



International Islamic University Chittagong (IIUC)

Computer and Communication Engineering (CCE)

INDEX

Exp. No	Experiment Name	Page No.	Lab Date	Submission Date	Remarks
1	Brief Introduction to EMU8086 and how to install it.	3-5	18/10/2021	13/12/2021	
2	Write a program called 'hello world' using emu8086.	6-9	18/10/2021	13/12/2021	
3	Understanding Syntax structure of assembly language using emu8086	10-11	19/10/2021	13/12/2021	
4	A program to understand basic input and output using assembly language.	12-13	13/11/2021	13/12/2021	
5	Printing the student credentials using assembly language.	14-18	03/11/2021	13/12/2021	
6	Use of loop in assembly language using emu8086 (must print number [0-9] and alphabet [A-Z	19-22	10/11/2021	13/12/2021	
7	Drawing multiple multi-color line using emu8086 graphical mode	23-26	10/12/2021	13/12/2021	



আন্তর্জাতিক ইসলামী বিশ্ববিদ্যালয় চট্টগ্রাম
الجامعة الإسلامية العالمية شيتاغونغ
International Islamic University Chittagong

Department of Computer and Communication Engineering

LAB REPORT

Course Name: Microprocessor and Assembly Language Sessional

Course Code: CCE-3502

Submitted By

Name: Arif Hasnat

ID No: E191034

Semester: 5th

Section: A

Email: ahparif.16@gmail.com

Date of Experiment: 10/12/2021

Date of Submission: 13/12/2021

Submitted To

Md. Imanul Huq

Adjunct Faculty, Dept. of ETE, IIUC

<u>Remarks</u>

Experiment No:01

Experiment_Name: Brief Introduction to EMUBO86 and how to install it.

Description:

EMUS086 - MICROPROCESSOR EMULATOR is a free emulator for multiple platforms. It provides its user with the ability to emulate old 8086 processors, which were used in Macintosh and Windows computers from the 1980s and early 1990s. It can emulate a large amount of software that was used on these microprocessors; we can also program their own assembly code to run on it.

EMUS086 - MICROPROCESSOR EMULATOR primarily emulates the processor, not the other functions that a microcomputer running an 8086 processor would have. However, it still serves many of the same functions that an emulator for a more specific microcomputer might have, and more besides. For example, both the NEC-P9801 and early IBM-compatible computers used the 8086. Using EMU8086, one might be able to write assembly software that can run on either of those devices. On the flip side, EMUS086 cannot access some of the more advanced hardware functionality that you might find in the monitors or other components of those devices.

Overall, EMUS086 - MICROPROCESSOR EMULATOR will be useful to computing enthusiasts and gear heads, and anyone who happens to work with this legacy processor even today: some computers, particularly in business and industrial applications, still use the 8086.

Installation_Procedure:

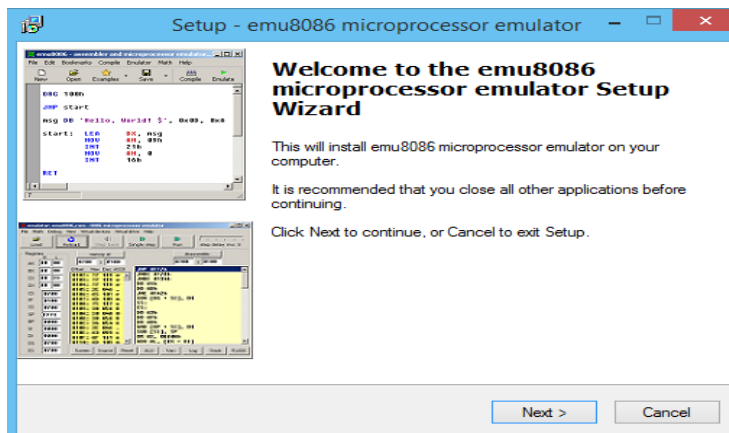


Fig 1: Downloaded file extracted and setup has been initiated

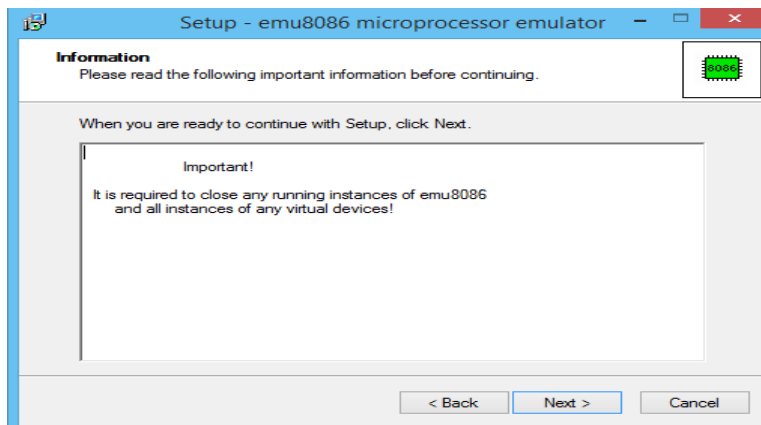


Fig 2: Setup procedure

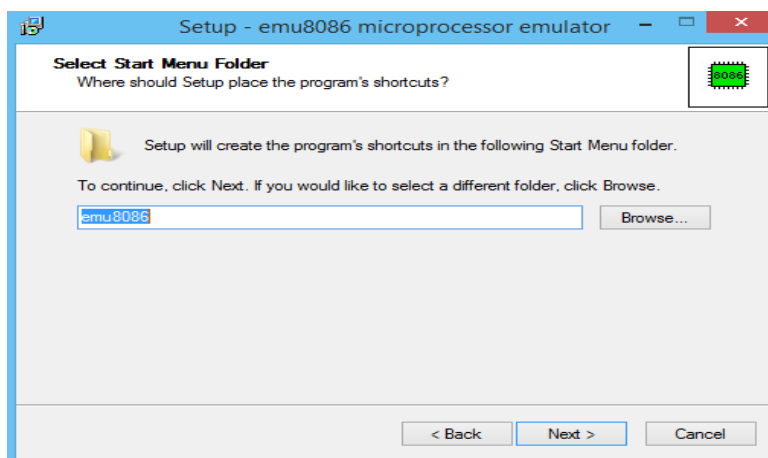


Fig 3: Installation procedure

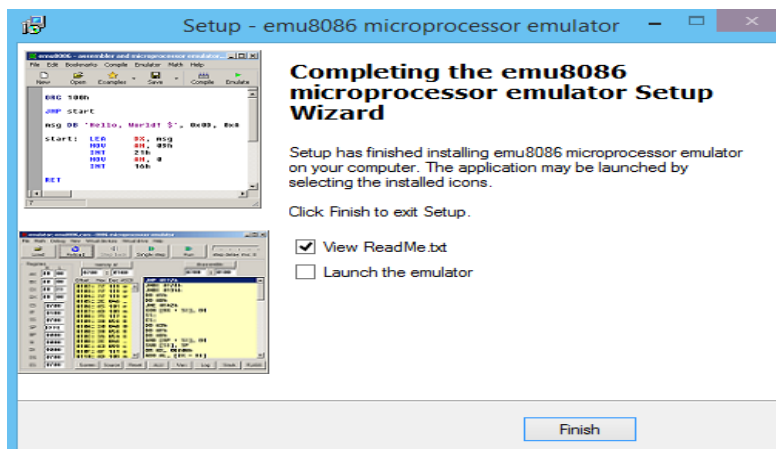


Fig 4: Installation Complete

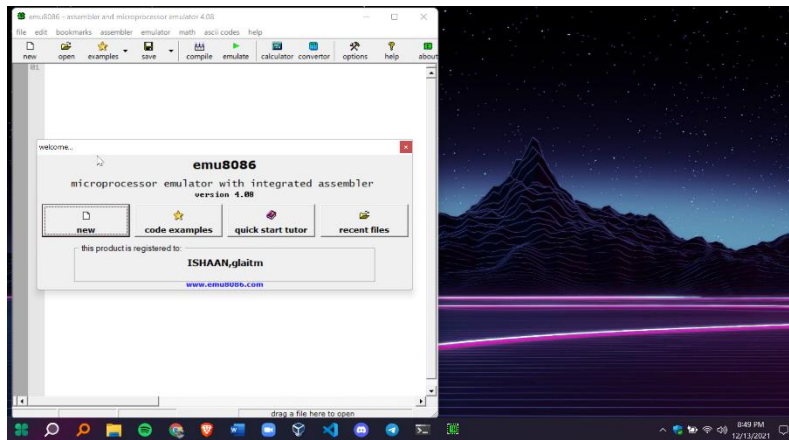


Fig 5: Installation complete and successfully running Software.

Discussion:

From the experiment we have learned about how to install Emu8086 Software, basic knowledge about this software and how to set up the Emu8086 and run the software properly. After the experiment we can run assembly code on this installed software.

Experiment No: 02

Experiment Name: Write a program called “hello world” using emu8086.

Required Equipment:

1. A computer.
2. Emu8086 software.

Description:

An example Assembly program explained so that we can understand the very basic terminology before you write more complex Assembly Applications. First Assembly program simply prints a text message “Hello World” on Screen. In this Assembly Language Programming. A single program is divided into four Segments which are 1. Data Segment, 2. Code Segment, 3. Stack Segment, and 4. Extra segment. Now, from these one is compulsory i.e. Code Segment if at all we do not need variable(s) for our program. if we need variable(s) for our program we will need two Segments i.e. Code Segment and Data Segment.

Source code:

```
name "hi-world"

; this example prints out "hello world!"

; by writing directly to video memory.

; in vga memory: first byte is ascii character, byte that follows is character attribute.

; if you change the second byte, you can change the color of

; the character even after it is printed.

; character attribute is 8 bit value,

; high 4 bits set background color and low 4 bits set foreground color.


; hex   bin    color
;
; 0     0000    black
; 1     0001    blue
; 2     0010    green
; 3     0011    cyan
; 4     0100    red
```

```

; 5    0101    magenta
; 6    0110    brown
; 7    0111    light gray
; 8    1000    dark gray
; 9    1001    light blue
; a    1010    light green
; b    1011    light cyan
; c    1100    light red
; d    1101    light magenta
; e    1110    yellow
; f    1111    white

```

```
org 100h
```

```
; set video mode
```

```
mov ax, 3    ; text mode 80x25, 16 colors, 8 pages (ah=0, al=3)
```

```
int 10h    ; do it!
```

```
; cancel blinking and enable all 16 colors:
```

```
mov ax, 1003h
```

```
mov bx, 0
```

```
int 10h
```

```
; set segment register:
```

```
mov ax, 0b800h
```

```
mov ds, ax
```

```
; print "hello world"
```

```
; first byte is ascii code, second byte is color code.
```

```
mov [02h], 'H'
```

mov [04h], 'e'

mov [06h], 'l'

mov [08h], 'l'

mov [0ah], 'o'

mov [0ch], ','

mov [0eh], 'W'

mov [10h], 'o'

mov [12h], 'r'

mov [14h], 'l'

mov [16h], 'd'

mov [18h], 'l'

; color all characters:

mov cx, 12 ; number of characters.

mov di, 03h ; start from byte after 'h'

c: mov [di], 11101100b ; light red(1100) on yellow(1110)

add di, 2 ; skip over next ascii code in vga memory.

