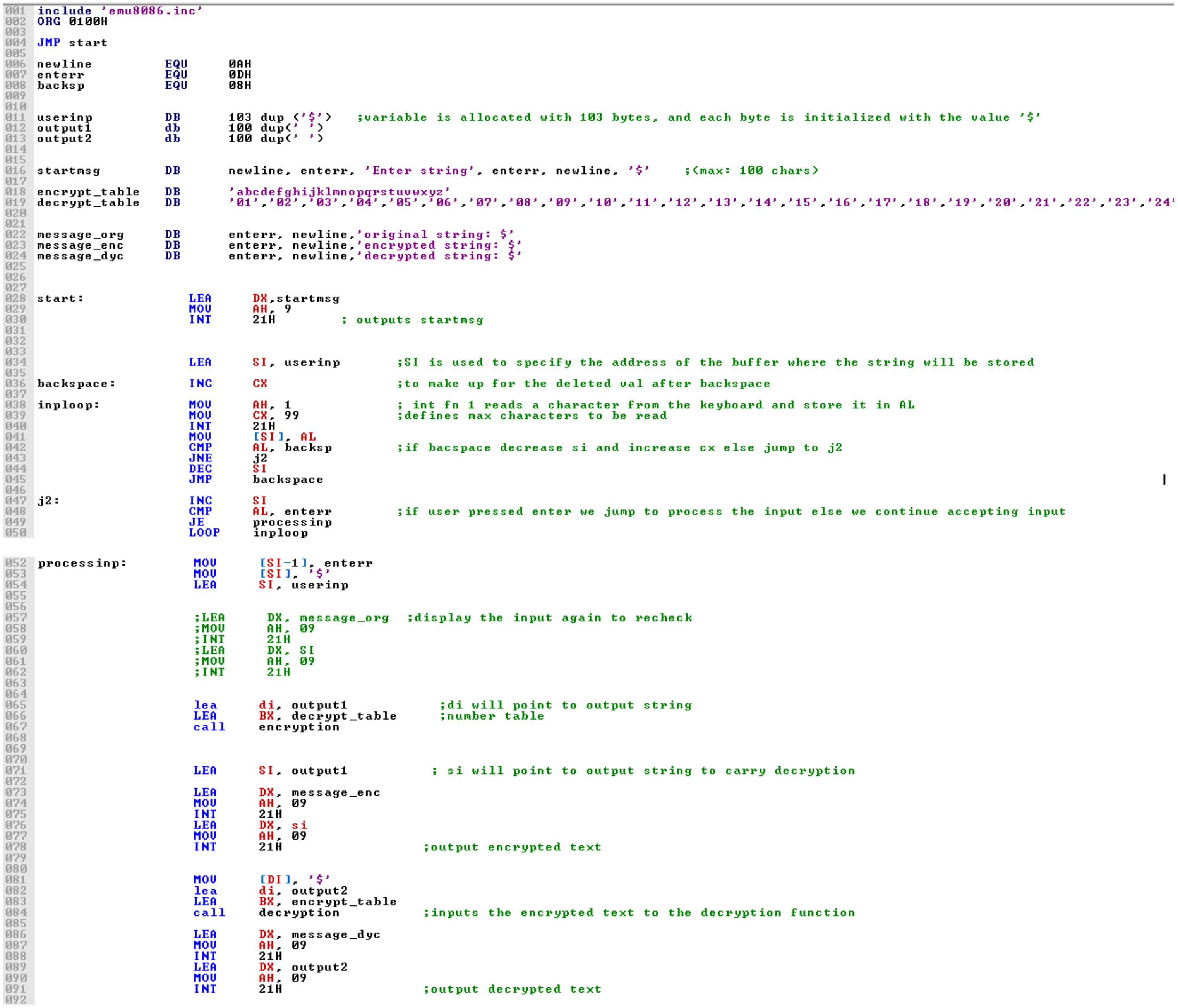
MICROPROCESSOR SYSTEMS

# PROJECT 9: Mononumeric substitution encryption

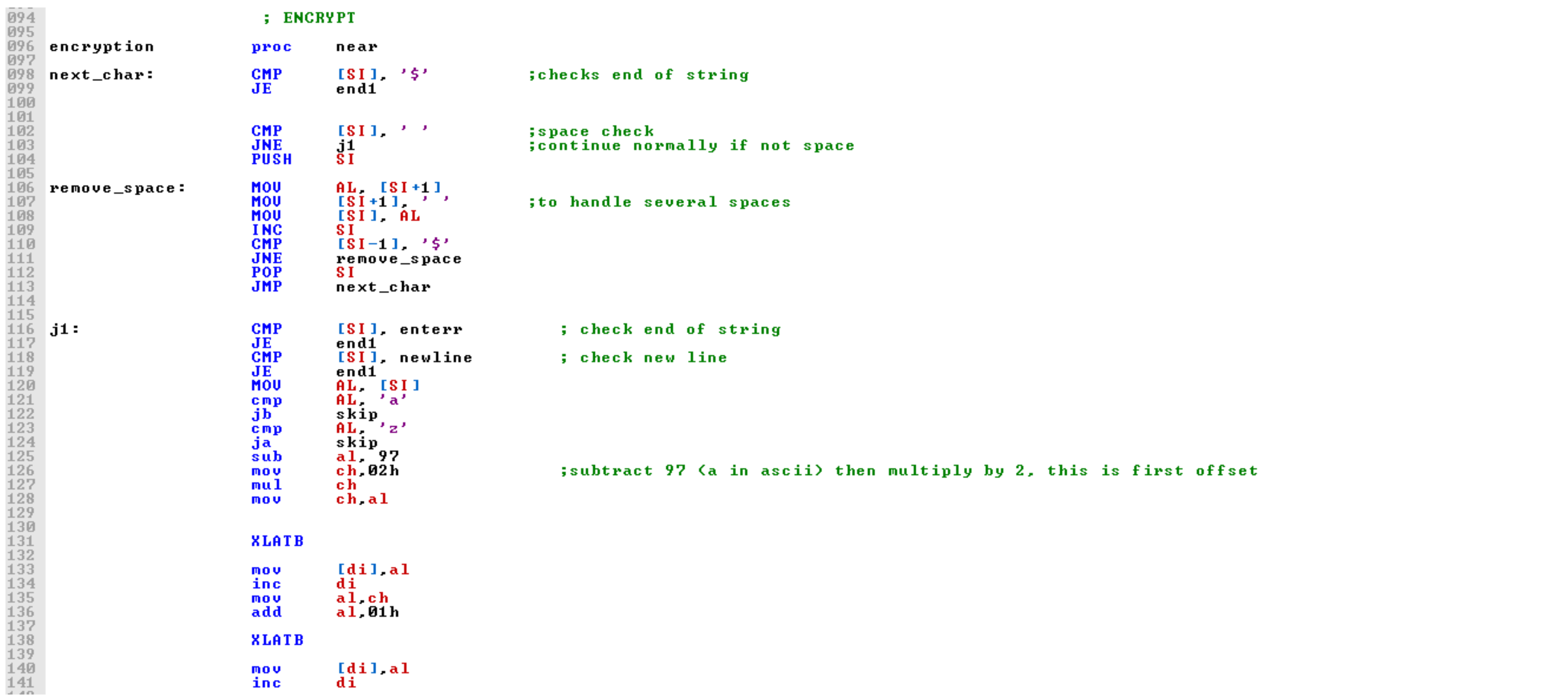
Mervat Tamer Ahmed

ID: 7779

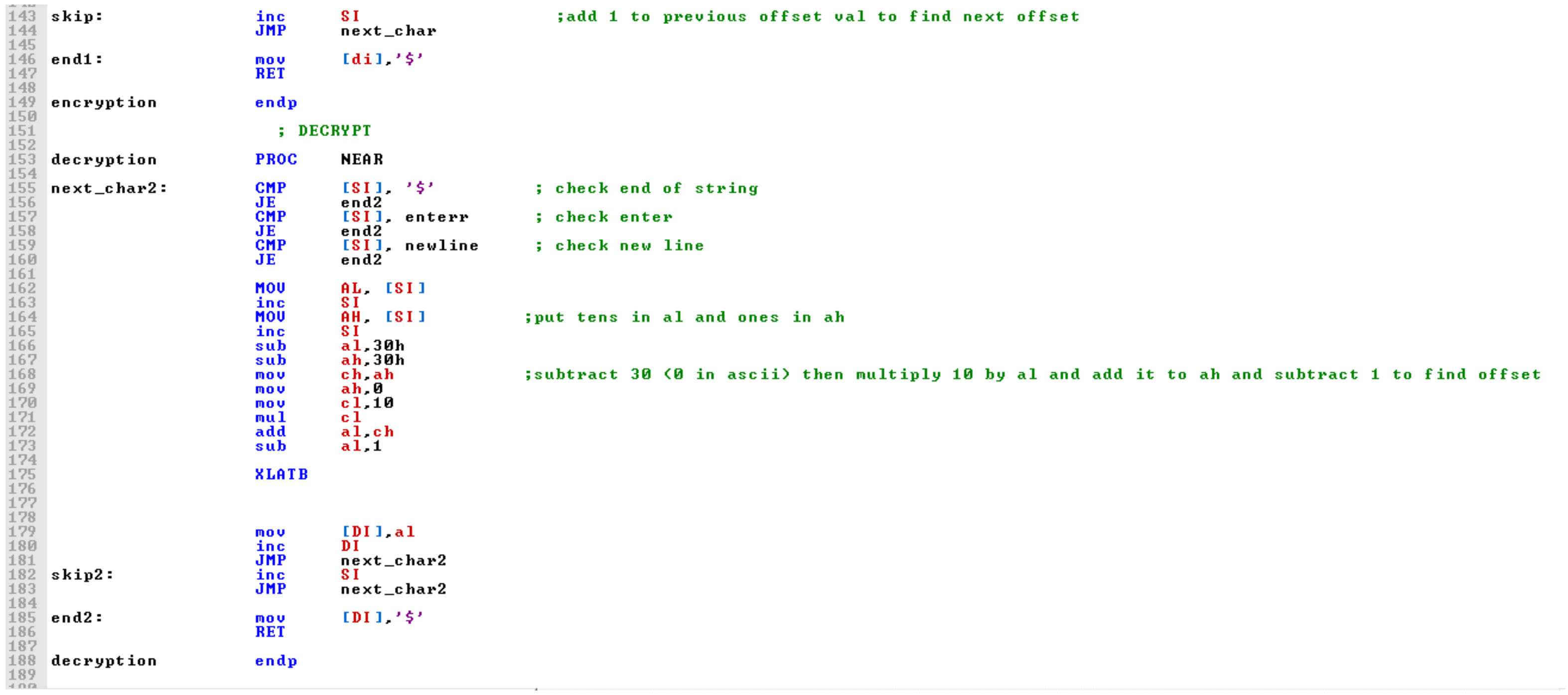
## CODE



```
include 'emu8086.inc'
ORG 0100H

JMP start

newline         EQU     0AH
enterr          EQU     0DH
backsp          EQU     08H


userinp         DB      103 dup ('$')   ;variable is allocated with 103 bytes, and each byte is initialized with the value '$'
output1         db      100 dup(' ')
output2         db      100 dup(' ')


startmsg        DB      newline, enterr, 'Enter string', enterr, newline, '$'    ;(max: 100 chars)

encrypt_table   DB      'abcdefghijklmnopqrstuvwxyz'
decrypt_table   DB      '01','02','03','04','05','06','07','08','09','10','11','12','13','14','15','16','17','18','19','20','21','22','23','24'


message_org     DB      enterr, newline,'original string: $'
message_enc     DB      enterr, newline,'encrypted string: $'
message_dyc     DB      enterr, newline,'decrypted string: $'



start:                  LEA     DX,startmsg
                        MOV     AH, 9
                        INT     21H         ; outputs startmsg



