|  |
| --- |
| **Ram Lal Anand College**  **University of Delhi**    **Department of Computer Science**    **Session :- November- March 2021**    **Assignment**  **Submitted to Ms. Nupur Tyagi**      **Program Name: - B.Sc(H) Computer Science**  **Semester - I**  **Title of the paper:- Computer System Architecture**  **Paper Code:- 32341102**  **Name of the Student:- Aman Goyal**  **Examination/Class Roll No:- 4037** |

Ques 1. Create a machine based on the following architecture:

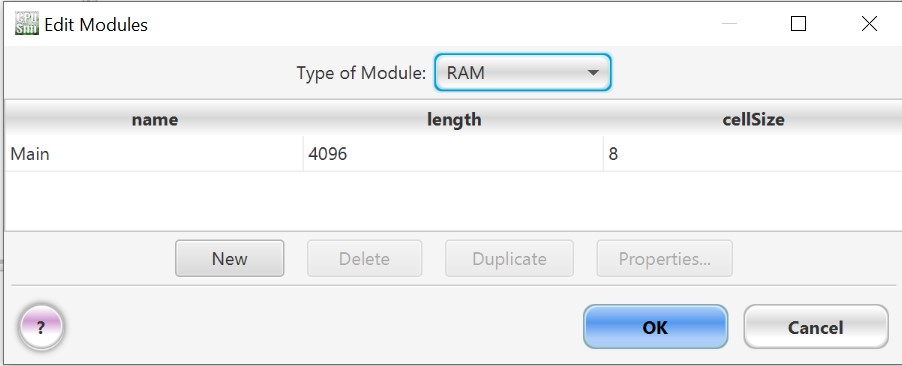
**Registers**

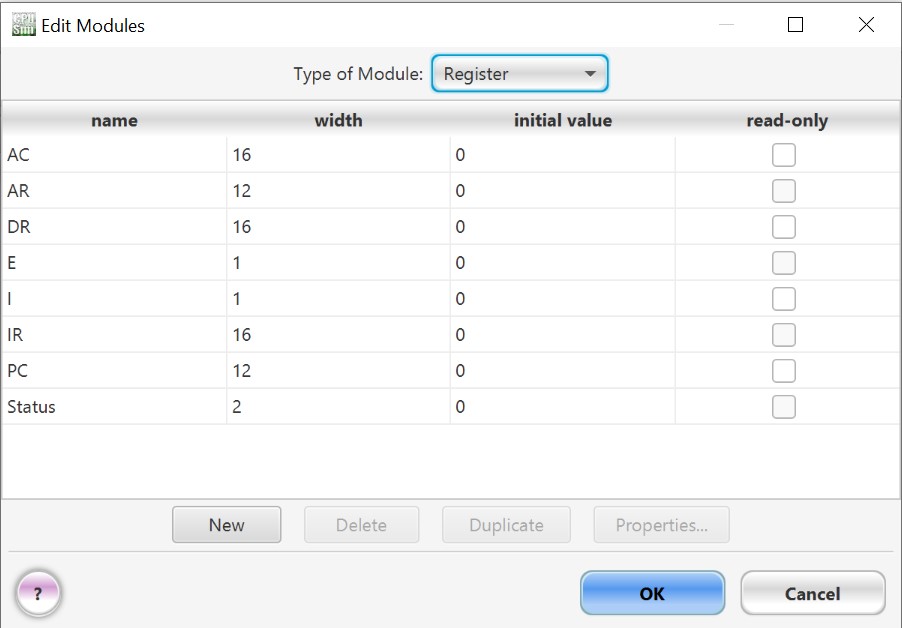
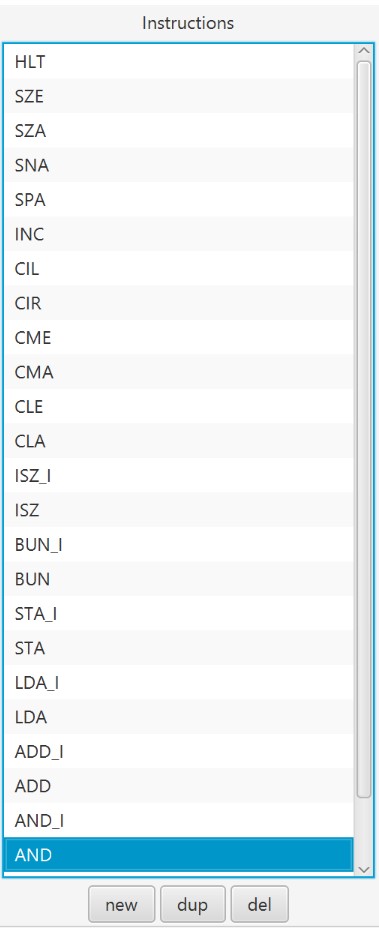
IR DR AC AR PC I E