loop c

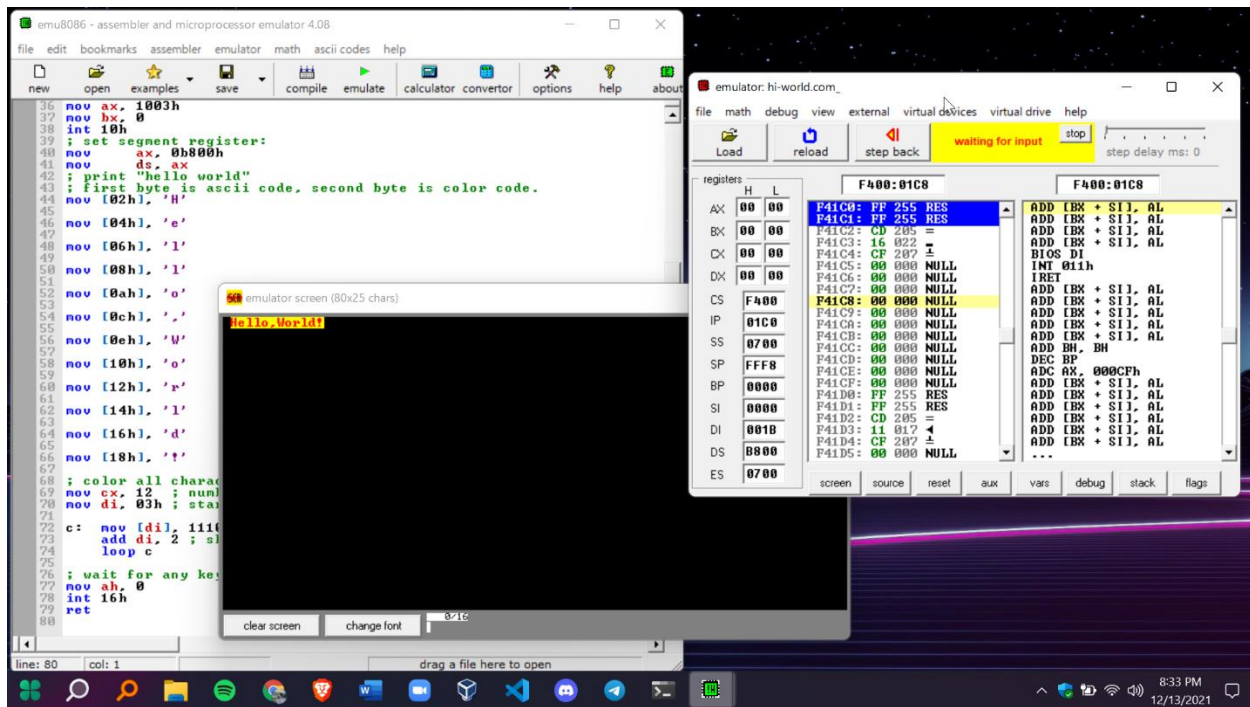
; wait for any key press:

mov ah, 0

int 16h

ret

Result / Screenshot:



Discussion:

From this experiment we have run our first code and executed the “Hello world” program. We have known how to change the color in the output result. In the result section we have shown the screenshot. For we must be compiled the written program and Run by clicking on the RUN button on the top. Then the output result will pop up with no error.

Experiment No:03

Experiment Name: Understanding syntax structure of assembly language using emu8086.

Required Equipment:

1. A computer
2. Emu8086 software

DESCRIPTION:

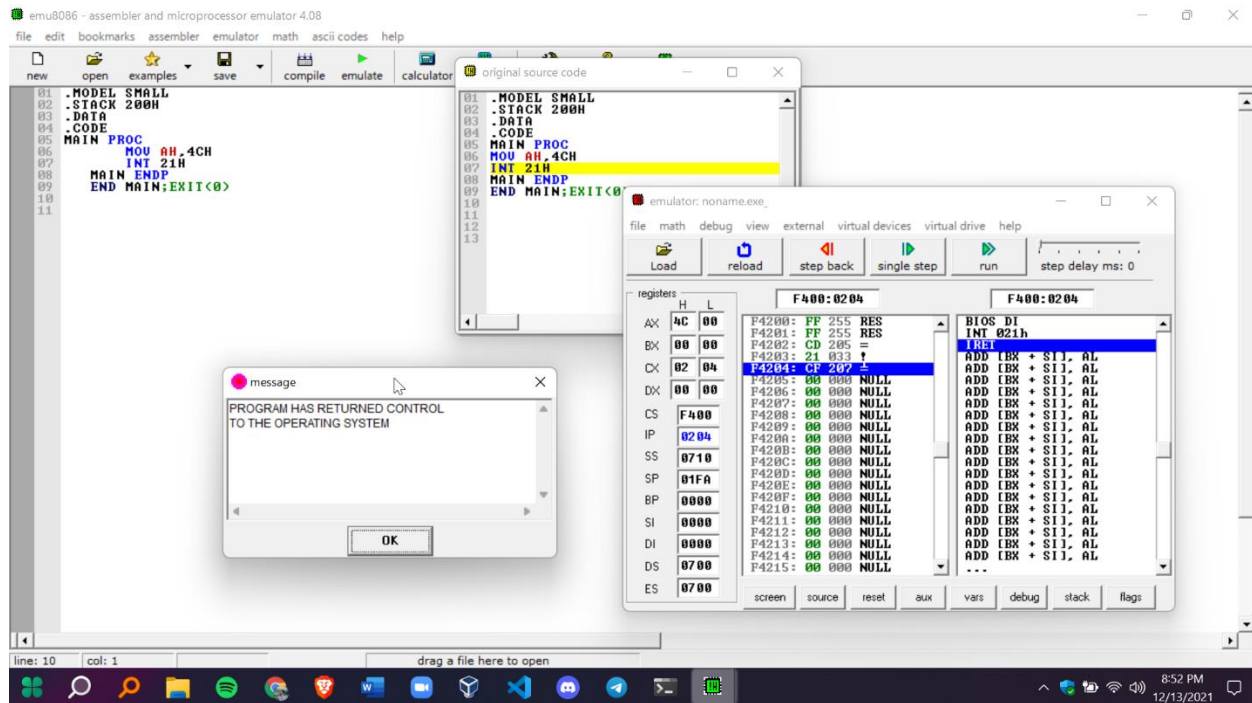
Assembly language is a low-level programming language. We need to get some knowledge about computer structure to understand anything. Basic syntax has discussed with example and we implemented the solution of some small problems. I/O DOS function calls: the main I/O functions. These functions are mainly used to read a character or a string from the keyboard, which could be an input data to a program, and display characters or strings, which could be results, or an output, of a program.

