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

Computer Architecture I - WS 02/03

(due: 02.12.2002)

Excercise 1: (32 Bit Constants) **(5 points)**

One needs more than one instruction to load a 32 bit constant into a register of the DLX. Because one knows the value of the constant when one writes a program one can build 32 bit constants with two instructions. Give a rule how to construct these instructions. In these rules you have to consider more than one case!

Excercise 2: (Cost and Delay of Booth Multipliers) **(4 + 6 points)**

Like in the delay and cost estimation for the adder tree in standard multipliers one can compute the number of excess full adders for the adder tree of a booth multiplier. The standard width of the partial sums S' is now $n' = n + 5$. Longer operands require excess full adders.

1. Take the definitions of the weight W of some nodes and of the number of input busses h of the left sub tree as in the lecture. In the cost estimation for the standard multiplier we saw that the number of excess full adders for a given node is $e = 2 * h$.

   Show that for booth multipliers the number of excess full adders for a given node is $e' = 4 * h$.

2. Let $m' = \lceil (m+1)/2 \rceil$, $M' = 2^{\lceil \log m' \rceil}$, $\mu' = \log(M/4)$ and $m \in \{27, 58\}$. Therefore it holds that $3/4 * M' \leq m' < M'$.

   Show[1] that the number of excess full adders for the whole addition tree has an upper bound of $E' \leq 2 * (\mu' * m')$.

Excercise 3: (instruction set architecture) **(2 + 1 points)**

1. What does the following DLX instruction do?[2]. Write it in DLX assembler (see Exercise 4) and describe it in words.
   01101100101001000000000001100100

2. Code the following assembler instruction as a bit string:
   add R1 R2 R3

Excercise 4: (DLX assembler programming) **(6 points)**

Let n be a natural number. Write a DLX assembler program which computes the sum of the numbers $1, \ldots, n$. At the start of the program n is stored in the register $GPR[1]$. Your program should write the result in $GPR[2]$.

Comment your program in a way that everyone can understand what it should do. Programs without a enough comments will give 0 points!!!

[1] similarly to the proof in the lecture
[2] higher most bit is on the left side

Write the assembler instructions as:

- r-type: Mnemonic / destination register / source register 1 / source register 2

  Example: add R1 R1 R2 (adds the value of register R1 and R2 and stores the result in register R1)

- i-type: Mnemonic / destination register / source register / 16 bit constant

  Example: addi R1 R2 523

- j-type: Mnemonic / constant

  Example: j 500

Excercise 5: (DLX assembler programming) **(6 points)**

Let $n, a_1, \ldots, a_n$ be natural numbers. Write a DLX assembler program which computes the maximum of the numbers $a_1, \ldots, a_n$. At the start of your program n will be stored in register $GPR[1]$ and the a_i in memory cell $M(i-1)$ ($i \in \{1, \ldots, n\}$). Your program should store its result in memory cell $M(n)$.