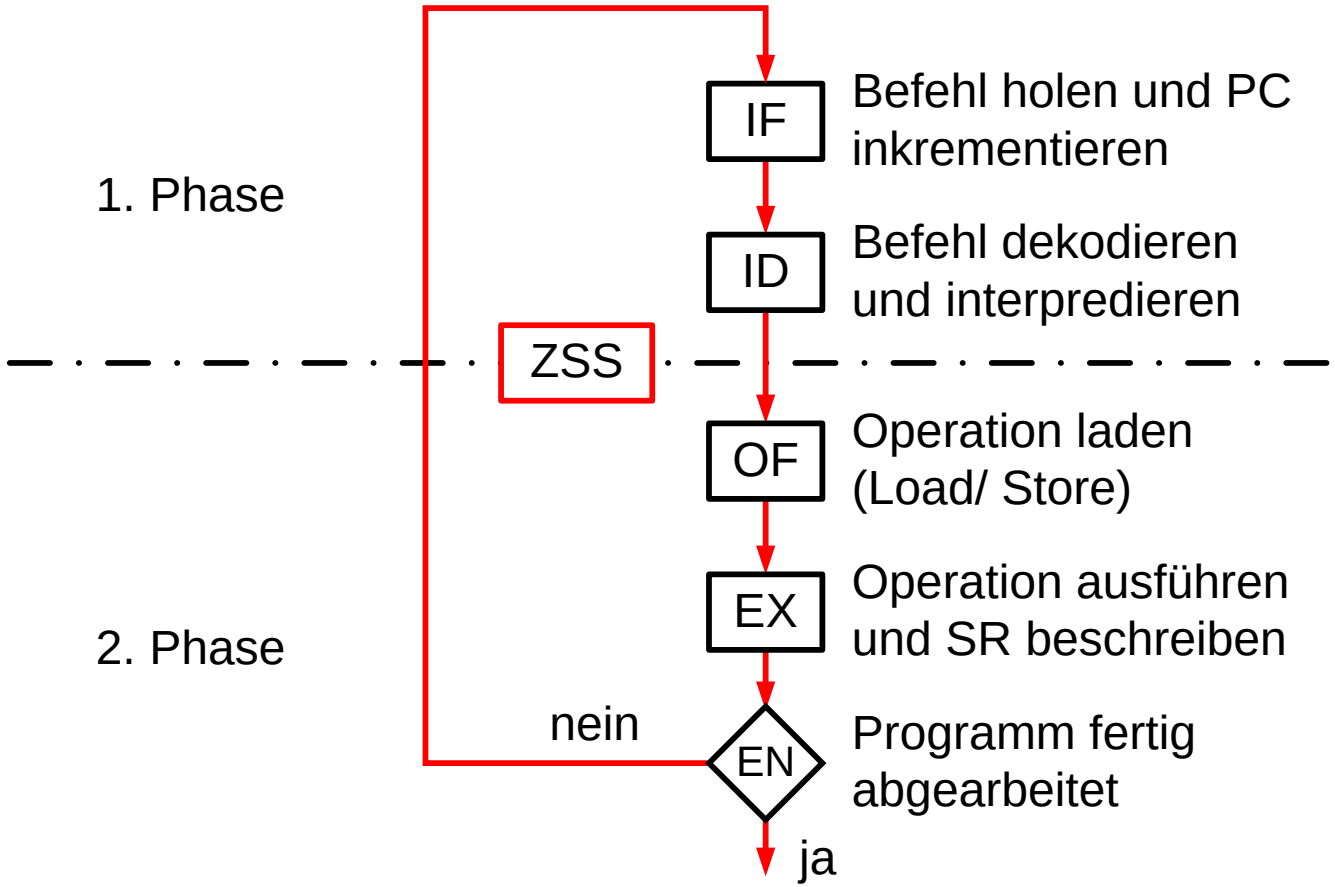
Operand holen (OF – Operand Fetch):

Der Operand wird entsprechend der Adresse aus dem Adressteil des Befehlsregisters aus dem Speicher geladen und dem Rechenwerk zugeführt. Er kann auch als Direktoperand direkt vom Adressteil des Befehlsregisters dem Rechenwerk zugeführt werden.
Bei **load** wird der Akkumulator mit dem Operanden aus dem Speicher entsprechend der Adresse des Adressteils des Befehlsregisters geladen.
Bei **store** wird der Inhalt des Akkumulators entsprechend der Adresse des Adressteils des Befehlsregisters in den Speicher geschrieben.

Operation ausführen (EX – Execute):

Die im Operationsteil codierte Operation wird durch ein Mikroprogramm ZSS gesteuert in der ALU mit dem Akkumulator und dem aus dem Speicher bzw. Adressteil zugeführten Operanden durchgeführt. Die Statusinformationen der Operation werden im Statusregister abgelegt. Der Befehlszähler kann ebenfalls durch das Rechenwerk beschrieben und gelesen werden.

Befehlszyklus, Instruction Cycle (IC)



Diese beiden zeitlich getrennten Phasen wechseln sich zyklisch ab (Befehlszyklus, Instruction Cycle, IC)
Ursache: Getrennter Speicherzugriff für Befehle und Daten im Speicher.

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

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

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

- mehrstufige Adressberechnungen, komplexe Verzweigungen
- das Lesen und Decodieren unterschiedlichster Befehlsformate
- Mehrfachzugriffe auf den Speicher (Bearbeitung größerer Datenformate)
- die Ausführung komplexer Mikroprogramme in der Ausführungsphase.

