UNIVERSITY OF BRISTOL

Not a real Examination Period

School of Computer Science

Practice Examination for the Degrees of Master of Science in Computer Science (conversion)

COMSM1302
Overview of Computer Architecture (Practical component)

TIME ALLOWED:
2 Hours

This paper contains **four** questions.
All questions will be marked.
The maximum for this paper is **50 marks**.

Other Instructions

1. **You are only permitted to use the following software: Calculator, KCalc, Text editor, the Hack assembler, the Hack CPU emulator, and a web browser.**
2. **You are not permitted to visit any web sites other than those in the "In-class test 2" sidebar of the unit Blackboard page.**
3. **You are not permitted the use of physical calculators.**
4. **You are not permitted external reference materials.**

TURN OVER ONLY WHEN TOLD TO START WRITING

For each question, write Hack assembly within a .asm file. Each file should be named according to its question number, e.g. `Q4.asm`. You **are** permitted (and encouraged) to use the Hack assembler and Hack CPU emulator to test your code. You **are not** permitted to use the Hack VM emulator.

Full marks will be given to all programs that display the correct behaviour and obey any restrictions explicitly specified in the question. Partial marks will be available. We recommend that you include **brief** comments describing your code's intended behaviour to ensure that we understand your thought process correctly while marking; however, beyond this you will not be marked on e.g. code complexity or efficiency unless otherwise stated.

You should zip your completed .asm files and submit them to the "In-Class Test 2 (Practical component) submission point" on Blackboard. Multiple submissions are allowed, but only the last one will be marked. No submissions are allowed from outside the exam room under any circumstances.

Question 1 (10 marks)

Your code should read the values in RAM[0] and RAM[1], then behave as follows.

- Add RAM[0] to RAM[1], then store the result in RAM[2].
- If RAM[2] is positive, then multiply RAM[2] by -1.
- Take a bitwise OR of RAM[2] with RAM[1] and store the result in RAM[2].

You may assume that RAM[0] and RAM[1] are between -10000 and 10000 (so that no integer overflows can occur).

Question 2 (15 marks)

Your code should do nothing until the letters 'O' and 'M' are pressed on the keyboard in sequence ('O' first, then 'M') with no other keys being pressed in between. It should then colour every pixel on the screen black. For example, if the key sequence 'OHMOMAAAAHWAITNOIDIDNTMEANTODOTHATUNDOUNDOUNDO' is entered, then the screen should turn black on the second 'M' and then not do anything further.

Your code for Question 2 must be at most 200 lines long, or you will receive very limited credit.

Question 3 (15 marks)

Your code should be a direct translation of the Hack VM instruction `call Main.exam 7` into assembly. In other words, your code should assume that the current contents of RAM are a state of the Hack VM, and then change RAM (and the program counter) in exactly the same way that the `call Main.exam 7` statement would. You may assume the standard memory map from Hack VM to assembly (given in the reference materials), that the label `auto$52` is unused by any other piece of assembly code, and that the assembly code corresponding to the `Main.exam` function starts with the label `(call$Main.exam)`. You do not need to know the corresponding `function Main.exam *` command to answer this question.

Question 4 (10 marks)

In decimal, the *digital root* of a number x is defined as follows. Take the digits of x and add them to get a new number y. Then take the digits of y and add them to get a new number z. Then take the digits of z and add them, repeating the process until you end up with a single-digit number, which is the output. For example, to find the digital root of 491, add $4 + 9 + 1 = 14$, then add $1 + 4 = 5$, then return 5.

Binary digital roots work the same way, but in base 2. For example, to take the binary digital root of $103 = 1100111$, we add $1 + 1 + 0 + 0 + 1 + 1 + 1$ to get $5 = 101$, then add $1 + 0 + 1$ to get $2 = 10$, then add $1 + 0 = 1$ to get 1. In fact, it can be shown that every binary digital root will be 1, but different numbers take different paths to get there.

Your code should take the binary digital root of the value in RAM[0]. You should write the result of every digit sum on the "path" from RAM[0] to 1 into RAM, starting from RAM[1]. For example, if RAM[0] = 103 and all other memory is zero before running your code, then after your code has been run then we should have

$$\text{RAM}[0] = 103, \qquad \text{RAM}[1] = 5, \qquad \text{RAM}[2] = 2, \qquad \text{RAM}[3] = 1.$$

If instead RAM[0] = 255 before running your code, then after running your code we should have

$$\text{RAM}[0] = 255, \qquad \text{RAM}[1] = 8, \qquad \text{RAM}[2] = 1.$$

(You do not need to be able to access the morphogenetic field in order to solve this question!)