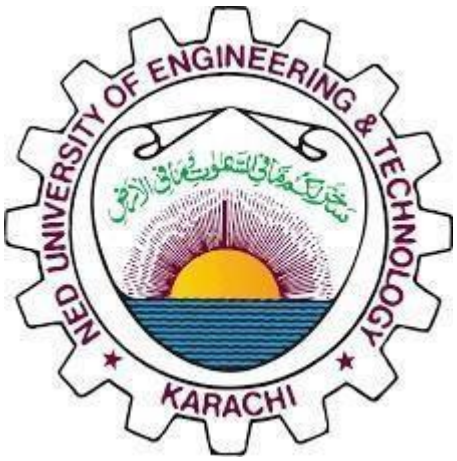
Exam System (Console Based)

Computer Architecture and Organization– CT-252



Bushra Ansar	AI-003
Tamia Naeem	AI-004
Hareem Nadeem	AI-025
Bushra Atiq	AI-028

Department of Computer Science and Information Technology NED University of Engineering and Technology, Pakistan

Contents

Certificate of Completion

This is to certify that,

Miss Bushra Ansar, Miss Tamia Naeem, Miss Hareem and Miss Bushra Atiq has successfully designed and developed project of CAO (4th semester) NED University Main Campus, Karachi.

Submitted To: Sir Wajhi Uddin, Sir Uzair.

Date of issue: 12-05-2025

Authorized Signature: _____

Introduction

The Smart Exam System is a command-line-based multiple-choice quiz program developed in MIPS Assembly Language and executed using the MARS (MIPS Assembler and Runtime Simulator). It mimics a basic exam environment where users must log in with credentials and then select a subject to answer 10 quiz questions. This project demonstrates the use of assembly programming in real-world scenarios, such as exam systems and user authentication.

Purpose

The main purpose of this project is to:

- Strengthen low-level programming skills using MIPS assembly language.
- Simulate a real-world system (examination platform) using assembly code.
- Understand how input/output, control structures, loops, and memory are managed at a low level.
- Implement a working quiz system that includes user login, subject selection, and quiz evaluation.

Project Scope

The project focuses on simulating the following core functionalities:

- A user authentication system with hardcoded credentials.
- A menu-driven subject selection system.
- A quiz for each subject with pre-defined questions and correct answers.
- Real-time score evaluation and result display after the quiz.
- Modular design allowing for future expansion (e.g., add more subjects or randomize questions).

Product Features

- User Login: Requires username and password before accessing quizzes.
- Subject Selection Menu: Includes 5 subjects - Programming Languages, Advanced Mathematics, Computer Science Basics, Software Engineering, and Data Structures.
- Quiz Functionality: Displays 10 questions per subject with 4 options (A/B/C/D).
- Answer Validation: User inputs are checked against stored correct answers.
- Score Display: Number of correct and wrong answers and final score shown at the end.

System Features

- Interactive Text-based Interface
- Preloaded Questions for Each Subject
- Answer Evaluation in Real-time
- Stateless Design (No persistent user data)

- Sequential Flow (Login → Menu → Quiz → Results)

Hardware Requirements

- Any system capable of running Java applications.
- Minimum 1 GHz processor.
- 1 GB RAM or higher.
- At least 50 MB of free disk space.

Software Requirements

- MARS (MIPS Assembler and Runtime Simulator) – For writing and running MIPS assembly code.
- Java Runtime Environment (JRE) – Required to run the MARS simulator.
- Any text editor or IDE (e.g., Notepad++, VSCode) for editing .asm files.

