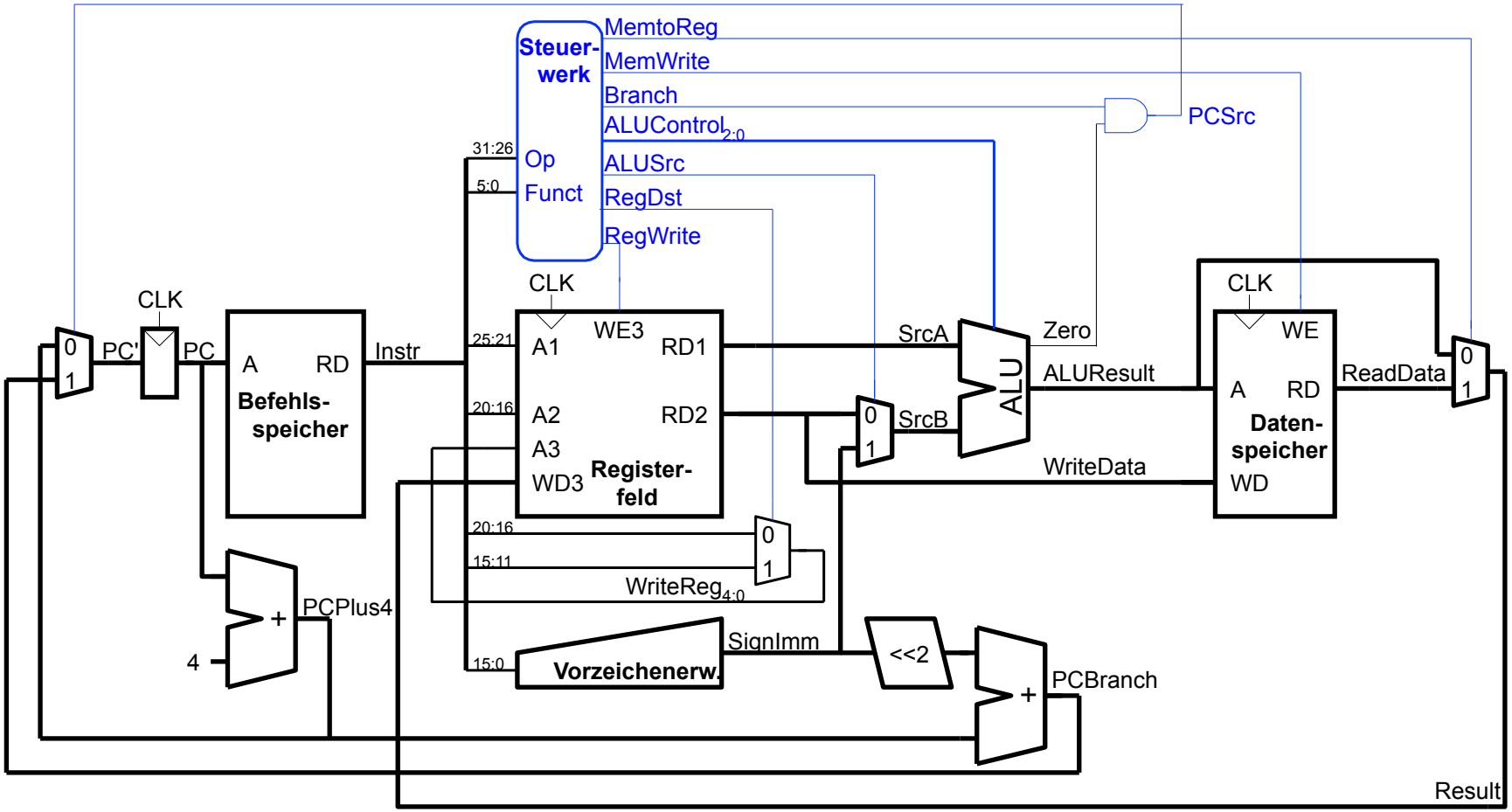
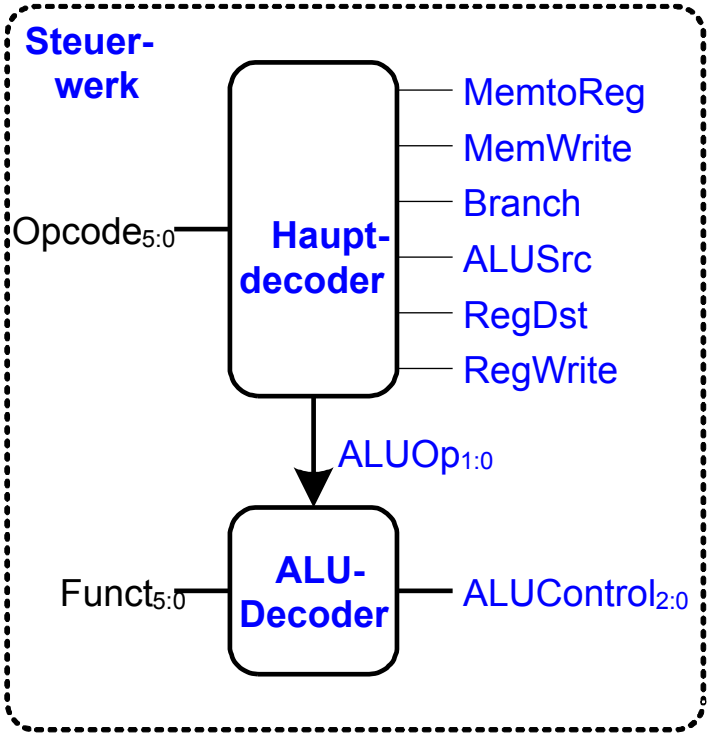
LEHRE WIKI FRAGE

- Bitte jetzt auf LEHRE WIKI eine Frage beantworten!

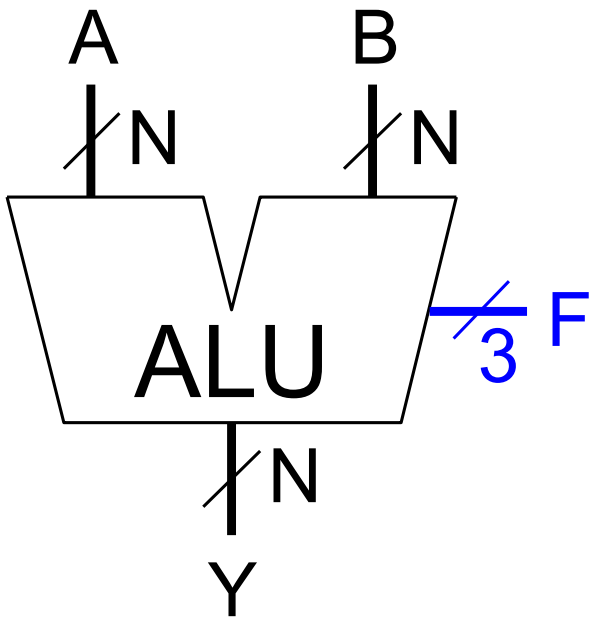
Vollständiger Ein-Takt-Prozessor



Steuerwerk

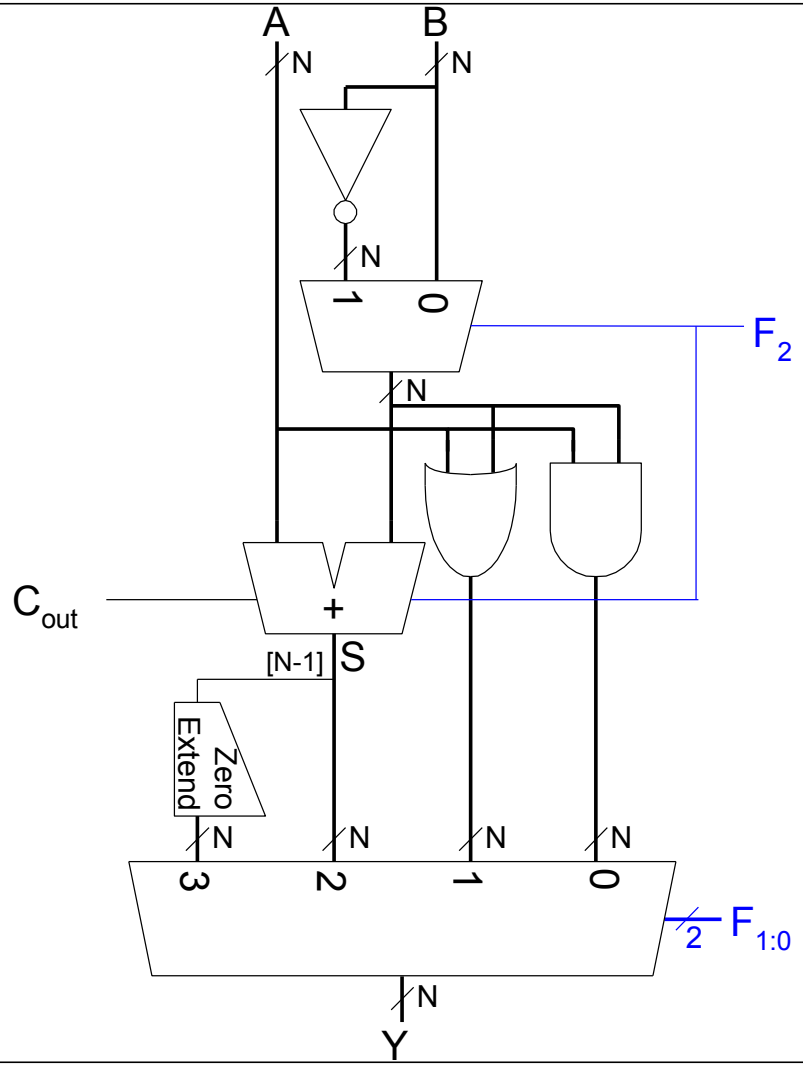

Steuer-
werk
Opcode5:0
Haupt-
decoder
MemtoReg
MemWrite
Branch
ALUSrc
RegDst
RegWrite
ALUOp1:0
Funct5:0
ALU-
Decoder
ALUControl2:0

Zur Erinnerung: ALU



$F_{2:0}$	Funktion
000	A & B
001	A | B
010	A + B
011	unbenutzt
100	A & ~B
101	A | ~B
110	A - B
111	SLT

Zur Erinnerung: ALU



Steuerwerk: ALU-Decoder

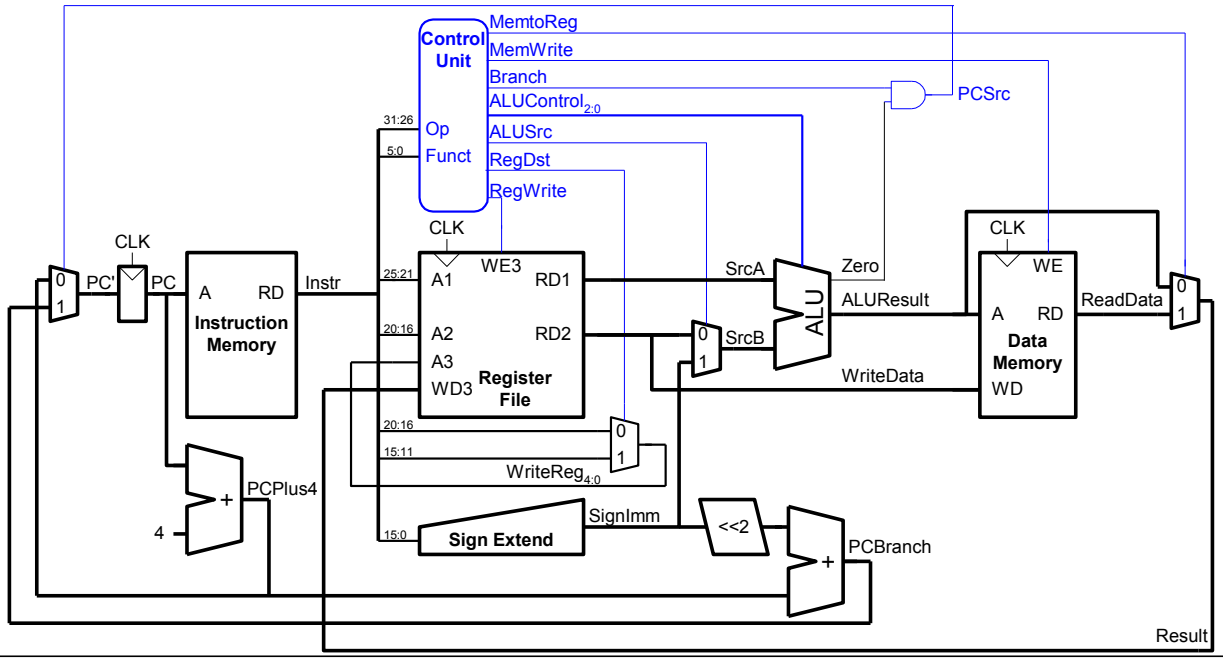
TECHNISCHE UNIVERSITÄT DARMSTADT

$ALUOp_{1:0}$	Bedeutung
00	Addiere
01	Subtrahiere
10	Werte Funct-Feld aus
11	unbenutzt

$ALUOp_{1:0}$	Funct	$ALUControl_{2:0}$
00	X	010 (Add)
X1	X	110 (Subtract)
1X	100000 (`add`)	010 (Add)
1X	100010 (`sub`)	110 (Subtract)
1X	100100 (`and`)	000 (And)
1X	100101 (`or`)	001 (Or)
1X	101010 (`slt`)	111 (SLT)

Steuerwerk: Hauptdecoder

Instruktion	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	$ALUOp_{1:0}$
R-Typ	000000							
`lw`	100011							
`sw`	101011							
`beq`	000100							

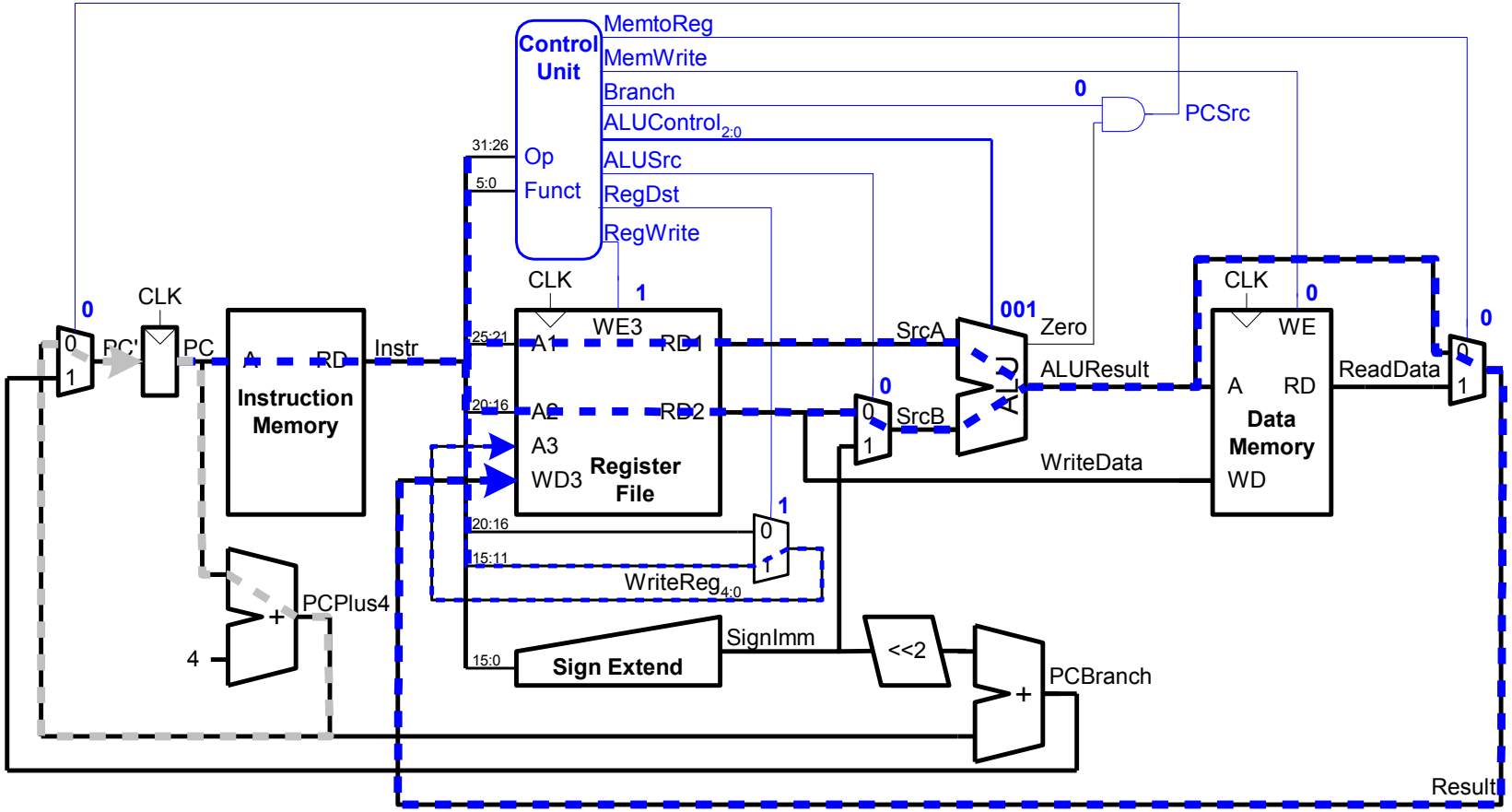


Steuerwerk: Hauptdecoder

TECHNISCHE UNIVERSITÄT DARMSTADT

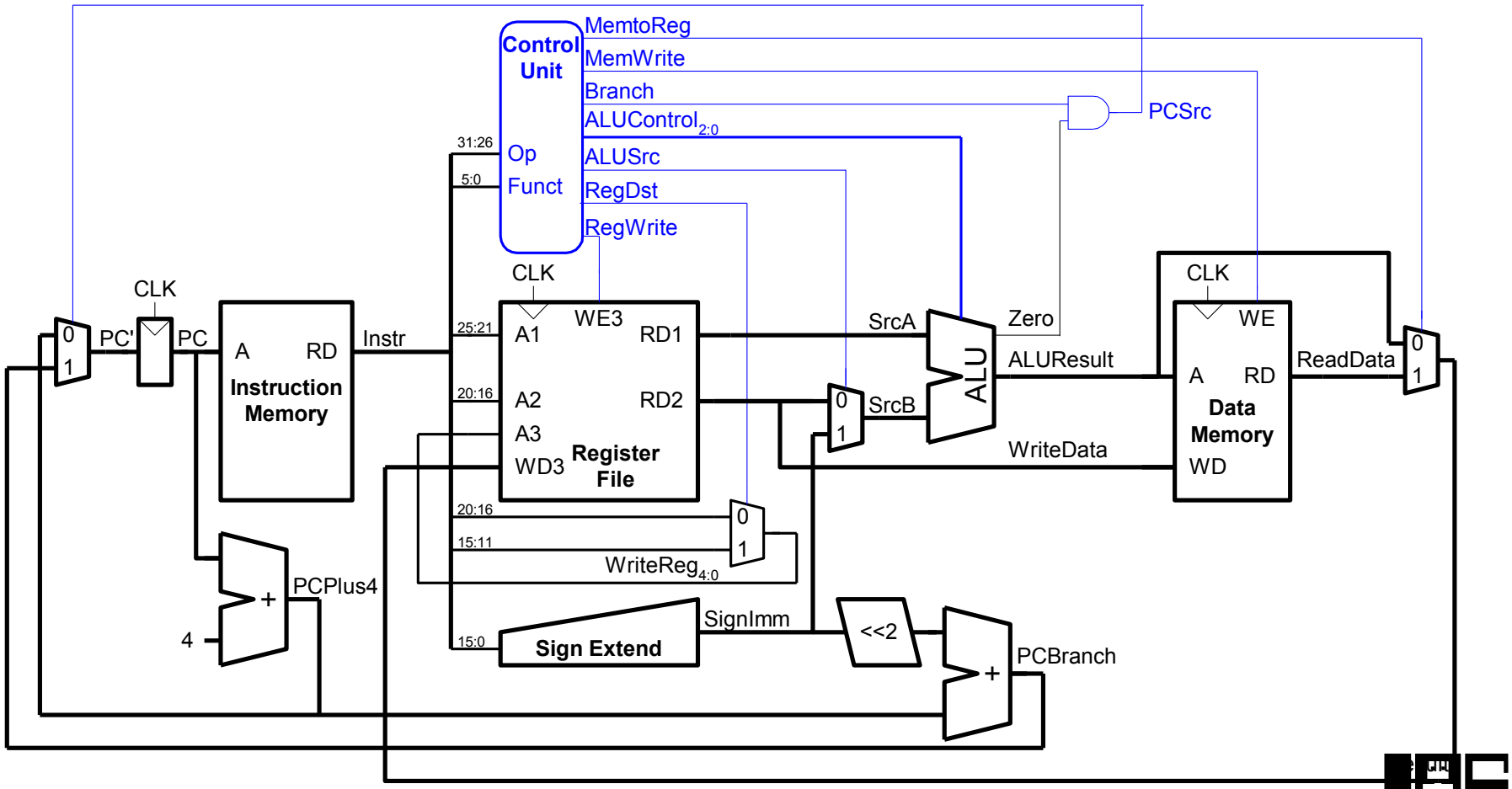
Instruction	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	$ALUOp_{1:0}$
R-type	000000	1	1	0	0	0	0	10
`lw`	100011	1	0	1	0	0	1	00
`sw`	101011	0	X	1	0	1	X	00
`beq`	000100	0	X	0	1	0	X	01

