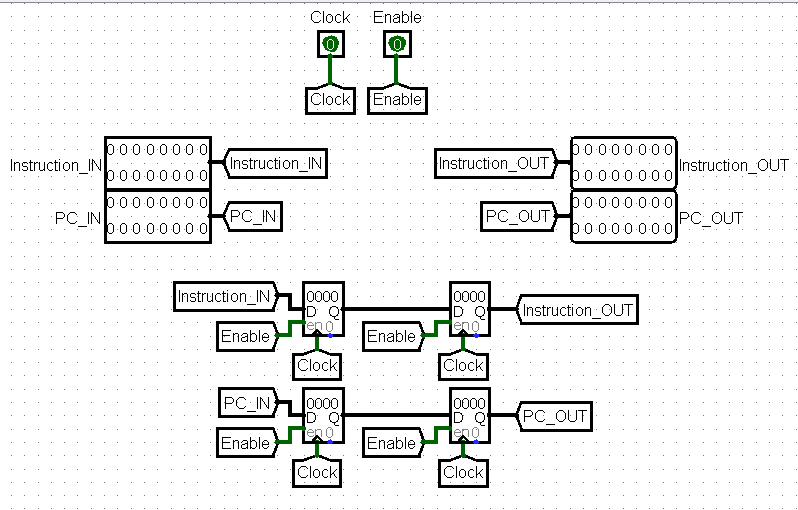
Project Title: 16-bit Pipelined Processor Design

Team Members: Kyle Johnson and Sandro Leon

Class: EECE 3026

Date Submitted: 8-3-16

**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.



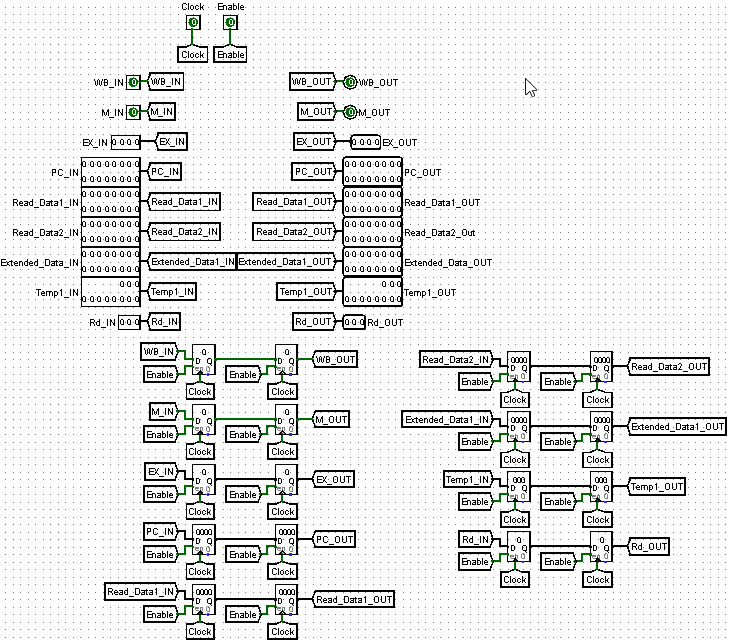
Writing Data to a Register

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.



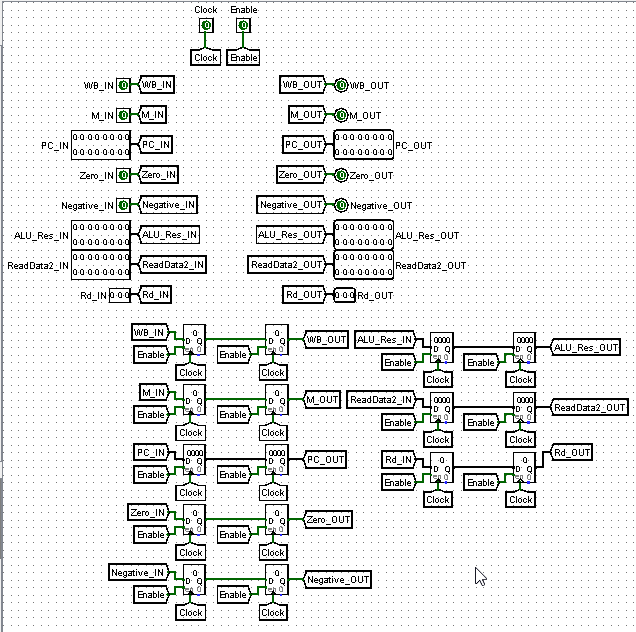
Writing Data to a Register

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.



