Date: 05/09/2022

High Performance Computing Lab

Assignment - 1

PRN: 2019BTECS00034
Name: Rutikesh Sawant

Q1. Differentiate between Software and Hardware Threads

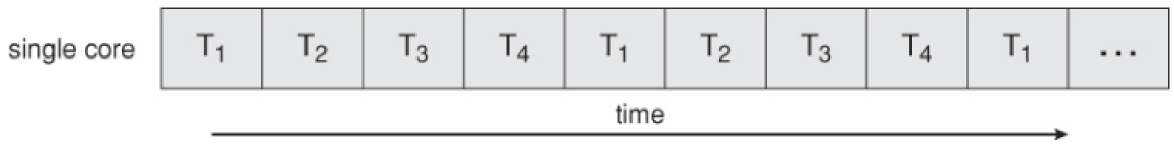
Hardware Thread: A "hardware thread" is a physical CPU or core. So, a 4 core CPU can genuinely support 4 hardware threads at once - the CPU really is doing 4 things at the same time. One hardware thread can run many software threads. In modern operating systems, this is often done by time-slicing - each 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.

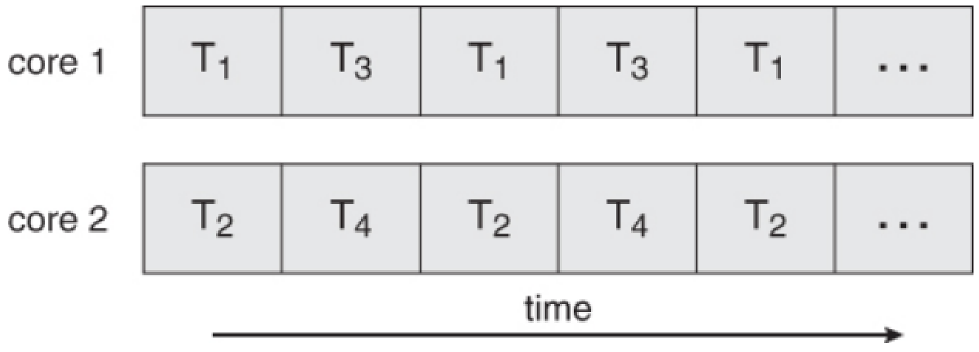
Q2. Which type of threads are supported by the processor?

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 would have 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.



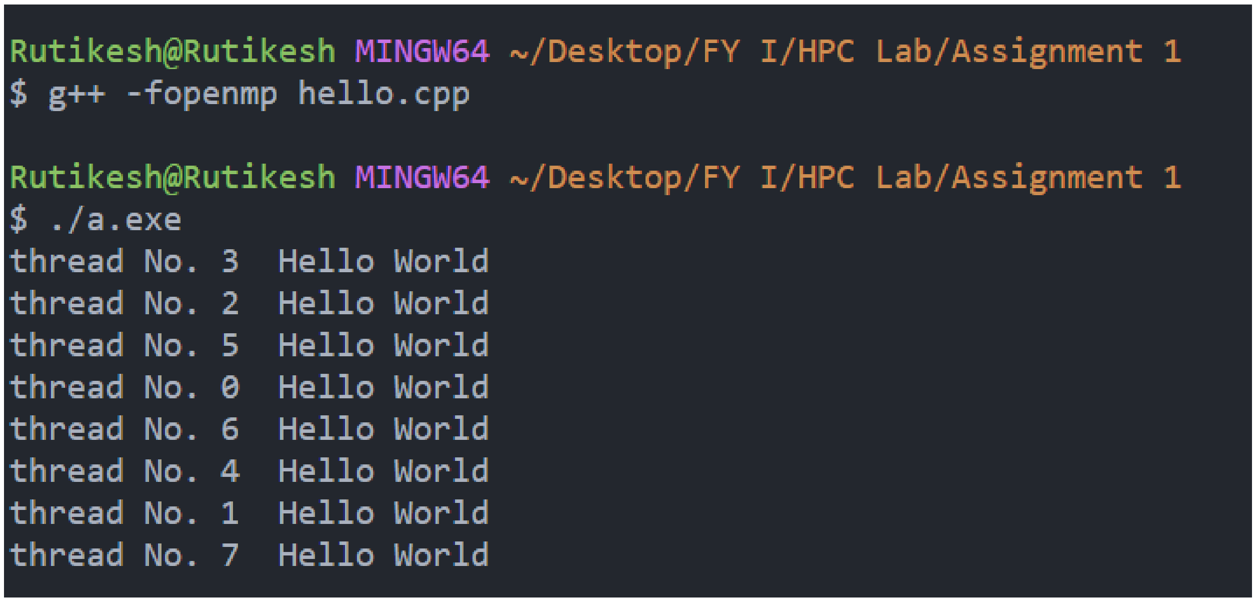
Concurrent execution on single core


core 1
T_1 T_3 T_1 T_3 T_1 ...
core 2
T_2 T_4 T_2 T_4 T_2 ...
time

Parallel execution on multiple cores

Q3 program to print Hello World!

```c
#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;
}
```



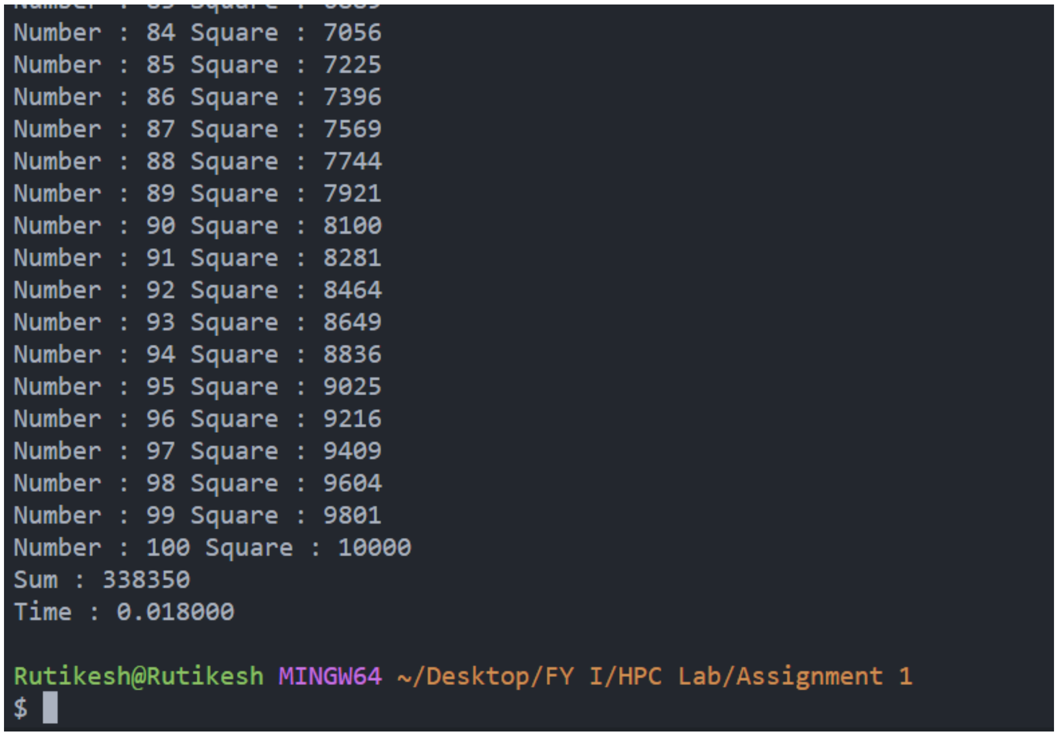
```
Rutikesh@Rutikesh MINGW64 ~/Desktop/FY I/HPC Lab/Assignment 1
$ g++ -fopenmp hello.cpp

Rutikesh@Rutikesh MINGW64 ~/Desktop/FY I/HPC Lab/Assignment 1
$ ./a.exe
thread No. 3  Hello World
thread No. 2  Hello World
thread No. 5  Hello World
thread No. 0  Hello World
thread No. 6  Hello World
thread No. 4  Hello World
thread No. 1  Hello World
thread No. 7  Hello World
```

Q4. Find the squares of first 100 numbers followed by their sum. Compare the speed in sequential and parallel algorithm.

Sequential Approach :

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

int main() {

    double time;
    clock_t begin = clock();

