

ENCS2110

DIGITAL ELECTRONICS AND COMPUTER ORGANIZATION LABORATORY

Experiment 10: Simple Computer Simulation

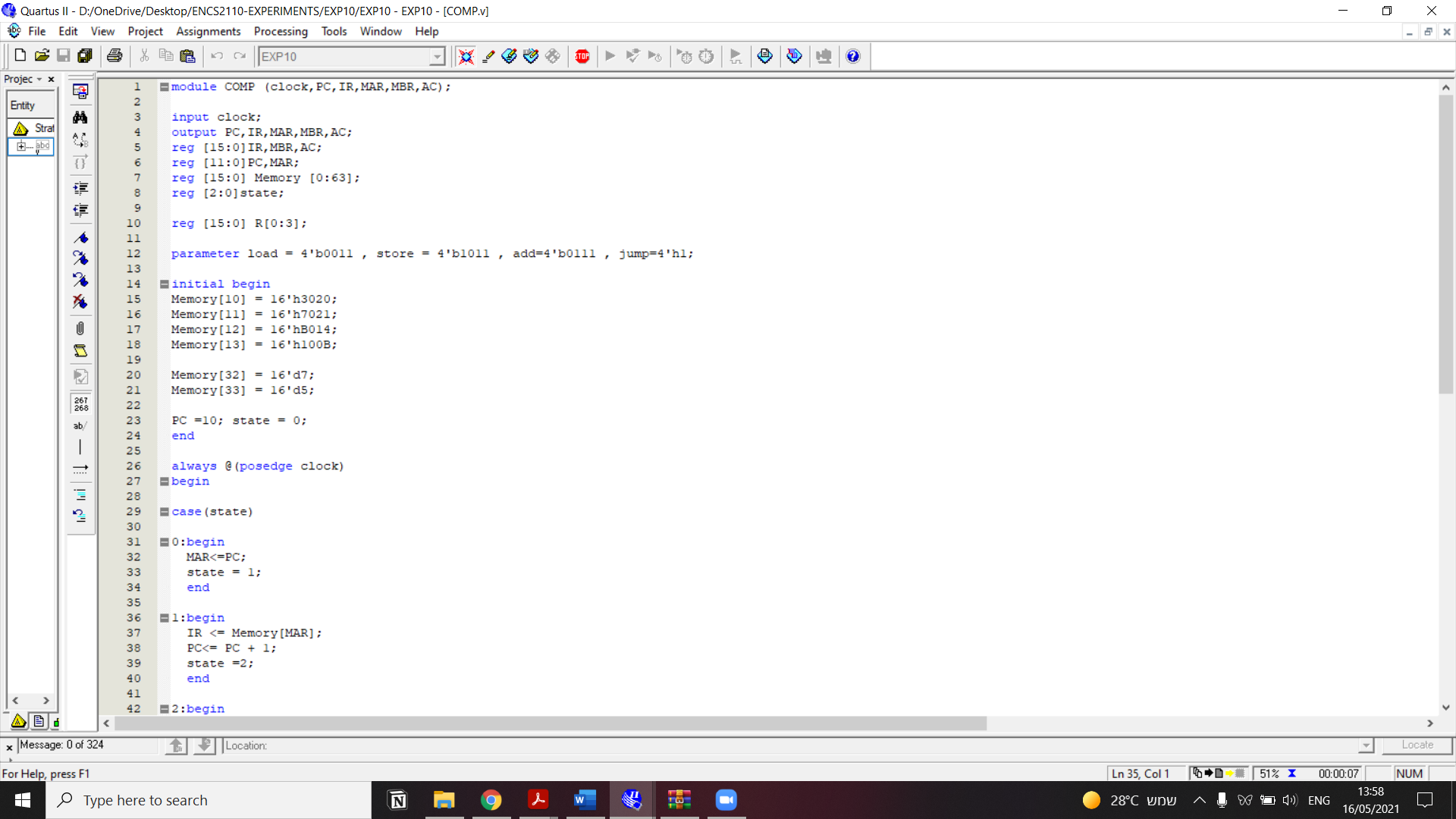
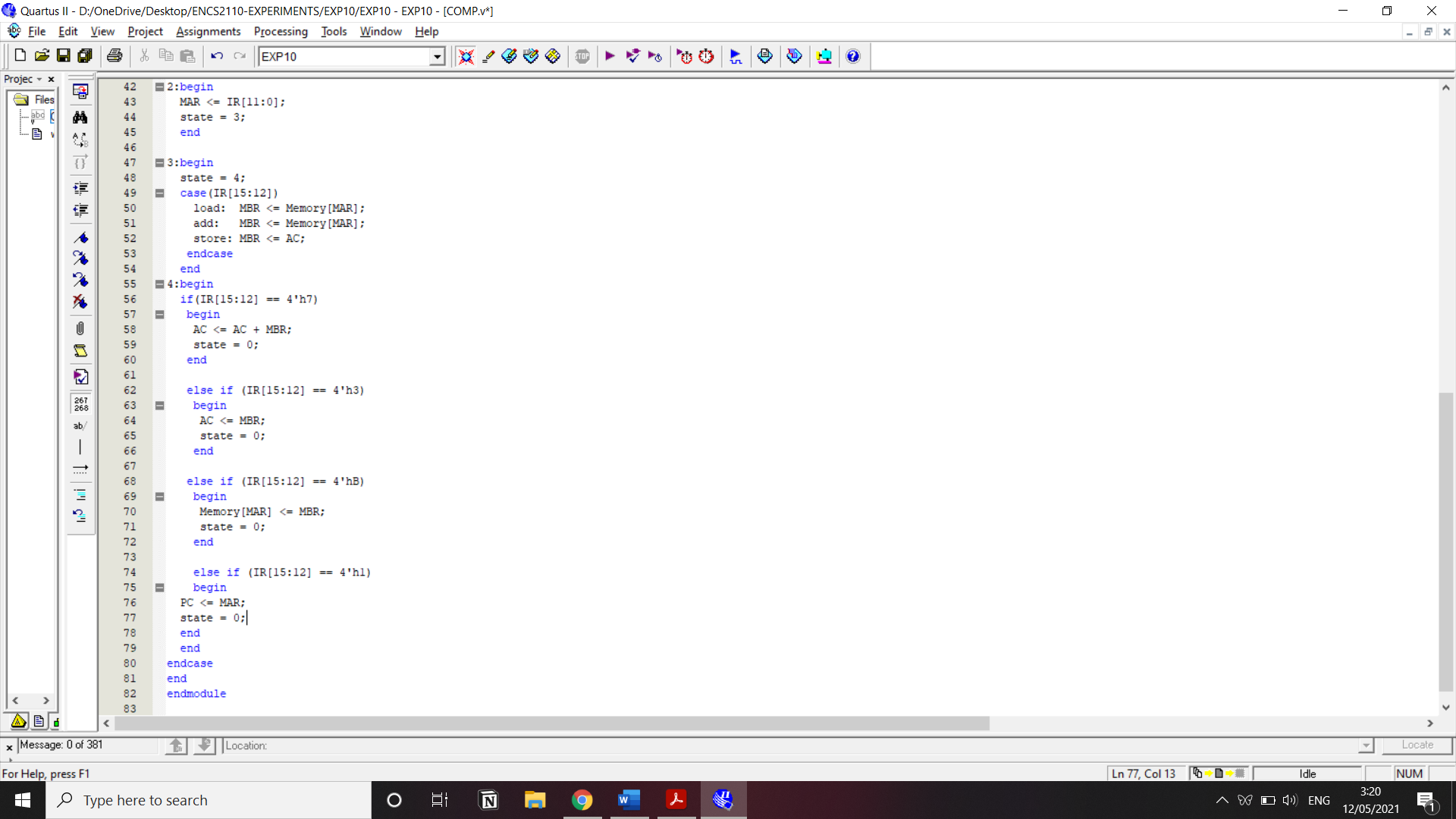
Name: Sara Totah

