Computer Architecture 2012/2013
Assignment 2b

Date: September 25th, 2012
Deadline: October 19th 2012, 23:59

Contents

1 Overview **1**

2 Instructions **2**

3 Prerequisites **2**

4 ALU instructions **2**
- 4.1 Decode + Execute for R instructions 3
- 4.2 Decode for I and J 3
- 4.3 Execute for ALU instructions with format I and J 3
- 4.4 Testing 3

5 Summary of submission contents **3**

6 Grading **4**

1 Overview

Reminder: The purpose of assignment series 2 is to implement the MIPS ISA in MGSim, so as to be able to run real MIPS code compiled using GCC and the GNU Binary utilities (assembler+linker). Assignment series 2 will be spread over multiple weeks, and split into individual assignments 2a, 2b, etc.

The goal of assignment 2b is to:

- extend the work started in assignment 2a
- add more logic to the pipeline decode stage;
- add more logic to the pipeline execute stage;
- test the results with simple programs.

2 Instructions

- For this assignment, you can work in groups of 2.
- Read this entire document before you start.
- Your must submit a compressed tarball[1], named after your last name and student ID, containing:
  - the files that you have produced during the assigment.
  - a file `report.rst` containing your write ups to open questions using reStructured Text. This must also contain your full name and student ID. Ensure that `report.rst` is valid by using `rst2html`.
- Your submission must be sent by e-mail before the deadline, at the e-mail address given by the assistants. Do not send your submission to the mailing list!

3 Prerequisites

You will need the following:

- the Alpha an MIPS cross-utilities and cross-compilers, and the MGSim simulator compiled for Alpha, as per assignment 1/2a.
- the MGSim source code and development environment, as per assignment 2a.
- a copy of the following files, which should accompany this document:

File	**Description**
mipsel-cc	Script to compile/assemble/link MIPS code.
minisim.ini	Configuration file for MGsim.
arith.c	Example micro-program.

Note

The files `alpha-cc`, `mipsel-cc` and `minisim.ini` are different from assignment 1.

4 ALU instructions

The goal of this task is to implement the extra ALU instructions that you have identified in assignment 1, question 7 in MGSim. You must also document your changes to MGSim in textual form in the separate report.

The following progression is strongly suggested.

[1]A compressed tarball is created with `tar -czf xxxx.tgz ....`

4.1 Decode + Execute for R instructions

Start with register-register instructions (MIPS instruction format "R"), to extend the support for `add` and `sub` from assignment 2a.

Focus on ALU instructions (arithmetic, logic, bitwise shift): do not implement branches (`jr`) just yet.

4.2 Decode for I and J

Add a new *decode* logic to identify the instruction formats "I" and "J" (register-immediate and jump).

Hint

For this you will also need to add new buffers in the decode-read latch.

Note

You do not need to add the new buffers explicitly to the read-execute latch too, because all buffers from the decode-read latch are already duplicated (and automatically propagated) to the read-execute latch.

4.3 Execute for ALU instructions with format I and J

Implement the *execute* logic for *ALU* instructions using the format "I". Again, focus on ALU instructions (`addi`, `ori`, `lui`, `slti`...), and avoid branches and memory operations.

Note

Try to focus on the ALU instructions that your test programs actually use. There are a couple MIPS ALU instructions that are more "difficult" to implement, namely `mult`, `div`, `mflo`, `mfhi`, `mfcZ`, `mtcZ` but these should not be used by your test programs.

4.4 Testing

At each step above, make micro-programs to test your progress. Use separate micro-programs for different categories of instruction.

Think of writing a shell script of Makefile to automate the execution of many test programs after one another. Suggestion: document yourself about the concepts of "unit testing" and "regression testing".

5 Summary of submission contents

Your final submission archive should contain the following files:

- `report.rst` (your report with explanations);
- `mgsim.patch` (your patch file generated with `git diff origin/mgsim`);
- any micro-programs and testing scripts you have created to support your work.

6 Grading

Assignment 2b and 2c will be graded together. You will be evaluated as follows:

- whether you have implemented decode + execute for all R instructions (1pt);
- whether you have implemented decode for R, I, J formats and defined the appropriate buffers, and documented the result (1pt);
- whether your decode logic properly uses a tree of conditions instead of a linear sequence of conditions (1pt);
- whether you have implemented execute for "I" ALU instructions (1pt);
- whether your code is properly structured, indented, documented, either as comments or in the separate report (2pt, will be shared with assignment 2c);
- how you can prove or illustrate that all your instructions work using a test suite or demonstration programs (2pt, will be shared with assignment 2c).