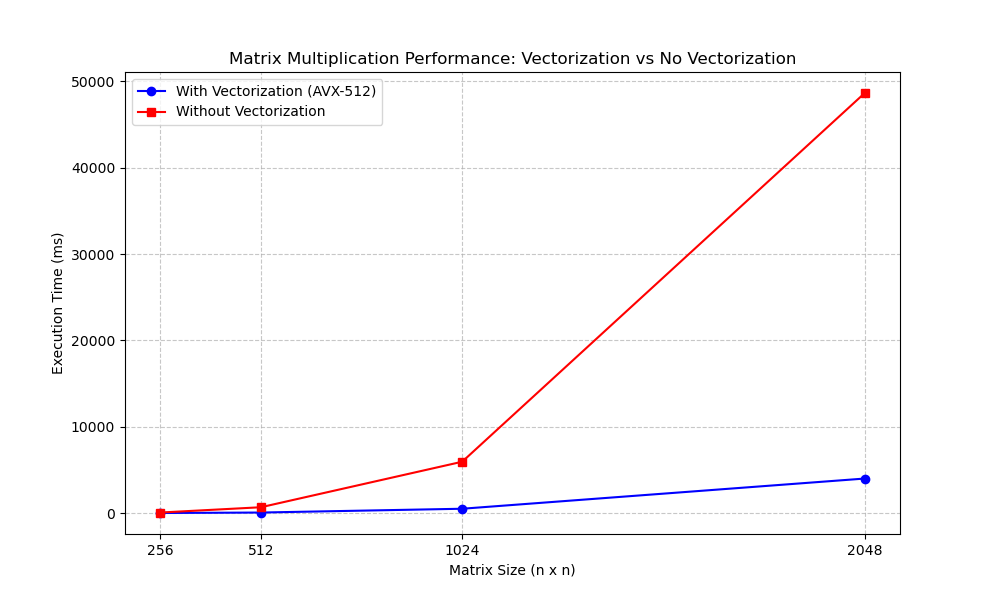
Question 2:

a.) Using the dotproduct.c example provided, develop an AVX512-accelerated version of matrix-vector multiplication. The example should give you a good start on the AVX intrinsics that you need to use in the program. Report on the speedup that you obtain as compared to a matrix-vector multiplication that does not use vectorization.

This table shows the runtime of matrix multiplication with vectorization versus with no vectorization for different matrix sizes.

|  |  |  |
| --- | --- | --- |
| Matrix Size (nxn) | Vectorization (ms) | No Vectorization (ms) |
| 256x256 | 5.036 | 58.819 |
| 512x512 | 58.097 | 671.758 |
| 1024x1024 | 497.88 | 5943.395 |
| 2048x2048 | 3996.381 | 48656.795 |

With vectorization, the speedup changes by a magnitude of 10, 100, 1000, and 10,000. The magnitude of speedup is much more visible as the size of the matrices grows.

b.) Using the same code, generate an assembly listing (using the -S flag) and identify 3 different vector instructions that the compiler generated, explaining their operation.

To find vector instructions, I looked for single instruction, multiple data operations (SIMD).

Instruction 1: **vbroadcastss –572(%rbp), %zmm0**

Vbroadcastss means “Vector broadcast scalar single-precision floating-point.” This instruction is used under .L25. It broadcasts a single float from memory to all 16 elements of the zmm0 register to prepare for SIMD computation. A 512-bit AVX register holds this broadcast’s value, so that all lanes of the vector register can operate in parallel.

Instruction 2: **(Line 1) vmovups (%rax), %zmm0, (Line 2) vmovaps %zmm0, -368(%rbp)**

There are a collection of vector loads and store instructions with vmovups and vmovaps. Vmovups moves the packed floats from memory to vector registers, and vmovaps stores both data from registers to memory and moves data between memory and registers. By loading all data into the registers using these lines and the same commands after them, the data in these registers can be used for SIMD operations in matrix multiplication.

Instruction 3: **vfmadd132ps %zmm2, %zmm1, %zmm0{%k1}**

Vfmadd132ps is a fused multiply-add instruction that performs zmm0 = zmm0 \* zmm2 + zmm1. This stores the result of the following operation in the register zmm0, and a mask register is used here so that only elements with this mask execute this operation. Zmm0, zmm1, and zmm2 contain floating point values, and k1 is a 16-bit mask register. This is valuable for operations on selective elements of a vector. Mask registers enable these operations with SIMD, and the parallelism greatly improves performance.

Snapshot of the Assembly:

