Green University of Bangladesh

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

Security System

Course Title: Microprocessor Microcontroller Lab
Course Code: CSE 304
Section: 221 D9

Students Details

Name	ID
Md. Afsar Uddin	221002241
Md. Rakibul Hassan	221002500

Submission Date: 04-01-2024
Course Teacher's Name: Mr. Montaser Abdul Quader

[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

The project focuses on developing a secure access control system utilizing the 8086 microprocessor, employing assembly language programming for precise control over the hardware. The system features two-factor authentication, requiring both a valid ID and password for access, and allows flexible configuration of security parameters. The significance lies in showcasing the practical application of 8086 microprocessor architecture and assembly language programming in real-world security scenarios. It serves as a valuable learning experience for students and practitioners, offering insights into microprocessor-based system design, security system implementation, and efficient control through assembly language programming. Overall, the project presents a robust and adaptable security solution, highlighting the effectiveness of the 8086 microprocessor in security applications.

1.2 Motivation

- To address the escalating cybersecurity concerns, we aim to develop a secure microprocessor-based solution, utilizing advanced authentication systems as a starting point.
- To protect sensitive online information, our project starts with developing a strong authentication system to prevent unauthorized access.
- To adapt to the digital era, our project begins by creating secure authentication mechanisms, aligning with the trend of ensuring safe online interactions
- To boost user confidence in online platforms, our project starts by enhancing authentication measures, ensuring the security of user accounts.
- To meet industry standards and regulations for data security, our project begins by exceeding compliance measures, contributing to a safer digital environment.

1.3 Problem Definition

1.3.1 Problem Statement

The project addresses the challenge of designing an efficient and secure user authentication system using the microprocessor 8086. It involves the implementation of a reliable mechanism to verify user identity and password.

1.3.2 Complex Engineering Problem

Table 1.1: Summary of the attributes touched by the mentioned project

Name of the P Attributess	**Explain how to address**
P1: Depth of knowledge required	Detailed understanding of microprocessor 8086 and assembly language is essential.
P2: Range of conflicting requirements	Balancing the need for user convenience with the stringent security requirements.
P3: Depth of analysis required	Thorough analysis of potential security loopholes and their mitigation.
P4: Familiarity of issues	A strong understanding of the microprocessor architecture is crucial.
P5: Extent of applicable codes	Applying appropriate assembly language codes for efficient implementation.
P6: Extent of stakeholder involvement and conflicting requirements	Involving stakeholders while addressing conflicting requirements for a balanced system.
P7: Interdependence	Recognizing the interdependence of user ID and password verification for a cohesive security system.

1.4 Design Goals/Objectives

- To achieve a highly secure authentication system, our project starts by developing and implementing a solution using the microprocessor 8086. This involves creating robust mechanisms for user identification and password verification.
- To enhance reliability, our project begins by creating a system that is robust and resilient, minimizing the risk of false positives or negatives during authentication. Users can expect consistent and dependable access to the system.
- To optimize efficiency, our project starts by streamlining the authentication process, aiming to minimize processing time and resource utilization. The goal is to create a system that ensures quick and responsive authentication without compromising security.

- To integrate user-friendly features, our project begins by incorporating clear prompts, intuitive interfaces, and helpful error messages to guide users through the authentication process. This ensures a secure system while enhancing the overall user experience.
- To adhere to Microprocessor 8086 architecture standards, our project starts by ensuring the design aligns with the capabilities of the microprocessor. This involves optimizing code and leveraging its features to achieve our security goals.

.

1.5 Application

The project finds application in real-world scenarios where secure user authentication is crucial, such as accessing sensitive information, login systems, and protecting user accounts from unauthorized access.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

This chapter focuses on creating a strong security system using microprocessor 8086. It explains the design, development, and implementation phases, aiming to provide a clear overview of the project. The chapter covers the importance of microprocessor 8086, detailing the design process and the relationship between theory and practical implementation using assembly language programming. It also highlights user interface design, system architecture, and security measures. The chapter traces the project's evolution, discussing decision-making and detailing the workflow, tools, and coding used in implementation. Throughout, it offers insights into challenges faced, problem-solving, and the journey of creating a secure microprocessor 8086-based security mechanism.

