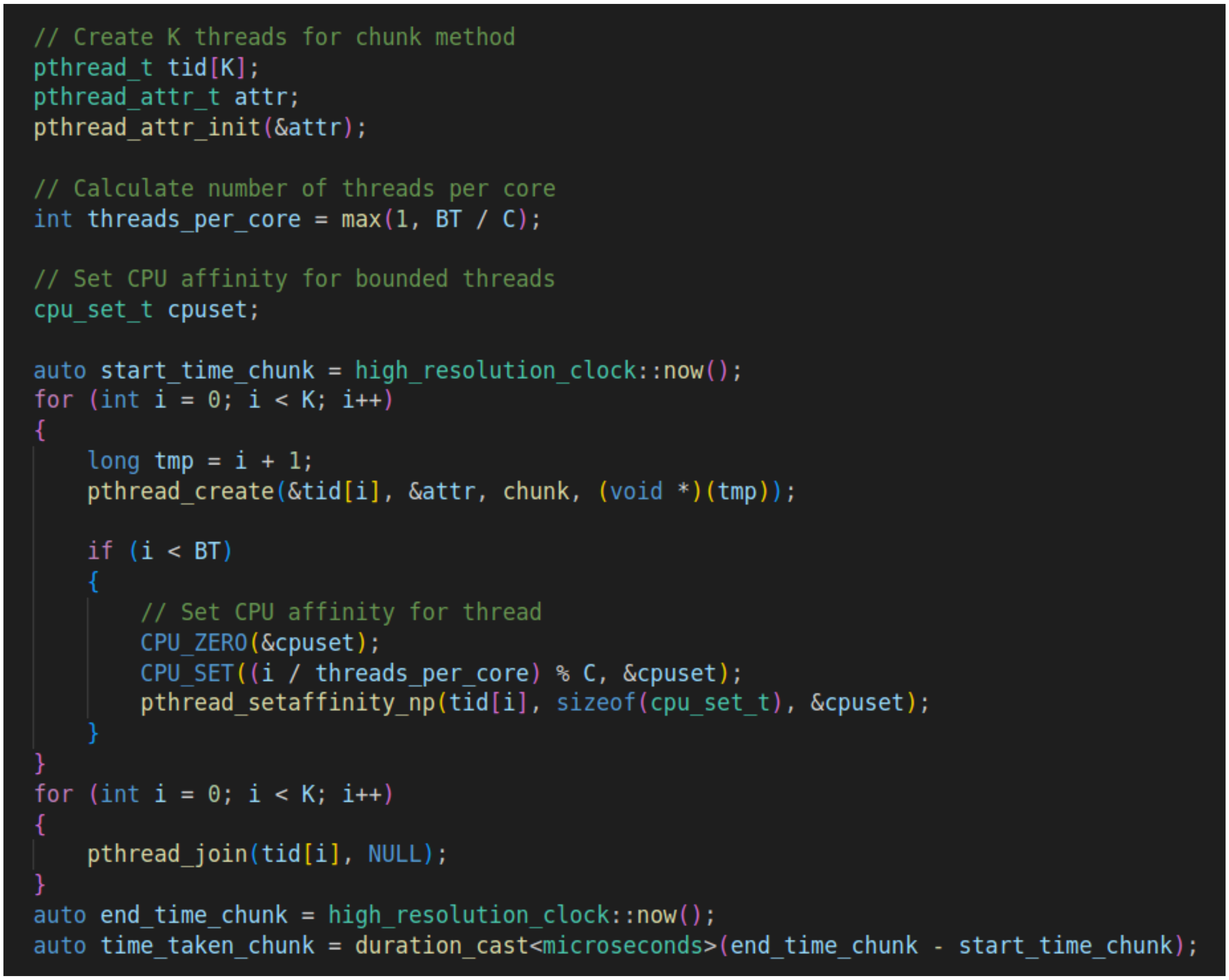
CS3523 - OS2 - Programming Assignment 2
CS23MTECH11026 - Report

System Specification:
OS: Windows 11
Installed Memory (RAM): 8GB
Chipset: 11th Gen Intel Core i5-1135G7 @ 2.40GHz × 8
Core: 4
Logical processor: 8

