22344 - Microprocessors

King Abdullah II School of Engineering

Department of Computer Engineering

Customizable Quiz Game in 8086 Microprocessor

Author:
Qabas Ahmad (ID: 20210786)
Supervisor:
Dr. Heba AbdelNabi

Date: May 12, 2025

Abstract

This project demonstrates the design and implementation of a quiz game with 8086 assembly language customization based on an MCQ approach. The game focuses on an interactive user interface, low-level memory handling, and a modular approach. Some of the key features are case-insensitive input handling, customized login for user authentication, scoring, random question picking, and two more features: ID-based countdown timer and session-persistent scoreboard. Each of these is implemented using system calls and low-level 8086 instructions.

1 Introduction

Programming in assembly language provides direct hardware control but is also demanding because of its low-level nature. This project uses both theoretical and practical material of the Microprocessors course to design an interactive 8086 assembly quiz game. It illustrates basic principles such as input/output using interrupts, memory handling, conditional branching, procedure calls, and optimizing timing.

1.1 Quiz Topic

The quiz is about computational complexity and the P vs. NP problem, a fascinating area of study in theoretical computer science with broad applications across numerous domains. Including the topic of computational complexity in the project helps make the project academically significant and impactful.

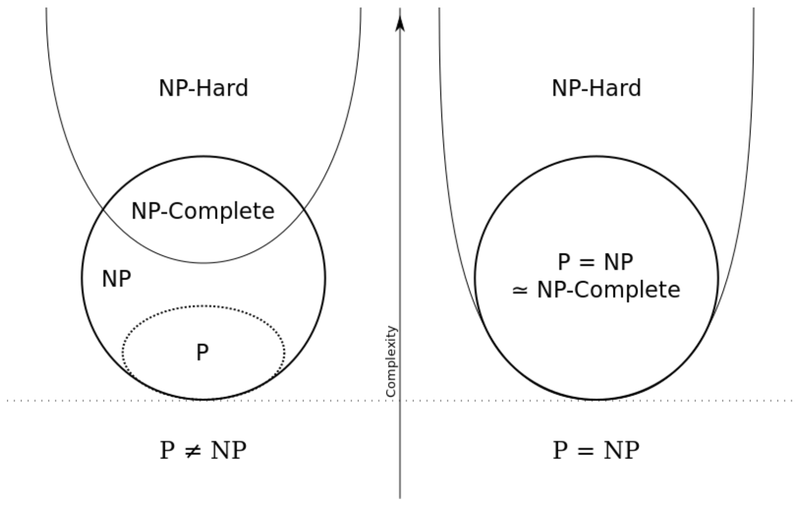


Figure 1: Computational Complexity

1.2 Project Objectives

- Implement user authentication system with 3 custom username-password pairs.
- Design an MCQ-based quiz system with 5 questions, storing questions in memory and shuffle them each game session.
- Accept user answers in a case-insensitive format.
- Analyze the performance of a selected procedure with clock cycle calculation.

2 Program Logic and Features

2.1 Data Segment

The data segment stores all static and dynamic information required by the program. It is logically organized into the following categories:

- Authentication Data: Contains usernames and passwords, along with input buffers to receive user credentials.

```
    UserX db "Name", 0
    PassX db "Pass00", 0
```

- Quiz Questions and Answers: Each question includes a prompt, multiple choice options, correct answers (both uppercase and lowercase), and buffers to hold user responses.

```
    Qx  db 'Qx: What does it mean ... ?$'
    QxA db 'A. ... $'
    QxB db 'B. ... $'
    QxC db 'C. ... $'
    QxD db 'D. ... $'
    QxSolC db 'Y', 0
    QxSolS db 'y', 0
    QxIn db 2Ah, 0, 2Ah dup(0)
    QxUserAnswer db 3 dup(0)
```

- Score Tracking: Keeps track of the current score and stores individual scores for each registered user.
- Leaderboard and Display Strings: Used to format and present the top scores after the quiz is completed.

```
    LBHeader   db 13,10,'=== Top Scores ===',13,10,'$'
    LB1        db '1st: $'
    LB2        db '2nd: $'
    LB3        db '3rd: $'
    LBScore    db '/5',13,10,'$'
    Display_User1 db 'Usr1   $'
    Display_User2 db 'Usr2   $'
    Display_User3 db 'Usr3   $'
```

- Timer Variables: Reserved for managing and displaying countdowns or timing behavior (planned extension).

2.2 Main Procedures

This section covers all the main procedures written to ensure modular programming all throughout. Helper procedures were also used to avoid redundancy in the main program and the procedures such as the Newline procedure, the Peep procedure as well as the Comparison procedure, among many others.

