Team Member 1: Kyle Johnson

Team Member 2: Sandro Leon

Class: EECE 3026

Due Date: 7-7-16

Project Title: Single Bus Processor design

Table of Contents

**Register Fileiii**

**INTRODUCTION1**

**TOPICS OF DISCUSION1**

Implementation of Recycling Bins1

Implementation of Motion Activated Lighting1

Incorporation of Electric Vehicles2

**FEASIBILITY2**

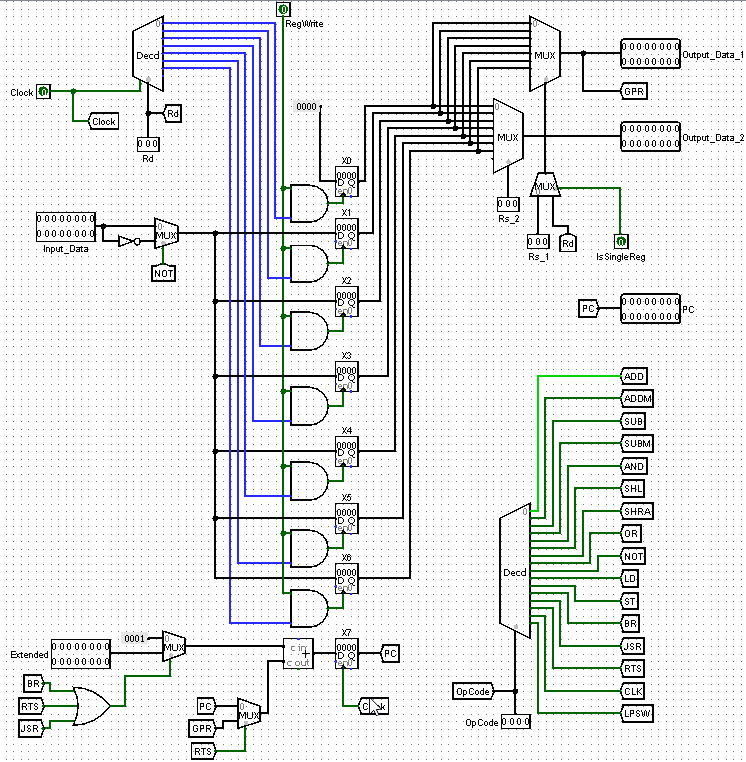
**PERSONNEL3**

**CONCLUSION3**

**REFERENCES4**

Tttttt

**Register File**



|  |  |
| --- | --- |
| **Register File** | |
| **Input(s)** | **Output(s)** |
| Rd | Output\_Data\_1 |
| Rs1 | Output\_Data\_2 |
| Rs2 | PC |
| Clock |  |
| RegWrite |  |
| InputData |  |
| Extended |  |
| IsSingleReg |  |
| OpCode |  |

Description of I/O

Rd – 3-bit Input of destination register index

Rs1 – 3-bit Input of operand #1 register index

Rs2 – 3-bit Input of operand #2 register index

Clock – 1-bit Input of clock signal used to enable decoder output

RegWrite – 1-bit Input of Register Write Enable signal

InputData – 1-bit Data to be written to the selected register if RegWrite is High

Extended – 16-bit Input sign extended signal (It can be short or long offset extended based on external logic)

IsSingleReg – 1-bit Input Control signal denoting a single register instruction (Only Rd is used)

OpCode – 4-bit Input Opcode used to determine which instruction is being executed

Output\_Data\_1 – 16-bit Output of a single selected Register (be either Rd or Rs1 depending on logic)

Output\_Data\_2 – 16-bit Output of a single selected Register (Rs2)

PC – 16-bit Output of program counter (Reg[7])

Summary of Circuit Functionality

The register file brings in critical functionality to any CPU. It allows the data in the registers to be read from and written depending on a given instruction and its inherent control signal.

Writing Data to a Register

A decoder (with selection signal Rd) is used to select which register will be written to. A register can only be written to if it both selected and the RegWrite control signal is high.

