CSCE 2303 – Computer Organization and Assembly Language Programming
Fall 2016

Project 1: RISC-V Simulator and Disassembler

1. Background

An *assembler* is a program that translates programs written in assembly language into machine code. Machine code is essentially the binary equivalent of the assembly language. Translating from assembly language to machine code is straightforward process using the instruction formats specifications.

A *disassembler* works in reverse. It takes machine code and produces the equivalent assembly language program. At times, the machine code may not have all the information that the assembly language program contains. In particular, comments are typically removed, and labels are also removed.

An ISA simulator is a program that accepts machine code of compiled/assembled applications 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 Division 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 a disassembler/simulator for RISC-V RV32I Base Integer Instruction Set using a programming language of your choice (C/C++ is preferred and a C++ skeleton is given). Your program must:

- Read RV32i machine code file. The file is just a binary file that contains the machine code of a program instructions. The first 4 bytes 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 0x00 in the main memory.
- Decode each machine code word, then translate it into a true RV32i instruction.
- Execute the decoded instruction. The execution involves: modifying a register, modifying a memory location or performing an I/O operation (SYSCALL).
- Print out the disassembled code on the screen.
- Be invoked from the command line using the syntax:

  ```
  mipsim <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, read this.
- The disassembler/simulator should be able to decode/execute all RV32i instructions except SBREAK as well as timers and counters instructions.
- Finally, the simulator should be able to execute the following MIPS SYSCALL services (in risc-v syscall is called “scall”):
  - Print an integer (1),
  - Print a string (4),
  - Read an integer (5),
  - Terminate execution (10).

3. Optional Requirements (50% bonus)

Create an online simulator similar to the one outlined at:
https://www.mschweighauser.com/make-your-own-assembler-simulator-in-javascript-part1/
You have to consult with Dr. Shalan if you are planning to do this bonus. You cannot do it unless you get his permission.

4. 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 is will be done during the lecture time of November 13th.
  - You have to upload the project files, including the report, to Google Drive and share it with Dr. Shalan and Ahmed Baracat (the graduate TA).
  - 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.