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

System Architecture - SS15

Exercise Sheet 12(due: July 13, 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: (6)

Erweitern Sie den Compiler aus der Vorlesung um Array-Bounds-Checking. D.h., komplieren Sie in der Ausdruckskomplierung Arrayzugriffe $e[e']$ so, dass bei $va(e', c) \notin [0 : n-1]$ mit

$$etype(e, c) = t[n]$$

die Instruktion *sysc* aufgerufen wird.

Solution: Nachdem wir die Ausdrücke e, e' kompiliert haben, machen wir folgenden Test, wobei j'' ein freies Register ist:

```
gpr(j'') = enc(n);
slt j'' j' j'' ; // j'' <=> e' < n
bgtz j'' +2; // unless e' < n: sysc
sysc
slti j'' j' 0 ; // j'' <=> e' < 0
blez j'' +2; // if e' < 0: sysc
sysc
```

Aufgabe 2: (6)

Definieren Sie die Kompilation des return-statements!

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

System Architecture - SS15

Exercise Sheet 12(due: July 13, 2015)

Solution: Steht im Buch.

Bonus Aufgabe 3: (12)

Beweisen Sie dass wenn

$$consis(d, c),$$

dann auch

$$consis(nc_{\mathrm{M}}(d), \delta_{\mathrm{C}}(c)).$$

Solution: Wir skizzieren den Beweis nur. Es gilt zu zeigen

1. es wird irgendwann ein Konsistenzpunkt erreicht
2. der pc an diesem Konsistenzpunkt ist die Startaddresse des head vom program rest der nächsten C0-Konfiguration

Man beachte dass der head vom program rest in der nächsten Konfiguration der successor vom program rest der aktuellen Konfiguration ist, ausser bei der Kompilation der return Anweisung. Das macht die folgenden Fälle des heads vom program rest der aktuellen Konfiguration interessant:

1. Letzte Anweisung in then-part von if/then/else Anweisung
2. Letzte Anweisung in body von while Anweisung
3. Return Anweisung, und wir müssen den succ vom caller erreichen.

Also muss man Induktiv (entlang der Rekursion der succ Definition) Zeigen, dass wir nur noch eine endliche Anzahl von Schritten von jenem Konsistenzpunkt entfernt sind. Es gilt offensichtlich für die o.g. Fälle:

1. Nach dem then-part und einem Schritt erreichen wir das ende vom code vom Fa, und per Induktionshypothese erreichen wir nach endlich vielen Schritten und als nächsten Konsistenzpunkt den start vom code vom succ vom Fa, was genau der succ ist.
2. Nach dem body und einem Schritt erreichen wir den beginn vom code vom Fa, und das ist genau der succ.
3. Durch die return Anweisung landen wir im code vom caller, und da der code vom Caller stets eine Zuweisung ist, erreichen wir nach endlich vielen Schritten das ende des codes vom caller. Danach erreichen wir entweder einen Konsistenzpunkt, oder wir wenden Fälle 1 und 2 an.

Offensichtlich manipulieren die zusätzlichen Schritte in Fällen 1 und 2 nur den pc, und die restliche Konsistenz bleibt erhalten.