



UGC & Govt. Approved  
**Sonargaon University (SU)**  
সন্দৰ্ভ ইউনিভার্সিটি (এসইউ)

# Lab Report

Course Title : Microprocessor & Assembly  
Language Programming  
Sessional

Course Code : CSE312

Submitted To : Md. Ashfakur Rahman  
Lecturer  
Department of Computer Science  
And Engineering  
Sonargaon University

Submitted By : Tanvirujjaman  
CSE2202026130  
26A(Loomis)  
Summer - 2024

**Submission Date : 24 August 2024**

# Lab Report

## • Task 1

**Name:** Loop and nested loop operations (Patterns, dynamic input etc.)

**Solve** - Loop Patterns with dynamic input

1. include emu8086.inc

2. org 100h

3.

4. .data

5. number db ?

6.

7. .code

8. mov ax,@data

9. mov ds,ax

10. mov ah,00

11. mov bl,10

12. mov number,0

13. mov cl,0

14.

15. print "Enter a 3 digit number: "

16. input:

17. mov ah,01h

18. int 21h

19. cmp al,13

20. je output

21. sub al,48

22. mov cl,al

23. mov al,number

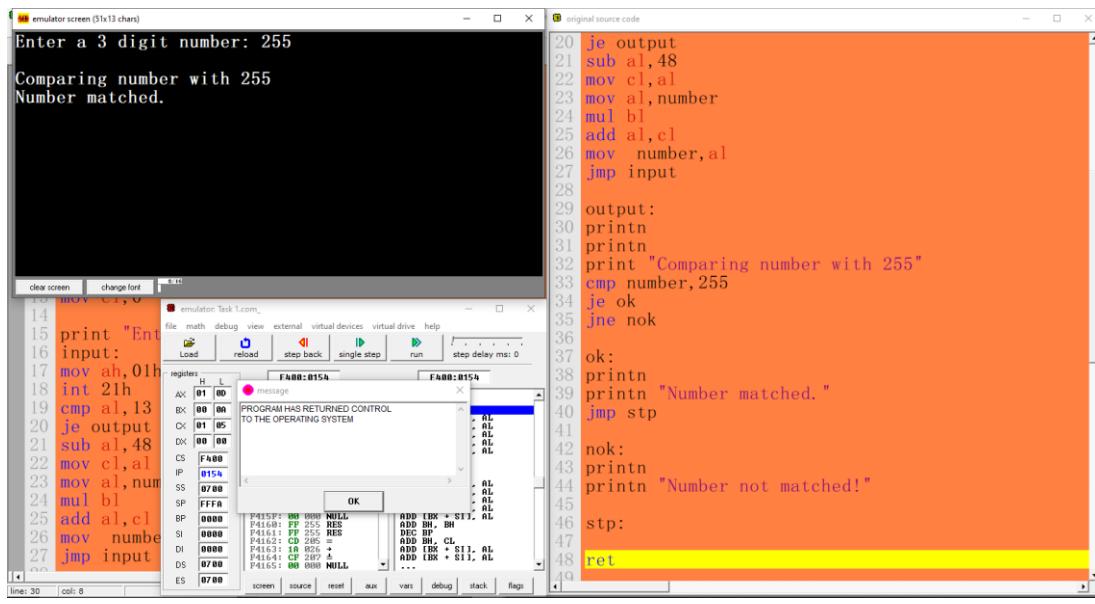
24. mul bl

25. add al,cl

26. mov number,al

27. jmp input

28.



29. output:  
 30. printn  
 31. printn  
 32. print "Comparing number with 255"  
 33. cmp number,255  
 34. je ok  
 35. jne nok  
 36.  
 37. ok:  
 38. printn  
 39. printn "Number matched."  
 40. jmp stp  
 41.  
 42. nok:  
 43. printn  
 44. printn "Number not matched!"  
 45.  
 46. stp:  
 47.  
 48. Ret

## • Task 2

**Name:** Conditions and nested conditions (Even/odd, positive/negative, vowel/consonant, validation etc.)

**Solve** - Even/odd with Conditions

1. include emu8086.inc
2. org 100h
- 3.
4. .data
5. res db ?
6. rem db ?
- 7.
8. .code
9. mov ax,@data

10. mov ds,ax

11. mov bl,2

12.

13. mov ah,01h

14. int 21h

15. sub al,48

16.

17. div bl

18. mov res,al

19. mov rem,ah

20.

