Report

I. Implementation

The current implementation is divided into three functions: main, naives_operation, and simd_operation. The "main" function starts by generating random floating values between -50.0 and 50.0 for groups of **m**, **x**, and **b**. Then, it calls the "naives_operation" function and "simd_operation" function and passes in the generated groups of **m**, **x**, and **b** into each function respectively. For each function call, the execution time and the return value are being recorded. The calling and recording processes are repeated a certain number of iteration and the results will be averaged over all iterations.

The "naives_operation" function calculates the sum of all **y** given groups of **m**, **x**, and **b** using the formula: **y=mx+b**. It does this by iterating over each row of **m**, **x**, and **b** and accumulating results into the variable **total**. Finally, it returns the variable.

The "simd_operation" function also calculates the sum of all **y** given groups of **m**, **x**, and **b** using the formula: **y=mx+b**. However, it starts by creating an **__m256** variable called **totals** to store the partial sum of **y** values. Then, it starts to iterate overs **m**, **x**, and **b** arrays in a group of 8 and stores extracted results into variables **p_m**, **p_x**, and **p_b**. Then, it multiplies **p_m** with **p_x**, adds the first resulting values with **p_b**, and accumulates the second resulting values into **totals**. Lastly, all values in **totals** are summed together and return as the result.

II. Compilation Steps

```
g++ -march=native -O3 -o slope .\slope.cpp

.\slope.exe
```

III. Speedup

Given 100000 groups of **m**, **x,** and **b** and 100000 iterations, the speedup is approximately 4.3 times.