**Chapter 8**
**Qno1**

```
[org 0x0100]

        jmp start
isTSR:  dw 0
oldISR: dd 0
;--------------------------------------------------------------------------

myISR:          cmp ah, 0x31                                    ;Agar tw 0x31 wali service call huwi tw kuch kaam karo, nahi tw humein koi kaam karney ki zaroorat
                                                                ;hi nahi because hum ne tw sirf 31h wali service pe kaam karna hai

                jnz chain

                cmp word [cs:isTSR], 0          ;Agar tw humari myISR TSR ban chuki hai, phir ab kisi aur naye program ko TSR nahi banney dena
                jz makeitTSR                    ;Lekin agar humari myISR TSR nahi bani, tw pehley usey TSR banao


                mov ah, 0x4c

chain:          jmp far [cs:oldISR]



makeitTSR:      mov word [cs:isTSR], 1
                jmp far [cs:oldISR]


;--------------------------------------------------------------------------

start:          xor ax,ax
                mov es,ax

                ;Saving the OLD ISR
                mov ax, [es:21h*4]
                mov [oldISR], ax

                mov ax, [es:21h*4 + 2]
                mov [oldISR + 2], ax

                ;Hooking our ISR
                mov word [es:21h*4], myISR
                mov word [es:21h*4+2], cs

                mov dx, start
                add dx, 15
                mov cl, 4
                shr dx, cl

exit:           mov ax, 0x3100
                int 21h
```

**qno2**

```
[org 0x0100]

        jmp start


address:        dd 0

;-----------------------------------------------------------------------------------------------------------------

;Clear Screen
clrscr:                 push bp
                                mov bp, sp

                                pusha
                                push es

                                mov ax, 0xb800
                                mov es, ax
                                xor di,di
                                mov ax,0x0720
                                mov cx,2000

                                cld
                                rep stosw

                                pop es


return:                 mov ax, 0
                                mov cx, 0

                                cmp     ax, [bp+4]
                                jz nearReturn

                                cmp ax, [bp+6]
                                jz farReturn

                                cmp ax, [bp+8]
                                jz interruptReturn


nearReturn:                     popa
                                        pop bp
                                        ret 2

farReturn:                      popa
                                        pop bp
                                        retf 2

interruptReturn: mov ax, [bp + 6]
                                        mov [bp + 8], ax

                                        mov ax, [bp + 4]
                                        mov [bp + 6], ax
```

```
                                        mov ax, [bp + 2]
                                        mov [bp + 4], ax

                                        popa
                                        pop bp

                                        add sp, 2

                                        iret
;-------------------------------------------------------------------------------------------------------------------



start:          xor ax, ax
                        mov es, ax

                        ;Hooking the interrupt
                        mov word [es:80h*4], clrscr
                        mov word [es:80h*4+2], cs

                        ;Saving address for far call
                        mov word [address], clrscr
                        mov word [address + 2], cs

                        push 0
                        call clrscr                             ;Near Call

                        push 0
                        call far [address]                      ;Far Call

                        push 0
                        int 80h                                 ;Interrupt Call (Extended Far Call)

                        mov ax, 0x4c00
                        int 21h
```

## Qno3

```
[org 0x0100]

        jmp start


XISR_Offset:            dw  0x0000
XISR_Segment:           dw  0x0000
N:                                      dw  0x80

;-------------------------------------------------------------------------------------------------------------------


hooker:         push bp
                        mov bp, sp
```

```
sub sp, 4                                   ;Making two local variables, one for old offset and one for old segment of the ISR previously hooked at N

pusha
push es


;bp - 2                                     ;Old segment
;bp - 4                                     ;Old offset

;bp + 4                                     ;XISR Offset
;bp + 6                                     ;XISR Segment
;bp + 8                                     ;Interrupt No. 'N'


xor ax, ax
mov es, ax


mov di, [bp + 8]            ;Interrupt No. 'N'


;First of all saving the offset, segment of the ISR previously hooked at N

shl di, 2                                   ;Multiplying by 4

;Saving the offset
mov bx, [es:di]
mov [bp - 4], bx

;Saving the segment
mov bx, [es:di + 2]
mov [bp - 2], bx



;Loading the segment of XISR in es
mov es, [bp + 6]


;Chaining the XISR to the old ISR previously hooked at N

mov bx, [bp + 4]            ;Offset of XISR

mov ax, [bp - 4]                    ;Offset
mov dx, [bp - 2]                    ;Segment


mov [es:bx + 2], ax
mov [es:bx + 4], dx


;Now hooking XISR at N
mov ax, 0
mov es, ax
```

```
                        mov di,[bp + 8]

                        shl di, 2                                   ;Multiplying by 4


                        mov ax,[bp + 4]             ;Offset of XISR
                        mov [es:di], ax
                        mov ax, [bp + 6]                    ;Segment of XISR
                        mov [es:di+ 2], ax


return:         pop es
                        popa

                        add sp, 4

                        pop bp
                        ret 6

;-------------------------------------------------------------------------------------------------------------------



;-------------------------------------------------------------------------------------------------------------------
XISR:   pushf
                        call 0:0

                        popf
                        ret


;-------------------------------------------------------------------------------------------------------------------

start:          push word [N]

                        mov  word [XISR_Offset], XISR
                        mov  word [XISR_Segment], cs

                        push word [XISR_Segment]
                        push word [XISR_Offset]

                        call hooker


                        mov ax, 0x4c00
                        int 21h
```

