Date: 05/09/2022

High Performance Computing Lab

Assignment - 1

PRN: 2019BTECS00029
Name: Harshal Kodgire

Q1. Differentiate between Software and Hardware Threads

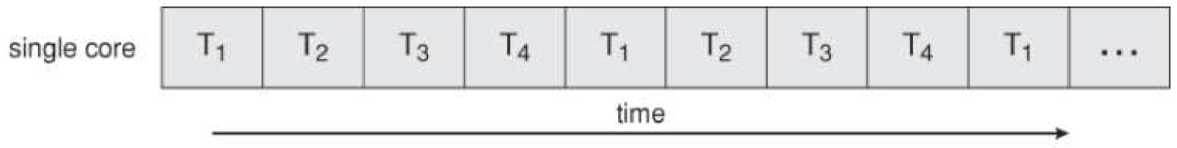
Hardware Thread: A "hardware thread" is a physical CPU or core. So, a 4 core CPU can genuinely support 4 hardware threads at once - the CPU really is doing 4 things at the same time. One hardware thread can run many software threads. In modern operating systems, this is often done by time-slicing - each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

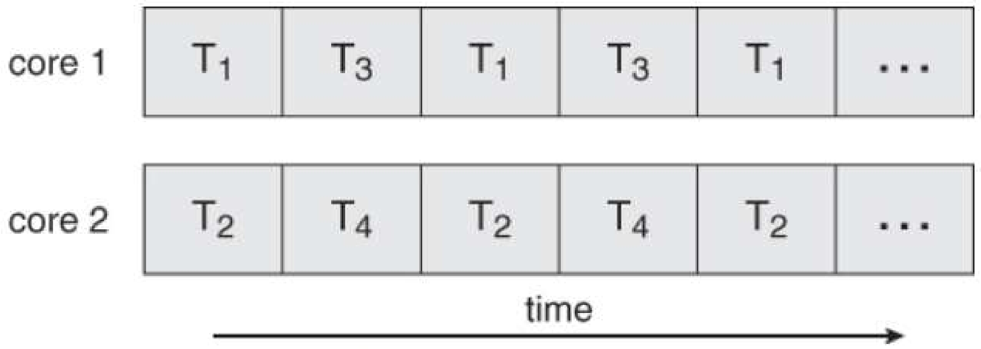
Q2. Which type of threads are supported by the processor?

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 multicore chip, however, the threads could be spread across the available cores, allowing true parallel processing.

