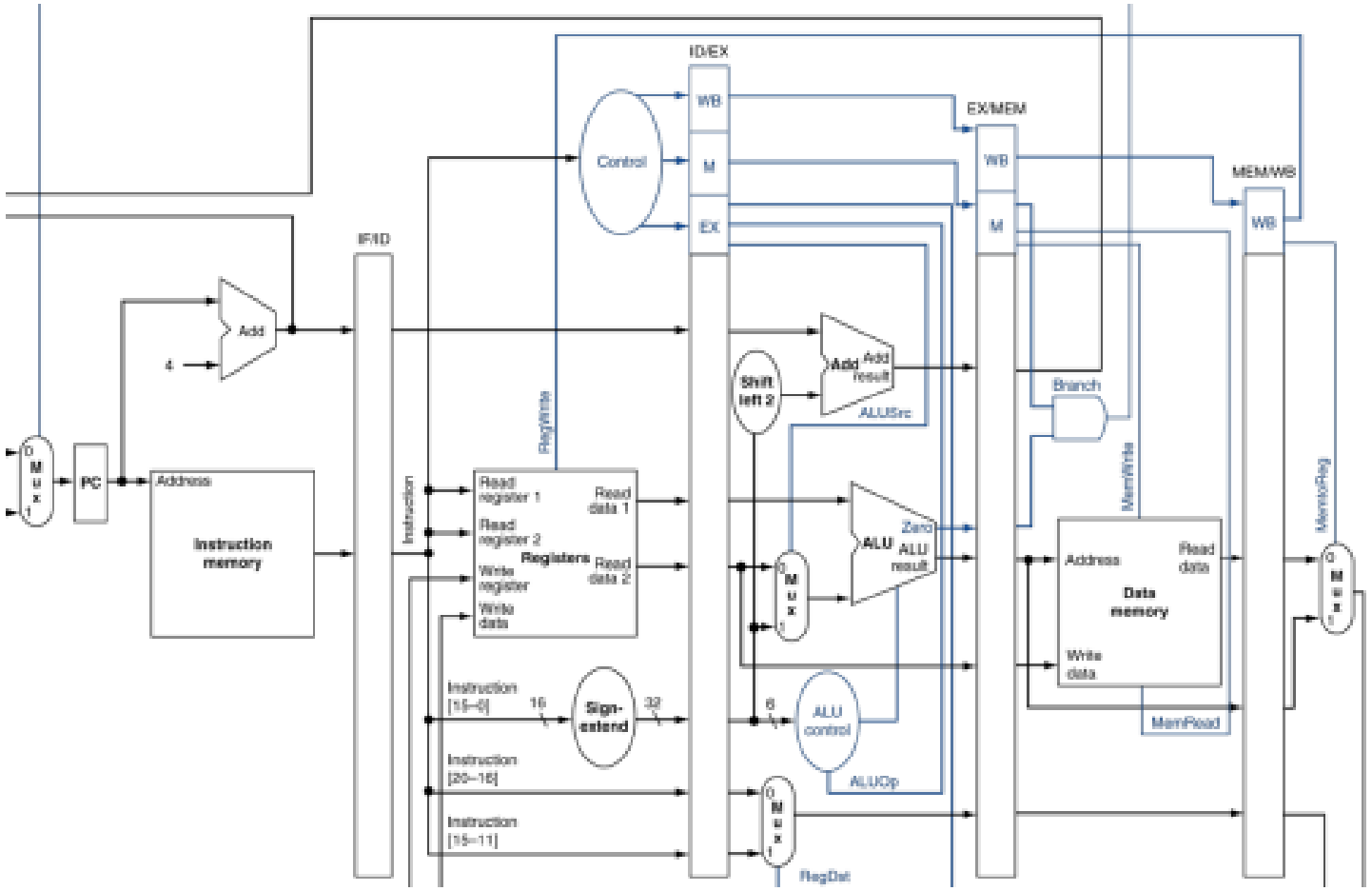
