Saarland University
Department 6.2 - Computer Science
Prof. Dr. W. J. Paul
Jonas Oberhauser

System Architecture - SS15

Exercise Sheet 7(due: June 8, 2015)

Wichtig:

- Sie Benötigen 50% aller Übungsblätter die für Klausur X relevant sind, um zu Klausur X zugelassen zu werden. Dieses Blatt ist Relevant für Haupt- und Nachklausur.
- Das Übungsblatt muss stets am Montag nach der Vorlesung bei mir in der Office Hour oder, falls zeitgleich, in der Übungsgruppe Ihrer Tutorin abgegeben werden.
- Geben Sie stets Ihren Namen, Ihre Matr. Nr., und den Namen ihrer Tutorin auf der vordersten Seite oben rechts an.
- Sie dürfen Ergebnisse von vorherigen Aufgaben verwenden, auch wenn Sie diese nicht gelöst haben. Markieren sie Gleichungen, in denen Sie ein vorheriges Ergebniss benutzen, mit dem Kürzel E+Aufgabenblatt+Aufgabennummer.
- Wenn Sie sich nicht für die Klausur vorbereiten möchten, aber trotzdem zugelassen werden möchten, schreiben Sie einfach Ihren Namen und Ihre Matrikelnummer auf die Lösung einer kompetenten Mitstudentin. Es besteht auch keine Anwesenheitspflicht in den Übungsgruppen.

Tutor: _____

Namen, Matr. Nummern: _____

Aufgabe 1: (2)

Wir möchten den Prozessor aus der Vorlesung so ändern, dass wir in jedem Hardwarezyklus genau eine Instruktion ausführen.

(a) (1 point) Welche Hardware-Komponenten können wir wiederverwenden, welche müssen wir ändern und wie?

(b) (1 point) Wie ändert sich die Simulationsrelation?

Solution: Das Execute-Register muss weg, wir brauchen einen 2-Port Ram um gleichzeitig Instruktionen laden und Speicherzugriffe ausführen zu können. Das Instruction-Register wird entfernt. Gleich bleiben alle anderen Komponenten: ALU, PC, GPR, ROM, Shifter, Instruction Decoder... Wobei natürlich der Wert des Execute-Registers keine Rolle mehr spielt (PC und GPR werden immer geclockt).

Die Simulationsrelation zwischen zwei Berechnungen vergleicht nun c^i, h^i.

Aufgabe 2: (4)

Geben Sie zwei Kontext-Freie Grammatiken für Boolsche Ausdrücke ohne Funktionssymbole an, e.g.,

$$x_1 \vee x_2,\ x_2 \wedge (x_2 \vee x_1),\ \overline{x_2} \vee x_1$$

($\oplus$ müssen sie nicht beachten, dürfen Sie aber. Die Prezedenz dürfen Sie frei wählen.)

(a) (1 point) Ihre erste Grammatik soll uneindeutig sein. Beweisen Sie, dass Ihre Grammatik uneindeutig ist.

Saarland University
Department 6.2 - Computer Science
Prof. Dr. W. J. Paul
Jonas Oberhauser

System Architecture - SS15

Exercise Sheet 7(due: June 8, 2015)

(b) (1 point) Ihre zweite Grammatik soll eindeutig sein. Warum ist das wichtig?

(c) (1 point (bonus)) Beweisen sie, dass Ihre zweite Grammatik eindeutig ist.

(d) (1 point) Definieren Sie eine Funktion, die Ableitungsbäume Ihrer Grammatik korrekt auf boolesche Ausdrücke abbildet. Insbesondere sollte Ihre Funktion Ableitungsbäume des Texts von e auf e abbilden.

(e) (1 point) Leiten Sie in Ihrer Grammatik den Ausdruck

$$x_0 \vee x_1 \wedge x_2$$

ab (Baumregion, Labels. Graphisch ist auch O.K., solange die Baumregion und Labels alle zu erkennen sind).

Solution: (XOR ist nicht notwendig).

Uneindeutige Grammatik:

$$\langle BE\rangle ::= \langle BE\rangle \wedge \langle BE\rangle | \langle BE\rangle \vee \langle BE\rangle | \langle BE\rangle \oplus \langle BE\rangle | \overline{\langle BE\rangle} | (\langle BE\rangle) | \mathrm{x}_i$$

$$x_0 \vee x_1 \wedge x_2$$

kann als

$$A_1 = \{\epsilon, (0), (0,0), (0,1), (0,2), (1), (2)\}$$

und

$$A_2 = \{\epsilon, (0), (1), (2), (2,0), (2,1), (2,2)\}$$

abgeleitet werden mit

$$L_1(a) = \begin{cases} x_0 & a = (0,0) \\ \vee & a = (0,1) \\ x_1 & a = (0,2) \\ \wedge & a = (1) \\ x_2 & a = (2) \\ \langle BE\rangle & a \in \{\epsilon, (0)\}, \end{cases}$$

und

$$L_2(a) = \begin{cases} x_0 & a = (0) \\ \vee & a = (1) \\ x_1 & a = (2,0) \\ \wedge & a = (2,1) \\ x_2 & a = (2,2) \\ \langle BE\rangle & a \in \{\epsilon, (2)\}. \end{cases}$$