6 Beispiel zum von Neumann Rechner

Befehlsformat:	8-Bit	3-Bit Opcode, 5-Bit Adressteil
Datenformat:	8-Bit	binärcodierte Dualzahlen, Wertebereich: 0 bis 255
Befehlsvorrat:	3-Bit	maximal 8 verschiedene Befehle codierbar
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 := O1 + O2*

	Befehls-adresse	Befehl	Opcode	Daten-adresse	Bemerkung
I1	0000 0001	LOAD O1	001	0000 1011	lade O1 in Akku
I2	0000 0010	ADD O2	100	0000 1100	addiere O2 mit Akku
I3	0000 0011	STORE R	010	0000 1101	speichere Akku auf R

Befehlszyklus 1
1. Phase
LOAD O1

BZ

0000 0001

BD

Opcode Adresse

000 | 0000 0000

SAR

0000 0000

BR

0000 0000

SDR

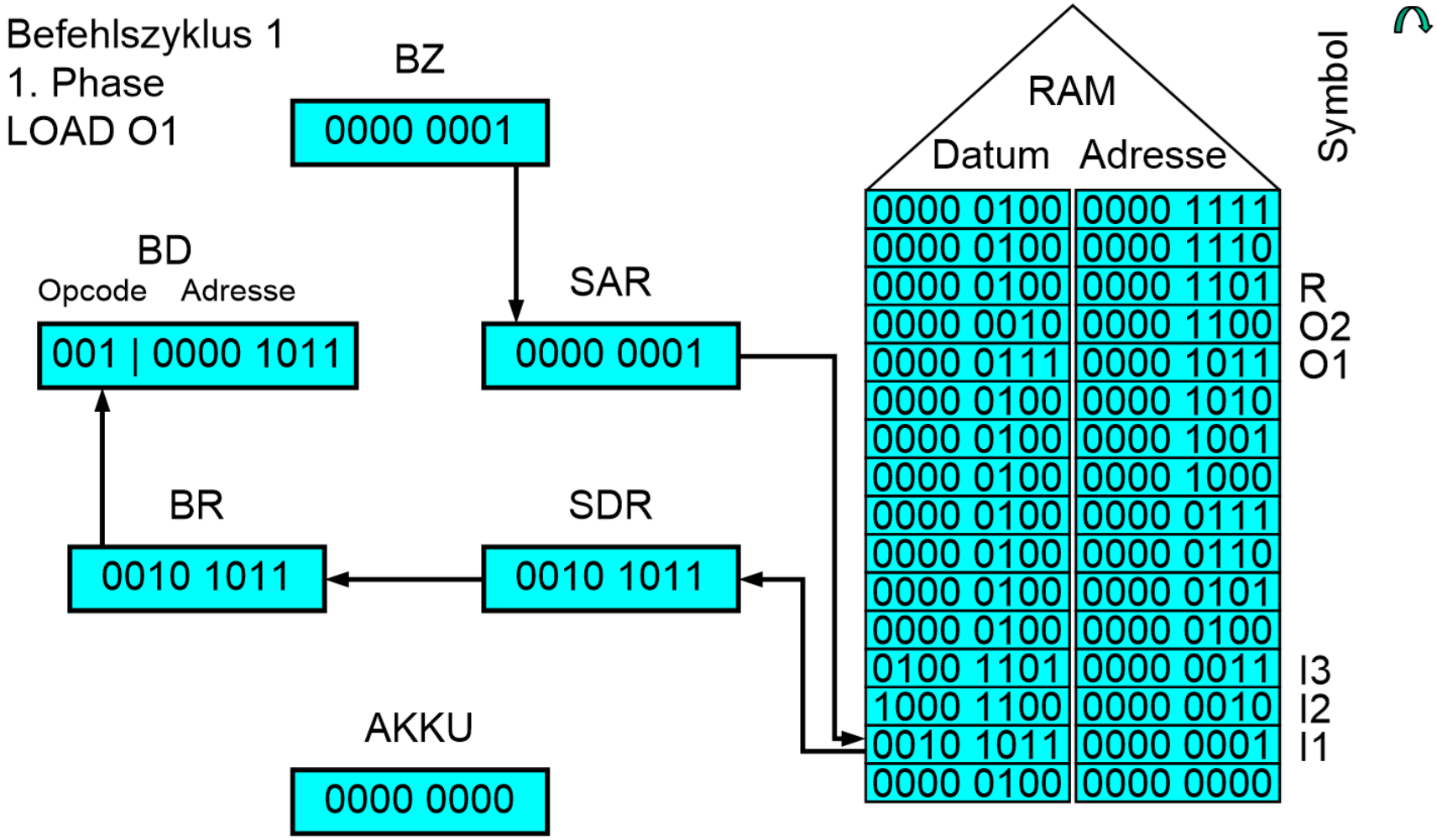
0000 0000

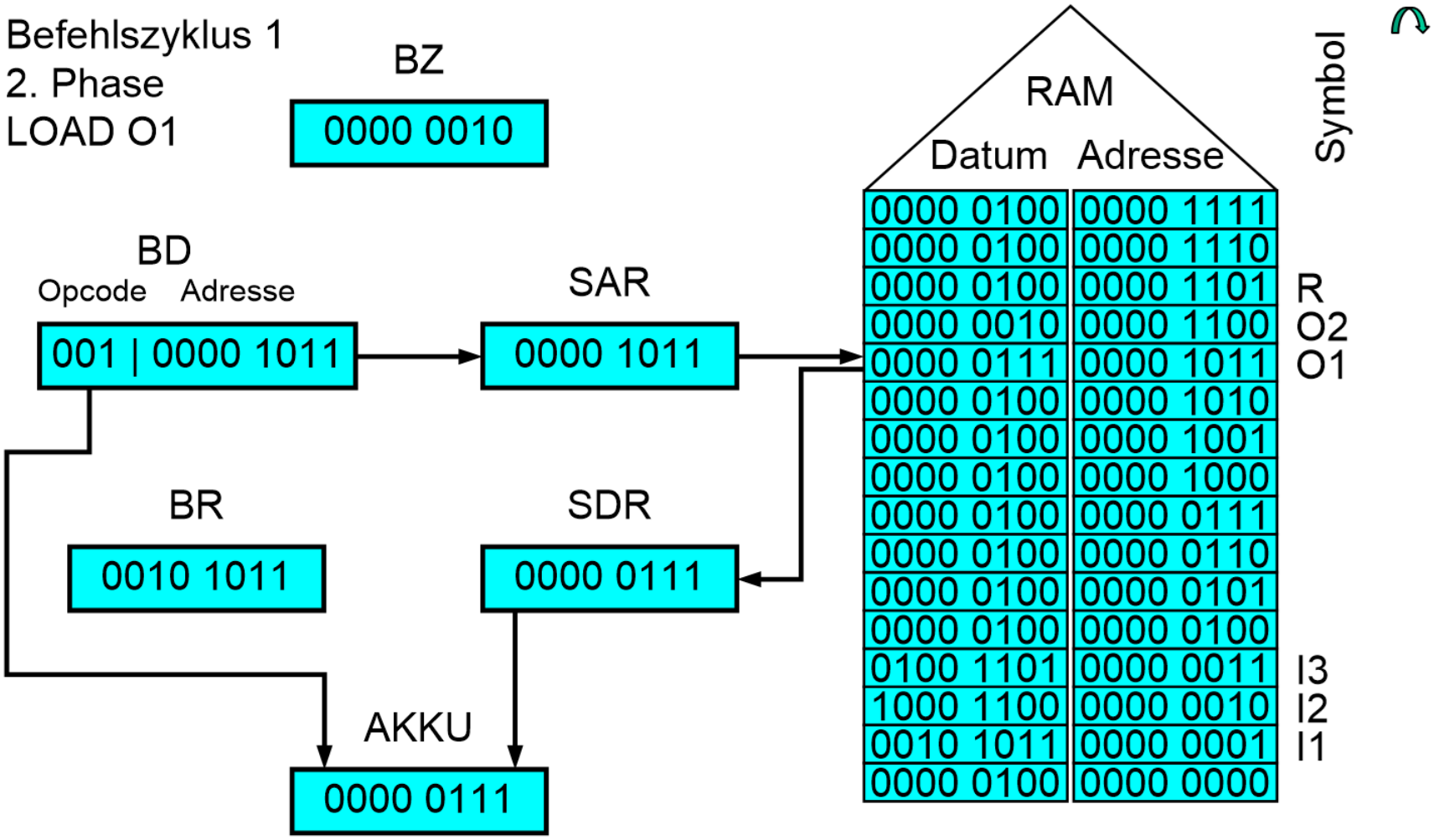
AKKU

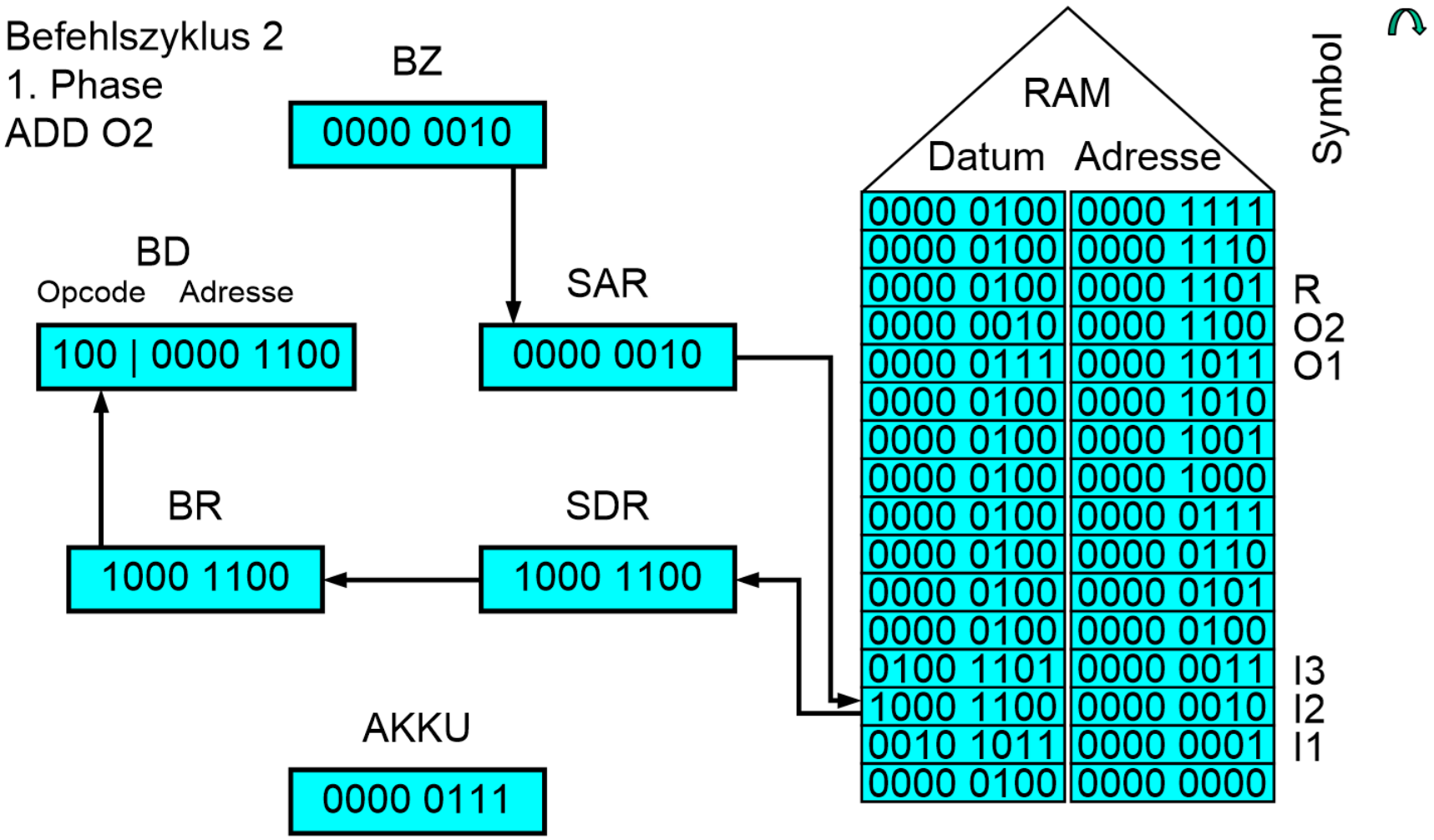
0000 0000

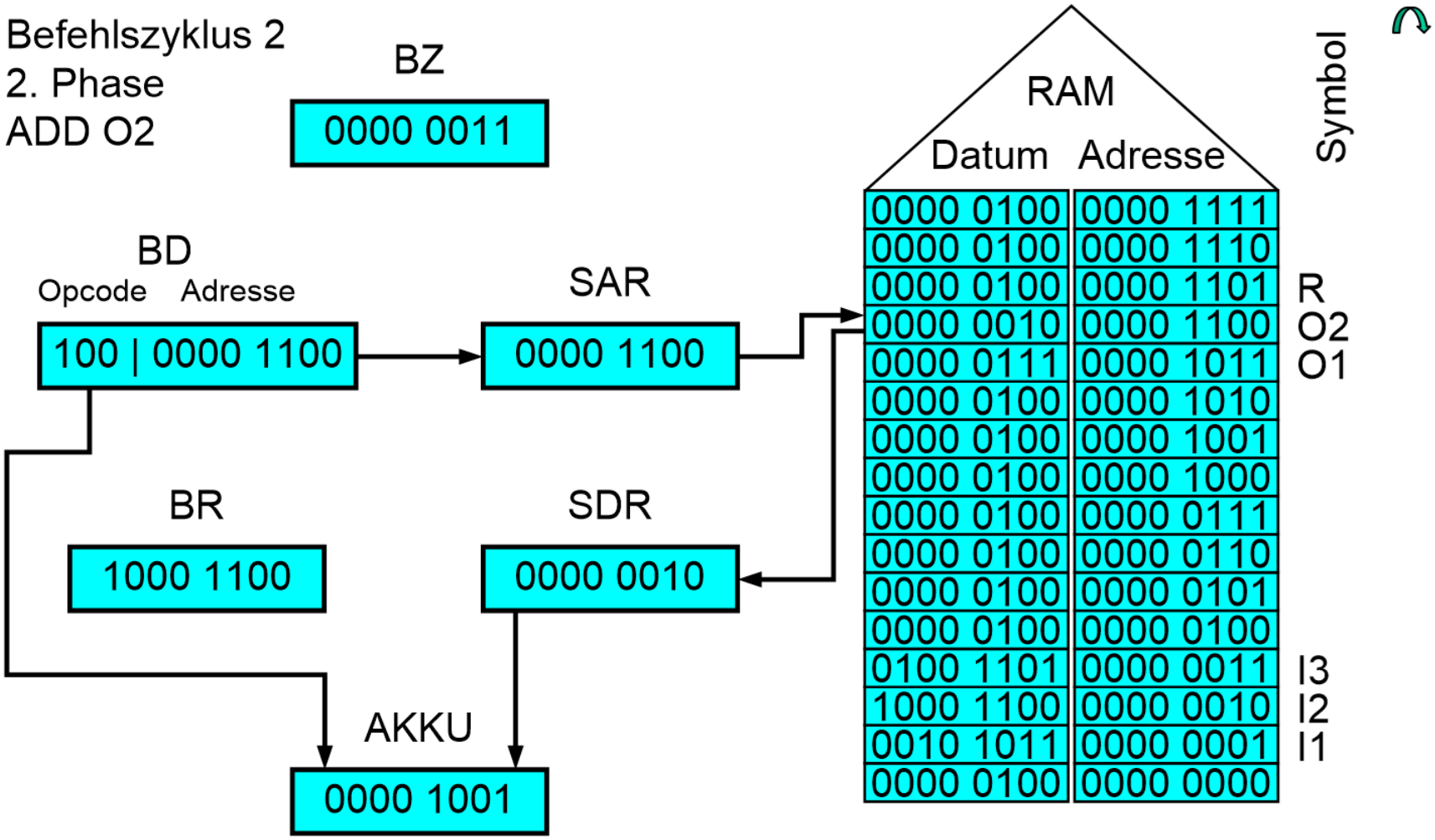
RAM

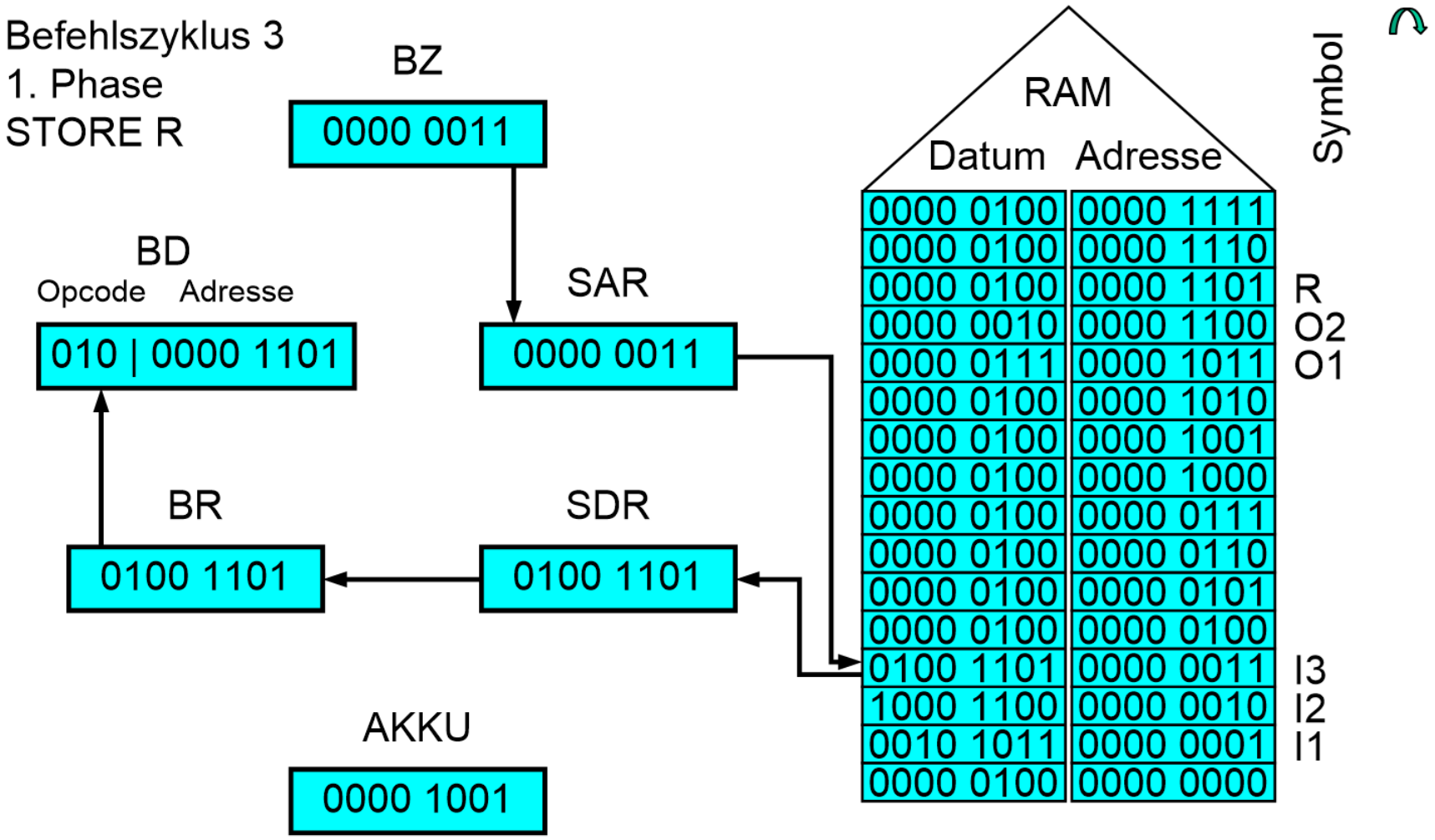
Datum	Adresse	Symbol
0000 0100	0000 1111	
0000 0100	0000 1110	
0000 0100	0000 1101	R
0000 0010	0000 1100	O2
0000 0111	0000 1011	O1
0000 0100	0000 1010	
0000 0100	0000 1001	
0000 0100	0000 1000	
0000 0100	0000 0111	
0000 0100	0000 0110	
0000 0100	0000 0101	
0000 0100	0000 0100	
0100 1101	0000 0011	I3
1000 1100	0000 0010	I2
0010 1011	0000 0001	I1
0000 0100	0000 0000	