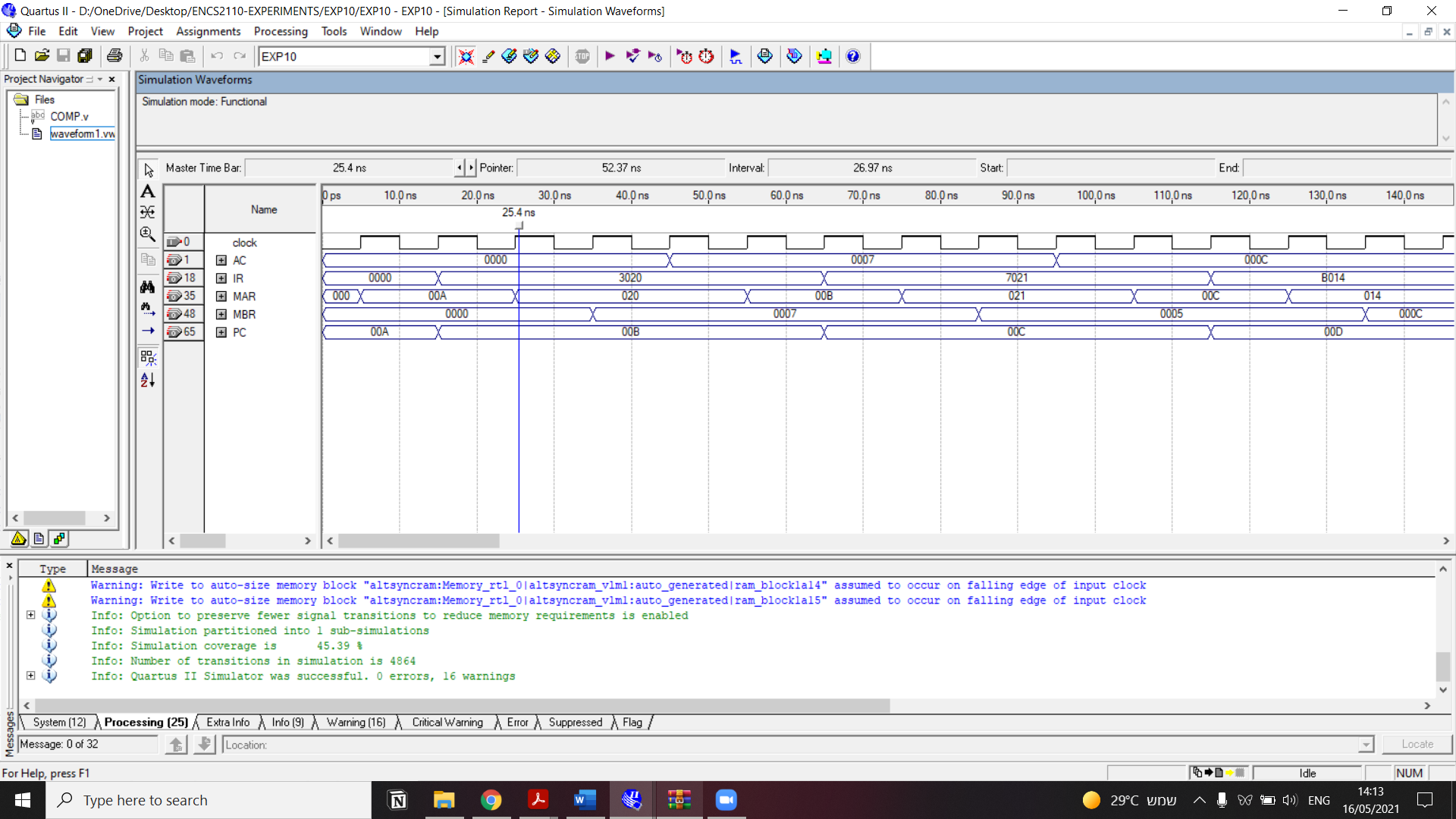
Student ID: 1181779

Instructor: Khader Mohammed

T.A: Enas Jawabreh

Date: 12/05/2021



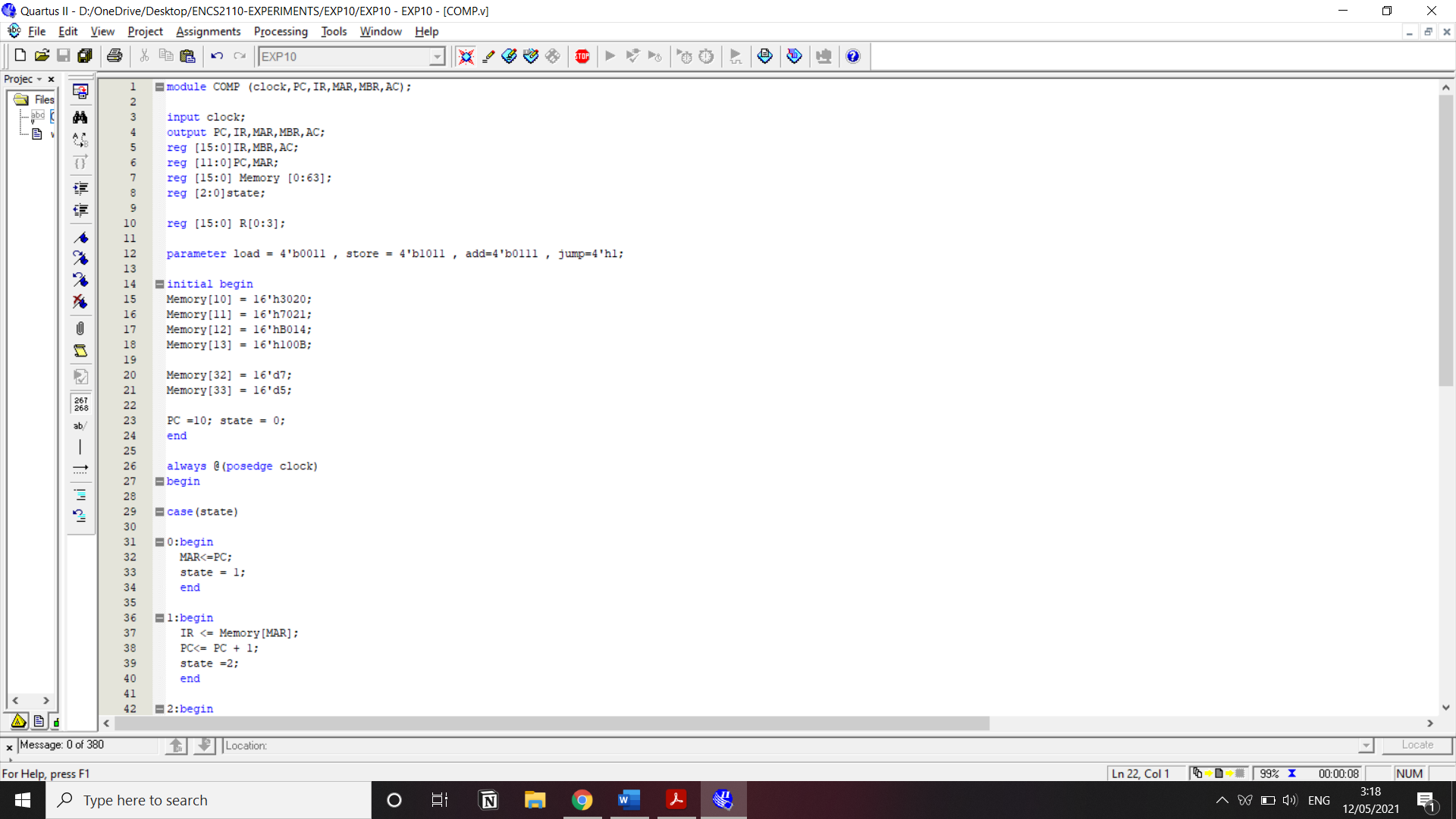
# Task 1:

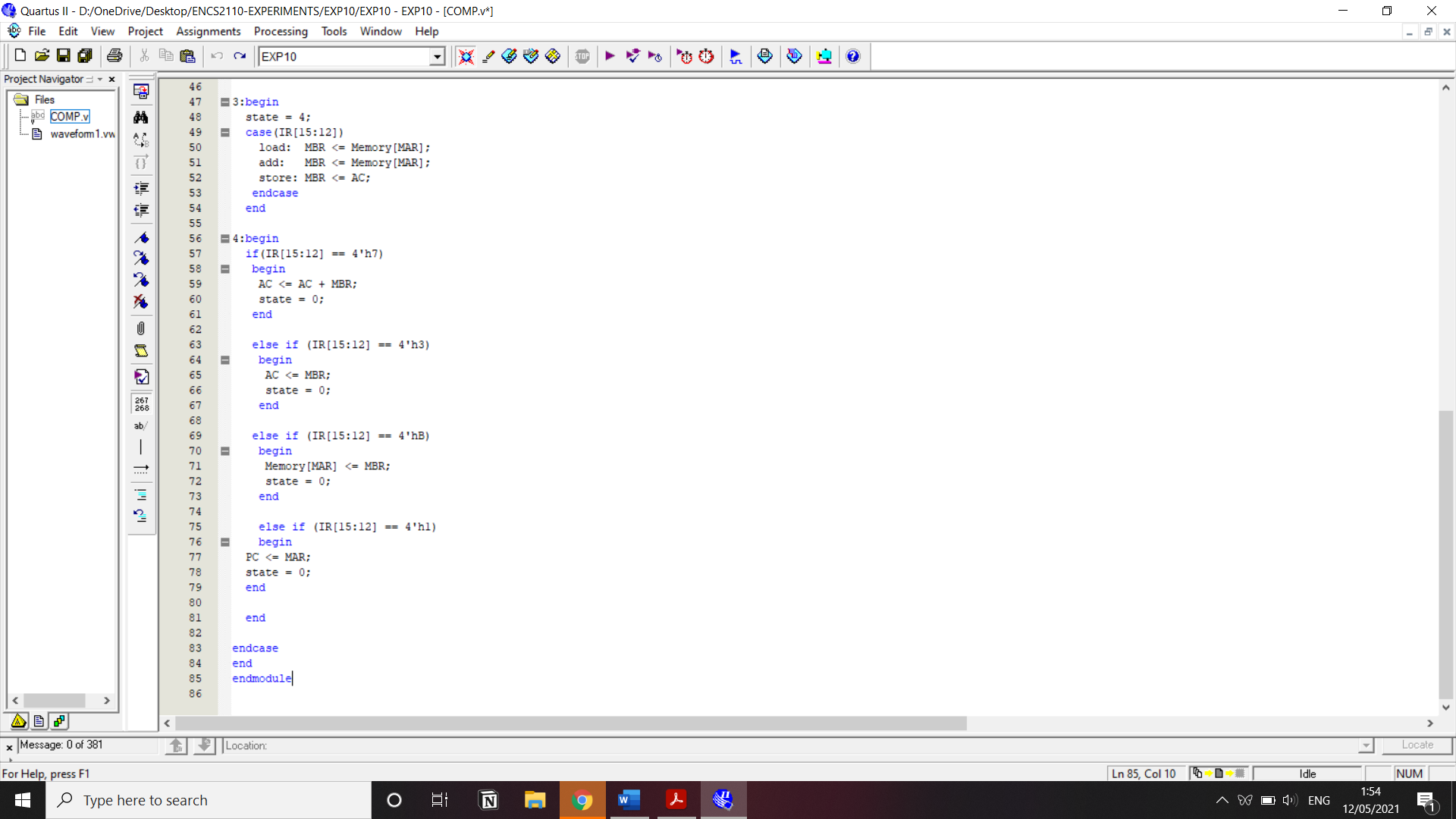
Modify the code to include the jump instruction.

