Lab 2
Chih Han “Josh” Yeh
ENGEC 527 Spring 2025

Part 1.

a.
For the following plots and data generation, I chose the following variables and coefficients that would fit right under the size of the L1 cache and the number of data points covering a large range of data.

Modified Variable/Coefficients	Value
NUM_TESTS	20
A (double, float, int, long double)	11, 22, 22, 5
B	0, 0, 0, 0
C	1, 1, 1, 1

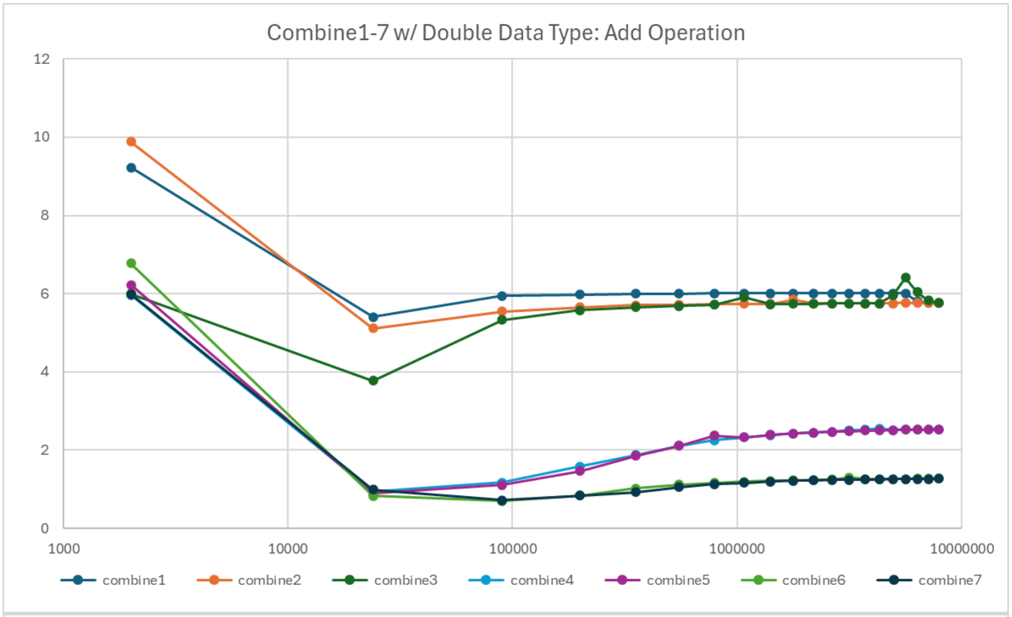
Maximum Array Sizes covered for each data type with corresponding coefficients:

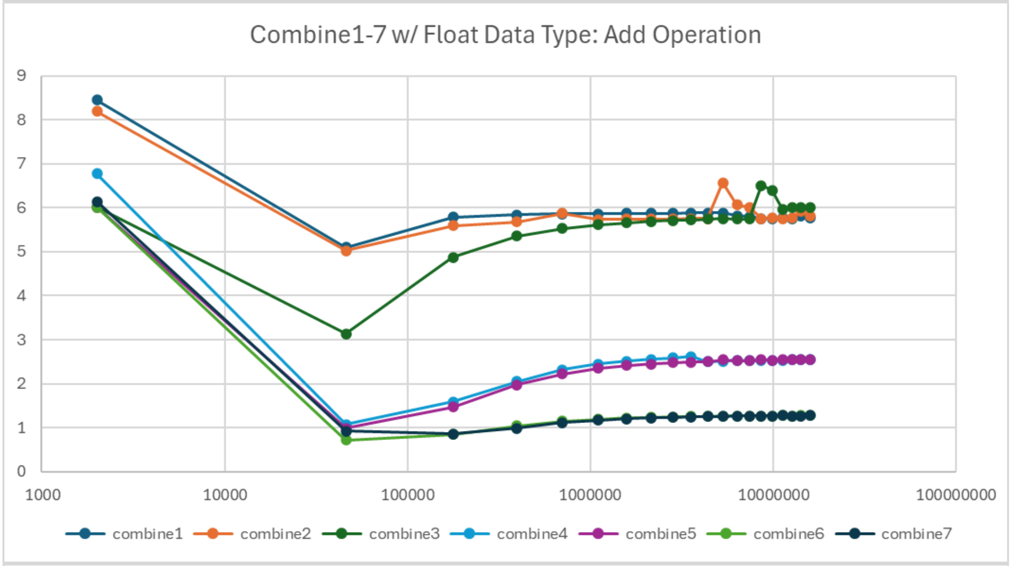
Data Type (and equation)	Memory
Int ($22x^2 + 1$)	31772 Bytes (31.8KB)
Float ($22x^2 + 1$)	31772 Bytes (31.8KB)
Double ($11x^2 + 1$)	31776 Bytes (31.8KB)
Long Double ($5x^2 + 1$)	28896 Bytes (28.9KB)

