Assignment 1

EG 212/Computer architecture- MIPS processor design

Student Name: Sarthak Raj
Student ID: IMT2023569
Student Name: Harjot Singh
Student ID: IMT2023064
Student Name: Archit Jaju
Student ID: IMT2023128

26 February 2024

Abstract

In this section, we will examine the inner workings of the MIPS assembly code. focusing on the design and implementation of a non-pipelined MIPS processor. Our objective is to understand the intricacies of processor architecture by translating C programs into MIPS assembly language and executing them on the MARS assembler platform.

1 Introduction

In this report, we have chosen to perform the calculation of Calculating the nth fibonacci sequence entry for a given number n. Calculating the frequency of a given element in an array of given length.and The sum of first N natural numbers by providing instructions in mars code. With the help of the assembler, it should read the assembly code and generate instruction code of 32bit for non-pipelined MIPS processor for to decode them, and then execute them.

2 C codes

2.1 (1)Fibonacci sequence

```
#include <iostream>
using namespace std;

int main() {
    int a = 0, b = 1, c, n;

    cout << "Enter the number of terms: ";
    cin >> n;

    cout << "Fibonacci Series:" << std::endl;

    if (n >= 1) {
        cout << a << " ";
    }
    if (n >= 2) {
        cout << b << " ";
    }

    for (int i = 3; i <= n; ++i) {
        c = a + b;
        std::cout << c << " ";
        a = b;
        b = c;
    }

}
```

:

2.2 (2) Sum of n natural numbers

```
#include <iostream>
using namespace std;
int main() {
    int n;
    cout << "Enter a positive integer: ";
    cin >> n;

    if (n < 0) {
        cout << "Please enter a positive integer." << std::endl;
        return 1;
    }

    int sum = 0;
    for (int i = 1; i <= n; ++i) {
        sum += i;
    }

    cout << "The sum of the first " << n << " natural numbers is: " << sum <<
        endl;
    return 0;
}
```

:

2.3 (3)Total number of occurrences in the list

```
#include <iostream>
using namespace std;
#include <vector>
int countOccurrences(const std::vector<int>& lst, int num) {
    int count = 0;
    for (int element : lst) {
        if (element == num) {
            count++;
        }
    }
    return count;
}
int main() {
    std::vector<int> lst;
    int n, num;

    cout << "Enter the number of integers in the list: ";
    cin >> n;
  cout << "Enter " << n << " integers separated by spaces: ";
    for (int i = 0; i < n; ++i) {
        int val;
      cin >> val;
        lst.push_back(val);
    }
  cout << "Enter the integer you want to find: ";
   cin >> num;
```

:

3 MIPS instructions

3.1 Fibonacci sequence

Edit Execute

fib.asm*

```

main:
    lui $s0, 0x1001

    li $v0, 5                   # System call code for reading integer
    syscall
    sw $v0, 0($s0)              # Store the user input in location input_num

loop:

    lw $s1, 16($s0)
    lw $s2, 0($s0)
    beq $s1, $s2, exit        # if iterator==input_num -> branch to exit

    lw $s3, 4($s0)
    lw $s4, 8($s0)
    lw $s5, 12($s0)
    sw $s4, 4($s0)            # a = b
    sw $s5, 8($s0)            # b = c
    add $s6, $s4, $s5
    sw $s6, 12($s0)           # c = a+b
    addi $s1, $s1, 1          # iterator++
    sw $s1, 16($s0)

    j loop                    # go to loop again
exit:
    lw $s7, 4($s0)
    sw $s7, 20($s0)           # storing the returned value(a) in ans

    li  $v0, 1                # System call code for printing integer
    add $a0, $s7, $zero
    syscall

    li $v0, 10                # System call to exit
    syscall
```

:

3.2 Sum of n natural numbers

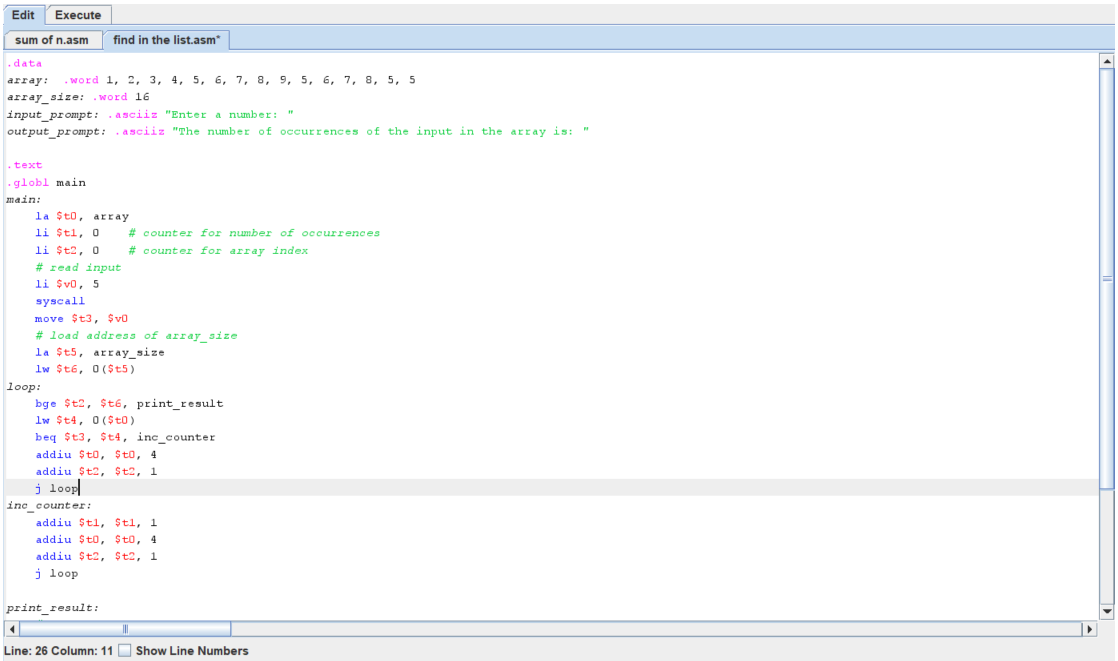
Edit Execute

sum of n.asm*

```
main:
    # load base address
    lui $s0, 0x1001

    # read integer input
    li $v0, 5
    syscall
    sw $v0, n       # store the user input in location n

loop:
    # load n and i
    lw $t0, n
    lw $t1, i

    # compute n+1
    addi $t0, $t0, 1

    # check if i == n+1, if yes then exit loop
    beq $t1, $t0, exit

    # load sum, compute sum + i, and store sum
    lw $t3, sum
    add $t3, $t3, $t1
    sw $t3, sum

    # increment i and store it
    addi $t1, $t1, 1
    sw $t1, i

    # jump to loop
    j loop

exit:
    # load sum, print integer, and exit
```

:

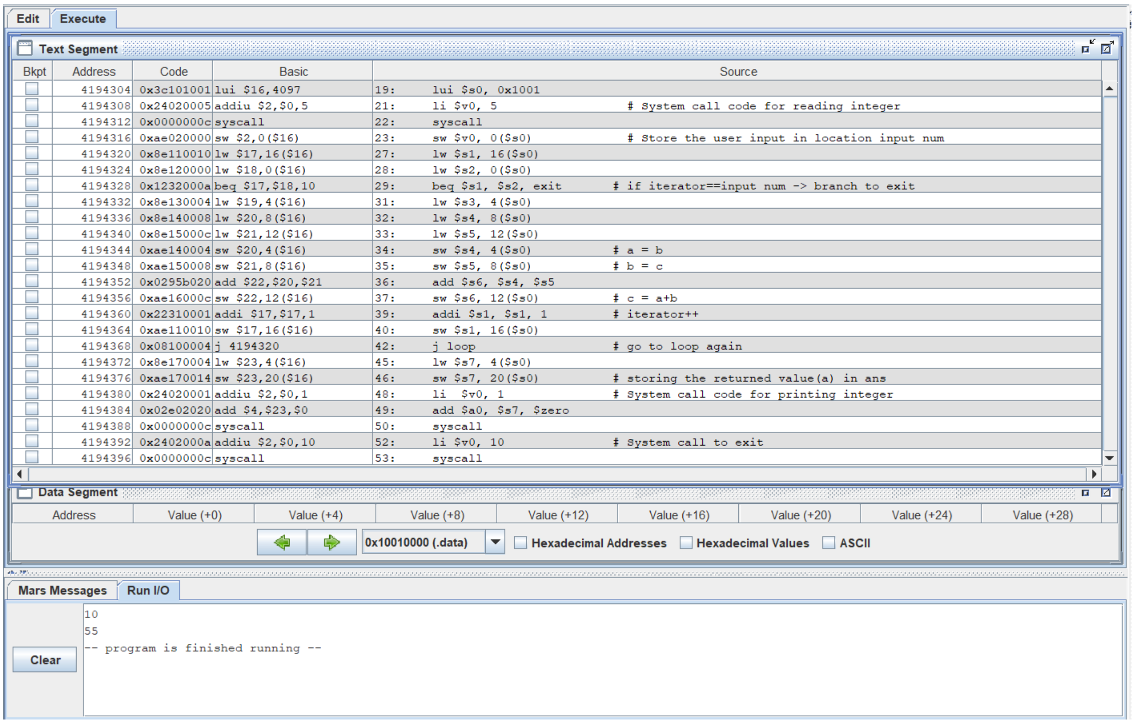
3.3 Total number of occurrences in the list



:

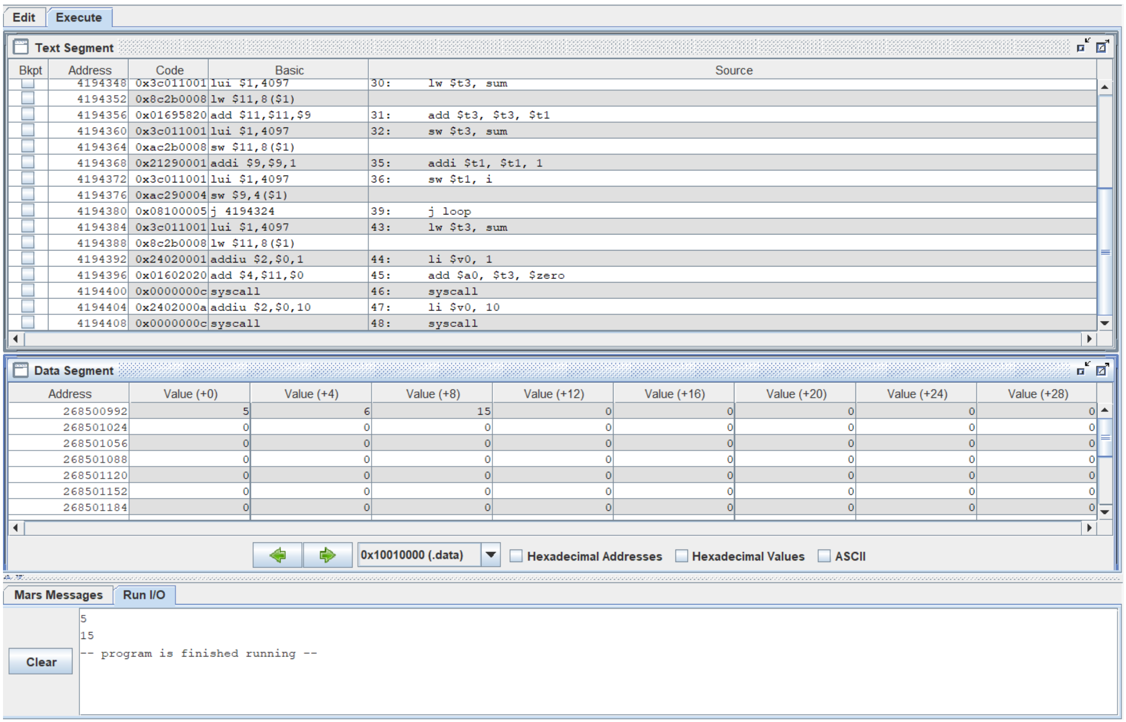
4 MIPS Result

4.1 Fibonacci sequence



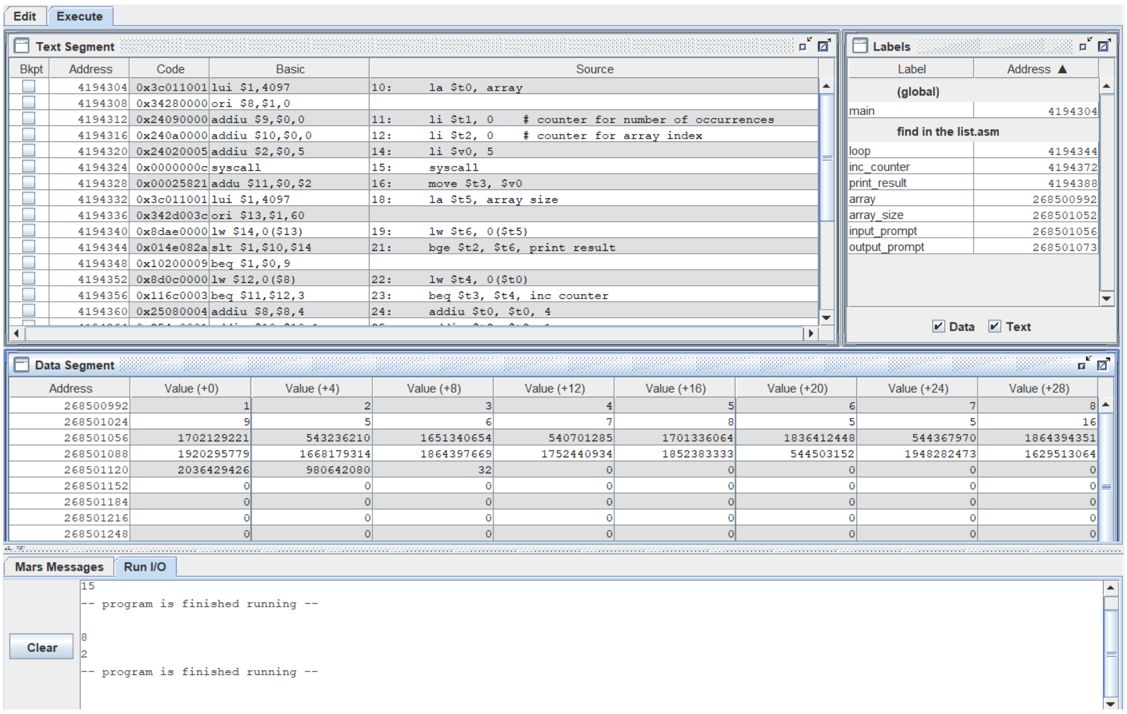
:

4.2 Sum of n natural numbers



:

4.3 Total number of occurrences in the list



:

5 Conclusion

Through this assignment, we delved into computer architecture and assembly language programming, focusing on developing a plague checker program. By translating C code into MIPS assembly and crafting a non-pipelined MIPS processor, we gained insights into computing systems for epidemiological analysis. Adhering to guidelines, we included essential instructions for plague checking algorithms, facilitating testing with the MARS assembler. Designing the MIPS processor provided hands-on understanding of architecture, memory management, and instruction execution for epidemiological computation. This experience broadened our knowledge and equipped us with practical skills for real-world challenges in public health. We are confident these lessons will support our future work in epidemiology and computational health