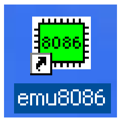
LAB#2

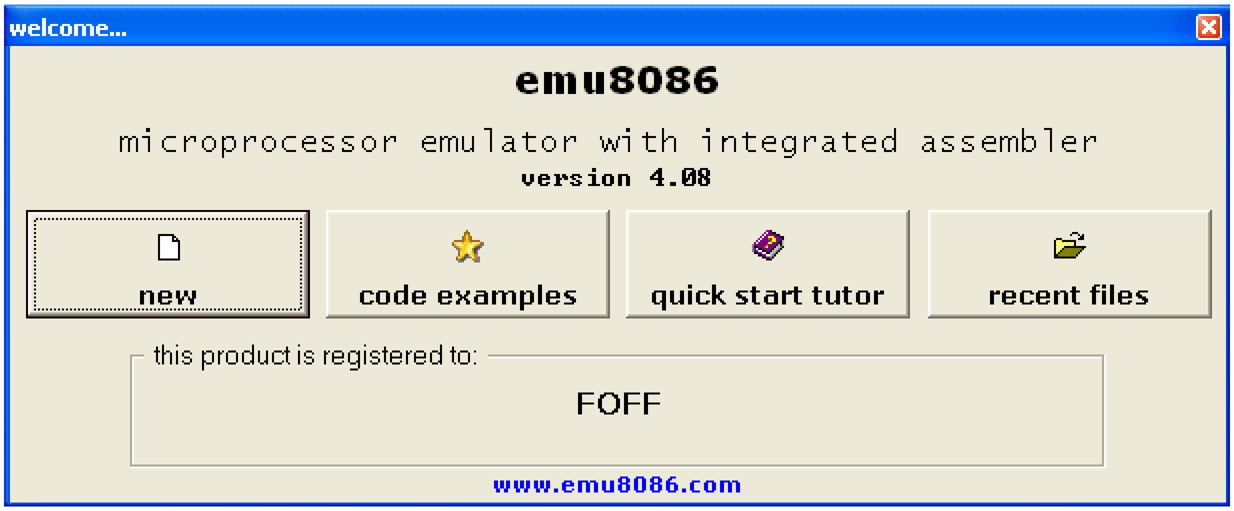
Objective

To understand and familiarize with the 8086 emulator environment.

8086 Microprocessor Emulator, also known as EMU8086, is an emulator of the program 8086 microprocessor. It is developed with a built-in 8086 assembler. This application is able to run programs on both PC desktops and laptops. This tool is primarily designed to copy or emulate hardware. These include the memory of a program, CPU, RAM, input and output devices, and even the display screen.

There are instructions to follow when using this emulator. It can be executed into one of the two ways: backward or forward. There are also examples of assembly source code included. With this, it allows the programming of assembly language, reverse engineering, hardware architecture, and creating miniature operating system (OS).



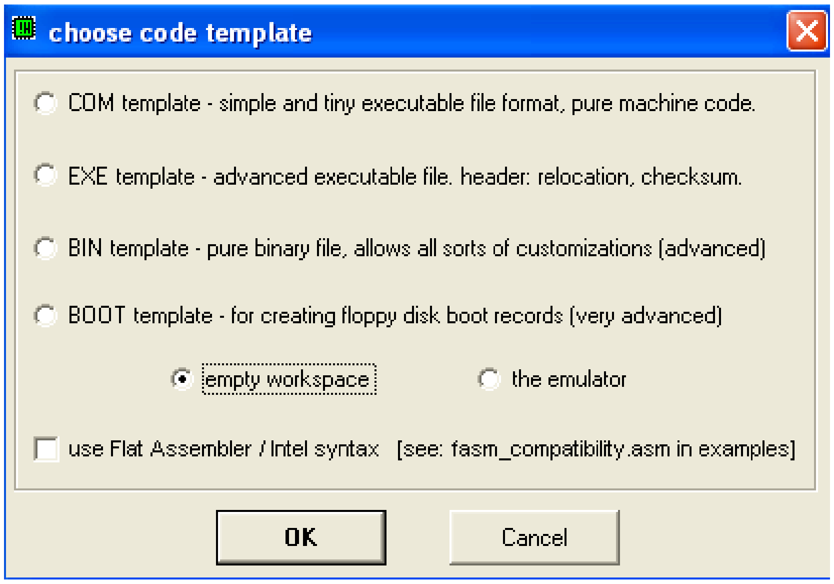

welcome...

emu8086

microprocessor emulator with integrated assembler
version 4.08

this product is registered to:
FOFF

www.emu8086.com



COM template - simple and tiny executable file format, pure machine code.

EXE template - advanced executable file. header: relocation, checksum.

