University of the Saarland
Department 6.2 - Informatik
Prof. Dr. W.J. Paul

Computer Architecture I - WS 02/03

(due: 02.12.2002)

Aufgabe 1: (32 Bit Konstanten) **(5 Punkte)**

Um eine 32 Bit Konstante in ein Register der DLX zu laden braucht man man mehr als eine Instruktion. Da man aber den Wert der Konstanten kennt, wenn man ein Programm schreibt, kann man 32 Bit Konstanten mit nur zwei Instruktionen erzeugen. Geben Sie eine Regel an, mit der diese Befehle generiert werden können. Dabei muss eine Fallunterscheidung vorgenommen werden.

Aufgabe 2: (Kosten und Delay von Booth Multiplizieren) **(4 + 6 Punkte)**

Wie bei der Delay- und Kostenabschätzung für den Addiererbaum in Standardmultiplizierern kann man die Anzahl von Excess-Volladdierern für den Addierbaum eines Booth Multiplizierers berechnen. Die Standardbreite der partiellen Produkte S' ist nun $n' = n + 5$. Längere Operanden benötigen Excess-Volladdierer.

1. Übernehmen Sie die Definitionen des Gewichts von Knoten W und der Anzahl der Eingabebusse des linken Teilbaums h wie in der Vorlesung. Bei der Kostenabschätzung bei Standardmultiplizierer sahen wir, dass die Anzahl der Excess-Volladdierer für einen bestimmten Knoten $e = 2 * h$ ist.

   Zeigen Sie, dass bei Booth Multiplizierern die Anzahl der Excess-Volladdierer für einen Knoten $e' = 4 * h$ beträgt.

2. Sei $m' = \lceil (m+1)/2 \rceil$, $M' = 2^{\lceil \log m' \rceil}$, $\mu' = \log(M/4)$ und $m \in \{27, 58\}$. Daraus ergibt sich: $3/4 * M' \leq m' < M'$.

   Zeigen Sie[1], dass die Anzahl der Excess-Volladdierer im kompletten Addiererbaum eine obere Grenze von $E' \leq 2 * (\mu' * m')$ hat.

Aufgabe 3: (Instruktionskodierung) **(2 + 1 Punkte)**

1. Was tut folgende DLX-Instruktion?[2] Schreiben Sie den Befehl in DLX Assembler (siehe Aufgabe 4) und beschreiben Sie seine Funktion in Worten.
   01101100101001000000000001100100

2. Kodieren Sie die folgende Assembler-Instruktion als Bit-String:
   add R1 R2 R3

Aufgabe 4: (DLX Assemblerprogrammierung) **(6 Punkte)**

Sei n eine natürliche Zahl. Schreiben Sie ein DLX Assemblerprogramm, das die Summe der Zahlen $1, \ldots, n$ berechnet. Zu Beginn stehe n im Register $GPR[1]$. Ihr Programm soll das Ergebnis in Register $GPR[2]$ speichern.

Kommentieren Sie Ihr Programm so, dass es gut zu verstehen ist. Nicht kommentierte Programme ergeben 0 Punkte!!!

[1] ähnlich wie beim Beweis in der Vorlesung
[2] Das höchstwertigste Bit befindet sich am linken Ende.

Die Assemblerbefehle sehen folgendermaßen aus:

- r-type: Mnemonic / Zielregister / Quellregister 1 / Quellregister 2

  Beispiel: add R1 R1 R2 (addiert die Inhalte der Register R1 und R2 und speichert das Ergebnis im Register R1)

- i-type: Mnemonic / Zielregister / Quellregister / 16 Bit Konstante

  Beispiel: addi R1 R2 523

- j-type: Mnemonic / Konstante

  Beispiel: j 500

Aufgabe 5: (DLX Assemblerprogrammierung) **(6 Punkte)**

Seien $n, a_1, \ldots, a_n$ natürliche Zahlen. Schreiben Sie ein Programm in DLX Assembler, das das Maximum der Zahlen $a_1, \ldots, a_n$ berechnet. Zu Beginn steht der Wert von n in Register $GPR[1]$ und die a_i in den Speicherstellen $M(i-1)$ $(i \in \{1, \ldots, n\})$. Ihr Programm soll das Ergebnis in Speicherstelle $M(n)$ speichern.