

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
```

[illegible]

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.

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

The screenshot displays a MIPS simulator interface. The main window is divided into several sections:

- Text Segment:** Shows assembly code with columns for Address, Code, Basic, and Source. The code includes instructions like `lw $s0, number`, `lw $t0, term1`, `lw $t1, term2`, `addi $t2, $t0, 1`, `addi $t3, $t1, 1`, `beq $s0, $t2, mis1`, `beq $s0, $t3, mis2`, and `sle $s1, $s0, $zero`.
- Data Segment:** A table showing memory addresses and their corresponding values. The values are mostly 0, with some non-zero values at specific addresses.
- Registers:** A table showing the state of MIPS registers. The `$zero` register is 0, `$at` is 1, `$v0` is 10, `$v1` is 3, `$a0` is 34, `$a1` is 5, `$a2` is 6, `$a3` is 7, `$t0` is 9, `$t1` is 34, `$t2` is 10, `$t3` is 11, `$t4` is 12, `$t5` is 13, `$t6` is 14, `$t7` is 15, `$s0` is 16, `$s1` is 17, `$s2` is 18, `$s3` is 19, `$s4` is 20, `$s5` is 21, `$s6` is 22, `$s7` is 23, `$t8` is 24, `$t9` is 25, `$k0` is 26, `$k1` is 27, `$gp` is 28, `$sp` is 29, `$fp` is 30, `$ra` is 31, `$lo` is 0, and `$hi` is 4194416.
- Mars Messages:** A text area showing the message: "program is finished running --".

Number of Roots of Quadratic:

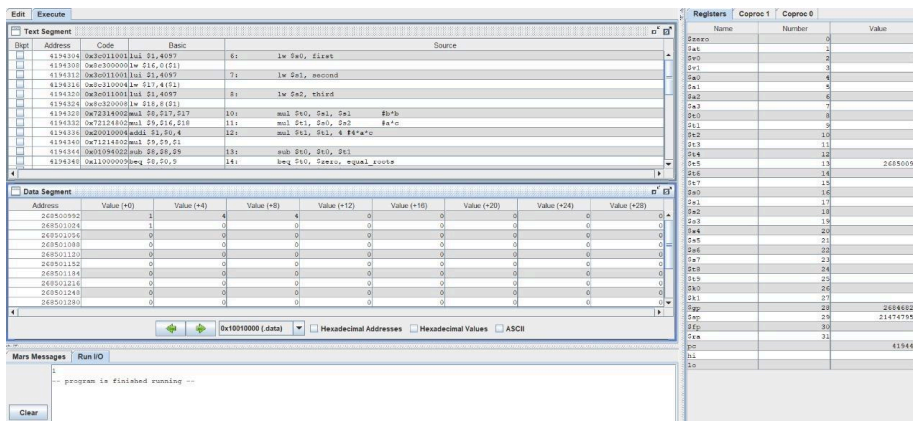
The third MIPS program takes three integers as input, representing the coefficients of a quadratic equation ($ax^2 + bx + c = 0$), and determines the number of real roots the quadratic equation has. Then, it prints the number of real roots of the given equation.

screenshot-

```

armstrong.asm  fib.asm  quadratic.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)

```
comp_arch_assign2 > mips_processor.py > pc
168     print("-"*10)
169     alu = int(registers[int(rs,2)],2) - int(registers[int(rt,2)],2)
170     print("Execute Ends")
171     print("-"*10)
172     print("Memory Access Begins")
173     print("-"*10)
174     print("Memory Access Ends")
175     print("-"*10)
176     print("Write Back Begins")
177     print("-"*10)
178     print("Write Back Ends")
179     print("-"*10)
180     pc = bin(int(pc,2) + 4)[2:].zfill(32)
181     if (alu != 0):
182         pc = bin(int(pc,2) + binary_to_decimal(immediate)*4)[2:].zfill(32)
183
184     elif (instruction[0:6] == '000100'): #beq
185         op = instruction[0:6]
186         rs = instruction[6:11]
187         rt = instruction[11:16]
188         immediate = instruction[16:32]
```

```
comp_arch_assign2 > mips_processor.py > pc
52 pc = '00000000100000000000000000000000'
53 hi = '0'*32
54 lo = '0'*32
55
56 registers = [reg_zero, reg_at, reg_v0, reg_v1, reg_a0, reg_a1, reg_a2, reg_a3, reg_t0, reg_t1, reg_t2, reg_t3, reg_t4, reg_t5, reg_t6, reg
57
58
59 memory = {}
60 address = 4194304
61 with open('./comp_arch_assign2/dumped/fib_text_machine','r') as file:
62     for line in file:
63         memory[address] = line[0:8]
64         memory[address+1] = line[8:16]
65         memory[address+2] = line[16:24]
66         memory[address+3] = line[24:32]
67         address += 4
68
69 address = 268500992
70
71 with open('./comp_arch_assign2/dumped/fib_data_machine','r') as file:
72     for line in file:
73         memory[address] = line[0:8]
74         memory[address+1] = line[8:16]
75         memory[address+2] = line[16:24]
76         memory[address+3] = line[24:32]
77         address += 4
78
79 while (True):
80     print(int(pc,2))
81     print("Instruction Fetch Begins")
82     print("-"*10)
83     instruction = memory[int(pc,2)] + memory[int(pc,2) + 1] + memory[int(pc,2) + 2] + memory[int(pc,2) + 3]
84     print("Instruction Fetch Ends")
85     print("-"*10)
86     print("Instruction Decode Begins")
87     print("-"*10)
88
89     if (instruction[0:6] == '001111'): exit()
```

Link:

https://iitbac-my.sharepoint.com/:f:/g/personal/harsh_gupta_iitb_ac_in/ErkDK8m6leNIptVb1tSHyeEBZrP044ueWR9qd5ZK-ltdTw?e=SXrLjd