­Hsiang Lo

**Project #4**

**Objective**

Part 1 – Test array multiplication, SIMD and non-SIMD

Part 2 – Test array multiplication and reduction, SIMD and non-SIMD

**Requirements**

* Use different array sizes from 1K to 32M
* Run each experience a certain number of trials. Use the peak value for record.
* Create a table and a graph showing SSE/Non-SSE speed-up as a function of array size. Speedup will be S = Psse/Pnon-sse = Tnon-sse/Tsse (P = Performance, T = time)

**Notes**

* \_asm = You’re no longer in C, but assembly
* Link openmp library for timing using -fopenmp
* Do not use intel, any optimization flags and do not use **-O3**
  1. **What machine you ran this on?**  My project was ran on my MacBook pro 2017, same as my previous projects in order to main consistency. The project was then ran on school’s server, flip.
  2. **Show the table and graph**
  3. **What patterns are you seeing in the speedups?**

Starting at around 40,000,000 array sizes, the speedup from using SIMD is very clear from opposed to regular methods without using SIMD. This is most noticeable between SIMDMUL and REGMUL.

* 1. **Are they consistent across a variety of array sizes?**

Yes, the array sizes appear to graph consistently from one another. The regular functions shows an extremely high increases in time as array size gets bigger. SIMD functions while also shows increases in time as data sizes get bigger, but not by much in comparison to regular functions.

* 1. **Why or why not, do you think?**

All functions behave this way is due to the fact that I tried to increase the data sizes at the multiple of 4 for all four functions, and due to the same usage of variables, the graph remain constant as a result.

* 1. **Knowing that SSE SIMD is 4-floats-at-a-time, why could you get a speed-up of < 4.0 or > 4.0 in the array-multiplication?**

This Is due to the fact that SSE SIMD code provided to us is in assembly language. Using assembly language provides faster compute time for computer as there is less communication time, or decomposing reading level code such as C to assembly which is necessarily for computer to understand and compute. Further, since the time result is extremely small to begin with, the difference in using machine language is more noticeable.

* 1. **Knowing that SSE SIMD is 4-floats-at-a-time, why could you get a speed-up of < 4.0 or > 4.0 in the array-mutiplication-reduction?**

This Is due to the fact that SSE SIMD code provided to us is in assembly language. Using assembly language provides faster compute time for computer as there is less communication time, or decomposing reading level code such as C to assembly which is necessarily for computer to understand and compute. Further, since the time result is extremely small to begin with, the difference in using machine language is more noticeable.