581305-6 Computer Organization I, miniexam 1, 17.11.2016 (10p)
Write your answer on this exam paper in the space given. Please notice, that the exam paper is 2-sided.

a) [3 p] Hardware level instruction fetch/execute cycle (instruction cycle)
   i. What are the stages in instruction cycle, and what happens in them?
      START, Fetch next instruction, Execute instruction, HALT
      START – start running the processor, Fetch – fetch next instruction to IR from memory location pointed by PC and increment PC, Execute – execute instruction in IR, possibly and possible modify PC, HALT – stop running the processor.
      In normal operation, Execute follows each Fetch, and Fetch follow each Execute stage.

   ii. Where in the processor and in the overall system can one machine instruction cause state changes? In which stages of the instruction cycle do those changes occur?
      User addressable register (in execute stage), PC (in fetch and execute stage), interrupt flags in SR (any time), comparison flags in SR (in execute stage), main memory (in execute stage)
      [memories of other I/O devices (in execute state) – not requested now]

b) [3 p] Processor execution modes: privileged (supervisor) and user mode.
   i. Why does the processor have two different execution modes? Why wouldn’t you just execute in one mode all the time?
      Two modes makes it feasible to build reliable (operating) systems. Normal, simpler mode prevents ordinary user program to do “dangerous” things or access data not belonging to it.

   ii. What all is allowed in privileged mode that is not allowed in user mode?
      You can access all of the memory, and not only your own memory areas.
      You can execute all machine instructions, including the one reserved for operating system work.

   iii. When and how does the execution mode change from user mode to privileged mode?
      When and how do you get back to user mode?
      All interrupt handlers (see question c) may execute in privileged mode, and the mode is changed at the same time as control is transferred to the interrupt handler. A program can also ask the mode to be changed by invoking (calling) proper supervisor call (SVC), which will first check whether the calling program is allowed to do that.
      To return to the earlier execution mode, you need to invoke some special (privileged) machine instruction (e.g., RTI, or ReTurn from Interrupt).

-- TURN --
c) [4 p] Interrupts

i. Define the concept “interrupt” (exception, fault).
   Predefined event that may occur pretty much at any time, and that requires special handling
   outside normal instruction cycle for currently running program. An interrupt can be caused by
   currently running machine instruction, or an external device like a timer or an I/O device.
   Interrupts are handled by the operating system.

ii. Describe three (3) different types of interrupts, and explain what might cause them.
   Integer overflow – add instruction result does not fit into register,
   Bad address – running program tried to access memory location out of bounds,
   I/O interrupt – some I/O device wants to signal that it had completed its task,
   Timer interrupt – predefined signal that some operating system utility needs to run now,
   Supervisor Call instruction – specific request to invoke some operating system service,
   etc.

iii. How does the hardware know than in interrupt has occurred, when does it check for interrupts,
     and what does it do if an interrupt had occurred? As an example, use one of interrupts you
     described in part (ii).
   State register (SR) has one bit for all possible interrupts, and these bits are checked in the
   instruction cycle after each Execute stage (unless interrupts are disabled, i.e., bit D in SR is set).
   For example, if the add-instruction result does not fit into result register, then the result is not
   stored but instead the corresponding interrupt bit (O) is set in SR. In the Check for Interrupts stage
   the bit O is found to be 1, current PC is saved somewhere, and PC will get the value for interrupt
   handler for integer overflow. For example, if bit O is the bit 28 in SR, then the interrupt handler
   address for it could be the (absolute) memory address 28, or it could be stored in (absolute)
   memory address 28.

iv. What is an interrupt handler belonging to the operating system? When and how is the interrupt
    handler executed? As an example, use one of interrupts you described in part (ii).
   Each interrupt type has its own interrupt handler, which is an operating system subroutine
   designed to handle that special case. Once an interrupt is found to have occurred, the normal
   instruction flow is interrupted, and the interrupt handler is executed (see part iii). After that the
   execution may return to interrupted program, or some other program. For example, the interrupt
   handler for integer overflow is executed immediately after the faulting machine instruction. It will
   advise the user of what happened, kill the offending program (you cannot continue after such
   fault), and ask the operating system to execute some other program.