

**PRN:** 2020BTECS00112

**Class:** Final Year (Computer Science and Engineering)

**Year:** 2023-24

**Semester:** 1

**Course:** High Performance Computing Lab

## Practical No. 2

### Title of practical: Study and implementation of basic OpenMP clauses

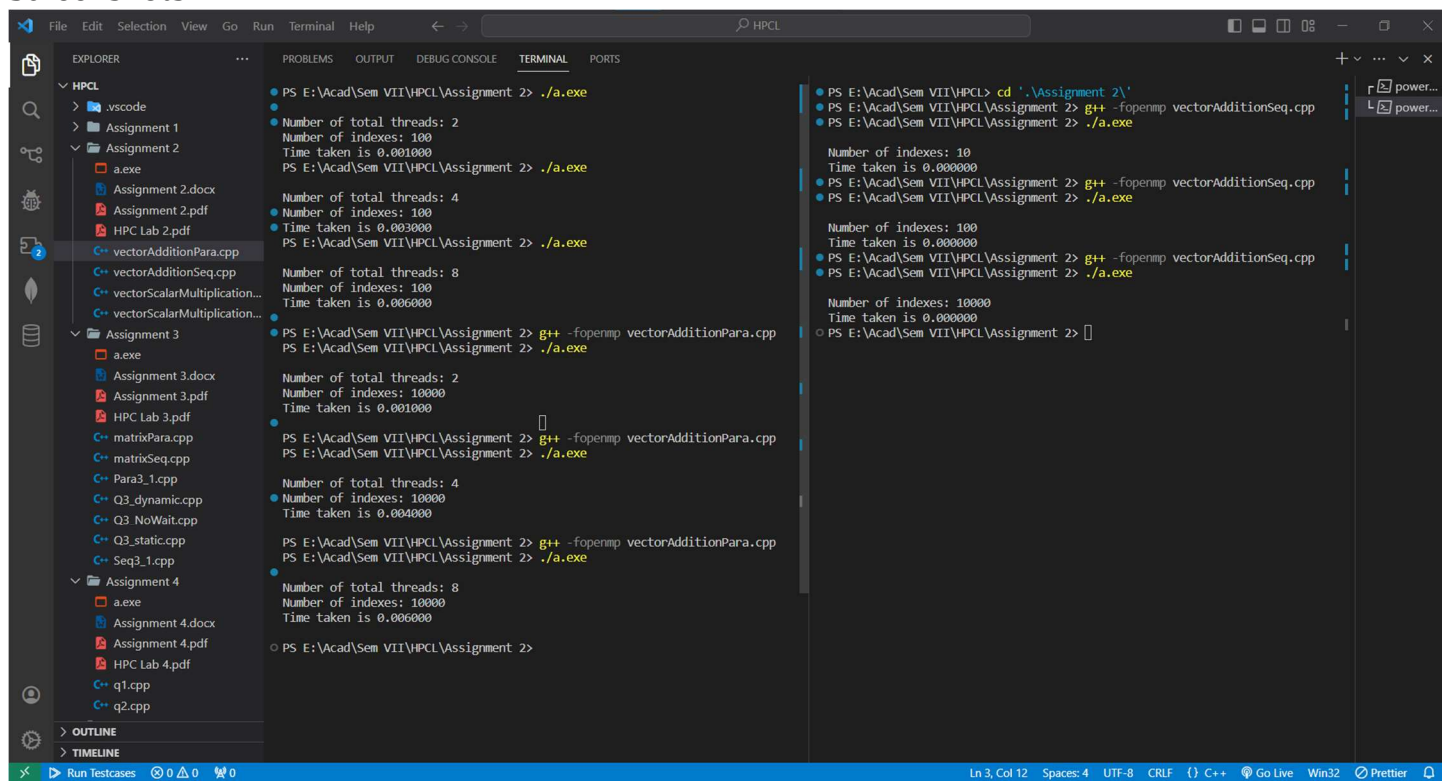
Implement following Programs using OpenMP with C:

1. Vector Scalar Addition
2. Calculation of value of Pi

Analyse the performance of your programs for different number of threads and Data size.