21.

22. cmp rem,0

23. je Even

24. printn

25. printn "Odd Number"

26. jmp stp

27.

28. Even:

29. printn

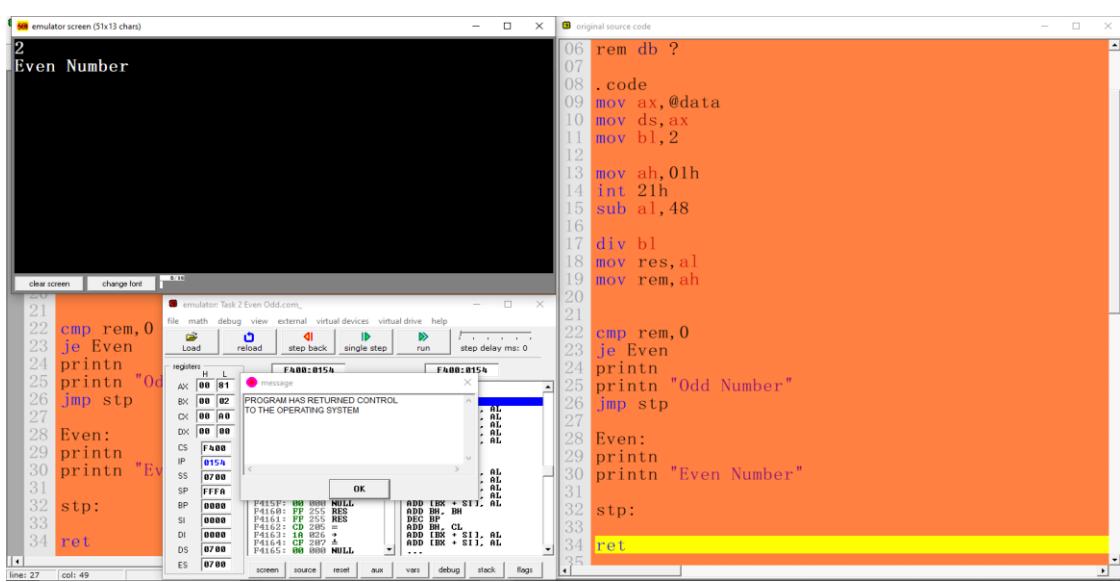
30. printn "Even Number"

31.

32. stp:

33.

34. Ret



## Solve - Vowel/Consonant with Conditions

1. include emu8086.inc

2. org 100h

3.

4. .data

5. str db 10,13,"Enter a alphabet: \$"

6.

7. .code

8. mov ax,@data

9. mov ds,ax

10. mov ax,00

11.

12. lea dx,str

13. mov ah,09h

14. int 21h

15.

16. mov ah,01h

17. int 21h

18. printn

19.

20. cmp al,'a'

21. je V

22. cmp al,'A'

23. je V

24.

25. cmp al,'e'

26. je V

27. cmp al,'E'

28. je V

29.

30. cmp al,'i'

31. je V

32. cmp al,'I'

33. je V

34.

35. cmp al,'o'

36. je V

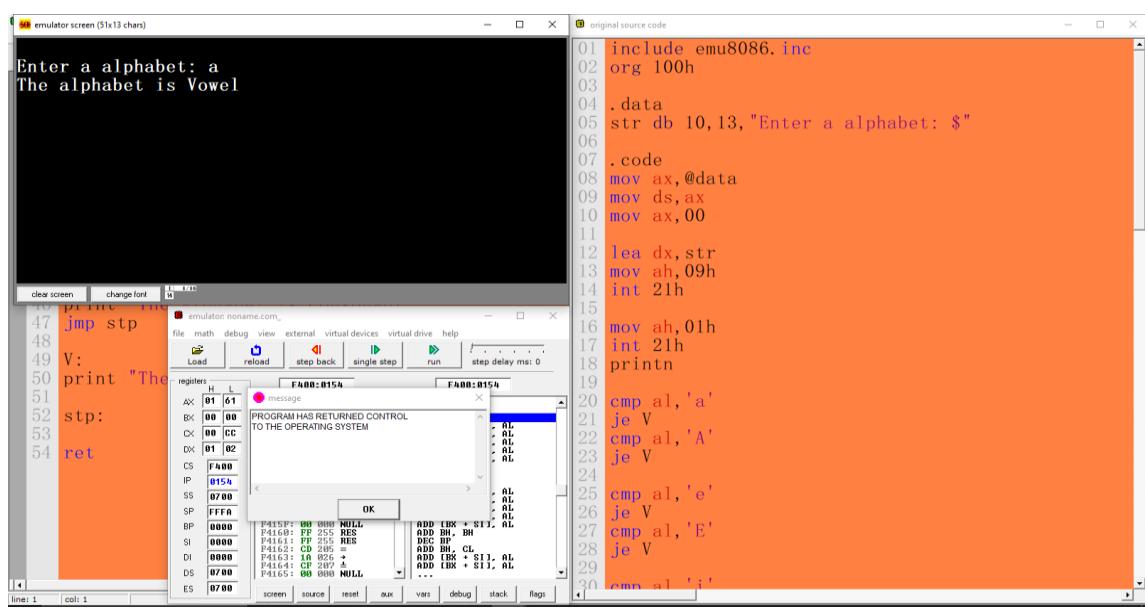
37. cmp al,'O'