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 15 | 0 15 | 0 15 | 0 | 11 | 0 11 | 1 bit | 1 bit |
|  | | | |  | | | |
| Memory  4096 Words  8 bits per word | | | | Instruction Format  0 3 4 15   |  |  | | --- | --- | | Opcode | Address | | | | |

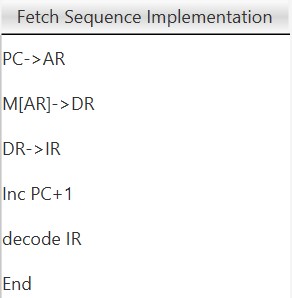
**Basic Computer Instructions**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Memory Reference | | | | | | | |  | Register Reference | | | | | | | |
|  | Symbol | | | Hex | | | |  |  | Symbol | | | Hex | | |  |
|  | AND |  |  | 0xxx |  | |  | CLA |  |  | E800 |  |
| ADD | 2xxx | CLE | E400 |
|  | Direct Address |
| LDA | 4xxx | CMA | E200 |
| STA | 6xxx | CME | E100 |
| BUN | 8xxx | CIR | E080 |
|  |
|  |  | CIL | E040 |
| ISZ | Cxxx | INC | E020 |
|  | SPA |  |  | E010 |  |
| |  | | --- | | AND\_I | | ADD\_I | | LDA\_I | | STA\_I | | BUN\_I | | ISZ\_I | | | | 1xxx  3xxx | | | | SNA | E008 |
| SZA | E004 |
| |  | | --- | | 5xxx | | 7xxx | | 9xxx | | Dxxx | | | | Indirect Address |
| SZE | E002 |
| HLT | E001 |
|  |  |  |
|  | | | | | |

Ans. 

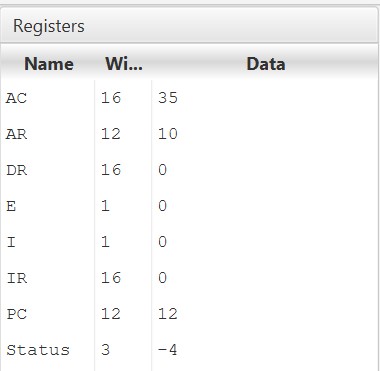
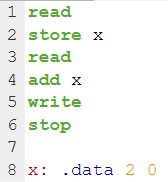


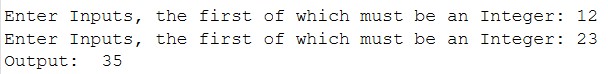
1. Create a Fetch routine of the instruction cycle.

Ans. 

1. Write an assembly program to simulate ADD operation on two user-entered numbers.

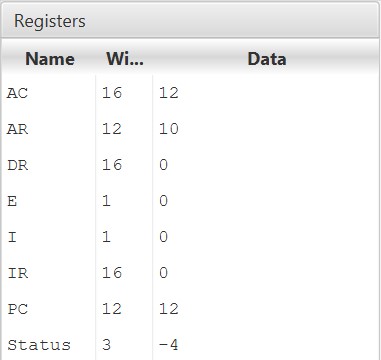
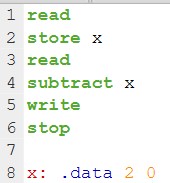
Ans. We take first input from user and save it in M[AR] using store microinstruction. The second input is stored in AC which is then added with the first input that is saved in M[AR] and then we display the result using write microinstruction. Then the stop micro-instruction ends the programs.

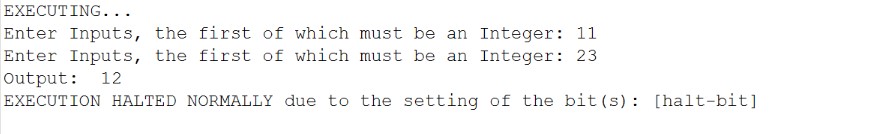




1. Write an assembly program to simulate SUBTRACT operation on two user entered numbers.

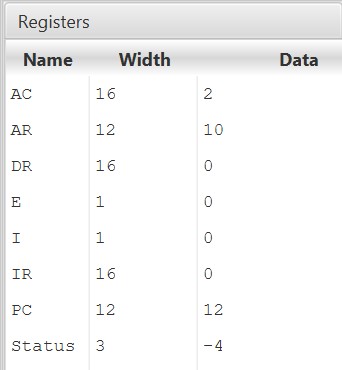
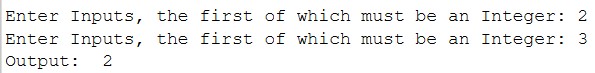
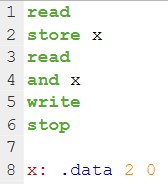
Ans. We take first input from user and save it in M[AR] using store microinstruction. The second input is stored in AC which is then subtracted with the first input that is saved in M[AR] and then we display the result using write micro-instruction. Then the stop micro-instruction ends the programs.



