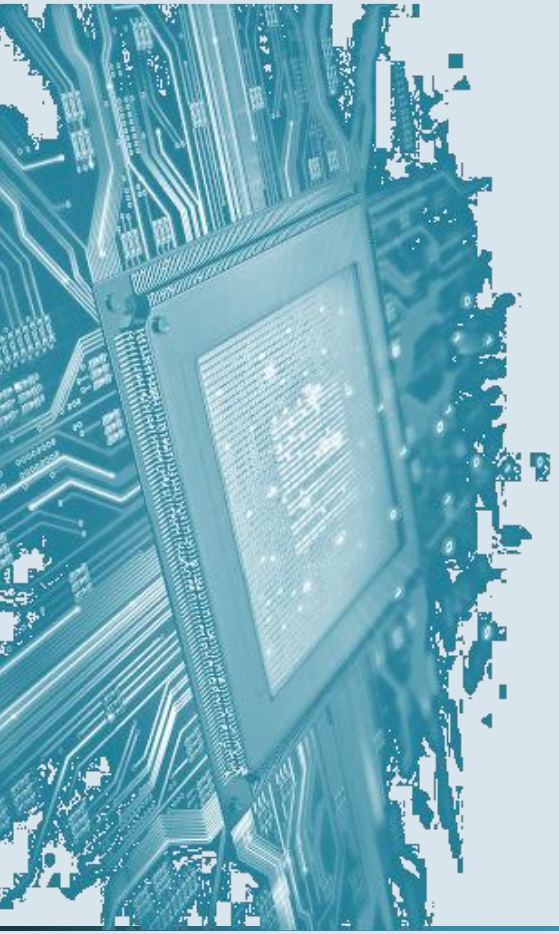


Rechnerarchitektur (AIN 2)

SoSe 2021

Kapitel 4

Multi-Cycle CPU – Pipeline-Architekturen

Prof. Dr.-Ing. Michael Blaich
mblaich@htwg-konstanz.de

Kapitel 4 - Gliederung

Kapitel 4: Multi-Cycle CPU – Pipeline-Architekturen

4.1 Prinzip einer Pipeline

4.2 Datapath der MIPS Pipeline

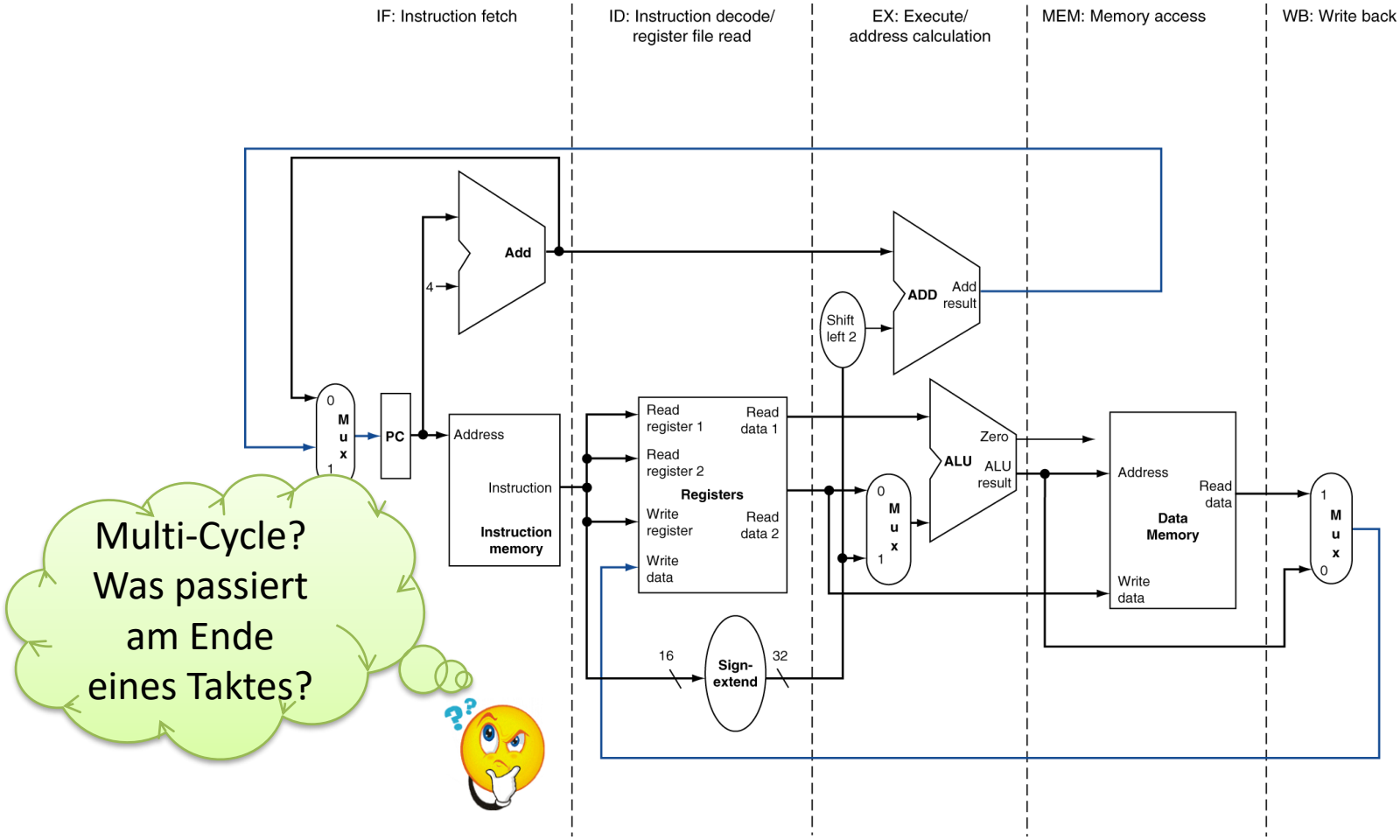
4.3 Control der MIPS Pipeline

4.4 Hazards

4.5 Exceptions

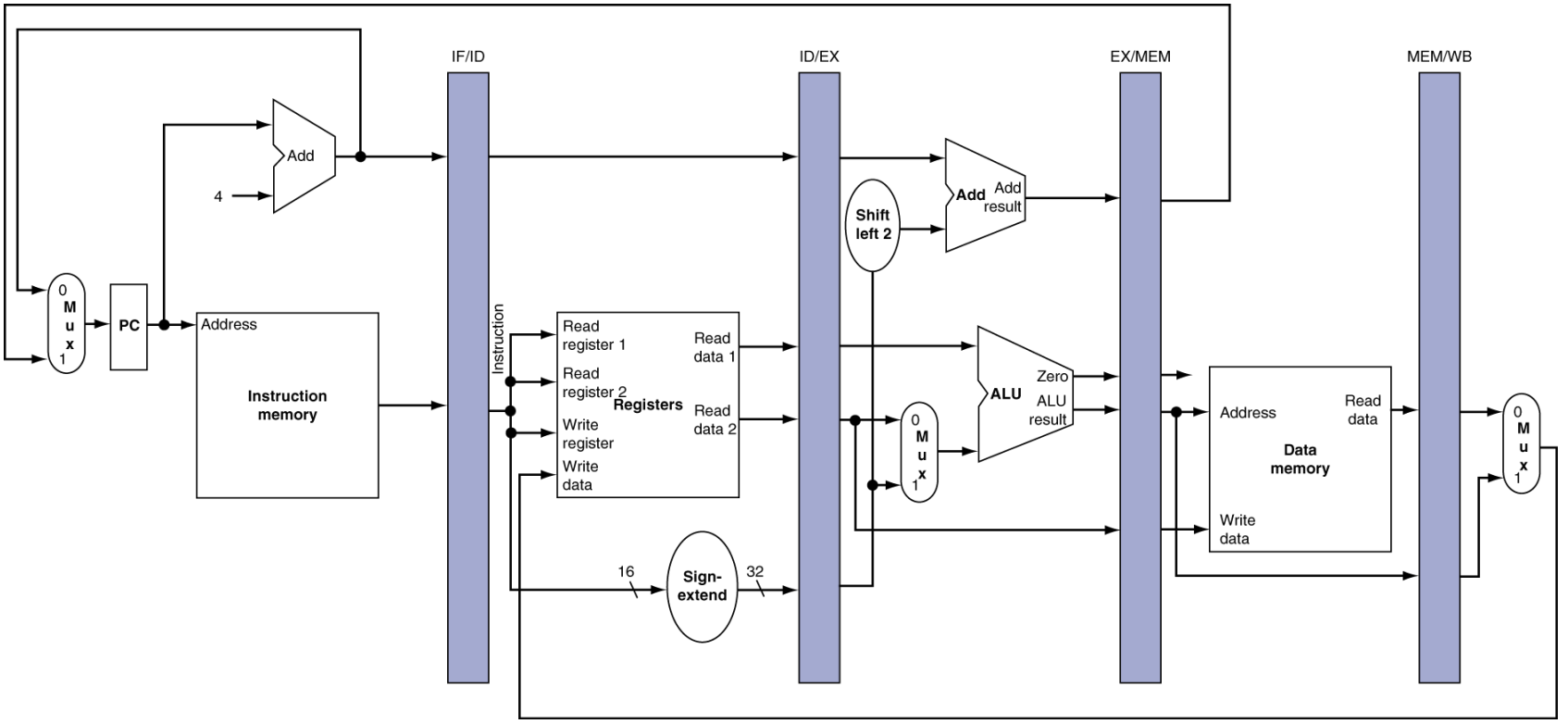
Single- und Multi-Cycle CPU

In der Single-Cycle-CPU wurde die gesamte CPU in einem Takt berechnet. In der Multi-Cycle CPU werden Instruktion in mehreren Takten berechnet, ein Stage pro Takt. Idealerweise sollten alle Stages die Berechnung in der gleichen Zeit durchführen. Die Grafik zeigt die Aufteilung der Hardware im Datenpfad der Single-Cycle-CPU auf die 5 Stages IF, ID, EX, MEM und WB.



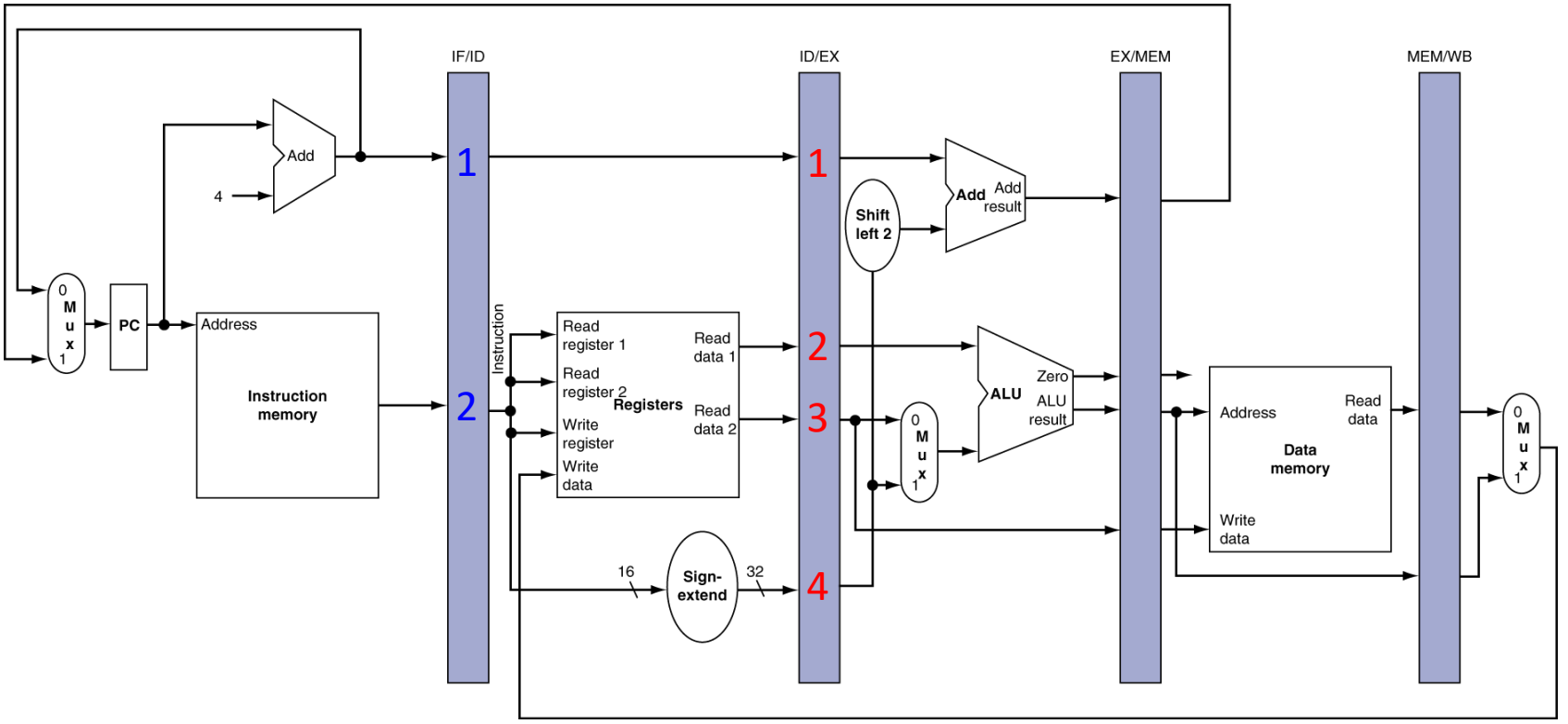
Pipeline-Register

- Die (Zwischen-) Ergebnisse einer Pipeline-Stage werden am Ende des Taktes in **Pipeline-Registern** gespeichert und dienen im nächsten Takt als Eingänge zur Berechnung der nächsten Stage.
- Es gibt 4 Pipeline-Register, die nach den beiden Stages zwischen denen sie liegen benannt sind:
  - IF/ID, ID/EX, EX/MEM, MEM/WB



