**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:

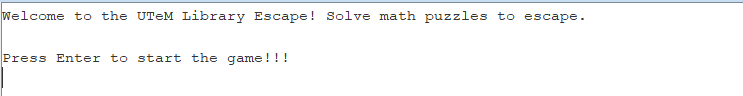
li $v0, 0

return\_compare:

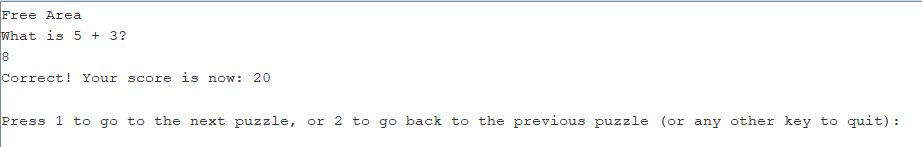
jr $ra

**Output Sample**

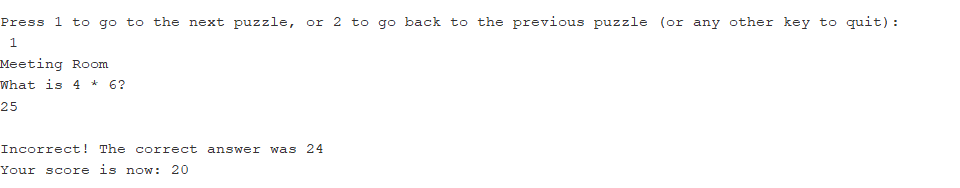
Program start with Welcome message and You must press Enter to start the game.



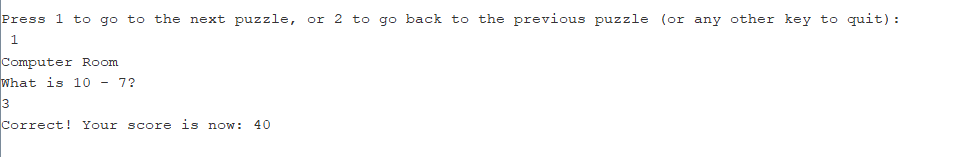
After you press Enter it will appear which rooms you are in and puzzle that you need to solve. Every correct answer you will get 20 points.



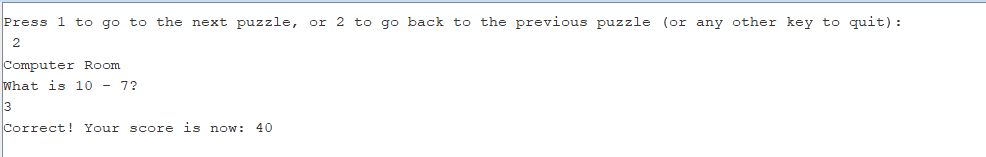
For every incorrect answer it will show the correct answer for the puzzle and your score will be deduct by 20 points.



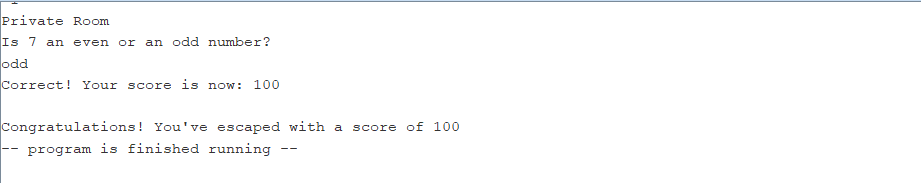
If you choose 1 then you will be move to the next puzzle.



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



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



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

