**Comprehensive Assembly Lab Final**

**BY**

**MD: KAMRUZZAMAN**

**ID: CSE2103024039**

**Course Title: microprocessor and assembly language sessional**

**Course code : CSE312**

**Semester : Fall\_2023**

**SUBMITTED BY**

**Md. Ashfakur Rahman**

Lecturer

# DEPARTMENT OF CSE

# ****SONARGAON UNIVERSITY****

****

# ****SONARGAON UNIVERSITY****

# ****DHAKA, BANGLADESH****

# ****December 2023****

TABLE OF CONTENTS

CONTENS PAGE

CHAPTER 1: **Assembly language** 01-12

1.1 Introduction and Syntax assembly language 02

1.2 Arithmetic and logical operations 03

1.3 Conditions and nested conditions 04-05

1.4 Loop and nested loop operations  06

1.5 String and Array operations 07-09

1.7 Procedures 10-11

1.8 Stack operation 11-12

CHAPTER 1

**Assembly language**

# 1.1 Introduction

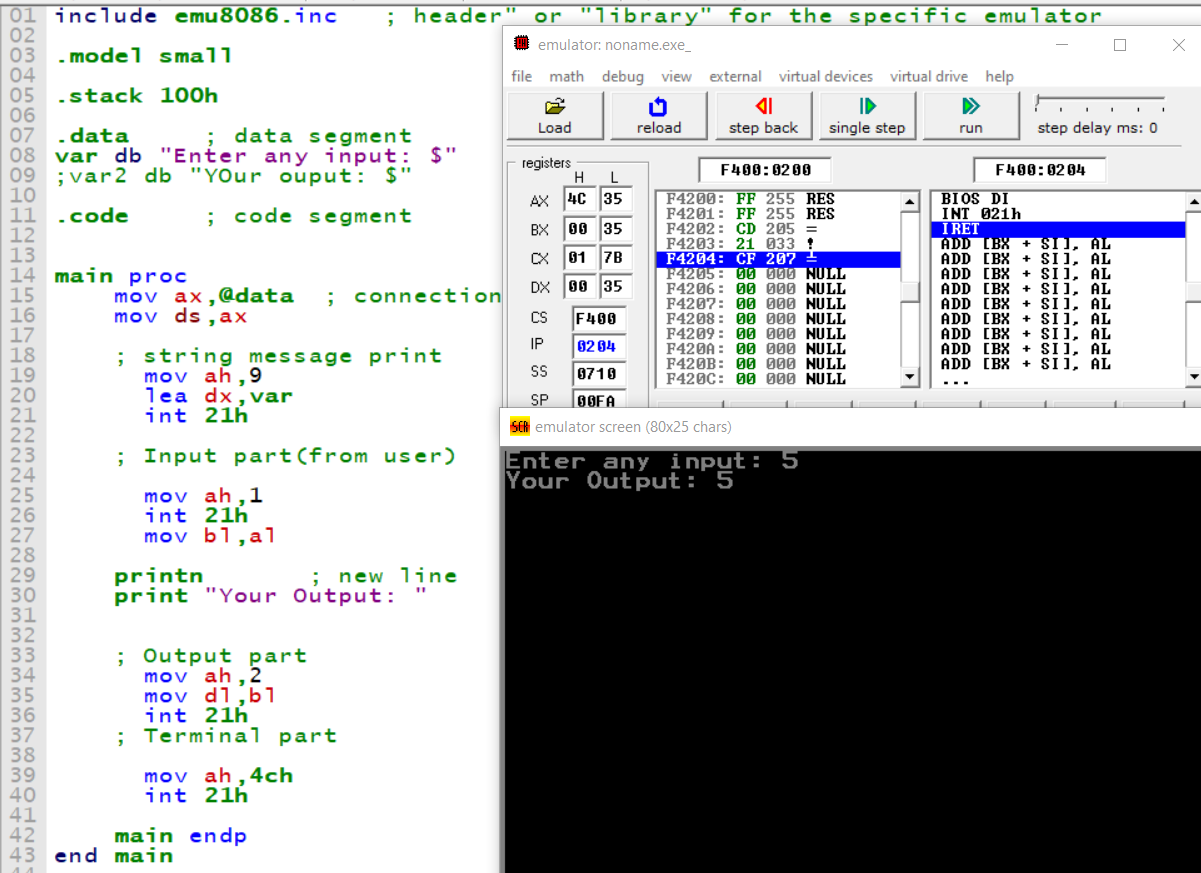
**Assembly language is a low level programming language that provides a human-readable presentation of machine code instructions for a specific computer architecture or micro-processor.**

The "Father of Assembly Language" is **Maurice Wilkes**, and he introduced the concept in the year **1947**.