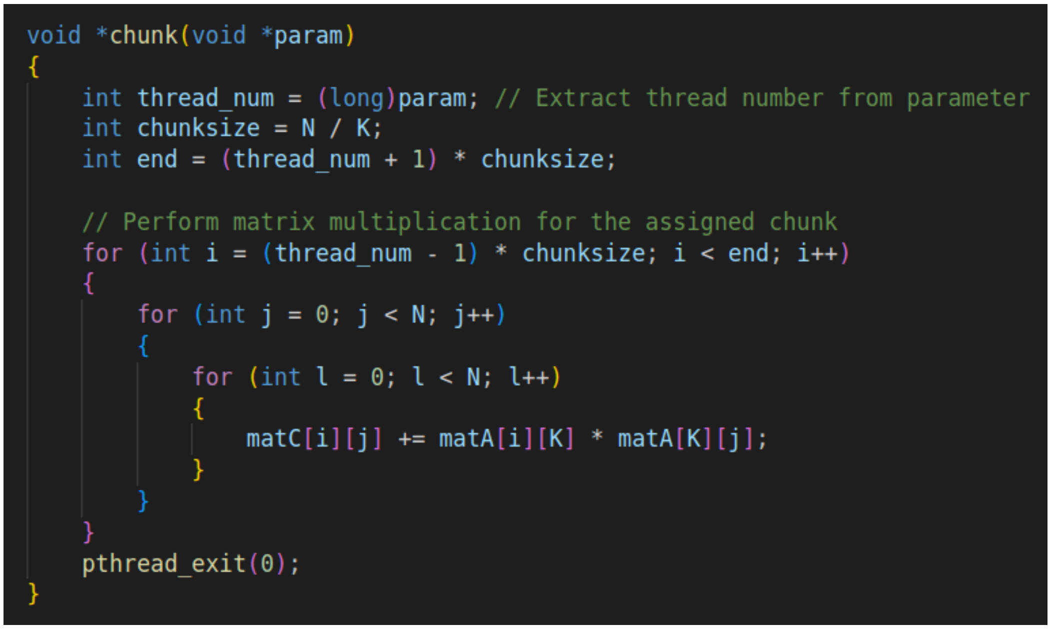
VMware: Ubuntu 22.04.3 LTS
Allocated Memory: 4GB

Low Level Design of Program (Calls chunk and mixed function)



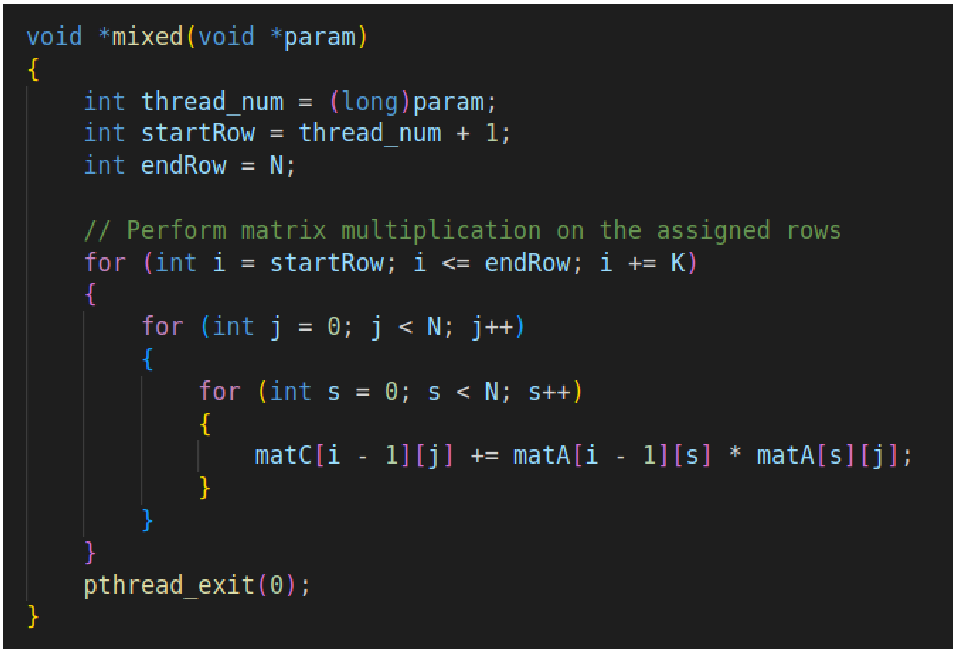
```
// Create K threads for chunk method
pthread_t tid[K];
pthread_attr_t attr;
pthread_attr_init(&attr);

// Calculate number of threads per core
int threads_per_core = max(1, BT / C);

// Set CPU affinity for bounded threads
cpu_set_t cpuset;

auto start_time_chunk = high_resolution_clock::now();
for (int i = 0; i < K; i++)
{
    long tmp = i + 1;
    pthread_create(&tid[i], &attr, chunk, (void *)(tmp));

    if (i < BT)
    {
        // Set CPU affinity for thread
        CPU_ZERO(&cpuset);
        CPU_SET((i / threads_per_core) % C, &cpuset);
        pthread_setaffinity_np(tid[i], sizeof(cpu_set_t), &cpuset);
    }
}
for (int i = 0; i < K; i++)
{
    pthread_join(tid[i], NULL);
}
auto end_time_chunk = high_resolution_clock::now();
auto time_taken_chunk = duration_cast<microseconds>(end_time_chunk - start_time_chunk);
```

