Explanation

For assignment 4, I used the previous code I had written of a ,,Sieve of Eratosthenes” in the last task. From what I gathered, openMP gives a programmer the ability to use parallelism without having to write any low-level details. In my case, previously I described, started and joined all the threads. In other words, the whole process of parallelism depended on me.

OpenMP in this case gave me the chance to write the same code with high-level constructs and as simple as it could get. The for cycle I had in the function that every thread launched just moved to the main section of the code and with just one sentence I did exactly the same thing I did previously.

So, by writing pragma omp parallel for line and specifying number of threads, I parallelized ,,for” cycle without having to manually launch every thread with the function that has that for cycle in it, the openMP does it for me. Also, by writing a schedule(static), the for cycle will be divided into chunks and passed onto threads in that way, again this prevents me from writing any additional code and dividing it in chunks on my own.

As for the for cycle itself, it iterates from square root of the number we inputted (num) to our num and inside is another for cycle from 2 to square root of the num, which gives us the seed, we check if the number is divisible by this seed and then mark it not prime if so. With the help of openMP the whole process happens parallelized on different threads and after returning, finishing all of the threads, we print the answers.

I also added the time to find out how long the process takes in milliseconds. For the numbers up to 10000 it takes 0 to 1ms and from there the number grows with just a few milliseconds, which is nearly the same result I had in the previous code and also, when I ran Sieve of Eratosthenes without parallelizing it, with just one difference, in some cases of big numbers this code ran just a little bit faster (1 or 2ms).