# CHAPTER NO 9

## Qno 3

```
;Write a program to make an asterisk travel the border of the screen,
;from upper left to upper right to lower right to lower left and back to upper left indefinitely.


[org 0x0100]

jmp start


start:          call clrscr

                call borderAsterisk

                mov ax, 0x4c00
                int 21h


;Clear Screen
clrscr:                 mov ax, 0xb800
                        mov es, ax
                        xor di,di
                        mov ax,0x0720
                        mov cx,2000

                        cld
                        rep stosw

                        ret


;Delay
delay:                  pusha
                        mov cx, 0xFFFF

b1:                     loop b1

                        popa
                        ret



borderAsterisk:         push bp
                        mov bp, sp
                        pusha


                        ;Loading the video memory
                        mov ax, 0xb800
                        mov es, ax

                        mov di, 0

                        mov ah, 01110000b
```

```
                                mov al, '*'

                                mov bh, 0x07
                                mov bl, 0x20

LefttoRight:        mov cx, 80

l1:                             mov [es:di], ax

                                call delay

                                mov [es:di], bx

                                call delay

                                add di, 2

                                loop l1

                                sub di, 2

RightToBottom:      mov cx, 25

l2:                             mov [es:di], ax

                                call delay

                                mov [es:di], bx

                                call delay

                                add di, 160

                                loop l2

                                sub di, 160

BottomToLeft:       mov cx, 80

l3:                             mov [es:di], ax

                                call delay

                                mov [es:di], bx

                                call delay

                                sub di, 2

                                loop l3

                                add di, 2
```

```
LefttoTop:          mov cx, 25

l4:                         mov [es:di], ax

                            call delay

                            mov [es:di], bx

                            call delay

                            sub di, 160

                            loop l4

                            add di, 160

                            ;Then repeat the whole process again resulting in an infinite loop
                            jmp LefttoRight


return:             popa
                            pop bp
                            ret




;---------------------------------------------------------------------------
; ALTERNATE SOLUTION
; Solution to this problem was developed by https://github.com/farhana1i
;---------------------------------------------------------------------------

; ; to display asterick movement every after 1 second
; [org 0x0100]

; jmp main

; seconds:    dw 0    ; number of seconds
; ticks:      dw 0    ; count of ticks
; isLeft:     db 0    ; left movement flag
; isRight:    db 0    ; right movement flag
; isTop:      db 0    ; up movement flag
; isBottom:   db 0    ; down movement flag
; col:        db 0    ; current row number
; row:        db 0    ; current column number

; ; to clear video screen
; clrscr:
; push    es
; push    ax
; push    di

; mov     ax, 0xb800
; mov     es, ax
; mov     di, 0
```

```
; nextchar:
; mov     word [es:di], 0x720
; add     di, 2
; cmp     di, 4000
; jne     nextchar

; pop     di
; pop     ax
; pop     es
; ret


; ; to print asteric
; ; DI == position
; printAsterick:
; push    ax
; push    es

; mov     ax, 0xb800
; mov     es, ax          ; points to video memory

; mov     word [es: di], 0x0720   ; clear previous location

; cmp     byte [col], 0
; JNE     nextCmp

; cmp     byte [row], 0
; JNE      checkUp
; mov     byte [isLeft], 1
; mov     byte [isRight], 0
; mov     byte [isTop], 0
; mov     byte [isBottom], 0
; jmp     update

; checkUp:
; cmp     byte [row], 24
; JNE     nextCmp
; mov     byte [isLeft], 0
; mov     byte [isRight], 0
; mov     byte [isTop], 1
; mov     byte [isBottom], 0
; jmp     update

; nextCmp:
; cmp     byte [col], 158
; JNE     update

; cmp     byte [row], 0
; JNE     checkRight
; mov     byte [isLeft], 0
; mov     byte [isRight], 0
; mov     byte [isTop], 0
; mov     byte [isBottom], 1
; jmp     update

; checkRight:
```

```
; cmp     byte [row], 24
; JNE     update
; mov     byte [isLeft], 0
; mov     byte [isRight], 1
; mov     byte [isTop], 0
; mov     byte [isBottom], 0
; jmp     update

; update:
; cmp     byte [isLeft], 1
; JNE     checkRightFlag
; add     di, 2
; add     byte [col], 2
; jmp     printScreen

; checkRightFlag:
; cmp     byte [isRight], 1
; JNE     checkUpFlag
; sub     di, 2
; sub     byte [col], 2
; jmp     printScreen

; checkUpFlag:
; cmp     byte [isTop], 1
; JNE     checkDownFlag
; sub     di, 160
; sub     byte [row], 1
; jmp     printScreen

; checkDownFlag:
; cmp     byte [isBottom], 1
; JNE     printScreen
; add     di, 160
; add     byte [row], 1
; jmp     printScreen


; printScreen:
; mov     ah, 0x07    ; attribute
; mov     al, '*'
; mov     word [es: di], ax

; pop es
; pop ax
; ret



; ; hook timer interrupt service routine
; timer:
; push    ax

; inc     word [cs: ticks]
; cmp     word [cs: ticks], 18        ; 18.2 ticks per second
; jne     exitTimer
```

```
; inc     word [cs: seconds]          ; increase total seconds by 1
; mov     word [cs: ticks], 0
; CALL    printAsterick

; exitTimer:
; mov     al, 0x20        ; send EOI
; out     0x20, al
; pop     ax
; iret


; main:
; ;call    clrscr      ; to clear screen
; mov     di, 0
; xor     ax, ax
; mov     es, ax

; ; hook interrupt
; cli
; mov     word [es: 8*4], timer
; mov     [es: 8*4+2], cs
; sti

; ; to make program TSR
; mov     dx, main
; add     dx, 15
; mov     cl, 4
; shr     dx, cl
; mov     ax, 0x3100
; INT     0x21
```