Beispiel im Ein-Takt Datenpfad: `or`



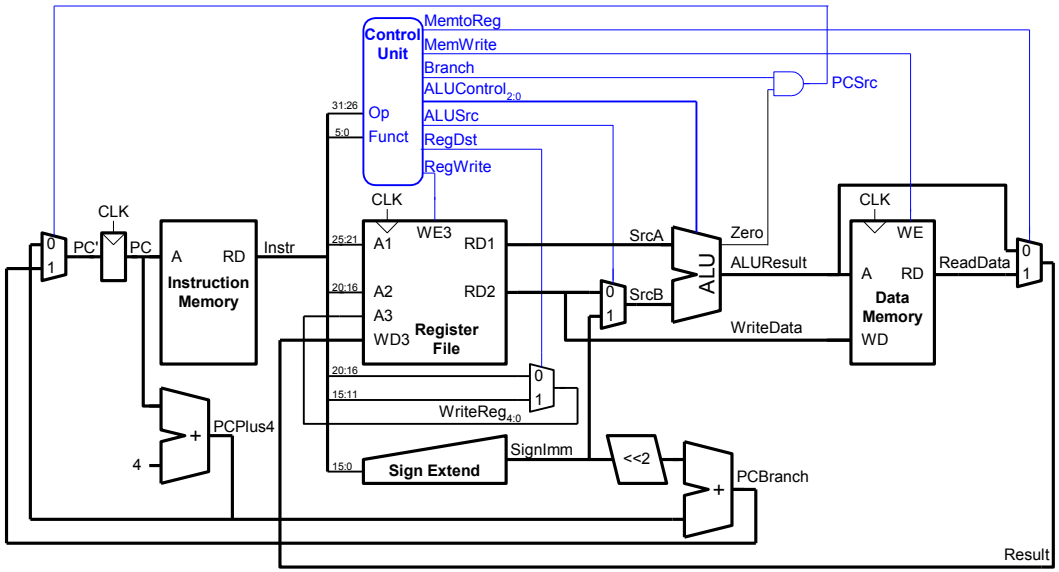
Erweitere Funktionalität: `addi`

- Keine Änderung am Datenpfad nötig



Erweitere Steuerwerk: `addi`

Instruktion	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	$ALUOp_{1:0}$
R-Typ	000000	1	1	0	0	0	0	10
`lw`	100011	1	0	1	0	0	1	00
`sw`	101011	0	X	1	0	1	X	00
`beq`	000100	0	X	0	1	0	X	01
`addi`	**001000**							

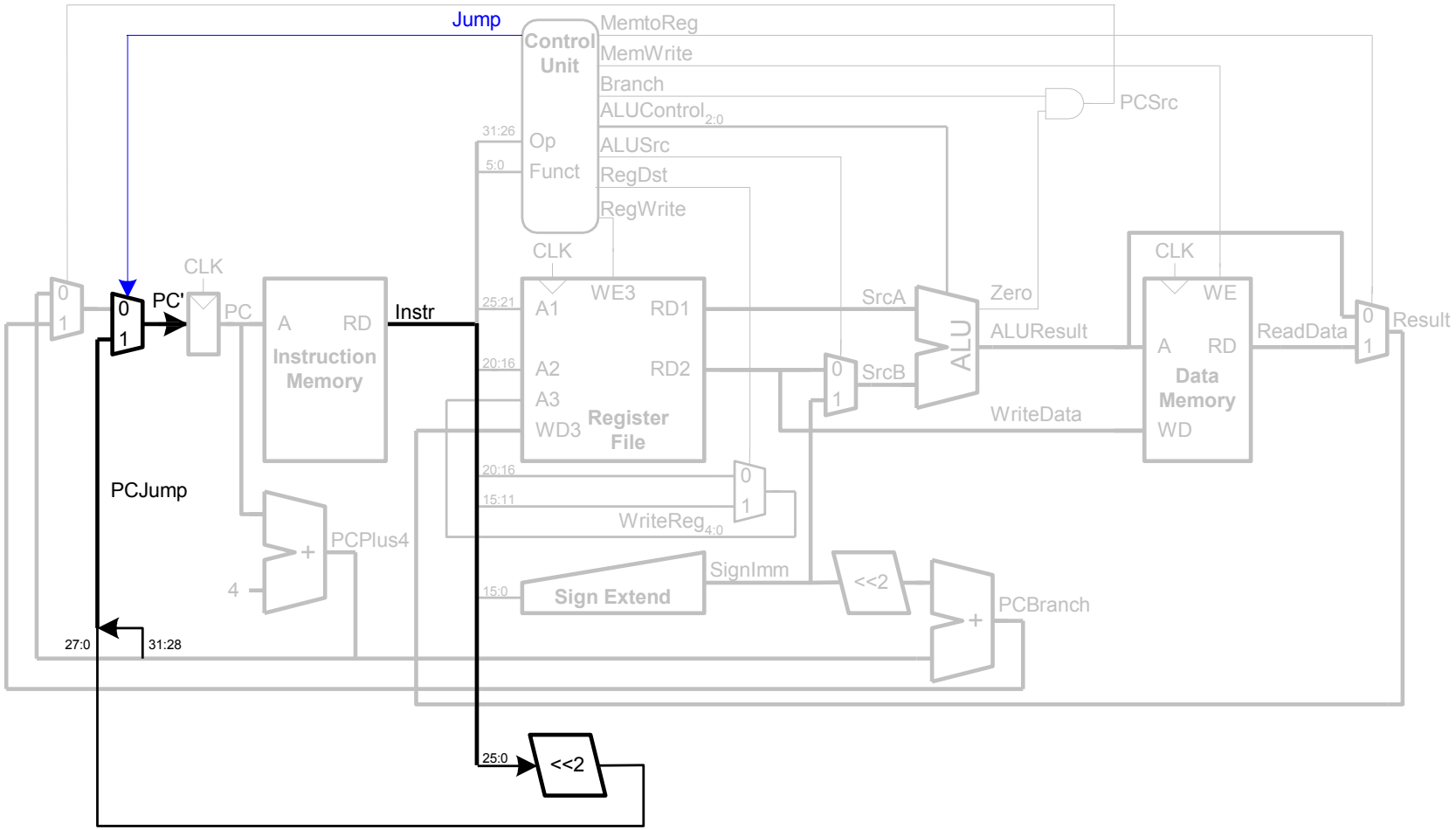


Erweitere Steuerwerk: `addi`

TECHNISCHE UNIVERSITÄT DARMSTADT

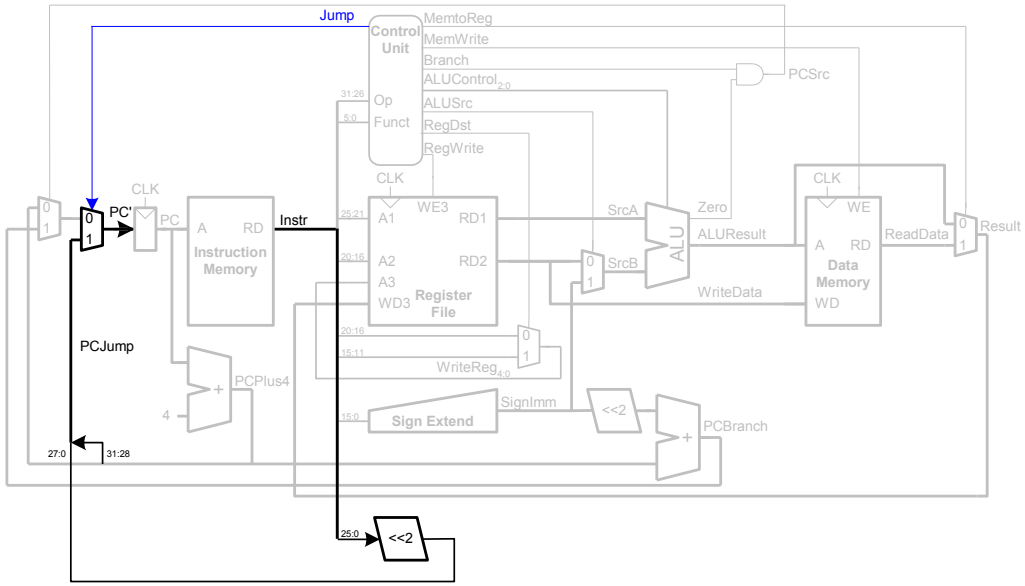
Instruktion	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	$ALUOp_{1:0}$
R-Typ	000000	1	1	0	0	0	0	10
`lw`	100011	1	0	1	0	0	1	00
`sw`	101011	0	X	1	0	1	X	00
`beq`	000100	0	X	0	1	0	X	01
`addi`	**001000**	**1**	**0**	**1**	**0**	**0**	**0**	**00**

Erweitere Funktionalität: j



Steuerwerk: Hauptdecoder

Instruktion	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	$ALUOp_{1:0}$	Jump
R-Typ	000000	1	1	0	0	0	0	10	0
`lw`	100011	1	0	1	0	0	1	00	0
`sw`	101011	0	X	1	0	1	X	00	0
`beq`	000100	0	X	0	1	0	X	01	0
`j`	**000010**								



Steuerwerk: Hauptdecoder

TECHNISCHE UNIVERSITÄT DARMSTADT

Instruction	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	$ALUOp_{1:0}$	Jump
R-type	000000	1	1	0	0	0	0	10	0
`lw`	100011	1	0	1	0	0	1	00	0
`sw`	101011	0	X	1	0	1	X	00	0
`beq`	000100	0	X	0	1	0	X	01	0
`j`	**000010**	**0**	**X**	**X**	**X**	**0**	**X**	**XX**	**1**

Wiederholung: Rechenleistung des Prozessors

TECHNISCHE UNIVERSITÄT DARMSTADT

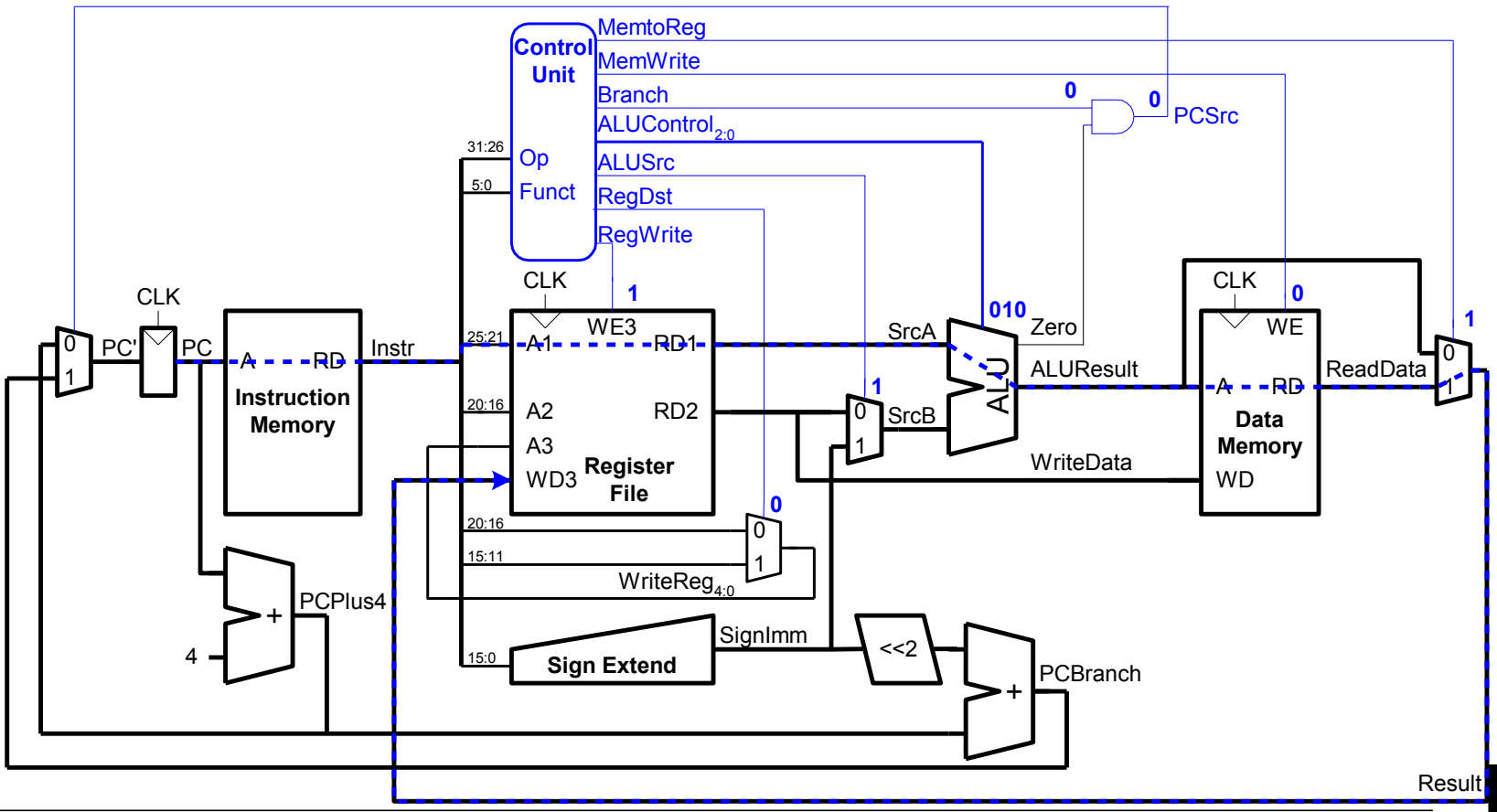
Programmausführungszeit

= (# Instruktionen) (Takte/Instruktion)(Sekunden/Takt)

= # Instruktionen CPI T_C

Rechenleistung des Ein-Takt-Prozessors

- T_C wird durch längsten Pfad bestimmt (`lw`)



Rechenleistung des Ein-Takt-Prozessors

TECHNISCHE UNIVERSITÄT DARMSTADT

- Kritischer Pfad:
  $T_c = t_{pcq_PC} + t_{\text{mem}} + \max(t_{RF\text{read}}, t_{sext} + t_{\text{mux}}) + t_{\text{ALU}} + t_{\text{mem}} + t_{\text{mux}} + t_{RF\text{setup}}$
- In vielen Implementierungen: Kritischer Pfad durch
  - Speicher, ALU, Registerfeld
- Damit:
  - $T_c = t_{pcq_PC} + 2t_{\text{mem}} + t_{RF\text{read}} + t_{\text{mux}} + t_{\text{ALU}} + t_{RF\text{setup}}$

LEHRE WIKI TEST IN DREI FOLIEN

Ein-Takt Prozessor Rechenleistung: Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

Element	Parameter	Verzögerung (ps)
Register Clock-to-Q	t_{pcq_PC}	30
Register setup	t_{setup}	20
Multiplexer	t_{mux}	25
ALU	t_{ALU}	200
Speicher lesen	t_{mem}	250
Registerfeld lesen	t_{RFread}	150
Registerfeld setup	$t_{RFsetup}$	20

$$T_c = t_{pcq_PC} + 2t_{mem} + t_{RFread} + t_{mux} + t_{ALU} + t_{RFsetup}$$
$$= [30 + 2(250) + 150 + 25 + 200 + 20] \text{ ps}$$
$$= 925 \text{ ps}$$

Ein-Takt Prozessor Rechenleistung: Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Auszuführen: Programm mit 100 Milliarden Instruktionen
  - Auf Ein-Takt MIPS Prozessor

Ausführungszeit =

Ein-Takt Prozessor Rechenleistung: Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Auszuführen: Programm mit 100 Milliarden Instruktionen
  - Auf Ein-Takt MIPS Prozessor

Ausführungszeit = # Instruktionen CPI T_C

$= (100 \times 10^9)\ (1)\ (925 \times 10^{-12}\ s)$

= 92,5 Sekunden

LEHRE WIKI FRAGE

- Bitte jetzt auf LEHRE WIKI eine Frage beantworten!

Mehrtakt-MIPS-Prozessor

TECHNISCHE UNIVERSITÄT DARMSTADT

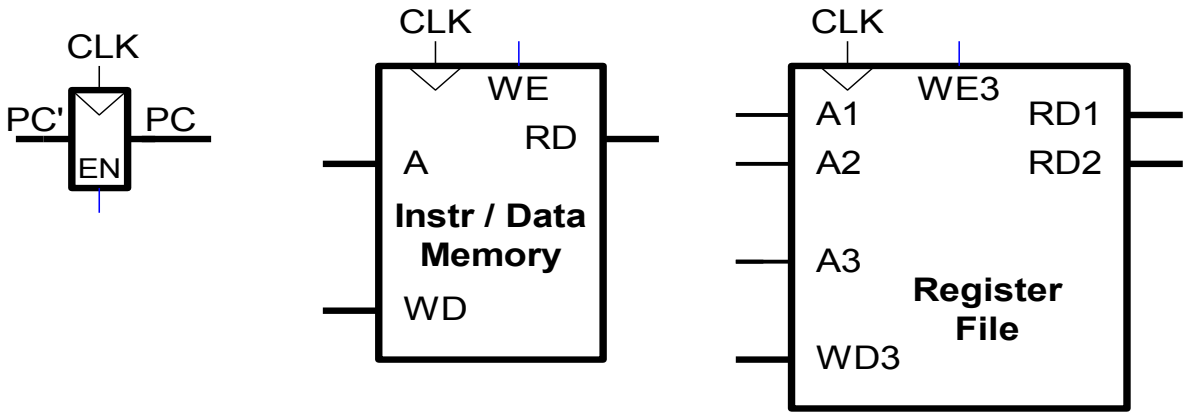
- Ein-Takt-Mikroarchitektur:
  + einfach
  - Taktfrequenz wird durch langsamste Instruktion bestimmt (`lw`)
  - Drei Addierer / ALUs und zwei Speicher

- Mehrtaktmikroarchitektur:
  + höhere Taktfrequenz
  + einfachere Instruktionen laufen schneller
  + bessere Wiederverwendung von Hardware in verschiedenen Takten
  - aufwendigere Ablaufsteuerung

- Gleiche Grundkomponenten
  - Datenpfad
  - Steuerwerk

Zustandselemente im Mehrtaktprozessor

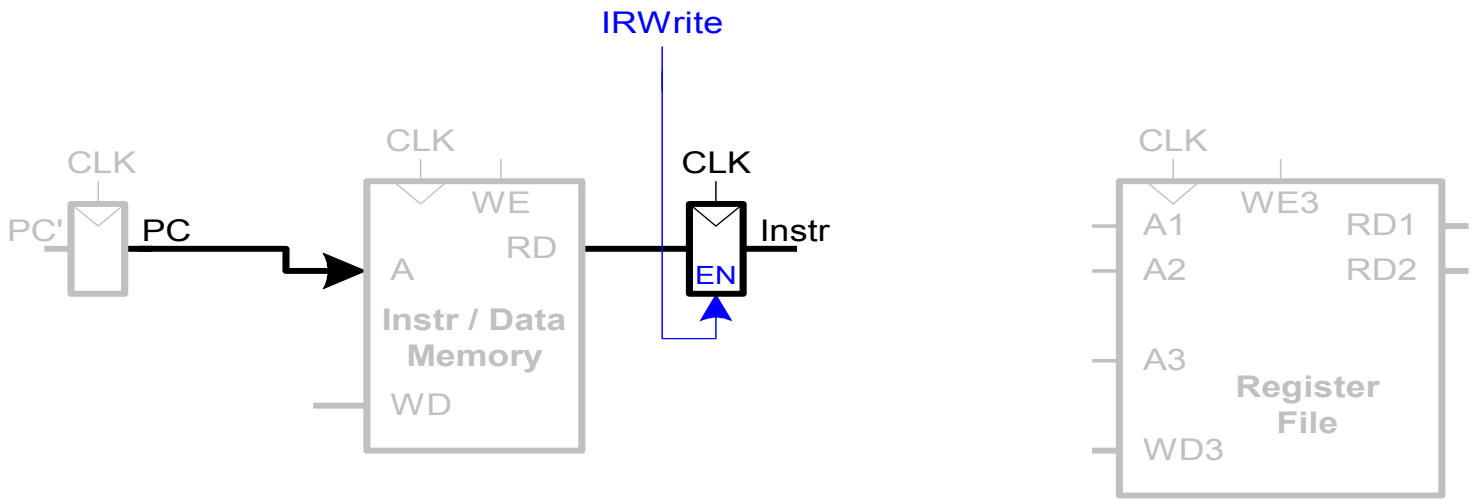
- Ersetze getrennte Instruktions- und Datenspeicher
  - Harvard-Architektur
- Durch einen gemeinsamen Speicher
  - Von Neumann-Architektur
  - Heute weiter verbreitet