Chunk Function:



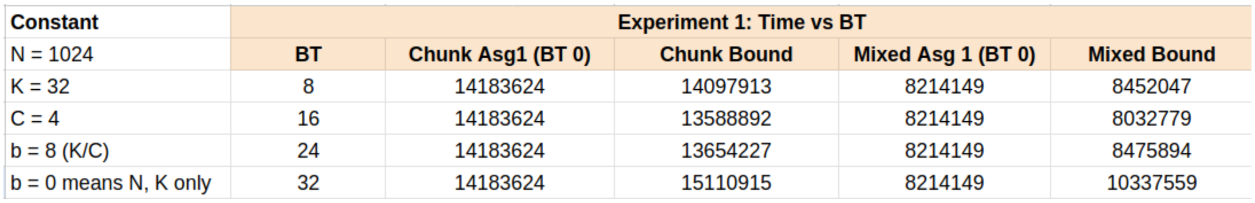
```
void *chunk(void *param)
{
    int thread_num = (long)param; // Extract thread number from parameter
    int chunksize = N / K;
    int end = (thread_num + 1) * chunksize;

    // Perform matrix multiplication for the assigned chunk
    for (int i = (thread_num - 1) * chunksize; i < end; i++)
    {
        for (int j = 0; j < N; j++)
        {
            for (int l = 0; l < N; l++)
            {
                matC[i][j] += matA[i][K] * matA[K][j];
            }
        }
    }
    pthread_exit(0);
}
```

Mixed Method:



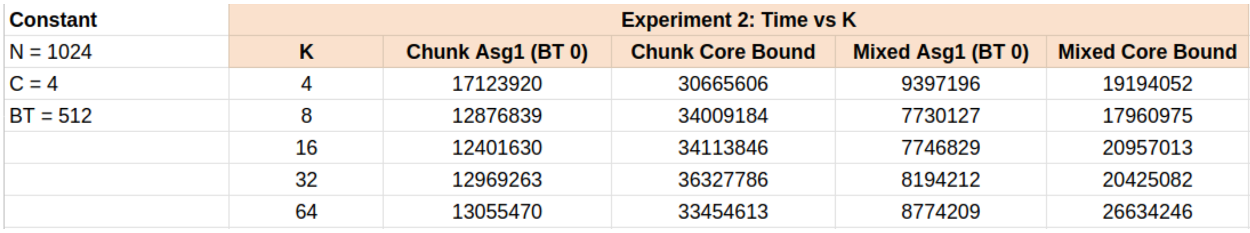
```c
void *mixed(void *param)
{
    int thread_num = (long)param;
    int startRow = thread_num + 1;
    int endRow = N;

    // Perform matrix multiplication on the assigned rows
    for (int i = startRow; i <= endRow; i += K)
    {
        for (int j = 0; j < N; j++)
        {
            for (int s = 0; s < N; s++)
            {
                matC[i - 1][j] += matA[i - 1][s] * matA[s][j];
            }
        }
    }
    pthread_exit(0);
}
```

Experiment 1: Total Time vs Number of Bounded Threads, BT



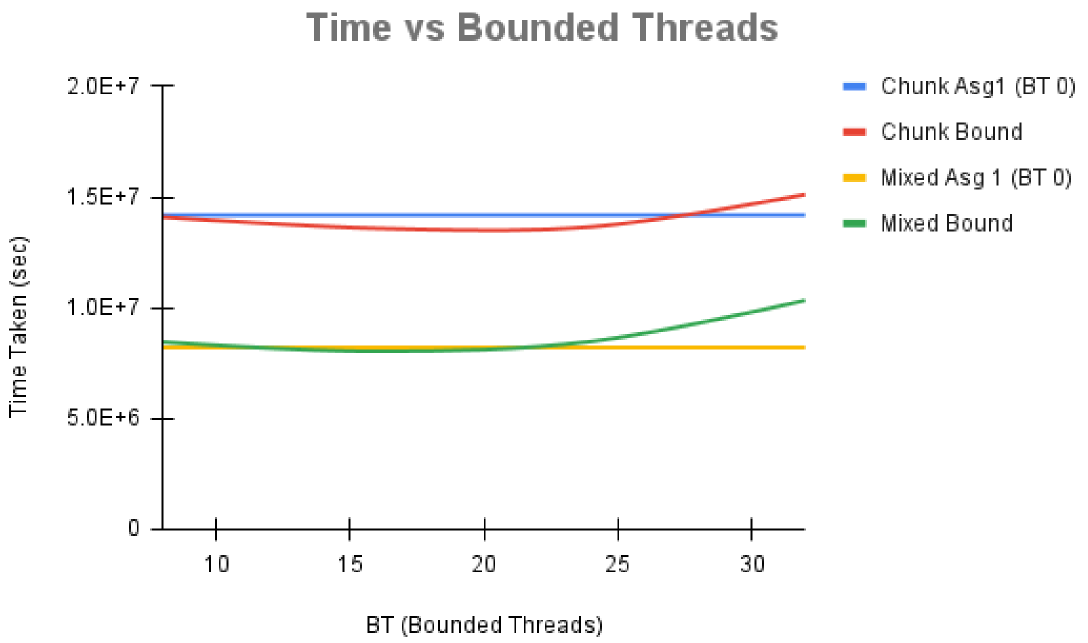
Constant	**Experiment 1: Time vs BT**				
N = 1024	**BT**	**Chunk Asg1 (BT 0)**	**Chunk Bound**	**Mixed Asg 1 (BT 0)**	**Mixed Bound**
K = 32	8	14183624	14097913	8214149	8452047
C = 4	16	14183624	13588892	8214149	8032779
b = 8 (K/C)	24	14183624	13654227	8214149	8475894
b = 0 means N, K only	32	14183624	15110915	8214149	10337559

