I used google benchmark for my measurements. I randomize the numbers and seed every time and after running each part several times I believe my report reflects the results on average.

```
user@DESKTOP-LAVVV9R:~/assignments$ g++ -std=c++17 -O2 -I/usr/local/include -L/usr/local/lib task1.cpp -lbenchmark -lpthread -o sum
user@DESKTOP-LAVVV9R:~/assignments$ ./sum
2025-11-23T21:09:06+01:00
Running ./sum
Run on (28 X 3417.6 MHz CPU s)
CPU Caches:
  L1 Data 48 KiB (x14)
  L1 Instruction 32 KiB (x14)
  L2 Unified 2048 KiB (x14)
  L3 Unified 33792 KiB (x1)
Load Average: 0.00, 0.00, 0.00
----------------------------------------------------
Benchmark           Time             CPU   Iterations
----------------------------------------------------
Part1          220497 ns       230940 ns         3013
Part2          711523 ns       745223 ns          944
Part3          320395 ns       335573 ns         2088
Part4          315015 ns       304175 ns         2312
user@DESKTOP-LAVVV9R:~/assignments$
```

Here is an image of all of the parts running and their results.
Part 1:

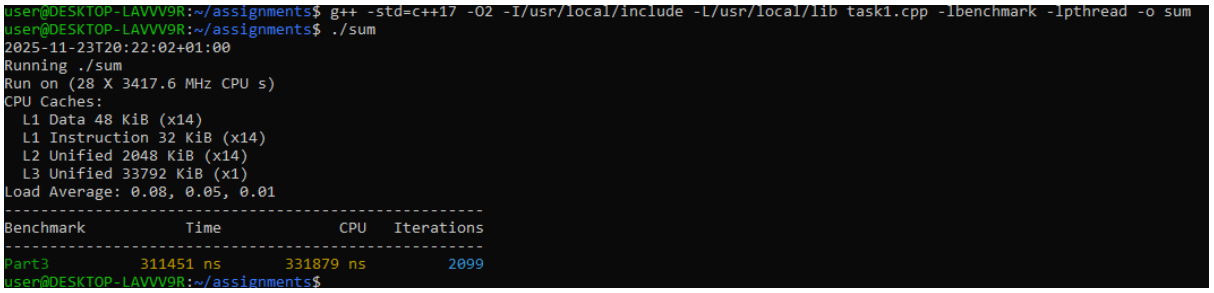
```
user@DESKTOP-LAVVV9R:~/assignments$ g++ -std=c++17 -O2 -I/usr/local/include -L/usr/local/lib task1.cpp -lbenchmark -lpthread -o sum
user@DESKTOP-LAVVV9R:~/assignments$ ./sum
2025-11-23T21:15:13+01:00
Running ./sum
Run on (28 X 3417.6 MHz CPU s)
CPU Caches:
  L1 Data 48 KiB (x14)
  L1 Instruction 32 KiB (x14)
  L2 Unified 2048 KiB (x14)
  L3 Unified 33792 KiB (x1)
Load Average: 0.04, 0.01, 0.00
----------------------------------------------------
Benchmark           Time             CPU   Iterations
----------------------------------------------------
Part1          227368 ns       227374 ns         3119
user@DESKTOP-LAVVV9R:~/assignments$
```

Part 2:

```
user@DESKTOP-LAVVV9R:~/assignments$ g++ -std=c++17 -O2 -I/usr/local/include -L/usr/local/lib task1.cpp -lbenchmark -lpthread -o sum
user@DESKTOP-LAVVV9R:~/assignments$ ./sum
2025-11-23T20:09:07+01:00
Running ./sum
Run on (28 X 3417.6 MHz CPU s)
CPU Caches:
  L1 Data 48 KiB (x14)
  L1 Instruction 32 KiB (x14)
  L2 Unified 2048 KiB (x14)
  L3 Unified 33792 KiB (x1)
Load Average: 0.14, 0.09, 0.02
----------------------------------------------------
Benchmark           Time             CPU   Iterations
----------------------------------------------------
Part2          700285 ns       740894 ns          931
user@DESKTOP-LAVVV9R:~/assignments$
```

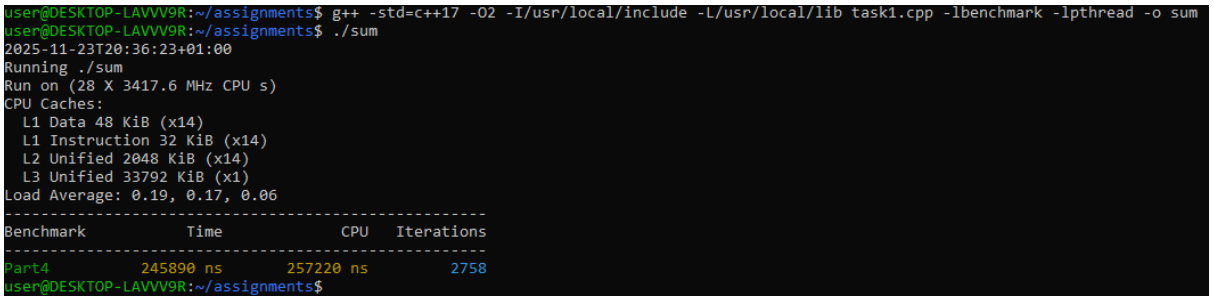
The time and CPU usage for part 2 is around 3 times higher for both.
This makes sense as significantly more operations are performed which requires more CPU usage. As iterations are significantly lower than any other task, pipelining is possibly a large contributing factor to performance as it will allow for more instructions and calculations to be done per cycle. As I ran these tests on a powerful PC, the amount of cores and fast cache reduces the time it takes to complete instructions and will reduce the time the CPU waits for data to move between memory.

Part 3:



The performance of part 3 is a lot better than part 2. One reason it may be faster is because the values have to be accessed from the array 2 times as much in part 2 than part 3 which increases the time the CPU is waiting for values to transfer across memory and registers. The compiler also may be more optimized when everything is in one expression.

Part 4:



It doesn’t affect throughput much. The interactions are lowered but CPU and Time actually increase. The if statement will make branches and any mis prediction of the branches will impact the performance. The slowdown may also be because not enough calculations are skipped as 500000 may be too high to impact the performance.