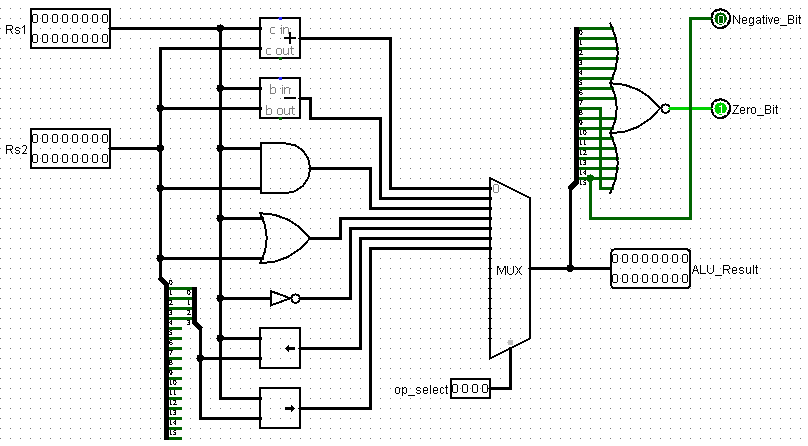
Reading Data from a Register

A multiplexer is used to select which register’s data will be read from. There are 2 multiplexers found in order to accommodate the reading of 2 registers at once. Inputs Rs1 and Rs2 are used as the selector bits in order to choose which registers to read from. Special accommodations must be made for D-Type instructions where there is no Rs1 or Rs2. The control signal IsSingleReg is used to select Rd as Read Data 1 instead of Rs1 during these instructions.

Program Counter

The logic for the program counter is implemented in the register file. The PC ( Reg[7] ) operates on the falling edge of the clock signal. During most instructions, the PC simply increments by one at the completion of the instruction. For certain instruction/conditions alternative values for PC must be applied. A decoder is also used (with the selector bit being the OpCode) in order to determine which instruction is active. Logic is implemented in order to handle when the PC is not just simply incremented by 1.

**ALU**





Description of I/O

Rs1 – 16-bit Input of operand #1

Rs2 – 16-bit Input of operand #2

op\_select – 4-bit ALU operation mode select

Negative\_Bit – 1-bit output denoting a negative result from ALU

Zero\_Bit – 1-bit output denoting a zero result from ALU

ALU\_Result – 16-bit containing the result of the ALU operation

Summary of Circuit Functionality

The ALU is used to perform a multitude of operations on the operand(s) given. The included functions of our ALU are ADD, SUB, AND, OR, NOT, Shift\_Left, and Shift\_Right. The ALU also has the functionality to determine whether the output is negative as well as zero.

Selection of ALU Operation

A 16x1 multiplexer is used to select which ALU operation is to be performed. The multiplexer’s selection is controlled by the 4-bit op\_select input.

Adding of Two Operands

The ALU accepts inputs Rs1 and Rs2. It then performs an addition operation using Logisim’s built-in adder module. The result is then passed to the ouput ALU\_Result.

Subtracting of Two Operands

The ALU accepts inputs Rs1 and Rs2. It then performs a subtraction operation using Logisim’s built-in subtraction module. The result is then passed to the ouput ALU\_Result.

AND of Two Operands

The ALU accepts inputs Rs1 and Rs2. It then performs a bit-wise AND operation using an AND gate. The result is then passed to the ouput ALU\_Result.

OR of Two Operands

The ALU accepts inputs Rs1 and Rs2. It then performs a bit-wise OR operation using an OR gate. The result is then passed to the ouput ALU\_Result.

NOT of Two Operands

The ALU accepts inputs Rs1 and Rs2. It then performs a bit-wise OR operation using an OR gate. The result is then passed to the ouput ALU\_Result.

Shift Left Operation

The ALU accepts bits 3-0 of input Rs2 using a splitter. It then performs a logical shift left operation using Logisim’s built-in left shift module. The result is then passed to the ouput ALU\_Result.

Shift Right Operation

The ALU accepts bits 3-0 of input Rs2 using a splitter. It then performs a logical shift right operation using Logisim’s built-in right shift module. The result is then passed to the ouput ALU\_Result.

