

CIS 570: Advanced Computer Systems

Fall 2009

Simulation Assignment #1: Simple simulations with SimpleScalar Due 11:59 PM, 10/5/09

Please ensure that your name is clearly visible on all submitted material. Electronic submission is required for this assignment. Directions for your deliverables are available in Section 3.

1. Introduction

As discussed in lecture, architectural simulators are used to profile applications and evaluate system designs without the cost of implementing actual hardware. The simulation assignments in this course are intended to give you some familiarity with these useful programs.

The simulator we will use in this course, SimpleScalar, is widely used in computer architecture research. The simulator is written in C and runs on Unix machines. I expect that you will have access to the Linux machines in Dion 305, either directly or through an SSH client (see <http://www.cis.umassd.edu/RemoteAccess.htm> for details). You may also download and run the software on your own machine; please note that SimpleScalar may not have been extensively tested in all environments.

2. Simulations

A. Building SimpleScalar

The first step to this assignment is to download the simulator, compile it, and run some simple tests to make sure it's built correctly. The steps below walk you through this process.

- We're using SimpleScalar version 3.0d, which is available from <http://www.simplescalar.com>. Under "Downloads," click on "Tools," then "simplesim-3v0d.tgz." Accept the terms of use to start the download.
 - Note that if you are remotely accessing a lab machine to do this assignment, you may have to download the archive to your desktop and then FTP it to the lab machine.
- Unpack the SimpleScalar archive by typing:

```
tar -xzvf simplesim-3v0d.tgz
```

The simulator files will be unpacked into the `simplesim-3.0` directory.

- Change directories to `simplesim-3.0` and build the simulators. We'll be using the SimpleScalar Portable ISA (PISA), a 32-bit RISC architecture with 64-bit instructions developed for use with these simulators, so you should configure your simulators to execute PISA binaries as follows:
 - Type `make config-pisa.` This command links architecture-specific files to the general file names used in compilation. For example, the ISA definition file `target-pisa/pisa.def` is linked to the general name `machine.def`, which is referenced in the Makefile.
 - Type `make.` This command builds all of the different simulators in the SimpleScalar toolset. You'll likely encounter a number of warnings in the compilation process, which you can ignore. When you modify source files in later simulation assignments, you simply have to rebuild the simulator(s) using those source files with this command.
 - Type `make sim-tests.` This command runs some short, simple programs and tests their outputs against previously generated reference outputs to ensure that the simulators have built correctly. If the `diff` commands run throughout this process give you any outputs, you may have encountered errors in the build process.

B. Functional simulations

The SimpleScalar toolset contains a number of different simulators. *sim-fast* and *sim-safe* are both functional simulators, one optimized for speed, the other for correctness. You'll use *sim-safe* to run a few simple benchmarks and examine the output.

- Download and unpack the `cis570_sim1.tgz` archive file from the CIS 570 web page. This archive gives you a folder containing a number of pre-compiled benchmarks and their inputs. Note that these benchmarks are pre-compiled for the PISA ISA on little-endian machines. If you happen to be running on a big-endian processor and experience problems, please contact me for the appropriate files.
- The resulting directory contains a README file that tells you how to execute each of the four benchmarks using *sim-safe*. The given command lines redirect the benchmark output into its own file so that it can be compared to a reference output file for correctness. However, the simulator output is displayed in the terminal window. Figure out how to redirect the simulator output into its own file, then run all four benchmarks and submit the simulator outputs.
 - Note that the command lines implicitly assume either that the simulator and benchmark are in the same directory (which is unlikely), or your `simplesim-3.0` directory is on your search path. If neither of those things is true, you're going to have to provide the full path for the simulator executable (i.e., `~/simplesim-3.0/sim-safe`).
- Also, use the simulator output files to answer the following questions:
 1. (5 points) How long does each program take to simulate?
 2. (7 points) What are the dynamic instruction counts for each program?
 3. (8 points) Can we determine the static instruction count for each program? If so, what are those values?

C. Profiling simulations

You may have noticed that the functional simulations don't give much information about the executions of these programs. To get more detail about the benchmark's execution, a profiling simulator can be used. SimpleScalar's profiling simulator is called *sim-profile*. To see what type of profiling *sim-profile* can do, simply run the simulator without any inputs and look at the different options.

- Once again, run the benchmarks and redirect the simulator output to a file. Enable the following *sim-profile* options: instruction class, instruction, branch instruction, and address mode profiling.
- Answer the following questions:
 4. (7 points) Which instruction class is the most common for each benchmark, and what percentage of the dynamic instructions does that class comprise?
 5. (8 points) List the 5 most common instruction types and their percentages for each benchmark.
 6. (8 points) Briefly describe the branch and address mode profiles—their purpose and the data generated for each benchmark.
 7. (7 points) One option I didn't ask you to simulate is text address profiling, primarily because the output files it generates are extremely long. What could you learn from such a profile?

D. Timing simulations

For detailed timing information and cycle-accurate simulations, SimpleScalar uses a program called *sim-outorder*. This simulator provides an in-depth hardware model for a general microprocessor, which can be used to model different hardware designs. We will explore these capabilities more fully in future assignments.

- As in the previous steps, simulate each benchmark and save the output files. Note that these simulations may take long periods of time—make sure you give yourself enough time to complete them before the assignment is due! To make sure you complete these, you may want to actually start with this step.
- Answer the following questions:
 8. (7 points) The downside of detailed simulation is the significant amount of time it takes to simulate each benchmark. How much slower is *sim-outorder* as compared to *sim-safe* for each benchmark?
 9. (7 points) Can you suggest a method for improving the performance of timing simulations?

3. Deliverables

You should submit the following documentation through Portal as proof you've completed the assignment:

- All simulation output files, clearly labeled with the program and simulator used. (3 points per submitted file) Follow the convention below:
 - All output files should be combined, along with a “README” file describing the various outputs, into a single archive. You should not submit each file separately.
 - When choosing your archive format, please use a format widely available on UNIX machines: .zip, .tar, or .tgz are my three preferences. Please do not use the .rar format.
- Your answers to the questions asked in parts B, C, and D of the assignment. Any common document format (e.g. Word .doc/.docx, .txt, .rtf) is acceptable.