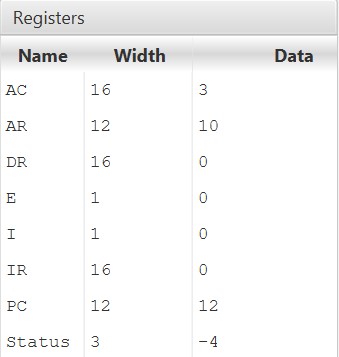
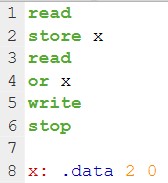


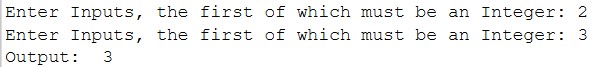
1. Write an assembly program to simulate the following logical operations on two user-entered numbers. a) AND
2. OR
3. NOT
4. XOR
5. NOR
6. NAND

Ans. a) We take first input from user and save it in M[AR] using store microinstruction. The second input is stored in AC which is then “and” according to binary value of the inputs with the first input that is saved in M[AR] and then we display the result using write micro-instruction. Then the stop microinstruction ends the programs.

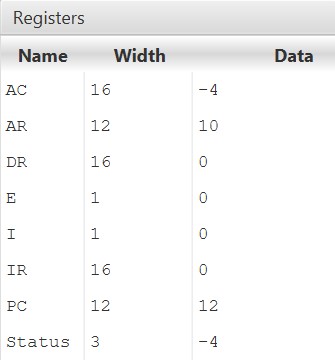
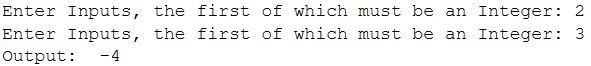
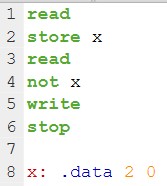


1. We take first input from user and save it in M[AR] using store microinstruction. The second input is stored in AC which is then “or” according to binary value of the inputs with the first input that is saved in M[AR] and then we display the result using write micro-instruction. Then the stop microinstruction ends the programs.

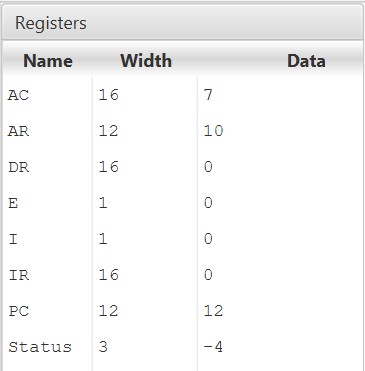
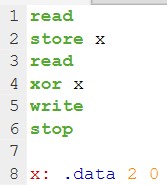


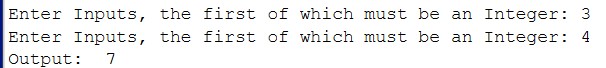


1. We take first input from user and save it in M[AR] using store microinstruction. The second input is stored in AC which is then “not” according to binary value of the inputs with the first input that is saved in M[AR] and then we display the result using write micro-instruction. Then the stop microinstruction ends the programs.

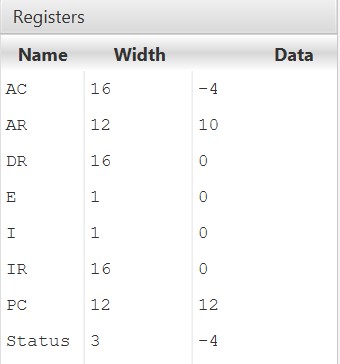
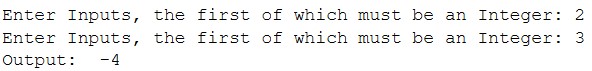
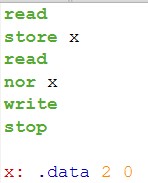


1. We take first input from user and save it in M[AR] using store microinstruction. The second input is stored in AC which is then “xor” according to binary value of the inputs with the first input that is saved in M[AR] and then we display the result using write micro-instruction. Then the stop microinstruction ends the programs.

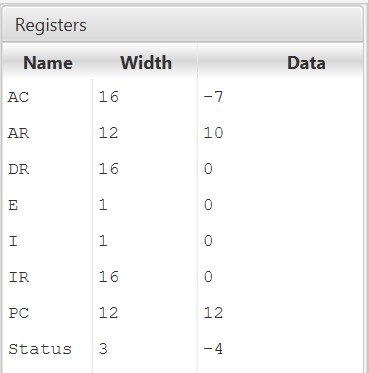
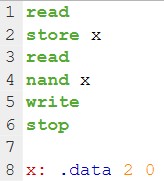
 

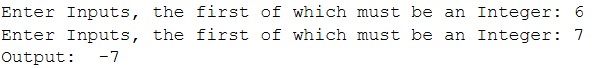


1. We take first input from user and save it in M[AR] using store microinstruction. The second input is stored in AC which is then “nor” according to binary value of the inputs with the first input that is saved in M[AR] and then we display the result using write micro-instruction. Then the stop microinstruction ends the programs.



