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

Exercises for
Rechnerstrukturen
Winter Semester 2016

Exercise Sheet 9 (Block C - 1)

(16 points)

Submission until Wednesday, 11th January 2017, 16:00 o'clock
Discussion begins on Monday, 16th January 2017

To solve the exercises, use the simulator presented in the lecture (MARS simulator). You can download the simulator from the following link:
http://courses.missouristate.edu/KenVollmar/MARS/index.htm

9.1 Assembly programming (4 points)

Consider the register transfer operation below:

```
Memory[0x10010000] := (Memory[0x10010004] - Memory[0x10010008] + 8)
                      * Memory[0x1001000C] + Memory[0x10010010]
```

Complete the following assembly code to perform the register transfer above. The use of lw- and sw- commands with offset values (which cannot be represented using 16 bits) is not allowed. Use as few registers and commands as possible.

```
li   $2,0x10010000 # Given: Reg[2] := 0x10010000
lw   $3,0x4($2)    # Given: Reg[3] := Memory[0x10010004]
lw   $4,0x8($2)    # Given: Reg[4] := Memory[0x10010008]
```

9.2 Simple Case Distinction (4 points)

Write a MIPS-Program to assist in the evaluation of exams' correction. The program should determine if the student passes (Points ≥ 40) or fails (Points < 40). You can use the code below to start your program.
First, the variable "Passed" is initialized to 99, which means that the grade is not assigned yet. Then, it is assigned to 1 in the case of "passed" and 0 in the case of "failed".

```
  .data
  Points: .word 42            # Exam points
  Passed: .word 99            # 0: No, 1: Yes, 99: Unknown
  Threashould: .word 40       # Passing threashould: 40 points

  .text
  .globl main
  .
  .
  .
```

9.3 Debugging (4 points)

Given a program to calculate the factorial iteratively. Unfortunately, it has 6 errors/bugs (Syntactic and semantic).
Hint: Definition of the factorial: $n! = n \cdot (n-1) \cdot (n-2) \cdots 1$ a $0! = 1$

a. Find and correct the errors/bugs. The number of program's lines should stay the same.

b. The result in register 3 should be interpreted as a two's complement number in another program part. How big (decimal) can the input "inp" size be, so that register 3 can be used as a correct result?

```
.data
  inp: .word 5          # input given by the user (e.g. 5! = 120 = 0x78)
  outp: .word 1         # output result

.text
.global main
main:
  lw $2,inp             # fetch the input
  li $3,0               # preallocate, in $3 could be something else
  bneq $2,$0,finished   # 0! there is no loop
jump:
  mul $3,$3,$2          # mul outp with a counter
  subi $2,1             # counting
  bgt $2,1, jump        # end of the loop, mul with 1 should not be performed
finished:
  sw outp, $3           # store the factorial result in outp
  li $2,$10             # Programmende
  syscall
```

9.4 Assembly programming (4 points)

Implement a program to calculate the parity bit of a variable called “value” (Typ .word). The parity value should be stored in a variable called “pari” (Typ .word).
If the binary representation of a number contains an even number of 1 bits, the value 0 should be returned, otherwise (i.e. odd) the value 1. For example, the program returns 0 as parity bit for the number $(23)_{10} = (0010111)_2$, and returns 1 for $(38)_{10} = (0100110)_2$.
Hint: You don’t need to use a multiplication command. The command “srl” could be used (see lecture notes [1] S. 13).

```
.data
value: .word 38          # An example of an input value (result: odd parity 1)
pari: .word 0            # parity result

.text
.globl main
main: lw   $2,value      # load the input example in R2

.
.
.
```

[1] http://ls12-www.cs.tu-dortmund.de/daes/media/documents/teaching/courses/ws1314/rs/script/rs2.pdf

Notes:
Submission until Wednesday, 11th January 2017, 16:00 pm in the mailbox number 40 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 31 and the mailbox is number 40.

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.