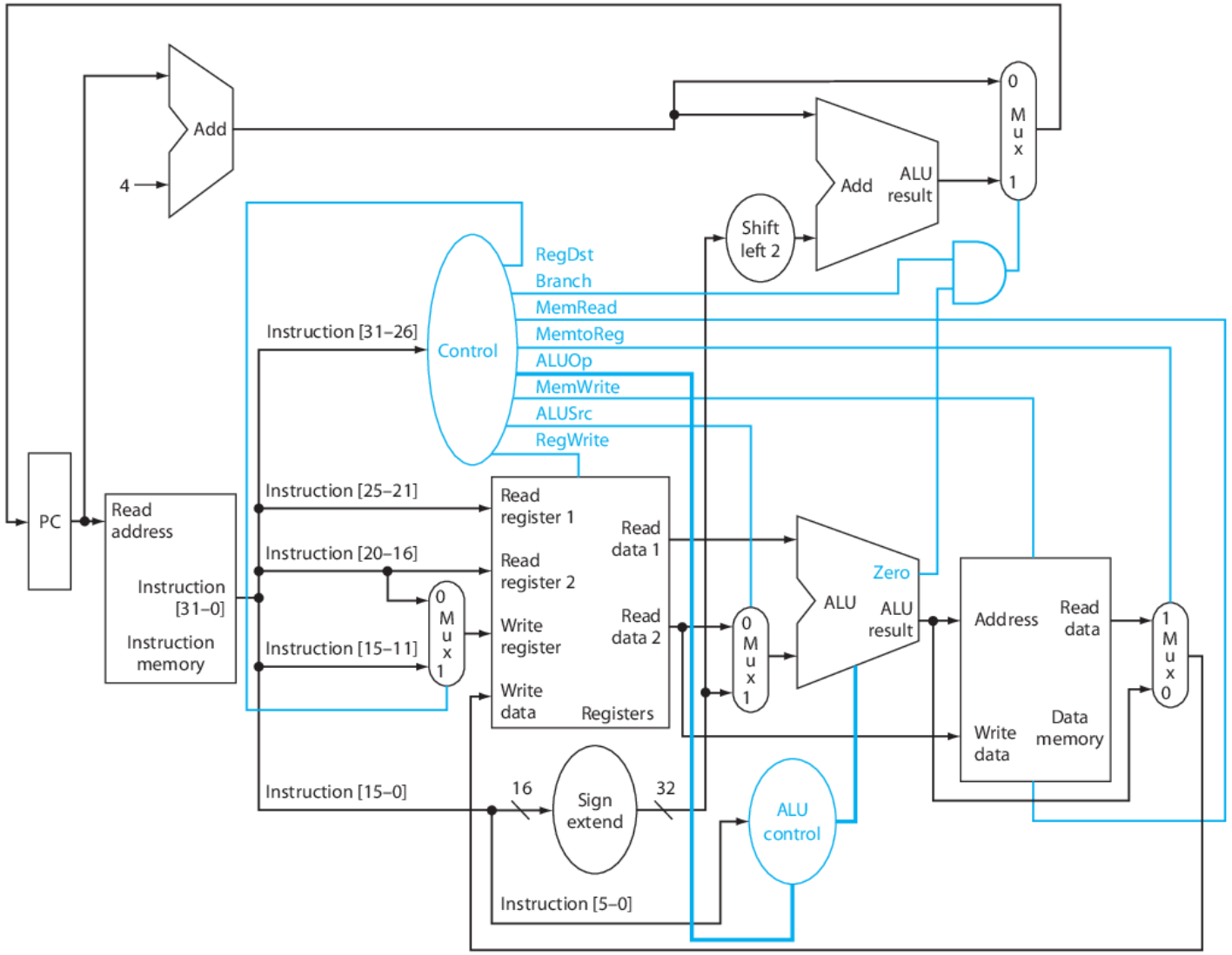
Pipelining