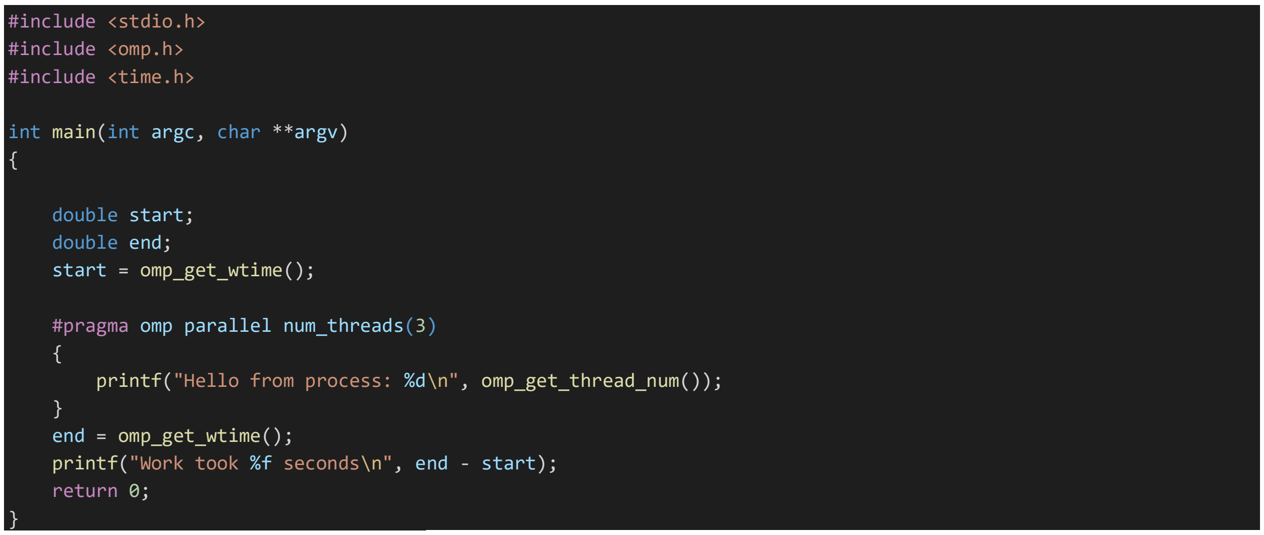


Concurrent execution on single core

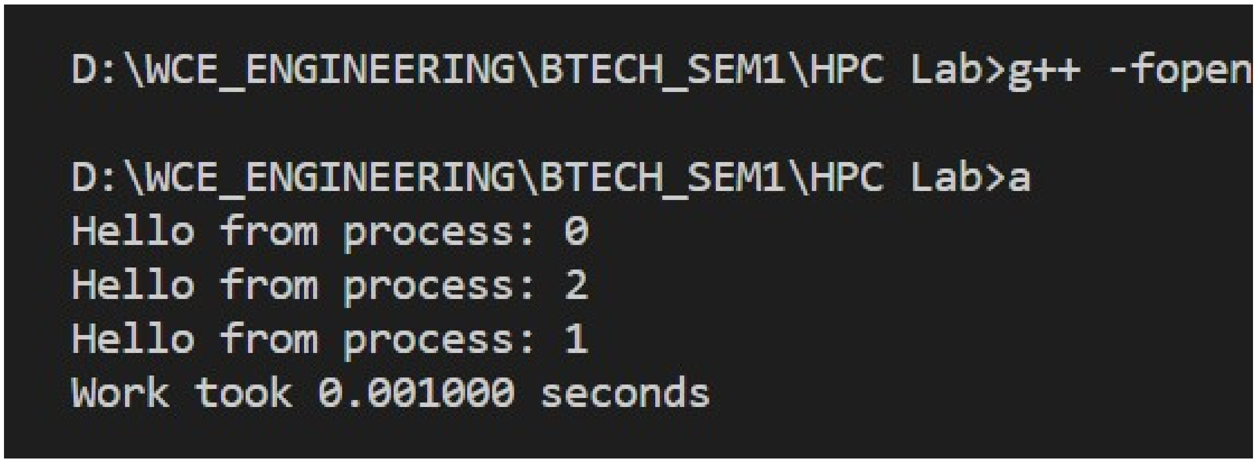

core 1
T_1 T_3 T_1 T_3 T_1 ...
core 2
T_2 T_4 T_2 T_4 T_2 ...
time

Parallel execution on multiple cores

Q3 program to print Hello World!



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

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

    double start;
    double end;
    start = omp_get_wtime();

