School of Computation, Information and Technology
Technische Universität München

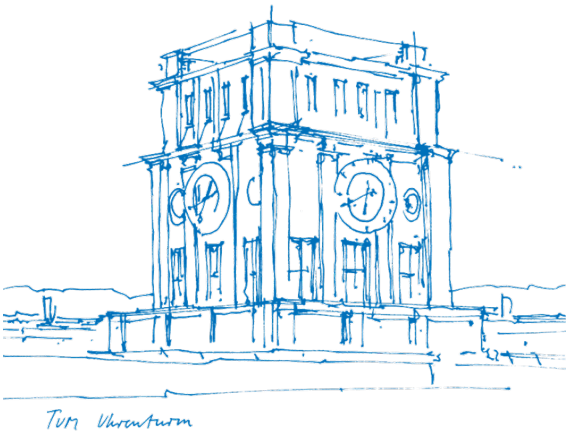
Übung 11: Pipelining

Einführung in die Rechnerarchitektur

Niklas Ladurner

School of Computation, Information and Technology
Technische Universität München

10. Januar 2025



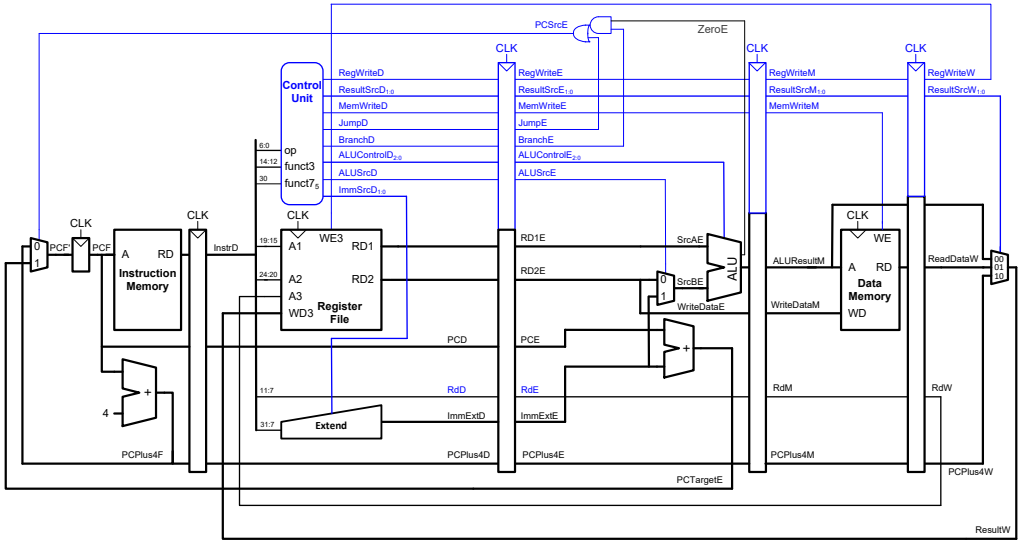
Keine Garantie für die Richtigkeit der Tutorfolien.

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

Pipelining

- Parallele Verarbeitung von mehreren Instruktionen
- Aufteilung der Instruktionsverarbeitung in 5 Teilschritte: Fetch, Decode, Execute, Memory, Writeback
- Daten- und Kontrollpfad des Prozessors wird aufgeteilt: Register zur Zwischenspeicherung dazwischen
- maximaler Speedup: Anzahl n der Pipelinestufen, Effekt bei großer Zahl an Instruktionen erkennbar

Pipelining


PCSrcE
ZeroE
CLK
CLK
CLK
Control Unit
RegWriteD
ResultSrcD1:0
MemWriteD
JumpD
BranchD
ALUControlD2:0
ALUSrcD
ImmSrcD1:0
RegWriteE
ResultSrcE1:0
MemWriteE
JumpE
BranchE
ALUControlE2:0
ALUSrcE
RegWriteM
ResultSrcM1:0
MemWriteM
RegWriteW
ResultSrcW1:0
6:0
op
14:12
funct3
30
funct75
CLK
CLK
CLK
PCF
PCF
A
RD
Instruction Memory
InstrD
19:15
24:20
A1
A2
A3
WD3
WE3
RD1
RD2
Register File
RD1E
RD2E
SrcAE
SrcBE
ALU
WriteDataE
ALUResultM
WriteDataM
CLK
WE
A
RD
Data Memory
WD
ReadDataW
PCD
PCE
RdD
RdE
RdM
RdW
11:7
31:7
Extend
ImmExtD
ImmExtE
4
PCPlus4F
PCPlus4D
PCPlus4E
PCPlus4M
PCPlus4W
PCTargetE
ResultW

