School of Computation, Information and Technology
Technische Universität München

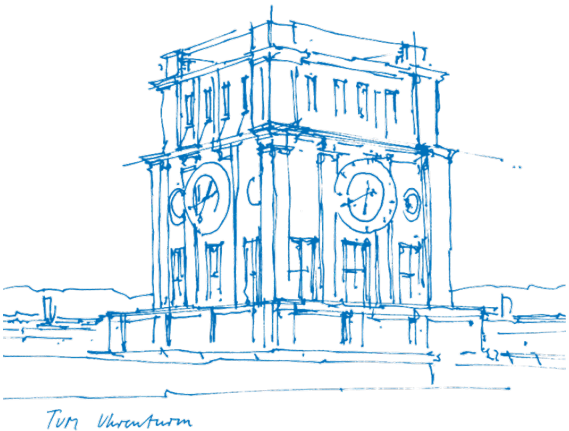
Übung 12: Pipelining

Einführung in die Rechnerarchitektur

Niklas Ladurner

School of Computation, Information and Technology
Technische Universität München

16. Januar 2026



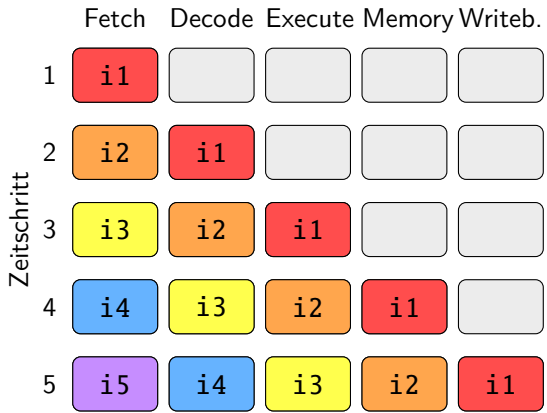
Keine Garantie für die Richtigkeit der Tutorfolien.

Bei Unklarheiten/Unstimmigkeiten haben VL/ZÜ-Folien recht!

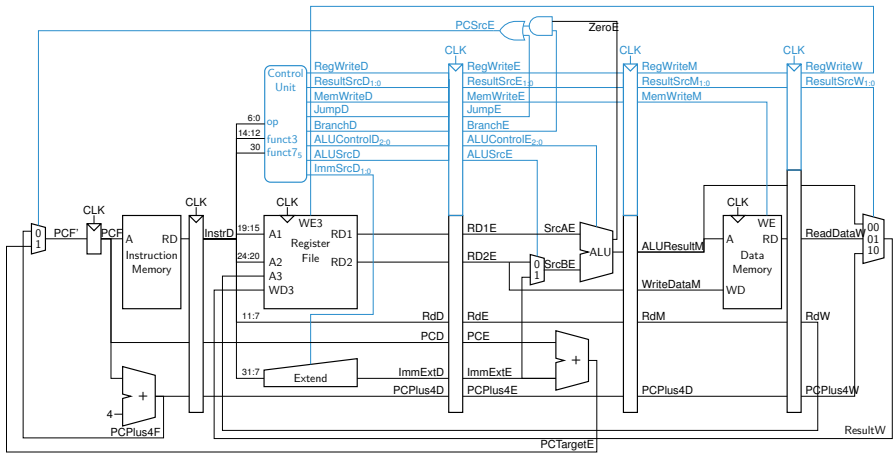
Pipelining

- Parallele 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-Abhängigkeit 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`

Pipelinekonflikte

Data Hazards
(Datenkonflikte[1])

- können nur bei RAW auftreten[2]
- abhängige Instruktion in Decode, aber Ergebnis noch nicht zurückgeschrieben

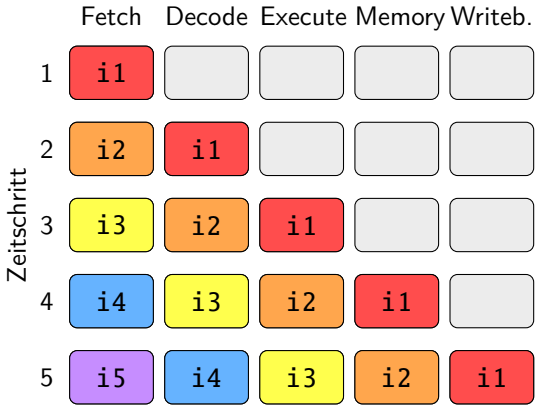
Control Hazards
(Steuerkonflikte)

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

[1] Nicht zu verwechseln mit den vorher genannten Daten*abhängigkeiten* (RAW, WAR, WAW)
[2] müssen aber nicht!

