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

System Architecture - SS15

Exercise Sheet 6(due: June 1, 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 Vor- 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: _____

Die Aufgaben auf diesem Blatt sind zum Großteil Bonuspunkte, damit Sie eine Möglichkeit haben vor der Vorklausur am 13.6. Punkte zu sammeln.

Aufgabe 1: (2)

Zeigen Sie das Dekompositionslemma der Addition.

(a) (2 points) Zeigen Sie: Für alle $a, b, s \in \mathbb{B}^n$ und $c_0, c_m, c_n \in \mathbb{B}$ sodass

$$\langle a[m-1:0]\rangle + \langle b[m-1:0]\rangle + c_0 = \langle c_m s[m-1:0]\rangle$$
$$\langle a[n-1:m]\rangle + \langle b[n-1:m]\rangle + c_m = \langle c_n s[n-1:m]\rangle$$

gilt auch

$$\langle a\rangle + \langle b\rangle + c_0 = \langle c_n s\rangle.$$

(b) (1 point (bonus)) Wo wurde das Lemma benutzt? (In welchem Beweis, und für welchen Schritt.)

Solution: Beweis durch mehrfaches anwenden des Dekompositionslemmas.

$$\begin{aligned}
&\langle a\rangle + \langle b\rangle + c_0 = \langle c_n s\rangle \\
\iff &(\langle a[n-1:m]\rangle + \langle b[n-1:m]\rangle) \cdot 2^m + (\langle a[m-1:0]\rangle + \langle b[m-1:0]\rangle + c_0) \\
&= \langle c_n s[n-1:m]\rangle \cdot 2^m + \langle s[m-1:0]\rangle \\
\iff &(\langle a[n-1:m]\rangle + \langle b[n-1:m]\rangle + c_m) \cdot 2^m + (\langle a[m-1:0]\rangle + \langle b[m-1:0]\rangle + c_0) \\
&= \langle c_n s[n-1:m]\rangle \cdot 2^m + \langle c_m s[m-1:0]\rangle \\
\iff &(\langle a[n-1:m]\rangle + \langle b[n-1:m]\rangle + c_m) \\
&= \langle c_n s[n-1:m]\rangle \qquad \wedge(\langle a[m-1:0]\rangle
\end{aligned}$$

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

System Architecture - SS15

Exercise Sheet 6(due: June 1, 2015)

Bonus Aufgabe 2:

Konstruieren Sie in Gatter-Schemata (i.e., Bildchen) einen Instruction-Decoder Schaltkreis mit input $I \in \mathbb{B}^{32}$ und outputs

- $af \in \mathbb{B}^4$
- $bf \in \mathbb{B}^4$
- $cad \in \mathbb{B}^5$
- $rd, rs, rt \in \mathbb{B}^5$
- $sa \in \mathbb{B}^5$
- $rtype, itype, jtype \in \mathbb{B}$
- $lw, sw, b, jump \in \mathbb{B}$
- $xtimm \in \mathbb{B}^{32}$
- $iindex \in \mathbb{B}^{28}$

Solution: N/A

Bonus Aufgabe 3:

Schreiben Sie eine MIPS-Routine, welche die Quadratwurzel einer Zahl berechnet.

(a) (4 points (bonus)) Ihr Programm soll eine kommentierte Folge von MIPS-Mnemonics sein. Verwenden Sie ausschliesslich Labels um die Ziele von direkten Sprüngen (j, jal) und Branches anzugeben. Sie Dürfen ein Macro $SQUARE(j)$ der länge s (also mit s Instruktionen) benutzen, das Register j mit sich selbst multipliziert (Modulo 2^{32}). Wenn Sie das Macro in Konfiguration c ausführen erreichen Sie eine Konfiguration c' mit

$$c'.gpr(j) = (\langle c.gpr(j)\rangle^2 \bmod 2^{32})_{32},\ c'.pc = c.pc +_{32} (4 \cdot s)_{32},$$

und alle anderen Komponenten haben sich nicht verändert.

Sie können annehmen, dass Ihre Routine durch $jalSQRT$ aufgerufen wird und sich die Zahl im General Purpose Register 1 befindet.

Sie können davon ausgehen, dass es eine Konstante X gibt sodass in Konfiguration c vor dem Aufruf Ihrer Routine der Speicher an Adressen $\{c.gpr(29), \ldots, X\}$ null ist:

$$c.m(c.gpr(29)) = 0^8, \ldots, c.m(X) = 0^8,$$

und ausserdem $X - c.gpr(29) > 70$.

(b) (2 points (bonus)) Verwenden Sie nicht das Macro $SQUARE(j)$.

(c) (2 points (bonus)) Beweisen Sie die Korrektheit Ihrer Routine, i.e., zeigen Sie dass wenn ihre Routine in Konfiguration c mit einem jal aufgerufen wurde, dass dann in der Konfikuration c' nach der Rückkehr aus ihrer Routine gilt:

$$c'.pc = c.pc +_{32} 4_{32},\ c'.gpr(1) = \lfloor \sqrt{\langle c.gpr(1)\rangle} \rfloor_{32},$$

und alle anderen Komponenten sind unverändert.

