**Semester Project Report**

**Project Title:** **Caesar Cipher in NASM x86 Assembly**

**Student's Name:**

a. Sameer Talreja (457795)

b. Abdul Basit (469017)

c. Bismillah (457434)

**Course Name:** CS-235: Computer Organization and Assembly Language (COAL)

**Instructor Name:** Engr Muhammad Abid Hussain

**Semester:** Spring 2025

Date: 23 June 2025



**Department of Computer Science**

**NUST Balochistan Campus (NBC), Quetta**

**Contents**

1. Motivation

2. Introduction

3. Background Knowledge

4. Problem Statement

5. Objectives

6. Flowchart

7.Program Structure

8. Subroutines

9.Caesar Cipher Logic

10. Features

11. Limitations

12.Conclusion

13. References

**1. Motivation**

The motivation for selecting the Caesar Cipher lies in its blend of simplicity and educational richness. It provides an approachable starting point for understanding cryptographic principles while requiring hands-on mastery of fundamental assembly language constructs. Implementing the Caesar Cipher in NASM x86 enhances understanding of:

* ASCII manipulation
* System-level I/O using Linux interrupts
* String and arithmetic operations at the hardware-near level

This project not only deepens theoretical knowledge of encryption but also strengthens practical skills in low-level programming, forming a base for more advanced cryptographic and systems programming in future studies.

**2. Introduction**

The Caesar Cipher is a substitution cipher in which each letter in the plaintext is shifted a certain number of positions in the alphabet. This classical technique is one of the earliest known encryption methods and is ideal for educational use due to its simplicity.

In this project, the Caesar Cipher is implemented in **NASM x86 assembly** using **Linux syscalls**. The program supports both **encryption** and **decryption**, takes input from the user, and performs character-by-character manipulation to transform the text.

**3. Background Knowledge**

To understand and implement this project, the following knowledge areas were essential:

* **Assembly Language (NASM):** Syntax, register usage, memory management.
* **Linux System Calls:** Using int 0x80 to perform I/O operations (read/write).
* **ASCII Encoding:** Recognizing and manipulating character codes.
* **String Processing:** Handling input buffers, string iteration, and modification.
* **Modular Arithmetic:** Essential for wrapping characters within the alphabet.

**4. Problem Statement**

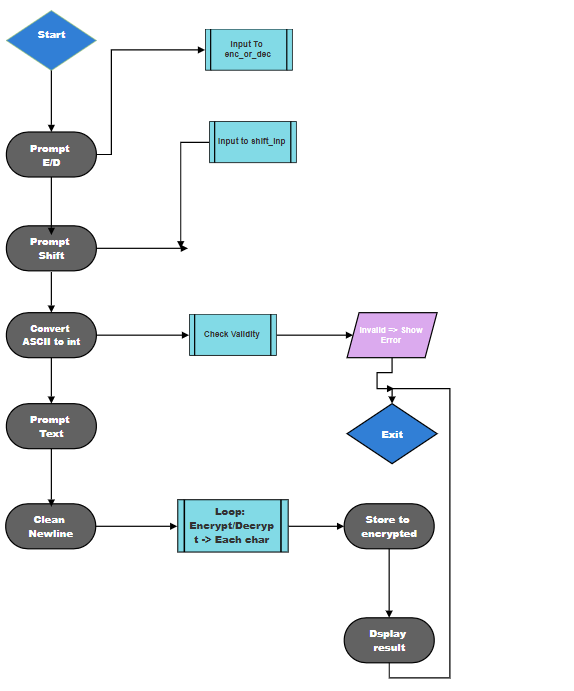
To design and implement a Caesar Cipher encryption/decryption program in NASM x86 assembly language, that:

* Takes a shift value and text input from the user.
* Allows the user to choose between encryption or decryption.
* Processes uppercase and lowercase letters correctly.
* Outputs the result to the terminal using system-level calls**.**

**5. Objectives**

* Implement a text-based Caesar Cipher in NASM x86 Assembly.
* Support both encryption and decryption modes.
* Learn and demonstrate low-level I/O handling using Linux syscalls.
* Practice string manipulation, loops, and modular arithmetic in assembly.
* Strengthen understanding of number conversion and memory handling in x86.

**6. Flowchart**

****

**7.Program Structure :**

The code is divided into the following sections:

**.data section**

Stores all the constant strings used for prompts and messages, such as:

* Prompts for encryption/decryption, shift value, input text
* Result label
* Error message
* Newline character

**.bss section**

Allocates uninitialized memory for:

* shift\_inp: User input for shift value (as a string)
* text\_inp: Input text to be encrypted/decrypted
* encrypted: Result of the encryption/decryption
* shift\_val: Final numeric value of shift
* enc\_or\_dec: Holds 'E' or 'D' for mode selection

**.text section**

**Main Program Flow**

1. **Mode Selection (E/D):**  
   Prompts the user for mode and reads into enc\_or\_dec.
2. **Shift Value Input and Conversion:**  
   Prompts the user, stores input in shift\_inp, converts ASCII to integer using str\_to\_int.
3. **Text Input:**  
   Prompts the user to enter text for encryption or decryption.
4. **Sanitization:**  
   Removes newline characters from input.
5. **Caesar Cipher Logic:**  
   Encrypts or decrypts the text based on the mode and stores in encrypted.
6. **Display Result:**  
   Prints the final result string.
7. **Error Handling:**  
   Displays an error if the shift value is invalid (e.g., negative).

**8. Subroutines**

**str\_to\_int**

* Converts ASCII string to numeric value stored in bl.
* Handles invalid digits and basic validation.
* Returns -1 on encountering a negative sign.

**print\_string**

* Calculates string length and prints using int 0x80 syscall (write syscall).

**9. Caesar Cipher Logic**

Implemented using ASCII manipulations:

* For **uppercase letters ('A'-'Z')**, the formula is:  
  ((char - 'A' + shift) % 26) + 'A' (or subtract shift for decryption)
* For **lowercase letters ('a'-'z')**, the formula is:  
  ((char - 'a' + shift) % 26) + 'a'
* All non-alphabetic characters are left unchanged.

**10. Features**

* Fully interactive input/output without external libraries.
* Supports both encryption and decryption.
* Handles uppercase and lowercase separately.
* Removes newline characters for clean output.
* Clears result buffer before each operation.

**11. Limitations**

* Shift value must be a positive integer less than 256.
* No handling of non-numeric input for shift value.
* No support for punctuation or non-English characters.

**12. Conclusion**

This project enhanced my understanding of x86 assembly by applying concepts such as register usage, syscalls, loops, conditionals, string processing, and modular arithmetic. It effectively demonstrated how a high-level concept like Caesar Cipher can be translated into low-level operations, which deepens understanding of how data is manipulated at the hardware-software interface.

**13. References**

* <https://www.nasm.us/doc/>
* <https://man7.org/linux/man-pages/man2/write.2.html>
* <https://en.wikipedia.org/wiki/Caesar_cipher>
* <https://www.geeksforgeeks.org/caesar-cipher-in-cryptography/>
* <https://caesar-cipher.com/>
* <https://www.youtube.com/watch?v=JtbKh_12ctg>