Language Used

- MIPS Assembly Language

Code

```
.data

newline:    .asciiz "\n"

welcome:    .asciiz "=== Welcome to Smart Exam System ===\n"

login_msg:  .asciiz "\nLogin Required\nUsername: "

user_ok:    .asciiz "Welcome, user123!\n"

wrong_user: .asciiz "Incorrect login. Try again.\n"

pass_msg:   .asciiz "Password: "

final_score:.asciiz "\nYour Final Score: "

correct_ans:.asciiz "Correct answers: "

wrong_ans:  .asciiz "Wrong answers: "

subject_prompt: .asciiz "\nSelect a Subject:\n1. Programming Languages\n2. Advanced Mathematics\n3. Computer Science Basics\n4. Software Engineering\n5. Data Structures\n0. Exit\n"

input_prompt: .asciiz "\nEnter your answer (A/B/C/D): "

correct_msg:  .asciiz "Correct!\n"

wrong_msg:    .asciiz "Wrong!\n"
```

```
user_input: .space 3

input_username:   .space 10

input_password:   .space 10

selected_subjects: .space 3


# --- Programming ---

q1: .asciiz "\nQ1: Who created Python?\nA. Dennis Ritchie\nB. Guido van Rossum\nC. James Gosling\nD. Bjarne Stroustrup\n"

q2: .asciiz "\nQ2: Python is...\nA. Interpreted\nB. Compiled\nC. Both\nD. None\n"

q3: .asciiz "\nQ3: Extension of Python files?\nA. .java\nB. .py\nC. .cpp\nD. .txt\n"

q4: .asciiz "\nQ4: Creator of Java?\nA. Guido\nB. Bjarne\nC. Gosling\nD. Dennis\n"

q5: .asciiz "\nQ5: Which language runs in a browser?\nA. C\nB. Java\nC. Python\nD. JavaScript\n"

q6: .asciiz "\nQ6: C++ is...\nA. Functional\nB. OOP\nC. Markup\nD. None\n"

q7: .asciiz "\nQ7: HTML used for?\nA. Logic\nB. Styling\nC. Structure\nD. Database\n"

q8: .asciiz "\nQ8: CSS used for?\nA. Logic\nB. Styling\nC. Structure\nD. Database\n"

q9: .asciiz "\nQ9: JS stands for?\nA. Java Super\nB. JavaScript\nC. Just Start\nD. None\n"

q10:.asciiz "\nQ10: Best for AI?\nA. C\nB. HTML\nC. Python\nD. Java\n"

prog_questions: .word q1,q2,q3,q4,q5,q6,q7,q8,q9,q10

prog_answers: .asciiz "BBCDDCBBCB"


# --- Math ---

mq1: .asciiz "\nQ1: Derivative of x^2?\nA. 2x\nB. x\nC. x^2\nD. 1\n"

mq2: .asciiz "\nQ2: Value of π?\nA. 2.14\nB. 3.14\nC. 4.14\nD. 1.41\n"

mq3: .asciiz "\nQ3: √16 = ?\nA. 2\nB. 4\nC. 8\nD. 6\n"

mq4: .asciiz "\nQ4: 5^2 = ?\nA. 10\nB. 20\nC. 25\nD. 15\n"

mq5: .asciiz "\nQ5: sin(90°) = ?\nA. 0\nB. 0.5\nC. 1\nD. Undefined\n"

mq6: .asciiz "\nQ6: log(1) = ?\nA. 1\nB. 0\nC. -1\nD. Infinity\n"

mq7: .asciiz "\nQ7: ∫x dx = ?\nA. x\nB. x^2/2 + C\nC. x^2 + C\nD. x + C\n"

mq8: .asciiz "\nQ8: x^0 = ?\nA. x\nB. 0\nC. 1\nD. Undefined\n"
```

```
mq9: .asciiz "\nQ9: Prime Number?\nA. 4\nB. 6\nC. 7\nD. 9\n"

mq10:.asciiz "\nQ10: Square root of 81?\nA. 8\nB. 9\nC. 7\nD. 6\n"

math_questions: .word mq1,mq2,mq3,mq4,mq5,mq6,mq7,mq8,mq9,mq10

math_answers: .asciiz "BABCCBBCCB"


# --- CS Basics ---

csq1: .asciiz "\nQ1: Full form of CPU?\nA. Central Process Unit\nB. Central Processing Unit\nC. Control Process Unit\nD. None\n"

csq2: .asciiz "\nQ2: RAM stands for?\nA. Read Access Memory\nB. Random Access Memory\nC. Run Access Memory\nD. None\n"

csq3: .asciiz "\nQ3: 1 byte = ? bits\nA. 8\nB. 16\nC. 32\nD. 4\n"

csq4: .asciiz "\nQ4: Brain of Computer?\nA. Monitor\nB. RAM\nC. CPU\nD. Mouse\n"

csq5: .asciiz "\nQ5: Which is input device?\nA. Printer\nB. Monitor\nC. Mouse\nD. Speaker\n"

csq6: .asciiz "\nQ6: GUI stands for?\nA. Graphical User Interface\nB. General User Input\nC. Graphics User Interaction\nD. None\n"

csq7: .asciiz "\nQ7: ASCII is for?\nA. Numbers only\nB. Images\nC. Text Encoding\nD. Sound\n"

csq8: .asciiz "\nQ8: 1 Kilobyte = ? bytes\nA. 1000\nB. 1024\nC. 512\nD. 2048\n"

csq9: .asciiz "\nQ9: .exe is?\nA. Image file\nB. Executable\nC. Audio\nD. Video\n"

csq10:.asciiz "\nQ10: HTML is?\nA. Language\nB. Protocol\nC. Script\nD. Software\n"

cs_questions: .word csq1,csq2,csq3,csq4,csq5,csq6,csq7,csq8,csq9,csq10

