CS 4431 - Computer Architecture

Term Project

Due: Thursday April 18, 2024, Class time)

In this project, we will study the performance of caches. In doing so, we will follow what is known as *trace driven simulation*. In trace driven simulation, a trace of program references produced by a processor simulator are input to a simulating program and at the end of the trace, collected statistics are printed. In this project, we will use a trace of the form:

```
@<I/D><R/W><address>
```

where `@` is the at sign, followed by `I` for instruction references, and `D` for data references, `R` for reading from memory, and `W` for writing to memory, followed by an 8 character hexadecimal address that is referenced. An example reference line could be:

```
@DRbbdaff00
```

meaning that this is a data reference reading a memory location 0xbbdaff00. The traces to be used in your testing are located under the directory `/local/classes/cs4431/trace` in the lab machines. There are six traces produced by running a simulator on Spec 95 benchmarks, namely, 099.go, 124.m88ksim, 126.gcc, 129.compress, 132.ijpeg, and 134.perl.

The basic steps of the project is outlined below:

1. Develop an implementation for a direct-mapped, write-back cache in a language of your choice. It should accept the cache attributes such as line size and cache size from the command line, read from the standard input and print a similarly formatted reference to the standard output upon a miss. It should discard any characters read from the standard input until the marker `@` is seen. This way, you will be able to pipe data from the simulator to your cache and pipe its output to another cache and so on, even though the program may print other information to its standard output.

2. Run your cache simulator using the supplied trace data by input directing the files to your cache implementation. Use cache sizes of 8 and 16Kb and line sizes of 4 and 8 words/line and find the miss rates for an instruction cache, data cache and a unified cache.

3. Now concentrate on the data references only. Use a line size of 8 words per line for all runs. Assume that the hit time for L1 cache is a single cycle, hit time for L2 is 16 cycles and an L3 hit takes 64 cycles. Also assume that the path between the caches and the path between the last cache and the main memory are all wide and an entire line can be transferred between layers in one cycle, upon a hit. Using an L1 cache only L1,L2 combination and L1,L2,L3 combination, record the miss rates and calculate the effective memory access time. Note that each configuration is easily established by piping the output from one to the next and the same program is used throughout the experiments. What is the most effective configuration (i.e., number of layers and their sizes) that yields the lowest average memory access time?

4. Simulate the following configurations with all the benchmarks:

   ```
   L1 sizes : 4, 16 KB
   L2 sizes : 32, 64 KB
   L3 sizes : 256, 1024KB
   ```

5. Main memory latency is 100 cycles.

6. Keep statistics of number of references received, number of hits, number of misses, number of read misses resulting in a write and number of write misses resulting in a write back and report these for each configuration in a table.

7. Write a report about your findings. In your report, it will be helpful to cover:
   (a) The implementation process, the difficulties you have encountered.
   (b) Your fully commented program and its description.
   (c) Measurements itemized above.
   (d) A discussion of the results you have obtained.