

CIS 775: Computer Architecture
Fall 2003
Laboratory Assignment #2

Instructor: S. Parthasarathy

Due: Tuesday, Nov. 25th before class.

Purpose

The purpose of this assignment is to understand some of the internals of the simplescalar simulators, learn how to use the gcc compiler for PISA-big and finally gain experience on pipelined processing unit design and its related issues. Like Lab#1, please use a SUN Solaris machine to carry out this lab. Please do not procrastinate, this lab is quite easy but it does involve a fair bit of work.

You will be using the program `matrix.c` located in `/usr/class/cis775` directory for these experiments. Please read the entire assignment before commencing to run experiments (see experiment 5 in objective 3 in particular). This assignment consists of 3 objectives.

1. **Objective 1: Modify `sim-outorder.c`:** Create a directory within your home directory space called *LAB2*. Within this space copy (use recursive copy) the files in the directory:

/usr/class/cis775/newsim/simplesim - 2.0

The modifications you will need to make to this file (`sim-outorder.c`) are so that you can add functionality which will allow you to specify the latency for a particular functional unit (e.g. fp adders). The current implementation of `sim-outorder` does not allow you to specify this off the command line interface. You can either choose to make the corresponding modification each time and recompile the simulator, or you can choose to make the modification in such a way that you can pass the latency arguments to the command-line interface. Which way you do it is immaterial. Note that the first thing you will need to do is to identify where the latency parameters for the functional units are specified. Please identify any changes to the source you make and describe them for full points. You will need to read through Objective 3 in order to state all the changes you made to the source code. [15 points]

2. **Objective 2: Compile `matrix.c`:** This objective is an easy one. Basically you will have to copy `matrix.c` to the above working directory and then compile it using the appropriate compiler located at: `/usr/class/cis775/newsim/bin`. Part of this objective is to figure out which of the several programs available in that location is the compiler you will need. An easy way to figure out if you have got the correct compiler is to run the simulator on the file that is produced to see if it works. [5 points]
3. **Objective 3: Understand various pipeline-related design tradeoffs** The overall objective of this lab is to understand the tradeoffs between different optimizations for this simple benchmark (`matrix.c`). Note that for all experiments below you will need to ensure that the simulator is in the `issue:inorder = true` and `issue:wrongpath = false` mode.

- **Experiment 1:** You are the newest whiz-kid architect hired by your company. Your first job is to evaluate the following hardware configuration on the given program. The configuration is one fully pipelined FP adder/ subtractor with 3 clock cycles latency, one fully pipelined FP multiplier with 4 clock cycles latency, and one separate *unpipelined* divider with 20 clock cycles latency. Note to ensure that you have a separate FP multiplier and divider (the new divider will also include the FP SQRT operation with unchanged latency and initiation interval), you will have to make changes to the simulator source. Use the default latencies for the integer units but configure it so that you have exactly one of each integer unit (ALU, Mult/Div). Report the average stall cycles per instruction for this environment. [5 points]
- **Experiment 2:**
Building on the environment evaluated in experiment 1, you have told your boss that with your logic wizardry you have added enough space on chip for another floating point unit (FPU: one of adder/subtractor, multiplier, divider may be added). You are commissioned to comparatively evaluate these three options and evaluate their impact on overall performance as compared with the baseline performance from experiment 1. Which would you choose and why? [10 points]
- **Experiment 3:** While conducting the previous evaluation you find out that you can reduce the latency of the adder to 2 clock cycles. You also find that you can further **either** reduce the latency of the multiplier to three clocks cycles *or* the divider latency to fifteen clock cycles. You are now asked to evaluate which of these six options (remember you still can choose which FPU to duplicate) is best and why in a manner similar to the previous set of experiments? Use the results from the previous experiment as a basis for comparison. [20 points]
- **Experiment 4:**
Going back in time when you conducted experiment 1, you have discovered that you can partially pipeline the divide unit into two roughly equal stages. The latency remains constant (w.r.t the above is at 20) but the initiation interval is reduced. Compare the results you obtained against the results obtained in experiment 1 and identify the impact of this optimization. [5 points]
- **Experiment 5: Impact of data size** Double the value assigned to the *size* variable in `matrix.c` and recompile and re-execute experiment(s), 1,2,3 and 4 under this new version. Comment on these new results with respect to the previous results. Do the same trends hold? Are there any unexpected results? [40 points]

You will need to submit a report answering all the questions. Use tables where appropriate and when comparing amongst different strategies. You will also need to provide the UNIX pathname for the directory where you did your work so that the TA can go over your modifications for the objectives.