---

## Qno 8

```
; Solution to this problem was developed by https://github.com/farhana1i

[org 0x0100]

    jmp main

oldisr: dd 0        ; old isr offset and segment
buffer: times 2000 dw 0 ; buffer to save video memory

; to clear video screen
clrscr:
push    es
push    ax
push    di

mov     ax, 0xb800
mov     es, ax
mov     di, 0
nextchar:
mov     word [es:di], 0x720
add     di, 2
```

```
cmp     di, 4000
jne     nextchar

pop     di
pop     ax
pop     es
ret

; to add some delay
delay:
push    cx
push    di

mov     cx, 0xFF
delay1:
mov     di, 0xFFF
delay2:
dec     di
jnz     delay2
loop    delay1

pop     di
pop     cx
ret

; to store video memory in buffer
store_buffer:
push    bp
mov     bp, sp
push    ax
push    cx
push    si
push    di
push    es
push    ds

mov     ax, 0xb800  ; points to video memory
mov     ds, ax
mov     si, 0
mov     ax, cs
mov     es, ax
mov     di, buffer
mov     cx, 2000

cld
rep     movsw    ; move data from video memory to buffer

pop     ds
pop     es
pop     di
pop     si
pop     cx
pop     ax
pop     bp
ret
```

```
; load buffer
load_buffer:
push    bp
mov     bp, sp
push    ax
push    cx
push    si
push    di
push    es
push    ds

mov     ax, 0xb800      ; points to video memory
mov     es, ax
mov     di, 0
; points to buffer
mov     ax, cs
mov     ds, ax
mov     si, buffer
mov     cx, 2000

cld
rep     movsw       ; load buffer in video memory

pop     ds
pop     es
pop     di
pop     si
pop     cx
pop     ax
pop     bp
ret


; hook key board interrupt with interrupt chaining
kbISR:
push    ax
in      al, 0x60     ; read a char from keyboard

cmp     al, 00011101b  ; snap code of ctrl == 29
JNE     nextCmp

CALL    store_buffer   ; store video memory in a buffer
CALL    clrscr         ; clear screen
jmp     exit

nextCmp:
cmp     al, 10011101b   ; snap code of ctrl == 29
JNE     noMatch

CALL    delay          ; add some delay
CALL    load_buffer    ; load buffer in video memory
jmp     exit

noMatch:
pop     ax
jmp     far [cs:oldisr] ; CALL the original ISR
```

```
exit:
mov     al, 0x20        ; send EOI
out     0x20, al
pop     ax
iret



main:
xor     ax, ax
mov     es, ax
; save old keyboard isr
mov     ax, [es:9*4]
mov     [oldisr], ax
mov     ax, [es:9*4+2]
mov     [oldisr+2], ax

; hook keyboard interrupt
cli
mov     word [es:9*4], kbISR
mov     [es:9*4+2], cs
sti

; to make program TSR
mov     dx, main
add     dx, 15
mov     cl, 4
shr     dx, cl
mov     ax, 0x3100
INT     0x21
```

