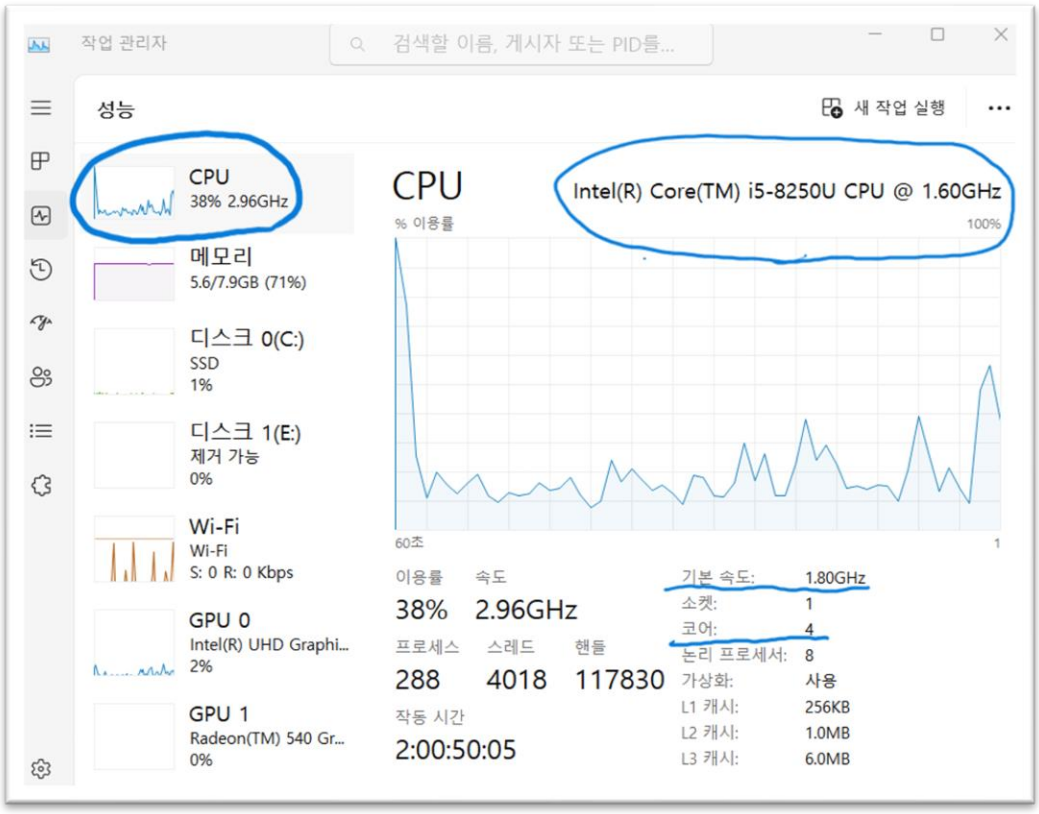
Problem2
Report

Student NO: 20183784
Student Name: 노현진

[Environment]

