Midterm Exam

(October 19th @ 7:30 pm)

- Implement SAXPY (Single-Precision A.X Plus Y), also called Scaled Vector Addition with both *pthreads* and TBB.

$$\vec{y} \leftarrow a\vec{x} + \vec{y}$$

  - ✓ SAXPY is a combination of scalar multiplication and vector addition. It takes as input two n-element input vectors $\vec{x}$ and $\vec{y}$ (whose elements are 32-bit floating point numbers), and a scalar value a. A simple C implementation looks like this:

```
void saxpy(int n, float a, float *x, float *y) {
  for (int i = 0; i < n; i++)
      y[i] = a*x[i] + y[i];
}
```

PROBLEM 1 (60 PTS)

- Implement SAXPY using *pthreads* in C (30 pts)
  - ✓ Your code should read the parameter *nthreads* (number of threads) and the length of the vectors (n).
    - Note that $nthreads \in [1, n]$.
  - ✓ Parallelization: each thread i ($i \in [1, n]$) computes a slice of the output vector $\vec{y}$ with the following indices:
    - From $\left\lfloor \frac{i \times n}{nthreads} \right\rfloor$ to $\left\lfloor \frac{(i+1) \times n}{nthreads} \right\rfloor$.
  - ✓ **Input data**: Given the length n, your code should initialize the vectors $\vec{x}$ and $\vec{y}$ as per the following pseudo-code:

```
a = 1.618
for i = 0:n-1
   x[i] = sinh(i*3.416/n);   y[i] = cosh(i*3.416/n);
```

  - ✓ **Verification**: To be fully sure that your results are correct, you need to create a sequential implementation and then compare the results with those of your multi-threaded implementation. This can be achieved by computing the sum of absolute differences (SAD), which should be 0.0:

$$diff = \sum_{i=0}^{n-1} \left| y_p(i) - y_s(i) \right|$$

    where $\vec{y}_p$ and $\vec{y}_s$ are the output vectors of the multi-threaded and sequential implementations respectively.

- Compile the code and execute the application on the DE2i-150 Board. Complete Table I (take an average of ~10 executions in order to get the computation time for each case). (20 pts).
  - ✓ Example: `./mysaxpy 1000 10`
    - It will compute SAXPY on 1000-element vectors $\vec{x}$ and $\vec{y}$ using 10 threads.

TABLE I. COMPUTATION TIME (US) VS. NUMBER OF THREADS AND VECTORS LENGTH

	nthreads									
n	1	2	3	4	5	6	7	8	9	10
1,000										
10,000										
100,000										
1,000,000										
2,000,000										

  - ✓ Comment on your results in Table I. Is there an optimal number of threads? At what point increasing the number of threads causes an increase in processing time?

- Take (and attach) a screenshot of the software running in the terminal for *nthreads=5, n=20*. It should show the computation times (for both the sequential and the *pthreads* implementations), the input vectors $\vec{x}$ and $\vec{y}$, the output vector $\vec{y}$, and the sum of absolute differences (SAD). Fig. 1 shows an execution example. (10 pts)

