UNIT – V :

TSR(Terminate And Stay Resident)

Prof. Reshma Pise

Computer Engg. Dept

Vishwakarma University

Contents

To know why we need TSR especially in DOS operating system

To understand the working of TSR

Learn how to write TSR programs.

Learn difference between active and passive TSR

Execute two TSR programs

1.To hook interrupt 0h(Passive TSR)

2.To hook interrupt 16h(Active TSR)

MULTIUSER / MULTITASKING OPERATING SYSTEM CONCEPT

An operating system that coordinates the action of the timesharing system is known as ***multi user operating system***

The program or section of a program for each user is referred to as task or process.

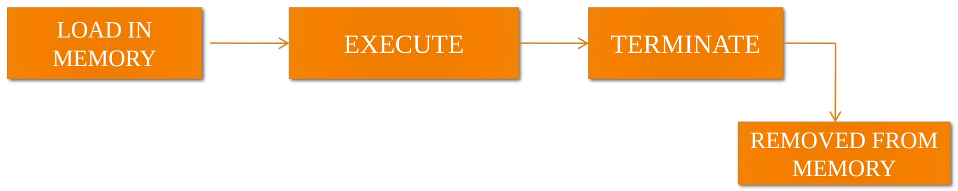
The multi user operating system is commonly referred to as ***multi tasking***.

DOS OPERATING SYSTEM

MS-DOS is designed as a single user-single tasking operating system.

Dos can execute only one program at a time.

Most MS-DOS applications are transient.



NORMAL DOS PROGRAMS

HOW TO MAKE DOS MULTI TASKING?

Using ***Terminate and Stay resident Programs***.

A resident program, upon termination, does not return all memory back to DOS.

Instead, a ***portion of the program*** remains resident, ready to be reactivated by some other program at a future time.

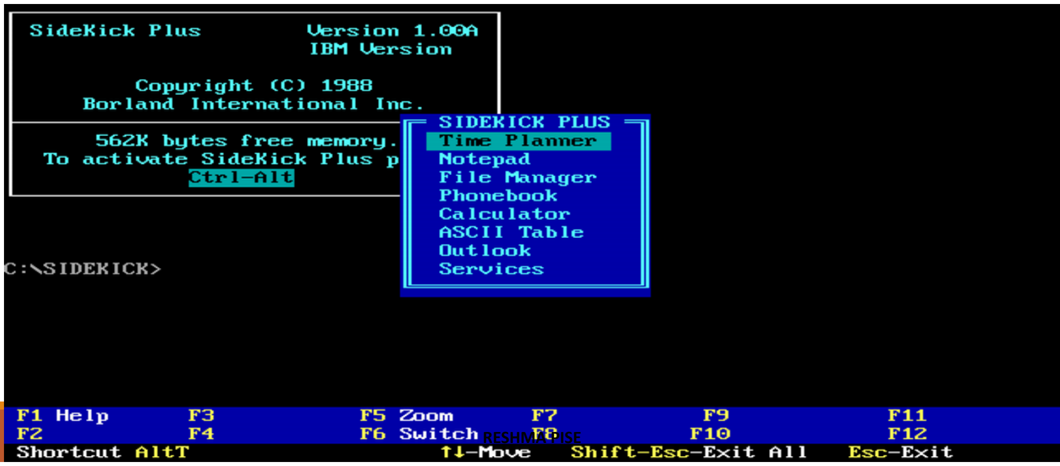
Thus they provide a ***tiny amount of multitasking*** to an otherwise single tasking operating system

Eg for ***TSRs are Sidekick , Prokey*** were developed which are pop up utility programs.

BORLAND'S SIDEKICK

Borland's **Sidekick** was an early Personal Information Manager (PIM) software application

They can be activated using a hot key combination (by default: Ctrl-Alt).