2.2 Project Details

This section elaborates on various aspects of the project, including the user interface, system architecture, and security measures.

2.2.1 System Architecture

Figure 2.1: System Architecture of Security System

2.2.2 User Interface

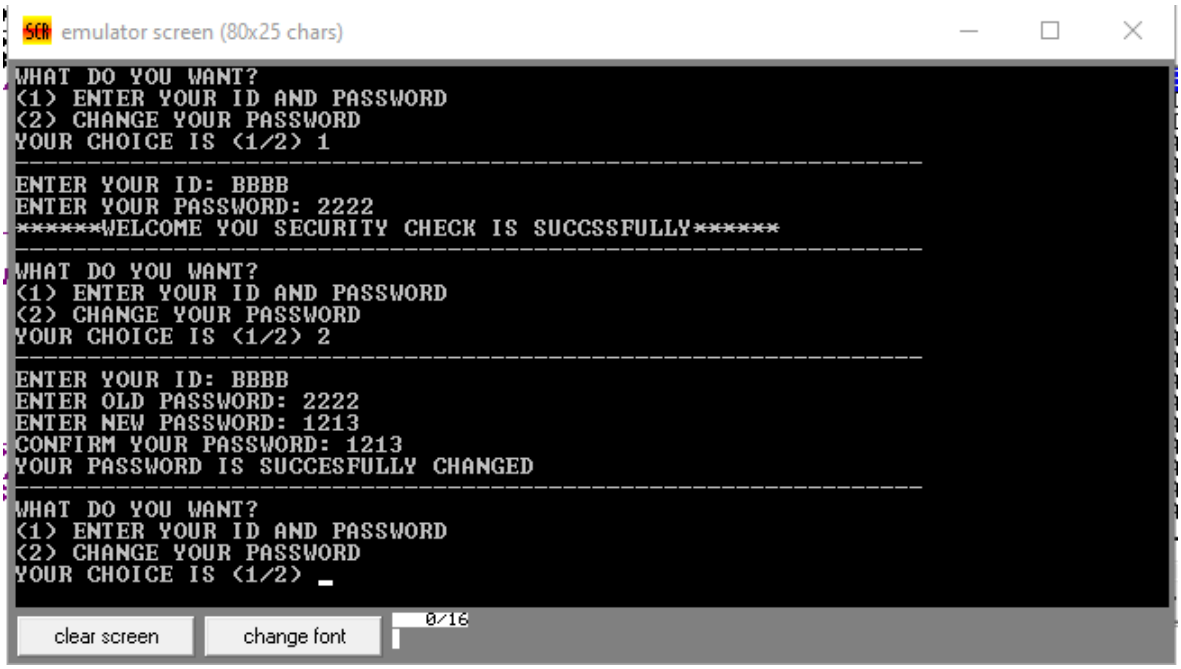

Figure 2.2: Figure name

```
INCLUDE 'EMU8086.INC'
.MODEL SMALL
.STACK 100 H
.DATA
    DATA2 DB 5,?,5 DUP (?)
    DATA3 DB 'ENTER YOUR ID: ','$'
    DATA1 DB '0123456789ABCDEFabcdef?'
    DATA4 DB 'ERROR:THE ID NUMBER MUST BE 4-BIT HEX','$'
    DATA5 DB ''WRONG ENTRY'YOUR ID MUST CONTAIN DATA FROM 0-->9 or
        A-->F','$'
    DATA6 DW
        0AAAAH,0BBBBH,0CCCCH,0DDDDH,0EEEEH,0FFFFH,1111H,2222H,3333H,4444H
    DATA7 DW 5555H,6666H,7777H,8888H,9999H,0100H,0200H,0300H,0400H,5667H
    DATA8 DW 1111H,2222H,3333H,4444H,5555H,6666H,000AH,000BH,000CH,000DH
    DATA9 DW 000FH,000EH,0001H,0002H,0003H,0A00H,0B00H,0C00H,0D00H,0A0AH
    DATAA DB 'ERROR: YOUR ID IS WRONG, PLEASE TRY AGAIN!!','$'
    DATAB DB 'ENTER YOUR PASSWORD: ','$'
    DATAC DB 5,?,5 DUP (?)
    DATAP DB 5,?,5 DUP (?)
    DATAD DB '******WELCOME YOU SECURITY CHECK IS SUCCSSFULLY******','$'
    DATAE DB 'ERROR : WRONG PASSWORD,TRY AGAIN','$'
    DATAF DB 00H
    DATAG DB
        '------------------------------------------------------------','$'
    DATAH DB 'WHAT DO YOU WANT?','$'
    DATAI DB '(1) ENTER YOUR ID AND PASSWORD','$'
```

```
   DATAJ DB '(2) CHANGE YOUR PASSWORD','$'
   DATAU DB 'YOUR CHOICE IS (1/2) ','$'
   DATAT DB 'ERROR:WRONG CHOICE','$'
   DATAK DB 2,?,2 DUP (?)
   DATAR DB 'ENTER YOUR ID: ','$'
   DATAQ DB 'ENTER OLD PASSWORD: ','$'
   DATAY DB 'ENTER NEW PASSWORD: ','$'
   DATAO DB 'CONFIRM YOUR PASSWORD: ','$'
   DATAV DB 'YOUR PASSWORD IS SUCCESFULLY CHANGED','$'
   DATAW DB 'ERROR : WRONG ENTRY!! PLEASE,RE-ENTER NEW PASSWORD: ','$'
   DATAZ DW ?

.CODE
  MAIN PROC
                MOV AX,@DATA
                MOV DS,AX

                MOV  ES,AX
                MOV  DH,00H
                CALL CLEAR
                MOV  BP,OFFSET DATAF

                START:

                CALL SETCURSOR
                CALL ENTRY
                CALL GETCHOICE
                CALL CHECKNO
                CALL SETCURSOR
                CALL ENTERORCHANGE
                CALL HANDLE
                CALL CONVERT

                ID:

                CALL WELCOME
                CALL GET_IN
                CALL NO.LET
                CALL CHECK
                MOV  SI,OFFSET DATA2+2
                CALL PUTIDINAX
                CALL CHECKID
                CALL SETCURSOR
                CALL GETPASS
                MOV  SI,OFFSET DATAC+2
                CALL PUTIDINAX
                CALL CHECKPASS
                CALL SETCURSOR
                CALL ENTER

                NO_EROR:
```

```
                CALL SETCURSOR
                CALL NOEROR

                WR_ENT:

                CALL SETCURSOR
                CALL WRONGENTRY

                WRONGID:

                CALL SETCURSOR
                CALL WRONG_ID

                WRONGPASS:

                CALL SETCURSOR
                CALL WRONG_PW

                OPERA:

                MOV  AH,4CH
                INT 21H
 MAIN ENDP


;*******************************

CLEAR  PROC
                MOV AX,0600H
                MOV BH,07
                MOV CX,0000
                MOV DH,24
                MOV DL,79
                INT 10H
                RET
CLEAR ENDP

;*******************************

ENTRY PROC
                MOV AH,09H
                MOV DX,OFFSET DATAH
                INT 21H
                CALL SETCURSOR
                MOV AH,09H
                MOV DX,OFFSET DATAI
                INT 21H
