MIPS PROCESSOR DESIGN

Course Name: EG 212 Computer Architecture – Processor

Design Group Number – 10

Members – Harsh Gupta (IMT2023121), Shivek Ranjan (IMT2023042), Navish Malik (IMT2023060)

INTRODUCTION

This report explores the fundamental concepts of an MIPS Processor through design and analysis. We have written three MARS simulations and designed a processor which takes the machine code from MARS and does the specified computations.

This report include these following files:

- armstrong.asm
- fib.asm
- quadratic.asm
- mips_processor.py

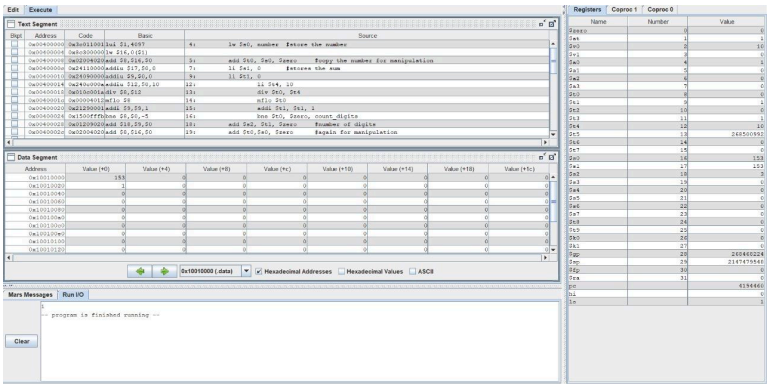
IMPLEMENTATION

Armstrong number:

The first MIPS program takes an integer and prints 1 if the given integer is an armstrong number, else 0. An Armstrong number is a number that is equal to the sum of its own digits each raised to the power of the number of digits.

Screenshot-

```
data
        number: .word 153
text
        lw $s0, number  #store the number
        add $t0, $s0, $zero     #copy the number for manipulation

        li $s1, 0       #stores the sum

        li $t1, 0

        count_digits:
                li $t4, 10
                div $t0, $t4
                mflo $t0
                addi $t1, $t1, 1
                bne $t0, $zero, count_digits

        add $s2, $t1, $zero     #number of digits
        add $t0,$s0, $zero      #again for manipulation

        calculate_sum:
                li $t4, 10
                div $t0, $t4
                mfhi $t1 #storing remainder
                mflo $t0
                add $t2, $s2, $zero
                li $t3, 1
```



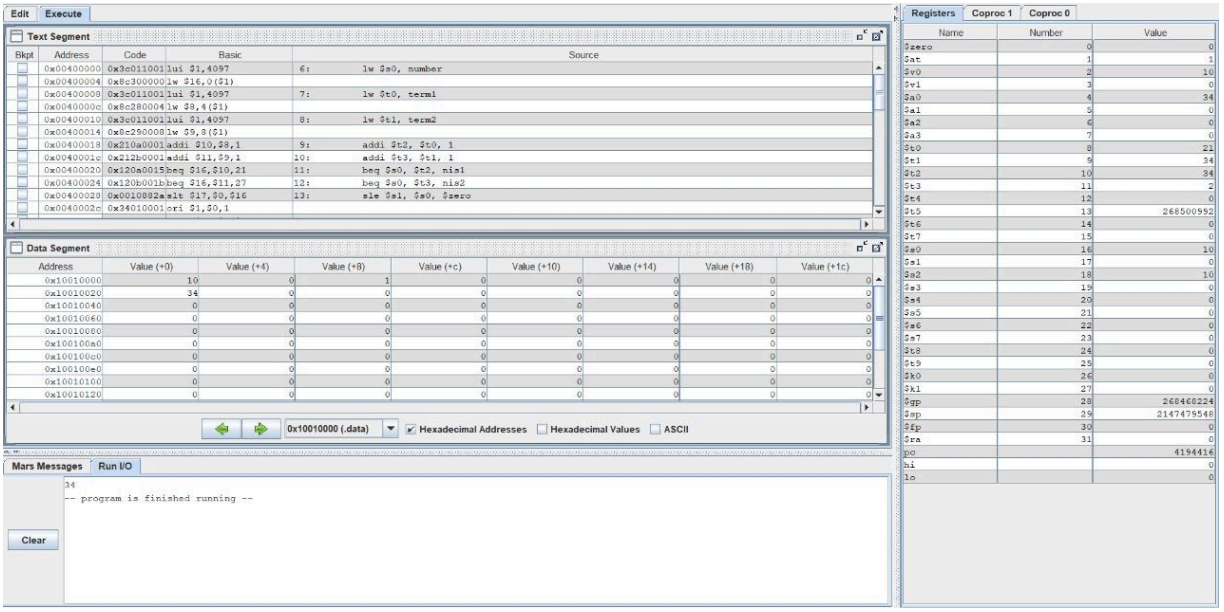
Fibonacci term:

The second MIPS program calculates and prints the given term of the Fibonacci sequence, where n is the integer input by the user. The Fibonacci sequence is a series of numbers where each number is the sum of the two preceding ones, starting with 0 and 1.

