The American University in Cairo

Computer Science and Engineering Department

RISC-V Simulator Project Report

Gehad Salem 900205068
Mario Mamdouh 900202178
Ahmed Asaad 900201621

Brief Description of Implementation

The goal of this project is to implement a functional RISC-V simulator capable of tracing the execution of RV32I instructions. This RISC-V simulator was programmed using C++. Our implementation starts in the main, where we open two files. The first file contains the data for the program, this includes the program counter and any values we want to initialize. The second file contains the file program which has the RISC-V instructions that we want to execute. We take these instructions in order to process and execute them in our simulator. When it comes to the processing, the first thing we do is search for labels, and if a label is found, the program will mark this label and its specific location in order to process it later in the program. Afterwards, the program goes through each RISC-V instruction and identifies which of forty user-level instructions were called. We split these 40 instructions into a number of categories depending on their type and function in order to make it easier to organize and process according to each instruction's specific logic. We did this using a series of vectors and if else conditions in order to execute a different unique instruction for each instruction in the program. Finally, we have a function that prints the contents of the registers and the memory by the end of the program.

When it comes to the bonus features we implemented, our simulator program bonus 3 and bonus 7. For bonus 3, we output all values in decimal, binary, and hexadecimal instead of just decimal which is assumed to be the default. We do this using the functions decimalToBase and decimalToHexa. Additionally, we did bonus 7, which is including a larger set of test programs (at least 6 meaningful programs) and their equivalent C programs. Please refer to the "List of Programs" section found at the end of this report.

Assumptions

In this project, we have made a few assumptions to ensure that the simulator is able to run properly. These assumptions are as follows:

1. The user should enter the values that are stored in the memory as well as the respective addresses they are stored in.
2. The label must be in the same line as the instruction, with the instruction being put after the colon. For example, it should be like this:
   L1: addi x10, x29, 3
3. The registers are from x0 to x31 they can also be addressed as t0, a0 etc.
4. The RISC-V program instructions should be saved in a .txt file.

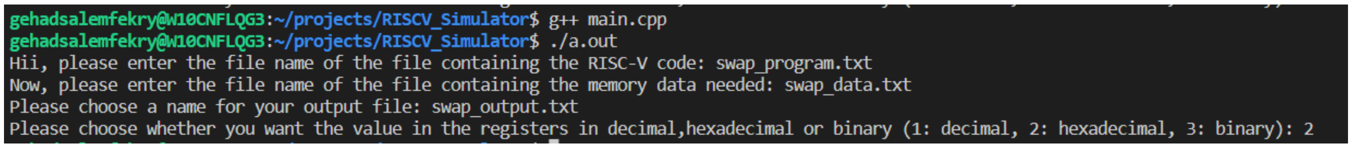
Known Bugs or Issues

After running the program dozens of times we have found the following bigs/issues:

1. Error occurs if there are comments on lines without any instructions in the program
2. Error occurs if there are comments placed BEFORE the instructions
3. Error occurs if the label is not in the same line as the instructions
4. Error occurs if the instructions or data are not in a .txt file format

User Guide

This RISC-V simulator is capable of tracing the execution of RV32I instructions. We have implemented all forty user-level instructions (listed in page 130 of the RISC-V Instruction Set Manual – Volume I). This user guide will show you how to compile and run this RISC-V simulator. First you should download and open the program with all its files on your respective Integrated Development Environment (IDE).



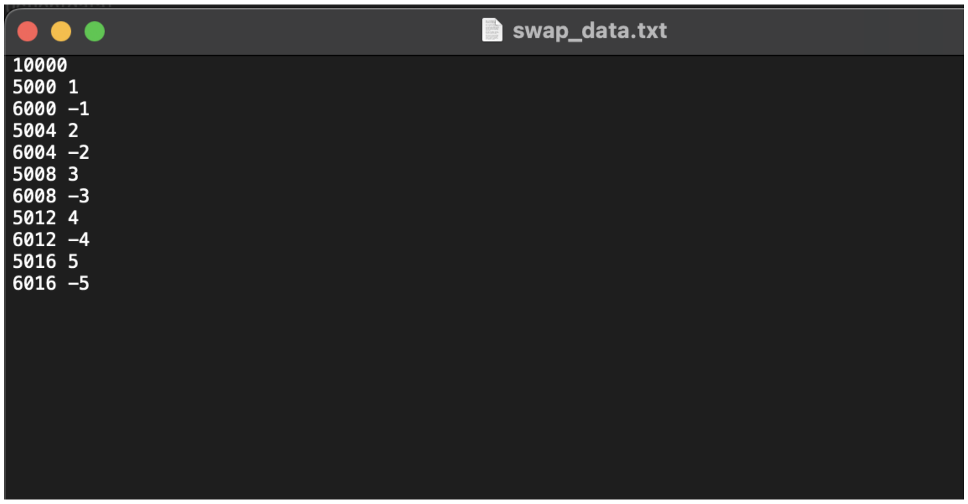
```
gehadsalemfekry@W10CNFLQG3:~/projects/RISCV_Simulator$ g++ main.cpp
gehadsalemfekry@W10CNFLQG3:~/projects/RISCV_Simulator$ ./a.out
Hii, please enter the file name of the file containing the RISC-V code: swap_program.txt
Now, please enter the file name of the file containing the memory data needed: swap_data.txt
Please choose a name for your output file: swap_output.txt
Please choose whether you want the value in the registers in decimal,hexadecimal or binary (1: decimal, 2: hexadecimal, 3: binary): 2
```

