CSCE 2303 – Computer Organization and Assembly Language Programming
Summer 2019

Project 1: RV32IC Simulator

1. Background

An ISA simulator (ISS) is a program that accepts machine code of compiled/assembled application and mimics what a real CPU hardware would do to execute the instructions that make the application. The simulator shows the effect of simulating the instructions execution as memory location(s) and register(s) changes.

RISC-V (pronounced "risk-five") is an open-source instruction set architecture (ISA) based on established reduced instruction set computing (RISC) principles. The project was originated in 2010 by researchers in the Computer Science Department at University of California, Berkeley, but many contributors are volunteers and industry workers otherwise unaffiliated with the university.

2. Requirements

You are required to develop ISS for RISC-V RV32IC Base Integer Instruction Set with support for instruction compression. 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 first 4 bytes (32-bit word) in the file belong to the first instruction; the 2nd 4 bytes belong to the 2nd instruction, etc. The first instruction is assumed to be at location 0x00000000 in the main memory.
- Decode each machine code word, then translate it into a true RV32IC instruction.
- Execute the decoded instruction. The execution involves: modifying a register, modifying a memory location or performing an I/O operation (ECALL).
- Print out the decoded instruction on the screen.
- Be invoked from the command line using the syntax:

  `rvsim <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 https://www.cprogramming.com/tutorial/lesson14.html
- The disassembler/simulator should be able to decode/execute all RV32i instructions except EBREAK, FENCE and CSR instructions.
- Finally, the simulator should be able to execute the following ECALL services: 1, 4, 5, 8, and 10 as defined by https://github.com/TheThirdOne/rars/wiki/Environment-Calls

3. Guidelines

- Work in a group of 2 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).
- Report submission and project demo will be done on July 7th 9:00-11:00AM.
  - You have to upload the project files, including the report, to BB.
  - You must bring a hardcopy of your report to present in the interview (.
  - The report should outline your design and how to use the simulator. Also, it should outline the challenges you faced as well as any limitation your simulator has.