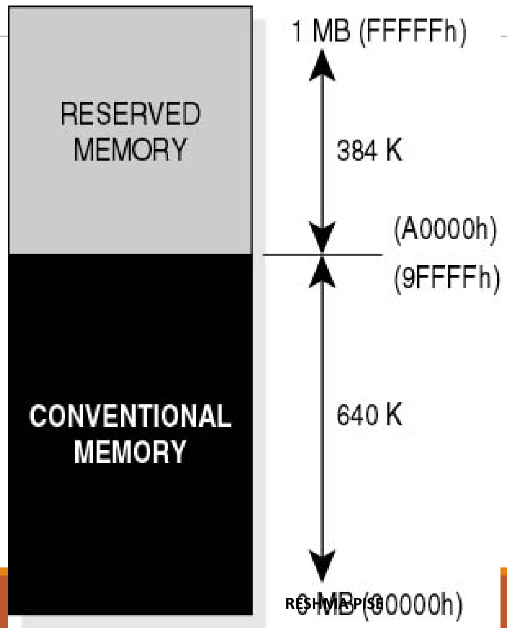


```
 SideKick Plus            Version 1.00A
                           IBM Version

          Copyright (C) 1988
     Borland International Inc.

       562K bytes free memory.
  To activate SideKick Plus p
            Ctrl-Alt

C:\SIDEKICK>
```

```
SIDEKICK PLUS
 Time Planner
 Notepad
 File Manager
 Phonebook
 Calculator
 ASCII Table
 Outlook
 Services
```

```
F1 Help     F3          F5 Zoom     F7          F9              F11
F2          F4          F6 Switch   F8          F10             F12
Shortcut AltT                  ↑↓-Move   Shift-Esc-Exit All   Esc-Exit
```

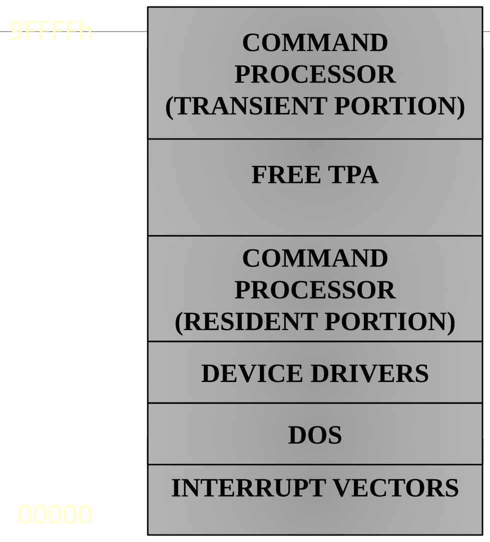
DOS MEMORY MAP



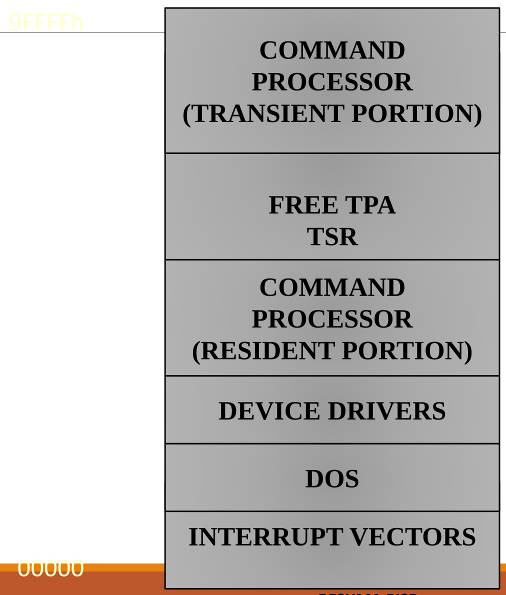
SYSTEM AREA

TPA(Transient program area)

CONVENTIONAL MEMORY WITHOUT TSR


0FFFFh
COMMAND PROCESSOR (TRANSIENT PORTION)
FREE TPA
COMMAND PROCESSOR (RESIDENT PORTION)
DEVICE DRIVERS
DOS
INTERRUPT VECTORS
00000

CONVENTIONAL MEMORY WITH TSR

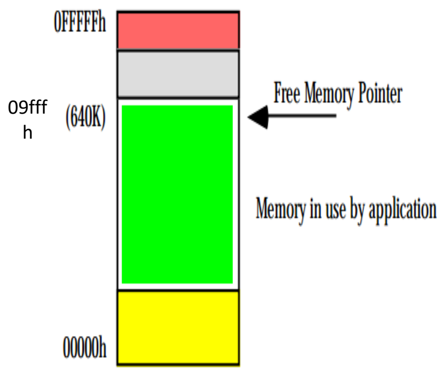

9FFFFh
COMMAND PROCESSOR (TRANSIENT PORTION)
FREE TPA
TSR
COMMAND PROCESSOR (RESIDENT PORTION)
DEVICE DRIVERS
DOS
INTERRUPT VECTORS
00000

