CSCI 315 Lab 9 Exercise

April 9, 2009

- **Objectives:** In this lab you will work with a simulation of a cache with a goal of understanding the effects of the entry replacement policy on the hit rate. While the simulator and data are for a memory cache, the concept of caching is general and covers Translation Lookaside Buffers (TLBs) and demand paged memories, which can be looked at as page caches, in addition to memory caches.

- **Preparation:** Copy the file `Driver.java` and the directory `CacheMgr` and its contents from ~cs315/Labs10/Lab09 to an appropriate Lab 9 directory. In the course of today's lab you will create graphs. You may use whatever package you prefer for this, including OpenOffice, Matlab, or gnuplot.

- **Problem 1:**

  Study the files `Driver.java` and `CacheMgr/Cache.java` to understand the nature of the simulation.

  The data for the simulation is derived from an execution session on a MIPS processor. The data is contained in files in the directory

  ```
  ~cs315/Labs/CacheData
  ```

  The class `Driver.java` reads the data from that directory. What you should know is that in the memory reference sequence (in the file) there are 265,755 memory references to 4,395 distinct memory pages - this should give realistic simulation results. During the lab you will implement various page replacement algorithms and test them using this data.

  You should also look over the `.java` files in `CacheMgr`, which include `Cache.java`, `CacheEntry.java`, `Memory.java` and `MemEntry.java`. The file `Cache.java` is especially important because you will modify that file during the lab session.

  Looking at `Cache.java` you will notice that it includes methods named `RandSelect, Optimal` and `SelectVictim`. Read these methods and understand them before moving on.

  Do This! Compile the file `Driver.java`. The `main()` method reads a command-line argument that specified the cache size. Execute the `Driver` program for each of the command-line arguments 8, 16, 32, 64, and 128. E.g.,

  ```
  java Driver 8
  ```

  The results printed will be based on the random page replacement algorithm. Edit `Cache.java` and modify `SelectVictim` to call the `Optimal` mthod instead. Recompile `Driver.java` and run it again for the same

cache sizes. The optimal algorithm takes considerably longer to run than the random, so be patient.

Record the results of your 10 test runs (five for each algorithm). **Create a graph plotting the results.** Plot cache sizes on the x axis and the number of hits on the y axis. You should plot a line for each of the replacement algorithms and label the lines. **Include your graphs in your handin for the lab.**

- **Problem 2:** In the first part of the lab you should have seen that the statistics you gathered convincingly show that the optimal algorithm is best at minimizing the cache miss/TLB miss/page-fault rate. However, there's a catch: we know the optimal algorithm can't actually be implemented. You will now implement two new page replacement algorithms that try to achieve optimal results. Gather similar statistics for each. You will plot this data and analyze the algorithms' performance.

  You should implement the two algorithms FIFO (first-in, first-out) and LRU (least recently used). The values you need to implement LRU are available in `CacheEntry` objects. You may need to add a data member to implement FIFO. Each algorithm should be implemented as a private method in the class `Cache`. To test these you can either add a new command-line argument to specify which algorithm to use, or edit the method `SelectVictim` as needed to call a different algorithm.

  Do This! Run the new simulations with the specified cache sizes as you did before. Plot the results for the two new algorithms. Explain the results. **Include your graphs, your analysis, and the code for your two new methods in your hand in file.**

  NOTE: Be careful about stating your conclusions too strongly. For example, try running the algorithms for cache sizes of 1024 and 2048. It appears that the LRU algorithm may be sensitive to the size of the working set.

- **Handin:** Hand in your graphs, analysis, and code as specified above by next Monday.