NAME: Meith Navlakha
SAP: 60004190068
POA Practical

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:
```

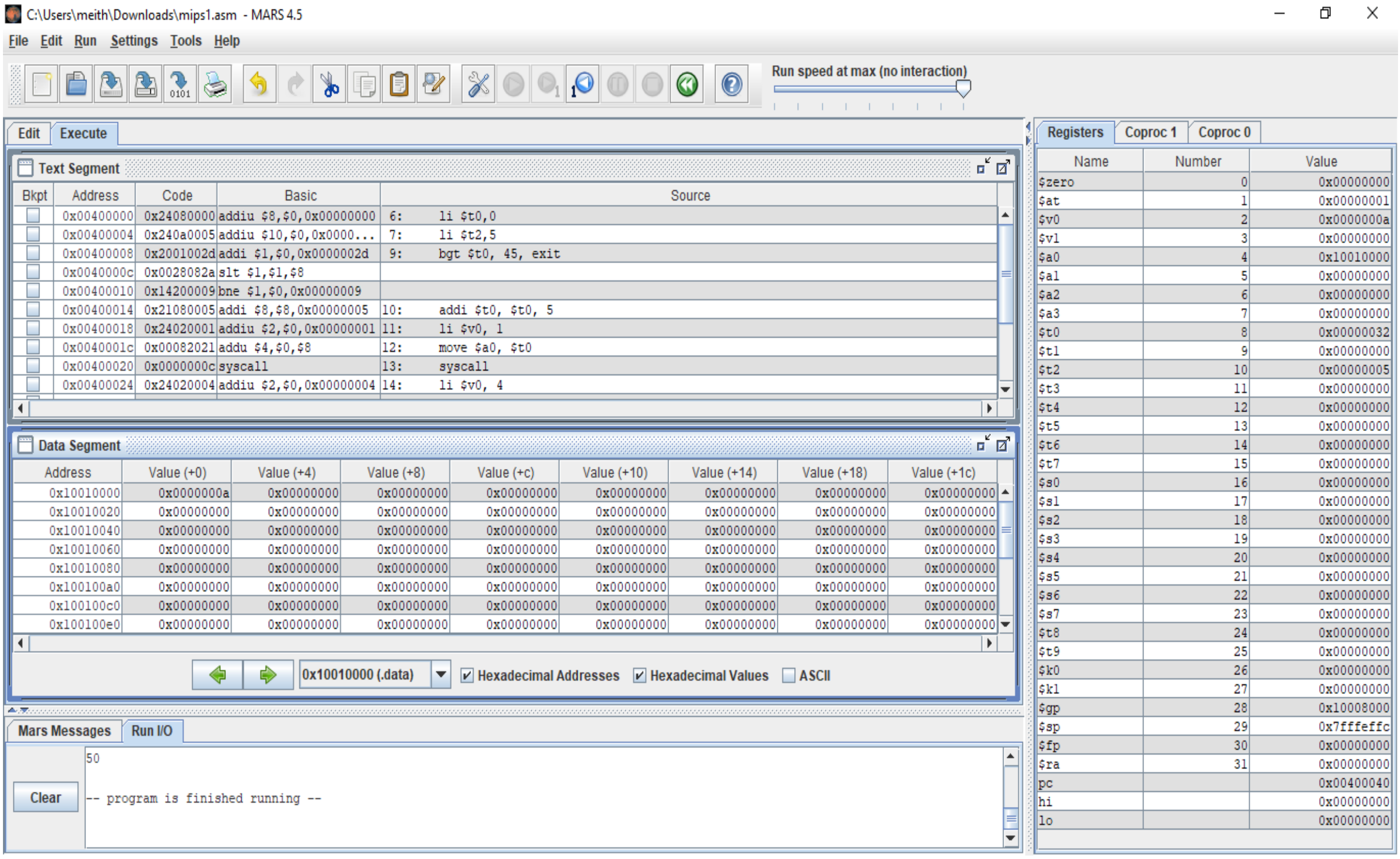
CODE:

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
```

