Name:	Yash Ravindra Kerkar
Roll No:	67
Class/Sem:	SE/IV
Experiment No.:	3
Title:	Program for drawing square using Assembly Language.
Date of Performance:	31/01/24
Date of Submission:	07/02/24
Marks:	
Sign of Faculty:	

Aim: Program for drawing square using Assembly Language.

Theory: INT 10h is a video service bios interrupt. It includes services like setting the video mode, character and string output and reading and writing pixels in graphics mode. To use the BIOS interrupt load ah with the desired sub-function. Load other required parameters in other registers and make a call to INT 10h.

INT 10h/AH = 0ch -Write graphics pixel.

Input:
AL = pixel colour
CX = column
DX = row

Algorithm:
1. Start

2. Initialize ax to 0013h for graphics mode.

3. Set the Counter bx to 60 h.

4. Initialize the co-ordinates cx and dx to 60h.

5. Set the Color.

6. Set Display Mode function by making ah = 0ch.

7. Increment cx and Decrement bx.

8. Repeat step 7 until bx = 0.

9. Initialize the counter by making bx = 60h.

10. Set the color.

11. Set Display Mode function by making ah = 0ch.

12. Increment dx & Decrement bx.

13. Repeat step 12 until bx = 0.

14. Initialize the counter by making bx = 60h.

15. Set the Color.

16. Set Display Mode function by making ah = 0ch.

17. Decrement cx and Decrement bx.

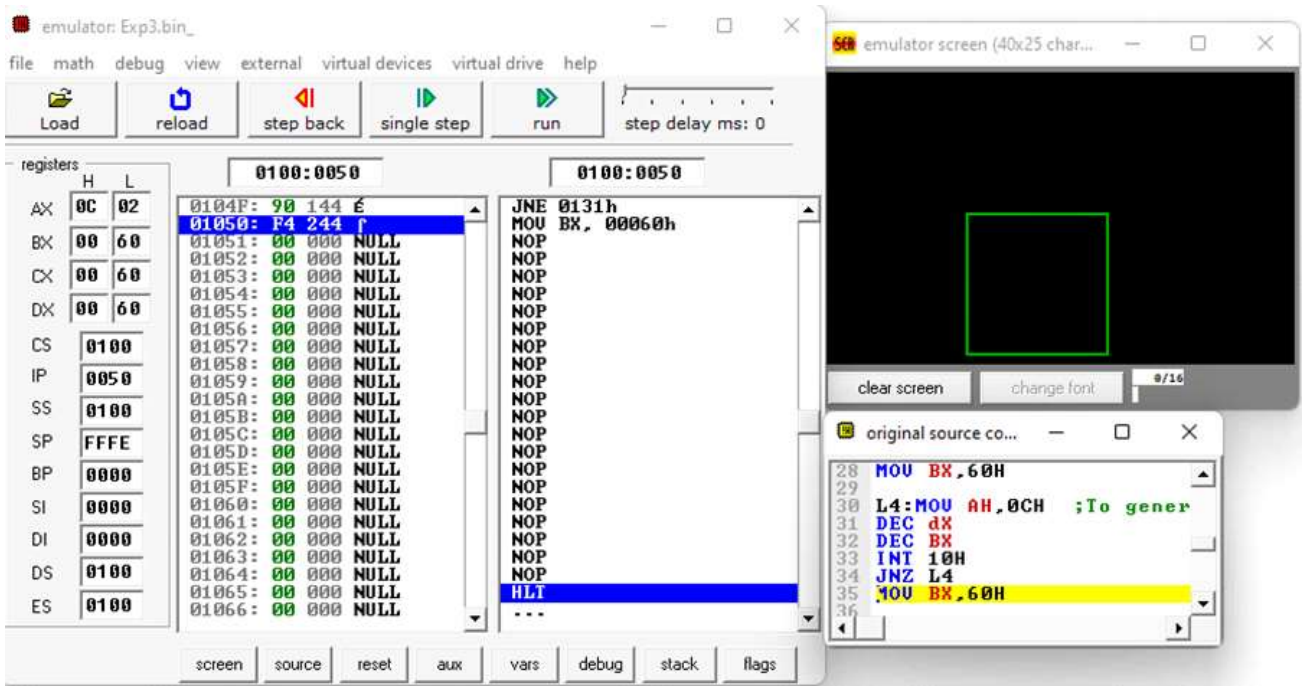
18. Repeat step 17 until bx = 0.
19. Initialize the counter by making bx = 60h.
20. Set the color.
21. Set Display Mede function by making ah = 0ch.
22. Decrement dx & Decrement bx.
23. Repeat step 22 until bx = 0.
24. To end the program use DOS interrupt:
    1) Load ah = 4ch.
    2) Call int 21h.
25. Stop.

