Mips 1 Simulator Documentation

Dylan Auty

November 19th, 2013

1 About this library

This library is intended to act as a complete simulator for the MIPS 1 processor architecture. The only exception to this is the SYSCALL command, which cannot be implemented using the present layout.

This library allows the user to write a binary file ending ".bin" using a hex editor, then a driver to go with it which will set up arguments and place the program in memory, before stepping it to completion and then exiting.

2 Implemented commands

The list of implemented commands can be found in full in the file "implemented_functions.csv". Every command from the MIPS 1 instruction set, save for the SYSCALL command, has been implemented

3 Usage

In order to use the library, the commands must be read into memory, then the function "mips_step(state)" must be called - this will return a 0 when execution is running normally, and a 1 when an error (such as an overflow, or an illegal instruction) has occurred.

The files involved in each driver are: tn_driver.c, tn_input-mips.bin, and driver_helper.c. If the file tn_driver.c file is included in the makefile, then the entire library can be linked and tested in one go by cding into the ./inputs directory and using the "make all" command. Each of the tests should only return any output if there is an error; otherwise they will execute without outputting. The exceptions to these are the t6_driver and t7_driver files, which were written by myself and will print out the contents of the registers after each instruction. However, a similar error message will be thrown if there is any error in execution, giving the arguments, the simulation's returned value, and the correct value.

4 Testing

Testing was done with the 5 driver files included with the project skeleton, and with the use of my own driver files, t6_driver and t7_driver.

T6_driver is the most complete, and its operation is documented in the T6_LOG.xlsx file included in the .zip file. In this file, BOLD outputs indicate a desired output. Only three registers are used - $4, $5, and $6. Output is always written to $6, and the program will indicate a pass or a fail for each test it runs. In addition, it will print out the contents of the registers immediately after executing each instruction, for further debugging use.