CS220
Introduction to Computer Organisation
Lab 7

Abhinav Agarwal*

1 Introduction

We are back to SMIPS design. The goal of this lab is to experiment with some local changes in the processor and reason about their performance implications. You have been given a single cycle implementation of the SMIPS processor to try out the modifications. You can use the given code identically to the previous labs, using the `smips` run script.

2 Princeton-style Architecture

Right now, the processor is implemented as a Harvard-style microarchitecture, in that it uses a separate memory for program and data. Most modern machines are implemented in a Princeton-style microarchitecture, in that it uses the same memory for program and data.

The current SMIPS can not be converted directly to use same memory for instructions and data as there is only a single "read" port. Therefore, we first need to have a 2-cycle implementation of SMIPS.

Exercise (15 points): Divide the `doProc` rule into two rules: `doFetch` and `doExecute`. The rule `doFetch` should load the instruction from memory and store it in register `ir`, while the rule `doExecute` should read the `ir`, and decode and execute the instruction. You need to use the following enumerated type to switch between the two states.

```
typedef enum Fetch, Execute State deriving (Bits, Eq);
```

Exercise (10 points): Change the `doFetch` rule to fetch instructions from dMem instead of iMem. Instructions are already loaded in dMem, so just changing one line should be sufficient.

*Adapted by Amey Karkare for CS220

Discussion (5 points): Before actually running the modified design, try to predict how the number of cycles will change. Does the result match your expectations? How can you make it better? Provide your answers in the `discussion.txt` file.