Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2024), B.Sc. in CSE (Day)

Cryptographic Algorithm Implementation on the 8086 Microprocessor Using Assembly Language

---

Project Report

Course Title: *Microprocessor and Microcontroller Lab*
Course Code: *CSE 304*
Section: *222 D9*

Students Details

Name	**ID**
Md. Mahim Hossain	221902082
Mohammad Habibullah	221902221

Submission Date: *12 Dec, 2024*
Course Teacher's Name: *Sagufta Sabah Nakshi*

[For teachers use only: Don't write anything inside this box]

Lab Report Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

This project aims to design a program that implements Caesar Cipher encryption and decryption using the EMU 8086. The Caesar Cipher is a basic cryptographic technique that shifts each character in the input text by a user-defined key value. This project will demonstrate the capability of the 8086 microprocessor in handling text-based cryptographic operations, utilizing its instruction set for efficient processing in assembly language.

1.2 Motivation

The growing importance of data security has made cryptography an essential field in computer science. This project focuses on implementing the Caesar Cipher in assembly language to provide practical experience and deeper understanding of cryptographic principles. Key motivations for this project include:

- Providing hands-on experience in low-level programming and text manipulation.
- Gaining insights into how encryption algorithms operate at the machine level.
- Highlighting the 8086 microprocessor's capabilities in solving real-world problems.
- Bridging the gap between theoretical knowledge of cryptography and its practical application.

1.3 Problem Definition

The problem addressed in this project is to implement the Caesar Cipher encryption and decryption algorithm on the 8086 microprocessor using assembly language. The challenge lies in performing cryptographic operations efficiently within the limited processing power and memory of the 8086 architecture. The project aims to handle both

uppercase and lowercase letters while ensuring the encryption and decryption processes work correctly. This solution demonstrates how basic cryptographic techniques can be executed on older hardware.

1.3.1 Problem Statement

In the digital age, securing information is essential to protect sensitive data from unauthorized access. Cryptography is one of the fundamental techniques used to achieve this, but many basic encryption methods, such as the Caesar Cipher, are often not fully understood at the low-level programming stage. While modern cryptographic systems rely on advanced algorithms, understanding basic methods and their implementation is critical for building a solid foundation in encryption techniques.

The problem lies in the challenge of implementing these basic cryptographic operations on resource-constrained microprocessors, like the 8086. Specifically, the issues to address include:

- Efficiently managing text input and output within the limitations of the microprocessor.
- Correctly handling character encoding and performing transformations with minimal errors.
- Designing a user-friendly interface while ensuring that the encryption/decryption process is both efficient and scalable.

These challenges highlight the need to bridge the gap between theoretical cryptographic methods and their practical application on hardware with limited computational resources, like the 8086 microprocessor.

1.3.2 Complex Engineering Problem

Table 1.1: Summary of the attributes addressed in the Caesar Cipher project

Name of the Problem Attributes	**Explain how to address**
Problem 01: User-friendly Interface	Use clear prompts for text input, key entry, and operation selection (encrypt/decrypt).
Problem 02: Comprehensive Coverage	Handle both uppercase and lowercase letters, non-alphabetic characters, and varying key sizes.
Problem 03: Real-time Feedback	Display encryption or decryption results immediately after the user inputs data.
Problem 04: Error Handling	Add error messages for invalid inputs and handle out-of-bound ASCII values gracefully.
Problem 06: Scalability	Design flexible code to support future cryptographic algorithms.

1.4 Design Goals

The primary design goals of this project are as follows:

- **Functionality:** The program should correctly perform Caesar Cipher encryption and decryption, handling both uppercase and lowercase letters and non-alphabetic characters.
- **Usability:** Ensure the program is user-friendly with clear prompts for text input, key entry, and operation selection.
- **Efficiency:** Optimize the use of CPU cycles and memory resources while maintaining reliable performance.
- **Scalability:** Design the program to support future enhancements, such as more complex encryption algorithms.

1.5 Application

The Caesar Cipher program showcases the 8086 microprocessor's capability in cryptographic operations. This project serves as a foundation for understanding basic cryptographic concepts. It also opens doors for exploring more advanced algorithms. Practical applications include:

- Teaching basic cryptography and assembly language in education.
- Implementing lightweight encryption in embedded systems.
- Building a foundation for developing and understanding secure algorithms like AES and RSA in low-level programming.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

The Caesar Cipher is used to implement cryptographic techniques in the project, Cryptographic Algorithm Implementation on the 8086 Microprocessor Using Assembly Language. The implementation shows how Assembly Language can be used to include data security into microprocessor-based systems. This project demonstrates the useful implementation of cryptographic concepts on outdated hardware, such as the 8086 microprocessor, by making it possible to encrypt and decrypt text communications.

