Walchand College of Engineering, Sangli
Department of Computer Science and Engineering

High Performance Computing Lab

Class: Final Year (Computer Science and Engineering) **Year:** 2022-23

PRN: 2019BTECS00089 – Piyush Pramod Mhaske **Batch**: B3

Practical No. 1

https://github.com/sakshid14/2019BTECS00021_HPC_LAB/tree/main/Practical1

GitHub Link:

Once the compiler encounters the parallel regions code, threads will get created and it will calculate square of all first 100 numbers with addition of all squares of numbers stored in variable sum. Threads works in parallel and do task given to it.

To implement a parallel code to print squares of first 100 numbers followed by addition of all these squares.

```
#include<omp.h>
#include<stdio.h>
#include<stdlib.h>
#include <time.h>

int main(int argc, char* argv)
{
  double start,end;
  start=omp_get_wtime();


      #pragma omp parallel
  {
    printf("Thread: %d\nHello World\n",omp_get_thread_num());
  }

    end=omp_get_wtime();

    printf("\nTime For Parallel Execution= %f",(end-start));


}
```

Output:

```
Hello World
Thread: 1
Hello World
Thread: 2
Hello World
Thread: 0
Hello World
Thread: 5
Hello World
Thread: 4
Hello World
Thread: 3
Hello World
Thread: 7
Hello World

Time For Parallel Execution= 0.007000
PS D:\Academics\Fourth Year\HPC Lab\Assignments>
```

```
PS D:\Academics\Fourth Year\HPC Lab\Assignments> gcc -o SerialHello -fopenmp SerialHello.c
PS D:\Academics\Fourth Year\HPC Lab\Assignments> .\SerialHello.exe
Thread: 0
Hello World
Runtime is 0.000000 seconds
PS D:\Academics\Fourth Year\HPC Lab\Assignments>
```

Problem Statement 2:

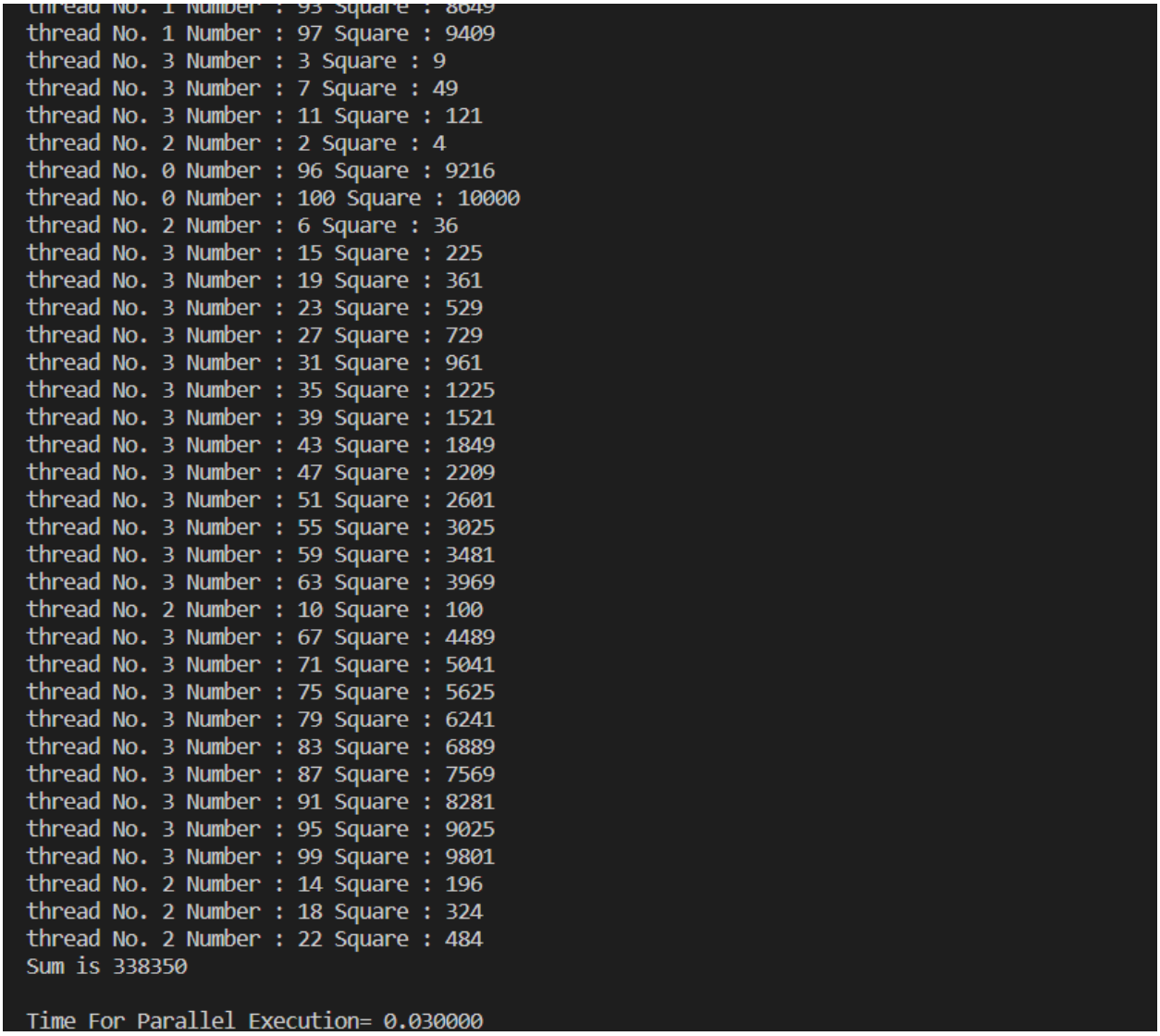
When the master thread reaches this line, it forks additional threads to carry out the work enclosed in the block following the #pragma construct. The block is executed by all threads in parallel. The original thread will be denoted as master thread with thread-id 0.Once the compiler encounters the parallel regions code, the master thread (thread which has thread id 0) will fork into the specified number of threads. Entire code within the parallel region will be executed by all threads concurrently. Once the parallel region ended, all threads will get merged into the master thread.