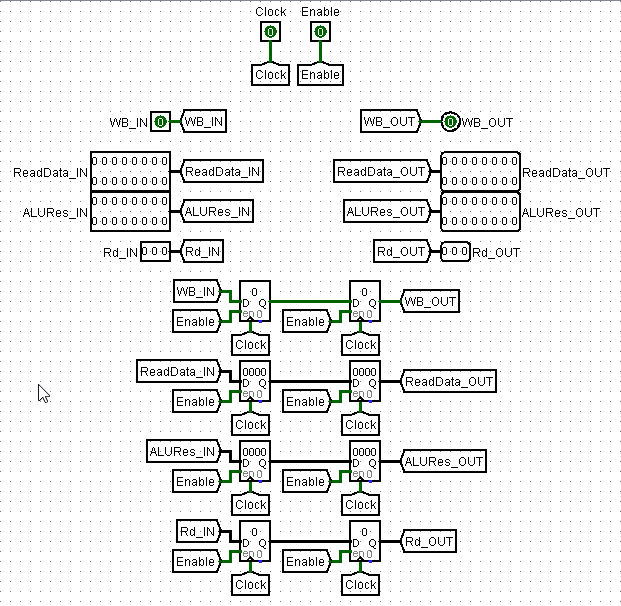
Writing Data to a Register

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.



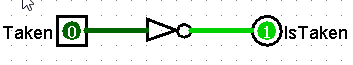
Writing Data to a Register

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.

**1- Bit Branch Prediction**





Description of I/O

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

IsTaken– 1-Bit output used to predict the branch outcome

Summary of Circuit Functionality

The Branch Prediction is used to reduce the amount of hazards encountered by reducing the time taken to flush stages when the branching instruction is taken. For a 1-bit branch prediction, the prediction persists of the input Taken which is the record of whether the prediction was taken, and the output IsTaken is the opposite of the previous record.

Not Taken

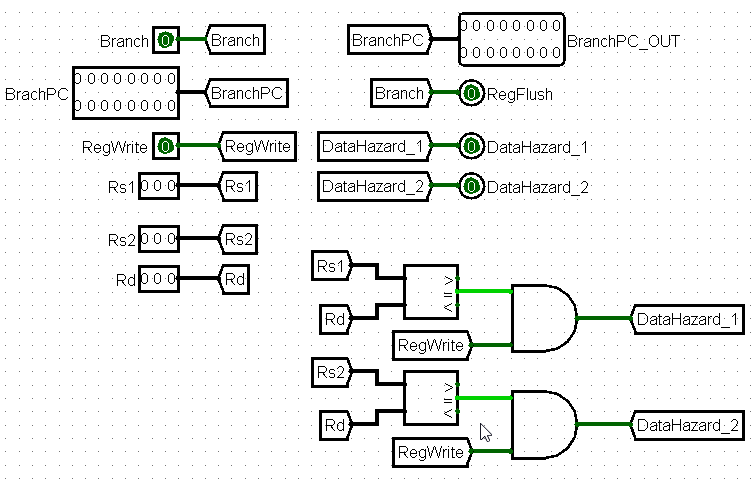
Taken

Predict

Not Taken

Predict Taken

**Hazard Detection Unit**

****

|  |  |
| --- | --- |
| **Hazard Detection Unit** | |
| **Input(s)** | **Output(s)** |
| Branch | BranchPC\_OUT |
| BranchPC | RegFlush |
| RegWrite | DataHazard\_1 |
| Rs1 | DataHazard\_2 |
| Rs2 |  |
| Rd |  |