DOS MEMORY POINTER

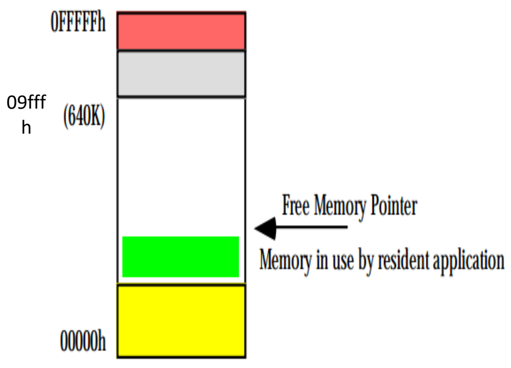
Dos maintains a memory pointer which points to the free memory area in RAM.

When TSR executes it resets the free-memory pointer to a higher position.

This leaves the program resident in a protected block of memory, even though it is no longer running.



DOS Memory Map (w/active application)



DOS Memory Map (w/resident application)

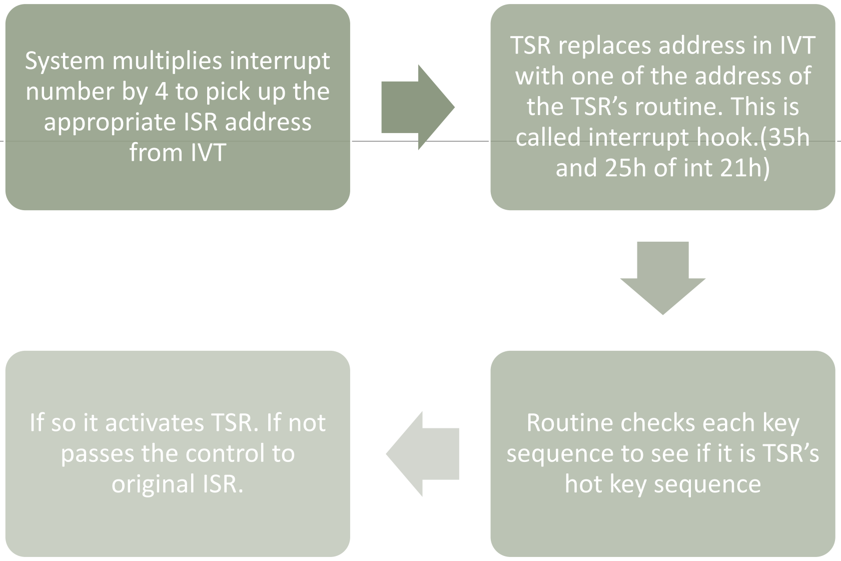
HOW A TSR WORKS?

We know that if we load more than one TSR , each one gets loaded on the top of each one like a TSR stack

When you hit the appropriate hot keys to bring up TSR,we are just activating the program that is already in the main memory.

But how does the TSR gets activated by hitting the appropriate key?

By using trick that involves ***interrupts and IVTs***.


System multiplies interrupt number by 4 to pick up the appropriate ISR address from IVT

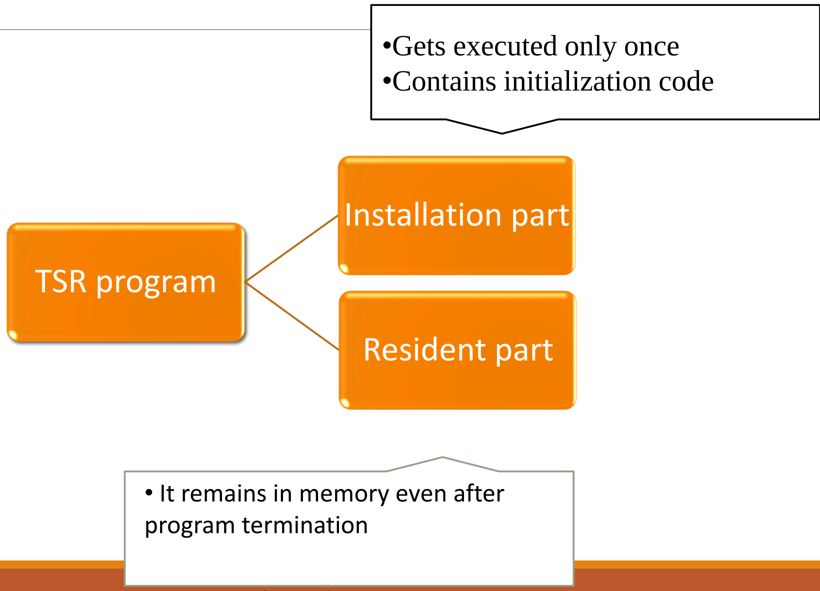
TSR replaces address in IVT with one of the address of the TSR's routine. This is called interrupt hook.(35h and 25h of int 21h)

