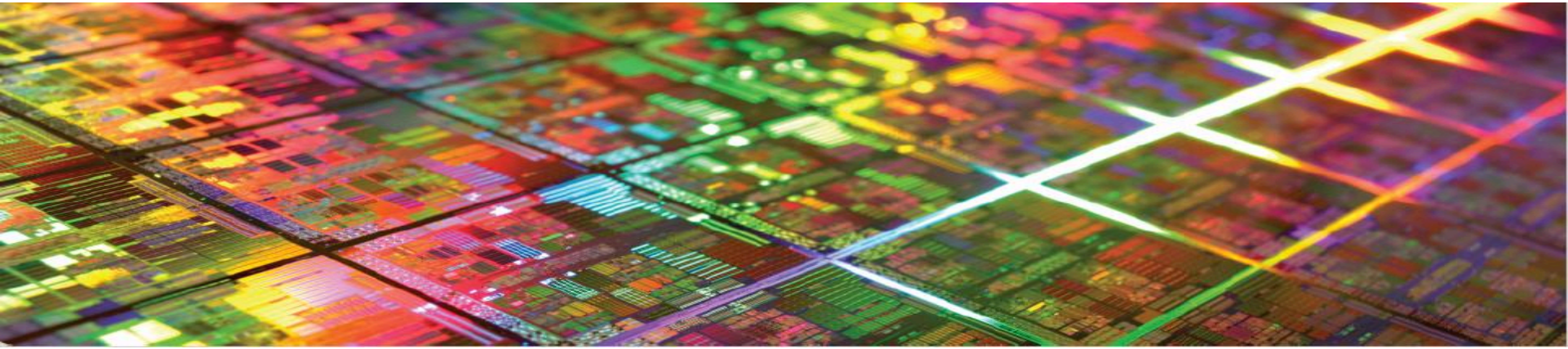
Karlsruher Institut für Technologie

Rechnerorganisation

Prof. Dr. Wolfgang Karl

Vorlesung im Wintersemester 2025/2026 – Foliensatz: RO25-FS06



Kapitel 5
Prozessororganisation

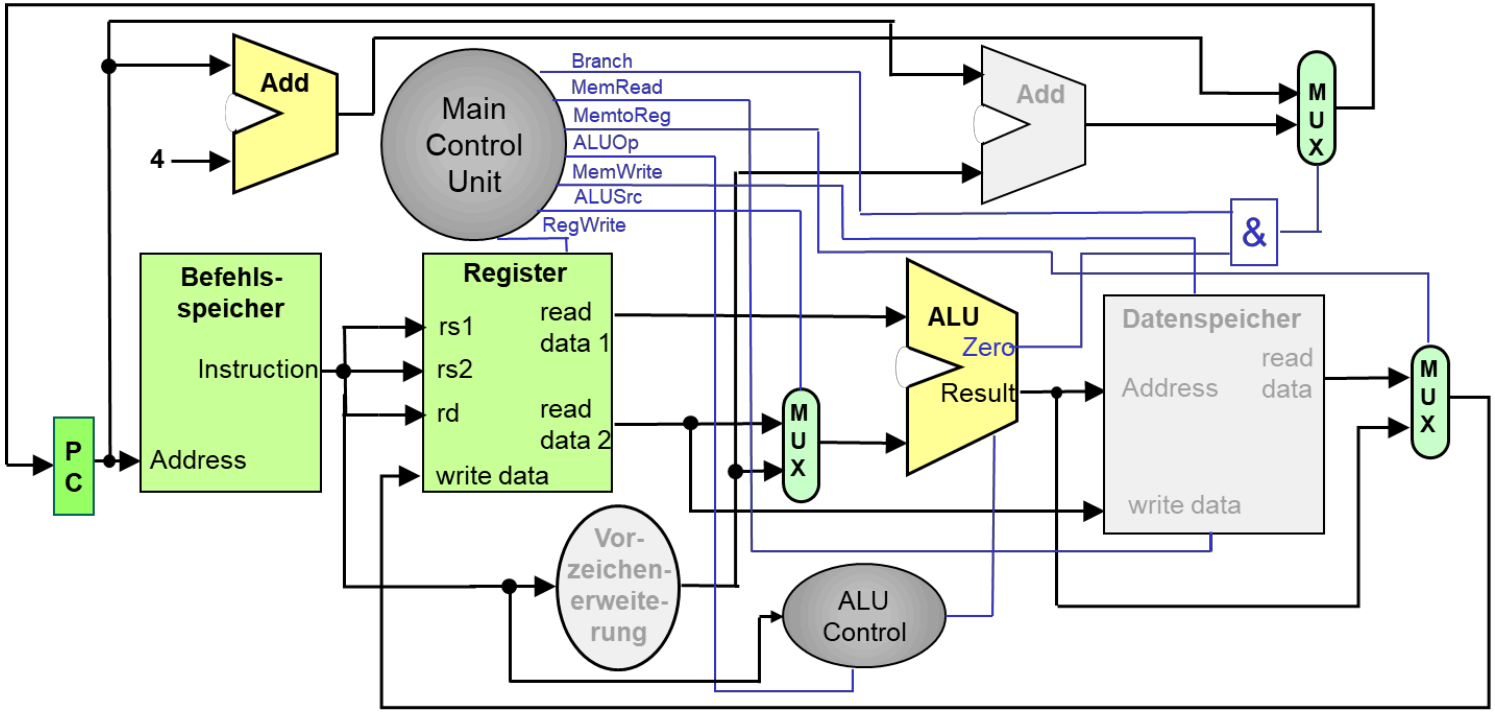
- **Aufbau eines RISC-V Prozessors**
- Implementierung der Steuerung
- Multizyklen-Implementierung

Aufbau des RISC-V-Prozessors

- **Grundlegende Implementierung eines RISC-V Prozessors für eine Teilmenge des Befehlssatzes**
  - **Datenpfad (data path)**
    - Einheiten auf einem Prozessor, die Daten verarbeiten oder speichern;
      - Funktionseinheiten (ALUs), Multiplexer, Registers, Speicherkomponenten, …
    - Ziel: schrittweiser Aufbau eines Datenpfades für RISC-V!

Aufbau des RISC-V-Prozessors

- **Ziel: Datenpfad für RISC-V**



Aufbau des RISC-V-Prozessors

- **Betrachten Teilmenge des RISC-V Befehlssatzes:**
  - Arithmetische und logische Befehle
    - **`add, sub, and, or`**
  - Speicherzugriffsbefehle
    - **`lw` (load word) , `sw` (store word)**
  - Verzweigung (bedingter Sprung)
    - **`beq`**

Aufbau des RISC-V-Prozessors

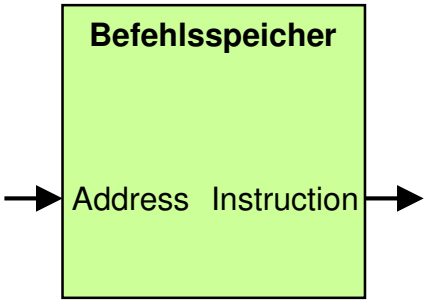
- **Entwurf des Datenpfades**
  - Auswahl der Komponenten für den Datenpfads, die für die Ausführung alle Befehle notwendig sind;
  - Auswahl der Komponenten, die für die Ausführung der Befehle einer Klasse notwendig sind;
  - Betrachtung der Steuersignale für die Komponenten;

Aufbau des RISC-V-Prozessors

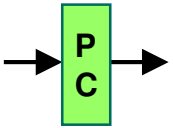
- **Entwurf des Datenpfades**
  - **Schritte, die für alle Befehle gleich sind:**
    - Sende den Inhalt des Befehlszählers (PC) an den Speicher, der das Programm enthält, und hole den Befehl;
    - Adressiere die zu lesenden Register mithilfe der entsprechenden im Befehl stehenden Registernummern und lese deren Inhalt;
    - Alle nachfolgenden Schritte, die zur vollständigen Bearbeitung des Befehls erforderlich sind, hängen von der Befehlsklasse ab.
    - Die Einfachheit und Regelmäßigkeit des RISC-V-Befehlssatzes vereinfachen die Implementierung.

Aufbau des Datenpfades für RISC-V

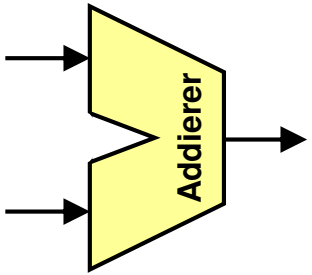
- **Hardware-Komponenten für das Holen der Befehle**
  - Folgende Komponenten sind für die Befehle aller Befehlsklassen notwendig:



Speicher für Befehle



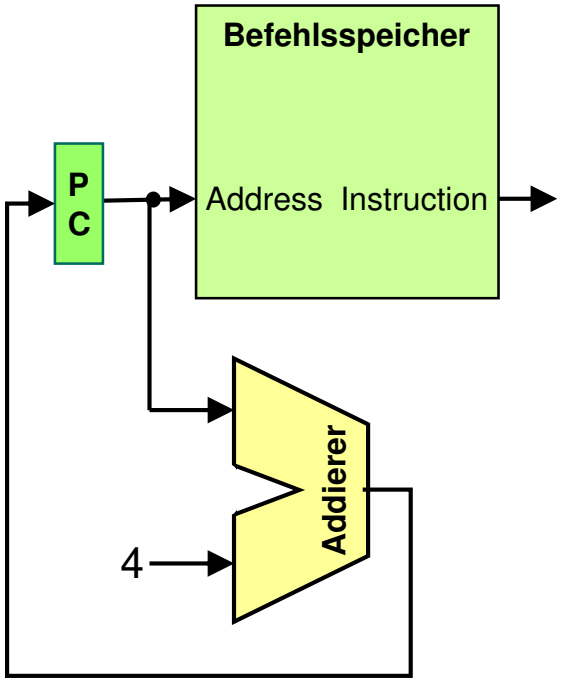
Befehlszähler



Addierer für die Berechnung der Adresse des nächsten Befehls

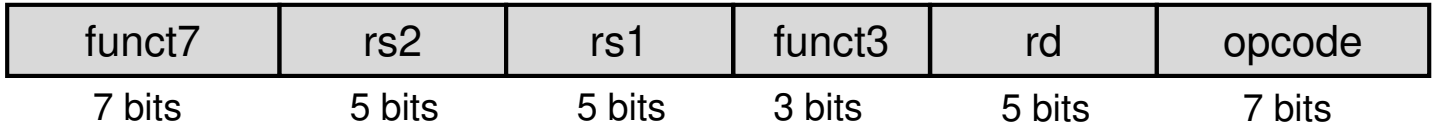
Aufbau des Datenpfades für RISC-V

- **Hardware-Komponenten für das Holen der Befehle**
  - **Aktionen:**
    - Jeder Befehl wird aus dem Speicher geholt.
    - Der Befehlszähler muss um 4 (4 Bytes) inkrementiert werden, um die Adresse des nächsten sequentiell auszuführenden Befehls zu erhalten.



Aufbau des Datenpfades für RISC-V

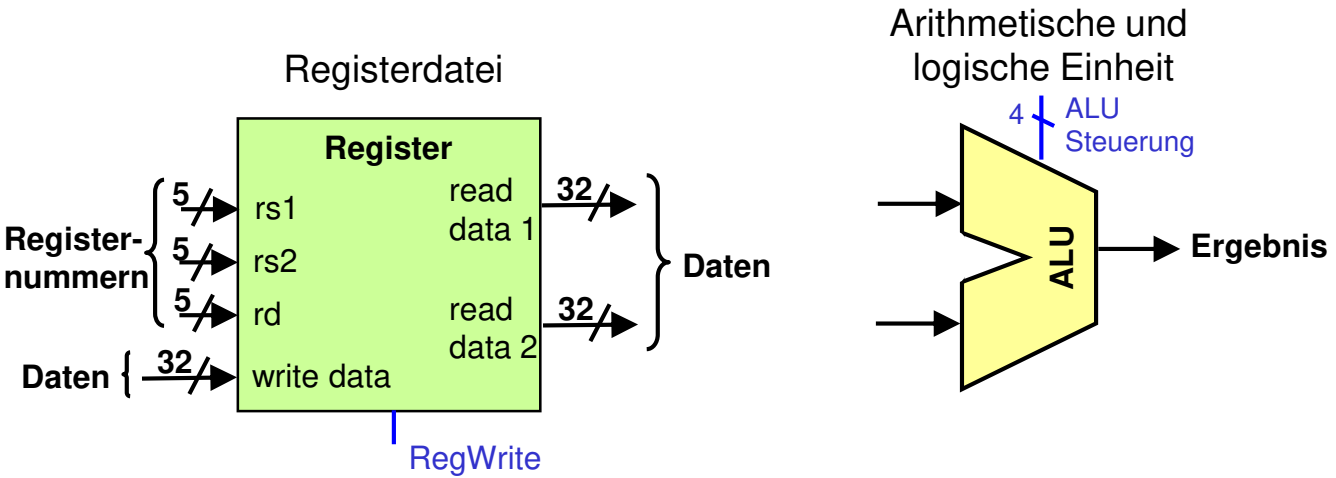
- **Hardware-Komponenten für Befehle vom R-Type**
  - **Arithmetische und logische Befehle**



  - Die im Befehl spezifizierte arithetische oder logische Operation wird auf Operanden ausgeführt, die in den Registern stehen, die im rs1-Feld und rs2-Feld adressiert werden.
  - Das Ergebnis wird in das im rd-Feld adressierte Register geschrieben.