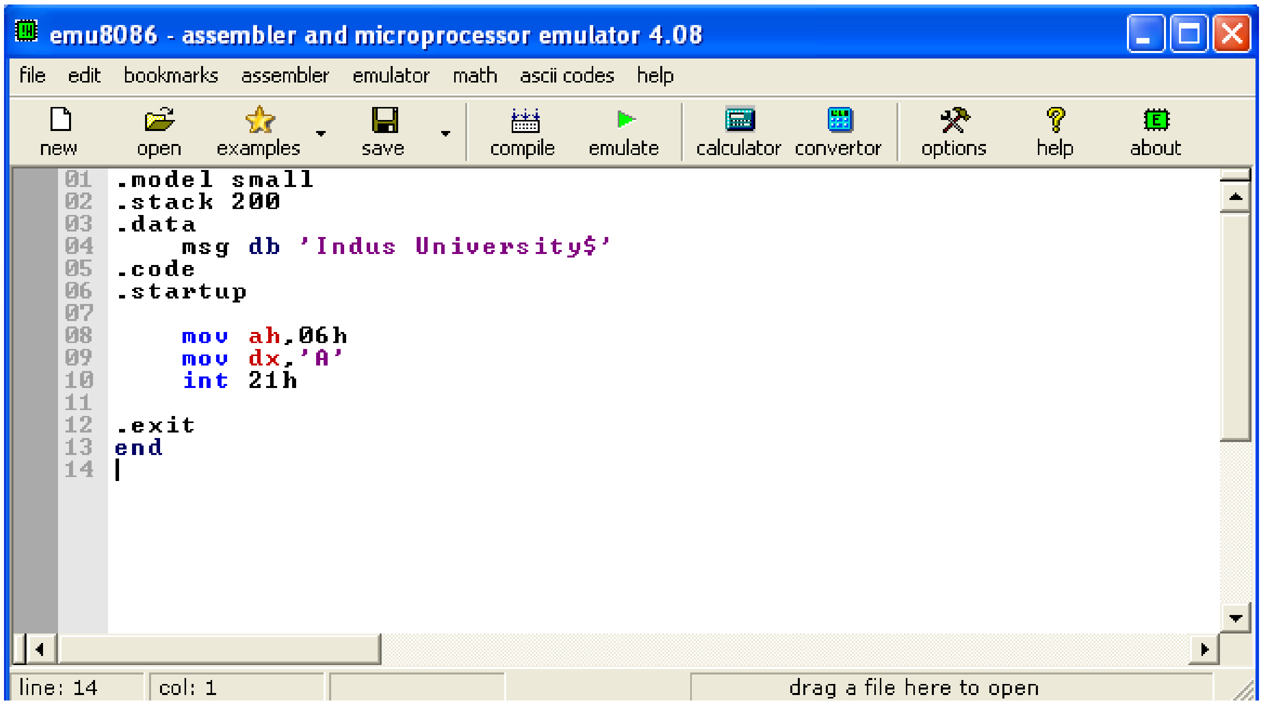
BIN template - pure binary file, allows all sorts of customizations (advanced)

BOOT template - for creating floppy disk boot records (very advanced)

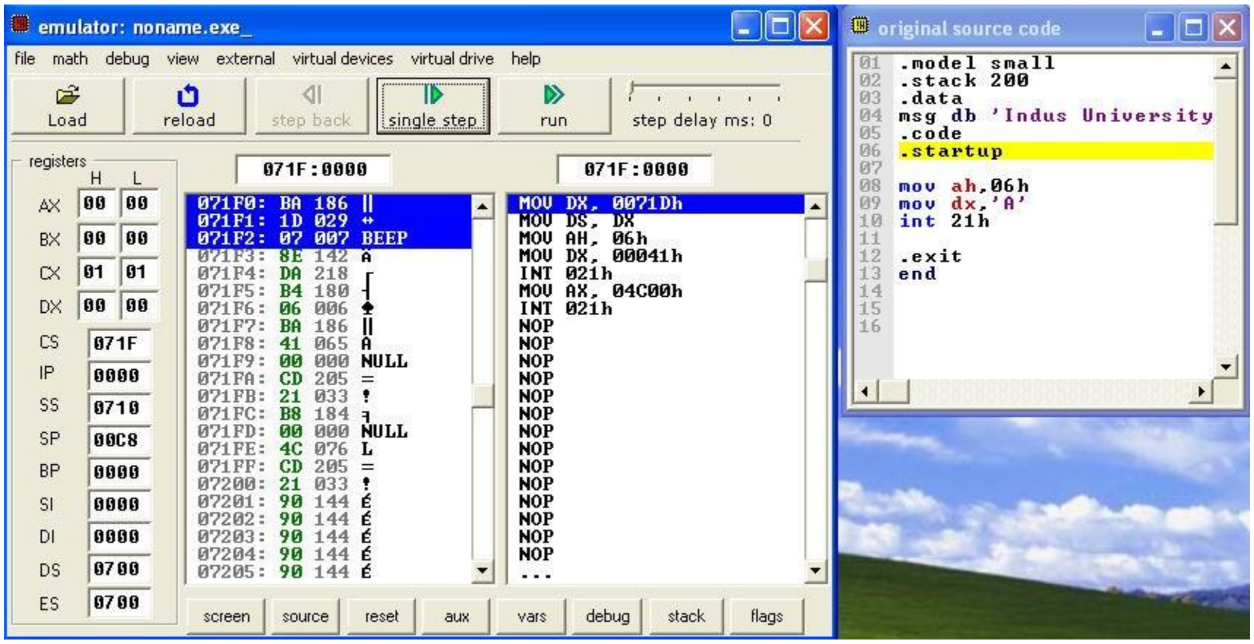
[✓ empty workspace] the emulator

☐ use Flat Assembler / Intel syntax [see: fasm_compatibility.asm in examples]

the directive `.model small` tells the assembler that you intend to use the small memory model - one code segment.
It simply tells the structure of the memory
The stack often holds temporary and local variables.


emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
line: 14 col: 1 drag a file here to open

```
.model small
.stack 200
.data
        msg db 'Indus University$'
.code
.startup

        mov ah,06h
        mov dx,'A'
        int 21h

.exit
end
```


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
screen source reset aux vars debug stack flags
original source code

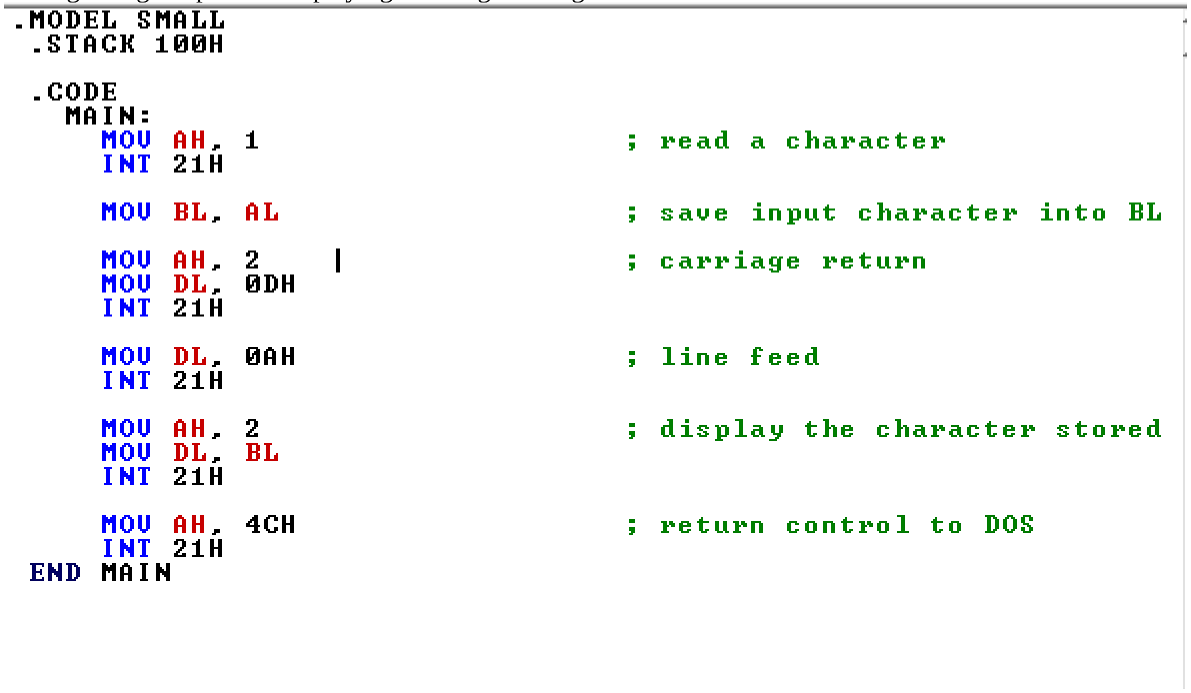
```
registers
     H   L
AX   00  00
BX   00  00
CX   01  01
DX   00  00
CS   071F
IP   0000
SS   0710
SP   00C8
BP   0000
SI   0000
DI   0000
DS   0700
ES   0700
```

```
071F:0000
071F0: BA 186 ||
071F1: 1D 029 ↔
071F2: 07 007 BEEP
071F3: 8E 142 Ä
071F4: DA 218 ┌
071F5: B4 180 ┤
071F6: 06 006 ♠
071F7: BA 186 ||
071F8: 41 065 A
071F9: 00 000 NULL
071FA: CD 205 ═
071FB: 21 033 !
071FC: B8 184 ╕
071FD: 00 000 NULL
071FE: 4C 076 L
071FF: CD 205 ═
07200: 21 033 !
07201: 90 144 É
07202: 90 144 É
07203: 90 144 É
07204: 90 144 É
07205: 90 144 É
```

```
071F:0000
MOV DX, 0071Dh
MOV DS, DX
MOV AH, 06h
MOV DX, 00041h
INT 021h
MOV AX, 04C00h
INT 021h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

