[MP2 Specs] x86-to-C interface programming project specifications

- Due Jul 31 at 11:59pm
- Points 0
- Questions 1
- Available Jul 18 at 2:30pm - Jul 31 at 11:59pm
- Time Limit 1 Minute

Instructions

- This assignment will provide only your MP2 specification/problem, and will not be graded.
- Submit your MP2 solution using the assignment titled "[MP2 Submission] x86-C Interface Programming Project."
- Accomplish the peer evaluation using the assignment titled "Peer Evaluation [for MP2]."
- Though MP2 is assigned by group/pair, students should treat this requirement as an individual assignment by independently working on their solution. Then, the opportunity to collaborate later on with your partner will hopefully provide a helpful advantage to both members. Groupings/pairings are automatically assigned in alphabetical order.
- Since groupmates may receive two different MP2 specs or problems, partners may discuss and decide which of the two specs to take.

Attempt History

	Attempt	Time	Score
LATEST	**Attempt 1**	less than 1 minute	0 out of 0

Score for this quiz: 0 out of 0
Submitted Jul 22 at 2:56pm
This attempt took less than 1 minute.

Write the kernel in (1) C program and (2) an x86-64 assembly language. The kernel must calculate the distances between the coordinate points across two vectors.

*Required to use functional scalar SIMD registers

*Required to use functional scalar SIMD floating-point instructions

Input: Scalar variable *n* (integer) contains the length of the vector; Vectors X_1, X_2, Y_1, Y_2, and Z are **single-precision float.**

Process: $Z[i] = \sqrt{(X_2[i] - X_1[i])^2 + (Y_2[i] - Y_1[i])^2}$

Example:

X_1 -> 1.5, 4.0, 3.5, 2.0

X_2 -> 3.0, 2.5, 2.5, 1.0

Y_1 -> 4.0, 3.0, 3.5, 3.0

Y_2 -> 2.0, 2.5, 1.0, 1.5

(answer)
Z -> 2.5, 1.58113883, 2.692582404, 1.802775638

Output: store result in ve*ctor Z.* Display the result of 1st ten elements of vector *Z* for all versions of kernel (i.e., C and x86-64).

Note:

1.) Write a C main program to call the kernels of the C version and x86-64 assembly language.

2.) Time the kernel portion only.

3.) For each kernel version, time the process for vector size *n* = {2^{20}, 2^{24}, and 2^{30}}. If 2^{30} is impossible, you may reduce it to the point your machine can support (i.e., 2^{28} or 2^{29}).

4.) You must run at least 30 times for each version to get the average execution time.

5.) For the data, you may initialize each vector and scalar variable with the same or different random value.

6.) You will need to check the correctness of your output. Thus, if the C version is your "sanity check answer key," then the output of the x86-64 version has to be checked with the C version and output correspondingly (i.e., the x86-64 kernel output is correct, etc.).

7.) Output in GitHub (make sure that I can access your Github):

a.) Github readme containing the following (C and x86-64):

i.) comparative execution time and short analysis of the performance of the kernels

ii.) Take a screenshot of the program output with the correctness check (C).

iii.) Take a screenshot of the program output, including the correctness check (x86-64).

iv.) short videos (5-10mins) showing your source code, compilation, and execution of the C and x86-64 program

b.) Visual Studio project folder containing **complete** files (source code: C, x86-64, and all other required files) for others to load and execute your program.

Rubric:

C main program with initialization and correct call/passing parameters to C and x86-64	10
Correct output (C version)	10
Correct output (x86-64)	40
Comparative result	20
Analysis of result	10
Video	10
not following instructions	-10/instructions
Note: No usage of functional scalar SIMD registers and scalar SIMD instructions	grade = 0

Quiz Score: 0 out of 0