ECS 154B – Winter 2014 – Lab 5
Due by 11:55 PM on Friday March 14, 2014

Objectives

- Handle Control hazards through stalling and forwarding
- Add limited Exception handling support to your 5 stage pipelined CPU

1 Control Hazards

Start by adding BEQ and J instruction handling to your 5 staged pipelined CPU in lab 4. Read After Write hazards can occur with BEQ and J. The following hazards must be solved with Forwarding.

`add $4,$5,$6`	`add $4,$5,$6`	`lw $10, 0($0)`	`lw $10, 0($0)`
`add $8, $9, $10`	`add $8,$9,$10`	`add $4,$5,$6`	`add $4,$5,$6`
`beq $0, $4, addr`	`j $4`	`add $8,$9,$10`	`add $8,$9,$10`
		`beq $0,$10,addr`	`j $10`

The following hazards should be resolved with stalls.

`add $4,$5,$6`	`add $4,$5,$6`	`lw $10, 0($0)`	`lw $10, 0($0)`	`lw $10, 0($0)`	`lw $10, 0($0)`
`beq $0, $4, addr`	`j $4`	`add $4,$5,$6`	`add $4,$5,$6`	`beq $0,$10,addr`	`j $10`
		`beq $0,$10,addr`	`j $10`		

1.1 Branch Prediction

In order to reduce the number of stall cycles in our CPU we will be using a branch not taken prediction strategy. This means that if a branch is taken we will need to provide hardware to squash the incorrectly predicted instructions. For example

`beq $0,$0,addr`	
`add $4,$5,$6`	`This instruction should be squashed.`

2 Exception Support

You will be adding support for two types of exceptions: overflow and unrecognized instruction. The book describes how to handle an overflow exception in some detail, so this might be a good starting point. You will need to add two registers: a 32 bit EPC register that holds [PC of exception]+4, and a 32 bit Cause register that holds the cause of the exception. If the exception was caused by overflow, Cause should be set to 0x00000001, if it was caused by an unrecognized instruction, Cause should be set to 0x00000002. These registers should only change when an exception occurs.

When an exception occurs, all instructions further down the pipeline should complete successfully. The excepting instruction itself should not complete or write back to the register file. None of the instructions further up the pipeline should complete or write back to the register file.

2.1 Overflow

There is an Overflow output on your ALU already, which will be 1 any time the ALU is performing an addition that yields overflow. If the instruction in the execute stage causes overflow, your CPU must flush the appropriate stages, set the EPC and Cause register, and set the PC to 400 (decimal), which corresponds to instruction (word) number 100 in your instruction memory.

2.2 Unrecognized Instruction

Your CPU must analyze each instruction during the decode stage to determine if it is an instruction that is supported by your CPU. If it is not, flush the appropriate stages, set the EPC and Cause register, and set the PC to 400 (decimal).

Notice that there is no way to access the EPC or Cause register, or a way to jump back to the excepting instruction, so we cannot actually do much in the way of exception handling given the limitations of our CPU. A convenient way to "end things" is to put `64 : 1000FFFF;` in your instruction memory. This corresponds to `LOOP: beq $0, $0 LOOP` located at instruction 100 in memory.

3 Output Pins

Please add the following output pins to your circuit:

1. `IF_Flush`: indicates IF/ID pipeline register should be flushed
2. `ID_Flush`: indicates ID/EX pipeline register should be flushed
3. `EX_Flush`: indicates EX/MEM pipeline register should be flushed
4. `Overflow`: indicates that the instruction currently in execute is causing overflow
5. `EPC[31..0]`: PC+4 of the excepting instruction
6. `Cause[31..0]`: Cause of the last exception

4 Grading

Grading is divided as follows:

1. `Forwarding`: 30
2. `Stall and Forwarding`: 30
3. `Exception Handling`: 30
4. `Q n A`: 10

Make sure to include the following in your README:

- names and SIDs
- A description of how you tested the exception support that you added to the CPU. This is very important because if you've not tested it, and not included this information, your circuit will not be evaluated.
- A description of how you implemented your exception detection, which stages get flushed for each exception.
- An indication of any difficulties you faced, varying in length from one sentence for no problems to multiple paragraphs if you faced serious difficulties. For example, if you have a known issue remaining in your final submission, any details you can give me about what the problem is will help me be able to give you partial credit.