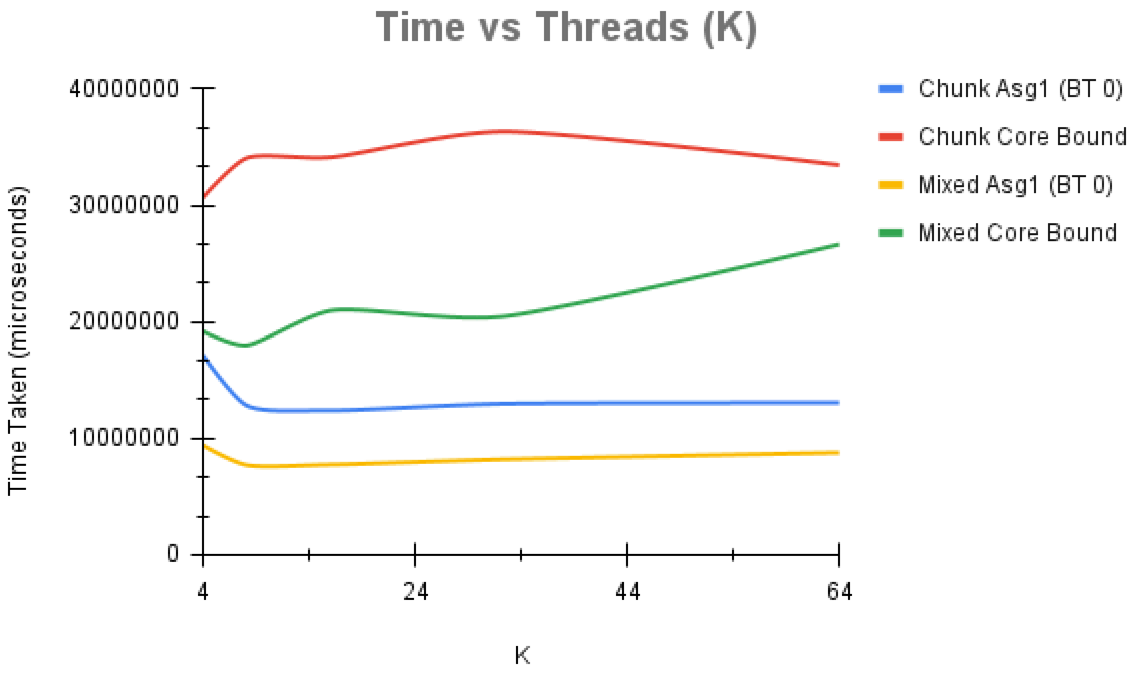
Experiment 2: Time vs Number of threads



Constant	**Experiment 2: Time vs K**				
N = 1024	**K**	**Chunk Asg1 (BT 0)**	**Chunk Core Bound**	**Mixed Asg1 (BT 0)**	**Mixed Core Bound**
C = 4	4	17123920	30665606	9397196	19194052
BT = 512	8	12876839	34009184	7730127	17960975
	16	12401630	34113846	7746829	20957013
	32	12969263	36327786	8194212	20425082
	64	13055470	33454613	8774209	26634246

Below given are the plots for the observation mentioned above respectively.





For Both the experiments it shows very strange behavior because generally as the no. of threads increases the time taken should decrease because the parallel computations increase. Also I have tried to run the program multiple times but still don’t know why this behavior is seen. I think there might be a problem with the aggregation of all the threads or threads not creating as per required or staying in particular core.