CS 429, Spring 2012
Optimizing the Performance of a Pipelined Processor
Assigned: March 9, Due: March 22, 11:59PM

Tyler Smith (`tms@cs.utexas.edu`) is the lead person for this assignment.

1 Introduction

In this lab, you will learn about the design and implementation of a pipelined Y86 processor, optimizing both it and a benchmark program to maximize performance. You are allowed to make any semantics preserving transformations to the benchmark program, or to make enhancements to the pipelined processor, or both. When you have completed the lab, you will have a keen appreciation for the interactions between code and hardware that affect the performance of your programs.

More specifically, you will extend the SEQ simulator with two new instructions.

2 Logistics

You will work on this lab alone.

Any clarifications and revisions to the assignment will be posted on the course Web page.

3 Handout Instructions

You can find the Y86 tools on the shedule of the course website.

1. You'll be using the same y86 tools that we used in lab3 (asmlab). So you can just use the same directory that you used for lab3. Or if that's no longer around, redownload it from the course webpage.
2. Then, change to the `sim` directory and build the Y86 tools (if you haven't already):

```
unix>  cd sim
unix>  make clean; make
```

4 The Lab

You will be working in directory `sim/seq` in this part.

Your task in The Lab is to extend the SEQ processor to support two new instructions: `iaddl` (described in Homework problems 4.47 and 4.49) and `leave` (described in Homework problems 4.48 and 4.50). To add these instructions, you will modify the file `seq-full.hcl`, which implements the version of SEQ described in the CS:APP2e textbook. In addition, it contains declarations of some constants that you will need for your solution.

Your HCL file must begin with a header comment containing the following information:

- Your name and ID.
- A description of the computations required for the `iaddl` instruction. Use the descriptions of `irmovl` and `OPl` in Figure 4.18 in the CS:APP2e text as a guide.
- A description of the computations required for the `leave` instruction. Use the description of `popl` in Figure 4.20 in the CS:APP2e text as a guide.

Building and Testing Your Solution

Once you have finished modifying the `seq-full.hcl` file, then you will need to build a new instance of the SEQ simulator (`ssim`) based on this HCL file, and then test it:

- *Building a new simulator.* You can use `make` to build a new SEQ simulator:

  ```
  unix>  make VERSION=full
  ```

  This builds a version of `ssim` that uses the control logic you specified in `seq-full.hcl`. To save typing, you can assign `VERSION=full` in the Makefile.

- *Testing your solution on a simple Y86 program.* For your initial testing, we recommend running simple programs such as `asumi.yo` (testing `iaddl`) and `asuml.yo` (testing `leave`) in TTY mode, comparing the results against the ISA simulation:

  ```
  unix>  ./ssim -t ../y86-code/asumi.yo
  unix>  ./ssim -t ../y86-code/asuml.yo
  ```

  If the ISA test fails, then you should debug your implementation by single stepping the simulator in GUI mode:

  ```
  unix>  ./ssim -g ../y86-code/asumi.yo
  unix>  ./ssim -g ../y86-code/asuml.yo
  ```

- *Retesting your solution using the benchmark programs.* Once your simulator is able to correctly execute small programs, then you can automatically test it on the Y86 benchmark programs in `../y86-code`:

```
unix> (cd ../y86-code; make testssim)
```

This will run `ssim` on the benchmark programs and check for correctness by comparing the resulting processor state with the state from a high-level ISA simulation. Note that none of these programs test the added instructions. You are simply making sure that your solution did not inject errors for the original instructions. See file `../y86-code/README` file for more details.

- *Performing regression tests.* Once you can execute the benchmark programs correctly, then you should run the extensive set of regression tests in `../ptest`. To test everything except `iaddl` and `leave`:

  ```
  unix> (cd ../ptest; make SIM=../seq/ssim)
  ```

  To test your implementation of `iaddl`:

  ```
  unix> (cd ../ptest; make SIM=../seq/ssim TFLAGS=-i)
  ```

  To test your implementation of `leave`:

  ```
  unix> (cd ../ptest; make SIM=../seq/ssim TFLAGS=-l)
  ```

  To test both `iaddl` and `leave`:

  ```
  unix> (cd ../ptest; make SIM=../seq/ssim TFLAGS=-il)
  ```

For more information on the SEQ simulator refer to the handout *CS:APP2e Guide to Y86 Processor Simulators* (`simguide.pdf`).

5 Evaluation

The lab is worth 60 points:

- 10 points for your description of the computations required for the `iaddl` instruction.
- 10 points for your description of the computations required for the `leave` instruction.
- 10 points for passing the benchmark regression tests in `y86-code`, to verify that your simulator still correctly executes the benchmark suite.
- 15 points for passing the regression tests in `ptest` for `iaddl`.
- 15 points for passing the regression tests in `ptest` for `leave`.

6 Handin Instructions

- You will be handing in one file:
  - Part B: `seq-full.hcl`.
- Make sure you have included your name and ID in a comment at the top of each of your handin files.
- To handin your file, go to your `datapath-handout` directory and type:

```
unix>  turnin --submit tms datapathlab seq-full.hcl
```

- You can verify your handin by using:

```
turnin --verify tms datapathlab
```