I have finished all the work of hack computer. Based on Nand and DFF, using the testing script to test,

ComputerRect.tst (Draws a rectangle of width 16 pixels and length RAM[0] at the top left of the screen)

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, table, Excel

Description automatically generated

ComputerMax.tst (Computes the maximum of RAM[0] and RAM[1] and writes the result in RAM[2])

Graphical user interface, text, application

Description automatically generated

ComputerAdd.tst (Adds the two constants 2 and 3 and writes the result in RAM[0])

Graphical user interface, text, application

Description automatically generated

Part 2

Graphical user interface, application

Description automatically generated

Graphical user interface, table

Description automatically generated

Table

Description automatically generated

Instruction 1: 0000000000000010 As the opcode (the first bit) is 0, this is an A instruction. The symbolic syntax is @value. So the given instruction can be expressed as @2. The A-register is set to 2. This is done by setting the load bit (the control bit) in the A register to 1. The A-register puts 2 to the address bus. This is used to access RAM[2]. The value of RAM[2] is put in to the data bus and is output as the M register (M input).

Instruction 2: 1110110000010000 As the opcode (the first bit) is 1, this is a C instruction. So the syntax is D=A. The D register is set to value of RAM[2] i.e., M. This is done by setting the load bit (the control bit) in the D register to 1. The A-register puts 2 to the address bus. This is used to access RAM[2]. The value of RAM[2] is put in to the data bus and is output as the D register (D input).

Instruction 3: 0000000000000011 As the opcode (the first bit) is 0, this is an A instruction. The symbolic syntax is @value. So the given instruction can be expressed as @3. The A-register is set to 3. This is done by setting the load bit (the control bit) in the A register to 1. The A-register puts 3 to the address bus. This is used to access RAM[3]. The value of RAM[3] is put in to the data bus and is output as the M register (M input).

Instruction 4: 1110000010010000 As the opcode (the first bit) is 1, this is an C instruction. So the syntax is D=A. Here the instruction is D=D+A. First the control bit of D and A (RAM[3]) is set to read the value of D and A (RAM[3])and to put into the data bus which will then be used as the input of ALU, Now the control bit of ALU is set to ADD these two values. The output of the ALU will then be stored in the D register. This is done by setting the load bit (the control bit) in the D register to 1 and is output as the D register (D input).

Instruction 5: 0000000000000000 As the opcode (the first bit) is 0, this is an A instruction. The symbolic syntax is @value. So the given instruction can be expressed as @0. The A-register is set to 0. This is done by setting the load bit (the control bit) in the A register to 1. The A-register puts 0 to the address bus. This is used to access RAM[0]. The value of RAM[0] is put in to the data bus and is output as the M register (M input).

Instruction 6: 1110001100001000 As the opcode (the first bit) is 1, this is an C instruction. So the syntax is D=A. Here the instruction is M=D. First the control bit of D is set to read the value of D and to put into the data bus. Now the control bit of M register is set to 1 to load the value of the data bus into M (RAM[0]) and is output as the M register (M input).