In this example, we will be using the swap program. Upon running the program, the user will be asked to enter the name of the file containing the RISC-V code and then he/she will be asked to enter the name of the file containing the memory data needed. For this swap example, the user should enter “swap_program.txt” and “swap_data.txt”. Each of these files should already be saved in the same directory as main.cpp and should contain the correct information needed. Additionally, the user will be asked to enter the name of the output file, here he/she should enter “swap_output.txt”.

Here is what each of the two former files should contain:

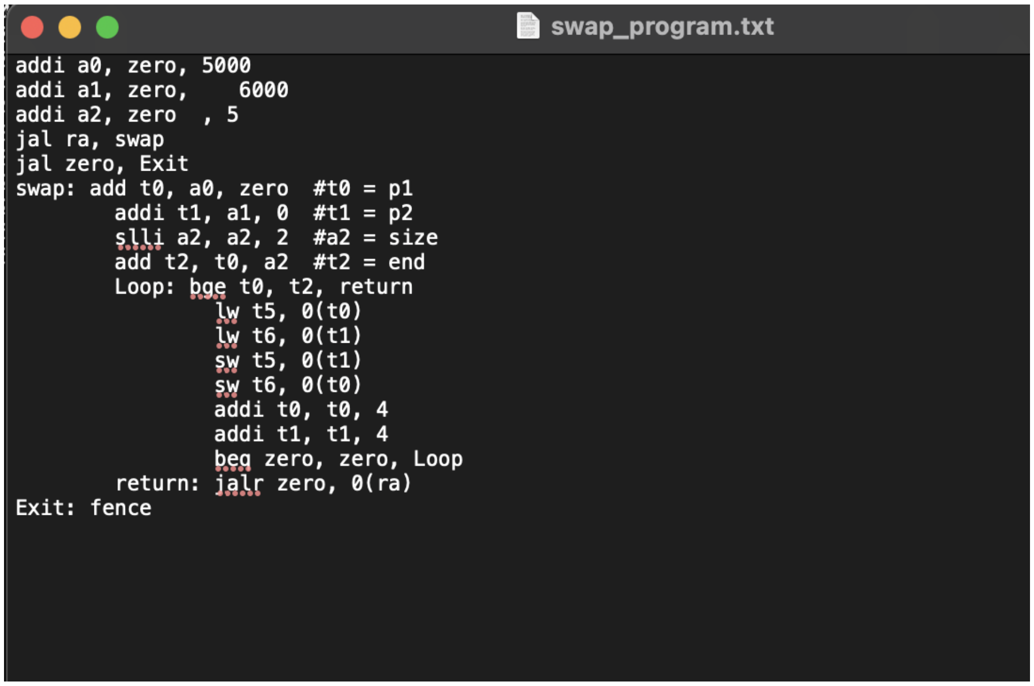
"swap_data.txt":

Where 10000 is the initial program counter



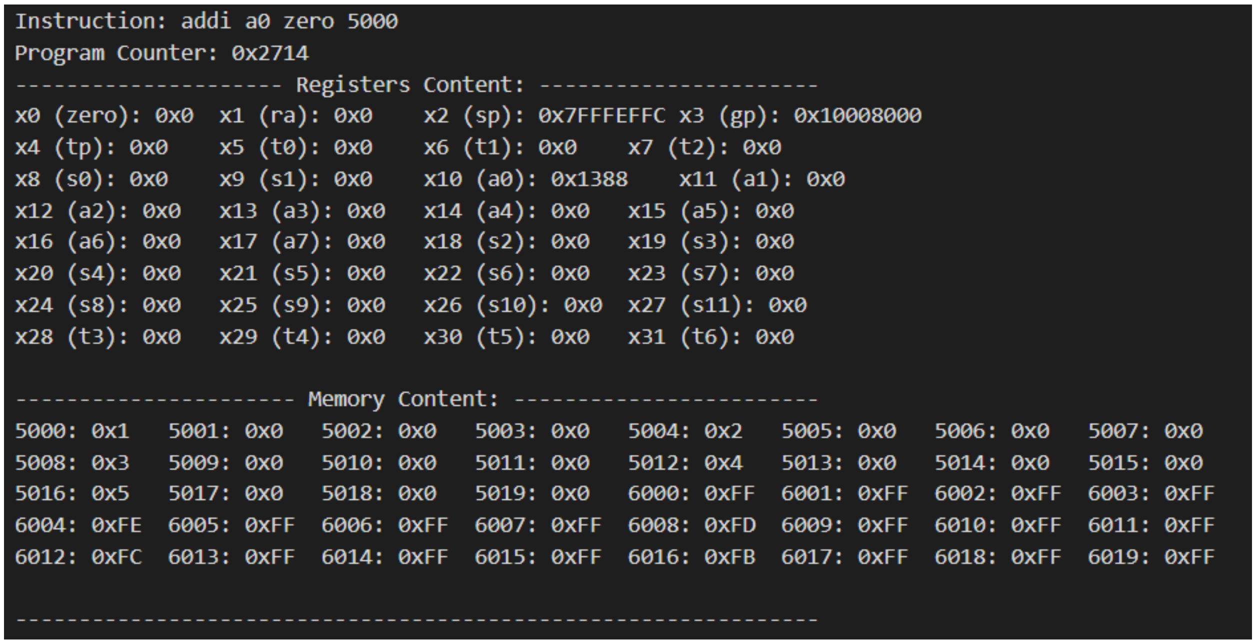
```
10000
5000 1
6000 -1
5004 2
6004 -2
5008 3
6008 -3
5012 4
6012 -4
5016 5
6016 -5
```

