CS220
Introduction to Computer Organisation
Lab 3

Amey Karkare

1 Introduction

This and the next few labs relate to the design of SMIPS processor. The ultimate goal of these labs is to take the provided (incomplete) implementation of the SMIPS processor and complete it. By the end of the course, you will have an implementation of SMIPS where the majority of the functionality is implemented by you. You will be able to run simple assembly programs and C programs compiled using a SMIPS cross compiler and related tools.

The first lab in the series requires you to complete the implementation of Register File and the ALU.

2 Using the provided code

Inside the lab directory, you should find the following files and directories:

```
Exec.bsv      Proc.bsv       RFile.bsv      Types.bsv       share
MemTypes.bsv  ProcTypes.bsv  TestBench.bsv  discussion.txt  smips
```

for this lab, you only have to modify two bsv files: `RFile.bsv` and `Exec.bsv`. Other bsv files are for your reference to understand the design of SMIPS. **Do not** modify other bsv files as the compilation ignores them, and uses appropriate libraries from `share` directory.

Instead of a makefile, we have provided a script file named `smips`. This script will take care of everything from compilling the SMIPS core and cross-compiling the benchmarks for SMIPS, to running the benchmarks. In order to compile the processor and all benchmarks and run them, you can use the following command:

```
./smips -c all -r
```

The `-c` flag specifies which benchmarks to compile, and the `-r` flag tells the script to run the benchmarks that has been compiled. After completing

execution, the script will tell you if your processor has passed the benchmark, and how many cycles it took to execute how many number of instructions.

You can also pick individual benchmarks to compile and run, instead of `all`. You can do this by pointing to the individual benchmark directory. These benchmarks can be found in the directory `~cs220/Labs/LABHOME/programs`. For example, the `multiply` benchmark can be run using the following command:

```
./smips -c ~cs220/Labs/LABHOME/programs/src/multiply/ -r
```

Cleaning the build directory is done using the following command:

```
./smips -x
```

You can learn other arguments that the scripts take, by running:

```
./smips -h
```

3 SMIPS implementation

The provided code implements a single-cycle non-pipelined SMIPS implementation. The Register File module (`RFile.bsv`) and the ALU functions (`Exec.bsv`) are incomplete/erroneous. Hence, the test programs fail when you run them using `smips` script. Your goal is to complete these parts to pass the programs.

Exercise 1 (4 point): Implement the rd1, rd2 and wr methods in `RFile.bsv`. These are the read and write ports of the Register File used by SMIPS.

NOTE: You need to consult the SMIPS manual (provided separately) to implement the ALU operations below.

Exercise 2 (7 point): Complete the implementation of function `alu` in the file `Exec.bsv`.

Exercise 3 (4 point): Complete the implementation of function `aluBr` in the file `Exec.bsv`.

Exercise 4 (10 point): Complete the implementation of function `brAddrCalc` in the file `Exec.bsv`.

Discussion (5 point): Once the tests start passing, note down the individual IPC (Instructions per cycle) obtained for each of the benchmarks ran by the `./smips -c all -r` script in discussion.txt file. (Only the C benchmarks will display the number of instructions executed and the cycles taken, the assembly instruction tests are only used to check correctness.)