Assignment 3

Code Due: 11:59 pm, Tuesday, March 1st, 2016.
HW Problems Due: Beginning of Class Tuesday, March 1st, 2016

1 Purpose:

Understand how to create and control multi-cycle implementations of a datapath. This is achieved by adding functionality to a multi-cycle datapath that implements a small subset of the MIPS instructions.

2 Installing the Tools

To complete this assignment, you can access ModelSim installed on the Klaus Windows laboratories. This is where the code will be graded

However, if you wish to install it on your own machine, you should download and install **ModelSim PE Student Edition per the instructions below**. ModelSim runs under Windows. If you have a Mac you can use a virtual machine (e.g., Parallels) and Windows. It is highly recommended that you go through the ModelSim tutorial that is available from the Help button on the toolbar.

The following steps describe how you may install ModelSim on your personal machine.

1. Download and install the free Student Edition from the Mentor Graphics Website. Note that this is free but you will need a license that you must request at the end of the installation process. Requesting the license involves the installer taking you to a web page, where you will fill out some basic information (incl. your email) and request a license from ModelSim. They will send you an email with a license file. This license file will have been customized for the ModelSim student edition installation on your machine. The email will instruct you on where to save the license file. This step is of course not necessary if you use the version in Klaus.
2. After you receive your license file via email follow the instructions. You should now be ready to go through the tutorial (Help → PDF Documentation → Tutorial).
3. Download the multi-cycle datapath code from the class website. It is posted under lecture notes and is associated with the module in single and multi-cycle datapaths. Also download the associated documentation posted along with the code on the class website. This document contains detailed instructions on how to get the multi-cycle datapath (MS_SPIM) running in ModelSim.
4. There is also a set of VHDL tutorial slides made available on the class website. The resources page will also link to some useful online sites for VHDL.

Now you can follow the execution process for the Multi-Cycle datapath (MS_SPIM) as provided in the documentation mentioned in Step 3. This should be a mechanical process. Even if it is not clear, go through the steps and ensure (with myself or the TA) that you can successfully complete all steps. Through the warm up in Section 3 and the assignment in Section 4 you will become more familiar with ModelSim and the model. When you use ModelSim, ensure that you add to the trace all signals in the top level of the model (MIPS.vhd). When you follow the ModelSim tutorial you will find a description of how to do this. If you have a problem seeing all of the signals, ask (in person or via email) the TA or me. Do not spend too much time on tool

specific issues. I would rather you spend most of the time on the assignment.

You may use other VHDL tools rather than ModelSim. If you do, please ensure that your submission is in line with the submission requirements for grading. You are responsible for ensuring that your simulation will execute correctly under ModelSim. If you stay with standard VHDL this will not be a problem.

3 Assignment Part I: Understanding the Model (0 points)

Now you can follow the execution process for the Multi-Cycle datapath (MS_SPIM) as provided in its documentation. This should be a mechanical process. Even if it is not clear, go through the steps and ensure (with myself or the TA) that you can successfully complete all steps. Do not spend too much time on tool specific issues. I would rather you spend most of the time on the assignment.

You may use other VHDL tools other than ModelSim if you wish. If you do, please ensure that your submission is in keeping with the detail required for the TA to grade. *You are responsible for ensuring that your simulation will execute correctly under ModelSim*. If you use standard VHDL and no simulator specific constructs this should not be a problem.

While this model has been used before and should be quite stable, please do report any discrepancies immediately.

1. To familiarize yourself with the model, determine the answer to the following. You may ask the TA and discuss this with your classmates and use the piazza forum that will be monitored by the TA. There are no submission requirements for this part. It is to your benefit to complete this and understand the answers before proceeding to Part II.
2. How large is the microcode ROM?
3. What microinstruction address stores the memory write state?
4. How large is instruction memory? What is the contents of memory word at address 0x8.
5. In which VHDL modules are the following components implemented?
   Note that this may not exactly correspond to what you understand from the figures in the text.
   1. The multiplexor controlled by the MemToReg control signal
   2. The logic for computing the branch address
   3. The multiplexor controlled by the RegDst control signal.
6. How many words of data memory are there?
7. Write a short MIPS program to load the word at location 0x4, the word at location 0x8, add the two values and store the result at location 0xc. You do not need to modify the VHDL datapath model – you will edit the file MS_IFETCH.VHD to place the program in instruction memory. Test this program, view the trace, and make sure the trace indicates that this program works. If you have assembled the program correctly it should run with no problems.
8. Compile and execute the multi-cycle data path. From the trace identify the following demonstrating to yourself that your program executes correctly
   1. Encoded instructions
   2. The operands as they show up at the output of the ALU

The results of the execution of each instruction as they show up at the input to the data memory.

4 Assignment Part II: Adding Instructions (150 Points)

Extend the multi-cycle datapath to add the jal instruction. The following sequence of steps is a recommended approach although it is not required, i.e., you may follow a different sequence of steps. **This component of the assignment must be done individually!** Note that the raw number of lines of VHDL code required is very little – on the order of a few lines. Most of the effort will be in understanding what needs to be done (item 1 below) and debugging (VHDL & ModelSim).

1. Using a figure of the multi-cycle datapath draw the modifications necessary for the datapath to correctly implement the extension.
2. From the preceding determine the values of all (and any new) control signals required to realize the desired functionality.
3. Update the controller state machine to implement the new functionality. The state machine is realized using a sequencer implementation with a ROM.
   a) Make updates to the controller ROM
   b) Make updates to the dispatch tables
   c) Any other additions as necessary
4. Modify the individual VHDL modules to make the hardware changes to the module.
5. Compile and test by creating test cases – short program sequences manually assembled.
6. Capture a trace for submission.

Test Program

The program below will need to be assembled manually and loaded into the instruction memory. The registers will also need to be initialized by making updates in the code.

Initialize $1,$2,$3 with values 1,2,3
Program

```
        add $4,$1,$2
        jal label1
        add $5,$4,$3

label1  add $6,$5,$4
          jal label2
          add $7,$6,$3

label2  add $8,$6,$3
          jal endlabel
          add $9,$8,$3

endlabel add $10,$8,$3
```

Expected results:

- $5, $7, $9 remain unchanged from whatever initial value they were set to. If they are changed, then the jump was not executed correctly.
- $4 = 3, $6 = 6, $8 = 9, $10 = 12

Submission Instructions

Section 3: No Submission required.

Section 4: Assignment Part II: Adding Instructions", the following must be submitted in a **single .zip file named MS_SPIM.zip to t-square** (Do not use any non-zip format, you will lose points).

1. **Design & Verification Report: A Single PDF file (2 pages max.) containing**
   - **a) Design: Modifications to the Data path**
     - i. Your name on all pages of the PDF
     - **ii.** Figure illustrating the modifications made to the multi-cycle datapath. Label and illustrate all modifications (logic, signals etc.). The powerpoint figure supplied in the notes can be used as a starting point for this.
     - **iii.** A bullet pointed summary (textual description) of the changes made to the datapath.
   - **b) Verification: Results from a test program which will be provided**
     - **i.** List the translated instructions from the test program (i.e copy paste from IFETCH.vhd). Include comments outlining what each instructions is doing.
     - **ii.** A screen shot of the simulation trace when the test program is executed.

2. **Code: ALL VHD FILES (modified and un-modified). Only VHD files. No .bak files, No Project files**
   - a. **Document your code.** Include your Name and GTID at the top of each VHDL module you submit (even the un-modified ones). Also include a brief description of what changes you made to each module at the top of the module.
   - b. Specifically for IFETCH.vhd: Each line of the test program must be documented.

Grading Guidelines

Report	Points
Design Section: Clear, complete, concise and correct design.	**35**
Verification: Clear, complete, concise and correct explanation of results. Well labeled screen shots of trace.	**15**
Code	
Compiles and a complete simulation can be executed.	**25**
Executes correctly. Specifically, execution of the test program should produce a correct trace which should be consistent with the one submitted in the report.	**60**
Documented per the guidelines in Submission Instructions	**15**
Total	**150**

Note: No late assignments will be accepted. You must make achieve a minimum average of 50% in the assignments to pass the course.

5 Assignment Part III: Problems (50 points)

1. (25 pts) Consider the execution of the following block of SPIM code on a multi-cycle datapath. The text segment starts at 0x00400000 and the data segment starts at 0x10010000. Assume immediate instructions take 4 cycles.

```
        .data
start:  .word 21, 22, 23,24
str:    .asciiz "CmpE"
        .align 4
        .word 24, 0x77

        .text
        .globl main

main:   li $t3, 4
        lui $t0, 0x1001
        lui $t1, 0x1002
move:   lw $t5, 0($t0)
        sw $t5, 0($t1)
        addiu $t0, $t0, 4
        addiu $t1, $t1, 4
        addi $t3, $t3, -1
end:    bne $t3, $zero, move
done:
```

Fill in the following values at the end of the requested cycle. Assume that the first cycle of program execution is cycle number **0**!

Cycle	ALUSrcB	ALUOp	RegDst	Instruction	PCWriteCond
12					
19					

2. (25 pts) Implement an overflow exception in the multi-cycle datapath. Your answer must include the following
   a) Figure of the multi-cycle datapath with all of the modifications clearly marked.
   b) Assume a sequencer implementation with a ROM.
      1. Show updates to the controller ROM
      2. Updates to the dispatch tables
      3. Any other additions as necessary
   c) Explain in a few sentences how the implementation will work.

Note: No late assignments will be accepted. You must make achieve a minimum average of 50% in the assignments to pass the course.