18-645: How to Write Fast Code

Assignment:- Benchmarking

General Instructions.

- 2 separate files need to be submitted via Canvas on the due date AoE. A PDF of the report and/or writeup, and a zipped files with all the code
- Both files should be named `asg_1_≪your andrew id≫`, and they should only differ in their extensions.
- 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. Readme.txt that explains how to run your implementations.

1. **Know Your Machine.** Determine the following parameters for a particular computer in the ECE cluster
   (a) Hostname of the machine
   (b) CPU model and manufacturer
   (c) Base and maximum frequencies
   (d) Number of physical cores
   (e) Number of hardware threads
   (f) Number of caches
   (g) Size of each cache

   Hint:
   - You can connect to the ECE clusters using instructions from `https://cmu-enterprise.atlassian.net/wiki/spaces/ITS/pages/2332131370/ECE+Community+Compute+Clusters#Connection-methods`
   - Use `lscpu` on Linux to identify the CPU model. You will need the version number as well. Search on the Internet (manufacturers' website or otherwise) to determine the rest of the information. Remember to cite your sources.

2. **Benchmarking your machine - Latency.**
   (a) Write separate benchmark programs to measure the latency (in cycles) of the following instructions assuming that α and χ are integers, and a, b, c are SIMD vectors of 4 double-precision floating point numbers: (please write each benchmark in a separate file):
      i. Scalar Multiplication (`latency_int_mul.c`), i.e. $\chi = \chi \times \alpha$;
      ii. SIMD Addition (`latency_simd_add.c`), i.e. $c = c + a$;
      iii. SIMD Fused multiply-add (`latency_simd_fma.c`), i.e. $c = c + a \times b$.

   Hint:
   - Depending on the compiler used, your benchmark program may be optimized in various manner. Use `objdump` or other relevant tools to check your assembly.

- You might need to vary the optimization flag (`-O1, -O2, -O3, -mavx, -mfma`) passed to the compiler to ensure that there are no non-essential code between your testing.
- There are often many variants of the assembly instruction that computes the same operation. Test the variant that reads from and writes to registers.
- You might want to look at `https://www.agner.org/optimize/instruction_tables.pdf` to find out what others have measured for your specific instruction.

3. **Benchmarking your machine - Throughput.** Write separate benchmark programs to measure the throughput of the following instructions (please write each benchmark in a separate file):

   (a) SIMD Addition (`throughput_simd_add.c`), i.e. $c = c + a$.

   (b) SIMD Fused multiply-add (`throughput_simd_fma.c`), i.e. $c = c + a \times b$.

   Based on the empirical results from this question and the latency question, and making the assumption that each functional unit can compute 1 instruction every cycle, answer the following:

   (a) For each of the SIMD instructions, provide a plot of the execution time to explain how you determine the throughput.

   (b) How many functional units are there for computing SIMD fused multiply-adds?

   Hint:

   - Depending on the compiler used, your benchmark program may be optimized in various manner. Use `objdump` or other relevant tools to check that your assembly.
   - You might want to look at `https://www.agner.org/optimize/instruction_tables.pdf` to find out what others have measured for your specific instruction.

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