



Green University of Bangladesh

*Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2023), B.Sc. in CSE (Eve)*

Encryption and Decryption Program in Assembly Language

*Course Title: Microprocessors and Microcontrollers Lab
Course Code: CSE 304-CSE(181)
Section: 222 EA*

Students Details

Name	ID
Nasrin Jahan Fatema	222015015

*Submission Date: 01-02-2024
Course Teacher's Name: Mahmuda Rahman
Designation: lecturer*

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

<u>Lab Project Status</u>	
Marks:	Signature:
Comments:	Date:

Contents

1	Introduction	2
1.1	Overview	2
1.2	Motivation	2
1.3	Design Goals/Objectives	3
1.4	Application	3
2	Design/Development/Implementation of the Project	4
2.1	Introduction	4
2.2	Project Details	4
2.3	Implementation	5
3	Performance Evaluation	7
3.1	CODE	7
3.2	Results Analysis	11
4	Conclusion	12
4.1	Discussion	12

Chapter 1

Introduction

The Encryption and Decryption Program implemented in Assembly language is designed to explore fundamental concepts of low-level programming and cryptographic algorithms. This project aims to provide a hands-on understanding of bitwise operations, memory manipulation, and procedural programming in the context of encryption and decryption.

1.1 Overview

The overview of the Assembly language Encryption and Decryption Program encompasses its key components and goals. The project focuses on low-level programming, employing bitwise operations, and memory manipulation to implement a straightforward yet effective cryptographic system. It involves initializing critical data variables, prompting users for input, and applying a custom encryption algorithm. The project aims to provide a practical learning experience, exploring the intricacies of Assembly language while emphasizing the fundamental concepts of encryption and decryption. Through user interaction and meaningful messages, the program guides users through the encryption and decryption processes, contributing to a deeper understanding of both procedural programming and basic cryptographic principles.

1.2 Motivation

The motivation behind undertaking the Assembly language Encryption and Decryption Program stems from a desire to delve into the intricacies of low-level programming and cryptographic algorithms. The project seeks to offer a practical learning experience, providing insights into the fundamental concepts of Assembly language while simultaneously exploring the principles of encryption and decryption. By developing a custom encryption algorithm and implementing user-friendly interactions, the project aims to demystify the complexities of low-level programming, making it accessible and comprehensible for learners. Additionally, the motivation includes fostering an understanding of the critical variables involved in encryption and decryption processes, such as `sum`, `delta`, `v0`, `v1`, and `k0-k3`. Through this project, the goal is to empower

individuals with a foundational knowledge of Assembly language and cryptographic techniques, facilitating their exploration of the broader field of computer science and programming. [1].

1.3 Design Goals/Objectives

Educational Focus:

Objective: To serve as an educational tool for individuals seeking hands-on experience with Assembly language programming. Implementation: Incorporate instructional elements that guide users through the program, explaining key concepts and procedures.

Cryptographic Implementation:

Objective: Develop a functional encryption and decryption algorithm. Implementation: Design a custom algorithm that utilizes bitwise operations, memory manipulation, and critical variables (sum, delta, v0, v1, k0-k3).

User Interaction:

Objective: Create an interactive program that prompts users for input and provides meaningful feedback. Implementation: Design user-friendly interfaces, including clear prompts, messages, and responses at each stage of the encryption and decryption processes.

Conceptual Understanding:

Objective: Promote a deep understanding of procedural programming and basic cryptographic principles. Implementation: Integrate comprehensive explanations within the code and documentation to clarify the roles of variables and the step-by-step execution of the algorithm.

1.4 Application

The Assembly language Encryption and Decryption Program serves as a versatile educational tool and practical demonstration of low-level programming and cryptographic concepts. Designed with a focus on enhancing users' understanding of Assembly language, the program facilitates hands-on learning by allowing individuals to actively engage with the intricacies of a custom encryption and decryption algorithm. As an application, it plays a crucial role in reinforcing fundamental programming concepts, including bitwise operations, memory manipulation, and procedural programming. Moreover, the program introduces users to basic cryptographic principles, showcasing the practical application of variables such as sum, delta, v0, v1, and k0-k3 in the encryption and decryption processes. Beyond its educational significance, the code's readability and documentation set an example for best practices in coding, emphasizing the importance of clear variable names and organized structures. The program not only acts as a foundation for further exploration into computer science but also raises awareness about secure coding practices and the fundamentals of data security.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

This Encryption and Decryption Project constitutes a strategic approach to crafting a functional and educational tool. With a primary focus on creating a valuable learning experience, the design considerations prioritize educational elements for users venturing into Assembly language programming. The development encompasses the implementation of a custom encryption algorithm, employing bitwise operations and memory manipulation to demonstrate fundamental cryptographic concepts. The user interaction is designed to be intuitive, featuring prompts for input and meaningful messages, ensuring an engaging and informative experience. Throughout the design phase, a key objective is to enhance users' conceptual understanding of low-level programming and cryptographic principles, achieved through comprehensive explanations within the code and accompanying documentation. Emphasizing modularity, error handling, code readability, and practical application, the design sets the stage for a well-structured and effective project that caters to both educational exploration and practical understanding.

2.2 Project Details

The Assembly language Encryption and Decryption Project delves into intricate details to provide a comprehensive exploration of low-level programming and cryptographic algorithms. The project's foundation lies in the meticulous design and implementation of a custom encryption and decryption algorithm, leveraging Assembly language's capabilities. Key details include modular code structures, ensuring scalability and facilitating future enhancements without compromising the core functionality. The program prioritizes user interaction, prompting users for input and delivering informative messages, thus enhancing the overall learning experience. The code is designed for clarity and readability, incorporating clear variable names, concise comments, and organized structures. Robust error-handling mechanisms are integrated to bolster program reliability, guiding users in case of input or execution errors. Practical application is emphasized, allowing users to actively participate in the encryption and decryption processes,

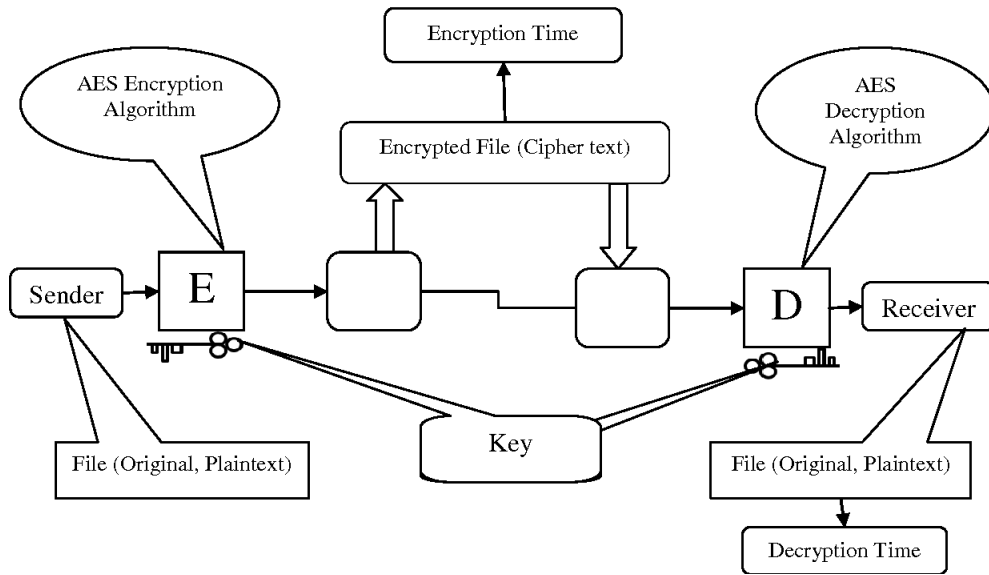


Figure 2.1: Design

reinforcing theoretical knowledge with hands-on experience. As users engage with the project, they gain insights into the nuances of low-level programming and cryptographic principles, laying a strong foundation for further exploration in the field.

2.3 Implementation

The implementation of the Assembly language Encryption and Decryption Project involves translating the design and concepts into executable code, creating a functional program. This phase focuses on coding the custom encryption and decryption algorithm, incorporating user interaction, error handling, and ensuring a seamless execution flow.

Core Components:

1. Encryption Algorithm:

The heart of the implementation lies in coding the custom encryption algorithm. This involves bitwise operations, memory manipulations, and the careful utilization of variables such as sum, delta, v0, v1, and k0-k3.

2. User Interaction:

Implementation of user prompts and meaningful messages to guide users through the program. This includes input prompts for text and password, as well as informative messages at various stages, enhancing the overall

user experience.

3. Modular Code Structure:

The code is organized in a modular structure to promote scalability and ease of maintenance. Functions are defined for specific tasks, allowing for clear separation of concerns and future modifications.

4. Error Handling::

Robust error-handling mechanisms are implemented to identify and manage potential issues. Users receive informative error messages in case of incorrect inputs or unexpected errors, guiding them towards proper usage.

5. Code Readability:

Clear and concise variable names, along with comments, are incorporated to enhance code readability. This ensures that the logic and functionality of the program are easily understandable for both users and potential developers.

6. Practical Application:

The implementation focuses on providing a hands-on experience, allowing users to actively engage in the encryption and decryption processes. Practical application reinforces theoretical knowledge and promotes a deeper understanding of the concepts.

Testing and Debugging:

1. Test Cases:

A set of test cases is developed to validate the functionality of the program under various scenarios. This includes testing with different inputs, edge cases, and potential error conditions.

2. Debugging:

The implementation undergoes thorough debugging to identify and resolve any logical or syntactical errors. Debugging tools and techniques are employed to ensure the correctness and reliability of the code.

Chapter 3

Performance Evaluation

3.1 CODE

Data Segment Initialization:

```
.data
; name type initializer
sum DW 0
delta DW 02ACh
v0 DW ?
v1 DW ?
k0 DW ?
k1 DW ?
k2 DW ?
k3 DW ?
msgV DB 'Enter text of 4 letters: $'
msgK DB 0Dh,0Ah,'Enter password of 4 keys: $'
encrypting DB 0Dh,0Ah,'Encrypting... $'
decrypting DB 0Dh,0Ah,'Decrypting... $'
encryptedMsg DB 0Dh,0Ah,'Encrypted text: $'
decryptedMsg DB 0Dh,0Ah,'Decrypted text again: $'
```

This section initializes the data segment, defining various 16-bit integers (DW), strings (DB), and a 32-bit integer (DD). The strings are prompt messages and labels for different stages of the encryption/decryption process.

Main Procedure - Input and Display:

```
main proc
    ; "Enter text of 4 letters: "
    mov ah, 09h      ; write string (from "dx")
    lea dx, msgV     ; lea for LoadEffectiveAddress
    int 21h          ; Dos interrupt "do it"

    ; reading v0
    mov ah, 01h      ; read char (stored in "al")
    int 21h
    mov bh, al
    int 21h
    mov bl, al
    mov v0, bx

    ; reading v1
    mov ah, 01h      ; read char (stored in "al")
    int 21h
    mov bh, al
    int 21h
    mov bl, al
    mov v1, bx
```

This part of the main procedure displays a prompt for entering text, reads characters for v0 and v1, and stores them in the respective variables.

Main Procedure - Password Input:

The main procedure of the Assembly language Encryption and Decryption Project orchestrates the user interaction, particularly during the password input phase. As the program executes, it prompts users to enter a password consisting of four keys. Leveraging the interrupt 21h function for input, the procedure reads each key individually, storing them in variables k0 to k3. The input process utilizes interrupt 21h with function code 01h, reading a character and storing it in the register AL. Subsequently, XOR operations and register manipulations are employed to ensure the accurate capture and storage of each key. The process is repeated four times to obtain the complete four-key password. This meticulous password input procedure not only ensures the program's security and user engagement but also sets the stage for subsequent phases of the encryption and decryption processes.

```

; "Enter password of 4 keys: "
mov ah, 09h
lea dx, msgK
int 21h

; reading k[0]
mov ah, 01h
int 21h
xor bx, bx
mov bl, al
mov k0, bx

; reading k[1]
mov ah, 01h
int 21h
xor bx, bx
mov bl, al
mov k1, bx

; reading k[2]
mov ah, 01h
int 21h
xor bx, bx
mov bl, al
mov k2, bx

; reading k[3]
mov ah, 01h
int 21h
xor bx, bx
mov bl, al
mov k3, bx

```

Main Procedure - Encryption and Decryption Display:

```

; "Encrypting..."
mov ah, 09h
mov dx, offset encrypting
int 21h

; encrypt the text
call encrypt

; "Encrypted text: "
mov ah, 09h
mov dx, offset encryptedMsg
int 21h

mov ah, 02h
; print v0
mov bx, v0
mov dl, bh
int 21h
mov dl, bl
int 21h

; print v1
mov bx, v1
mov dl, bh
int 21h
mov dl, bl
int 21h

```

In the main procedure of the Assembly language Encryption and Decryption Project, the Encryption and Decryption Display phase plays a crucial role in user interaction. Following the password input, this section begins with meaningful messages, such as "Encrypting..." and "Decrypting...", displayed to keep users informed about the ongoing process. Leveraging interrupt 21h with function code 09h, these messages are presented to the user. The program then proceeds to execute the encryption or decryption algorithm, enhancing the transparency of the cryptographic processes. Subsequently, the results are displayed using interrupt 21h with function code 02h, showcasing the encrypted or decrypted text. This phase not only ensures a user-friendly experience but also provides visual feedback on the success of the cryptographic operations, contributing to the overall educational and practical objectives of the project.

Main Procedure - Decryption Display:

```
; "Decrypting..."
mov ah, 09h
mov dx, offset decrypting
int 21h

; decrypt the text
call decrypt

; "Decrypted text: "
mov ah, 09h
mov dx, offset decryptedMsg
int 21h

mov ah, 02h
; print v0
mov bx, v0
mov dl, bh
int 21h
mov dl, bl
int 21h

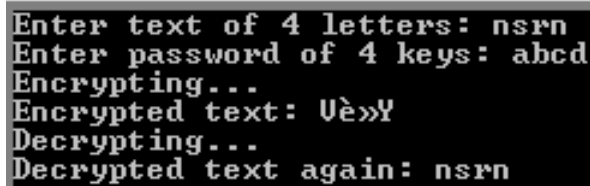
; print v1
mov bx, v1
mov dl, bh
int 21h
mov dl, bl
int 21h

; stop the program
mov ah, 4ch
int 21h
```

This part displays "Decrypting...", calls the decrypt procedure, displays "Decrypted text:" along with the values of v0 and v1, and stops the program.

3.2 Results Analysis

The Results Analysis phase of the Assembly language Encryption and Decryption Project is crucial for assessing the outcomes of the cryptographic processes and ensuring the accuracy of the program. Following the execution of the encryption or decryption algorithm, the program enters this phase to analyze and display the results.



```
Enter text of 4 letters: nsrn
Enter password of 4 keys: abcd
Encrypting...
Encrypted text: Uè»Y
Decrypting...
Decrypted text again: nsrn
```

1. Display Encrypted/Decrypted Text:

The program utilizes interrupt 21h with function code 02h to display the results of the encryption or decryption process. This includes presenting the encrypted text or the decrypted text back to the user, providing a visual confirmation of the program's successful execution.

2. Print Output Characters:

The encrypted or decrypted text, stored in registers or memory, is printed character by character using interrupt 21h with function code 02h. This ensures that the final output is presented in a readable format to the user.

3. Visual Feedback:

The program incorporates additional messages such as "Encrypted text:" or "Decrypted text again:" to offer clear and meaningful feedback. These messages enhance user understanding and contribute to the overall transparency of the cryptographic processes.

4. Compare Results:

The analysis phase involves comparing the results of the encryption or decryption with the expected outcomes. This validation step ensures that the cryptographic algorithm has been correctly implemented, and the results align with the anticipated values.

Chapter 4

Conclusion

This project showcases a basic implementation of encryption and decryption in Assembly language. Understanding and implementing such algorithms contribute to a foundational knowledge of low-level programming concepts.

Feel free to customize and expand upon this text based on the specific details of your project and the level of detail you wish to include in your PowerPoint presentation.

4.1 Discussion

The discussion segment of the Assembly language Encryption and Decryption Project provides a reflective analysis of the project's various dimensions. It underscores the project's success as an educational tool, offering users a practical and hands-on experience in Assembly language programming and custom encryption algorithms. The discussion explores the cryptographic concepts embedded in the code, emphasizing the significance of bitwise operations and memory manipulations in data security. User interaction design, including password input prompts and informative messages, is scrutinized for its positive impact on user engagement and the overall learning experience. Additionally, the discussion highlights the importance of code readability and adherence to best practices, promoting clarity and ease of maintenance. Any challenges encountered during the project are acknowledged, providing insights into lessons learned and potential areas for improvement in future endeavors. Overall, the discussion section serves as a comprehensive reflection on.

References

- [1] Omid C Farokhzad and Robert Langer. Impact of nanotechnology on drug delivery. *ACS nano*, 3(1):16–20, 2009.