```
.model small
.stack 200
.data
msg db 'Indus University
.code
.startup

mov ah,06h
mov dx,'A'
int 21h

.exit
end
```

Taking a single input and displaying it through carriage return

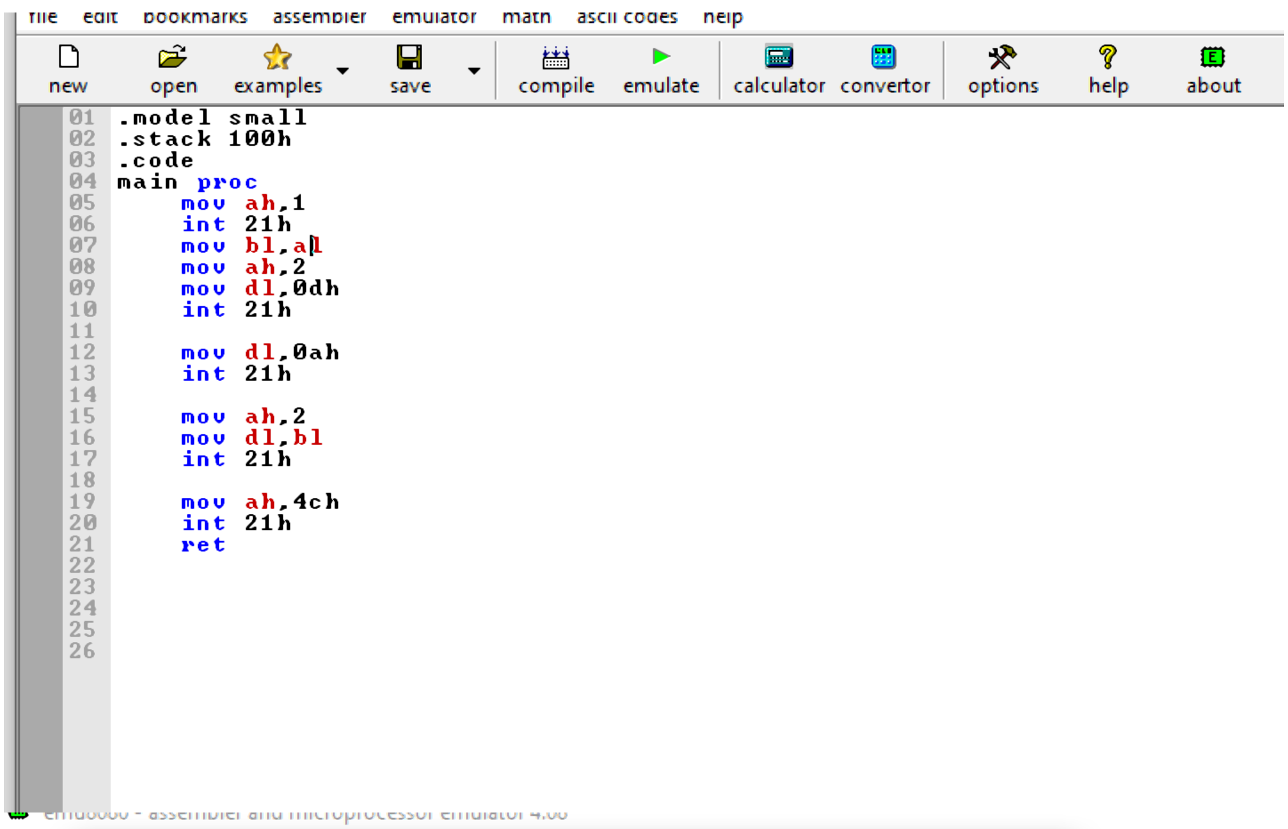


```
.MODEL SMALL
 .STACK 100H

 .CODE
   MAIN:
     MOV AH, 1                    ; read a character
     INT 21H

     MOV BL, AL                   ; save input character into BL

     MOV AH, 2                    ; carriage return
     MOV DL, 0DH
     INT 21H

     MOV DL, 0AH                  ; line feed
     INT 21H

     MOV AH, 2                    ; display the character stored
     MOV DL, BL
     INT 21H

