Question 1

1 / 1 pts

Why does the Hack machine have separate A and C instructions?

- Because the A-register cannot be accessed by any other sort of instruction.

Correct!

- Because there are not enough bits in a C instruction to allow direct access to the addresses in memory.

- Because there are not enough possible C instructions.

- Because it saves confusion

Question 2

1 / 1 pts

Why is the following instruction - which is valid hack assembler - unlikely to be useful when run on the Hack machine?

```
D=M;JMP
```

- The a register gets used and reset by its use in accessing M leaving it set to zero for the jump which returns it to the start of the program which is very rarely what you want.

- Because it implicitly overwrites the A register and then jumps to that overwritten location.

Correct!

- Because it accesses the data memory at M using A and then also jumps to that same A address in ROM. It is very unlikely that this address is useful in both memories at the same time.

- Because we can't combine jumps with other instructions in C-instructions.

Question 3

1 / 1 pts

The Hack ALU has only 18 instructions listed in figure 2.6 of the textbook but there are $2^6 = 64$ possible input wire configurations. Why aren't all of these instructions available?

- To implement all 64 instructions would greatly increase the complexity of the processor and for no benefit since we can do almost everything we want with the 18 instructions we have.
- Because most of the other instructions would conflict with the 18 instructions that we have - making it so that they don't work properly.

Correct!

- Actually there are 64 instructions possible - in the machine code - but most are redundant so the assembler doesn't give the programmer a way to express these.
- The other instructions are mostly slower than the 18 very fast instructions. We don't implement these other instructions because we don't want to slow down the machine.

This is about understanding the relationship between the instruction set and the architecture.

Question 4

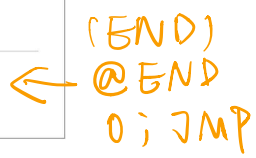
1 / 1 pts

How does a programmer finish a program in Hack Assembler?

- The programmer doesn't need to do anything to finish a program.
- By setting the address of the A register to zero.
- By including an instruction that is all zero bits at the end of the code and then jumping conditionally to that instruction.
- By using a halt instruction
- Using a no-op instruction.

Correct!

- By specifying a label followed by an instruction that unconditionally jumps to itself.



Question 5

1 / 1 pts

In the Hack machine there is an A register to hold the contents of the A-instruction but there is no equivalent register to hold the values of the C-instruction. Why not?

Correct!

- (◉) Because it is possible to wire up the hack CPU so that the wires holding the instruction can be routed directly to the chips controlling the CPU without the need for an intervening C-register.
- ( ) The D register in the hack machine serves this purpose.
- ( ) The fact that instruction memory is ROM in the hack machine design means that a C register is not necessary. If the instruction memory were RAM then there would be a need to store the instruction in a C-register.
- ( ) The C register is not needed because we make use of the A-register to hold the C-instruction while decoding takes place.