Homework 1

Instructions:

- Please type your solutions into a document and convert it into a PDF file. Your solution document should contain your name, student ID, the course name, and homework number. Please submit your solution PDF via Canvas.
- Make reasonable assumptions where necessary and clearly state them!
- You may discuss concepts with your classmates. This fosters group learning and improves the class' progress as a whole. However, make sure to submit your own independent and individual solutions.
- Combine all files of your submission into a single zip file.

Problem 1

Install the Dinero cache simulator. You can find the full installation guide here: http://www.dejazzer.com/coen4730/doc/hw2_cache_dinero.pdf

Dinero allows you to configure a cache using command line. The simulator takes as input an address trace. We will use the "din" input format, which is an ASCII file with one LABEL and one ADDRESS per line. Here are some examples of trace used as an input for DineroIV cache simulator:

2 0	This is an instruction fetch at hex address 0.
0 1000	This is a data read at hex address 1000.
1 70f60888	This is a data write at hex address 70f60888.

Using the trace provided as input, use DineroIV to model an instruction cache and a data cache with a combined total cache space of 32KB (for a split cache, assume a 16KB instruction cache and a 16KB data cache. The block size should be varied (8B, 32B and 128B) and the associativity should be varied (direct mapped and 4-way). Model both split (separate instruction and data caches) and shared (all accesses go to a single cache that holds both instructions and data) caches. There are a total of 12 simulations. No other parameters should be varied. Graph the results you get from these experiments (cache hit rate and miss rate) and discuss in detail why you see different trends in the graphs (30pts).

Make sure to use lowercase for your command line!

Problem 2

Repeat the problem 1, but this time develop your own cache simulator. Your cache simulator should able to read the provided memory access trace file and simulate the behavior of cache. You can use (C/C++, Python) to implement your cache simulator. Repeat all 12 simulation scenarios of the previous problem, this time on your own simulator (assume LRU as the replacement policy). Plot the results in a graph similar to problem1. Also, compare your results (cache miss and hit rates) against DineroIV simulator. Ideally, you should receive similar results with slight variations. Make sure to submit your source codes separately (50pts).

Problem 3

In this problem, you will use the allcache.so Pin tool to study the relationship between a program, compiler optimizations, and memory behavior. Use linpack and drystone programs from the previous homework (homework_0).
The documents including (manual, installation guide and source files) are available here:
https://software.intel.com/en-us/articles/pintool-downloads

Here is a short installation guideline:

1- Download version 81205 for Linux

2- tar -xvf pin-3.2-81205-gcc-linux.tar

3- cd source/tools/Memory

4- make

Then run the tool as follows:

pin –t allcache.so -- Your Program

You will see a large number of memory hierarchy values for both the instruction stream and the data stream. Now recompile your program applying different optimization switches, rerun the new binary with Pin, and compare the results. Try out 3 different compiler switch settings that generate a significant change in the memory stream. Plot these results and attempt to explain what trends you are seeing across these two applications. Note: for the results report the cache hit and miss rate across different level of hierarchy, including hit/miss for L1 (data and instruction), L2 and L3 (30pts).

Problem 4

Use Pin to create memory trace for Dhrystone and Linpack benchmarks. PinTools provides the instruction of generating memory trace. Use your cache simulator (developed in probem2) to explore the cache miss/hit rates for four cache configurations:

Data cache size: 16KB, and 64KB
Cache block size: 32B and 128B
Associativity: 4-way

For this example, only focus on the data memory access. Make sure to plot the results (hit and miss rate) for both benchmarks across four configurations. Argue your results and finding in your report (40pts).

NOTE: Due to a long simulation time, you can narrow down the simulation to the first 4 million data memory access. However, it would be nice to see the results for a larger memory access trace.