38. je V

39.

40. cmp al,'u'

41. je V



42. cmp al,'U'  
 43. je V  
 44.  
 45. C:  
 46. print "The alphabet is Consonant"  
 47. jmp stp  
 48.  
 49. V:  
 50. print "The alphabet is Vowel"  
 51.  
 52. stp:  
 53.  
 54. Ret

## • Task 3

**Name:** String operations (String comparison, length of string, array etc.)

**Solve** - String comparison operation

1. include emu8086.inc

2. org 100h

3.

4. .data

5. username db  
 "Tanvir\$"

6. pass db "123\$"

7. .code

8. mov ax,@data

9. mov ds,ax

10.

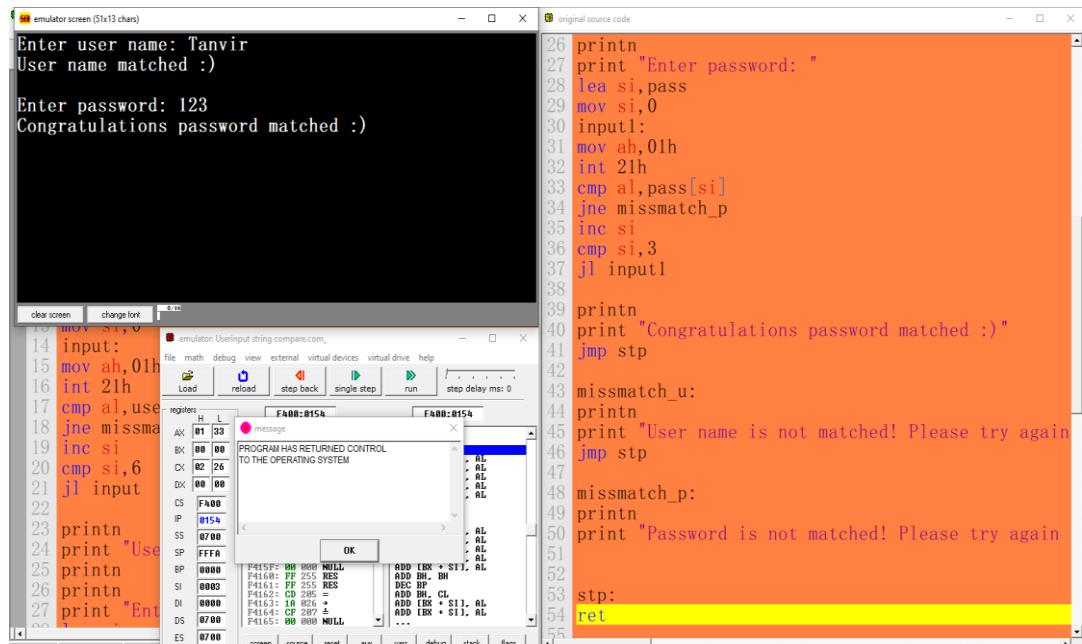
11. print "Enter user  
 name: "

12. lea si,username

13. mov si,0

14. input:

15. mov ah,01h



The screenshot shows the emulator interface with two windows. The left window is titled 'emulator screen (51x13 chars)' and displays the following text:  
 Enter user name: Tanvir  
 User name matched :)  
 Enter password: 123  
 Congratulations password matched :)