2.2.1 Read Credentials and Login Authentication

Three username-password pairs are hard-coded with the names of close relatives. The user has 3 attempts at logging in, after which the program terminates. Input is handled via INT 21h / 0Ah.

Listing 1: Read Credentials - Username

```
    lea dx, EnterUsername
    mov ah, 9
    int 21h
    lea dx, UsernameBuffer
    mov ah, 0Ah
    int 21h
    lea si, UsernameBuffer + 2
    lea di, UsernameInput
    call CopyInputTo
```

The CopyInputTo procedure is crucial in the correctness of the authentication process to remove the return carriage when reading from the user. To analyze, the input is compared with every single user; if it did not match any of the user, it prompts for another input which gets repeated three times.

2.2.2 Shuffle and SolveQuestionX

Dynamic shuffling is crucial to ensure that random questions are asked in every session. To guarantee randomness of behavior, a seed is defined using the BIOS system clock with interrupt INT 1ah. It reads the low byte of tick count and stores it within a seed variable.

Listing 2: Seed Generation

```
    mov ah, 00h
    int 1ah
    mov seed, dl
    cmp seed, 0
    jne SeedValid
    mov Seed, 1
SeedValid:
    call Shuffle
    jmp  StartQuiz
```

Since it may destabilize randomness in the case of zero seed, the code checks if it is zero and replaces it with 1. This seed is subsequently transferred to the Shuffle function to facilitate pseudorandomness.
Shuffling a list of indices, it is later used to access the questions in random order per session.

2.2.3 Calculate and Display Timer

Each question is expected to have a specific timer based on the last digit of the ID. For higher accuracy, BIOS based delay was used with the amount of delay in the appropriate registers. Each question will wait 12 seconds (6 * 2), if no input is provided before time-out, the wrong answer is assumed, and the score will not increase.
The procedure to display the timer is called in the solver for every question, right after displaying the question text and options for fairness sake.

2.2.4 Push Scores and Leader-board

To display all players with their scores at the end of the game, the push score function determines which user is playing and then stores its score to be later displayed.

Listing 3: Storing Scores

```
    UpdateUser:
    mov al, [Score]
    cmp [UserScore], al
    jb  SetUser
    ret
SetUser:
    mov [UserScore], al
    ret
```

The comparison ensures that each new score is only stored if it is the maximum, avoiding overwriting the maximum score for the user. Then, at the end of the game session in which all users played, the ShowLeaderboard function showcases the maximum value per user.

2.3 Flowchart

The flowchart illustrates the logical sequence of the quiz program, from user authentication to quiz completion and leaderboard display. Here is a breakdown of the key stages:

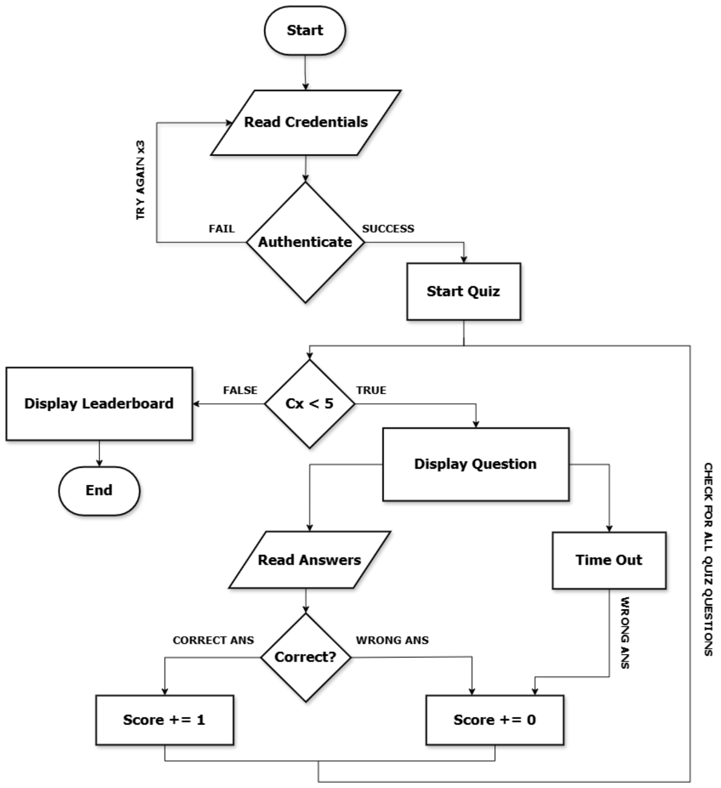
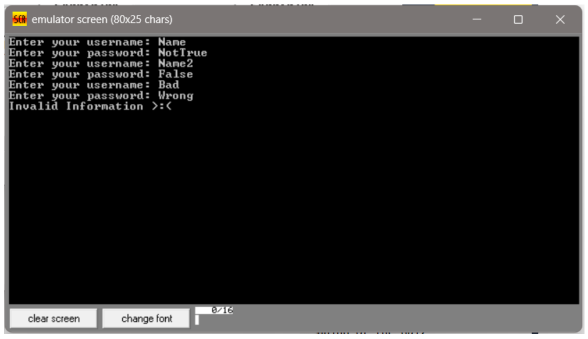


