Assignment 5

Due November 25, 17:00

NOTE: Late submissions will **NOT** be accepted. Please submit a single PDF file with your answers via the **ECE 355 Brightspace** webpage.

1. [5 points] Consider the code portion of the matrix-vector product computation as shown below: **(double) 128x128** matrix **A** is multiplied by **(double) 128x1** vector **X**, producing **(double) 128x1** result **Y** (initially all 0's).

```
for (i = 0; i < 128; i++) {
      for (j = 0; j < 128; j++) {
            Y[i] = Y[i] + A[i][j]*X[j];
      }
}
```

Storing **X**, **Y**, and **A** (each **double** array element is 8 bytes in size) requires 128*8 + 128*8 + 128*128*8 = **130KB** of memory. If the cache (assume fully associative) is smaller than 130KB, the above code will cause many misses, considerably slowing down the program execution. Alternatively, one can perform blocked computation: partition **A** into smaller blocks and perform the product computation block-by-block. If our data blocks can fit into the cache, such blocked computation may significantly outperform the original code.

Rewrite the code fragment above using blocked computation and letting matrix **A**'s blocks be of size **64x64** (i.e., 4 blocks total). Assuming that such blocking yields the best performance, what can you say about the size of the cache?

2. [10 points] Consider a C code fragment below, working on a given square matrix **float X[N][N]** (stored row by row, i.e., in the row-major order), where **N = 256**:

```
float trace = 0;
for (i = 0; i < N; i++) {
      trace = trace + X[i][i];          /* sum diagonal elements of X */
}
for (i = 0; i < N; i++) {
      for (j = 0; j < N; j++) {
            X[i][j] = X[i][j]/trace;      /* normalize elements of X */
      }
}
```

Determine the **X**-related page fault rate in the following two cases: 1) the main memory uses **1-KB** paging with four pages allocated for **X**, and 2) the main memory uses **4-KB** paging with only one page allocated for **X**. Initially, no part of **X** is in the main memory. **Note:** Type **float** is 32 bits (4 bytes).

3. [10 points] Consider a C code fragment below, working on a given positive matrix **`float X[M][N]`** (stored row by row, i.e., in the row-major order), where **`M = 128`** and **`N = 512`**:

```
for (i = 0; i < M; i++) {
      float max = 0;
      for (j = 0; j < N; j++) {
            float y = X[i][j];
            if (max < y) max = y;   /* find max. element of row i */
      }
      for (j = 0; j < N; j++) {
            X[i][j] = X[i][j]/max;  /* normalize elements of row i */
      }
}
```

Determine the **`X`**-related page fault rate in the following two cases: 1) the main memory uses **1-KB** paging with four pages allocated for **`X`**, and 2) the main memory uses **4-KB** paging with only one page allocated for **`X`**. Initially, no part of **`X`** is in the main memory. **Note:** Type **`float`** is 32 bits (4 bytes).