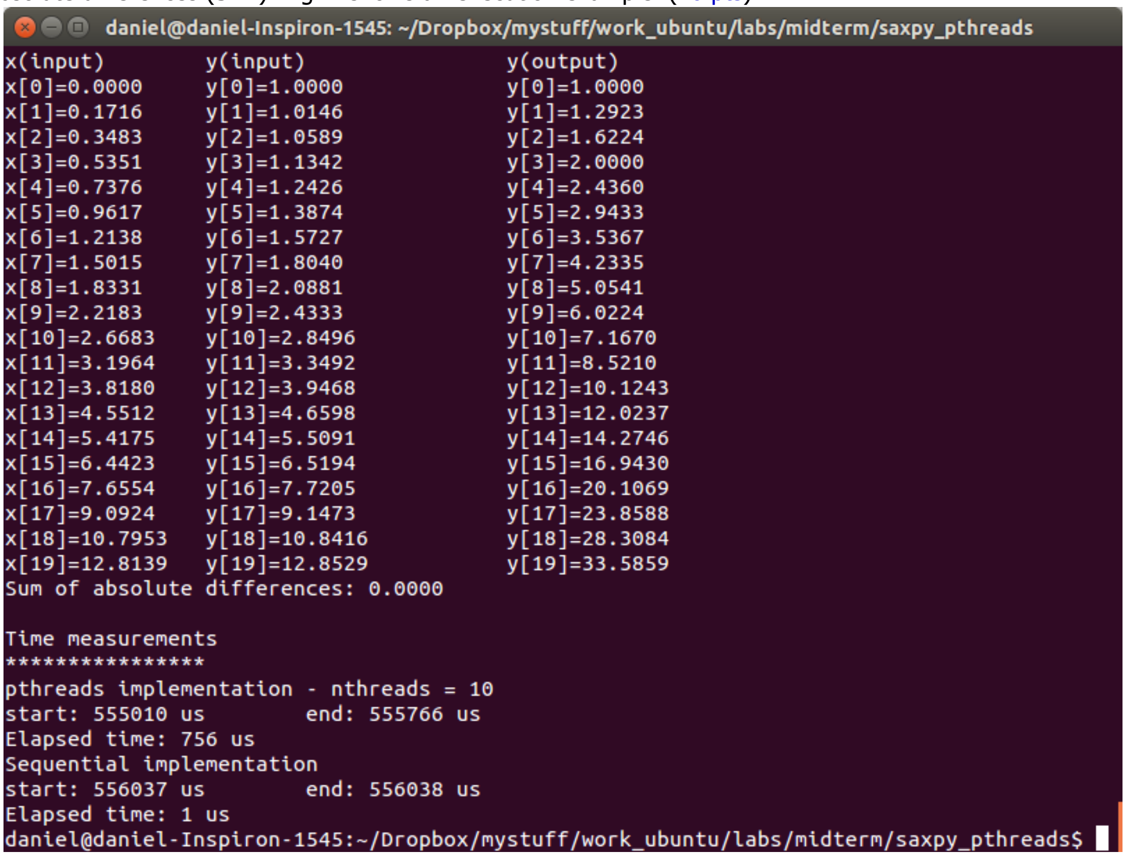

Figure 1. SAXPY execution showing three 20-element sets of values. Computation times obtained from execution on a Dell Inspiron laptop.

PROBLEM 2 (40 PTS)

- Implement SAXPY using TBB *parallel_for* in C++ (15 pts)
  - ✓ Follow the same procedure as in Problem 1, but instead of using *pthreads* to implement slices of the output vector, use *parallel_for* to fully parallelize the sequential SAXPY. Make sure to include a sequential implementation in C++.
  - ✓ Your code should read the parameter input data set size (n).

- Compile the code and execute the application on the DE2i-150 Board. Complete Table II (take an average of ~10 executions for each case). (15 pts)
  - ✓ Example: `./mysaxpy_tbb 1000`
    - ▫ It will compute SAXPY on 1000-element vectors $\vec{x}$ and $\vec{y}$.

TABLE II. COMPUTATION TIME (US) VS. VECTORS LENGTH

	n				
Implementation	10,000	100,000	1,000,000	2,000,000	5,000,000
Sequential					
TBB					

  - ✓ Comment on your Table II results. Is there any point at which the TBB implementation is faster than the sequential one? Yes or No? If No, can you venture a guess as to why this is happening?

- Take (and attach) a screenshot of the software running in the terminal for *n=20*. It should show the computation times (both sequential and the TBB implementations), the input vectors $\vec{x}$ and $\vec{y}$, the output vector $\vec{y}$ and the SAD (as in Fig. 1). (10 pts)

SUBMISSION

- Demonstration: In this Midterm, the requested screenshots of the software routines running in the Terminal suffices.

- Submit to Moodle (an assignment will be created):
  - ✓ Two .zip files (one for Problem 1 and one for Problem 2).
    - Problem 1: The .zip file must contain the source files (`.c, .h, Makefile`).
    - Problem 2: The .zip file must contain the source files (`.cpp, .h, Makefile`).
  - ✓ Your Midterm work (a PDF file): This must include the completed Tables I and II, your comments, as well as the requested screenshots (2).