SECURITY LOCK

Project submitted to the
SRM University – AP, Andhra Pradesh
Submitted in partial fulfilment of the requirement for the award of the degree of
Bachelor of Technology
in
Computer Science and Engineering

School of Engineering and Sciences
Submitted By

P.DURGA SRAVANTHI(AP23110011597)

Y.LAKSHMI MAHESWARI(AP23110011590)

CH.HIMAKSHI(AP23110011596)

B.KAIVALYA(AP23110011311)

D.GOPIKA(AP23110011571)

Under the guidance of

Dr.Kakumani KC Deepthi

SRM University, AP
Neerukonda,
Mangalagiri, Guntur
Andhra Pradesh – 522 240

TABLE OF CONTENT

ABSTRACT

Security and authentication are essential components of modern banking systems to protect sensitive financial data and secure physical access to restricted areas. This project presents the design and implementation of a **Security Lock System using 8086 Assembly Language**, which simulates access control in a bank. Each employee is assigned a unique 16-bit Employee ID and a 4-bit password stored in memory. The system verifies login credentials, tracks incorrect attempts, locks the account after three failures, and allows users to change passwords after successful authentication. The project demonstrates fundamental concepts of Computer Organization, memory mapping, array-based searching, conditional execution, input handling, and interrupt-driven I/O using 8086 microprocessor.

INTRODUCTION

In banking environments, access to certain areas such as vaults, server rooms, and document archives is restricted to authorized personnel only. To control access, digital authentication mechanisms are used. In this project, we implement a **software-based simulation of a secure lock system** using the **8086 microprocessor**.

The system accepts two inputs:

- **Employee Identification Number (ID)** – 16-bit number
- **Password** – 4-bit number (range 0–15)

The system compares user inputs with a stored database of employees and either **grants or denies access** based on validity. It also implements a **lockout mechanism** after three incorrect password attempts and provides an option for employees to change their passwords upon successful login.

BACKGROUND

In real-world applications, banks use security systems like RFID locks, keypads, biometric scanners, and digital PIN access. While hardware-based systems are common, software-based simulation helps in understanding **how authentication logic, password matching, memory storage, and attempt tracking work at the processor level**.

The 8086 microprocessor is ideal for learning such concepts because:

- It supports addressing modes for arrays (employee database).
- It allows interrupt-driven I/O using INT 21h.
- It enables direct memory manipulation and conditional logic using Assembly language.

The challenge is to simulate an authentication system that:

- Stores multiple employee IDs and passwords in memory
- Accepts inputs and compares them with stored data
- Tracks multiple login attempts
- Locks the account after repeated failures
- Allows password update after successful access

PROBLEM STATEMENT

Design Security Lock project by using 8086 assembly language In the bank, a security lock is used to access some rooms. This lock accepts two inputs: the employee identification number (16 bits) and his/her password (4 bits). If the bank has 20 employees, construct their database and store it in the memory. Then write a program to access these rooms. The inputs of the program are the employee identification and the password. The output is one bit (0/1) that means (denied/allowed).

METHODOLOGY

1 **Database Design and Memory Allocation**

- Use DW to store 16-bit Employee IDs (EmpTable)
- Use DB to store associated passwords (PassTable)
- Use DB array Attempts to track incorrect login tries

2 **Input Handling**

- Use `SCAN_NUM` macro to read user input (ID and password)
- INT 21h interrupt used for printing messages (display prompts)

3 **Employee ID Search (FindEmp Procedure)**

- Linear search (array traversal) is used to match ID
- If found → store index, else show “ID not found”

4 **Password Verification**

- Compare entered password with stored password
- If incorrect → increment attempt count
- If attempts ≥ 3 → lock account

5 **Successful Login Handling**

- Show “Access Granted” message
- Reset attempt count to 0

6 **Password Change Feature**

- Ask if user wants to change password (Y/N)
- If yes → update PassTable at stored index