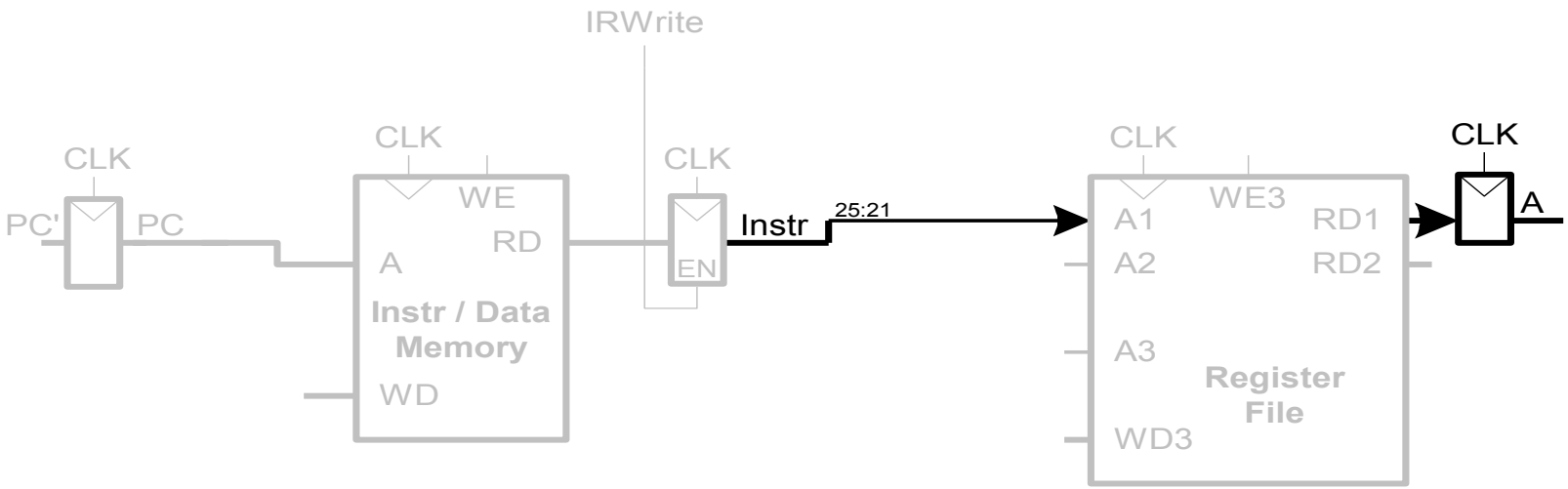
Mehrtaktdatenpfad: Instruktionen holen (*fetch*)

TECHNISCHE UNIVERSITÄT DARMSTADT

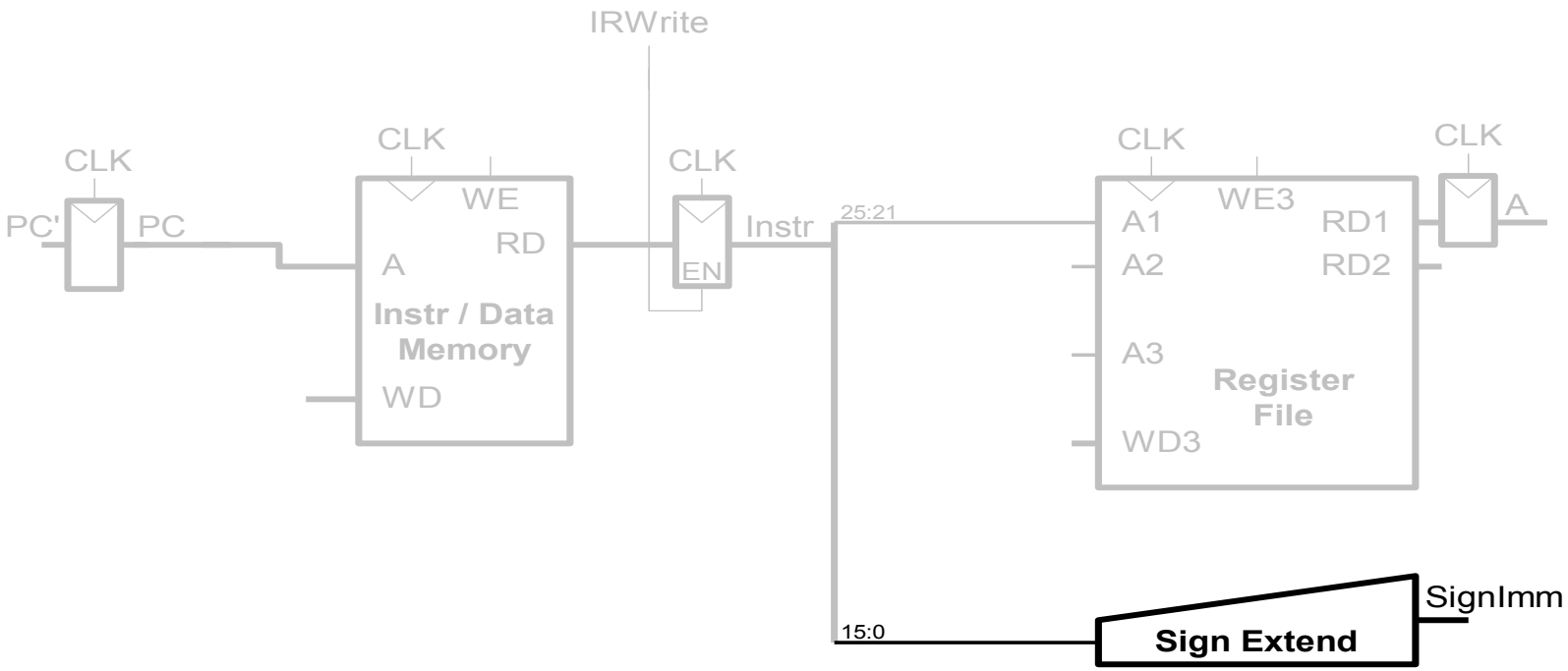
- Beispiel: Ausführung von `lw`
- **Schritt 1:** Hole Instruktion



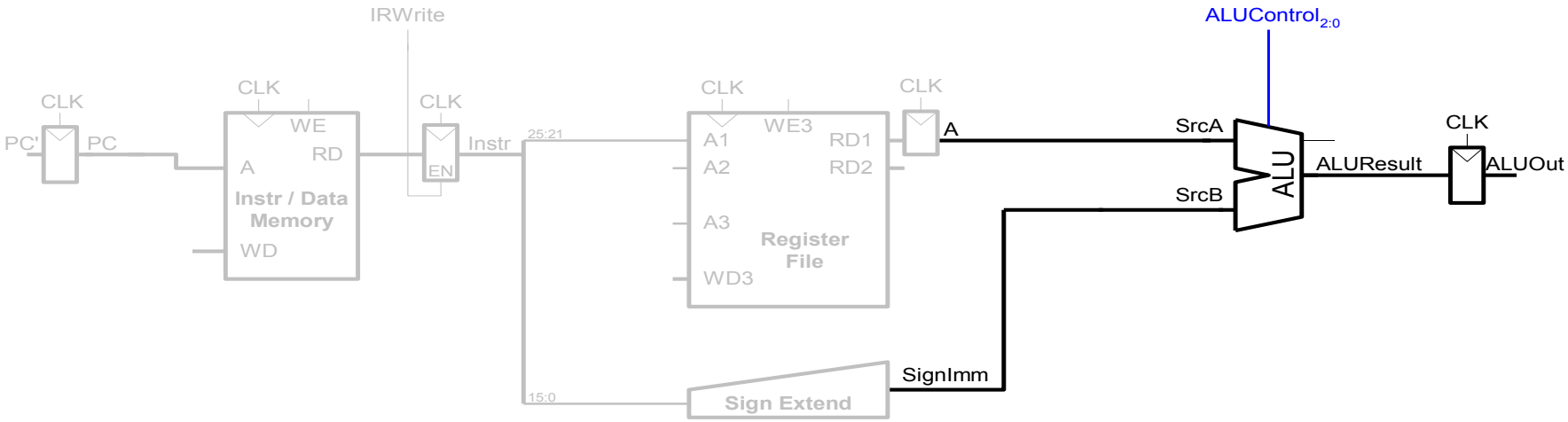
Mehrtaktdatenpfad: Lese Register für `lw`



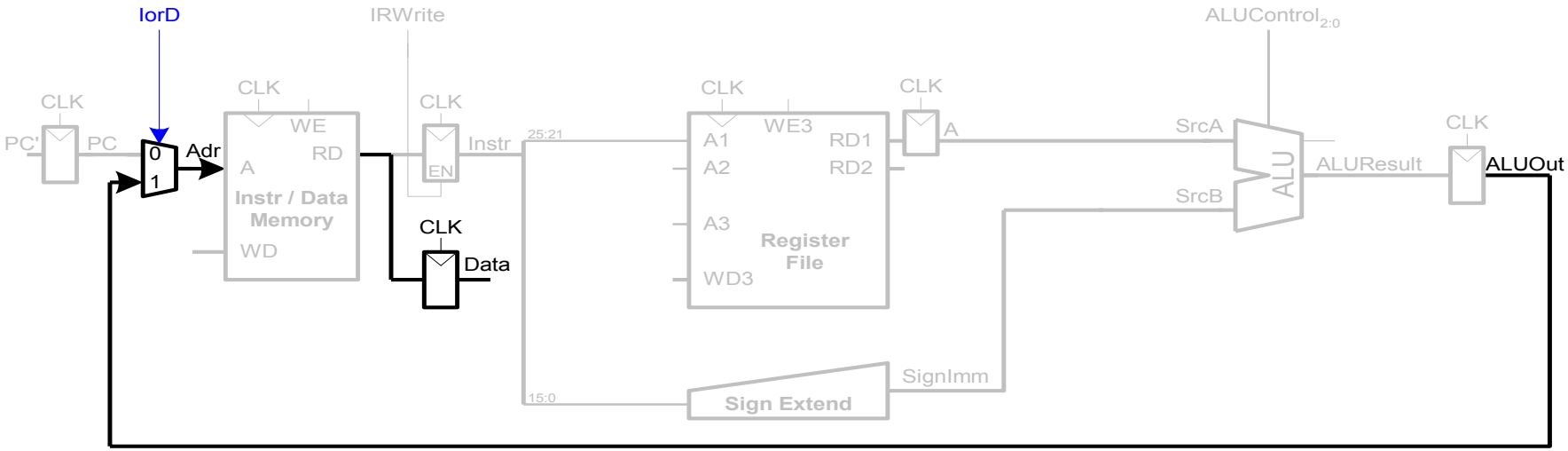
Mehrtaktdatenpfad: Werte `lw` Direktwert aus



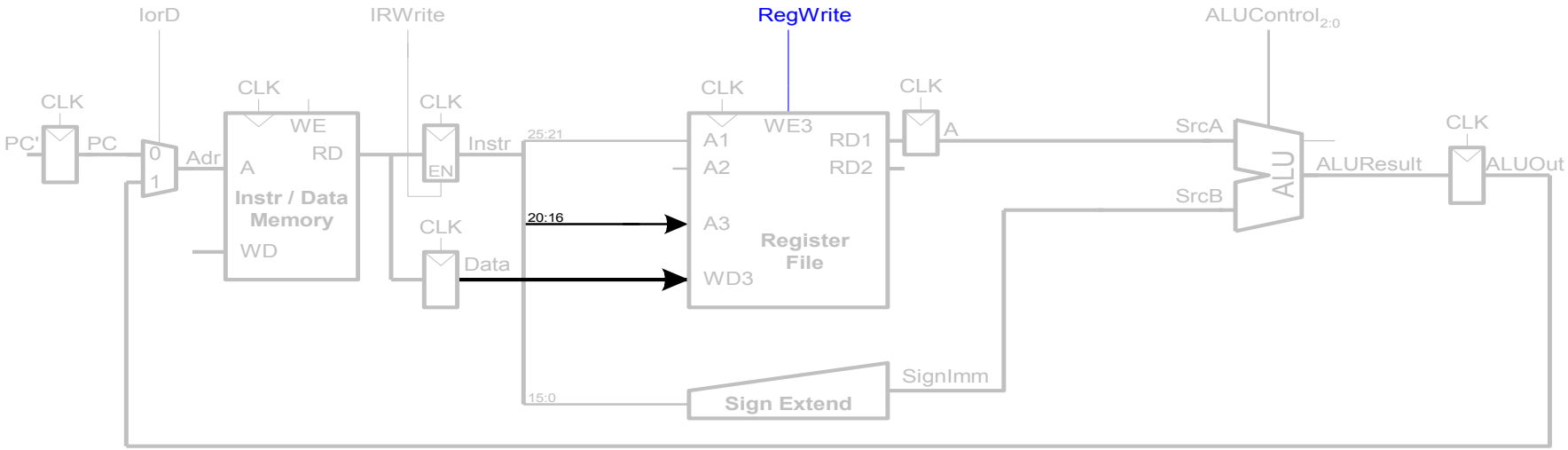
Mehrtaktdatenpfad: Bestimme effektive Adresse für `lw`



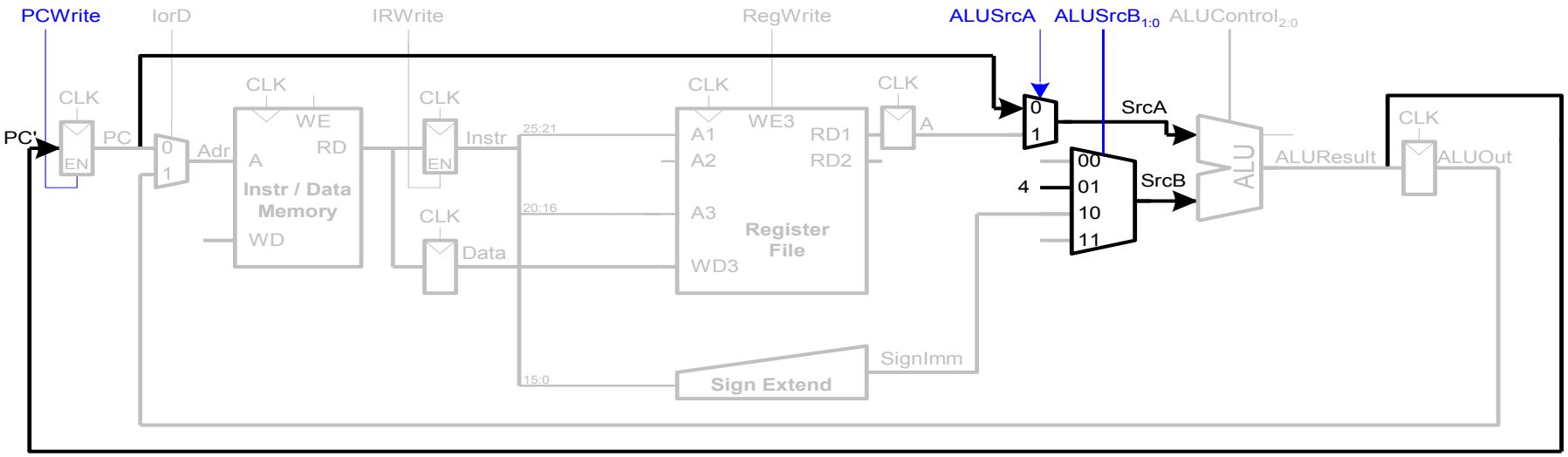
Mehrtaktdatenpfad: Lesezugriff von `lw`



Mehrtaktdatenpfad: Schreibe Register in `lw`



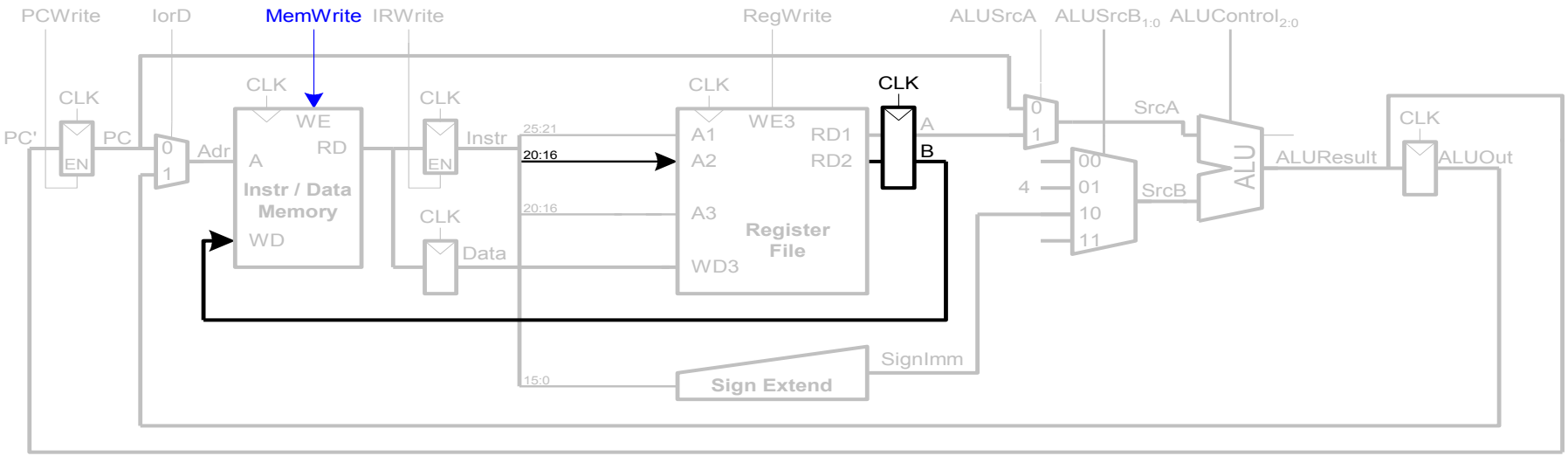
Mehrtaktdatenpfad: Erhöhe PC



Mehrtaktdatenpfad: Nun Ausführung von `sw`

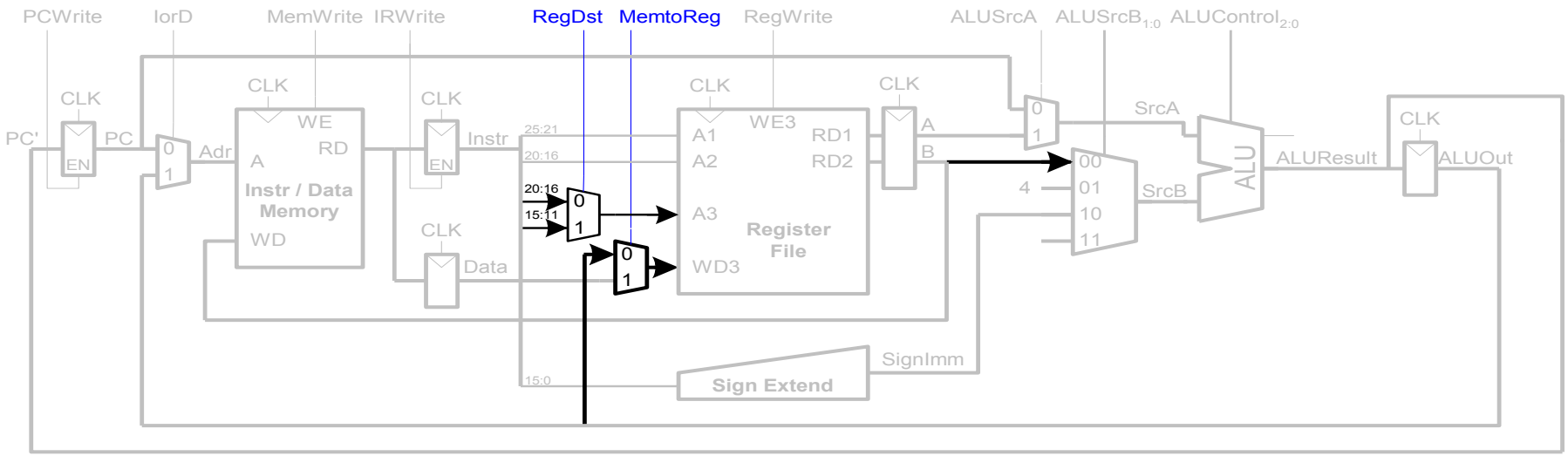
LEHRE WIKI TEST IN DREI FOLIEN

- Schreibe Daten aus `rt` in Speicher



Mehrtaktdatenpfad: Instruktion vom R-Typ

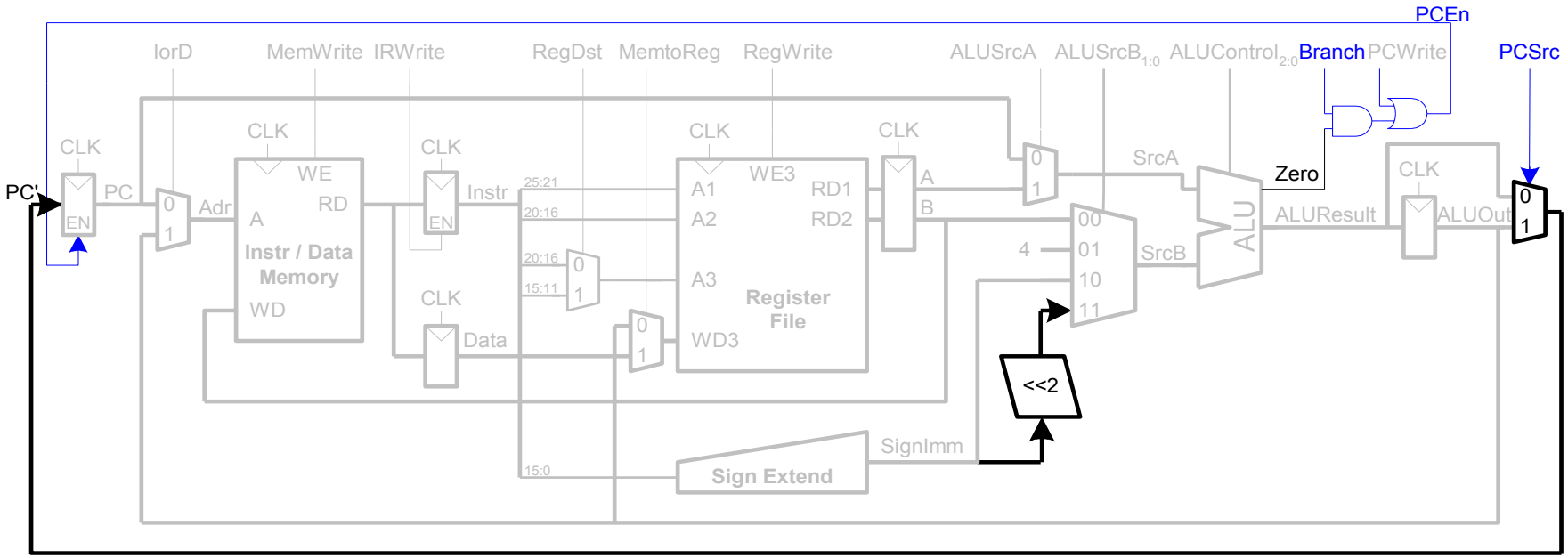
- Lese Werte aus `rs` und `rt`
- Schreibe *ALUResult* ins Registerfeld
- Schreibe Wert nach `rd` (statt nach `rt`)