### Problem Statement 1: Vector Scalar Addition

#### Screenshots:

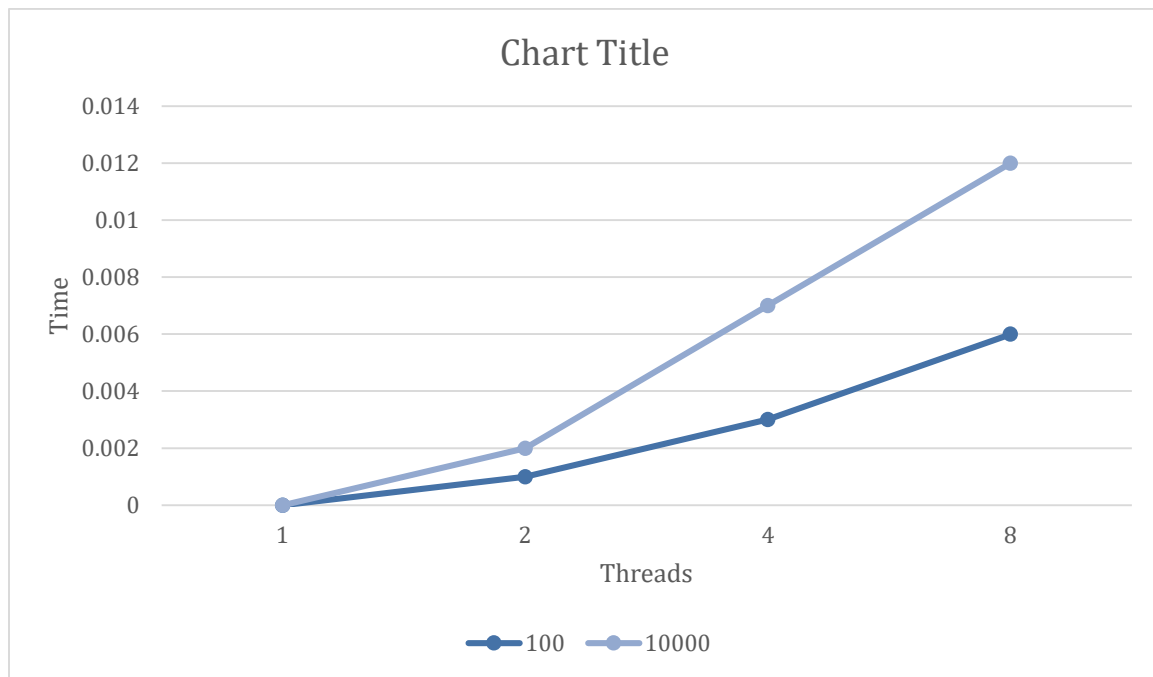


```
PS E:\Acad\Sem VII\HPCL\Assignment 2> .\a.exe
Number of total threads: 2
Number of indexes: 100
Time taken is 0.001000
PS E:\Acad\Sem VII\HPCL\Assignment 2> .\a.exe
Number of total threads: 4
Number of indexes: 100
Time taken is 0.003000
PS E:\Acad\Sem VII\HPCL\Assignment 2> .\a.exe
Number of total threads: 8
Number of indexes: 100
Time taken is 0.006000
PS E:\Acad\Sem VII\HPCL\Assignment 2> g++ -fopenmp vectorAdditionPara.cpp
PS E:\Acad\Sem VII\HPCL\Assignment 2> .\a.exe
Number of total threads: 2
Number of indexes: 10000
Time taken is 0.001000
PS E:\Acad\Sem VII\HPCL\Assignment 2> g++ -fopenmp vectorAdditionPara.cpp
PS E:\Acad\Sem VII\HPCL\Assignment 2> .\a.exe
Number of total threads: 4
Number of indexes: 10000
Time taken is 0.004000
PS E:\Acad\Sem VII\HPCL\Assignment 2> g++ -fopenmp vectorAdditionPara.cpp
PS E:\Acad\Sem VII\HPCL\Assignment 2> .\a.exe
Number of total threads: 8
Number of indexes: 10000
Time taken is 0.006000
PS E:\Acad\Sem VII\HPCL\Assignment 2>
```

#### Information:

The reduction clause in OpenMP is used to perform a reduction operation on one or more variables across multiple threads. It allows you to automatically compute the final result of a variable after a parallel region. It avoids false sharing.