Description of I/O

Branch – 1-Bit input of whether or not a Branch has been detected

BranchPC– 16-Bit input of the Branch target PC value

RegWrite – 1-bit input of the RegWrite control signal

Rs1 – 3-Bit input of Rs1

Rs2 – 3-Bit input of Rs2

Rd – 3-Bit input of Rd

BranchPC\_OUT– 16-Bit output of the Branch target PC value

RegFlush – 1-Bit output controlling whether or not to flush the control bit of the pipeline buffers

DataHazard\_1 – 1-Bit output indicating if a data hazard involving Rs1 will occur

DataHazard\_2 – 1-Bit output indicating if a data hazard involving Rs2 will occur

Summary of Circuit Functionality

Incorporating a pipelined architecture creates many unforeseen difficulties. Things like data, structural, and control hazards may occur and create erroneous results. A Hazard Detection Unit is implemented in order to handle these potential hazards.

Detecting Data Hazards

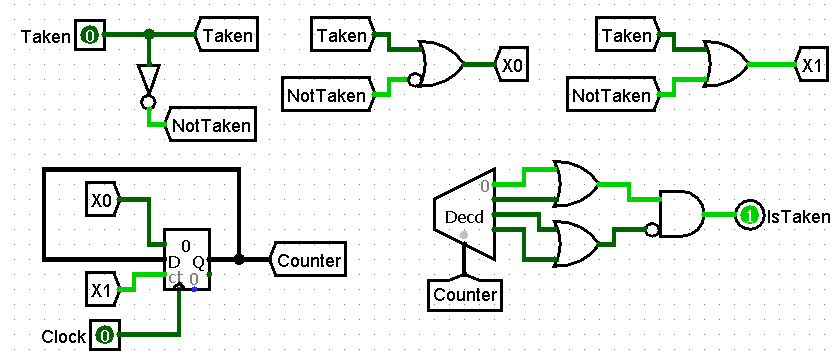
If the Rd register coming out of the ID/EX stage is the same as either the Rs1 OR Rs2 register coming out of the IF/ID register, a potential for a data hazard too occur may emerge. This will only become an issue if the given instruction involves writing data into the register file. This means that Rd must equal Rs1 (or Rs2) AND the RegWrite control signal must be high in order to trigger the Hazard in the Unit.

Detecting Control Hazards

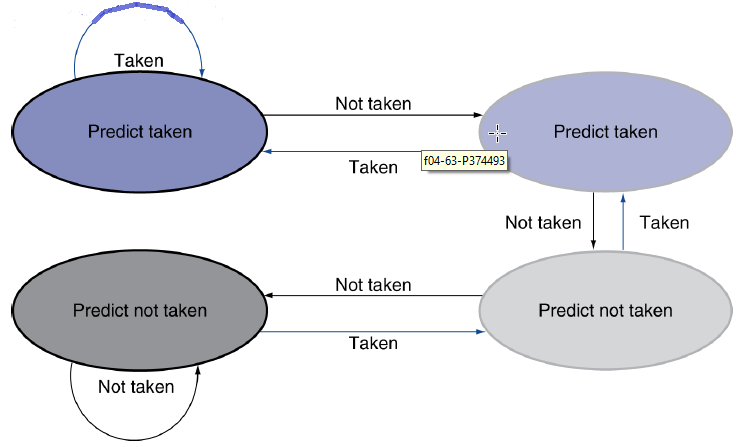
The Hazard Detection Unit will receive an input indicating if a branch command is to be taken or not. If the input is HIGH, a control signal will be sent out of the unit indicating to a multiplexer that the appropriate pipeline registers must have their control signals flushed (set to 0).

**Optimization**

2-Bit Branch Predictor

****



****

Description of I/O

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

Taken – 1-Bit input of the Branch result computed by the ALU.