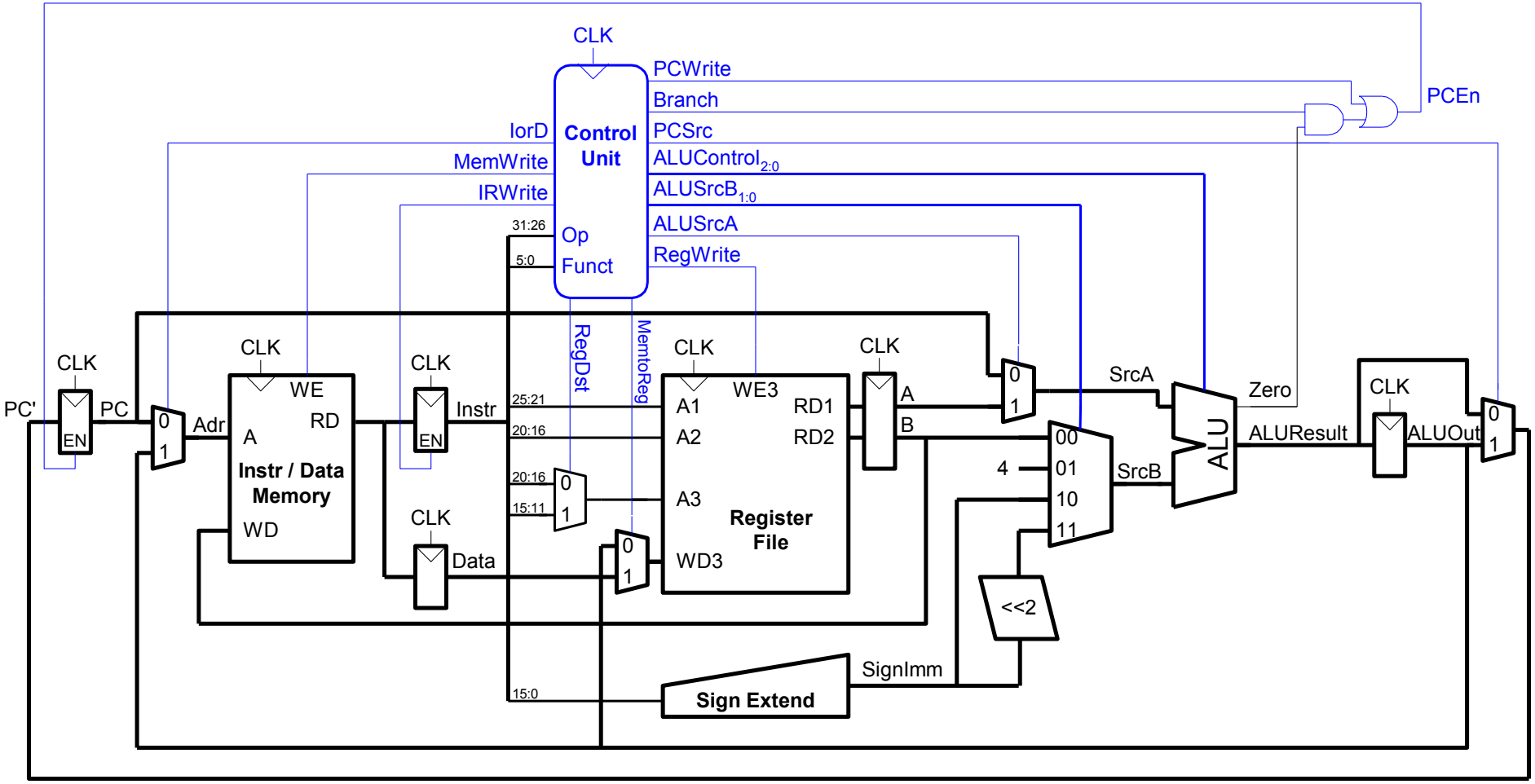
Mehrtaktdatenpfad: `beq`

- Prüfe, ob Werte in `rs` und `rt` gleich sind
- Bestimme Adresse des Sprungziels *(branch target address):*

  BTA = (vorzeichenerweiterter Direktwert << 2) + (PC+4)



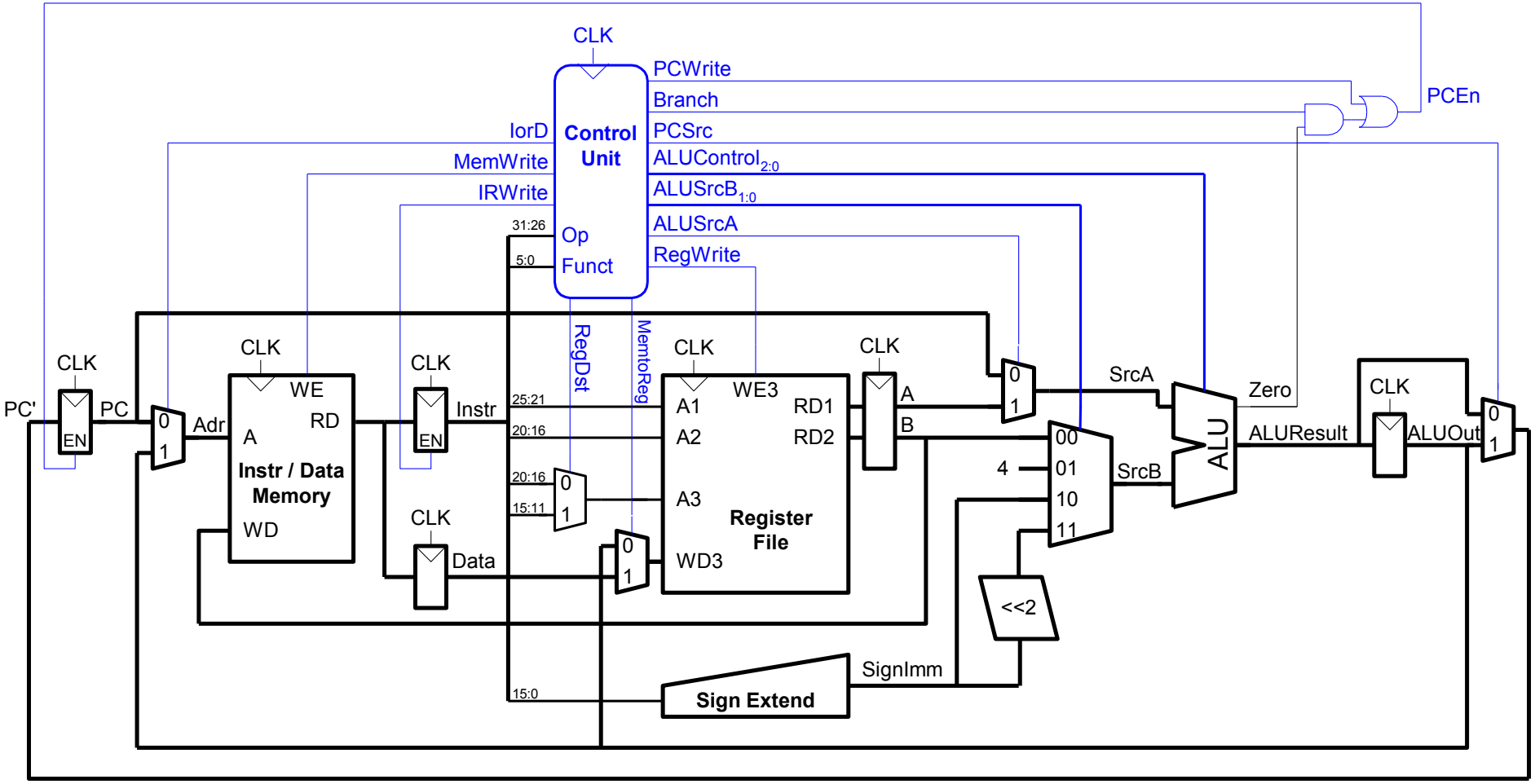
Vollständiger Mehrtaktprozessor



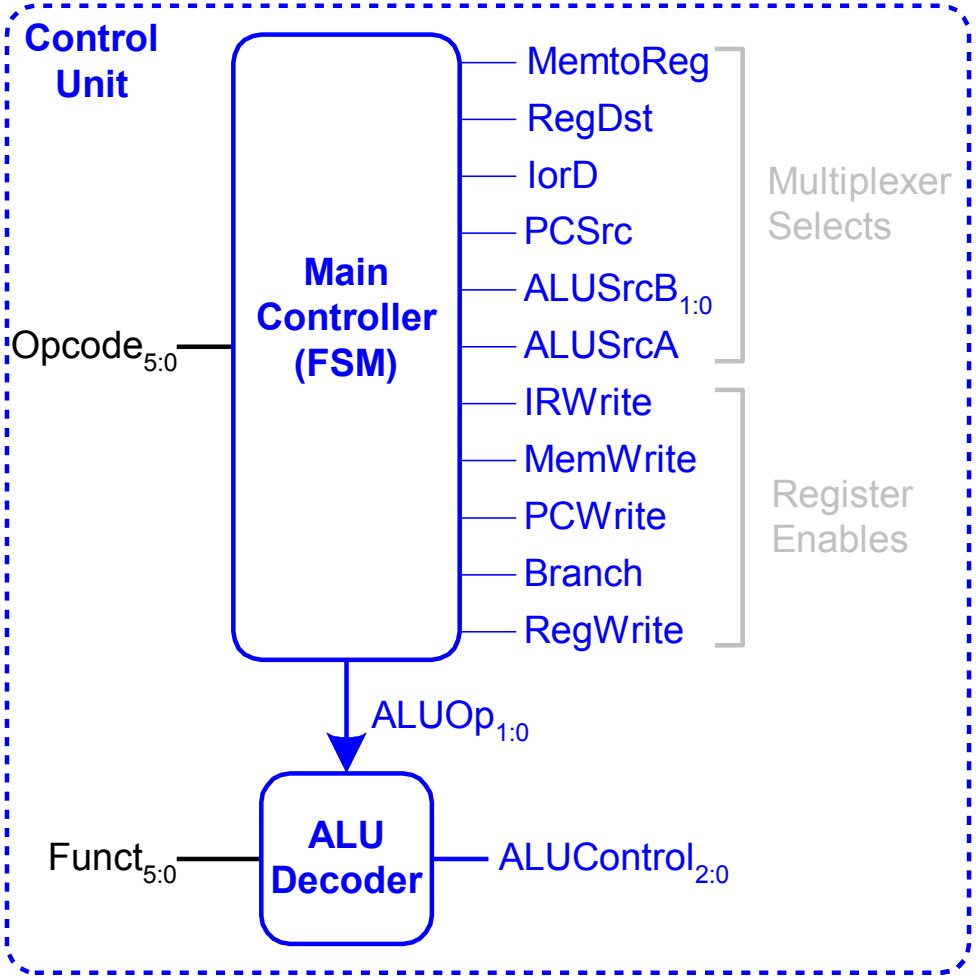
LEHRE WIKI FRAGE

- Bitte jetzt auf LEHRE WIKI eine Frage beantworten!

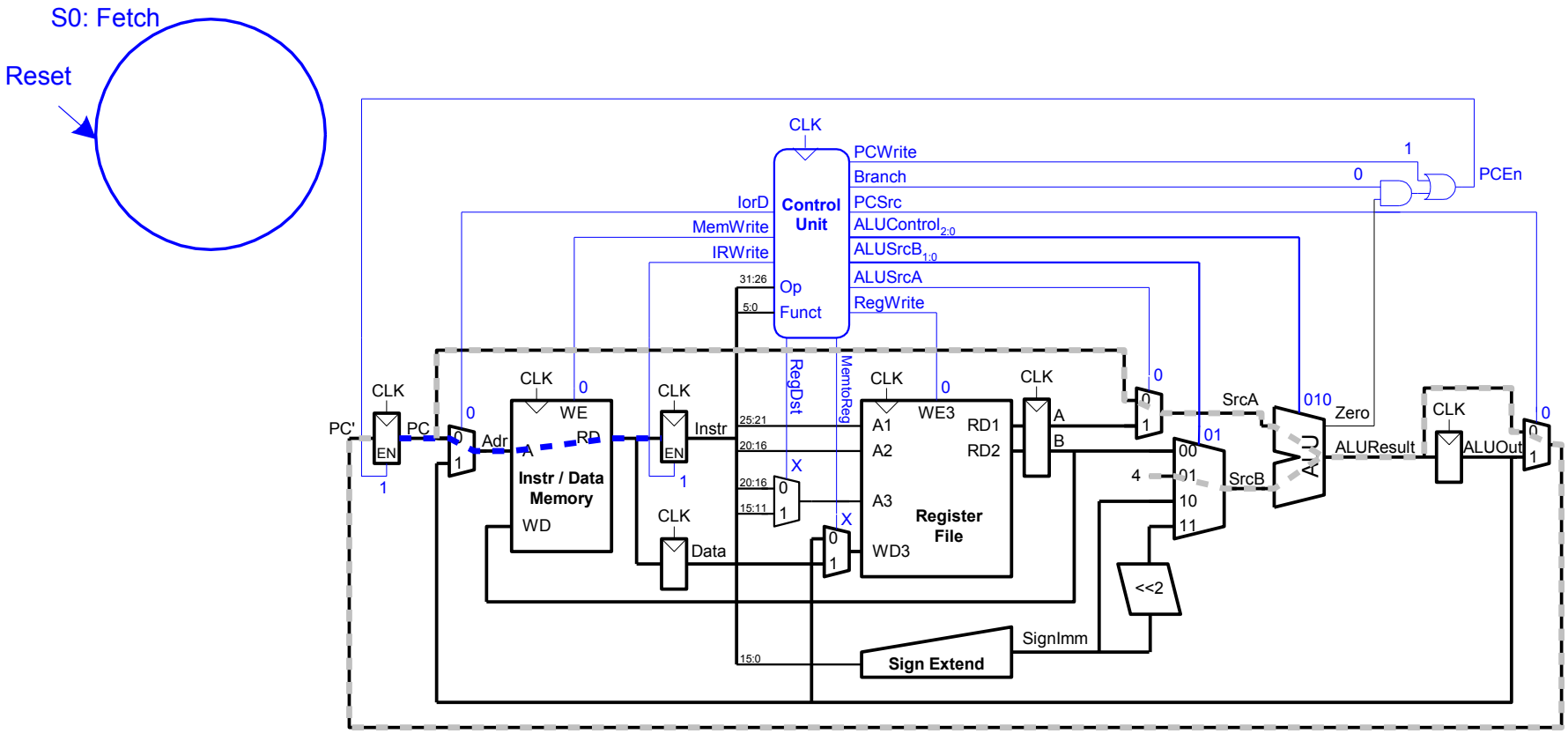
Vollständiger Mehrtaktprozessor



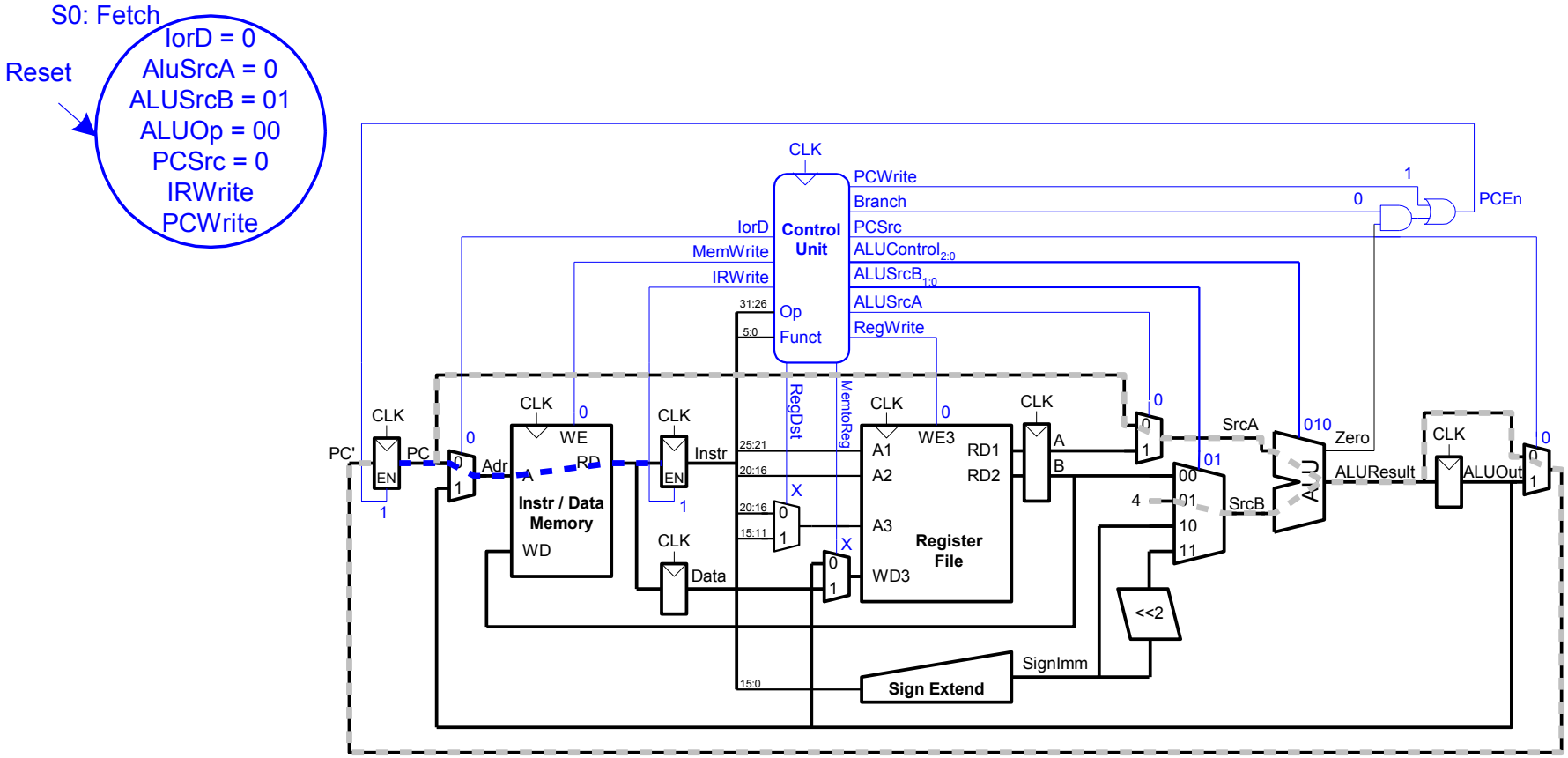
Steuerwerk


Control Unit
Main Controller (FSM)
Opcode5:0
MemtoReg
RegDst
IorD
PCSrc
ALUSrcB1:0
ALUSrcA
IRWrite
MemWrite
PCWrite
Branch
RegWrite
Multiplexer Selects
Register Enables
ALUOp1:0
ALU Decoder
Funct5:0
ALUControl2:0

