LAB 1 - USING OPENMP FOR ACCELERATING VECTOR MATRIX MULTIPLICATION

List of files:

- *matrix.c*: containing c source code for the matrix multiplication application
- *run.sh*: executing script
- *lab1.pdf*: this file - the description file

During the first lab, you are learning/practising how to accelerate an application with the OpenMP library. The application in this lab is vector matrix multiplication. Given a vector v with n element, a matrix M with $n \times n$ elements, the application computes the product vector $r[n] = v[n] \times M[n \times n]$ using both normal execution (sequential) and OpenMP execution (parallel).

How to use the script file? The script file contains all things you need for this lab. First, it compiles the application. Then it sets up the number of OpenMP threads. Finally, it executes the application. The output is the execution time for both normal execution and OPEMMP execution. To execute the application, you need to specify the number of OpenMP threads and the size of the matrix and the vector (value n). Below is an example in which we use 4 OpenMP threads and $n = 1000$.

```
$ > ./run.sh 4 1000
```

Here is the example output:

```
SEQUENTIAL EXECUTION: 0.013533 (sec)
PARALLEL EXECUTION WITH 4 (threads) ON 2 (processors): 0.012302 (sec)
```

It means that the execution time of the normal execution is $13.522ms$ and of the OpenMP is $12.302ms$. The OpenMP execution uses 4 threads on a 2-processor CPU (the number of processors depends on the CPU running the application).

What will you do during the lab:

1. **Task 1**: While the execution time of the normal execution is based on the size of data, the execution time with OpenMP library is based on both the size of data and the number of threads. In this task, you must find how many OpenMP threads is the most suitable for your system? To do this, you should keep a fixed data size (at a reasonable size) and change the number of OpenMP threads value. Then, you compare the speed-up of the OpenMP execution to the normal execution. The most suitable of the number of threads achieves the maximum speed-up. To clarify your work, please show your data in some medium such as table, graph, etc. What is your conclusion?
2. **Task 2**: With the most suitable of the number of OpenMP threads for your system in the previous task, you change the data size in some range (let's say from $n = 10$ to $n = 100000$) and sketch the speed-up of the OpenMP execution with respect to the normal execution. If the execution time of the OpenMP execution is longer than of the normal, please explain why?
3. **Task 3**: Modify the application to have a new application which can calculate the product of two matrices with OpenMP.
4. **Task 4 - bonus**: Accelerate the application with whatever techniques such as OpenMP, SSE to acheive a higher speed-up than the current OpenMP implementation.