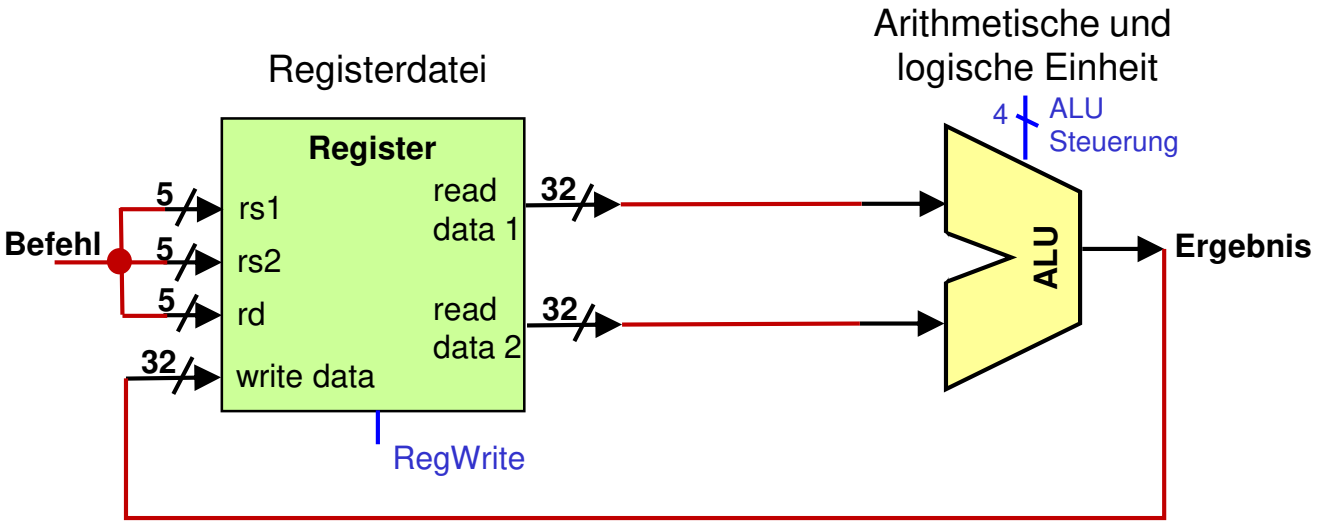
Aufbau des Datenpfades für RISC-V

- **Hardware-Komponenten für Befehle vom R-Type**



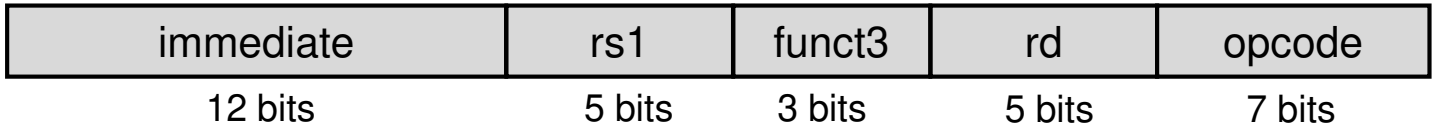
Aufbau des Datenpfades für RISC-V

- **Hardware-Komponenten für Befehle vom R-Type**



Aufbau des Datenpfades für RISC-V

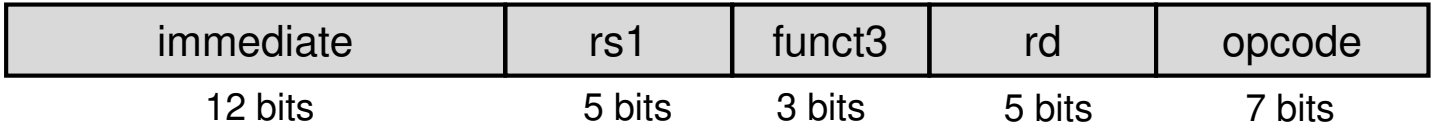
- **Hardware-Komponenten für Befehle vom I-Type und S-Type**
  - **Arithmetische Befehle mit einer Konstanten**



  - Auf den Operanden im Register, das im rs1-Feld adressiert wird, wird eine Konstante addiert. Diese ergibt sich durch Vorzeichenerweiterung des im immediate-Feld stehenden Werts.
  - Das Ergebnis wird in das im rd-Feld adressierte Register geladen.

Aufbau des Datenpfades für RISC-V

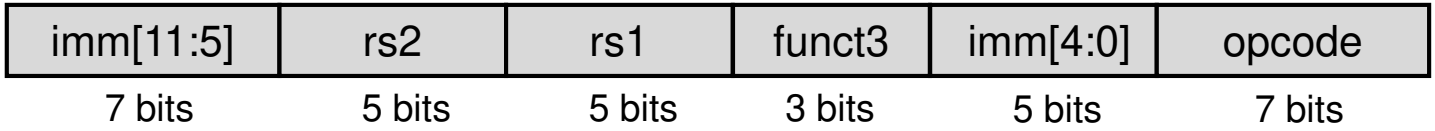
- **Hardware-Komponenten für Befehle vom I-Type und S-Type**
  - **Lade-Befehle**



  - Auf die Basisadresse im Register, das im rs1-Feld adressiert wird, wird eine konstante Abstandsgröße addiert. Diese ergibt sich durch Vorzeichenerweiterung des im immediate-Feld stehenden Werts.
  - Der Inhalt der adressierten Speicherzelle wird in das im rd-Feld adressierte Register geladen.

Aufbau des Datenpfades für RISC-V

- **Hardware-Komponenten für Befehle vom I-Type und S-Type**
  - **Speicher-Befehle**

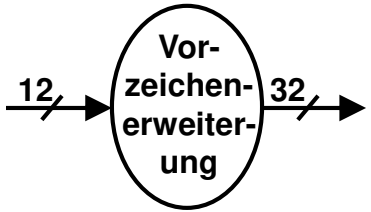


  - Auf die Basisadresse im Register, das im rs1-Feld adressiert wird, wird eine konstante Abstandsgröße addiert. Diese ergibt sich durch Vorzeichenerweiterung des in den immediate-Feldern stehenden Werts.
  - In die adressierte Speicherzelle wird der Wert des Registers gespeichert, das im rs2-Feld adressiert wird

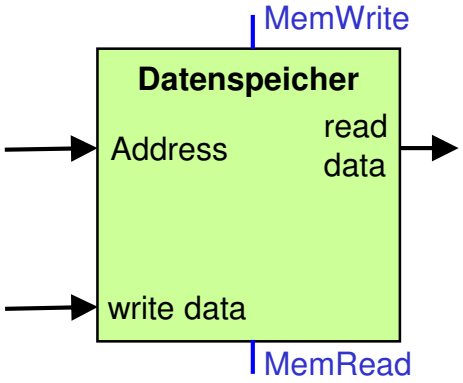
Aufbau des Datenpfades für RISC-V

- **Hardware-Komponenten für Befehle vom I-Type und S-Type**

Einheit für Vorzeichen-erweiterung



Speicher für Daten

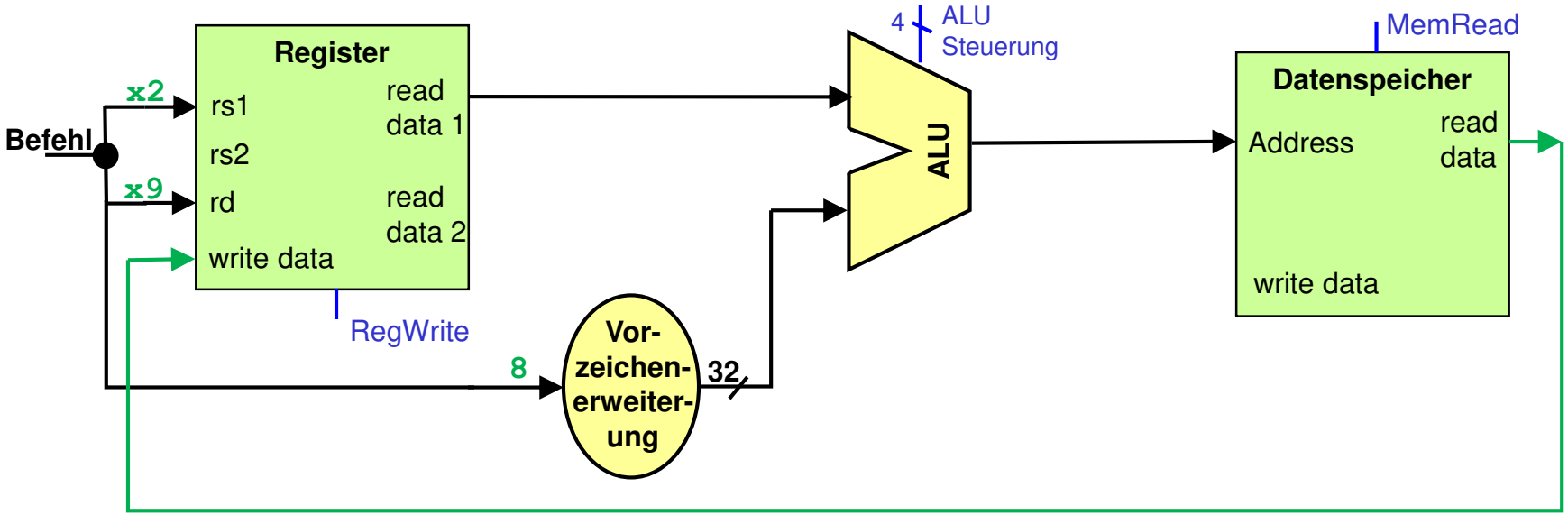


Multiplexer

Aufbau des Datenpfades für RISC-V

- **Hardware-Komponenten für Befehle vom I-Type und S-Type**

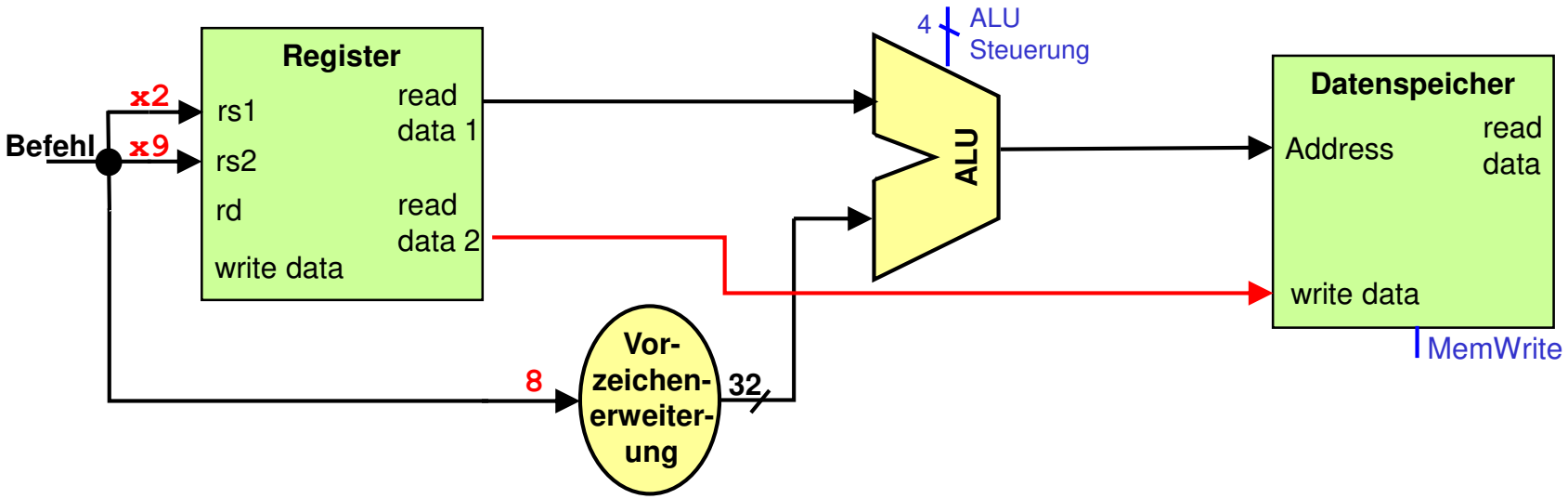
```
lw x9,8(x2)
```



Aufbau des Datenpfades für RISC-V

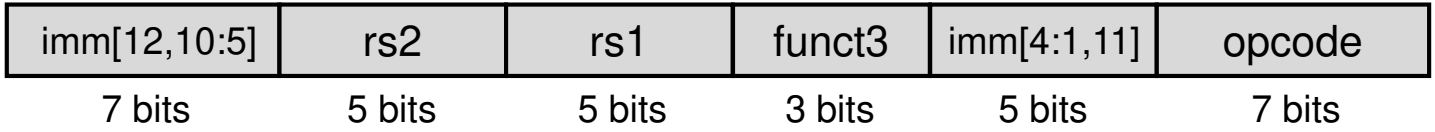
- **Hardware-Komponenten für Befehle vom I-Type und S-Type**

```
sw x9,8(x2)
```