Diesen Ableitungen entsprechen unterschiedliche boolesche Ausdrücke, die auch unterschiedlich auswerten (z.B. für $x_0 = 1, x_1 = 0, x_2 = 0$). Insbesondere respektieren die

Saarland University
Department 6.2 - Computer Science
Prof. Dr. W. J. Paul
Jonas Oberhauser

System Architecture - SS15

Exercise Sheet 7(due: June 8, 2015)

Ableitungen nicht zwingenderweise Prezedenzregeln für logische Operatoren. Da ein Kompiler nicht nur syntaktisch richtige Programme erkennen, sondern auch übersetzen soll, ist es hilfreich wenn die Semantik der Programmiersprache eindeutige Prezedenzregeln definiert, und der Kompiler diese Regeln auch anwenden kann. Eine eindeutige Grammatik kann dieses Problem für uns lösen, wenn sie ausschliesslich Ableitungen erlaubt, die den Prezedenzregeln folgen.

Eindeutige Grammatik (Beweis analog zum Beweis für arithmetische Ausdrücke):

$$\begin{aligned}
\langle BF\rangle &::= \mathrm{x}_i|\overline{\langle BF\rangle}|(\langle BE\rangle)\\
\langle BT\rangle &::= \langle BF\rangle|\langle BT\rangle \wedge \langle BF\rangle\\
\langle BO\rangle &::= \langle BT\rangle|\langle BO\rangle \oplus \langle BT\rangle\\
\langle BE\rangle &::= \langle BO\rangle|\langle BE\rangle \vee \langle BO\rangle
\end{aligned}$$

Diese Grammatik liefert folgenden Ableitungsbaum:

$$\begin{aligned}
A = \{&\epsilon, (0), (1), (2), (0,0), (0,0,0), (0,0,0,0), (0,0,0,0,0),\\
&(2,0), (2,0,0), (2,0,0,0), (2,0,0,0,0), (2,0,1), (2,0,2), (2,0,2,0)\}
\end{aligned}$$

mit

$$L(a) = \begin{cases}
\langle BE\rangle & a \in \{\epsilon, (0)\}\\
\langle BO\rangle & a \in \{(0,0), (2)\}\\
\langle BT\rangle & a \in \{(0,0,0), (2,0), (2,0,0)\}\\
\langle BF\rangle & a \in \{(0,0,0,0), (2,0,0,0), (2,0,2)\}\\
x_0 & a = (0,0,0,0,0)\\
x_1 & a = (2,0,0,0,0)\\
x_2 & a = (2,0,2,0)\\
\vee & a = (1)\\
\wedge & a = (2,0,1)
\end{cases}$$

Aufgabe 3: (1)

Wie lange dauert folgende Berechnung des Quadrats einer Zahl im Schlimmsten Fall? Als Grundlage der Berechnung können Sie annehmen, dass das Programm auf dem MIPS-Prozessor aus der Vorlesung läuft, der mit 10^{10} Taktzyklen pro Sekunde betrieben wird (das sind 10 GHz).

Saarland University
Department 6.2 - Computer Science
Prof. Dr. W. J. Paul
Jonas Oberhauser

System Architecture - SS15

Exercise Sheet 7(due: June 8, 2015)

	$MOV(0,5);$	$sum_0 = 0$
	$MOV(3,4);$	$rem_0 = x$
$LOOP:$	$beqz\ 4\ DONE;$	until $rem_i = 0$, INV: $sum_i + rem_i \cdot x = x^2$
	$addu\ 5\ 5\ 3;$	$sum_{i+1} = sum_i + x$
	$addi\ 4\ 4\ -1;$	$rem_{i+1} = rem_i - 1$
	$j\ LOOP$	
$DONE:$		

Solution: Schlimmstenfalls gilt $x = 2^{32} - 1$. In diesem Fall beträgt die Zahl der MIPS-Schritte mindestens $(2^{32} - 1) \cdot 4 \approx 2^{34}$, die Zahl der Prozessorschritte $2 \cdot 2^{34}$, und damit die gesamte Berechnung etwa 3,4 Sekunden:

$$\frac{2^{35}}{10^{10}} \approx 3.44.$$

Aufgabe 4: (1)

Zeigen Sie: Für alle Baumregionen A und Knoten $a \circ i \in A$ gilt

$$a \in A.$$

Solution: Induktion über die Baumregion A. Wenn $A = \{\epsilon\}$ gilt $a \circ i = \epsilon$ was nicht sein kann.

Ansonsten gibt es A' und $a' \in A'$ sodass

$$A = A' \cup \{a' \circ 0, \ldots, a' \circ (k-1)\},$$

und entweder $a \circ i \in A'$ und per Induktionshypothese $a \in A' \subseteq A$, oder $a = a'$ und $i \in [0 : k-1]$ und $a = a' \in A' \subseteq A$.

Aufgabe 5: (2)

Zeigen Sie, dass wir in der Arithmetischen Grammatik nicht zwischen unärem und binärem Minus unterscheiden müssen.

Solution: Das binäre Minus steht stets rechts von einem Operanden, das unäre Minus niemals. Also sind die Operationen einfach zu unterscheiden.