CpE 404: Modern Processor Architecture

Dr. Harris *Spring 2019*

HW 4: MIPS Pipelined Processor Due: Thursday, Feb 28th

Points Available: 200

Introduction

In this homework assignment you will build a MIPS pipelined processor using Verilog. First, you will design the Hazard Unit and test it by writing your own testbench and testvectors. Next, you will build your pipelined processor with hazard detection. Finally, you will test your MIPS pipelined processor with hazard detection using a test program.

Before starting this homework assignment, you should be very familiar with the pipelined implementation of the MIPS processor discussed in class and in your text, *Digital Design and Computer Architecture*. The pipelined processor schematic from the text is repeated at the end of this homework assignment for your convenience. Your MIPS pipelined processor should be able to execute the following instructions:

`add`, `sub`, `and`, `or`, `slt`, `lw`, `sw`, `beq`, `addi` and `j`

Remember to **start and complete this assignment early** and get any of your misunderstandings cleared up early. Also, remember to gauge your work. You have two weeks to complete this assignment, so you should get at least half of the work done in the first week – for example, you should aim to have **at least** the hazard unit and its testbench/testvector file complete by February 21st.

1. Building the Hazard Unit

Build the Hazard Unit described in class and in Chapter 7 of *Digital Design and Computer Architecture.* Write your own testbench and test vector files (based on the ones we've discussed in class) for testing the Hazard Unit. The testvectors should include values for all of the inputs of the Hazard Unit as well as the expected outputs. Remember that you can use the code posted on WebCampus (particularly see the testbench file: "Ex4.39_testbench3.v") as a starting point for your testbench.

2. Building the MIPS Pipelined Processor

Build the MIPS Pipelined Processor. Recall that you can reuse parts (Verilog modules) from your single-cycle MIPS processor from Homework 3.

3. Testing the pipelined MIPS processor

In this section, you will test the pipelined MIPS processor. Modify the testbench from Homework 3 to test your pipelined processor. You can run the first program from Homework 3 (memfile.dat). Simulate your processor with ModelSim. Refer to the Homework 3 instructions for a reminder of how to do this.

Run the simulation. At this point, you will likely get to hone your debugging skills. Look at the waveforms and drill down into the pipelined processor module to pull up signals that are needed to debug your pipelined processor.

What to Turn In

Please turn in each of the following items, clearly labeled and in the following order:

0. **[-5 points if omitted] Please indicate how many hours you spent on this assignment.** This will not affect your grade (unless omitted), but will be helpful for calibrating the workload for next semester's labs.
1. **[20 pts]** Your Verilog code for the **Hazard Unit**.
2. **[30 pts]** Your **testbench** and **testvector** files for the Hazard Unit.
3. **[20 pts]** Your ModelSim Simulation **waveforms** output **for the Hazard Unit testbench**.
4. **[100 pts]** Your Verilog code for your **MIPS pipelined processor with Hazard Unit**. Remember to submit clean, commented code. Messy, difficult to read code will lose points.
5. **[30 pts]** An image of the **simulation waveforms** showing correct operation of the processor executing the memfile.asm program. Display only the necessary signals and highlight important signals/values. Add brief notes to describe clearly what is occurring.

```
# memfile.asm
#
# Test the MIPS processor.
#  add, sub, and, or, slt, addi, lw, sw, beq, j
# If successful, it should write the value 7 to address 84

#Machine   Assembly               Description           Address
        main:
20020005   addi $2, $0, 5         # initialize $2 = 5     0
2003000c   addi $3, $0, 12        # initialize $3 = 12    4
2067fff7   addi $7, $3, -9        # initialize $7 = 3     8
00e22025   or   $4, $7, $2        # $4 <= 3 or 5 = 7      c
00642824   and  $5, $3, $4        # $5 <= 12 and 7 = 4    10
00a42820   add  $5, $5, $4        # $5 = 4 + 7 = 11       14
10a7000a   beq  $5, $7, end       # shouldn't be taken    18
0064202a   slt  $4, $3, $4        # $4 = 12 < 7 = 0       1c
10800001   beq  $4, $0, around    # should be taken       20
20050000   addi $5, $0, 0         # shouldn't happen      24
        around:
00e2202a   slt  $4, $7, $2        # $4 = 3 < 5 = 1        28
00853820   add  $7, $4, $5        # $7 = 1 + 11 = 12      2c
00e23822   sub  $7, $7, $2        # $7 = 12 - 5 = 7       30
ac670044   sw   $7, 68($3)        # [80] = 7              34
8c020050   lw   $2, 80($0)        # $2 = [80] = 7         38
08000011   j    end              # should be taken       3c
20020001   addi $2, $0, 1         # shouldn't happen      40
        end:
ac020054   sw   $2, 84($0)        # write [84] = 7        44
```