Program:

```
MOV AX, 0013H ;To enter into the graphics mode
INT 10H
MOV BX,60H      ;Counter
MOV CX,60H      ;x-axis
MOV DX,60H      ;y-axis
MOV AL,02H      ;Color:Green
L1:MOV AH,0CH  ;To generate a pixel
  INC CX
  DEC BX
  INT 10H
  JNZ L1
  MOV BX,60H
L2:MOV AH,0CH  ;To generate a pixel
  INC DX
  DEC BX
  INT 10H
  JNZ L2
```

```
    MOV BX,60H
L3:MOV AH,0CH  ;To generate a pixel
    DEC CX
    DEC BX
    INT 10H
    JNZ L3
    MOV BX,60H
L4:MOV AH,0CH  ;To generate a pixel
    DEC dX
    DEC BX
    INT 10H
    JNZ L4
    MOV BX,60H
```

Output:



Conclusion: We successfully implemented a routine to draw a square. By utilizing simple arithmetic operations to manipulate memory locations corresponding to display pixels, we effectively generated a square shape on the screen. This exercise not only reinforced our understanding of memory manipulation and control flow in assembly language but also provided a practical illustration of how low-level programming can interact with hardware components to achieve graphical output.

1) Explain the use of int 10.

Ans. In assembly language programming, int 10h is a software interrupt that allows interaction with the video display services provided by the BIOS (Basic Input/Output System). It is commonly used for displaying text, graphics, and controlling the cursor on the screen.

Here's how `int 10h` is typically used:

Text Mode Functions:

`int 10h, AH = 0x0E`: Display a character in the teletype mode.

`int 10h, AH = 0x0A`: Display a character in the graphics mode.

`int 10h, AH = 0x02`: Set the cursor position.

`int 10h, AH = 0x06`: Scroll the screen up.

`int 10h, AH = 0x07`: Scroll the screen down.

`int 10h, AH = 0x09`: Write a character and attribute at the current cursor position.

Graphics Mode Functions:

`int 10h, AH = 0x00`: Set the video mode.

`int 10h, AH = 0x01`: Get the current video mode.

`int 10h, AH = 0x02`: Set the cursor position.

`int 10h, AH = 0x0C`: Write a pixel.

`int 10h, AH = 0x0D`: Read a pixel.

`int 10h, AH = 0x0E`: Write a character with attribute at the current cursor position.

Palette Control Functions:

`int 10h, AH = 0x10, AL = 0x00`: Set the palette registers.

`int 10h, AH = 0x10, AL = 0x01`: Set the border color.

2) Explain hardware interrupts.

Ans. Interrupt Request (IRQ) Lines: Each hardware device is assigned a specific interrupt request line (IRQ) on the system's interrupt controller. When a device wants to communicate with the CPU, it asserts its IRQ line, signaling that it requires attention.

Interrupt Controller: The interrupt controller is responsible for managing the flow of interrupts within the system. It prioritizes interrupts based on their importance and routes them to the CPU. Older systems might use the Programmable Interrupt Controller (PIC), while modern systems typically use the Advanced Programmable Interrupt Controller (APIC) or the Interrupt Controller Architecture for PCI (PCI-ICA).

Interrupt Service Routine (ISR): When an interrupt occurs, the CPU interrupts its current execution and jumps to the corresponding Interrupt Service Routine (ISR), also known as interrupt handler. The ISR is a piece of code responsible for handling the interrupt. It typically saves the CPU's current state, acknowledges the interrupt to the hardware device, performs the necessary actions to handle the interrupt, and then restores the CPU's state before returning control to the interrupted program.

Interrupt Vector Table (IVT): The Interrupt Vector Table is a data structure that maps interrupt numbers to the memory addresses of their corresponding ISRs. When an interrupt occurs, the CPU uses the interrupt number to index into the IVT and find the address of the appropriate ISR.

Interrupt Prioritization: Interrupts can have different priorities, and the interrupt controller ensures that higher-priority interrupts are serviced before lower-priority ones. This prioritization mechanism helps ensure that critical tasks receive prompt attention from the CPU.