

ECE 5752 - Fall '02

Homework Exercise

Introduction to SimpleScalar 3.0c

Assigned: Thursday 9/12

Due: Tuesday 9/24

All work should be done individually, no group work allowed for this assignment

Purpose:

The purpose of this exercise is to introduce the students to the SimpleScalar 3.0c architectural simulation environment. This introduction will include building to tools from available FTP source code, and running some of the tools on provided benchmarks and input sets. No modification of the source code is required for this assignment, though changes may be required for environmental reasons.

Problem Specification:

For completion of this assignment, each student will be required to download, build, test, and run the SimpleScalar 3.0c architectural simulation tool upon programs which are not part of the SimpleScalar distribution. You will have provided to you a set of PISA (little-endian) benchmark binaries, and the input set for those binaries. As described in class - the SimpleScalar architectural simulation tool allows you to model a system architecture. When modeling an architecture, it is of little interest to model an idle system, thus a workload must be used to exercise the model. The benchmarks and associated input sets serve as a workload in this project.

It is recommended that you follow the suggested procedure, but if you are able to complete the requirements of the problem, the procedure is secondary.

Suggested Procedure:

It is suggested that the following procedure be followed on one of the COLOR machines in the btdpool, or any available Linux machine. Beware, the btdpool is not designed for, and is rarely used in a classroom environment - if there are any machine failures or other issues to report, please direct them to *btdavis@mtu.edu*.

1. Download the SimpleScalar 3.0c distribution from the ftp site below.
<http://www.simplescalar.com/tools.html>
`/home/Public/simulators/simplesim-3v0c.tgz`

2. Untar and build the SimpleScalar 3.0c package following the instructions contained in the README file contained in the root directory of the distribution.

If using Linux on an x86 machine, the PISA-little architectural choice is recommended

3. You can access the benchmarks for the pisa-little architecture either from the webpage above, or if you are running on the btdpool machines, the workload benchmarks are available in the shared directory below.

/home/Public/benchmarks/ss_bin_little

4. Execute *sim-outorder* upon one of the benchmarks for which an invocation method is shown below. Verify the operation of the simulation toolset. It is recommended - but certainly not necessary - that you follow the below convention for invoking the simulator. Simulation invocations can be (a) manually input to the command line, (b) static lines in a shell script or (c) dynamically produced by a shell/perl script. Some benchmark/input_set combinations which are known to work are included in the table below. Execution of the simulator may take a long time, depending upon the benchmark and input set, so execution in the background is advised.

Benchmark	sample invocation
compress.ss	sim-outorder \ /home/Public/benchmarks/ss_bin_little/compress95.ss \ < /home/Public/benchmarks/benchData/compress/data/test/input/test.in
mcf	sim-outorder \ /home/Public/benchmarks/ss_bin_little/mcf00.ss \ /home/Public/benchmarks/benchData/mcf00/data/test/input/inp.in
cc1.ss	sim-outorder \ ss_bin_little/cc1.ss benchData/cc1/data/test/input/cccp.i
go.ss	sim-outorder \ /home/Public/benchmarks/ss_bin_little/go.ss 9 9 \ /home/Public/benchmarks/benchData/go/data/test/input/null.in
vpr	sim-outorder \ ss3.0_pisa_big_bin/vpr \ benchData/vpr00/data/test/input/net.in \ benchData/vpr00/data/test/input/arch.in \ benchData/vpr00/data/test/input/place.in \ vpr00.route.out

nice \

```
(sim_path/sim-bin {sim_params} \  
bmark_path/benchmark_bin {bmark_parms i.e. input_set_path/input_set} \  
> output_file) &> err_out_file
```

The above is known to work using tcsh, YMMV if you choose another shell (i.e. csh, bash, etc).

5. Generate a dump-config file for sim-outorder. Using your favorite editor, create (5) unique configuration files. Each of these five configurations represents one of the machines for which statistics are given below. When parameters are not given in the table below, leave them as they

Configuration Name	Cache Parameters
None	No Cache
Small	L1U : 4KB, 2-way Set Associative, 32 Byte Linesize, Random Replacement, 1-cycle latency
Medium	L1I : 8KB, Direct Mapped, 64 Byte Linesize, 1-cycle latency L1D : 8KB, 2-way Set Associative, 32 Byte Linesize, LRU Replacement, 1-cycle latency L2U : 64KB, 4-way Set Associative, 64 Byte Linesize, LRU Replacement, 6-cycle latency
Large	L1I : 16 KB, Direct Mapped, 128 Byte Linesize, 1-cycle latency L1D : 16 KB, 2-way Set Associative, 64 Byte Linesize, LRU Replacement, 1-cycle latency L2U : 256KB, 2-way Set Associative, 128 Byte Linesize, LRU Replacement, 6-cycle latency
Excessive	L1I : 32 KB, 2-way Set Associative, 128 Byte Linesize, Random Replacement, 1-cycle latency L1D : 64 KB, 2-way Set Associative, 64 Byte Linesize, LRU Replacement, 1-cycle latency L2I : 256 KB, 4-way Set Associative, 256 Byte Linesize, LRU Replacement, 4-cycle latency L2D : 512 KB, 8-way Set Associative, 128 Byte Linesize, LRU Replacement, 6-cycle latency

are in the default configuration generated by -dumpconfig.

6. Run a (3) simulations for each of your (5) configuration files. You may choose a benchmark & input set of your liking, provided that (a) the benchmark output shows correct operation and (b) at least 1 Million instructions are simulated. It is recommended that you use a script to run these benchmarks. Be aware, some benchmarks will take multiple hours to simulate. (15) such simulations may require DAYS.
7. Generate the graphs required for the deliverables.

Considerations:

1. Describe the difference between the multiple simulation tools (sim-*), why is more than one necessary?
2. Describe an appropriate usage of one or more of the SimpleScalar simulation tools in an industrial ASIC design environment.
3. After examining the output files of the SimpleScalar simulation runs (deliverable # 2) how would you expedite the generation of the graphical representations of this data?
4. Examining the performance of these five cache systems for a “default” processor - how significant is the impact of cache upon simulated microprocessor performance?
5. SimpleScalar is relatively limited in the functionality of it’s cache simulator, yet the parameters to specify a cache hierarchy are quite complex. If you were to add victim caches and stream buffers to the simulation environment, how would you propose that this increased functionality be integrated into the a parameters contained in the configuration file?

Deliverables

1. Assignment report, in an appropriate format. Including answers to the Consideration questions above. Reports should include: Purpose, Procedure - including suggested revisions or issues which caused problems, Results, Conclusions, answers to Consideration questions, all Deliverables and whatever else you feel is appropriate.
2. An edited (to one page) configuration file for each of the (5) configurations, showing those parameters which were changed or unique to each of the (5) configurations.
3. An edited (to one page) output from each of the (15) simulation runs (3 benchmarks, 5 configurations) showing the following statistics:

Mem_size		
sim_cycle	sim_num_insn	sim_num_refs
sim_num_loads	sim_num_stores	sim_num_branches
sim_inst_rate	sim_CPI	ruu_occupancy
and for all caches		
*.accesses	*.hits	*.miss_rate

And anything else you feel is relevant

4. A graphical representation of the output from each of the (15) simulation runs. The X-axis should be the memory model-size of the simulated processor. Three lines/groups on the graph should be each of the three benchmarks. Using this format, provide graphs for each of the following:
 - a). Execution Time (sim_cycle)
 - b). Cycles per instruction (sim_CPI)
 - c). ruu_occupancy