ASSIGNMENT NO 12

AIM: - Write a TSR program in 8086 Assembly Language Program to implement Real Time Clock (RTC). Read the Real Time from CMOS chip by suitable INT and FUNCTION and display the RTC at the bottom right corner on the screen. Access the video RAM directly in your routine.

APPARATUS:-

- Core 2 duo/i3/i5/i7 - 64bit processor
- OS – windows 32bit/64bit OS
- Assembler used –MASM / TASM assembler for assembling. Link / Tlink for Linking.
- Editor Used – Edit / Notepad / WordPad
- Debugger : Debug

OBJECTIVE:

1) To understand the structure of IVT,
2) To understand the structure of TSR program

THEORY:-

1. **IVT: Interrupt Vector Table:**

   (Draw diagram of IVT and explain)

2. **TSR programs:**

   There are two types of programs which we execute on system: Transient programs and resident programs. After termination of transient programs the complete memory occupied by the program will be released whereas in case of resident programs part of program (resident portion) will remain in memory and part of program (initialisation routine) will be removed from the memory.

   - These programs are .COM programs
   - Can be loaded after DOS is loaded
   - Stay in the memory, even if they are not active on your screen
   - They appear to exit, but remain in the memory to perform tasks in the background

3. **Structure of TSR programs:** These programs are divided in three different parts
   i. Data Area
   ii. Resident routine
   iii. Initialization routine

- **Data Area:** Where different data definitions are included as per the requirement of program e.g. Original entry in interrupt vector table, temporary space for registers etc.
- **Resident Routine:** Portion of program which will be made resident in the memory to perform specified task. During execution of specified task original register contents

may get change so these contents must be preserved and again loaded before calling original interrupt service routine

- **Initialisation routine:** Does the preliminary work to make resident routine stay resident in the memory, It executes only once.

  It performs following steps
  - Get the original address of specified interrupt from IVT and save it
  - Store the address of resident program in the IVT in place of original address,
  - Calculate the size of the resident routine including Data area and PSP. Reserve the memory area of this size and make the program resident

  To perform these three steps we use three INT 21H functions
  - Function 35H: Get Interrupt Address
    - Call with AH=35H,
      AL= int #
    - Returns vector address in ES:BX

  - Function 25H: Set Interrupt Address
    - Call with AH=25H,
      AL=int # ,
      DS:DX= Segment:Offset of interrupt handling routine

  - Function 31H: Terminate and Stay resident
    - Call with AH=31H,
      AL=return code
      DX= Amount of Memory to reserve

4. To read time from the CMOS time/clock chip, we use INT 1Ah (Real Time Clock Driver) and its function 02h (Get Time). It returns CH= Hours in BCD, CL= minutes in BCD, DH= seconds in BCD, DL= daylight-saving-time code (00h if standard time and 01h if daylight saving time), and carry will be clear if clock is running else carry will be set if clock is stopped.
   This time we are displaying at specific location continuously for which we will use display RAM

5. **Video Display Adaptors:** The video display adapters found in IBM PC-compatible computers have a hybrid interface to the central processor. The overall display characteristics, such as vertical and horizontal resolution, background color, and palette, are controlled by values written to I/O ports whose addresses are hardwired on the adapter, whereas the appearance of each individual character or graphics pixel on the display is controlled by a specific location within an area of memory called the regen buffer or refresh buffer. Both the CPU and the video controller access this memory; the software updates the display by simply writing character codes or bit patterns directly into the regen buffer.

Address for regen buffer for CGA, EGA, MCGA and VGA is B8000h (B800:0000)

Each character-display position is allotted 2 bytes in the regen buffer. The first byte (even address) contains the ASCII code of the character, which is translated by a special hardware character generator into a dot-matrix pattern for the screen. The second byte (odd address) is the attribute byte.

6. Difference between .EXE and .COM Files

No	.EXE program	.COM program
1	These programs can have multiple code, data and stack segments.	Max only one segment is present. Data ,code and stack reside in one segment.
2	.EXE programs can be as large as available memory.	.COM program can have maximum size of 64KB. **(Actual Size is 64KB - Size of PSP – 2 Bytesfor stack)**
3	.EXE programs fit in the small, medium or large model.	.COM programs fit in the tiny model, in which all segment registers contain the same value.
4	NEAR and FAR calls can be used.	Only NEAR calls.
5	Due to the file header .EXE programs is larger than corresponding .COM program	File header is not present in .COM programs hence it always compact than .EXE.
6	The entry point of .EXE program will never be fixed and varies according to the file header size.	The entry point of .COM program is 0100H (the length of PSP) which is always fixed.
7	.EXE files contains unique header, a relocation map and other information used by DOS along with the execution code.	The .COM files are compact and are loaded slightly faster then equivalent .EXE file since these contain only the execution code.

Algorithm of program: (Resident Routine)

1. ORG 100H
2. Unconditionally jump to initialisation routine
3. Reserve memory lactations to store the registers and original vector address
4. Store the registers temporarily
5. Read time
6. Initialise base address(B800h) of page-0 of video RAM in ES and offset(3984h) of a location where we want to display the RTC,
7. Display HH:MM:SS
8. Restore the original register contents
9. Call the original interrupt service procedure

Algorithm of initialisation routine

1. Clear interrupt flag to avoid any hardware interrupt during the process of initialisation,
2. Read the original vector address entry and store is in data area
3. Set the vector address to our interrupt service routine
4. Set interrupt flag
5. Terminate and make it resident

Conclusion:

_____

_____