    int sum = 0;
#pragma omp parallel for
    // {
    for (int i = 1; i <= 100; i++) {
        printf("thread No. %d  Number : %d Square :
%d\n", omp_get_thread_num(), i, i * i);
```

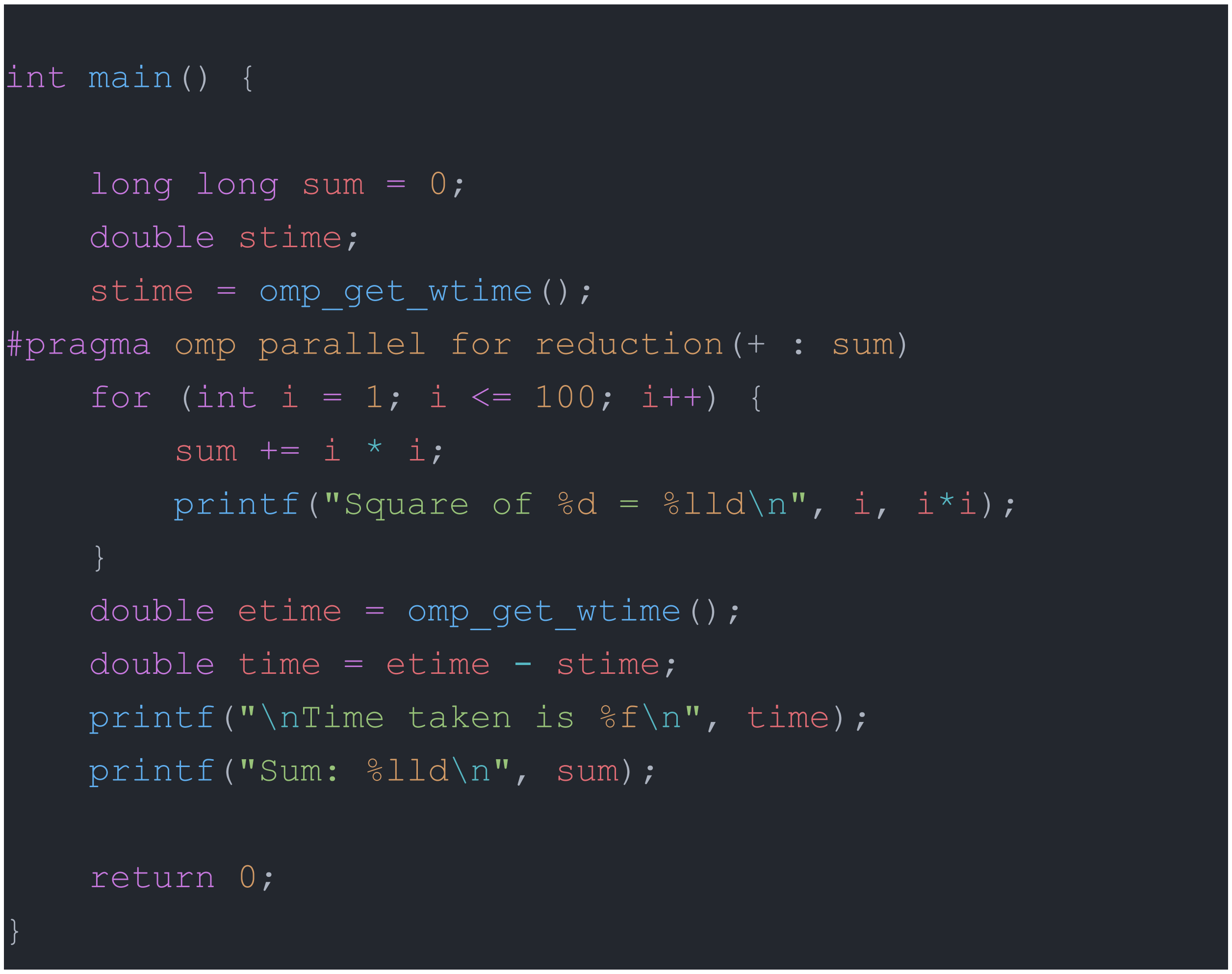
```
        sum += i * i;
    }
    printf("Sum : %d\n", sum);