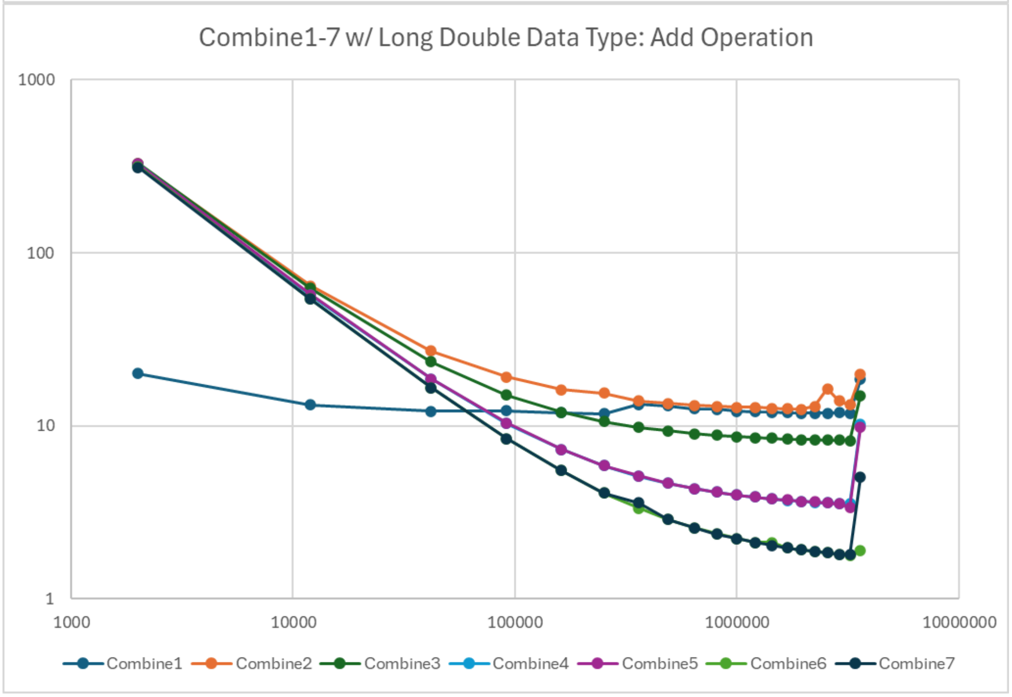
Below are the average CPE of different methods for different operations and data types

Data Types	Int		Double		Float		Long Double	
Methods	+	*	+	*	+	*	+	*
combine1	3.76	4.88	6.10	6.09	5.92	5.84	12.93	12.91
combine2	3.55	4.29	5.91	6.13	5.91	5.91	33.06	32.67
combine3	3.46	4.78	5.69	5.62	5.66	5.71	28.93	27.72
combine4	1.14	1.99	2.41	2.43	2.59	2.51	24.31	23.25
combine5	0.90	2.00	2.41	2.42	2.51	2.51	24.41	23.40
combine6	0.81	1.20	1.44	1.43	1.43	1.43	21.77	21.79
combine7	0.93	1.20	1.39	1.40	1.44	1.43	21.75	21.72

Below are some plots to display a distinct difference in CPE between data types with the add (+) operation (CPE vs Log (Array Size)).

