georg.von-der-brueggen [☺] tu-dortmund.de
ulrich.gabor [☺] tu-dortmund.de
pascal.libuschewski [☺] tu-dortmund.de
olaf.neugebauer [☺] tu-dortmund.de

Exercises for
Rechnerstrukturen
Winter semester 2015

Exercise Sheet 8

(16 points)

Submission until Wednesday, 4th November 2015, 16:00 pm
Discussion begins on Tuesday, 10th November 2015

To solve the exercises use the simulator presented in the lecture. You can download the simulator at http://courses.missouristate.edu/KenVollmar/MARS/index.htm

8.1 Simulator familiarization (4 points)

Select 'File' 'New' and enter the following assembler program into the editor. Save the source code into a file. (e.g. ExerciseSheet8A1.asm)

```
.data                # 01 Line numbers used
x: .word 4711        # 02 for discussion
y: .word 10          # 03
z: .word 0x0a92      # 04
e: .word 0           # 05 result variable
                     # 06
.text                # 07
.globl main          # 08
main:                # 09
      lw  $2,x       # 10
      lw  $3,y       # 11
      lw  $4,z       # 12
      add $2,$2,$3   # 13
      sub $3,$2,$4   # 14
      sw  $3,e       # 15
      li  $2,10      # 16
      syscall        # 17 exit program
```

Assemble the program(F3). The program is now listed under 'Execute' in the text segment. It start with the line: **lui $1,**0x1001 (column 'Basic'). The source code is listed under 'Source'. Execute the program in single step mode (F7). Observe the changes in register 2 and 4 after each step (Registers and values are listed on the right side in the IDE).

a. After termination of the program, the tab 'Run I/O' displays – program is finished running –. Which value is stored in register **$3**? Provide also a decimal representation of the value.

b. What calculates the program in register Reg[3] (with respect to variables x,y,z)? Provide your answer in Register-Transfer-Notation and the calculated results. (in Register-Transfer-Notation: x,y,z and the result register must be used)

c. The first program line in the 'Execute' tab of the simulator is: **lui $1,**0x1001. What is the reason for 0x1001?

d. Where (part of the IDE) is the result stored in the memory after executing line 15? Provide the name of area of the IDE, the exact memory address and the content stored at this address in the form 0x..... . What does the prefix 0x mean?

8.2 Loading of constants (4 points)

With the MIPS-pseudo command '**li** r,k', one can load a 32-bit constant k into register r. Depending on the value of k, different *real* instructions are generated for the command '**li** r,k'. Now, we would like to examine these different instructions. Please provide an example (with an appropriate k) for each instructions generated. Use **$3** for r and **$1** to store temporary results.

a. $-2^{15} \leq k \leq 2^{15}$: 16 bit two's complement, 0x?000????

b. $0 \leq k \leq 2^{16}$: 16 bit absolut value, 0x0000????

c. $k = k1 * 2^{16}, 2^{16} \leq k1 \leq 2^{32}$: 32 bit absolut value, 0x????0000

d. $0 \leq k \leq 2^{32}$: 32 bit absolut value (Tip: $k = k1 * 2^{16} + k2, 0 \leq k1 \leq 2^{16} und 0 \leq k2 \leq 2^{16}$), 0x????????

? represents a number from {0 ... F}

8.3 Multiplication (4 points)

Write an assembler program to analyse the different multiplication variants provided by the instructions **mul**, **mult**, **multu** and **mulo** as follows: Save the two constants +1 and -1 into to different memory cells (value1 and value2 in .data segment) and multiply them with **mult** and **multu**. Then, multiply 1030 and 4721471 with the instructions **mul** und **mulo**.

a. Fill out the table with the results of the different multiplications (register Hi and low and Reg[4] if necessary). Reg[4] should be used as a target register if necessary.

Instruction	Hi	Lo	$4
mult			
multu			
mul			
mulo			

b. What are the results of **mult** and **multu** in decimal. Why does **mulo** have no result?

8.4 Instruction sequence (4 points)

Consider the following sequence of MIPS-instructions: (Provide only changes after the second row)

	Register content after execution of an instruction			
	$2	$3	$4	$lo
at program start	?	?	?	?
li $2,0x05				
ori $4,$0,48				
mul $3,$4,$2				
mfhi $3				
add $2,$2,$4				
addi $4,$2,0xAB39				
and $3,$4,0x7F56				
ori $3,$3,0x87BA				

Advide:
Use the hexadecimal system for calculation. Logical immediate-instructions uses the zero-extend and not the sign-extend function. You should be able to solve this task without using the simulator.

Notes:
Submission until Wednesday, 4th November 2015, 16:00 pm in the mailbox number 46 at Otto-Hahn-Straße 12.

You can find the mailboxes in the first floor of the Otto-Hahn-Straße 12 near the transition to the ground floor of the Otto-Hahn-Straße 14. The mailboxes are labeled with “Rechnerstrukturen”, the exercise group number and time/place of the exercise. The English exercise group is number 30 and the mailbox is number 46.

Please write your **name**, your **student registration number** and your **exercise group number** at the top right corner of your submission. You can make submissions in teams with up to two more students. To make a team submission put names, student registrations numbers and group numbers of all members of the team on the submission. Only one submission per team has to be made.

Tack you submission. Please do not fold your submission and do not put it into an envelope. Use the correct mailbox, you will need your exercise group number for that.

In total there are 12 exercises in 3 blocks (A, B, C). In each block you have to achieve at least 30 points of 64 possible ones to get access to the exam.