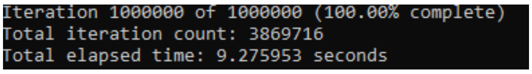
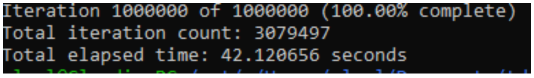
1. The von Neumann bottleneck is caused by the bus system used to transfer data between the part of the architecture. To overcome it it’s possible to implement a cache, that is a fast memory very close to the CPU that contains the last data fetched by CPU, this is possible thanks to spatial and temporal locality.
2. In this snippet of code we have, for sure a Write After Write dependency that is a Name Dependence since the instruction #4 and #6 write in the same memory location. Since is a for-loop has control dependence. And for instruction #5 since it need the result of instruction #4 is a Data Dependence.
3. The difference between SIMD and MIMD is the amount of instruction they can perform in a single clock cycle.
4. These 3 different solvers have very important differences:
    - Jacobi, since is an iterative method need a lot of iteration, and so time, to converge to correct result. Also, if is possible to parallelize it, seems like is not parallelized in this code.

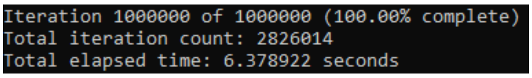
      ```
      Iteration 1000000 of 1000000 (100.00% complete)
      Total iteration count: 3869716
      Total elapsed time: 9.275953 seconds
      ```

    - Gauss-Seidel, it’s the more time-consuming method, because we introduce the data-dependence to perform the convergence.

      ```
      Iteration 1000000 of 1000000 (100.00% complete)
      Total iteration count: 3079497
      Total elapsed time: 42.120656 seconds
      ```

    - Red-black Gauss-Seidel, this method takes the advantage of Gauss-Seidel but with the possibility to parallelize more. In this way we can achieve the best results, in time and space complexity in this set of method.

      ```
      Iteration 1000000 of 1000000 (100.00% complete)
      Total iteration count: 2826014
      Total elapsed time: 6.378922 seconds
      ```