UNIVERSITY OF INFORMATION TECHNOLOGY AND SCIENCES

Assignment on

Lab Report- 01

Course Title

Microprocessor and MicroControllers

Course Code

CSE 360

Submitted by

KM Jakaria

Section: 6A

Batch: 53

ID: 0432310005101037

Submitted to

Md. Ismail

Lecturer

Department of CSE

UITS

Date of Submission

27 Aug, 2025

Problem No: 01

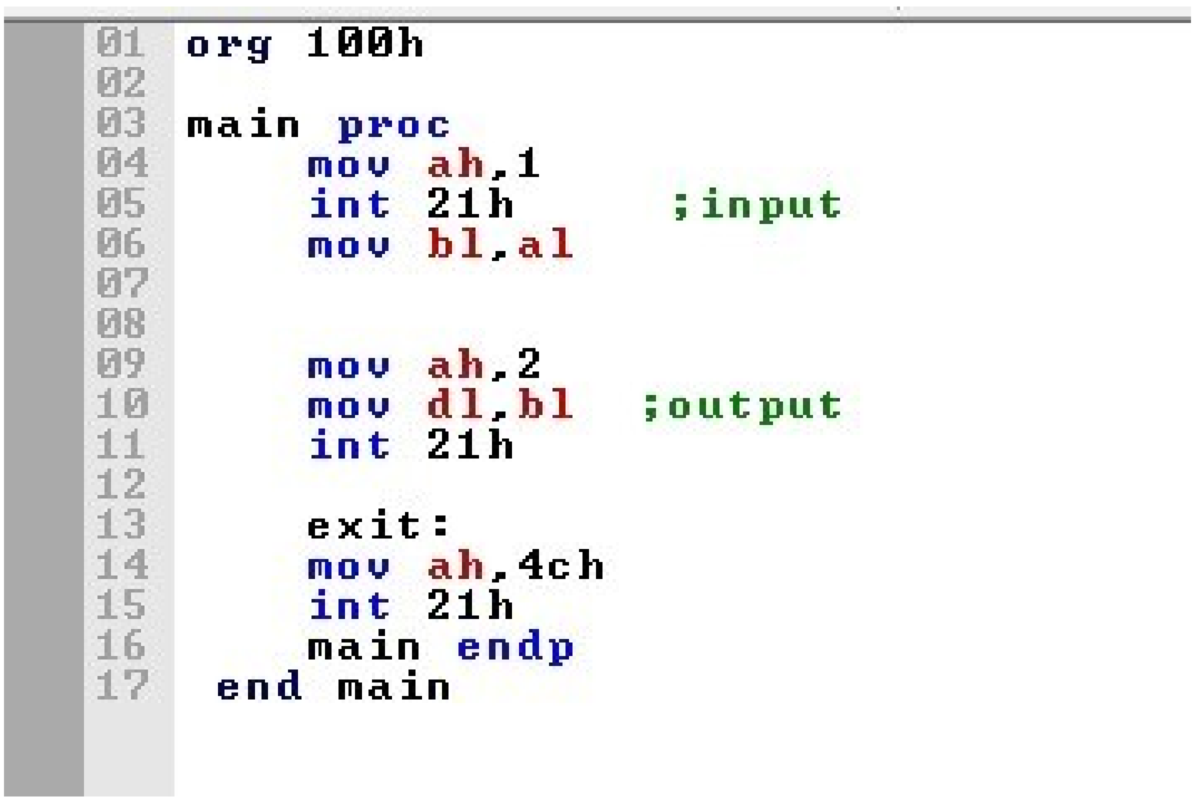
Experiment No: 01

Experiment Name: Taking input from the user and printing as output on the same line.

Process:

Interrupt service procedure AH = 01H is used for user input; it receives a character from standard input and echoes it to standard output. The AL register contains the input character. After then, this value is transferred to BL for short-term storage. The character in the DL register is written to standard output using an interrupt service function with AH = 02H.