cs_answers: .asciiz "BABCCACBBB"


# --- Software Engineering ---

seq1: .asciiz "\nQ1: SDLC stands for?\nA. System Development Life Cycle\nB. Software Direct Life Cycle\nC. Software Data Life Cycle\nD. None\n"

seq2: .asciiz "\nQ2: Agile is?\nA. Waterfall\nB. Methodology\nC. Code\nD. Testing\n"

seq3: .asciiz "\nQ3: UML is?\nA. Unified Modeling Language\nB. User Mode Line\nC. Unstructured Map\nD. None\n"

seq4: .asciiz "\nQ4: Bug is?\nA. Feature\nB. Virus\nC. Error\nD. Backup\n"

seq5: .asciiz "\nQ5: SRS means?\nA. Software Requirement Specification\nB. System Result Source\nC. Software Resource\nD. None\n"

seq6: .asciiz "\nQ6: V&V is?\nA. Verify & Validate\nB. View & Vote\nC. Virtual Value\nD. None\n"
```

```
seq7: .asciiz "\nQ7: Design Phase is after?\nA. Testing\nB. Coding\nC. Planning\nD. Requirement\n"
seq8: .asciiz "\nQ8: Unit Testing is?\nA. Whole software\nB. Module testing\nC. Alpha test\nD. Beta test\n"
seq9: .asciiz "\nQ9: Spiral Model is?\nA. Iterative\nB. Sequential\nC. Static\nD. None\n"
seq10:.asciiz "\nQ10: Feasibility study checks?\nA. Market\nB. Time\nC. Cost\nD. All\n"
se_questions: .word seq1,seq2,seq3,seq4,seq5,seq6,seq7,seq8,seq9,seq10
se_answers: .asciiz "ABCACADBAD"

# --- Data Structures ---
dsq1: .asciiz "\nQ1: Stack uses?\nA. FIFO\nB. LIFO\nC. FILO\nD. None\n"
dsq2: .asciiz "\nQ2: Queue uses?\nA. LIFO\nB. Random\nC. FIFO\nD. None\n"
dsq3: .asciiz "\nQ3: Linked List stores?\nA. Only data\nB. Only address\nC. Data and address\nD. None\n"
dsq4: .asciiz "\nQ4: BST stands for?\nA. Binary Search Tree\nB. Binary Structure Table\nC. Basic Search Tree\nD. None\n"
dsq5: .asciiz "\nQ5: Array index starts from?\nA. 1\nB. 0\nC. -1\nD. 2\n"
dsq6: .asciiz "\nQ6: Queue insert called?\nA. Push\nB. Append\nC. Enqueue\nD. Add\n"
dsq7: .asciiz "\nQ7: Heap used in?\nA. Sorting\nB. Merging\nC. Searching\nD. Traversing\n"
dsq8: .asciiz "\nQ8: DFS stands for?\nA. Data File Search\nB. Depth First Search\nC. Disk File System\nD. None\n"
dsq9: .asciiz "\nQ9: Hash Table uses?\nA. Stack\nB. Indexing\nC. Hash function\nD. Sorting\n"
dsq10:.asciiz "\nQ10: Queue removal called?\nA. Pop\nB. Remove\nC. Dequeue\nD. Shift\n"
ds_questions: .word dsq1,dsq2,dsq3,dsq4,dsq5,dsq6,dsq7,dsq8,dsq9,dsq10
ds_answers: .asciiz "BCABCCBCCD"

.text
.globl main

main:
   li $t4, 0  # correct
   li $t5, 0  # wrong

   li $v0, 4
   la $a0, welcome
```

```
    syscall

login:
    li $v0, 4
    la $a0, login_msg
    syscall

    li $v0, 8
    la $a0, input_username
    li $a1, 10
    syscall

    li $v0, 4
    la $a0, pass_msg
    syscall

    li $v0, 8
    la $a0, input_password
    li $a1, 10
    syscall

    la $t0, input_username
    lb $t1, 0($t0)
    li $t2, 'u'
    bne $t1, $t2, login_failed
    lb $t1, 1($t0)
    li $t2, 's'
    bne $t1, $t2, login_failed
    lb $t1, 2($t0)
    li $t2, 'e'
```

```
bne $t1, $t2, login_failed
lb $t1, 3($t0)
li $t2, 'r'
bne $t1, $t2, login_failed
lb $t1, 4($t0)
li $t2, '1'
bne $t1, $t2, login_failed
lb $t1, 5($t0)
li $t2, '2'
bne $t1, $t2, login_failed
lb $t1, 6($t0)
li $t2, '3'
bne $t1, $t2, login_failed

la $t0, input_password
lb $t1, 0($t0)
li $t2, '1'
bne $t1, $t2, login_failed
lb $t1, 1($t0)
li $t2, '2'
bne $t1, $t2, login_failed
lb $t1, 2($t0)
li $t2, '3'
bne $t1, $t2, login_failed

li $v0, 4
la $a0, user_ok
syscall

j subject_loop
```

```
login_failed:
    li $v0, 4
    la $a0, wrong_user
    syscall
    j login

subject_loop:
    li $v0, 4
    la $a0, subject_prompt
    syscall

    li $v0, 8
    la $a0, selected_subjects
    li $a1, 3
    syscall

    lb $t0, selected_subjects
    li $t1, '1'
    beq $t0, $t1, call_prog
    li $t1, '2'
    beq $t0, $t1, call_math
    li $t1, '3'