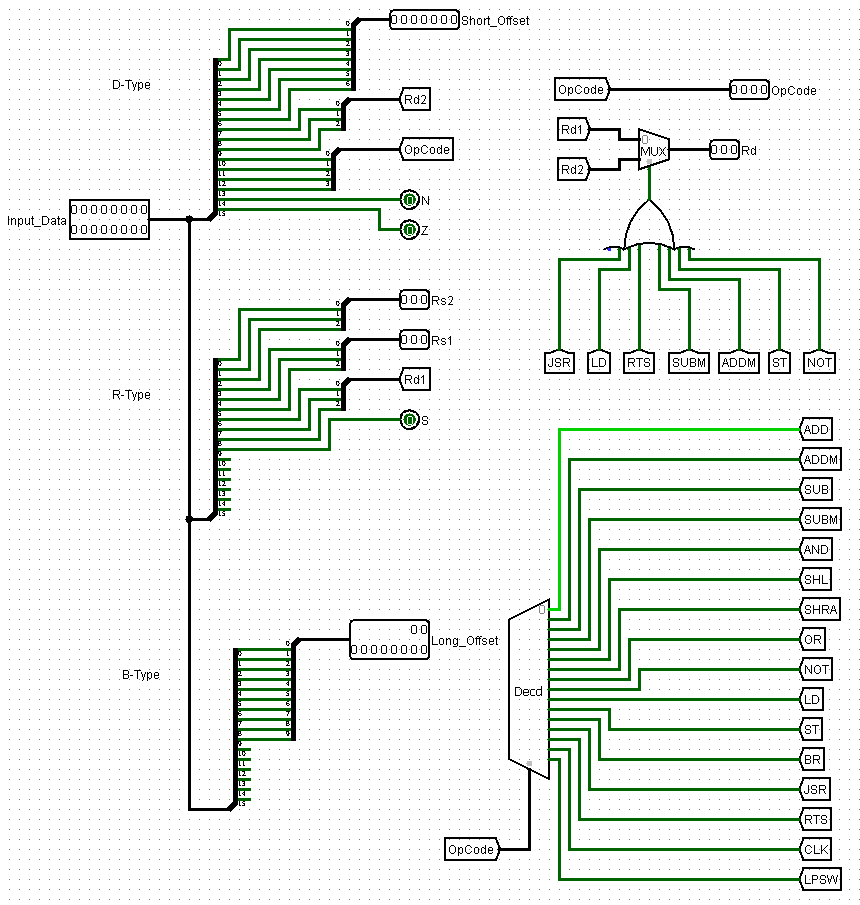
Negative Bit Detection

Using a splitter, the ALU result is parsed into 16 individual bits. The negative bit detection works by tying the most significant bit of the ALU result to the Negative\_Bit output.

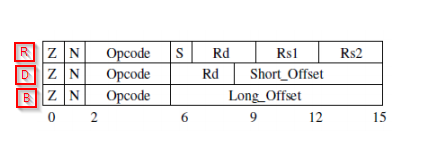
Zero Result Detection

Using a splitter, each bit of the ALU result is fed into a NOR gate. The Zero\_Bit output only enables when all bits of the ALU\_Result are 0.

**Instruction Decode**



Instruction Type Table



In order to more clearly identify the three instruction types, we have assigned types of R, D, and B for Arithmetic, Data, and Branch type instructions respectively as seen in the figure above.



Description of I/O

Input\_Data – 16-bit input instruction read from ROM

Short\_Offset - 7-bit output used in the instructions requiring the short offset

N – 1-bit output denoting whether the N bit of the instruction is HIGH

Z – 1-bit output denoting whether the Z bit of the instruction is HIGH

S – 1-bit output denoting whether the S bit of the instruction is HIGH

Rs1 – 3-bit output denoting the index of Rs1 in the register file

Rs2 – 3-bit output denoting the index of Rs2 in the register file

Rd – 3-bit output denoting the index of Rd in the register file

Long\_Offset - 10-bit output used in the instructions requiring the long offset

OpCode – 4-bit output denoting which operations will be performed for the given instruction

Summary of Circuit Functionality

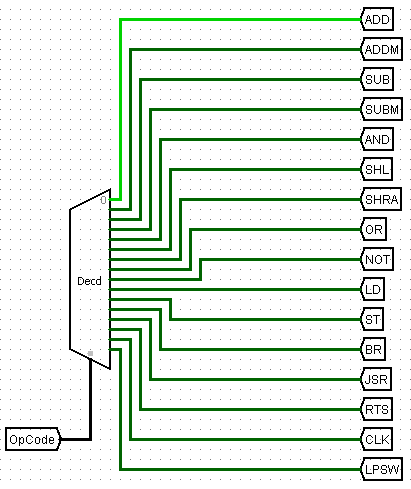
The instruction decoder (ID) is used to decode the instruction memory. The main functionality of the circuit consists of parsing the 16-bit instruction with splitters into functional indexes and values.

