Christopher Lall

CSc 342/343 Comp Org – Professor Gertner

ALU Lab Experiment

Due 4/10/22

Table of Contents

[Objective: 3](#_Toc100250239)

[Code: 3](#_Toc100250240)

[Simulation/ModelSim: 13](#_Toc100250241)

[Conclusion: 19](#_Toc100250242)

# Objective:

The goal of this lab is to use an Instruction Register (IR), a 3-Port RAM (which accesses registers Rs, Rt, and Rd as needed), an Immediate Register for I-Format Instructions, a Data Memory (1-Port RAM), an ADD/SUB unit with flags overflow, negative, and zero, and finally a Bitwise Operation Unit integrated within the former unit to implement 16 MIPS Instructions in VHDL. Students must evaluate the validity of their design using ModelSim waveform simulation and compare the VHDL executed version of the MIPS instruction to its MARS equivalent after creating these instructions.

# Code:

Text

Description automatically generated

Figure 1. RD register VHDL code

Text

Description automatically generated

Figure 2. RS register VHDL code

Text

Description automatically generated

Figure 3. RT register VHDL code

Those are the three register files we had to create that IR register accesses based on the instructions. They are all 32-bit registers where RS and RT are used to store the two data values and RD writes the output signal once the arithmetic logic is complete.

Text

Description automatically generated

Figure 4. ADD/SUB VHDL code

When the opcode is 0 or 1 representing add and sub, we use the N-bit add/sub-component that we constructed in the previous experiments to perform adder/subtraction with overflow and negative flags.

A picture containing text

Description automatically generated

Figure 5. ALU VHDL code part 1

Text

Description automatically generated with medium confidence

Figure 6. ALU VHDL code part 2

A picture containing scatter chart

Description automatically generated

Figure 7. ALU VHDL code part 3

Scatter chart

Description automatically generated

Figure 8. ALU VHDL code part 4

The ALU file operates as a control unit, redirecting to other components based on the opcode to accomplish the logic operation. It redirects to the n-bit adder sub if the opcode is 0 or 1.

Text

Description automatically generated with medium confidence

Figure 9. Bitwise op VHDL code

This component is called when OPCODE points to a bitwise operation, and it, along with the sign extender, conducts bitwise operations for two 32-bit values.

Text

Description automatically generated

Figure 10. Sign Extender VHDL code

When we have a 32-bit value and a 16-bit IMM value, we use a sign extender to make the 16-bit value 32 bits by adding 0s to the tail of the 16-bit value. This allows us to complete the logic with two 32-bit values rather than one 32-bit and one 16-bit integer.

Text

Description automatically generated

Figure 11. IMM16 VHDL code

This 16-bit immediate register that helps us perform the immediate operations with the help of sign extenders which turns the 16-bit to 32- bits.

Text

Description automatically generated

Figure 12. MAR VHDL code

Text

Description automatically generated

Figure 13. MDR VHDL code

# Simulation/ModelSim:

Graphical user interface

Description automatically generated

Figure 14. ADD waveform neg and overflow modelsim.

Graphical user interface

Description automatically generated

Figure 15. Add underflow waveform modelsim

Graphical user interface

Description automatically generated

Figure 16. ADD with IMM ModelSim waveform

Graphical user interface

Description automatically generated

Figure 17. ADDIU IMM modelsim waveform

Graphical user interface

Description automatically generated

Figure 18. Subtraction of ALU modelsim waveform

Graphical user interface

Description automatically generated

Figure 19. SUBU modelsim waveform

Graphical user interface

Description automatically generated with medium confidence

Figure 20. AND modelsim waveform

A picture containing graphical user interface

Description automatically generated

Figure 21. NOR modelsim waveform

Graphical user interface

Description automatically generated

Figure 22. OR modelsim waveform

A picture containing graphical user interface

Description automatically generated

Figure 23. ADDI modelsim waveform

Graphical user interface

Description automatically generated

Figure 24. ORI modelsim waveform

Graphical user interface

Description automatically generated

Figure 25. SLL modelsim waveform

Graphical user interface

Description automatically generated

Figure 26. SRL modelsim waveform

Graphical user interface

Description automatically generated

Figure 27. SRA modelsim waveform

Graphical user interface

Description automatically generated with medium confidence

Figure 28. SW modelsim waveform

Graphical user interface

Description automatically generated with low confidence

Figure 29. LW modelsim waveform

# Conclusion:

We learned how to design an ALU unit capable of simulating the behavior of 16 MIPS instructions in both R-Format and I-Format formats. We learned how to create a data memory unit and how to utilize it to do operations like store and load words. We learnt how to configure and use access registers RS, RT, and RD for these instructions. Finally, we learned how arithmetic/logic operations are conducted among register values and how these values compare to the MIPS operations when these components are combined.