- **CPU**

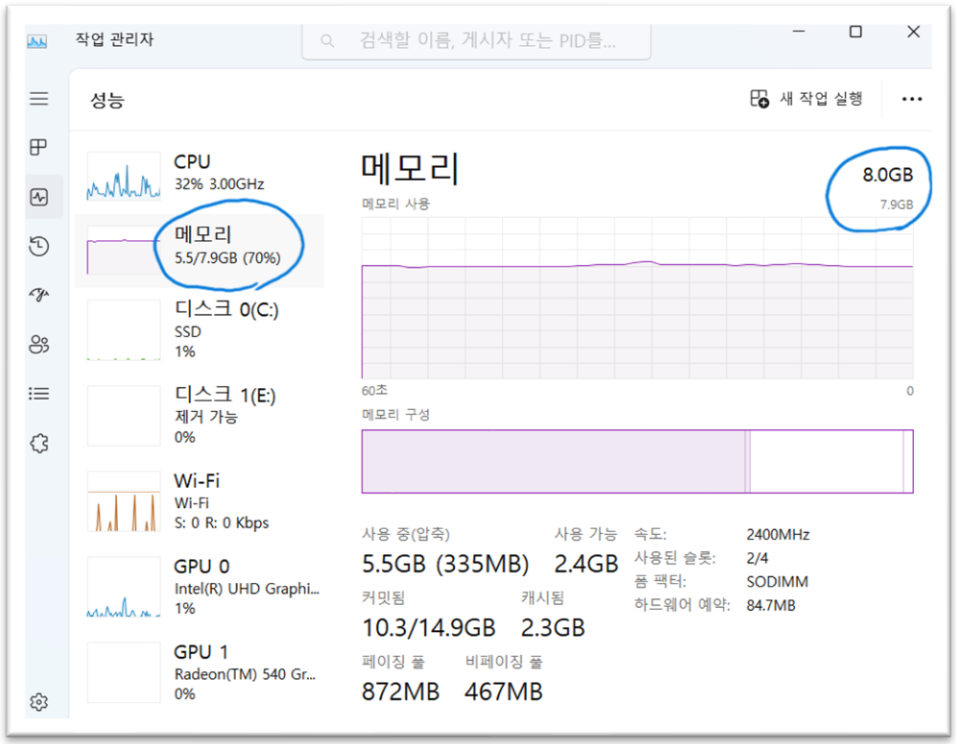


CPU type: Intel® Core™ i5-8250U CPU

Clock Speed: 1.80GHz

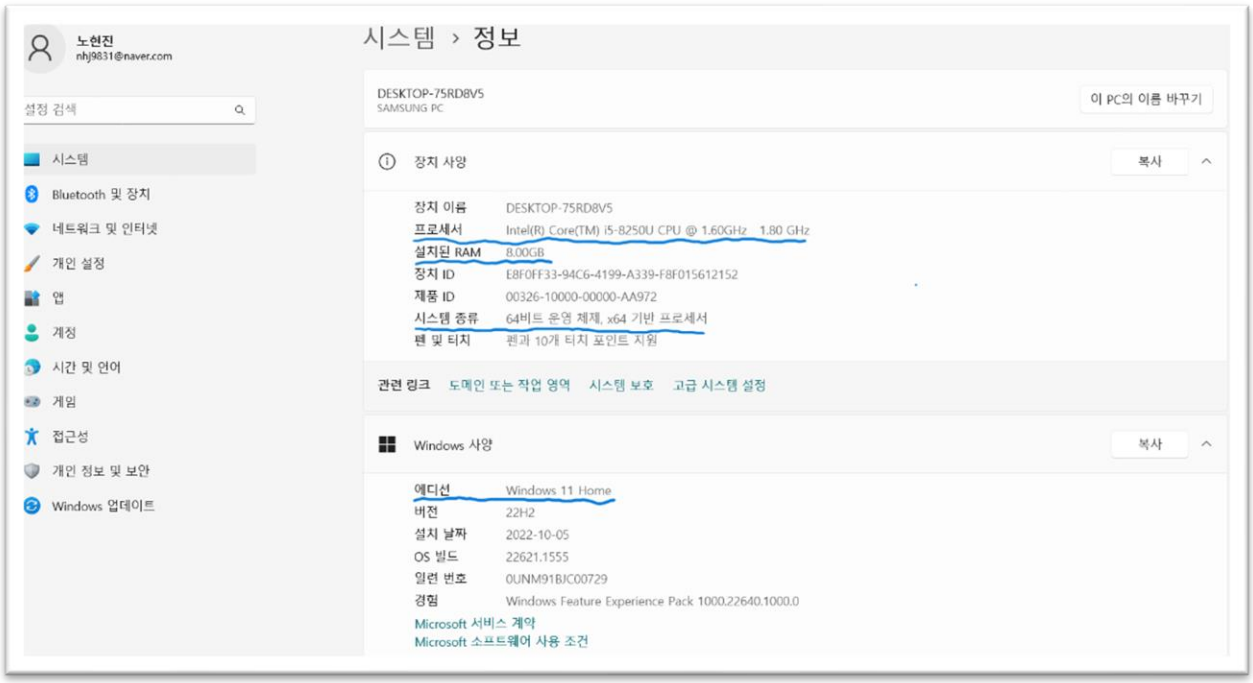
Number of cores: 4

- **Memory**



Memory size: 8.0GB

- **OS**



OS type: Windows 11

[Source Code]