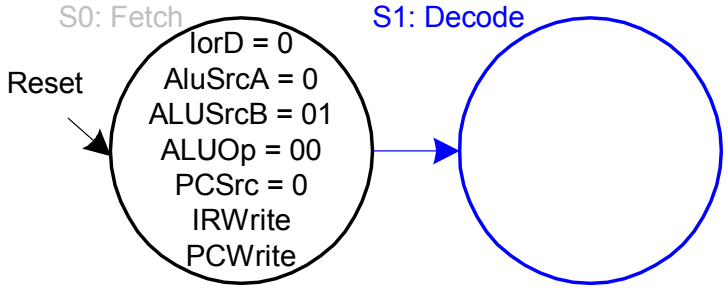
Hauptsteuerwerk: Holen eines Befehls

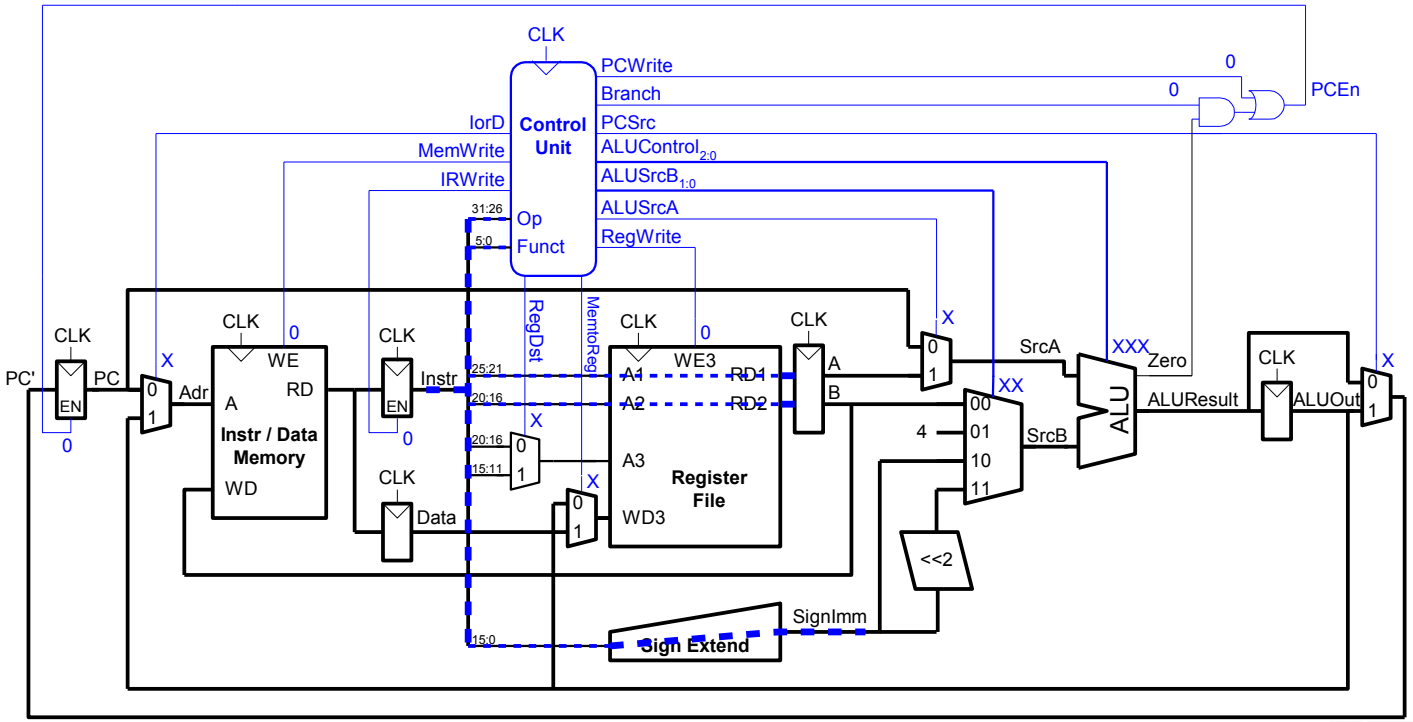


Hauptsteuerwerk: Holen eines Befehls

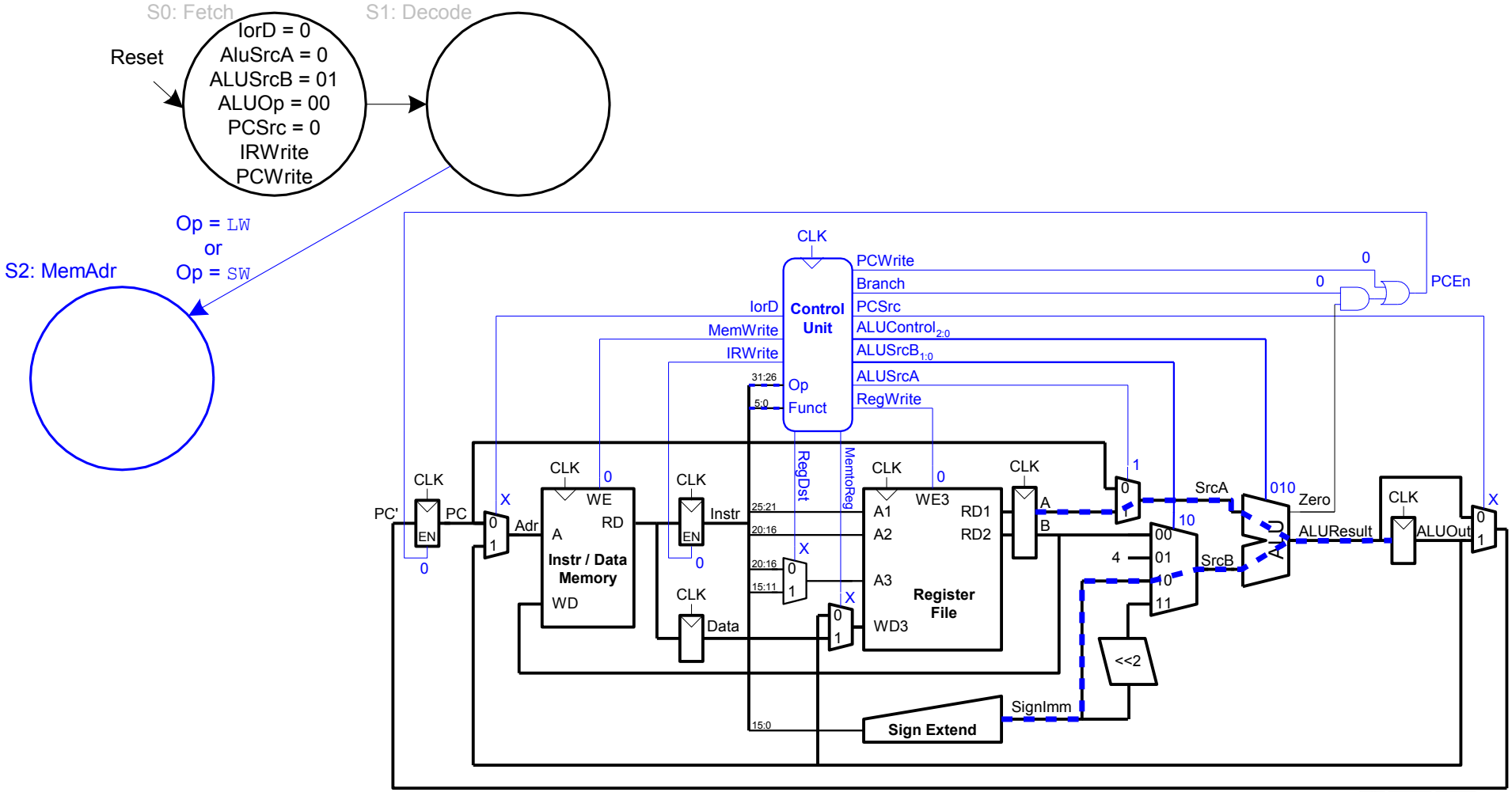


Hauptsteuerwerk: Dekodieren eines Befehls

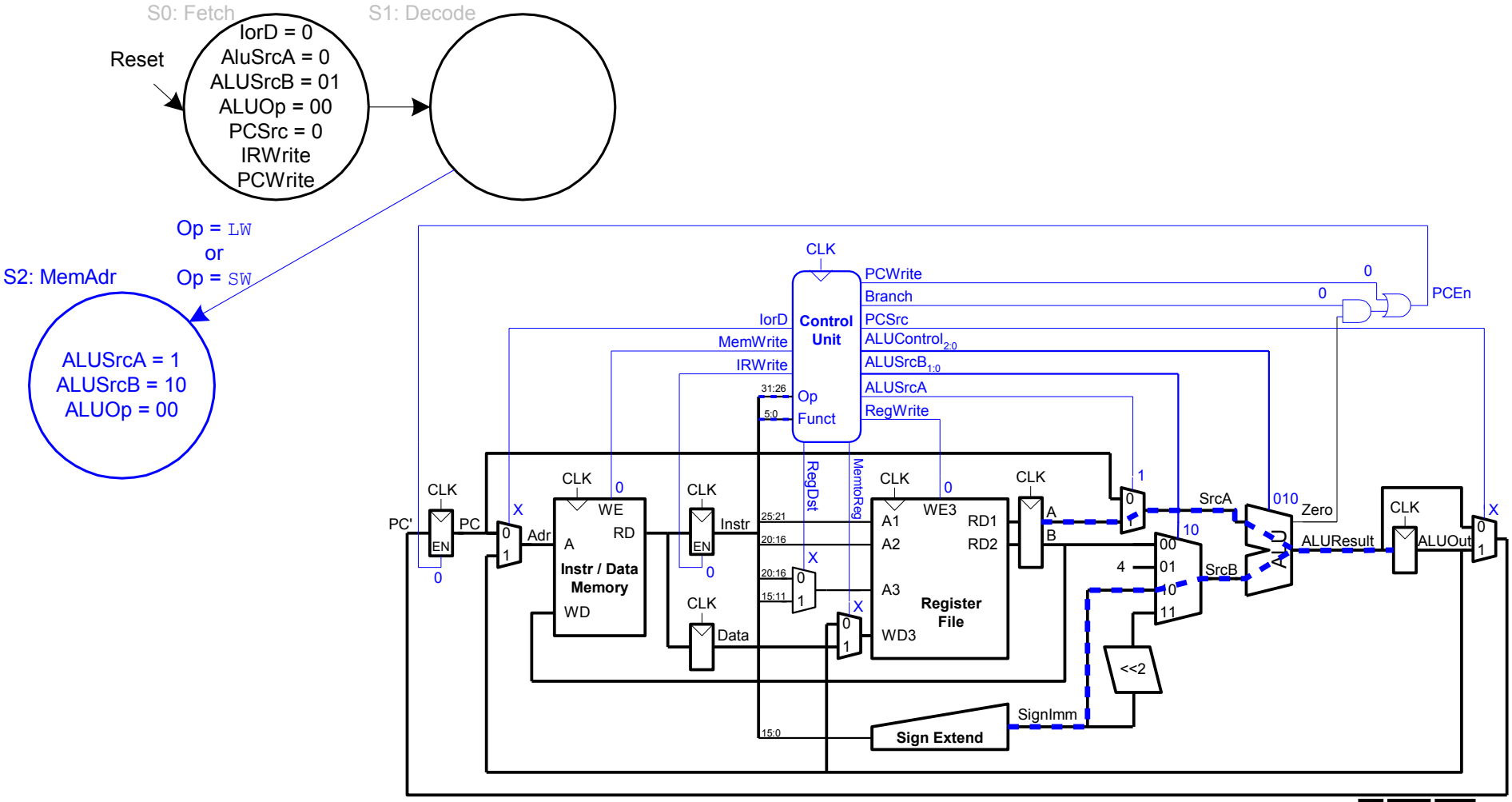




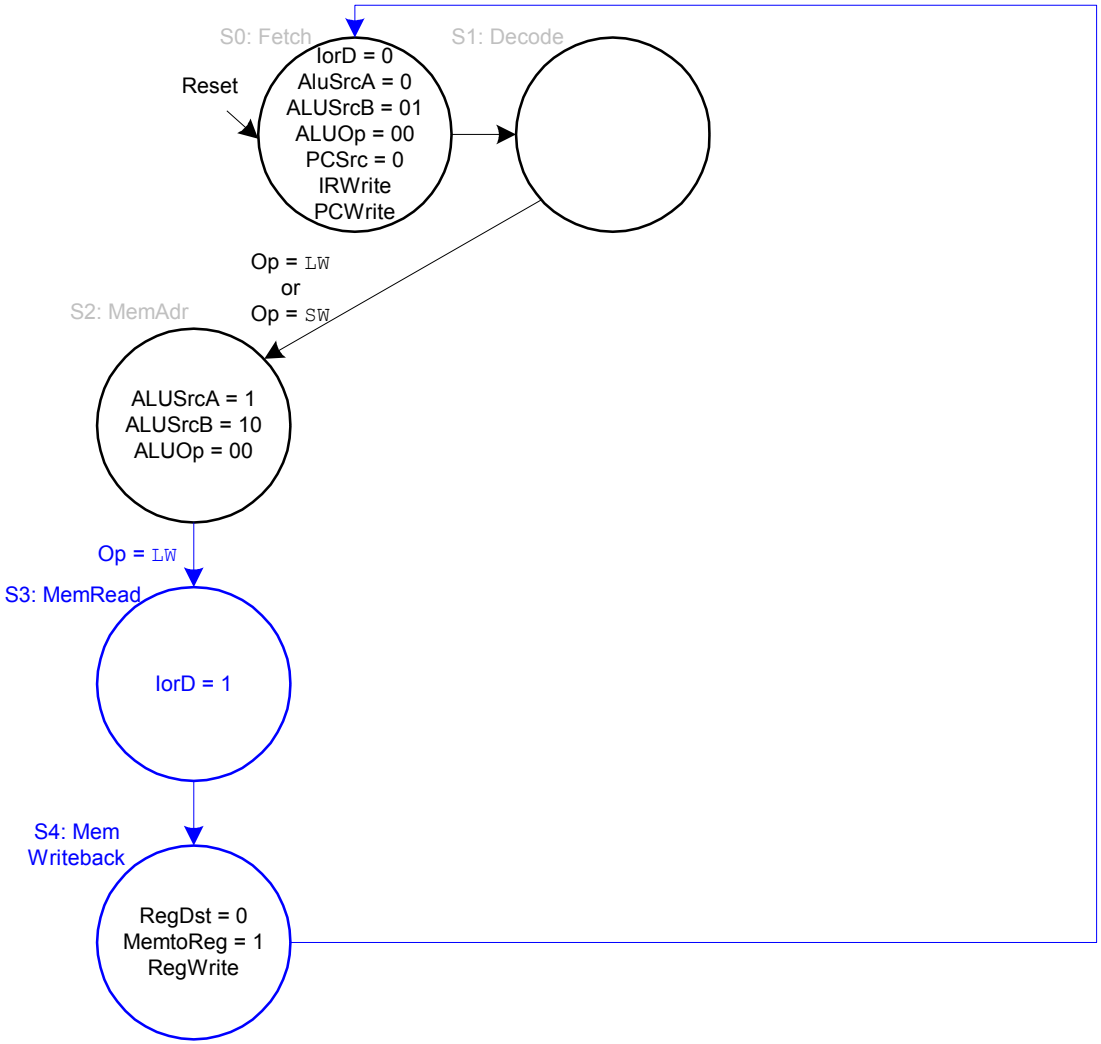
Hauptsteuerwerk: Adressberechnung



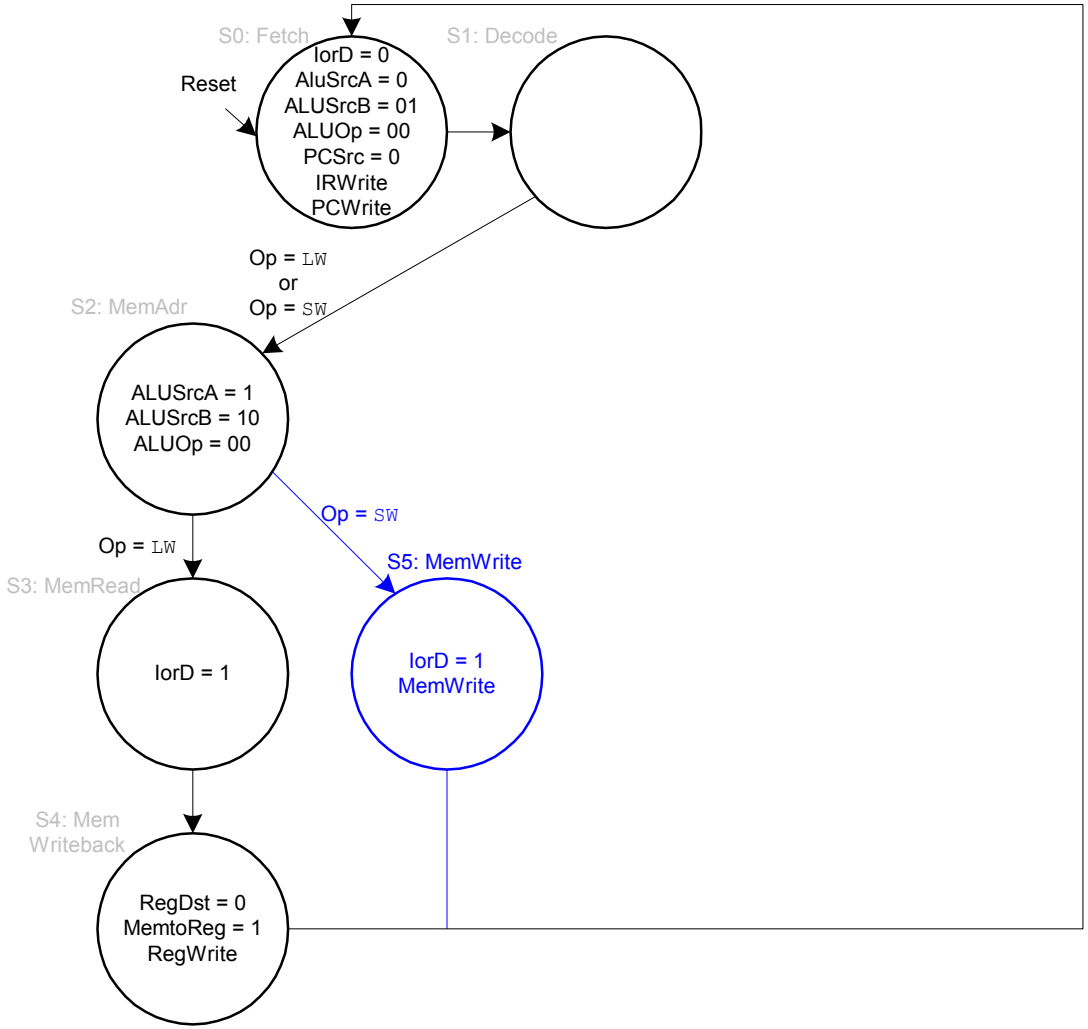
Hauptsteuerwerk: Adressberechnung



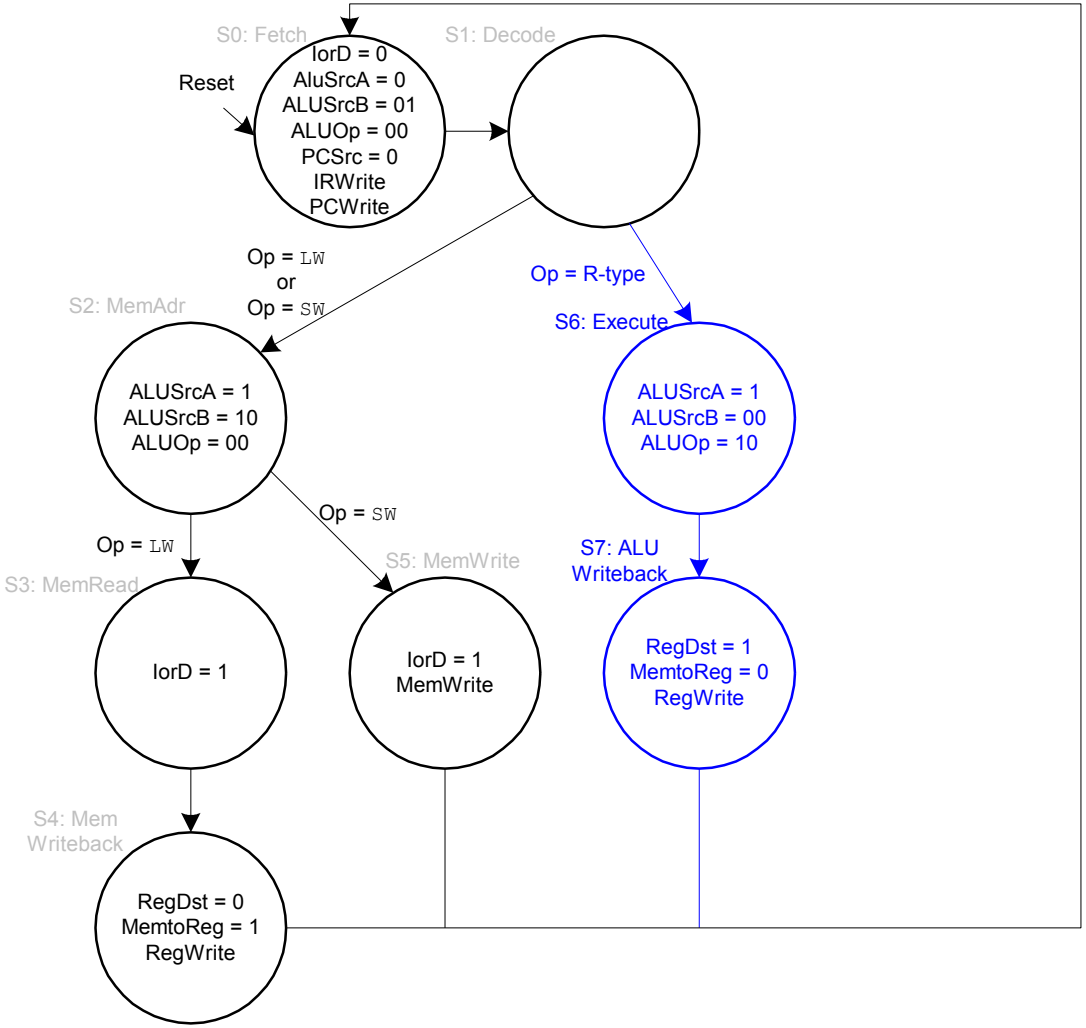