Hazards,
Programmoptimierung,
Performanceverlgeich



Eintaktprozessor

- Eintaktprozessor
- Befehle werden sequentiell au sgeführt
- Nur ein Befehl im Prozessor
- Viel Datenpfad ungenutzt
- Ein Befehl pro Takt

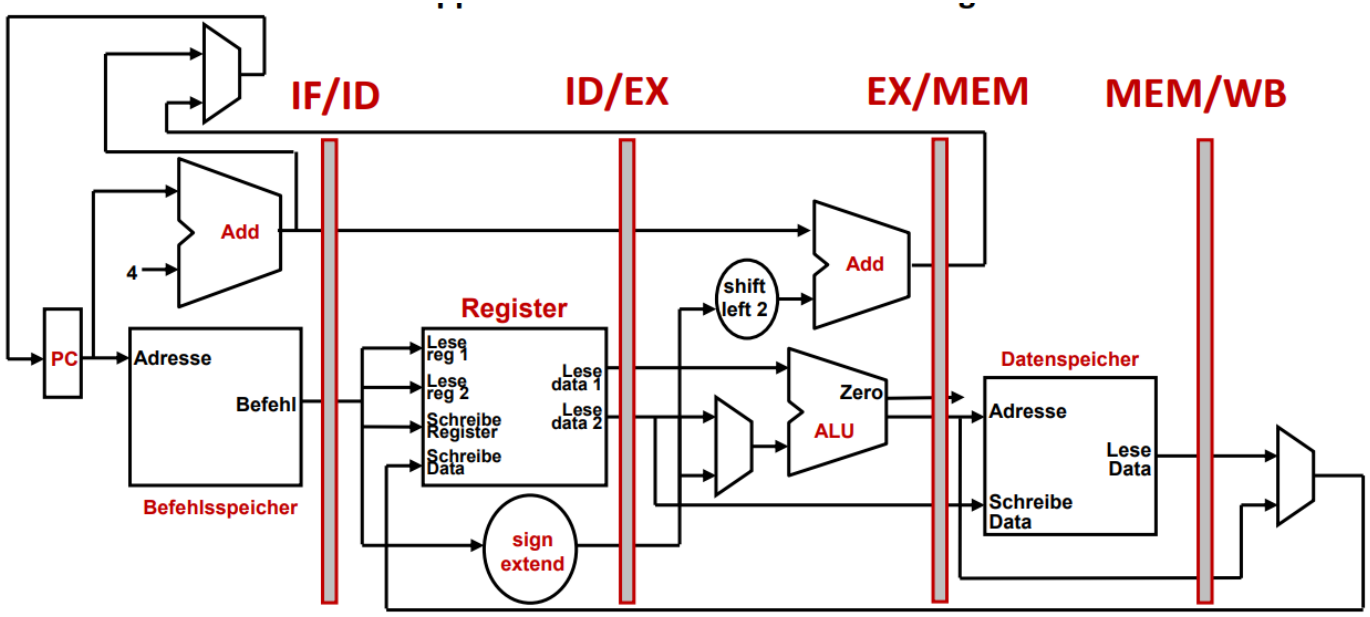


Pipelining

- Prozessor in Phasen aufgeteilt
- Eine Phase pro Takt
- In jeder Phase anderer Befehl

Befehl	Zyklus								
	1	2	3	4	5	6	7	8	9
Befehl 1	IF	ID	EX	MEM	WB				
Befehl 2		IF	ID	EX	MEM	WB			
Befehl 3			IF	ID	EX	MEM	WB		
Befehl 4				IF	ID	EX	MEM	WB	
Befehl 5					IF	ID	EX	MEM	WB

