

MICROPROCESSOR SYSTEMS

PROJECT 9: Mononumeric substitution encryption

Mervat Tamer Ahmed

ID: 7779

CODE

```
001 include 'emu8086.inc'
002 ORG 0100H
003
004 JMP start
005
006 newline EQU 0AH
007 enterr EQU 0DH
008 backsp EQU 08H
009
010
011 userinp DB 103 dup('<del>') ;variable is allocated with 103 bytes, and each byte is initialized with the value '<del>'
012 output1 db 100 dup('<del>')
013 output2 db 100 dup('<del>')
014
015
016 startmsg DB newline, enterr, 'Enter string', enterr, newline, '$' ;<max: 100 chars>
017
018 encrypt_table DB 'abcdefghijklmnopqrstuvwxyz'
019 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'
020
021
022 message_org DB enterr, newline, 'original string: $'
023 message_enc DB enterr, newline, 'encrypted string: $'
024 message_dyc DB enterr, newline, 'decrypted string: $'
025
026
027
028 start: LEA DX, startmsg
029 MOV AH, 9
030 INT 21H ; outputs startmsg
031
032
033
034 LEA SI, userinp ;SI is used to specify the address of the buffer where the string will be stored
035
036 backspace: INC CX ;to make up for the deleted val after backspace
037
038 inploop: MOV AH, 1 ; int fn 1 reads a character from the keyboard and store it in AL
039 MOV CX, 99 ;defines max characters to be read
040 INT 21H
041 MOV [SI], AL
042 CMP AL, backsp ;if backspace decrease si and increase cx else jump to j2
043 JNE j2
044 DEC SI
045 JMP backspace
046
047 j2: INC SI
048 CMP AL, enterr ;if user pressed enter we jump to process the input else we continue accepting input
049 JE processing
050 LOOP inploop
051
052
053 processing: MOV [SI-1], enterr
054 MOV [SI], '$'
055 LEA SI, userinp
056
057 ;LEA DX, message_org ;display the input again to recheck
058 ;MOV AH, 09
059 ;INT 21H
060 ;LEA DX, SI
061 ;MOV AH, 09
062 ;INT 21H
063
064 lea di, output1 ;di will point to output string
065 LEA BX, decrypt_table ;number table
066 call encryption
067
068
069
070 LEA SI, output1 ; si will point to output string to carry decryption
071
072 LEA DX, message_enc
073 MOV AH, 09
074 INT 21H
075 LEA DX, si
076 MOV AH, 09
077 INT 21H ;output encrypted text
078
079
080 MOV [DI], '$'
081 lea di, output2
082 LEA BX, encrypt_table
083 call decryption ;inputs the encrypted text to the decryption function
084
085 LEA DX, message_dyc
086 MOV AH, 09
087 INT 21H
088 LEA DX, output2
089 MOV AH, 09
090 INT 21H ;output decrypted text
091
092
```

```

094         ; ENCRYPT
095
096 encryption    proc    near
097 next_char:    cmp     [SI], '$'    ;checks end of string
098              je      end1
099
100
101
102              cmp     [SI], ' '    ;space check
103              jne     j1           ;continue normally if not space
104              push    SI
105
106 remove_space:  mov     AL, [SI+1]
107              mov     [SI+1], '',   ;to handle several spaces
108              mov     [SI], AL
109              inc     SI
110              cmp     [SI-1], '$'
111              jne     remove_space
112              pop     SI
113              jmp     next_char
114
115
116 j1:           cmp     [SI], enter    ; check end of string
117              je      end1
118              cmp     [SI], newline   ; check new line
119              je      end1
120              mov     AL, [SI]
121              cmp     AL, 'a'
122              jnb     skip
123              cmp     AL, 'z'
124              ja      skip
125              sub     al, 97
126              mov     ch, 02h         ;subtract 97 (a in ascii) then multiply by 2, this is first offset
127              mul     ch
128              mov     ch, al
129
130
131             XLATB
132
133             mov     [di], al
134             inc     di
135             mov     al, ch
136             add     al, 01h
137
138             XLATB
139
140             mov     [di], al
141             inc     di
142
143 skip:         inc     SI             ;add 1 to previous offset val to find next offset
144             jmp     next_char
145
146 end1:         mov     [di], '$'
147             rei
148
149 encryption    endp
150
151         ; DECRYPT
152
153 decryption    PROC    NEAR
154 next_char2:    cmp     [SI], '$'    ; check end of string
155              je      end2
156              cmp     [SI], enter    ; check enter
157              je      end2
158              cmp     [SI], newline   ; check new line
159              je      end2
160
161
162              mov     AL, [SI]
163              inc     SI
164              mov     AH, [SI]       ;put tens in al and ones in ah
165              inc     SI
166              sub     al, 30h
167              sub     ah, 30h
168              mov     ch, ah         ;subtract 30 (0 in ascii) then multiply 10 by al and add it to ah and subtract 1 to find offset
169              mov     ah, 0
170              mov     cl, 10
171              mul     cl
172              add     al, ch
173              sub     al, 1
174
175             XLATB
176
177
178
179             mov     [DI], al
180             inc     DI
181             jmp     next_char2
182 skip2:        inc     SI
183             jmp     next_char2
184
185 end2:         mov     [DI], '$'
186             rei
187
188 decryption    endp
189
190

