pr5 - Simulated RAM and Disassembler

1 Lab Assignment 2 (Graded)

In this lab, we will simulate the RAM and implement a mechanism to load the input ELF into the simulated RAM. The ELF is loaded into the memory by first generating the "binary" file (already done by the Makefile `*.r5ob`). A binary file goes through the ELF file, and places the sections at appropriate addresses. The lowest address is placed at offset 0 (of the `*.r5ob`), and the highest address is placed just before the EOF.

1. Modify the linker script programs/custom/test.ld to place .data section at the address `0x80008000`. The binary file will have 0s at locations between the lowest and highest addresses where there is no valid data/instruction.

2. Implement the following tasks in Python. You are strongly encouraged to organize the code into appropriate modules. All the code should you develop should be in the folder `pr5/src`.

   - Implement a simulated RAM. The simulated RAM should support `read()` and `write()` operations. The `read()` operation should take in an address and return the byte stored at that address. `write()` should take an address and a data (byte) as input and store inside the simulated RAM.
   - Load the contents of the binary file into the simulated RAM. Use a driver code for this if necessary. The driver code should be in `pr5/src/main.py`.
   - Disassemble the instructions present in the .text section (that was loaded into the memory starting at 0x80000000). The code for disassembler should be in `pr5/src/disassembler.py`, and should read the instructions from the simulated RAM (and not from the ELF or the binary file).
   - Print the contents of the .data section that was loaded into the simulated RAM (starting at `0x80008000`).