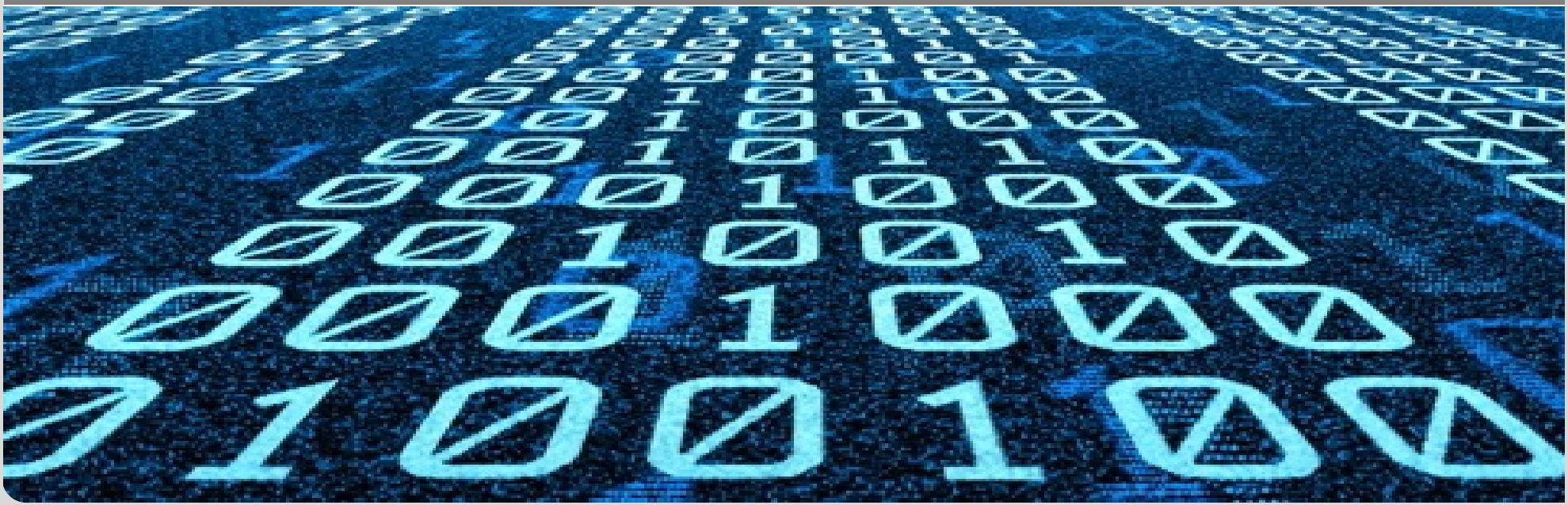
Karlsruher Institut für Technologie

6. Tutorium
DLX-Pipeline

Rechnerorganisation, Tutorium #13

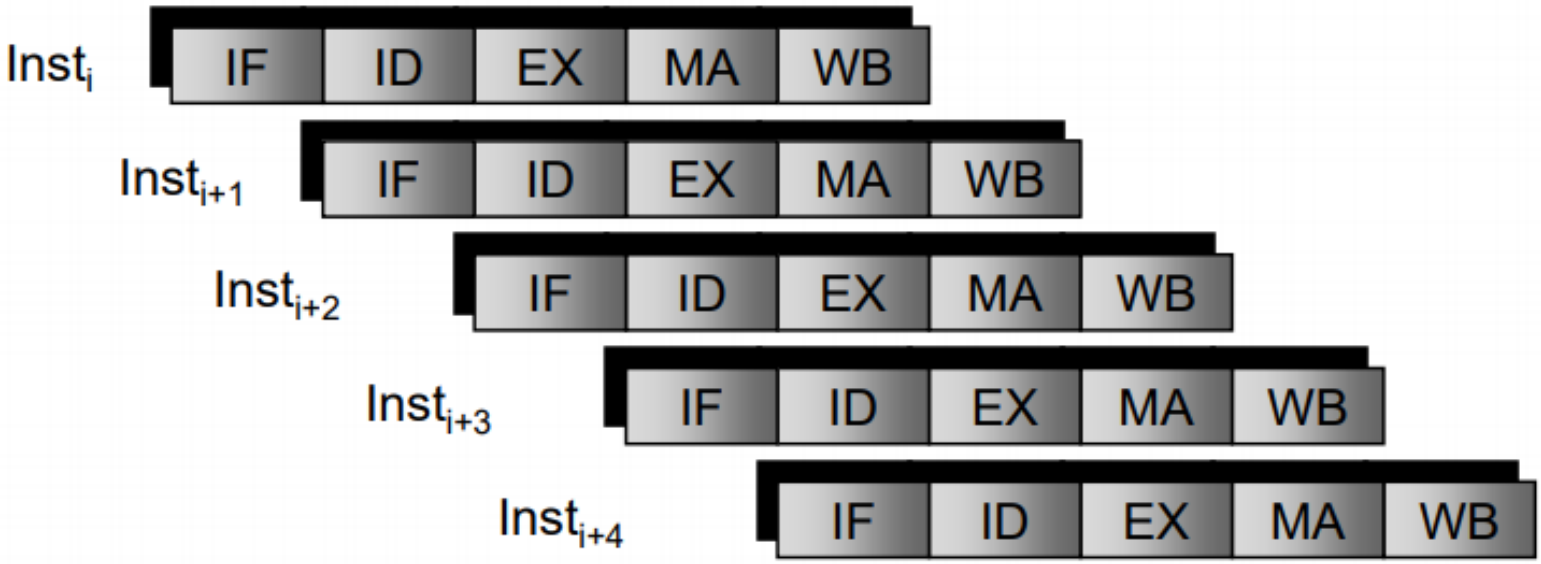
Patrick Röper | 10. Dezember 2019

FAKULTÄT FÜR INFORMATIK



Roadmap

Pipeline



Logische Phasen:
IF: Befehl holen
ID: Befehl dekodieren / Operanden bereitstellen
EX: Befehl ausführen
MA: Speicherzugriff
WB: Zurückschreiben

Pipeline-Konflikte

- Datenkonflikte
  - echte Datenabhängigkeit
  - Gegenabhängigkeit
  - Ausgabeabhängigkeit
- Steuerkonflikte
- Strukturkonflikte

Beispiel

- **Beispiel:**

```
S1:     add     r1,r2,2         #       r1 := r2 + 2
S2:     add     r4,r1,r3        #       r4 := r1 + r3
S3:     mul     r3,r5,3         #       r3 := r5 * 3
S4:     mul     r3,r6,3         #       r3 := r6 * 3
```

Beispiel

- **Beispiel:**

```
S1:     add     r1,r2,2         #       r1 := r2 + 2
S2:     add     r4,r1,r3        #       r4 := r1 + r3
S3:     mul     r3,r5,3         #       r3 := r5 * 3
S4:     mul     r3,r6,3         #       r3 := r6 * 3
```

Echte Datenabhängigkeit δ^t

Beispiel

- **Beispiel:**

```
S1:     add     r1,r2,2         #       r1 := r2 + 2
S2:     add     r4,r1,r3        #       r4 := r1 + r3
S3:     mul     r3,r5,3         #       r3 := r5 * 3
S4:     mul     r3,r6,3         #       r3 := r6 * 3
```

Gegenabhängigkeit δ^a