Key Features of Assembly Language

* Low-Level Programming Language.
* Mnemonics and Symbols instructions.
* Close to Machine Code.
* Direct Access to Hardware

The following represents the syntax of assembly language



**Figure 1.1: syntax of assembly language**

# 1.2 Arithmetic and logical operations

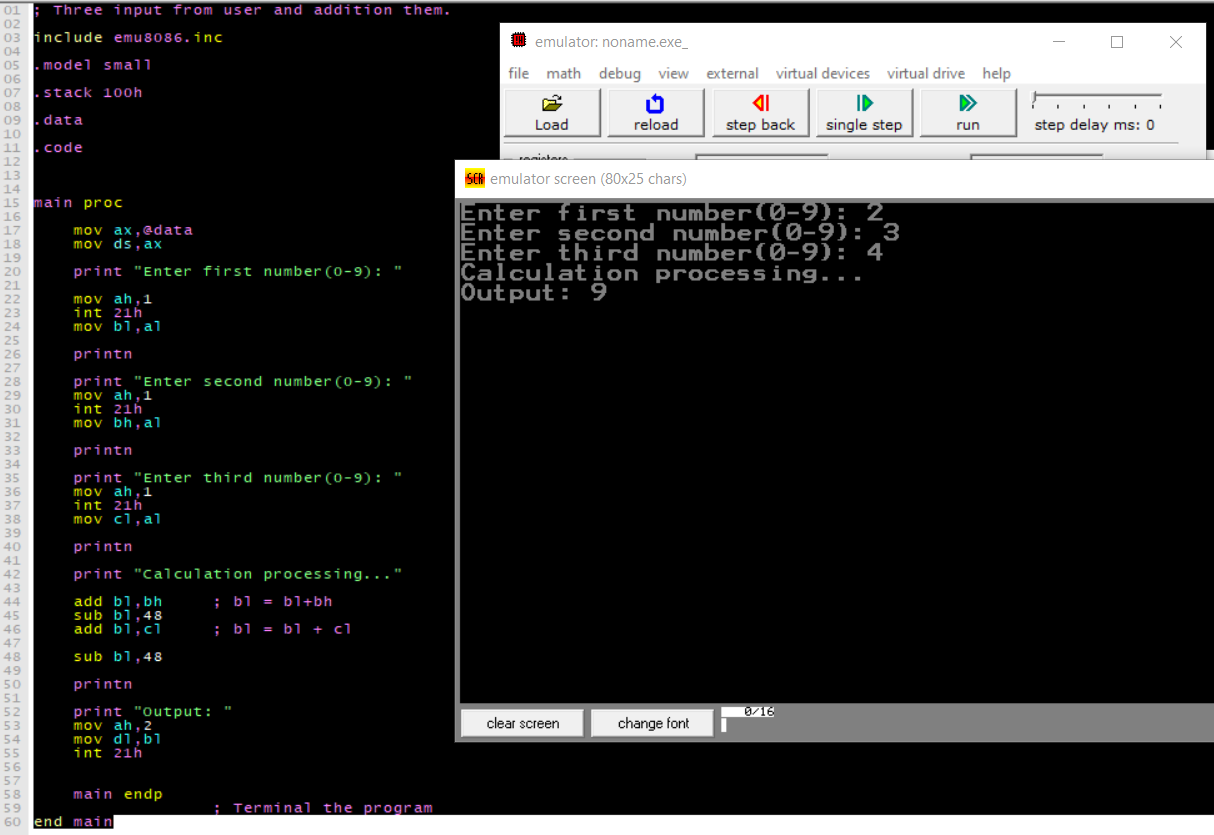
**Task Name: Three inputs from the user and addition.**