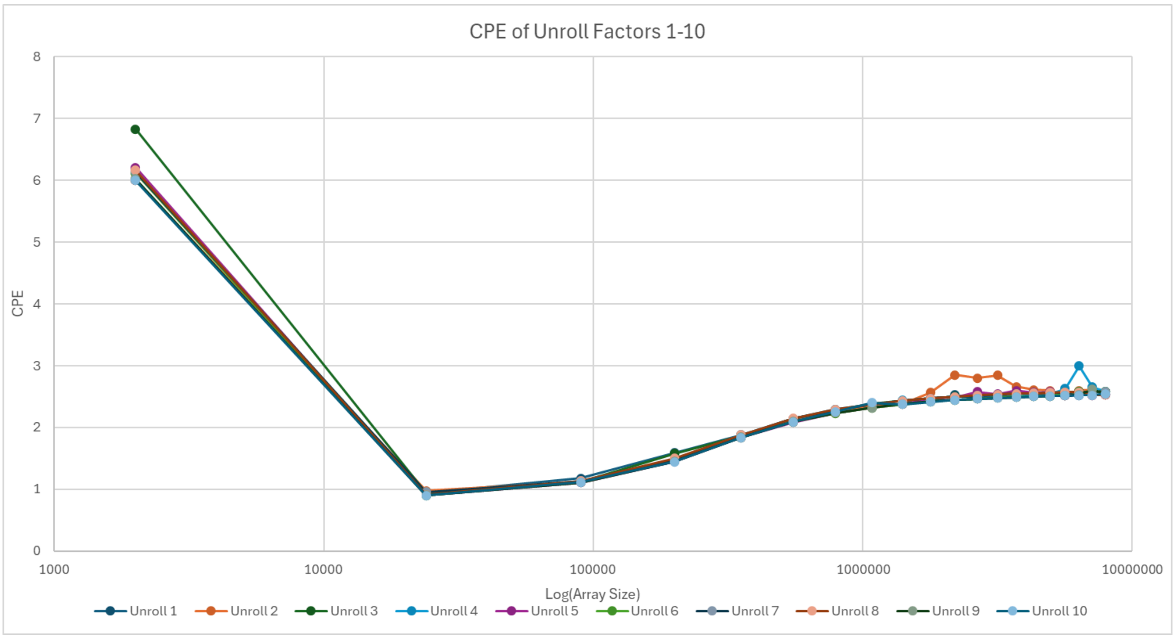






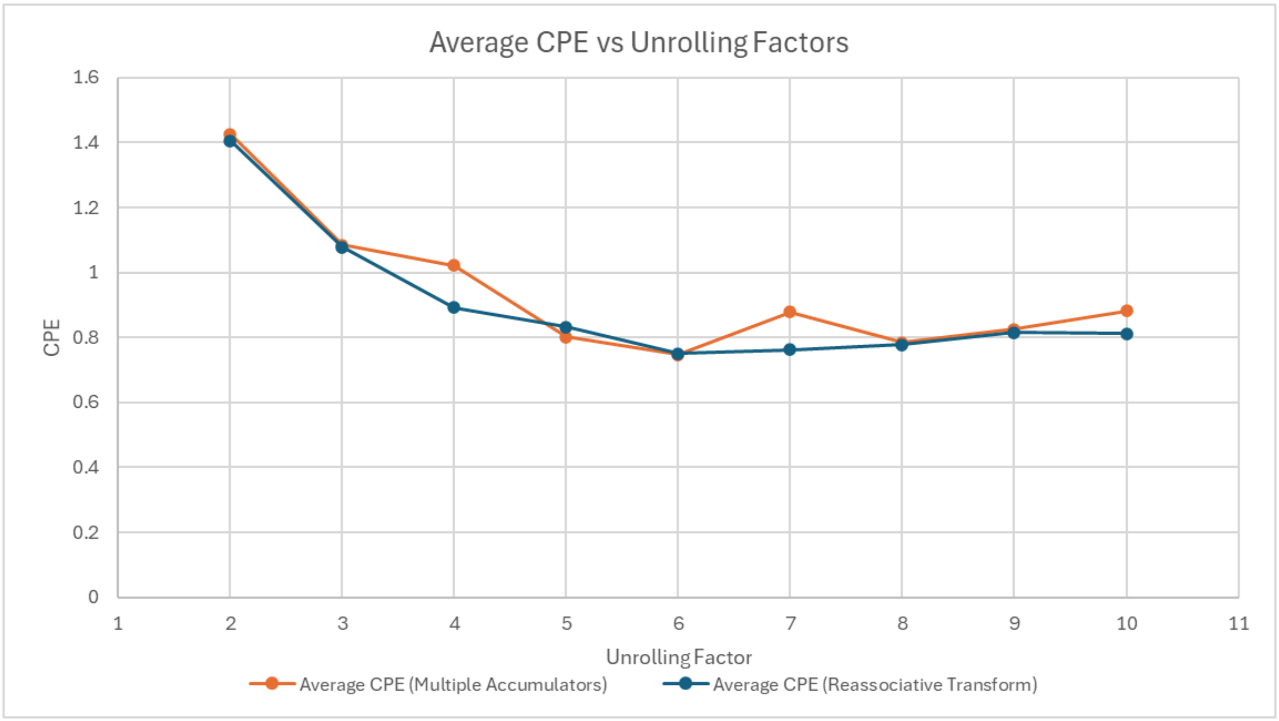
b.

i. Below is the CPE comparison between unrolling factors from 1 through 10 with the **double** data type and **+** (addition) operation.

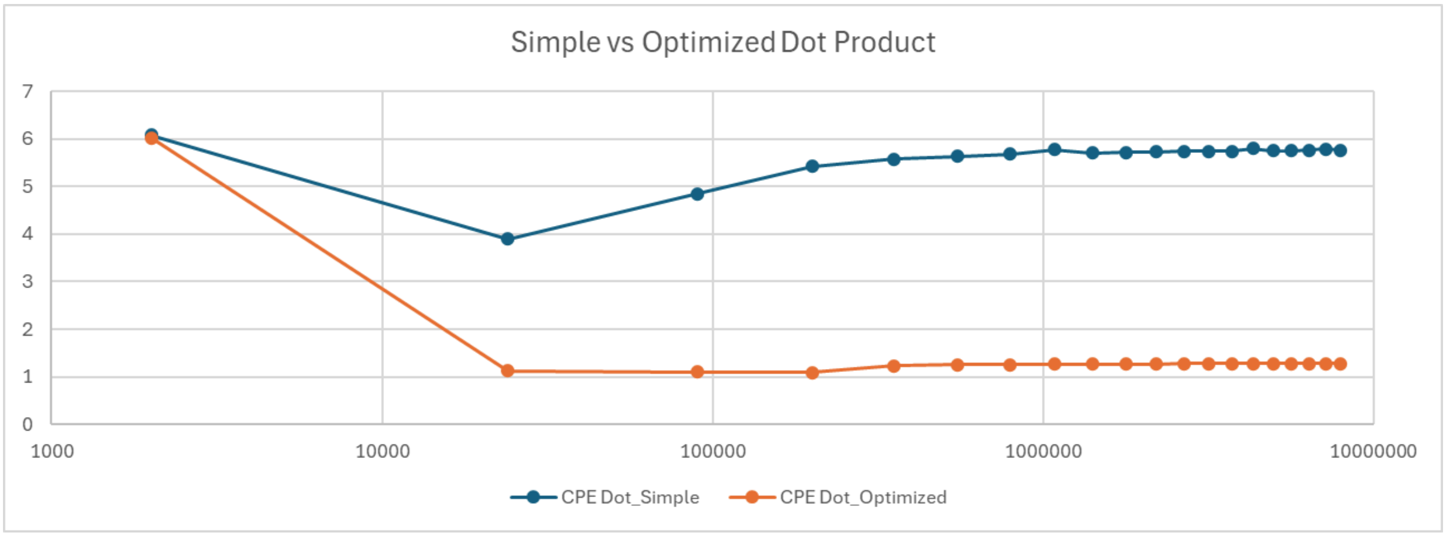


ii. The figure is comparable to the plot provided in the textbook. The CPE between each unrolling factor is close. The only noticeable difference is the starting CPE, and the slight jump for unrolling factor 2 when the array size ranges between 200,000 to 400,000; and unrolling factor 4 at an array size of 700,00.

c. Below is a plot comparing the average CPE against unrolling factors featuring the two parallelization methods of accumulators and re-associative transform with unrolling factors from 2 through 10.



Part 2.



