Preliminary Assignment

ECE 154B
COMPUTER ARCHITECTURE
The ZEN of Simulator Design

- Performance: Speeds up design cycle
- Flexibility: Maximizes changes that can be made in design
- Detail: Makes simulation more realistic, minimizes risk

Pick Two

The SimpleScalar Tool Set
- optimizes performance and flexibility
- in addition, provides portability and varied detail
Simulator types

- Full-system vs micro-architecture simulators
  - Full system simulators perform high level behavioral simulation. Involve OS overheads. Provide “complete picture” of results.
  - Micro-architectural simulators simulate individual functional block

- Execution Driven vs Trace Driven
  - Execution driven simulators simulate the u-architecture and do not contain any pre-assumed information. Very dynamic. Difficult to implement. Functional components not needed.
  - Traces are pre recorded streams of instructions, whose order has been fixed (branch outcomes known) and allow deterministic simulation each time. Fast. Easy to implement.

- What SS does: performs execution, gets feedback for a specific trace, then performs trace driven sim
Simulator Types in Simple Scalar

- `sim-safe.c` - minimal functional simulator
- `sim-fast.c` - faster (and twisted) version of `sim-safe`
- `sim-eio.c` - EIO trace and checkpoint generator
- `sim-profile.c` - profiling simulator
- `sim-cache.c` - two-level cache simulator (no timing)
- `sim-cheetah.c` - Cheetah single-pass multiple-configuration cache simulator
- `sim-bpred.c` - branch predictor simulator (no timing)
- `sim-outorder.c` - detailed OoO issue performance simulator (with timing)
Objective

• To get you accustomed to the Simple Scalar tool-set
  • Performing SimpleScalar Installation (linux machines only, use virtualbox if reqd)
    – Installation files will be provided
  • Running simple functional simulation with pre-existing binaries
  • Fiddle around with the tool and use -h prefix to understand how to configure simulators for user defined runs (very important)
• Basic understanding of SimpleScalar architecture
  • Read users manual (very important)
  • Tutorial v2 (has a lot of details, dont need to go into them)
Installation

- Tarball will be provided on the website
- Compiler simplesim-3.0 (the SimpleScalar processor simulator), type the following commands:
  - cd $HOME/simplesim-3.0 (depends on where you unpacked the .tar file)
  - make config-pisa
  - Make
Installation-test

- Type the following commands
  - cd $HOME/simplesim-3.0
  - ./sim-outorder ./tests/bin.little/test-math (by default: out-of-order issue)
  - You should see this somewhere on the screen:
    sim: ** starting performance simulation **
    pow(12.0, 2.0) == 144.000000
    pow(10.0, 3.0) == 1000.000000
    ...
    -1e-17 != -1e-17 Worked!

- From the output, you will find that sim_CPI (cycles per instruction) is about 1.04
Installation-test2

- Type the following commands:
  - `cd $HOME/simplesim-3.0`
  - `./sim-outorder -issue:inorder ./tests/bin.little/test-math` (in-order issue)

- Wade through output for `sim_CPI` (cycles per instruction) is about 1.51. Why are the CPIs different?

- Look for all `sim_*` outputs. Try to explain why they are different.
Installation Wattch

- Compile the Wattch simulator, type the following commands:
  - cd $HOME/sim-wattch-1.02d (depends on where you unpacked the .tar file )
  - make config-pisa
  - Make

- Wattch incorporates four different power models: avg_total_power_cycle, avg_total_power_cycle_cc1, avg_total_power_cycle_cc2, and avg_total_power_cycle_cc3. What are the differences?