IsTaken– 1-Bit output used to predict the branch outcome

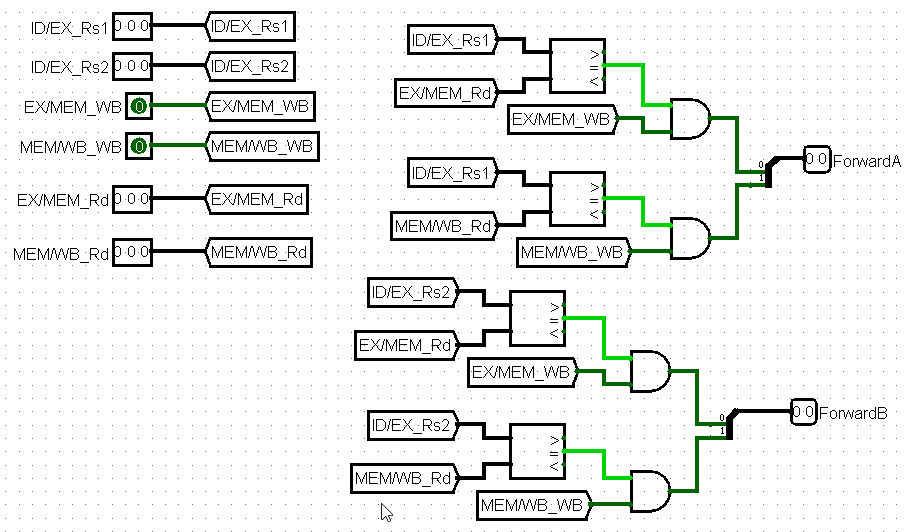
Summary of Circuit Functionality

The Branch Prediction is used to reduce the amount of hazards encountered by reducing the time taken to flush stages when the branching instruction is taken.

Determining IsTaken

A counter register is used to either increment or decrement from 0-3. If Taken is LOW (Branch Not Taken), The counter is incremented (Moved to the next state). If Taken is HIGH (Branch Taken), then the counter is decremented. A decoder then is used to which state it is in. If it is in state 00 or 01 (Predict Taken) then the IsTaken output is HIGH. If it is in state 10 or 11 (Prediction Not Taken) then IsTaken is LOW.

Data Forwarding Unit

****



Description of I/O

ID/EX\_Rs1 – 3-Bit input of Rs1 from the ID/EX Pipeline Register

ID/EX\_Rs2 – 3-Bit input of Rs2 from the ID/EX Pipeline Register

EX/MEM\_WB – 1-Bit input of WB control bit from the EX/MEM Pipeline Register

MEM/WB\_WB – 1-Bit input of WB control bit from the MEM/WB Pipeline Register

EX/MEM\_Rd – 3-Bit input of Rd from the EX/MEM Pipeline Register

MEM/WB\_Rd – 3-Bit input of Rd from the MEM/WB Pipeline Register

ForwardA – 2-Bit output used to as a choose the default ALU operand 1 (00), the forwarded EX/MEM\_Rd value as operand 1 (01), or the forwarded MEM/WB\_Rd value as operand 1 (10)

ForwardB – 2-Bit output used to as a choose the default ALU operand 2 (00), the forwarded EX/MEM\_Rd value as operand 2 (01), or the forwarded MEM/WB\_Rd value as operand 2 (10)

Summary of Circuit Functionality

The data forwarding unit is used to address data hazards on the fly. Instead of stalling, it solves some data dependencies by forwarding data from one stage directly to the ALU.

Forwarding Data from The EX/MEM Pipeline Register to the ALU

When it is detected (via a comparator) that ID/EX\_Rs1 and EX/MEM\_Rd both point to the same register and EX/MEM\_WB is HIGH, the output ‘ForwardA’ will hold a binary value of 01. The ‘ForwardA’ value will be used as a multiplexer’s selector input on the outer circuit.

When it is detected (via a comparator) that ID/EX\_Rs2 and EX/MEM\_Rd both point to the same register and EX/MEM\_WB is HIGH, the output ‘ForwardB’ will hold a binary value of 01. The ‘ForwardB’ value will be used as a multiplexer’s selector input on the outer circuit.