Aufbau des Datenpfades für RISC-V

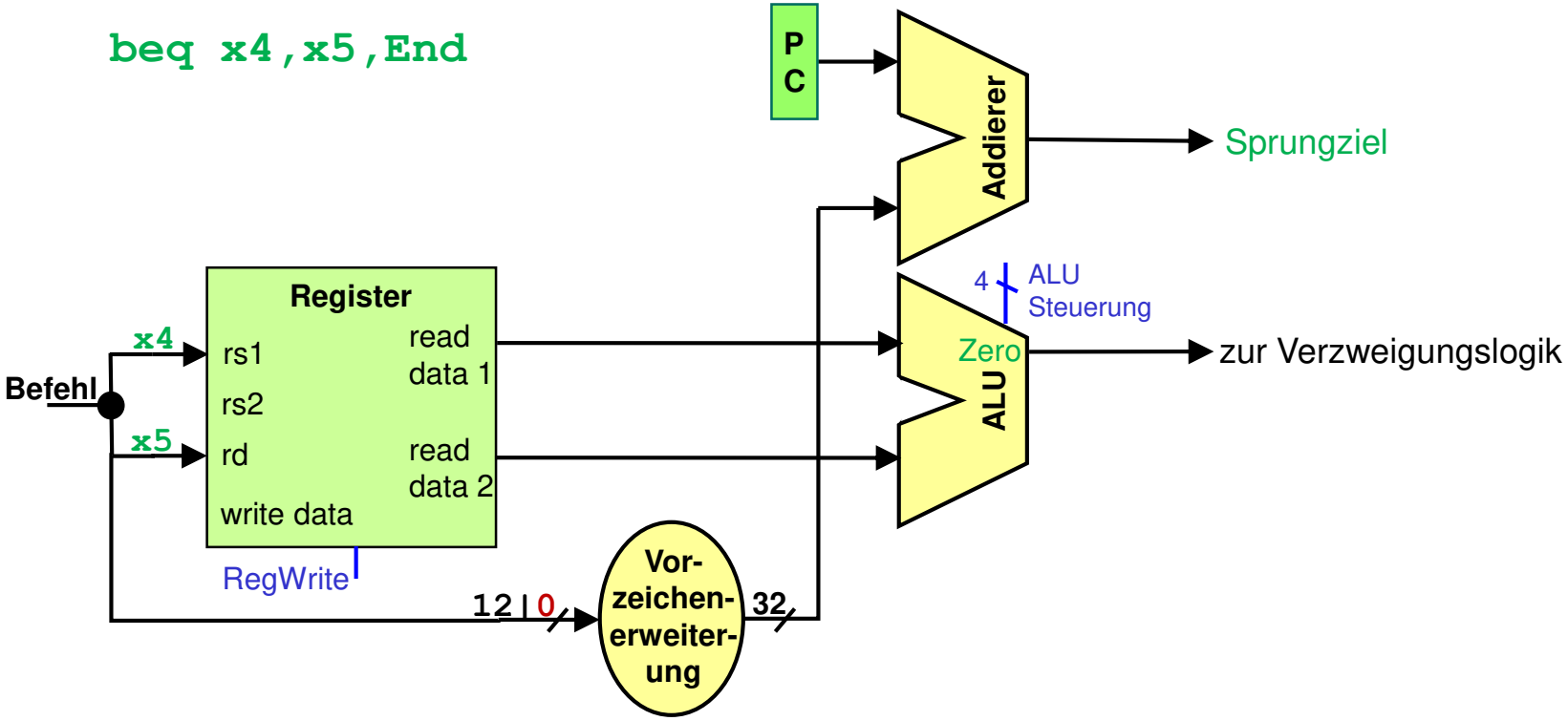
- **Hardware-Komponenten für Befehle vom SB-Type**
  - **Verzweigungen**



  - Auf den Inhalt des PCs wird eine Konstante addiert. Diese ergibt sich aus dem Offset (Felder imm[12,10:5] und imm[4:1,11]), der um 1 nach links geschoben wird, und durch Vorzeichenerweiterung. Das Ergebnis ist das Sprungziel.
  - Die Inhalte der in den rs1- und rs2-Feldern adressierten Register werden verglichen. Falls die Bedingung erfüllt ist, erfolgt der Sprung an das Sprungziel

Aufbau des Datenpfades für RISC-V

Hardware-Komponenten für Befehle vom SB-Type



Aufbau des Datenpfades für RISC-V

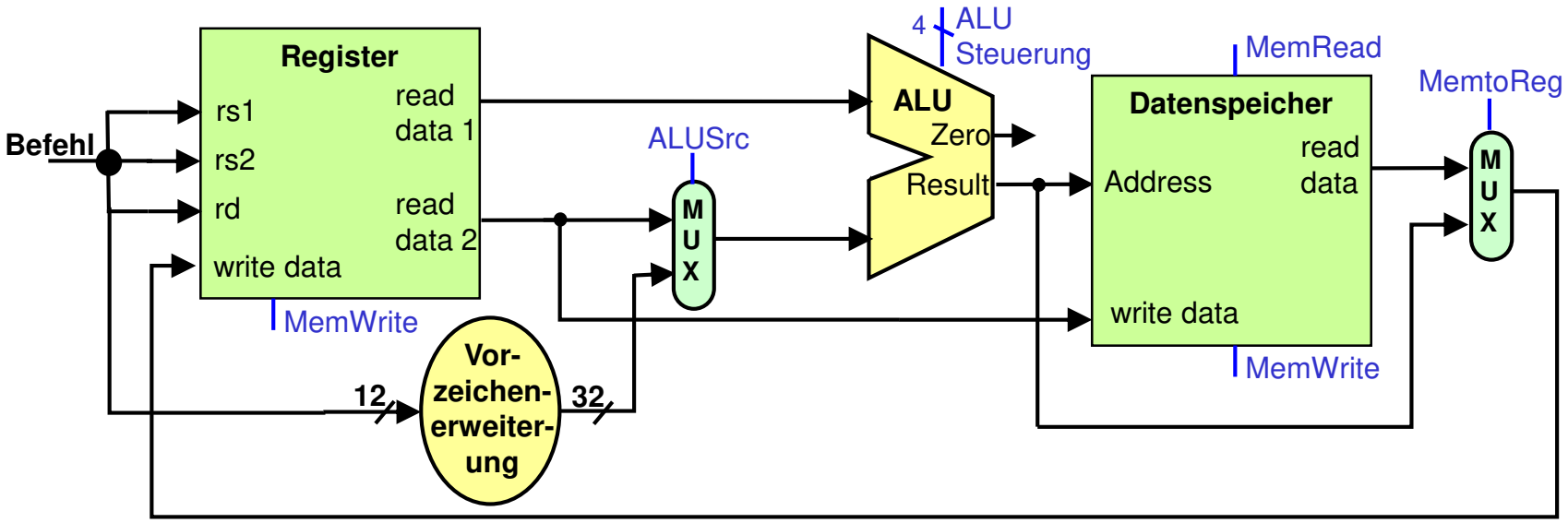
- **Hardware-Komponenten für Befehle vom SB-Type**
  - RISC-V–Befehle sind 4 Byte lang, aber die RISV-V-Architekten wollten auch die Möglichkeit geben, dass RISC-V-Befehle nur 2 Byte lang sind. Aus diesem Grund werden bei Verzweigungen Halbwörter adressiert.
  - Anstelle den in den imm-Feldern stehenden Wert um 1 Bit nach links zu schieben, wird eine 0 konkateniert.

Aufbau des Datenpfades für RISC-V

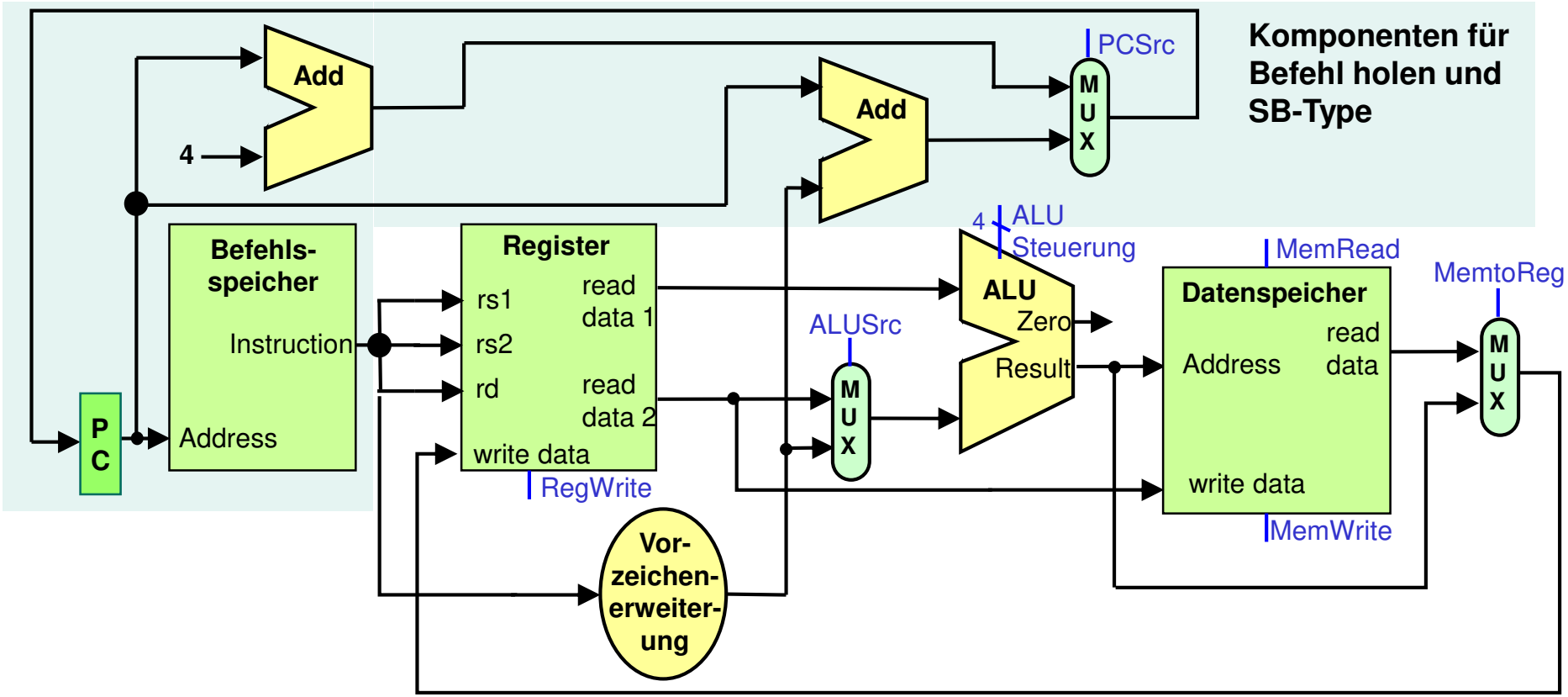
- **Hardware-Komponenten für Befehle vom I-, S- und SB-Type**
  - **Vorzeichenerweiterung**
    - Die Einheit zur Vorzeichenerweiterung muss auswählen:
      - Lade-Befehle: Feld immediate (Bits 31-20) des Befehlswortes
      - Speicher-Befehle: Felder imm[11:5] (Bits 31-25) und imm[4:0] (Bits 11-7) des Befehlswortes
      - Verzweigungen: Felder imm[12,10:5] (Bits 31, 29-25) und imm[4:1,11] (11-8, 30) des Befehlswortes
    - Im Opcode-Feld werden die Steuersignale für einen 3:1 Multiplexer der Einheit zur Auswahl der entsprechenden 12-Bit-Feldes kodiert:
      - Bit 6=0: Speicherzugriffsoperation
      - Bit 6=1: Verzweigung
      - Bit 5=0: Lade-Befehl
      - Bit 5=1: Speicher-Befehl

Aufbau des Datenpfades für RISC-V

- Zusammenschaltung der Hardware-Komponenten
  - Lade-/Speicher-Befehle und R-Type-Befehle



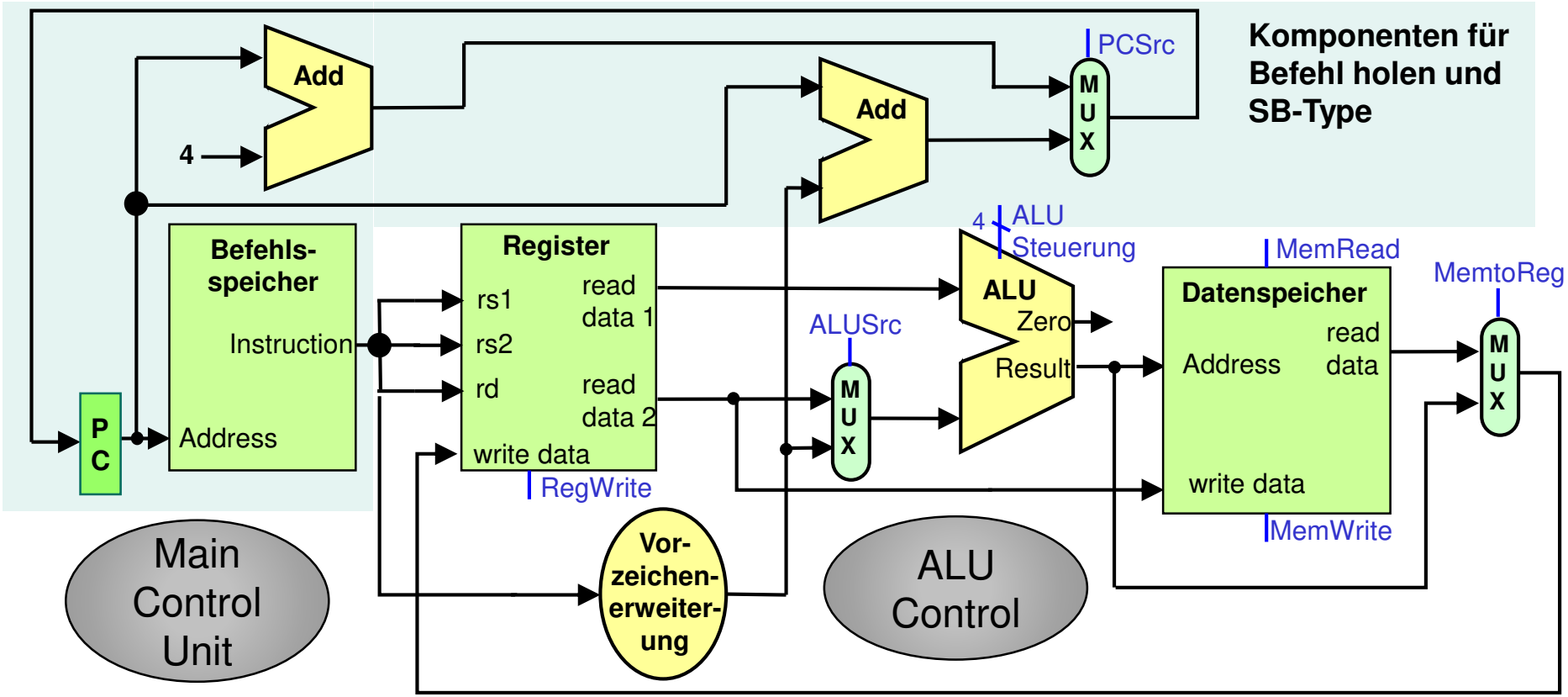
Aufbau des Datenpfades für RISC-V



Kapitel 5
Prozessororganisation

- Aufbau eines RISC-V Prozessors
- **Implementierung der Steuerung**
- Lösung für Multizyklenbefehle