## ***Qno11***

```
;Write a TSR to calculate the current typing speed of the user.
;Current typing speed is the number of characters typed by the user in the last five seconds.
;The speed should be represented by printing asterisks at the right border (80th column) of the screen
;starting from the upper right to the lower right corner (growing downwards).
;Draw n asterisks if the user typed n characters in the last five seconds. The count should be updated
every second.

;-----------------------------------------------------------------------------------------------------------------------------------


;CTS - Current Typing Speed




;-----------------------------------------------------------------------------------------------------------------------------------
[org 0x0100]

            jmp start


ms:                 dw 0                                                ;Milli seconds
```

```
count:  dw 0, 0, 0, 0, 0                ;Count of the characters typed

tCount: dw -1                   ;Note: The tCount of first second is initialized to -1 for one time because when you type the command and press ENTER
                                        ;       then the program gets loaded. And it takes you a few milliseconds to release the ENTER key
                                        ;       and since the program was loaded before, it counts this release of ENTER key as one. So this release count
                                        ;   is ignored by initializing the count to -1

iNo : dw 0

location: db 0                  ;Location where the next star is to be printed


;-----------------------------------------------------------------------------------------------------------------

;Clear Screen
clrscr:                 pusha
                                push es

                                mov ax, 0xb800
                                mov es, ax
                                xor di,di
                                mov ax,0x0720
                                mov cx,2000

                                cld
                                rep stosw

                                pop es
                                popa
                                ret
;-----------------------------------------------------------------------------------------------------------------


;Program to print the stars
printStars:             pusha
                                push es

                                mov ax, 0xb800
                                mov es, ax

                                mov al, 80
                                mul byte [cs:location]
                                add ax, 159
                                shl ax, 1

                                mov di, ax

                                mov cx, [cs:tCount]

                                cmp cx, 0
                                jle return
```

```
l1:                     mov byte [es:di], '*'
                        inc byte [cs:location]
                        add di, 160
                        loop l1

return:         pop es
                        popa
                        ret


;-------------------------------------------------------------------------------------------------------------

CTS:            pusha

                        ;These lines will execute for the very first five seconds
                        cmp word [cs:iNo], 10
                        jz l2

                        add word [cs:ms], 55
                        cmp word [cs:ms], 1000
                        jl EOI2

                        mov word [cs:ms], 0             ;Resetting the MilliSeconds to
zero
                        call printStars                 ;Because the count is to be
updated every second i.e
                                                                                ;the
stars are to be printed after every second

                        mov ax, [cs:tCount]
                        mov bx, [cs:iNo]

                        mov word [cs:count + bx], ax

                        mov word [cs:tCount], 0
                        add word [cs:iNo], 2

                        jmp EOI2


l2:                     add word [cs:ms], 55
                        cmp word [cs:ms], 1000
                        jl EOI2

                        mov word [cs:ms], 0             ;Resetting the MilliSeconds to
zero

                        ;Shifting the counts towards the right, to create a space for this current
second

                        mov dx, 0
```

```
                        mov ax, [cs:count + 2]
                        add dx, ax
                        mov [cs:count], ax


                        mov ax, [cs:count + 4]
                        add dx, ax
                        mov [cs:count + 2], ax


                        mov ax, [cs:count + 6]
                        add dx, ax
                        mov [cs:count + 4], ax

                        mov ax, [cs:count + 8]
                        add dx, ax
                        mov [cs:count + 6], ax

                        mov ax, [cs:tCount]
                        add dx, ax
                        mov [cs:count + 8], ax

                        jmp a1


EOI2:                   jmp EOI
;Intermediate Jump


                        ;Now dx contains the count of the last five seconds

a1:                     mov [cs:tCount], dx

                        call clrscr

                        mov byte [cs:location], 0
                        call printStars

                        mov word [cs:tCount], 0


EOI:                    mov al, 0x20
                        out 0x20, al



exit:                   popa
                        iret



;-----------------------------------------------------------------------------------------------------------------

;Keyboard ISR

kbisr:    push ax
```

```
                in al, 0x60

                shl al, 1
                jnc EOI1

                inc word [cs:tCount]            ;If a key is released, only then increase the count

EOI1:   mov al, 0x20
                out 0x20, al

                pop ax

                iret

;-------------------------------------------------------------------------------------------------------------------



start:          mov ax, 0
                        mov es, ax

                        mov bx, 0

                        call clrscr

                        ;Hooking the interrupts
                        cli

                        mov word [es: 9*4], kbisr
                        mov [es:9*4+2], cs

                        mov word [es:8*4], CTS
                        mov [es:8*4+2], cs

                        sti


                        ;Code for making it TSR
                        mov dx, start                           ;End of resident portion
                        add dx, 15                                      ;round up to
next para
                        mov cl, 4
                        shr dx, cl                                      ;number of
paras

end:            mov ax, 0x3100                          ;terminate and stay resident
                        int 21h
```