# Lab 3

## Files Provided:

A tarball containing two full SMIPS processors, a C compiler for the ISA, test cases, applications, and documentation.

## Introduction

In this lab, you will be working on the code for a SMIPS microprocessor with the intention of learning about how the code works in preparation for the next two labs. SMIPS is a simplified MIPS ISA. You can find documentation for the ISA in the doc-smips-isa directory.

**Part 1:** You will start with a two stage, unpipelined sequential circuit version of the pipeline that can be found in simulator/procs/Unpipelined/2cyc\_harvard.bsv. You will be doing two things: completing the code and adding a third stage (still unpipelined) to the code. You must pass all of the provided tests. You should also write any tests you think are needed to test your code and the modifications you made.

Add a third stage. The three stages are as follows:

1. Fetch

2. Decode, RegisterRead, Execute, Memory

3. Writeback

**Part 2:** We are now going to look at a two stage pipelined version of the SMIPS processor. It is very similar to the unpipelined version from Part 1, but now must deal with mispredicted branches. The code contains support for epochs as described in Lecture 6.

We have carefully removed code relating to epochs and resolving mispredicted branches from simulator/procs/ControlHazardOnly/pcMsg\_epoch.bsv. Search for TBD (standing for “To Be Done”). Fill in the missing code. The code that was removed took 4 lines total (but yours may be longer or shorter based on how you format.)

## How to build:

- First, untar the file from your home directory. You should have a Lab3 directory in your home directory.

- Edit the Lab3/simulator/lib/scemi/project.bld directory. Change the scemi-tcp-port to 3400 + your student account number (ye5 would be 5). Save the file (of course.)

- Create a build directory off of your home directory.

mkdir -p ~/tmp/build\_SMIPS

先输入csh

setenv SMIPS\_HOME ~/Lab3

setenv SMIPS\_BUILD ~/tmp/build\_SMIPS

cd to the utility-bins directory. Then do the following commands that build the entire processor, compiles and runs all programs, including assembly micro-benchmarks.

For Part 1:

“./smips -b ../simulator/procs/Unpipelined/2cyc\_harvard.bsv -c all --run”

For Part 2:

./smips -b ../simulator/procs/ControlHazardOnly/pcMsg\_epoch.bsv -c all --run”

假如出现错误的话，需要在\Lab3\simulator\lib\scemi\TestBench.cpp 里面添加sleep(0)的声明，

#include <unistd.h>

即可通过编译

A trace of the run will be generated and placed in the following files for the assembly programs:

~/tmp/build\_SMIPS/ControlHazardOnly/pcMsg\_epoch.bsv/assembly/<assemblybenchmark>/bsimOut and tbOut.

Or in the following files for the C benchmarks (filters, median, multiply, print, qsort, towers, vvadd):

~/tmp/build\_SMIPS/ControlHazardOnly/pcMsg\_epoch.bsv/<benchmark name>/<assemblybenchmark>/bsimOut and tbOut.

If things get strange, try ./smips -d that deletes the build directory.

## Submitting Your Solution

Provide all of the files that you changed in the same format as Lab 1 (but, of course, using Lab3 as the prefix to the directory.)

结果请看实验截图、结果文字和实验代码。

Directory structure:

Processor skeletons:

simulator/procs/Unpipelined

simulator/procs/ControlHazardOnly

Processor modules (Exec.bsv, Decode.bsv, etc.)

simulator/lib