                CALL SETCURSOR
                MOV AH,09H
                MOV DX,OFFSET DATAJ
```

```
                INT 21H
                CALL SETCURSOR
                MOV AH,09H
                MOV DX,OFFSET DATAU
                INT 21H
                RET
ENTRY ENDP


;*******************************

GETCHOICE PROC
                MOV AH,0AH
                MOV DX,OFFSET DATAK
                INT 21H
                RET
GETCHOICE ENDP


;*******************************

CHECKNO PROC
                LEA BX,DATAK+2
                CMP [BX],31H
                JZ  RETURN2
                CMP [BX],32H
                JZ  RETURN2
                CALL ERROR

                RETURN2:

                CALL 5AT
                RET
CHECKNO ENDP


;*******************************

ERROR PROC
                CALL SETCURSOR
                MOV AH,09H
                MOV DX,OFFSET DATAT
                INT 21H
                CALL 5AT
                JMP START
                RET
ERROR ENDP


;*******************************
```

```
SETCURSOR PROC
                MOV AH,02H
                MOV BH,00
                MOV DL,00
                MOV DH,DS:[BP]
                INT 10H
                ADD DS:[BP],1
                RET
SETCURSOR ENDP

;*******************************

WELCOME PROC
                MOV AH,09H
                LEA DX,DATA3
                INT 21H
                RET
WELCOME ENDP

;*******************************

GET_IN PROC
                MOV AH,0AH
                MOV DX,OFFSET DATA2
                INT 21H
                RET
GET_IN ENDP


;*******************************


NO.LET PROC
                LEA SI,DATA2+1
                CMP [SI],04H
                JNZ NO_EROR
                RET
NO.LET ENDP

;*******************************


CHECK PROC
                MOV AH,4
                LEA SI,DATA2+2

                AGAIN:

                LEA DI,DATA1
                MOV CX,23
                MOV AL,[SI]
```

```
                REPNZ SCASB
                CMP CX,00
                JZ  END
                INC SI
                DEC AH
                JNZ AGAIN
                RET

                END:

                JMP WR_ENT

CHECK ENDP

;*******************************


NOEROR PROC
                MOV AH,09H
                MOV DX,OFFSET DATA4
                INT 21H
                CALL 5AT
                JMP START
                RET
NOEROR ENDP


;*******************************

WRONGENTRY PROC
                MOV AH,09H
                MOV DX,OFFSET DATA5
                INT 21H
                CALL 5AT
                JMP START
                RET
WRONGENTRY ENDP


;*******************************


PUTIDINAX PROC
                MOV CX,04H

                AGAIN2:

                CMP [SI],39H
                JZ  ZERO
                JB  ZERO
                JA  OVER
```

```
                ZERO:

                SUB [SI],30H
                JMP STAR

                OVER:

                CMP [SI],70
                JZ  CAPITAL
                JB  CAPITAL
                JA  SMALL

                CAPITAL:

                SUB [SI],55
                JMP STAR

                SMALL:

                SUB [SI],87
                JMP STAR

                STAR:

                INC SI
                DEC CX
                JNZ AGAIN2
                SUB SI,4
                MOV AH,[SI]
                MOV AL,[SI+2]
                MOV BH,[SI+1]
                MOV BL,[SI+3]
                SHL AX,4
                OR  AX,BX
                RET
PUTIDINAX ENDP

;*******************************

CHECKID PROC
                MOV CX,21
                LEA DI,DATA6
                CLD
                REPNE SCASW