                        LEA     SI, userinp         ;SI is used to specify the address of the buffer where the string will be stored

backspace:              INC     CX                  ;to make up for the deleted val after backspace

inploop:                MOV     AH, 1               ; int fn 1 reads a character from the keyboard and store it in AL
                        MOV     CX, 99              ;defines max characters to be read
                        INT     21H
                        MOV     [SI], AL
                        CMP     AL, backsp          ;if bacspace decrease si and increase cx else jump to j2
                        JNE     j2
                        DEC     SI
                        JMP     backspace

j2:                     INC     SI
                        CMP     AL, enterr          ;if user pressed enter we jump to process the input else we continue accepting input
                        JE      processinp
                        LOOP    inploop

processinp:              MOV     [SI-1], enterr
                         MOV     [SI], '$'
                         LEA     SI, userinp


                         ;LEA      DX, message_org   ;display the input again to recheck
                         ;MOV      AH, 09
                         ;INT      21H
                         ;LEA      DX, SI
                         ;MOV      AH, 09
                         ;INT      21H


                         lea     di, output1             ;di will point to output string
                         LEA     BX, decrypt_table       ;number table
                         call    encryption



                         LEA     SI, output1            ; si will point to output string to carry decryption

                         LEA     DX, message_enc
                         MOV     AH, 09
                         INT     21H
                         LEA     DX, si
                         MOV     AH, 09
                         INT     21H                   ;output encrypted text


                         MOV     [DI], '$'
                         lea     di, output2
                         LEA     BX, encrypt_table
                         call    decryption            ;inputs the encrypted text to the decryption function

                         LEA     DX, message_dyc
                         MOV     AH, 09
                         INT     21H
                         LEA     DX, output2
                         MOV     AH, 09
                         INT     21H                   ;output decrypted text