    beq $t0, $t1, call_cs
    li $t1, '4'
    beq $t0, $t1, call_se
    li $t1, '5'
    beq $t0, $t1, call_ds
    li $t1, '0'
    beq $t0, $t1, show_result
```

```
    j subject_loop

call_prog:
    jal programming_exam
    j subject_loop
call_math:
    jal math_exam
    j subject_loop
call_cs:
    jal cs_exam
    j subject_loop
call_se:
    jal se_exam
    j subject_loop
call_ds:
    jal ds_exam
    j subject_loop

programming_exam:
    li $t3, 0
prog_loop:
    li $t6, 10
    beq $t3, $t6, prog_done

    la $t0, prog_questions
    sll $t1, $t3, 2
    add $t1, $t0, $t1
    lw $a0, 0($t1)
    li $v0, 4
```

```
    syscall

    li $v0, 4
    la $a0, input_prompt
    syscall

    li $v0, 8
    la $a0, user_input
    li $a1, 3
    syscall

    lb $t1, user_input
    la $t2, prog_answers
    add $t2, $t2, $t3
    lb $t6, 0($t2)

    beq $t1, $t6, prog_correct
prog_wrong:
    li $v0, 4
    la $a0, wrong_msg
    syscall
    addi $t5, $t5, 1
    j prog_next
prog_correct:
    li $v0, 4
    la $a0, correct_msg
    syscall
    addi $t4, $t4, 1
prog_next:
    addi $t3, $t3, 1
```

```
    j prog_loop
prog_done:
    jr $ra

math_exam:
    li $t3, 0               # Question index = 0
math_loop:
    li $t6, 10              # Total 10 questions
    beq $t3, $t6, math_done     # Exit loop when 10 questions are done

    la $t0, math_questions      # Load base address of questions
    sll $t1, $t3, 2             # Multiply index by 4 (word size)
    add $t1, $t0, $t1           # Calculate question address
    lw $a0, 0($t1)              # Load question string
    li $v0, 4
    syscall                 # Print question

    li $v0, 4
    la $a0, input_prompt
    syscall                 # Prompt for input

    li $v0, 8
    la $a0, user_input
    li $a1, 3
    syscall                 # Take input (2 chars max + null terminator)

    lb $t1, user_input          # Load first char of user input
    la $t2, math_answers        # Load base address of answers
    add $t2, $t2, $t3           # Move to correct answer index
    lb $t6, 0($t2)              # Load correct answer
```

```
    beq $t1, $t6, math_correct  # Compare input with answer
math_wrong:
    li $v0, 4
    la $a0, wrong_msg
    syscall                 # Print wrong answer message
    addi $t5, $t5, 1          # wrong_count++
    j math_next
math_correct:
    li $v0, 4
    la $a0, correct_msg
    syscall                 # Print correct answer message
    addi $t4, $t4, 1          # correct_count++
math_next:
    addi $t3, $t3, 1          # question_index++
    j math_loop
math_done:
    jr $ra                  # Return to caller

cs_exam:
    li $t3, 0               # Question index = 0
cs_loop:
    li $t6, 10               # Total 10 questions
    beq $t3, $t6, cs_done      # Exit loop when 10 questions are done

    la $t0, cs_questions       # Load base address of questions
    sll $t1, $t3, 2           # Multiply index by 4 (word size)
    add $t1, $t0, $t1          # Calculate question address
    lw $a0, 0($t1)            # Load question string
    li $v0, 4
```

```
    syscall                 # Print question

    li $v0, 4
    la $a0, input_prompt
    syscall                 # Prompt for input

    li $v0, 8
    la $a0, user_input
    li $a1, 3
    syscall                 # Take input (2 chars max + null terminator)

    lb $t1, user_input          # Load first char of user input
    la $t2, cs_answers          # Load base address of answers
    add $t2, $t2, $t3           # Move to correct answer index
    lb $t6, 0($t2)              # Load correct answer

    beq $t1, $t6, cs_correct    # Compare input with answer
cs_wrong:
    li $v0, 4
    la $a0, wrong_msg
    syscall                 # Print wrong answer message
