WESTERN
WASHINGTON UNIVERSITY

Computer Science Department

CSCI 247 Computer Systems I

Laboratory Exercise 7

## Objectives

Understanding the caching impact of spatial locality.

## Submitting Your Work

Submit your C program files as the Lab Exercise 8 Submission item on the course web site. You must submit your program by 11:59pm on the Friday following your scheduled lab session.

## Lab Tasks

### Task 1: Write a C Program to add all the elements of a 2 dimensional array

Implement a C program that takes 2 arguments in the command line as follows:

```
./CacheLab        <matrix size>      <computation Method>
```

Where the name of the executable is “CacheLab”, the first argument represents the size of a 2 dimensional array (equal number of rows and columns) and the third argument represents the calculation method type (only 2 method types required).

The program will first allocate memory for a 2D array using a C dynamic memory allocation function (eg. malloc) and initialize each matrix element to be the sum of the 2 indexes representing the matrix element.

Once the matrix is initialized, you will access each element in the matrix by implementing two “for” loops. If method “1” is specified as an argument for the calculation method, the outer loop will represent the rows and the inner loop will represent the columns. If method “2” is specified as an argument for the calculation method, the outer loop will represent the columns and the inner loop will represent the rows. The calculation will be a simple accumulator (initialized to “0”) that keeps adding each matrix element to itself.

Finally you will print the value of the accumulator.

## Task 2: Use the Perf tool to track the d-cache and i-cache misses

Use the following commands to run your program.

```
sudo perf stat –B –e L1-dcache-loads –e L1-dcache-load-misses –e cpu-clock –e context-switches
./CacheLab <Size> <Method>
```

```
sudo perf stat –B –e L1-icache-loads –e L1-icache-load-misses –e cpu-clock –e context-switches
./CacheLab <Size> <Method>
```

Use the values 64, 128, 256 and 512 for the size of the matrix and for each size, use both possible methods for computation.

## Task 3: Explain your results

Explain the output of “perf” when running your program with the different sizes and different computation methods. You should have a total of 8 sets of results to explain.