**; Three input from user and addition them**

**include emu8086.inc**

**.model small**

**.stack 100h**

**.data**

**.code**

**main proc**

**mov ax,@data**

**mov ds,ax**

**print "Enter first number: "**

**mov ah,1**

**int 21h**

**mov bl,al**

**printn**

**print "Enter second number: "**

**mov ah,1**

**int 21h**

**mov bh,al**

**printn**

**print "Enter third number: "**

**mov ah,1**

**int 21h**

**mov cl,al**

**printn**

**print "Calculation processing..."**

**add bl,bh ; bl = bl+bh**

**sub bl,48**

**add bl,cl ; bl = bl + cl**

**Figure 1.2: Addition three numbeRS**

**sub bl,48**

**printn**

**print "Output: "**

**mov ah,2**

**mov dl,bl**

**int 21h**

**main endp**

**; Terminal the program**

**end main**

**1.3 Conditions and nested conditions**

**Task Name: Using assembly programming to find the larger of three user inputs**

**include emu8086.inc**

**.model small**

**.stack 100h**

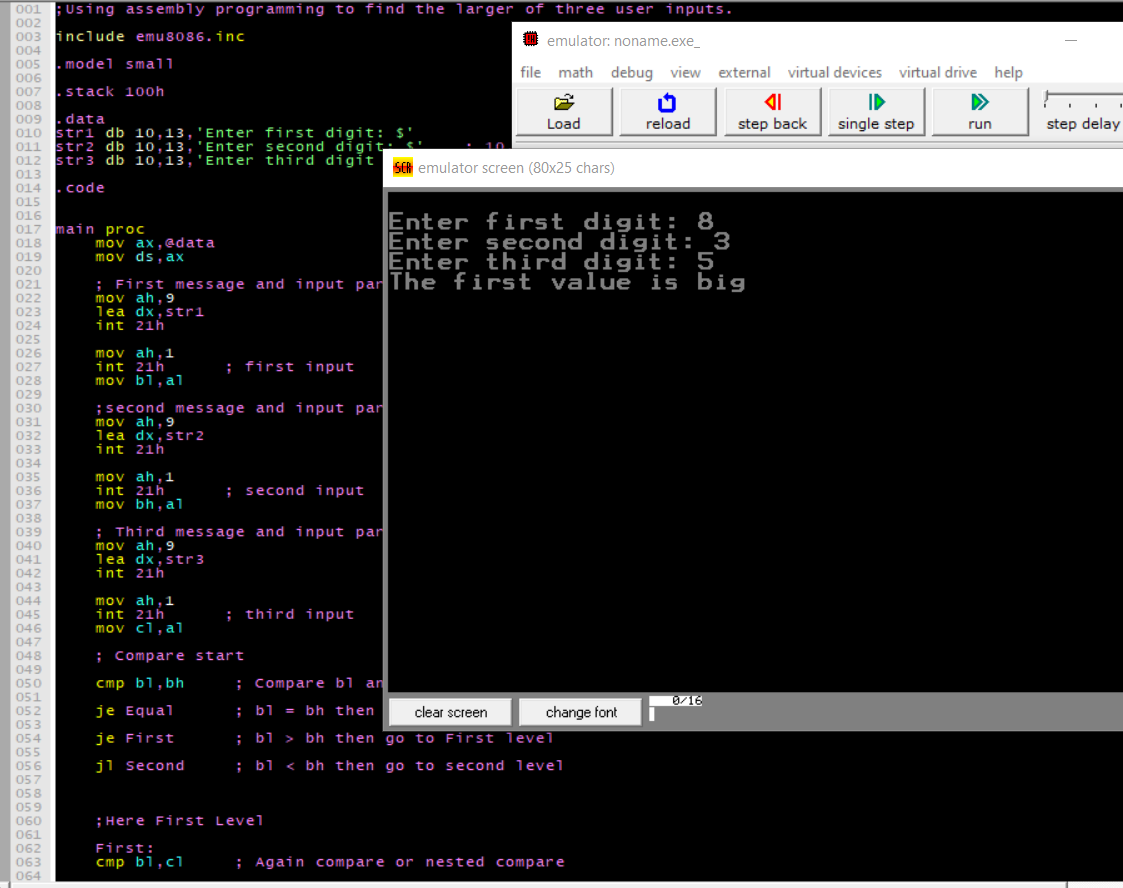