7 **Loop back for next user**

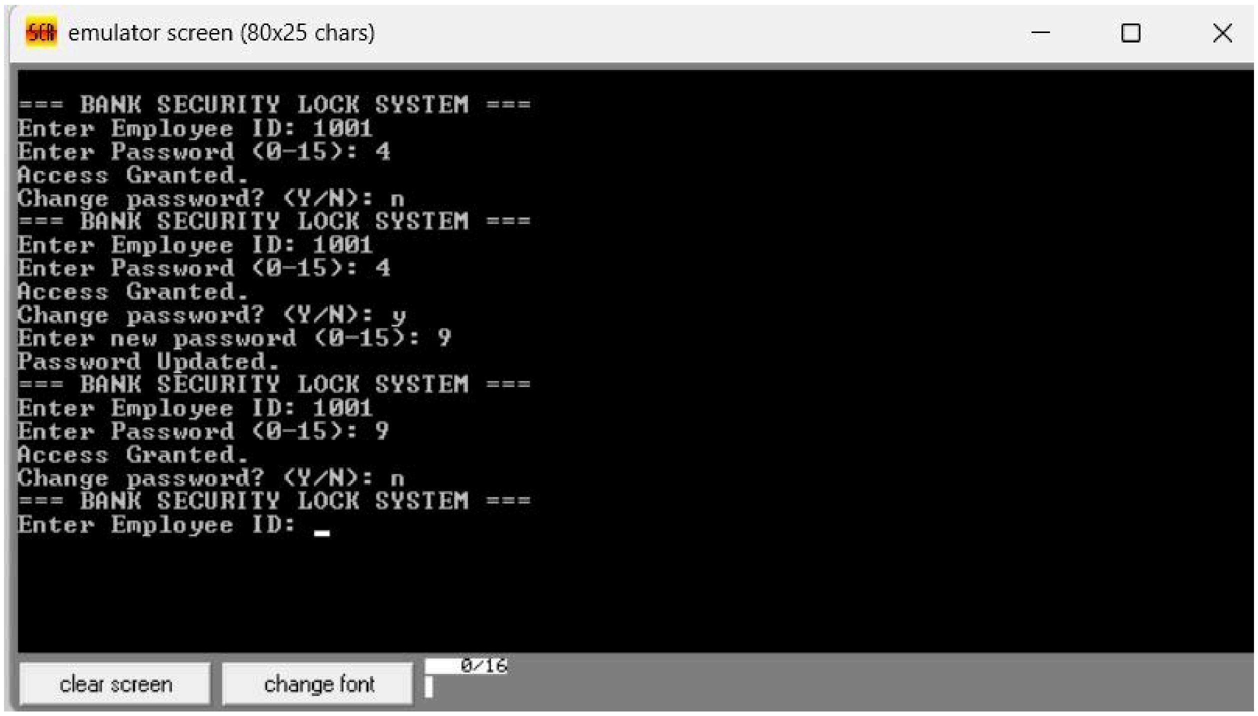
- System repeats for continuous authentication testing

IMPLEMENTATION

```
INCLUDE emu8086.inc
.MODEL SMALL
.STACK 100h
MAX_EMP EQU 20
MAX_ATTEMPTS EQU 3
.DATA
TitleMsg   DB 13,10,'=== BANK SECURITY LOCK SYSTEM ===$'
AskIDMsg   DB 13,10,'Enter Employee ID: $'
AskPassMsg DB 13,10,'Enter Password (0-15): $'
IDNotFound DB 13,10,'ID not found.$'
LockedMsg  DB 13,10,'Account locked!$'


WrongMsg   DB 13,10,'Wrong password.$'
OKMsg      DB 13,10,'Access Granted.$'

AskChange DB 13,10,'Change password? (Y/N): $'
AskNew    DB 13,10,'Enter new password (0-15): $'
Changed   DB 13,10,'Password Updated.$'

CurrentIndex DB ?

; IDs (20 employees)
EmpTable DW 1001,1002,1003,1004,1005,1010,1011,1012,1013,1014
     DW 2001,2002,2003,2004,2005,3001,3002,3003,3004,3005

; Passwords
PassTable DB 4,7,3,0,6,5,8,1,9,10,11,12,2,14,15,2,5,7,9,13

Attempts DB 20 DUP(0)

.CODE
MAIN PROC
  mov ax, @data
  mov ds, ax

MAIN_LOOP:
   lea dx, TitleMsg
   mov ah, 09h
   int 21h

; -------- ID INPUT --------
GetID:
   lea dx, AskIDMsg
   mov ah, 09h
   int 21h

   call SCAN_NUM      ; CX = entered ID
   call FindEmp
   jz Found
   lea dx, IDNotFound
   mov ah, 09h
   int 21h
   jmp GetID

Found:
   mov CurrentIndex, bl

   mov bl, CurrentIndex
   mov al, Attempts[bx]
   cmp al, MAX_ATTEMPTS
   jb AskPass
   lea dx, LockedMsg
   mov ah, 09h
   int 21h
   jmp MAIN_LOOP

; -------- PASSWORD INPUT --------
AskPass:
   lea dx, AskPassMsg
   mov ah, 09h
   int 21h

   call SCAN_NUM
   mov al, cl

   mov bl, CurrentIndex
   cmp al, PassTable[bx]
   je Correct
; wrong password
   mov bl, CurrentIndex
   inc Attempts[bx]

   mov al, Attempts[bx]
   cmp al, MAX_ATTEMPTS
   jae LockNow

   lea dx, WrongMsg
   mov ah, 09h
   int 21h

   jmp MAIN_LOOP

LockNow:
   lea dx, LockedMsg
   mov ah, 09h
   int 21h
   jmp MAIN_LOOP

; -------- CORRECT PASSWORD --------
Correct:
   mov bl, CurrentIndex
   mov Attempts[bx], 0

   lea dx, OKMsg
   mov ah, 09h
   int 21h

   ; ------------ TIME REMOVED ------------
   ; (No ShowTime call)
   ; --------------------------------------

   ; change password?
   lea dx, AskChange
   mov ah, 09h
   int 21h

   mov ah, 01h
   int 21h
```

