SEM - VII - 2022-23
High-Performance Computing Lab
Assignment 2
Name: Borse Vishal Subhash
PRN: 2019BTECS00066

SPMD: Single Program Multiple Data
It is a parallel programming style. In SPMD style tasks are split up and run simultaneously on multiple processors with different data. This is done to achieve results faster.

Worksharing
Threads are assigned an independent subset of the total workload For example, different chunks of an iteration are distributed among the threads. OpenMP's loop worksharing construct splits loop iterations among all active threads
#pragma omp for

Types of variables 1.
Shared Variables :
There exist one instance of this variable which is shared among all threads
2. Private Variables :
Each thread in a team of threads has its own local copy of the private variable

Implicit and Explicit
Implicit : All the variables declared outside of the pragma are by default shared and all the variables declared inside pragma are private Explicit:
Shared Clause eg. #pragma omp parallel for shared(n, a) => n and a are declared as shared vavriables Private Clause eg. #pragma omp parallel for shared(n, a) private(c) => here c is private variable Default Clause eg. #pragma omp parallel for default(shared) => now all variables are shared
#pragma omp parallel for default(private) => now all variables are private

Schedule
a specification of how iterations of associated loops are divided into contiguous non-empty subsets.
syntax: #pragma omp parallel for schedule([modifier [modifier]:]kind[,chunk_size])

Q) Program for Vector to Vector Addition:

Parallel Algorithm:

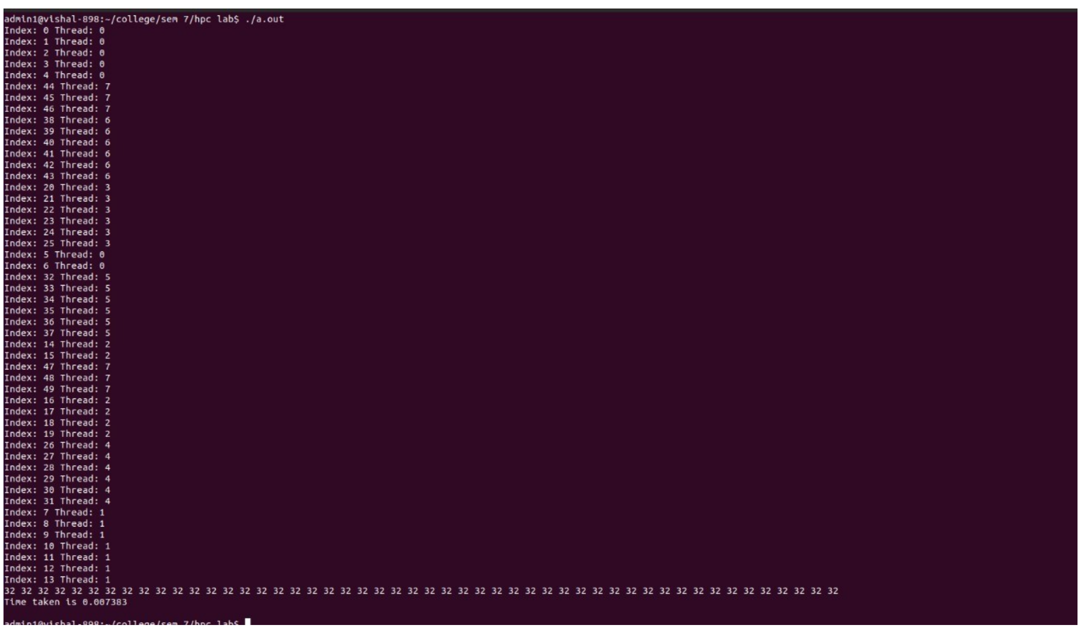
```
#include <omp.h>
#include <stdio.h>

void main()
{
    printf("Adding Two Arrays\
    n"); int a1[] = {1, 2, 3, 4, 5};
    int a2[] = {11, 12, 13, 14, 15};
    int a3[5] = {0};

    #pragma omp parallel for
     for (int i = 0; i < 5; i++)
    { a3[i] = a1[i] + a2[i];
    }


    for (int i = 0; i < 5; i++)
    { printf("%d ", a3[i]);
    }
}
```

```
admin1@vishal-898:~/college/sem 7/hpc lab$ ./a.out
Adding Two Arrays
12 14 16 18 20 admin1@vishal-898:~/college/sem 7/hpc lab$
```

Q) Parallel program for scaler sum of vector

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

int main() {
    int N = 50;
    int A[100];
    for (int i = 0; i < N; i++)
```

```
        A[i] = 30;
    int S = 2;

    double itime;
    itime = omp_get_wtime();
    #pragma omp parallel for
    for (int i = 0; i < N; i++) {
        A[i] += S;
        printf("Index: %d Thread: %d\n", i, omp_get_thread_num());
    }

    for (int i = 0; i < N; i++) {
        printf("%d ", A[i]);
    }

    double ftime = omp_get_wtime();
    double exec_time = ftime - itime;
    printf("\nTime taken is %f\n", exec_time);
    printf("\n");
}
```



```
admin1@vishal-898:~/college/sem 7/hpc lab$ ./a.out
Index: 0 Thread: 0
Index: 1 Thread: 0
Index: 2 Thread: 0
Index: 3 Thread: 0
Index: 4 Thread: 0
Index: 44 Thread: 7
Index: 45 Thread: 7
Index: 46 Thread: 7
Index: 38 Thread: 6
Index: 39 Thread: 6
Index: 40 Thread: 6
Index: 41 Thread: 6
Index: 42 Thread: 6
Index: 43 Thread: 6
Index: 20 Thread: 3
Index: 21 Thread: 3
Index: 22 Thread: 3
Index: 23 Thread: 3
Index: 24 Thread: 3
Index: 25 Thread: 3
Index: 5 Thread: 0
Index: 6 Thread: 0
Index: 32 Thread: 5
Index: 33 Thread: 5
Index: 34 Thread: 5
Index: 35 Thread: 5
Index: 36 Thread: 5
Index: 37 Thread: 5
Index: 14 Thread: 2
Index: 15 Thread: 2
Index: 47 Thread: 7
Index: 48 Thread: 7
Index: 49 Thread: 7
Index: 16 Thread: 2
Index: 17 Thread: 2
Index: 18 Thread: 2
Index: 19 Thread: 2
Index: 26 Thread: 4
Index: 27 Thread: 4
Index: 28 Thread: 4
Index: 29 Thread: 4
Index: 30 Thread: 4
Index: 31 Thread: 4
Index: 7 Thread: 1
Index: 8 Thread: 1
Index: 9 Thread: 1
Index: 10 Thread: 1
Index: 11 Thread: 1
Index: 12 Thread: 1
Index: 13 Thread: 1
32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32
Time taken is 0.007383

admin1@vishal-898:~/college/sem 7/hpc lab$
```