CIS 314 Project 6 Written Solutions

========
Problem 2
========

a) The only line that cannot be parallelized is adds %xmm0, %xmm1. This corresponds to the line in the function where we have sum += u[i] * v[i]. The only ops that cannot be parallelized are those that depend on previous loop iterations to perform their calculations, and the only op that does is sum +=. In every loop we update our value of sum to be sum + u[i] * v[i], which means every subsequent iteration of the loop needs to remember the value of sum from the previous iteration to perform the sum + u[i] * v[i] part, hence it cannot be parallelized.

b)
.L87:

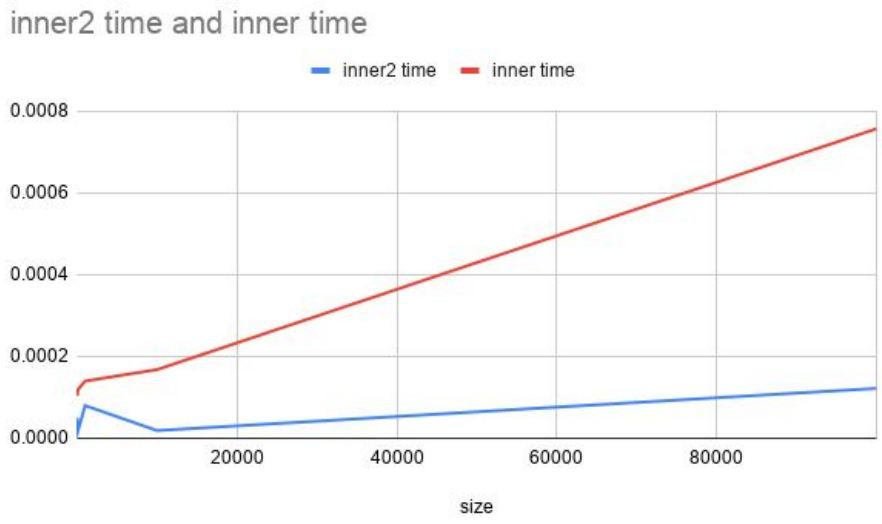
These two lines compose the u[i] * v[i] multiplication. We know that each multiplication requires 5 clock cycles.

```
movss (%rbx, %rdx, 4), %xmm0 # Get u[i]
mulss (%rax, %rdx, 4), %xmm0 # Multiply by v[i]
```

The following three lines are all contained within the float addition, which we know composes 3 clock cycles.

```
adds %xmm0, %xmm1 # Add to sum
addq $1, %rdx # Increment i
cmpq %rcx, %rdx # Compare i to length
```

Thus, we have 5 cycles + 3 cycles = 8 total cycles for our best case CPE.
d)



I used the sizes 10, 100, 1000, 10000, and 100000 to perform my tests. Based on the graph, I think it can be said for certain that inner2 is much more efficient than the original inner. They start off pretty similar, but we can see that as our sizes increase, inner2 remains relatively quick while inner itself is ramping up to significantly longer processing times.