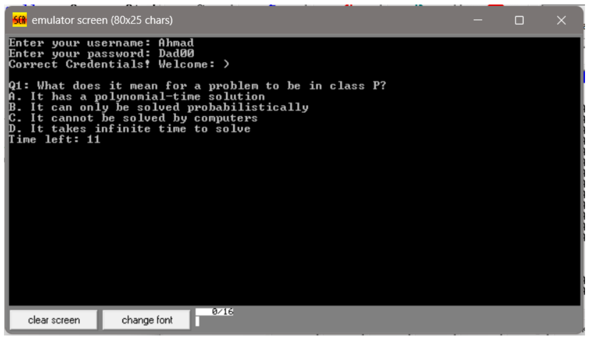
Figure 2: Program Flowchart

3 Screenshots



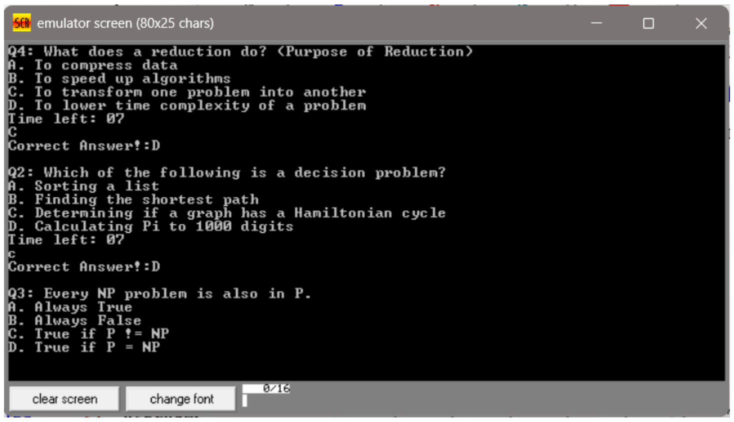
```
Enter your username: Name
Enter your password: NotTrue
Enter your username: Name2
Enter your password: False
Enter your username: Bad
Enter your password: Wrong
Invalid Information >:(
```

Figure 3: Incorrect Credentials



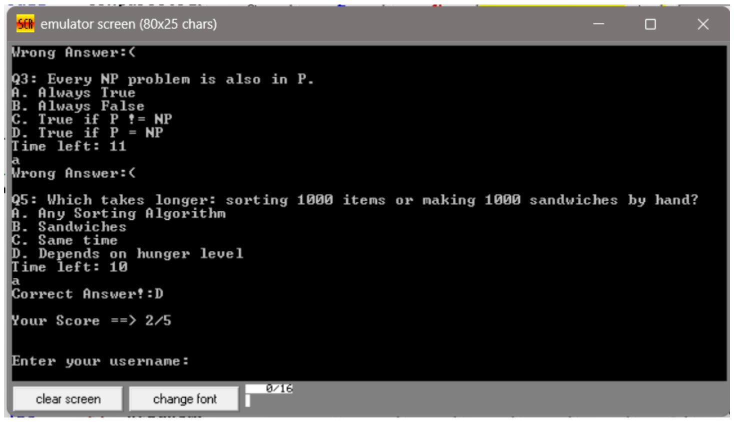
```
Enter your username: Ahmad
Enter your password: Dad00
Correct Credentials! Welcome: >

Q1: What does it mean for a problem to be in class P?
A. It has a polynomial-time solution
B. It can only be solved probabilistically
C. It cannot be solved by computers
D. It takes infinite time to solve
Time left: 11
```

