Name	Signature	Student Id Nr	*Points*
Model solutions			12

581305-6 Computer Organization I, mini exam 4, 9.12.2019 (12 p)

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

a) [4 p] I/O implementation. The keyboard is implemented with interrupt-driven (indirect) I/O. User process (P) has asked(e.g., with service call *c=readchar(kbd)*) the keyboard device driver (DD) to read one key press. The device driver is implemented as a (privileged) subroutine. We assume for now, that the user has not yet pressed that key ('a') but will soon press it.

i. [1 p] How does the DD give the command to read next key press to the keyboard device controller process (DCP) running on the keyboard device controller (DC)?
If any (CPU) process management events happen at this time, please explain them.
DD writes the read command to DC's control register. (First it may have reset the status register in DC.)
DD (and P which called DD) suspends until it is activated again via I/O interrupt.

ii. [2 p] How will DD know, that some key ('a') is pressed? How will DCP tell DD about it?
How will DD know which key was pressed?
If any (CPU) process management events happen at this time, please explain them.
Once DCP notices that a key ('a') is pressed, it will store character code for 'a' in data register,
set "read OK" to status register, and finally cause an I/O-interrupt.
The I/O-interrupt handler moves DD to R-to-R queue.
Eventually DD (or which called DD) will run. It will first check that status register contains "read OK", and then reads the character 'a' from the data register.

iii. [1 p] When and how will P get the character code for the pressed key ('a') from DD?
If any (CPU) process management events happen at this time, please explain them.
DD returns read character 'a' to P, and P resumes execution with read character 'a'.

b) [4 p] Linking. Module N has (e.g.) variable X. Module M has calls to subroutine S in module N.
Modules M and N are linked so that M is first, and N second.

i. [1 p] How are the references to X inside module N modified during linking?
How do you know which addresses must be modified and onto what values?
The relocation table for N has a list of all locations where X is referenced, in addition to local address (within N) of X. All module addresses are relative to beginning of that module, which will be the relocation constant (RC) for that module. RC for M is zero and there is no need for changes. RC for N is the size of M, and this RC must be added to each occurrence of X in N.

-- TURN --

ii. [1 p] How are the calls in module M to subroutine S in module N modified during linking?
How do you know which addresses must be modified and onto what values?
The relocation table (import table in it) for M has a list of all locations where S in N is referenced to.
The relocation table (export table in it) for N has symbol S and its relative location within N.
The RC for N is added to symbol S value, and that address ($S+RC_N$) is used in M for all references to S.

iii. [1 p] When is static linking better than dynamic linking? Explain. (One example is enough)
The OS does not have dynamic linker, because of the OS being very simple.
Dynamic linking is not possible, because there is not mass storage or network connection.
Program is small, and there is no need for extra complexity.
…

iv. [1 p] When is dynamic linking better than static linking? Explain. (One example is enough)
You want to have an easy way for library updates.
You want small load module, and short process starting time.
You do not want to have rarely used modules present in load module.
Some library module is not even ready, when load module is linked together.
…

c) [4 p] Java-ohjelmien suoritus

i. [2 p] How is statement Y=X*X done in JVM byte code, when X and Y are local variables in method.
Where are X and Y located? How do you reference X and Y in (numeric) JVM byte code?
(It does not matter, if you do not remember exact byte code syntax. Use opcodes similar to ttk-91.)
X and Y are local variables in stack, and you refer to them using their relative locations in current frame, which is pointed by register LV. Assume now, that X is in LV+3 and Y is in LV+4.

```
push 3      ; push X in mem(LV+3) to top of stack   ("push   =3"  is ok here also)
push 3                  ("push  LV, =3"  is not ok, because references to LV are implicit)
mul         ; pop 2 operands from stack, multiply, push result to stack
pop 4       ; pop top of stack to Y in mem(LV+4)
```

ii. [2 p] What does statement "Java program P is executed with JIT-compilation" mean?
In what form is P at execution time, and how is that form achieved?
P has been already translated to Java byte code. When P is executed, each referenced module is compiled from byte code to native machine code on first (or second or third…) reference. Some modules may be interpreted, if they are used only once. Compiled modules must also be dynamically linked.

-- TURN --