NAME: Meith Navlakha
SAP: 60004190068
POA Practical

OUTPUT



Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x24080000	addiu $8,$0,0x00000000	6: li $t0,0
□	0x00400004	0x240a0005	addiu $10,$0,0x0000...	7: li $t2,5
□	0x00400008	0x2001002d	addi $1,$0,0x0000002d	9: bgt $t0, 45, exit
□	0x0040000c	0x0028082a	slt $1,$1,$8	
□	0x00400010	0x14200009	bne $1,$0,0x00000009	
□	0x00400014	0x21080005	addi $8,$8,0x00000005	10: addi $t0, $t0, 5
□	0x00400018	0x24020001	addiu $2,$0,0x00000001	11: li $v0, 1
□	0x0040001c	0x00082021	addu $4,$0,$8	12: move $a0, $t0
□	0x00400020	0x0000000c	syscall	13: syscall
□	0x00400024	0x24020004	addiu $2,$0,0x00000004	14: li $v0, 4

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x0000000a	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

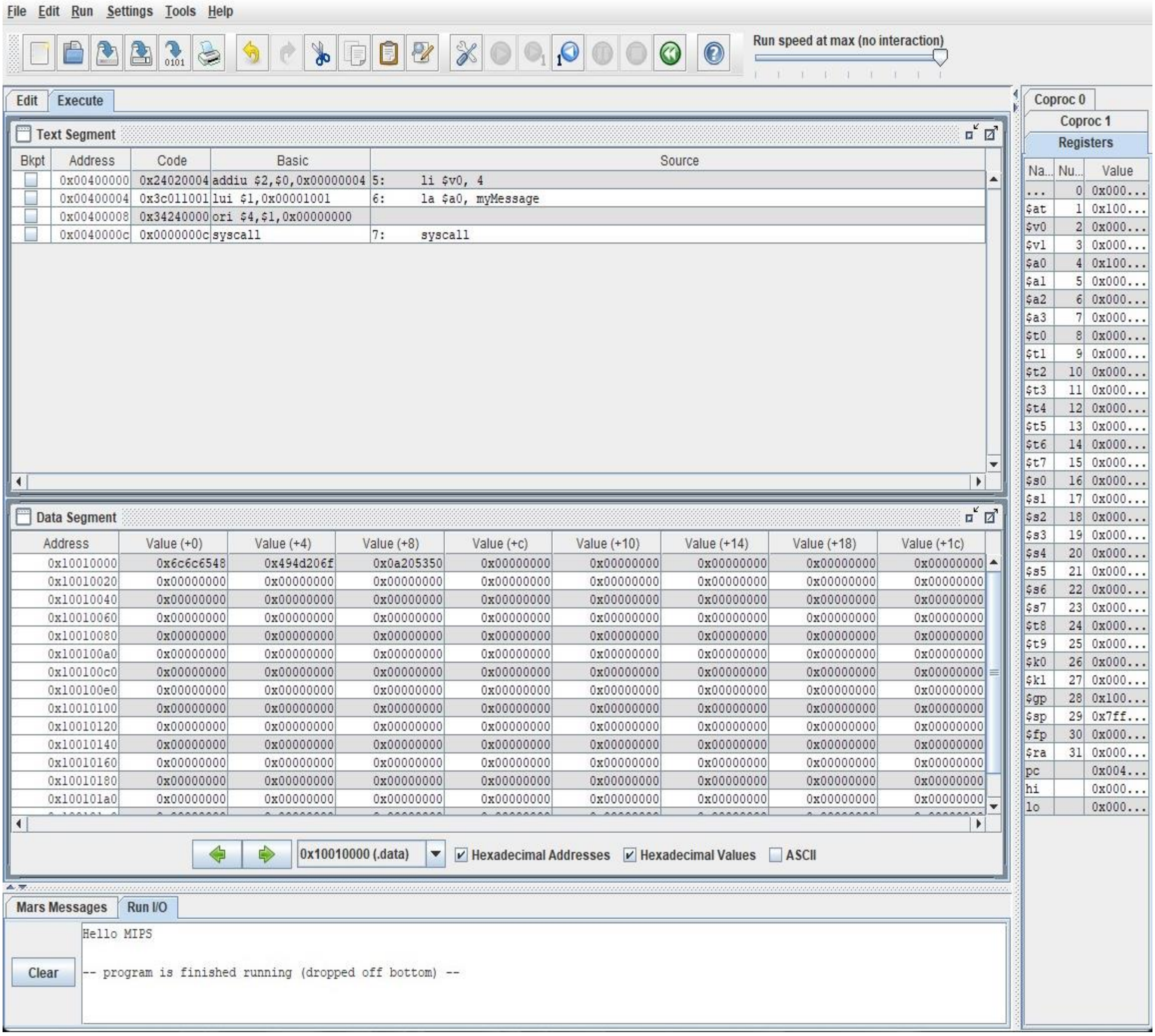
```
50

-- program is finished running --
```