**.data**

**str1 db 10,13,'Enter first digit: $'**

**str2 db 10,13,'Enter second digit: $' ; 10,13 denotes printing a new line following the message display.**

**str3 db 10,13,'Enter third digit : $'**

**.code**

**main proc**

**mov ax,@data**

**mov ds,ax**

**; First message and input part**

**mov ah,9**

**lea dx,str1**

**int 21h**

**mov ah,1**

**int 21h ; first input**

**mov bl,al**

**;second message and input part**

**mov ah,9**

**lea dx,str2**

**int 21h**

**mov ah,1**

**int 21h ; second input**

**mov bh,al**

**; Third message and input part**

**mov ah,9**

**lea dx,str3**

**int 21h**

**mov ah,1**

**int 21h ; third input**

**mov cl,al**

**; Compare start**

**cmp bl,bh ; Compare bl and bh**

**je Equal ; bl = bh then go to Equal level**

**je First ; bl > bh then go to First level**

**jl Second ; bl < bh then go to second level**

**;Here First Level**

**First:**

**cmp bl,cl ; Again compare or nested compare**

**je Equal ; bl = cl then go to Equal level**

**je Level ; bl > cl then go to Level level**

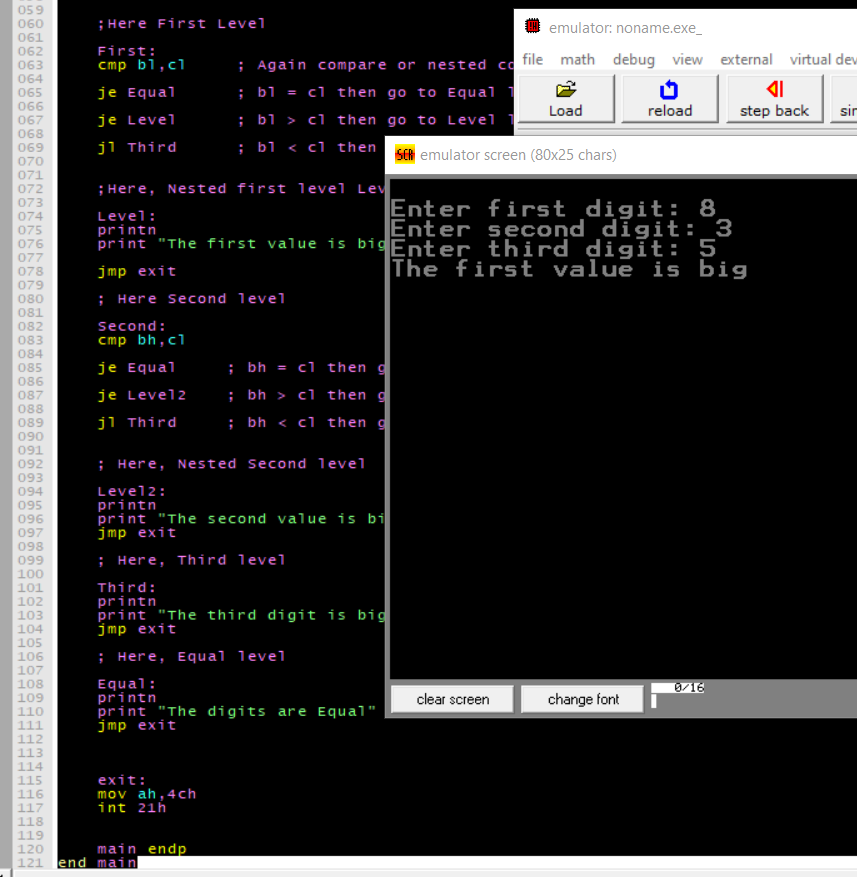
**jl Third ; bl < cl then go to Third level**