1. Choose any opcode e.g., jump=4'b0001, you have to include the execution of jump which changes the PC to the specified address in the instruction.

|  |  |  |  |
| --- | --- | --- | --- |
| 10 | Load | 32 | 0011-0000-0010-0000b=16'h3020 |
| 11 | Add | 33 | 0111-0000-0010-0001b=16'h7021h |
| 12 | Store | 20 | 1011-0000-0001-0100b=16'hB014h |
| 13 | Jump | 11 | 0001-0000-0000-1011b=16'h100B |
| 32 | Data | 7 | Memory[32]=16'd7 |
| 33 | Data | 5 | Memory[32]=16'd5 |

The code added to include the jump instruction:





Discussion

Variables used:

Clock: simulates computer internal clock

* Registers: PC, IR, MBR, AR, AC
* Memory: 64\*16
* State: To manage instruction cycle steps

Instruction types our program supports:

* Load: [0011]
* Store: [1011]
* Add: [0111]
* Jump: [0001]

Initialization:

* Memory:

initial begin

Memory[10] = 16'h3020; //load data at address 20Hex into AC

Memory[11] = 16'h7021; //add data at address 21Hex into AC

Memory[12] = 16'hB014; //store data into address 14Hex

Memory[13] = 16'h100B; //jump to address

* Data

Memory[32] = 16'd7; // store at address 20Hex or 32 decimal data = 7 decimal

Memory[33] = 16'd5; // store at address 21Hex or 33 decimal data = 5 decimal

* PC = 10 // the program should start from the address 10D = 0AH
* State = 0
* Instruction Cycle:
  + Instruction Fetch
    - The address in PC -> MAR
    - Data at the address saved in MAR Memory[MAR] -> IR
    - Increment PC
  + Operand Fetch
    - Take the address of the operand saved at IR -> MAR
    - Load data at the memory in MAR Memory[MAR] -> MBR
    - Store data in AC -> MBR
  + Instruction Execute //based on first 4-bits of IR
    - If instruction was ADD then AC = AC + MBR
    - If instruction was LOAD, then AC = MBR
    - If instruction was STORE then Memory[MAR] = MBR
    - If instruction was JUMP then PC = MAR

**Code Explanation:**

Instruction #1: LOAD [20H]

|  |  |  |  |
| --- | --- | --- | --- |
| State  0 | | PC  0AH | |
| IR | MAR | MBR | AC |

­­­­­­­­­­­­­

**MAR<=PC;**

**state = 1;**

**end**

|  |  |  |  |
| --- | --- | --- | --- |
| State  1 | | PC  0AH | |
| IR | MAR  0000 0000 0000 1010 | MBR | AC |

**1:begin**

**IR <= Memory[MAR];**

**PC<= PC + 1;**

**state =2;**

**end**

|  |  |  |  |
| --- | --- | --- | --- |
| State  2 | | PC  0BH | |
| IR  0011 0000 0010 0000 | MAR | MBR | AC |

**2:begin**

**MAR <= IR[11:0];**

**state = 3;**

**end**

|  |  |  |  |
| --- | --- | --- | --- |
| State  3 | | PC  0BH | |
| IR  0011 0000 0010 0000 | MAR  0000 0010 0000 | MBR | AC |

**3:begin**

**state = 4;**

**case(IR[15:12])**

**load: MBR <= Memory[MAR];**

**add: MBR <= Memory[MAR];**

**store: MBR <= AC;**

**endcase**

**end**

|  |  |  |  |
| --- | --- | --- | --- |
| State  4 | | PC  0BH | |
| IR  0011 0000 0010 0000 | MAR  0000 0010 0000 | MBR  0000 0000 0000 0111 | AC |

**else if (IR[15:12] == 4'h3)**

**begin**

**AC <= MBR;**

**state = 0;**

**end**

|  |  |  |  |
| --- | --- | --- | --- |
| State  0 | | PC  0BH | |
| IR  0011 0000 0010 0000 | MAR  0000 0010 0000 | MBR  0000 0000 0000 0111 | AC  0000 0000 0000 0111 |

Instruction #2: ADD [21H]

|  |  |  |  |
| --- | --- | --- | --- |
| State  0 | | PC  0BH | |
| IR | MAR | MBR | AC  0000 0000 0000 0111 |