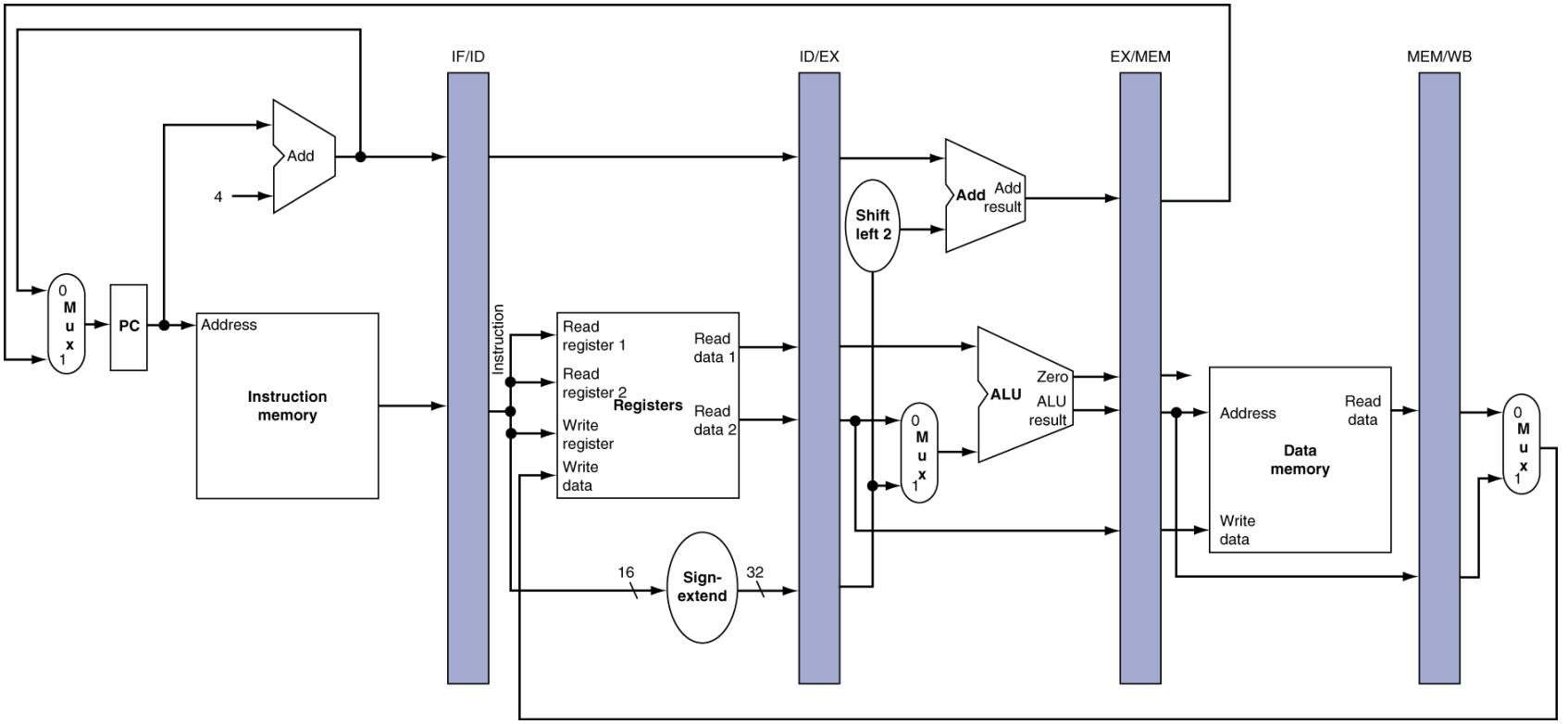
Beispiel: Pipeline-Register der ID-Stage

- Die ID-Stage arbeitet mit den Werten, die im IF/ID-Register gespeichert sind:
  - nächster Program Counter (1) und 32-Bit-Instruktion (2)
- Am Ende des Taktes werden die in der ID-Stage berechneten Werte im ID/EX-Register gespeichert:
  - nächster Program Counter (1), Registeroperand 1 (2), Registeroperand 2 (3), Konstante (4)).
- Im nächsten Takt werden die Werte aus dem ID/EX-Register dann in für die EX-Stage der Instruktion verwendet.



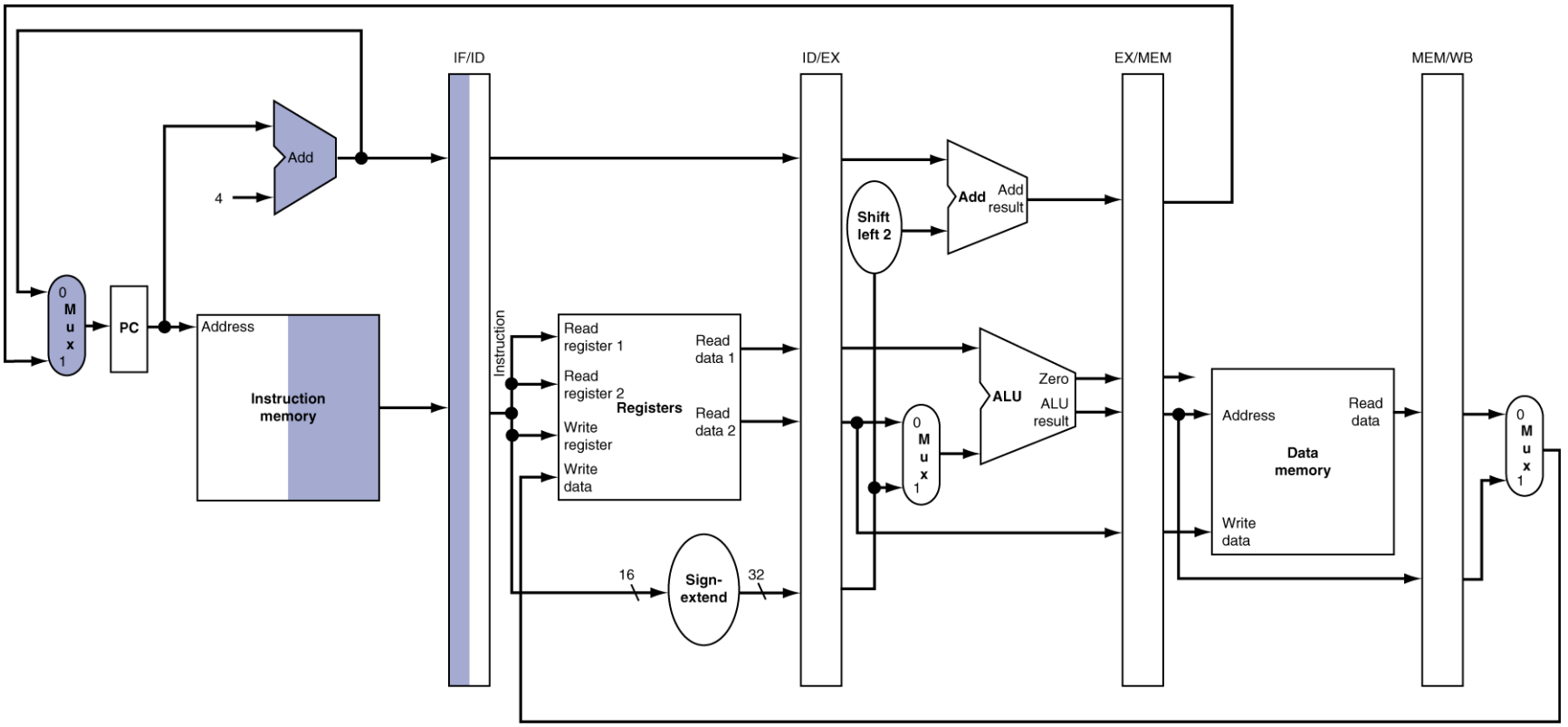
Pipeline-Register

Nutzung der Pipeline-Register am Beispiel von "Load" und "Store"



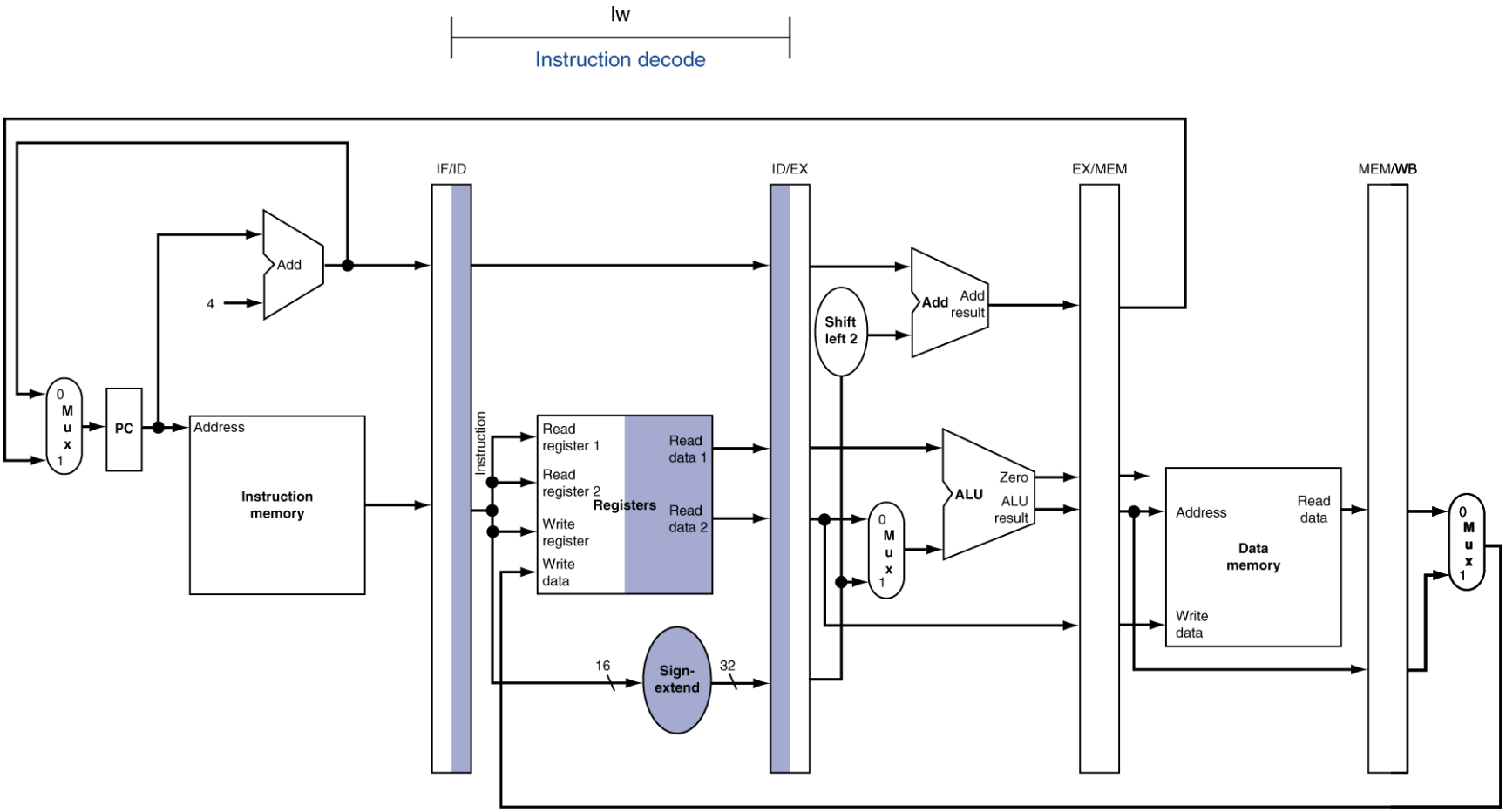
Stage 1 (IF) für load, store: Instruktion holen

- Instruktion an Speicheradresse in PC wird aus dem Speicher geladen und in IF/ID geschrieben
- PC wird um 4 erhöht und in IF/ID geschrieben
  - PC der Instruktion wird gespeichert
- Adresse der nächsten Instruktion (PC+4, branch, jump) wird in PC geschrieben
  - PC + 4: diese Instruktion
  - branch, jump: frühere Instruktion



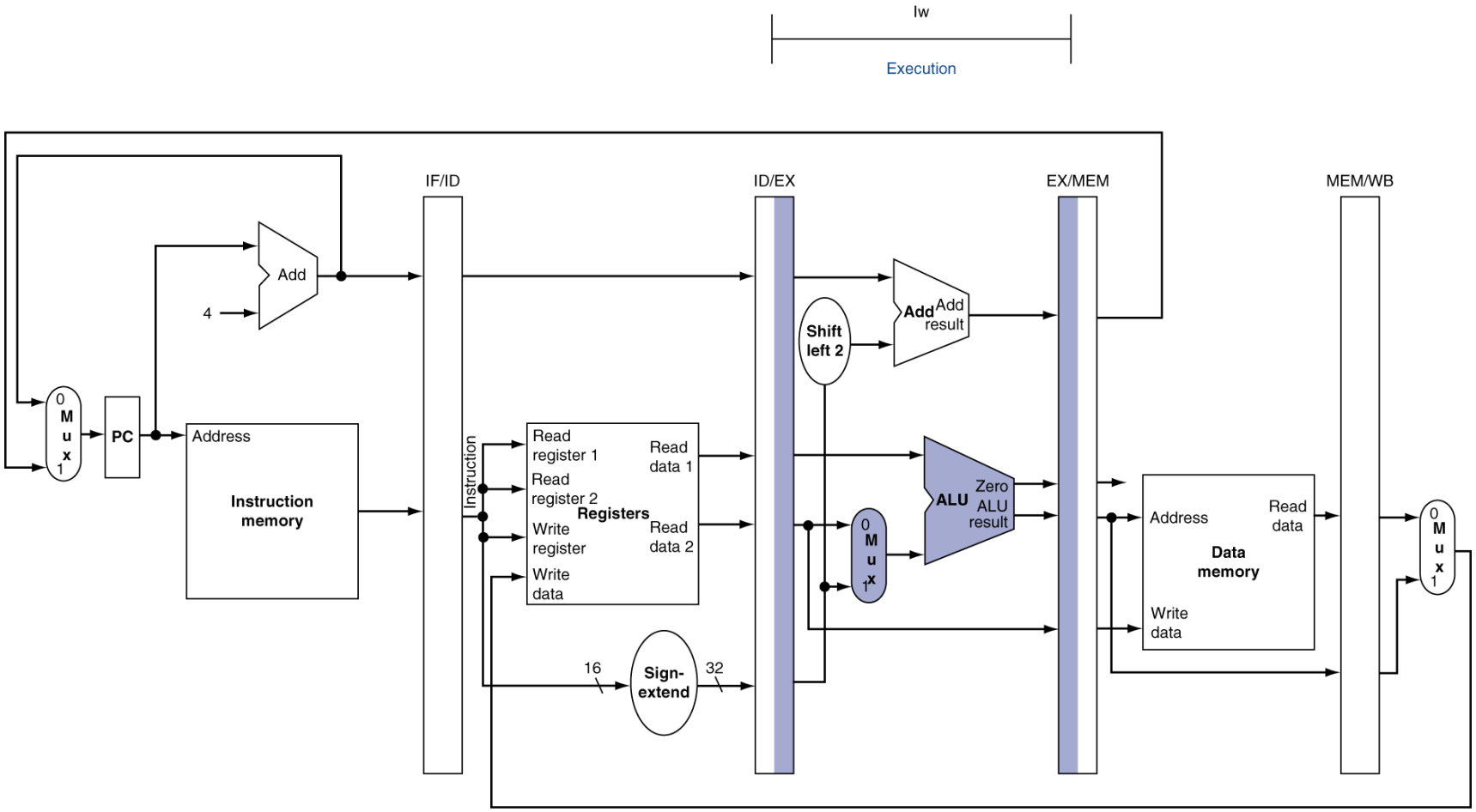
Stage 2 (ID) für load, store: Instruktion decodieren, Operanden

