CDA 4150 Project #1[1]

Due: 10/18/05, 11:59pm

School of EECS, University of Central Florida, Orlando, FL 32816.

1. Objective

You will be given a non-pipelined Verilog model of a MIPS R3000 microprocessor with some bugs and a few instructions missing. You will be asked to fix the problems and to add the missing instructions. Remember to submit your project once you are finished.

2. Model Description

The Verilog model is able to execute real MIPS code generated by our MIPS gcc cross-compiler, including emulation of all syscall (operating system call) instructions. The Verilog code is broken down into 10 files with the top-level file called mips.v. The mips.v file handles the interface between the processor and the memory system (load and stores) and also instantiates cpu.v. cpu.v in turn instantiates the various processor components which roughly follow the 5-stages of the execution of a MIPS instruction: Instruction Fetch (IF), Register Decode (RD), Execution (EX), Memory (MEM), and Writeback (WB).

3. Lab Setup

- Go to your home directory (type cd)
- Go to your CDA 4150 directory (type cd cda4150)
- cp ~hgao/cda4150/lab1.tgz .
- Uncompress the file: tar -xzf lab1.tgz
- All the files needed for Project 1 will be in the lab1 directory or sub-directories. Your work should be done in that directory.

4. Running Your First MIPS Program

- In the directory ~/cda4150/lab1/ type: make

This will compile and cross-assemble some C files inside the test directory. Once you are asked for a file name, enter test/hello. The vbs simulator is somewhat slow (it is simulating in detail!), so be patient when running larger C programs. In the directory ~/cda4150/lab1/test/ you will find a file called Makefile. Take a look at this file using a text editor; when you write a new C program of your own, this is the file that must be modified to make make work.

- The vbs simulator for CDA 4150 has a very useful command called $disasm. The $disasm command takes as input a 32-bit variable which represents and instruction and displays on the standard output the corresponding assembly code. - Go to the always block around line 79 in the mips.v file and uncomment the $disasm statement. Execute the test/hello.c program again as described above. - Another useful debugging command is the $display statement (in the same always block you will find a $display statement; try uncommenting it and see what happens). - The initial block in cpu.v is a good place to put $monitor statements for debugging. Uncomment the $monitor on line 100 of cpu.v and see what happens.

5. Running the Verilog Checker

There is a useful script called vcheck that checks the syntax of your Verilog code. For instance, vcheck checks whether all the RHS variables used inside of an always combinational block appear in the sensitivity list. vcheck also checks whether you used <= in always blocks (and checks for the 'TICK delay). Run vcheck on your design, and fix the problems that it reports. Use this script frequently on your code and you will avoid many common Verilog bugs! Your submitted design should be vcheck "clean" . You can run vcheck manually (e.g. vcheck *.v or vcheck file.v) or you can check your whole design with make check.

[1] The project was designed by Prof. Mark Heinrich for CDA4150 Fall 2004.

6. Fixing the Broken Model

Inside the test directory there is a file called problem.c.

- Check the contents of this file. Before executing the program in the MIPS model, make sure you understand what is supposed to happen, assuming a perfect execution of the C code. Also note that when you DO fix problems in the code and problem.c runs correctly, this does not necessarily mean that you have fixed all the bugs in your processor model! Please write your own tests and examine the code carefully for problems.
- Fix the bugs. You will not be given hints on how to do this (that is how things happen in real life!).
- You can get an assembly language listing of the test files (e.g. problem, or hello, etc.) by going to the test directory and disassembling the corresponding .ld binary file. To disassemble you must use the cross-disassembler: mips-sgi-irix5-objdump –D filename.ld | less. I often add these lines to my .cshrc and then source .cshrc again:

  alias dis 'mips-sgi-irix5-objdump -D'

  alias less 'more'

  And then the disassemble command can just be: dis file.ld | more.

7. Adding the SH, LH and LHU Instructions

Your MIPS model is currently missing the sh (store signed half word), the lh (load signed half word), and the lhu (load unsigned half word) instructions (a half word is 16 bits). Implement those instructions, and use the test programs in the test directory to test your instructions. The names of the test programs are loadh.c, loadhu.c, and storeh.c. The opcodes for the three instructions can be found in the file mips.h.

8. Submitting Your Results

Submit your files electronically once you have finished. This procedure will be used in all the labs.

IMPORTANT: To correctly process your group submission, you must have in your lab1 directory a file called README with the first four lines as follows: (suppose your login IDs are turing and maxwell)

GROUP: turing maxwell

Full name of Alan Turing

Full name of James Clerk Maxwell

LAB XX (where XX is the number of the lab)

If you have special instructions for the graders, or wish to denote the use of your Freebie, you must inform us in the README file (after the header).

To submit your files, make sure you are in your lab1 directory and type:

submit4150 lab1

Only one person in each group should submit. You may submit several times if you wish, but the newest submit will completely overwrite the previous one. The timestamp on the submission will determine the exact time of submission and determine whether it is on time or late. Late submissions are ignored in the grading process unless it is your Freebie. If for some reason your design is not completely functional, describe what does work, what the known problems are, and what you think is causing them. If your design is working properly, you do not need to put anything in the README file other than the header above.

NOTE: The submission script is not finished now. Please check course website for the announcement about it when it works (should be earlier than Oct 8th).