

1 About SimpleScalar

This document is a very brief introduction to the simplescalar tools as they are installed locally. Be sure to read through the user's guide, which contains much more detailed information about SimpleScalar.

- The original SimpleScalar files are available at <http://www.simplescalar.com/>
- SimpleScalar is now installed locally in:
 - `/cse/classes/cmpe202/SimpleScalar/` for Solaris
 - `/cse/classes/cmpe202/SimpleScalar-linux/` for Linux
- Code has been tested (to some extent) on sundance, emperor, & fyglia
- Let me know about problems running it (stuart@soe)
- There is now a class newsgroup for simplescalar questions and answers: `ucsc.class.cmpe202`. Please ask questions and share ideas there!
- Important subdirectories of `/cse/classes/cmpe202/SimpleScalar`:
 - `simplesim-2.0` – code for several simulators. This is the code you will modify for your own purposes!
 - `bin` – binaries, including the `ssbig-na-sstrix-gcc` compiler, which will compile C code to PISA (portable instruction set architecture) instructions which run on the simulator programs.
 - * To compile your code into files that the SimpleScalar simulators can execute:

```
% /cse/classes/cmpe202/SimpleScalar/bin/ssbig-na-sstrix-gcc  
myprog.c
```
 - * Please note: if running on a little-endian machine (i.e. Intel), use the versions of files with `little` in the name rather than `big`, e.g. `sslittle-na-sstrix` vs. `ssbig-na-sstrix`.
 - `instructbench` – precompiled “instructional benchmarks” – just as useful as the SPEC benchmarks for testing your code.
 - `f2c` – Fortran-to-C translator, useful if you want to compile some f77 programs to run on the simulators.

2 Installation

- Avoid installing anything if you can avoid it; small differences between machines can cause problems.
- My suggestion: install/compile **only** the `simplesim-2.0` directory in your own SOE disk space, as follows

```
% cd
% cp /cse/classes/cmpe202/SimpleScalar/simplesim-2.0 ss
# linux: /cse/classes/cmpe202/SimpleScalar-linux/simplesim-2.0
% cd ss
% make # it's already made, but you'll want to
# re-'make' after you make changes
% sim-outorder tests/bin.big/test-math
# linux x86: sim-outorder tests/bin.little/test-math
```

- Use any other files, such as the gcc PISA compiler, from the class directories (probably a good idea to add the `SimpleScalar/bin` directory to your path)

3 Running a simulator

- The simulators are the programs in the `simplescalar-2.0` directory starting with “sim-”, like `sim-fast` and `sim-outorder`. Running one of the simulators without any command-line parameters gives you important information on the simulator and how to use it.
- Several different simulators exist; the most complex, and the one you'll be modifying, is `sim-outorder`, the least complex is probably `sim-safe`.
- Suggestion: start by figuring out generally how `sim-safe` works, move on to `sim-outorder` later.

Example of how to run the `sim-outorder` simulator from your local `simplescalar-2.0` (or “ss”) directory:

```
% ln -s /cse/classes/cmpe202/SimpleScalar/instructbench ib
% more ib/README
% sim-outorder ib/anagram.pisa-big ib/words < ib/anagram.in
> OUT
```

Note: these “instructbench” programs are nice, but they only exist for running on a big-endian architecture (e.g. sundance & moondance). You can also compile your own programs to run (with the appropriate SimpleScalar version of gcc). You may also compile the SPEC benchmarks from the `/cse/classes/cmpe202/SPEC*` directories; be warned, this may require some work.

4 What next?

- Experiment with other programs, other simulators, simulator options...
- **Look at the documentation**, it can be very helpful:
 - User’s Guide (v2)
 - Hacker’s Guide & Tutorial
 - .h interface files