Matrix Multiplication Documentation

# Results

|  |  |  |  |
| --- | --- | --- | --- |
| **Paradigm** | **Number of Threads** | **Size of Matrix** | **Time** |
| Sequential | 1 | 10 | 41 microseconds |
| Parallel | 12 | 10 | 7822 microseconds |
| OpenMP | 12 | 10 | 62 microseconds |

# Parallel Program Decomposition

The matrix multiplication program consists of three stages:

1. Allocating memory
2. Calculating the matrix
3. Printing results

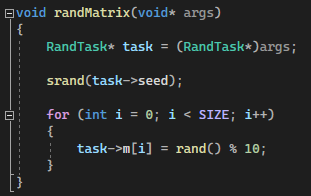
Stage 1 and 2 are parallelised. In stage 1, each thread is responsible for allocating one row of each of the three matrices. Each thread is given a task that is structured like this:

Graphical user interface, text

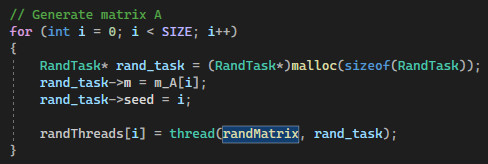
Description automatically generated

‘m’ is a pointer to the row in the matrix while the seed is given to the random number generator.

Each thread executes the following code to allocate the memory:



The threads are instantiated here:



Stage 2 follows a very similar process. Each thread is given a row from each of the matrices. Here is the task structure for multiplying the rows of the matrices:

Text

Description automatically generated

The code below shows what each thread is doing when calculating matrix C. It is finding the dot product from a row in matrix A and a column in matrix B. The result is stored in matrix C.

Text

Description automatically generated

Here is the code that instantiates the threads:

Text

Description automatically generated

# OpenMP Program Decomposition

The OpenMP version of the program is almost identical to the sequential version of the program. The code is structured exactly the same. The only difference is that the OpenMP version includes calls to OpenMP functions that parallelise the code automatically.

The code below shows the function that allocates the memory for each of the matrices. It is exactly the same as the sequential code.

Text

Description automatically generated

A feature called dynamic teams needed to be disabled to allow the number of threads to be defined by the program.

Here is the code for multiplying the matrices together:

