Simulator inputs:

Assembly program:

- We will take a text file as the source code file
- User should specify the program starting address

Program data:

- Formatted text file for value and memory address input of any data required by the program initially

Program Logic:

1-22

23-42

ECALL, EBREAK, PAUSE, FENCE, and FENCE.TSO instructions. Instead, your implementation should interpret any of these 5 instructions as a halting instruction that ends the execution of any program (by preventing the program counter from being updated anymore)

Simulator outputs:

The simulator should start by reading the inputs provided by the user and then it should simulate the input program's execution by keeping track of the program counter value, the register file contents, and the memory contents. The program should repeatedly output all these values (after each instruction's execution) until the program ends.

1.

To keep track of program counter value, the register file contents, and the memory contents you need to initialize them. The program counter should be initialized to the program's starting address as specified by the user. All registers should be initialized to zeros. The memory should also be assumed to be empty except for the locations containing the instructions (which can be ignored unless you intend to implement the assembling to machine code bonus described below) and the locations containing the data values provided by the user.

3.

Register 0 always contains the value 0. You will need to make sure that no instructions can modify it.

Bonus:

1. Building the application as a GUI application (either desktop, web-based, mobile app).

3. Outputting all values in decimal, binary, and hexadecimal (instead of just decimal which is assumed to be the default)

6. Add support for integer multiplication and division to effectively support the full RV32IM instruction set. The integer multiplication and division specifications can also be found in the document above.

7. Including a larger set of test programs (at least 6 meaningful programs) and their equivalent C programs.

Project Report: The report should include:

1. Team member names, student IDs, and emails.
2. A brief description of your implementation including any bonus features included.
3. Any design decisions and/or assumptions you made.
4. Any known bugs or issues in your simulator.
5. A user guide showing how to compile and run your simulator including a full simulation example step-by-step with screenshots.
6. A list of programs (and associated data if any) you simulated. You should at least provide 3 programs. The programs must cover all instructions supported and one of them at least must have a loop.
7. Optionally, you can include a section about your experience working on this project. This section will NOT affect your grade in any way.

General Guidelines

- Every group member must log all her/his activities in a journal (a text file). A journal entry may look like the following:

```
Oct 30, 1:55PM: added support for sw, sh, sb instructions. Fixed issue in
        program counter update in case of conditional branching.
```

- Only one member of each group should submit on Canvas. The submission should consist of a single compressed folder (**zip** or **rar**) which must include the following:
  - A journal folder that contains the journal file of each team member.
  - A source code folder that contains the code you wrote using your chosen programming language.
  - A test folder that contains all input files (assembly and data files) used for testing.
  - A **PDF** report that contains the information described above.