Lösung von Konflikten: Data Hazards

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

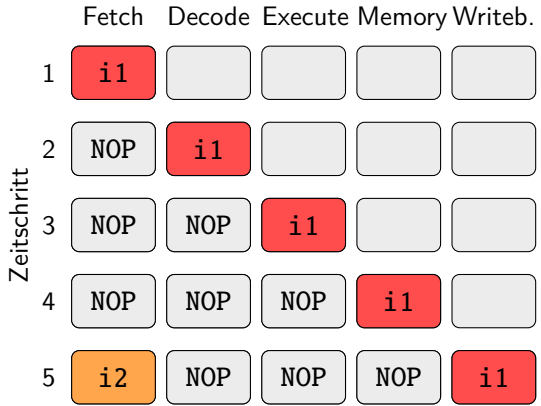


Konflikt zwischen `i1` und `i2`

Lösung von Konflikten: Data Hazards

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

- NOPs (Stalling)

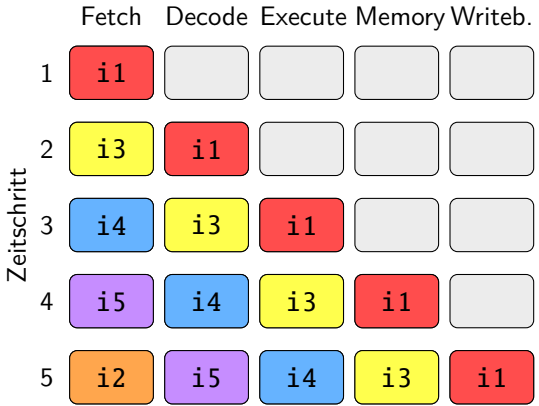


Konflikt zwischen `i1` und `i2`

Lösung von Konflikten: Data Hazards

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

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



Konflikt zwischen `i1` und `i2`

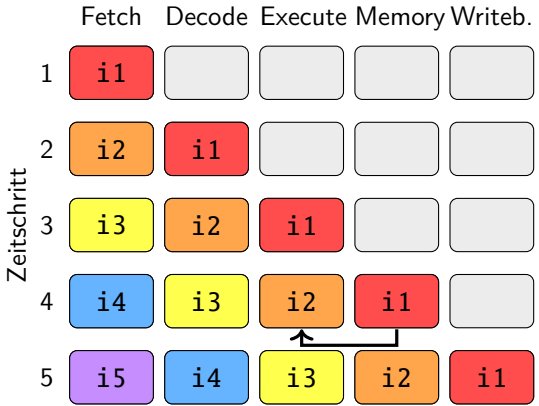
Lösung von Konflikten: Data Hazards

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

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

Alternativ:

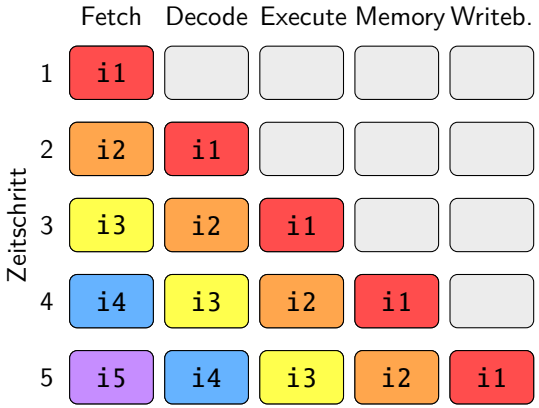
- Forwarding: noch nicht zurückgeschriebenes Ergebnis kann von einer Stage in eine andere geleitet werden



Konflikt zwischen `i1` und `i2`

Lösung von Konflikten: Control Hazards

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

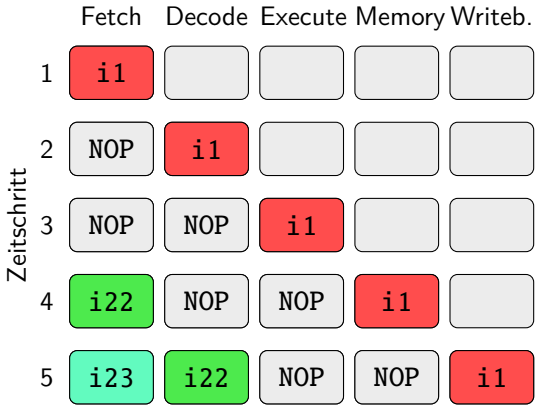


Branch durch `i1`

Lösung von Konflikten: Control Hazards

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

- NOPs (Stalling)



Branch durch `i1`

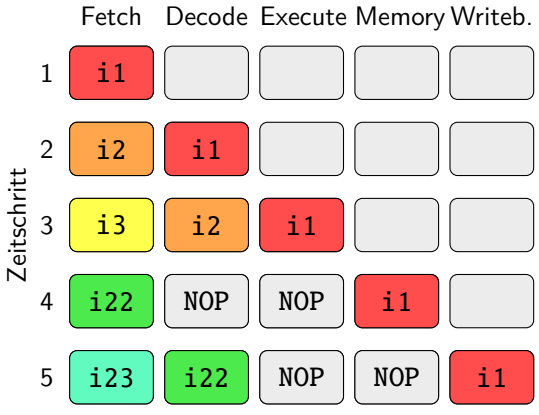
Lösung von Konflikten: Control Hazards

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



Branch durch `i1`

Fragen?

Links

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

School of Computation, Information and Technology
Technische Universität München

Übung 12: Pipelining

Einführung in die Rechnerarchitektur

Niklas Ladurner

School of Computation, Information and Technology
Technische Universität München

16. Januar 2026