Deep Dive: SIMD programming project specification

Started: Jan 27 at 3:01pm

Quiz Instructions

General directions:

0.) This is a group project. Only the group representative will receive this specification.

1.) Take a snapshot/screenshot of your project specification as a reference. That will be your project specification regardless of the attempts.

2.) There is no set time limit for the session when you open the project. After reading the project specification, you can navigate away or close the page. When you go back to the page, it will allow you to resume. DO NOT press submit since you are NOT going to submit here.

3.) Upload the screenshot of your project specification here

filename: SurnameFirstcharofyourfirstname1.* (surname max of 7 characters only; example:delacruj1.*) [where *- jpg, bmp, png, pdf]

*Note: Surname_name of the representative

Submit your deliverable(s) labeled "SIMD programmng project submission here".

Reminder again: DO NOT SUBMIT MAIN DELIVERABLES HERE, except the screenshot of the project specification. Any main submission done here WILL NOT be considered.

Question 1 0 pts

Write the kernel in (1) C program; (2) an x86-64 assembly language; (3) x86 SIMD AVX2 assembly language using XMM register; (4) x86 SIMD AVX2 assembly language using YMM register. The kernel is to perform 1-D stencil of vector *X* place the result in vector *Y*.

Input: Memory location *n* (integer) contains the length of the vector; Vectors *X* and *Y* are both **32-bit integer**.

Process:

$$Y[i] = X[i-3] + X[i-2] + X[i-1] + X[i] + X[i+1] + X[i+2] + X[i+3]$$

Example: X-> 1,2,3,4,5,6,7,8; output: Y-> 28, 35

Output: store result in vec*tor Y.* Also, display the result of first ten elements and last 10 elements of vector *Y* for all versions of kernels.

Note:

1.) Write a C main program to call the kernels of the C version, non-SIMD x86-64 assembly language, SIMD x86 assembly language using the XMM register, and SIMD x86 assembly language using the YMM register.

2.) Time the kernel portion only. Do this for both DEBUG and RELEASE versions of Visual Studio

3.) For each kernel version, time the process for vector size *n* = {2^{20}, 2^{26}, and 2^{30}}. If 2^{30} is not possible, reduce it to 2^{28} or $2^{29,}$ depending on the memory capability of your machine.

3a.) Initialize also vector size of with extra elements to check if your program can handle boundary situation. For example if SIMD register can load 4 elements at a time but the vector size is 6 elements, how will your system handle "boundary" elements?

4.) For each kernel, execute at least 30 times and get the average execution time.

5.) For the data, initialize each vector with values of your choice. Please document this value.

6.) 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 and both SIMD versions must be checked with the C version and output correspondingly (i.e., x86-64 kernel output is correct, etc.).

7.) Place the result in Github (ensure I can access your Github). The GitHub report be inline and NOT linked to external files.

8.) The GitHub repository contains the following:

a.) Readme section containing the following report:

i.) screenshot of the program output with execution time for all cases

ii.) comparative table of execution time as well as analysis of the performance of different kernels (how many times faster, why is it faster, etc.)

iii.) screenshot of the program output with correctness check (C)

iv.) screenshot of the program output, including correctness check (x86-64)

v.) screenshot of the program output, including correctness check (SIMD XMM register)

vi.) screenshot of the program output, including correctness check (SIMD, YMM register)

vii.) Discuss the problems encountered and solutions made, unique methodology used, AHA moments, etc.

b.) Visual Studio project folder containing complete files (source code: x86-64 non-SIMD, x86-64 SIMD version, and all other required files) to load and execute your program.

Rubric:

C main program with initialization and correct call/passing parameters to C, x86-64, and SIMD version	5
Correct output (C version)	10
Correct output (x86-64)	10
Correct output (SIMD XMM)	10
***with boundary handling	5
Correct output (SIMD YMM)	10
Comparative result	5
Analysis of result	15
Discussion	10
Completeness of file uploaded	5
following instructions	5

Edit View Insert Format Tools Table

12pt Paragraph

p

0 words

Quiz saved at 3:01pm

Submit Quiz