Hauptsteuerwerk: FSM für `lw`



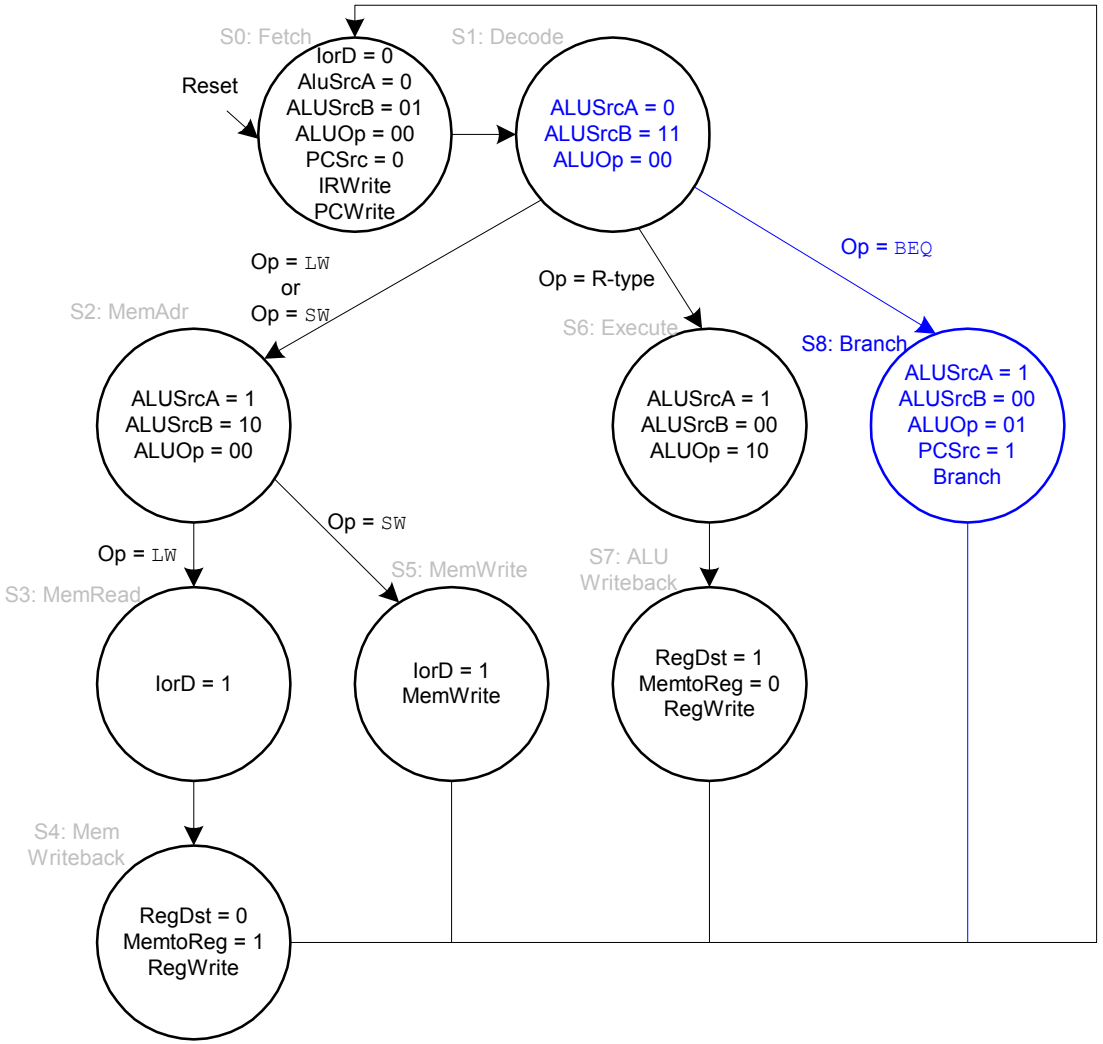
Hauptsteuerwerk: FSM für `sw`



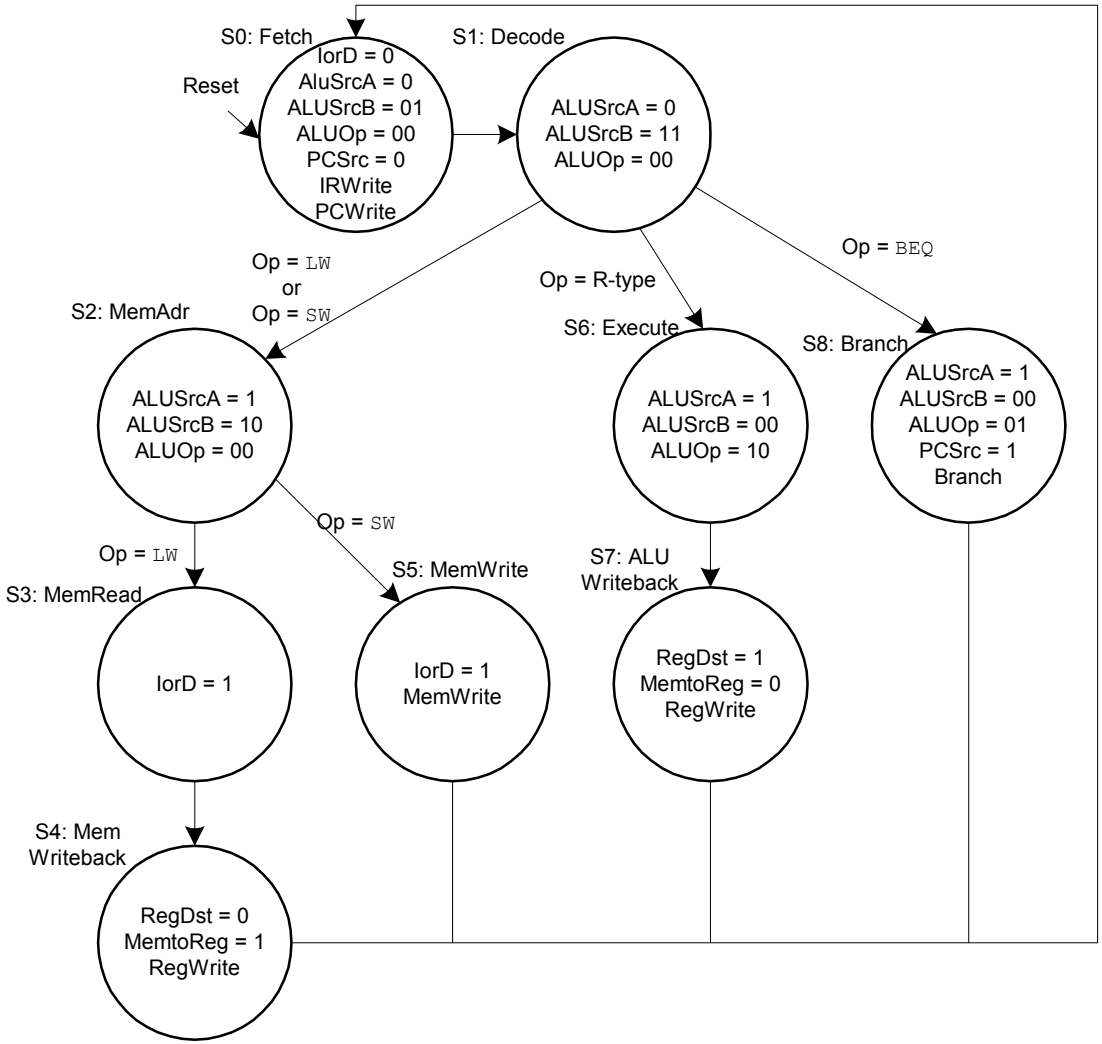
Hauptsteuerwerk: FSM für R-Typ



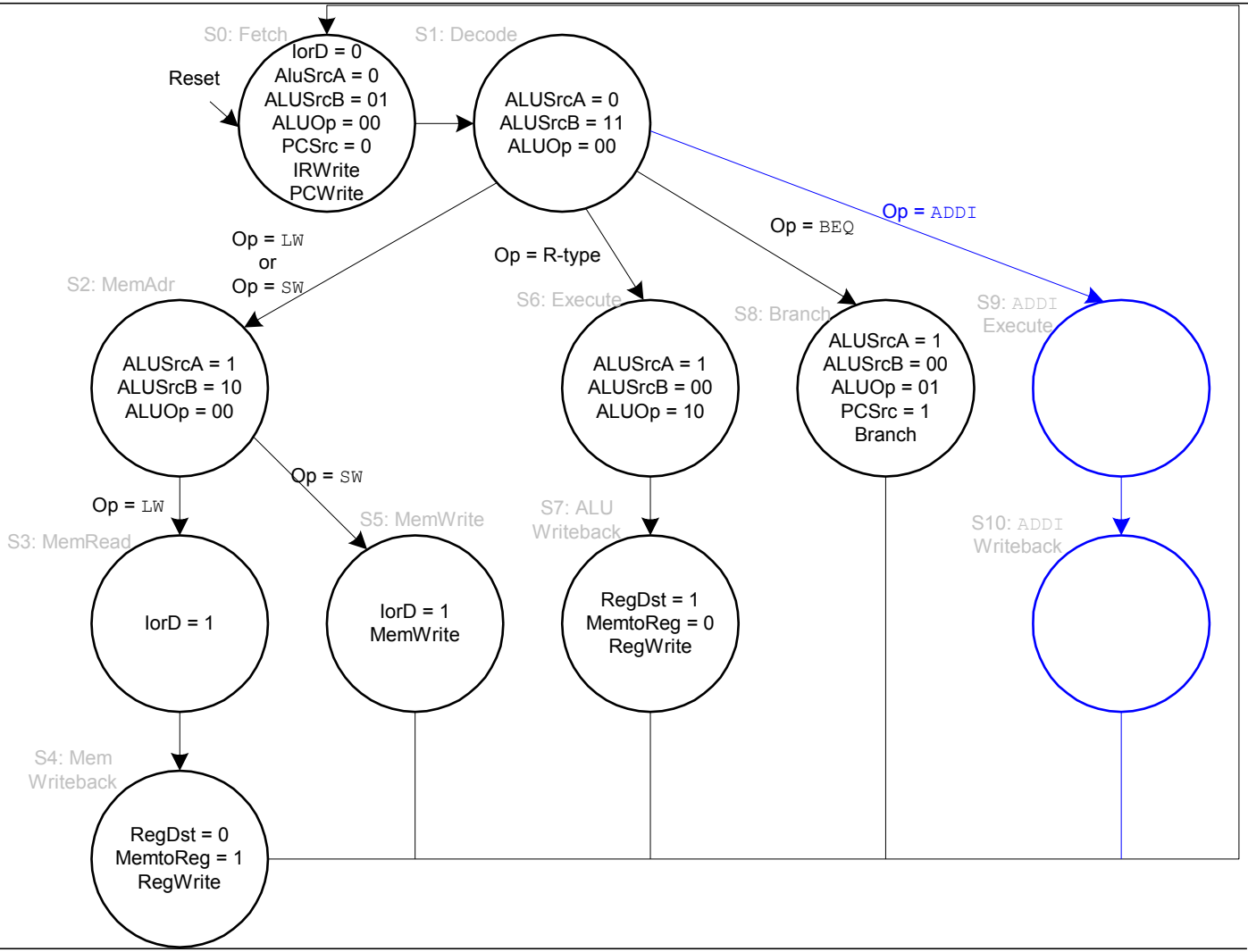
Hauptsteuerwerk: FSM für `beq`



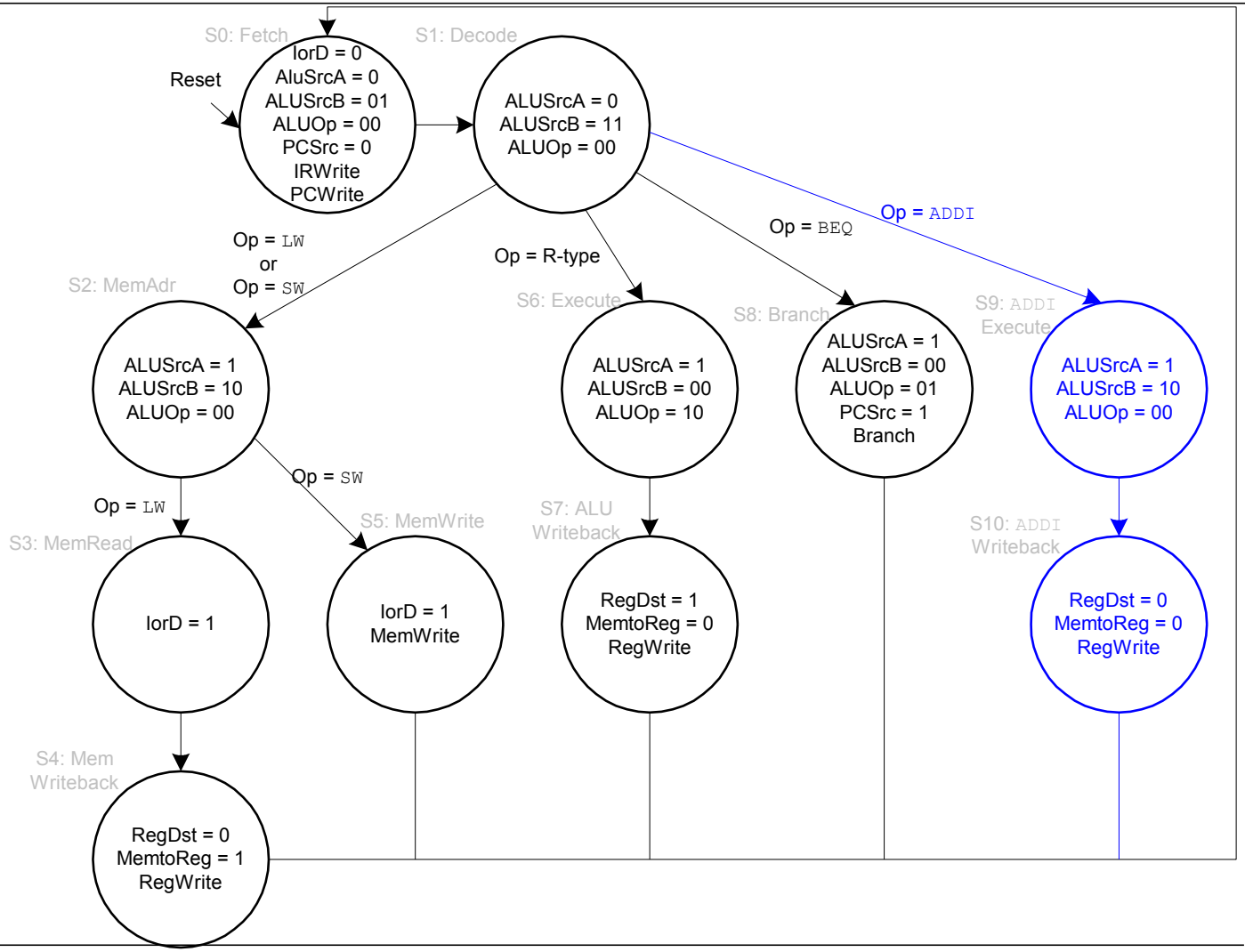
Vollständiges Hauptsteuerwerk für Mehrtakt-CPU



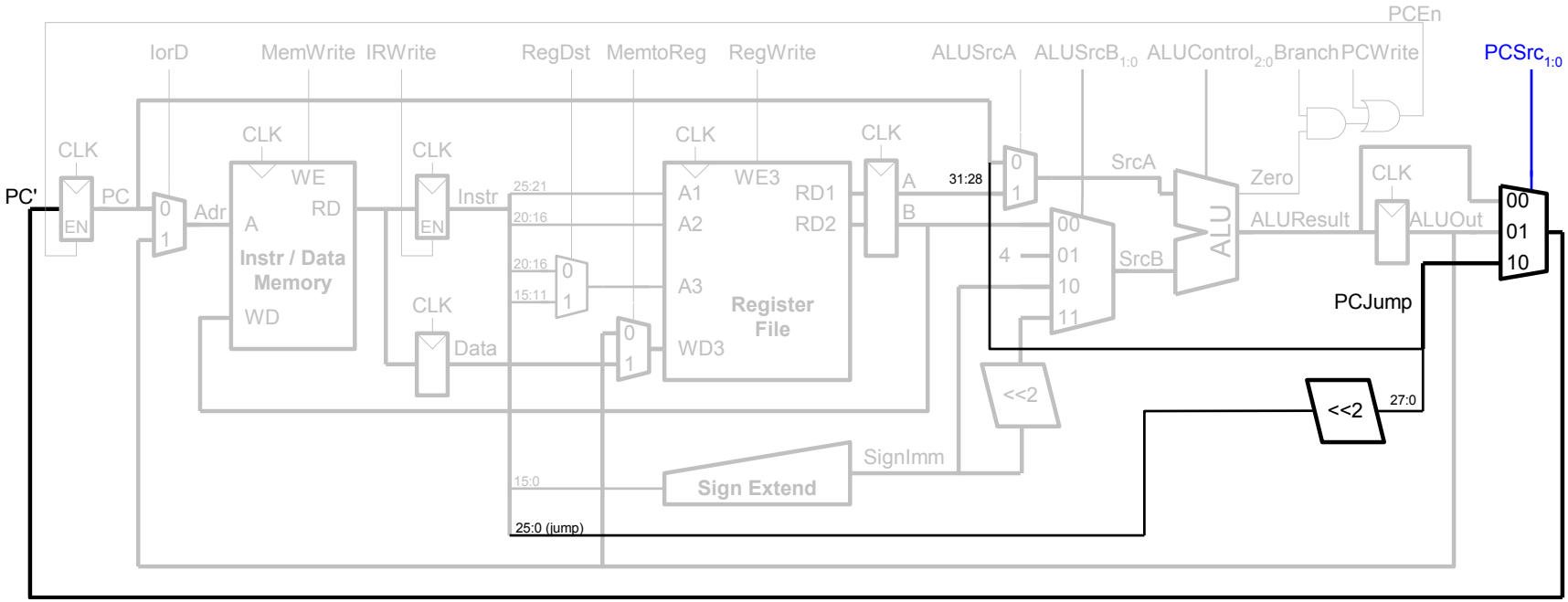
Erweiterung des Hauptsteuerwerks: `addi`



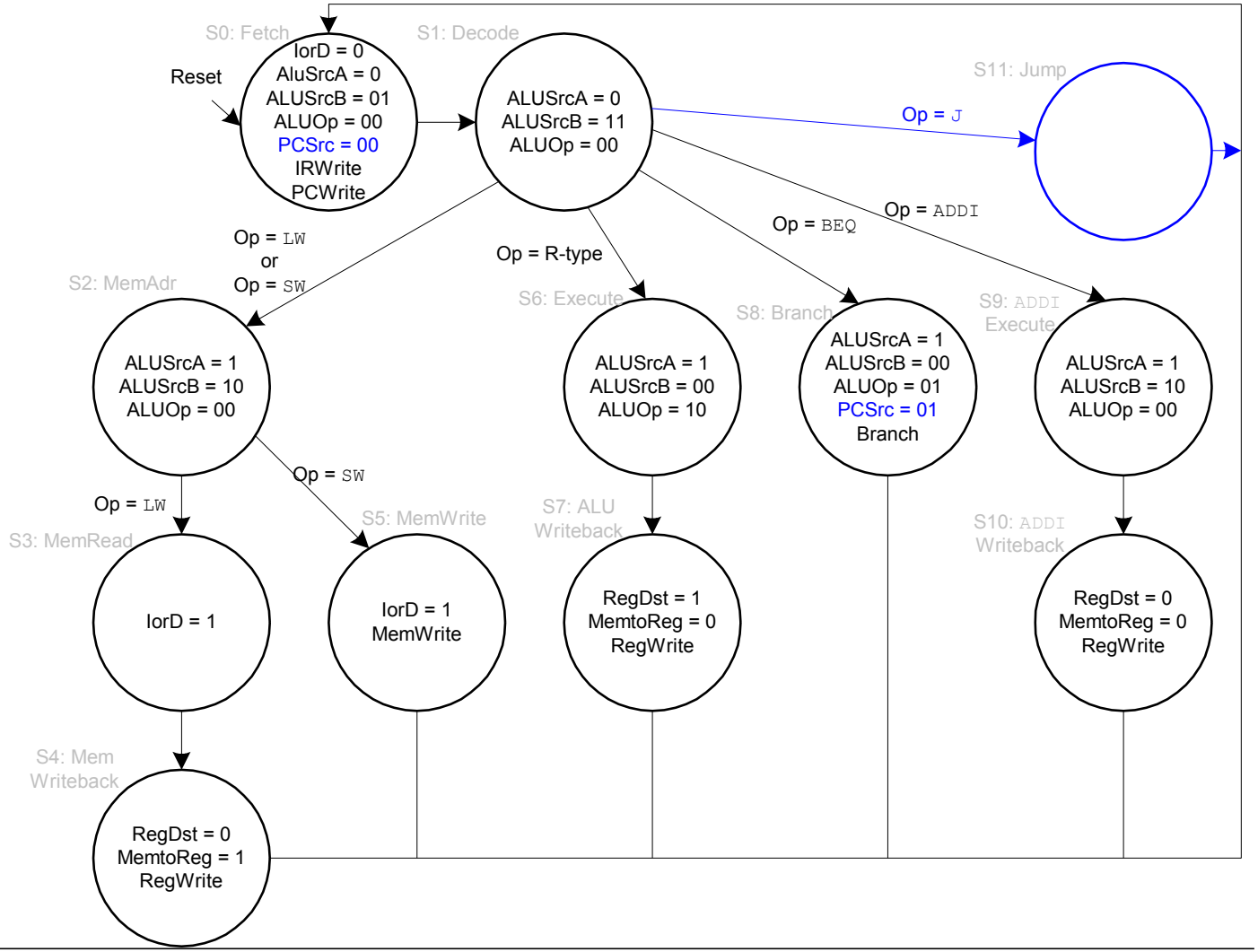
Erweiterung des Hauptsteuerwerks: `addi`