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000001
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x10010000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000032
$t1	9	0x00000000
$t2	10	0x00000005
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400040
hi		0x00000000
lo		0x00000000

2) To print hello message

```
.data
myMessage _: .asciiz "Hello MIPS"

.text
main:
li $v0, 4
myMessage syscall

li $v0, 10
syscall
```

OUTPUT


File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Coproc 0 Coproc 1 Registers
0x10010000 (.data)
☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII
Mars Messages Run I/O
Clear

Text Segment

Bkpt	Address	Code	Basic	Source
□	0x00400000	0x24020004	addiu $2,$0,0x00000004	5: li $v0, 4
□	0x00400004	0x3c011001	lui $1,0x00001001	6: la $a0, myMessage
□	0x00400008	0x34240000	ori $4,$1,0x00000000	
□	0x0040000c	0x0000000c	syscall	7: syscall

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x6c6c6548	0x494d206f	0x0a205350	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010160	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010180	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100101a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Na...	Nu...	Value
...	0	0x000...
$at	1	0x100...
$v0	2	0x000...
$v1	3	0x000...
$a0	4	0x100...
$a1	5	0x000...
$a2	6	0x000...
$a3	7	0x000...
$t0	8	0x000...
$t1	9	0x000...
$t2	10	0x000...
$t3	11	0x000...
$t4	12	0x000...
$t5	13	0x000...
$t6	14	0x000...
$t7	15	0x000...
$s0	16	0x000...
$s1	17	0x000...
$s2	18	0x000...
$s3	19	0x000...
$s4	20	0x000...
$s5	21	0x000...
$s6	22	0x000...
$s7	23	0x000...
$t8	24	0x000...
$t9	25	0x000...
$k0	26	0x000...
$k1	27	0x000...
$gp	28	0x100...
$sp	29	0x7ff...
$fp	30	0x000...
$ra	31	0x000...
pc		0x004...
hi		0x000...
lo		0x000...

```
Hello MIPS

-- program is finished running (dropped off bottom) --
```

3) To reverse the input string

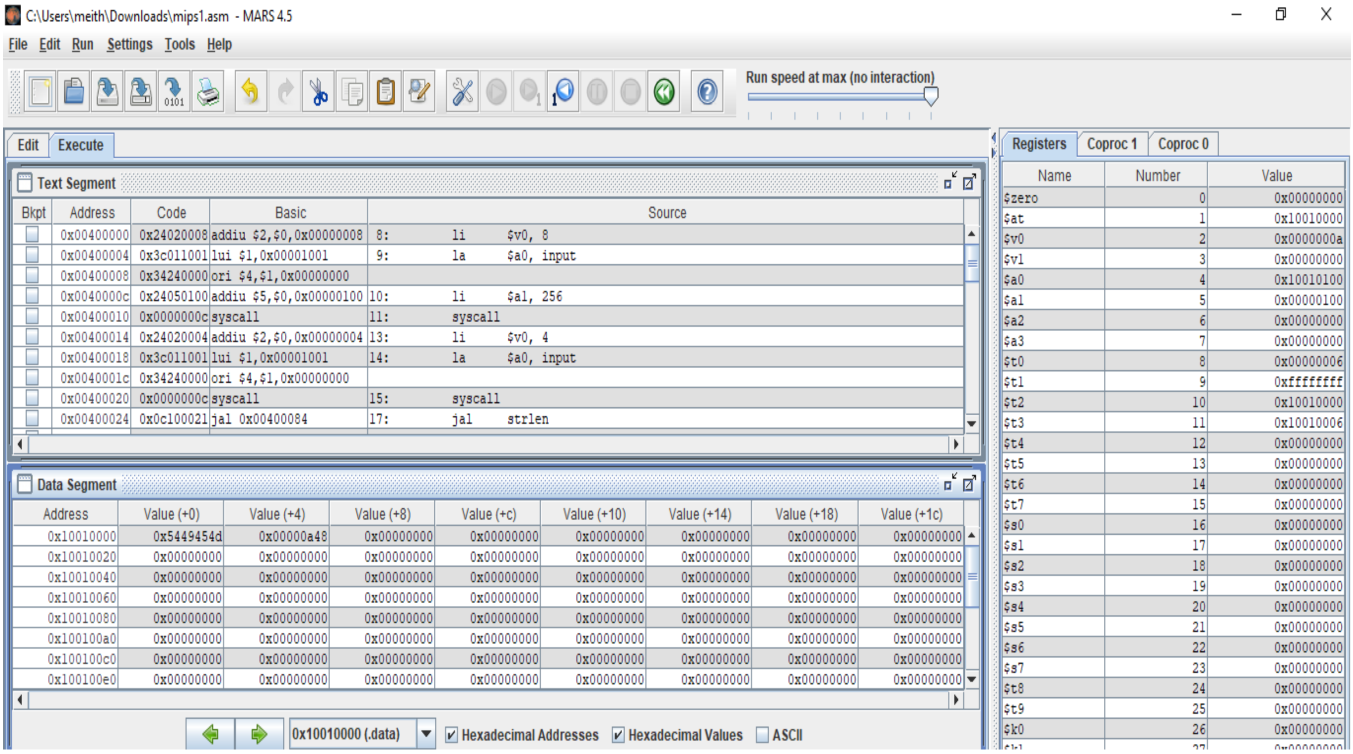
```
.data
input:  .space 256
output:.space 256

.text
.globl main
```