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

System Architecture - SS15

Exercise Sheet 6(due: June 1, 2015)

(d) (2 points (bonus)) Läuft Ihr Programm auf dem in der Vorlesung gezeigten Prozessor möglicherweise trotz des Korrektheitsbeweises Fehlerhaft? Warum/Warum nicht?

Solution: Wir definieren ein Makro $MOV(i,j)$, das den Wert des Registers i in Register j kopiert:

$$MOV(i,j) = or\ j\ 0\ i.$$

Wir definieren auch ein Makro $SHL(j,x)$, das j um x nach links shiftet:

$$SHL(j,x+1) = SHL(j,x);\ add\ j\ j\ j,\ SHL(j,0) = \epsilon.$$

Wir definieren ein Makro zur Berechnung des i-ten bits des Ergebnis, unter der Annahme dass bits $[32 : i+1]$ bereits berechnet sind. Wir nehmen an, dass diese bits in R1 gespeichert sind, während in R2 die Differenz zwischen dem ursprünglichen $x = c.gpr(1)$ und $R1^2$ ist. Wir setzen dabei das i-te bit Testweise auf 1, und überprüfen ob wir dadurch grösser werden als die Wurzel von x.

$BIT(i) =$	$MOV(1,3);$	$R3 = c_{i+1}$
	$SHL(3,i+1);$	$R3 = 2 \cdot c_{i+1} \cdot 2^i$
	$\begin{cases} lui\ 4\ +(2^{2\cdot(i-8)});\ add\ 3\ 3\ 4; & i > 7 \\ addiu\ 3\ 3\ +(2^{2\cdot i}); & \text{o.w.} \end{cases}$	$R3 = 2 \cdot c_{i+1} \cdot 2^i + 2^{2\cdot i}$
	$sub\ 3\ 2\ 3;$	$R3 = x^2 - (c_{i+1} + 2^i)^2$
	$bltz\ 3\ DONE_i;$	$if\ x^2 >= (c_{i+1} + 2^i)^2$
	$xori\ 1\ 1\ +(2^i);$	$c_i = c_{i+1} + 2^i$
	$MOV(3,2);$	$R2 = x^2 - c_i^2$
$DONE_i :$		

Jetzt können wir den Programmtext angeben:

$SQRT :addi\ 29\ 29\ +12;\ sw\ 2\ 29\ -12;\ sw\ 3\ 29\ -8;\ sw\ 4\ 29\ -4;$	store registers
$MOV(1,2);\ MOV(0,1);$	$c_{16} = 0, x^2 - c_{16} = x^2$
$BIT(15);\ \ldots;\ BIT(0);$	compute bits
$lw\ 2\ 29\ -12;\ sw\ 0\ 29\ -12;$	restore registers and memory
$lw\ 3\ 29\ -8;\ sw\ 0\ 29\ -8;$	
$lw\ 4\ 29\ -4;\ sw\ 0\ 29\ -4;$	
$addi\ 29\ 29\ -12;$	
$jr\ 31$	return from routine

Für die Korrektheit Zeigen wir zuerst, dass

$$\langle c_i \rangle^2 \leq x^2 \wedge (\langle c_i \rangle + 2^i)^2 > x^2$$

Wir zeigen dies per Induktion abwärts.

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

System Architecture - SS15

Exercise Sheet 6(due: June 1, 2015)

- $i = 16$: $c_{16}^2 = 0^2 \leq x^2$, und

$$(c_{16} + 2^{16})^2 \geq 2^{32} > 2^{32} - 1 \geq x^2$$

.

- $S(i) \to i$:

$$c_i = \begin{cases} c_{i+1} + 2^i & (c_{i+1} + 2^i)^2 \leq x^2 \\ c_{i+1} & \text{o.w.} \end{cases}$$

  - Im ersten Fall gilt $c_i^2 \leq x^2$ per Definition, und

$$(c_i + 2^i)^2 = (c_{i+1} + 2^i + 2^i)^2 = (c_{i+1} + 2^{i+1})^2 \overset{IH}{>} x^2.$$

  - Im zweiten Fall gilt $(c_i + 2^i)^2 > x^2$ per Definition, und

$$c_i^2 = c_{i+1}^2 \overset{IH}{\leq} x^2.$$

Daraus folgt direkt dass

$$c_0^2 \leq x^2 \wedge (c_0 + 1)^2 > x^2$$

und c_0 ist die gesuchte Zahl.

Weiterhin springt der jr befehl zu $c'.gpr(31) = c.pc +_{32} 4_{32}$ (spezifikation jal, jr).

Es bleibt noch zu Zeigen, dass der Speicher und die anderen Register unverändert bleiben, das ist jedoch nur dann möglich wenn R29 aligned ist und nicht in der ROM-Portion liegt (das wurde bei den Annahmen jedoch nicht vermerkt). Dann ist es allerdings offensichtlich.