Name	Signature	Student Id Nr	*Points*

581305-6 Computer Organization I, miniexam 1, 11.11.2019 (12 p)

Write your answer on this exam paper in the space given. Please notice, that the exam paper is 2-sided.

a) [3 p] How does the processor execute the machine instructions? Into which parts is that execution of instructions divided to, and what do those parts do?

Instruction fetch-execute cycle, containing the following phases:

i. Fetch instruction. The instruction from memory location given in register PC is loaded into register IR.
ii. Increment PC. This is the default location for next instruction.
iii. Execute the instruction in IR. The instruction is split into fields, some operands are possibly fetched from memory, the operation according to the opcode field is executed (possibly changing the value PC), the result is possible stored into memory, comparison results are stored in SR, possible interrupts caused by this instruction are marked in SR.
iv. Check for interrupts. If any interrupt has happened, old PC (and SR) are saved (into stack?), and control is given to the interrupt handler for that interrupt.
v. go back to phase i.

b) [3 p] Ordinary programs can only reference their own memory areas (code and data).
How is this implemented in the example computer (ttk-91) and how does the implementation effect the execution of instructions in the processor?

The memory area allowed to given program (P) is defined by registers BASE and LIMIT. Register BASE contains the starting memory address for P, and LIMIT the size of that memory area.
All addresses used in P are relative to BASE.
For every (code or data) memory reference (Ref), the hardware (in MMU) checks first that $0 \leq Ref < LIMIT$.
If there is no problem, the memory reference is made to physical memory address BASE+Ref.
If there is a problem, the instruction execution is terminated and appropriate interrupt (memory reference out of bounds) is raised. The execution the continues next in the interrupt handler for that interrupt, and program P execution is terminated.

TURN

c) [3 p] Parts of the operating system must be able to reference all data in memory, and perform tasks that are not allowed to ordinary programs.
How is this implemented in the example computer (ttk-91) and how does the implementation effect the execution of instructions in the processor?
There are two processor execution states, privileged and user. Normal programs execute in user-state and (certain) operating systems components (e.g., interrupts handlers) in privileged state.
In user state, the machine instructions can only access program's own memory area and use only "normal" instructions.
In privileged state the processor can use all machine instructions (e.g., read and write registers BASE and LIMIT, or empty the cache) and reference all memory (e.g., code and data areas of all programs in system, and memory areas reserved for operating system data).

d) [3 p] Parts of the operating system must be able to execute whenever needed.
How is this implemented in the example computer (ttk-91) and how does the implementation effect the execution of instructions in the processor?
There is a (small) finite set of predefined interrupts that are used as a gateway to operating system whenever need arises. For every interrupt, there is an interrupt handler, that is part of the operating system.
Whenever an interrupt occurs, it is marked into special register (SR in ttk-91).
Interrupts are checked at the end of each fetch-execute cycle, and if any has occurred, then the control is moved to the appropriate interrupt handler. At the same, the processor execution state becomes privileged. Once the interrupt handler (i.e., operating system) is done, control (and processor execution state) is moved back to the interrupted program, or some other program.

Mention three types, different from each other, of reasons to give operating system the turn to execute in the middle of executing some ordinary program. Give an example for each type.
Interrupts can be classified (e.g.) to

i. Errors in current instruction execution (e.g., arithmetic overflow, divide by zero, memory reference out of bounds).
ii. Signals from system (e.g., timer interrupt every 10 ms).
iii. Signals from outside the system (e.g., I/O interrupt from disk or network device controller).
iv. Program wants to use operating systems service (e.g., SVC instruction).
SVC instruction is often implemented as an interrupt of its own kind, because it is easy to implement that way.

TURN