

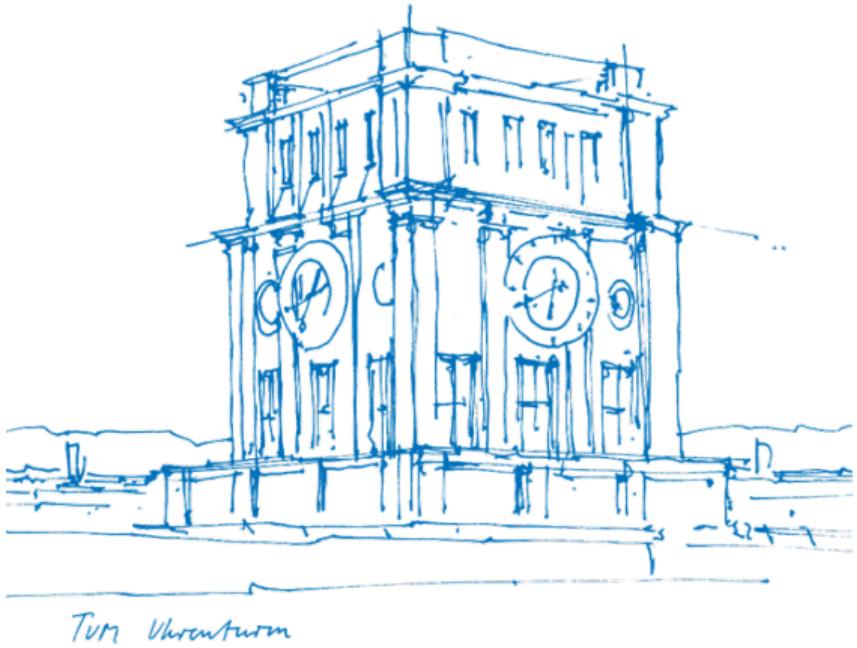
# Übung 12: Pipelining

## Einführung in die Rechnerarchitektur

Niklas Ladurner

School of Computation, Information and Technology  
Technische Universität München

