**EXPERIMENT 7**

**AIM:** Study MIPS Assembly Language Programming using MIPS simulator and implement the following:

1. To add 10 numbers
2. To print message "Hello MIPS".
3. To Reverse the input string ("ABC" - "CBA").

**THEORY:**

**MIPS Assembly Language Programming:**

MIPS is an acronym for Microprocessor without Interlocked Pipeline Stages. It is a reduced instruction set architecture developed by an organization called MIPS Technologies. The MIPS assembly language is a very useful language to learn because many embedded systems run on the MIPS processor. Knowing how to code in this language brings a deeper understanding of how these systems operate on a lower level.

A MAL program is divided into two types of sections:

**Data sections** specify actions to be taken during assembly. Usually declare memory variables used by the program.

**Text sections** define sequences of instructions executed by the program at run time.

Syntax: *label operation operand\_list # comment*

**Looping in MIPS:**

**for**

li $t0, 10 # t0 is a constant 10

li $t1, 0 # t1 is our counter (i)

loop:

beq $t1, $t0, end # if t1 == 10 we are done

loop body

addi $t1, $t1, 1 # add 1 to t1

j loop # jump back to the top

end:

**while:**

top\_while:

t0 = evaluate Cond

beqz $t0,end\_while

execute Statements

**j** top\_while

end\_while:

**CODE:**

**1)Program to add 10 nos.**

.data

newline: .asciiz "\n"

.text

main:

li $t0,0

li $t2,5

loop:

bgt $t0, 45, exit

addi $t0, $t0, 5

li $v0, 1

move $a0, $t0

syscall

li $v0, 4

la $a0, newline

syscall

j loop

exit:

li $v0,10

syscall

Graphical user interface, application

Description automatically generated**OUTPUT**

**2) To print hello message**

.data

myMessage \_: .asciiz "Hello MIPS"

.text

main:

li $v0, 4

myMessage syscall

li $v0, 10

syscall

Graphical user interface, application

Description automatically generated**OUTPUT**

**3) To reverse the input string**

.data

input: .space 256

output:.space 256

.text

.globl main

main:

li $v0, 8

la $a0, input

li $a1, 256

syscall

li $v0, 4

la $a0, input

syscall

jal strlen

add $t1, $zero, $v0

add $t2, $zero, $a0

add $a0, $zero, $v0

li $v0, 1

syscall

reverse:

li $t0, 0

li $t3, 0

reverse\_loop:

add $t3, $t2, $t0

lb $t4, 0($t3)

beqz $t4, exit

sb $t4, output($t1)

subi $t1, $t1, 1

addi $t0, $t0, 1

j reverse\_loop

exit:

li $v0, 4

la $a0, output

syscall

li $v0, 10

syscall

strlen:

li $t0, 0

li $t2, 0

strlen\_loop:

add $t2, $a0, $t0

lb $t1, 0($t2)

beqz $t1, strlen\_exit

addiu $t0, $t0, 1

j strlen\_loop

strlen\_exit:

subi $t0, $t0, 1

add $v0, $zero, $t0

add $t0, $zero, $zero

jr $ra

Graphical user interface, application

Description automatically generated**OUTPUT:**

Graphical user interface, application

Description automatically generated

**CONCLUSION:** In this experiment, I implemented 3 programs in MIPS Assembly Language (MAL) viz, addition of 10 numbers, print “Hello MIPS” and reverse the input string (‘MEITH’ -> ‘HTIEM’) using MIPS stimulator. The MAL is divided into 2 sections: Data and Text section. The Data section usually have declaration of memory variables used by the program and the Text sections define sequences of instructions executed by the program at run time. In this experiment I learnt how to implement loops in the MAL, how to take inputs and print the outputs and convert a complex expression into an assembly language.