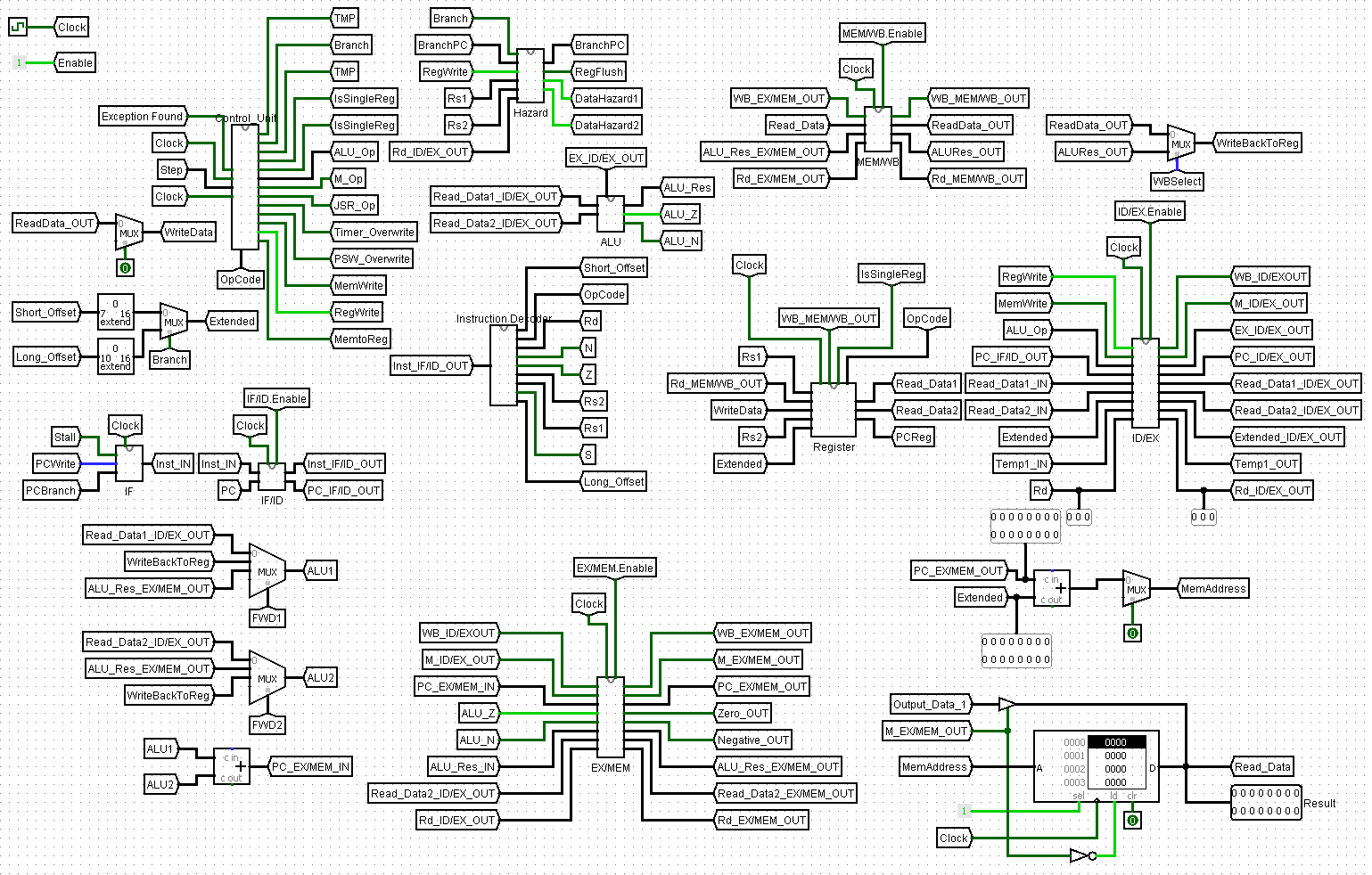
Forwarding Data from The MEM/WB Pipeline Register to the ALU

When it is detected (via a comparator) that ID/EX\_Rs1 and MEM/WB\_Rd both point to the same register and MEM/WB \_WB is HIGH, the output ‘ForwardA’ will hold a binary value of 01. The ‘ForwardA’ value will be used as a multiplexer’s selector input on the outer circuit.

When it is detected (via a comparator) that ID/EX\_Rs2 and MEM/WB \_Rd both point to the same register and MEM/WB \_WB is HIGH, the output ‘ForwardB’ will hold a binary value of 10. The ‘ForwardB’ value will be used as a multiplexer’s selector input on the outer circuit.

**Appendix**

Final Circuit Overview



* Reference Project 1 for Instruction Decoder, Register File, and ALU documentation