# Introduction

***Digital Design and Computer Architecture***

# **Assignment #6**

# **MIPS Single-Cycle Processor Due: 4/24/18**

In this lab you will build a simplified MIPS single-cycle processor using SystemVerilog. Most of this code is taken from the textbook. Then you will load a test program and confirm that the system works. Next, you will implement two new instructions, and then test another program that confirms the new instructions work as well. By the end of this lab, you should thoroughly understand the internal operation of the MIPS single-cycle processor as well as the role of an ALU in a CPU.

Please read and follow the instructions in this lab carefully.

Before starting this lab, you should be very familiar with the single-cycle implementation of the MIPS processor described in Section 7.3 of your text, *Digital Design and Computer Architecture*. The single-cycle processor schematic from the text is repeated at the end of this lab assignment for your convenience. This version of the MIPS single-cycle processor can execute the following instructions: add, sub, and, or, slt, lw, sw, beq, addi, and j.

Our model of the single-cycle MIPS processor divides the machine into two major units: the control and the datapath. Each unit is constructed from various functional blocks. For example, as shown in the figure on the last page of this lab, the datapath contains the 32-bit ALU, the register file, the sign extension logic, and five multiplexers to choose appropriate operands.

1. MIPS Single-Cycle Processor

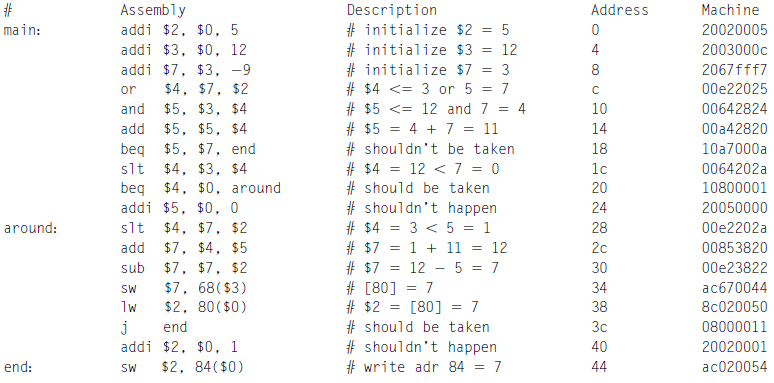
The SystemVerilog single-cycle MIPS modules are given in Section 7.6 of the text. Open these files by using the Vivado project file from Blackboard.

Study the modules until you are familiar with their contents. Look at the mips module, which instantiates two sub-modules, controller and datapath. Then take a look at the controller module and its submodules. It contains two sub-modules: maindec and aludec. The maindec module produces all control signals except those for the ALU. The aludec module produces the control signal, alucontrol[2:0], for the ALU. Make sure you thoroughly understand the controller module. Correlate signal names in the SystemVerilog code with the wires on the schematic.

After you thoroughly understand the controller module, take a look at the datapath SystemVerilog module. The datapath has quite a few submodules. Make sure you understand why each submodule is there and where each is located on the MIPS single-cycle processor schematic.

The highest-level module, top, includes the instruction and data memories as well as the processors. Each of the memories is a 64-word × 32-bit array. The instruction memory needs to contain some initial values representing the program. The test program is given below. Study the program until you understand what it does. The machine language code for this program is stored in memfile.dat.





2. Testing the single-cycle MIPS processor

In this section, you will test the processor.

In a complex system, if you don’t know what to expect the answer should be, you are unlikely to get the right answer. Begin by predicting what should happen on each cycle when running the program. Complete the chart in Table 1 at the end of the lab with your predictions. What address will the final sw instruction write to and what value will it write? The description of the program above will greatly aid you.

Simulate your processor with Vivado by running the simulation. If all goes well, the testbench will print “Simulation succeeded.” Look at the waveforms and check that they match your predictions in Table 1.

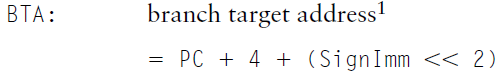
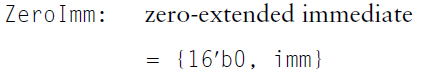
If you need to debug, you’ll likely want to view more internal signals. However, on the final waveform that you turn in, show the following signals in this order: clk, reset, pc, instr, aluout, writedata, memwrite, and readdata. **All the values need to be readable to get full credit.**

After you have fixed any bugs, print out your final waveform.