```



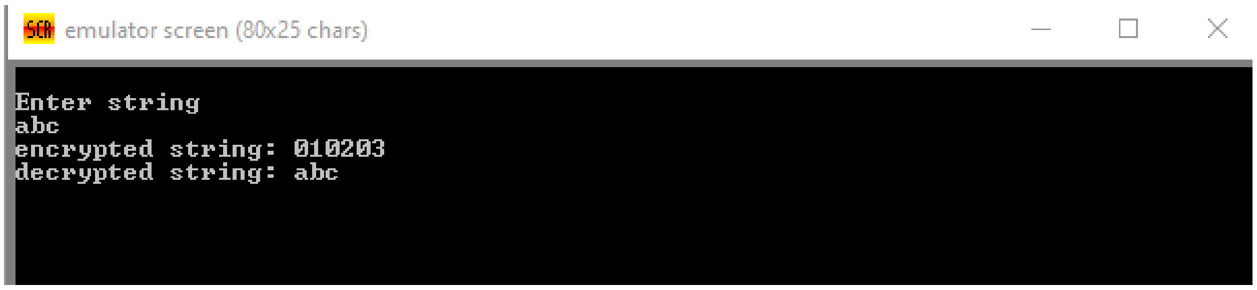
```
                          ; ENCRYPT

encryption              proc     near

next_char:              CMP      [SI], '$'          ;checks end of string
                        JE       end1


                        CMP      [SI], ' '          ;space check
                        JNE      j1                 ;continue normally if not space
                        PUSH     SI

remove_space:           MOV      AL, [SI+1]
                        MOV      [SI+1], ' '        ;to handle several spaces
                        MOV      [SI], AL
                        INC      SI
                        CMP      [SI-1], '$'
                        JNE      remove_space
                        POP      SI
                        JMP      next_char


j1:                     CMP      [SI], enterr          ; check end of string
                        JE       end1
                        CMP      [SI], newline         ; check new line
                        JE       end1
                        MOV      AL, [SI]
                        cmp      AL, 'a'
                        jb       skip
                        cmp      AL, 'z'
                        ja       skip
                        sub      al, 97
                        mov      ch,02h                ;subtract 97 (a in ascii) then multiply by 2, this is first offset
                        mul      ch
                        mov      ch,al


                        XLATB

                        mov      [di],al
                        inc      di
                        mov      al,ch
                        add      al,01h

                        XLATB

                        mov      [di],al
                        inc      di
```