The Caesar Cipher, one of the oldest known encryption methods, was selected due to its historical relevance and ease of use. The project emphasizes low-level programming and data security while showcasing both theoretical ideas and real-world application.

2.2 Project Details

In this section, we provide an in-depth explanation of the cryptographic algorithm implementation, the tools and techniques used, and the project's architecture. This project demonstrates the use of the Caesar Cipher encryption and decryption algorithm on the 8086 microprocessor using assembly language. The primary goal is to implement basic cryptographic functions and evaluate their performance within the context of a simple yet effective encryption scheme.

2.2.1 Project Overview

This project implements the Caesar Cipher encryption and decryption algorithm on the 8086 microprocessor, utilizing assembly language to demonstrate low-level operations such as bitwise arithmetic, ASCII handling, and user interaction through console input and output. The implementation aims to encrypt and decrypt messages by shifting letters in the alphabet, providing a simple introduction to cryptographic techniques.

2.2.2 System Design and Architecture

The system is designed around the 8086 microprocessor, which serves as the computational core for all operations. The project utilizes various assembly language constructs such as macros, loops, condition checks, and interrupts to handle user inputs and outputs efficiently.

Key architectural components include:

- User Interface: The user is prompted to select between encryption, decryption, and exiting the program, with text and numeric inputs.
- Encryption Module: Handles the encryption process by shifting the characters of the plaintext based on a key value.
- Decryption Module: Handles the reverse process by shifting the characters back to their original form.
- Memory Handling: Uses registers for key storage, text input, and output handling.
- Output Handling: Displays the encrypted or decrypted text on the console.

2.2.3 Project Functionality

This project is designed to:

- Encrypt: It takes plaintext input, applies a Caesar Cipher shift, and returns the encrypted text.
- Decrypt: It takes the encrypted text and applies a reverse shift based on the provided key, returning the original plaintext.
- Input Handling: The program allows the user to input a key (shift value), the number of characters in the text, and the actual text for encryption/decryption.
- Error Handling: Provides error messages for invalid choices or incorrect inputs, ensuring the program remains functional under various scenarios.

2.3 Implementation

In this section, we will detail the implementation of the cryptographic algorithm, Caesar Cipher, using Assembly language on the 8086 microprocessor. The implementation includes the overall workflow, the tools used, and the specific code logic. Screenshots and code snippets will be used to provide a clear understanding of the process.

2.3.1 2.3.1 Workflow

The workflow of the project is designed to be simple yet effective. It involves the following steps:

1. **User Input**: The program prompts the user to choose between encryption, decryption, or exit. Once an option is chosen, the program proceeds to the appropriate function.
2. **Text and Key Input**:
    - If encryption or decryption is chosen, the program will prompt the user to input the number of characters in the text, the actual text, and the key (the shift value for the Caesar Cipher).
3. **Processing**:
    - **Encryption**: Each character of the input text is shifted by the specified key in the alphabet, ensuring it wraps around if the character goes beyond 'Z' or 'z'.
    - **Decryption**: Similar to encryption, but the characters are shifted in the opposite direction.
4. **Display Result**: After processing, the program will display the encoded or decoded text on the screen.
5. **Return to Home**: The program will return to the main menu, allowing the user to perform another operation or exit.

2.3.2 2.3.2 Tools and Libraries

This project was implemented using the following tools and libraries:

- **8086 Microprocessor**: This serves as the base for the implementation, with Assembly language used to program it. The 8086 architecture provides support for memory addressing, input/output operations, and conditional branching.
- **EMU8086**: An EMU 8086 to run the Assembly program in an environment that simulates the 8086 microprocessor behavior.
- **Assembly Language**: Assembly language was chosen for low-level interaction with the hardware, providing control over the input/output operations, memory management, and performance optimization.
- **Int 21h (`EMU` Interrupt)**: Used for performing I/O operations such as displaying messages, taking input from the user, and printing the output to the screen.

