Experiment 7

Code:

```
.model small

.stack 100h

.data

.code

start:

        mov ax, 0Dh

        int 10h


        mov ah, 0Ch

        mov al, 0Eh

        mov cx, 160

        mov dx, 100


DrawTop:

        int 10h

        dec cx

        cmp cx, 75

        jnz DrawTop


DrawLeft:

        int 10h

        inc dx

        cmp dx, 145

        jnz DrawLeft


DrawBottom:

        int 10h
        inc cx
        cmp cx, 160
        jnz DrawBottom

DrawRight:
        int 10h
        dec dx
        cmp dx, 100
        jnz DrawRight

WaitKey:
        mov ah, 08h
        int 21h
        cmp al, 13
        jnz WaitKey

        mov al, 03h
        mov ah, 00h
        int 10h


        mov ah, 4Ch
        int 21h
END
```

Explanation:

1. **.**model small
   - Assembler directive. Use the small memory model (one 64KB code segment + one 64KB data segment). Not executed at runtime.
2. .stack 100h
   - Reserve 0x100 (256) bytes for the stack. Not executed at runtime.

3. .data
    - Start of the data segment. (Empty in this program.)
4. .code
    - Start of the code segment.
5. start:
    - Label — entry point of the program. Execution begins here.
6. mov ax, 0Dh
    - Load AX with 000Dh. This places AL = 0Dh and AH = 00h. We are preparing for a BIOS video-mode call.
7. int 10h
    - BIOS video interrupt. With AH = 00h and AL = 0Dh this sets the video mode to mode 0Dh.
    - Effect: Switches display to a graphics mode (commonly a 320×200, 16-color graphics mode on VGA/EGA-compatible BIOSes). The screen clears and becomes graphical.
8. mov ah, 0Ch
    - Set AH = 0Ch → BIOS function Write Graphics Pixel (INT 10h, AH=0Ch). This function writes one pixel at (CX,DX) in color AL (and optional page in BH).
9. mov al, 0Eh
    - Set AL = 0Eh → the color index used to plot pixels (palette-dependent; often a bright/yellow color).
10. mov cx, 160
    - Set CX = 160 → the X (column) coordinate for the first pixel to plot. (Approximately horizontal center of 320-pixel width.)
11. mov dx, 100
    - Set DX = 100 → the Y (row) coordinate for the first pixel to plot. (Approximately vertical center of 200-pixel height.)

Top edge loop (draws from right → left)

12. DrawTop:
    - Label for the top-edge drawing loop.
13. int 10h

- Call INT 10h (AH=0Ch) to plot one pixel at (X = CX, Y = DX) with color AL.
    - On first iteration this draws pixel at (160, 100).

14. dec cx
    - Decrement CX by 1 → move one pixel left.
15. cmp cx, 75
    - Compare CX with 75 (left boundary X coordinate).
16. jnz DrawTop
    - If CX != 75, jump back to DrawTop and draw another pixel.
    - Loop behavior & pixels drawn: Starting with CX=160, the code draws pixels at X = 160, 159, 158, …, 76 (inclusive) all at Y = 100. The loop exits when CX has been decremented to 75 (75 is the stopping value — pixel at X=75 is not drawn here).

Left edge loop (draws from top → bottom)

17. DrawLeft:
    - Label for left-edge loop.
18. int 10h
    - Plot a pixel at current (CX, DX). At this moment CX = 75 and DX = 100, so this first pixel for the left edge is at (75, 100).
19. inc dx
    - Increment DX by 1 → move one pixel down.
20. cmp dx, 145
    - Compare DX with 145 (bottom boundary Y coordinate).
21. jnz DrawLeft
    - If DX != 145, loop.
    - Loop behavior & pixels drawn: Draws the left column at X = 75, Y = 100, 101, 102, …, 144 (inclusive). The loop exits when DX becomes 145 (the pixel at Y=145 is not drawn by this loop).

Bottom edge loop (draws from left → right)

22. DrawBottom:

- Label for bottom horizontal edge.

23. int 10h
    - Plot pixel at current (CX, DX). At loop entry CX = 75 and DX = 145, so the first bottom pixel is at (75, 145).
24. inc cx
    - Increment CX → move one pixel right.
25. cmp cx, 160
    - Compare CX to 160 (right boundary X coordinate).
26. jnz DrawBottom
    - If CX != 160, loop.
    - Loop behavior & pixels drawn: Draws the bottom line at Y = 145, X = 75, 76, 77, …, 159 (inclusive). The loop stops when CX increments to 160 (the pixel at X=160 will be drawn in the next stage).

Right edge loop (draws from bottom → top)

27. DrawRight:
    - Label for the right vertical edge.
28. int 10h
    - Plot pixel at the current (CX, DX). At this point CX = 160, DX = 145 so this draws (160,145).
29. dec dx
    - Decrement DX → move one pixel up.
30. cmp dx, 100
    - Compare DX to 100 (top boundary Y coordinate).
31. jnz DrawRight
    - If DX != 100, loop.
    - Loop behavior & pixels drawn: Draws right column at X = 160, Y = 145, 144, 143, …, 101 (inclusive). The loop stops when DX reaches 100 (the top-right pixel at (160,100) was already drawn at the very beginning when the top edge loop started).

Wait for ENTER

32. WaitKey:
    - Label for the input wait loop.
33. mov ah, 08h
    - Set AH = 08h → DOS function to read a character without echo (INT 21h, AH=08h).
34. int 21h
    - Call DOS; returns one character in AL.
35. cmp al, 13
    - Compare the returned character with ASCII 13 (Carriage Return = Enter key).
36. jnz WaitKey
    - If it is not Enter, jump back and wait again. The program therefore loops until the user presses Enter.

Restore text mode and exit

37. mov al, 03h
    - Set AL = 03h (video mode code for 80×25 text mode).
38. mov ah, 00h
    - Set AH = 00h to prepare for BIOS Set Video Mode function.
39. int 10h
    - BIOS: Set video mode to 03h — restores the normal DOS text screen.
40. mov ah, 4Ch
    - Set AH = 4Ch → DOS Terminate Program function.
41. int 21h
    - Call DOS to terminate the program and return to DOS. (AL holds an exit code; currently AL still contains 03h, so DOS will receive exit code = 3.)
42. END
    - Assembler directive marking end of source. (Some assemblers expect END start to indicate the entry point.)