Routine checks each key sequence to see if it is TSR's hot key sequence

If so it activates TSR. If not passes the control to original ISR.

This happens in a fraction of second that the user normally cannot find the difference.

STRUCTURE OF A TSR PROGRAM


•Gets executed only once
•Contains initialization code

• It remains in memory even after program termination

HOW TO MAKE PROGRAM RESIDENT?

To make program resident when it terminates ***use 31h sub function of int 21h***

AH - 31H
DX – LENGTH OF TSR PROGRAM

◦In current versions of MS-DOS, Interrupt 27h also provides a terminate-and-stay-resident service.

◦However, Microsoft cannot guarantee future support for Interrupt 27h and does not recommend its use.

EXAMPLE

```
mov    ah, 31h        ;  Request DOS Function 31h
mov    dx, 60         ; Reserve 60 paragraphs
                     ; (60*16 = 960 bytes)
 int    21h            ; Terminate-and-stay-resident
```

INTERRUPT HOOK

As said the interrupt hook is accomplished through the use of DOS int 21h , function 25h and 35h

The steps involved are:

1.Get vector interrupt to hook(function 35h)

2.Save old interrupt vector.

3.Make a new vector for the interrupt

4.Set new vector for hooked interrupt(function 25h)

The original vector of the ISR is saved in data area of the TSR and can be called using FAR call or FAR jump to call the original ISR as required

INT 25H AND INT 35H

int 21h function 35h

```
mov al,16h
 mov ah,35h                    ;get interrupt vector
 Int 21h
 Es:bx is stored with the cs:ip of the interrupt 16h
```

int 21h function 25h

```
Mov al,16h
 Mov ah,25h                    ;set interrupt vector
 Int 21h
 ds:dx is stored with the cs:ip of the TSR
```

Ex: HOOK 16H

```
;storage for old interrupt vector
        Oldipcs dd ?            ; need a double word for cs:ip

;save current vector address
        mov al,16h
        mov ah,35h              ;get interrupt vector
        Int 21h

 ;Es:bx is stored with the cs:ip of the interrupt 16h
```

```
Mov wordptr oldpics , bx              ; save int 16h ip
Mov word ptr oldpics[2],es   ; save int 16h code segment

;set new vector address
    Mov al,16h                ;interrupt number
    Mov ah,25h                ;set new vector address
    Lea dx,newisr             ;load ip of new isr
    Push cs
    Pop ds                    ;put copy of cs in to ds
    Int 21h
```

