# MICS6002C-Assignment 1

**Xiwei Pan (50038447)**

## 1. Description of Single-threaded Version

The original single-threaded program initializes four $N \times N$ integer matrices with pseudorandom values and performs four matrix multiplications (index $0 \times 1$, $1 \times 2$, $2 \times 3$, $1 \times 3$), accumulating the sum of all resulting elements modulo $10,000,000$ and printing the final total.

## 2. Runtime Experimental Results

To evaluate the performance difference, both the single-threaded and multithreaded versions were compiled with identical options and executed multiple times on the same machine. The average runtimes over 10 runs were measured using the `time` command. The results, summarized in the table below, clearly show that the multithreaded implementation achieves a substantial reduction in execution time compared to the single-threaded version.

- The system is equipped with dual Intel Xeon Platinum 8378A processors (x86_64), providing 64 cores and 128 threads at 3.0 GHz.
- Compile command:

```
❯ g++ -std=c++14 -O2 -pthread -o multi multi.cpp
```

| **Averaged over 10 runs** | Single-threaded | Multithreaded (4 threads) |
|---|---|---|
| Real Runtime (s) | 2.791 | 0.793 |
| User CPU Time (s) | 2.779 | 2.859 |
| System CPU Time (s) | 0.005 | 0.012 |
| Speedup(×) | – | 3.52 |

Table 1: Runtime Comparison

The multithreaded run achieved a 3.52× speedup. However, the User CPU time is less than four times the Real time, indicating incomplete parallel scaling across the four threads, unlike the single-threaded case where User and Real time nearly coincide.

## 3. Description of Multithreaded Version

Each of the four threads multiplies two matrices and stores partial results in `sum`. After computing, the threads acquire a `mutex` to safely update the shared global `total`.