**Report:**

**Simple Encryption and Decryption Using 8086 Assembly Language with GUI**



**Names:**

Nashra Azhar -081

Fizzah Rafiq-075

Humna Sheraz-083

**Class:**

BCS-3(A)

**Subject:**

COAL Lab

**Professor:**

Sir Majid Shafiq

**1. Problem Statement**

In today’s digital world, data security is a major concern. Plain text data can be easily accessed or modified by unauthorized users. There is a need for a simple system that can protect textual data using encryption and also restore it using decryption.

The problem addressed in this project is to design and implement a simple encryption and decryption system that:

* Accepts user input
* Encrypts alphabetic characters
* Displays encrypted output
* Decrypts the data back to its original form
* Provides both command-line (Assembly) and graphical user interface (GUI) based interaction

**2. Objectives**

* To understand the working of encryption and decryption
* To implement a substitution cipher using 8086 Assembly Language
* To use lookup tables for fast character mapping
* To calculate total character count
* To design a GUI using C++ WinForms API
* To integrate low-level logic with a user-friendly interface

**3. Tools and Software Used**

|  |  |
| --- | --- |
| **Component** | **Description** |
| Language | 8086 Assembly, C++ |
| Assembler | MASM |
| Emulator | DOSBox 0.74 |
| GUI Framework | Windows API (Win32) |
| IDE | Dev-C++ |
| OS | Windows |

**4. System Overview**

The project consists of two main parts:

1. **Assembly Program (Core Logic)**Handles encryption, decryption, lookup tables, and character counting.
2. **C++ GUI Application**Provides an interactive interface for user input and output display.

**5. Encryption & Decryption Technique**

A substitution cipher is used in which:

* Each alphabetic character is replaced by another character
* Separate tables are maintained for:
  + Lowercase encryption
  + Lowercase decryption
  + Uppercase encryption
  + Uppercase decryption

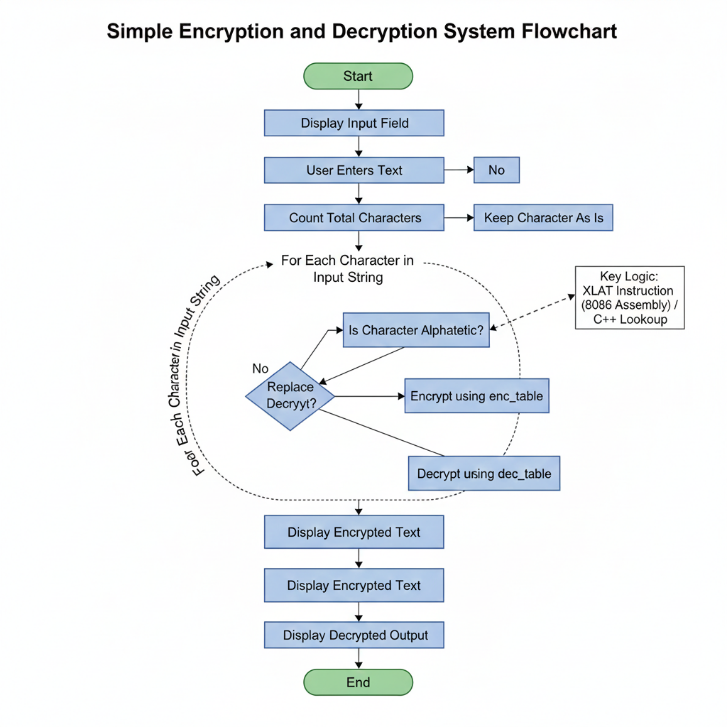
Non-alphabetic characters remain unchanged.

**6. Lookup Tables**

* enc\_lo → Encrypts lowercase letters
* dec\_lo → Decrypts lowercase letters
* enc\_up → Encrypts uppercase letters
* dec\_up → Decrypts uppercase letters

In Assembly, the XLAT instruction is used for fast lookup.  
In GUI, the same logic is implemented using C++ strings.