Datenpfad für RISC-V mit Steuerung

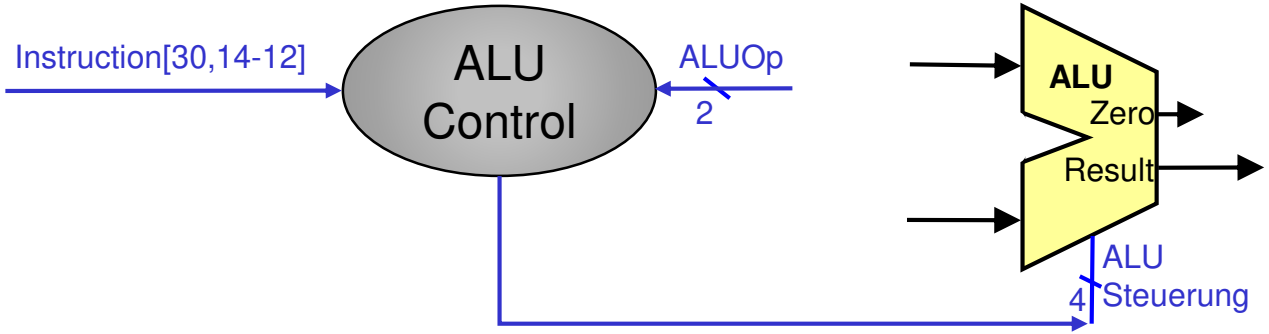


Datenpfad für RISC-V mit Steuerung

- **Implementierung beschränkt sich auf folgende Befehle:**
  - Lade- und Speicher-Befehle
    - **`lw`** (load word)
    - **`sw`** (store word)
  - Verzweigung
    - **`beq`** (branch if equal)
  - Arithmetische und logische Operationen
    - **`add`** (Addition)
    - **`sub`** (Subtraktion)
    - **`and`** (logisches Und)
    - **`or`** (logisches Oder)

Datenpfad für RISC-V mit Steuerung

- **ALU Control**
  - Je nach Befehlsklasse führt die ALU eine von vier Operationen aus:
    - R-Type: ALU führt eine der 4 ALU-Operationen (**`and`**, **`or`**, **`add`**, **`sub`**) aus;
      - Hängt vom Wert im funct7-Feld (Bits 31:25) und dem im funct3-Feld (Bit 14:12) ab;
    - Lade- / Speicher-Befehle: ALU berechnet die Speicheradresse und führt eine Addition (**`add`**) aus;
    - **`beq`**-Befehl: ALU subtrahiert (**`sub`**) die beiden Operanden und prüft, ob das Ergebnis 0 ist;



Datenpfad für RISC-V mit Steuerung

- **ALU Control**
  - Belegung der ALU Steuerung (Steuereingang der ALU)

ALU Steuerung Eingang	Operation
0000	`and`
0001	`or`
0010	`add`
0110	`sub`

Wegen der Beschränkung auf 4 Operationen werden nur 4 der 16 möglichen Eingabekombinationen verwendet.

Datenpfad für RISC-V mit Steuerung

ALU Control

- Belegung der ALU Steuerung (Steuereingang der ALU [3:0])

Opcode des Befehls	ALUOp	Operation	funct7-Feld (Bits 31:25)	funct3-Feld (Bits 14:12)	ALU Operation	ALU Steuerung (Eingang)
`lw`	00	load word	XXXXXXX	XXX	`add`	0010
`sw`	00	store word	XXXXXXX	XXX	`add`	0010
`beq`	01	branch if equal	XXXXXXX	XXX	`sub`	0110
R-Type	10	Addition	0**0**00000	000	`add`	0010
R-Type	10	Subtraktion	0**1**00000	000	`sub`	0110
R-Type	10	AND	0**0**00000	111	`and`	0000
R-Type	10	OR	0**0**00000	110	`or`	0001

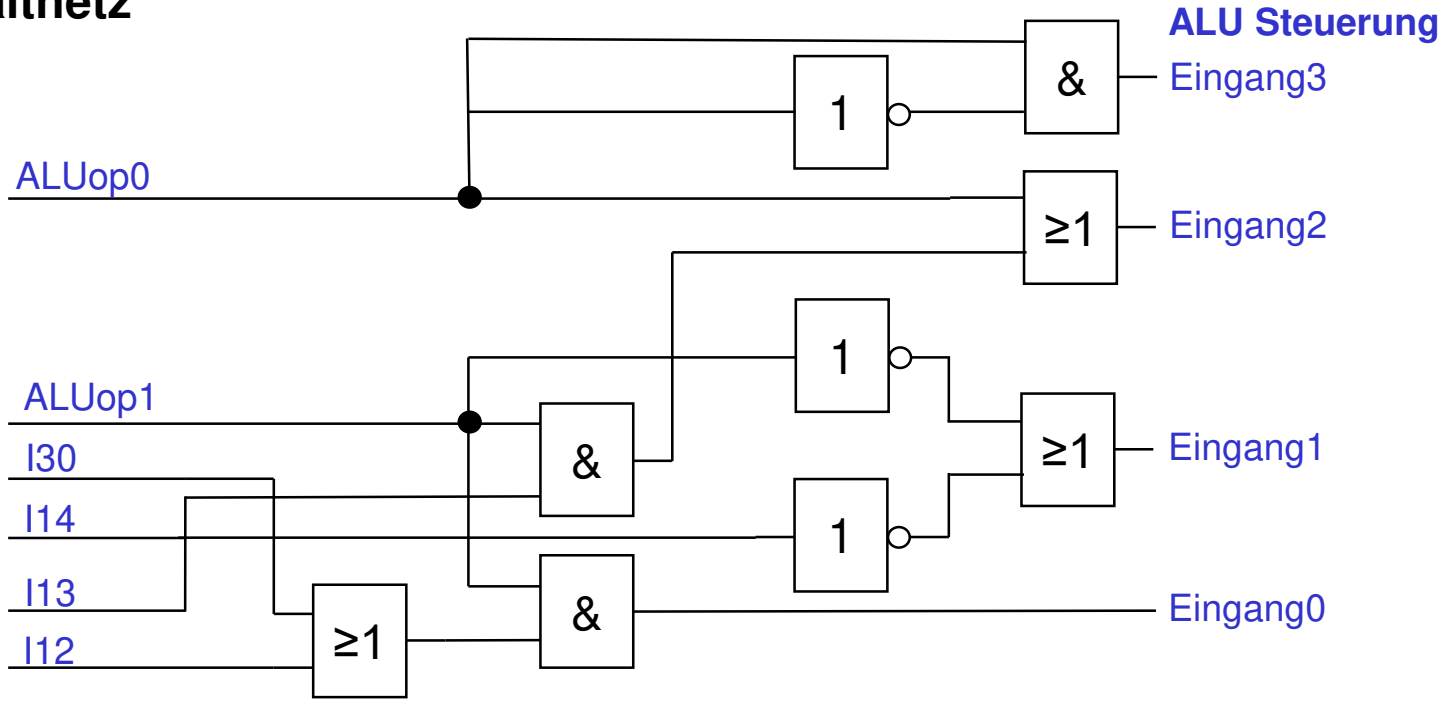
Datenpfad für RISC-V mit Steuerung

- **ALU Control**
  - Belegung der ALU Steuerung (Steuereingang der ALU [3:0])
  - Wahrheitstabelle für ALU Steuerbits (Operation)

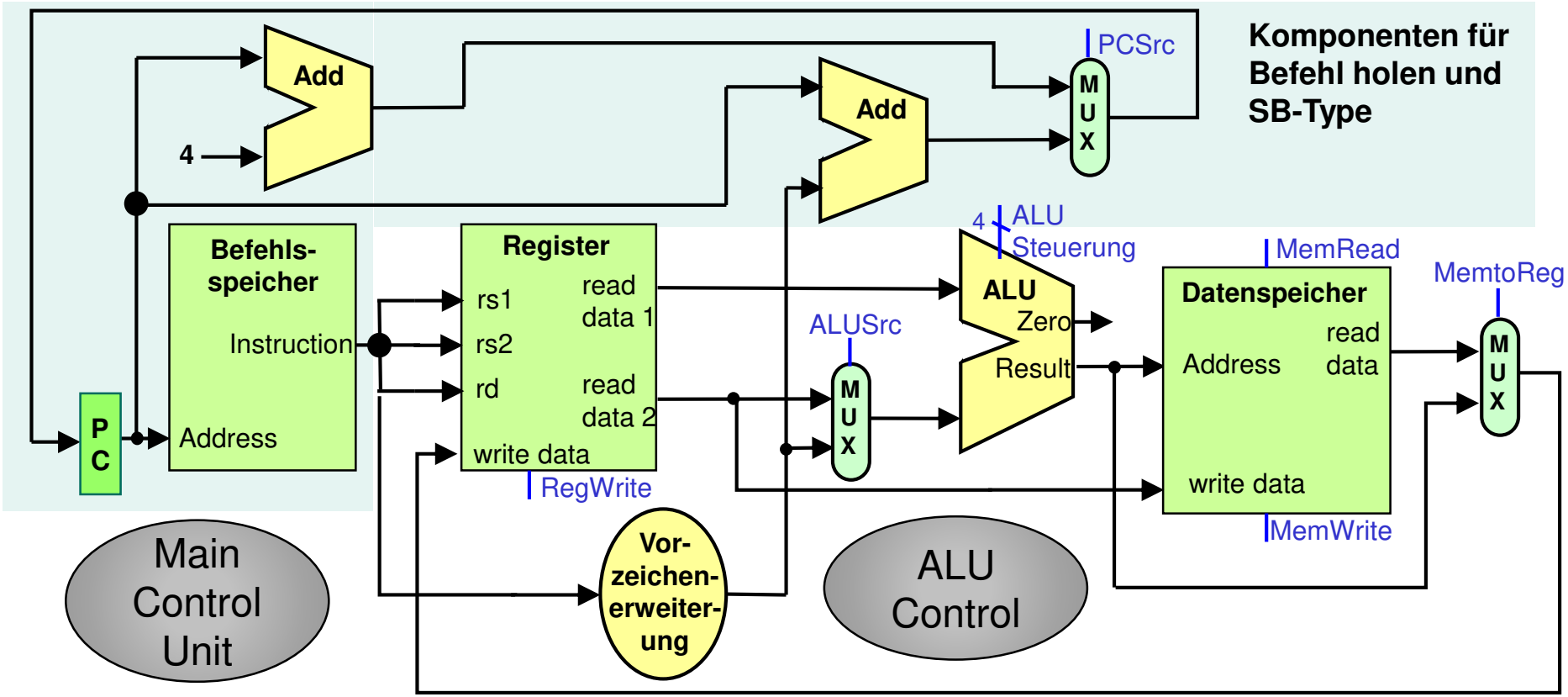
ALUOP		Funct7 Feld							Funct3 Feld			
ALU Op1	ALU Op2	I[31]	**I[30]**	I[29]	I[29]	I[27]	I[26]	I[25]	**I[14]**	**I[13]**	**I[12]**	Opera-tion
0	0	X	**X**	X	X	X	X	X	**X**	**X**	**X**	0010
X	1	X	**X**	X	X	X	X	X	**X**	**X**	**X**	0110
1	X	0	**0**	0	0	0	0	0	**0**	**0**	**0**	0010
1	X	0	**1**	0	0	0	0	0	**0**	**0**	**0**	0110
1	X	0	**0**	0	0	0	0	0	**1**	**1**	**1**	0000
1	X	0	**0**	0	0	0	0	0	**1**	**1**	**0**	0001

Datenpfad für RISC-V mit Steuerung

- **ALU Control**
  - **Schaltnetz**

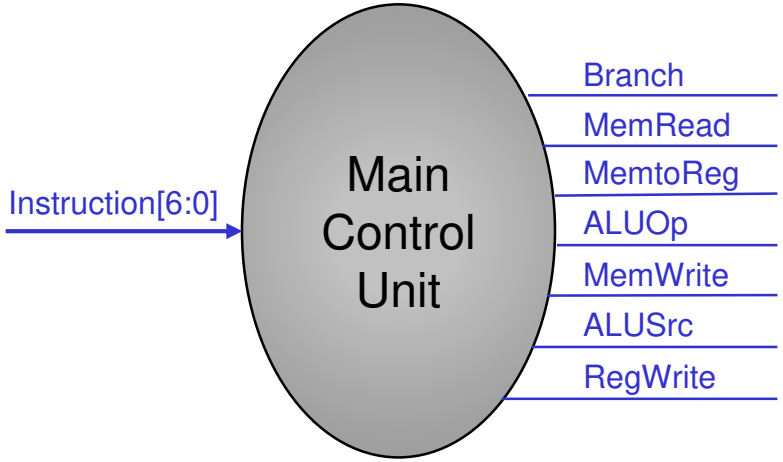


Datenpfad für RISC-V mit Steuerung



Datenpfad für RISC-V mit Steuerung

- **Main Control Unit (Control)**



Datenpfad für RISC-V mit Steuerung

- **Main Control Unit (Control)**
  - **Wirkung der Steuersignale**

Signalname	Wirkung, wenn logisch 0	Wirkung, wenn logisch 1
RegWrite	Keine	Das am **rd-**Eingang adressierte Register wird mit dem Wert am **write data** Eingang beschrieben
ALUSrc	Als zweiter ALU-Operand wird der am **read data 2** Ausgang der Registerdatei ausgewählt	Als zweiter ALU-Operand wird die vorzeichenerweiterte 12-Bit-Konstante des Befehls ausgewählt
PCSrc	Der neue Wert des Befehlszählers (**PC**) wird durch den Ausgangswert des Addierers ersetzt, der PC+4 berechnet	Der neue Wert des Befehlszählers (**PC**) wird durch den Ausgangswert des Addierers ersetzt, der das Sprungziel berechnet