Screenshot –

armstrong.asm | fib.asm

```
1   .data
2           number: .word 10
3           term1: .word 0
4           term2: .word 1
5   .text
6           lw $s0, number
7           lw $t0, term1
8           lw $t1, term2
9           addi $t2, $t0, 1
10          addi $t3, $t1, 1
11          beq $s0, $t2, nis1
12          beq $s0, $t3, nis2
13          sle $s1, $s0, $zero
14          beq $s1, $t1, nisnegative
15          li $t2, 0         #fib3
16          li $s2, 2         #term counter
17
18          fib:
19                  add $t2, $t1, $t0
20                  add $t0, $t1, $zero
21                  add $t1, $t2, $zero
22                  addi $s2, $s2, 1
23                  bne $s2, $s0, fib
24          li $v0, 1
25          add $a0, $t1, $zero
26          syscall
27          lui $t5, 4097
```



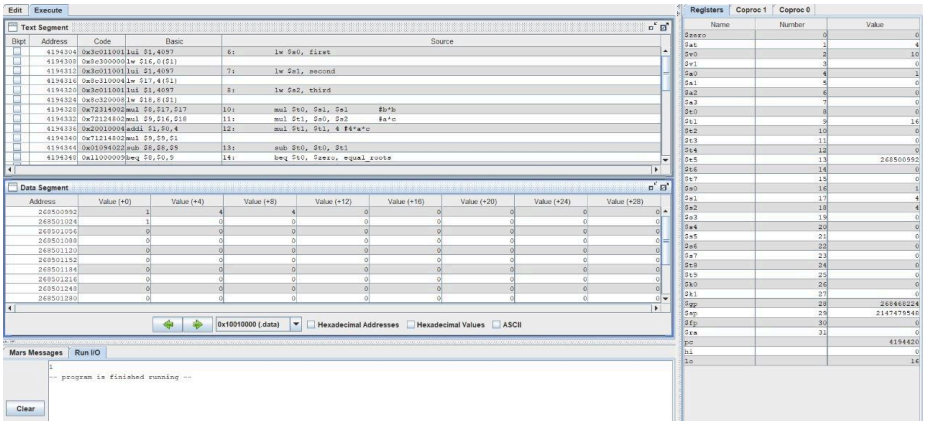
Number of Roots of Quadratic:

The third MIPS program takes three integers as input, representing the coefficients of a quadratic equation (ax^2 + bx + c = 0), and determines the number of real roots the quadratic equation has. Then, it prints the number of real roots of the given equation.

screenshot-

armstrong.asm | fib.asm | quardratic.asm

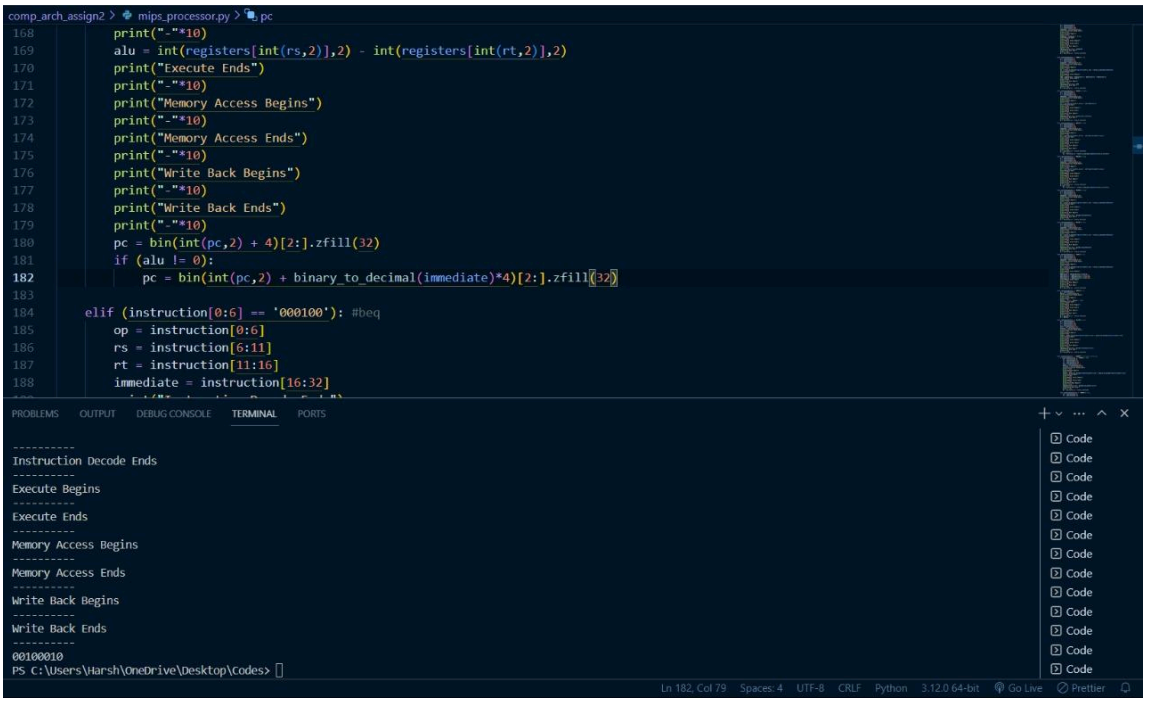
```
1   .data
2           first: .word 1
3           second: .word 4
4           third: .word 4
5   .text
6           lw $s0, first
7           lw $s1, second
8           lw $s2, third
9
10          mul $t0, $s1, $s1       #b*b
11          mul $t1, $s0, $s2       #a*c
12          mul $t1, $t1, 4 #4*a*c
13          sub $t0, $t0, $t1
14          beq $t0, $zero, equal_roots
15          slt $t1, $t0, $zero
16          beq $t1, 1, no_roots
17
18          real_roots:
19                  li $v0, 1
20                  li $a0, 2
21                  syscall
22                  lui $t5, 4097
23                  sw $a0, 32($t5)
24                  j exit
25
26          equal_roots:
27                  li $v0, 1
```