Phasenunterteilung



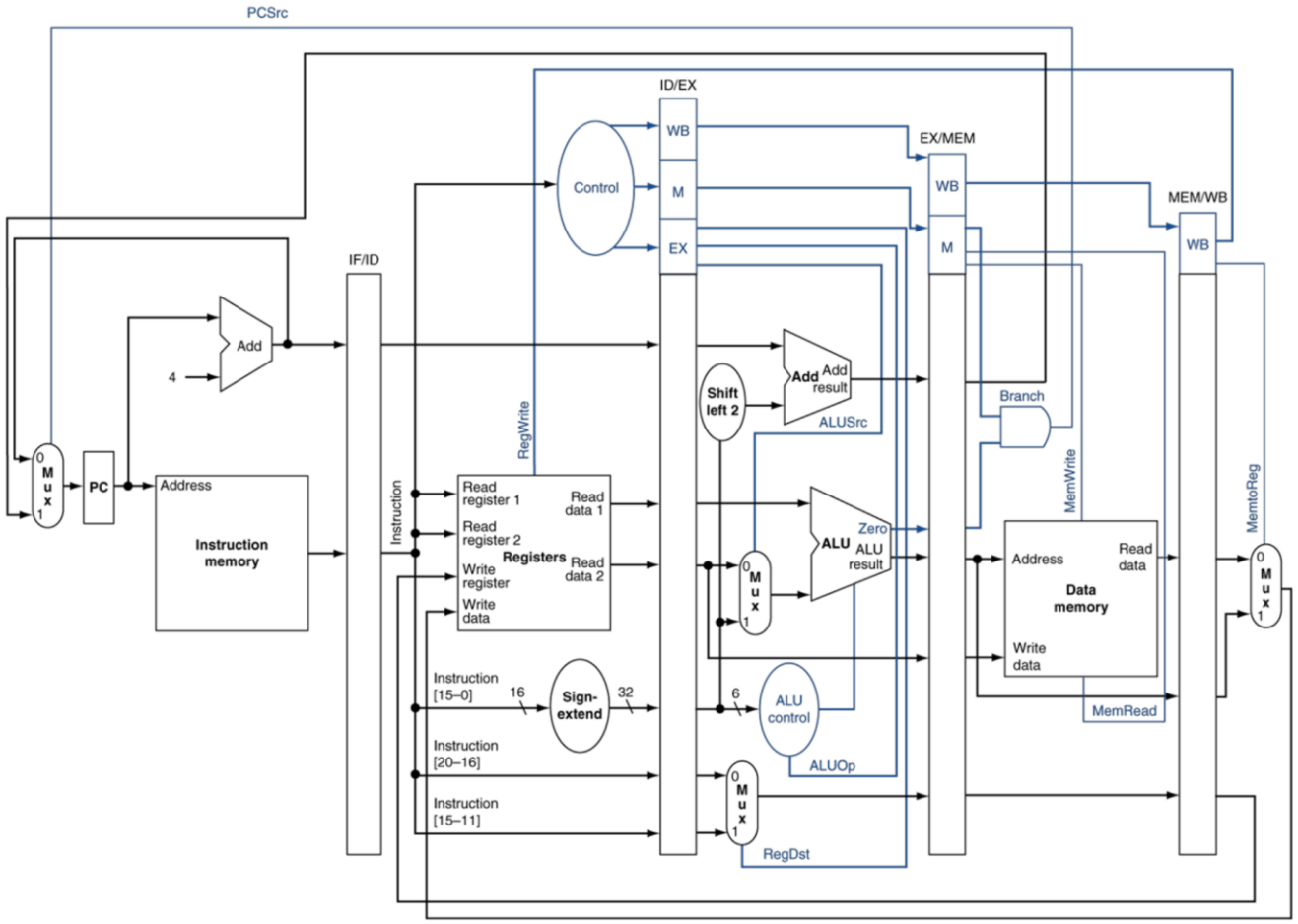
- Phasen benötigen Daten aus vorherigen Phasen
- Eingangsdaten müssen für Phase konstant anliegen