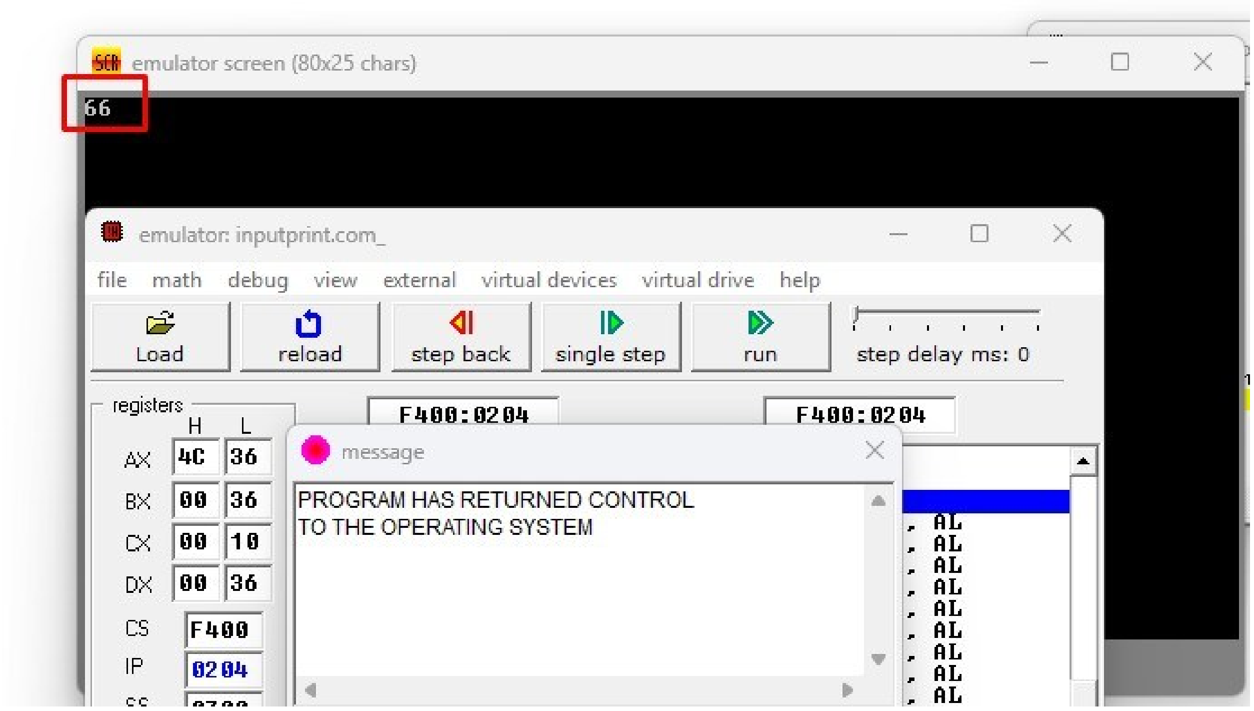
Implementation:



```
org 100h

main proc
    mov ah,1
    int 21h       ;input
    mov bl,al


    mov ah,2
    mov dl,bl   ;output
    int 21h

    exit:
    mov ah,4ch
    int 21h
    main endp
 end main
```

Result:

When the user enters "6", the program shows "6" once more, creating the visual "6 6" (the second '6" is from our output, and the first '6' is from the input echo). When the user enters "6", the program shows "6" once more, creating the visual "6 6" (the second '6" is from our output, and the first '6' is from the input echo).


emulator screen (80x25 chars)
emulator: inputprint.com_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
message

```
66
```

```
AX 4C 36
BX 00 36
CX 00 10
DX 00 36
CS F400
IP 0204
F400:0204
```

```
PROGRAM HAS RETURNED CONTROL
TO THE OPERATING SYSTEM
```

Conclusion:

The application effectively illustrates 8086 assembly language input and output processes. A simple way for character input and output is provided by the INT 21H interrupt with the proper function codes in the AH register. We see the input character twice—once from the explicit output instruction and once from the automatic echo—because when function 01H is used for input, the character is automatically echoed to the output. Our comprehension of fundamental I/O operations in 8086 assembly programming is validated by this experiment.