- PC wird von IF/ID nach ID/EX übernommen
- Instruktion wird dekodiert
- Operanden (Register 1, Register 2, Konstante) werden bestimmt und in ID/EX geschrieben



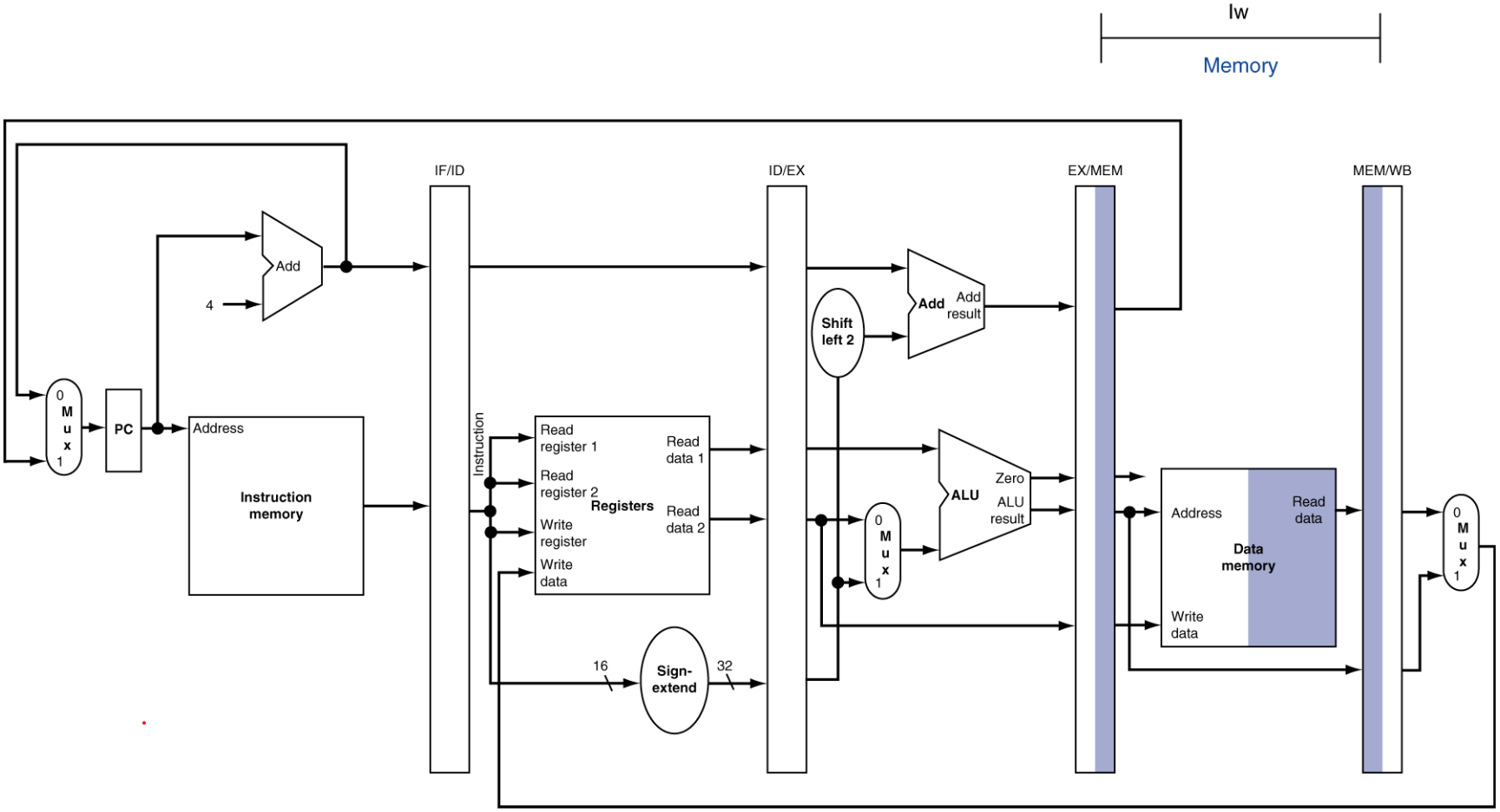
Stage 3 (EX) für load: Adresse berechnen

- Sprungadresse berechnen und in EX/MEM speichern
- ALU Ergebnisse (ALU result und zero Flag) in EX/MEM speichern
- Zweiten Operanden (Register 2) in EX/MEM übernehmen



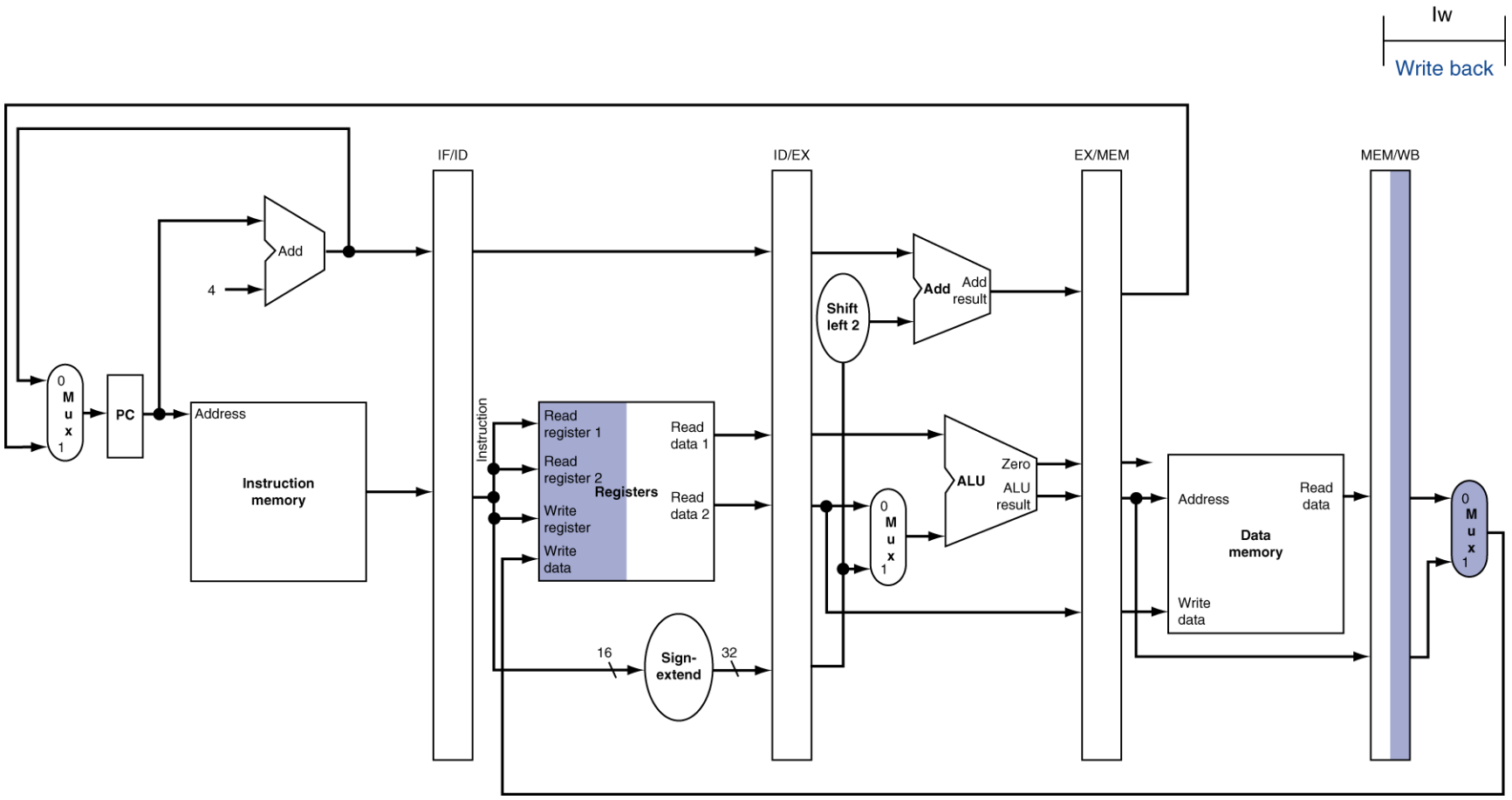
Stage 4 (MEM) für load: Speicherzugriff

- Speicherinhalt mit ALU Ergebnis aus EX/MEM als Adresse lesen und in MEM/WB schreiben
- Zweiten Operanden aus EX/MEM an Speicherstelle mit ALU Ergebnis aus EX/MEM als Adresse schreiben
- ALU Ergebnis in MEM/WB schreiben
- Sprung-Adresse in PC schreiben (wenn Bedingung erfüllt)



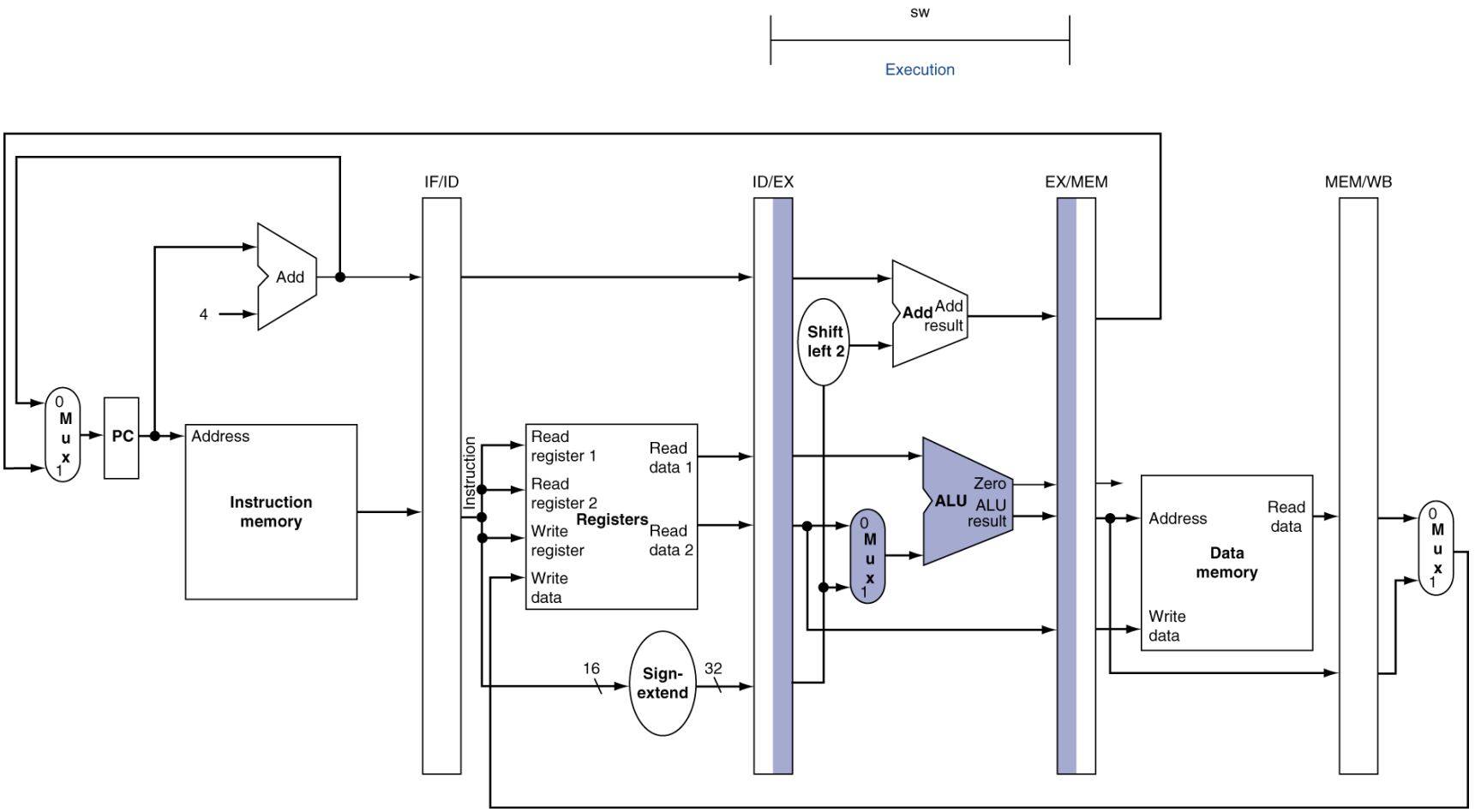
Stage 5 (WB) für load: Register schreiben

