1. Introduction

This program is a DOS-based assembly application that reads characters from the user, counts the occurrences of the letters 'x' and 'y' (case-insensitive), and displays the counts a specified number of times. It demonstrates basic **input/output handling**, **looping constructs**, **conditional branching**, and **subroutine usage** in assembly language.

The program runs as a **.COM file** using the **tiny memory model**, meaning that code, data, and stack all reside in a single 64 KB segment. It uses **DOS interrupts (INT 21h)** for all input/output operations.

---

2. Model Used

Memory Model

- **Tiny Model**: The program uses .model tiny and org 100h, which is the standard for MS-DOS .COM files.
- **Characteristics**:
  - Code and data share the same segment.
  - The program starts at offset 100h.
  - Maximum size: 64 KB, including code, data, and stack.

Registers Used

Register	Purpose
AX	Holds values for arithmetic and as input to subroutine for printing numbers.
BX	Counter for letter 'y'.
CX	Counter for letter 'x'.
DX	Used for I/O (printing characters, strings) and division remainder in number printing.
SI	Stores repetition count input by the user.
DI	Loop counter for printing results multiple times.
AL	Stores individual input characters from the user.

---

3. Algorithm Used

The program uses a **simple linear algorithm** with sequential steps:

Step 1: Initialize Counters

- Clear CX and BX to 0. These will store counts of 'x' and 'y'.

Step 2: Input Loop

- Use DOS interrupt 21h function 01h to read characters.
- Check each character:
  - If 'x' or 'X' → increment CX.
  - If 'y' or 'Y' → increment BX.
  - If 'q' or 'Q' → exit loop.
  - Otherwise, ignore character.
- Repeat until 'q' or 'Q' is entered.

Step 3: Get Repetition Count

- Prompt the user for a number 0-9.
- Convert the ASCII character to a numeric value by subtracting '0'.
- Store the repetition count in SI.

Step 4: Print Results Loop

- Loop SI times:
  - Print "X count: " and the value of CX.
  - Print "Y count: " and the value of BX.
  - Print a blank line after each repetition except the last.
- Uses subroutine print_number to display decimal numbers:
  - If value = 0 → print '0'.
  - Otherwise:
    - Divide the number repeatedly by 10.

        - Push remainders onto the stack.
        - Pop and print digits to display the number in correct order.

Step 5: Exit Program

- Use DOS interrupt 21h function 4Ch to terminate program execution.

---

4. Key Features

1. **Case-insensitive counting**: Handles both uppercase and lowercase inputs.
2. **Repetition mechanism**: The results can be displayed multiple times as specified by the user.
3. **Numeric printing subroutine**:
    - Converts binary numbers to decimal ASCII representation.
    - Handles zero as a special case.
    - Uses stack to reverse digits for correct display.
4. **Pure DOS interrupt I/O**: Demonstrates knowledge of DOS system calls for strings and characters.

---

5. Conclusion

This program demonstrates:

- Effective use of **DOS interrupts** for input/output in assembly language.
- Implementation of **loops and conditional branching**.
- Usage of **registers for counters and arithmetic operations**.
- Writing **subroutines** to handle reusable tasks (printing numbers).

The **tiny memory model** is ideal for small .COM programs where code and data share a single segment. The program efficiently counts and displays character occurrences while teaching fundamental assembly programming concepts such as:

- Handling user input
- Case-insensitive comparisons

- Looping constructs
- Stack usage for number conversion

Overall, this program is a solid example of combining **basic assembly logic with DOS I/O**, suitable for educational purposes in learning low-level programming and understanding the architecture of MS-DOS.