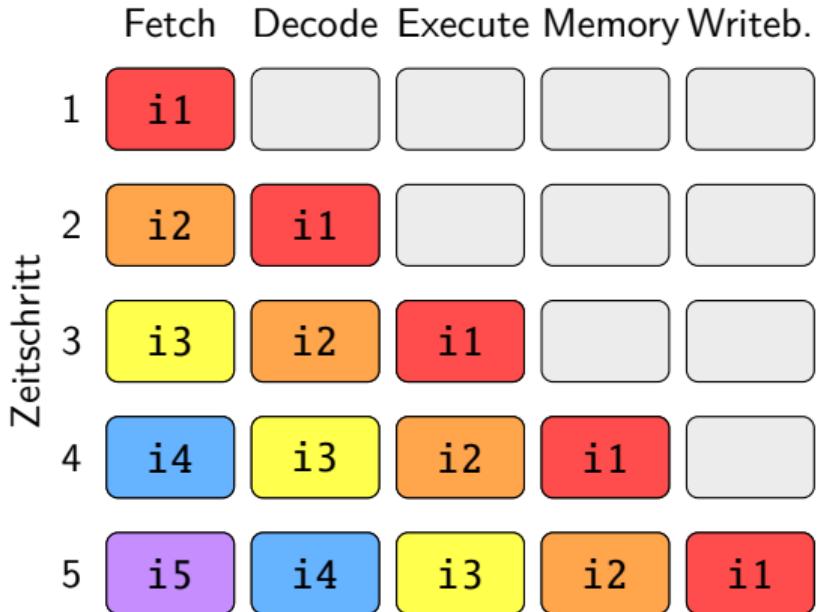
16. Januar 2026



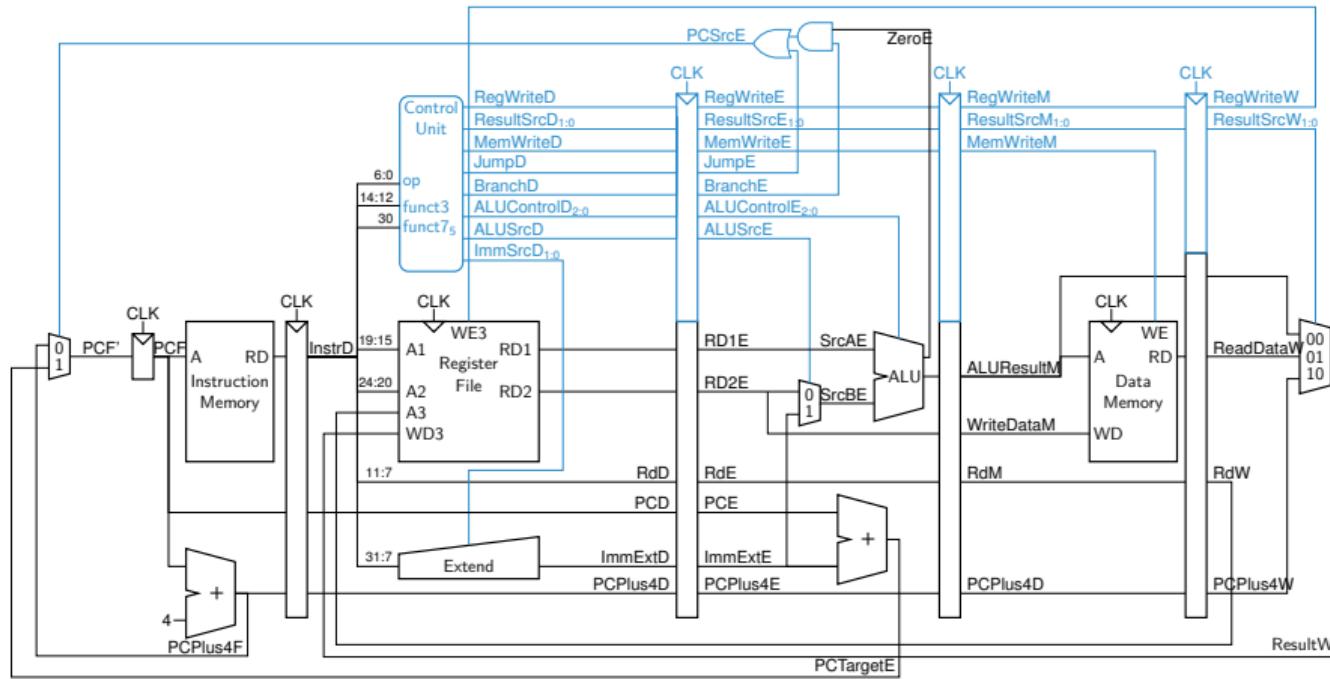
Keine Garantie für die Richtigkeit der Tutorfolien.  
Bei Unklarheiten/Unstimmigkeiten haben VL/ZÜ-Folien recht!

# Pipelining

- Parallel Verarbeitung von mehreren Instruktionen
- Aufteilung in 5 Teilschritte: Fetch, Decode, Execute, Memory, Writeback
- Daten- und Kontrollpfad des Prozessors wird aufgetrennt:  
Register zur Zwischenspeicherung
- maximaler Speedup: Anzahl  $n$  der Pipelinestufen
- .



# Pipelined Prozessor: Schaltbild



# Datenabhängigkeiten

- **RAW** (Read-After-Write): potentiell ein Problem, falls schreibende Instruktion noch nicht zurückgeschrieben hat

```
lw t0, 0(a1)  
add t2, t0, s5
```

RAW-Abhangigkeit bzgl. t0

# Datenabhängigkeiten

- **RAW** (Read-After-Write): potentiell ein Problem, falls schreibende Instruktion noch nicht zurückgeschrieben hat
- **WAR** (Write-After-Read): Reihenfolge der betroffenen Instruktionen darf nicht vertauscht werden

```
lw t0, 0(a1)  
add t2, t0, s5
```

RAW-Abhängigkeit bzgl. t0

```
xor a0, a2, a1  
sub a2, a4, a5
```

WAR-Abhängigkeit bzgl. a2

# Datenabhängigkeiten

- **RAW** (Read-After-Write): potentiell ein Problem, falls schreibende Instruktion noch nicht zurückgeschrieben hat
- **WAR** (Write-After-Read): Reihenfolge der betroffenen Instruktionen darf nicht vertauscht werden
- **WAW** (Write-After-Write): Reihenfolge der betroffenen Instruktionen darf nicht vertauscht werden

```
lw t0, 0(a1)  
add t2, t0, s5
```

RAW-Abhängigkeit bzgl. t0

```
xor a0, a2, a1  
sub a2, a4, a5
```

WAR-Abhängigkeit bzgl. a2

```
and s0, s1, s2  
lw s0, 0(s3)
```

WAW-Abhängigkeit bzgl. s0

## Data Hazards (Datenkonflikte<sup>1</sup>)

- können nur bei RAW auftreten<sup>2</sup>
- abhängige Instruktion in Decode, aber Ergebnis noch nicht zurückgeschrieben

