1. Hello and welcome to the presentation video on 4-bit computer assignment for the course EEE 315. My name is Mir Sayeed Mohammad and I will be presenting my solution for the assignment task.
2. The main task is to implement a computer for the given instruction set. In my solution, I have done the main task using verilog, tested the solution on EDA-playground and also wrote a python assembler for programming machine code instructions from assembly files for my 4-bit computer
3. The verilog computer module has data\_in, oscillator clock, reset, program address, program in, program load pins as input and data\_out pins as output
4. A detailed description of the module is given here
5. The reset pin resets all the registers of the module excluding the memory units
6. Oscillator clock acts as the primary clock source
7. Program in pin is the 8 bit input pins for loading instructions on the program memory
8. Data in is the input port for loading data on the data memory which is separate from the program memory
9. Program address pins are for selecting the memory location of both program and data memory while uploading instructions
10. On the positive edge of program clock, data and instruction is loaded on the module
11. A and B are the two main registers of the computer
12. The temporary register is used for the exchange instruction in particular
13. Instruction pointer register points to a location on the program memory. The instruction opcode is loaded on the INS register and operand is loaded on the ADD register on negative clock edge of each cycle
14. ADD register is used to access data from data memory
15. Stack pointer is used to access the stack memory of the computer
16. Halt flag is used to check whether the program execution reached its end.
17. The instruction decoding is done by verilog synthesis using if-else conditional statements
18. Each positive clock edge acts as execution step and each negative clock edge acts as the fetch step. So the computer can execute instructions every clock cycle – which is a unique aspect of this 4-bit computer design.
19. Here is the set of blocking statements that are used for execution of each instruction
20. The rest of the instructions are given on this slide. There are total 16 instructions
21. On EDA playground, a testbench code was written to program the computer and test performance. The testbench module contains the machine code which is first uploaded to the computer module, and then by releasing the reset pin, the main module starts executing instructions.
22. Here is the first sample program running on EDA playground
23. On the left we can see the instructions and data that is being uploaded by the testbench to the computer module. On the right we can see the instructions and data on a table
24. After running the simulation, the following waveform shows up on the web interface
25. The first part of the runtime is program being uploaded to computer module. Here we can see the program addressed being incremented
26. On each positive edge of program clock, program and data is being uploaded byte by byte
27. The reset flag is held high while uploading, and released for starting execution
28. While loading program, the computer does not operate
29. After computer starts execution of program, the instruction pointer steadily increases in value
30. On each negative clock edge, instruction and operand are loaded on to the two registers
31. Instructions are executed on positive clock edges
32. Firstly decimal data 2 is loaded to register A from memory location 0
33. Then the values of A and B are exchanged
34. Again, decimal data 1 is loaded to register A from memory location 1
35. A and B register data is added and loaded to register A
36. Now and operation is done on register A and decimal 0 stored on memory location 3 giving the result of 0 which makes zero flag go high
37. The contents of register A is sent to the output port of the computer module
38. Finally the instruction pointer stops incrementing further – essentially brining the computer to a halt
39. Here is another demo program being uploaded and executed on the 4-bit computer. We wont be going into the details for this program.
40. In this program, the call, return, push and pop capabilities of the 4-bit computer has been demonstrated.
41. As part of the assignment, a python assembler was implemented. The python code parses the input assembly code and converts it to machine code. It can detect data and code segments in the program, and make use of x, y, z variables for accessing data memory. The assembler can also decode jump instructions and procedures.
42. On the left is a demo assembly code written for our computer, and on the right we can see the command line output while the assembler decodes the assembly code and generates machine code
43. Finally, a verilog snippet is output by the assembler.
44. The verilog snippet replaces the program previously used on the testbench module, and then we have to run the simulation
45. As expected, the generated machine code runs flawlessly on the 4-bit computer
46. And with this, we have reached the end of this presentation, additional details on the operation of the computer will be presented on the report. Thank you for your patience.