**7. Flowchart**

****

**7.1 Flowchart Explanation**

The flowchart illustrates the logical sequence and operational flow of the Encryption and Decryption system. The process is divided into the following key stages:

1. **System Initialization (Start):** The process begins when the GUI application is launched. The system initializes the environment and waits for the user to interact with the input interface.
2. **Data Acquisition:** The user enters the desired plain text into the provided input field. Upon clicking the "Process" button, the system captures this string for processing.
3. **Character Enumeration:** Before encryption starts, the system iterates through the string to calculate the Total Character Count. This ensures the loop runs for the correct number of iterations and provides feedback to the user via the GUI.
4. **Character Analysis & Substitution (The Core Logic):** The system processes each character individually using a loop:
   * **Decision Phase:** It checks if the character is an alphabet (A-Z or a-z).
   * **Substitution Mapping:** If it is an alphabet, the 8086 Assembly XLAT instruction is utilized. It maps the character to its encrypted counterpart using the predefined Lookup Tables (enc\_lo or enc\_up).
   * **Bypass Logic:** If the character is a digit, space, or symbol, the system skips the mapping process and retains the original character to maintain data integrity.
5. **Bi-Directional Processing:**
   * **Encryption Cycle:** The plain text is transformed into cipher text and displayed in the "Encrypted" field.
   * **Decryption Cycle:** Immediately after encryption, the system applies the reverse logic using the decryption lookup tables (dec\_lo and dec\_up) to restore the data.
6. **Output & Termination (End):** The final results, the character count, the encrypted string, and the decrypted original string are displayed simultaneously in the GUI. This confirms the accuracy of the algorithm before the process terminates.

**8. Working of GUI Application**

* User enters text in the input box
* On clicking **Process**:
  + Character count is calculated
  + Text is encrypted
  + Encrypted text is displayed
  + Encrypted text is decrypted back
* All outputs are shown in separate fields

**9. Program Modules**

**Assembly Modules:**

* print\_count → Displays total characters
* crypt → Performs encryption/decryption
* map\_char → Maps characters using lookup tables

**GUI Modules:**

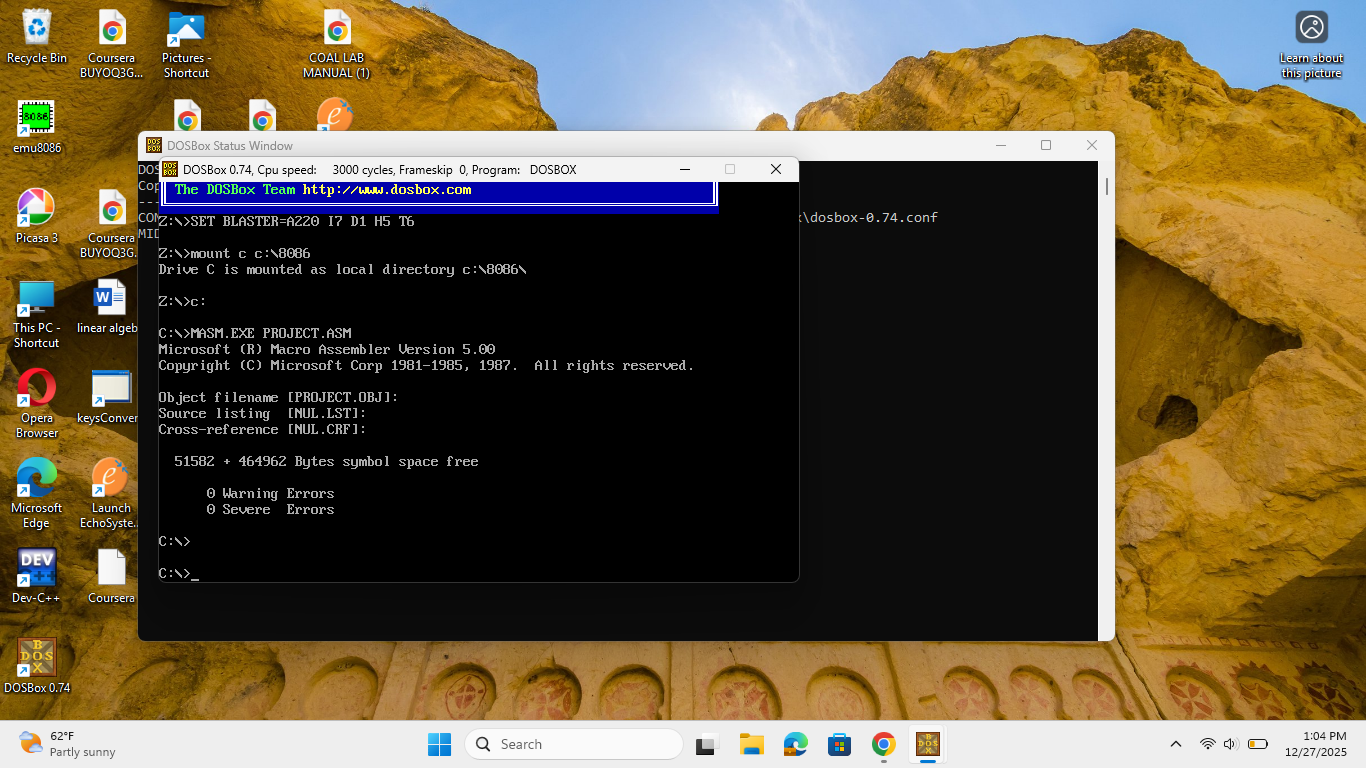
* map\_char() → Encrypts/decrypts one character
* crypt() → Processes complete string
* WindowProc() → Handles GUI events
* WinMain() → Application entry point