Parsing the 16-Bit Instruction

A 16-bit input instruction is fed into the instruction decoder. Splitters are implemented in order to extract instruction components in the following ways:

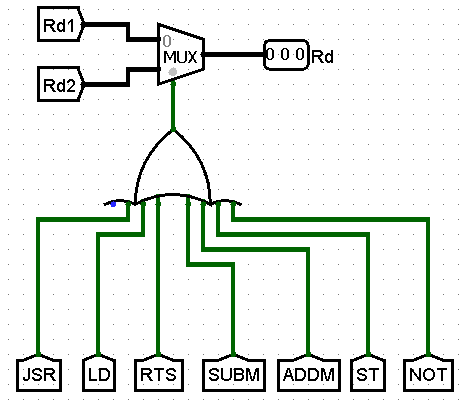
* Short\_Offset
  + 7-bit output is read from the instruction bits 6-0
* N
  + 1-bit output is read from the instruction bit 15
* Z
  + 1-bit output is read from the instruction bit 14
* S
  + 1-bit output is read from the instruction bit 9
* Rs1
  + 3-bit output is read from the instruction bits 5-3
* Rs2
  + 3-bit output is read from the instruction bits 2-0
* Rd1
  + 3-bit output is read from the instruction bits 8-6
* Rd2
  + 3-bit output is read from the instruction bits 9-7
* Long\_Offset
  + 10-bit output is read from the instruction bits 9-0
* OpCode
  + 4-bit output is read from the instruction bits 13-10

Determining The OpCode



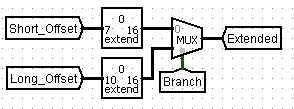
The 4-bit OpCode is used as the decoder’s selector input. The output of the decoder indicates which instruction is active. The active instruction is used as a control bit in other parts of the instruction decoder.

Determining The Rd Output



A multiplexer is used to determine whether Rd1 or Rd2 should be assigned as the Rd output. The zero-state of the multiplexer assigns Rd1 to the Rd output for R type instructions. The HIGH-state of the multiplexer assigns Rd2 to the Rd output for D type instructions. Using an OR gate, the selector bit for the multiplexer is HIGH if a D type instruction is HIGH.

Sign-Extend





Description of I/O

Short\_Offset - 7-bit input given to Logisim’s sign extension module

Long\_Offset - 10-bit input given to Logisim’s sign extension module

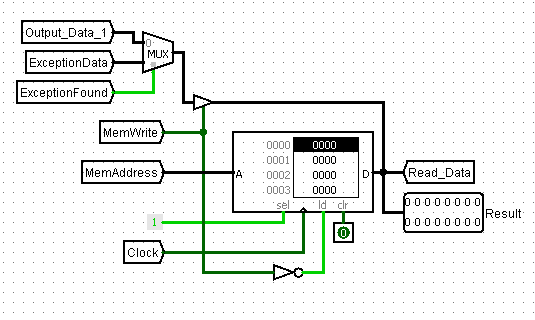
Branch – 1-bit input used to control the selection of the multiplexer

Extended – 16-bit output denoting the offset needed for the given instruction

Branch Multiplexer

The zero-state of the multiplexer assigns a 16-bit extended Short\_Offset to the Extended output for D type instructions. The HIGH-state of the multiplexer assigns a 16-bit extended Long\_ Offset to the Extended output for B type instructions. The selector bit, Branch, for the multiplexer is HIGH if a B type instruction is HIGH, and the zero-input state is used for D type instructions.

RAM





Description of I/O

Output\_Data\_1 – 16-bit input data to be written to the RAM

ExceptionData - 16-bit input data to be written to the RAM

ExceptionFound – 1-bit input to control data to be written to the RAM

MemWrite – 1-bit input to enable data write to the RAM

MemAddress – 16-bit input denoting which address in RAM should be accessed/ written

Clock – 1-bit input used to update RAM on rising edge of clock

Read\_Data – 16-bit output denoting the data stored in the given memory address

ExceptionFound Multiplexer

The zero-state of the multiplexer assigns a 16-bit Output\_Data\_1 to the output for non-exception instructions. The HIGH-state of the multiplexer assigns a 16-bit ExceptionData to the output for exception instructions. The selector bit, ExceptionFound, for the multiplexer is HIGH if an exception has been detected, and the zero-input state is used if no exception has been flagged.