3. Modifying the MIPS single-cycle processor

You now need to modify the MIPS single-cycle processor by adding the ori and bne instructions. First, modify the MIPS processor schematic at the end of this lab to show what changes are necessary. You can draw your changes directly onto the schematic. Then modify the main decoder and ALU decoder as required. Show your changes in the tables at the end of the lab. Finally, modify the SystemVerilog code as needed to include your modifications.

The following descriptions of ori and bne may be useful:









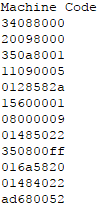
**4. Testing your modified MIPS single-cycle processor**

Next, you’ll need a test program to verify that your modified processor work. The program should check that your new instructions work properly and that the old ones didn’t break. Use the program below.

# test2.asm and memfile2.dat

# 23 March 2006 S. Harris sharris@hmc.edu

# Test MIPS instructions.

#Assembly Code

main: ori $t0, $0, 0x8000

addi $t1, $0, -32768

ori $t2, $t0, 0x8001

beq $t0, $t1, there

slt $t3, $t1, $t0

bne $t3, $0, here

j there

here: sub $t2, $t2, $t0

ori $t0, $t0, 0xFF

there: add $t3, $t3, $t2

sub $t0, $t2, $t0

sw $t0, 82($t3)

**Figure 1. MIPS assembly program and corresponding machine code memfile2.dat**

The machine code memfile2.dat should already be added to your project. Determine the address and data value written by the sw instruction (note the mix of decimal and hexadecimal in the assembly program). Modify the testbench to check for the appropriate address and data value indicating that the simulation succeeded. Run the program and check your results. Debug if necessary. When you are done, print out the waveforms as before and indicate the address and data value written by the sw instruction.

# What to Turn In

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

1. **Please indicate how many hours you spent on this lab.** This will not affect your grade (unless omitted), but will be helpful for calibrating the workload for next semester’s labs.
2. A completed version of Table 1.
3. An image of the simulation waveforms showing correct operation of the processor. Does it write the correct value to address 84?

The simulation waveforms should be in the following order: clk, reset, pc, instr, aluout, writedata, memwrite, and readdata. Do not display any other signals in the waveform. Check that the waveforms are zoomed out enough so the bus values are readable. Use several pages and multiple images as necessary.

1. Marked up versions of the datapath schematic and decoder tables that adds the ori and bne instructions.
2. Your SystemVerilog code for your modified MIPS computer (including ori and bne functionality) with the changes highlighted and commented in the code.
3. An image of the simulation waveforms showing correct operation of your modified processor on the new program. What address and data value are written by the sw instruction?
4. **(Extra Credit)** Implement this processor on your BASYS3 and display the 2 least significant bytes of Data Memory address 84 to the seven segment display.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Cycle** | **reset** | **pc** | **instr** | **branch** | **srca** | **srcb** | **aluout** | **zero** | **pcsrc** | **writedata** | **memwrite** | **read data** |
| 1 | 1 | 00 | addi $2,$0,5  20020005 | 0 | 0 | 5 | 5 | 0 | 0 | 0 | 0 | x |
| 2 | 0 | 04 | addi $3,$0,12  2003000c | 0 | 0 | c | c | 0 | 0 | 0 | 0 | x |
| 3 | 0 | 08 | addi $7,$3,-9  20067fff7 | 0 | c | -9 | 3 | 0 | 0 | 0 | 0 | x |
| 4 | 0 | 0C |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

**Table 1. First sixteen cycles of executing mipstest.asm**



**Single-cycle MIPS processor**

# Extended functionality. Main Decoder:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **Op5:0** | **RegWrite** | **RegDst** | **AluSrc1:0** | **Branch** | **MemWrite** | **MemtoReg** | **Jump** | **ALUOp1:0** |
| R-type | 000000 | 1 | 1 | 00 | 0 | 0 | 0 | 0 | 10 |
| lw | 100011 | 1 | 0 | 01 | 0 | 0 | 1 | 0 | 00 |
| sw | 101011 | 0 | X | 01 | 0 | 1 | X | 0 | 00 |
| beq | 000100 | 0 | X | 00 | 1 | 0 | X | 0 | 01 |
| addi | 001000 | 1 | 0 | 01 | 0 | 0 | 0 | 0 | 00 |
| j | 000010 | 0 | X | XX | X | 0 | X | 1 | XX |
| ori | 001101 |  |  |  |  |  |  |  |  |
| bne | 000101 |  |  |  |  |  |  |  |  |

**Extended functionality. ALU Decoder:**

|  |  |
| --- | --- |
| **ALUOp1:0** | **Meaning** |
| 00 | Add |
| 01 | Subtract |
| 10 | Look at funct field |
| 11 |  |