ECE 201/401 Project 1

October 10, 2019

1 Description

In this project, you are given a extended version of the 5 stage MIPS pipeline, with 7 'dummy' stages, that do nothing besides passing values onward. In effect, the ID stage receives its instruction 7 stages later than it normally would so that the given pipeline emulates a 12-stage, high-speed pipeline. In order to complete the project, you must add a branch predictor to the pipeline and modify the WB stage so that you can handle mis-predictions.

2 Requirements

To complete the project you must:

1. Implement two conditional branch predictors with the following features:
   (a) **Always Not Taken:** you will predict every branch as not taken and when you find that a branch is taken, you must push the pipeline and continue fetching from the correct branch target.
   (b) **Bimodal Branch Predictor with a Branch Target Buffer(BTB),** with following parameters:
      - 2-bit FSMs (saturated up/down counters)
      - 1024 entries
      - Use delayed updates
      - 1024 entry and 2-way set associative BTB with 4-bytes blocks(one instruction per block)
      - Handle Jumps using BTB

2.1 Extra Credit

For 100% extra credit, you may implement a hybrid branch predictor with the following properties:

1. Global Branch Predictor with 12 bit global history register(GHR)
2. Local Branch Predictor
   - 1024 entry Branch History Table and Pattern History Table
   - Each entry BHT should have 10 bit history
3. Meta Predictor
   - 1024 entries
   - 2-bit FSMs
4. Use delayed updates
5. Handle jumps using the BTB, handle JAL/JR using a Return Address Stack(RAS) with 32 entries.

3 Testing

Testing will be done in the same fashion as in the first project: we will run all tests (asm/cpp) to see that they produce the correct input. Credit will be given for each predictor you have implemented.

Initialize all BTBs, PHTs, and saturated counters to zero. Please report number of branch misses in your report.

4 Getting Started

Download and unpack the project files using the command line below:
tar -xvzf ECE401-Project1-F19.tar.gz
The extracted directory will have the following file structure.

- verilog/
  This contains the verilog design files for the processor. All necessary changes for this project will be made here. Additionally, any additional verilog files you write should be placed here.
- sim_main/
  This contains the c++ source for the simulator that will run your processor. While you do not make any changes to this file, some cursory knowledge of how it works will be useful for the extra credit opportunities.
- tests/
  We provide several tests for you. Tests can be downloaded by, in terminal, switching directory to the same folder as the makefile and typing 'make tests'.

5 Compile

Compilation is made easy by the makefile, allowing you to compile by being in the directory of the makefile and typing 'make'. We encourage you to develop on the ECE cluster machines. You may develop on your personal machines, however if your code does not run on the ECE machines you will receive a score of zero (0).

6 Simulate

Once successfully built, you need to test your system. This can be done by running the executable that's now in your topmost directory 'VMIPS'. You will need to provide VMIPS a file to run as well. It should look something like this.

```
./VMIPS -f tests/cpp/class
```

Additionally, you can provide a number of cycles to run. For example, to run for 12345 cycles.

```
./VMIPS -f tests/cpp/class -d 12345
```

If you find that you are encountering a problem at a specific address, you may also provide a breakpoint to VMIPS:

```
./VMIPS -f tests/cpp/class -b 0x0413ABCD
```

7 Some Advice

You may find the following advise useful for completing the project:

- Start the Project early; right now is a good time. Otherwise you won't be able to finish it before the deadline.
- Many instructions can have side effects that take effect before the WB stage(store/syscalls). Make sure you only change the architectural/memory state on instructions that you know happen.
- You can use the working code from Project 1 to find the correct execution path for your new pipeline
- Display whatever you can to show your project is working (e.g. how the BTB is getting updated and how how the saturation counters are getting updated each cycle)

8 Turn In

You submission should be in the form of a tarball (.tar or .tarbz2). You must include all files necessary to compile and run your processor, and the accompanying report in the tarball. The tarball you create should be submitted via blackboard. If you are working with a partner, the submission should specify who the partner is. Only one submission is needed per group.
You must provide a project report explaining what you did, and explaining how you did it. You should specify what is and is not working. If something does not work, try to explain why. If you have added any changes to sim main, you must justify them in your report. The report must be submitted in PDF format.

9 Plagiarism

What you submit must be your own work (or that of your partner, if you are working in a two-person group). It is permissible to use general code snippets having nothing to do with MIPS or microprocessors (eg: bit twiddling methods or module templates) that are found online as long as you comment the code to attribute its source. You are also permitted (and encouraged) to discuss ideas with other groups, but you must not share code to avoid accidental appropriation.