Information 1:

```
#include<omp.h>
#include<stdio.h>
#include <time.h>

static int sum =0;
int main(int argc, char* argv)
{
  double start,end;
  start=omp_get_wtime();

    #pragma omp parallel
    {
        for(int i=1; i<=100;i++)
        {
            if(i%4==omp_get_thread_num())
            {
                printf("thread No. %d Number : %d Square : %d\n", omp_get_thread_num(), i, i * i);
                sum+=i*i;
            }
        }

    }
    printf("Sum is %d \n",sum);
    end=omp_get_wtime();

    printf("\nTime For Parallel Execution= %f",(end-start));
}
```

output:

```
PS D:\Academics\Fourth Year\HPC Lab\Assignments> gcc -o squaresof100 -fopenmp squaresof100.c
PS D:\Academics\Fourth Year\HPC Lab\Assignments> .\squaresof100
thread No. 0 Number : 4 Square : 16
thread No. 0 Number : 8 Square : 64
thread No. 0 Number : 12 Square : 144
thread No. 0 Number : 16 Square : 256
thread No. 0 Number : 20 Square : 400
thread No. 0 Number : 24 Square : 576
thread No. 0 Number : 28 Square : 784
thread No. 0 Number : 32 Square : 1024
thread No. 0 Number : 36 Square : 1296
thread No. 0 Number : 40 Square : 1600
thread No. 0 Number : 44 Square : 1936
thread No. 0 Number : 48 Square : 2304
thread No. 0 Number : 52 Square : 2704
thread No. 0 Number : 56 Square : 3136
thread No. 0 Number : 60 Square : 3600
thread No. 0 Number : 64 Square : 4096
thread No. 0 Number : 68 Square : 4624
thread No. 0 Number : 72 Square : 5184
thread No. 0 Number : 76 Square : 5776
thread No. 0 Number : 80 Square : 6400
thread No. 0 Number : 84 Square : 7056
thread No. 0 Number : 88 Square : 7744
thread No. 0 Number : 92 Square : 8464
thread No. 1 Number : 1 Square : 1
thread No. 1 Number : 5 Square : 25
thread No. 1 Number : 9 Square : 81
thread No. 1 Number : 13 Square : 169
thread No. 1 Number : 17 Square : 289
thread No. 1 Number : 21 Square : 441
thread No. 1 Number : 25 Square : 625
thread No. 1 Number : 29 Square : 841
thread No. 1 Number : 33 Square : 1089
thread No. 1 Number : 37 Square : 1369
thread No. 1 Number : 41 Square : 1681
thread No. 1 Number : 45 Square : 2025
thread No. 1 Number : 49 Square : 2401
thread No. 1 Number : 53 Square : 2809
thread No. 1 Number : 57 Square : 3249
thread No. 1 Number : 61 Square : 3721
thread No. 1 Number : 65 Square : 4225
thread No. 1 Number : 69 Square : 4761
thread No. 1 Number : 73 Square : 5329
```



Information 2:

Once the compiler encounters the parallel regions code, threads will get created and it will calculate square of all first 100 numbers with addition of all squares of numbers stored in variable sum. Threads works in parallel and do task given to it.