Problem No: 02

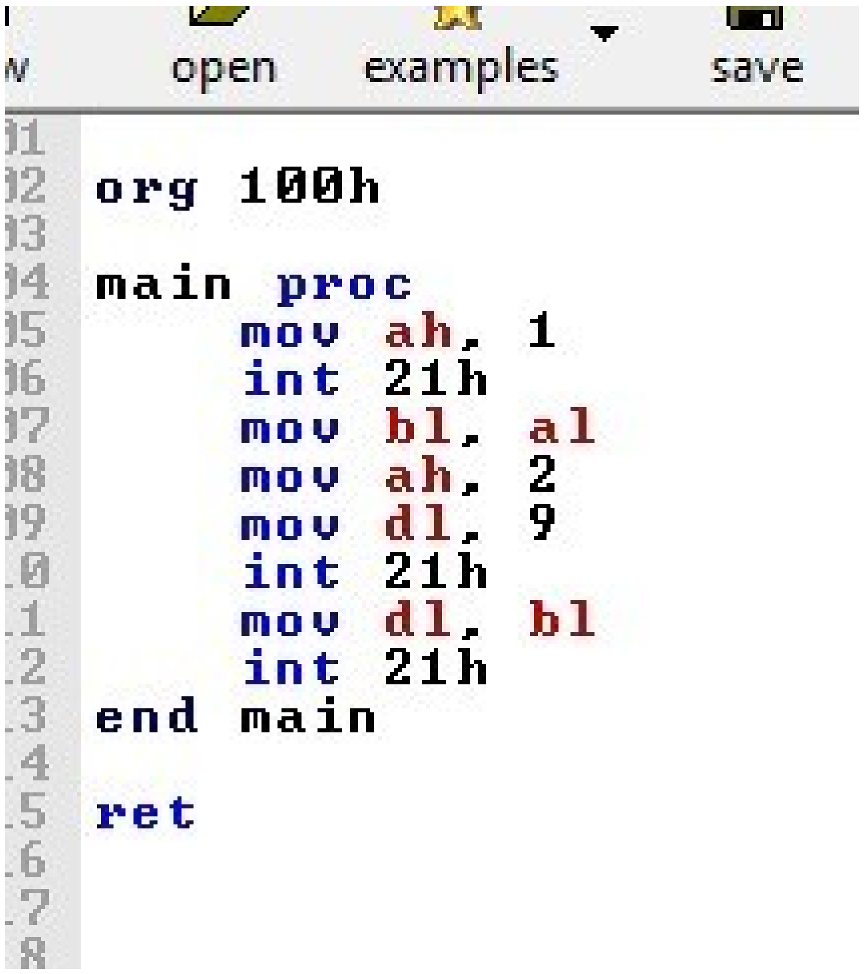
Experiment No: 02

Experiment Name: Taking input from user and printing as an output at the same line with tab

Process:

Interrupt service procedure AH = 01H is used for user input; it receives a character from standard input and echoes it to standard output. The AL register contains the input character. After then, this value is transferred to BL for short-term storage. Display the input character stored in BL after first displaying a tab character (ASCII 9) using an interrupt service function with AH = 02H.