Datenabhängigkeiten

```
s1: lw t0, 0(a1)
s2: lw t1, 0(a0)
s3: add t2, t0, s5
s4: xor a0, t2, a1
s5: lw a0, 0(t1)
s6: sub t2, t3, t0
```

Mit * markierte Abhängigkeiten sind Datenkonflikte

i	F	D	E	M	W
1	s1	-	-	-	-
2	s2	s1	-	-	-
3	s3	s2	s1	-	-
4	s4	s3	s2	s1	-
5	s5	s4	s3	s2	s1
6	s6	s5	s4	s3	s2
7	?	s6	s5	s4	s3
8	?	?	s6	s5	s4

Read after Write (RAW):

i. `s3, s1, t0*`

ii. `s4, s3, t2*`

iii. `s5, s2, t1`

iv. `s6, s1, t0`

Datenabhängigkeiten

```
s1: lw t0, 0(a1)
s2: lw t1, 0(a0)
s3: add t2, t0, s5
s4: xor a0, t2, a1
s5: lw a0, 0(t1)
s6: sub t2, t3, t0
```

Mit * markierte Abhängigkeiten sind Datenkonflikte

i	F	D	E	M	W
1	s1	-	-	-	-
2	s2	s1	-	-	-
3	s3	s2	s1	-	-
4	s4	s3	s2	s1	-
5	s5	s4	s3	s2	s1
6	s6	s5	s4	s3	s2
7	?	s6	s5	s4	s3
8	?	?	s6	s5	s4

Read after Write (RAW):

i. `s3, s1, t0*`

ii. `s4, s3, t2*`

iii. `s5, s2, t1`

iv. `s6, s1, t0`

Write after Read (WAR):

v. `s4, s1, a1`

vi. `s4, s2, a0`

vii. `s5, s2, a0`

viii. `s6, s4, t2`

Datenabhängigkeiten

```
s1: lw t0, 0(a1)
s2: lw t1, 0(a0)
s3: add t2, t0, s5
s4: xor a0, t2, a1
s5: lw a0, 0(t1)
s6: sub t2, t3, t0
```

Mit * markierte Abhängigkeiten sind Datenkonflikte

i	F	D	E	M	W
1	s1	-	-	-	-
2	s2	s1	-	-	-
3	s3	s2	s1	-	-
4	s4	s3	s2	s1	-
5	s5	s4	s3	s2	s1
6	s6	s5	s4	s3	s2
7	?	s6	s5	s4	s3
8	?	?	s6	s5	s4

Write after Read (WAR):

v. `s4, s1, a1`

vi. `s4, s2, a0`

vii. `s5, s2, a0`

viii. `s6, s4, t2`

Write after Write (WAW):

ix. `s5, s4, a0`

x. `s6, s3, t2`

Datenkonflikte

- Daten*abhängigkeiten*: RAW, WAR, WAW
- Pipelinekonflikte: Datenkonflikte (data hazards) und Steuerkonflikte (control hazards)
- Daten*konflikte* können nur bei RAW auftreten (müssen aber nicht): Abhängige Instruktion ist in der Execute-Phase, aber das Ergebnis wurde noch nicht zurückgeschrieben
- Steuerkonflikte treten bei Änderung der Kontrollflusses auf (branches, jumps)

Lösung von Konflikten

Bei **data hazards** müssen mindestens **3 Befehle** zwischen zwei Instruktionen mit RAW-Abhängigkeit stehen:

- NOPs (Stalling)
- Befehlsumordnung (ohne Änderung der Semantik)
- Forwarding: noch nicht zurückgeschriebenes Ergebnis kann von der ALU direkt an den nächsten Befehl gegeben werden, falls dieser das Ergebnis benötigt

Bei **control hazards** müssen mindestens **2 Befehle** zwischen der Sprungentscheidung und möglicherweise falsch geladenen Instruktion stehen.

- NOPs (Stalling)
- Branch Prediction (statisch/dynamisch): Falls Vorhersage falsch, müssen geladene Instruktionen entfernt werden

weitere Konzepte: Out-of-Order-Execution, Register Renaming, ...

Artemis-Hausaufgaben

- „H11 – Pipeline-Konflikte“ bis 19.01.2025 23:59 Uhr
- Datenabhängigkeiten auflisten und Reordering: ähnlich wie in Klausur!
- Semantik des Programms darf nicht verändert werden

Links

- Zulip: „ERA Tutorium - Do-1600-1“ bzw. „ERA Tutorium - Fr-1500-2“
- ERA-Moodle-Kurs
- ERA-Artemis-Kurs

School of Computation, Information and Technology
Technische Universität München

Übung 11: Pipelining

Einführung in die Rechnerarchitektur

Niklas Ladurner

School of Computation, Information and Technology
Technische Universität München

10. Januar 2025