ECE 2500 Project 3
Fall 2020

Total points: 100

Version: 1 (Any updates will be announced on Canvas.)

Deadline: Wednesday December 9th at 11:59 PM

Late Policy: Projects have strict deadlines and have to be digitally submitted at 11:59 PM on the day they are due. If submitted before 1 AM, a 5% reduction in grade will apply and the project will still be accepted. If submitted by 11:59 PM of the day after the due date, a 20% reduction in grade will be applied and the project will still be accepted. No project will be accepted after that point without a letter from Dean's Office. In order for your projects to be accepted, you need to complete the submission process. Uploading files but not making them available to the instructor, submitting the wrong version, and other technical or non-technical issues do not make you eligible for a late submission.

Note 1: The report submitted for this project, including all figures and text, should be your individual work or else properly cited. Any suspected plagiarism will be submitted to the Office of Undergraduate Academic Integrity.

Note 2: You can use any development environment such as Visual Studio or Linux to compile and run your code.

Project Overview

There is a continuum of cache management strategies ranging from direct mapped through set associative to fully associative. A direct mapped cache has only one possible cache entry for a memory block, while a fully associative cache allows a memory block to occupy any cache entry. Computer architects use simulations and analysis to help them make decisions on cache parameters. Compared to previous projects, this project requires less code writing and more time spent doing data collection, analysis, and presenting your results. Rather than write cache simulation software, you will make use of open source code available at:

https://github.com/adhnguyen/CacheSimulation

The code is written in C and is fairly readable and well structured. The code reads a memory trace file from standard input and writes cache performance statistics to standard output.

Part 1 – Compiling the code

Download the ZIP from the link above and find the top-level code (i.e. the `main()` function) contained in a file named `cache.c`. The code was developed using the GNU tools available in Linux, and the following changes are required to run it in Visual Studio:

1. Delete the `#include <netinet/in.h>` line.
2. Replace

   ```
   cache newCache[set];
   ```

   with

   ```
   cache * newCache = (cache *) malloc(set * sizeof(cache));
   ```

3. Replace

   ```
   char output[32];
   ```

   with

   ```
   char output[33];
   ```

4. Add preprocessor directive _CRT_SECURE_NO_WARNINGS before compiling. Add it by going to Project > Properties > C/C++ > All Options > Preprocessor Definitions.

Compile the code either in Linux (following the README.txt) or in Visual Studio using a Windows Console Application project. To ensure you compiled correctly check whether you get the same results for example 1 in the README.txt file. When running in Windows use the command shell (cmd.exe) and use a command as follows (example 1 from README.txt):

```
type art.trace | cache.exe -a 1 -s 16 -l 16 -mp 30
```

Where I have assumed the compiled executable and extracted trace file are in the same directory.

Part 2 – Trading off the cache size and associativity while keeping the block size constant

For the ‘art.trace’ memory trace, use a cache block size of 32 and a miss penalty of 40 cycles to determine the cache effectiveness for the following pairs of (cache size, associativity):

(256KB, 1), (128KB, 2), (64KB, 4), (32KB, 8), (16KB, 16), (8KB, 32)

Remember, use command line arguments to run your compiled program with the settings above. Your results for the above tests should be presented in a single graph that uses some way to identify and distinguish the data for each setting (e.g. color coding). Your report should summarize and explain the results/trends/anomalies obtained.

What to turn in

1. PDF Report: Create a report that briefly summarizes and explains your findings on how the above cache parameters impacted cache performance. Your report should be titled `Project3_report_PID`. Include the following:
    a. A title.
    b. Your name and affiliation.
    c. A one-paragraph abstract of the problem and results.
    d. The following sections:
        i. An introduction providing an overview of cache organizations and parameters. Define all the terms used in the rest of the paper.
        ii. A description of the simulation software and test data used. Summarize the input data format, command line arguments, data structures, and code structure.
        iii. A section for the experiment described in Part 2. State the question being investigated and the parameter combinations used. Results should be presented graphically. Provide an explanation of the results/trends/discrepancies.
        iv. Summarize the conclusions drawn from the work performed.

    Upload your report to Canvas before the deadline.