**Final Year B. Tech., Sem VII 2022-23**

**High Performance Computing Lab**

**PRN/ Roll No: 2020BTECS00206**

**Full Name: SAYALI YOGESH DESAI**

**Batch: B4**

**Assignment No. 1**

**Que 1. Difference between Software and Hardware threads**

**Hardware Thread:**

1. 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.
2. One hardware thread can run many software threads.
3. 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.
4. 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:**

1. Software threads are threads of execution managed by the operating system.
2. Software threads are abstractions to the hardware to make multi-processing possible.
3. 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.

**Que 2. Which type of threads are supported by the processor?**

1. Threads are the virtual components or codes, which divides the physical core of a CPU into virtual multiple cores. A single CPU core can have up-to 2 threads per core.
2. For example, if a CPU is dual core (i.e., 2 cores) it will have 4 threads. And if a CPU is Octal core (i.e., 8 core) it will have 16 threads and vice-versa.
3. There is single thread on the core which when gets the information from the user, creates another thread and allocates the task to it. Similarly, if it gets another instruction, it forms second thread and allocates the task to it. Making a total of two threads.
4. 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.

**Que 3. Simple Hello World Program with 4 threads**

* **Code**

#include<omp.h>

#include<stdio.h>

#include<stdlib.h>

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

#pragma omp parallel

{

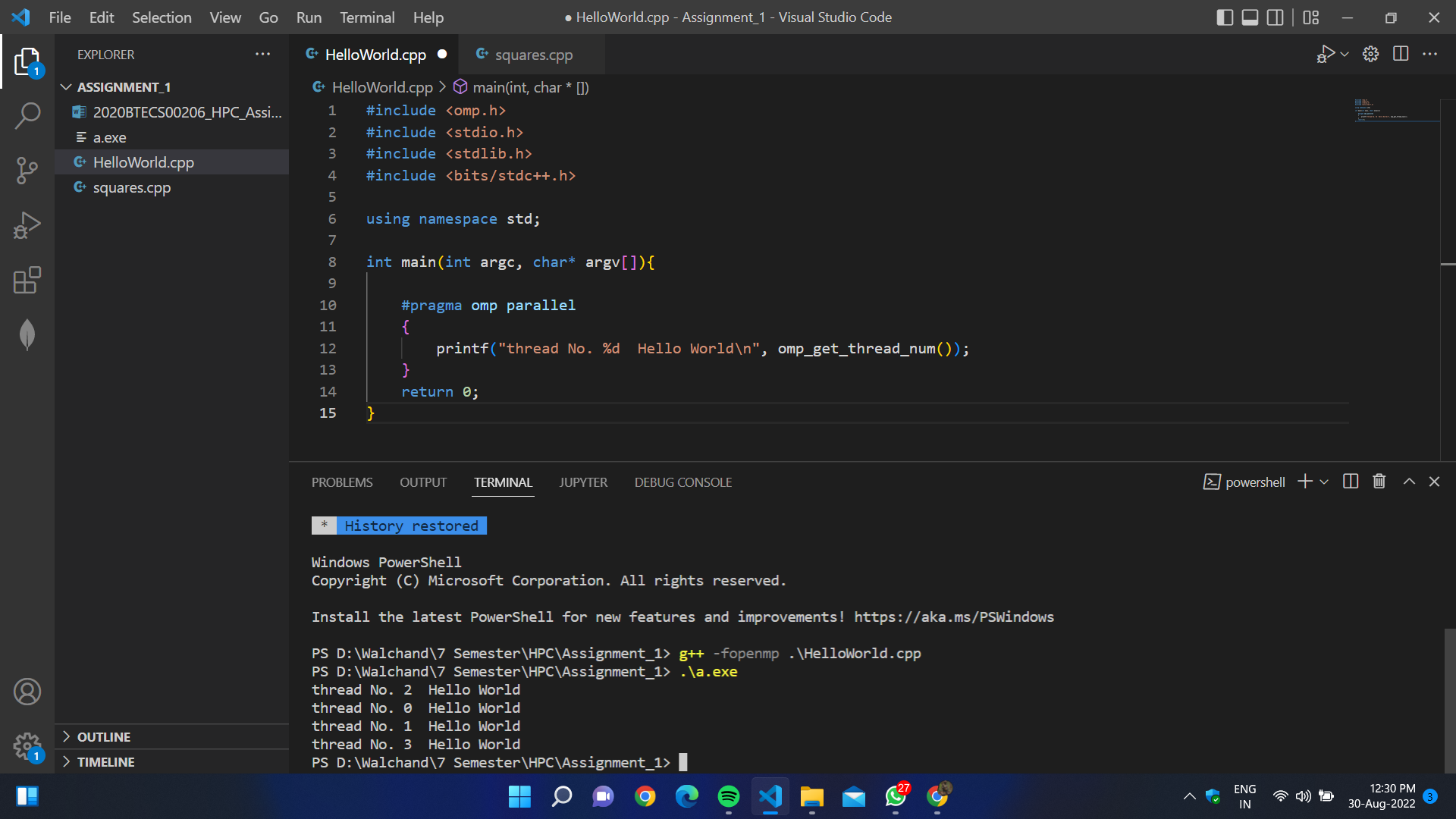
printf("thread No. %d Hello World\n", omp\_get\_thread\_num());

}

return 0;

}

* **Output**



- fopenmp is a flag which is required to execute openmp parallel programs on GCC compiler