    addi $t5, $t5, 1            # wrong_count++
    j cs_next
cs_correct:
    li $v0, 4
    la $a0, correct_msg
    syscall                 # Print correct answer message
    addi $t4, $t4, 1            # correct_count++
cs_next:
    addi $t3, $t3, 1            # question_index++
```

```
    j cs_loop
cs_done:
    jr $ra                  # Return to caller

se_exam:
    li $t3, 0               # Question index = 0
se_loop:
    li $t6, 10              # Total 10 questions
    beq $t3, $t6, se_done   # Exit loop when 10 questions are done

    la $t0, se_questions    # Load base address of questions
    sll $t1, $t3, 2         # Multiply index by 4 (word size)
    add $t1, $t0, $t1       # Calculate question address
    lw $a0, 0($t1)          # Load question string
    li $v0, 4
    syscall                 # Print question

    li $v0, 4
    la $a0, input_prompt
    syscall                 # Prompt for input

    li $v0, 8
    la $a0, user_input
    li $a1, 3
    syscall                 # Take input (2 chars max + null terminator)

    lb $t1, user_input      # Load first char of user input
    la $t2, se_answers      # Load base address of answers
    add $t2, $t2, $t3       # Move to correct answer index
    lb $t6, 0($t2)          # Load correct answer
```

```
    beq $t1, $t6, se_correct    # Compare input with answer
se_wrong:
    li $v0, 4
    la $a0, wrong_msg
    syscall                 # Print wrong answer message
    addi $t5, $t5, 1          # wrong_count++
    j se_next
se_correct:
    li $v0, 4
    la $a0, correct_msg
    syscall                 # Print correct answer message
    addi $t4, $t4, 1          # correct_count++
se_next:
    addi $t3, $t3, 1          # question_index++
    j se_loop
se_done:
    jr $ra                  # Return to caller

ds_exam:
    li $t3, 0               # Question index = 0
ds_loop:
    li $t6, 10               # Total 10 questions
    beq $t3, $t6, ds_done      # Exit loop when 10 questions are done

    la $t0, ds_questions       # Load base address of questions
    sll $t1, $t3, 2           # Multiply index by 4 (word size)
    add $t1, $t0, $t1          # Calculate question address
    lw $a0, 0($t1)            # Load question string
    li $v0, 4
```

```
    syscall                 # Print question

    li $v0, 4
    la $a0, input_prompt
    syscall                 # Prompt for input

    li $v0, 8
    la $a0, user_input
    li $a1, 3
    syscall                 # Take input (2 chars max + null terminator)

    lb $t1, user_input        # Load first char of user input
    la $t2, ds_answers        # Load base address of answers
    add $t2, $t2, $t3         # Move to correct answer index
    lb $t6, 0($t2)            # Load correct answer

    beq $t1, $t6, ds_correct    # Compare input with answer
ds_wrong:
    li $v0, 4
    la $a0, wrong_msg
    syscall                 # Print wrong answer message
    addi $t5, $t5, 1          # wrong_count++
    j ds_next
ds_correct:
    li $v0, 4
    la $a0, correct_msg
    syscall                 # Print correct answer message
    addi $t4, $t4, 1          # correct_count++
ds_next:
    addi $t3, $t3, 1          # question_index++
```

```
    j ds_loop
ds_done:
    jr $ra                  # Return to caller

show_result:
    li $v0, 4
    la $a0, final_score
    syscall

    li $v0, 1
    li $t7, 10
    mul $a0, $t4, $t7
    syscall

    li $v0, 4
    la $a0, newline
    syscall

    li $v0, 4
    la $a0, correct_ans
    syscall

    li $v0, 1
    move $a0, $t4
    syscall

    li $v0, 4
    la $a0, newline
    syscall
```

```
li $v0, 4
la $a0, wrong_ans
syscall
li $v0, 1
move $a0, $t5
syscall
li $v0, 10
syscall
```