    #pragma omp parallel num_threads(3)
    {
        printf("Hello from process: %d\n", omp_get_thread_num());
    }
    end = omp_get_wtime();
    printf("Work took %f seconds\n", end - start);
    return 0;
}
```



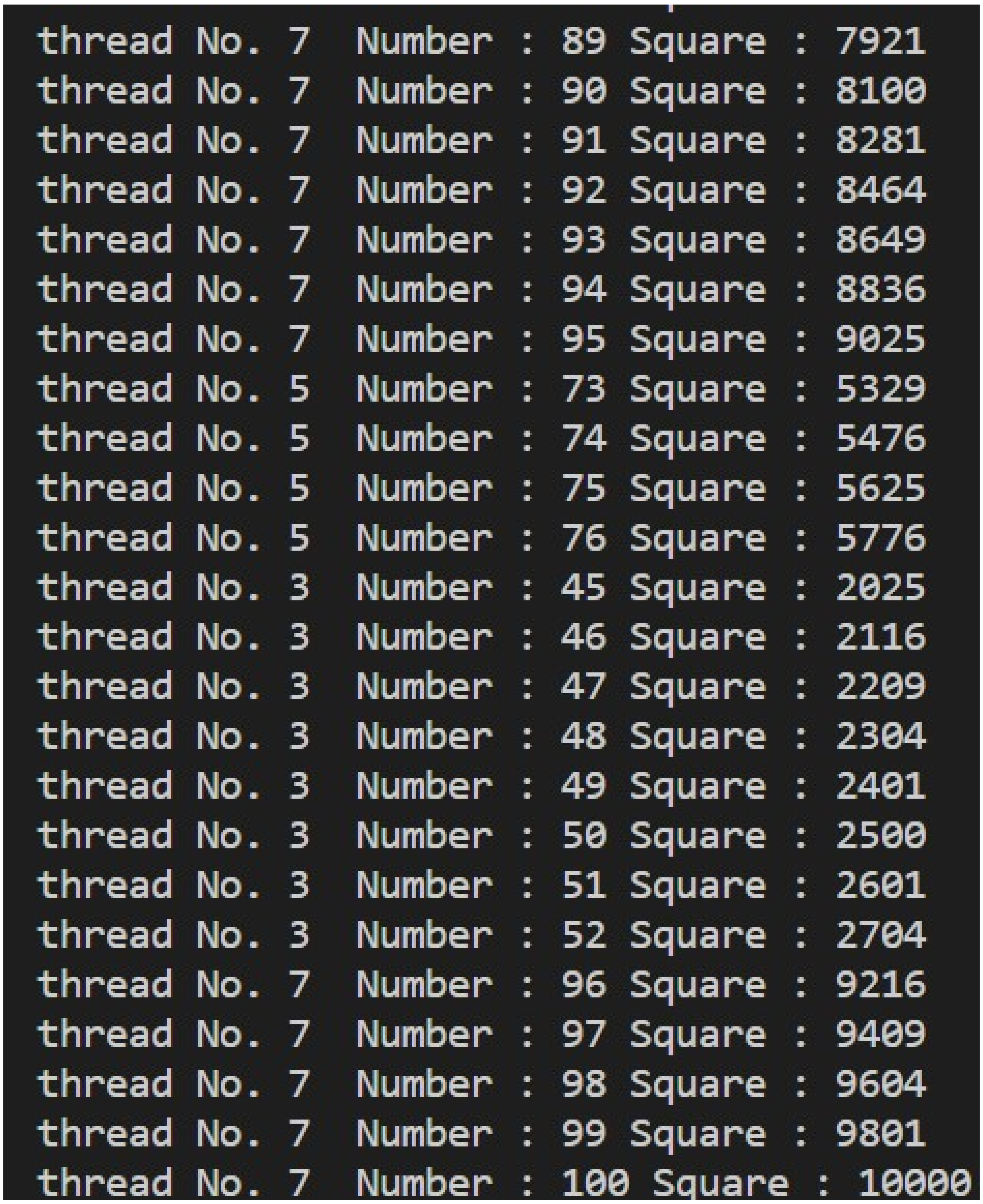
```
D:\WCE_ENGINEERING\BTECH_SEM1\HPC Lab>g++ -fopen

D:\WCE_ENGINEERING\BTECH_SEM1\HPC Lab>a
Hello from process: 0
Hello from process: 2
Hello from process: 1
Work took 0.001000 seconds
```

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);