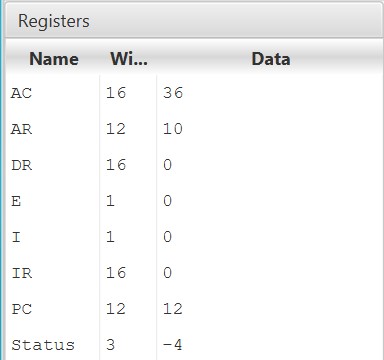
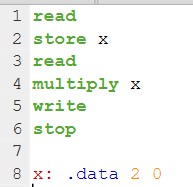
1. We take first input from user and save it in M[AR] using store microinstruction. The second input is stored in AC which is then “nand” according to binary value of the inputs with the first input that is saved in M[AR] and then we display the result using write micro-instruction. Then the stop microinstruction ends the programs.

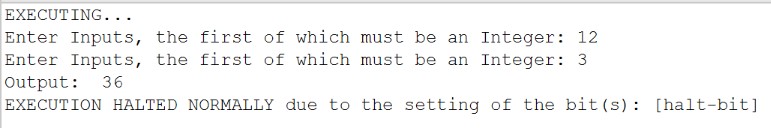




1. Write an assembly program to simulate MULTIPLY operation on two user entered numbers.

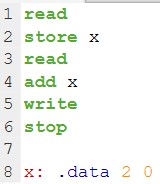
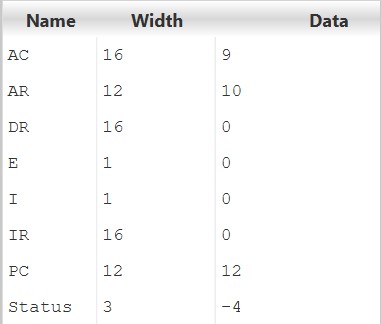
Ans. We take first input from user and save it in M[AR] using store microinstruction. The second input is stored in AC which is then multiplied with the first input that is saved in M[AR] and then we display the result using write micro-instruction. Then the stop micro-instruction ends the programs.





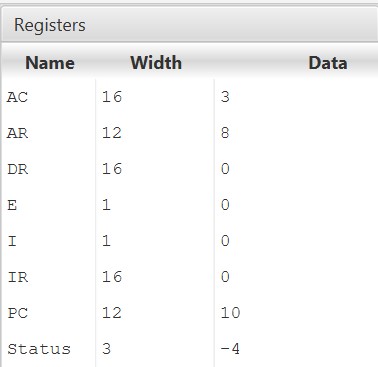
1. Write an assembly program to simulate following memory-reference instructions. a) ADD
2. LDA
3. STA
4. BUN
5. ISZ

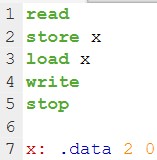
Ans. a) We take first input from user and save it in M[AR] using store microinstruction. The second input is stored in AC which is then added with the first input that is saved in M[AR] and then we display the result using write microinstruction. Then the stop micro-instruction ends the programs.



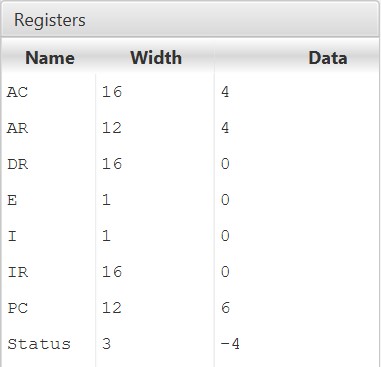
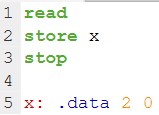
1. We take input from user and save it in M[AR] using store micro-instruction. We then load the value from M[AR] into AC using load micro-instruction and then we display the result using write micro-instruction. Then the stop microinstruction ends the programs.







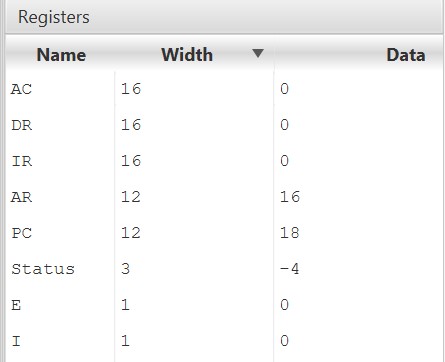
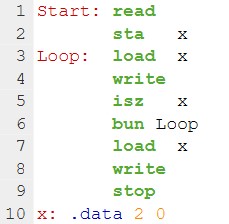
1. We take first input from user and save it in M[AR] using store microinstruction. Then the stop micro-instruction ends the programs.

