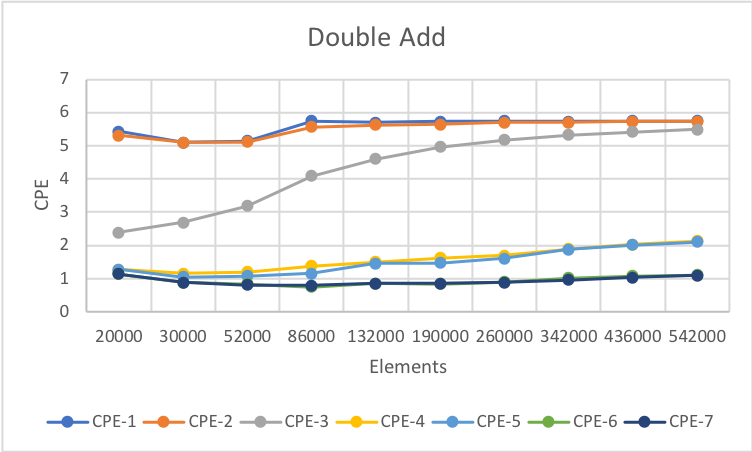
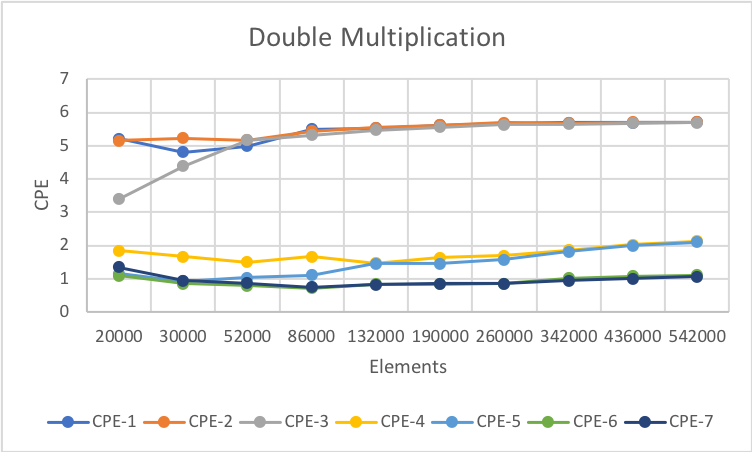
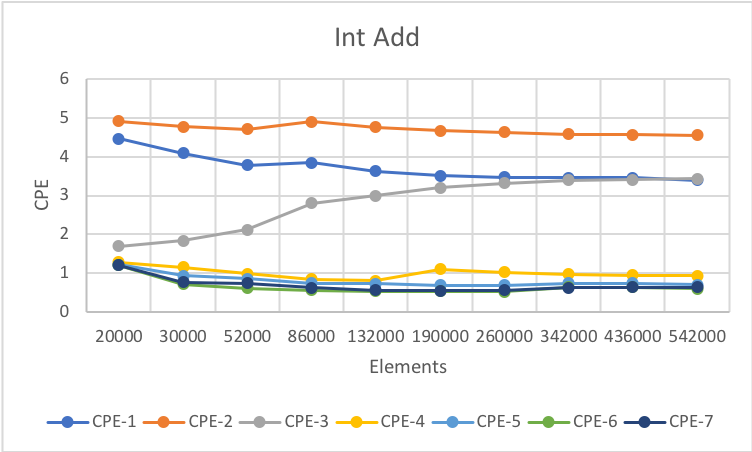
EC527 Assignment 2 Chen-Yu Chang

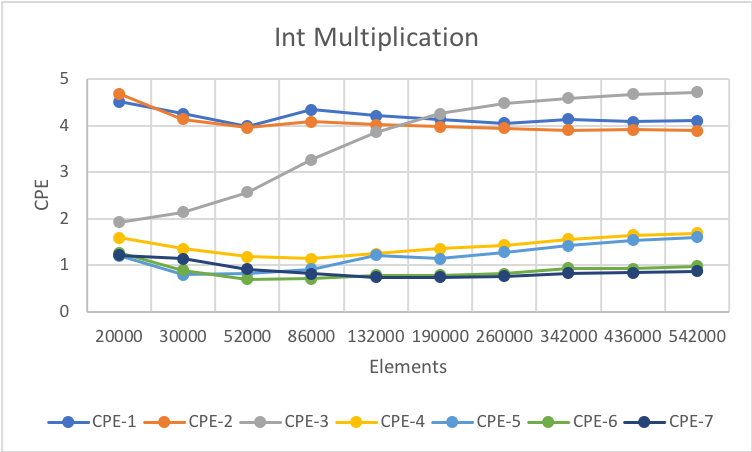
Part 1:

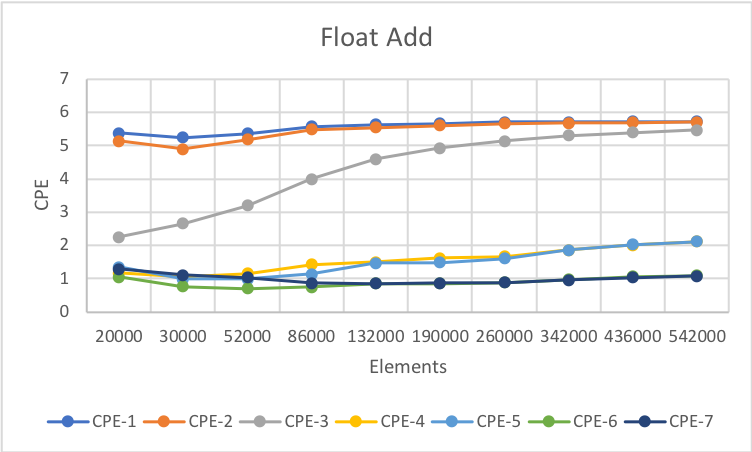
1a.

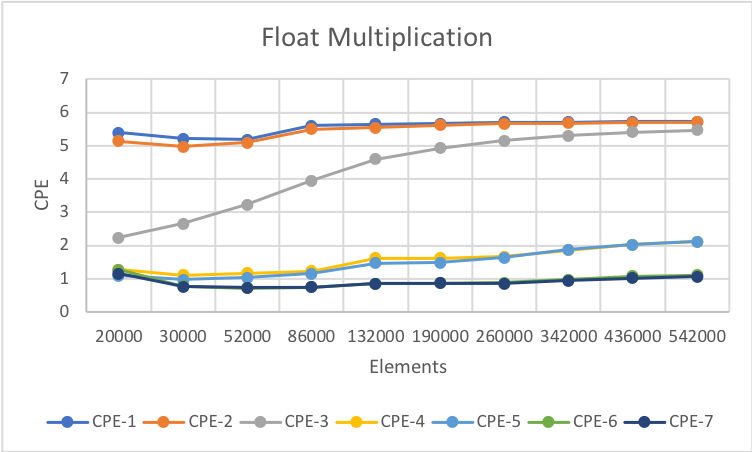






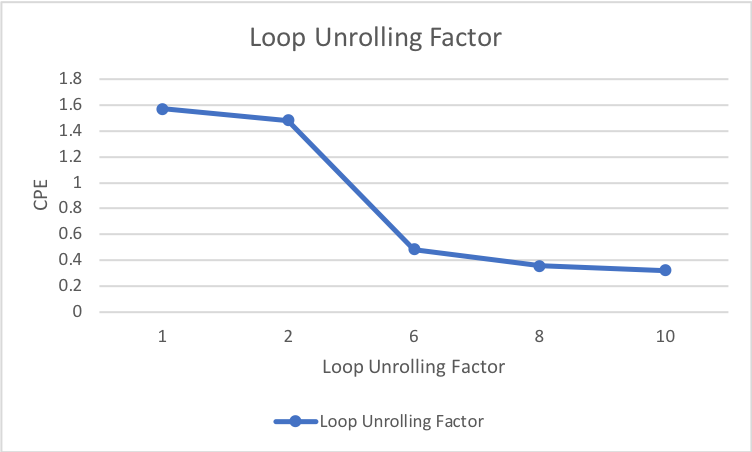






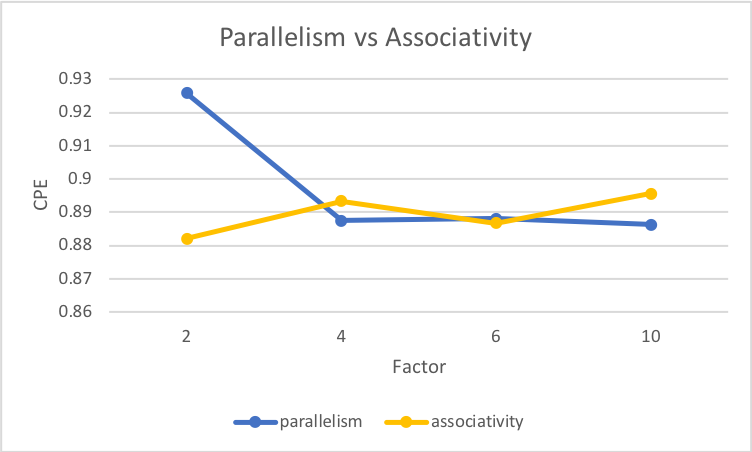
Overall, the results I got are quite similar to what it is given in the book. CPE tends to decrease for every optimization from combine 1 to combine 7. Even though some type of data has a sudden spike, most tests still follows the general trend. The CPE does not vary too much since they are all about 1 to 5. When using float and double, it should take more time than int, and generally, multiplication spends more time than addition.

