Final Year B. Tech, Sem VII 2022-23
PRN – 2020BTECS00211
Name – Aashita Narendra Gupta
High Performance Computing Lab
Batch: B4

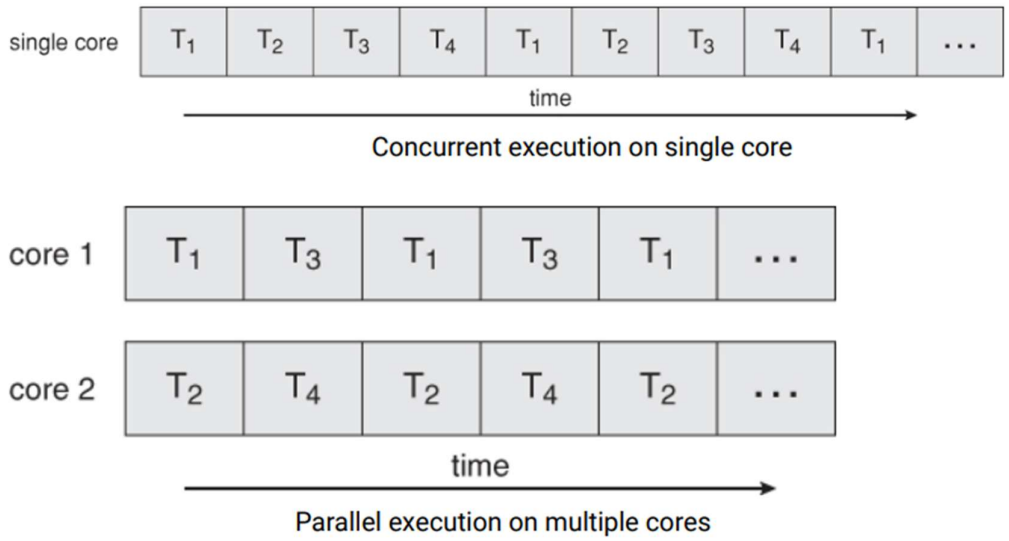
Practical No – 1

Github Link for Code - https://github.com/Aashita06/HPC_Practicals

Q.1) Difference between Hardware Threads and Software Threads:
→
Hardware Thread:
A "hardware thread" is a physical CPU or core. So, a 4 core CPU can genuinely support 4 hardware threads at once - the CPU really is doing 4 things at the same time. One hardware thread can run many software threads. In modern operating systems, this is often done by time-slicing - each thread gets a few milliseconds to execute before the OS schedules another thread to run on that CPU. Since the OS switches back and forth between the threads quickly, it appears as if one CPU is doing more than one thing at once, but in reality, a core is still running only one hardware thread, which switches between many software threads.

Software Thread:
Software threads are threads of execution managed by the operating system. Software threads are abstractions to the hardware to make multi-processing possible. If you have multiple software threads but there are not multiple resources then these software threads are a way to run all tasks in parallel by allocating resources for limited time(or using some other strategy) so that it appears that all threads are running in parallel. These are managed by the operating system.

Q.2) Which type of threads are supported by the processor?
→
Generally, the Hardware Threads are supported by the processor.
The hardware threads are mostly based on the muti-core architecture which is latest architecture to achieve high performance. A multi-threaded application running on a traditional single-core chip wouldhave to interleave the threads, as shown in Figure 4.3. On a multi-core chip, however, the threads could be spread across the available cores, allowing true parallel processing.



Q.3) Simple Hello World with eight threads:

→

Code:

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

int main(int argc, char* argv[]){

   omp_set_num_threads(8);
   #pragma omp parallel //generating threads
   {
      int id = omp_get_thread_num();
      printf("thread No. %d  Hello World\n", omp_get_thread_num());
      printf("ID = %d\n",id);
   }
   return 0;
}
```

- #pragma is used to generate threads.
- omp_set_num_thread() is used to number of threads manually.
- omp_get_thread_num() is used to get the id of current thread.
- The "#pragma omp for" advises the compiler to distribute the work load of the following loop within the team of threads which you have to create first.

Output:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

PS C:\Users\Ashitra\OneDrive\Desktop\7th sem\Practicals\HPC\Programs> ./a.exe
thread No. 2  Hello World
ID = 2
thread No. 3  Hello World
ID = 3
thread No. 4  Hello World
ID = 4
thread No. 6  Hello World
ID = 6
thread No. 1  Hello World
ID = 1
thread No. 5  Hello World
ID = 5
thread No. 0  Hello World
thread No. 7  Hello World
ID = 0
ID = 7
PS C:\Users\Ashitra\OneDrive\Desktop\7th sem\Practicals\HPC\Programs>
```

Q.3) Calculate sum of the squares using serial and parallel processing respectively and compare the time taken by both processing and calculate Speedup:

→

Serial Processing:

Code:

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

static long long sum =0;
int main()
{
    double itime, ftime, exec_time;
    itime = omp_get_wtime();

        for(long long i=1; i<=1000;i++)
        {
            sum += i * i;
        }

        printf("Sum is %d  ", sum);

        ftime = omp_get_wtime();
        exec_time = (ftime - itime);
        printf("\nTime taken is %f\n", exec_time);
        return 0;
}
```

Output:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

PS C:\Users\Ashitra\OneDrive\Desktop\7th sem\Practicals\HPC\Programs> gcc -fopenmp SquareSumS.cpp
PS C:\Users\Ashitra\OneDrive\Desktop\7th sem\Practicals\HPC\Programs> ./a.exe

Sum is 333833500
Time taken is 0.002000
PS C:\Users\Ashitra\OneDrive\Desktop\7th sem\Practicals\HPC\Programs>
```

Parallel Processing:

Code:

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

static long long sum =0;
int main()
{
    double itime, ftime, exec_time;
    itime = omp_get_wtime();

    #pragma omp parallel
     omp_set_num_threads(4);

        for(long long i=1; i<=1000;i++)
        {
            sum += i * i;
        }

        printf("\nSum is %d  ", sum);

        ftime = omp_get_wtime();
        exec_time = (ftime - itime);
        printf("\nTime taken is %f\n", exec_time);
        return 0;
}
```

Output:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

PS C:\Users\Ashitra\OneDrive\Desktop\7th sem\Practicals\HPC\Programs> gcc -fopenmp SquareSumP.cpp
PS C:\Users\Ashitra\OneDrive\Desktop\7th sem\Practicals\HPC\Programs> ./a.exe

Sum is 333833500
Time taken is 0.001000
PS C:\Users\Ashitra\OneDrive\Desktop\7th sem\Practicals\HPC\Programs>
```

Serial processing will take time – 0.002000
Parallel processing will take time – 0.001000

Parallel programming will take less time for execution in comparison with serial programming.

The speedup will be:
Speedup = Ts/Tp = 0.002000/0.001000 = 2