Question – 2

Assignment - 3

In this problem, explore the benefits of compiling on the Explorer cluster with floating point vector extensions (e.g., AVX). To allocate a node on Explorer with AVX512 support, you will need to specify "—constraint=cascadelake". To utilize AVX512 instructions, make sure to compile using the -mavx512f flag.

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.
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.

2 (a):

This is a speedup on MV using AVX512 that should take about X ms to complete a 10000 times another 10000 matrix and a 1000-length vector or so initialized to 1.0, whereas the non-vectorized version would take about Y ms or so. Therefore speedup is equal to Y/X or better said, AVX512 does vectorization 16 elements in parallel with fused mul-add operations.

```
[sridhar.pray@explorer-02 ~]$ nano matvec_avx512
[sridhar.pray@explorer-02 ~]$ nano matvec_avx512.s
[sridhar.pray@explorer-02 ~]$ ./matvec_avx512 10000 10000
Time naive (ms):    155.768127
Time AVX-512 (ms):  27.347706
Speedup:            5.695839
Max diff:           0.019531
```

2 (b):

Here are three specific vector instructions from the assembly output and a brief explanation of each:

1) **vmovups**:
   - **Assembly Instruction**: vmovups (%rdx,%rax), %zmm6
   - **Description**: This instruction loads unaligned packed single-precision floating-point values from memory into a ZMM register. In the context of matrix-vector multiplication, it's used to load segments of the matrix A or vector x into registers for vectorized computation.

2) **vfmadd231ps**:

- **Assembly Instruction**: vfmadd231ps (%rsi,%rax), %zmm6, %zmm1
- **Description**: This is a fused multiply-add instruction, which performs multiplication and addition in a single operation to maintain precision and reduce latency. Specifically, vfmadd231ps multiplies the elements of the second and third operands (%zmm6 and memory at (%rsi,%rax)) and adds the result to the elements of the first operand (%zmm1). This is crucial in the inner loop of matrix-vector multiplication for accumulating dot products efficiently.

3) **vaddss**:

- **Assembly Instruction**: vaddss %xmm1, %xmm0, %xmm0
- **Description**: This instruction adds scalar single-precision floating-point values from the lower bits of the second operand (%xmm1) and the first operand (%xmm0) and stores the result in the destination operand (%xmm0). It is typically used here to accumulate the results of different vector operations into a single scalar result, which might represent a single element in the result vector y.