The right window is titled 'original source code' and shows the assembly code with some lines highlighted in orange:  
 26 printn  
 27 print "Enter password: "  
 28 lea si,pass  
 29 mov si,0  
 30 input1:  
 31 mov ah,01h  
 32 int 21h  
 33 cmp al,pass[si]  
 34 jne mismatch\_p  
 35 inc si  
 36 cmp si,3  
 37 jl input1  
 38  
 39 printn  
 40 print "Congratulations password matched :)"  
 41 jmp stp  
 42  
 43 mismatch\_u:  
 44 printn  
 45 print "User name is not matched! Please try again"  
 46 jmp stp  
 47  
 48 mismatch\_p:  
 49 printn  
 50 print "Password is not matched! Please try again"  
 51  
 52  
 53 stp:  
 54 ret  
 55

16. int 21h  
17. cmp al,username[si]  
18. jne missmatch\_u  
19. inc si  
20. cmp si,6  
21. jl input  
22.  
23. printn  
24. print "User name matched :)"  
25. printn  
26. printn  
27. print "Enter password: "  
28. lea si,pass  
29. mov si,0  
30. input1:  
31. mov ah,01h  
32. int 21h  
33. cmp al,pass[si]  
34. jne missmatch\_p  
35. inc si  
36. cmp si,3  
37. jl input1  
38.  
39. printn  
40. print "Congratulations password matched :)"  
41. jmp stp  
42.  
43. missmatch\_u:  
44. printn  
45. print "User name is not matched! Please try again :"  
46. jmp stp  
47.  
48. missmatch\_p:  
49. printn

50. print "Password is not matched! Please try again :!"

51.

52. stp:

53. ret

### Solve - Length of a string

1. include emu8086.inc

2. org 100h

3.

4. .data

5.

6. .code

7. mov ax,@data

8. mov ds,ax

9.

10. print "Enter a stirng under 9 number: "

11.

12. mov bx,0

13. input:

14. mov ah,01h

15. int 21h

16. cmp al,13

17. je output

18. inc bx

19. jmp input

20.

21. output:

22. printn

23. printn

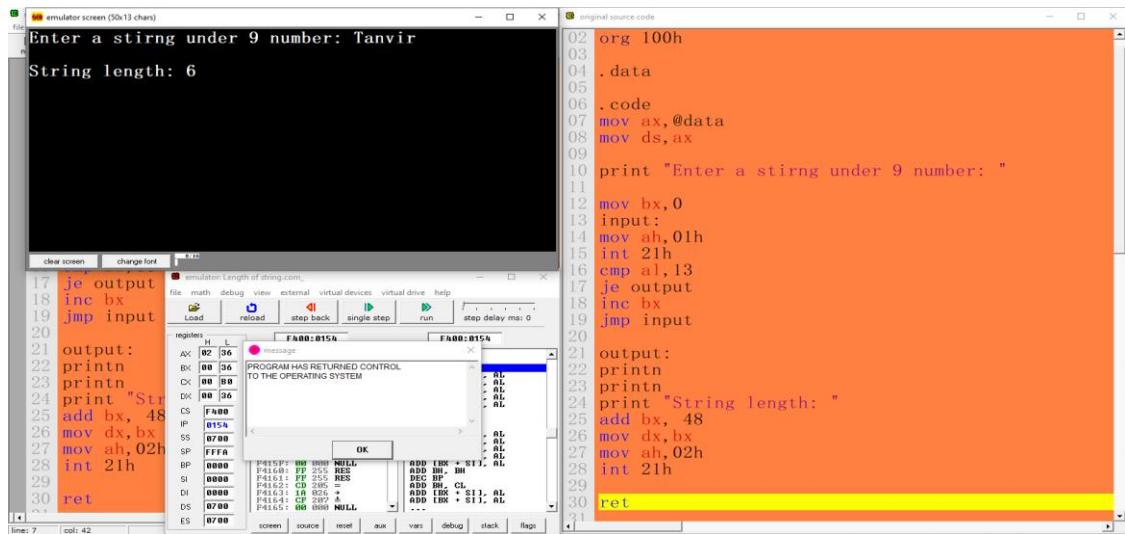
24. print "String length: "

25. add bx, 48

26. mov dx,bx

27. mov ah,02h

28. int 21h



29.

30. ret

## • Task 4

**Name:** ASCII value operations (Decimal to ASCII vice versa, Dynamic input, Upper case/lower case etc.)

**Solve -** Decimal to ASCII vice versa

1. include emu8086.inc

2. org 100h

3.