## Control Hazards (Steuerkonflikte)

- Kontrollflussänderungen durch branches/jumps
- falsche Sprungentscheidung: falsche Instruktionen in Pipeline geladen

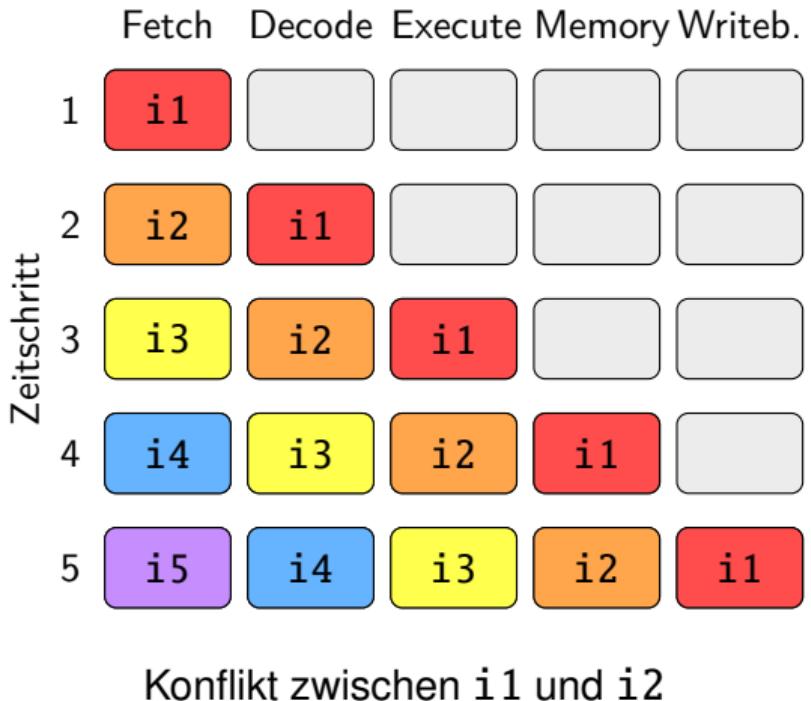
---

<sup>1</sup> Nicht zu verwechseln mit den vorher genannten Datenabhängigkeiten (RAW, WAR, WAW)

<sup>2</sup> müssen aber nicht!

# Lösung von Konflikten: Data Hazards

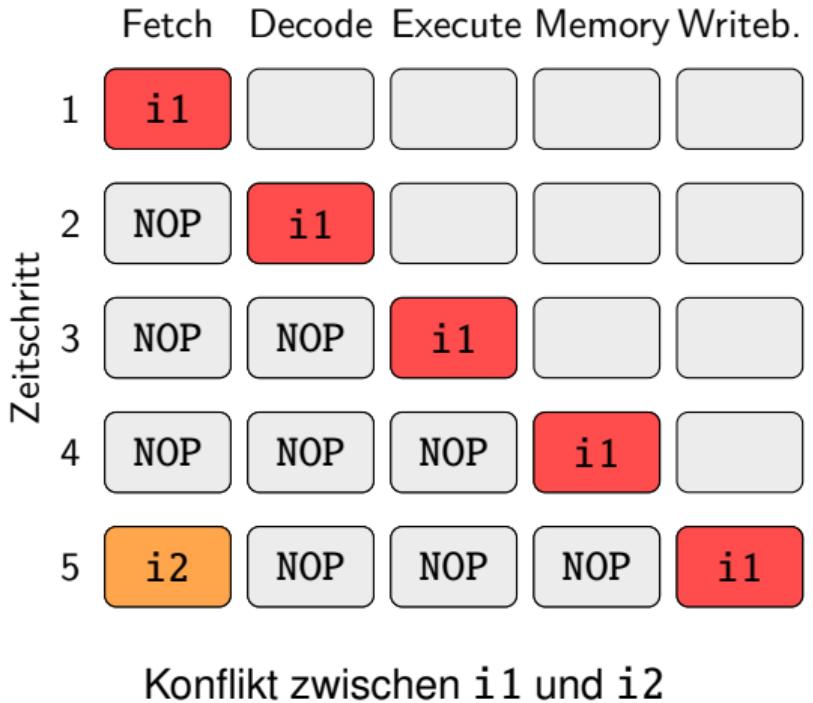
Mindestens **3 Befehle** zwischen zwei Instruktionen mit RAW-Abhängigkeit:



# Lösung von Konflikten: Data Hazards

Mindestens **3 Befehle** zwischen zwei Instruktionen mit RAW-Abhängigkeit:

- NOPs (Stalling)



# Lösung von Konflikten: Data Hazards

Mindestens **3 Befehle** zwischen zwei Instruktionen mit RAW-Abhängigkeit:

- NOPs (Stalling)
- Befehlsumordnung (ohne Änderung der Semantik)



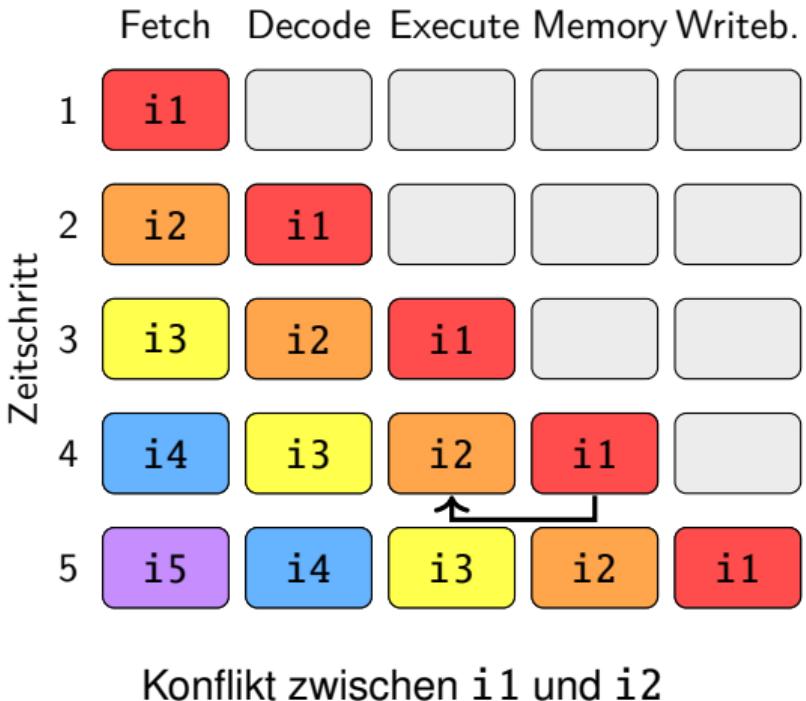
# Lösung von Konflikten: Data Hazards

Mindestens **3 Befehle** zwischen zwei Instruktionen mit RAW-Abhängigkeit:

- NOPs (Stalling)
- Befehlsumordnung (ohne Änderung der Semantik)

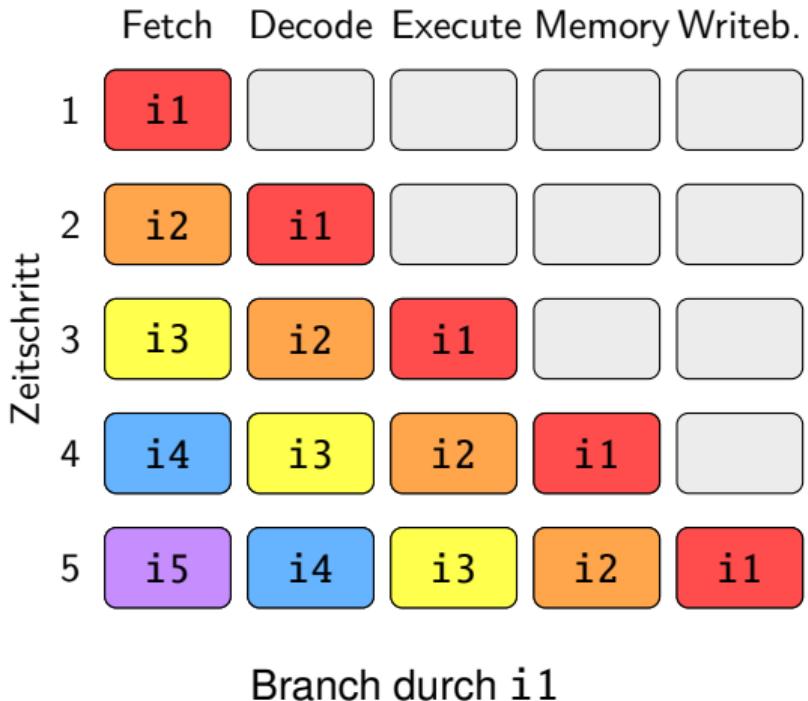
Alternativ:

- Forwarding: noch nicht zurückgeschriebenes Ergebnis kann von einer Stage in eine andere geleitet werden