Datenpfad für RISC-V mit Steuerung

- **Main Control Unit (Control)**
  - **Wirkung der Steuersignale**

Signalname	Wirkung, wenn logisch 0	Wirkung, wenn logisch 1
MemRead	Keine	Mit der am **Address**-Eingang des Datenspeichers anliegenden Adresse wird die Speicherzelle ausgewählt, deren Inhalt über den **read data** Ausgang ausgegeben wird
MemWrite	Keine	Mit der am **Address**-Eingang des Datenspeichers anliegenden Adresse wird die Speicherzelle ausgewählt, deren Inhalt mit dem am **write data** Eingang anliegenden Wert überschrieben wird
MemToReg	Der am **write data** Eingang der Registerdatei anzulegende Wert kommt vom ALU Ausgang	Der am **write data** Eingang der Registerdatei anzulegende Wert kommt vom **read data** Ausgang des Datenspeichers

Datenpfad für RISC-V mit Steuerung

- **Main Control Unit (Control)**
  - **Die Werte der Steuersignale werden von dem opcode-Feld bestimmt**

Bitposition	31:25	24:20	19:15	14:12	11:7	6:0
R-Type	funct7	rs2	rs1	funct3	rd	opcode
	7 bits	5 bits	5 bits	3 bits	5 bits	7 bits
I-Type	immediate[11:0]		rs1	funct3	rd	opcode
	12 bits		5 bits	3 bits	5 bits	7 bits
S-Type	imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode
	7 bits	5 bits	5 bits	3 bits	5 bits	7 bits
SB-Type	imm[12,10:5]	rs2	rs1	funct3	imm[4:1,11]	opcode
	7 bits	5 bits	5 bits	3 bits	5 bits	7 bits

Datenpfad für RISC-V mit Steuerung

- **Main Control Unit (Control)**
  - **Die Werte der Steuersignale werden von dem opcode-Feld bestimmt**

Befehl	ALUSrc	Mem-ToReg	Reg-Write	Mem-Read	Mem-Write	Branch	ALU-Op1	ALU-Op2
R-Type	0	0	1	0	0	0	1	0
`lw`	1	1	1	1	0	0	0	0
`sw`	1	X	0	0	1	0	0	0
`beq`	0	X	0	0	0	1	0	1

Datenpfad für RISC-V mit Steuerung

- **Main Control Unit (Control)**
  - **Die Werte der Steuersignale werden von dem opcode-Feld bestimmt**
    - **R-Type**
      - Für die Befehle vom R-Type sind die Registeradressen für die Quelloperanden in den **rs1** und **rs2** Feldern und die Adresse für das Ergebnisregister im **rd** Feld.
      - Es werden die Steuersignale für ALUSrc=0 bestimmt
      - Das Ergebnis wird in ein Register geschrieben und nie in den Datenspeicher, deshalb ist RegWrite =1.
      - Die Steuersignale ALUOp1=1 und ALUOp0=0 zeigen an, dass die Signale für die ALU Steuerung von den funct7- und funct3-Feldern des Befehls kommen.

Datenpfad für RISC-V mit Steuerung

- **Main Control Unit (Control)**
  - **Die Werte der Steuersignale werden von dem opcode-Feld bestimmt**
    - **SB-Type**
      - Wenn das Steuersignal Branch=0 ist, wird der Inhalt des PCs unbedingt um 4 inkrementiert. Ansonsten wird der PC mit dem Sprungziel geladen, falls der Zero-Ausgang der ALU gesetzt ist.
      - ALUOp1=0 und ALUOp0=1 (Steuersignale für die Subtraktion)

Datenpfad für RISC-V mit Steuerung

- **Main Control Unit (Control)**
  - **Die Werte der Steuersignale werden von dem opcode-Feld bestimmt**
    - **Speicherzugriff `lw, sw`**
      - Die Steuersignale ALUSrc und ALUOp1 und ALUOp0 werden gesetzt, um die Adressrechnungen durchführen zu können.
      - Die Steuersignale MemRead und MemWrite werden gesetzt, um die jeweiligen Zugriffe auf den Datenspeicher durchführen zu können.
      - Das Steuersignal RegWrite wird bei dem Befehl **`lw`** gesetzt, um den aus dem Speicher geholten Wert in das Register zu laden.
      - Das Steuersignal MemToReg ist irrelevant, wenn das Steuersignal RegWrite=0 ist, da kein Register beschrieben werden kann. Aus diesem Grund steht in der Tabelle X (für don‘t care)

Datenpfad für RISC-V mit Steuerung

- **Main Control Unit (Control)**
  - **Die Werte der Steuersignale werden von dem opcode-Feld bestimmt**
    - **Speicherzugriff `lw, sw`**
      - ALUSrc = 1
      - ALUOp1 = 0 und ALUOp0 = 0

      } Durchführung der Adressrechnung

      - MemRead = 1
      - MemWrite = 1

      } Durchführung der Speicherzugriffe

      - RegWrite = 1 für den Befehl **`lw`**

      } Laden des aus dem Speicher geholten Wertes in das Register

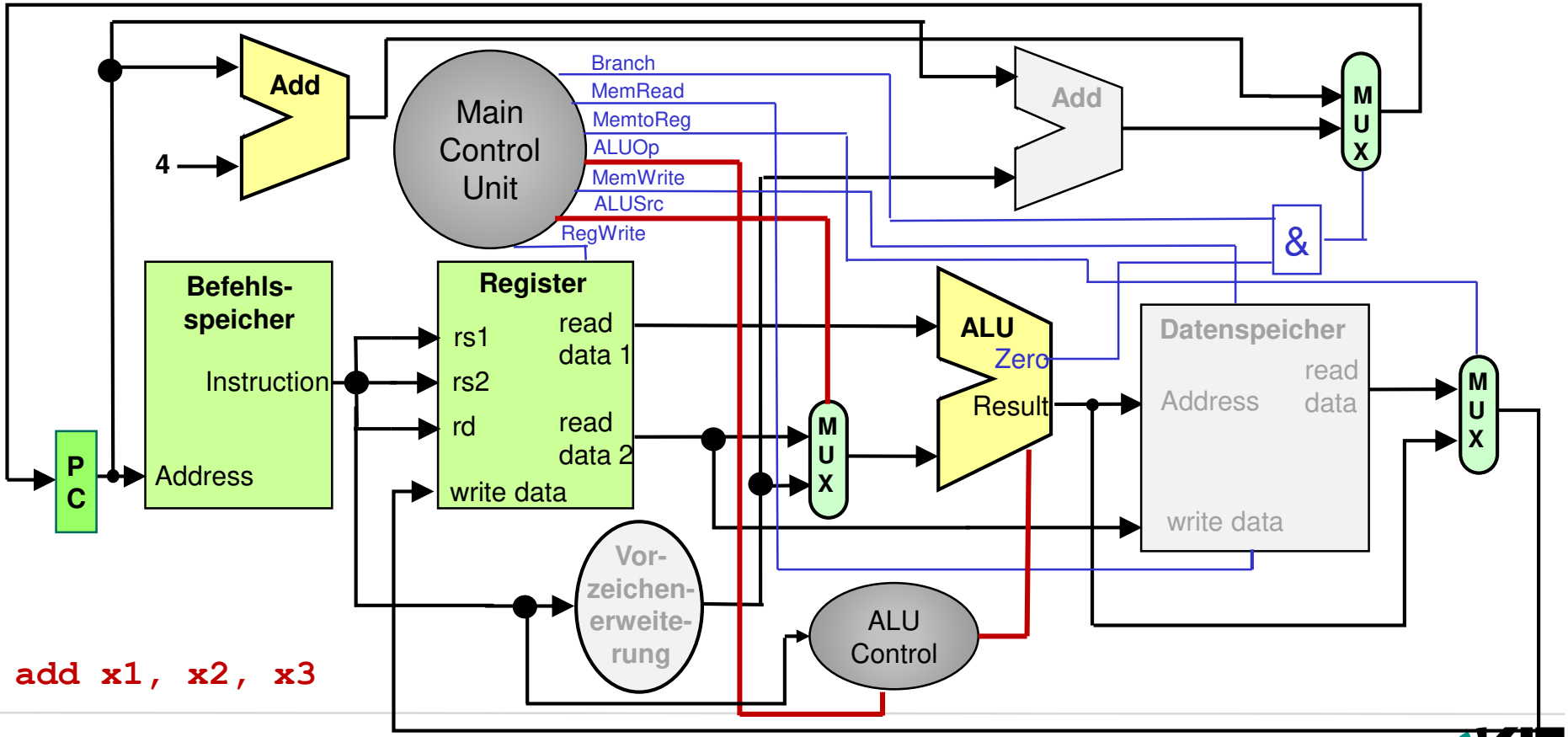
      - MemToReg ist irrelevant, wenn
      - RegWrite=0

      } Kein Register wird geschrieben.

Datenpfad für RISC-V mit Steuerung

- **Arbeitsweise des Datenpfads für Befehle vom R-Type**
  - Beispiel: **`add x1, x2, x3`**
  - **Ausführungsschritte:**
    1. Holen des Befehls und Inkrementieren des Inhalts des PCs um 4
    2. Lesen der Register **`x2`** und **`x3`**. Die Main Control Unit setzt die Steuersignale.
    3. Die ALU führt die Operation (**`add`**) auf den von der Registerdatei an den ALU-Eingängen bereitgestellten Operanden aus.
    4. Das Ergebnis wird in das Zielregister **`x1`** geschrieben

Datenpfad für RISC-V mit Steuerung

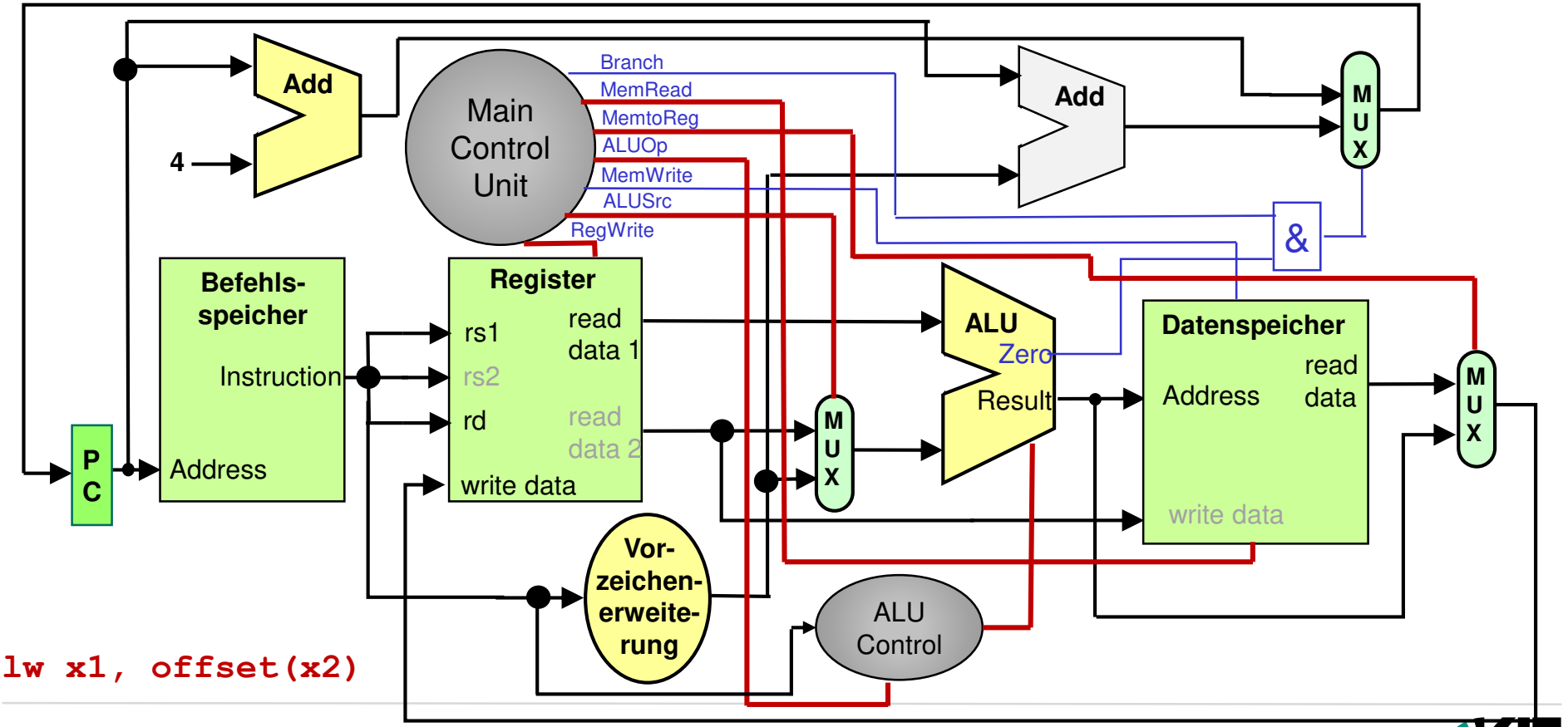


Datenpfad für RISC-V mit Steuerung

- **Arbeitsweise des Datenpfads für Befehle vom R-Type**
  - Beispiel: `lw x1, offset (x2)`

  - **Ausführungsschritte:**
    1. Holen des Befehls und Inkrementieren des Inhalts des PCs um 4
    2. Der Inhalt des Registers **`x2`** wird gelesen.
    3. Die ALU berechnet die Summer des aus dem Register **`x2`** gelesenen Wertes und der vorzeichenerweiterten 12-Bit-Konstante im Befehl (**`offset`**).
    4. Das Ergebnis der Berechnung in der ALU ist die Adresse für den Datenspeicher.
    5. Die Daten werden aus dem Speicher in die Registerdatei geschrieben (**`x1`**)

