Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2022), B.Sc. in CSE (Day)

Monoalphabetic-substitution-encryption-system

Course Title: Microprocessor and microcontroller Lab
Course Code: CSE 304
Section: 222-D10

Students Details

Name	**ID**
Md. Tanvir Alam Anik	221902283
Maruf Hossain	221902318

Submission Date: 24 Dec 2024
Course Teacher's Name: Sagufta Sabah Nakshi

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

Lab Project Status

Marks: **Signature:**

Comments: **Date:**

Contents

Chapter 1

Introduction

1.1 Overview

This project focuses on the development of a software application that implements the Monoalphabetic Substitution Cipher using assembly language. The software allows the user to input text, which is then converted into ciphertext for encryption, or the ciphertext is converted back into plaintext for decryption. The project focuses on providing a simple encryption and decryption system for short strings of text, ranging from 1 to 9 characters. Monoalphabetic substitution is a type of encryption where each letter of the plaintext is substituted with a corresponding letter from a fixed alphabet.

1.2 Motivation

In today's world, data security and privacy are crucial, especially with the increasing use of digital communication. Although modern cryptography methods are complex, learning about simpler encryption techniques helps in understanding the basics of how secure communication works. I chose to implement a Monoalphabetic Substitution Cipher using assembly language because it helps in learning about low-level programming, the structure of encryption algorithms, and how computers handle text and numbers at the hardware level.

1.3 Problem Definition

The main problem being solved by this project is to create a simple system that can both encrypt and decrypt text using a Monoalphabetic Substitution Cipher. The input text is small, with a length between 1 to 9 characters. The challenge is to handle the encryption and decryption processes efficiently in assembly language, which requires a strong understanding of low-level programming and memory management. The program should be user-friendly, accepting inputs, processing them correctly, and providing the expected output (ciphertext or plaintext).s

1.4 Objectives

1. **Implement a Monoalphabetic Substitution Cipher:** Create an algorithm that converts plaintext to ciphertext and vice versa using a fixed substitution key.
2. **User Interaction:** Develop a simple interface where the user can input text and receive either the encrypted or decrypted output.
3. **Efficiency:** The program should handle text input of 1 to 9 characters efficiently and without errors.
4. **Assembly Languages: ss** Implement the cipher purely in assembly language to understand its functioning at a lower hardware level.

1.5 Application

1. **Educational Purpose:** This project helps users understand basic encryption methods and the fundamentals of cryptography.
2. **Data Protection:** Though Monoalphabetic substitution is not secure for modern standards, it demonstrates how text can be manipulated for the purpose of confidentiality.
3. **Learning Assembly Language:** This project is also useful for those learning assembly language programming, as it demonstrates how to handle string manipulation and mathematical operations at a low level.

Chapter 2

Implementation and Result

2.1 Implementation

```
include "emu8086.inc"

 encryption macro
     Lea dx, string
     mov ah, 09h
     int 21h


     mov ah, 01
     int 21h
     sub al, 48
     mov ah, 0
     mov bx, ax

     LEA DX, NEWLINE
     MOV AH, 09H
     INT 21H

     mov cx, bx
     mov textSize, cx
     mov si, 0

     LEA DX, enterTextString
     MOV AH, 09H
     INT 21H




     takeInputFromUsers:
        mov ah, 01
        int 21h
```

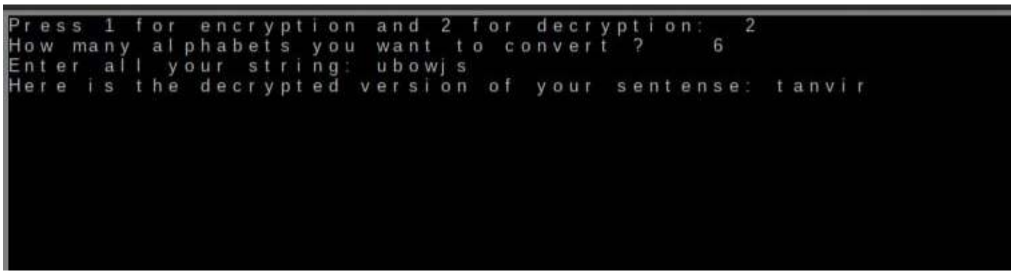
```
   mov ah, 0
   sub al,48
   mov array[si],al
   inc si
loop takeInputFromUsers


LEA DX, NEWLINE
MOV AH, 09H
INT 21H




LEA DX, encryptedString
MOV AH, 09H
INT 21H


mov cx,textSize
mov si,0
printEncryptedText:
   mov dl, array[si]
   add dl, 48

   cmp dl, 'z'
     je printz
   cmp dl, ' '
     je printSpace
     add dl, 1
     mov ah, 02h
     int 21h
     inc si


     jmp go:
     printz:
     sub dl, 25   ; z --> a
     mov ah, 02h
     int 21h
     inc si
     jmp go:


     printSpace:
     mov ah, 02h
     int 21h
     inc si
```

```
        go:

    loop printEncryptedText


ENDM




decryption macro
    Lea dx, string
    mov ah, 09h
    int 21h


    mov ah, 01
    int 21h
    sub al, 48
    mov ah, 0
    mov bx, ax

    LEA DX, NEWLINE
    MOV AH, 09H
    INT 21H

    mov cx, bx
    mov textSize, cx
    mov si, 0

    LEA DX, enterTextString
    MOV AH, 09H
    INT 21H



    takeInputFromUsersForDec:
       mov ah, 01
       int 21h
       mov ah, 0
       sub al,48
       mov array[si],al
       inc si
    loop takeInputFromUsersForDec


    LEA DX, NEWLINE
    MOV AH, 09H
    INT 21H



    LEA DX, decriptedString
    MOV AH, 09H
    INT 21H


    mov cx,textSize
    mov si,0
    printdeccryptedText:
       mov dl, array[si]
       add dl, 48

       cmp dl, ’a’
         je printa
       cmp dl, ’ ’
         je printSpaces
         sub dl, 1
         mov ah, 02h
         int 21h
         inc si


         jmp goes:
         printa:
         add dl, 25
         mov ah, 02h
         int 21h
         inc si
         jmp goes:


         printSpaces:
         mov ah, 02h
         int 21h
         inc si


         goes:

    loop printdeccryptedText


ENDM
```

```
org 100h

    .DATA
       string dw "How many alphabets you want to convert ?    $"
       encryptedString dw "Here is the encrypted version of your sentense : $"
       enterTextString dw "Enter all your string : $"
       decriptedString dw "Here is the decrypted version of your sentense : $"


       NEWLINE DB 0Dh, 0Ah, "$"
       array db 30 DUP(?)
       textSize dw 0


       whatYouWannado db "Press 1 for encryption and 2 for decryption :  $"
       selection db 0

    .CODE
       MOV AX, @DATA
       MOV DX, AX

       Lea dx, whatYouWannado
       mov ah, 09h
       int 21h

       mov ah, 01
       int 21h
       sub al, 48

       mov selection, al

       Lea dx, NEWLINE
       mov ah, 09h
       int 21h

       mov ax, 0
       mov bx, 0


       cmp selection, 1
          je encr
       cmp selection, 2
          je decr

       print "Invalid input selection ..!"
       mov ah, 04ch
       int 21h

       encr:
        encryption
        ret

       decr:
        decryption
        ret

ret
```

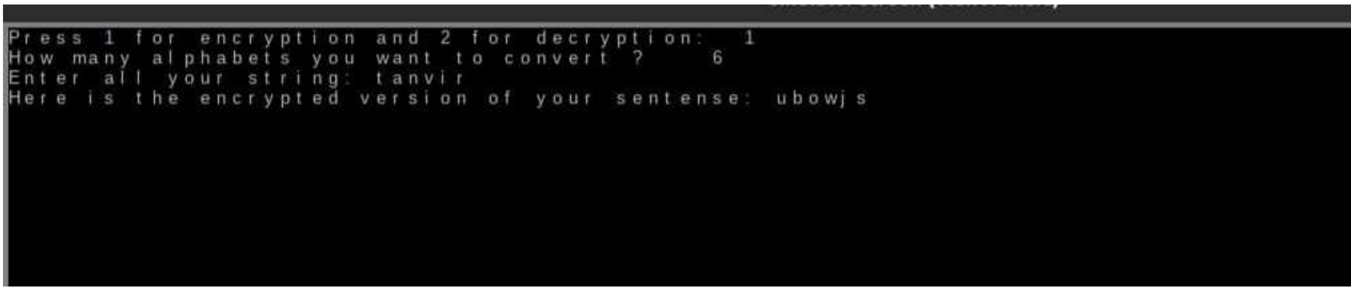
2.2 Result

Encryption and Decryption Result :



```
Press 1 for encryption and 2 for decryption:  2
How many alphabets you want to convert ?    6
Enter all your string: ubowjs
Here is the decrypted version of your sentense: tanvir
```

Figure 2.1: Decryption



```
Press 1 for encryption and 2 for decryption:  1
How many alphabets you want to convert ?    6
Enter all your string: tanvir
Here is the encrypted version of your sentense: ubowjs
```

Figure 2.2: Encryption

Chapter 3

Conclusion

3.1 Discussion

The Monoalphabetic Substitution Cipher is one of the simplest encryption techniques. In this project, the plaintext input from the user is converted into ciphertext by substituting each character with a predefined character from a different alphabet (e.g., shifting letters in the alphabet or using a random key). For decryption, the reverse process is applied using the same key. Since each letter in the plaintext is substituted with one fixed letter in the ciphertext, the security of this system is limited.

Assembly Language Considerations:

Working with assembly language involves manually managing memory and understanding the CPU's instruction set, which makes it more challenging but also educational. The challenge is to correctly implement the algorithm, perform character manipulations, and manage data flow, all while keeping the code simple and efficient.

3.2 Limitations

Security: Monoalphabetic substitution is not secure for modern applications because the pattern of the cipher can easily be broken by frequency analysis, especially with longer texts.
Input Length: The program currently supports only 1 to 9 characters of input. This limits its usability for larger texts, and further development would be required to support longer inputs.
Fixed Cipher Key: The substitution is based on a static key or pattern. This means that if an attacker knows the key, they can easily decrypt the text.
No Error Handling: The software does not handle errors well, such as invalid inputs (non-alphabetical characters) or exceeding the character limit.

3.3 Scope of Future Work

Support for Longer Inputs: One possible improvement would be to allow the program to handle longer inputs (more than 9 characters).
Key Generation and Randomization: Instead of using a fixed key, future versions could implement a random key generator, enhancing the security of the cipher.
Error Handling and User Interface: The program could be improved by adding better error handling, such as checking for invalid characters or input length, and providing a more user-friendly interface.
Stronger Encryption Algorithms: While Monoalphabetic Substitution is a good starting point, the project could evolve to incorporate more advanced cryptographic algorithms, such as transposition ciphers or even modern encryption techniques like AES.
Cross-platform Support: Currently, the program is written in assembly language, which may limit portability. Future work could involve creating similar functionality in higher-level languages for broader application.

3.4 References

- GitHub.