Above is my generated plot comparing the CPEs of Simple Dot Product and Optimized Dot Product methods. Since the data type and plotting parameters match that of the data type "double", I made sure my quadratic equation and other parameters were kept the same if not similar to problem 1:

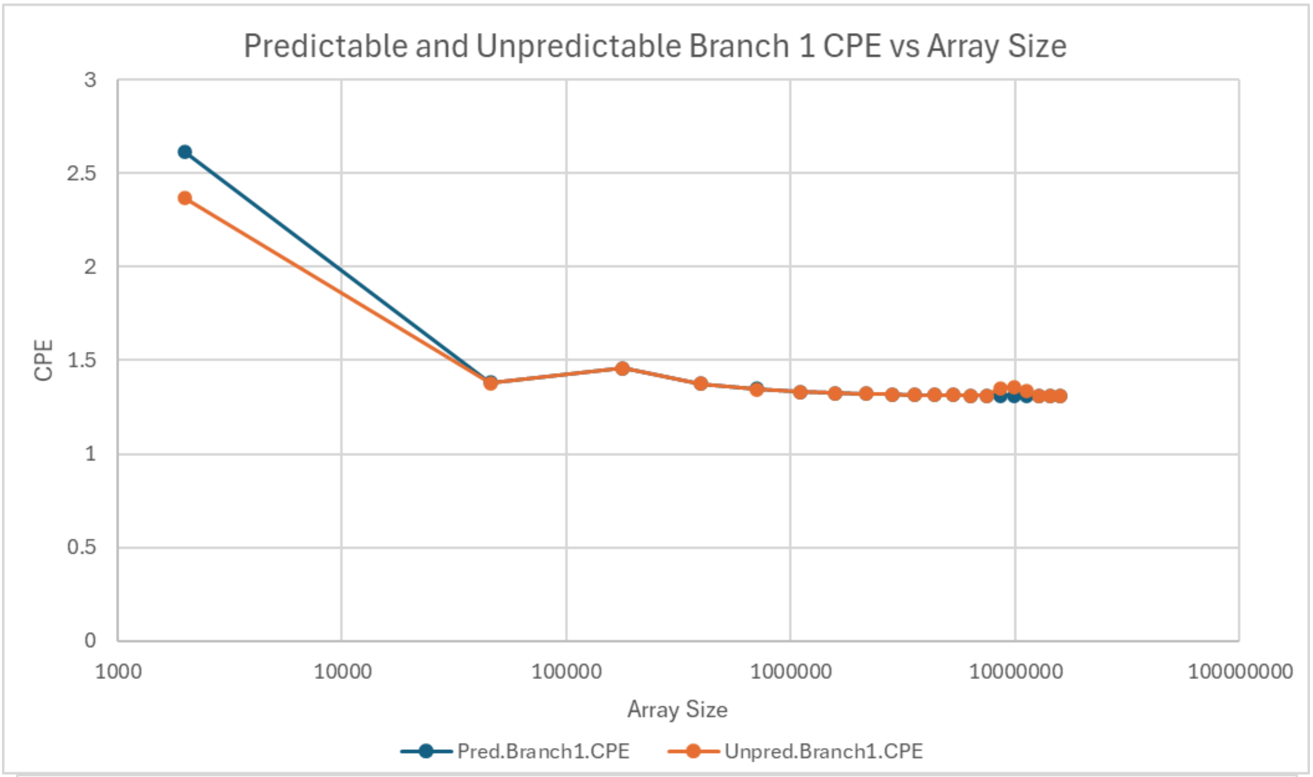
Parameters	Value
NUM_TEST	20
A, B, C	11, 0, 1 $(11x^2 + 1)$