**;Here, Nested first level Level**

**Level:**

**printn**

**print "The first value is big"**

****

**jmp exit**

**; Here Second level**

**Second:**

**cmp bh,cl**

**je Equal ; bh = cl then go to Equal level**

**je Level2 ; bh > cl then go to Level2**

**jl Third ; bh < cl then go to Third level**

**; Here, Nested Second level**

**Level2:**

**printn**

**print "The second value is big"**

**jmp exit**

**; Here, Third level**

**Third:**

**printn**

**print "The third digit is big"**

**jmp exit**

**; Here, Equal level**

**Equal:**

**printn**

**print "The digits are Equal"**

**jmp exit**

**exit:**

**mov ah,4ch**

**int 21h**

**main endp**

**end main**

**1.4 Loop and nested loop operations**

**Task Name: 04 Write an assembly program to display Capital letters in increment order and decrement (reverse) order.**

**include emu8086.inc**

**.model small**

**.stack 100h**

**.data**

**str1 db "-------- Capital Letter Increment Order -------$"**

**str2 db '-------- Capital Letter Decrement order -------$'**

**.code**

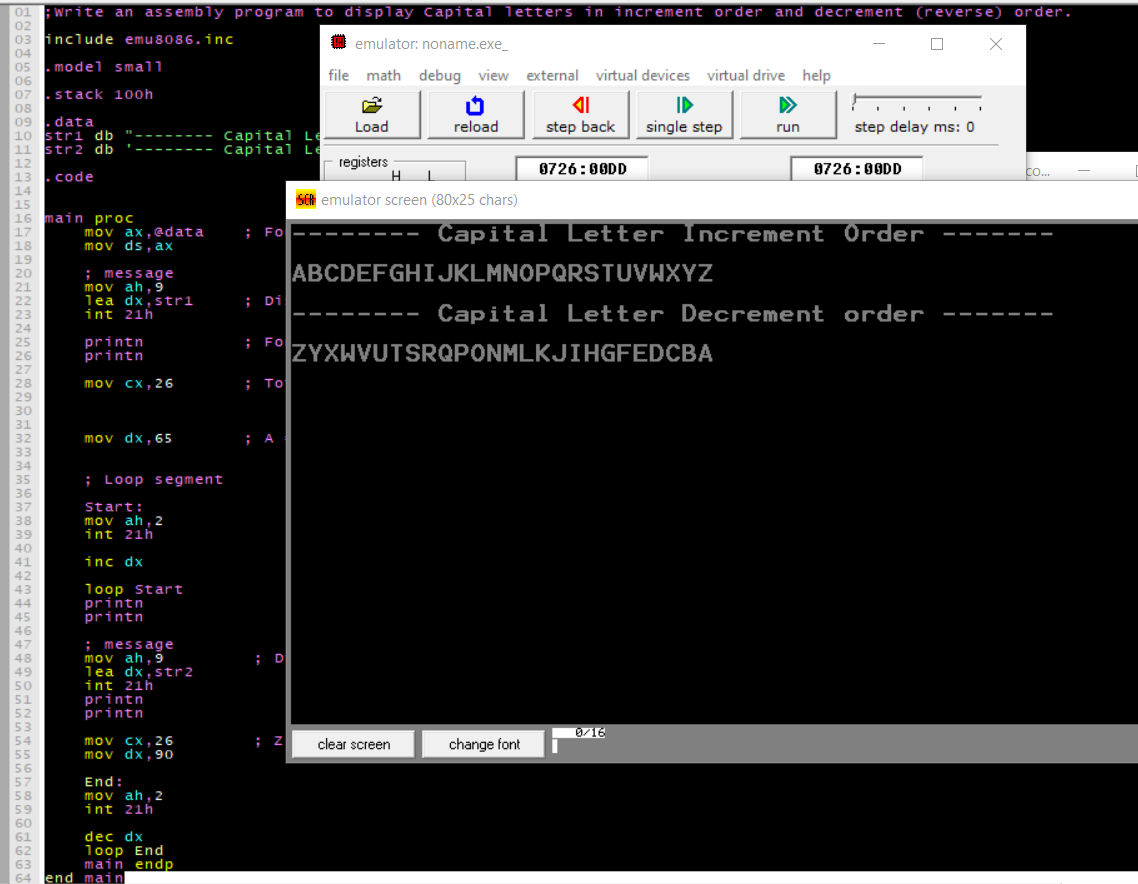
**main proc**

**mov ax,@data ; For connection data segment to code segment**

**mov ds,ax**

**; message**

**mov ah,9**

** lea dx,str1 ; Display String Message(Accending order)**

**int 21h**

**printn ; For New line**

**printn**

**mov cx,26 ; Total alphabet 26. so, the loop is round by 26 times (that way cx=26).**

**mov dx,65 ; A = 65 and Z = 90 . so , dx is start from 65**

**; Loop segment**

**Start:**

**mov ah,2**

**int 21h**

**inc dx**

**loop Start**

**printn**

**printn**

**; message**

**mov ah,9 ; Display String Message(descending order)**

**lea dx,str2**

**int 21h**

**printn**

**printn**

**mov cx,26 ; Z = 90**

**mov dx,90**

**End:**

**mov ah,2**

**int 21h**

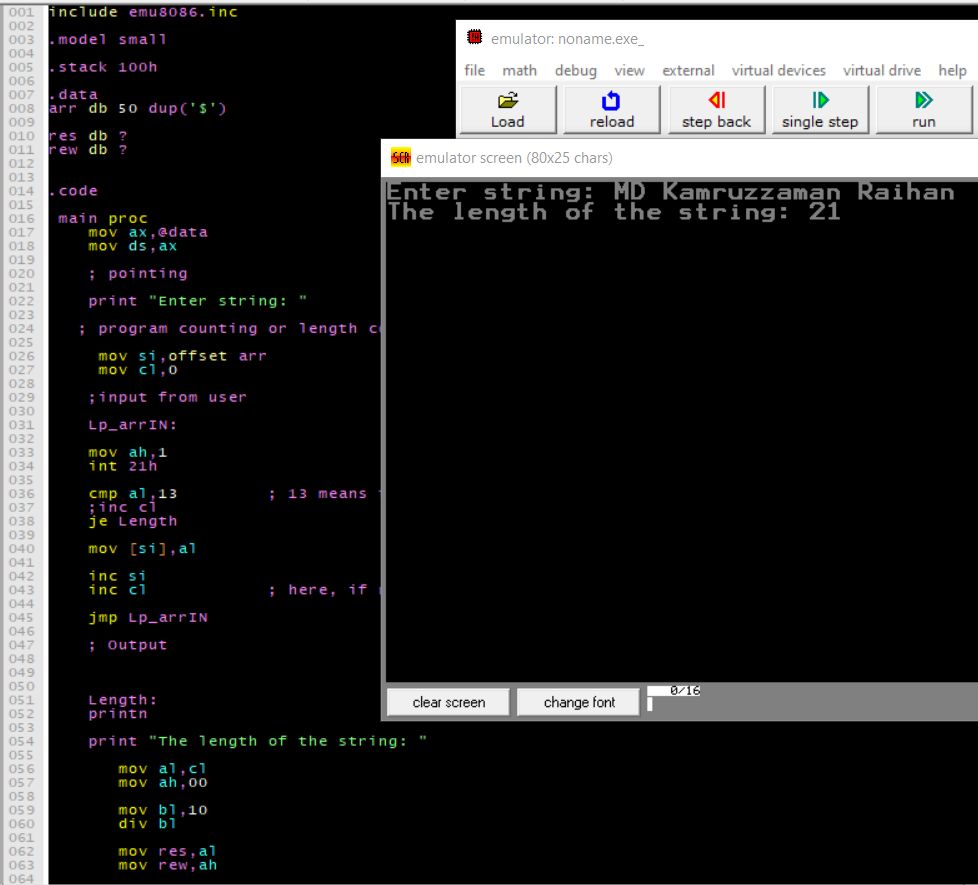