Output

WELCOME PAGE

```
=== Welcome to Smart Exam System ===

Login Required
Username: user123
Password: 123
Welcome, user123!
```

SUBJECT MENU

```
Select a Subject:
1. Programming Languages
2. Advanced Mathematics
3. Computer Science Basics
4. Software Engineering
5. Data Structures
0. Exit
```

10 QUESTIONS OF PROGRAMMING LANGUAGES

```
Q1: Who created Python?
A. Dennis Ritchie
B. Guido van Rossum
C. James Gosling
D. Bjarne Stroustrup

Enter your answer (A/B/C/D): B
Correct!
```

```
Q2: Python is...
A. Interpreted
B. Compiled
C. Both
D. None

Enter your answer (A/B/C/D): C
Correct!
```

```
Q3: Extension of Python files?
A. .java
B. .py
C. .cpp
D. .txt

Enter your answer (A/B/C/D): B
Correct!
```

```
Q4: Creator of Java?
A. Guido
B. Bjarne
C. Gosling
D. Dennis

Enter your answer (A/B/C/D): B
Wrong!
```

```
Q5: Which language runs in a browser?
A. C
B. Java
C. Python
D. JavaScript

Enter your answer (A/B/C/D): D
Correct!
```

```
Q6: C++ is...
A. Functional
B. OOP
C. Markup
D. None

Enter your answer (A/B/C/D): B
Correct!
```

```
Q7: HTML used for?
A. Logic
B. Styling
C. Structure
D. Database

Enter your answer (A/B/C/D): C
Correct!
```

```
Q8: CSS used for?
A. Logic
B. Styling
C. Structure
D. Database

Enter your answer (A/B/C/D): B
Correct!
```

```
Q9: JS stands for?
A. Java Super
B. JavaScript
C. Just Start
D. None

Enter your answer (A/B/C/D): B
Correct!
```

```
Q10: Best for AI?
A. C
B. HTML
C. Python
D. Java

Enter your answer (A/B/C/D): C
Correct!
```

FINAL SCORE

```
Your Final Score: 90
Correct answers: 9
Wrong answers: 1
-- program is finished running --
```