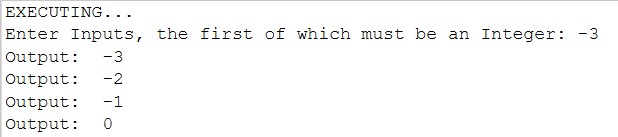




1. We take first input from user and save it in M[AR] using sta microinstruction. We then loop our code from load micro-instruction.

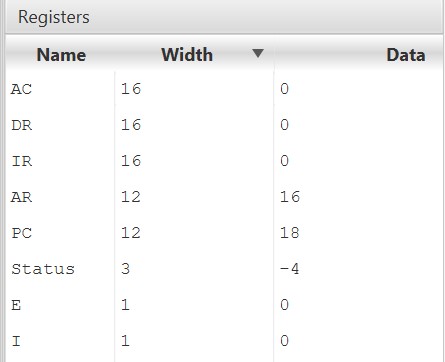
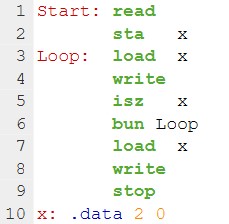
Inside the loop, we first load the value from M[AR] into DR using load microinstruction. We then display the value using write micro-instruction. After this we increment the value of the input and check if it is equal to 0 or not. If the value is equal to 0, we skip the next instruction and load the value and display it and stop the program using stop micro-instruction. If the value is not equal to 0, we don’t skip the next instruction which is “bun”. We go to beginning of the Loop again using bun and this process continues until value of the number is not equal to 0.

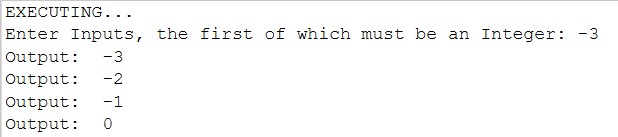




1. We take first input from user and save it in M[AR] using sta microinstruction. We then loop our code from load micro-instruction.

Inside the loop, we first load the value from M[AR] into DR using load microinstruction. We then display the value using write micro-instruction. After this we increment the value of the input and check if it is equal to 0 or not. If the value is equal to 0, we skip the next instruction and load the value and display it and stop the program using stop micro-instruction. If the value is not equal to 0, we don’t skip the next instruction which is “bun”. We go to beginning of the Loop again using bun and this process continues until value of the number is not equal to 0.

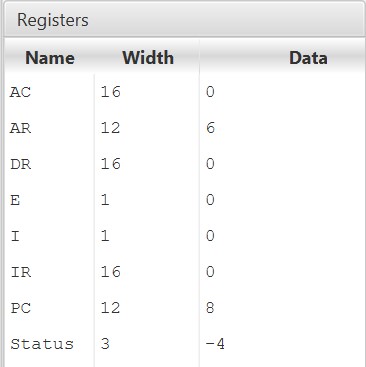




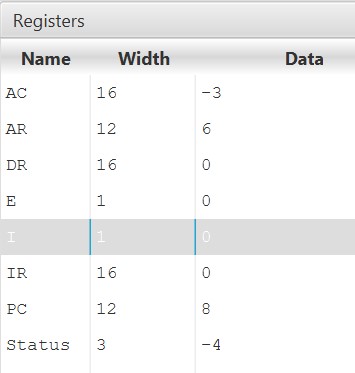
8. Write an assembly language program to simulate the machine for following register reference instructions and determine the contents of AC, E, PC, AR and IR registers in decimal after the execution: a) CLA

1. CMA
2. CME
3. HLT

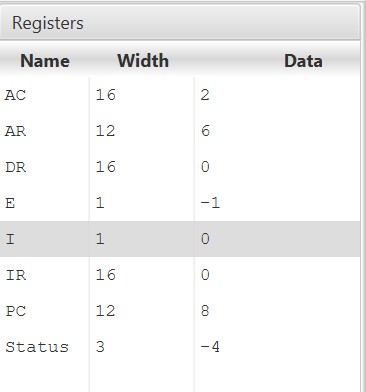
Ans. a) We take input from user which is stored in AC. This value is then cleared from AC using CLA and then we display the result using write microinstruction. Then the stop micro-instruction ends the programs.



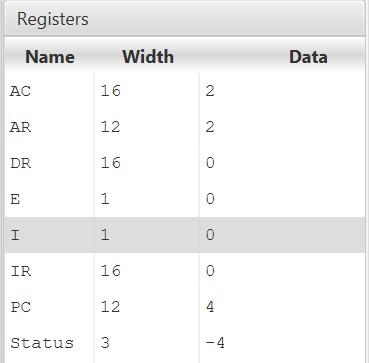
1. We take input from user which is stored in AC. The value of AC register is complemented using CMA and then we display the result using write microinstruction. Then the stop micro-instruction ends the programs.



