EECE5640
High Performance Computing
Homework 3
*Submit your work on Canvas in a single zip file.

1. (20) In this problem, you will utilize the IEEE 754 format and evaluate the performance implications of using floats versus doubles in a computation.
   a.) Compute f(x) = sin (x) using a Taylor series expansion. To refresh your memory:

   $\sin(x) = \sum_0^\infty \frac{(-1)^n}{(2n+1)!}\, x^{2n+1}$

   $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} \ldots .$

   You should select the number of terms you want to compute (but at least 10 terms). Compute sin(x) for 4 different values, though be careful not to use too large a value. Generate two versions of your code, first defining x and sin(x) to use floats (SP), and second, defining them as doubles (DP). Discuss any differences you find in your results for f(x). You should provide an in-depth discussion on the results you get and the reasons for any differences.

   b.) Provide both IEEE 754 single and double precision representations for the following numbers: 2.1, 6300, and -1.044.

2. (30) In this problem, explore the benefits of compiling on the Explorer cluster with floating point vector extensions (e.g., AVX). To allocate a node on Explorer with AVX512 support, you will need to specify "—constraint=cascadelake". To utilize AVX512 instructions, make sure to compile using the -mavx512f flag.

   a.) Using the dotproduct.c example provided, develop an AVX512-accelerated version of matrix-vector multiplication. The example should give you a good start on the AVX intrinsics that you need to use in the program. Report on the speedup that you obtain as compared to a matrix-vector multiplication that does not use vectorization.
   b.) Using the same code, generate an assembly listing (using the -S flag) and identify 3 different vector instructions that the compiler generated, explaining their operation.

3. (20) In this problem, you will modify the matmul.c program provided, optimizing the execution of the matrix multiplication with first a dense matrix, and second with a sparse matrix. You are welcome to use pthreads, OpenMP or any of the optimizations that were presented in class to accelerate this code. Do not change the sparsity of the matrices in matmul.c. Do not use a GPU and do not use OpenBLAS in your solution. There will be prizes awarded for the fastest dense and the fastest sparse implementations.

4. (20) In this problem, you will utilize the OpenBLAS library available on Explorer. To use OpenBLAS, you will need to issue load openblas/0.3.29. You can refer to the blas_simple.c code provided for an example code.

   a.) Develop a matrix-matrix multiplication, multiplying two single-precision floating point matrices that are 256x256 elements. Compare your implementation to your dense implementation in problem 3. Discuss which result is faster and why.

   b.) Run your program OpenBLAS accelerated program on two different CPU platforms in Explorer. Discuss the CPUs you are using, and the performance differences you obtain.

5. (10) Find a published paper from an ACM or IEEE conference that discusses a novel sparse matrix format that was not covered in class. Discuss why the proposed format is superior to the CSR or CSC format. Make sure cite your sources.
   (Optional for undergraduate/Plus-One students, required for MS/PhD students. For undergraduate/Plus-One student, completing can add up to 20 points to your quiz grade.)

* Written answers to the questions should be included in your homework 3 write-up in pdf format. You should include your C/C++ programs and the README file in the zip file submitted.