Project Title: 16-bit Pipelined Processor Design

Team Members: Kyle Johnson and Sandro Leon

Class: EECE 3026

Date Submitted: 8-3-16

# Register File

## Description of IO

## Summary of Circuit Functionality

## Writing Data to a Register

## Reading Data from a Register

## Program Counter

# ALU

## Description of IO

## Summary of Circuit Functionality

## Selection of ALU Operation

## Addition of Two Operands

## Subtraction of Two Operands

## AND of Two Operands

## OR of Two Operands

## NOT of Two Operands

## Shift Left Operation

## Shift Right Operation

## Negative Bit Detection

## Zero Result Detection

# Instruction Decode

## Instruction Type Table

## Description of IO

## Summary of Circuit Functionality

## Parsing the 16-bit instruction

## Determining the OpCode

## Determining the Rd output

# Sign Extend

## Description of IO

## Summary of Circuit Functionality

## Branch Multiplexer

# RAM

## Description of IO

## Summary of Circuit Functionality

## Writing to RAM

## Reading from RAM

# ROM

## Description of IO

## Summary of Circuit Functionality

## Reading from ROM

# PSW

## Description of IO

## Summary of Circuit Functionality

## Choosing PSW Register’s Input Data

# Control

## Description of IO

## Summary of Circuit Functionality

## Determining the OpCode

## Determining RegWrite

## Determining MemtoReg

## Determining MemWrite

## Determining ALUOp

## Determining Instruction Type

## Outputting remaining control signals

# Exception Handling

## Detecting a program check violation (PCV)

## Detecting Program Timeout

## Resolving exceptions

# Optimization

## Reworking RegWrite, MemWrite, and MemtoReg Logic

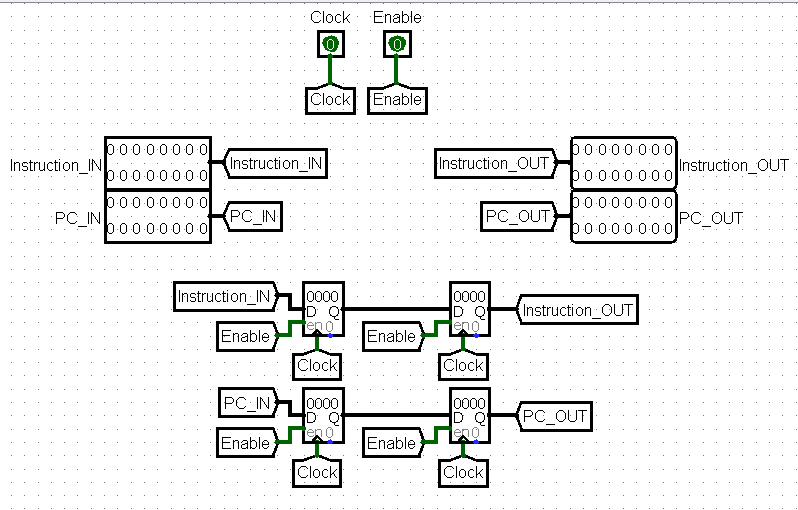
## Calculating RAM Address

## Use of the tunnel object in Logisim

# Appendix

## Final Circuit Overview

**Instruction Fetch/Instruction Decode Pipeline Register**





Description of I/O

Clock – 1-Bit input of Clock signal used to update the value in the intermediate registers.

Enable – 1-Bit input used to keep pipeline registers active for writing.

Instruction\_IN – 16-Bit input of the instruction retrieved from ROM in the instruction fetch stage.

PC\_IN – 16-Bit input of the program counter from the instruction fetch stage.

Instruction\_OUT – 16-Bit output of the instruction to be instruction decode stage.

PC\_OUT – 16-Bit output of the program counter to be passed to the instruction decode stage.

Summary of Circuit Functionality

The Instruction Fetch/Instruction Decode Pipeline Register is used to store the output of the previous stage, and to be read from in the next stage. This is to ensure that each stage can pass data onto the next stage before moving on to a new instruction.