Beispiel

- **Beispiel:**

```
S1:     add     r1,r2,2         #       r1 := r2 + 2
S2:     add     r4,r1,r3        #       r4 := r1 + r3
S3:     mul     r3 r5,3         #       r3 := r5 * 3
S4:     mul     r3 r6,3         #       r3 := r6 * 3
```

Ausgabeabhängigkeit δ^o

Roadmap

Aufgabe 1

Aufgabe

Diskutieren Sie die folgende Situation:

Der Befehl n legt sein Ergebnis in der Speicherstelle a ab. Der nächste abzuarbeitende Befehl (n + 1) benutzt den soeben erzeugten Wert als Operanden. Zeigen Sie die Probleme auf, die dadurch entstehen, dass die beiden Befehle in einer Pipeline ausgeführt werden, die aus den obigen Bearbeitungseinheiten besteht.

Aufgabe 1

Aufgabe

Die Befehlsabarbeitung in einem Prozessor erfolgt in den folgenden Stufen:

- B: Befehl holen
- D: Befehl decodieren
- O: Operand(en) holen
- A: Befehl ausführen
- S: Ergebnis speichern

Aufgabe 1

Takt	B	D	O	A	S
1	B1				
2	B2	D1			
3	B3	D2	O1		
4	...	D3	O2[1]	A1	
5	...	...	O3	A2	S1[2]
6	...	...	...	A3	S2
7	...	...	...	...	S3

1: An dieser Stelle (also einen Takt zu früh) will der Befehl (n + 1) (hier: der zweite Befehl) auf das Ergebnis von Befehl n zugreifen. Das Ergebnis des n-ten Befehls wird jedoch erst einen Takt später gespeichert.

2: Erst an dieser Stelle wird das Ergebnis vom Befehl n (hier: der erste Befehl) an der Speicherstelle a abgelegt.