- omp.h - header file in C which have all openmp functions defined

- omp\_get\_thread\_num() - function to get thread ID/ number

**Que 4. Calculate squares and sum of first 100 numbers using serial processing. Compare speed in parallel and serial processing.**

* **Code:**

#include<omp.h>

#include<stdio.h>

#include<stdlib.h>

#include<bits/stdc++.h>

using namespace std;

static int sum =0;

int main()

{

double itime, ftime, exec\_time;

itime = omp\_get\_wtime();

for(int i=1; i<=100;i++)

{

printf("Thread No. %d Number : %d Square : %d \n", omp\_get\_thread\_num(), i, i \* i);

sum += i \* i;

}

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

cout << endl;

ftime = omp\_get\_wtime();

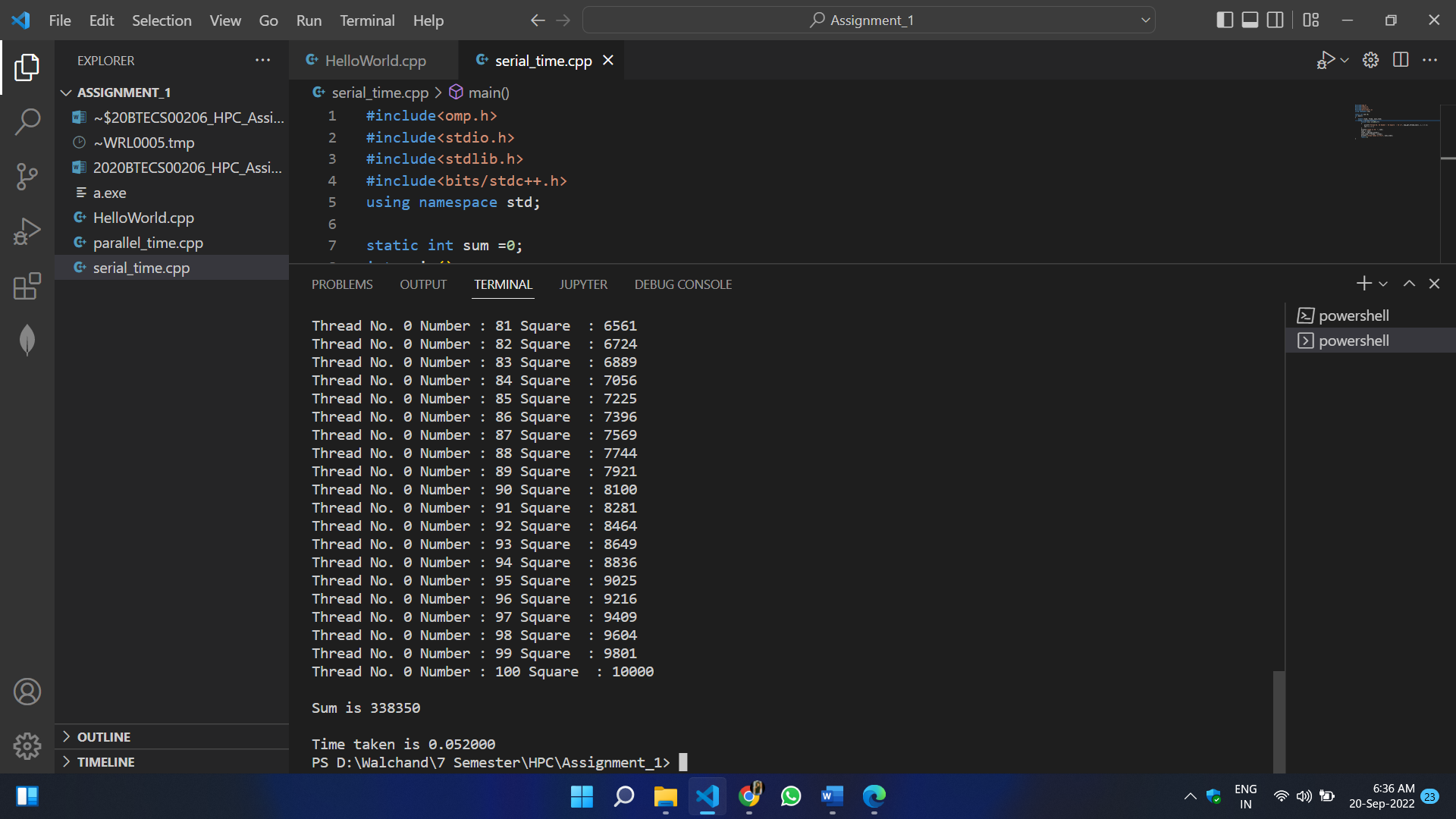
exec\_time = (ftime - itime);

printf("\nTime taken is %f\n", exec\_time);

return 0;

}

* **Output:**



**Que 5. Calculate squares and sum of first 100 numbers using parallel processing. Compare speed in parallel and serial processing.**

* **Code:**

#include<omp.h>

#include<stdio.h>

#include<stdlib.h>

#include<bits/stdc++.h>

using namespace std;

static int sum =0;

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

{

double itime, ftime, exec\_time;

itime = omp\_get\_wtime();

#pragma omp parallel

for(int i=1; i<=100;i++)

{

if(i%4==omp\_get\_thread\_num())

{

printf("\nThread No. %d Number : %d Square : %d", omp\_get\_thread\_num(), i, i \* i);

sum+=i\*i;

}

}

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

cout << endl;

ftime = omp\_get\_wtime();

exec\_time = (ftime - itime);

printf("\nTime taken is %f\n", exec\_time);

return 0;

}

* **Output:**