4. .data

5.

6. .code

7. mov ax,@data

8. mov ds,ax

9.

10. print "Enter a integer number:  
number: "

11. mov ah,01h

12. int 21h

13. sub al,48

14. printn

15. printn

16. print "The integer number is: "

17. mov dl,al

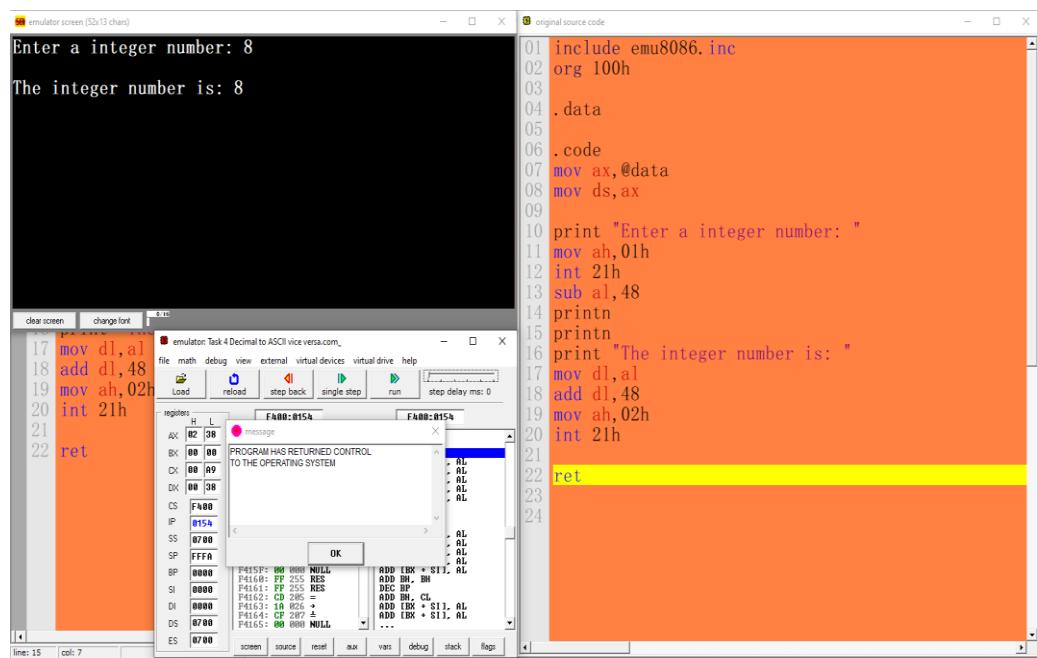
18. add dl,48

19. mov ah,02h

20. int 21h

21.

22. Ret



## Solve - Upper case/lower case

1. include emu8086.inc
2. org 100h
- 3.
4. .data
5. str1 db 10,13,"Enter a uppercase alphabet: \$"
6. str2 db 10,13,"Changed alphabet to lowercase: \$"
7. str3 db 10,13,"Enter a lowercase alphabet: \$"
8. str4 db 10,13,"Changed alphabet to uppercase: \$"
- 9.
10. .code
11. mov ax,@data
12. mov ds,ax
- 13.
14. lea dx,str1
15. mov ah,09h
16. int 21h
- 17.
18. mov ah,01h
19. int 21h
20. mov bl,al
- 21.
22. mov dx,offset str2
23. mov ah,09h
24. int 21h
- 25.
26. add bl,32
27. mov dl,bl
28. mov ah,02h
29. int 21h
- 30.
31. println
32. lea dx,str3
33. mov ah,09h
34. int 21h
- 35.
36. mov ah,01h
37. int 21h
38. mov bl,al
- 39.
40. mov dx,off
41. mov ah,09h
42. int 21h
- 43.
44. sub bl,32
45. mov dl,bl
46. mov ah,02h
47. int 21h
- 48.
49. ret

The screenshot shows the emulator interface with two windows. The left window displays the emulator screen (50x13 chars) with the following text:  
 Enter a uppercase alphabet: Z  
 Changed alphabet to lowercase: z  
 Enter a lowercase alphabet: k  
 Changed alphabet to uppercase: K

The right window shows the original source code in assembly language:

```

21 mov dx, offset str2
22 mov ah, 09h
23 int 21h
24
25 add bl, 32
26 mov dl, bl
27 mov ah, 02h
28 int 21h
29
30
31 println
32 lea dx, str3
33 mov ah, 09h
34 int 21h
35
36 mov ah, 01h
37 int 21h
38 mov bl, al
39
40 mov dx, offset str4
41 mov ah, 09h
42 int 21h
43
44 sub bl, 32
45 mov dl, bl
46 mov ah, 02h
47 int 21h
48
49 ret

```

A message dialog box is visible in the center of the screen, displaying "PROGRAM HAS RETURNED CONTROL TO THE OPERATING SYSTEM".

20. mov bl,al
- 21.
22. mov dx,offset str2
23. mov ah,09h
24. int 21h
- 25.
26. add bl,32
27. mov dl,bl
28. mov ah,02h
29. int 21h
- 30.
31. println
32. lea dx,str3
33. mov ah,09h

34. int 21h
- 35.
36. mov ah,01h
37. int 21h
38. mov bl,al
- 39.
40. mov dx,offset str4
41. mov ah,09h
42. int 21h
- 43.
44. sub bl,32
45. mov dl,bl
46. mov ah,02h
47. int 21h
- 48.
49. Ret

## • Task 5

**Name:** Stack operation (PUSH/POP, Reverse string etc.)

**Solve** – PUSH + POP with output Reverse string

1. include emu8086.inc

2. org 100h

3.

4. .data

5.

6. .code

7. mov ax,@data

8. mov ds,ax

9. mov ax,00

10.

11. print "Enter any  
string: "

12. mov cx,0

13. forl:

```

07 mov ax,@data
08 mov ds,ax
09 mov ax,00
10
11 print "Enter any string: "
12 mov cx,0
13 forl:
14 mov ah,01h
15 int 21h
16 cmp al,13
17 je output
18 mov bl,al
19 push bx
20 inc cx
21 jmp forl
22
23 output:
24 prints
25 prints
26 print "Reversed string is: "
27 for2:
28 pop dx
29 mov ah,02h
30 int 21h
31 loop for2
32
33 stp:
34
35 ret

```

The screenshot shows the emulator interface with the assembly code in the editor window. The code reads a string from the user, reverses it, and prints it back. The registers window shows the state of the CPU during execution. A message box is displayed with the text "PROGRAM HAS RETURNED CONTROL TO THE OPERATING SYSTEM".

```

14. mov ah,01h
15. int 21h
16. cmp al,13
17. je output
18. mov bl,al
19. push bx
20. inc cx
21. jmp for1
22.
23. output:
24. printn
25. printn
26. print "Reversed string is: "
27. for2:
28. pop dx
29. mov ah,02h
30. int 21h
31. loop for2
32.
33. stp:
34.
35. Ret

```

## • Task 6

**Name:** Arithmetic and logical operations (Double digit input/output etc.)

**Solve -** Double digit input/output

1. include emu8086.inc
2. org 100h
- 3.
4. .data
5. res db ?
6. rem db ?
- 7.

8. .code  
 9. mov ax,@data  
 10. mov ds,ax  
 11. mov bl,10  
 12. mov ah,00  
 13.  
 14. print "Enter a double digit number: "

15. mov ah,01h

16. int 21h

17. sub al,48

18. mul bl

19. mov cl,al

20.

21. mov ah,01h

22. int 21h

23. sub al,48

24. add al,cl

25. mov cl,al

26.

27. mov ax,cx

28. div bl

29. mov res,al

30. mov rem,ah

31.

32. println

33. println

34. print "The double digit number is: "

