10320 CS410001 - Computer Architecture 2015

Project #1

1. Project Objective
   a. Implement a single-cycle, functional processor simulator according to the reduced MIPS R3000 ISA specified in *Appendix A*, "*Datasheet for the Reduced MIPS R3000 ISA*."
   b. Design your own test case to test the functionality of your simulator.

2. About the Simulation
   a. The execution of the single-cycle processor simulator should **terminate** after **executing the "halt"** instruction.
   b. Both the instruction memory and data memory are in **big-endian** format.
   c. **Register 0** is a **hard-wired constant 0**; any attempt to write to register 0 takes no effect.
   d. Assume that both the instruction memory and data memory are of 1K bytes size.
   e. You should implement an **error handler** to deal with erroneous instruction executions.
   For more details please refer to *Appendix D*, "*Error Detection Samples*."
   f. The simulator should get input files and generate output files at the same directory as that of the executable. Note that the executable file should be named **single_cycle** and takes no command-line arguments.

3. Input Format
   a. For each test case, **all registers, except PC and $sp, are initialized to 0's**; other initializations are specified in the following two files.
      i. **iimage.bin**: The instruction image (big-endian format, encoded in binary). The first four bytes indicate the initial value of PC, **i.e. the starting address to load the instruction image**. The next four bytes specify the number of words to be loaded into instruction memory (I-memory). The remaining are the program text to be loaded into I-memory. The contents of all other addresses not covered by the image are assumed to have been initialized to 0's.
      ii. **dimage.bin**: The data image (big-endian format, encoded in binary). The first four bytes show the initial value of $sp. The next four bytes specify the number of words to be loaded into data memory (D-memory). The remaining are the data to be loaded into D-memory, starting from address 0. The contents of all other addresses not covered by the image are assumed to have been initialized to 0's.
   b. For details about input format, please refer to *Appendix B*, "*Input Samples*."

4. Output Requirement
   For each test case, please respectively output to the following two files:
   a. **snapshot.rpt :** contains all register values at each cycle.

b. **error_dump.rpt :** contains error messages.

For more details please refer to *Appendix C-1*, *“Output Samples for Project 1”* and *Appendix D*, *“Error Detection Samples*”.

5. Test Case Design
   a. Your test case should be of the same format as described in item 3 (Input Format) and *Appendix B*. Additionally, both ”**iimage.bin**” and ”**dimage.bin**” should be placed under a folder named **testcase**.
   b. Note that the initial value of PC is also the starting point of the execution of your program specified in **iimage.bin**.
   c. Your test case should run **no more 500,000 cycles**.

6. Submission
   a. Submit a folder named ***studentID_01*** under */home/archi/**studentID***. Inside the folder, it provides the following **two folders** and corresponding files required:
      i. ***simulator/*** : contains your「**Makefile**」,「**README**」, and source code files.
         - Your **Makefile** should support the following two functions:
           make – to build your simulation environment
           make clean – to erase from the build tree the files built by make all.
      ii. ***testcase/*** : contains your test case files.
   b. Finally, compress the folder ***studentID_01*** as ***studentID_01.tar.gz***, and upload ***studentID_01.tar.gz*** and ***studentID_report.pdf*** to the iLMS system.

- **Note**: Each project has *two* due dates. The second due date is one week after the first one.

7. Evaluation
   a. Correctness of simulator: 70%
      i. First submission : 30%
         TA’s open test cases (12%)
         TA’s hidden test cases (18%)
      ii. Second submission: 40%
         TA’s open+hidden test cases (10%)
         Students’ valid test cases (30%)
      - **Note 1**: We will release TA’s open test cases before first due date, and you may use them to verify your simulator.
      - **Note 2**: After first due date, we will release TA’s hidden test cases.

   b. Report: 10% (evaluated at the second submission)

i. Report is limited to at most 10 pages.

ii. Your report can be either in Chinese or English, or mixed. The report file should be named ***studentID_report.pdf***.

iii. Grading principle:

1. Simulator design (2.5%): Discuss if any special data structures, design patterns, code structure, and general execution flow, etc.
2. Simulator elaboration (2.5%): Clearly explain your design with aids of tables, figures, etc.
3. Test case design (2.5%): How you implement your test case in C code and assembly code? What corner cases are you targeting at? We will also look for the creative ideas used in your test case.
4. Test case elaboration (2.5%): Comments to both C code and assembly code of your test case; the mapping between variables and register/memory locations and that between codes and labels; etc.

c. Test case: 20% * (1 – $[1.5]^{-n}$), n: number of defeated groups

i. First submission: 0%

TA won't use your test case for grading, but will generate reference output files (**snapshot.rpt, error_dump.rpt)** for you by using TA's golden simulator. You may use the reference files to correct your test case and simulator for second submission.

ii. Second submission: 20% * (1 – $[1.5]^{-n}$) , n: number of defeated groups

- **Note 1**: A test case is valid if the output generated by TA's simulator is the same as that from your own simulator.
- **Note 2**: You get zero points if your test case is invalid.

- Note that we use **nthucad workstation** as our official testing environment. Furthermore, we will evaluate your project using scripts. Please make sure that your project can be executed by the script provided by TA's. **If it cannot run through the official script, you will get zero points even if your program or result is correct.**

8. Etiquette

a. Do not plagiarize others' work, or you will fail this course.

b. No acceptance of late homework.

c. Please frequently check the class website announcements for possible updates.