PACS LABORATORY 3

by

Daniel Ariza Antón (775545) and Marcos Ilarraza Sarto (797720)

All the tests realized in this lab have been realized in the *pilgore.cps.unizar.es* machine because the computers of the team can't execute the *perf* command for not executing natively Unix (WSL doesn't support it). If the case is different, it will be mentioned in the propper section.

QUESTION 1

The function std::stoll's purpose is to convert a string object into a long long object, however, a long long object can store up to 16 bytes and the code is using size_t, which can store up to 8 bytes, so the function std::stoll is incorrect in this case.
As we have deemed that we don't need to use the function std::stoll instead we decided to use std::stoul which can save up to 8 bytes of a number without sign which let's us to save up some memory space and further optimize the number of cycles required to the execution of the program.

QUESTION 2

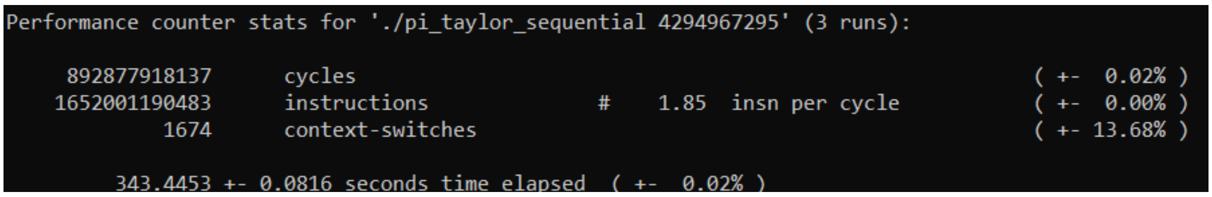
The times we have measured are the following:

Time\Steps	16	128	1.024	1.048.576	4.294.967.295
Time	6us	14us	83us	80ms	343.495ms

With some calculations, we can check with the three three numbers of iterations are multiplied by 8 in each case. From the first to the second the increase of time is of 2'33, and from the second to the third is of 5'93. This already proves that **it's not linear with the number of steps**. The fourth amount of steps is 1.024 times the third, and the time relation is of 963'86. The fifth is around 4.095 times the fourth, and the increase of time is of 4.293'69.

QUESTION 3

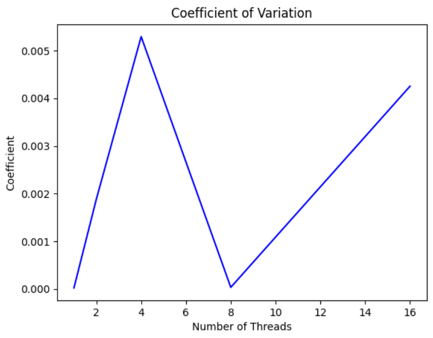
In order to extract this information, we executed the perf command 3 times to avoid spurious. The results were the following:

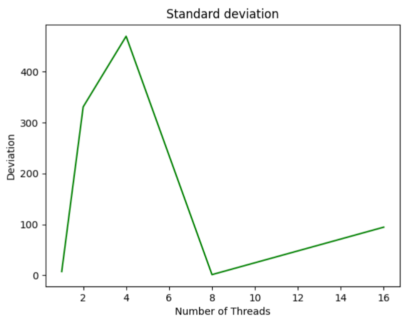


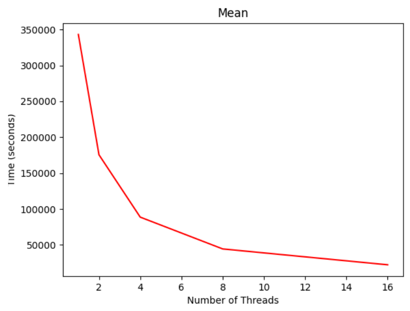
```
Performance counter stats for './pi_taylor_sequential 4294967295' (3 runs):

    892877918137      cycles                                                  ( +-  0.02% )
   1652001190483      instructions              #    1.85  insn per cycle     ( +-  0.00% )
            1674      context-switches                                        ( +- 13.68% )

       343.4453 +- 0.0816 seconds time elapsed  ( +-  0.02% )
```