"swap_program.txt":



```
addi a0, zero, 5000
addi a1, zero,    6000
addi a2, zero   , 5
jal ra, swap
jal zero, Exit
swap: add t0, a0, zero  #t0 = p1
        addi t1, a1, 0  #t1 = p2
        slli a2, a2, 2  #a2 = size
        add t2, t0, a2  #t2 = end
        Loop: bge t0, t2, return
                lw t5, 0(t0)
                lw t6, 0(t1)
                sw t5, 0(t1)
                sw t6, 0(t0)
                addi t0, t0, 4
                addi t1, t1, 4
                beq zero, zero, Loop
        return: jalr zero, 0(ra)
Exit: fence
```

The user will then be asked whether they want the value in the registers in decimal, hexadecimal or binary. Subsequently, the program will begin processing and executing each of the instructions in the code file. After each instruction is executed, the contents of the registers, the memory and the program counter are outputted, allowing the user to see each change taking place step by step, with the values displayed in either decimal, hexadecimal, or binary depending on the user’s choice in the beginning of the program (Which is hexadecimal in this example). The output will then be saved and placed in a a file named swap_output.txt. Below is a screenshot of the contents of this file:



```
Instruction: addi a0 zero 5000
Program Counter: 0x2714
--------------------- Registers Content: ----------------------
x0 (zero): 0x0  x1 (ra): 0x0    x2 (sp): 0x7FFFEFFC x3 (gp): 0x10008000
x4 (tp): 0x0    x5 (t0): 0x0    x6 (t1): 0x0    x7 (t2): 0x0
x8 (s0): 0x0    x9 (s1): 0x0    x10 (a0): 0x1388    x11 (a1): 0x0
x12 (a2): 0x0   x13 (a3): 0x0   x14 (a4): 0x0   x15 (a5): 0x0
x16 (a6): 0x0   x17 (a7): 0x0   x18 (s2): 0x0   x19 (s3): 0x0
x20 (s4): 0x0   x21 (s5): 0x0   x22 (s6): 0x0   x23 (s7): 0x0
x24 (s8): 0x0   x25 (s9): 0x0   x26 (s10): 0x0  x27 (s11): 0x0
x28 (t3): 0x0   x29 (t4): 0x0   x30 (t5): 0x0   x31 (t6): 0x0

---------------------- Memory Content: -----------------------
5000: 0x1   5001: 0x0   5002: 0x0   5003: 0x0   5004: 0x2   5005: 0x0   5006: 0x0   5007: 0x0
5008: 0x3   5009: 0x0   5010: 0x0   5011: 0x0   5012: 0x4   5013: 0x0   5014: 0x0   5015: 0x0
5016: 0x5   5017: 0x0   5018: 0x0   5019: 0x0   6000: 0xFF  6001: 0xFF  6002: 0xFF  6003: 0xFF
6004: 0xFE  6005: 0xFF  6006: 0xFF  6007: 0xFF  6008: 0xFD  6009: 0xFF  6010: 0xFF  6011: 0xFF
6012: 0xFC  6013: 0xFF  6014: 0xFF  6015: 0xFF  6016: 0xFB  6017: 0xFF  6018: 0xFF  6019: 0xFF

---------------------------------------------------------------
```

List of Programs:

(Please note that each of these programs has their associated data found in the zip file)

1. **Sort Program**

   Sorts numbers using the insertion sort algorithm.

   File name: sort_program.txt

   Data file name: sort_data.txt

2. **String Length Program**

   Takes a string and finds its length.

   File name: strlen_program.txt

   Data file name: strlen_data.txt

3. **Fibonacci Sequence Program**

   Executes the Fibonacci sequence.

   File name: fibonacci_program.txt

   Data file name: fibonacci_data.txt

4. **Get Max Program**

   Finds the greatest number in an array.

   File name: max_program.txt

   Data file name: max_data.txt

5. **Sum Difference Program**

   Takes two arrays and finds the sum and difference between them.

   File name: sum_difference_program.txt

   Data file name: sum_difference_data.txt

6. **Swap Program**

Swaps the contents of two arrays with one another.

File name: swap_program.txt

Data file name: swap_data