Project 1
CSCI 540
Advance Computer Architecture
Jordan Manier

In Order pipeline

In an in-order pipeline, instructions are fetched, executed & completed in compiler- generated order. In this design if one of the instructions stalls then this will cause all of the instructions to stall. Instructions are statically scheduled in the In-ordered pipeline
Attached you will find the following files.

- **Project1.cpp**

Written in c++ is my source code for the LC. Also Included is the branch prediction fragment. This code is used to read the .MC file produced by the compiling the .ASM.

- **a.c**

the a.c file was provided to the class. This is used to convert valid assembly code for the LC into machine code.

Simulator Operation

LC is a basic 5 stage pipeline design. It consist of both data forwarding and basic branch predication using a Branch Table and 2-bit state machine for predictions. The pipelines in conducted:

1. Instruction Fetch
2. Instruction Decode
3. Execute
4. Memory
5. Writeback

The all of these pipeline registers are used. IFID: Between fetch and decode IDEX: Between decode and execute EXMEM: Between execute and memory MEMWB: Between memory and write-back WBEND: After writeback to ease data-forwarding

- **The Test Cases**

memory_test

This test consists of a for loop in which regi1 is loaded from memory 11, reg 2 is added with the number in reg 1 and stored to reg 2, and then that is stored to mem 11. Basically, this loop is one big data hazard. We expect that the data will be forwarded when possible and that the system will stall in order to prevent corruption. Additionally, this test tests the sw instruction. When running this program through the simulator we see that data us appropriately forward added between the add and sw instructions resulting in no stall between those instructions. Additionally, the system does stall (and forwards data) between lw and add thus preventing that hazard. Additionally, we see that the branch prediction correctly predicts the branch all but two times like in the previous test because they are similar.

nand_test

I implemented this test case to make sure the nand test work properly. We get the correct value of the nand when executed.

noop_test

This test was designed to test basic functionality of the simulator it also tests
the noop instruction.