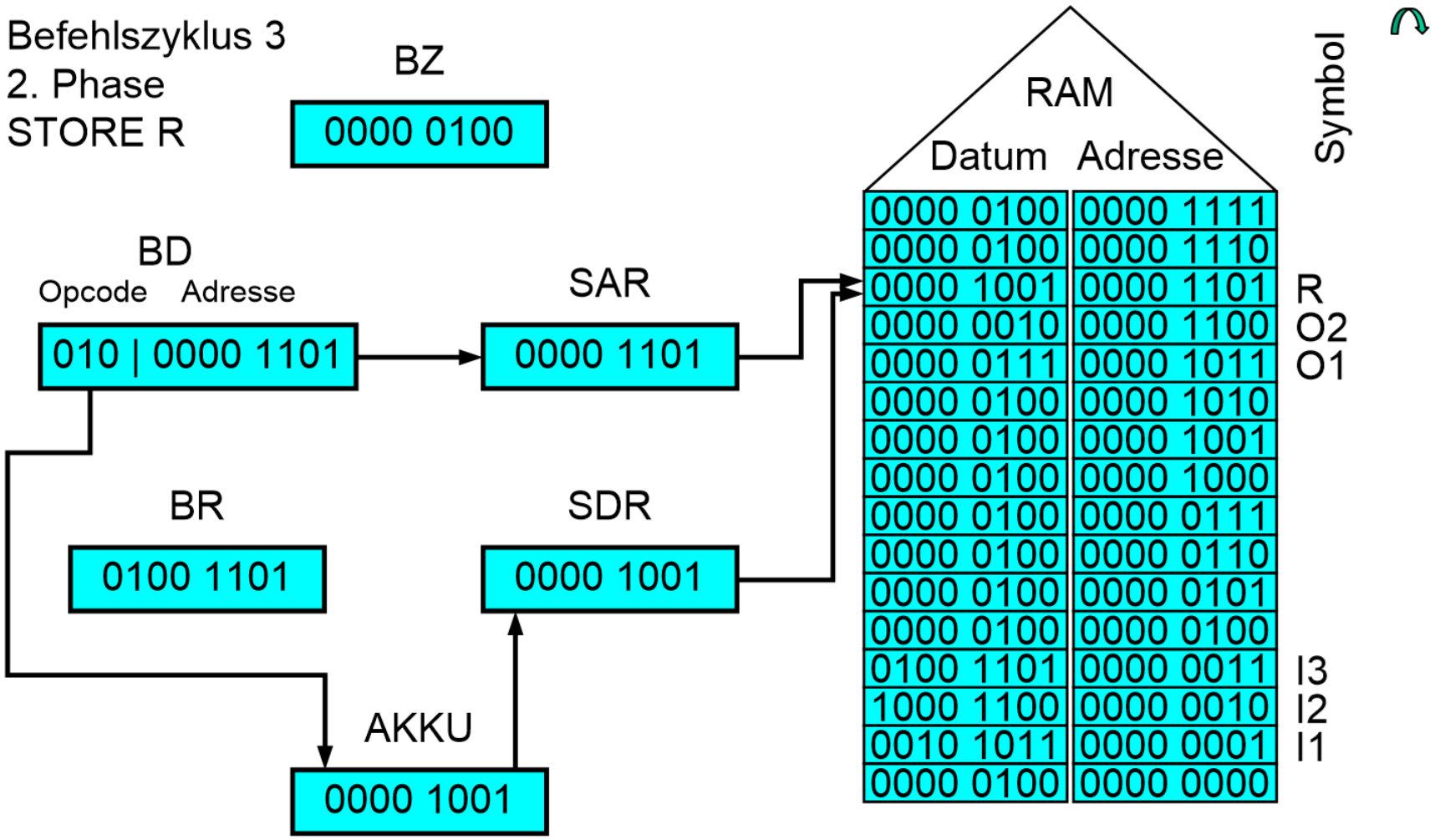

Befehlszyklus 1
1. Phase
LOAD O1
BZ
0000 0001
BD
Opcode Adresse
001 | 0000 1011
SAR
0000 0001
BR
0010 1011
SDR
0010 1011
AKKU
0000 0000
RAM
Datum Adresse
Symbol
0000 0100 0000 1111
0000 0100 0000 1110
0000 0100 0000 1101 R
0000 0010 0000 1100 O2
0000 0111 0000 1011 O1
0000 0100 0000 1010
0000 0100 0000 1001
0000 0100 0000 1000
0000 0100 0000 0111
0000 0100 0000 0110
0000 0100 0000 0101
0000 0100 0000 0100
0100 1101 0000 0011 I3
1000 1100 0000 0010 I2
0010 1011 0000 0001 I1
0000 0100 0000 0000


Befehlszyklus 1
2. Phase
LOAD O1
BZ
0000 0010
BD
Opcode Adresse
001 | 0000 1011
SAR
0000 1011
BR
0010 1011
SDR
0000 0111
AKKU
0000 0111
RAM
Datum Adresse
Symbol
0000 0100 0000 1111
0000 0100 0000 1110
0000 0100 0000 1101 R
0000 0010 0000 1100 O2
0000 0111 0000 1011 O1
0000 0100 0000 1010
0000 0100 0000 1001
0000 0100 0000 1000
0000 0100 0000 0111
0000 0100 0000 0110
0000 0100 0000 0101
0000 0100 0000 0100
0100 1101 0000 0011 I3
1000 1100 0000 0010 I2
0010 1011 0000 0001 I1
0000 0100 0000 0000


Befehlszyklus 2
1. Phase
ADD O2
BZ
0000 0010
BD
Opcode Adresse
100 | 0000 1100
SAR
0000 0010
BR
1000 1100
SDR
1000 1100
AKKU
0000 0111
RAM
Datum Adresse
Symbol
0000 0100 0000 1111
0000 0100 0000 1110
0000 0100 0000 1101 R
0000 0010 0000 1100 O2
0000 0111 0000 1011 O1
0000 0100 0000 1010
0000 0100 0000 1001
0000 0100 0000 1000
0000 0100 0000 0111
0000 0100 0000 0110
0000 0100 0000 0101
0000 0100 0000 0100
0100 1101 0000 0011 I3
1000 1100 0000 0010 I2
0010 1011 0000 0001 I1
0000 0100 0000 0000


Befehlszyklus 2
2. Phase
ADD O2
BZ
0000 0011
BD
Opcode Adresse
100 | 0000 1100
SAR
0000 1100
BR
1000 1100
SDR
0000 0010
AKKU
0000 1001
RAM
Datum Adresse
Symbol
0000 0100 0000 1111
0000 0100 0000 1110
0000 0100 0000 1101 R
0000 0010 0000 1100 O2
0000 0111 0000 1011 O1
0000 0100 0000 1010
0000 0100 0000 1001
0000 0100 0000 1000
0000 0100 0000 0111
0000 0100 0000 0110
0000 0100 0000 0101
0000 0100 0000 0100
0100 1101 0000 0011 I3
1000 1100 0000 0010 I2
0010 1011 0000 0001 I1
0000 0100 0000 0000


Befehlszyklus 3
1. Phase
STORE R
BZ
0000 0011
BD
Opcode Adresse
010 | 0000 1101
SAR
0000 0011
BR
0100 1101
SDR
0100 1101
AKKU
0000 1001
RAM
Datum Adresse
Symbol
0000 0100 0000 1111
0000 0100 0000 1110
0000 0100 0000 1101 R
0000 0010 0000 1100 O2
0000 0111 0000 1011 O1
0000 0100 0000 1010
0000 0100 0000 1001
0000 0100 0000 1000
0000 0100 0000 0111
0000 0100 0000 0110
0000 0100 0000 0101
0000 0100 0000 0100
0100 1101 0000 0011 I3
1000 1100 0000 0010 I2
0010 1011 0000 0001 I1
0000 0100 0000 0000


Befehlszyklus 3
2. Phase
STORE R
BZ
0000 0100
BD
Opcode Adresse
010 | 0000 1101
SAR
0000 1101
BR
0100 1101
SDR
0000 1001
AKKU
0000 1001
RAM
Datum Adresse
Symbol
0000 0100 0000 1111
0000 0100 0000 1110
0000 1001 0000 1101 R
0000 0010 0000 1100 O2
0000 0111 0000 1011 O1
0000 0100 0000 1010
0000 0100 0000 1001
0000 0100 0000 1000
0000 0100 0000 0111
0000 0100 0000 0110
0000 0100 0000 0101
0000 0100 0000 0100
0100 1101 0000 0011 I3
1000 1100 0000 0010 I2
0010 1011 0000 0001 I1
0000 0100 0000 0000

7 Engpässe (Flaschenhälse)

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 so gegenseitig (Flaschenhals)
→ Trennung des Befehlszyklus in 2 Phasen (1. Befehl- 2. Daten holen).