Datenpfad für RISC-V mit Steuerung

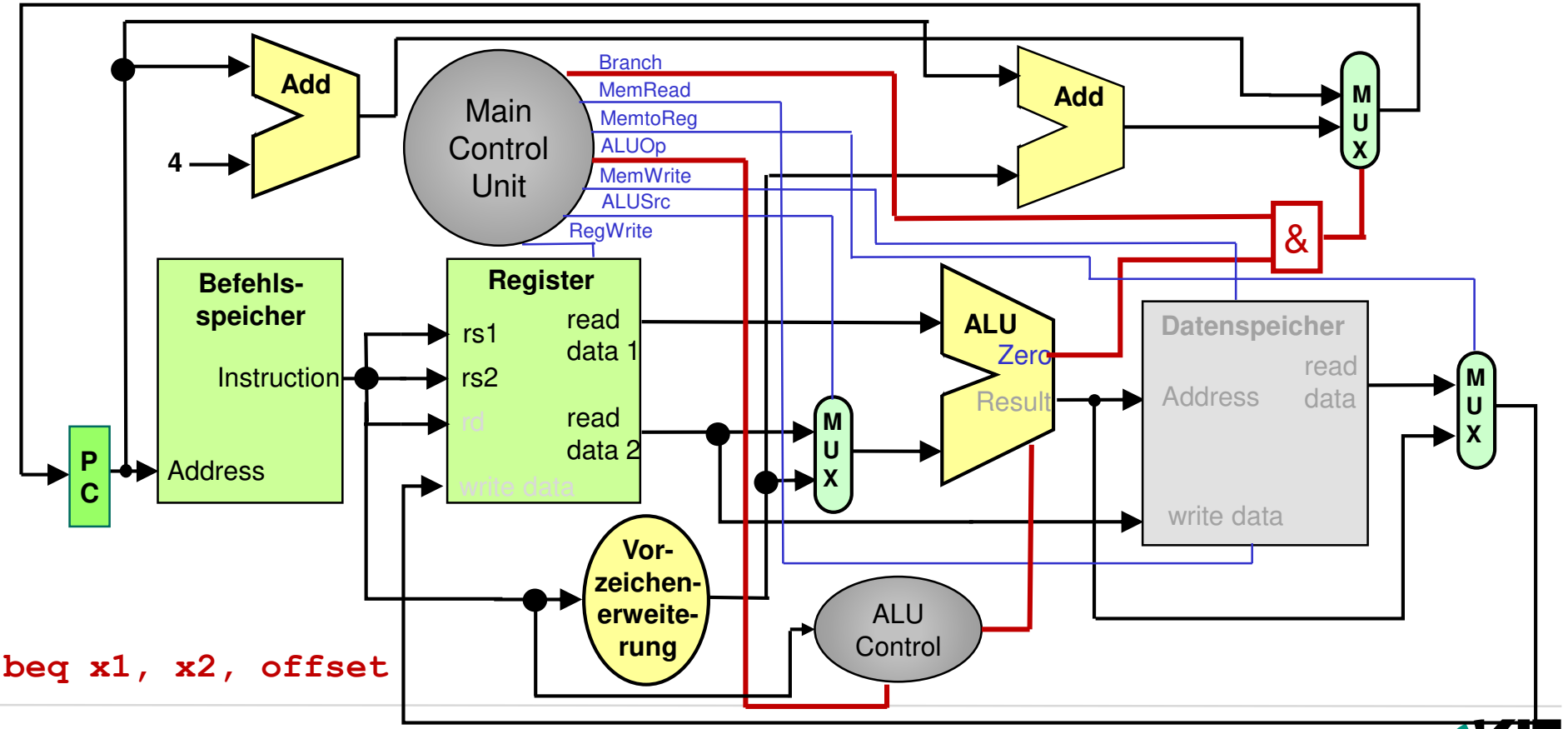



```
lw x1, offset(x2)
```

Datenpfad für RISC-V mit Steuerung

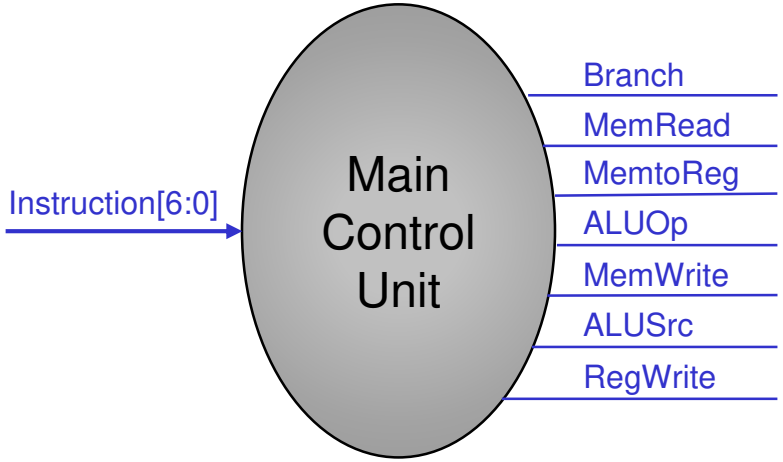
- **Arbeitsweise des Datenpfads für Befehle vom R-Type**
  - Beispiel: **`beq x1, x2, offset`**
  - **Ausführungsschritte:**
    1. Holen des Befehls und Inkrementieren des Inhalts des PCs um 4
    2. Die Inhalte der Registers **`x1`** und **`x2`** werden aus der Registerdatei gelesen.
    3. Die ALU bildet die Differenz von den aus der Registerdatei gelesenen Werte. Der im PC stehende Wert wird auf die vorzeichenerweiterte konstante 12-Bit-Abstandsgröße (**`offset`**) addiert Das Ergebnis bildet die Sprungadresse.
    4. Der Wert des Statussignals Zero wird verwendet, um zu entscheiden, welcher Addierer den Wert für den PC liefert.

Datenpfad für RISC-V mit Steuerung



Datenpfad für RISC-V mit Steuerung

- **Main Control Unit (Control)**



Datenpfad für RISC-V mit Steuerung

- **Main Control Unit (Control)**
  - **Wahrheitstabelle 1**

	Eingabe						
Befehl	**I[6]**	**I[5]**	**I[4]**	**I[3]**	**I[2]**	**I[1]**	**I[0]**
R-Type	0	1	1	0	0	1	1
`lw`	0	0	0	0	0	1	1
`sw`	0	1	0	0	0	1	1
`beq`	1	1	0	0	0	1	1

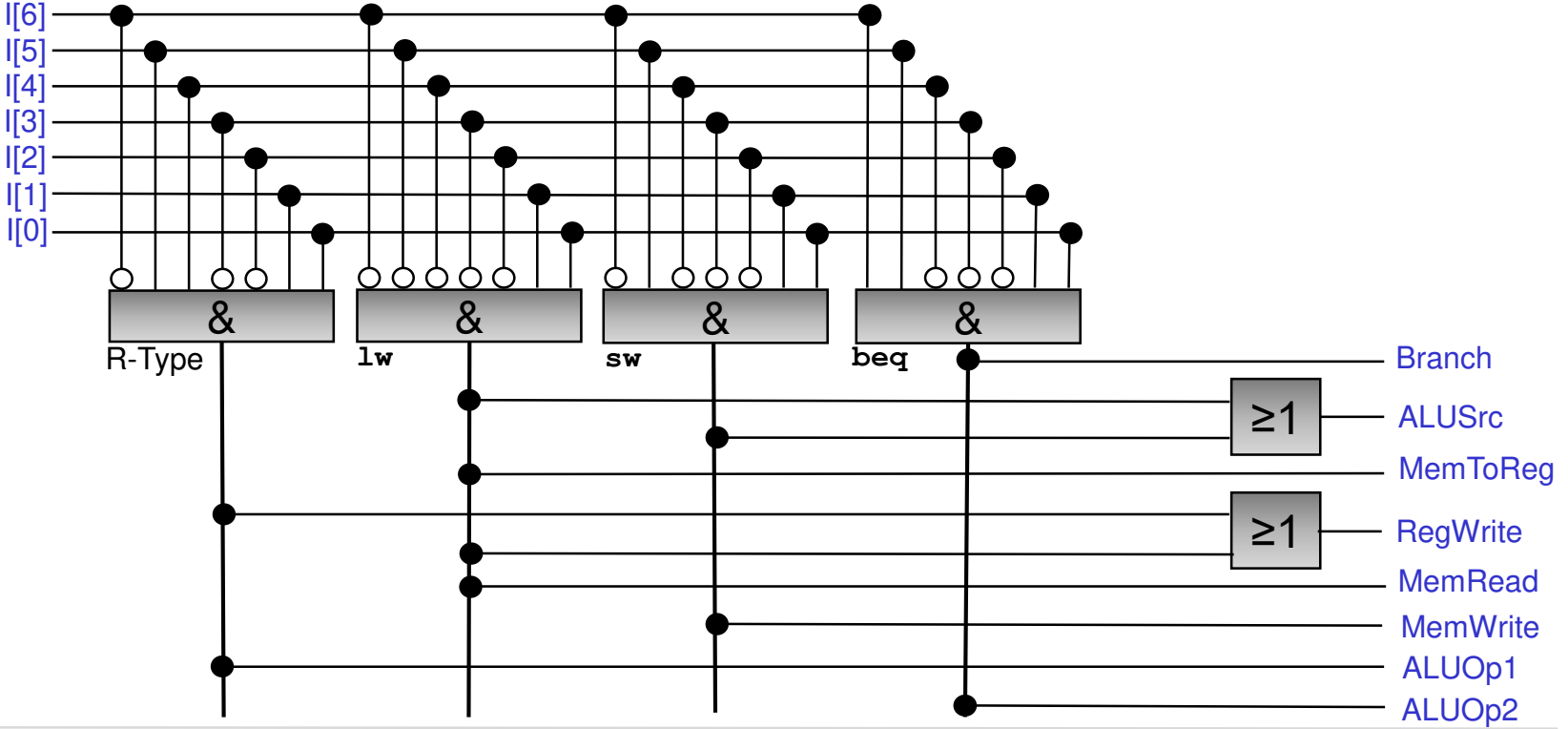
Datenpfad für RISC-V mit Steuerung

- **Main Control Unit (Control)**
  - **Wahrheitstabelle 2**

	Ausgabe							
Befehl	**ALUSrc**	**Mem-ToReg**	**Reg-Write**	**Mem-Read**	**Mem-Write**	**Branch**	**ALU-Op1**	**ALU-Op2**
R-Type	0	0	1	0	0	0	1	0
`lw`	1	1	1	1	0	0	0	0
`sw`	1	X	0	0	1	0	0	0
`beq`	0	X	0	0	0	1	0	1

Datenpfad für RISC-V mit Steuerung

- **Main Control Unit (Control): Implementierung als PLA**



Kapitel 5
Prozessororganisation

- Aufbau eines RISC-V Prozessors
- Implementierung der Steuerung
- **Multizyklen-Implementierung**

Datenpfad für RISC-V mit Steuerung

- **Einzyklen-Implementierung**
  - Beispiel: **`add x1, x2, x3`**
  - **Ausführungsschritte:**
    1. Holen des Befehls und Inkrementieren des Inhalts des PCs um 4
    2. Lesen der Register **`x2`** und **`x3`**. Die Main Control Unit setzt die Steuersignale.
    3. Die ALU führt die Operation (**`add`**) auf den von der Registerdatei an den ALU-Eingängen bereitgestellten Operanden aus.
    4. Das Ergebnis wird in das Zielregister **`x1`** geschrieben
  - Ausführung der Schritte für einen Befehl in einem Taktzyklus

Datenpfad für RISC-V mit Steuerung

- **Einzyklen-Implementierung**
  - **Ausführung der Schritte für einen Befehl in einem Taktzyklus**
    - Der Taktzyklus dauert für jede Instruktion gleich lang
    - Die Dauer eines Taktzyklus richtet sich nach dem längsten Pfad (**kritischer Pfad**)
      - Beispiel: **lw**-Befehl greift auf Speicher zu
    - Die Rechenleistung für die Ausführung eines Programms ist wahrscheinlich sehr schlecht
      - Zur Erinnerung: Gleichung für die Prozessorzeit

$$\mathbf{CPU - Zeit = IC * CPI * Zyklendauer}$$

    - **Gibt es einen besseren Ansatz?**

Datenpfad für RISC-V mit Steuerung

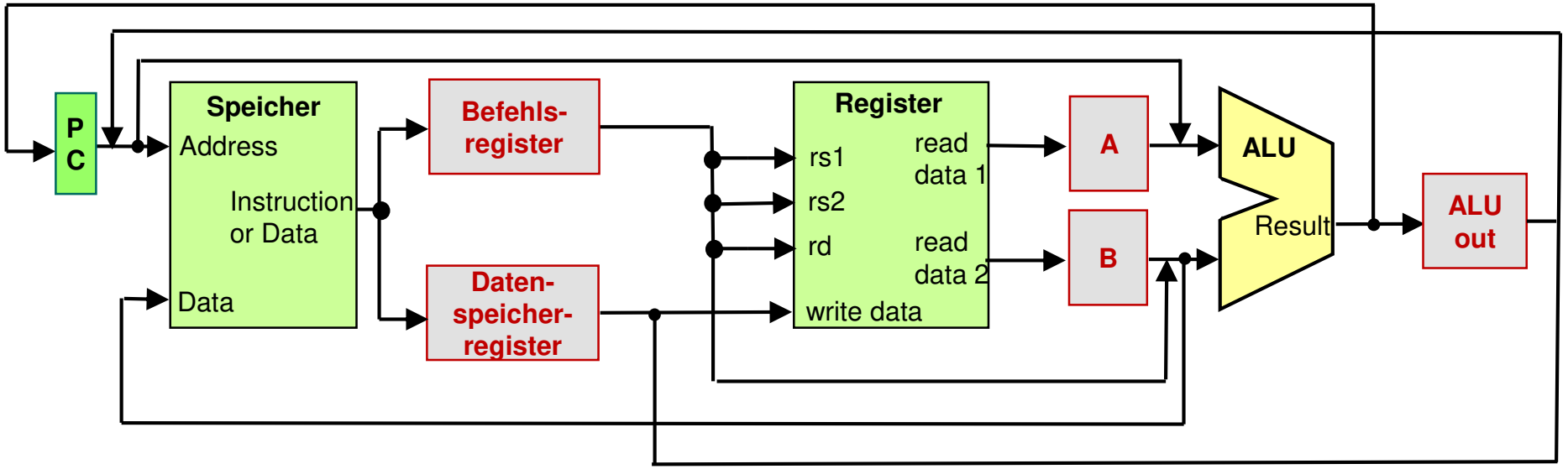
- **Multizyklen-Implementierung**
  - **Die Ausführung eines Arbeitsschrittes für einen Befehl dauert einen Taktzyklus**
    - Die Abarbeitung eines Befehls dauert mehrere Taktzyklen.
    - Eine Hardware-Komponente kann in verschiedenen Taktzyklen von mehr als einem Befehl genutzt werden.
      - Mögliche Reduzierung der notwendigen Hardware-Komponenten
    - **Ermöglicht das Pipelining des Maschinenbefehlszyklus!**