1. We take input from user which is stored in AC. The value of E register is complemented using CME and then we display the result using write microinstruction. Then the stop micro-instruction ends the programs.



1. We take input from user which is stored in AC. Then the stop microinstruction ends the programs.

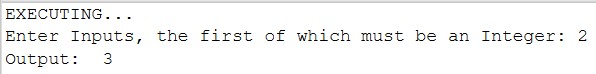
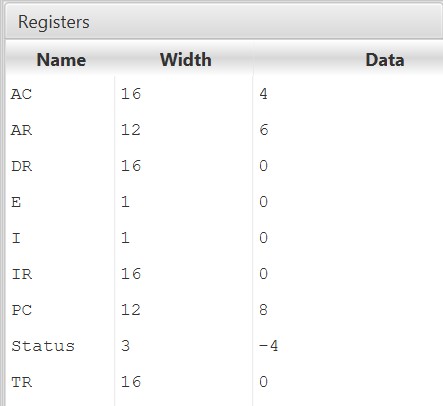
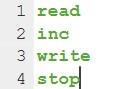




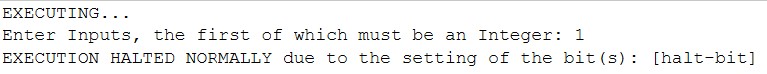
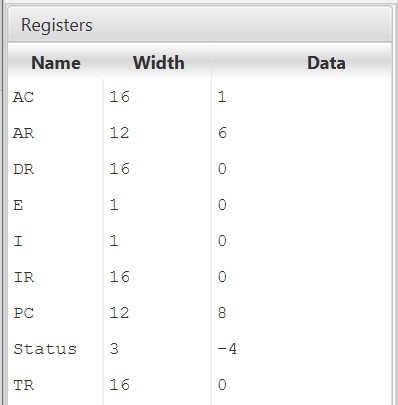
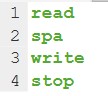
9. Write an assembly language program to simulate the machine for following register reference instructions and determine the contents of AC, E, PC, AR and IR registers in decimal after the execution: a) INC

1. SPA
2. SNA
3. SZE

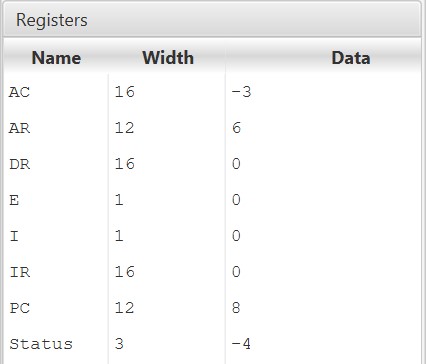
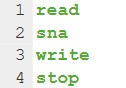
Ans. a) We take input from user which is stored in AC. The value of the input in AC register is increased by 1 using inc micro-instruction and then we display the result using write micro-instruction. Then the stop micro-instruction ends the programs.

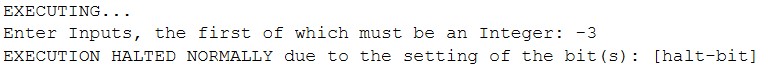


1. We take input from user which is stored in AC. We then check the value of the input using spa micro-instruction. If the value is positive, we skip the next instruction, else we display the result using write micro-instruction. Then the stop micro-instruction ends the programs.

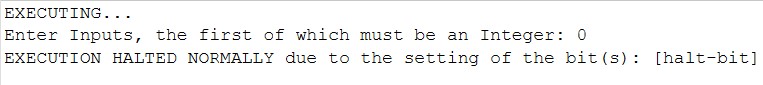
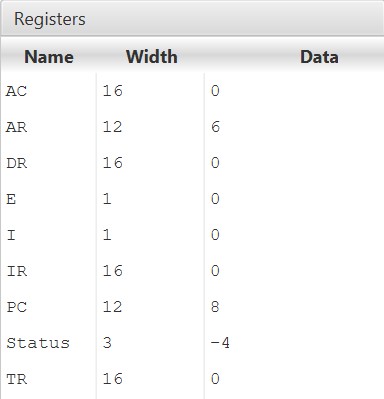
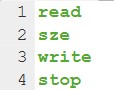


1. We take input from user which is stored in AC. We then check the value of the input using sna micro-instruction. If the value is negative, we skip the next instruction, else we display the result using write micro-instruction. Then the stop micro-instruction ends the programs.





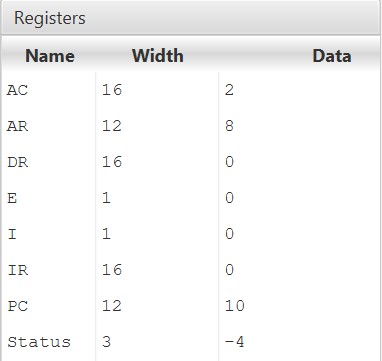
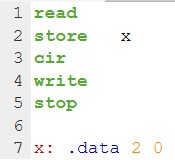
1. We take input from user which is stored in AC. We then check the value of the input using sze micro-instruction. If the value is zero, we skip the next instruction, else we display the result using write micro-instruction. Then the stop micro-instruction ends the programs.

