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**

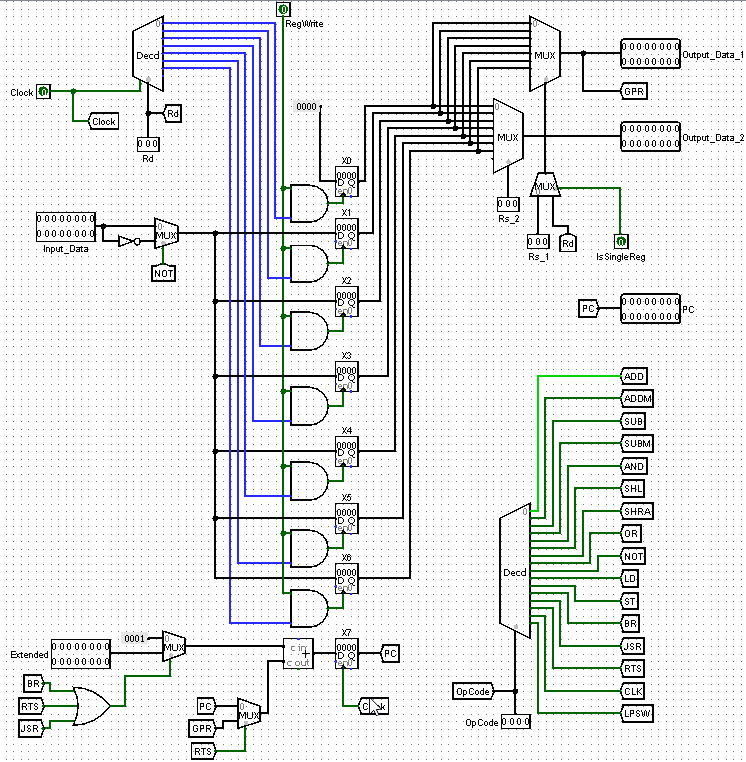
**PERSONNEL3**

**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.

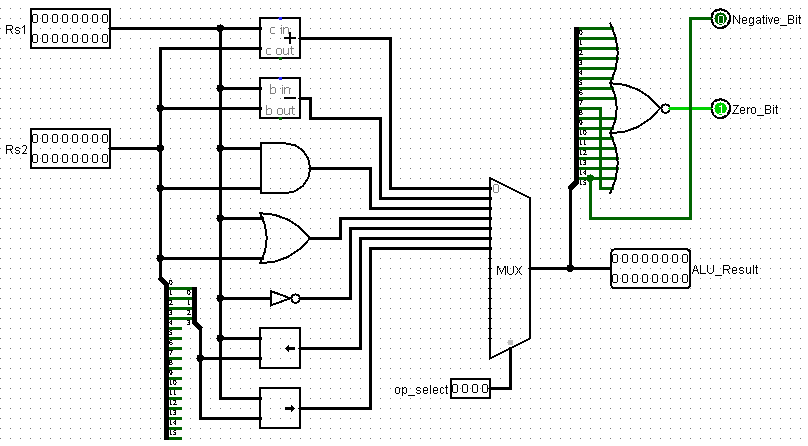
Reading Data from a Register

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.

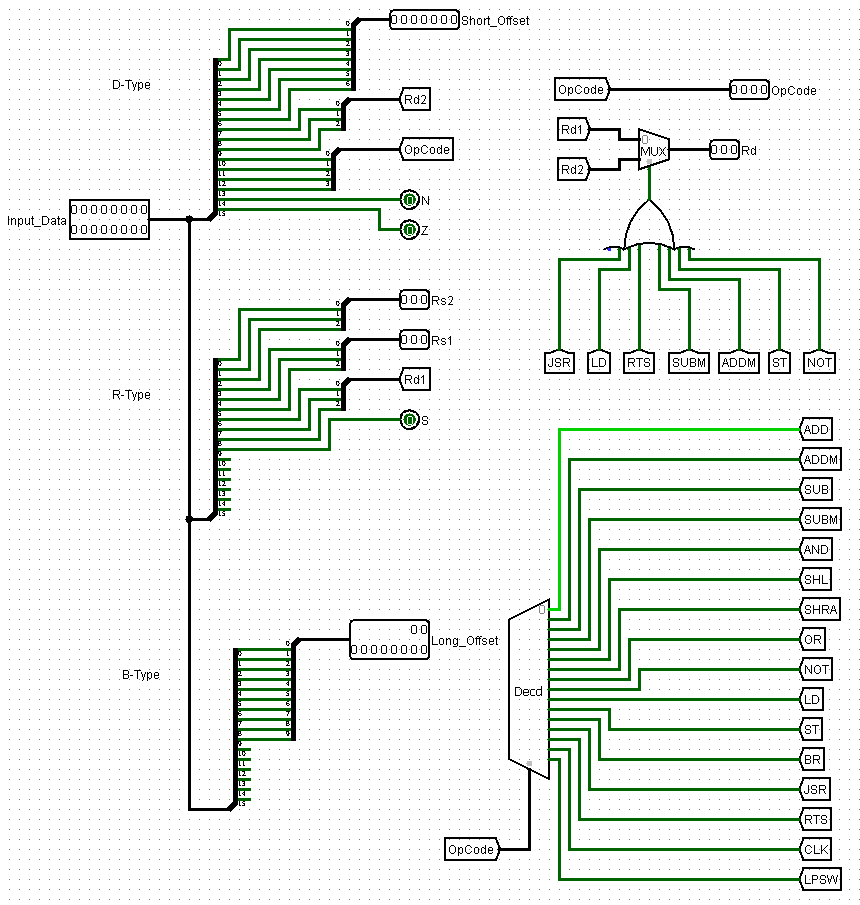
Negative Bit Detection

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**





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 enabled

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

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

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.

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.