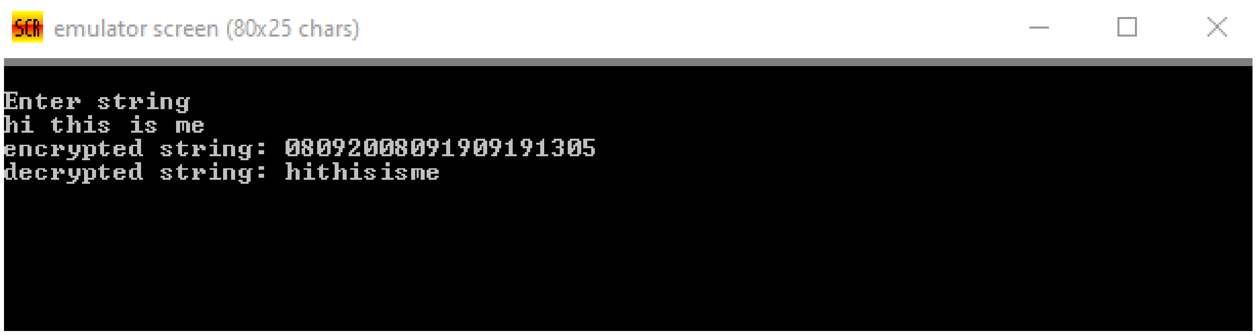
```
skip:                   inc      SI                    ;add 1 to previous offset val to find next offset
                        JMP      next_char

end1:                   mov      [di],'$'
                        RET

encryption              endp

                          ; DECRYPT

decryption              PROC     NEAR

next_char2:             CMP      [SI], '$'          ; check end of string
                        JE       end2
                        CMP      [SI], enterr       ; check enter
                        JE       end2
                        CMP      [SI], newline      ; check new line
                        JE       end2

                        MOV      AL, [SI]
                        inc      SI
                        MOV      AH, [SI]          ;put tens in al and ones in ah
                        inc      SI
                        sub      al,30h
                        sub      ah,30h
                        mov      ch,ah             ;subtract 30 (0 in ascii) then multiply 10 by al and add it to ah and subtract 1 to find offset
                        mov      ah,0
                        mov      cl,10
                        mul      cl
                        add      al,ch
                        sub      al,1

                        XLATB



                        mov      [DI],al
                        inc      DI
                        JMP      next_char2
skip2:                  inc      SI
                        JMP      next_char2

end2:                   mov      [DI],'$'
                        RET

decryption              endp
```

The previous code accepts text input, omits the spaces then encrypts the text with the encryption table, outputs it. It then decrypts the cipher text that was output again and outputs the decrypted (original text).

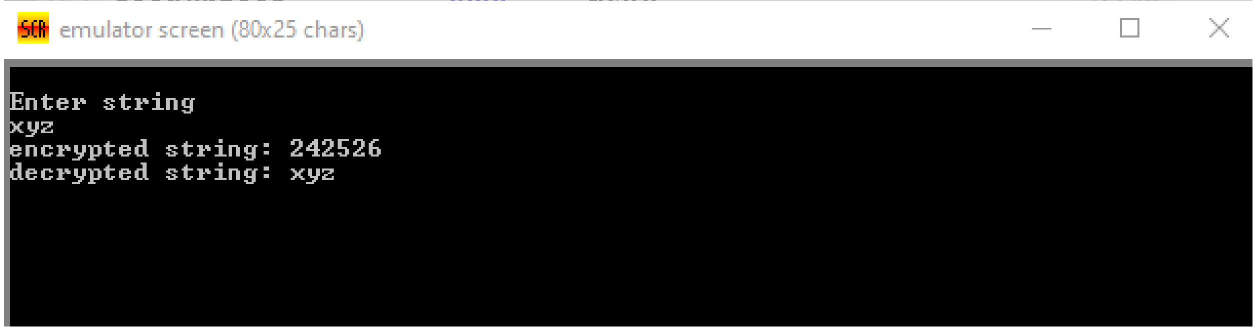
## SAMPLE RUNS:


emulator screen (80x25 chars)

```
Enter string
abc
encrypted string: 010203
decrypted string: abc
```


emulator screen (80x25 chars)

```
Enter string
hi this is me
encrypted string: 080920080919091913O5
decrypted string: hithisisme
```


emulator screen (80x25 chars)

```
Enter string
xyz
encrypted string: 242526
decrypted string: xyz
```