- **MatmultD.java**

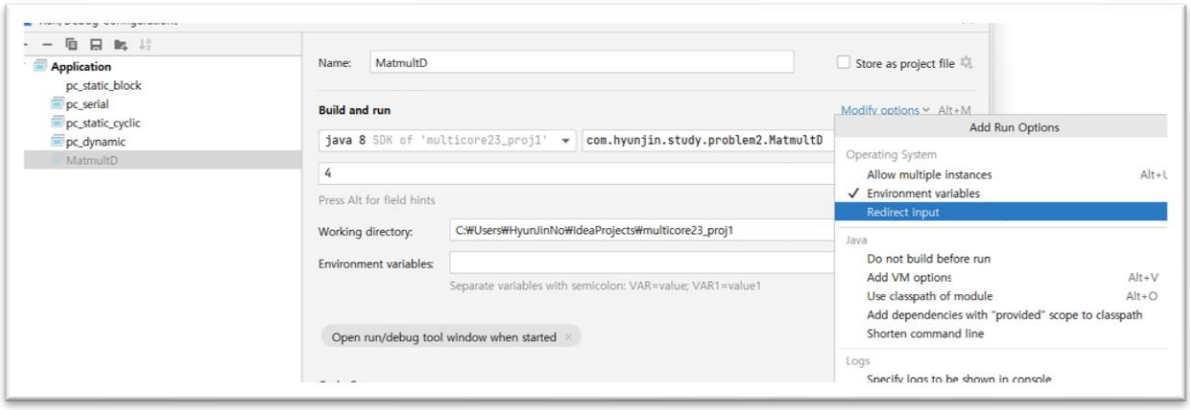
```
package com.hyunjin.study.problem2;

import java.util.*;
import java.lang.*;

/*
 This program should print the following values:
 (1) execution time of each thread
 (2) execution time when using all threads
 (3) sum of all elements in the resulting matrix
*/

public final class MatmultD {
    private static final Scanner sc = new Scanner(System.in);

    public static void main(String[] args) {
        int thread_no = 1;
        if (args.length == 1) {
            thread_no = Integer.parseInt(args[0]); // Possible number of threads:
1, 2, 4, 6, 8, 10, 12, 14, 16, 32
        }

        int[][] a = readMatrix();
        int[][] b = readMatrix();

        if (a.length == 0) {
```

```
            System.out.println("Invalid Input.");
            return;
        }
        if (a[0].length != b.length) {
            System.out.println("Invalid Dimension.");
            return;
        }

        int[][] c = new int[a.length][b[0].length];
        MatrixThread[] matrixThreads = new MatrixThread[thread_no];

        int endRow = a.length;
        int cycle = thread_no;
        for (int i = 1; i <= thread_no; i++) {
            int startRow = i - 1;
            matrixThreads[i - 1] = new MatrixThread(a, b, c, startRow, endRow,
cycle);
        }

        long startTime = System.currentTimeMillis();

        for (MatrixThread matrixThread : matrixThreads) {
            matrixThread.start();
        }

        try {
            for (MatrixThread matrixThread : matrixThreads) {
                matrixThread.join();
            }
        } catch (Exception ignored) {}

        long endTime = System.currentTimeMillis();
        long timeDiff = endTime - startTime;

        printMatrix(c);
        System.out.printf("\n[thread_no]:%2d , [Time]:%4d ms\n", thread_no,
timeDiff);
    }

    public static int[][] readMatrix() {
        int rows = sc.nextInt();
        int cols = sc.nextInt();
        int[][] result = new int[rows][cols];

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result[i][j] = sc.nextInt();
            }
        }
        return result;
    }

    public static void printMatrix(int[][] mat) {
        System.out.println("\nMatrix[" + mat.length + "][" + mat[0].length + "]");
        int cols = mat[0].length;
        int sum = 0;
        for (int[] ints : mat) {
            for (int j = 0; j < cols; j++) {
                //System.out.printf("%4d " , ints[j]);
                sum += ints[j];
            }
            //System.out.println();
        }
        //System.out.println();
        System.out.println("Matrix Sum = " + sum + "\n");
    }
```