     MOV AH, 4CH                  ; return control to DOS
     INT 21H
 END MAIN
```

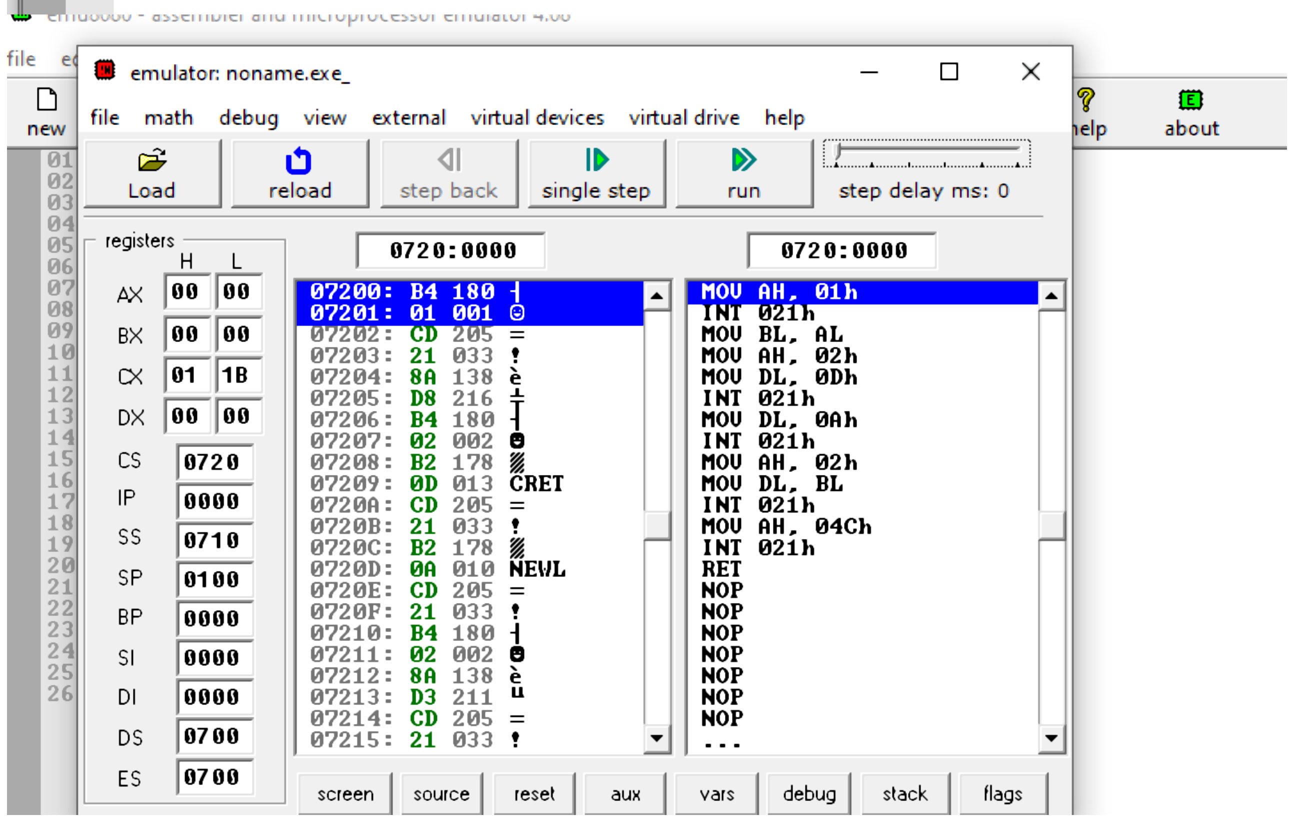
Use single step option for line by line execution

Lab Objective

Task#1: Write a program that takes a single character input and displays it in a new line and observe the contents of registers by using single stepping and record them



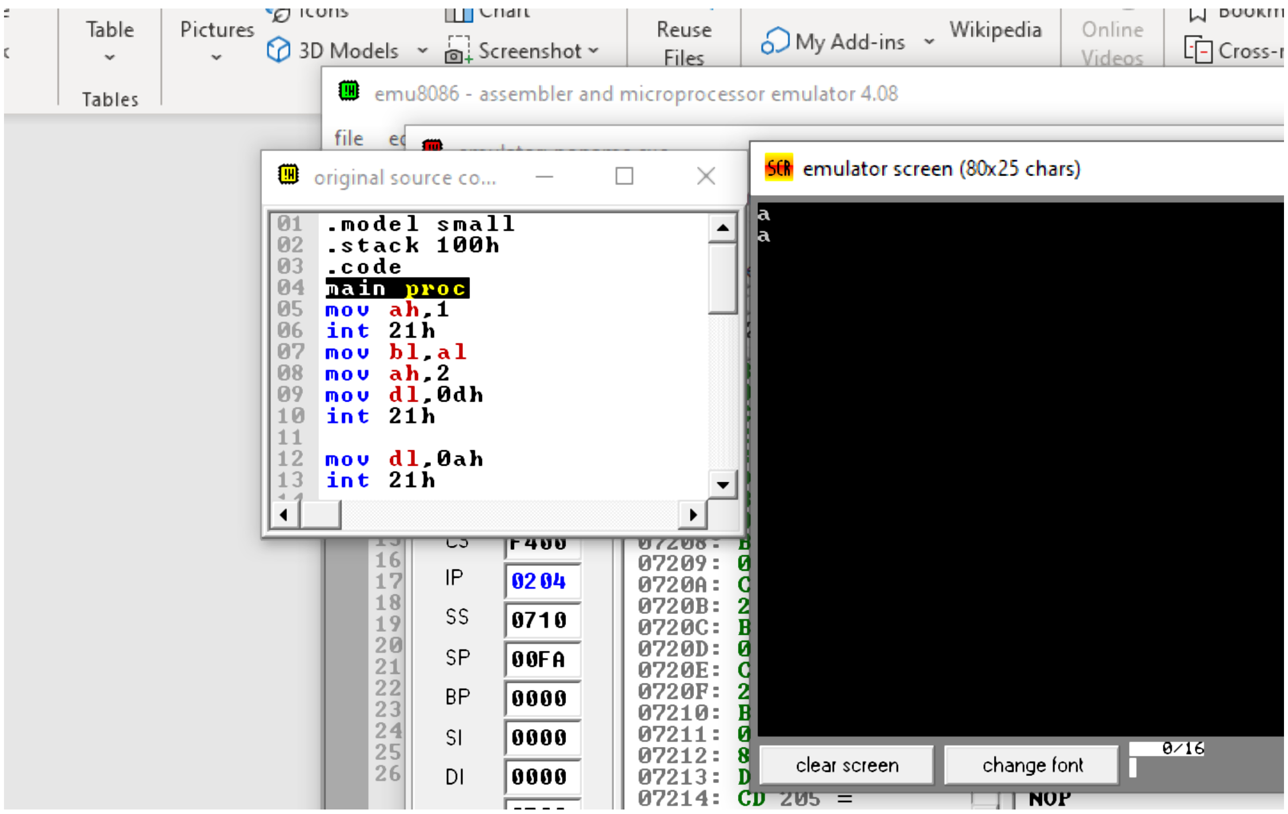
```
.model small
.stack 100h
.code
main proc
        mov ah,1
        int 21h
        mov bl,al
        mov ah,2
        mov dl,0dh
        int 21h

        mov dl,0ah
        int 21h

        mov ah,2
        mov dl,bl
        int 21h

        mov ah,4ch
        int 21h
        ret
```



	H	L
AX	00	00
BX	00	00
CX	01	1B
DX	00	00

```
CS 0720
IP 0000
SS 0710
SP 0100
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
```

```
0720:0000                    0720:0000

07200: B4 180 ┤              MOV AH, 01h
07201: 01 001 ☺              INT 021h
07202: CD 205 ═              MOV BL, AL
07203: 21 033 !              MOV AH, 02h
07204: 8A 138 è              MOV DL, 0Dh
07205: D8 216 ╪              INT 021h
07206: B4 180 ┤              MOV DL, 0Ah
07207: 02 002 ☻              INT 021h
07208: B2 178 ▓              MOV AH, 02h
07209: 0D 013 CRET           MOV DL, BL
0720A: CD 205 ═              INT 021h
0720B: 21 033 !              MOV AH, 04Ch
0720C: B2 178 ▓              INT 021h
0720D: 0A 010 NEWL           RET
0720E: CD 205 ═              NOP
0720F: 21 033 !              NOP
07210: B4 180 ┤              NOP
07211: 02 002 ☻              NOP
07212: 8A 138 è              NOP
07213: D3 211 ╙              NOP
07214: CD 205 ═              NOP
07215: 21 033 !              ...
```

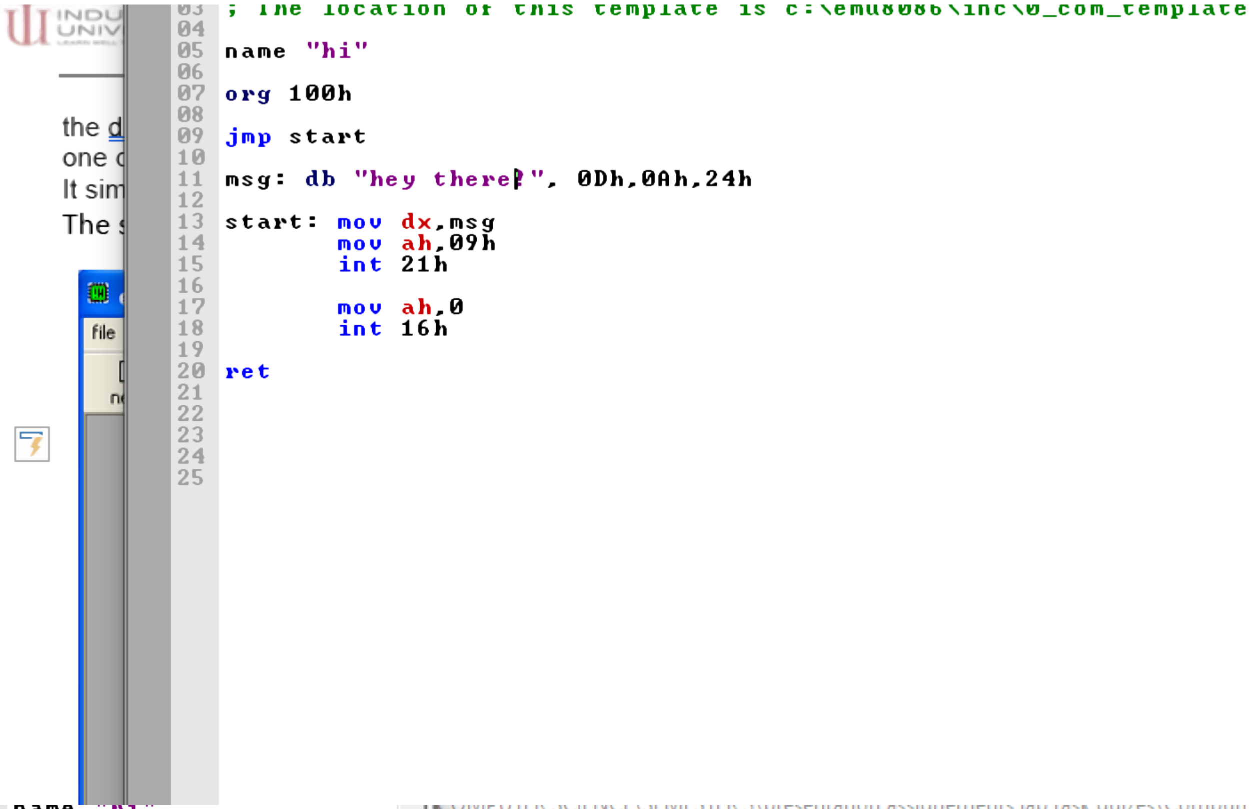

emu8086 - assembler and microprocessor emulator 4.08
original source co...
emulator screen (80x25 chars)
clear screen
change font