-> werden durch Zwischenspeicher gehalten

-> vor jeder Phase

-> halten vorherige Signale für den ganzen nächsten Takt

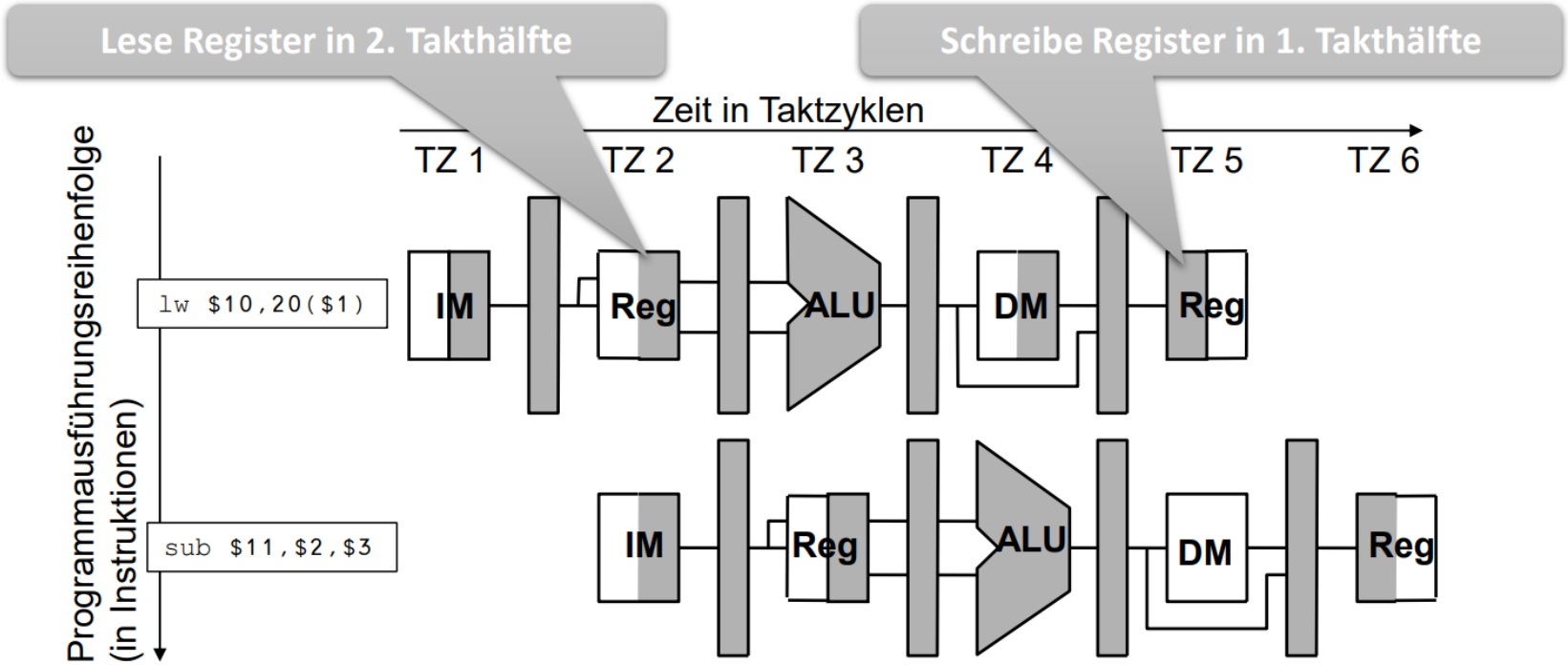
Pipelineprozessor



- Steuersignale werden bis in die Phase, wo sie gebraucht werden, weitergegebn
- Steuersignale und Eingangsdaten werden für den Takt gehalten

Write Back – Decode Überschneidung

- Aufteilung des Taktes in 2 Hälften
- Schreiben in Register zu steigender Taktflanke
- Lesen aus Register zu fallender Taktflanke



Hazards

Datenkonflikt:

- auf Ergebnis vorherigen Befehls muss gewartet werden

Steuerkonflikte:

- Bei Verzweigung falsche Befehle in Pipeline

Strukturkonflikt:

- benötigte Ressource belegt, von mehreren Befehlen benötigt
- Fehldesign

Optimierung

- Betrachte die Abhängigkeiten (z.B. durch Graph) -> erhalte die Abhängigkeiten bei der Umsortierung
- Wenn man mehere wählen kann, wähle den Befehl der:
  - keinen Konflikt mit den 2 vorher ausgeführten Instr. hat
  - wahrscheinlich einen Konflikt später erzeugt (Abhängigkeiten zu folgenden Befehlen)
  - am Weitesten von einer offensichtlich letzten Instruktion entfernt ist
  - Ansonsten nop-Befehl
- Muss nicht immer funktionieren
- Ansonsten einfach ausprobieren