Source Code:

Code 1:

```
.MODEL SMALL  
.STACK 200H  
.DATA  
.CODE  
MAIN PROC  
    MOV AH,4CH  
    INT 21H  
MAIN ENDP  
END MAIN;EXIT(0)
```

Result / Screenshot:



DISCUSSION: In this experiment we have learned about the syntax structure of assembly language using emu8086. Syntax 1) MOV mem/reg1, mem/reg2 MOV instruction copies the second operand(source) to the first operand(destination). The source operand can be an immediate value, general-purpose register, or memory location. The destination register can be a general-purpose register, or memory location. Both operands must be same size, which can be a byte or a word.

Experiment No: 04

Experiment Name: A program to understand basic input and output using assembly language.

Required Equipment:

1. A computer
2. Emu8086 software

Description:

For this experiment first, we make the basic structure for the program. For that first we must write the code on MAIN PROC. For taking a keystroke first we must keep the value 1 in ah register. Then we have to call the int 21h for taking the value from keyboard. It will keep the value in al register, but we cannot keep it here as it is unsafe. So, we will take the value from al to bl. Then for printing the value we have to keep the value 2 in ah register and call the int 21h. This will print the value have on dl so we have to move the value of bl to dl. As it will print the value in same line, we have to put the value 0ah on dl. It will print the value in new line with backspace. To remove backspace, we have put the value 08h on dl. Then it will print the value without backspace in new line.

Source Code:

```
.MODEL SMALL
```

```
.STACK 200H
```

```
.DATA
```

```
.CODE
```

```
MAIN PROC
```

```
mov ah,1
```

```
Int 21h
```

```
mov bl,al
```

```
mov ah,2
```

```
mov dl,0ah
```

```
Int 21h
```

```

mov ah,2
mov dl,bl

int 21h

exit:

mov ah,4ch

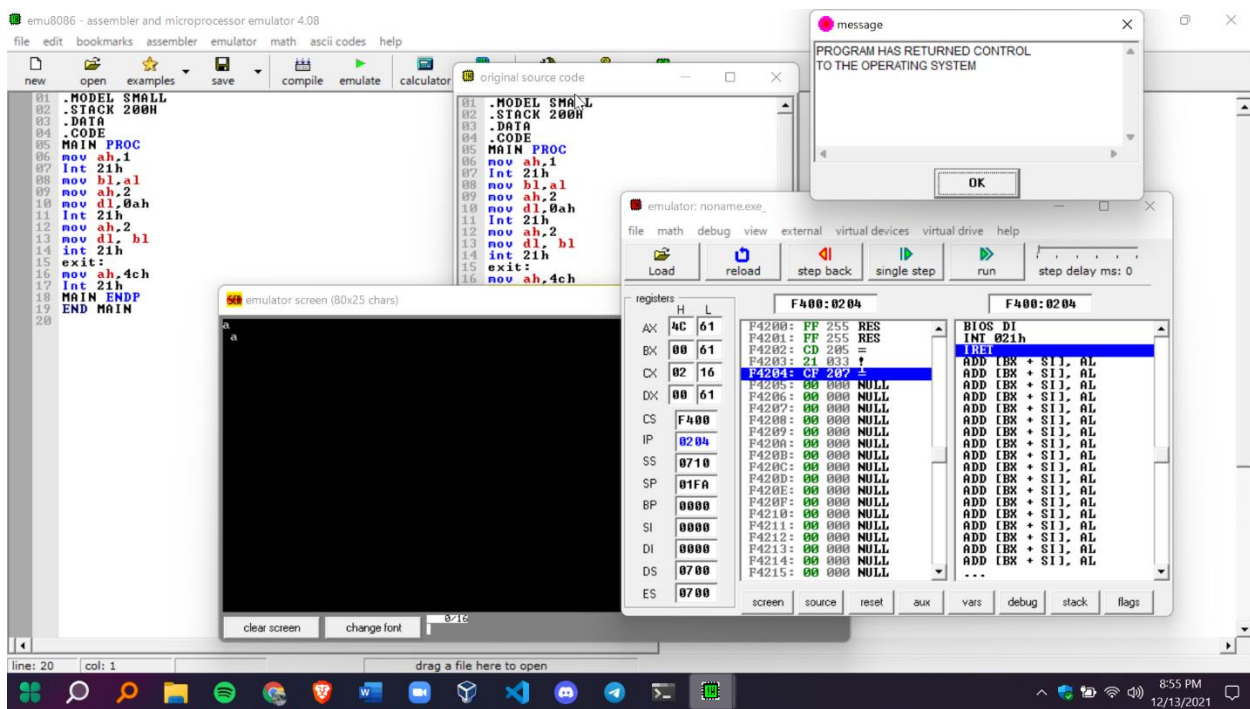
Int 21h

MAIN ENDP

END MAIN

```

Result / Screenshot:



Discussion: in this experiment we have learned about the basic input and output by using the emu8086 software. When the input given the same output has shown in the output screen. In / out is a crucial part of most programs since few programs operate in isolation from the outside world. At least the results of the program must be output, either to a person or for another program or computer. Input / output is basically asynchronous.