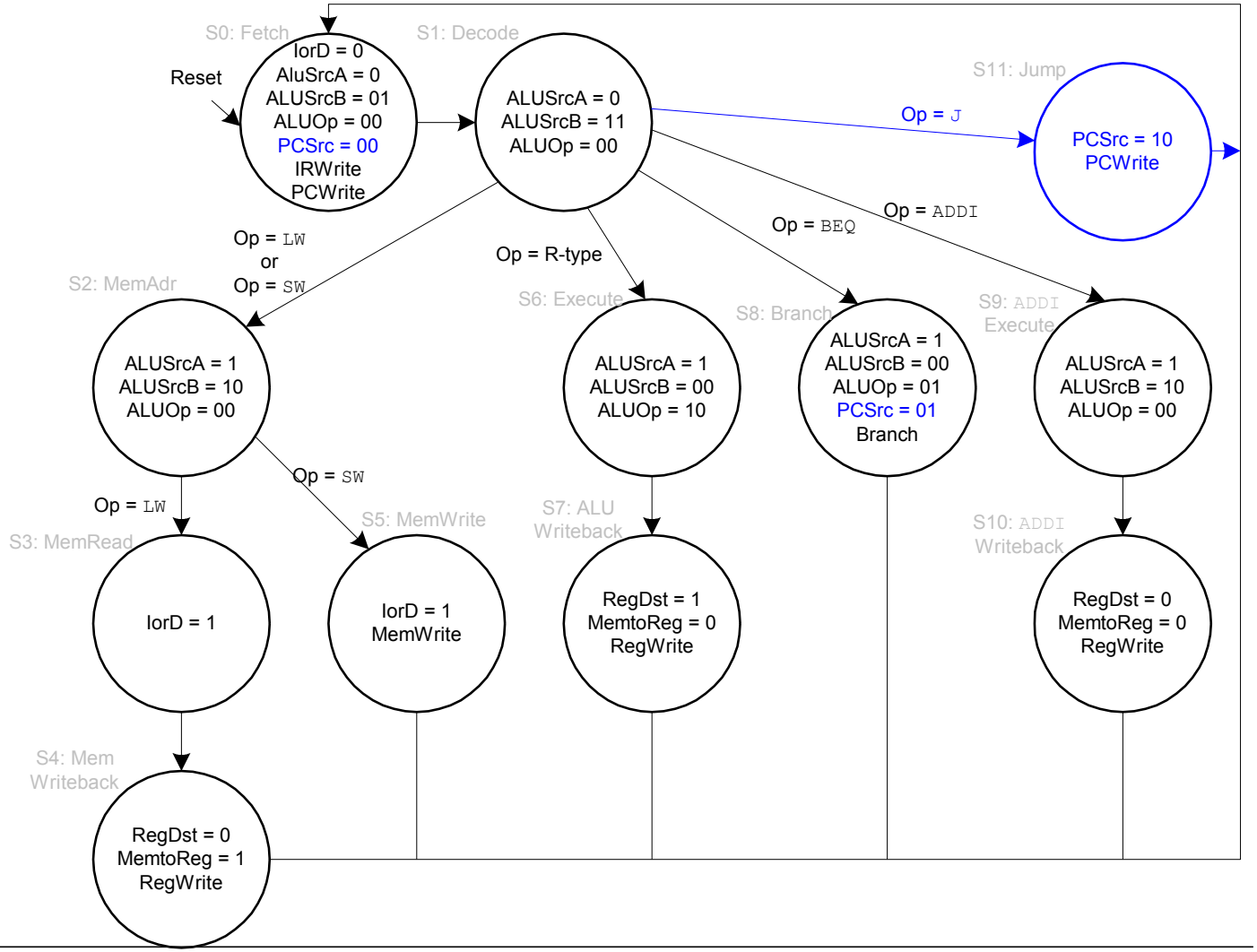
Erweiterung des Datenpfads für j



Erweiterung des Hauptsteuerwerks um j



Erweiterung des Hauptsteuerwerks um `j`



Rechenleistung des Mehrtaktprozessors

TECHNISCHE UNIVERSITÄT DARMSTADT

- Instruktionen benötigen unterschiedliche viele Takte:
  - 3 Takte : `beq`, `j`
  - 4 Takte : R-Typ, `sw`, `addi`
  - 5 Takte : `lw`

- CPI wird bestimmt als gewichteter Durchschnitt

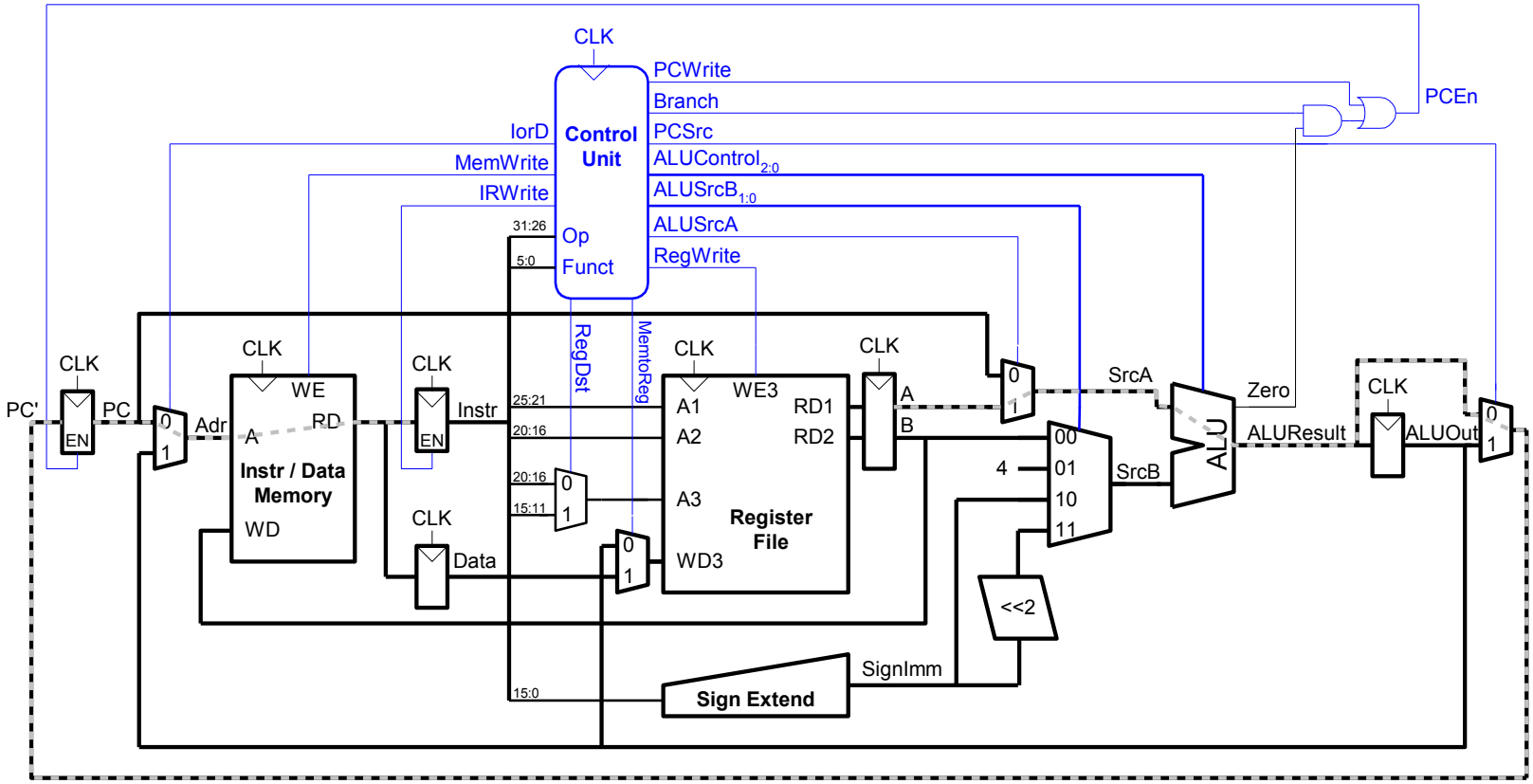
- SPECint 2000 Benchmark:
  - 25% Laden
  - 10% Speichern
  - 11% Verzweigungen
  - 2% Sprünge
  - 52% R-Typ

Durchschnittliche CPI = (0,11 + 0,02)(3) + (0,52 + 0,10)(4) + (0,25)(5) = 4,12

Rechenleistung des Mehrtaktprozessors

- Kritischer Pfad:

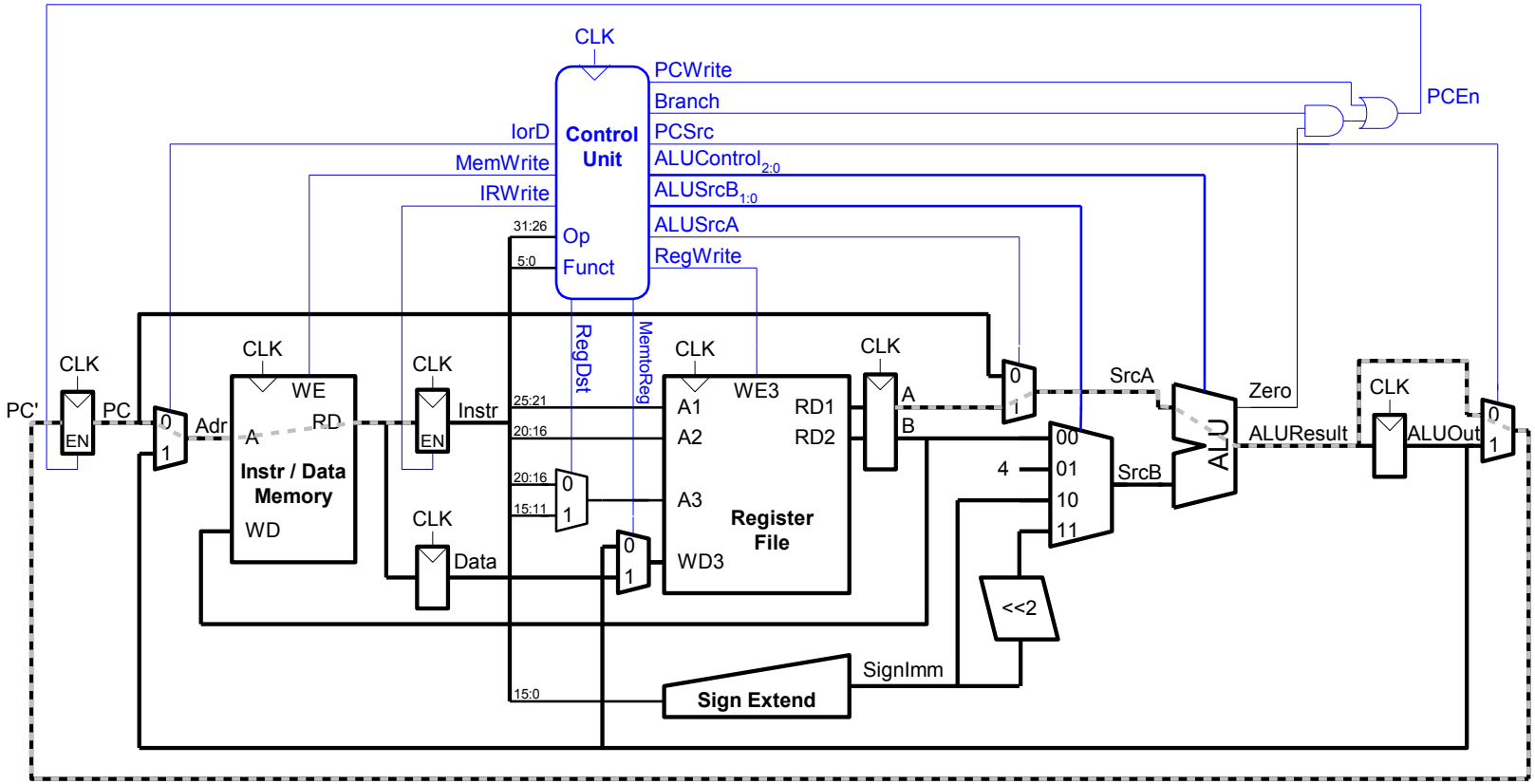
$T_c =$



Rechenleistung des Mehrtaktprozessors

- Kritischer Pfad :

$$T_c = t_{pcq} + t_{mux} + \max(t_{ALU} + t_{mux}, t_{mem}) + t_{setup}$$



Beispiel: Rechenleistung Mehrtaktprozessor

TECHNISCHE UNIVERSITÄT DARMSTADT

Element	Parameter	Verzögerung (ps)
Register Clock-to-Q	t_{pcq}	30
Register Setup	t_{setup}	20
Multiplexer	t_{mux}	25
ALU	t_{ALU}	200
Speicher Lesen	t_{mem}	250
Registerfeld Lesen	t_{RFread}	150
Registerfeld Setup	$t_{RFsetup}$	20

$T_c =$

Beispiel: Rechenleistung Mehrtaktprozessor

TECHNISCHE UNIVERSITÄT DARMSTADT

Element	Parameter	Verzögerung (ps)
Register Clock-to-Q	t_{pcq}	30
Register Setup	t_{setup}	20
Multiplexer	t_{mux}	25
ALU	t_{ALU}	200
Speicher Lesen	t_{mem}	250
Registerfeld Lesen	t_{RFread}	150
Registerfeld Setup	$t_{RFsetup}$	20

$$
\begin{aligned}
T_c &= t_{pcq_PC} + t_{mux} + \max(t_{ALU} + t_{mux}, t_{mem}) + t_{setup} \\
&= t_{pcq_PC} + t_{mux} + t_{mem} + t_{setup} \\
&= [30 + 25 + 250 + 20]\ \text{ps} \\
&= 325\ \text{ps}
\end{aligned}
$$

Beispiel: Rechenleistung Mehrtaktprozessor

TECHNISCHE UNIVERSITÄT DARMSTADT

- Führe Programm mit 100 Milliarden Instruktionen auf Mehrtaktprozessor aus
  - CPI = 4,12
  - T_c = 325 ps

Ausführungszeit =

Beispiel: Rechenleistung Mehrtaktprozessor

TECHNISCHE UNIVERSITÄT DARMSTADT

- Führe Programm mit 100 Milliarden Instruktionen auf Mehrtaktprozessor aus
  - CPI = 4,12
  - T_c = 325 ps

Ausführungszeit = (# Instruktionen) × CPI × T_c

$= (100 \times 10^9)\ (4{,}12)\ (325 \times 10^{-12})$

= 133,9 Sekunden

- Langsamer als Ein-Takt-Prozessor (brauchte 92,5 Sekunden).
  - Warum?

LEHRE WIKI TEST IN DREI FOLIEN

Beispiel: Rechenleistung Mehrtaktprozessor

TECHNISCHE UNIVERSITÄT DARMSTADT

- Führe Programm mit 100 Milliarden Instruktionen auf Mehrtaktprozessor aus
  - CPI = 4,12
  - T_c = 325 ps

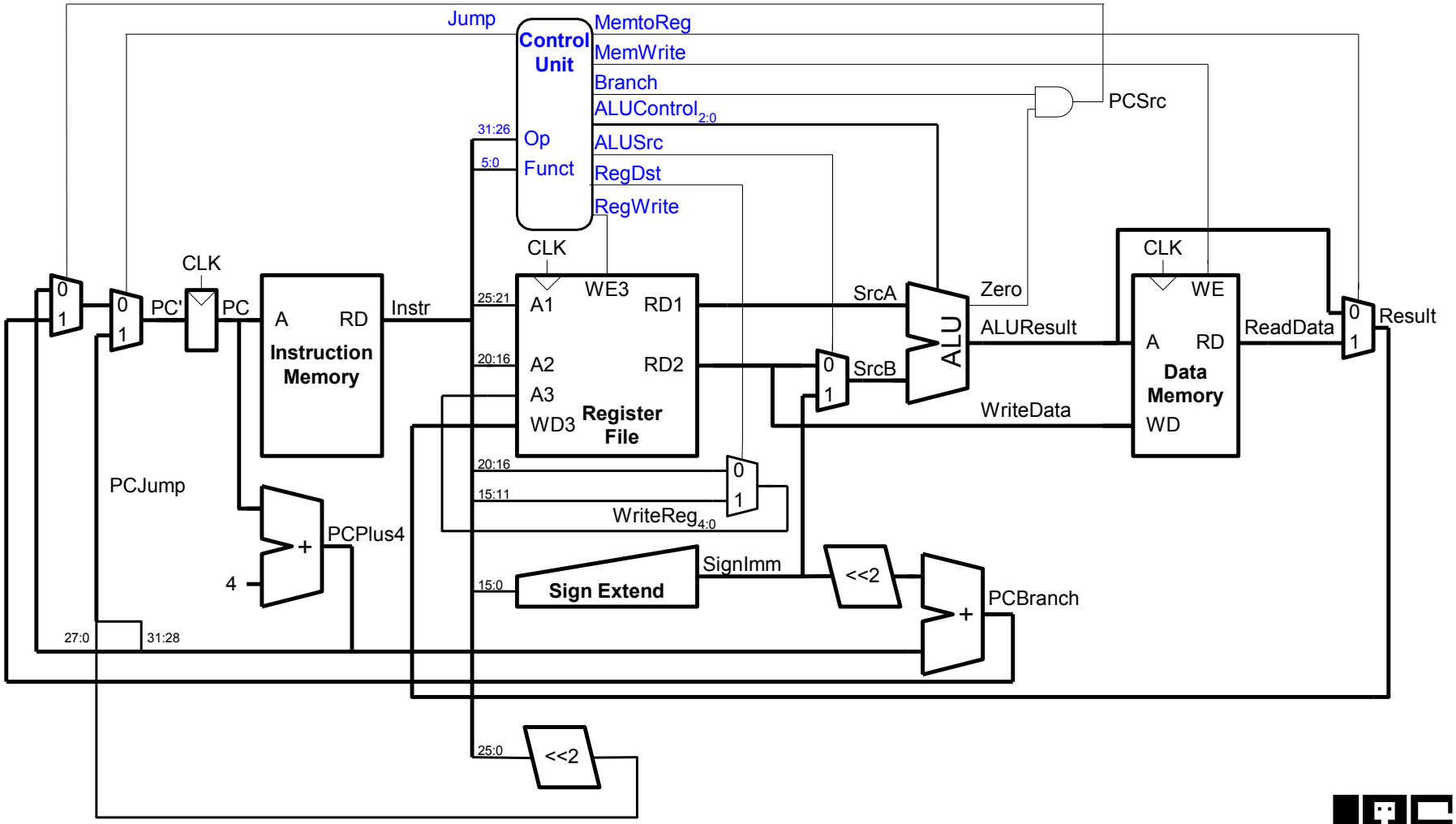
Ausführungszeit = (# Instruktionen) × CPI × T_c

$= (100 \times 10^9)\ (4{,}12)\ (325 \times 10^{-12})$

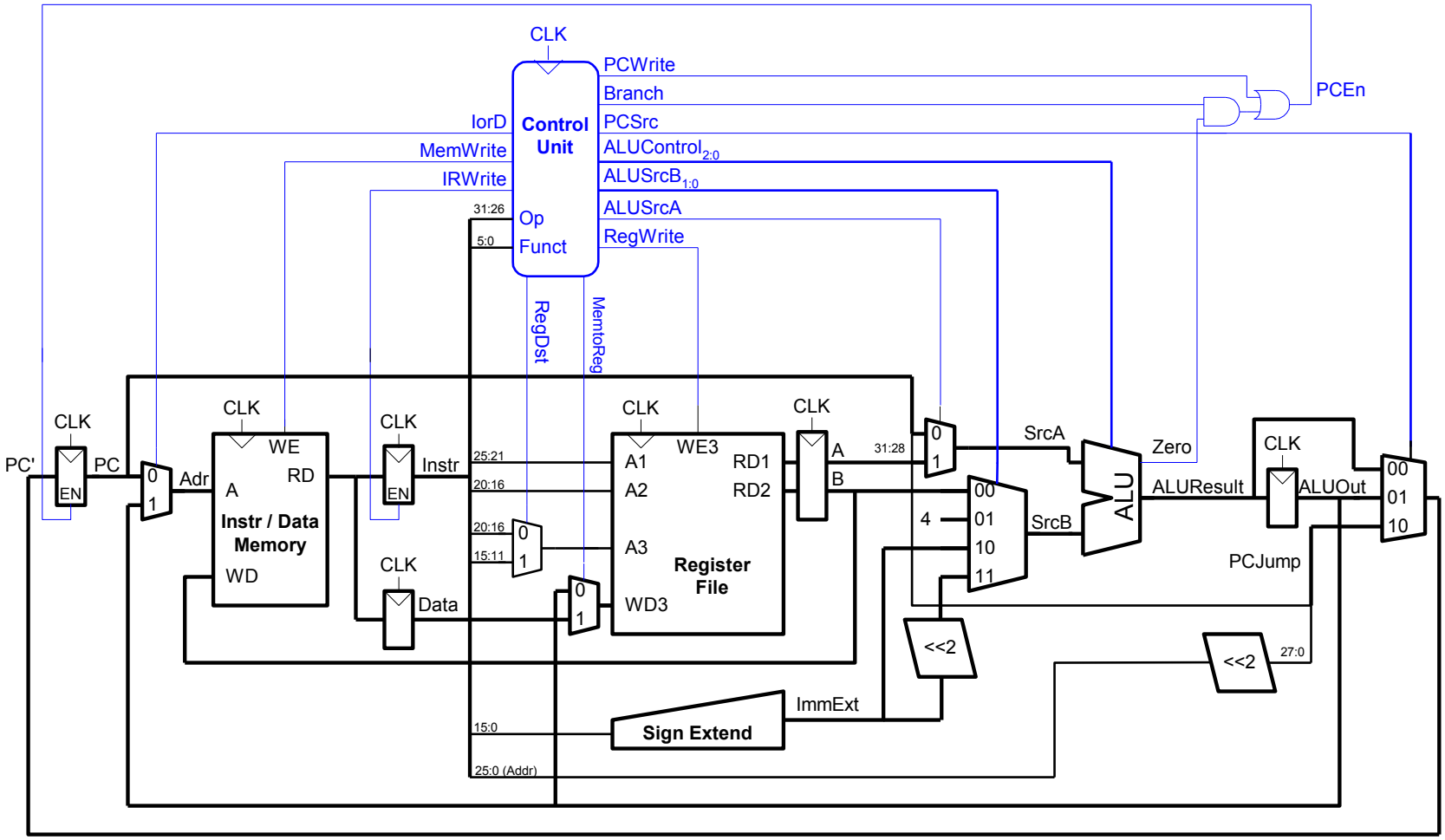
= 133,9 Sekunden

- Langsamer als Ein-Takt-Prozessor (brauchte 92,5 Sekunden).
  - Unterschiedlich lange Anzahl von Ausführungstakten (bis zu 5 für `lw`)
    - Aber nicht 5x schnellere Taktfrequenz
  - Nun zusätzliche Verzögerungen für sequentielle Logik mehrfach je Befehl
    - $t_{pcq} + t_{setup} = 50$ ps

Rückblick: Ein-Takt MIPS Prozessor



Rückblick: Mehrtakt-MIPS-Prozessor



Potentiell etwas kleiner.