QUESTION 4

We repeated each experiment three times to calculate its means and standard deviations, and from it calculate their coefficients of variation:







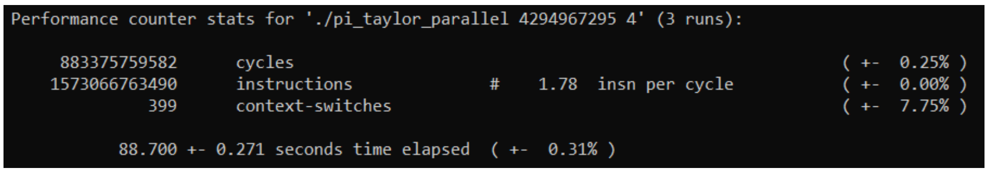
As we can see, the program improves greatly by dividing the task in different threads. When duplicating its amount, the time needed to finish is reduced by 50%. And even though the CV and standard deviation varies differently depending on the amount of threads, even seeming big in the graphics, their actual values regarding the units we are working with are small.

QUESTION 5

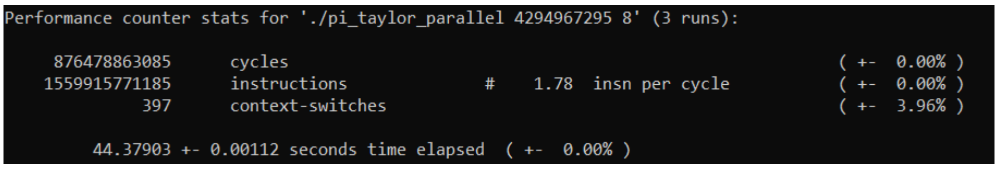
With the parallelization of the code, we only have changed the order in which the values of each step are added. If there are 4 threads, there will be four groups of steps: $a(x_o,...,x_a)$, $b(x_{a+1},...,x_b)$, $c(x_{b+1},...,x_c)$ and $d(x_{c+1},...,x_d)$. According to the ***Handbook of Floatina Point Arithmetic***, there is a rounding operation executed after every floating point operation. Therefore, in the sequential code the sums were: $o(o(o(o(x_o + x_1) + x_2)\dots+x_d)$.
But in the parallel version, the sums get the following order:
$o(o(o(o(o(o(o(x_o + x_1) + x_2)\dots+x_a) + o(o(o(o(x_{a+1} + x_{a+2}) + x_{a+3})\dots+x_b)) +$
$o(o(o(o(x_{b+1} + x_{b+2}) + x_{b+3})\dots+x_c)) + o(o(o(o(x_{c+1} + x_{c+2}) + x_{c+3})\dots+x_d)) =$
$o(o(o(a + b) + c) + d)$
According to the section *2.4* of the ***Handbook of Floatina Point Arithmetic***, the associativity property of the sum is lost because of this rounding. Therefore, even thought the results will be similar, they are not guaranteed to be exactly the same.

QUESTION 6

The results of the perf after 3 executions for cycles, instructions and context-switches are:



```
Performance counter stats for './pi_taylor_parallel 4294967295 4' (3 runs):

     883375759582      cycles                                                        ( +-  0.25% )
    1573066763490      instructions              #    1.78  insn per cycle           ( +-  0.00% )
              399      context-switches                                              ( +-  7.75% )

           88.700 +- 0.271 seconds time elapsed  ( +-  0.31% )
```



```
Performance counter stats for './pi_taylor_parallel 4294967295 8' (3 runs):

     876478863085      cycles                                                        ( +-  0.00% )
    1559915771185      instructions              #    1.78  insn per cycle           ( +-  0.00% )
              397      context-switches                                              ( +-  3.96% )

         44.37903 +- 0.00112 seconds time elapsed  ( +-  0.00% )
```