**dec dx**

**loop End**

**main endp**

**end main**

**1.5 String and Array operations**

**Task Name : 05 find length of string in assembly language 8086**

**include emu8086.inc**

**.model small**

**.stack 100h**

**.data**

**arr db 50 dup('$')**

**res db ?**

**rew db ?**

**.code**

**main proc**

**mov ax,@data**

**mov ds,ax**

**; pointing**

**print "Enter string: "**

**; program counting or length count**

**mov si,offset arr**

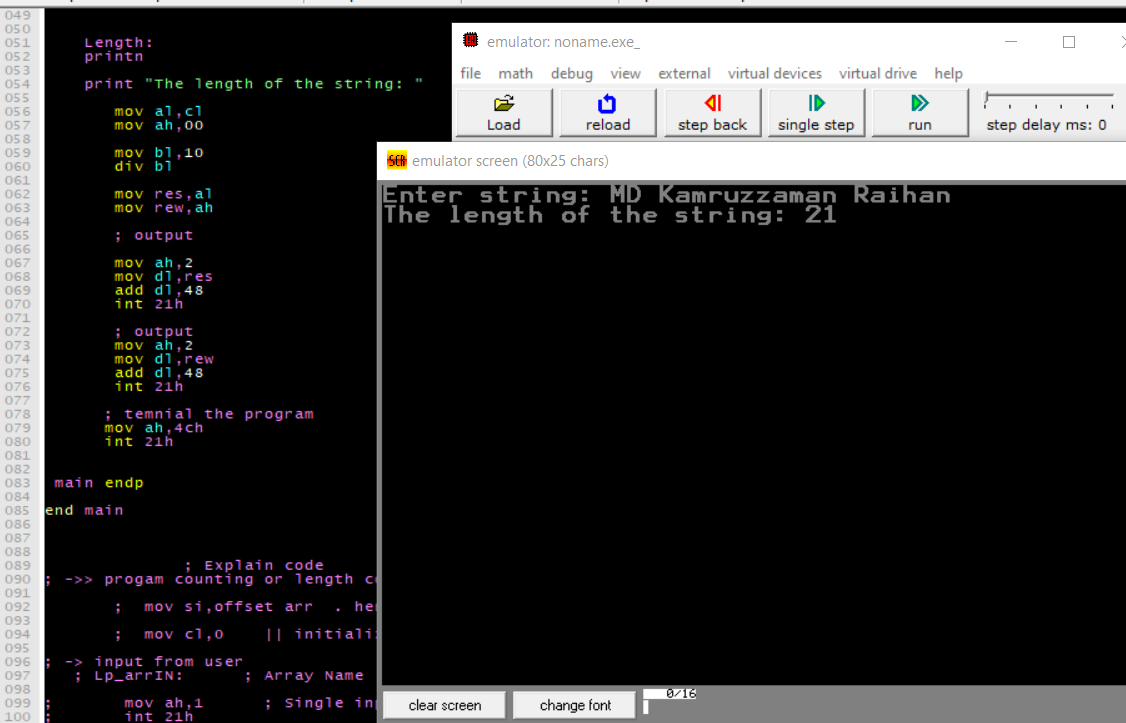
**mov cl,0**

**;input from user**

**Lp\_arrIN:**

**mov ah,1**

**int 21h**

** cmp al,13 ; 13 means the keyboard enter key.**

**;inc cl**

**je Length**

**mov [si],al**

**inc si**

**inc cl ; here, if necessary cx register then use jmp condition.**

**jmp Lp\_arrIN**

**; Output**

**Length:**

**printn**

**print "The length of the string: "**

**mov al,cl**

**mov ah,00**

**mov bl,10**

**div bl**

**mov res,al**

**mov rew,ah**

**; output**

**mov ah,2**

**mov dl,res**

**add dl,48**

**int 21h**

**; output**

**mov ah,2**

**mov dl,rew**

**add dl,48**

**int 21h**

**; temnial the program**

**mov ah,4ch**

**int 21h**

**main endp**

**end main**

**; Explain code**

**; ->> progam counting or length count**

**; mov si,offset arr . here si is the sources index and offset in the pointing and store the base address**

**; mov cl,0 || initialize program counter here, length of strign is zero.**

**; -> input from user**

**; Lp\_arrIN: ; Array Name**

**; mov ah,1 ; Single input string function.