    clock_t end = clock();
    time = (double)(end - begin) / CLOCKS_PER_SEC;
    printf("Time : %lf\n", time);
    return 0;
}
```



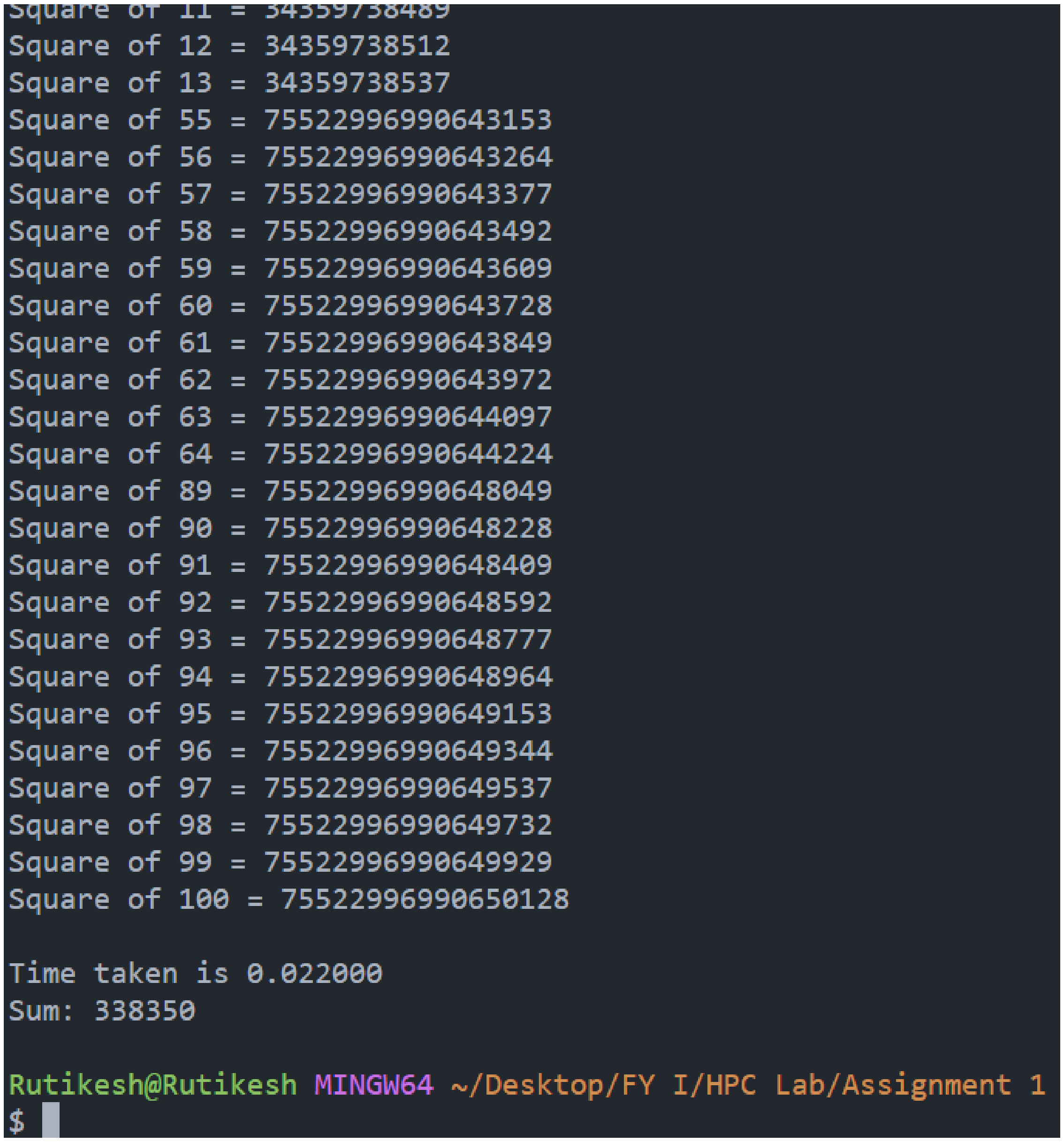
```
Number : 84 Square : 7056
Number : 85 Square : 7225
Number : 86 Square : 7396
Number : 87 Square : 7569
Number : 88 Square : 7744
Number : 89 Square : 7921
Number : 90 Square : 8100
Number : 91 Square : 8281
Number : 92 Square : 8464
Number : 93 Square : 8649
Number : 94 Square : 8836
Number : 95 Square : 9025
Number : 96 Square : 9216
Number : 97 Square : 9409
Number : 98 Square : 9604
Number : 99 Square : 9801
Number : 100 Square : 10000
Sum : 338350
Time : 0.018000

Rutikesh@Rutikesh MINGW64 ~/Desktop/FY I/HPC Lab/Assignment 1
$
```

Parallel approach using openMP:

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



```c
int main() {

    long long sum = 0;
    double stime;
    stime = omp_get_wtime();
#pragma omp parallel for reduction(+ : sum)
    for (int i = 1; i <= 100; i++) {
        sum += i * i;
        printf("Square of %d = %lld\n", i, i*i);
    }
    double etime = omp_get_wtime();
    double time = etime - stime;
    printf("\nTime taken is %f\n", time);
    printf("Sum: %lld\n", sum);

    return 0;
}
```



```
Square of 11 = 34359738489
Square of 12 = 34359738512
Square of 13 = 34359738537
Square of 55 = 7552296990643153
Square of 56 = 7552296990643264
Square of 57 = 7552296990643377
Square of 58 = 7552296990643492
Square of 59 = 7552296990643609
Square of 60 = 7552296990643728
Square of 61 = 7552296990643849
Square of 62 = 7552296990643972
Square of 63 = 7552296990644097
Square of 64 = 7552296990644224
Square of 89 = 7552296990648049
Square of 90 = 7552296990648228
Square of 91 = 7552296990648409
Square of 92 = 7552296990648592
Square of 93 = 7552296990648777
Square of 94 = 7552296990648964
Square of 95 = 7552296990649153
Square of 96 = 7552296990649344
Square of 97 = 7552296990649537
Square of 98 = 7552296990649732
Square of 99 = 7552296990649929
Square of 100 = 7552296990650128

Time taken is 0.022000
Sum: 338350

Rutikesh@Rutikesh MINGW64 ~/Desktop/FY I/HPC Lab/Assignment 1
$
```