CPEG655

Final Project Proposal

Michael Hutts

Introduction

My proposed final project is matrix multiplication via SSE (Streaming SIMD Extensions). The goal of this project is to perform multiplication on an NxN matrices using only vector operations with SSE. To start, I will attempt to perform matrix multiplication on 2 small matrices of matching square dimensions by the first deadline, with the end goal of supporting larger matrices and maximizing performance.

Overview

Regarding the technical approach of the project, from my initial testing no greater than SSE2 compatibility is required to achieve my desired goals, but I'll be curious in experimenting with later versions if time allows. I've already confirmed SSE is working within WSL on my system and compiles correctly with g++, the compiler I plan to use for this project.

Regarding syntax, there are some instructions I believe I'll need to familiarize myself with to achieve the goals of this project:

- _mm_setzero_ps()
  - Meaning: Initializes 4 single-precision floats to 128-bit registers to 0.
  - Purpose: Initialize our registers before they can be used.
- _mm_loadu_ps(const float* mem_addr)
  - Meaning: Load 4 single-precision floats to 128-bit registers from memory.
  - Purpose: Load values to our registers from memory to be multiplied.
- _mm_storeu_ps(float* mem_addr, __m128 a)
  - Meaning: Store 4 single-precision floats from 128-bit register to memory.
  - Purpose: Store values from register to memory after multiplication.
- _mm_mul_ps(__m128 a, __m128 b)
  - Meaning: Multiply values of two 128-bit floating-point vectors.
  - Purpose: Perform relevant multiplication along the two matrices.
- _mm_add_ps(__m128 a, __m128 b)
  - Meaning: Add values of two 128-bit floating-point vectors.
  - Purpose: Accumulate the results of the multiplication to the resulting matrix.

Verification

Verifying the correctness of any matrix multiplication functions made will be relatively easy as we've already implemented single-threaded loop-based on-CPU matrix multiplication previously in the class. Although they might perform poorly regarding execution time, they'll make a good way to verify the correctness of any results produced by SSE functions regardless of the input.