Universität des Saarlandes
FR 6.2 - Informatik
Prof. Dr. W.J. Paul
Dipl.-Ing. Dominik Rester

4. Übungsblatt Informatik II

(Abgabe: 18.05.2005)

1. **Aufgabe: (Instruktionsdekodierer)** **(10 Punkte)**
   Konstruiere einen Schaltkreis, der aus I(c)[31:0] die folgenden Prädikate berechnet:

   - RD(c)
   - sxtimm(c)
   - jump(c)
   - branch(c)
   - lw(c)
   - sw(c)
   - comp(c)
   - compimm(c)
   - jal(c)
   - jalr(c)
   - RS1(c)
   - RS2(c)
   - aluf(c)

   Gib das Delay deiner Konstruktion an.

2. **Aufgabe: (Laden von 32bit Konstanten)** **(5 Punkte)**
   Der Instruktionssatz aus der Vorlesung bietet keine Möglichkeit eine 32-bit Konstante mit einer Instruktion zu laden. Wie kann man eine beliebige 32-bit Konstante mit zwei Instruktionen laden?
   Hinweis: Dies ist normalerweise Aufgabe des Compilers. Dieser kennt die Konstante, wenn er die DLX Assembler Befehle erzeugt. Daher können die erzeugten Instruktionen von der Konstante abhängen.

Universität des Saarlandes
FR 6.2 - Informatik
Prof. Dr. W.J. Paul
Dipl.-Ing. Dominik Rester

4. Übungsblatt Informatik II

(Abgabe: 18.05.2005)

3. **Aufgabe: (DLX Assembler)** **(3 + 3 + 5 Punkte)**
Je nach Instruktionstyp (I-, R-, J-type) sieht die Syntax einer Instruktion folgendermaßen aus:

I-type	R-type	J-type
mnemonic(RD, RS1, imm);	mnemonic(RD, RS1, RS2);	mnemonic(imm);

Ausnahmen sind die I-type instruktionen: *beqz* und *bnez*, die jeweils kein RD Argument nehmen, also mnemonic(RS1, imm), sowie *jr* und *jalr*, die jeweils nur das RS1 Argument nehmen. Es stehen alle Instruktionen aus dem Instruktionssatz zur Verfügung. Für 'mnemonic' ist ein Eintrag aus der Spalte 'mnemonic' zu wählen. Die Semantik der Instruktion wird (in abgekürzter Schreibweise) durch die Spalte 'Effekt' definiert. Pro Zeile darf nur eine Instruktion stehen. Nach dem ';' ist Platz für Kommentare. Die immediate Konstante wird (der Einfachheit halber) als Dezimalzahl angegeben. RD, RS1 und RS2 sind als Ri anzugeben, wobei i $\in \{0, \cdots 31\}$. Zu Begin jeder Zeile steht die Adresse der Instruktion (= PC) gefolgt von ':'. Unsere Programme beginnen stets an Adresse 0. Beachte, daß Adressen immer durch 4 teilbar sein müssen.

Beispiele:
0: addi(R1, R0, 20); R1:=20 (lädt 20 in R1, da R0=0)
4: sw(R0, R0, 400); m[403:400]:=0 (speichert 0_{32} in Adresse 400)
8: subi(R17, R4, 9); R17:=R4-9
12: beqz(R0, -8); PC:=PC-8 = 4

Programme ohne ausführliche Kommentare erhalten keine Punkte!

(a) Schreibe ein DLX Assembler Program, das zwei Zahlen a1 und a2 addiert und das Ergebnis in Speicherzelle 1024 ablegt. Die beiden Zahlen stehen zu Begin an Addresse 100 und 104.

(b) Nun stehen die beiden Zahlen an Adresse 20000 und 20004. Was ändert sich an dem Program aus Aufgabe (a)?

(c) Schreibe ein DLX Assembler Program, welches die Zahlen von 0 bis n addiert und das Ergebnis in Adresse 1024 ablegt. Zu Begin ist n in Adresse 200 gespeichert.

4. **Aufgabe: (Misalignment)** **(5 Punkte)**
Konstruiere einen Schaltkreis, der erkennt ob ein Speicherzugriff misaligned ist. Misalignment liegt vor, wenn die Adresse eines Speicherzugriffes nicht durch 4 teilbar ist.

5. **Aufgabe: (Illegal Instruction)** **(5 Punkte)**
Konstruiere einen Schaltkreis, der erkennt ob eine gegebene Instruktion eine gültige gemäß dem Instruktionssatz ist.
Ill: $\{0,1\}^{32} \rightarrow \{0,1\}$ mit

$$Illi(I(c)) = \begin{cases} 0 & : \text{I(c) ist gültige Instruktion} \\ 1 & : \text{sonst} \end{cases}$$