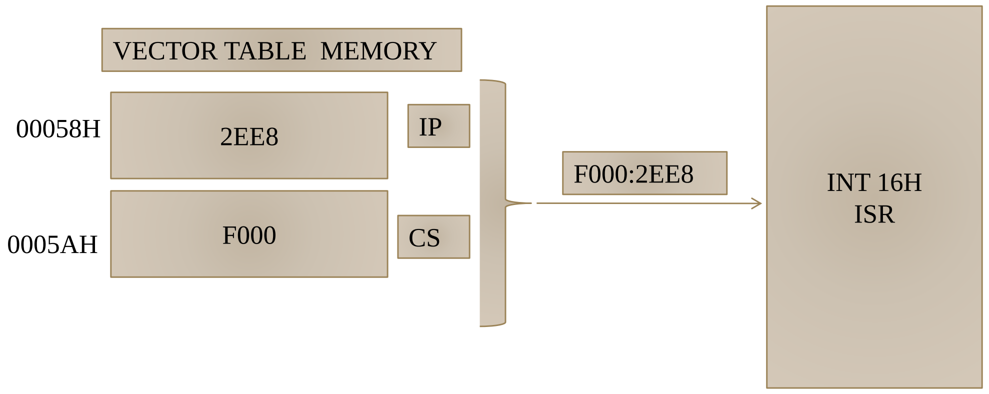
RETURN BACK TO ORIGINAL TSR

The TSR give control to the ISR in two ways:through use of FAR JMP or FAR CALL

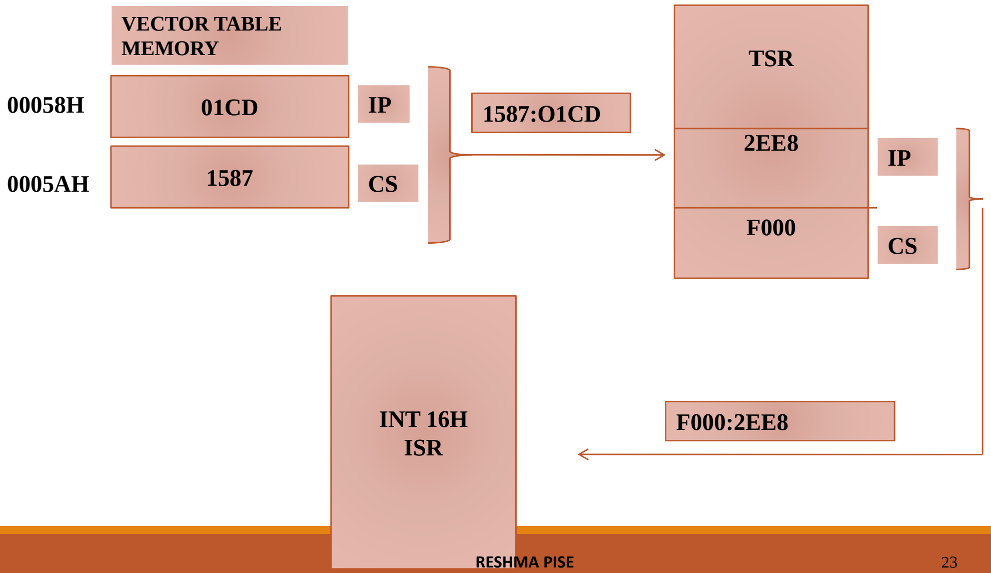
Ex: JMP OLDIPCS

In this case the iret at the end of the original TSR will cause a return to the program that caused interrupt.

ORIGINAL INTERRUPT VECTOR (HOOK 16H)



MODIFIED INTERRUPT CHAIN



ACTIVE AND PASSIVE TSR

The simplest way to execute a TSR is to transfer control to it explicitly from another program.

The TSRs are generally classified in to two types:

ACTIVE and PASSIVE

PASSIVE TSR – activated through software interrupts

ACTIVE TSR- activated through hardware interrupts(hotkey combinations)

TSR VIRUSES

Many viruses install themselves as TSR’S.

Being memory resident provides several **advantages** for viruses. This would account for the fact that the majority of viruses stay in the memory.

They will reside there until something external occurs like inserting a disk or executing a program.

By being in the memory, a resident virus can do anything the operating system can do. Some will modify or damage the computer's software to hide in the memory.

Another advantage to being memory resident is that the user cannot delete the virus while the computer is running.

Deleting these viruses may require doing a cold boot (restarting the computer after it has been off for a short time period) .

Examples of TSR viruses: all boot viruses, many file viruses, some macro viruses, some network viruses

But another good example of a TSR is a Virus Scanner, which must remain loaded in memory to help protect your computer from computer viruses.

TSR – PROS AND CONS

TSR’s reduce the size of the TPA.

Naturally it would consume some of the DOS’s precious memory

If we don’t exercise sufficient care it would end up with not enough memory to load the application.

TSR bring great power to user at the same time great responsibility

We must be very careful in designing TSR such that it’s operation may not disrupt the operations of other program.

SUMMARY

The TSRs help us to bring multitasking facility in DOS

The TSRs use function 31h,25h and 35h of int 21h.

The TSRs are both active and passive

The active TSRs are initiated through hardware interrupts

The passive TSRs are initiated through software interrupts

Many virus programs and even anti viruses are TSRs.

Care must be taken while working with TSRs

Thank You