Geladenen Speicherinhalt (oder ALU Result) in Schreibregister speichern



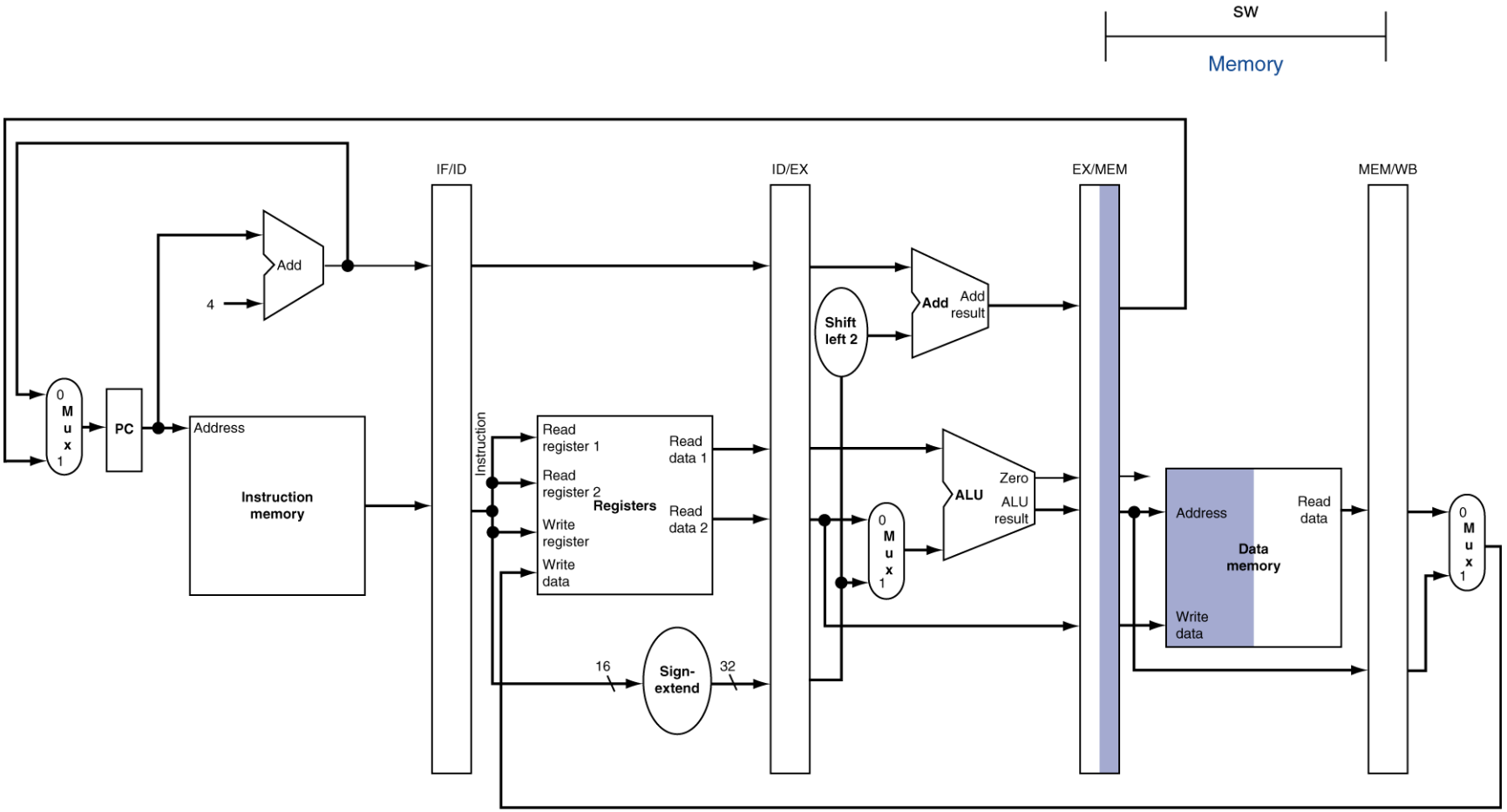
EX für "Store"

- Sprungadresse berechnen und in EX/MEM speichern
- ALU Ergebnisse (ALU result und zero Flag) in EX/MEM speichern
- Zweiten Operanden (Register 2) in EX/MEM übernehmen



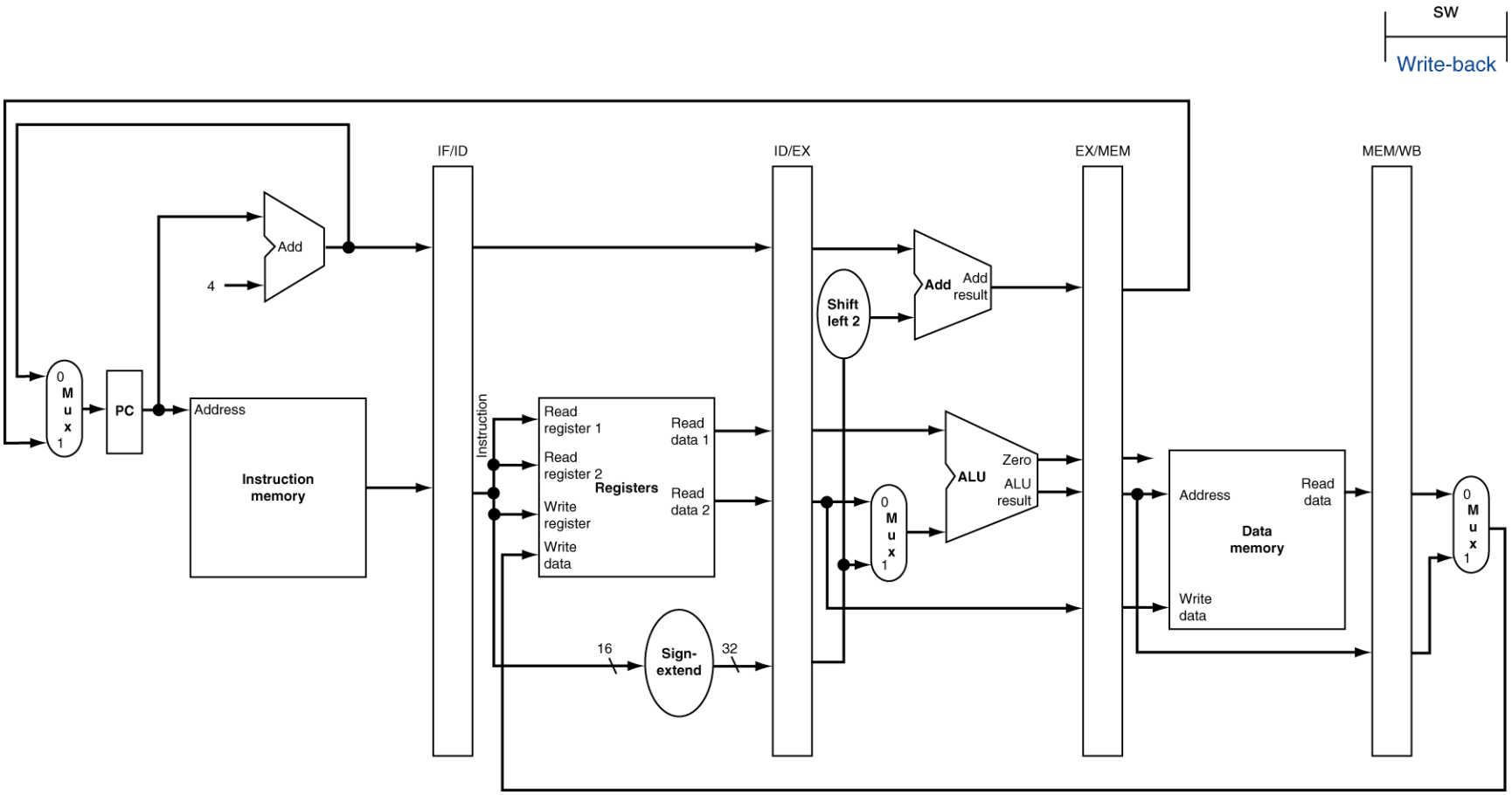
MEM für "Store"

- Speicherinhalt mit ALU Ergebnis aus EX/MEM als Adresse lesen und in MEM/WB schreiben
- Zweiten Operanden aus EX/MEM an Speicherstelle mit ALU Ergebnis aus EX/MEM als Adresse schreiben
- ALU Ergebnis in MEM/WB schreiben
- Sprung-Adresse in PC schreiben (wenn Bedingung erfüllt)



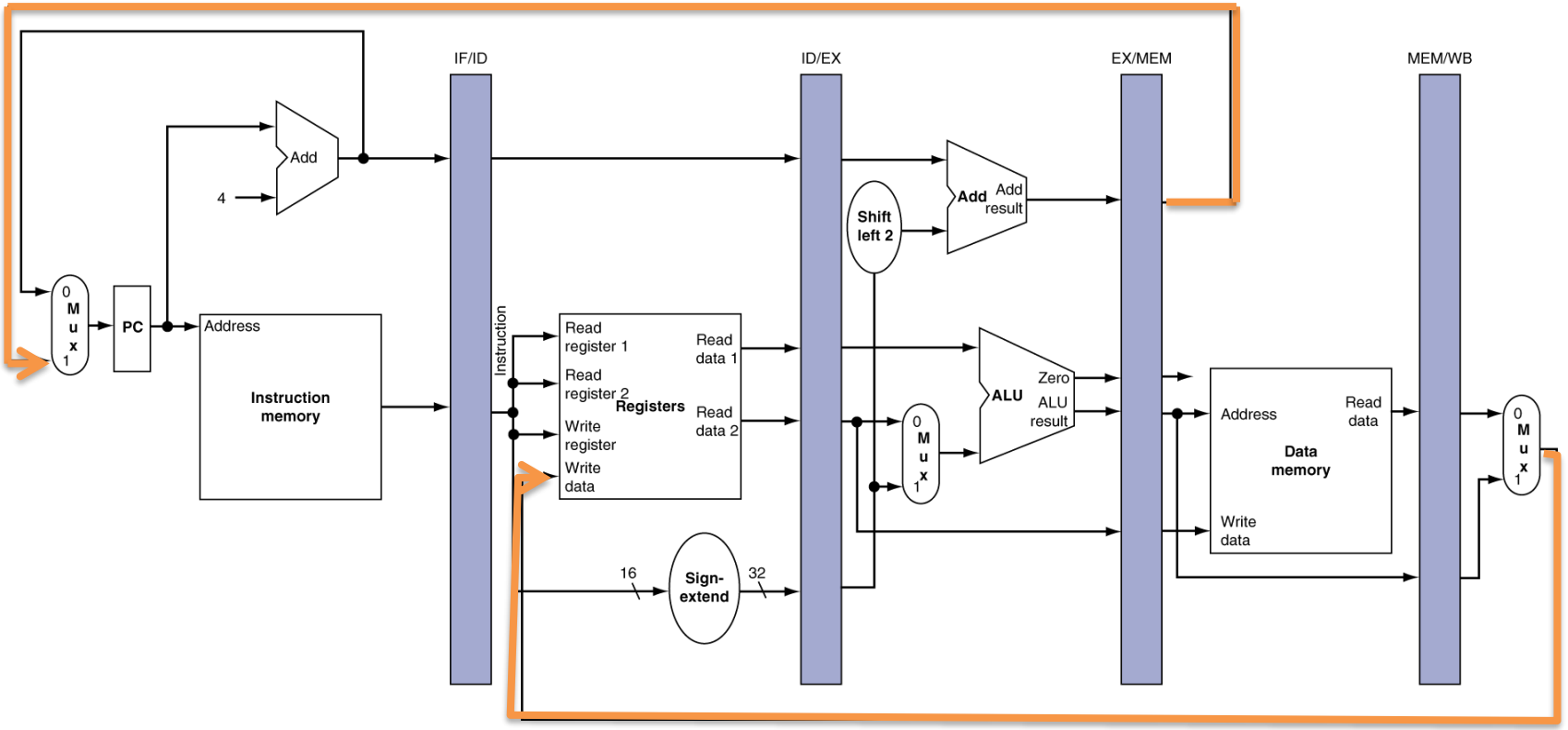
WB für store

- Geladenen Speicherinhalt (oder ALU Result) in Schreibregister speichern



Datenpfad mit Pipeline-Register

- Die meisten Daten durchlaufen die Pipeline von links nach rechts und haben somit nur Auswirkung auf den eigenen Befehl
- Daten die von rechts nach links laufen haben Auswirkungen auf nachfolgende Befehle
- Die Rückführung der Instruktionsadresse bei einem Sprung kann zu einem Steuerkonflikt führen
- Die Rückführung der Daten in der WB Stage zum Registersatz kann zu einem Datenkonflikt führen

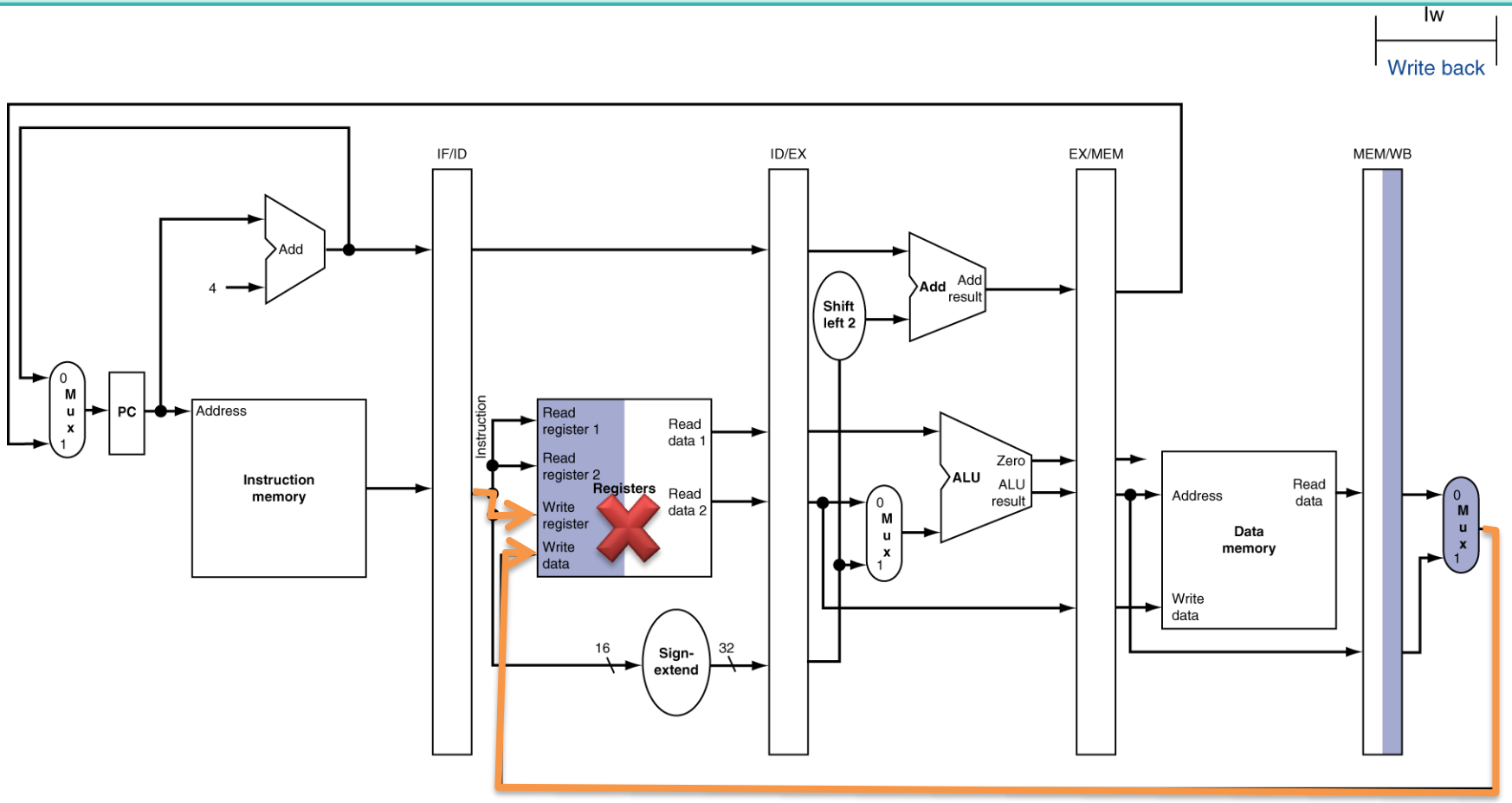


Stage 5 (WB) für load: Register schreiben

Problem:

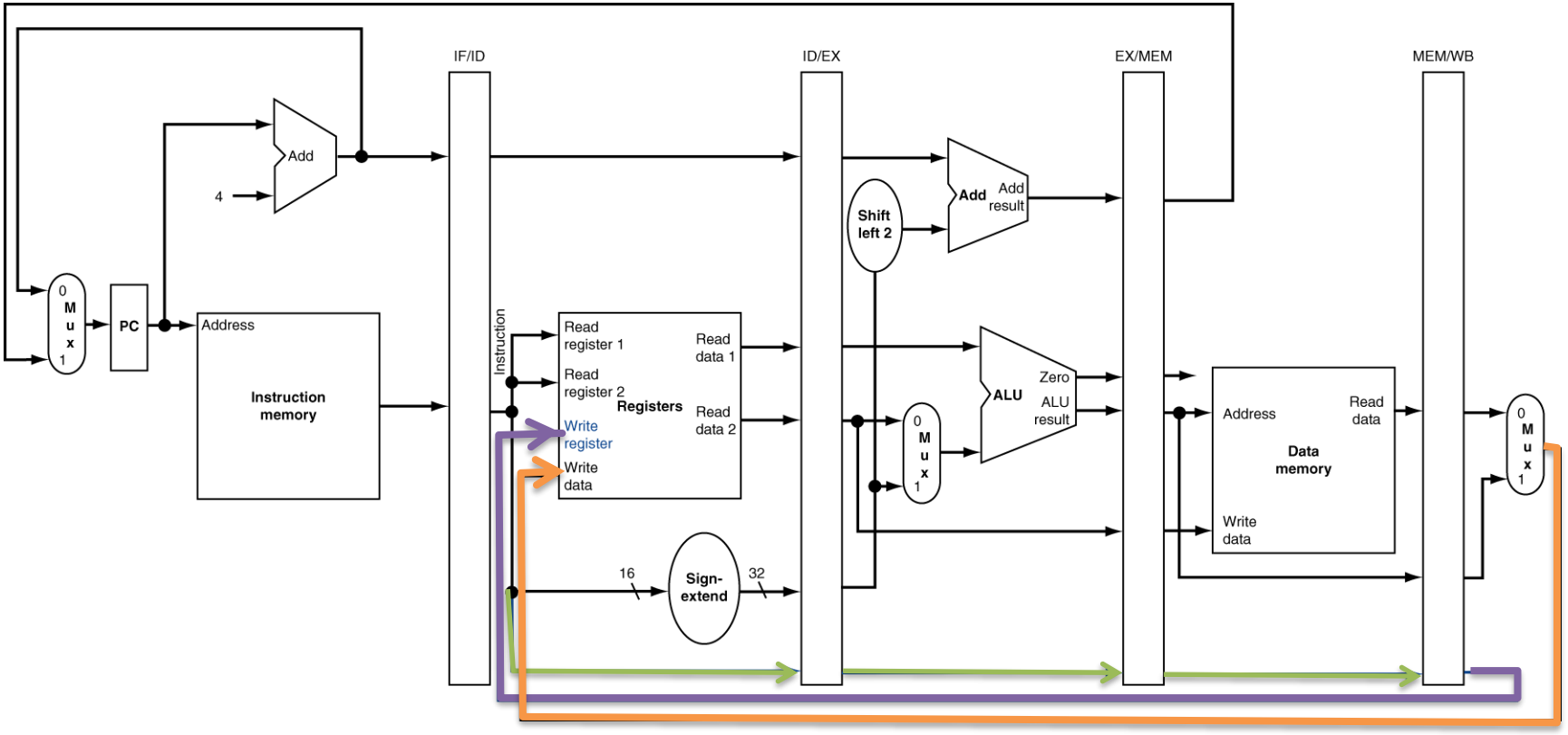
- Der Ergebnis-Wert der Instruktion in der WB-Stage liegt am Write-Data-Eingang des Registerblock an.
- Das Zielregister am Write-Register-Eingang des Registerblocks wird aber von der Instruktion in der ID-Stage bestimmt.

Lösung: Der Write-Register-Eingang des Registerblocks wird von einem Wert im MEM/WB-Register bestimmt.



Korrekter Datapath als Pipeline für load

- Nummer des Schreibregisters (rt) wird von Pipeline-Register zu Pipeline-Register kopiert
- Schreibregister wird nicht über Instruktion in IF/ID Pipeline-Register sondern über Nummer des Schreibregisters (rd) aus dem Pipeline-Register MEM/ID bestimmt

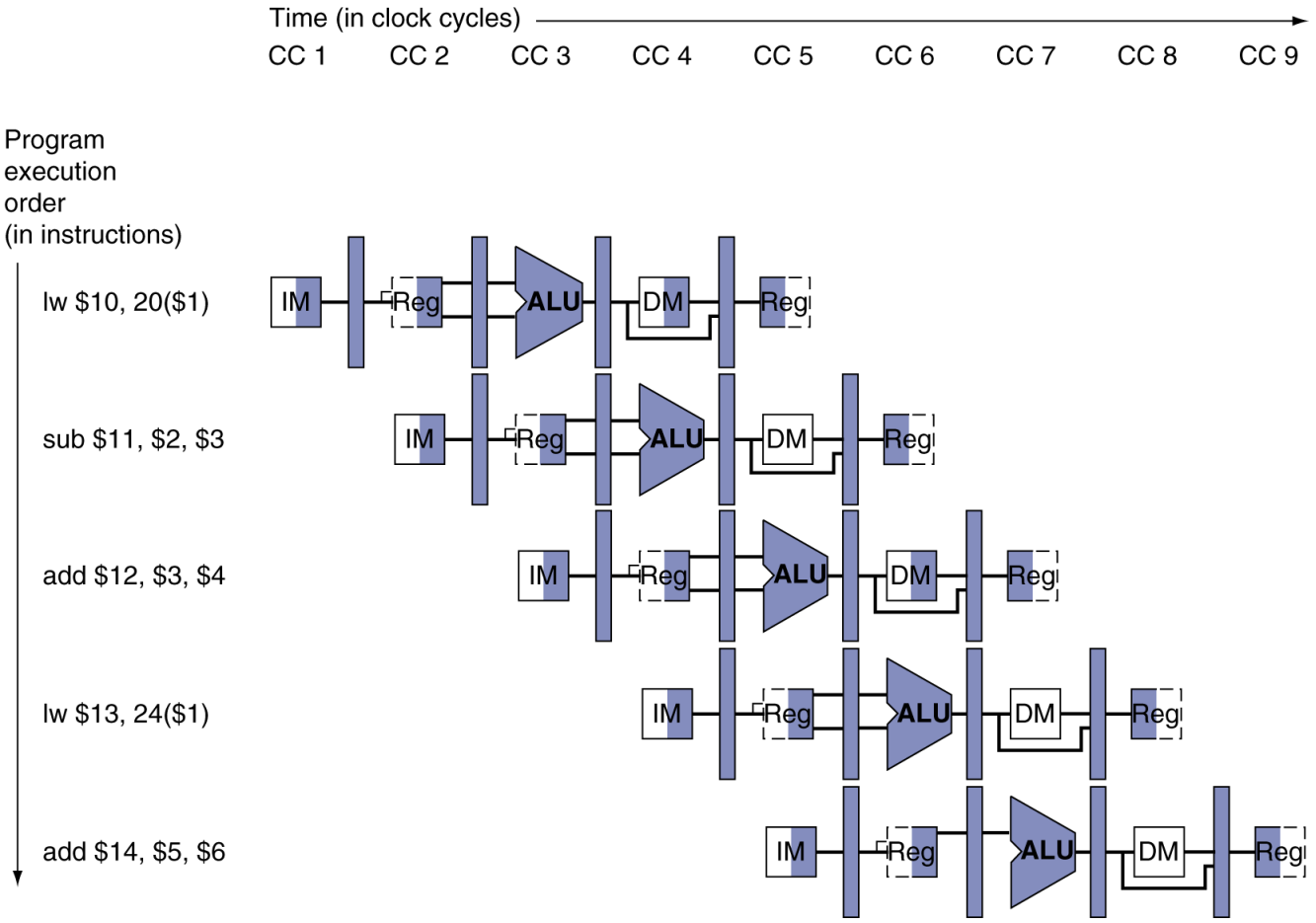


Darstellung einer Pipeline

- Betrachten den Instruktionsdurchlauf einer Pipeline Takt für Takt
  - "Single-Clock-Cycle" Pipeline Diagramm
    - zeigt den Gebrauch der Pipeline während eines einzigen Taktzyklus
      - Instruktionen in verschiedenen Stages
    - hebt die genutzten Ressourcen hervor
  - "Multi-Clock-Cycle" Diagramm
    - veranschaulicht die zeitliche Abfolge der Instruktionen in der Pipeline
    - weniger Details zu den einzelnen Stages

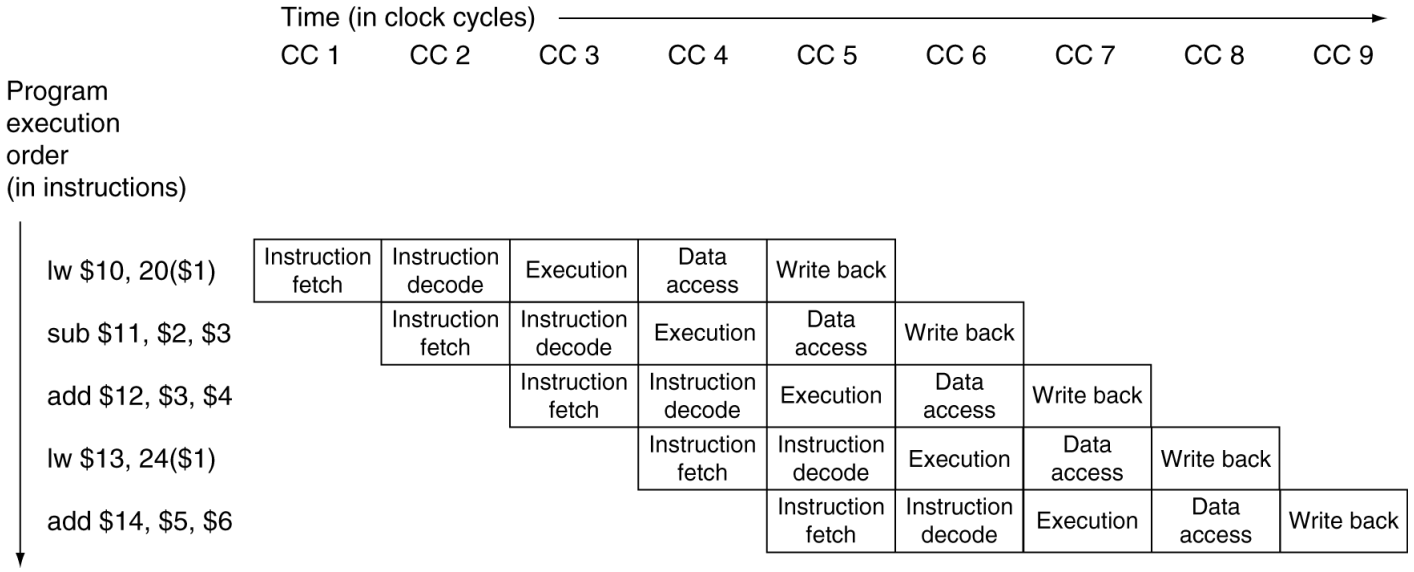
Multi-Clock-Cycle-Pipeline-Diagramm

- Zeigt die Nutzung der Ressourcen durch mehrere aufeinanderfolgende Instruktionen