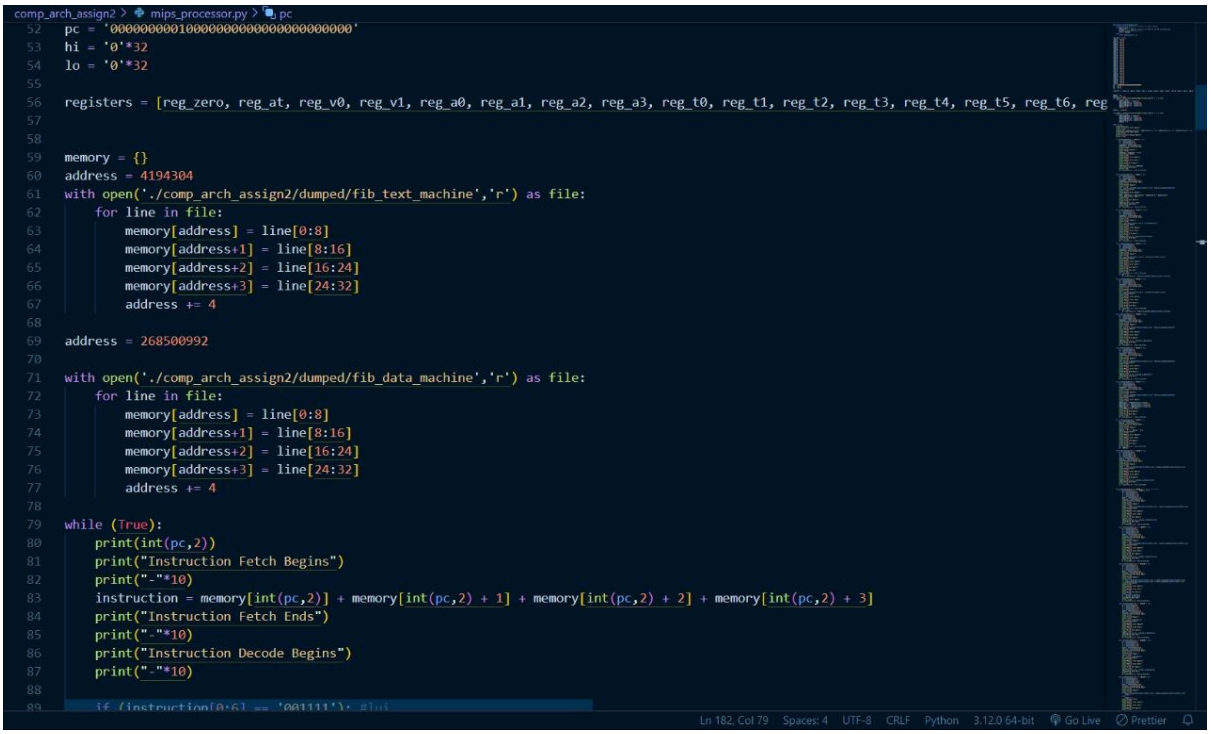


MIPS Processor:

The processor reads the machine code (data and text) dumped by MARS. It performs certain operations corresponding to different instruction opcodes. It also contains a memory that is used to depict system memory. We save the instructions in the memory and feed to the processor using a while loop. Each register is initialized as an empty string in the beginning.

Screenshot:(only a snippet of the original code)





Link:
https://iiitbac-my.sharepoint.com/:f:/g/personal/harsh_gupta_iiitb_ac_in/ErkDK8m6leNIptVb1tSHyeEBZrP044ueWR9qd5ZK-IdtTw?e=SXrLjd