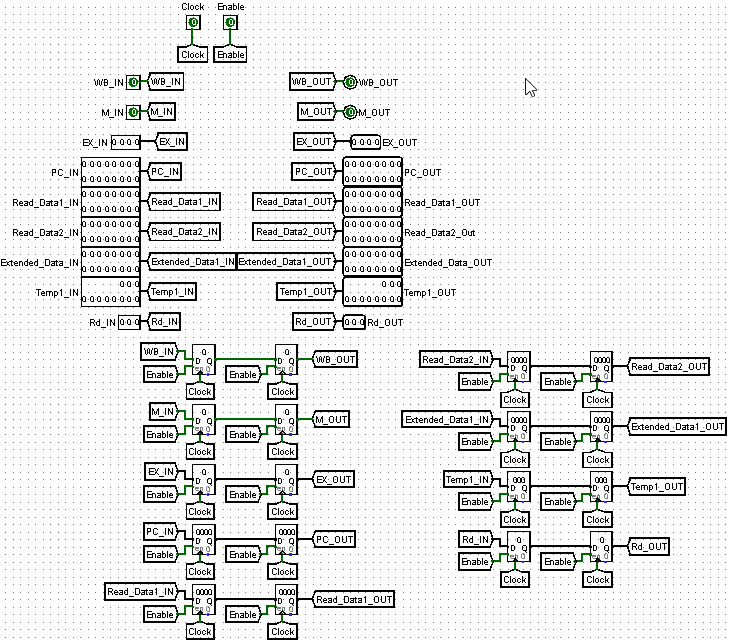
Writing Data to a Register

A write bit is enabled (with selection signal Enable) to allow the pipelined registers to store input data from the previous stage. This is accomplished by using a falling edge trigger to enable new data to be transferred into the input register (the one on the left). This ensures that the data from the stage is only saved when the clock is falling and the stage cycle completes.

Reading Data from a Register

A write bit is enabled (with selection signal Enable) to allow the pipelined registers to store input data from the previous register. This is accomplished by using a rising edge trigger to enable new data to be transferred from the input register (the one on the left) into the output register (the one on the right). By having a delay where the input register is written to on the falling edge, and the output register is written to on the rising edge, there is a period where the output register is able to contain the previous register’s values so that the previous register can save new data from the input stage and stored data can be read into the next stage. This ensures that the data from the previous stage is only read into the next stage when the clock is rising and the next stage cycle starts.

**Instruction Decode/Instruction Execute Pipeline Register**





Description of I/O

Clock – 1-Bit input of Clock signal used to update the value in the intermediate registers.

Enable – 1-Bit input used to keep pipeline registers active for writing.

WB\_IN – 1-Bit input used to save the RegWrite control signal value for a given instruction

M\_IN – 1-Bit input used to save the MemWrite control signal value for a given instruction

PC\_IN – 16-Bit input used to save the program counter for a given instruction

Read\_Data1\_IN – 16-Bit input used to store the data of Rs1

Read\_Data2\_IN – 16-Bit input used to store the data of Rs2

Extended\_Data\_IN – 16-Bit input of the offset value associated with a given instruction

Rd\_IN - 3-Bit input used to store the register number Rd

WB\_OUT – 1-Bit output used to output the RegWrite control signal value for a given instruction

M\_OUT – 1-Bit output used to output the MemWrite control signal value for a given instruction

PC\_OUT – 16-Bit output used to output the program counter for a given instruction

Read\_Data1\_OUT – 16-Bit output used to output the data of Rs1

Read\_Data2\_OUT – 16-Bit output used to output the data of Rs2

Extended\_Data\_OUT – 16-Bit output of the offset value associated with a given instruction

Rd\_OUT - 3-Bit output used to store the register number of Rd

Summary of Circuit Functionality

The Instruction Decode/Instruction Execute Pipeline Register is used to store the output of the previous stage, and to be read from in the next stage. This is to ensure that each stage can pass data onto the next stage before moving on to a new instruction.