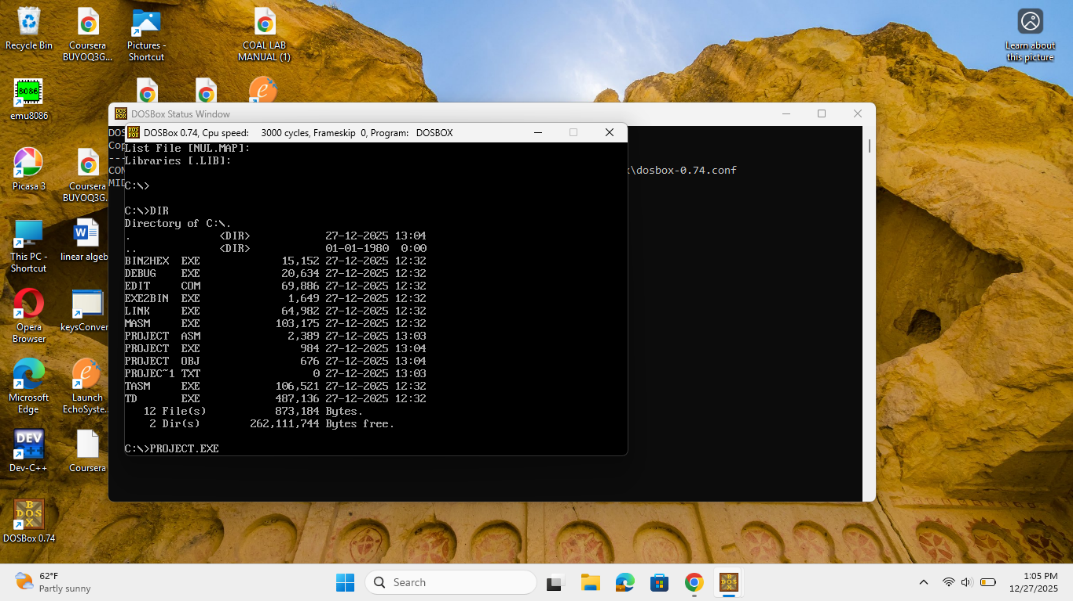
**10. Output Explanation**

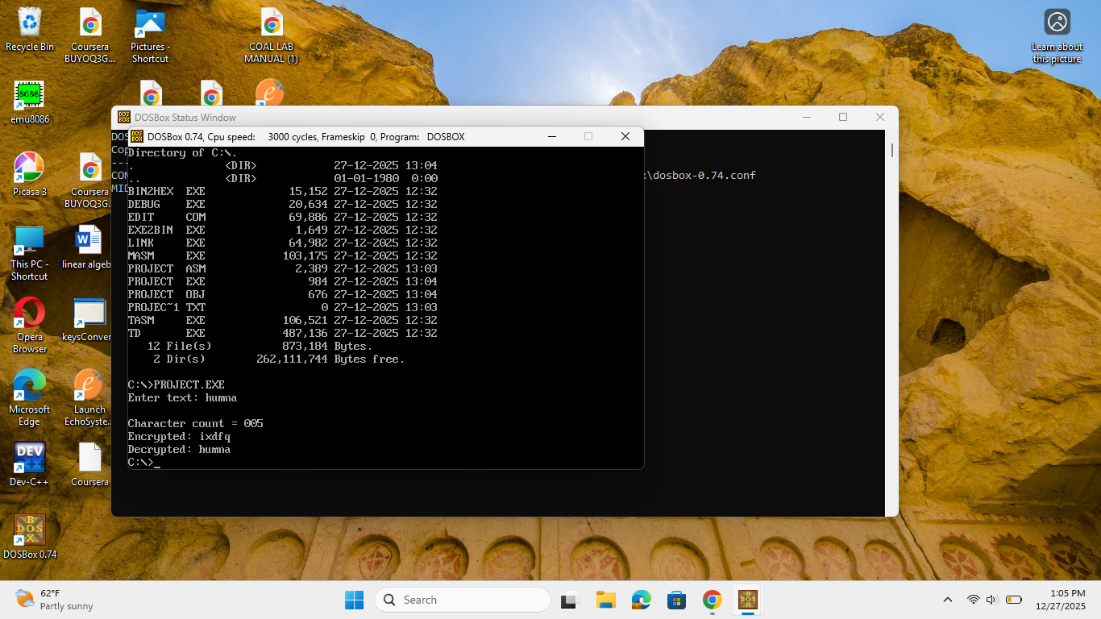
The GUI output shows:

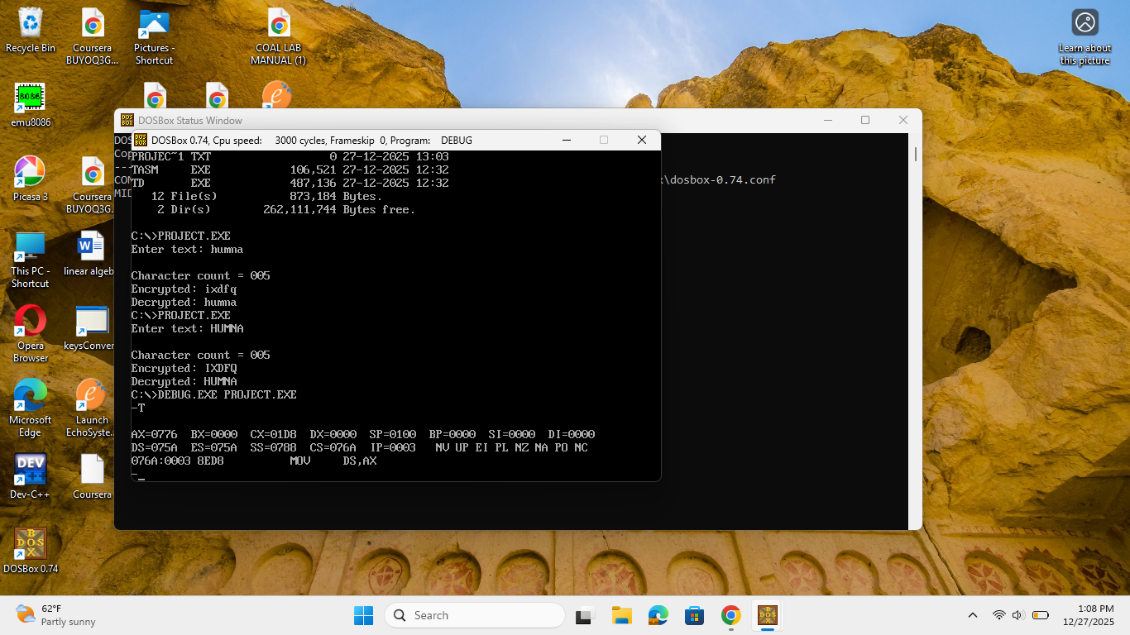
* Entered text
* Total character count
* Encrypted text
* Decrypted original text

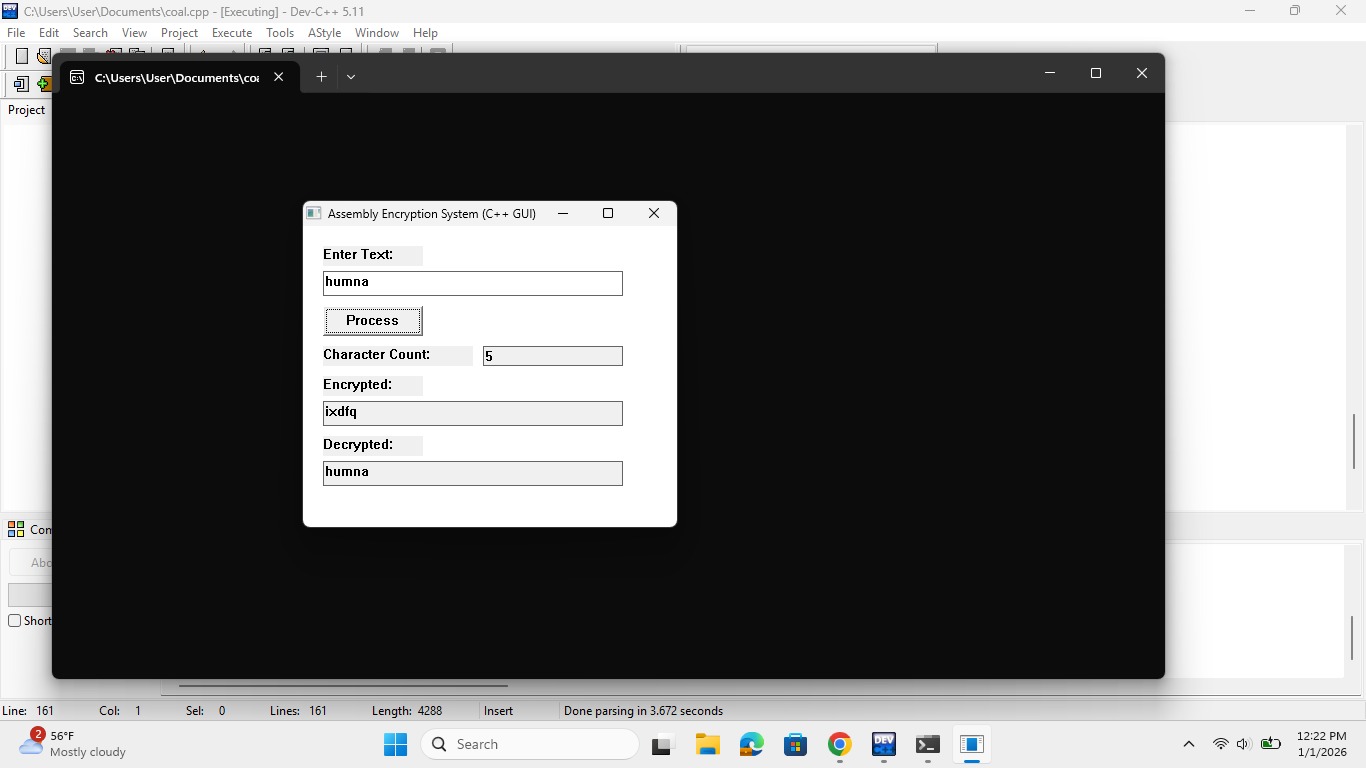
The decrypted output exactly matches the input, confirming correctness.











**11. Advantages**

* Simple and efficient design
* User-friendly GUI
* Fast encryption using lookup tables
* Supports both uppercase and lowercase letters
* Demonstrates Assembly + High-Level Language integration

**12. Limitations**

* Only alphabetic characters are encrypted
* No advanced cryptographic security
* Fixed substitution mapping

**13. Conclusion**

This project successfully demonstrates a **Simple Encryption and Decryption System** using **8086 Assembly Language** with a **C++ GUI interface**. The system efficiently encrypts and decrypts user data while maintaining simplicity and clarity. It is highly useful for understanding low-level programming concepts, lookup table usage, and GUI-based interaction.

**14. Group Members’ Roles and Contribution Summary**

This project was completed through collaborative effort, where each group member contributed actively to different aspects of the system.

**Member 1:** Humna Sheraz

* Designed and developed the C++ GUI using Windows API.
* Implemented encryption and decryption logic within the GUI.
* Assisted in Assembly logic integration and testing.
* Managed user interaction and output display.

**Member 2:** Fizzah Rafiq

* Prepared the project report and documentation.
* Designed and implemented core 8086 Assembly logic for encryption/decryption using lookup tables and XLAT.
* Assisted in testing, debugging, Assembly logic verification, and final integration.

**Member 3:** Nashra Azhar

* Prepared and delivered the project presentation.
* Designed flowcharts and explanations.
* Assisted in character counting and DOSBox execution.

**All members:**

Contributed equally to planning, testing, and finalizing the project.