**

**; int 21h**

**; cmp al,13 ; compare al and Keyboard enter key. if user press enter key or not**

**; je Length ; if user press then go to level Length**

**; if not press then working step by step next instruction**

**; mov [si],al ; --> Step-01:**

**; suppose user input : 2 or al store the value al = 2 ; here, si store or pointing the base address not value**

**; move or load this value of si first index or si = 0 or arr[0] = 2 || arr[si] = 2**

**; inc si ; increment si index so, si = 1**

**; inc cl ; cl = 1**

**; jmp Lp\_arrIN ; jump to Lp\_arrIN**

**;--> Step\_02:**

**; input = 8**

**; cmp 8 == Enter False go to next instruction**

**; si = 1 or arr[1] = 8**

**; si = 3**

**; cl = 2**

**;--> Step\_03:**

**; input = Enter or (13)**

**; cmp Enter == Enter. True go to Length level**

**; Length level**

**; Length:**

**; printn ; new line**

**; print "The length of the string: "**

**; mov al,cl ; al = cl . here, cl = 2 || so, al == 2**

**; mov ah,00 ; ah == 0**

**; mov bl,10 ; bl == 10**

**; div bl ; div bl means**

**; al = al / bl here result go to al register and remainder go to bl register.**

**; al = 2 / 10**

**; 10) 2 ( 0 <-- res**

**; 0**

**; ----**

**; 2 <-- rew**

**; al == 0**

**; and**

**; ah == 2**

**; mov res,al ; res == 0**

**; mov rew,ah ; rew == 2**

**; output**

**; mov ah,2 ; Single output function**

**; mov dl,res ; dl == 0**

**; add dl,48 ; dl = dl + 48 convert numerical value to ascii value in the range of 0-9.**

**; int 21h**

**; mov ah,2**

**; mov dl,rew**

**; add dl,48**

**; int 21h**

**1.7 Procedures**

**Task No. : 07 Procedure in assembly language.**

**include emu8086.inc**

**.model small**

**.stack 100h**

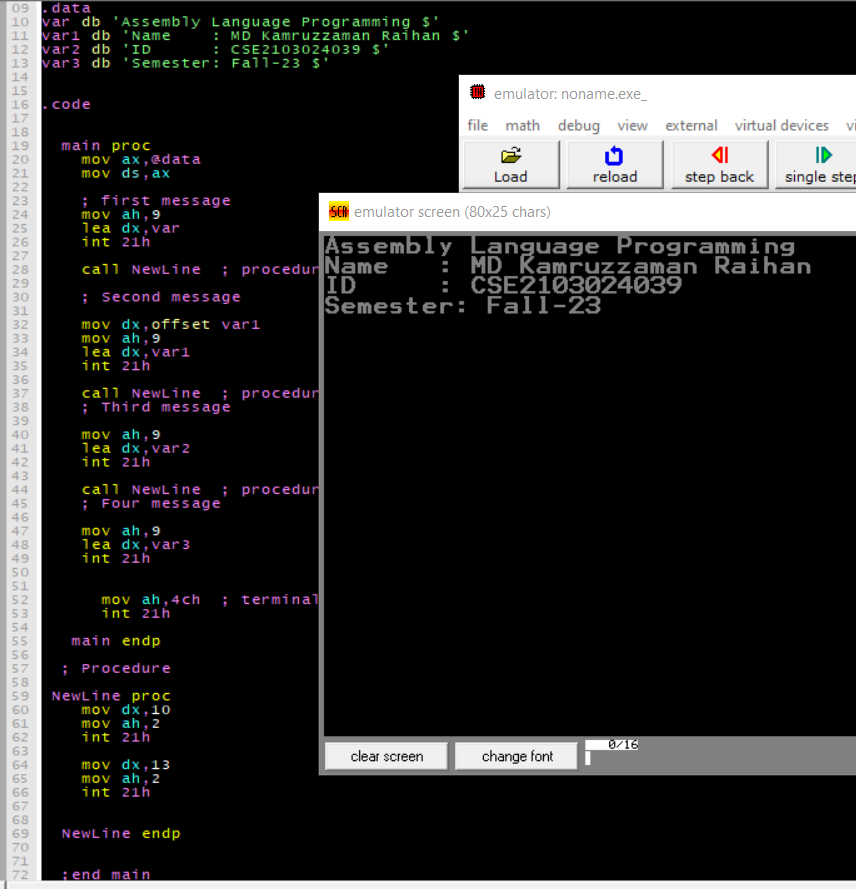