1b.



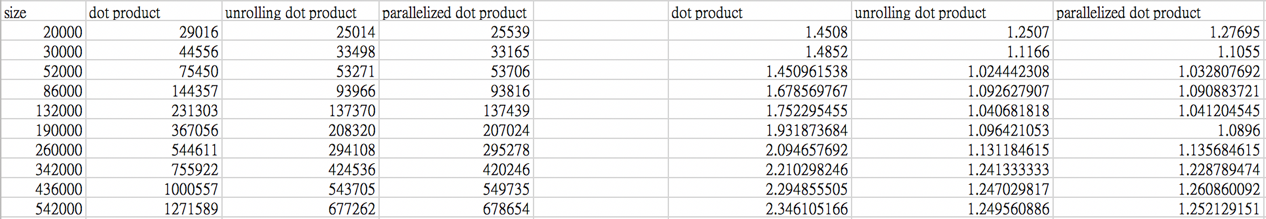
From the graph, increasing unrolling factor will make CPE decrease. However, after the factor reach 6, the difference of factor does not seem to have a dramatic change. I think that it increased the number of operations in a single iteration, so less unrolling factor would perform more operations.

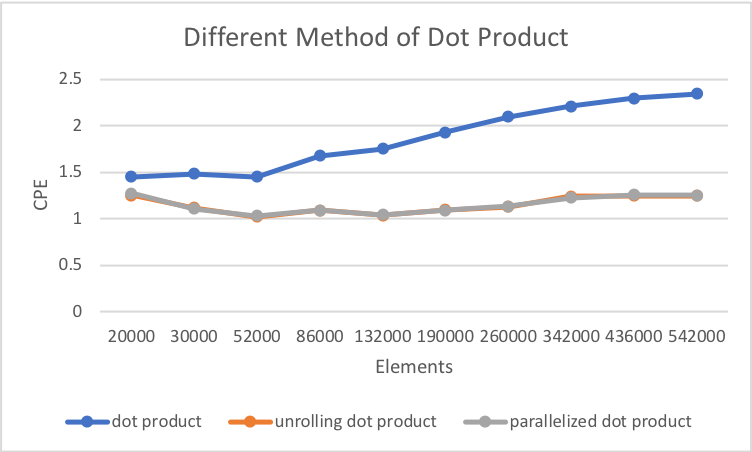
1c.



From the data we obtained, I found that the smaller loop unrolling factor performs better than bigger, while the data on the graph is an average, so from the excel file in the file, we can see that smaller factor performs better. Overall, those whose factor is smaller than 6 have a major change on the CPE, while for parallelism, it tends to be stagnant, and for associativity, it starts to increase.

Part 2:





Optimization Methods:

1. Unrolling Dot Product:

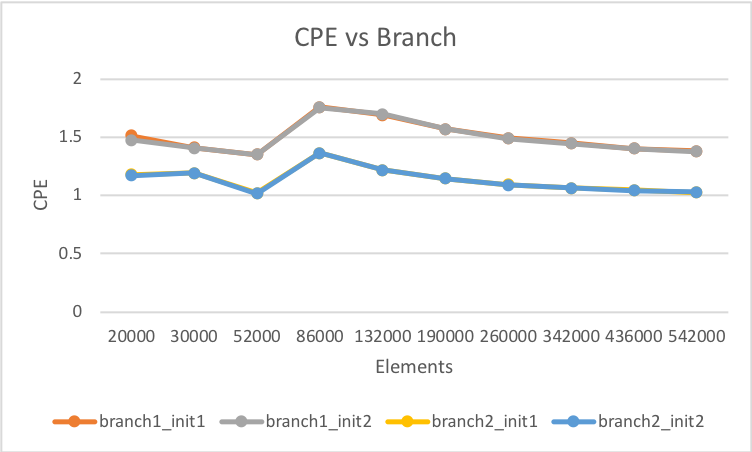
This method decreases the number of iterations by increasing the operations of elements per iteration. Also, it operates the dot product and adds it to the accumulator. This improves the performance in the way of reducing the operations.

