CS6600 : Computer Architecture
Assignment 1 : Cache Profiling

Submission Deadline: 11:59 PM, 15th August 2022

In this assignment you will utilize Cachegrind simulator to collect various cache statistics such as instruction references, L1 I-Cache misses, L1 D-Cache misses, L2 cache misses etc... for different cache configurations such as associativity, block size, cache sizes etc...

Resources:

1. Install Valgrind simulator: https://installati.one/ubuntu/20.04/valgrind/
2. Understand the Cachegrind simulator: https://valgrind.org/docs/manual/cg-manual.html, the various configuration option it supports and the output it returns.

Problem:

Explore the cache behaviour for different sorting algorithms:

1. Bubble sort
2. Selection sort
3. Merge sort
4. Quick sort
5. Radix sort

Consider an array of *long int* elements. The sorting code should take array size as command-line parameter. Create an array of following sizes: (i). 250_000 (ii). 500_000 (iii). 750_000 (iv). 1_000_000, initialized randomly. The code should sort the array in ascending order. Use the Cachegrind simulator to collect various cache statistics for different cache configurations. Also find the IPC of the program with the help of instruction count provided by Cachegrind. Summarize and expand the results in a report and provide your observations.

Submission guidelines:

1. This is an individual assignment.
2. Submit only file: *RollNumber_A1*.tar.gz file, containing the following:
   (a) Your analysis report in *RollNumber_A1*.pdf format. The report should contain your analysis based on cache statistics collected across different cache configurations with same/different programs. Also explain the method you used to find out the IPC of the program.

(b) Folder: *Sorting*, which contains the relevant source codes and script files with all the commands you executed for running Cachegrind, collecting the cache statistics and IPC.

3. All implementation should be in C language alone.
4. Please provide your system configuration and the function used to find the IPC in the report.
5. Please use two levels of Cache(L1 and LLC) in all your experiments.
6. For initializing the array with random numbers, use a constant seed input to *srand()*, so that you are able to reproduce the same array input across different sorting algorithms.
7. Note: Use suitable values w.r.t cache configurations so that you are able to differentiate the cache performance.