2.3.3 2.3.3 Implementation Details

The core of the implementation involves the following sections:

2.3.3.1 Encryption Process

The encryption process in the Caesar Cipher algorithm works by shifting each letter in the text by the specified key. For example, if the input is "HELLO" and the key is 3, the output will be "KHOOR". The following code snippet demonstrates the encryption logic:

```
; Encrypt the text
encry proc
    print msg5              ; Prompt for key input
    call input              ; Get the key
    mov key, al             ; Store the key

    xor cx, cx              ; Clear register for loop
    print msg3              ; Prompt for number of characters
    call input              ; Get the number of characters
    mov n, al               ; Store number of characters

    print msg4              ; Prompt for the text input
    mov cl, n               ; Load number of characters into CL register
    mov si, 0               ; Initialize index for text storage
l1:
    call charin             ; Take character input
    mov bl, al              ; Store input character in BL
    add bl, key             ; Shift character by key
    ; Check if character exceeds 'Z' or 'z'
    cmp bl, 'Z'
    jle save1
    sub bl, 26              ; Wrap around if character goes beyond 'Z'

save1:
    mov str[si], bl         ; Save the character to memory
    inc si                  ; Increment index
    loop l1                 ; Loop for all characters
    ret
endp
```

This code handles both uppercase and lowercase letters, wrapping around the alphabet if necessary.

2.3.3.2 Decryption Process

The decryption process is the reverse of encryption, shifting characters in the opposite direction by the key value. The following code snippet demonstrates the decryption process:

```
; Decrypt the text
```

```
decry proc
    print msg5              ; Prompt for key input
    call input              ; Get the key
    mov key, al             ; Store the key

    xor cx, cx              ; Clear register for loop
    print msg3              ; Prompt for number of characters
    call input              ; Get the number of characters
    mov n, al               ; Store number of characters

    print msg4              ; Prompt for the text input
    mov cl, n               ; Load number of characters into CL register
    mov si, 0               ; Initialize index for text storage
l3:
    call charin             ; Take character input
    mov bl, al              ; Store input character in BL
    sub bl, key             ; Shift character by key (opposite direction)
    ; Check if character goes below 'A' or 'a'
    cmp bl, 'A'
    jge save3
    add bl, 26              ; Wrap around if character goes below 'A'

save3:
    mov str[si], bl         ; Save the character to memory
    inc si                  ; Increment index
    loop l3                 ; Loop for all characters
    ret
endp
```

This code is similar to the encryption process but reverses the shift for each character.

2.3.4 Implementation Details

The Caesar Cipher was implemented on the 8086 microprocessor using assembly language. The program accepts user inputs for the text to be encrypted or decrypted, as well as the encryption key. It handles both uppercase and lowercase letters, applying the Caesar Cipher shift accordingly. The program uses EMU interrupt services for input and output operations. The encryption process shifts characters forward by the key value, while the decryption process reverses the shift. This straightforward implementation demonstrates the use of assembly language for cryptographic algorithms on a microprocessor.

2.4 Algorithms

In this section, the algorithms for encryption and decryption of the Caesar Cipher implemented on the 8086 microprocessor using assembly language are detailed. The algorithms outline the basic steps taken to encrypt and decrypt text, along with handling of different types of characters.

2.4.1 Caesar Cipher Algorithm

The Caesar Cipher is a simple encryption technique where each letter in the plaintext is shifted by a fixed number, called the key. It supports both encryption and decryption using the same key but in reverse directions. Non-alphabetical characters remain unchanged during the process.

Algorithm 1: Caesar Cipher Algorithm

```
1  Text to be processed, Key (shift value) Encrypted or Decrypted Text
       Data: Input text string of length n, Shift key k
   /* Encryption Process                                              */
2  if Mode is "Encrypt" then
3      for i = 1 to n do
4          if character x_i is uppercase letter then
5              x_i = (x_i + k)  mod 26
6          else if character x_i is lowercase letter then
7              x_i = (x_i + k)  mod 26
8          else
9              Keep character unchanged
10     Encrypted text
   /* Decryption Process                                              */
11 else if Mode is "Decrypt" then
12     for i = 1 to n do
13         if character x_i is uppercase letter then
14             x_i = (x_i - k + 26)  mod 26
15         else if character x_i is lowercase letter then
16             x_i = (x_i - k + 26)  mod 26
17         else
18             Keep character unchanged
19     Decrypted text
20 else
21     Error: Invalid Mode
```

Chapter 3

Performance Evaluation

This chapter evaluates the performance of the Caesar Cipher implementation on the 8086 microprocessor using the EMU8086 emulator. It includes details on the simulation environment, the results of the encryption and decryption processes, and an analysis of the project's performance.

3.1 Simulation Environment

3.1.1 Simulation Setup

The simulation for this project was conducted using the EMU8086 emulator, which is designed to simulate the 8086 microprocessor. The 8086 assembly code was executed in the emulator to test both the encryption and decryption functionalities of the Caesar Cipher. The inputs (plaintext, ciphertext, and key) were provided through the console, and the output was observed within the emulator.

The steps for setting up the environment were as follows:

- EMU8086 was installed on a compatible system.
- The assembly code for the Caesar Cipher was written and compiled using the EMU8086 assembler.
- The program was run in the emulator, where user inputs for the key and text to be encrypted or decrypted were provided interactively.

3.1.2 Environment Installation

The EMU8086 setup involved:

- Installing the EMU8086 emulator on a Windows-based system.
- Writing the Caesar Cipher program in 8086 Assembly language.

- Running the code in the emulator, providing inputs for the key and text to be encrypted or decrypted, and observing the program's output in the emulator's console.

3.2 Results Analysis

3.2.1 Result

The Caesar Cipher was tested using various text inputs and keys. The output was verified to ensure that:

- Uppercase and lowercase letters were shifted correctly based on the key.
- Non-alphabetic characters (spaces, punctuation) remained unchanged.
- Both encryption and decryption processes functioned as expected with different key values.

The results confirmed that the program performs the encryption and decryption operations as intended. The following screenshots illustrate the encryption and decryption processes.

3.3 Results Overall Discussion

The project was successful in implementing the Caesar Cipher on the 8086 microprocessor using `EMU8086`. Both encryption and decryption processes worked correctly, with the program shifting letters as per the given key and maintaining the integrity of non-alphabetic characters. The results showed that the cipher could operate on small to moderate text inputs efficiently within the constraints of the 8086 microprocessor's resources.

3.3.1 Performance Observations:

- The time taken for encryption and decryption was minimal for shorter texts, but it gradually increased as the text length grew.
- There were no significant performance issues with the emulator when handling typical input sizes.

3.4 Complex Engineering Problem Discussion

The project addresses the problem of implementing a simple cryptographic algorithm on a resource-constrained system, the 8086 microprocessor. The challenge was to ensure that the Caesar Cipher was implemented efficiently, given the limited processing

power and memory of the 8086. The successful implementation of the encryption and decryption functions demonstrates that basic cryptography can be efficiently executed even on older, less powerful hardware using optimized assembly language code.

Chapter 4

Conclusion

4.1 Discussion

The Caesar Cipher was successfully implemented on the 8086 microprocessor using assembly language, with both encryption and decryption working correctly for uppercase, lowercase, and non-alphabetic characters. The project demonstrated that basic cryptographic operations could be efficiently performed even on older, resource-constrained systems. The program handled moderate-sized texts within the time and memory limits of the 8086 microprocessor, providing a valuable understanding of cryptography and assembly language programming on limited hardware.

This project was a valuable learning experience, offering insights into both cryptographic techniques and assembly language programming on the 8086 microprocessor.

4.2 Limitations

While the project achieved the intended goals, there were several limitations:

- The program is limited to handling only alphabetic characters (both uppercase and lowercase) and does not support special characters like digits, punctuation, or other symbols.
- The encryption and decryption are relatively simple and lack advanced security features that modern cryptographic algorithms provide, making the cipher vulnerable to brute-force attacks.
- The 8086 microprocessor has limited memory and processing power, which constrained the size of the input that could be handled efficiently.
- The project relied on a simple key-based shift without incorporating more complex techniques such as key management or dynamic key generation.

4.3 Scope of Future Work

There are several opportunities for expanding and improving this project:

- Implementing more complex cryptographic algorithms, such as the Advanced Encryption Standard (AES), on the 8086 microprocessor or other microprocessors.
- Adding support for additional character sets, including special characters, digits, and symbols, to make the encryption and decryption more versatile.
- Enhancing the security of the Caesar Cipher by incorporating a more dynamic key generation process and implementing key exchange protocols.
- Testing the encryption and decryption algorithms on real-world data to evaluate the performance and security of the implementation in practical scenarios.
- Exploring hardware optimizations to improve the performance of the cryptographic algorithm, such as using specialized instructions or hardware accelerators.

The future work could involve more sophisticated cryptographic methods and enhanced security mechanisms, as well as optimizing the program for better efficiency and scalability on modern systems.

References

1. **M. E. Hellman**, *Cryptography and Data Security*, Addison-Wesley, 1989.
2. **8086 Microprocessor Documentation**: Intel, https://www.intel.com/content/www/us/en/processors/8086-processor.html
3. **R. K. Gupta**, *8086 Microprocessor Programming and Interfacing*, PHI Learning, 2012.
4. **S. Singh**, *Introduction to Cryptography*, Springer, 2009.
5. **Sagufta Sabah Nakshi**, *Project Advisor*.
6. **W. Stallings**, *Cryptography and Network Security: Principles and Practice*, 7th Edition, Pearson, 2017.