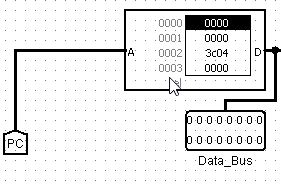
Writing To RAM

When the input MemWrite is HIGH, the RAM can be written to on the rising edge of the clock signal. A tri-state buffer conditionally allows the flow of the input data to the RAM.

Reading From RAM

When the input MemWrite is LOW, the RAM loads the contents of the current address into Read\_Data. The tri-state buffer disables the flow of the input data to the RAM.

ROM





Description of I/O

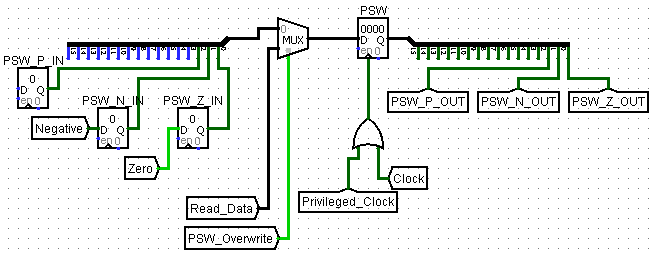
PC – 16-bit input Program Counter to indicate instruction address

Data\_Bus – 16-bit output storing the instruction retrieved from ROM

Reading From ROM

The ROM uses the Program Counter input in order to select the address of the an instruction.

PSW





Description of I/O

PSW\_P\_IN – 1-bit input of the PSW register’s privileged bit

PSW\_N\_IN – 1-bit input of the PSW register’s negative bit

PSW\_Z\_IN – 1-bit input of the PSW register’s zero bit

Read\_Data – 16-bit input to the multiplexer

PSW\_Overwrite – 1-bit input used to control the multiplexer

Privilaged\_Clock – 1-bit input of the privileged clock signal

Clock – 1-bit input of the clock signal

PSW\_P\_OUT – 1-bit output of the PSW register’s privileged bit

PSW\_N\_OUT – 1-bit output of the PSW register’s negative bit

PSW\_Z\_OUT – 1-bit output of the PSW register’s zero bit

Choosing PSW Register’s Input Data

The zero-state of the multiplexer assigns a 16-bit output containing the P, Z, and N values to be written to the PSW register. This multiplexer output is only used when a single bit of the PSW is to be written to. The HIGH-state of the multiplexer assigns a 16-bit output to the PSW register. This value is only selected when P, Z, and N must be written to at once. The selector bit, PSW\_Overwrite, for the multiplexer is HIGH if the PSW’s current P, Z, and N values must be overwritten at once, and the zero-input state is used if the PSW’s current P, Z, and N bits are to be used.

Control



Description of I/O

ExceptionFound – 1-bit input indicating whether an exception has been flagged in the circuit

Clock – 1-bit input clock signal

Step – 2-bit input indicating which step of the exception handling process the circuit is on

PSW\_P\_OUT – 1-bit input indicating the status of the PSW’s privileged bit

OpCode – 4-bit input indicating which instruction the circuit is handling

PCV – 1-bit output indicating the status of whether or not a program check violation has been found

ALUSrc – 4-bit output indicating where the ALU should be selecting the operands from

Branch – 1-bit output indicating if a B type instruction is being used

IsSingleReg – 1-bit output indicating if a single register instruction is executing

M\_Type – 1-bit output indicating if an ADDM or SUBM instruction is being executed

M\_Op – 1-bit output indicating if a SUBM instruction is being executed

JSR\_Op – 1-bit output indicating if a JSR instruction is being executed

Timer\_Overwrite – 1-bit output indicating whether or not the current timer value should be enabled to be overwritten

PSW\_Overwrite – 1-bit output indicating whether or not the current PSW value should be enabled to be overwritten

MemWrite - 1-bit output indicating whether or not the current RAM value should be enabled to be overwritten

RegWrite - 1-bit output indicating whether or not the current selected register value should be enabled to be overwritten

MemtoReg - 1-bit output indicating whether or not the current memory value should be written back to the selected register

ALUOp – 4-bit output indicating which ALU operation will be required for the current instruction