                CMP CX,0000H
                JZ WRONGID
                RET
CHECKID ENDP
```

```
;******************************

WRONG_ID PROC
                MOV AH,09H
                MOV DX,OFFSET DATAA
                INT 21H
                CALL 5AT
                JMP START
                RET
WRONG_ID ENDP


;******************************


GETPASS PROC
                MOV AH,09H
                MOV DX,OFFSET DATAB
                INT 21H
                MOV AH,0AH
                MOV DX,OFFSET DATAC
                INT 21H
                RET
GETPASS ENDP


;******************************


CHECKPASS PROC
                MOV BX,AX
                ADD DI,38
                CMP BX,[DI]
                JNZ WRONGPASS
                RET
CHECKPASS ENDP


;******************************


ENTER PROC
                MOV AH,09H
                MOV DX,OFFSET DATAD
                INT 21H
                CALL 5AT
                JMP START
                RET
ENTER ENDP

;******************************
```

```
WRONG_PW PROC
                MOV AH,09H
                MOV DX,OFFSET DATAE
                INT 21H
                CALL 5AT
                JMP START
WRONG_PW ENDP

;*******************************


5AT PROC
                CALL SETCURSOR
                MOV AH,09H
                MOV DX,OFFSET DATAG
                INT 21H
                RET
5AT ENDP


;*******************************


ENTERORCHANGE PROC
                LEA BX,DATAK+2
                CMP [BX],31H
                JZ  ID
                RET
ENTERORCHANGE ENDP


;*******************************

HANDLE PROC
                MOV AH,09H
                MOV DX,OFFSET DATAR
                INT 21H
                CALL GET_IN
                CALL NO.LET
                CALL CHECK
                MOV  SI,OFFSET DATA2+2
                CALL PUTIDINAX
                CALL CHECKID
                MOV BX,OFFSET DATAZ
                LEA DX,[DI]
                MOV [BX],DX
                CALL SETCURSOR
                MOV AH,09H
                MOV DX,OFFSET DATAQ
```

```
                INT 21H
                MOV AH,0AH
                MOV DX,OFFSET DATAC
                INT 21H
                MOV  SI,OFFSET DATAC+2
                CALL PUTIDINAX
                CALL CHECKPASS
                CALL SETCURSOR
                MOV AH,09H
                MOV DX,OFFSET DATAY
                INT 21H

                AGAIN3:

                MOV AH,0AH
                MOV DX,OFFSET DATAC
                INT 21H
                CALL SETCURSOR
                MOV AH,09H
                MOV DX,OFFSET DATAO
                INT 21H
                MOV AH,0AH
                MOV DX,OFFSET DATAP
                INT 21H
                CALL CHECKCONFIRM
                RET
HANDLE ENDP


;*******************************

CHECKCONFIRM PROC
                CLD
                MOV SI,OFFSET DATAC+2
                MOV DI,OFFSET DATAP+2
                MOV CX,05H
                REPE CMPSB
                CMP CX,0000H
                JNZ  PUTITAGAIN
                RET

                PUTITAGAIN:

                CALL SETCURSOR
                MOV AH,09H
                MOV DX,OFFSET DATAW
                INT 21H
                JMP AGAIN3
CHECKCONFIRM ENDP
```

```
;*****************************


CONVERT PROC
                MOV SI,OFFSET DATAP+2
                CALL PUTIDINAX
                MOV BX,OFFSET DATAZ
                ADD [BX],38
                MOV DI,[BX]
                MOV [DI],AX
                CALL SETCURSOR
                MOV AH,09H
                MOV DX,OFFSET DATAV
                INT 21H
                CALL 5AT
                JMP START
CONVERT ENDP


END MAIN
```