**.data**

**var db 'Assembly Language Programming $'**

**var1 db 'Name : MD Kamruzzaman Raihan $'**

**var2 db 'ID : CSE2103024039 $'**

**var3 db 'Semester: Fall-23 $'**

****

**.code**

**main proc**

**mov ax,@data**

**mov ds,ax**

**; first message**

**mov ah,9**

**lea dx,var**

**int 21h**

**call NewLine ; procedure call first time**

**; Second message**

**mov dx,offset var1**

**mov ah,9**

**lea dx,var1**

**int 21h**

**call NewLine ; procedure call second time**

**; Third message**

**mov ah,9**

**lea dx,var2**

**int 21h**

**call NewLine ; procedure call third time**

**; Four message**

**mov ah,9**

**lea dx,var3**

**int 21h**

**mov ah,4ch ; terminal the program**

**int 21h**

**main endp**

**; Procedure**

**NewLine proc**

**mov dx,10**

**mov ah,2**

**int 21h**

**mov dx,13**

**mov ah,2**

**int 21h**

**NewLine endp**

**;end main**

**ret**

**1.8 Stack operation**

**Task No. : 08**

**; String reverse using push, pop operation.**

**include emu8086.inc**

**.model small**

**.stack 100h**

**.data**

**string db "Raihan"**

**.code**

**main proc**

**mov ax,@data**

**mov ds,ax**

**; connection or pointing or store the base address**

**mov si,offset string**

**mov cx,6 ; length of string is 6.**

**; push operation**

**Lp\_push:**

**mov bx,[si]**

**push bx ; push operation (store or push in the stack)**

**; n <-- Last store on the stack**

**; a .**

**; h .**

**; i .**

**; a .**

**; R <-- first store on the stack**

**inc si**

**loop Lp\_push**

**printn**

**print "Reverse of the string; "**

**mov si,offset string**

**mov cx,6**

**Lp\_pop: ; pop operation (pop in the stack)**

**; n <-- first out on the stack**

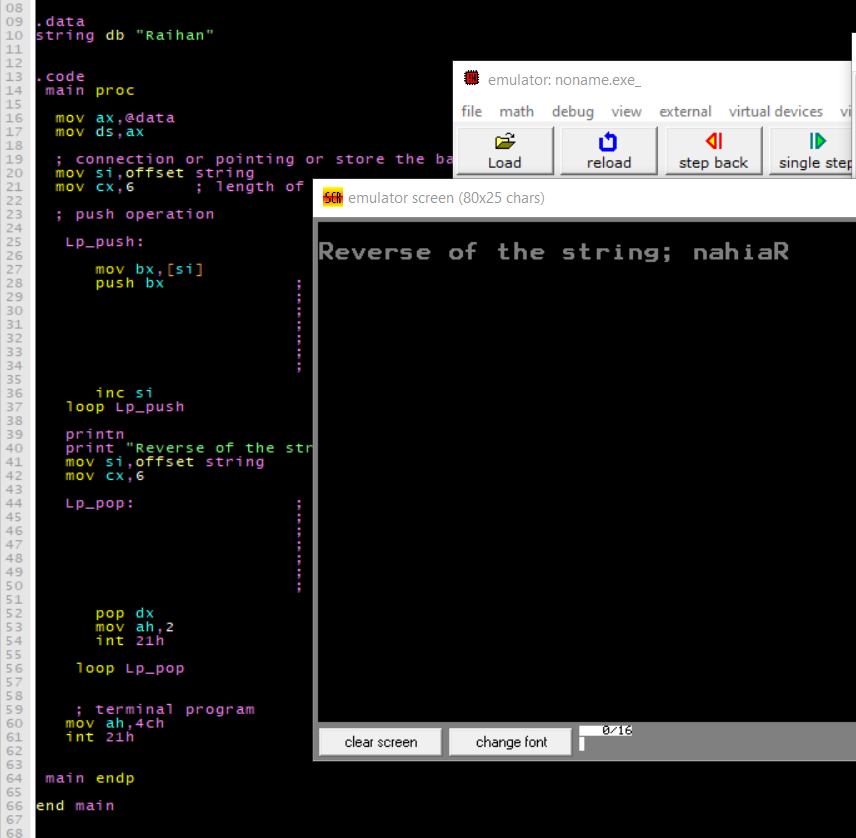
**; a .**

**; h .**

**; i .**

**; a .**

**; R <-- Last out on the stack**

** pop dx**

**mov ah,2**

**int 21h**

**loop Lp\_pop**

**; terminal program**

**mov ah,4ch**

**int 21h**

**main endp**

**end main**