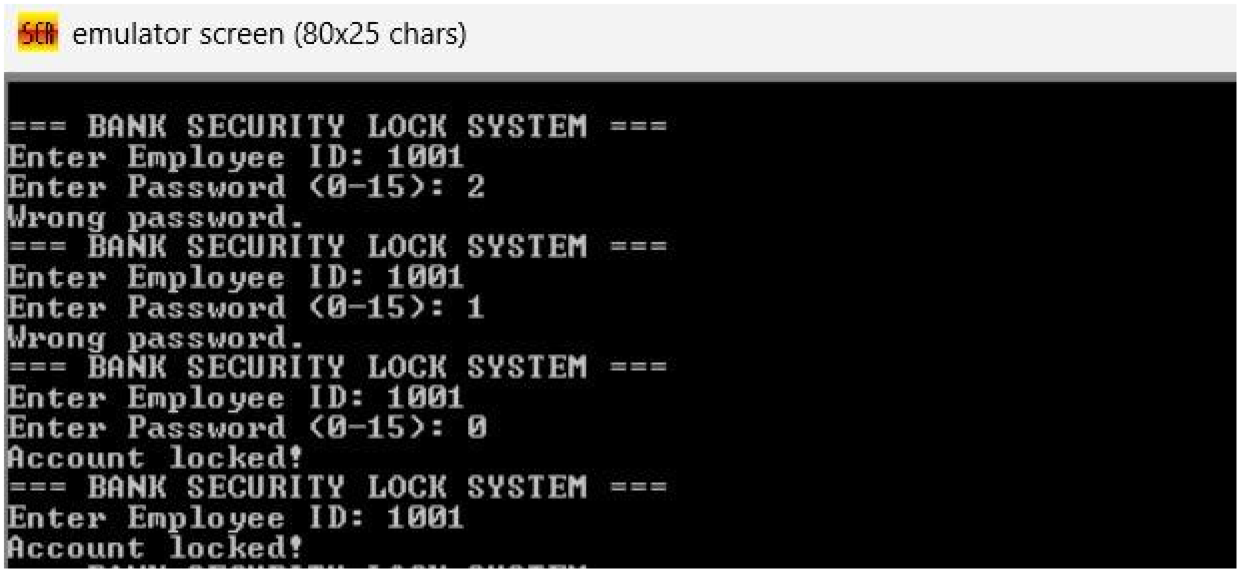
```
    cmp al, 'Y'
    je ChangePass
    cmp al, 'y'
    je ChangePass
    jmp MAIN_LOOP

; -------- CHANGE PASSWORD --------
ChangePass:
    lea dx, AskNew
    mov ah, 09h
    int 21h

    call SCAN_NUM
    mov al, cl
    mov bl, CurrentIndex
    mov PassTable[bx], al

    lea dx, Changed
    mov ah, 09h
    int 21h

    jmp MAIN_LOOP

MAIN ENDP

; ---------- FIND EMPLOYEE ----------
FindEmp PROC
    xor bx, bx
FLoop:
    cmp bx, MAX_EMP
    jge NotFound
    mov si, bx
    shl si, 1
    mov ax, EmpTable[si]
    cmp ax, cx
    je FoundEmp
    inc bx
    jmp FLoop

FoundEmp:
   cmp bx, bx
   ret

NotFound:
   mov bx,1
   cmp bx,0
   ret
FindEmp ENDP
DEFINE_SCAN_NUM
END MAIN
```