NAME: Meith Navlakha
SAP: 60004190068
POA Practical

```
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
```

OUTPUT:


C:\Users\meith\Downloads\mips1.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Registers Coproc 1 Coproc 0

Text Segment

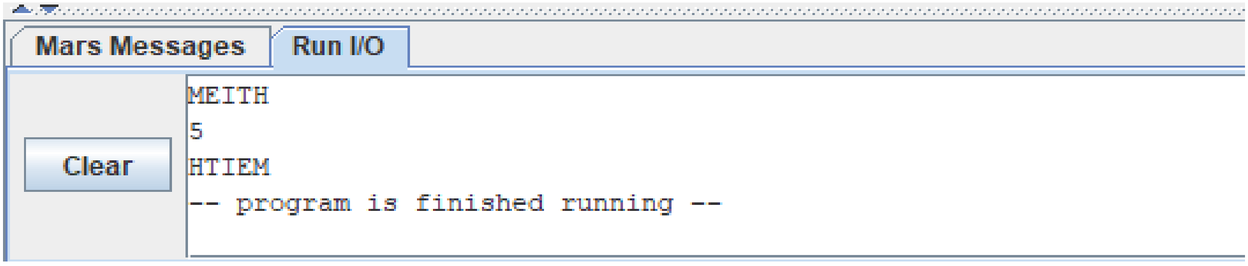
Bkpt	Address	Code	Basic	Source
□	0x00400000	0x24020008	addiu $2,$0,0x00000008	8: li $v0, 8
□	0x00400004	0x3c011001	lui $1,0x00001001	9: la $a0, input
□	0x00400008	0x34240000	ori $4,$1,0x00000000	
□	0x0040000c	0x24050100	addiu $5,$0,0x00000100	10: li $a1, 256
□	0x00400010	0x0000000c	syscall	11: syscall
□	0x00400014	0x24020004	addiu $2,$0,0x00000004	13: li $v0, 4
□	0x00400018	0x3c011001	lui $1,0x00001001	14: la $a0, input
□	0x0040001c	0x34240000	ori $4,$1,0x00000000	
□	0x00400020	0x0000000c	syscall	15: syscall
□	0x00400024	0x0c100021	jal 0x00400084	17: jal strlen

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x5449454d	0x00000a48	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x10010000 (.data) ☑ Hexadecimal Addresses ☑ Hexadecimal Values □ ASCII

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x10010100
$a1	5	0x00000100
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000006
$t1	9	0xffffffff
$t2	10	0x10010000
$t3	11	0x10010006
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000



```
MEITH
5
HTIEM
-- program is finished running --
```

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.