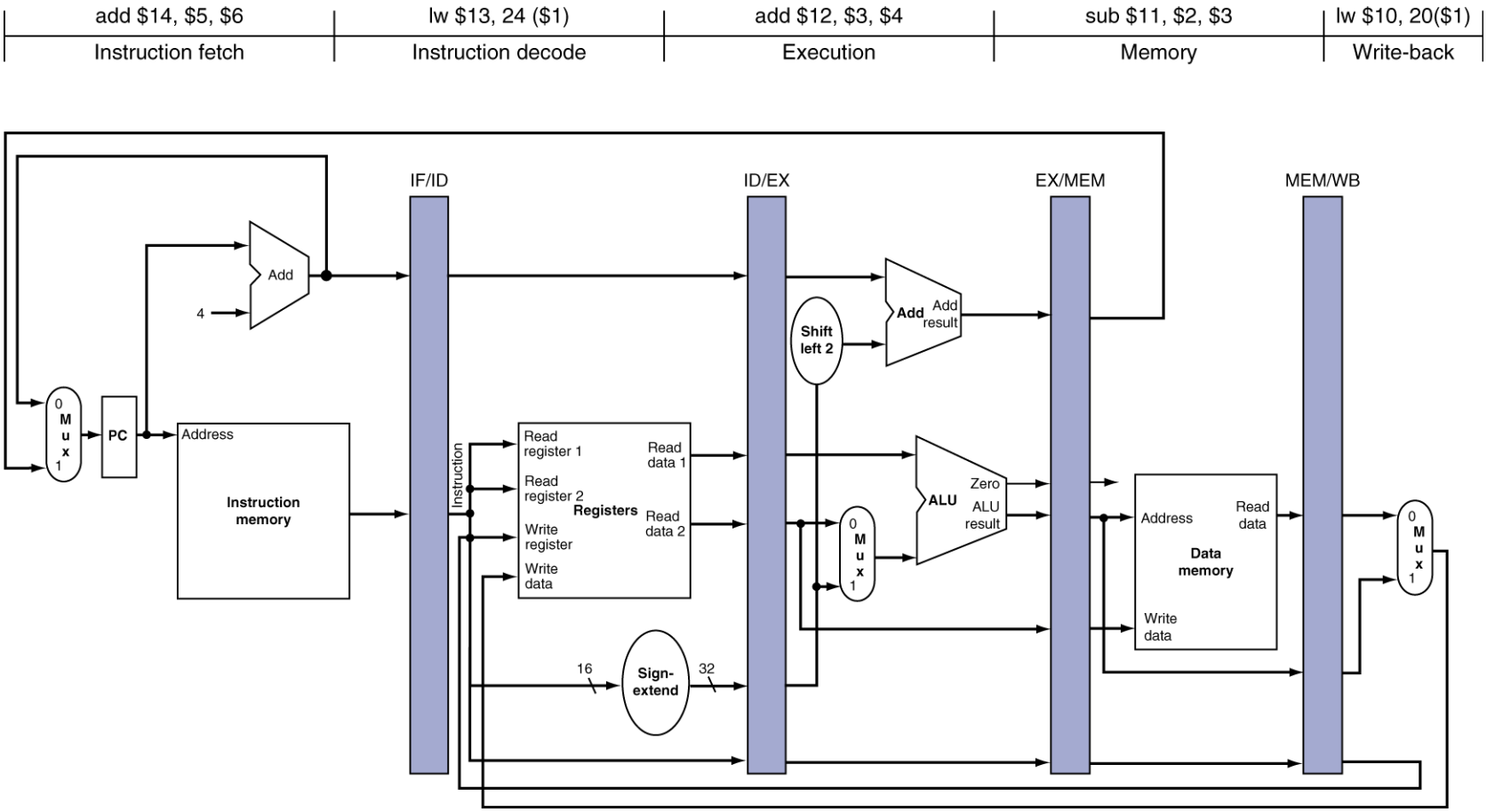
Multi-Clock-Cycle-Pipeline-Diagramm (traditionell)

- In traditioneller Form als Kästchen pro Pipeline-Stage und Ressource



Single-Clock-Cycle-Pipeline-Diagramm

- Zeigt den Zustand der Pipeline in einem bestimmten Takt
- Frage: Was steht vor und nach Ausführung des Takts in den Pipeline-Registern?
  ➔ Übungsaufgabe



Kapitel 4 - Gliederung

Kapitel 4: Multi-Cycle CPU – Pipeline-Architekturen

4.1 Prinzip einer Pipeline

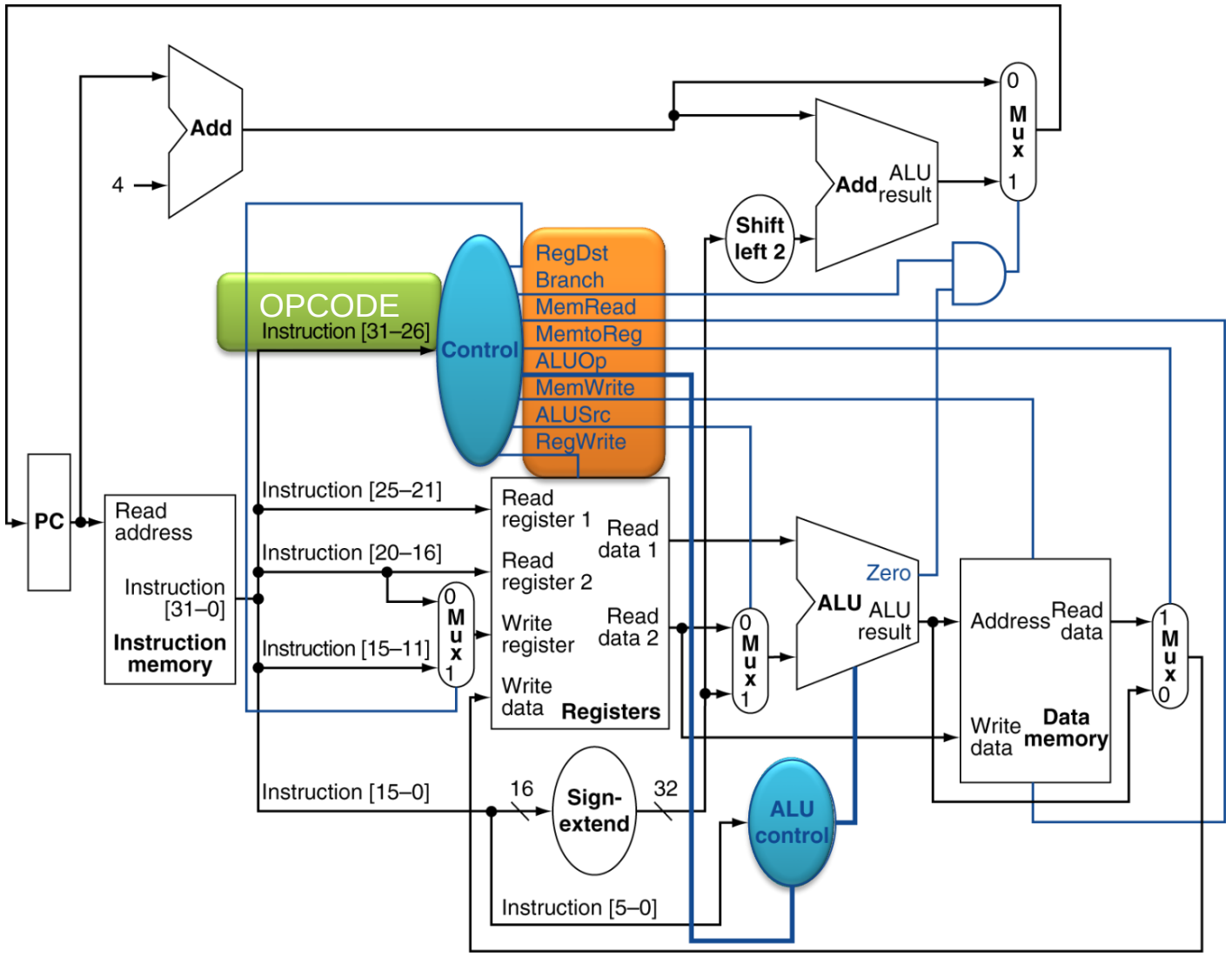
4.2 Datapath der MIPS Pipeline

4.3 Control der MIPS Pipeline

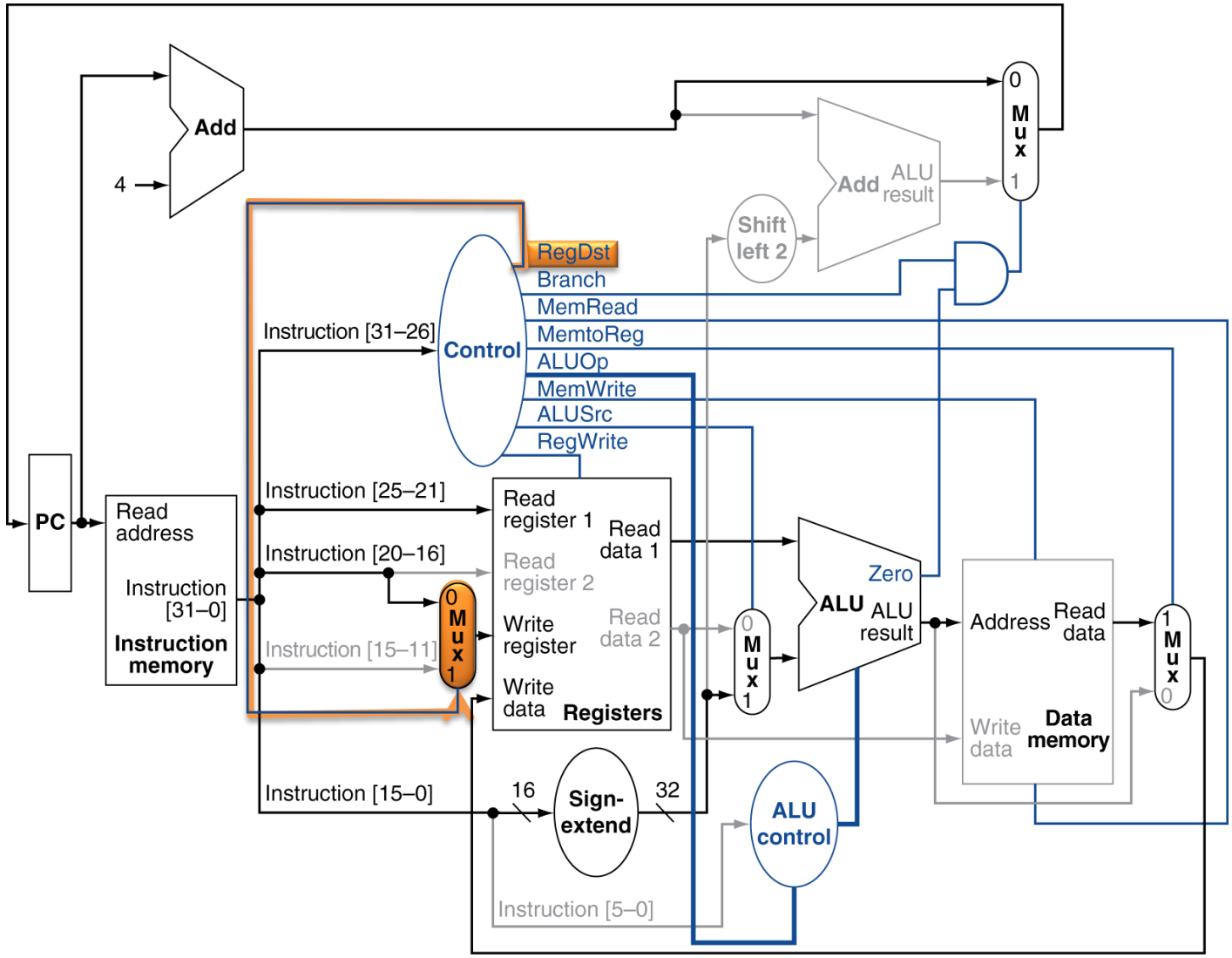
4.4 Hazards

4.5 Exceptions

Wiederholung Single-Cycle Control

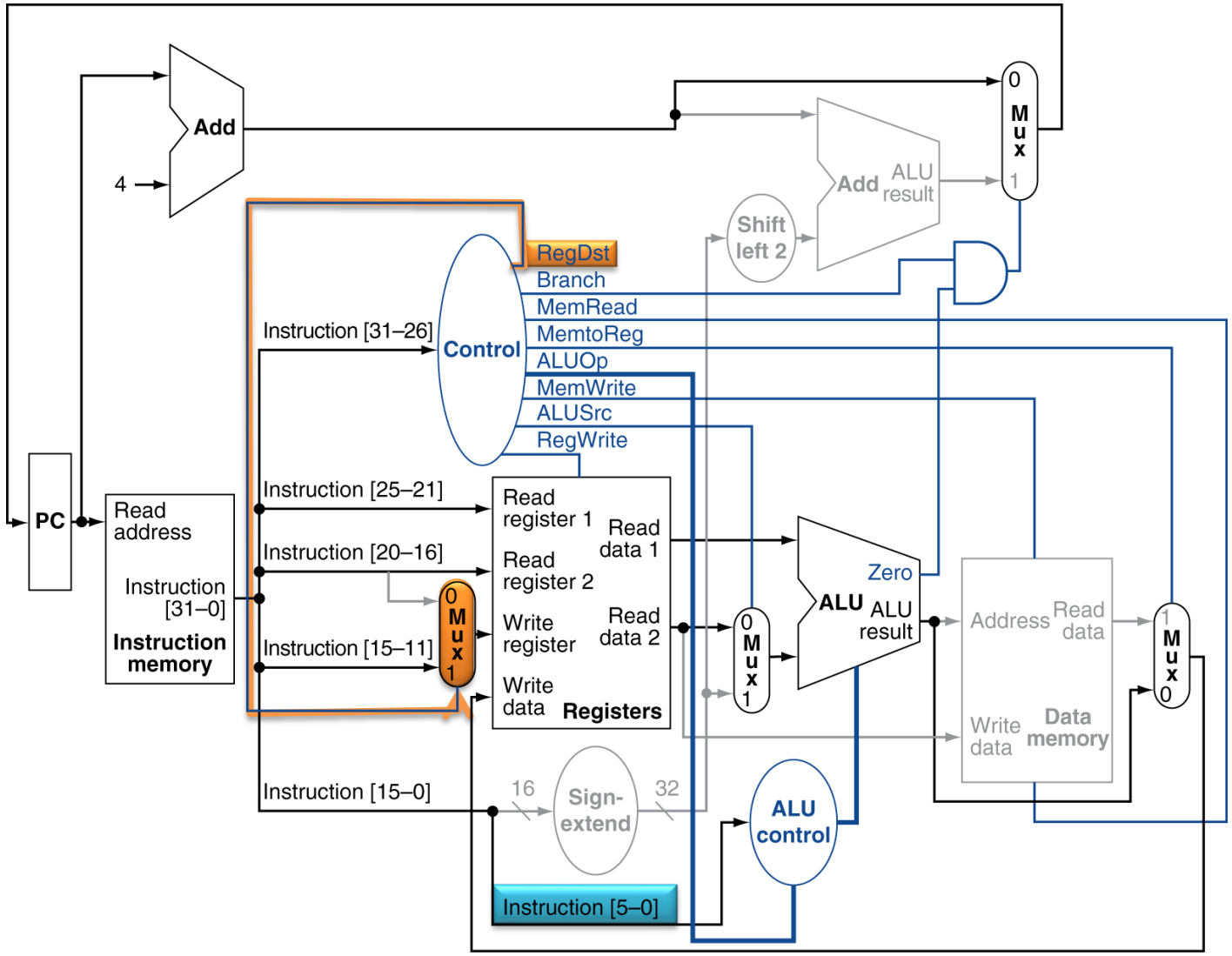


Wiederholung Single-Cycle Control - Load



Steuer-Bits	lw
RegDst	0
Branch	0
MemRead	1
MemToReg	1
ALUOp	00
MemWrite	0
ALUSrc	1
RegWrite	1

