CS 698L Semester 2019–2020-I: Assignment 1

9th August 2019

Due Your assignment is due by Aug 18 2019 11:59 PM IST.

General Policies

- You should do this assignment ALONE.
- Do not plagiarize or turn in solutions from other sources. You will be PENALIZED if caught.
- We MAY check your submission(s) with plagiarism checkers.

Submission

- Submission will be through Canvas.
- Submit a PDF file with name "<roll-no>.pdf". You are encouraged to use LaTeX typesetting system for generating the PDF file.
- Submitting your assignments late will mean losing points automatically. You will lose 10% for each day that you miss, for up to three days.

Problem 1 **[10 points]**

Consider a processor which uses 32 bit addresses and can address 2^{32} B of memory. The address is split into a *t* bit tag, an *s* bit set index, and a *b* bit block offset. The processor has a cache with capacity 256 KB, and block size of 64 B. Answer each of the following for (i) direct-mapped, (ii) 4-way set-associative, and (iii) fully-associative cache.

(a) What is the total number of lines contained in the entire cache?

(b) What is b?

(c) What is s?

(d) What is t?

Problem 2 **[20 points]**

Consider the following loop:

```
float s = 0.0, A[size];
int i, it, stride;
for (it = 0; it < 100 * stride; it++) {
    for (i = 0; i < size; i += stride) {
        s += A[i];
    }
}
```

Assume a 4-way set-associative cache with capacity of 256 KB, line size of 32 B, and word size of 4 B (for float). Furthermore, assume cache is empty before execution, and uses an LRU replacement policy.

Given size=32K, determine the total number of cache misses on A for the following access strides: 1, 4, 16, 32, 2K, 8K, and 32K.

Problem 3

[40 points]

Consider a cache of size 64K words and line of size 8 words, and arrays 512 x 512. Perform cache miss analysis for the *ikj* and the *jik* forms of matrix multiplication (shown below) considering direct-mapped and fully-associative caches. The arrays are stored in row-major order. To simplify analysis, ignore misses due to cross-interference between elements of different arrays (i.e., perform the analysis for each array, ignoring accesses to the other arrays).

Listing 1: ikj form

```
for (i=0; i<N; i++)
    for (k=0; k<N; k++)
        for (j=0; j<N; j++)
            C[i][j] += A[i][k]*B[k][j];
```

Listing 2: jik form

```
for (j=0; j<N; j++)
    for (i=0; i<N; i++)
        for (k=0; k<N; k++)
            C[i][j] += A[i][k]*B[k][j];
```

Your solution should have a table to summarize the total cache miss analysis for each loop nest variant and cache configuration, so there will be four tables in all. Briefly justify your computations.

Problem 4

[30 points]

Consider the following code:

```
double y[4096], X[4096][4096], A[4096][4096];
for (k = 0; k < 4096; k++)
    for (j = 0; j < 4096; j++)
        for (i = 0; i < 4096; i++)
            y[i] = y[i] + A[i][j] * X[k][j];
```

Assume a direct-mapped cache of capacity 16 MB, with 32 B cache line and a word of 8 B. Assume that there is negligible interference between the arrays A, X, and y (i.e., each array has its own 16 MB cache for this question), and arrays are laid out in row-major form.

Estimate the total number of cache misses for A and X.