9.1P - Computer System

PART 1

The Hack computer's fundamental parts are as follows:

Memory chip:- Instructions and data are stored on a memory chip. It has a keyboard for input, a screen to display output, and RAM for temporary data and variables.
CPU chip:- The central processing unit, or CPU chip, manages the flow of instructions and carries out calculations. It has registers to momentarily store data or addresses during operations and an Arithmetic Logic Unit (ALU) to carry out math operations.
ROM chip:- A read-only memory chip, or ROM chip, is used to store permanent programme code, such as firmware or BIOS.

Hardware Description Language is used to merge these chips into a single master chip called Computer.hdl, which is then used to construct the computer.
RAM from earlier tasks is already present in the Memory chip. It doesn't need to be changed.
Programmes are stored in the ROM after being encoded in binary machine language, just like our tests.
The CPU reads the first instruction from ROM upon power-up and decodes/executes it. In order to temporarily store addresses and data during operations, registers are used. After finishing, it retrieves the following command, and so forth.

We utilize sample test programmes that are pre-encoded in the ROM to test the computer:
In RAM, an additional test adds numbers and saves the result.
The maximum test locates the higher of two RAM values and stores it in a different RAM address.
The Rectangle test creates a box on the display.

I ran my computer.hdl file and all my test scripts all worked fine. Below is my computer.hdl files.

https://drive.google.com/file/d/1jgu1SP1uIL7GnJrE7ecRvtwIcAlXOpW9/view?usp=sharing

---

PART 2

This section explains each line of the Add.hack code that adds to the variables in the ComputerAdd.tst script mentioned above. Six lines of code make up the Add.hack and provide the instructions to be followed. The six lines of instructions are listed below.

Instruction 1: 0000000000000010
This is an A instruction since the opcode, or first bit, is 0. @value is the symbolic syntax. Thus, the instruction provided can be represented as @2.The value of the A-register is 2. To accomplish this, set the A register's load bit, also known as the control bit, to 1. Two are sent to the address bus by the A-register. RAM can be accessed using this[2]. The data bus receives the value of RAM[2], which is then output as the M register (M input).

Instruction 2:1110110000010000
This instruction is in C since the opcode (first bit) is 1. The syntax that is symbolic is est=comp;jmp. It is possible to represent the given instruction as 111ac1c2c3c4c5c6d1d2d3j1j2j3.
Therefore, calculation instruction A corresponds to the computation bits 110000. It is evident from destination bits 010 that the value is kept in the D register. Furthermore, it is stated in the final three jump bits 000 that no jump condition is given.

Instruction 3:0000000000000011
This is an A instruction since the opcode (first bit) is 0. RAM's memory location is chosen in this way[3]. The value of the A register is 3. To accomplish this, set the A register's load bit to 1. Three is put into the address bus by the A-register. RAM can be accessed using this[3].The RAM[3] value is output as the M register after being placed onto the data bus.

Instruction 4:1110000010010000
This instruction is in C since the opcode (first bit) is 1. As previously stated, a C instruction has the syntax dest=comp;jmp. The computation bits 000010 in this instruction equate to the computation D+A. The value will be put in the D register since the destination bits are 010. The jump condition is not defined because the jump bits are 000.

Instruction 5:0000000000000000
This is an A instruction since the opcode (first bit) is 0. RAM[0] is the memory location that will be chosen. To achieve this, set the A-register's load bit to 1. The address bus is then filled with 0. RAM is accessed in this way[0]. The data bus receives the value of RAM[0], which is then sent as output to the M register.

Instruction 6:1110001100001000
This instruction is in C since the opcode (first bit) is 1. The computing instruction D is matched by the computation bits 001100. It will therefore deal with the D register. The value is going to be put in the M register, as indicated by the destination bits 001.

Jump bits have a value of 000. As a result, no jump condition is given.
In a nutshell, the instructions are written to store two constants in RAM: one in RAM[2] and one in RAM [3]. After that, sum the values, and RAM[0] will hold the outcome.