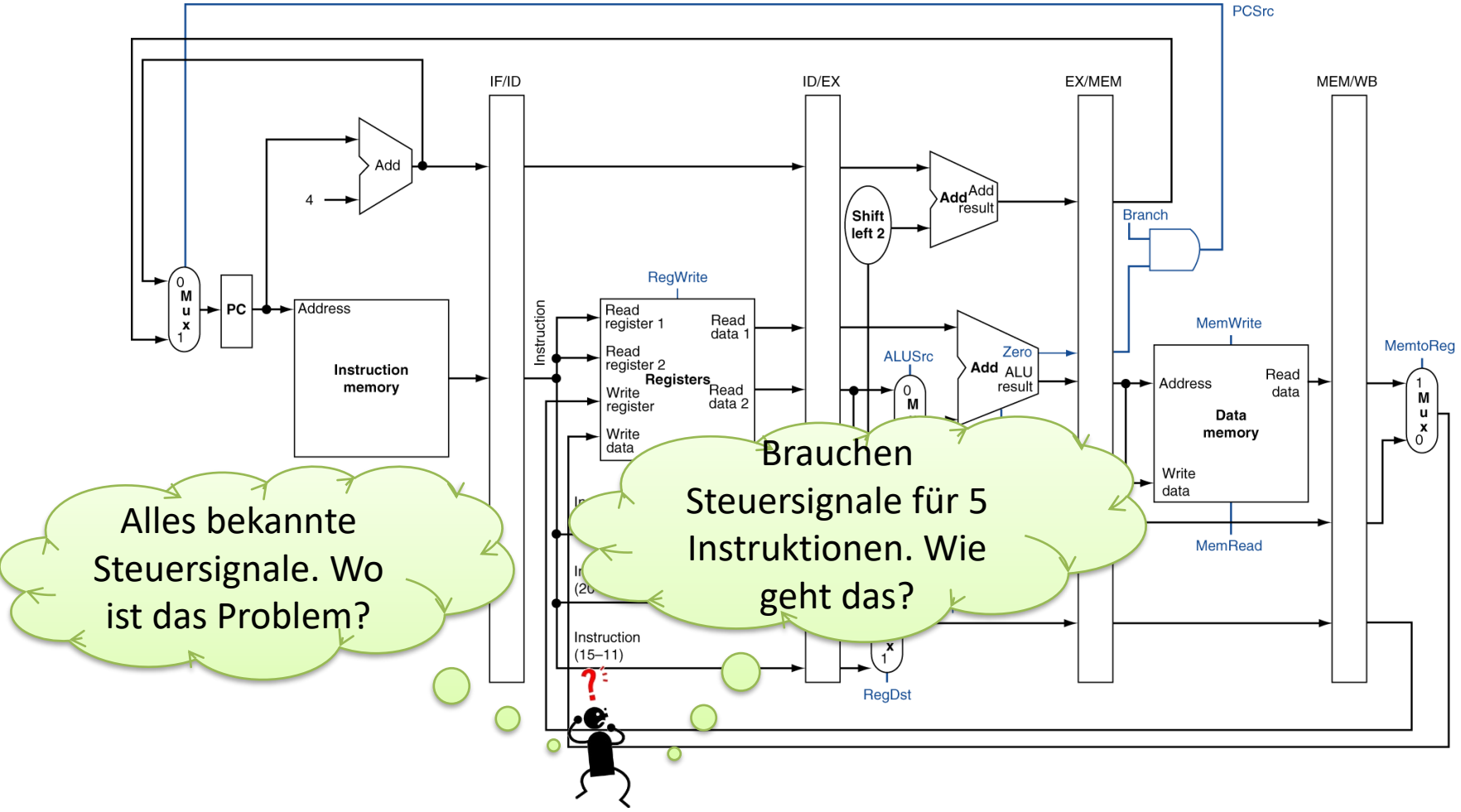
Wiederholung Single-Cycle Control - R-Format Instruktionen



Steuer-Bits	R-Format
RegDst	1
Branch	0
MemRead	0
MemToReg	0
ALUOp	10
MemWrite	0
ALUSrc	0
RegWrite	1

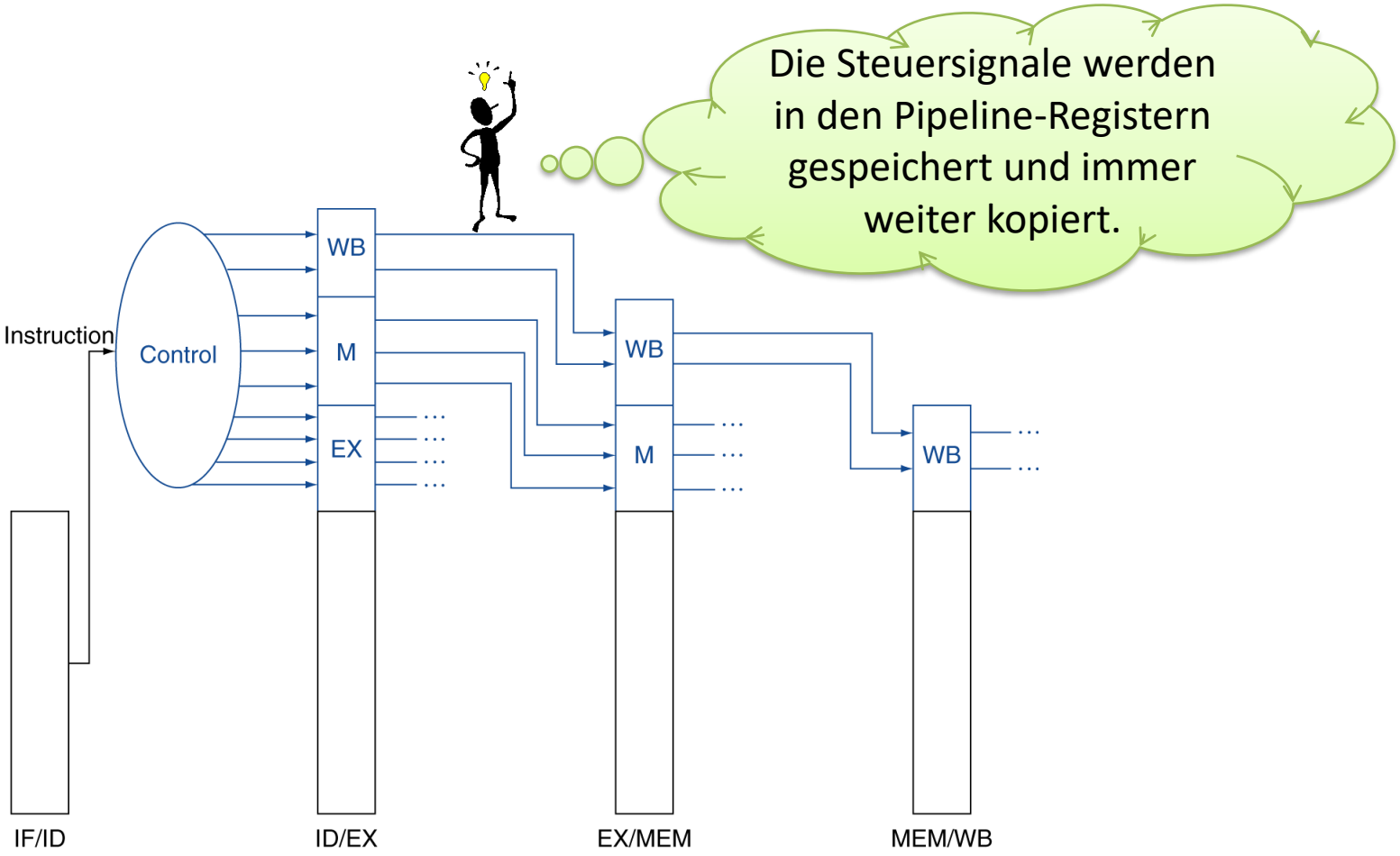
Pipelined Control – Eine erste Übersicht

- Die Steuersignale der einzelnen Komponenten sind in einem Multi-Cycle DataPath die gleichen wie in einem Single-Cycle-Datapath.
- Aber bislang gab es nur ein Steuerwerk (Control), das die Steuersignale aus dem OPCODE bestimmt hat
- Wie wird das Steuerwerk in einer Multi-Cycle-CPU realisiert?