Experiment No: 05

Experiment Name: Printing the student credentials using assembly language.

Required equipment:

1. A computer.
2. Emu8086 software

Description:

For this experiment first, we must make the basic structure for the program. For that first we must declare the code segment. After that we must write the code on MAIN PROC. For printing the value, we have to keep the value 2 in ah register and call the int 21h. This will print the value have on dl so we have to give the value of our credential on dl. As it will print the value in same line, we must put the value 0ah on dl. it will print the value in new line with backspace. To remove backspace, we have put the value 08h on dl. Then it will print the value without backspace in new line.

Source code:

```
.MODEL SMALL
```

```
.STACK 203H
```

```
.DATA
```

```
.CODE
```

```
MAIN PROC
```

```
;For printing the Name
```

```
mov ah,2
```

```
mov dl,"A"
```

```
int 21h
```

```
mov ah,2
```

```
mov dl,"r"
```

```
int 21h
```

```
mov ah,2
```

```
mov dl,"i"
```

```
int 21h
```

```
mov ah,2  
mov dl,"f"  
int 21h  
mov ah,2
```

```
mov dl," "  
int 21h  
mov ah,2
```

```
mov dl,"H"  
int 21h  
mov ah,2  
mov dl,"a"  
int 21h  
mov ah,2
```

```
mov dl,"s"  
int 21h  
mov ah,2
```

```
mov dl,"n"  
int 21h  
mov ah,2
```

```
mov dl,"a"  
int 21h  
mov ah,2
```

```
mov dl,"t"
```

```
int 21h
```

```
mov ah,2
```

```
mov dl, " "
```

```
int 21h
```

```
mov ah,2
```

```
mov dl," "
```

```
int 21h
```

```
mov ah,2
```

```
mov dl,"E"
```

```
Int 21h
```

```
mov ah,2
```

```
mov dl, "1"
```

```
int 21h
```

```
mov ah,2
```

```
mov dl, "9"
```

```
int 21h
```

```
mov ah,2
```

```
mov dl,"1"
```

```
int 21h
```

```
mov ah,2
```

```
mov dl, "0"
```



```
int 21h
```

```
mov ah,2
```

```
mov dl, "3"
```

```
int 21h
```

```
mov ah,2
```

```
mov dl, "4"
```

```
int 21h
```

```
mov ah,2
```

```
;For taking a new line with backspace
```

```
mov ah,2
```

```
mov dl,0ah
```

```
int 21h
```

```
;For removing backspace
```

```
mov ah,2
```

```
mov dl, 08h
```

```
int 21h
```

```
mov ah,2
```

```
mov dl,08h
```

```
int 21h
```

```
mov ah,2
```

```
mov dl,08h
```

```
int 21
```

```
mov ah,2
```

```
mov dl, 08h
```

int 21

mov ah,2

mov dl, 08h

int 21

;For printing Id

mov ah,2

mov dl, "T"

int 21

mov ah,2

mov dl, "."

int 21

mov ah,2

mov dl, "1"

int 21

mov ah,2

mov dl, "4"

int 21

mov ah,2

mov dl, "3"

int 21

mov ah,2

mov dl, "0"

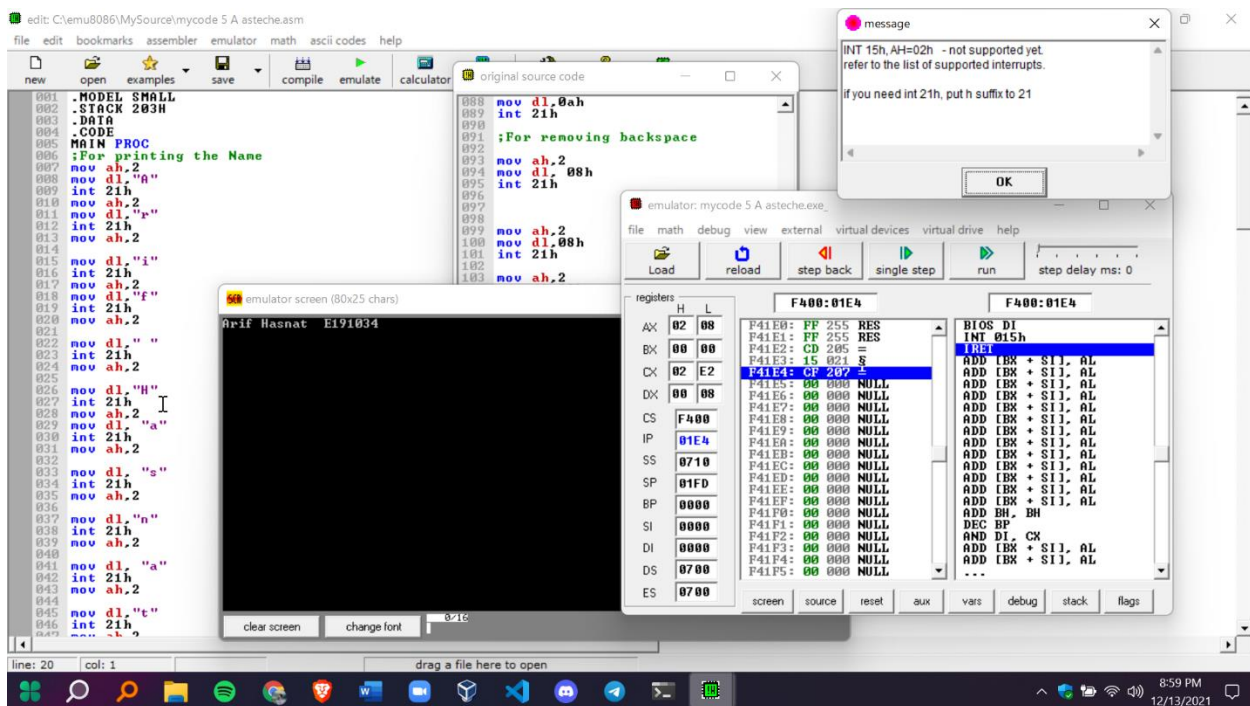
int 21

```

mov ah,2
mov dl, "1"
int 21
mov ah,2
mov dl, "8"
int 21
exit:
mov ah, 4ch
int 21h
MAIN ENDP
END MAIN