Aufgabe 2.1

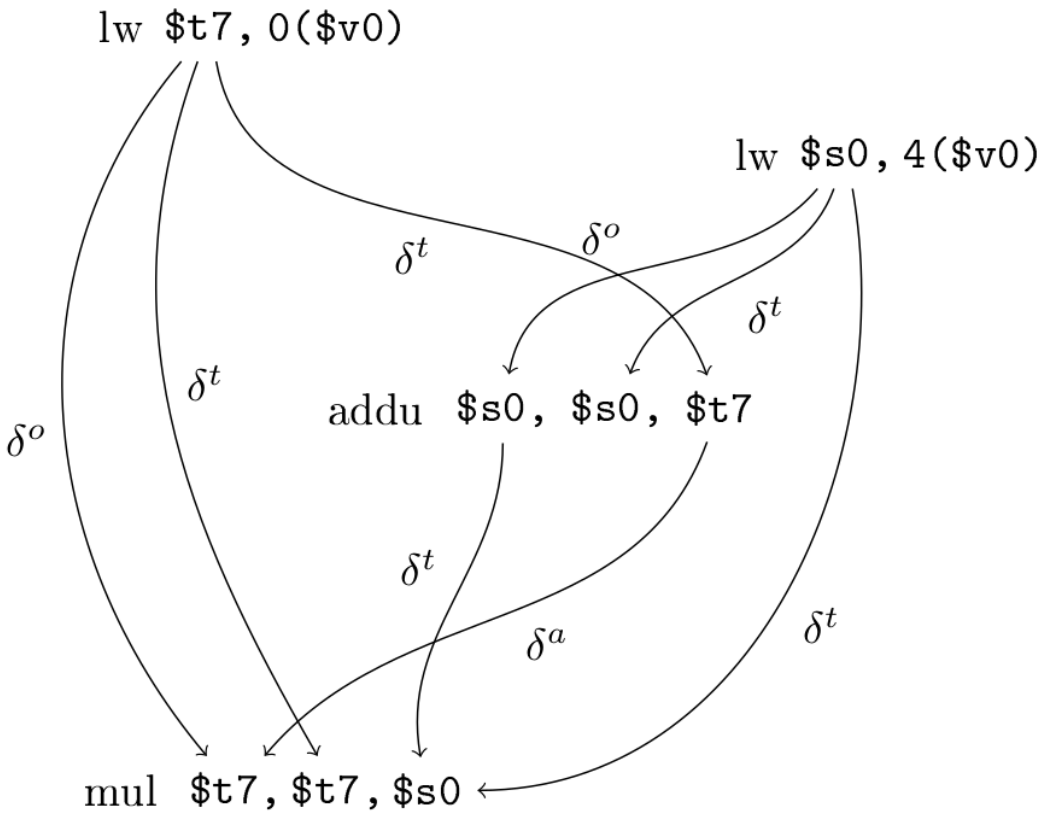
Aufgabe

Bestimmen Sie alle Daten- und Kontrollabhängigkeiten in dem gegebenen Programmstück.

```
# In Register $v0 steht die Adresse 0x10008000

lw   $t7, 0($v0)
lw   $s0, 4($v0)
add  $s0, $s0, $t7          # $s0 = $s0 + $t7
mul  $t7, $t7, $s0          # $t7 = $t7 * $s0
```

Aufgabe 2.1



Anmerkung: Echte Datenabhängigkeit δ^t, Gegenabhängigkeit δ^a ,
Ausgabeabhängigkeit δ^o
Keine Kontrollabhängigkeiten!

Aufgabe 2.2

Aufgabe

Wieviele Pipelinekonflikte treten auf?

Begründen Sie Ihre Antwort.

Aufgabe 2.2

Aufgabe

Wieviele Pipelinekonflikte treten auf?

Begründen Sie Ihre Antwort.

Lösung

Es gibt vier Pipelinekonflikte.

Aufgabe 2.3

Aufgabe

Unter der Voraussetzung, dass die auftretenden Pipelinekonflikte von der Hardware nicht erkannt werden, müssen die Pipelinekonflikte vom Compiler behandelt werden. Ergänzen Sie obiges Programmstück, so dass die auftretenden Pipelinekonflikte berücksichtigt werden.

Aufgabe 2.3

Aufgabe

Unter der Voraussetzung, dass die auftretenden Pipelinekonflikte von der Hardware nicht erkannt werden, müssen die Pipelinekonflikte vom Compiler behandelt werden. Ergänzen Sie obiges Programmstück, so dass die auftretenden Pipelinekonflikte berücksichtigt werden.

Lösung

```
# In Register $v0 steht die Adresse 0x10008000
lw $t7, 0($v0)
lw $s0, 4($v0)
nop
nop
add $s0, $s0, $t7 # $s0 = $s0 + $t7
nop
nop
mul $t7, $t7, $s0 # $t7 = $t7 * $s0
```

Aufgabe 2.4

Aufgabe

Welche der von Ihnen getroffenen Maßnahmen im letzten Aufgabenteil sind noch erforderlich, falls die auftretenden Pipelinekonflikte von der Hardware erkannt und durch Load-Forwarding und Result- Forwarding behandelt werden.

Aufgabe 2.4

Aufgabe

Welche der von Ihnen getroffenen Maßnahmen im letzten Aufgabenteil sind noch erforderlich, falls die auftretenden Pipelinekonflikte von der Hardware erkannt und durch Load-Forwarding und Result- Forwarding behandelt werden.

Lösung

```
# In Register $v0 steht die Adresse 0x10008000
lw $t7, 0($v0)
lw $s0, 4($v0)
nop
add $s0, $s0, $t7 # $s0 = $s0 + $t7
mul $t7, $t7, $s0 # $t7 = $t7 * $s0
```

Was ihr jetzt kennen und können solltet...

- Einführung in die Grundlagen der Pipelineverarbeitung (Definitionen, Speedup und Durchsatz)
- Datenpfad der DLX-Pipeline
- Daten- und Steuerflussabhängigkeiten und -konflikte
- Software- und Hardware-Behebungsmaßnahmen für Pipeline-Konflikte

Das wars