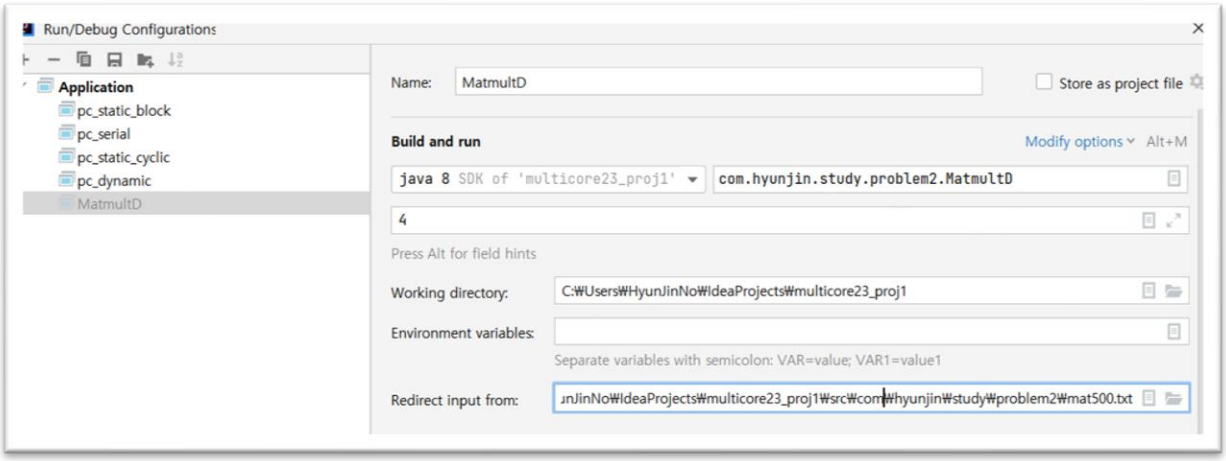
```
}

class MatrixThread extends Thread {
    private final int[][] a; // a[m][n]
    private final int[][] b; // b[n][p]
    private final int[][] c; // c[m][p]
    private final int startRow;
    private final int endRow;
    private final int cycle;

    public MatrixThread(int[][] a, int[][] b, int[][] c, int startRow, int endRow,
int cycle) {
        this.a = a;
        this.b = b;
        this.c = c;
        this.startRow = startRow;
        this.endRow = endRow;
        this.cycle = cycle;
    }

    @Override
    public void run() {
        long startTime = System.currentTimeMillis();

        for (int i = startRow; i < endRow; i += cycle) {
            for (int j = 0; j < c[0].length; j++) {
                for (int k = 0; k < a[0].length; k++) {
                    c[i][j] += a[i][k] * b[k][j];
                }
            }
        }

        long endTime = System.currentTimeMillis();
        long timeDiff = endTime - startTime;
        System.out.println(this.getName() + " ==> Execution Time: " + timeDiff +
"ms");
    }
}
```

- **How to compile and execute**
  1. **Firstly, install Intellij IDEA.**
  2. **After installation, open the submitted file.**
  3. **Before execution, click "Edit Configurations.."**