```

Result/Screenshot:



Discussion: In this experiment we have learned about printing the credentials using the emu8086 software. We used the MOV instruction that is used for moving data from one storage space to another. The MOV instruction takes two operands. We had to keep in mind for this program that both the operands in MOV operation should be of same size. And the value of source operand remains unchanged.

Experiment No: 06

Experiment Name: Use of loop in assembly language using emu8086 (must print number [0-9] and alphabet [A-z])

Required Equipment:

1. A Computer
2. Emu8086 software

Description: For this experiment first, we must make the basic structure for the program. For that first we must declare the code segment. Input – output instructions. Print upper case A to Z – while loop simulation Print upper case A to Z – Do while loop simulation Read letters display in next in sequence. Although the loop instruction's name suggests that we would normally create loops with it, keep in mind that all it is really doing is decrementing CX and branching to the target address if CX does not contain zero after the decrement. We can use this instruction anywhere you want to decrement CX and then check for a zero result, not just when creating loops. Nonetheless, it is a very convenient instruction to use if we simply want to repeat a sequence of instructions some number of times.

Source code:

1. For Number [0-9]

```
.MODEL SMALL
```

```
.STACK 10H
```

```
.CODE
```

```
main proc
```

```
mov cl, 10
```

```
mov dl, 47
```

```
Bangladesh:
```

```
inc dl
```

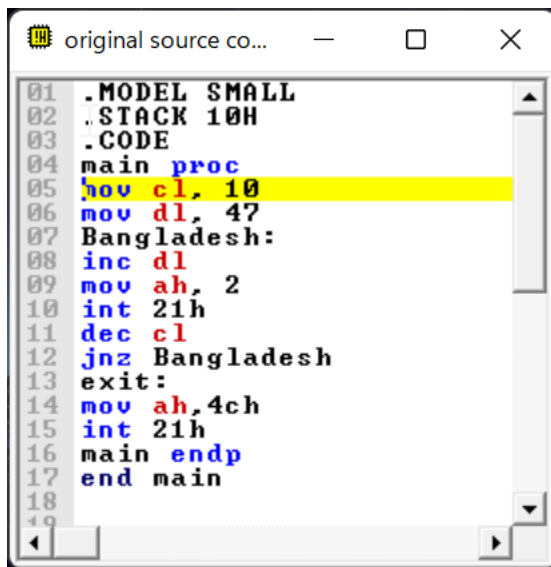
```
mov ah, 2
```

```
int 21h
```

```
dec cl
```

```
jnz Bangladesh
```

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



```
01 .MODEL SMALL
02 .STACK 10H
03 .CODE
04 main proc
05 mov cl, 10
06 mov dl, 47
07 Bangladesh:
08 inc dl
09 mov ah, 2
10 int 21h
11 dec cl
12 jnz Bangladesh
13 exit:
14 mov ah, 4ch
15 int 21h
16 main endp
17 end main
18
19
```

Fig 1: source code (0-9)

2. For Alphabet [a-z]

```
.MODEL SMALL
.STACK 10H
.CODE
main proc
mov cl, 26
mov dl, 64
    Bangladesh:
inc dl
mov ah, 2
int 21h
dec cl
```

jnz Bangladesh

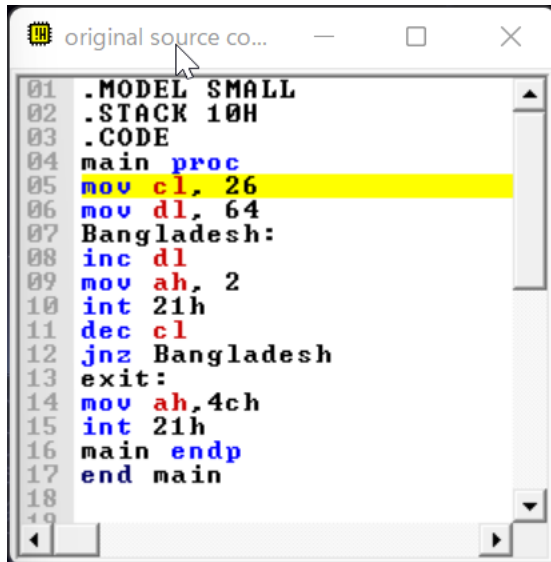
exit:

mov ah,4ch

int 21h

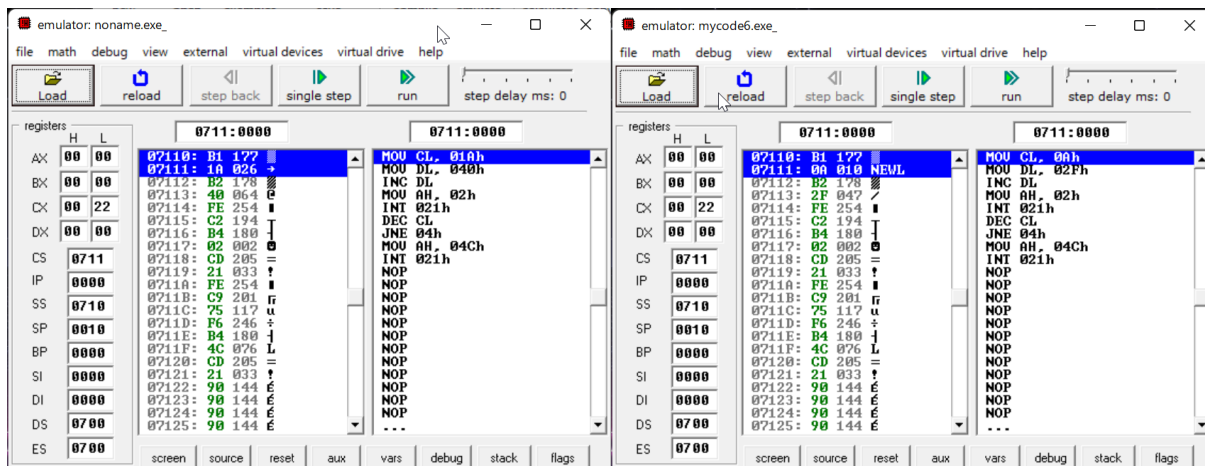
main endp

end main



```
01 .MODEL SMALL
02 .STACK 10H
03 .CODE
04 main proc
05 mov cl, 26
06 mov dl, 64
07 Bangladesh:
08 inc dl
09 mov ah, 2
10 int 21h
11 dec cl
12 jnz Bangladesh
13 exit:
14 mov ah, 4ch
15 int 21h
16 main endp
17 end main
```

Result/ Screenshot:



emulator: noname.exe_

file math debug view external virtual devices virtual drive help

Load reload step back single step run step delay ms: 0

registers

	H	L
AX	00	00
BX	00	00
CX	00	22
DX	00	00
SI	00	00
DI	00	00
DS	07	00
ES	07	00

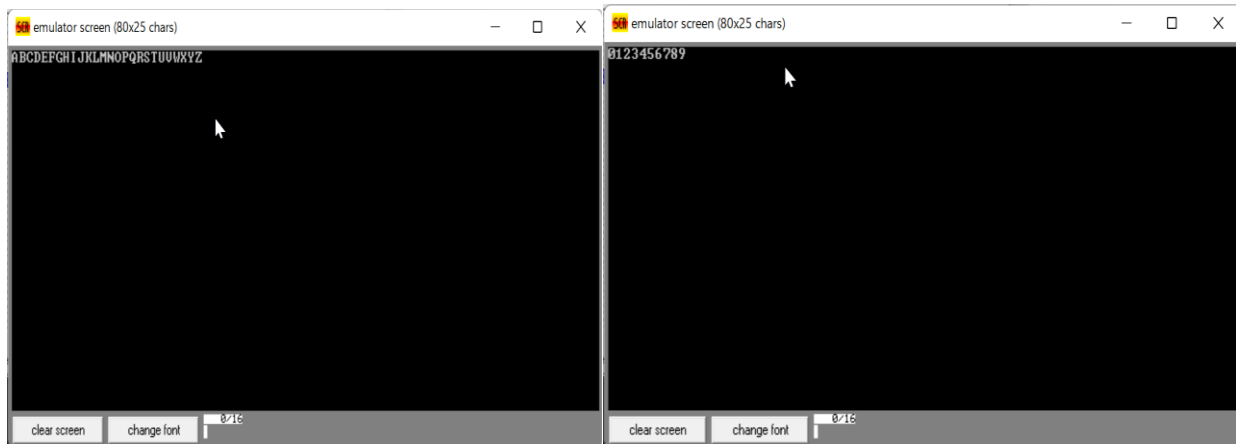
emulator: mycode6.exe_

file math debug view external virtual devices virtual drive help

Load reload step back single step run step delay ms: 0

registers

	H	L
AX	00	00
BX	00	00
CX	00	22
DX	00	00
SI	00	00
DI	00	00
DS	07	00
ES	07	00



Discussion: In this experiment we have learned about the giving any numerical and alphabetical sequence in the input and getting the output by using the emu8086 software. In this session we have learned how to give alphabetic words and numerical values also by a sequence. We have used ASCII table for this experiment. ASCII stands for American Standard Code for Information Interchange. Computers can only understand numbers, so an ASCII code is the numerical representation of a character such as 'a' or '@' or an action of some sort. ASCII was developed a long time ago and now the non-printing characters are rarely used for their original purpose.

Experiment No: 07

Experiment Name: Drawing multiple multi-color line using emu8086

Required Equipment:

1. A Computer
2. Emu8086 Software

Description: we must write the code on MAIN PROC. For graphical work we must take 13h in al and must call the int 10h. Then we have selected the 1st row by giving the value of dx = 1. Then we must give a particular color in al from which it will start the different color line. Then we started the loop. After that we must increase the value of dx and as our wish. We must give the value of ch, Ofor pixel print. Then we must value of cl and need to start another loop where to jump. Then we have to call the interrupt 10h and decrease the value of cl. Then we have to print column by jnz and print row by jz. Finally, we have to end our main program.

