SEMESTER 1/1

BITS1123

COMPUTER ARCHITECTURE AND ORGANIZATION

BITC

GROUP PROJECT:
TEXT -BASED ADVENTURE GAME

Name	Matric No.
MUHAMMAD SYAKIR BIN ABD RAZAK	B032410068
HAZRILL IZKHWAN BIN DOLLAH	B032410167
MUHAMMAD HAIKAL BIN MOHD SHAHRIM	B032410166

PENSYARAH :

DR. ZURINA BINTI SAAYA

Introduction to MIPS Assembly Language Programming

MIPS, or Microprocessor Interlocked Pipeline Stage, is a reduced instruction set computer(RISC) instructon set architecture invented by MIPS Technologies.

Because so many embedded systems use the MIPS processor, learning the MIPS assembly language is tremendously beneficial. It is possible to gain a deeper grasp of how these systems function on a more fundamental level by learning to code in this language.

In MIPS, every instruction is 32 bits long. In the MIPS architecture, a byte correesponds to 8 bits, a halfword to 2bytes, and a word to 4bytes. The MIPS architecture needs 1bytes of storage for each character. There are 4bytes storage requirement for each integer.

Pseudocode

```
START
        score = 0
        current_puzzle = 0
        Display "Welcome to the UTeM Library Escape! Solve math puzzles to escape."
        Display "Press Enter to start..."
        Wait for user to press Enter

                Game_Loop:
                Display "Room: " + rooms[current_puzzle]
                Display  puzzles[current_puzzle]
                Display "Your answer: "
                Read user_answer

                IF user_answer == correct_answers[current_puzzle]
                        score += 20
                        Display "Correct! Your score is now: " + score
                ELSE
                        score -= 20 (minimum is 0)
                        Display "Incorrect! The correct answer was " +
                        correct_answers[current_puzzle]
                        Display "Your score is now: " + score
                        IF score == 0
                            Display "You didn't manage to escape this time. Keep trying."
                            End IF
                IF score >= 100
```

```
                Display "Congratulations! You've escaped with a score of " + score
                End IF
            Display "\nPress 1 for next puzzle, 2 for previous puzzle, or any other key to
            quit: "

        Read user_answer
            IF user_answer == "1"
             current_puzzle += 1 (max is 4)
            ELSE IF user_answer == "2"
                current_puzzle -=1 (min is 0)

            ELSE
                Display "You didn't manage to escape this time. Keep trying."
                End IF

STOP
```

