CSCE 230 Project

Phase III

Group #9

Members:Johnathan Carlson,Tyler Zinsmnaster,Jake Ediger

Outline: This report will detail the following topics from Phase III:

1. Current Capabilities of Processor

2. Current Components/working parts
3. Process of connecting, implementing, and testing the various parts/functions

1) Current Capabilities of Processor:

Phase III involved adding more working instructions,commands. This gives the processor many more capabilities than phase II, and a more complete processor. In a broad sense, there is added compatibility for **B-type**,**J-type**, and **D-type** instructions.**R-type** instructions were already implemented in phase II, except for jr.

- **B-type** instructions gives the ability to branch and link, which are very important instructions for a program to flow. With the ability to branch, it is now possible to “call” other functions and to loop through a task. This greatly improves efficiency when writing a program, to the point that it is indispensable.
- **J-type** instructions deal with jump, jump and link, and load immediate jump. Jumping addresses can be used with PC to change certain values. Jump and link jumps to an instruction, and stores the next instruction. Load immediate loads a const into a specific register.
- **D-type** contains important instructions like load word, store word, addi, and set interrupt. Loading and storing words is crucial if you want to go to any functions or hold values. Addi allows the ability to add a numerical value to register, rather than just adding registers to registers. Subi is not implemented, but its functionality can be entirely implemented through addi.
- **R-Type** instructions were already implemented in phase II, which contains commands like addi, subi, and, or, and xor.

2) Current Components and Working parts

There are several different components in the processor as of now. Designed components include: the register file, ALU, and control unit. Given components as part of the project: Instruction Address generator, I/O memory interface, Memory interface, and Immediate block. The current working parts include the register file, the ALU, the control Unit, and partially working I/O.

Phase III:

Components were added in phase III that were needed to implement the new capabilities. Added/Changed components include:

- Fast adder
- Memory Interface
- New parts to assembler able to interpret new instructions
- Control Unit heavily modified/Datapath
- Necessary Muxes

After testing, each of these components was fully functional and able to implement the types of instructions needed to complete this phase.

3) Process of connecting, implementing, and testing the various parts/functions

To bring all of the components together, V.H.D.L., which stands for “Very High Speed Integrated Circuit Hardware Description Language”, was used. It is a language developed for implementing logic onto a circuit, used to design the processor. Each of the parts of the processor are stored in different files in one overarching project. Similar to using functions in a language like C, it is possible to use different components in different parts of the program. After all of the parts are connected(via PORT MAP), it is possible to begin implementing the various components. In order to implement or test anything, it is necessary to first compile the program, and resolve any errors that may appear because of bad logic or syntax errors. After everything is resolved, the program can then be tested to see if any problems remain.

Testing is done by using a .do file to test certain(if not all) possible values, and then see if the output matches what is expected. “ModelSim” is the environment used to simulate inputs and outputs. Adding many different “waves” in the do file allowed checking everything in modelSim to see what the program was doing at any given point. Hand checking the values was necessary to confirm the accuracy of the output.

1. **Phase II:** Phase II involved setting up R-type instructions, as well as initial designing/testing of the assembler (assembler created in Python).
    a. Phase II began with a certain number of components either provided or previously created. The main difficulty was making these parts compatible and able to be implemented together. The ALU and the Control Unit required minimal changes. Multiple multiplexers and Register configurations were needed, some of which was made easier using MegaWizard, a tool in Quartus. VHDL allows all components to be stored in different files, and mapped into a “main function” that puts everything together.
    b. Testing was mainly done putting each implemented instruction onto a .do file, multiple times. Bugs came up when more than one command was implemented onto one register in one program run. After this was solved, the rest of the issues were solved by setting register 0(r0) set to 1, because otherwise there was no way to add/manipulate numbers (addi had not been implemented yet). After several rigorous testing trials, there was confirmation the commands were working correctly.
2. **Phase III:**
    a. Phase III added on to phase II instructions, adding B-type,J-type, and D-type. The assembler is almost entirely completed to the point where all of these types are functional. The control unit needed major changes in order to be compatible

with the new instructions/parts. After the CU was fixed, simulations were run on Modelsim to ensure correctness of all new instructions. Running this program on the FPGA would not always result in the same results as running the exact same program in Modelsim. The difference happens in the clock, which requires different specifications for different hardware like the FPGA.