### Analysis:



### Problem Statement 2: Calculation of value of Pi

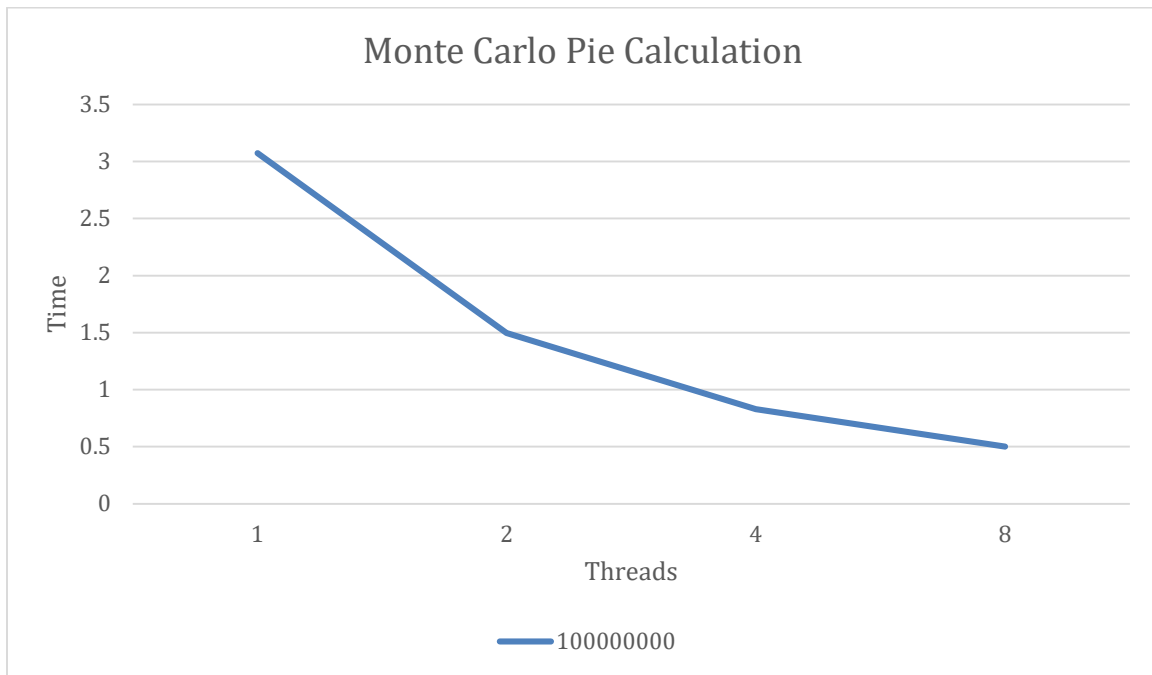
#### Screenshots:

```
PS E:\Acad\Sem VII\HPCL\Assignment 2> g++ piePara.cpp -fopenmp
PS E:\Acad\Sem VII\HPCL\Assignment 2> ./a.exe
• Pi Approximation: 3.141320
• Total points: 100000000
• Number of threads: 1
• Total time: 3.074000
PS E:\Acad\Sem VII\HPCL\Assignment 2> g++ piePara.cpp -fopenmp
PS E:\Acad\Sem VII\HPCL\Assignment 2> ./a.exe
• Pi Approximation: 3.141528
• Total points: 100000000
• Number of threads: 2
• Total time: 1.496000
PS E:\Acad\Sem VII\HPCL\Assignment 2> g++ piePara.cpp -fopenmp
PS E:\Acad\Sem VII\HPCL\Assignment 2> ./a.exe
• Pi Approximation: 3.141288
• Total points: 100000000
• Number of threads: 4
• Total time: 0.829000
PS E:\Acad\Sem VII\HPCL\Assignment 2> g++ piePara.cpp -fopenmp
PS E:\Acad\Sem VII\HPCL\Assignment 2> ./a.exe
• Pi Approximation: 3.141095
• Total points: 100000000
• Number of threads: 8
• Total time: 0.501000
PS E:\Acad\Sem VII\HPCL\Assignment 2> |
```

#### Information:

Monte Carlo methods are a way of estimating numerical results through random sampling. The Monte Carlo method for approximating  $\pi$  involves randomly generating points within a square and determining how many fall within a quarter circle inscribed within that square. The ratio of points inside the quarter circle to the total points generated is an approximation of  $\pi/4$ . The final approximation of  $\pi$  is calculated by multiplying the ratio of points inside the circle to the total points by 4, as we are using only one quarter of the unit circle.

**Analysis:**



**Github Link:**