35. mov dl,res

36. add dl,48

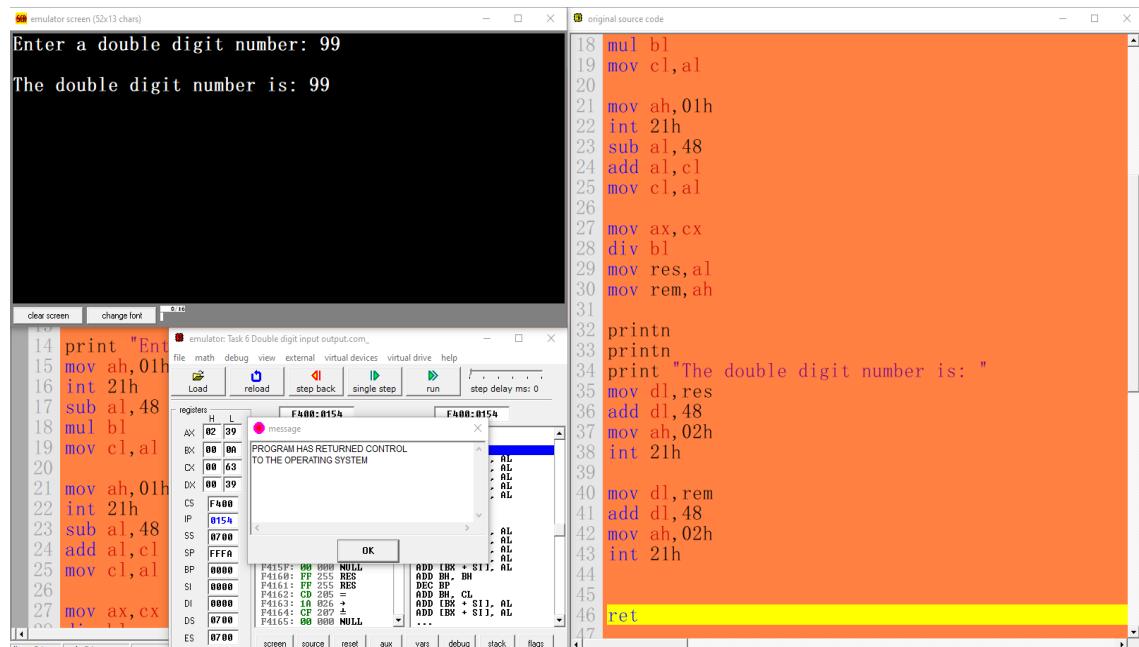
37. mov ah,02h

38. int 21h

39.

40. mov dl,rem

41. add dl,48



The screenshot shows a debugger interface with several windows:

- Registers Window:** Shows CPU registers (AX, BX, CX, DX, SI, DI, BP, SP, CS, DS, SS, IP, CS, DS, ES) with their values. A message box is displayed: "PROGRAM HAS RETURNED CONTROL TO THE OPERATING SYSTEM".
- Stack Window:** Displays memory dump from F400 to F4165, showing various assembly instructions like ADD, DEC, ADD, and CP.
- Code Window:** Shows the original source code in assembly language, with lines numbered 18 to 47.
- Output Window:** Displays the emulator screen output: "Enter a double digit number: 99" and "The double digit number is: 99".

42. mov ah,02h
43. int 21h
- 44.
- 45.
46. Ret

## • Task 7

**Name:** Array related operations (I/O array, comparison etc.)

**Solve** – Input + Output array, comparison with a number

1. include emu8086.inc

2. org 100h

3.

4. .data

5. arr3 db 20 dup(?)

6. var1 db ?

7.

8. .code

9. mov ax,@data

10. mov ds,ax

11.

12. print "Enter array input:"

13. lea si,arr3

14. mov si,0

15.

16. for1:

17. mov ah,01h

18. int 21h

19. cmp al,13

20. je next

21. sub al,48

22. mov arr3[si],al

23. inc si

24. jmp for1

```

37
38 mov cx,si
39 lea si,arr3
40 mov si,0
41 for2:
42 mov dl,arr3[si]
43 cmp dl,var1
44 je found
45 inc si
46 cmp si,cx
47 jl for2
48
49 not_found:
50 printn
51 print "Not Found :)"
52 jmp stp
53
54 invalid:
55 printn
56 print "Invalid input!"
57 jmp stp
58
59 found:
60 printn
61 print "Found :)"
62
63 stp:
64
65 ret
66

```

25.  
26. next:  
27. printn  
28. printn  
29. print "Enter what you want to find: "  
30. mov ah,01h  
31. int 21h  
32. cmp al,13  
33. je invalid  
34. sub al,48  
35. mov var1,al  
36. printn  
37.  
38. mov cx,si  
39. lea si,arr3  
40. mov si,0  
41. for2:  
42. mov dl,arr3[si]  
43. cmp dl,var1  
44. je found  
45. inc si  
46. cmp si,cx  
47. jl for2  
48.  
49. not\_found:  
50. printn  
51. print "Not Found :!"  
52. jmp stp  
53.  
54. invalid:  
55. printn  
56. print "Invaid input!"  
57. jmp stp  
58.

59. found:
60. printn
61. print "Found :)"
- 62.
63. stp:
- 64.
65. Ret

**The End.**