Lab 7 - Caches

Introduction

In this lab you will be exploring cache design trade offs. You will build a number of different caches, and see how these design choices affect the number of memory accesses. A sample simulation program, written in C, can be downloaded here and a version in C++ can be downloaded here

Note : You can use a modified version of the PTLSIM simulator to get an executable's memory trace. This executable should work on any compiled C/C++ application. To use it call ./ptlsim with your executable as an argument (i.e. "$./ptlsim a.out"). The simulator should output two files ptlsim.log, and ptlsim.cache. The .cache file will hold a trace of instruction and data loads for your executable. PTLSIM is used to generate traces of any compiled program you have installed. You can generate traces if you want to test your simulator on multiple programs, but we are only grading the performance on the trace below.

Deliverables

You should build a cache simulator that reads one address trace file and simulates multiple cache architectures and reports the miss rate (# misses / total accesses). Your simulator should support all the following attributes:

- 64 bit addresses
- Block Size: 16 elements
- Replacement Policies: LRU, FIFO
- Cache Sizes: 1024, 2048, 4096, 8192, 16384 locations
- Associativity: Direct Mapped, 2-way, 4-way, and 8-way

The output of your simulator should have to following format.

- First output the LRU policy data, followed by the Fifo policy data
- The x-axis should hold the cache sizes
- The y-axis should hold the cache associativity
- The output should go to 2 decimal places. (leading zeros for the whole numbers are optional)

You can build your simulator in C/C++, or Python. If you want to use a different language please check with the TA first. Your program should read from standard input, and write to standard

output. To test your simulation you can use the memory trace file here. (If you are having problems with the zip, you can try the original file).The output for this file should look like:

LRU Replacement Policy

	1024	2048	4096	8192	16384
1	55.01	42.07	29.30	20.74	13.91
2	51.58	36.44	23.70	13.98	08.49
4	48.85	33.97	20.04	11.33	06.37
8	47.44	32.20	18.63	10.02	05.72

FIFO Replacement Policy

	1024	2048	4096	8192	16384
1	55.01	42.07	29.30	20.74	13.91
2	53.31	38.32	25.47	15.37	09.49
4	51.86	37.07	23.11	13.67	07.86
8	51.14	35.98	22.40	12.79	07.44

Turn-In

Each group should turn in one tar file to iLearn. The contents of which should be:

- A README file with the group members names, and any incomplete or incorrect functionality
- The source code for your simulator
- A makefile to compile your code on WELL. When compiling I should only have to type "make". If you are using python you do not need a makefile, but your code should be executable.