EEL 6935 Embedded Systems - Fall 2011

Assignment 2

SESC

Assigned: 11/05/11

Due date: 11/21/11 @ 8 PM via Sakai

Since this assignment is very hands-on and you cannot learn the tools unless you actually follow the steps, all work must be done ***individually***. ***No group work allowed***.

In this assignment you will:

1. Install SESC [1]
2. Analyze miss rates of SPLASH-2 [2] benchmarks
3. Vary number of processors and analyze IPC for SPLASH-2 benchmarks

1. Installing SESC

The installation instructions are based on instructions in [1] and have been tested on the grid.ecel.ufl.edu Redhat x86_64 machines. You are welcome to install these tools on your own Linux OS machine or on Cygwin, however, the TA will not give you installation help for problems associated with these installations. You will have to debug your installation using web resources.

Download SESC

Follow the instructions on SESC Download page [1] for **Anonymous CVS Access**. The SESC installation files are now in *$HOME/sesc.*

Note: If you do not use grid.ecel.ufl.edu you may be required to install/update *m4, bison, flex, zlib1g-dev, and cvs*.

Compile SESC for a SMP System

Create a "build_smp" folder inside *$HOME/sesc.* To configure SESC for SMP (--enable-smp):

cd $HOME/sesc/build_smp
../configure --enable-smp

To see the list of configuration options use:

../configure --help

Compile the simulator:

$ make
$ make sesc

Test the SESC installation

The compiled SESC executable is called *sesc.smp*. To test the installation run the *crafty* benchmark with the *smp.conf* SESC configuration file and the *tt.in* input file:

./sesc.smp -c../confs/smp.conf ../tests/crafty < ../tests/tt.in

The output statistics are an output file called *sesc_crafty.xxxxxx* in your *$HOME/sesc/build_smp* directory.

To see a list of all sesc.smp options type:
./sesc.smp

Questions

1. How would you configure SESC to a) use MIPS emulation, b) use an in-order pipeline, and c) use debug features in an SMP system?
2. What are the readMiss, writeMiss, readHit, and writeHit statistics for the IL1 and DL1 caches in P(0) for the crafty benchmark run with the smp.conf configuration file?
3. Run the *crafty* benchmark with the *sesc.conf* SESC configuration file and *tt.in* input file again. This time have SESC set the name of the output file to *crafty_test.txt*, skip the first 1,000,000 instructions, and simulate for 4,000,000 instructions.
   a. What command did you execute? (./sesc.smp <sesc arguments>)
   b. What are the readMiss, writeMiss, readHit, and writeHit statistics for the IL1 and DL1 caches in P(0)?

2. Analyzing Miss-Rates

Download SESC_files.zip. SESC_files.zip contains the SESC binaries (water-spatial and lunon), necessary input files, and instructions on running the benchmarks.

For each benchmark you will analyze the dL1 miss rate as you vary the dL1 cache size, line size, and associativity in a single processor system.

a) Vary Cache Size

Set the cache line size to 16 bytes and associativity to direct-mapped for the data L1 cache. Set the iL1 cache to 64KB, 128 byte line size, 4-way associativity, and the L2 cache to 512KB, 128 byte line size, 4-way associativity. Vary the dL1 cache size from 1KB to 256KB (the cache size must be a power of 2) while keeping all other parameters constant. Plot a graph of the dL1 miss rate vs. dL1 cache size and discuss the results.

In this step choose an appropriate cache size. The goal is to minimize the dL1 miss rate without choosing an unnecessarily large cache. For example, if increasing the cache size from 128KB to 256KB results in a negligible decrease in miss rate, choose the 128KB cache.

b) Vary Line Size

Set the cache size to the size chosen in part a, and the associativity to direct-mapped for the data L1 cache. Vary the dL1 line size from 16 bytes to 128 bytes (the line size must be a power of 2) while keeping all other parameters constant. Plot a graph of the dL1 miss rate vs. dL1 line size and discuss the results. In this step choose the line size that gives the smallest miss rate.

c) Vary Associativity

Set the cache size to the size chosen in part a, and the line size to the line size chosen in part b for the data L1 cache. Vary the dL1 associativity from direct-mapped to 8-way (the associativity must be a power of 2) while keeping all other parameters constant. Plot a graph of the dL1 miss rate vs. dL1

associativity rate and discuss the results. In this step choose the associativity that gives the smallest miss rate.

Create a file called *section2_benchmark.x* to record the following information for each benchmark:
The benchmark name
The dL1 cache size, line size, and associativity chosen
The graphs and other discussion

3. Varying Number of Processors

For this section you will simulate a system with private iL1 and dL1 caches, and a shared L2 cache therefore you need to modify *cmp.conf* as necessary. Before varying the number of processors you need to replace the */sesc/src/libmint/subst.cpp* with the *subst.cpp* file found in the *SESC_files.zip* folder, and create a new sesc.smp simulator.

Set both the dL1 and iL1 caches to 64KB, 4-way associativity, 64 byte line size, and the shared L2 cache to 256KB, 4-way associativity, 64 byte line size.

For each benchmark vary the number of processors: 1p, 2p, 4p, 8p, and 16p. Create a file *section3_benchmark.x* and analyze the average execution time (ClockTicks in SESC output), average number of instructions executed, average IPC, average speedup, and L2 misses wrt. increasing the number of processors. For example does the IPC increase or decrease as the number of processors are increased? Why? Note: You may use */sesc/scripts/report.pl <sesc_output_file>* to find the IPC or do your own calculations using the SESC output files.

Turn in the *.conf* file used to simulate a 16p system.

4. What to turn in

You must submit the following via Sakai in a zipped file named *Lastname_Firstname_SESC*:

- Answers to the questions in section 1.
- The *section2_benchmark.x* file for each benchmark from section 2.
- The *section3_benchmark.x* file for each benchmark from section 3.
- The 16p *.conf* file from section 3.

References

[1] SESC: cycle accurate architectural simulator http://sesc.sourceforge.net/index.html

[2] Woo, S. C., Ohara, M., et al. 1995. The splash-2 programs: Characterization and methodological considerations. In Proceedings of the International Symposium on Computer Architecture, pp. 24–36, June 1995.