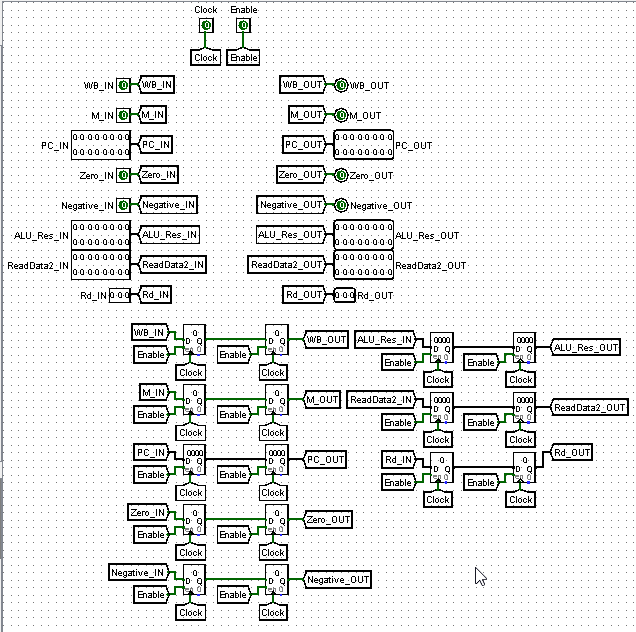
Writing Data to a Register

A write bit is enabled (with selection signal Enable) to allow the pipelined registers to store output data from the previous stage. This is accomplished by using a falling edge trigger to enable new data to be transferred into the output register (the one on the left). This ensures that the data from the stage is only saved when the clock is falling and the stage cycle completes.

Reading Data from a Register

A write bit is enabled (with selection signal Enable) to allow the pipelined registers to store output data from the previous register. This is accomplished by using a rising edge trigger to enable new data to be transferred from the output register (the one on the left) into the output register (the one on the right). By having a delay where the output register is written to on the falling edge, and the output register is written to on the rising edge, there is a period where the output register is able to contain the previous register’s values so that the previous register can save new data from the output stage and stored data can be read into the next stage. This ensures that the data from the previous stage is only read into the next stage when the clock is rising and the next stage cycle starts.

**Instruction Execute/Memory Pipeline Register**





Description of I/O

Clock – 1-Bit input of Clock signal used to update the value in the intermediate registers.

Enable – 1-Bit input used to keep pipeline registers active for writing.

WB\_IN – 1-Bit input used to save the RegWrite control signal value for a given instruction

M\_IN – 1-Bit input used to save the MemWrite control signal value for a given instruction

PC\_IN – 16-Bit input used to save the program counter for a given instruction

Zero\_IN – 1-Bit input used to save the zero-bit output of the ALU for a given instruction

Negative\_IN – 1-Bit input used to save the negative-bit output of the ALU for a given instruction

Read\_Data2\_IN – 16-Bit input used to save the data of Rs2

Rd\_IN - 3-Bit input used to save the register number of Rd

WB\_OUT – 1-Bit output used to output the RegWrite control signal value for a given instruction

M\_OUT – 1-Bit output used to output the MemWrite control signal value for a given instruction

PC\_OUT – 16-Bit output used to output the program counter for a given instruction

Zero\_OUT – 1-Bit input used to output the zero-bit output of the ALU for a given instruction

Negative\_OUT – 1-Bit input used to output the negative-bit output of the ALU for a given instruction

Read\_Data2\_OUT – 16-Bit input used to output the data of Rs2

Rd\_OUT - 3-Bit input used to output the register number of Rd

Summary of Circuit Functionality

The Instruction Decode/Instruction Execute Pipeline Register is used to store the output of the previous stage, and to be read from in the next stage. This is to ensure that each stage can pass data onto the next stage before moving on to a new instruction.



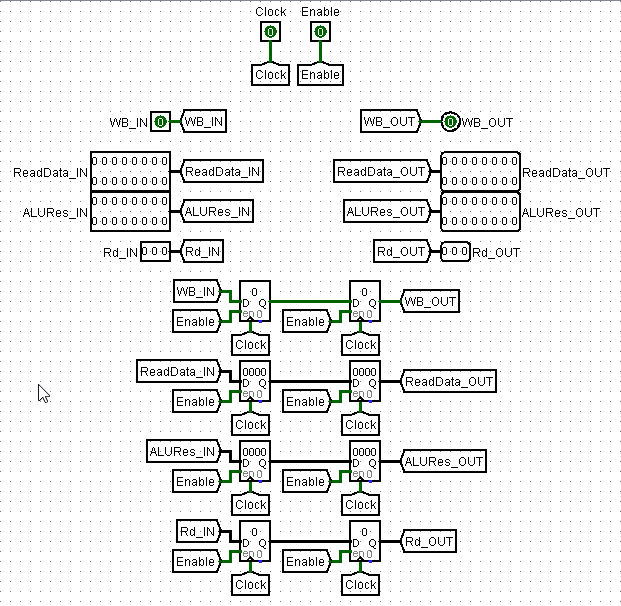
Writing Data to a Register

A write bit is enabled (with selection signal Enable) to allow the pipelined registers to store output data from the previous stage. This is accomplished by using a falling edge trigger to enable new data to be transferred into the output register (the one on the left). This ensures that the data from the stage is only saved when the clock is falling and the stage cycle completes.