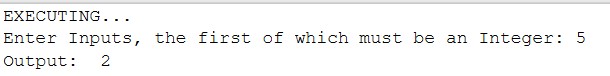


10. Write an assembly language program to simulate the machine for following register reference instructions and determine the contents of AC, E, PC, AR and IR registers in decimal after the execution: a) CIR

b) CIL

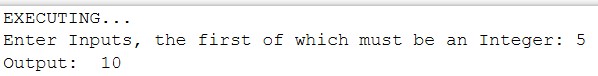
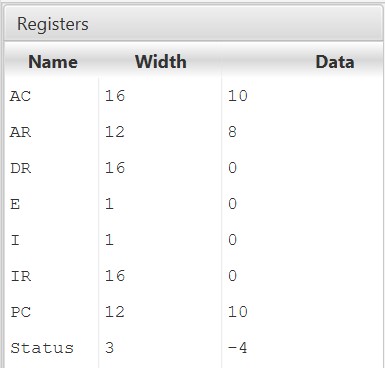
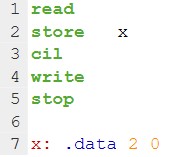
Ans.(a) We take input from user and save it in M[AR] using store microinstruction. The input is shifted circularly right by taking the binary form the input using cir micro-instruction and then we display the result using write micro-instruction. Then the stop micro-instruction ends the program.





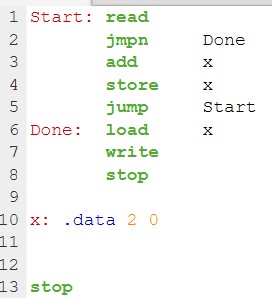
(b) We take input from user and save it in M[AR] using store micro-instruction. The input is shifted circularly left by taking the binary form the input using cil micro-instruction and then we display the result using write micro-instruction.

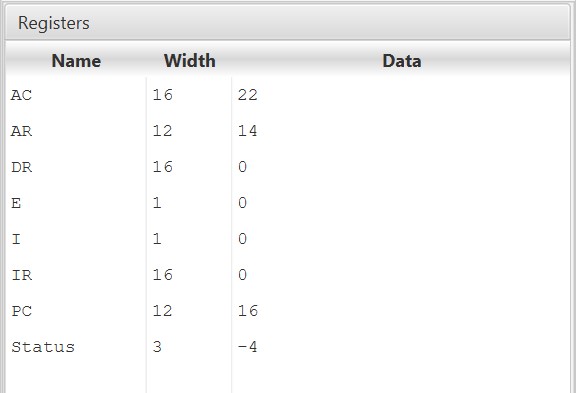
Then the stop micro-instruction ends the program.

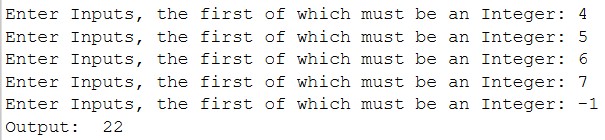


1. Write an assembly language program that reads in integers and adds them together, until a negative non-zero is read in. Then it outputs the sum (not including the last number).

Ans. We take input from user which is stored in AC. Then we check if the input is negative or not. If the input is negative, we jump to Done statement and load the value and display it and end the program. If the input is not negative, we don’t jump to Done loop and add the value and store the value, then we jump to the beginning of the Start loop and again check if the new input is negative or not.

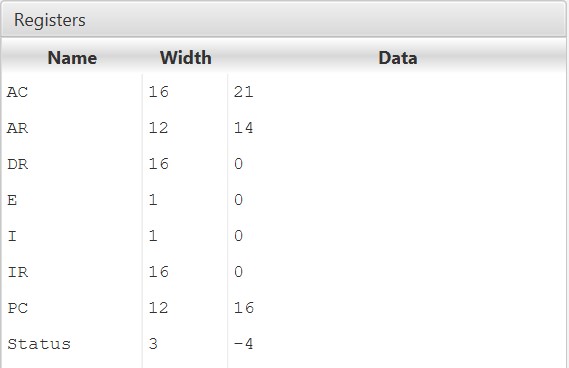
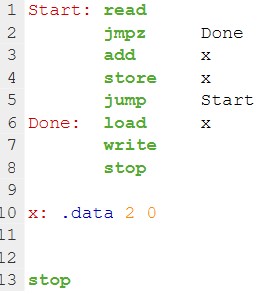


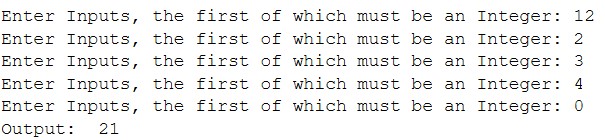




1. Write an assembly language program that reads in integers and adds them together, until zero is read in. Then it outputs the sum.