```
.model small
.stack 100h
.code
main proc
mov ah,1
int 21h
mov bl,al
mov ah,2
mov dl,0dh
int 21h

mov dl,0ah
int 21h
```

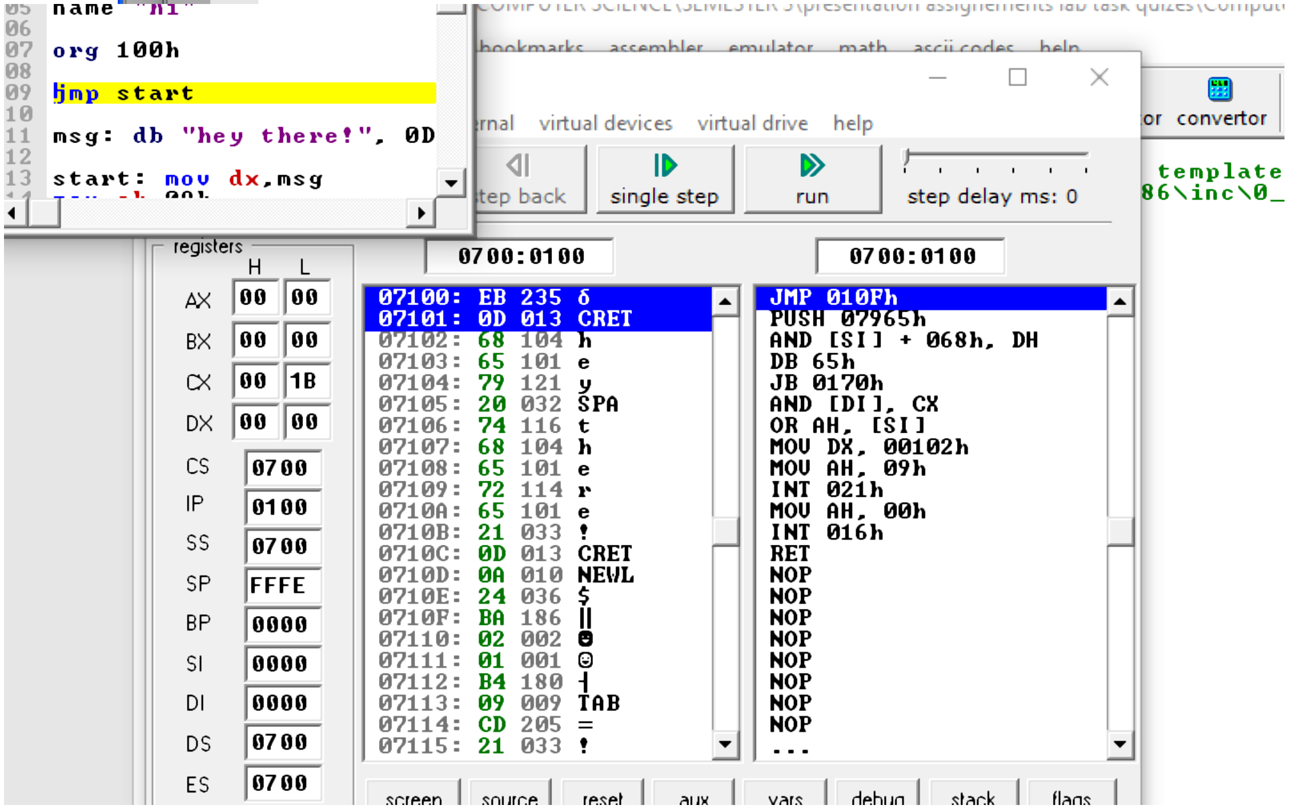
```
a
a
```

IP	0204
SS	0710
SP	00FA
BP	0000
SI	0000
DI	0000

Task#2: Write a program to display a string in Assembly Language**.**



```
; The location of this template is c:\emu8086\inc\0_com_template

name "hi"

org 100h

jmp start

msg: db "hey there!", 0Dh,0Ah,24h

start: mov dx,msg
       mov ah,09h
       int 21h

       mov ah,0
       int 16h

ret
```



```
name "hi"

