Karlsruhe Institute of Technology
Institut für Anthropomatik und Robotik (IAR)

Rechnerorganisation im WS 2017/18

Musterlösungen zum 6. Übungsblatt

Prof. Dr. Wolfgang Karl
Haid-und-Neu-Str. 7

Dr.-Ing. Ömer Terlemez
Adenauerring 2, Geb. 50.20

Email: ti@ira.uka.de
Web: http://ti.ira.uka.de

Lösung 1 (11 Punkte)

1. 4 P.

   - Echte Abhängigkeiten:

     $S1 \rightarrow S3$ $S1 \rightarrow S5$ $S1 \rightarrow S6$ $S1 \rightarrow S7$ $S1 \rightarrow S9$ $S1 \rightarrow S10$
     $S2 \rightarrow S4$ $S3 \rightarrow S4$ $S4 \rightarrow S5$ $S6 \rightarrow S7$ $S6 \rightarrow S9$ $S6 \rightarrow S10$
     $S7 \rightarrow S8$ $S9 \rightarrow S10$

   - Steuerflussabhängigkeiten:

     $S8 \rightarrow S9$

2. Behebung der Pipelinekonflikte: 3 P.

```
S1:              add  $t1, $zero, $zero
S2:              lw   $t3, 0x1500($zero)
S3:      loop:   lw   $t4, 0x5000($t1)
                 nop
S4:              add  $t5, $t4, $t3
S5:              sw   $t5, 0x400($t1)
S6:              addi $t1, $t1, 4
S7:              subi $t2, $t1, 0x400
S8:              bnez $t2, loop
                 nop
                 nop
                 nop
S9:      end:    srli $t1, $t1, 2
S10:             sw   $t1, 0x2000($zero)
```

3. Struktur- oder Ressourcenkonflikte: 2 P.

   Treten auf, wenn zwei oder mehrere Pipeline-Stufen gleichzeitig dieselbe Ressource benötigen, auf diese aber nur einmal zugegriffen werden kann.

   Sie können bei der DLX-Pipeline nicht auftreten, da diese entsprechend entworfen ist.

4. Ausgabeabhängigkeiten (*Output dependence*) und Gegenabhängigkeiten (*Anti-dependence*) können in der DLX-Pipeline nicht zu Konflikten führen, da 1 P.

   - das Lesen aus Registern immer in der Stufe 2 (ID/RF) und
   - das Schreiben in Register immer in der Stufe 5 (WB) erfolgt.

5. In den häufig in Programmen auftretenden Schleifen wird die Mehrzahl der Sprünge genommen. Falls ein Sprungbefehl genommen wird, müssen die drei Befehle hinter dem Sprungbefehl aus der Pipeline gelöscht werden (*pipeline flushing*). 1 P.

Lösung 2 (13 Punkte)

1. Echte Datenabhängigkeiten im Programmstück 3 P.

   - S1-S5 (`$s0`)
   - S2-S3 (`$t0`)
   - S2-S4 (`$t0`)
   - S2-S7 (`$t0`)
   - S3-S4 (`$t0`)
   - S3-S7 (`$t0`)
   - S4-S5 (`$s1`)
   - S4-S6 (`$s1`)
   - S5-S6 (`$s1`)
   - S6-S7 (`$s2`)

2. Bei der folgenden Tabelle wird angenommen, dass Schreibvorgänge in den Registersatz im jeweiligen Taktzyklus bereits abgeschlossen werden. 3 P.

	IF	ID	EX	MEM	WB	`$s0`	`$s1`	`$t0`
1	addi	(nop)	(nop)	(nop)	(nop)	0	0	0
2	lui	addi	(nop)	(nop)	(nop)	0	0	0
3	ori	lui	addi	(nop)	(nop)	0	0	0
4	lw	ori	lui	addi	(nop)	0	0	0
5	add	lw	ori	lui	addi	100	0	0
6	srl	add	lw	ori	lui	100	0	0x20140000
7	sw	srl	add	lw	ori	100	0	0x00001234
8	(nop)	sw	srl	add	lw	100	⊛	0x00001234
9	(nop)	(nop)	sw	srl	add	100	100	0x00001234
10	(nop)	(nop)	(nop)	sw	srl	100	100	0x00001234
11	(nop)	(nop)	(nop)	(nop)	sw	100	100	0x00001234

   ⊛: Datenwort an Adresse $0x00000000$ (undefiniert)

3. DLX-Pipeline ohne Forwarding: 2 P.

```
S1
S2
NOP
NOP
S3
NOP
NOP
S4
NOP
NOP
S5
NOP
NOP
S6
NOP
NOP
S7
```

4. Modifizierte DLX-Pipeline mit Result Forwarding und Load Forwarding: 2 P.

```
S1
S2
S3
S4
NOP
S5
S6
S7
```

5. Berechnung der Ausführungszeit: 3 P.

   - Sequentielle Ausführung: $7 \cdot 5 = 35$ Taktzyklen
   - DLX-Pipeline ohne Forwarding: $17 + 4 = 21$ Taktzyklen
   - DLX-Pipeline mit Forwarding: $8 + 4 = 12$ Taktzyklen

Lösung 3 (3 Punkte)

- Echte Abhängigkeit (*true dependency*):

```
lb $s4, 0x12345678
sub $s3, $s4, $s5
```

- Gegenabhängigkeit (*anti dependency*):

```
sub $s3, $s4, $s5
ori $s4, $zero, 42
```

- Ausgabeabhängigkeit (*output dependency*):

```
lb $s4, 0x12345678
ori $s4, $zero, 42
```

  Betroffen ist jeweils das Register `$s4`.