CSCE 2303 – Computer Organization and Assembly Language Programming
Summer 2022

Project 1: RV32IC Disassembler

1. Background

A disassembler is a computer program that translates machine language into assembly language—the inverse operation to that of an assembler.

The objective of this project is to create a disassembler that supports RV32IC instructions.

2. Requirements

You are required to develop a command line application that gets a file that has the content of the text section of an RV32IC application and outputs the disassembly of the program. You may use any programming language of your choice (C/C++ is preferred and a C++ skeleton is given). Your program must:

- Read RV32IC machine code file. The file is just a binary file that contains the machine code of a program instructions. The text segment is assumed to be at location 0x00000000 in the main memory.
- Decode each machine code word, then translate it into a true RV32IC instruction.
- Be invoked from the command line using the syntax:

  `rvcdiss <machine_code_file_name>`

  Where:

  `<machine_code_file_name>` is the file name of the input machine code
- If you don't know how to pass parameters using the command line to a C/C++ program, check [1].
- The disassembler should be able to decode all RV32IC instructions [2].
- Your output must use RISC-V ABI register names [3].

3. Guidelines

- Work in a group of 3 students
- The skeleton is in C++. You may use other programming language for your implementation. If you are planning to do so, please contact Dr. Shalan first before get his approval (you have to have a strong reason for that).
- You must use github or similar free online services to host your code and to collaborate with your partner.
- Grading Criteria:
  - 50%: Correctness of the output based on hidden test files.
  - 25%: Comprehensive test cases covering all the instructions.
  - 10%: The project readme file which should contains: the implementation details, limitations, known issues, contributions of each team member.
  - 10%: Code organization and comments.
  - 5%: Consistent development using github (or similar systems).
  - +10% (Bonus): Use labels for branch and jump instructions.
- Deadline: Monday July 5th, Class meeting time. More details will be posted.

4. References

[1] Command line arguments in C++ using argc and argv, https://www.cprogramming.com/tutorial/lesson14.html
[2] The RISC-V Compressed Instruction Set Manual, https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf
[3] RISC-V Calling Convention, https://riscv.org/wp-content/uploads/2015/01/riscv-calling.pdf