**Project**

**Code:**

.model small

.stack 100h

.data

; Define lookup tables for encryption and decryption

table1 db 97 dup(' '), 'klmnxyzabcopqrstvuwdefghij'

table2 db 97 dup(' '), 'hijtuvwxyzabcdklmnoprqsefg'

; Messages

welcome\_msg DB 'Welcome to the Encryption/Decryption Program!$', 0DH, 0AH, '$'

msg1 DB 'Enter the message: $'

msg2 DB 'Encrypted message: $'

msg3 DB 'Decrypted message: $'

n\_line DB 0DH, 0AH, '$' ; for new line

; Buffers

str db 256 dup('$') ; input buffer (first byte will store max length)

enc\_str db 256 dup('$') ; buffer for encrypted string

dec\_str db 256 dup('$') ; buffer for decrypted string

.code

start:

; Set up the data segment

mov ax, @data

mov ds, ax

; Print welcome message

lea dx, welcome\_msg

mov ah, 09h

int 21h

; Print message to prompt for input

lea dx, msg1

mov ah, 09h

int 21h

; Read user input (Using DOS function 0Ah: Buffered input)

lea dx, str ; point to buffer for input

mov ah, 0Ah ; DOS function: Buffered input

int 21h

; Print newline

lea dx, n\_line

mov ah, 09h

int 21h

; Encrypt message using table1

lea bx, table1 ; point bx to table1

lea si, str+2 ; skip over the length byte and go to the first character

lea di, enc\_str

call parse

; Print encrypted message

lea dx, msg2

mov ah, 09h

int 21h

lea dx, enc\_str

mov ah, 09h

int 21h

lea dx, n\_line

mov ah, 09h

int 21h

; Decrypt message using table2

lea bx, table2 ; point bx to table2

lea si, enc\_str

lea di, dec\_str

call parse

; Print decrypted message

lea dx, msg3

mov ah, 09h

int 21h

lea dx, dec\_str

mov ah, 09h

int 21h

lea dx, n\_line

mov ah, 09h

int 21h

; Wait for any key before exiting

mov ah, 00h

int 16h

; Terminate program

mov ax, 4C00h

int 21h

; Subroutine to encrypt/decrypt

; Parameters:

; si - address of string to encrypt/decrypt

; bx - table to use (either table1 or table2)

parse proc near

next\_char:

cmp [si], '$' ; End of string check

je end\_of\_string

cmp [si], ' ' ; Skip spaces

je skip

mov al, [si] ; Load current character

; Check if it's a lowercase letter (a-z)

cmp al, 'a'

jb check\_uppercase

cmp al, 'z'

ja skip

; Handle lowercase letters (a-z)

; Use xlatb to map the character using the table pointed by bx

xlatb

mov [di], al

inc di

jmp skip

check\_uppercase:

; Check if it's an uppercase letter (A-Z)

cmp al, 'A'

jb skip

cmp al, 'Z'

ja skip

; Handle uppercase letters (A-Z)

; Convert uppercase to lowercase for table lookup by adding 32 ('A' to 'a')

add al, 20h ; Convert uppercase to lowercase ('A' to 'a')

xlatb

sub al, 20h ; Convert back to uppercase

mov [di], al

inc di

skip:

inc si

jmp next\_char

end\_of\_string:

inc si

mov [si], '$' ; Null-terminate the string

ret

parse endp

**output:**