Source Code:

MAIN PROC

mov al,13h

mov ah,0

int 10h

mov dx,1

mov al,0001b

mov ch,0

Arif:

inc dx

inc dx

inc al

mov ah,0ch

int 10h

mov cl, 254

Hasnat:

int 10h

dec cl

jnz Hasnat

jz Arif

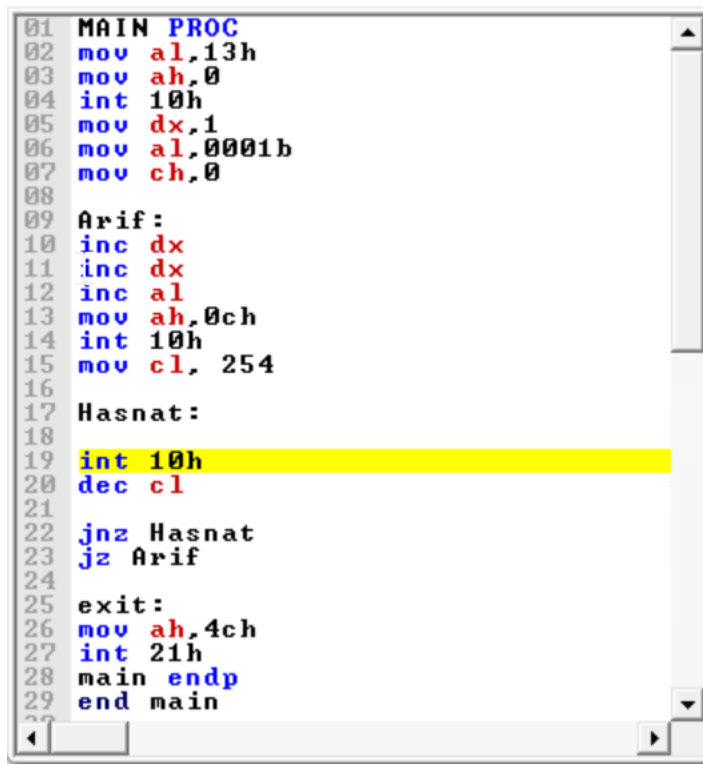
exit:

mov ah,4ch

int 21h

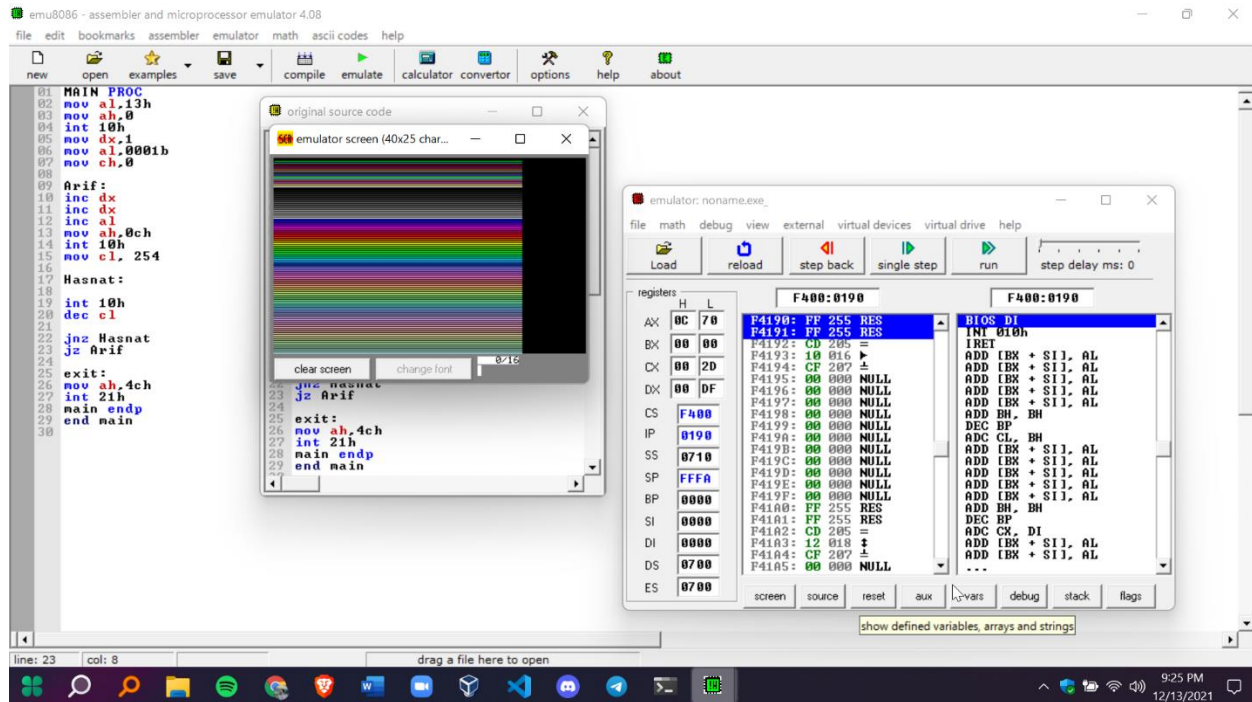
main endp

end main



```
01 MAIN PROC
02 mov al,13h
03 mov ah,0
04 int 10h
05 mov dx,1
06 mov al,0001b
07 mov ch,0
08
09 Arif:
10 inc dx
11 inc dx
12 inc al
13 mov ah,0ch
14 int 10h
15 mov cl, 254
16
17 Hasnat:
18
19 int 10h
20 dec cl
21
22 jnz Hasnat
23 jz Arif
24
25 exit:
26 mov ah,4ch
27 int 21h
28 main endp
29 end main
```

Result:



Discussion: In this experiment we have learned about the drawing multiple multicolor line using emu8086 graphical mode. In the output we can see the multi-color line. Its goes' line by line with different colors. We have done this by loop. We had to increase the value of dx and as our wish. Graphics mode is a simple call to the BIOS.