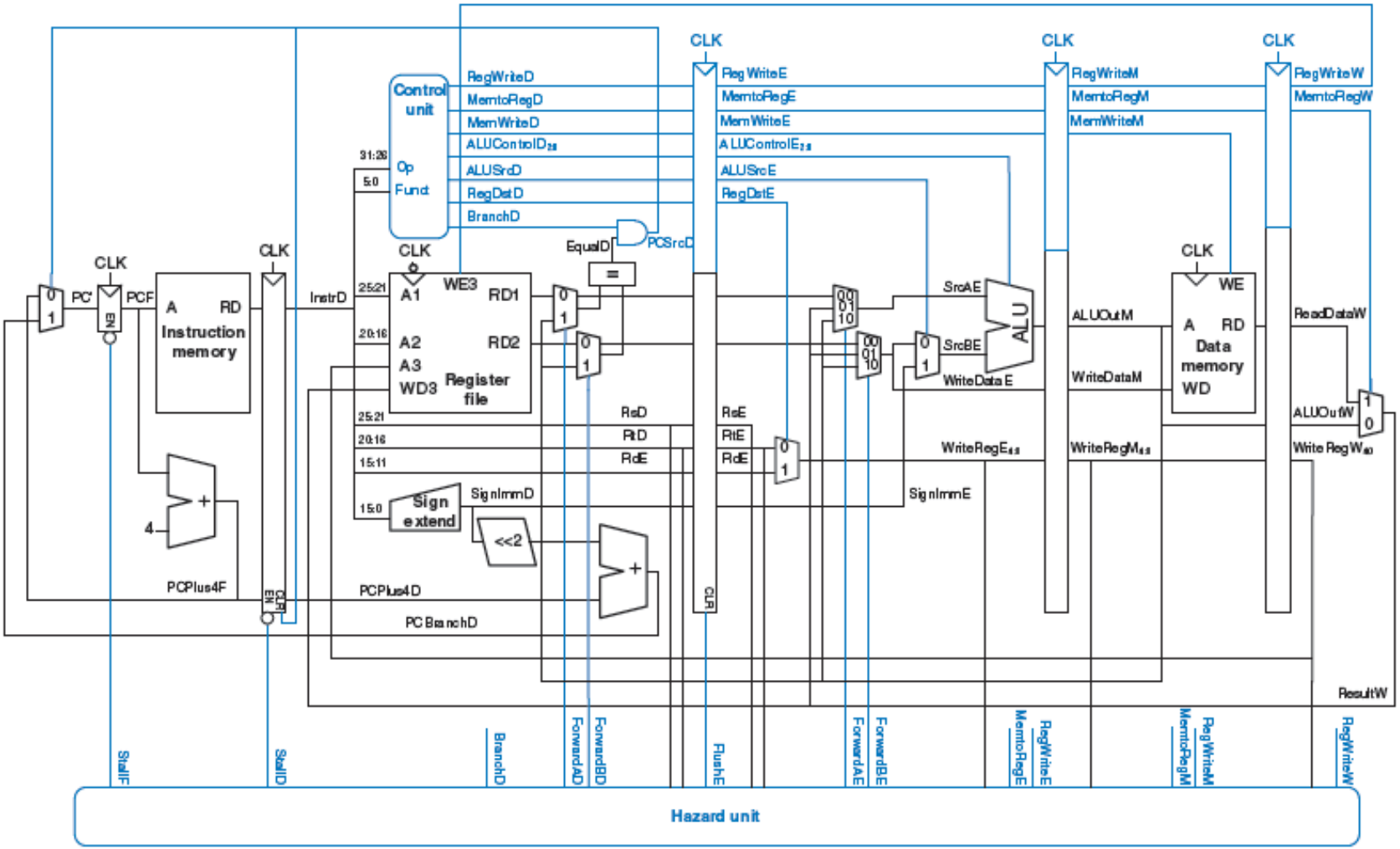

Figure 1. MIPS pipelined processor with Hazard Unit (Note: Hardware for `j` not shown in figure – you will need to add it)