RESULT



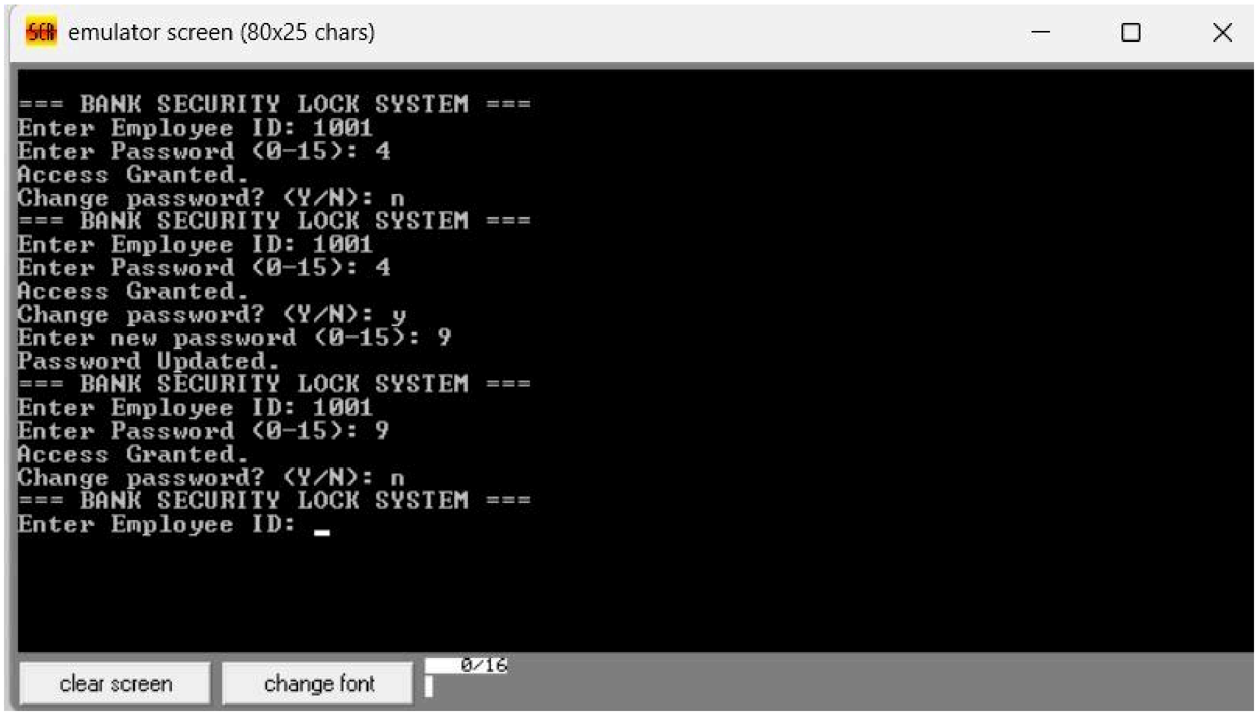
```
=== BANK SECURITY LOCK SYSTEM ===
Enter Employee ID: 1001
Enter Password (0-15): 4
Access Granted.
Change password? (Y/N): n
=== BANK SECURITY LOCK SYSTEM ===
Enter Employee ID: 1001
Enter Password (0-15): 4
Access Granted.
Change password? (Y/N): y
Enter new password (0-15): 9
Password Updated.
=== BANK SECURITY LOCK SYSTEM ===
Enter Employee ID: 1001
Enter Password (0-15): 9
Access Granted.
Change password? (Y/N): n
=== BANK SECURITY LOCK SYSTEM ===
Enter Employee ID: _
```

LIMITED TRY:



```
=== BANK SECURITY LOCK SYSTEM ===
Enter Employee ID: 1001
Enter Password (0-15): 2
Wrong password.
=== BANK SECURITY LOCK SYSTEM ===
Enter Employee ID: 1001
Enter Password (0-15): 1
Wrong password.
=== BANK SECURITY LOCK SYSTEM ===
Enter Employee ID: 1001
Enter Password (0-15): 0
Account locked!
=== BANK SECURITY LOCK SYSTEM ===
Enter Employee ID: 1001
Account locked!
```

PASSWORD CHANGE:



```
=== BANK SECURITY LOCK SYSTEM ===
Enter Employee ID: 1001
Enter Password (0-15): 4
Access Granted.
Change password? (Y/N): n
=== BANK SECURITY LOCK SYSTEM ===
Enter Employee ID: 1001
Enter Password (0-15): 4
Access Granted.
Change password? (Y/N): y
Enter new password (0-15): 9
Password Updated.
=== BANK SECURITY LOCK SYSTEM ===
Enter Employee ID: 1001
Enter Password (0-15): 9
Access Granted.
Change password? (Y/N): n
=== BANK SECURITY LOCK SYSTEM ===
Enter Employee ID: _
```

CONCLUSION

The Security Lock System using 8086 Assembly Language successfully demonstrates the core concepts of authentication, memory-based verification, and access control at the microprocessor level. By storing employee IDs, passwords, and login attempts in memory, the system effectively validates user credentials, restricts unauthorized access, and locks accounts after multiple failed attempts, just like real-world security mechanisms used in banks. The inclusion of a password update feature after successful authentication adds flexibility and enhances security. This project not only fulfills the objective of designing a secure access system but also strengthens understanding of array handling, conditional logic, interrupt-based input/output, and procedure implementation in 8086 assembly. Overall, it provides a strong foundation for implementing advanced embedded security systems in future applications.

REFERENCES

https://github.com/yousefkotp/8086-Assembly-Projects