```

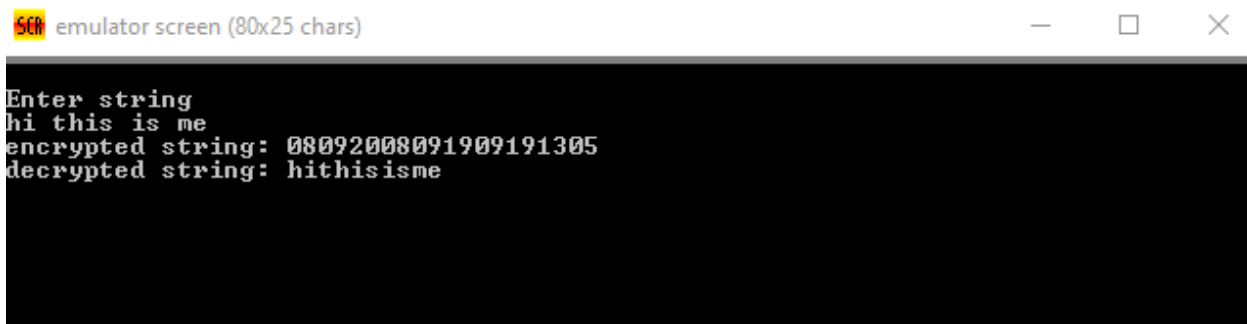
The previous code accepts text input, omits the spaces then encrypts the text with the encryption table, outputs it. It then decrypts the cipher text that was output again and outputs the decrypted (original text).

SAMPLE RUNS:



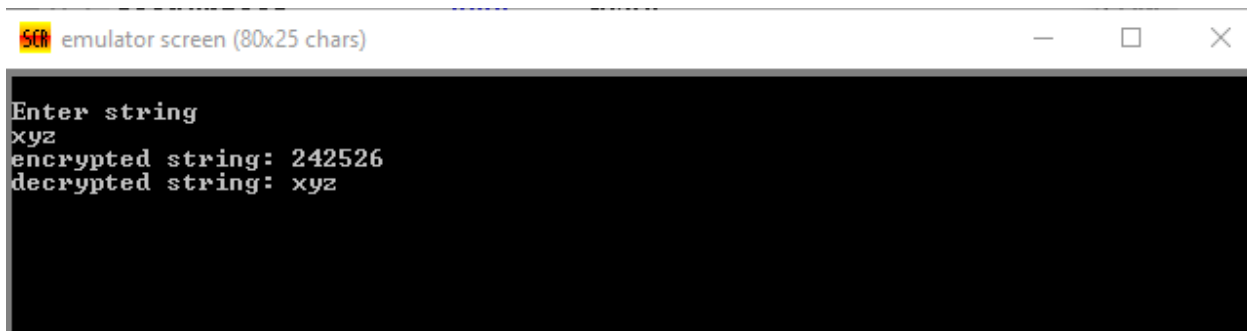
```
emulator screen (80x25 chars)

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



```
emulator screen (80x25 chars)

Enter string
hi this is me
encrypted string: 08092008091909191305
decrypted string: hithisisme
```



```
emulator screen (80x25 chars)

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