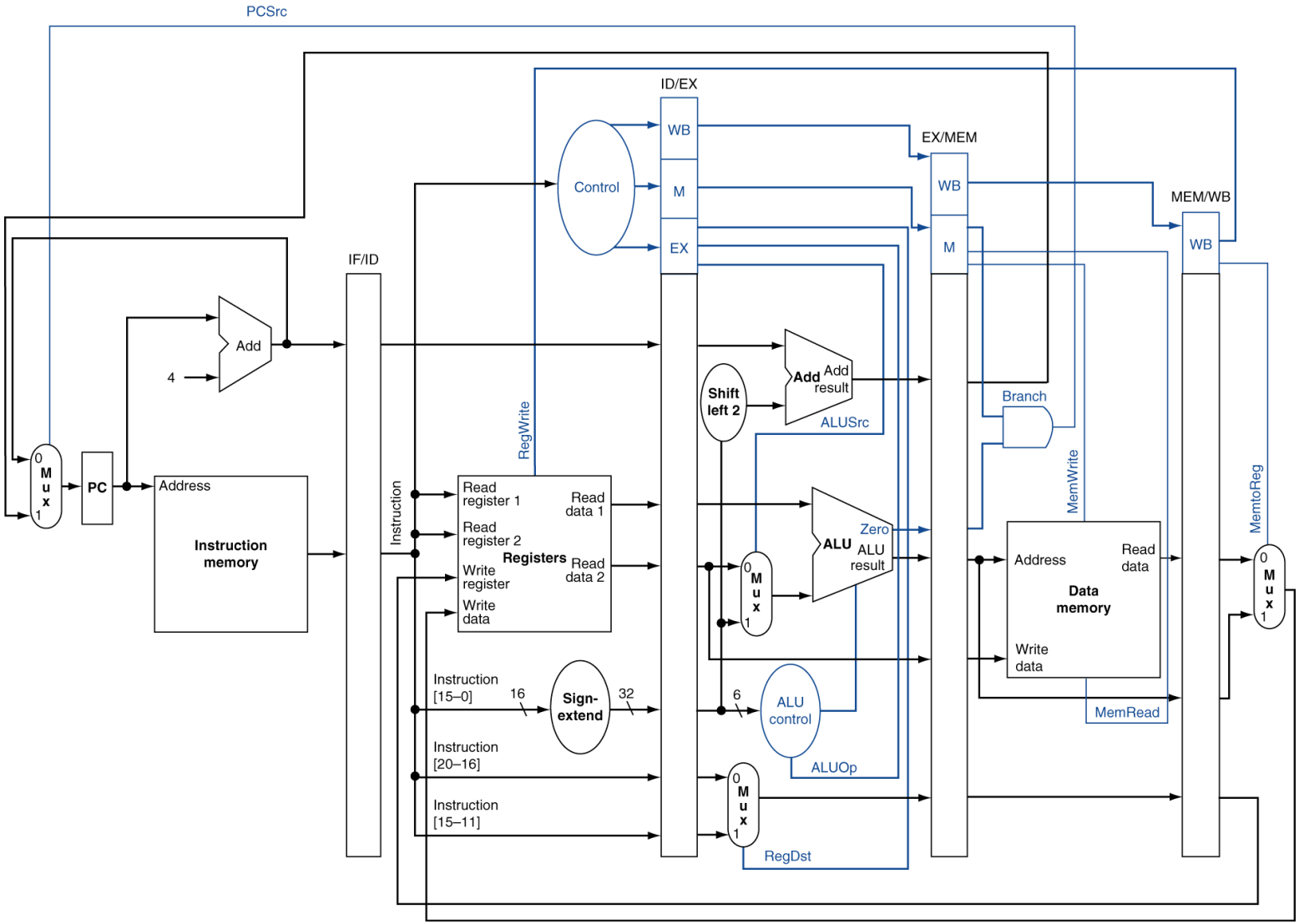


Pipelined Control – Steuersignale aus OPCODE

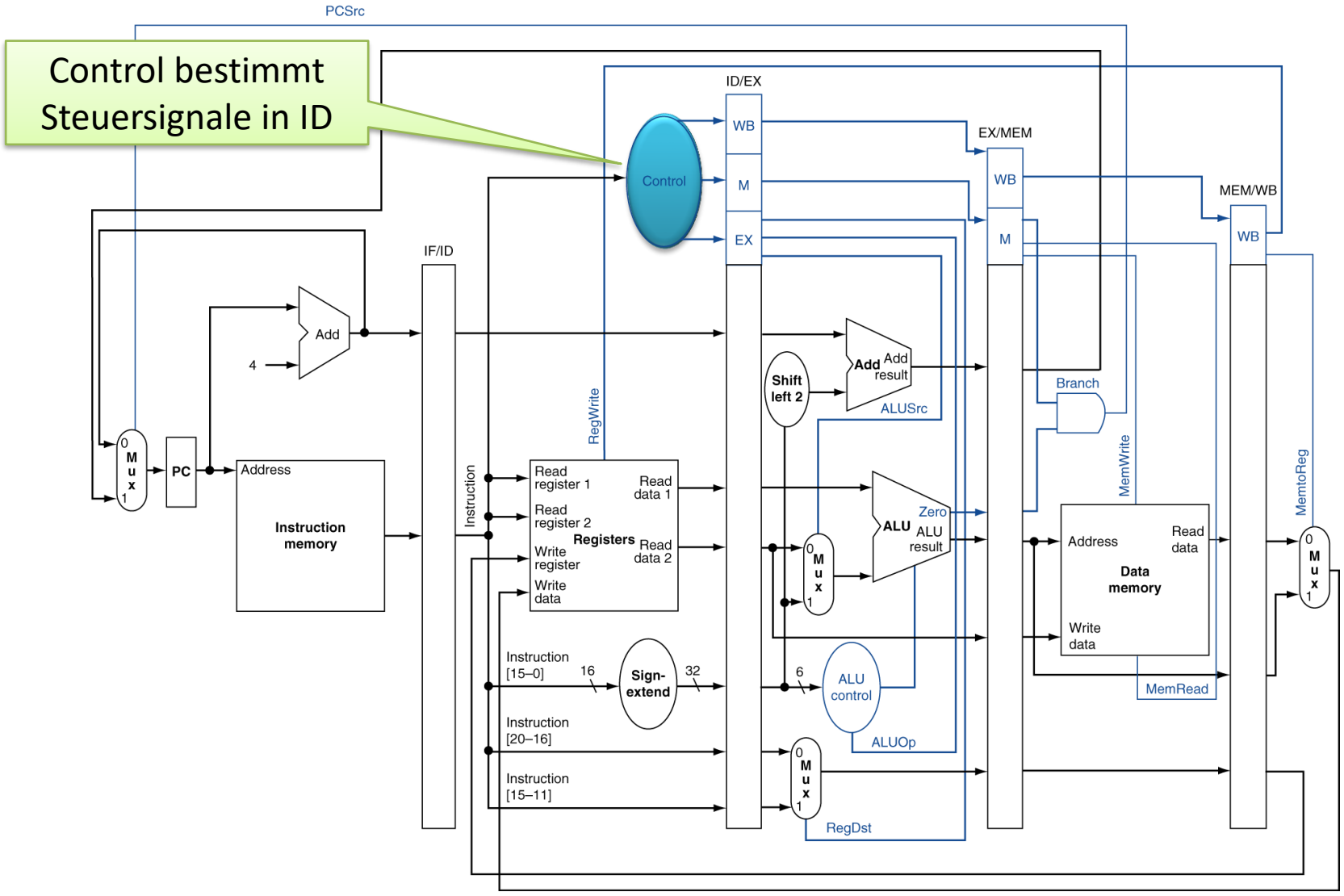
Auch in einer Multi-Cycle-CPU gibt es nur ein Steuerwerk, das die Steuersignale aus dem OPCODE der Instruktion bestimmt. Es befindet sich in der ID-Stage. Zur Verwendung in späteren Stages werden die Steuersignale in den Pipeline-Registern gespeichert.



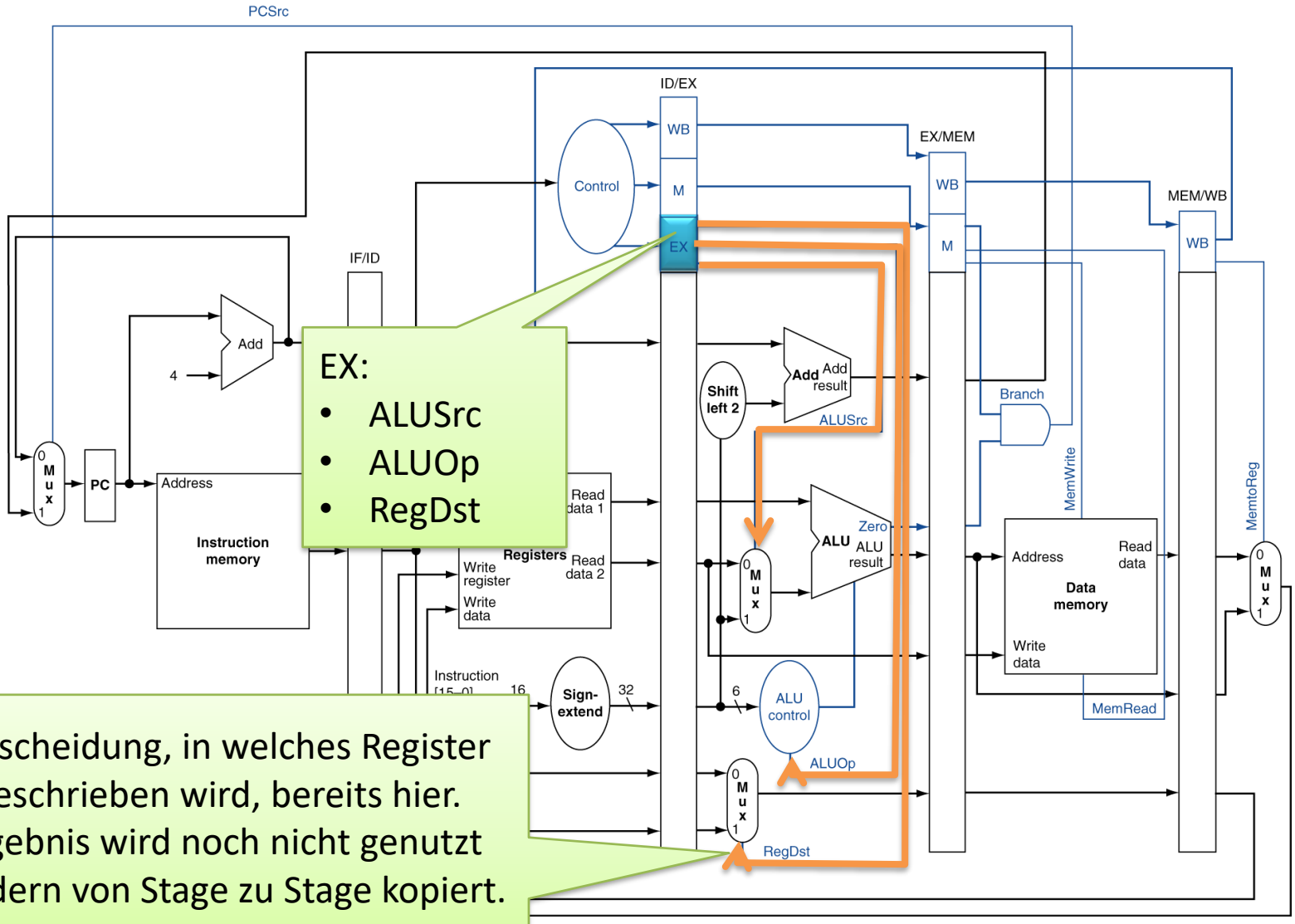
Pipelined Control – The BIG Picture



Pipelined Control – The BIG Picture



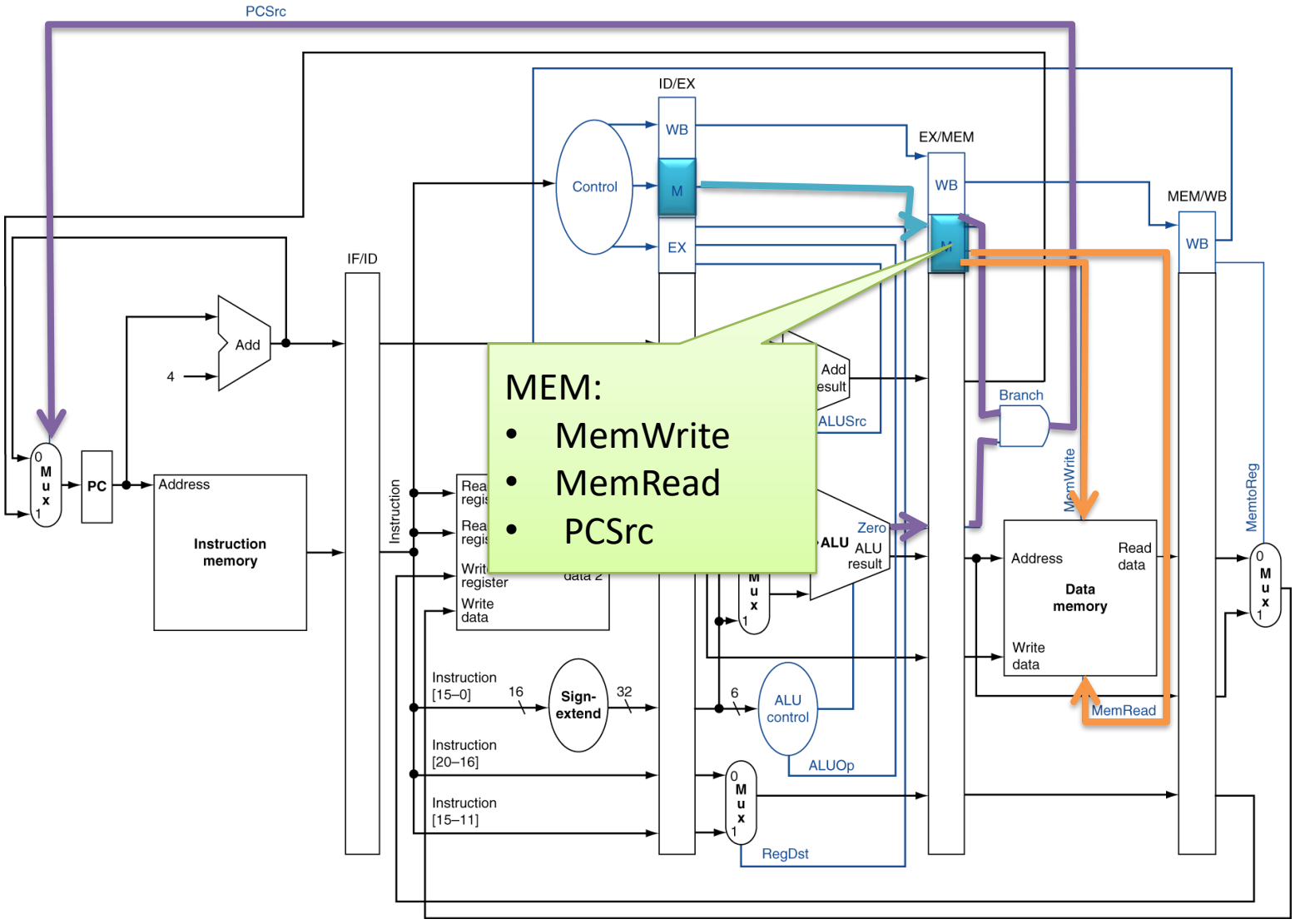
Pipelined Control – EX-Stage



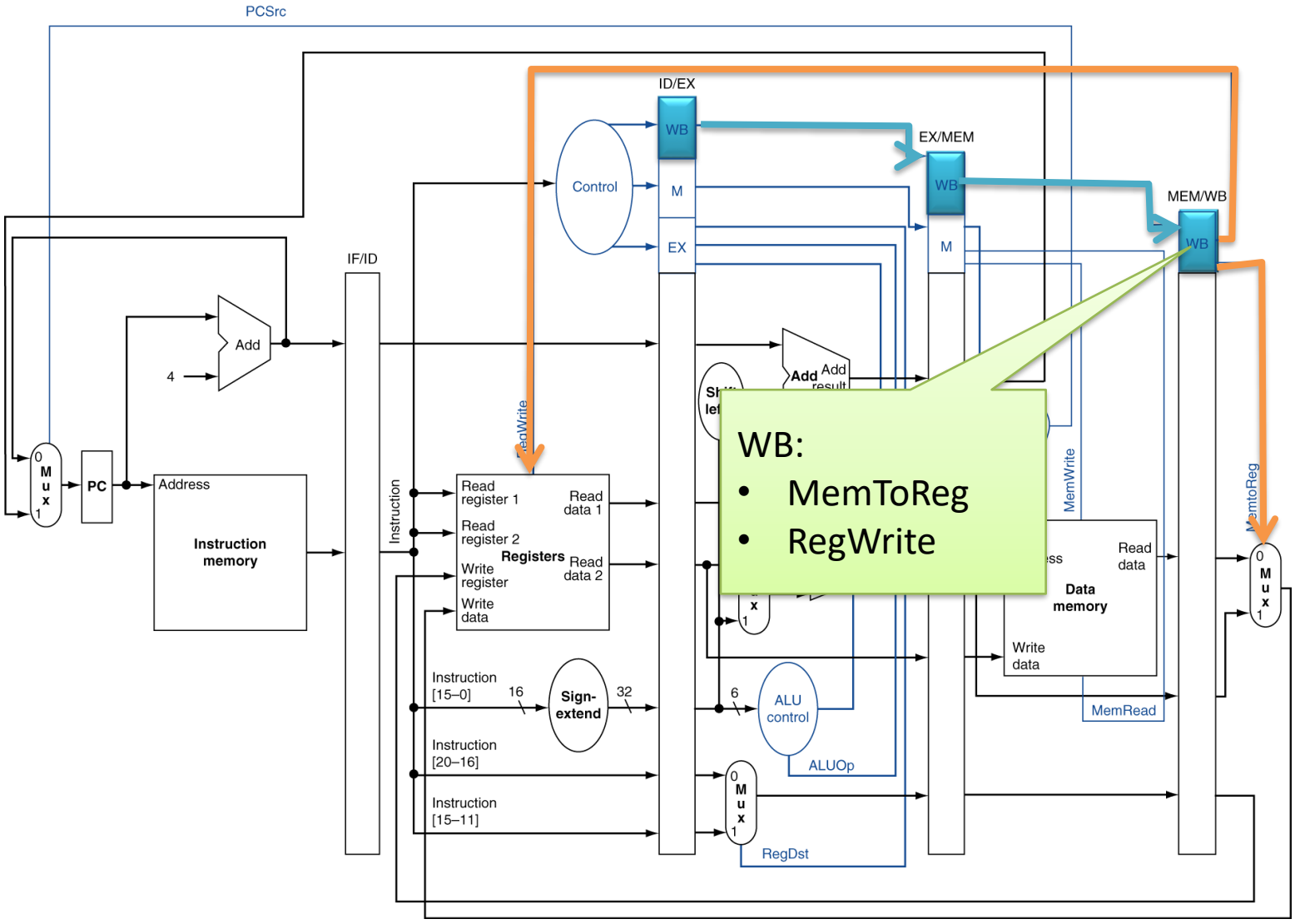
EX:
- ALUSrc
- ALUOp
- RegDst

Entscheidung, in welches Register geschrieben wird, bereits hier. Ergebnis wird noch nicht genutzt sondern von Stage zu Stage kopiert.

Pipelined Control – MEM Stage



Pipelined Control – WB Stage



Pipeline – The BIG Picture