Group 2

Mohanad Abouserie - 900213567

Amr Abdelbaky - 900204834

Mahmoud Nour - 900202978

A brief description of the implementation:

We have a struct named instruction, which stores the following information associated with the instruction:

-Name of instruction

-Source registers and destination registers, if present

-Immediate value, if present

-Label name, if present

We have a struct named dataval, which stores a byte as a string.

Note that the struct type is used instead of the string type to facilitate for initializing the value of the byte to be zero.

We implemented the instruction memory, data memory, and stack using the map data structure. This allows us to “record the contents of relevant memory locations only ” as required.

The instruction memory maps the address to the instruction..

The data memory maps the memory address to its corresponding value.

The stack memory maps the stack address to its corresponding value.

The registers are implemented as an int array of size 32 named registers. The stack pointer register value is initialized with 12000, and the rest of registers are initialized with 0.

The program begins by the reading_initial function, which reads the data required by the program entered by the user and stores it in the data memory.

The program then executes the read_instructions_from_file function which takes as an input the file and loops over the code, ignoring the empty lines and comments, and converts each valid instruction into an instruction object and stores the instruction in the instruction memory. The conversion is done using the convert_riscv_instruction function. Note that labels have the same memory address as the instruction directly after it and they are stored in a separate map.

After storing the instructions and data in their memories, the program will use the PC counter and will loop over the instruction memory, executing each instruction using the execute function. The execute function will manipulate the values in the registers and memory as required by the instruction. Execution will begin at the PC initialized by the user. Note that an execution of an instruction can change the PC counter. Our program takes this into consideration. The execute takes as a parameter the instruction at the current PC, thus changing the PC causes the execute function to continue execution normally at the new PC as required.

We use the rars_format function to print the program counter value, the register file contents, and the memory content as required. The user can specify at the beginning of the execution wether

they want to displacy the program counter value, the register file contents, and the memory content after each instruction execution or to print the program counter value, the register file contents, and the memory content at the end of the program only.

Bonus Features Implemented:

We Implemented the following 2 bonus features:

- Including a larger set of test programs (at least 6 meaningful programs) and their equivalent C++ programs.
- Outputting all values in decimal, binary, and hexadecimal (instead of just decimal which is assumed to be the default).

Design decisions and/or assumptions we made:

- We have assumed that for a Risc-v code to execute correctly the instructions need to have a certain format, where after each comma separator we include a whitespace.
- The program accepts the two formats for the register names, namely the registers beginning with x or the other pseudo format, thanks to our replace_reg function.

Any known bugs or issues in the simulator:

To our knowledge, the simulator has no bugs and it runs successfully on the 7 programs included. We implemented intense validation to ensure we have covered as much errors as possible. We have the minor short-back for the requirement of the format, which is mentioned in the assumptions section above. Also, if you encounter any “Error in line #” then you should count the lines of instructions without countring empty lines, comment lines or label lines.

User guide for running the simulator:

https://drive.google.com/file/d/1YaHrOFrCD3SDloEs-fPLFSNnFGAP8QRt/view?usp=sharing