These problems are done in advance and discussed in practice session on the 2. course week 7-11.11.2016. Exercises are based on lectures 1 and 5 (web lectures 1 and 5).

1. Basics
   a. What are the four basic tasks for a computer?
   b. Which components in computer system take care of those four tasks?
   c. How are those four basic tasks implemented in the instruction fetch-execute cycle (Fig 3.12 [Sta13] [Sta10])?

2. Program location in memory. Assume that ttk-91 program P is stored in main memory area 50200-50711.
   a. What are the values for base and limit registers?
   b. Assume that you are executing machine instruction "JUMP 200" (jump to location 200) in memory location 156 (programs address space). What happens in the system hardware (in the instruction fetch-execute cycle)?
   c. Assume that you are executing machine instruction "LOAD R1, 200" (load contents of memory location 200 to register R1) in memory location 176. What happens in the system?
   d. Assume that you are executing machine instruction "JUMP 20000" in memory location 186. What happens in the system?
   e. Assume now that the operating system decides to give that memory area to some other use, and relocates program P to memory area 134000-134511. What would the base and limit register values be now? What should be changed in program P (code) so that it would still work properly (i.e., exactly the same as in the original location)?

3. Interrupts
   a. How does an interrupt differ from a subroutine (function, method)?
   b. How are the interrupts implemented at hardware level (in fetch-execute cycle)?
   c. How are the interrupts implemented at software level (in the operating system)?
   d. Why would you want to access all of the memory and execute in privileged mode during interrupt handling?
   e. How else (in addition interrupt) could the operating system get to execute?
   f. What does the operating system do while the processor is executing user program?

4. [2 htp] Interrupt handler. Assume that the interrupt handler can access all of the memory and that it executes in privileged mode. Assume also, that at the end of fetch-execute cycle one has observed
   i. integer overflow interrupt (interrupt nr 28, state register bit O in ttk-91), or
   ii. I/O interrupt (interrupt nr 24, state register bit I)
We want to implement an interrupt handler (e.g., to ttk-91 architecture).
   a. How do you implement the transfer of control to the interrupt handler?
      What do you need to save before transferring control to the interrupt handler?
   b. What does the IRET-instruction (interrupt return) actually do?
   c. Describe the general structure of the interrupt handler in cases i and ii. What does it do?
   d. What happens if during the execution of interrupt handler nr 24 an integer overflow interrupt occurs?

5. Assume that you are coding the ttk-91 simulator with some high level programming language, like Javalla or C. Simulator reads machine language ttk-91 code (stored in simulated ttk-91 memory) and emulates instruction execution with simulated registers one instruction at a time.
   a. How would you represent the simulated structures (ttk-91 registers, memory) in your program?
   b. How would you code the instruction fetch phase?
   c. How you code disassembling instruction parts (opcode, Rj, M, Ri, ADDR) from the instruction?
   d. How would you code fetching and operand to temporary register TR?
   e. How would you code the execution phase of instruction "add r2, r3"?
      You may assume that the value of the latter operand (r3) is already in simulated register TR.
f. How would you program the execution phase of instruction "jump loop"?
   You may assume that the value of the latter operand (memory address "loop") is already in
   simulated register TR.