org 100h

jmp start

msg: db "hey there!", 0D

start: mov dx,msg
```

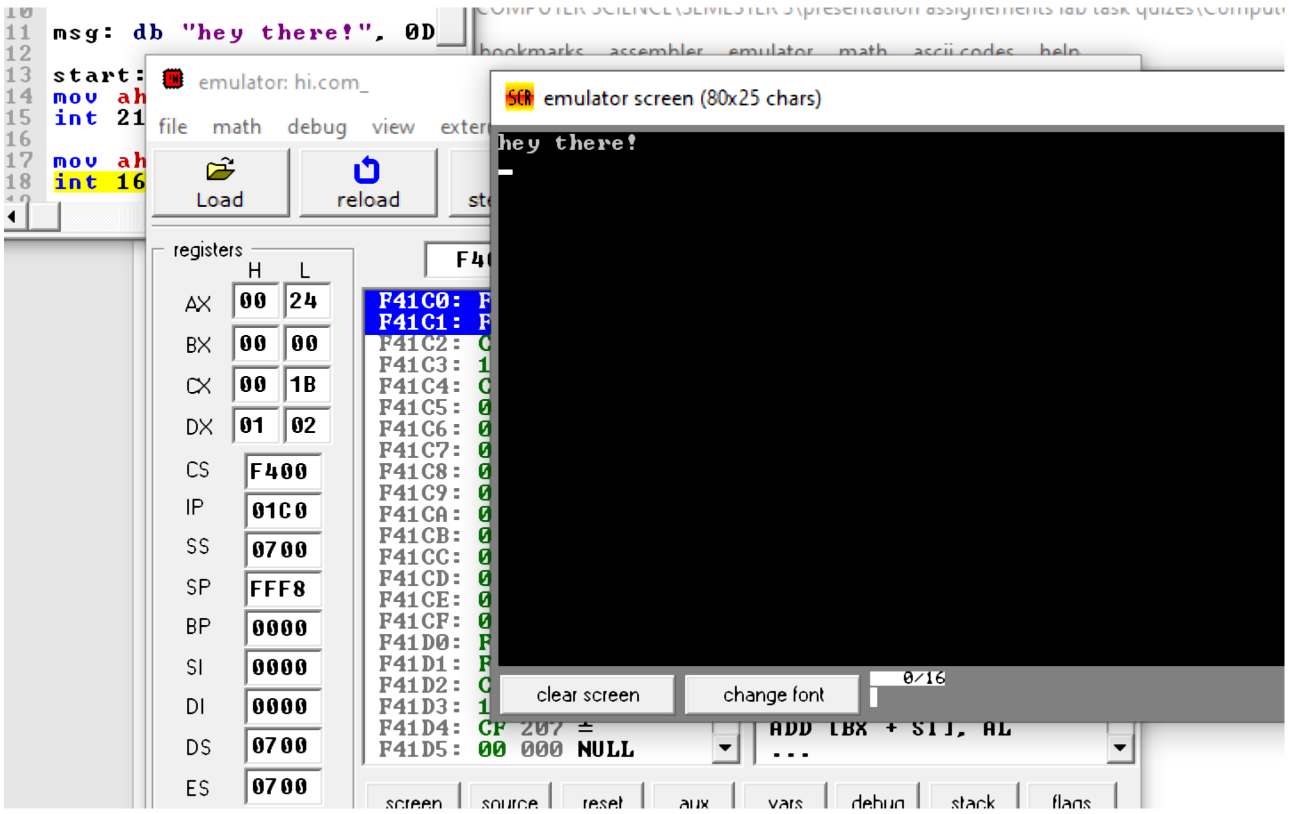
registers

	H	L
AX	00	00
BX	00	00
CX	00	1B
DX	00	00

CS	0700
IP	0100
SS	0700
SP	FFFE
BP	0000
SI	0000
DI	0000
DS	0700
ES	0700

```
0700:0100
07100: EB 235 δ
07101: 0D 013 CRET
07102: 68 104 h
07103: 65 101 e
07104: 79 121 y
07105: 20 032 SPA
07106: 74 116 t
07107: 68 104 h
07108: 65 101 e
07109: 72 114 r
0710A: 65 101 e
0710B: 21 033 !
0710C: 0D 013 CRET
0710D: 0A 010 NEWL
0710E: 24 036 $
0710F: BA 186 ||
07110: 02 002 ☻
07111: 01 001 ☺
07112: B4 180 ┤
07113: 09 009 TAB
07114: CD 205 =
07115: 21 033 !
```

```
0700:0100
JMP 010Fh
PUSH 07965h
AND [SI] + 068h, DH
DB 65h
JB 0170h
AND [DI], CX
OR AH, [SI]
MOV DX, 00102h
MOV AH, 09h
INT 021h
MOV AH, 00h
INT 016h
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```



```
msg: db "hey there!", 0D
start:
mov ah
int 21
mov ah
int 16
```

emulator: hi.com_

emulator screen (80x25 chars)

```
hey there!
```

registers	H	L
AX	00	24
BX	00	00
CX	00	1B
DX	01	02

CS	F400
IP	01C0
SS	0700
SP	FFF8
BP	0000
SI	0000
DI	0000
DS	0700
ES	0700

```
F41C0: F
F41C1: F
F41C2: C
F41C3: 1
F41C4: C
F41C5: 0
F41C6: 0
F41C7: 0
F41C8: 0
F41C9: 0
F41CA: 0
F41CB: 0
F41CC: 0
F41CD: 0
F41CE: 0
F41CF: 0
F41D0: F
F41D1: F
F41D2: C
F41D3: 1
F41D4: CF 207 ±
F41D5: 00 000 NULL
```

```
ADD [BX + SI], AL
...
```

Task#2:

Observe the contents of registers by using single stepping and record them.

```
.MODEL SMALL
.STACK 100H
.DATA
        MESSAGE1 DB 0AH, 0DH, "INDUS UNIVERSITY$"
.CODE
MAIN:
        MOV AX, @DATA
        MOV DS, AX
        MOV DX, OFFSET MESSAGE1
        MOV AH, 09H
        INT 21H

        MOV AH, 4CH
        INT 21H
END MAIN
```

Registers	After 1st Instruction	After 2nd Instruction	After 3rd Instruction	After 4th Instruction	After 5th Instruction
AX	07 | 20	07 | 20	07 | 20	09 | 20	09 | 20
BX	00|00	00 | 00	00 | 00	00 | 00	00 | 00
CX	01 | 30	01 | 30	01 | 30	01 | 30	01 | 30
DX	00|00	00 | 00	00 | 00	00 | 00	00 | 00

Registers	After 6th Instruction	After 7th Instruction	After 8th Instruction	After 9th Instruction	After 10th Instruction
AX	09 | 24	09 | 24	4C | 24	4C | 24	4C | 24
BX	00|00	00|00	00|00	00|00	00|00
CX	01 | 30	01 | 30	01 | 30	00|00	00|00
DX	00|00	00|00	00|00	00|00	00|00