



Prof. Dr. Rolf Drechsler, drechsler@informatik.uni-bremen.de, MZH 3510
Dr. Robert Wille, rwille@informatik.uni-bremen.de, MZH 3485
Oliver Keszöcze, keszocze@informatik.uni-bremen.de, MZH 3440

3. Übungsblatt zur Vorlesung

Technische Informatik 1

Aufgabe 1

(4 Punkte)

Vereinfache die Ausdrücke des folgenden Programmstücks so weit wie möglich unter der Annahme, dass

- lediglich der Wert aus Register $\$t4$ in zeitlich auf S_6 folgenden Instruktionen verwendet wird und
- für die zwei Konstanten a und b immer $a > 0$ und $b > 0$ gilt.

S_0	li	$\$t1, a$	# $\$t1 := a$ (konstant)
S_1	li	$\$t2, b$	# $\$t2 := b$ (konstant)
S_2	sllv	$\$t3, \$t1, \$t2$	# $\$t3 := \$t1 \ll \$t2$
S_3	move	$\$t4, \$t2$	# $\$t4 := \$t2$
S_4	add	$\$t4, \$t4, \$t1$	# $\$t4 := \$t4 + \$t1$
S_5	subi	$\$t2, \$t2, 1$	# $\$t2 := \$t2 - 1$
S_6	bgtz	$\$t2, S_4$	# do ... while($\$t2 > 0$)

Begründe deine Überlegungen.

Aufgabe 2

(4+2 Punkte)

Unter der Annahme, dass in einem sequentiellen Programm mit n Anweisungen die Anweisung S_i vor der Anweisung S_j mit $0 \leq i < j \leq n - 1$ steht, kann man *Datenabhängigkeiten* folgendermaßen klassifizieren:

- *True Dependence* (read after write): S_j liest eine Variable, die in S_i beschrieben wird.
 - *Anti Dependence* (write after read): S_j schreibt auf eine Variable, die in S_i gelesen wird.
 - *Output Dependence* (write after write): S_j schreibt auf eine Variable, die auch in S_i beschrieben wird.
- a) Betrachte das Programmstück aus Aufgabe 1. Bestimme alle Datenabhängigkeiten in diesem Programmstück (die dort genannten Annahmen gelten hier *nicht*). Unterscheide dabei zwischen True, Anti und Output Dependencies.
- b) Wie kann die Information über Datenabhängigkeiten genutzt werden?

Aufgabe 3

(3 + 3 Punkte)

Betrachte noch einmal das Programmstück aus Aufgabe 1 (ohne die dort genannten Annahmen). Die einzelnen Befehle werden in einer fünfstufigen Befehlspipeline (Befehl holen, Befehl dekodieren, Operanden holen, Operation ausführen, Operand speichern) verarbeitet. Erst am Ende der Operand-speichern-Phase ist ein Schreibvorgang in das entsprechende Zielregister abgeschlossen. Bei Sprungbefehlen wird der Befehlszähler während der Operation-ausführen-Phase auf den neuen Wert gesetzt.

- a) Wie viele und welche Pipeline-Konflikte können auftreten?
- b) Behebe alle Pipeline-Konflikte durch Einfügen einer minimalen Anzahl von NOP-Befehlen.

Aufgabe 4

(2 + 2 Punkte)

- a) Zur Lösung von Control-Hazards wird Branch-Prediction verwendet. In welchen Fällen greift dieses Verfahren zum Vermeiden von Pipeline-Konflikten besonders gut? Gib ein Beispiel an! Was ist der Nachteil bei der Verwendung von Branch-Prediction?
- b) Welche softwareseitigen Lösungen existieren für Data-Hazards? Wo werden diese realisiert? Gibt es auch hardwareseitige Lösungen?

Abgabetermin: vor Beginn der Vorlesung am 2. Mai 2013