Source Code

```
.data
    puzzles:
    .asciiz "What is 5 + 3?\n"
    .asciiz "What is 10 - 7?\n"
    .asciiz "What is 4 * 6?\n"
    .asciiz "What is 15 / 3?\n"
    .asciiz "Is 7 an even or an odd number?\n"

    correct_answers:
    .asciiz "8\n"
    .asciiz "3\n"
    .asciiz "24\n"
    .asciiz "5\n"
    .asciiz "odd\n"

    rooms:
    .asciiz "Free Area\n"
    .asciiz "Computer Room\n"
    .asciiz "Meeting Room\n"
    .asciiz "Chill Area\n"
    .asciiz "Private Room\n"

    welcome_msg: .asciiz "Welcome to the UTeM Library Escape! Solve math puzzles to escape.\n"
    correct_msg: .asciiz "Correct! Your score is now: "
    incorrect_msg: .asciiz "\nIncorrect! The correct answer was "
    score_now: .asciiz "Your score is now: "
    escape_msg: .asciiz "\nCongratulations! You've escaped with a score of "
    navigation_msg: .asciiz "\nPress 1 to go to the next puzzle, or 2 to go back to the previous puzzle (or any other key to quit):\n "
    failed_msg: .asciiz "You didn't manage to escape this time. Keep trying\n"
    newline: .asciiz "\n"
```

```
    score: .word 0
    current_puzzle: .word 0
    user_answer: .space 20
    one_str: .asciiz "1\n"
    two_str: .asciiz "2\n"

.text
main:

# Display welcome message
    li $v0, 4
    la $a0, welcome_msg
    syscall

    # Display start message
    li $v0, 4
    la $a0, start
    syscall

    # Get user input to start game
    li $v0, 8
    la $a0, user_answer
    li $a1, 20
    syscall

    # Add newline for separation
    li $v0, 4
    la $a0, newline
    syscall

    # Initialize score and puzzle index
    li $t0, 0
    sw $t0, score
```

```
    sw $t0, current_puzzle

game_loop:
    # Display current rooms
    lw $t0, current_puzzle
    la $t1, rooms
    jal get_string_offset
    move $a0, $t1
    li $v0, 4
    syscall

    # Display current puzzle
    lw $t0, current_puzzle
    la $t1, puzzles
    jal get_string_offset
    move $a0, $t1
    li $v0, 4
    syscall

    # Get user input
    li $v0, 8
    la $a0, user_answer
    li $a1, 20
    syscall

    # Compare answer
    lw $t0, current_puzzle
    la $t1, correct_answers
    jal get_string_offset
    move $a1, $t1
    la $a0, user_answer
    jal compare_strings

    bnez $v0, incorrect
```

```
correct:
    # Increase score
    lw $t0, score
    addi $t0, $t0, 20
    sw $t0, score

    # Display correct message and score
    li $v0, 4
    la $a0, correct_msg
    syscall

    li $v0, 1
    lw $a0, score
    syscall

    # Add newline for separation
    li $v0, 4
    la $a0, newline
    syscall

    j navigation

incorrect:
    # Decrease score, but not below 0
    lw $t0, score
    subi $t0, $t0, 20
    bgtz $t0, valid_score
    li $t0, 0
valid_score:
    sw $t0, score

    # Check if score reached zero to terminate
    beqz $t0, end_game
```

```
    # Display incorrect message and correct answer
    li $v0, 4
    la $a0, incorrect_msg
    syscall
    lw $t0, current_puzzle
    la $t1, correct_answers
    jal get_string_offset
    move $a0, $t1
    li $v0, 4
    syscall
    li $v0, 4
    la $a0, score_now
    syscall
    li $v0, 1
    lw $a0, score
    syscall

    # Add newline for separation
    li $v0, 4
    la $a0, newline
    syscall

navigation:
    # Check if score is high enough to escape
    lw $t0, score
    li $t1, 100
    bge $t0, $t1, escape

    # Display navigation options
    li $v0, 4
    la $a0, navigation_msg
    syscall
```

```
    # Read user choice as string
    li $v0, 8
    la $a0, user_answer
    li $a1, 20
    syscall

    # Check input
    la $a0, user_answer
    la $a1, one_str
    jal compare_strings
    beqz $v0, next_puzzle

    la $a0, user_answer
    la $a1, two_str
    jal compare_strings
    beqz $v0, prev_puzzle

    j end_game

next_puzzle:
    lw $t0, current_puzzle
    addi $t0, $t0, 1
    li $t1, 4
    ble $t0, $t1, update_puzzle_index
    li $t0, 4
update_puzzle_index:
    sw $t0, current_puzzle
    j game_loop

prev_puzzle:
    lw $t0, current_puzzle
    addi $t0, $t0, -1
    bgez $t0, update_puzzle_index
    li $t0, 0
```

```
    j update_puzzle_index

escape:
    # Display escape message
    li $v0, 4
    la $a0, escape_msg
    syscall
    li $v0, 1
    lw $a0, score
    syscall

exit:
    li $v0, 10
    syscall

end_game:
    li $v0, 4
    la $a0, failed_msg
    syscall
    j exit

get_string_offset:
    move $t2, $zero
loop_offset:
    beq $t0, $zero, return_offset
    subi $t0, $t0, 1
find_next_string:
    lb $t3, 0($t1)
    beqz $t3, update_offset
    addi $t1, $t1, 1
    j find_next_string
update_offset:
    addi $t1, $t1, 1
    j loop_offset
```

```
return_offset:
    jr $ra

# Simplified string comparison function
compare_strings:
    move $t0, $a0
    move $t1, $a1
compare_loop:
    lb $t2, 0($t0)
    lb $t3, 0($t1)
    beqz $t2, check_end
    bne $t2, $t3, not_equal
    addi $t0, $t0, 1
    addi $t1, $t1, 1
    j compare_loop
check_end:
    beqz $t3, equal
not_equal:
    li $v0, 1
    j return_compare
equal:
    li $v0, 0
return_compare:
    jr $ra
```

Output Sample

Program start with Welcome message and You must press Enter to start the game.

```
Welcome to the UTeM Library Escape! Solve math puzzles to escape.

Press Enter to start the game!!!
```

After you press Enter it will appear which rooms you are in and puzzle that you need to solve. Every correct answer you will get 20 points.

```
Free Area
What is 5 + 3?
8
Correct! Your score is now: 20

Press 1 to go to the next puzzle, or 2 to go back to the previous puzzle (or any other key to quit):
```

For every incorrect answer it will show the correct answer for the puzzle and your score will be deduct by 20 points.

```
Press 1 to go to the next puzzle, or 2 to go back to the previous puzzle (or any other key to quit):
 1
Meeting Room
What is 4 * 6?
25

Incorrect! The correct answer was 24
Your score is now: 20
```

If you choose 1 then you will be move to the next puzzle.

```
Press 1 to go to the next puzzle, or 2 to go back to the previous puzzle (or any other key to quit):
 1
Computer Room
What is 10 - 7?
3
Correct! Your score is now: 40
```

If you choose 2 then you will be move to the previous puzzle.

```
Press 1 to go to the next puzzle, or 2 to go back to the previous puzzle (or any other key to quit):
 2
Computer Room
What is 10 - 7?
3
Correct! Your score is now: 40
```

Once you get 100% score there will be escape message appears.

```
Private Room
Is 7 an even or an odd number?
odd
Correct! Your score is now: 100

Congratulations! You've escaped with a score of 100
-- program is finished running --
```

Otherwise , If you got 0% score the program will display failed message.

```
Incorrect! The correct answer was 24
Your score is now: 20

Press 1 to go to the next puzzle, or 2 to go back to the previous puzzle (or any other key to quit):
 1
Chill Area
What is 15 / 3?
1
You didn't manage to escape this time. Keep trying
```