```
ORG 100H
#DS=0000H#
include 'emu8086.inc'
MOV DX,offset msg0    ; DX holds the address of the string
MOV AH,09h
INT 21h             ; output the message

input:
MOV AH, 0           ;get keystoke from keyboard
INT 16h

MOV  AH, 0Eh        ;this functions display a character on the screen
INT  10h

E:
CMP  AL, 45h        ; Check if the user entered E
JNZ  D              ; Jump if not equal to D
JZ   Encryption     ; Jump if equal to Encryption

D:
CMP  AL, 44h        ; Check if the user entered D
JNZ  TryAgain       ; Jump if not equal to TryAgain
JZ   Encryption     ; Jump if equal to Decryption

TryAgain:
MOV DX, offset msg2
MOV AH, 09h
INT 21h             ; Output the message msg2
```

```
JMP input


msg0 db "  Enter 'E' for encryption or 'D' for decryption: ", 0Dh,0Ah
db , 0Dh,0Ah, "$"

msg1 db "  Enter '4' for 4 bit encryption/decrpytion or '8' for 8 bit encryption/decryption or 'f' for 16 bit
encryptoin/decryption " , 0Dh,0Ah
db , 0Dh,0Ah, "$"

msg2 db "  Enter 'E' or 'D' only " , 0Dh,0Ah
db , 0Dh,0Ah, "$"

msg3 db "  Enter '4' or '8' or 'f' only " , 0Dh,0Ah
db , 0Dh,0Ah, "$"

msg9 db "  Enter '4' for 4 bit decryption or '8' for 8 bit decryption or 'f' for 16 bit decryption" , 0Dh,0Ah
db , 0Dh,0Ah, "$"


Encryption:
;*************************************************************************************
****************************************

MOV DX,offset msg1          ; DX holds the address of the string
MOV AH,09h
INT 21h

input1:
```

```
MOV AH, 0                   ;get keystoke from keyboard
INT 16h

MOV  AH, 0Eh                ;this functions display a character on the screen
INT  10h

bit8:
CMP    AL,38h               ; Check if the user entered 8
JNZ    bit4                 ; if not equal 8 jump tryagain1
JZ     Encryption8bit       ; if equal jump encryption8bit

bit4:
CMP    AL,34h
JNZ    bit16
JZ     Encryption4bit

bit16:
CMP    AL,66h               ; enter "f" for 16 bit encryption
JNZ    TryAgain1
JZ     Encryption16bit

TryAgain1:
MOV    DX,offset msg3
MOV    AH,09h
INT    21h
JMP    input1
```

```
Encryption8bit:
;*********************************************************************************
****************

JMP start              ; skip over the declarations and data

buffer db "empty buffer"

size = $ - offset buffer  ; declare constant

msg5  db 13,10, "Enter the word you want to encrypt/decrypt : ", 0

MOV AX,0000h

MOV DS,AX

MOV CX,AX

off=0000h              ;offset of di

start:

LEA     SI, msg5       ;stores the address of a memory variable in a  general register. Print message msg5
or or:  MOV  SI, OFFSET msg5

CALL    print_string    ; macro with 1 parameter, prints out a string.

; get string to ds:di

LEA   DI,[off]    ; buffer offset.

MOV   DX, 16      ; buffer size.

CALL  get_string
```

```
MOV AL,[DI+4]
MOV AH,[DI+6]
MOV CL,[DI+1]
MOV CH,[DI+3]

        MOV [DI+1],AL
        MOV [DI+3],AH
        MOV [DI+4],CL
        MOV [DI+6],CH


putc    0Dh
putc    10          ; next line.

print "The encrypted/decrypted word is : "

; print string in ds:si using procedure:
MOV  SI, DI
CALL print_string
JMP  EXIT


Encryption4bit:
;*************************************************************************
*****************


JMP start1              ; skip over the declarations and data
```

```
buffer1 db "empty buffer"
size1 = $ - offset buffer  ; declare constant
msg6  db 13,10, "Enter the word you want to encrypt/decrypt: ", 0

MOV AX,0000h
MOV DS,AX
MOV CX,AX

off=0000h              ;offset of di

start1:

LEA     SI, msg6       ;stores the address of a memory variable in a  general register. Print message msg5 or or:  MOV  SI, OFFSET msg5
CALL    print_string    ; macro with 1 parameter, prints out a string.

; get string to ds:di
LEA   DI,[off]    ; buffer offset.
MOV   DX, 16      ; buffer size.
CALL  get_string

MOV AL,[DI+1]
MOV AH,[DI+2]
```

```
        MOV [DI+2],AL
        MOV [DI+1],AH


putc    0Dh
putc    10          ; next line.

print "The encrypted/decrypted word is : "

; print string in ds:si using procedure:
MOV  SI, DI
CALL print_string
JMP  EXIT

Encryption16bit:
;*************************************************************************

JMP start15; skip over the declarations and data

msg23 db 13,10, "Enter the word you want to encrypt/decrypt: ", 0


MOV AX,0000h
MOV DS,AX
MOV CX,AX
off=0000h ;offset of di
```

```
start15:
; print a welcome message:
LEA     SI, msg23
CALL    print_string

; get string to ds:di
LEA     DI,[off]    ; buffer offset.
MOV     DX, 16      ; buffer size.
CALL    get_string

MOV AL,[DI+1]
MOV AH,[DI+8]
MOV [DI+8],AL
MOV [DI+1],AH

MOV AL,[DI+2]
MOV AH,[DI+4]
MOV [DI+4],AL
MOV [DI+2],AH

MOV AL,[DI+3]
MOV AH,[DI+12]
MOV [DI+12],AL
MOV [DI+3],AH

MOV AL,[DI+5]
MOV AH,[DI+10]
MOV [DI+10],AL
MOV [DI+5],AH
```

```
MOV AL,[DI+7]
MOV AH,[DI+14]
MOV [DI+14],AL
MOV [DI+7],AH

MOV AL,[DI+11]
MOV AH,[DI+13]
MOV [DI+13],AL
MOV [DI+11],AH

putc    0Dh
putc    10 ; next line.

; print using macro:
print "the encrypted/decrypted word is: "

; print string in ds:si using procedure:
MOV     SI, DI
CALL    print_string
JMP EXIT

EXIT:
HLT
DEFINE_PRINT_STRING
DEFINE_GET_STRING
RET
```