LEHRE WIKI FRAGE

- Bitte jetzt auf LEHRE WIKI eine Frage beantworten!

MIPS Prozessor mit Pipelining

TECHNISCHE UNIVERSITÄT DARMSTADT

- Zeitliche Parallelität

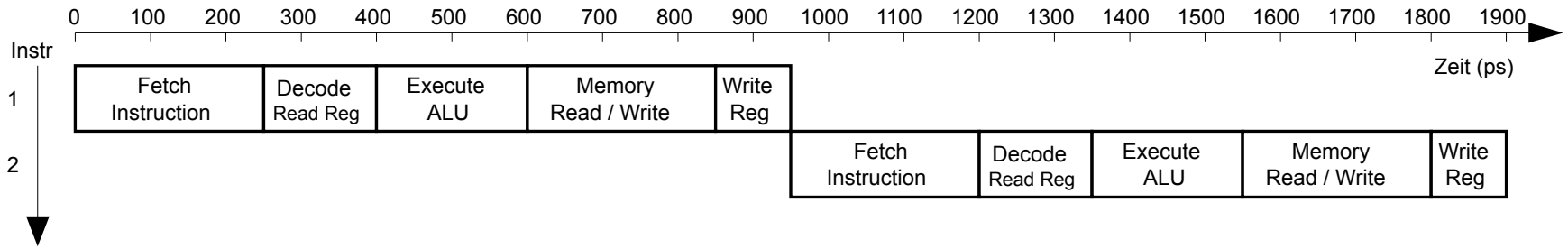
- Teile Ablauf im Ein-Takt-Prozessor in fünf Stufen:
  - Hole Instruktion (*Fetch*)
  - Dekodiere Bedeutung von Instruktion (*Decode*)
  - Führe Instruktion aus (*Execute*)
  - Greife auf Speicher zu (*Memory*)
  - Schreibe Ergebnisse zurück (*Writeback*)

- Füge Pipeline-Register zwischen den Stufen ein

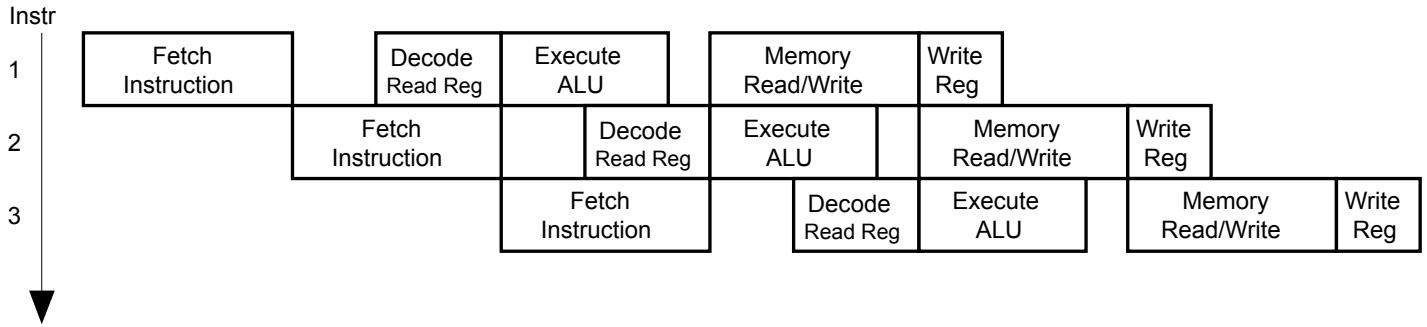
Rechenleistung: Ein-Takt und Pipelined

TECHNISCHE UNIVERSITÄT DARMSTADT

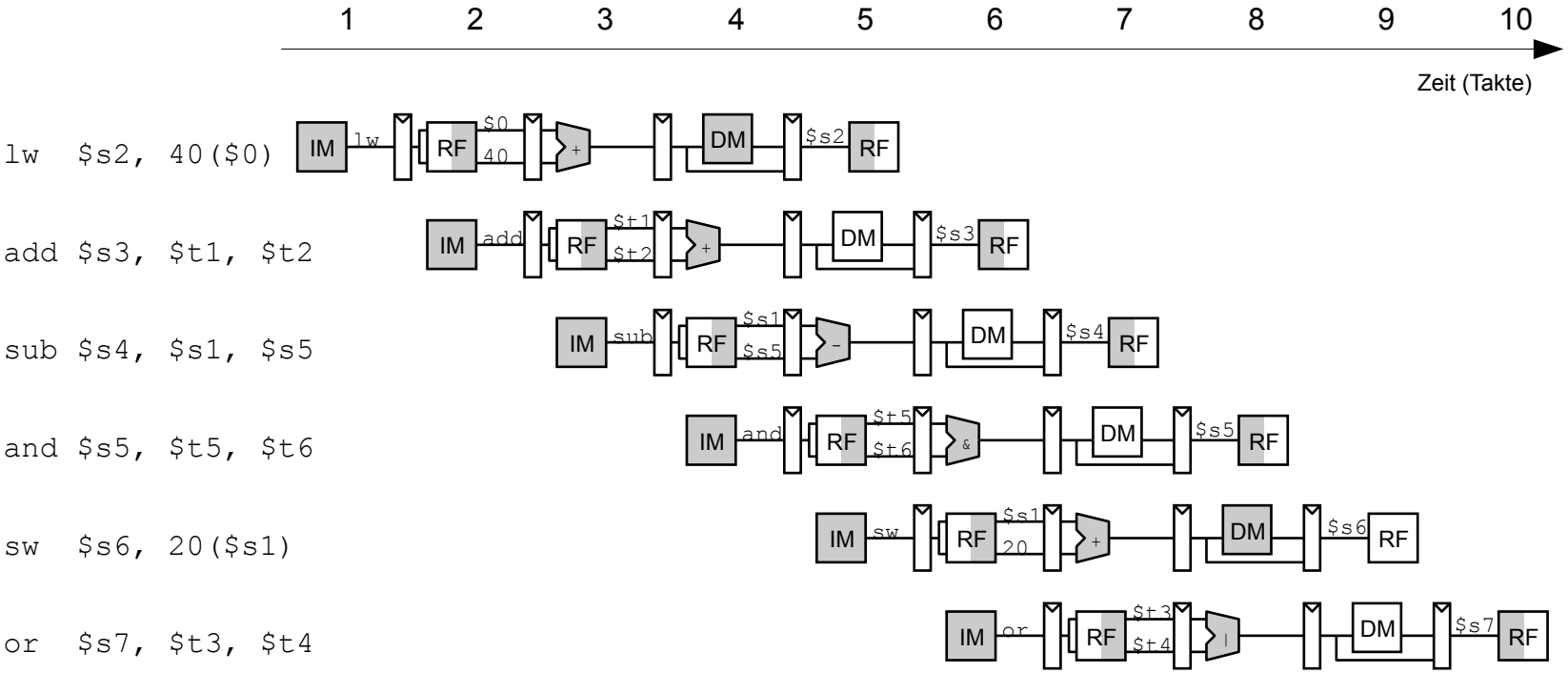
Ein-Takt



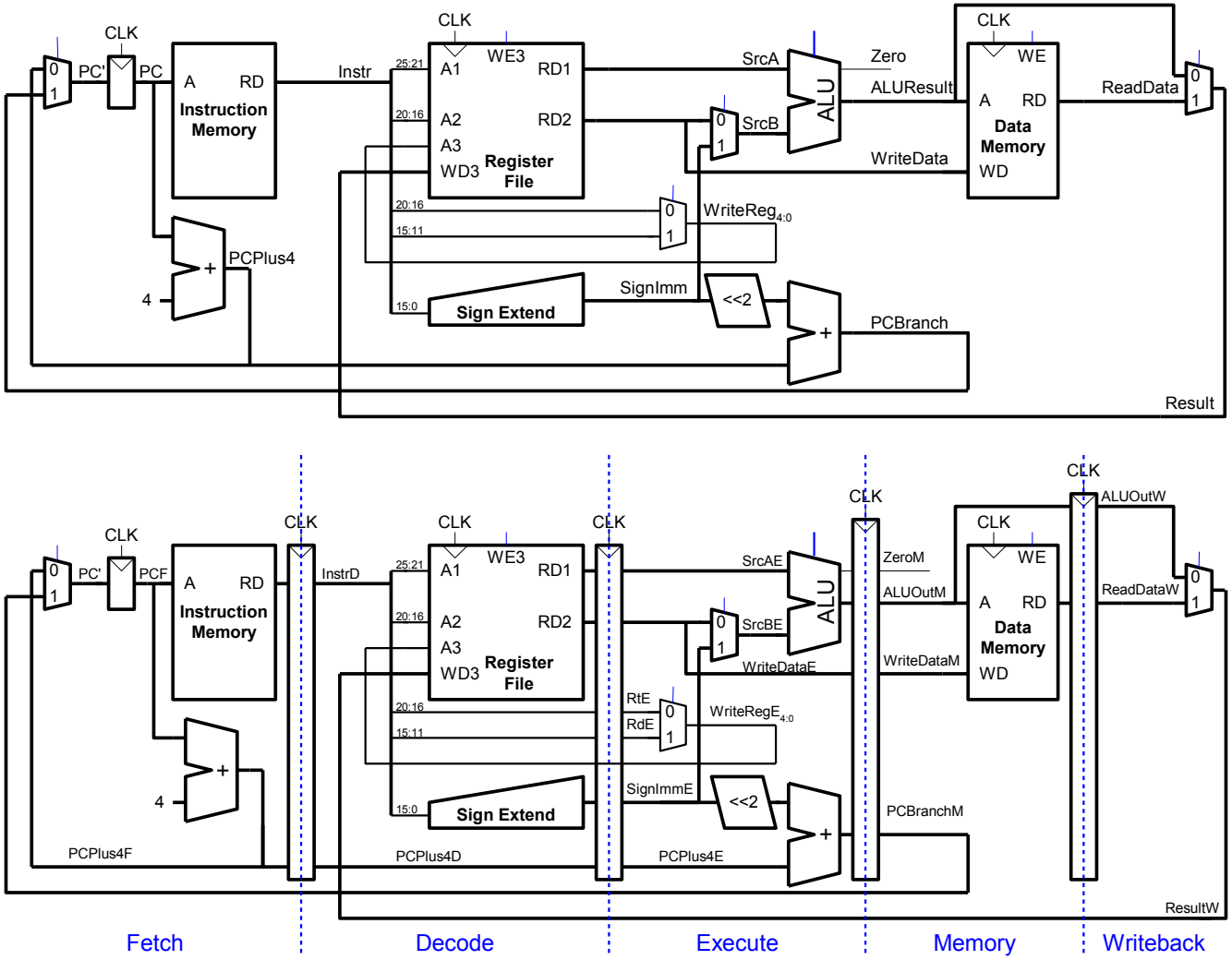
Pipelined



Abstraktere Darstellung des Pipelinings

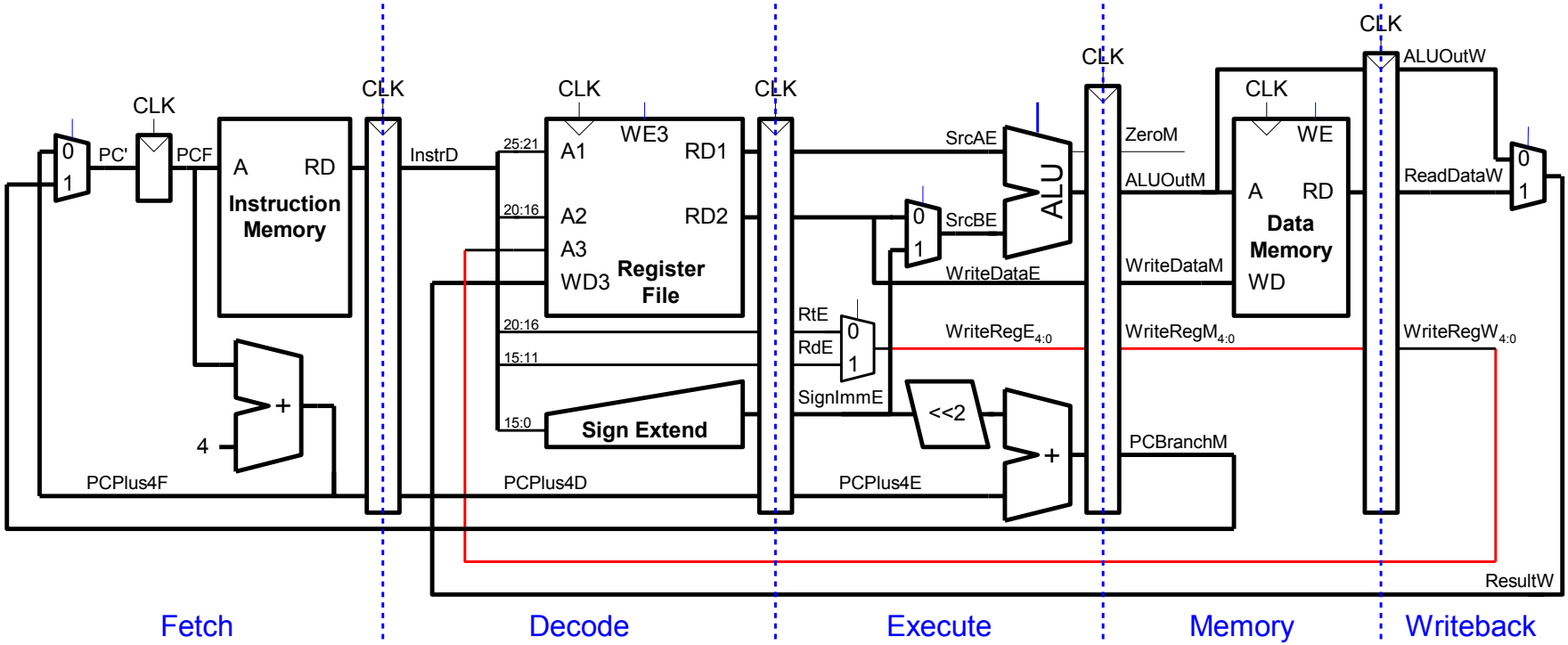


Ein-Takt- und Pipelined-Datenpfad

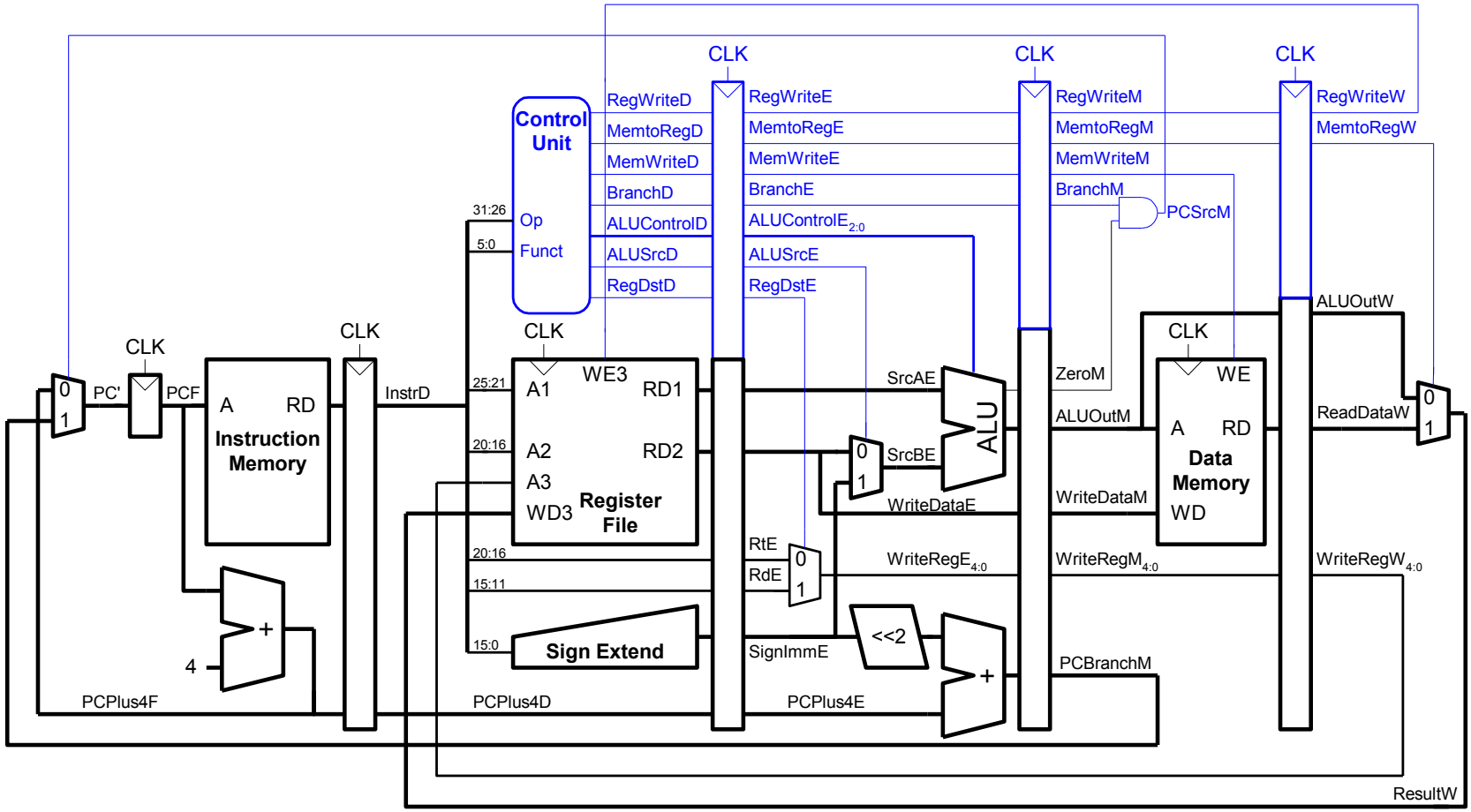


Korrigierter Pipelined-Datenpfad

- WriteReg muss zur gleichen Zeit am Registerfeld ankommen wie Result



Steuersignale für Pipelined-Datenpfad



Identisch zu Ein-Takt-Steuerwerk, aber Signale verzögert über Pipeline-Stufen

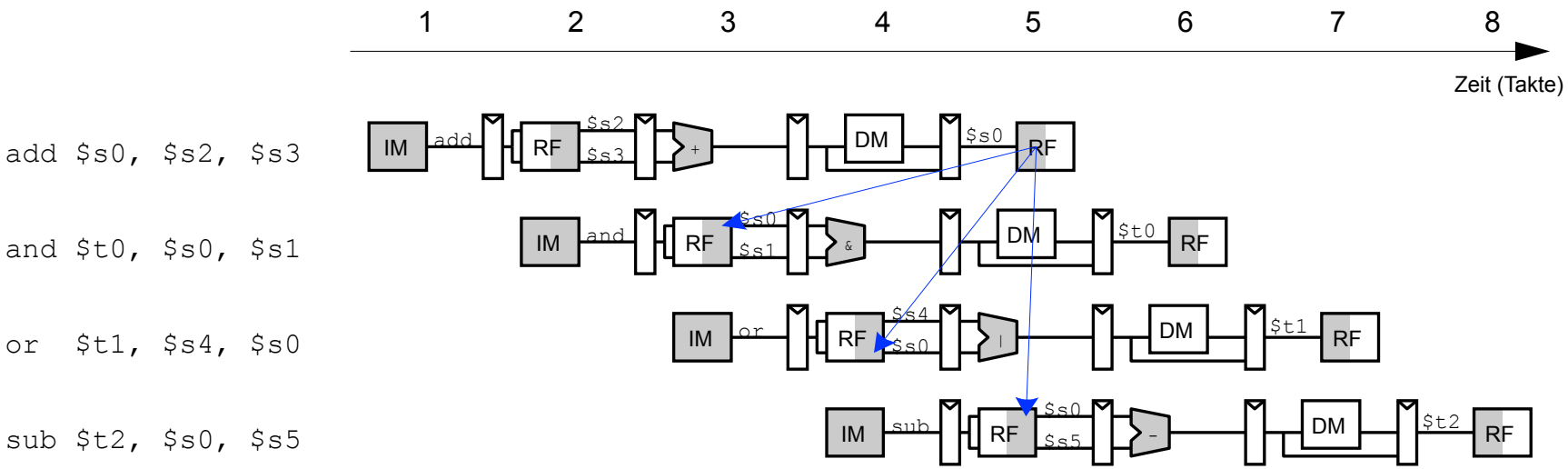
Abhängigkeiten zwischen Pipeline-Stufen (*hazards*)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Treten auf wenn eine
  - Instruktion vom Ergebnis einer vorhergehenden abhängt
  - … diese aber noch kein Ergebnis geliefert hat

- Arten von Hazards
  - **Data Hazard:** z.B. Neuer Wert von Register noch nicht in Registerfeld eingetragen
  - **Control Hazard:** Unklar welche Instruktion als nächstes ausgeführt werden muss
    - Tritt bei Verzweigungen auf

Data Hazard



Hier: Read-after-Write Hazard (RAW)
- $s0 „muss vor Lesen geschrieben werden“

Umgang mit Data Hazards

TECHNISCHE UNIVERSITÄT DARMSTADT

- Plane Wartezeiten von Anfang an ein
  - Füge `nop`s zur Compile-Zeit ein
  - *scheduling*
- Stelle Maschinencode zur Compile-Zeit um
  - *scheduling / reordering*
- Leite Daten zur Laufzeit schneller über Abkürzungen weiter
  - *bypassing / forwarding*
- Halte Prozessor zur Laufzeit an bis Daten da sind
  - *stalling*