```

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

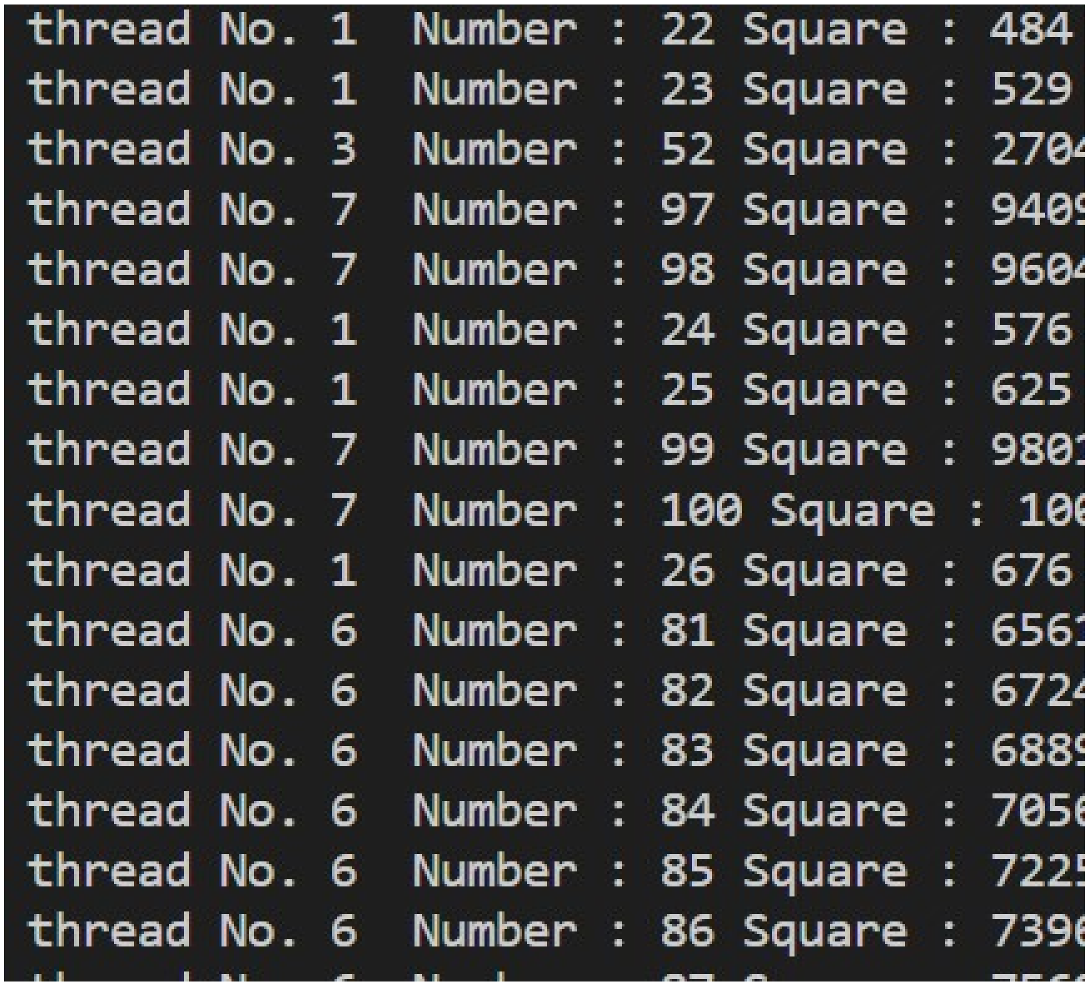


```
thread No. 7  Number : 89 Square : 7921
thread No. 7  Number : 90 Square : 8100
thread No. 7  Number : 91 Square : 8281
thread No. 7  Number : 92 Square : 8464
thread No. 7  Number : 93 Square : 8649
thread No. 7  Number : 94 Square : 8836
thread No. 7  Number : 95 Square : 9025
thread No. 5  Number : 73 Square : 5329
thread No. 5  Number : 74 Square : 5476
thread No. 5  Number : 75 Square : 5625
thread No. 5  Number : 76 Square : 5776
thread No. 3  Number : 45 Square : 2025
thread No. 3  Number : 46 Square : 2116
thread No. 3  Number : 47 Square : 2209
thread No. 3  Number : 48 Square : 2304
thread No. 3  Number : 49 Square : 2401
thread No. 3  Number : 50 Square : 2500
thread No. 3  Number : 51 Square : 2601
thread No. 3  Number : 52 Square : 2704
thread No. 7  Number : 96 Square : 9216
thread No. 7  Number : 97 Square : 9409
thread No. 7  Number : 98 Square : 9604
thread No. 7  Number : 99 Square : 9801
thread No. 7  Number : 100 Square : 10000
```

Parallel approach using openMP:

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

```c
#include <stdlib.h>
#include <time.h>

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;
}
```



```
thread No. 1  Number : 22 Square : 484
thread No. 1  Number : 23 Square : 529
thread No. 3  Number : 52 Square : 270
thread No. 7  Number : 97 Square : 940
thread No. 7  Number : 98 Square : 960
thread No. 1  Number : 24 Square : 576
thread No. 1  Number : 25 Square : 625
thread No. 7  Number : 99 Square : 980
thread No. 7  Number : 100 Square : 10
thread No. 1  Number : 26 Square : 676
thread No. 6  Number : 81 Square : 656
thread No. 6  Number : 82 Square : 672
thread No. 6  Number : 83 Square : 688
thread No. 6  Number : 84 Square : 705
thread No. 6  Number : 85 Square : 722
thread No. 6  Number : 86 Square : 739
```