Datenpfad für RISC-V mit Steuerung

- **Multizyklen-Implementierung**
  - **Änderungen im Datenpfad**
    - Eine Speichereinheit für Befehle und Daten.
    - Eine ALU anstelle einer ALU und 2 Addierer.
    - Mehrere Register, in denen der jeweilige Wert am Ausgang einer funktionalen Einheit festgehalten wird, damit der Wert im nächsten Takt verwendet werden kann.
      - Daten, die in einem späteren Taktzyklus benötigt werden, werden in einem für den Programmierer sichtbaren Speicherelement (Speichereinheit, PC Register) gehalten.
    - Befehlsregister (IR)
    - Datenspeicherregister (MR)
    - Register A und B speichern die an den Registerausgängen read data 1 und read data 2 anliegenden Werte
    - Das Register ALUOut speichert den am Ausgang der ALU anliegenden Wert

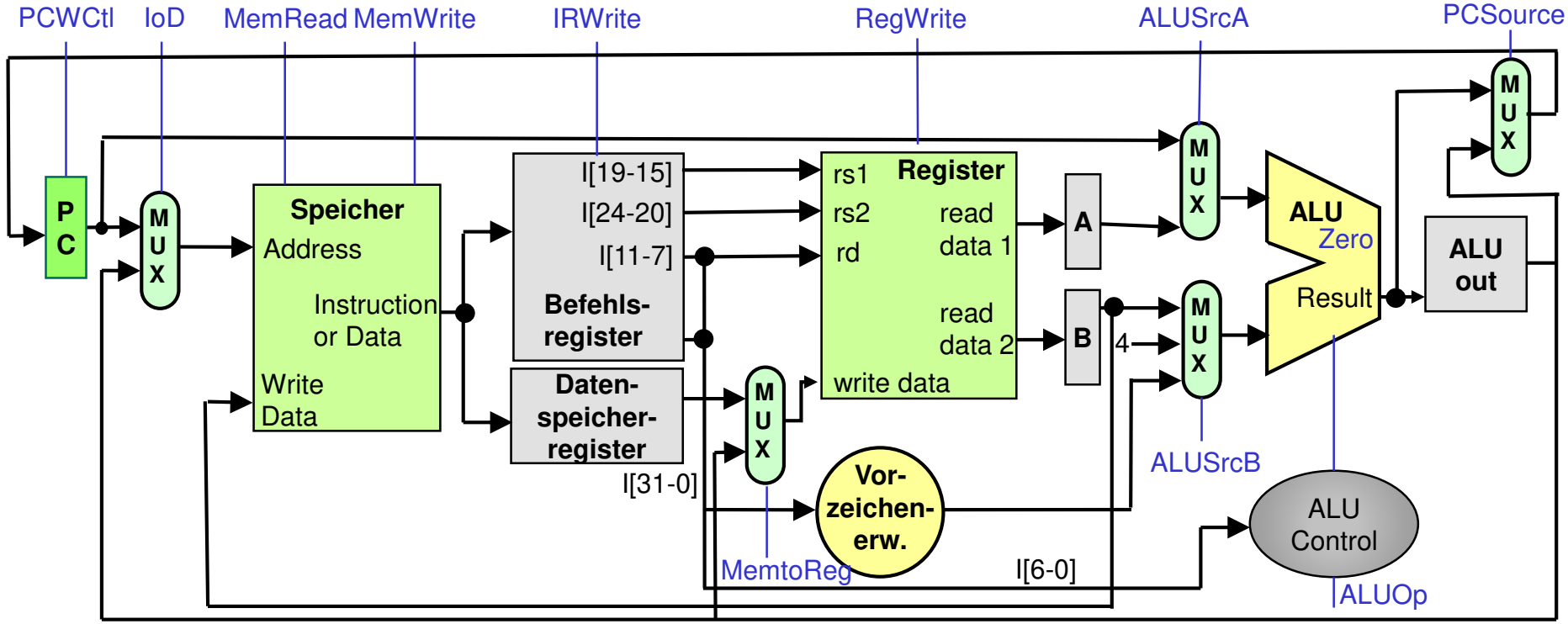
Datenpfad für RISC-V mit Steuerung

- **Multizyklen-Implementierung**
  - **Änderungen im Datenpfad: vereinfachte Darstellung**



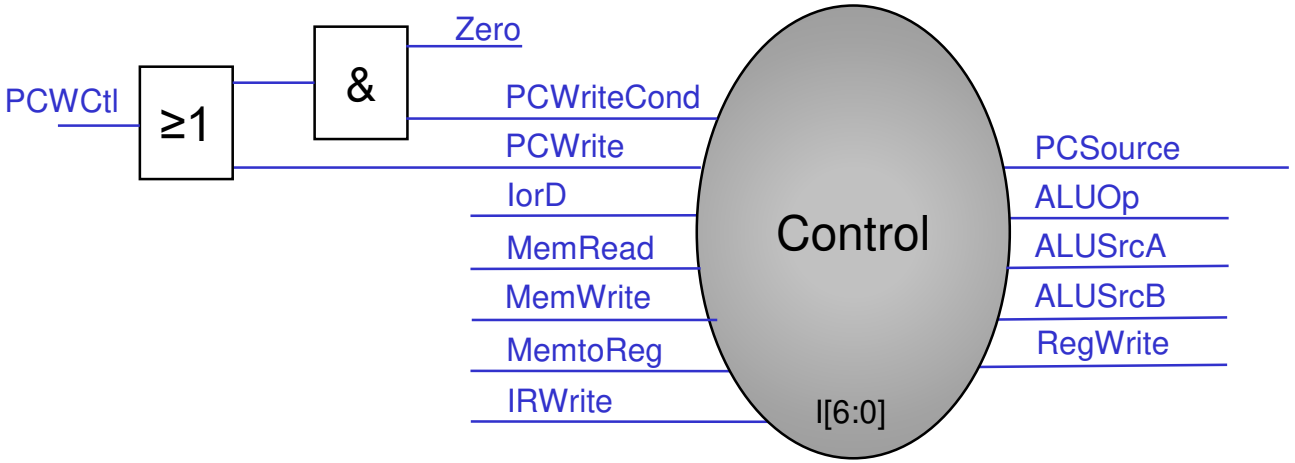
Datenpfad für RISC-V mit Steuerung

Multizyklen-Implementierung



Datenpfad für RISC-V mit Steuerung

- **Main Control Unit (Control)**



Datenpfad für RISC-V mit Steuerung

Main Control Unit (Control)

Wirkung der 1-Bit-Steuersignale

Signalname	Wirkung, wenn logisch 0	Wirkung, wenn logisch 1
RegWrite	Keine	Das am **rd-**Eingang adressierte Register wird mit dem Wert am **write data** Eingang beschrieben
ALUSrcA	Als erster ALU-Operand wird der Inhalt des PCs ausgewählt	Der erste ALU-Operand kommt A-Register
MemRead	Keine	Mit der am **Address**-Eingang des Datenspeichers anliegenden Adresse wird die Speicherzelle ausgewählt, deren Inhalt über den **read data** Ausgang ausgegeben wird
MemWrite	Keine	Mit der am **Address**-Eingang des Datenspeichers anliegenden Adresse wird die Speicherzelle ausgewählt, deren Inhalt mit dem am **write data** Eingang anliegenden Wert überschrieben wird

Datenpfad für RISC-V mit Steuerung

- **Main Control Unit (Control)**
  - **Wirkung der 1-Bit-Steuersignale**

Signalname	Wirkung, wenn logisch 0	Wirkung, wenn logisch 1
MemToReg	Der am **write data** Eingang der Registerdatei anzulegende Wert kommt von ALUOut	Der am **write data** Eingang der Registerdatei anzulegende Wert kommt vom **Datenspeicherregister**
IorD	Der Inhalt des PCs liefert die Adresse für den Speicher	Der Inhalt von ALUOut liefert die Adresse für den Speicher
IRWrite	Keine	Der am Speicherausgang anliegende Wert wird in das Befehlsregister geschrieben
PCWrite	Keine	Der PC wird beschrieben. Die Quelle bestimmt wird durch PCSource bestimmt
PCWriteCond	Keine	Der PC wird beschrieben, falls das Zero-Statussignal gesetzt ist

Datenpfad für RISC-V mit Steuerung

- **Main Control Unit (Control)**
  - **Wirkung der 1-Bit-Steuersignale**

Signalname	Wirkung, wenn logisch 0	Wirkung, wenn logisch 1
PCSource	Der Ausgang der ALU (PC+4) wird an den PC geschickt	Der Inhalt von ALUOut (Zieladresse) wird an den PC geschickt

Datenpfad für RISC-V mit Steuerung

- **Main Control Unit (Control)**
  - **Wirkung der 2-Bit-Steuersignale**

Signalname	Wert (binär)	Wirkung,
ALUOp	00	ALU führt Addition aus
	01	ALU führt Subtraktion aus
	10	Das funct-Feld der Instruktion bestimmt die ALU Operation
ALUSrcB	00	Der zweite Operand der ALU kommt vom B-Register
	01	Der zweite Operand ist die Konstante 4
	10	Der zweite Operand ist die vorzeichenerweiterte Konstante

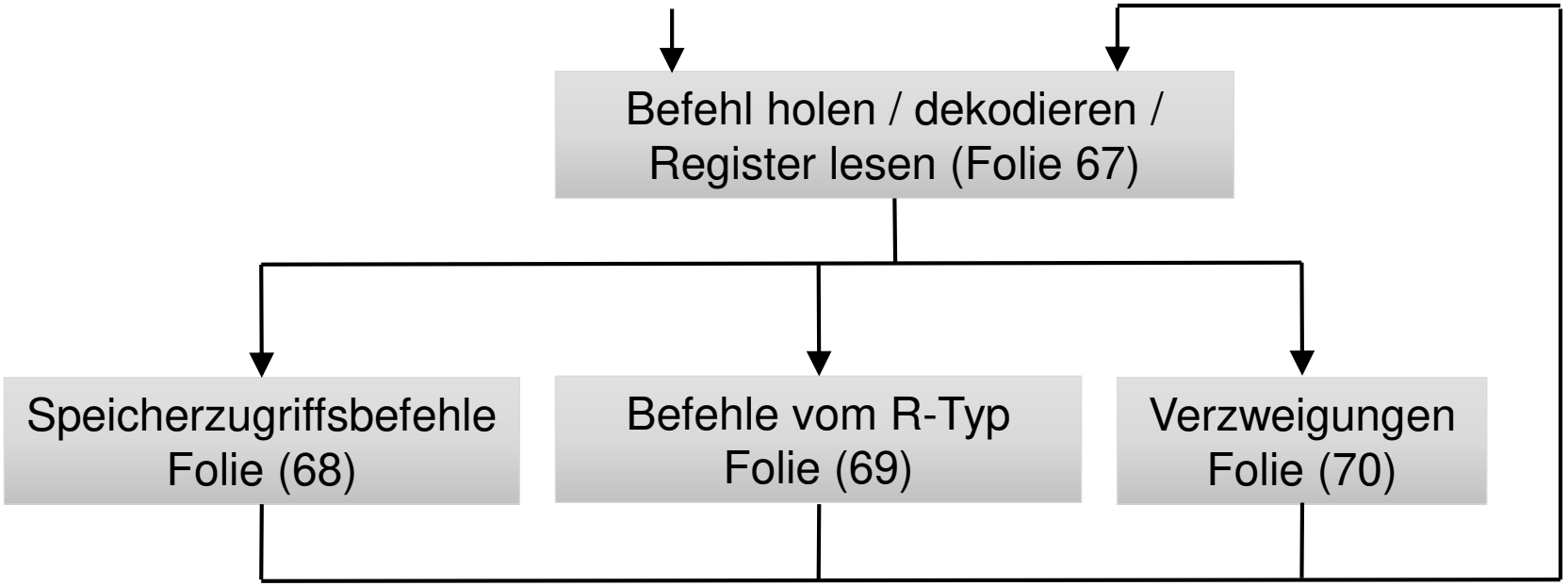
Datenpfad für RISC-V: Multizyklen-Implementierung

Aufbrechen der Befehlsausführung in Taktzyklen

Schritt	Aktionen für Befehle vom R-Type	Aktionen für Speicherzugriffsbefehle	Aktionen für Verzweigungen
Befehl holen	IR <= Memory[PC] PC <= PC + 4		
Befehl dekodieren / Register lesen	A <= Reg [IR[19:15]] B <= Reg [IR[24:20]] ALUOut <= PC + immediate		
Befehl ausführen, Adressberechnung, Verzweigung	ALUOut <= A op B	ALUOut <= A + immediate	if (A==B) PC <= ALUOut
Speicherzugriff oder Abschluss R-Type Befehl	Reg[11:7] <= ALUOut	**`lw`**: SR <= Memory [ALUOut] oder **`sw`**: Memory [ALUOut] <= B	
Abschluss Speicherzugriff		**`lw`**: Reg[IR[11:7] <= SR	