Figure 4: Correct Credentials with Timer



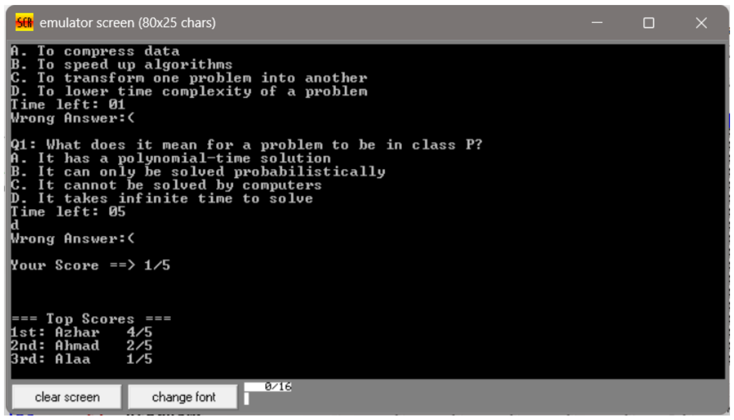
```
Q4: What does a reduction do? (Purpose of Reduction)
A. To compress data
B. To speed up algorithms
C. To transform one problem into another
D. To lower time complexity of a problem
Time left: 07
C
Correct Answer!:D

Q2: Which of the following is a decision problem?
A. Sorting a list
B. Finding the shortest path
C. Determining if a graph has a Hamiltonian cycle
D. Calculating Pi to 1000 digits
Time left: 07
c
Correct Answer!:D

Q3: Every NP problem is also in P.
A. Always True
B. Always False
C. True if P != NP
D. True if P = NP
```

Figure 5: Case-Insensitive Input and Shuffling



```
Wrong Answer:(

Q3: Every NP problem is also in P.
A. Always True
B. Always False
C. True if P != NP
D. True if P = NP
Time left: 11
a
Wrong Answer:(

Q5: Which takes longer: sorting 1000 items or making 1000 sandwiches by hand?
A. Any Sorting Algorithm
B. Sandwiches
C. Same time
D. Depends on hunger level
Time left: 10
a
Correct Answer!:D

Your Score ==> 2/5


Enter your username:
```

Figure 6: Score per Session



```
A. To compress data
B. To speed up algorithms
C. To transform one problem into another
D. To lower time complexity of a problem
Time left: 01
Wrong Answer:(

Q1: What does it mean for a problem to be in class P?
A. It has a polynomial-time solution
B. It can only be solved probabilistically
C. It cannot be solved by computers
D. It takes infinite time to solve
Time left: 05
d
Wrong Answer:(

Your Score ==> 1/5



=== Top Scores ===
1st: Azhar    4/5
2nd: Ahmad    2/5
3rd: Alaa     1/5
```

Figure 7: Leaderboard

4 Clock Cycle Analysis

The procedure used for the clock cycle analysis is LoginAuthentication due to its importance in the authentication process before the beginning of the quiz.
The following table showcases the instructions used in the procedure.
Calculating the average bytes for the jumps in the program to determine the type of jump.

```
    jnz   CheckUserx
    lea   si, PasswordInput   ; 4 bytes + 1 byte
    lea   di, Passx           ; 4 bytes + 1 byte
    call  CompareStrings      ; 3 bytes
    cmp   ax, 0               ; 4 bytes + 1 byte + 1 byte
    jz    Authen_Success      ; 2 bytes
    jmp   Authen_Done         ; 2 bytes
CheckUserx+1:
```

$$\text{Total} = 4 + 4 + 4 + 3 + 3 + 2 + 2 = 23 \text{ bytes} \tag{1}$$

Showcasing that the total number of bytes is well within -128 to 127. This results makes it safe to assume that for conditional and unconditional jumps of this procedure are short and can be represented with an 8-bit displacement.

Instruction	Clock Cycles
mov (reg - imm)	4 + 4
lea (direct)	2 + 6
call (worst case)	28
cmp (reg - imm)	4 + 4
jnz (worst case)	16
jz (worst case)	16
jmp (intrasegment direct)	15
ret (worst case)	18 + 8

So the total number of cycles for the procedure:

$$\text{CC's per User} = 2*(2+6)+28+(4+4)+16+2*(2+6)+28+(4+4)+16+15 = 151 \text{ cycles} \tag{2}$$

$$\text{CC's} = (4+4) + 3 * \text{CC's per User} + (4+4) + 18 + 8 = 495 \text{ cycles} \tag{3}$$

Assuming we have a frequency of

$$f = 5MHz \tag{4}$$

we get a total time of

$$495 * 200ns = 99\mu s \tag{5}$$

5 Conclusion

This project successfully integrates multiple low-level programming concepts into an interactive application. From user input via BIOS interrupts to timing analysis and memory-managed storage, the game serves as a strong demonstration of the power and complexity of 8086 assembly programming. Bonus features like a personalized timer and leaderboard further highlight advanced skills in delay loop design and memory structuring.