

# x86-to-C interface programming project

Started: Nov 24 at 8:43pm

## Quiz Instructions

**\*\*Remember the academic honor pledge that you signed.**

-----

General directions:

- 1.) This is assigned to a pair, only one member will access the quiz and submit. Deduction 10% if both member accessed and submitted. The member that should access the quiz should be the one whose lastname occurs first.
- 2.) Ignore the file upload button in the specification question.
- 3.) Submission via GitHub. Place your Github link in the next question. Make sure I can access your GitHub.
- 4.) Follow the directions found in the specifications.
- 5.) Take a screenshot of your project specification for reference purposes.

**The first project specification is your project specification regardless of the attempts.**

⋮

Question 1 0 pts

Write the kernel in (1) C program and (2) an x86-64 assembly language. The kernel is to perform a dot product between vector  $A$  and vector  $B$  and place the result in  $sdot$ .

\*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  $A$  and  $B$  are both **single-precision float**. Scalar  $sdot$  is a single-precision float.

**Process:**  $sdot = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$

**Output:** store the result in memory location  $sdot$ . Display the result for all versions of the 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  $n = \{2^{20}, 2^{24}, \text{ and } 2^{30}\}$ . 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 20 times for each version to get the average execution time.
- 5.) For the data, you may initialize each vector 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 (debug and release mode; 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
Short video	10
not following instructions	-10 for each instruction
Note: No usage of functional scalar SIMD registers and scalar SIMD instructions	grade = 0

Upload

Choose a File



Question 2 90 pts

Place your GitHub link here.

Edit View Insert Format Tools Table

12pt ▾ Paragraph ▾ | **B** *I* U A ▾  ▾  $T^2$  ▾ |  ▾  ▾  ▾  ▾ |

   ▾ |  ▾  ▾  ▾ |   ▾  $\sqrt{x}$  

p



0 words

</>



Not saved

Submit Quiz