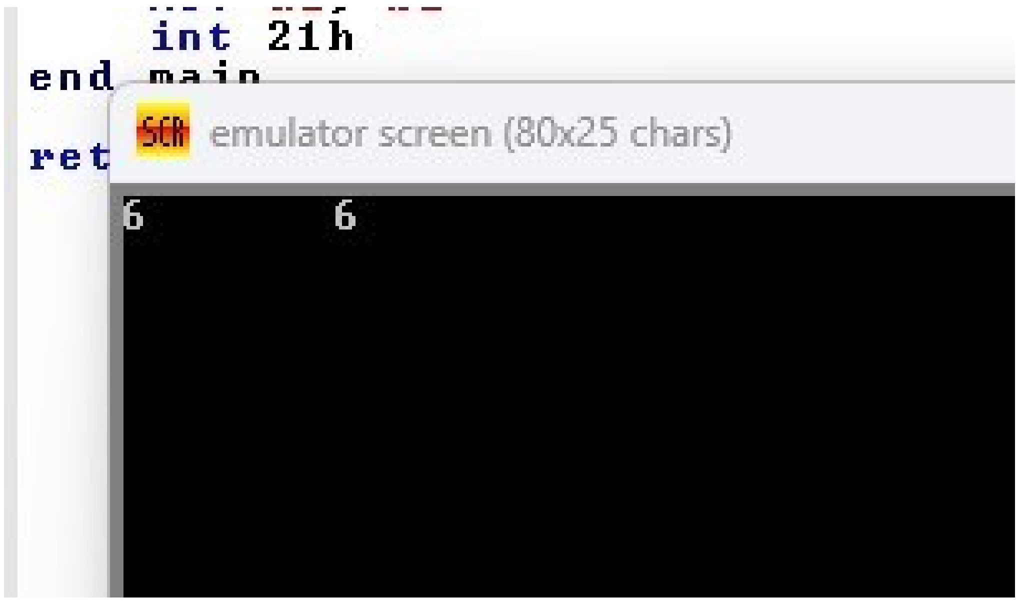
Implementation:



```
org 100h

main proc
     mov ah, 1
     int 21h
     mov bl, al
     mov ah, 2
     mov dl, 9
     int 21h
     mov dl, bl
     int 21h
end main

ret
```

Result:

When '6' is entered by the user, the application shows a tab and then '6', which makes "6" appear on the screen (where the space is a tab character). The second '6' following the tab comes from our explicit output instruction, while the first '6' comes from the input echo.



```
        int 21h
end main

ret
```

```
6       6
```

Conclusion:

In 8086 assembly code, the program effectively illustrates input and output processes with formatting. We can produce output that is correctly structured by adding a tab character between the automatically repeated input and our explicit output. This experiment strengthens the fundamental I/O operations we previously learnt by confirming our comprehension of how to format output in assembly language programming using ASCII control characters (such as tab).

Problem No: 03

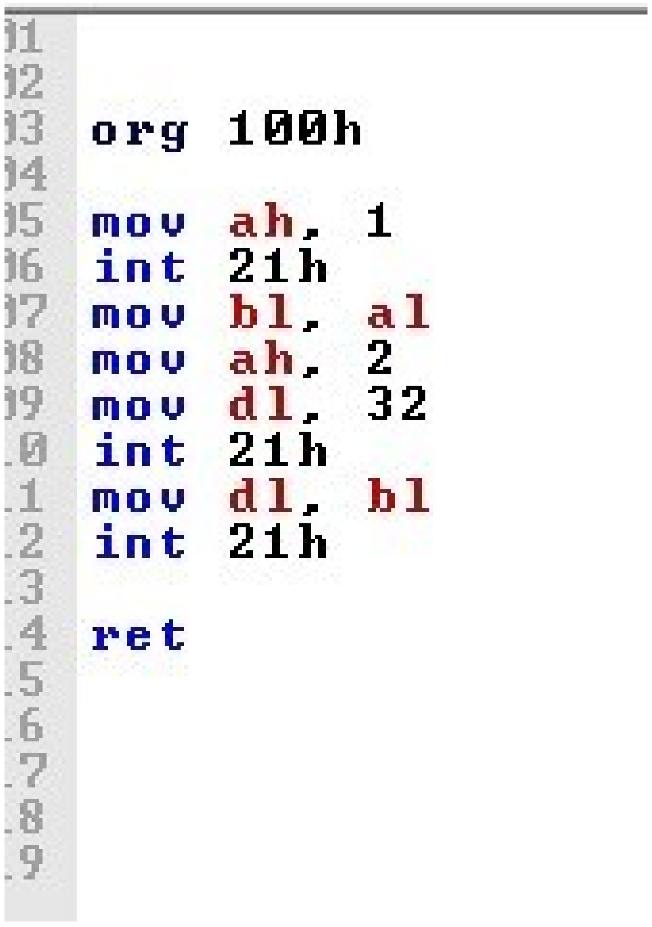
Experiment No: 03

Experiment Name: Input and Output Operations with Space Formatting in 8086 Assembly

Process:

Interrupt service procedure AH = 01H is used for user input; it receives a character from standard input and echoes it to standard output. The AL register contains the input character. After then, this value is transferred to BL for short-term storage. In order to display the input character stored in BL, first use an interrupt service procedure with AH = 02H to display a space character (ASCII 32).