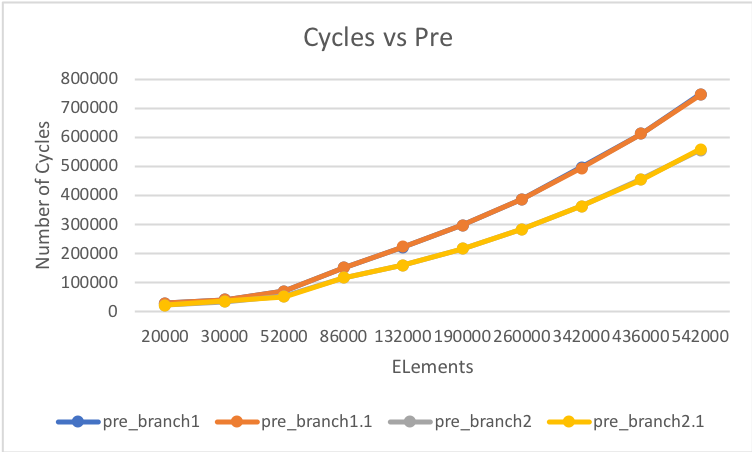
1. Parallelized Dot Product:

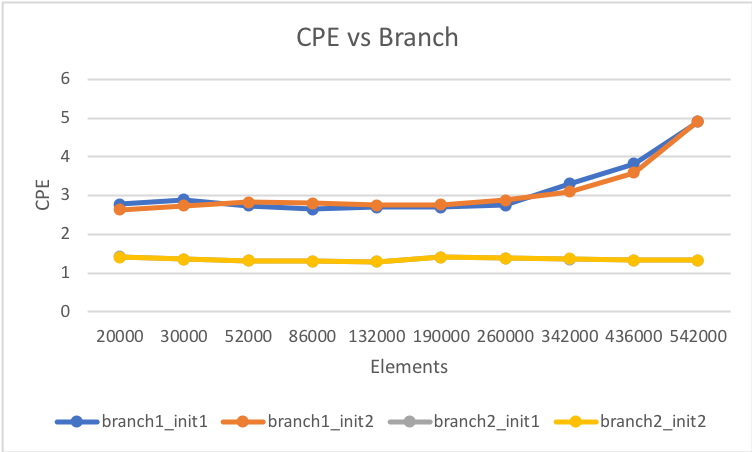
This method separates the operations so that it increases the performance. Therefore, it combines set of operations into two parts and combines the results in the end.

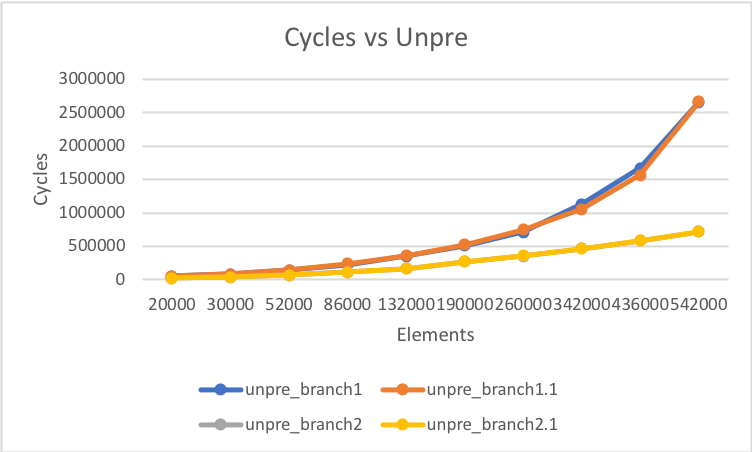
Overall, we can see that the two optimized methods all decreases the CPE of the dot product compared to the simplest way of dot product.

Part 3:









The top 2 graphs are predictable branches and the bottom 2 graphs are unpredictable branches. Overall, we can see that branch 2 is faster than branch1, which has a lower CPE. Also, the predictable branches have a lower CPE than the randomized branches. Branch 2 contains a smaller loop than branch 1 in the assembly language, so branch 2 is faster. Furthermore, processor is more accurate in predictable branches.

Part 4:

4a. It took about 8 hours.

4b. It took more time on writing the dot product code.

4c. Not really.

4d. Not now.