Reading Data from a Register

A write bit is enabled (with selection signal Enable) to allow the pipelined registers to store output data from the previous register. This is accomplished by using a rising edge trigger to enable new data to be transferred from the output register (the one on the left) into the output register (the one on the right). By having a delay where the output register is written to on the falling edge, and the output register is written to on the rising edge, there is a period where the output register is able to contain the previous register’s values so that the previous register can save new data from the output stage and stored data can be read into the next stage. This ensures that the data from the previous stage is only read into the next stage when the clock is rising and the next stage cycle starts.

**Memory/Write back Pipeline Register**





Description of I/O

Clock – 1-Bit input of Clock signal used to update the value in the intermediate registers.

Enable – 1-Bit input used to keep pipeline registers active for writing.

WB\_IN – 1-Bit input used to save the RegWrite control signal value for a given instruction

ReadData\_IN – 16-Bit input used to save the output value of the RAM for a given instruction

ALU\_Res\_IN – 16-Bit input used to save the output value of the ALU for a given instruction

Rd\_IN - 3-Bit input used to save the register number of Rd

WB\_OUT – 1-Bit input used to output the RegWrite control signal value for a given instruction

ReadData\_OUT – 16-Bit input used to output the output value of the RAM for a given instruction

ALU\_Res\_OUT – 16-Bit input used to output the output value of the ALU for a given instruction

Rd\_OUT - 3-Bit input used to output the register number of Rd

Summary of Circuit Functionality

The Instruction Decode/Instruction Execute Pipeline Register is used to store the output of the previous stage, and to be read from in the next stage. This is to ensure that each stage can pass data onto the next stage before moving on to a new instruction.



Writing Data to a Register

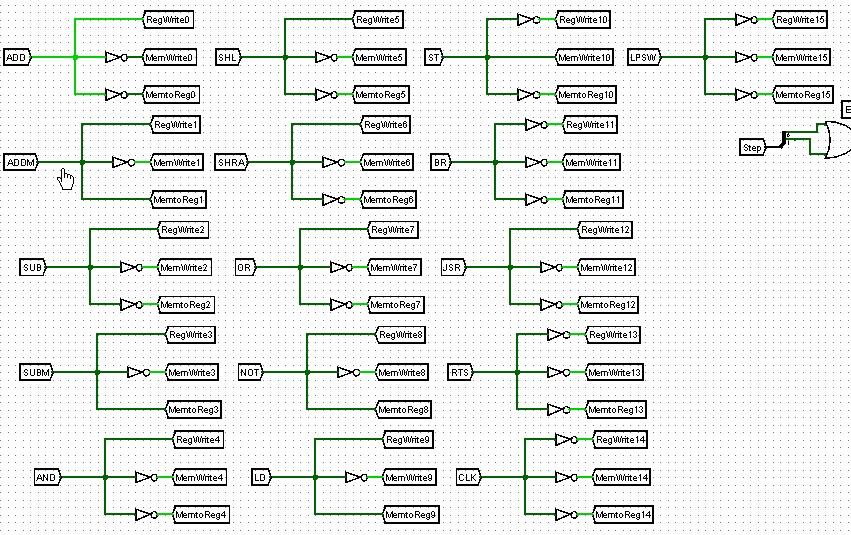
A write bit is enabled (with selection signal Enable) to allow the pipelined registers to store output data from the previous stage. This is accomplished by using a falling edge trigger to enable new data to be transferred into the output register (the one on the left). This ensures that the data from the stage is only saved when the clock is falling and the stage cycle completes.