# Lösung von Konflikten: Control Hazards

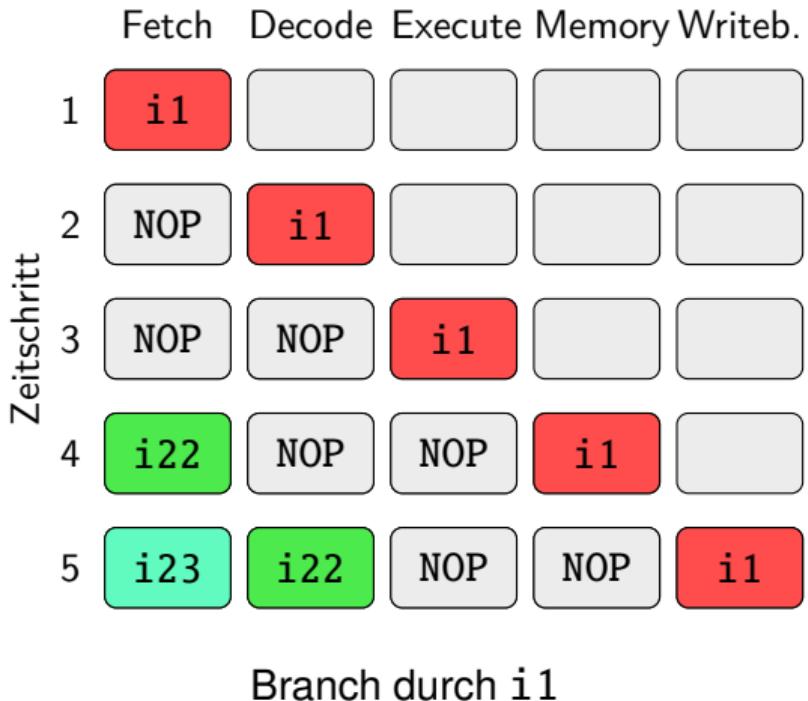
Mindestens **2 Befehle** zwischen Sprungentscheidung und möglicherweise falsch geladener Instruktion:



# Lösung von Konflikten: Control Hazards

Mindestens **2 Befehle** zwischen Sprungentscheidung und möglicherweise falsch geladener Instruktion:

- NOPs (Stalling)



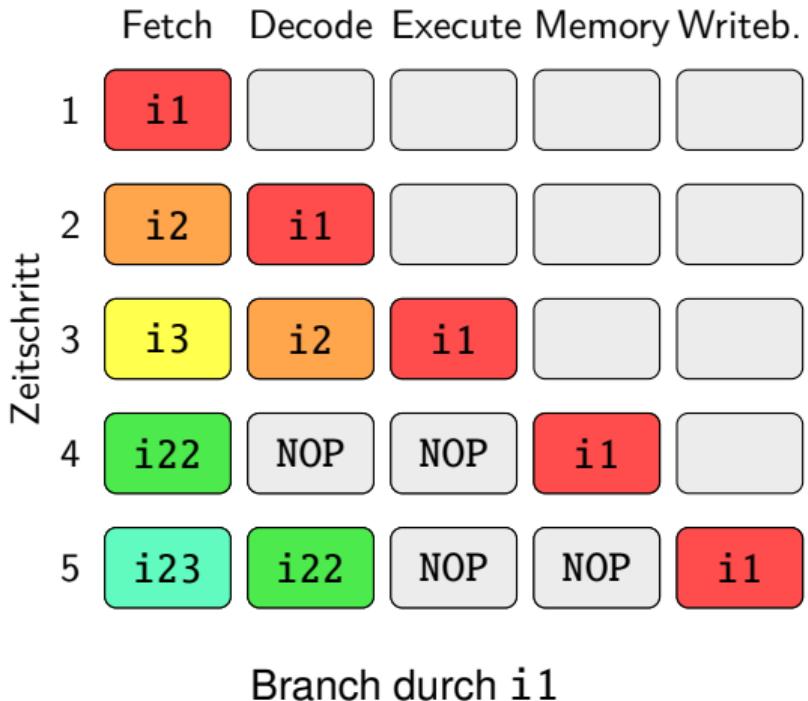
# Lösung von Konflikten: Control Hazards

Mindestens **2 Befehle** zwischen Sprungentscheidung und möglicherweise falsch geladener Instruktion:

- NOPs (Stalling)

Alternativ:

- Flushing: Falls Vorhersage der branch prediction falsch war, müssen geladene Instruktionen aus Pipeline entfernt werden



# Links

- „ERA Tutorium – Mi-1600-3“ bzw. „ERA Tutorium – Fr-1500-1“
- ERA-Moodle-Kurs
- ERA-Artemis-Kurs

# Übung 12: Pipelining

## Einführung in die Rechnerarchitektur

Niklas Ladurner

School of Computation, Information and Technology  
Technische Universität München

16. Januar 2026

