18-645: How to Write Fast Code

Assignment:- Designing for the Memory Hierarchy

General Instructions.

- A single zipped file with the name `asg_3_≪your andrew id≫` is to be submitted via Canvas on the due date AoE.
- The zipped file should contain the following
  1. Source code (appropriately named) for all implementations;
  2. A Makefile that compiles all implementations on one of the machine on the ECE cluster;
  3. PDF version of your written report;
  4. Readme.txt that explains how to run your implementations. (Optional, but strongly recommended)
- Each program has to be named using the exact name found in parenthesis in the question.
- Start early.

1. **Cache Oblivious Algorithms & Z-ordering** Write a recursive cache oblivious program (`morton.c`) using 256bits SIMD for transposing a square ($n \times n$) matrix of double-precision elements into a different $n \times n$ matrix. Both input and output matrices are assumed to be in Morton(Z)-ordering.
   (a) What is the size of your kernel? Explain why you chose that as your kernel size.
   (b) Time your implementation for different sizes of $n \in \{8, 16, 32, 64, 128, 256, 512\}$, and provide a plot of execution times against problem sizes.

   Hint:
   - The implementations need only handle matrices of size $n \times n$ where n is a power of 2.
   - Do not use assembly to implement this assignment.

2. **Cache-aware Matrix Matrix Multiplication.** In the previous assignment, we implemented a matrix multiplication kernel that computes a matrix C of size $m \times n$, by multiplying matrices A ($m \times k$) and B ($k \times n$) together. m, and n were determined based on the latency and number of functional units available.

   In this assignment, we want to determine the value of k, and use the kernel to implement an even larger matrix multiplication. For the remaining questions, assume that the L1 cache is a 32kB (32×1024 bytes) 8-way set associative cache. The cache replacement policy is least recently used.
   (a) Assume that m and n are 6 and 8 respectively. If matrices A, B and C are in the L1 cache, what is the largest size of k that fit in cache. Recall that we made the assumption that A is stored in column major order while B is stored in row-major order.
   Explain your answer.
   (b) Based on your previous answer, how much of the L1 cache is used?
   (c) A larger matrix multiplication can be performed if we stack multiple A matrices vertically to form a larger $m_c \times k$ matrix $\mathcal{A}$. $\mathcal{A}$ is multiplied with the same B matrix to compute a new $m_c \times n$ matrix $\mathcal{C}$. If these larger matrices must reside in L2, what is the largest value of m_c if the L2 cache is 256kb, and is 8 way-set associative with LRU policy. Assume that the L2 cache is inclusive (i.e. data stored in the L1 cache is also stored in the L2 cache). Explain your answer.

(d) Based on your previous answer, how much of the L2 cache is used?

(e) Implement a SIMD packing routine to pack the matrix A so that a larger matrix multiplication (described in the previous part) can be computed using a loop around the 6×8 kernel provided. A should be organized into m_c/m contiguous panels and each panel is stored in column-major order.

Use your packing routine to pack the input matrix A, and compute the larger matrix multiplication

$$C = AB + C$$

where $\mathcal{C}$ is a $m_c \times n$ output matrix.

Report the performance (in FLOPS per cycle) attained by your implementation for computing an output matrix that is $m_c \times n$. How close to the theoretical peak is your implementation?

Hint:

- Use values of k and m_c computed previously.
- The purpose of the last question is to show that good memory layout can help sustain performance for larger problem sizes.

3. **Miscellaneous.** How many hours did it take you to complete this assignment?