**0:begin**

**MAR<=PC;**

**state = 1;**

**end**

|  |  |  |  |
| --- | --- | --- | --- |
| State  1 | | PC  0BH | |
| IR | MAR  0000 0000 0000 1011 | MBR | AC  0000 0000 0000 0111 |

**1:begin**

**IR <= Memory[MAR];**

**PC<= PC + 1;**

**state =2;**

**end**

|  |  |  |  |
| --- | --- | --- | --- |
| State  2 | | PC  0CH | |
| IR  0111 0000 0010 0001 | MAR  0000 0000 0000 1011 | MBR | AC  0000 0000 0000 0111 |

**2:begin**

**MAR <= IR[11:0];**

**state = 3;**

**end**

|  |  |  |  |
| --- | --- | --- | --- |
| State  2 | | PC  0CH | |
| IR  0111 0000 0010 0001 | MAR  0000 0010 0001 | MBR | AC  0000 0000 0000 0111 |

**3:begin**

**state = 4;**

**case(IR[15:12])**

**load: MBR <= Memory[MAR];**

**add: MBR <= Memory[MAR];**

**store: MBR <= AC;**

**endcase**

**end**

|  |  |  |  |
| --- | --- | --- | --- |
| State  3 | | PC  0CH | |
| IR  0111 0000 0010 0001 | MAR  0000 0010 0001 | MBR  0000 0000 0000 0101 | AC  0000 0000 0000 0111 |

**4:begin**

**if(IR[15:12] == 4'h7)**

**begin**

**AC <= AC + MBR;**

**state = 0;**

**end**

|  |  |  |  |
| --- | --- | --- | --- |
| State  4 | | PC  0CH | |
| IR  0111 0000 0010 0001 | MAR  0000 0010 0001 | MBR  0000 0000 0000 0101 | AC  0000 0000 0000 1100 |

Instruction #3: STORE [14H]

|  |  |  |  |
| --- | --- | --- | --- |
| State  0 | | PC  0CH | |
| IR | MAR | MBR | AC  0000 0000 0000 1100 |

**0:begin**

**MAR<=PC;**

**state = 1;**

**end**

|  |  |  |  |
| --- | --- | --- | --- |
| State  1 | | PC  0CH | |
| IR | MAR  0000 0000 0000 1100 | MBR | AC  0000 0000 0000 1100 |

**1:begin**

**IR <= Memory[MAR];**

**PC<= PC + 1;**

**state =2;**

**end**

|  |  |  |  |
| --- | --- | --- | --- |
| State  2 | | PC  0CH | |
| IR  1011 0000 0001 0100 | MAR | MBR | AC  0000 0000 0000 1100 |

**2:begin**

**MAR <= IR[11:0];**

**state = 3;**

**end**

|  |  |  |  |
| --- | --- | --- | --- |
| State  3 | | PC  0DH | |
| IR  1011 0000 0001 0100 | MAR  0000 0001 0100 | MBR | AC  0000 0000 0000 1100 |

**3:begin**

**state = 4;**

**case(IR[15:12])**

**load: MBR <= Memory[MAR];**

**add: MBR <= Memory[MAR];**

**store: MBR <= AC;**

**endcase**

**end**

|  |  |  |  |
| --- | --- | --- | --- |
| State  4 | | PC  0DH | |
| IR  1011 0000 0001 0100 | MAR  0000 0001 0100 | MBR  0000 0000 0000 1100 | AC  0000 0000 0000 1100 |

**else if (IR[15:12] == 4'hB)**

**begin**

**Memory[MAR] <= MBR;**

**state = 0;**

**end**

|  |  |  |  |
| --- | --- | --- | --- |
| State  4 | | PC  0DH | |
| IR  1011 0000 0001 0100 | MAR  0000 0001 0100 | MBR  0000 0000 0000 1100 | AC  0000 0000 0000 1100 |

**Memory [14H] = 0000 0000 0000 1100**

Instruction #4: JUMP [1H]

**else if (IR[15:12] == 4'h1)**

**begin**

**PC <= MAR;**

**state = 0;**

**end**

Changes the address that should be executed next, in other words, the new address saved at the PC is the address of the new instruction to be excuted.