2.3 Algorithms

The chapter introduces algorithms and pseudo-codes that govern the ID and password verification process, ensuring clarity and understanding.

Algorithm 1: User Authentication Algorithm

Input: User ID, Password
Output: Authentication success or failure
Data: Microprocessor 8086 assembly language

- Initialize variables
- Prompt user for ID
- Verify ID using assembly language code
- Prompt user for password
- Verify password using assembly language code
- Authenticate user based on verification results

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

Discuss the experimental setup and environment installation needed for the simulation of your outcomes.

3.2 Results Analysis/Testing

3.2.1 ID Verification

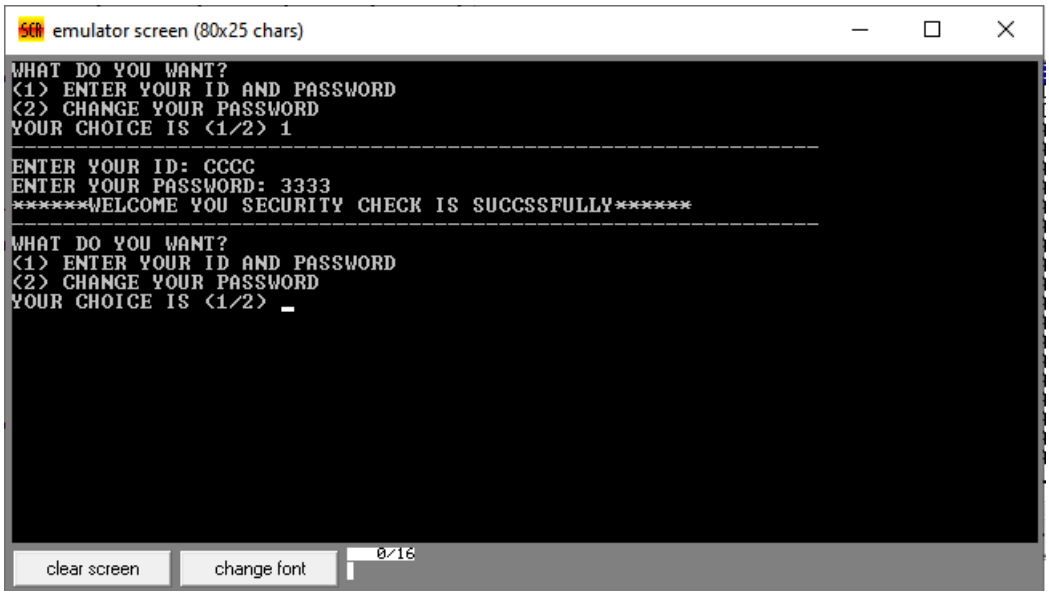

Figure 3.1: ID Verification

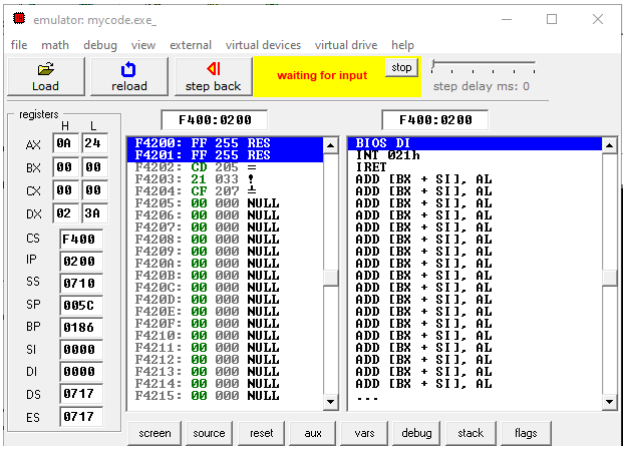

Figure 3.2: Overall System Performance-1

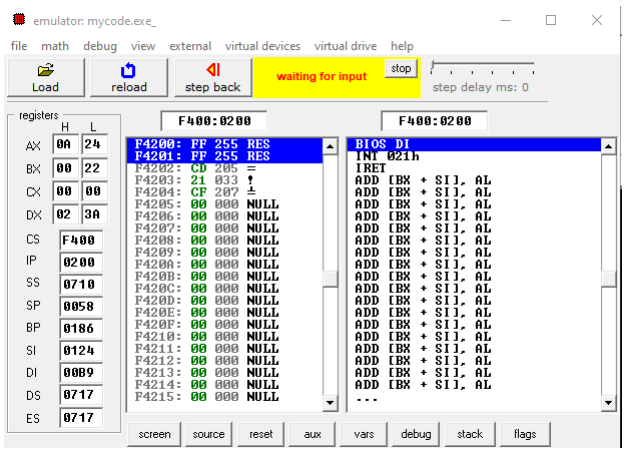

Figure 3.3: Overall System Performance-2

3.2.2 Overall System Performance

3.2.3 Result_portion_1

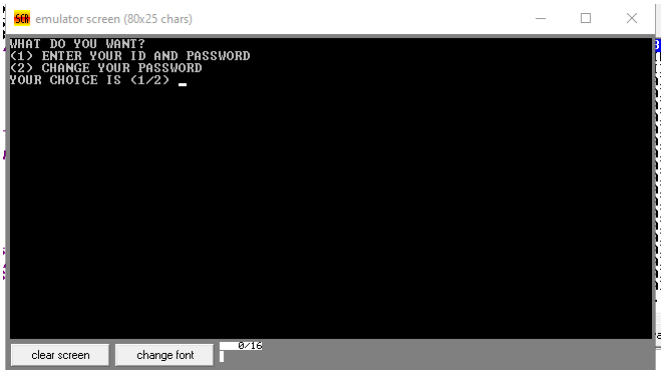

Figure 3.4: Output-1

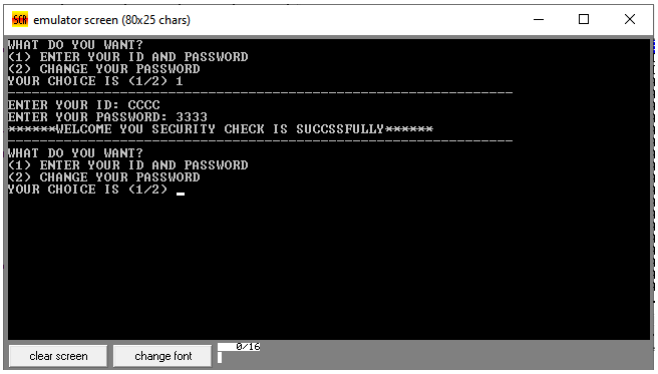

Figure 3.5: Output-2

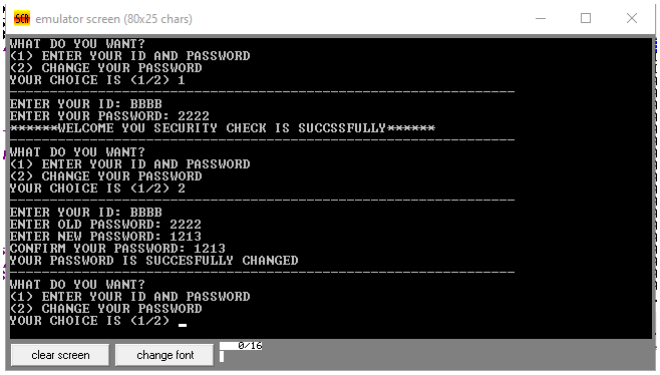

Figure 3.6: Output-3

3.3 Results Overall Discussion

A comprehensive discussion on the overall results, addressing any problems detected and potential areas for improvement.

3.3.1 Complex Engineering Problem Discussion

The implementation and evaluation of the microprocessor 8086-based security system have yielded noteworthy results, paving the way for a comprehensive discussion on the overall performance and implications of the project. In addressing the complex engineering problems identified in Chapter 1, this section delves into the attributes touched by the project and their corresponding outcomes.