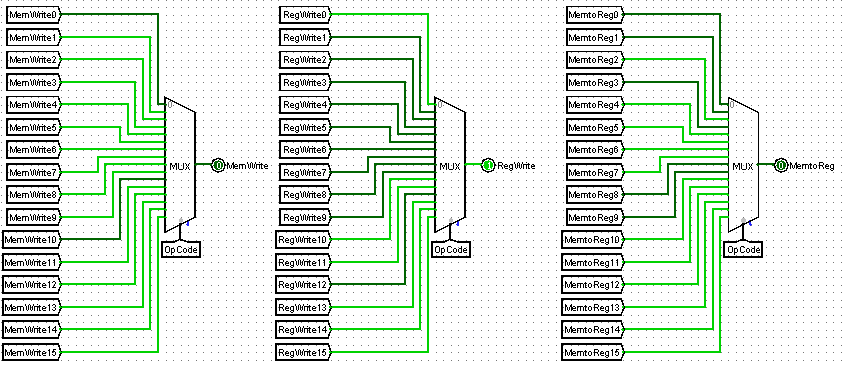
Reading Data from a Register

A write bit is enabled (with selection signal Enable) to allow the pipelined registers to store output data from the previous register. This is accomplished by using a rising edge trigger to enable new data to be transferred from the output register (the one on the left) into the output register (the one on the right). By having a delay where the output register is written to on the falling edge, and the output register is written to on the rising edge, there is a period where the output register is able to contain the previous register’s values so that the previous register can save new data from the output stage and stored data can be read into the next stage. This ensures that the data from the previous stage is only read into the next stage when the clock is rising and the next stage cycle starts.

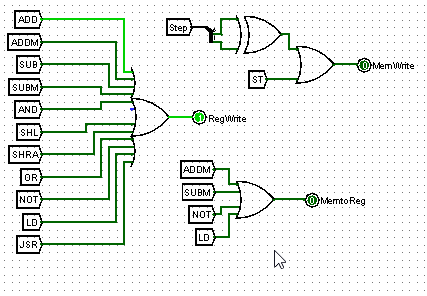
**Optimization**

Reworking RegWrite, MemWrite, and MemtoReg Logic

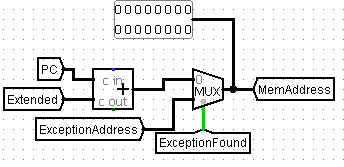




Originally MemWrite, RegWrite, and MemtoReg values were determined for each and every instruction regardless if the instruction used the control signal or not. The correct values of the control signals were then determined using multiplexers with the OpCode being their selector. This convoluted but functional method was then replaced with the much more streamlined use of OR gates pictured below.

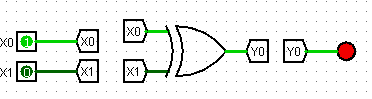


Calculating RAM Address

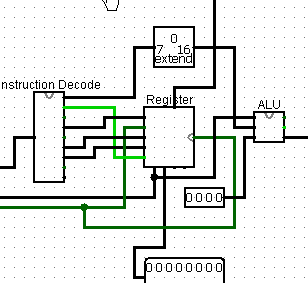


Originally, the main memory address was calculated in the ALU. As the instruction began getting more complex, it became more apparent that a better solution could be implemented. We added hardware to our circuit outside the of the ALU. This additional hardware (an adder and 2x1 multiplexer) allowed for addresses to be more easily selected for instructions with varying complexity. The address will either be the program counter + 16-bit extended offset (can be either short or long) or the address associated with the step # of the exception (see exception handling section).

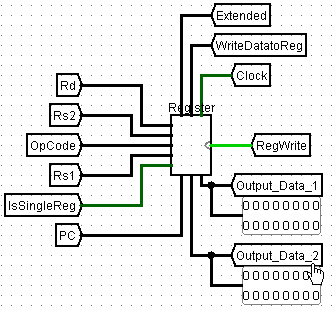
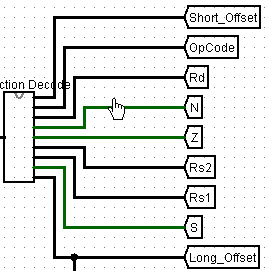
Use of The Tunnel Object in Logisim



We made the decision as a team to utilize the Tunnel Object in Logisim. The tunnels allow you to connect I/O without tracing wires from point A to point B. Although this is not necessarily an optimization that translates to the physical circuit design. The ease of understanding it brought during the design process was invaluable.



Before Use of Tunnels



After Use of Tunnels

**Appendix**

Final Circuit Overview