4. **Click “Modify Options” > “Redirect Input”**

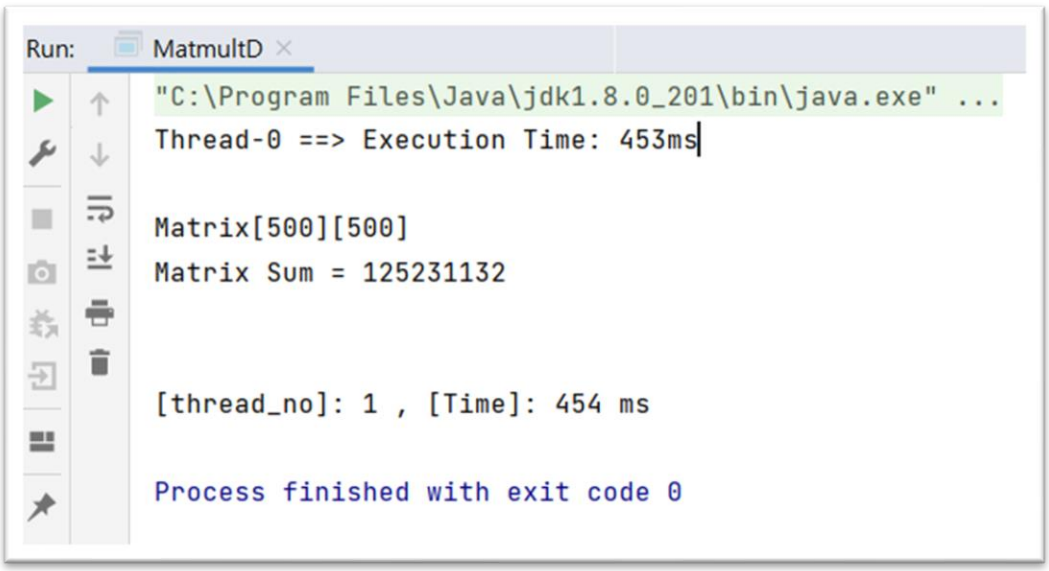


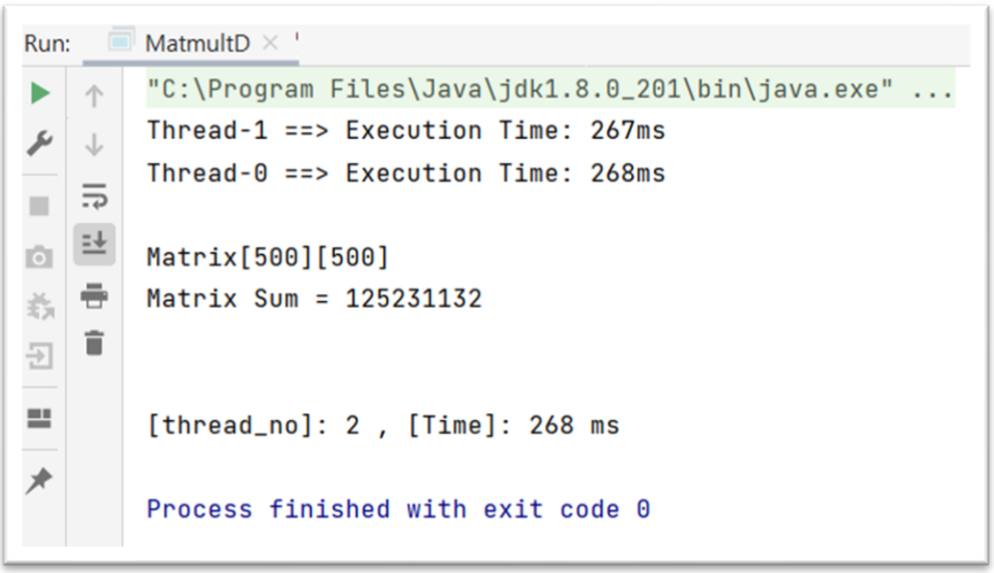
5. **Set the argument. In this picture, “4” means “thread_no”. And pass the absolute file path of “mat500.txt” in the “Redirect Input from:”**