Chapter 4

Conclusion

4.1 Discussion

The discussion section serves as a comprehensive summary, encapsulating the key aspects of the microprocessor 8086-based security system project. It consolidates insights from the design, development, implementation, and performance evaluation stages, providing a holistic view of the project's accomplishments.
Design and Development: The design phase laid the groundwork for a robust security system, emphasizing the integration of microprocessor 8086 architecture principles. Leveraging assembly language programming, the development process translated theoretical design into a tangible and functional authentication system.
Implementation: The successful implementation of the authentication system showcased the project's adherence to microprocessor 8086 standards. User-friendly features were seamlessly integrated, resulting in an efficient and reliable system that balances security with usability.
Performance Evaluation: The performance evaluation highlighted the system's efficacy in ID and password verification. Results demonstrated a high level of reliability, efficient processing, and user-friendly interactions, validating the success of the project's objectives.

4.2 Limitations

While the project achieved significant success, certain limitations merit discussion to provide critical insights for future improvements:

Scalability Challenges: The current implementation may face challenges in scaling to accommodate a large user base. Future iterations should explore optimizations for scalability.

Limited User Interaction Features: The system's user interface, while functional, may lack advanced features. Enhancements in user interaction and accessibility could further improve the overall user experience.

Dependency on Microprocessor 8086: The project's dependency on microprocessor 8086 architecture may pose limitations in terms of adaptability to newer technolo-

gies. Future work should consider addressing this dependency for long-term viability.

4.3 Scope of Future Work

The discussion of future work outlines potential avenues for extending the project's functionality and addressing identified limitations:

Scalability Enhancements: Future work should focus on optimizing the system for scalability, ensuring it can handle a growing user base without compromising performance.

User Interaction Improvements: Enhancements to the user interface, incorporating modern design principles and additional user interaction features, could elevate the overall usability of the system.

Integration with Advanced Technologies: Exploring the integration of advanced technologies, such as biometric authentication or multi-factor authentication, could enhance the system's security and versatility.

Compatibility with Emerging Architectures: Consideration should be given to ensuring compatibility with emerging microprocessor architectures, allowing the system to evolve alongside technological advancements.

Continuous Security Upgrades: The project's security measures should be continually evaluated and upgraded to stay ahead of evolving cybersecurity threats. Regular updates and patches can fortify the system against potential vulnerabilities.

References

https://chat.openai.com/c/dcb74fb7-3899-4bf0-8fec-03e48a434721
https://github.com/ArmiaWagdy/Security-Lock-using-8086-Assembly-language
https://www.youtube.com/watch?v=J01dt-QU4FQ
https://www.scribd.com/document/426592153/115225268-APPLICATIONS-OF-8086