Ans.We take input from user which is stored in AC. Then we check if the input is negative or not. If the input is zero, we jump to Done statement and load the value and display it and end the program. If the input is not negative, we don’t jump to Done loop and add the value and store the value, then we jump to the beginning of the Start loop and again check if the new input is zero or not.





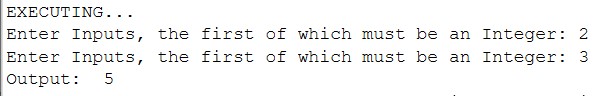
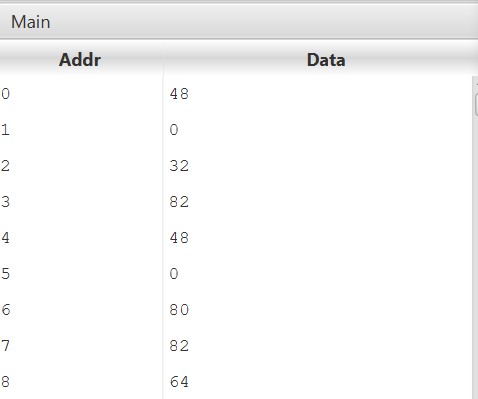
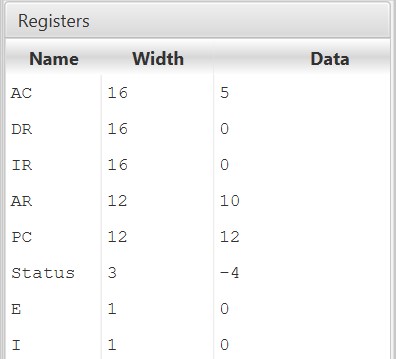
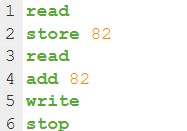
1. Create a machine for the following instruction format:

15 14 13 12 11 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OP Code |  | I |  | Address |

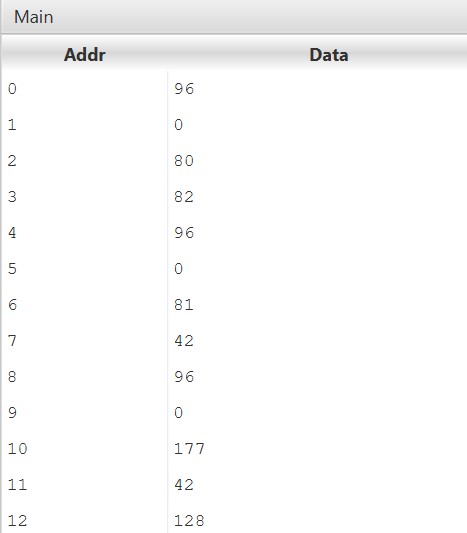
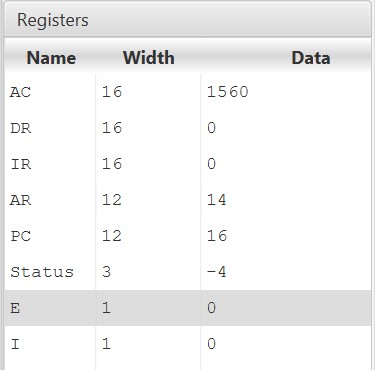
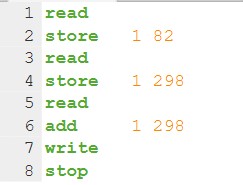
Write an Assembly Program to simulate the machine for addition of two numbers with I = 0 (Direct Address) and address part = 082. The instruction to be stored at address 022 in RAM, initialize the memory word with any decimal value at address 082. Determine the contents of AC, DR, PC, AR and IR in decimal after the execution.

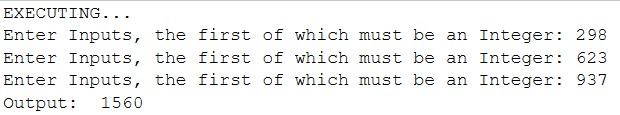
Ans



14. Simulate the machine for the memory-reference instruction referred in above question with I = 1 (Indirect Address) and address part = 082. The instruction to be stored at address 026 in RAM. Initialize the memory word at address 082 with the value 298. Initialize the memory word at address 298 with operand 632 and AC with 937. Determine the contents of AC, DR, PC, AR and IR in decimal after the execution.

Ans.





Ques 15. The instruction format contains 3 bits of opcode, 12 bits for address and 1 bit for addressing mode. There are only two addressing modes, I = 0 is direct addressing and I = 1 is indirect addressing. Write an assembly program to check the I bit to determine the addressing mode and then jump accordingly.

Ans.

