BR Test:

1. Want to be sure NZP is set from the immediately previous instruction
2. Test BR as first instruction, when NZP may not be set.

LDR:

1. Test dozens of back-to-back LDRs to ensure that any delay doesn’t accumulate and cause problems.

JSR

1. Test JSR within a JSR and make sure it can still return up the call stack correctly.

FUN BUGS

1.

A taken branch asserts the taken signal for exactly one cycle, no matter the state that the rest of the pipeline is in. If the IF stage is already fetching an instruction when this happens, and there is a cache miss such that the memory does not respond by the end of the clock cycle, then the taken signal will go low and the taken address will then not be written. The pipeline continues as if the branch was determined as not taken.  
  
With control hazard correction implemented in our pipeline, a taken branch must flush the pipeline in the IF, ID, and EX stages so that they have NOPs, and then it must place the branched-to address into the PC register for loading in the next cycle. The problem is that because there was a cache miss, a flush will not be able to stop the flushed address from being fetched. Once the branched-to address begins being fetched, the original instruction that caused the cache miss will at some point finish its read, and from the datapath’s point of view, the resulting memory response signal indicates that the *branched-to* address just finished, when in reality the branched-to address is still being read in parallel with the flushed instruction that was just returned.  
  
Possible solution: Save the branched-to address into the PC whenever the BRTaken signal is high. If memory has not responded by end of the clock cycle, save BRTaken into a *register*; else if memory has responded, make sure it stores 0. If after not responding immediately, the taken signal persists until the memory signal responds, in which case the register should flip to 0. The write signal for this register is (BRtaken OR TakenRegOut). The input port is ((BRtaken AND (NOT IMRESP\_H)) OR (TakenRegOut XOR MRESP\_H)). The purpose of storing this taken signal in a register is to prevent the PCreg from writing on a clock edge. So the new write signal for the PCreg is ((StageLoad AND IMRESP\_H AND (NOT TakenRegOut)) OR BRtaken).

|  |  |  |
| --- | --- | --- |
| BRtaken, TakenRegOut | MRESP\_H | TakenRegInput |
| 0,0 | 0 / 1 | (input not written) |
| 0,1 | 0 / 1 | 1 / 0 (don’t have a taken signal asserted, waiting for mem to finish) |
| 1,0 | 0 / 1 | 1 / 1 (taken signal asserted from Branch. make sure it’s written) |
| 1,1 | 0 / 1 | 1 / 1 (this state can’t happen because of flushing) |

^THIS DID NOT WORK. The address cannot change during a read, otherwise the fetched line from the cache miss will overwrite the line in the new address.

So we need to load the PC at its normal times (on StageLoad and when Memory has responded), but store the next PC address in a latch that can be overwritten at will when there is a new branched-to address that needs to be stored in it. Now, at each clock edge, the PC takes its input from the NextPClatch. Or from PC+2, if there was no branch. ~~Or directly from the branch address if the taken signal is currently high (i.e. the read that will be discarded is not a miss)~~. NextPClatch is not a register so that the PC can read its value on the very next clock edge after the latch is written. As before, NOPs are fed into the pipeline anytime the memory hasn’t returned, but there is an additional case now. When the previous instruction load had a branch at some point during it, also feed in a NOP. This ensures that the read is properly flushed by ignoring that read and waiting for the next cycle, when the PC has the correct address and can issue a new read for that correct address.

^ THIS DID WORK :D:D:D

2. Next big bug...

final reg values for mp3-cp3.asm:

r0: 4x064A (1610)

r1: 4x0608 (1544)

r2: 4x869C (34460)

r3: 4xC34E (49998)

r4: 4xFBE6 (64486)

r5: 4xE1B0 (57776)

r6: 4x7DF3 (32243)

r7: 4x7DF3 (32243)