Ausweg: Getrennte Speicher und Busse für Befehle und Daten.
→ Harvard – Architektur oder getrennte Caches für Befehle und Daten

2. Sequentielle Abarbeitung:

Sequentielle Abarbeitung der einzelnen Befehle (Instruktionen), einzeln, nacheinander. Keine Möglichkeit der gleichzeitigen, parallelen Ausführung mehrerer Befehle oder eines Befehls mit mehreren Daten.

Ausweg: Nutzung von Parallelität auf allen Ebenen
→ Bitebene → Befehlsebene → Kontrollflussebene
→ Programmebene → Datenflussebene.

3. Kontrollpfad:

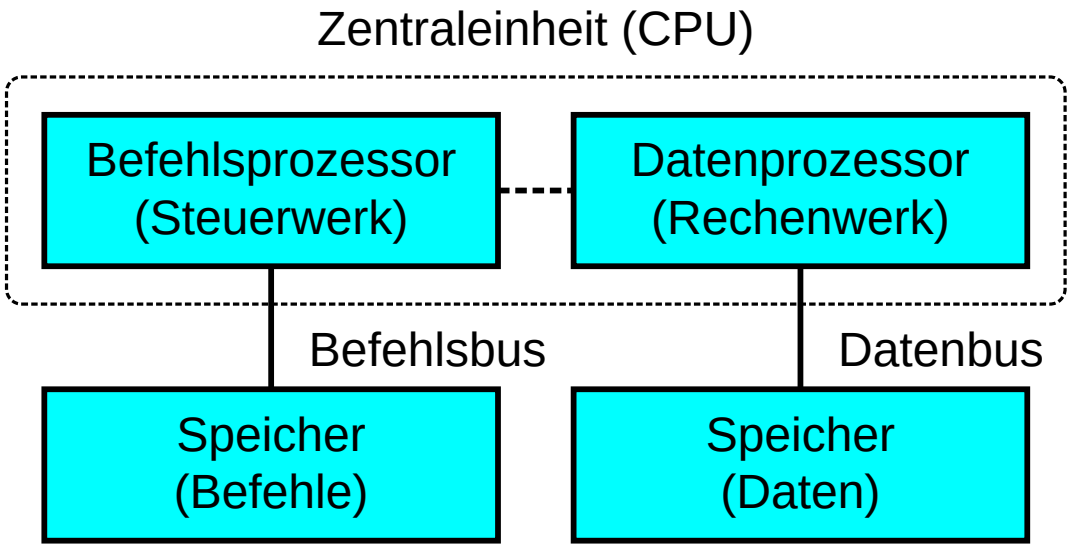
Es gibt nur einen Kontrollfluss mit festem Kontrollpfad. Die Befehlsabfolge und Steuerung ist nur in einer Ebene möglich (nur ein Befehlszähler, Befehlsregister, Akkumulator). Daraus resultieren Probleme bei der Unterprogrammtechnik, wie auch der Multitasking bzw. Multiuser Nutzung.

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

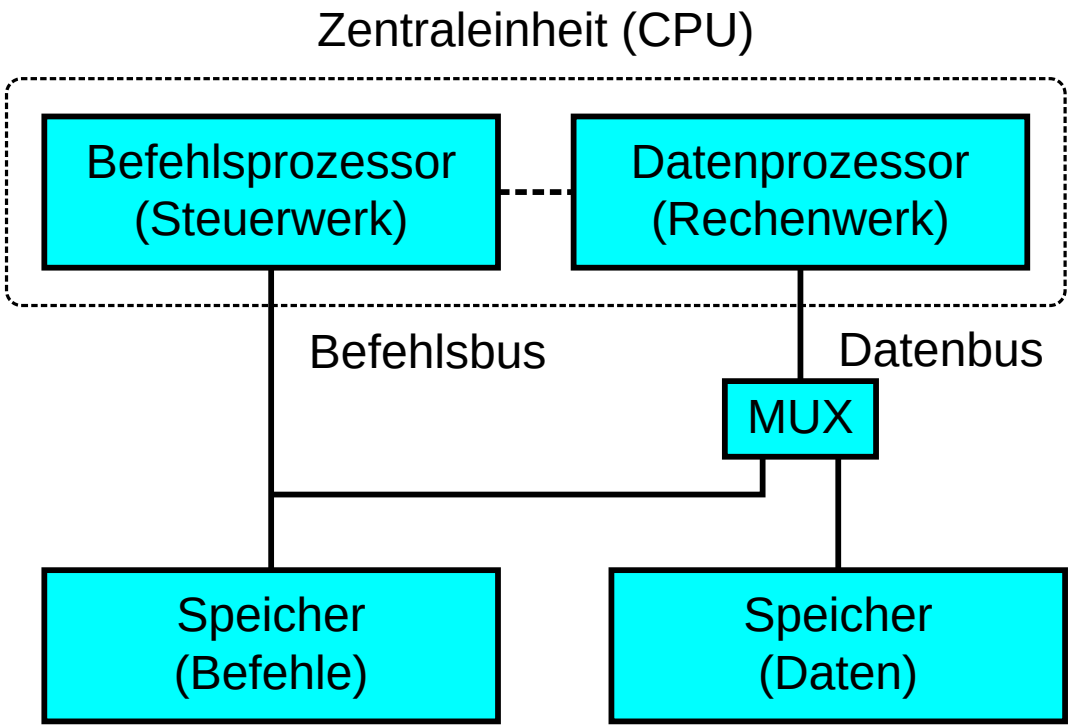
8 Alternative Architekturen

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.



Erweiterte Harvard – Architektur



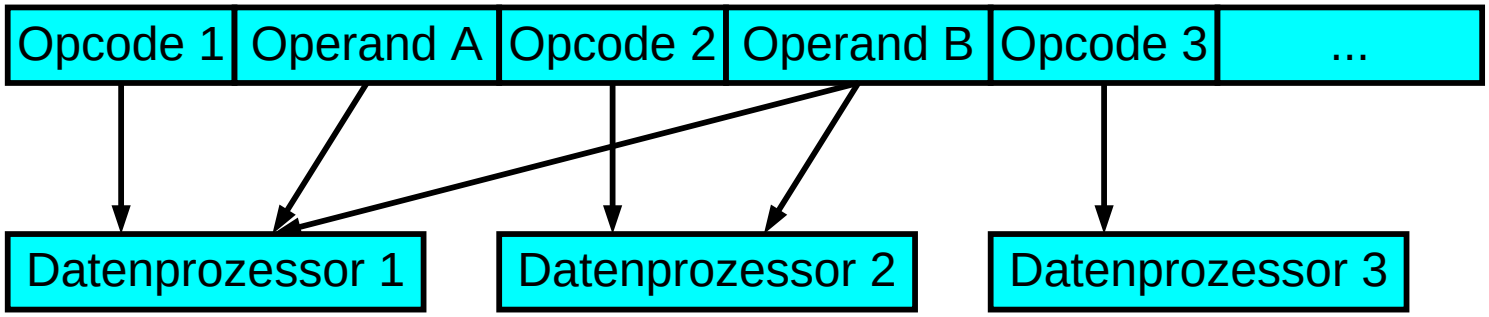
Durch den Multiplexer kann ein Operandenzugriff auch auf den Befehlsspeicher erfolgen (→ Operandenzugriff auf zwei Speicher).

VLIW – Architektur (Very Long Instruction Word)

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

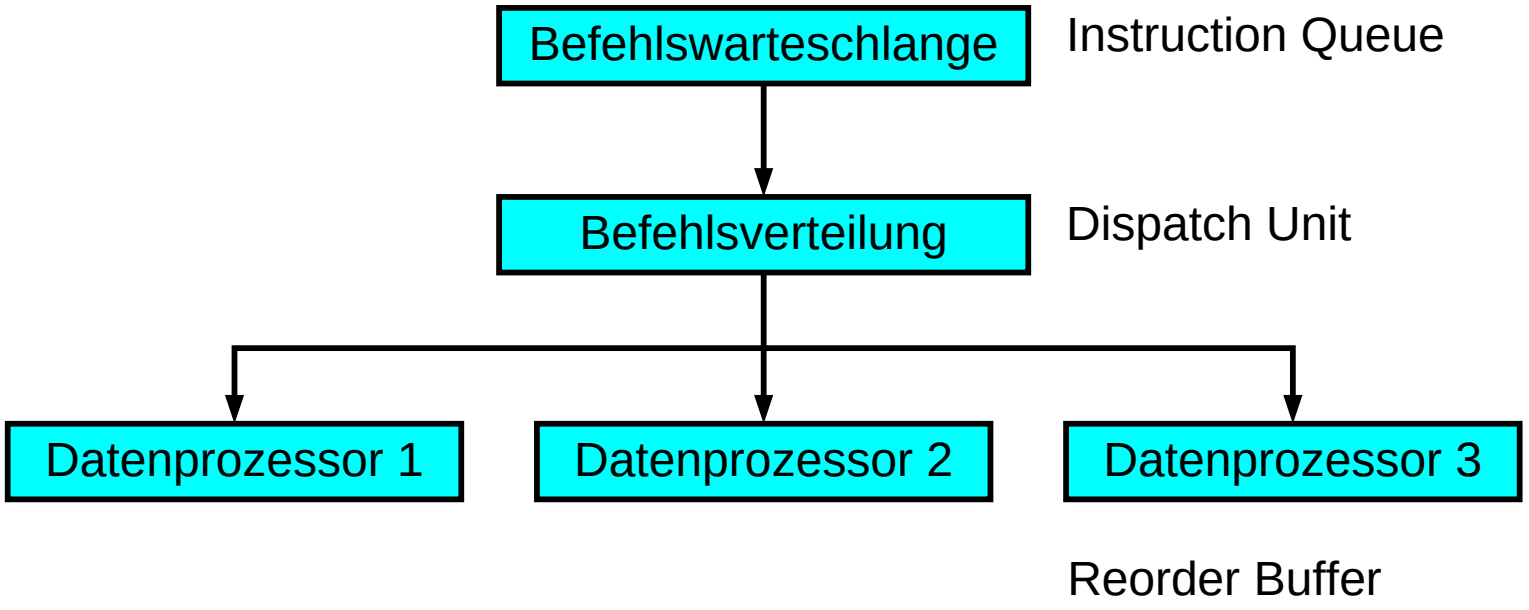
Das Befehlswort hat reservierte Bereiche (Operationsteil und Operandenteil) für jede einzelne Funktionseinheit. Die Operandenteile können sich überschneiden.

VLIW – Befehlswort



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.



9 Zusammenfassung von Neumann Rechner

- Einfache abstrakte Beschreibung eines komplexen Rechners.
- Hardware-optimale, robuste, vollprogrammierbare Rechnerarchitektur.
- Vollständig binärcodierte Darstellung, Speicherung und Verarbeitung.
- Trennung von Kontroll- und Datenfluss, einfacher Befehlsaufbau.
- Optimiertes Speicherkonzept führt zu Engpässen im Befehlszyklus.
- Sequentielle Befehlsabarbeitung führt zu Engpässen bei der Abarbeitung.
- Aufwändiges Steuerwerk zur Steuerung aller Komponenten.
- Grundkonzept mit Erweiterungen und Abwandlungen aller modernen Universalrechner.