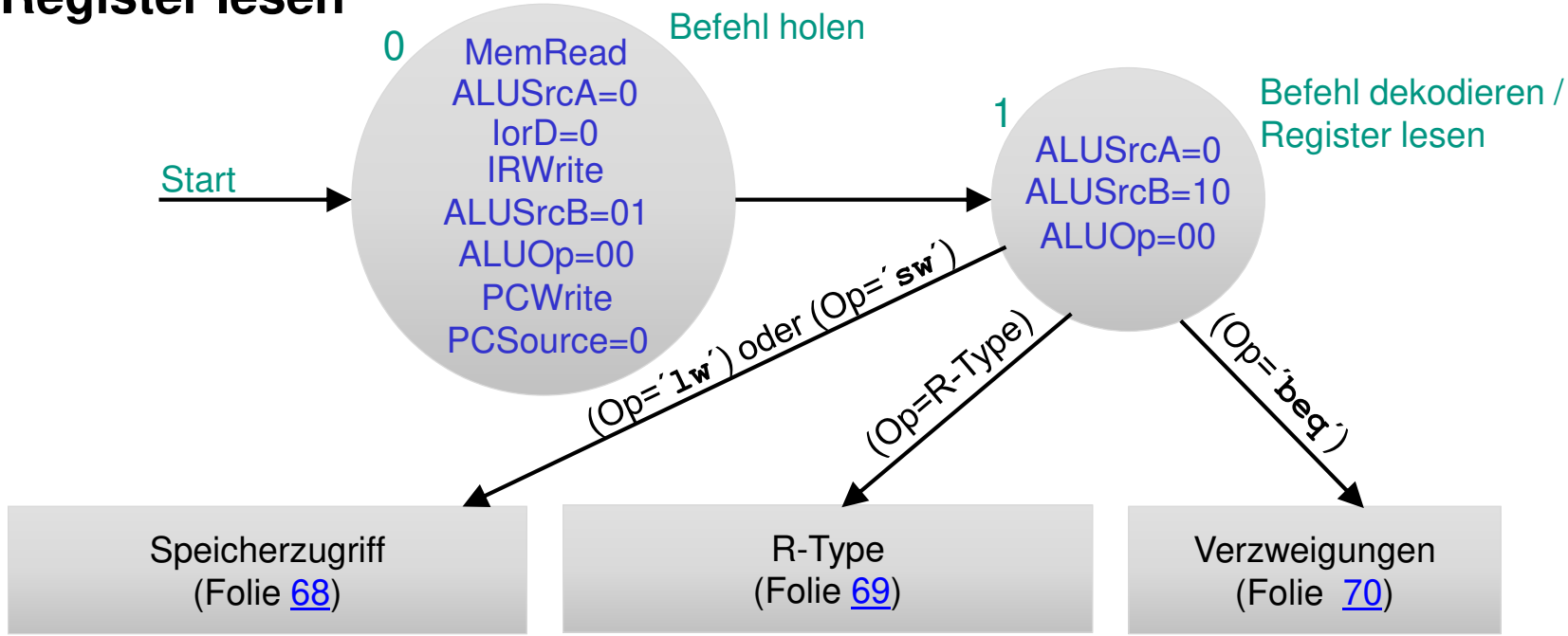
Datenpfad für RISC-V: Multizyklen-Implementierung

- **Zustandsautomat für die Steuerung**



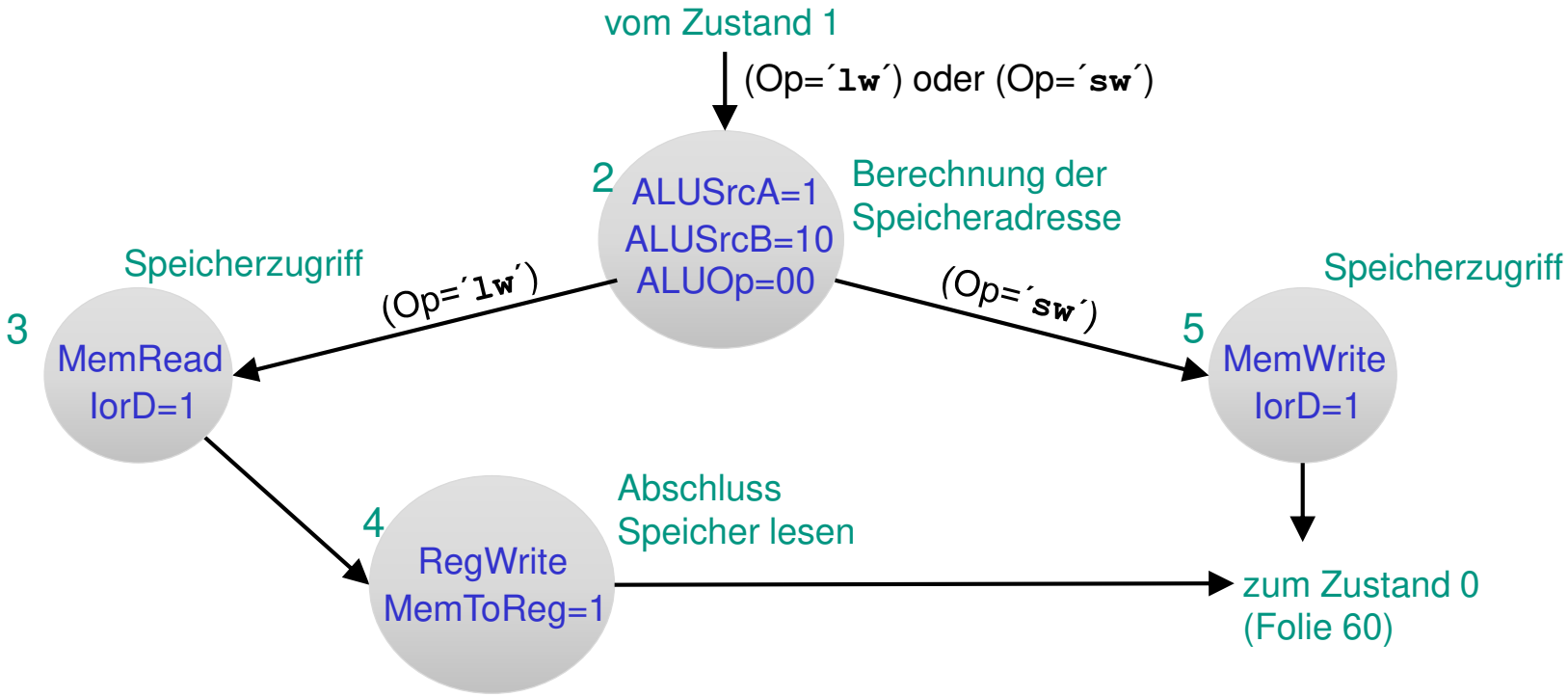
Datenpfad für RISC-V: Multizyklen-Implementierung

- **Zustandsautomat für die Steuerung: Befehl holen / dekodieren / Register lesen**



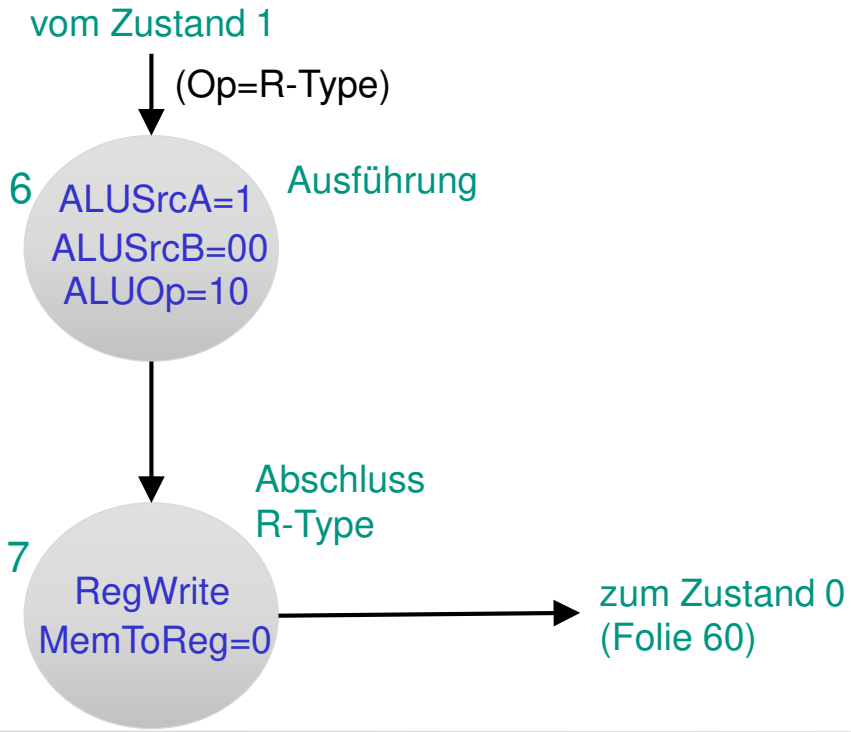
Datenpfad für RISC-V: Multizyklen-Implementierung

Zustandsautomat für die Steuerung: Speicherzugriff



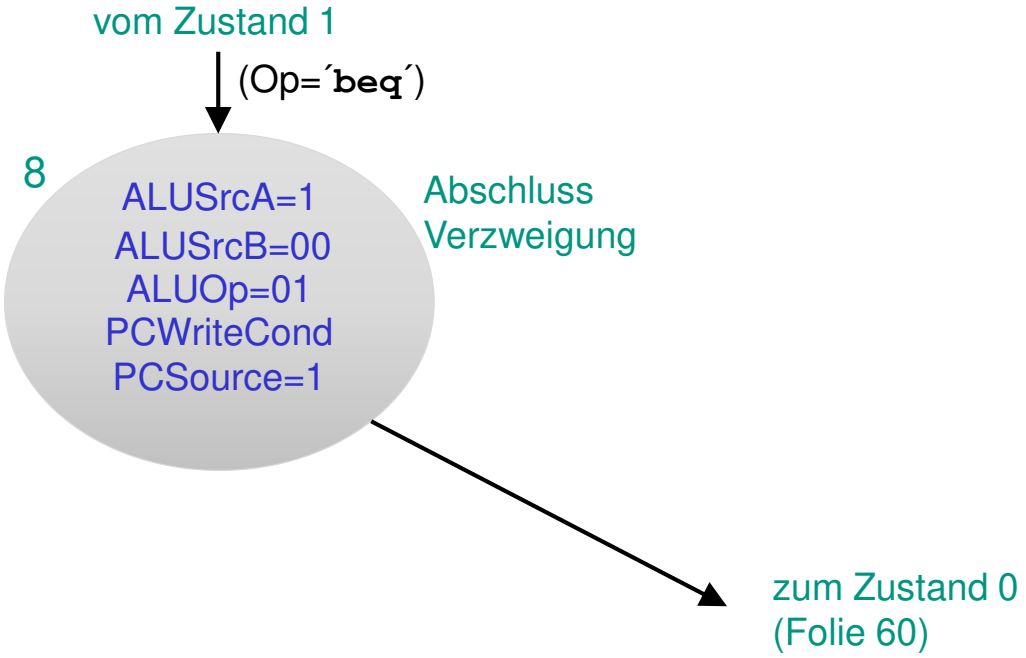
Datenpfad für RISC-V: Multizyklen-Implementierung

Zustandsautomat für die Steuerung: R-Type



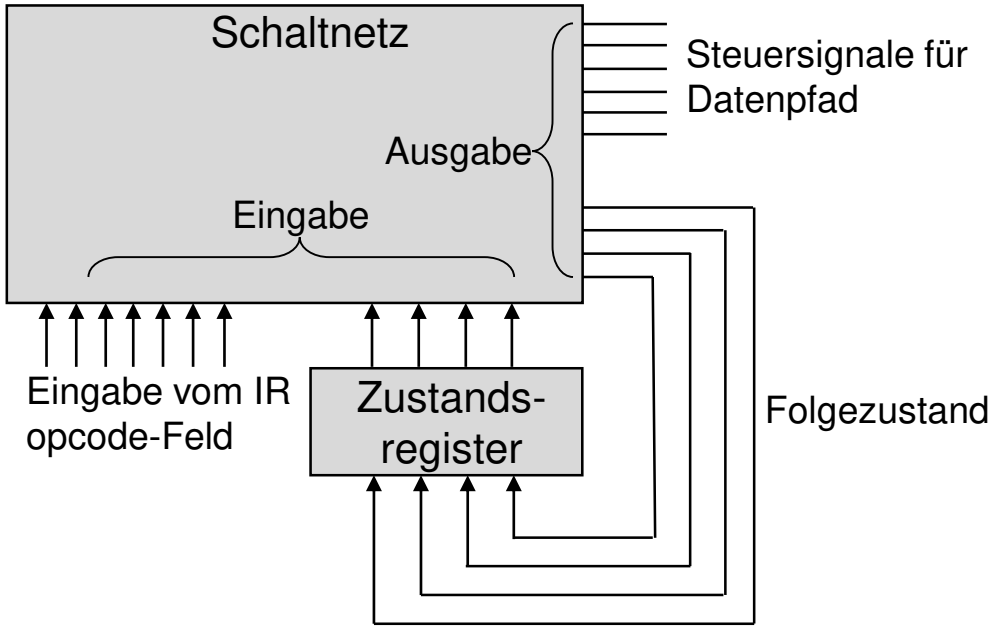
Datenpfad für RISC-V: Multizyklen-Implementierung

Zustandsautomat für die Steuerung: Verzweigungen



Datenpfad für RISC-V: Multizyklen-Implementierung

- **Implementierung des Zustandsautomaten (Moore Automat) als Schaltwerk**



Eingabe des Schaltnetzes:
- Aktueller Zustand
- Bits des IR opcode-Feldes

Ausgabe des Schaltnetzes:
- Nummer des Folgezustands
- gesetzte Steuersignale für den Datenpfad