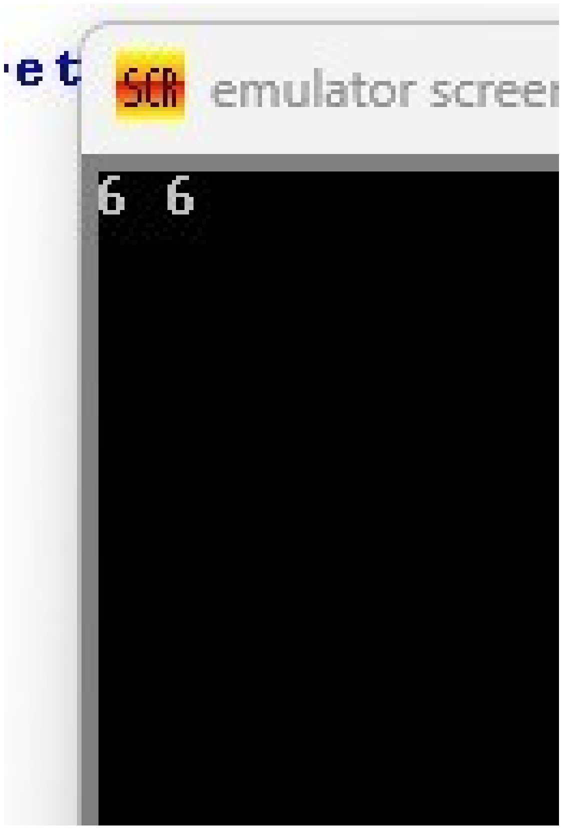
Implementation:



```


org 100h

mov ah, 1
int 21h
mov bl, al
mov ah, 2
mov dl, 32
int 21h
mov dl, bl
int 21h

ret
```

Result:

Inputting '6' causes the application to display a space after '6', which makes the screen read "6 6". Our explicit output command is responsible for the second '6' that follows the space, whereas the first '6' comes from the input echo. Inputting '6' causes the application to display a space after '6', which makes the screen read "6 6". Our explicit output command is responsible for the second '6' that follows the space, whereas the first '6' comes from the input echo.



```
6 6
```

Conclusion:

In 8086 assembly code, the program effectively illustrates input and output procedures with space formatting. We can produce output that is correctly formatted and clearly separated by adding a space character between the automatically echoed input and our explicit output. Building on the fundamental I/O operations we previously studied, this experiment validates our comprehension of how to format output in assembly language programming using ASCII characters. In contrast to the tab character, which might vary in width based on console settings, the space character offers a more straightforward and reliable separation.

Problem No: 04

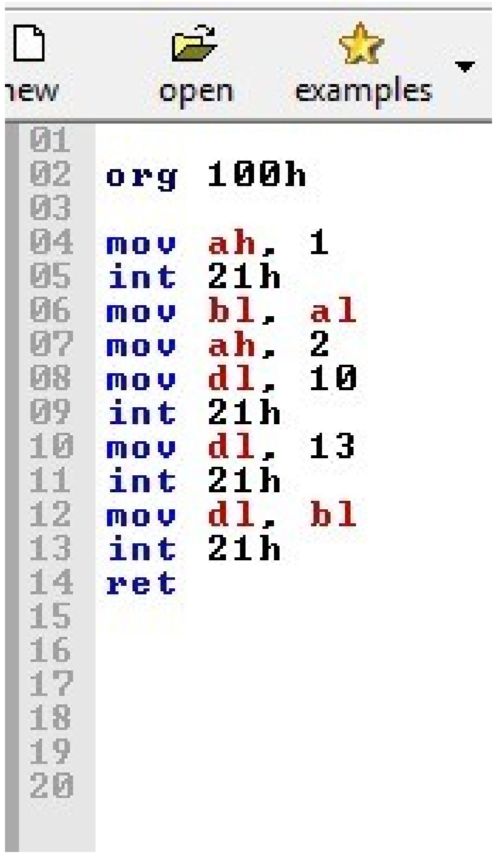
Experiment No: 04

Experiment Name: Taking input from user and printing as a output at the new line

Process:

Interrupt service procedure AH = 01H is used for user input; it receives a character from standard input and echoes it to standard output. The AL register contains the input character. After then, this value is transferred to BL for short-term storage. For output, display the input character stored in BL after first displaying a carriage return (ASCII 13) and line feed (ASCII 10) using an interrupt service routine with AH = 02H to go to a new line.

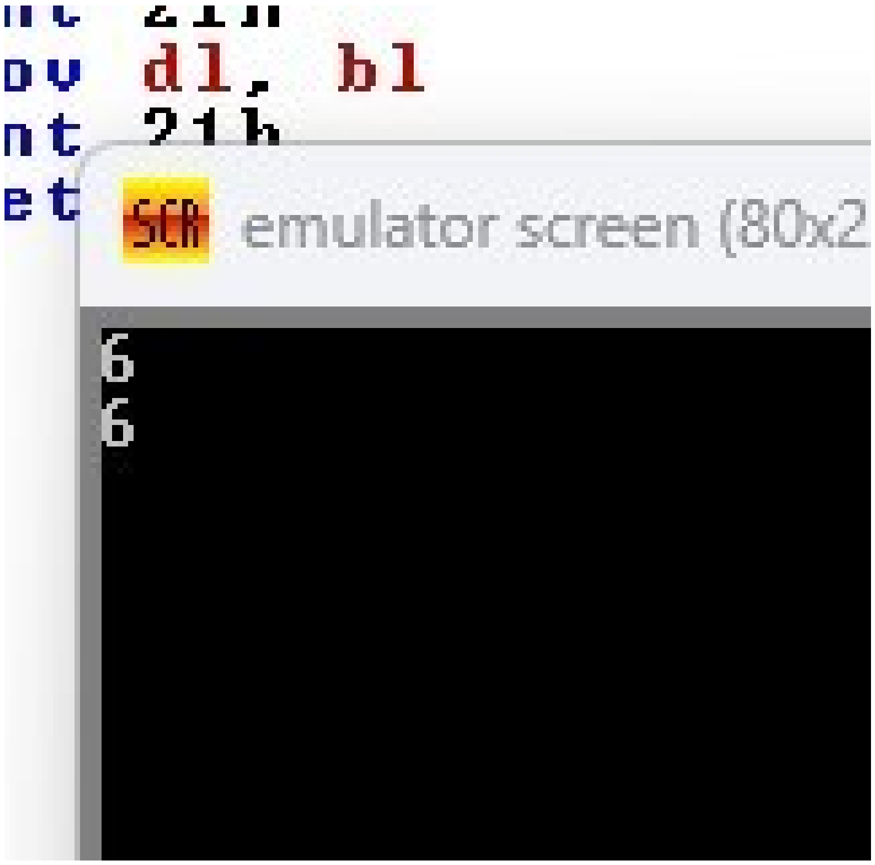
Implementation:



```

org 100h

mov ah, 1
int 21h
mov bl, al
mov ah, 2
mov dl, 10
int 21h
mov dl, 13
int 21h
mov dl, bl
int 21h
ret
```

Result:

1. Shows '6' from the input echo on the first line when the user enters it
2. Goes to a different line
3. Returns to '6' from our explicit output command on the second line.



```
ov  dl, bl
nt  21h
et
```

```
6
6
```

Conclusion:

In 8086 assembly code, the program effectively illustrates input and output procedures using new line formatting. We can construct output on a new line by adding carriage return (ASCII 13) and line feed (ASCII 10) characters between the automatically repeated input and our explicit output. This exercise demonstrates that we understand how to employ ASCII control characters for multi-line output and other sophisticated formatting in assembly language programming. For new lines to function properly in DOS settings, carriage return and line feed must be combined.