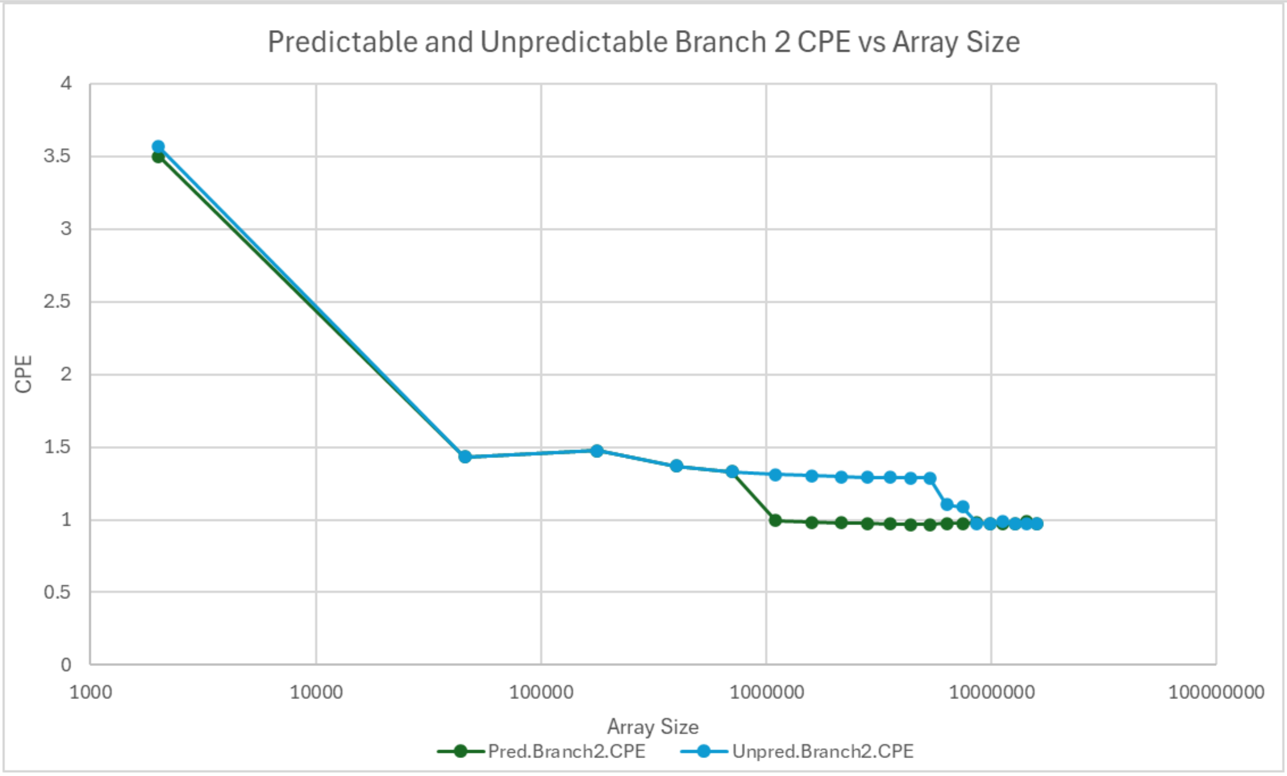
The optimization method I chose is loop unrolling, and in the case of the dot product, I chose an unrolling factor of 2, in which the instruction would add the dot product of a specific index and one of the next index after. I was expecting the performance of loop unrolling to be double that of the simple method, comparing the CPE values, the optimized version seems to have over 4.5x the performance of the simple dot product.

Part 3 in next page

Part 3.

Below are two plots comparing the performance of branch1() and branch2() using the predictable and unpredictable data types. The plot shows the CPE plotted against the log size of the different branch methods, highlighting the difference between random and predictable data generation between the two branch methods. The setup and parameters for output generation are the same as problem 1 for the float data type.





The method I used for the predictable data was: (data_t)(i % 100). Meaning data[i] = i%100, so the number would repeat every 100 turns, making it a predictable pattern. The unpredictable data was generated using (data_t) fRand(0.0, 100.0), meaning data[i] = a random number between 0.0 to 100.

The CPE of both data for branch1() are the same, or at the very least very close to each other their deviation is negligible. The assembly code describes two different code paths, with one specifically branching if the initial comparison is true (jump to .L6 if n1 <- n2).

branch2() has a more pronounced difference between the data sets. When the array size reaches above 100,000 the CPEs for predictable branch2() are significantly faster than the unpredictable branch, with performance up to 1.3x of the predictable data set.

Part 4.

a. This particular assignment took almost 20 hours of total work time to read, understand, and also data generation as there are numerous data sets that are required to be compared against each other.
b. Part 1 took the longest but not really in terms of excessive time consuming.
c. I am still only a beginner at C and reading both the main code and Assembly posed an issue.
d. Not particularly no