

This submission did not make use of a machine that is running Linux. Rather, this submission was done on a Windows 11 machine using an MSYS2 MinGW 64-bit Shell in order to simulate multithreading via OpenMP.

Additionally, I used `time`, which is shell-provided, in order to get the machine's execution time of each task.

These commands were used to compile the C files for each task:

Shell

```
gcc -O2 task1.c -o task1
gcc -O2 -mavx2 task2.c -o task2
gcc -O2 task3.c -o task3 -fopenmp
gcc -O2 -mavx2 task4.c -o task4 -fopenmp
```

I made use of the shell-provided `time` instruction in order to get execution time for running a task. I then made a shell script `execution_time.sh` which runs a task 10 times, and then gets the average run time of the runs. For example, for running task 1:

Shell

```
./execution_time.sh ./task1
```

## Execution Times

### *Task 1 - Basic Implementation*

Shell

```
gcc -O2 task1.c -o task1
./execution_time.sh ./task1
```

```
Shiro@LAPTOP-U2KIJPRU MINGW64 /c/Users/Shiro/cs21lab09
$ ./execution_time.sh ./task1
Run 1:
Appr: 3.1415926636
Time: 0.670 seconds

Run 2:
Appr: 3.1415926636
Time: 0.265 seconds

Run 3:
Appr: 3.1415926636
Time: 0.239 seconds

Run 4:
Appr: 3.1415926636
Time: 0.247 seconds

Run 5:
Appr: 3.1415926636
Time: 0.213 seconds

Run 6:
Appr: 3.1415926636
Time: 0.239 seconds

Run 7:
Appr: 3.1415926636
Time: 0.223 seconds

Run 8:
Appr: 3.1415926636
Time: 0.249 seconds

Run 9:
Appr: 3.1415926636
Time: 0.283 seconds

Run 10:
Appr: 3.1415926636
Time: 0.269 seconds

Average runtime over 10 runs: 0.2897 seconds
```

## Task 2- Vectorized Implementation

Shell

```
gcc -O2 -mavx2 task2.c -o task2  
./execution_time.sh ./task2
```

```
Shiro@LAPTOP-U2KIJPRU MINGW64 /c/Users/Shiro/cs21lab09  
$ ./execution_time.sh ./task2  
Run 1:  
Appr: 3.1415926436  
Time: 0.291 seconds  
  
Run 2:  
Appr: 3.1415926436  
Time: 0.148 seconds  
  
Run 3:  
Appr: 3.1415926436  
Time: 0.138 seconds  
  
Run 4:  
Appr: 3.1415926436  
Time: 0.147 seconds  
  
Run 5:  
Appr: 3.1415926436  
Time: 0.165 seconds  
  
Run 6:  
Appr: 3.1415926436  
Time: 0.155 seconds  
  
Run 7:  
Appr: 3.1415926436  
Time: 0.139 seconds  
  
Run 8:  
Appr: 3.1415926436  
Time: 0.154 seconds  
  
Run 9:  
Appr: 3.1415926436  
Time: 0.148 seconds  
  
Run 10:  
Appr: 3.1415926436  
Time: 0.151 seconds  
  
Average runtime over 10 runs: 0.1636 seconds
```

### Task 3 - Multicore Implementation

Shell

```
gcc -O2 task3.c -o task3 -fopenmp  
./execution_time.sh ./task3
```

```
Shiro@LAPTOP-U2K1JPRU MINGW64 /c/Users/Shiro/cs21lab09  
$ ./execution_time.sh ./task3  
Run 1:  
Appr: 3.1415926636  
Time: 0.391 seconds  
  
Run 2:  
Appr: 3.1415926636  
Time: 0.101 seconds  
  
Run 3:  
Appr: 3.1415926636  
Time: 0.091 seconds  
  
Run 4:  
Appr: 3.1415926636  
Time: 0.096 seconds  
  
Run 5:  
Appr: 3.1415926636  
Time: 0.087 seconds  
  
Run 6:  
Appr: 3.1415926636  
Time: 0.106 seconds  
  
Run 7:  
Appr: 3.1415926636  
Time: 0.087 seconds  
  
Run 8:  
Appr: 3.1415926636  
Time: 0.107 seconds  
  
Run 9:  
Appr: 3.1415926636  
Time: 0.091 seconds  
  
Run 10:  
Appr: 3.1415926636  
Time: 0.089 seconds  
  
Average runtime over 10 runs: 0.1246 seconds
```

## Task 4 - Hybrid Implementation

Shell

```
gcc -O2 -mavx2 task4.c -o task4 -fopenmp  
./execution_time.sh ./task4
```

```
Shiro@LAPTOP-U2KI3PRU MINGW64 /c/Users/Shiro/cs21lab09  
$ ./execution_time.sh ./task4  
Run 1:  
Appr: 3.1415926436  
Time: 0.259 seconds  
  
Run 2:  
Appr: 3.1415926436  
Time: 0.093 seconds  
  
Run 3:  
Appr: 3.1415926436  
Time: 0.099 seconds  
  
Run 4:  
Appr: 3.1415926436  
Time: 0.097 seconds  
  
Run 5:  
Appr: 3.1415926436  
Time: 0.098 seconds  
  
Run 6:  
Appr: 3.1415926436  
Time: 0.092 seconds  
  
Run 7:  
Appr: 3.1415926436  
Time: 0.094 seconds  
  
Run 8:  
Appr: 3.1415926436  
Time: 0.089 seconds  
  
Run 9:  
Appr: 3.1415926436  
Time: 0.106 seconds  
  
Run 10:  
Appr: 3.1415926436  
Time: 0.104 seconds  
  
Average runtime over 10 runs: 0.1131 seconds
```

## Speedup Computations

Say you want to compute the speedup from Implementation A to Implementation B, the formula to be used for the Speedup is:

$$Speedup_{A,B} = \frac{AvgRuntime_A}{AvgRuntime_B}$$

$$Speedup_{1,2} = \frac{AvgRuntime_1}{AvgRuntime_2} = \frac{0.2897}{0.1636} = 1.7708 \text{ times faster}$$

$$Speedup_{1,3} = \frac{AvgRuntime_1}{AvgRuntime_3} = \frac{0.2897}{0.1246} = 2.3250 \text{ times faster}$$

$$Speedup_{1,4} = \frac{AvgRuntime_1}{AvgRuntime_4} = \frac{0.2897}{0.1131} = 2.5615 \text{ times faster}$$

$$Speedup_{2,4} = \frac{AvgRuntime_2}{AvgRuntime_4} = \frac{0.1636}{0.1131} = 1.4465 \text{ times faster}$$

$$Speedup_{3,4} = \frac{AvgRuntime_3}{AvgRuntime_4} = \frac{0.1246}{0.1131} = 1.1017 \text{ times faster}$$