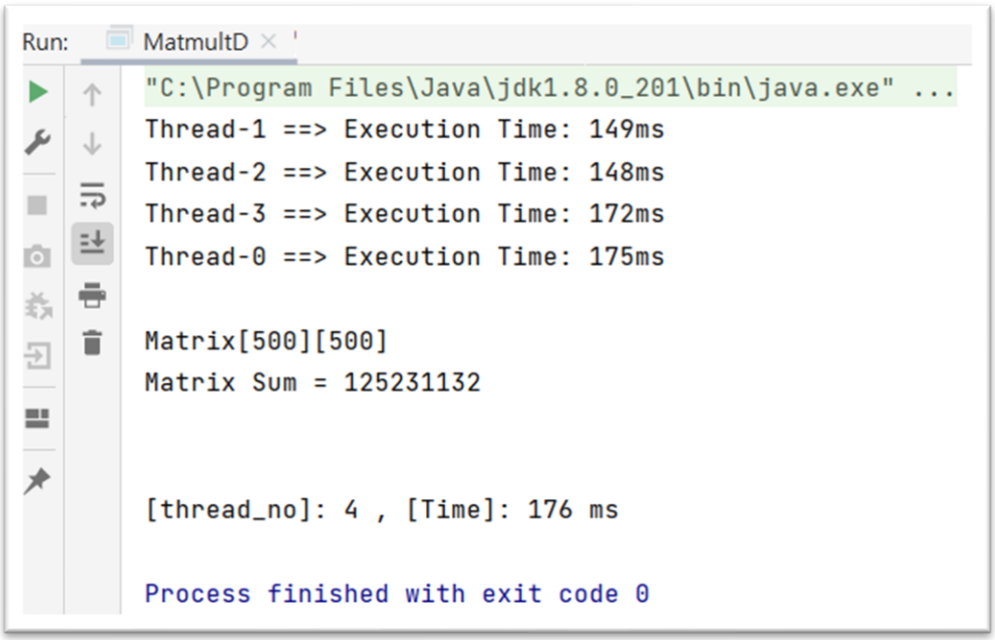


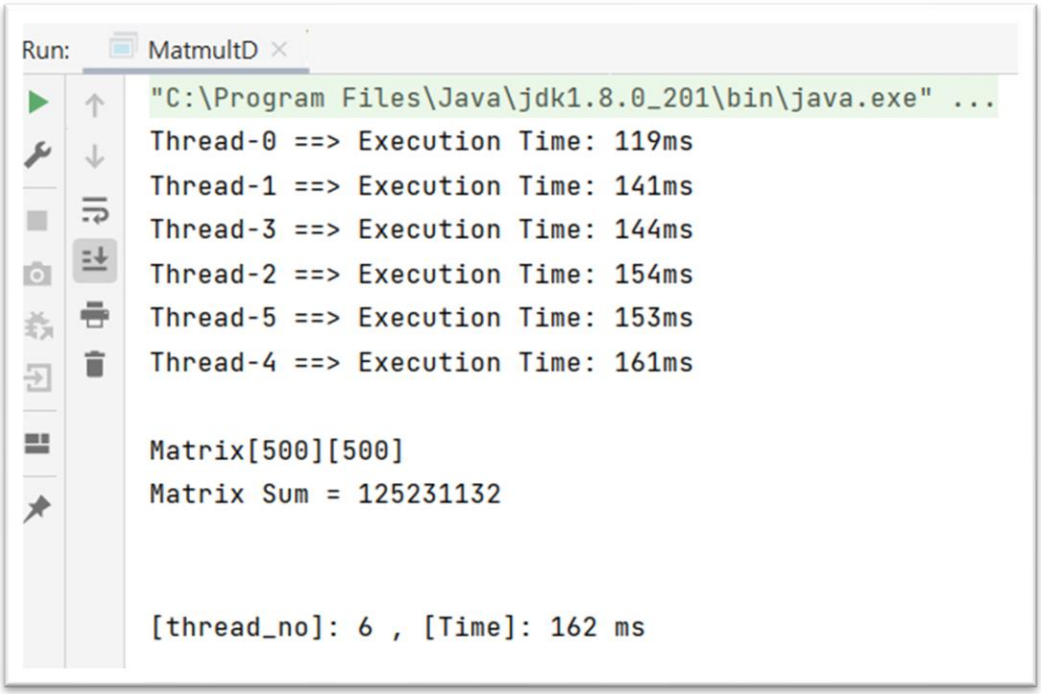
6. **Finally, run the source code.**

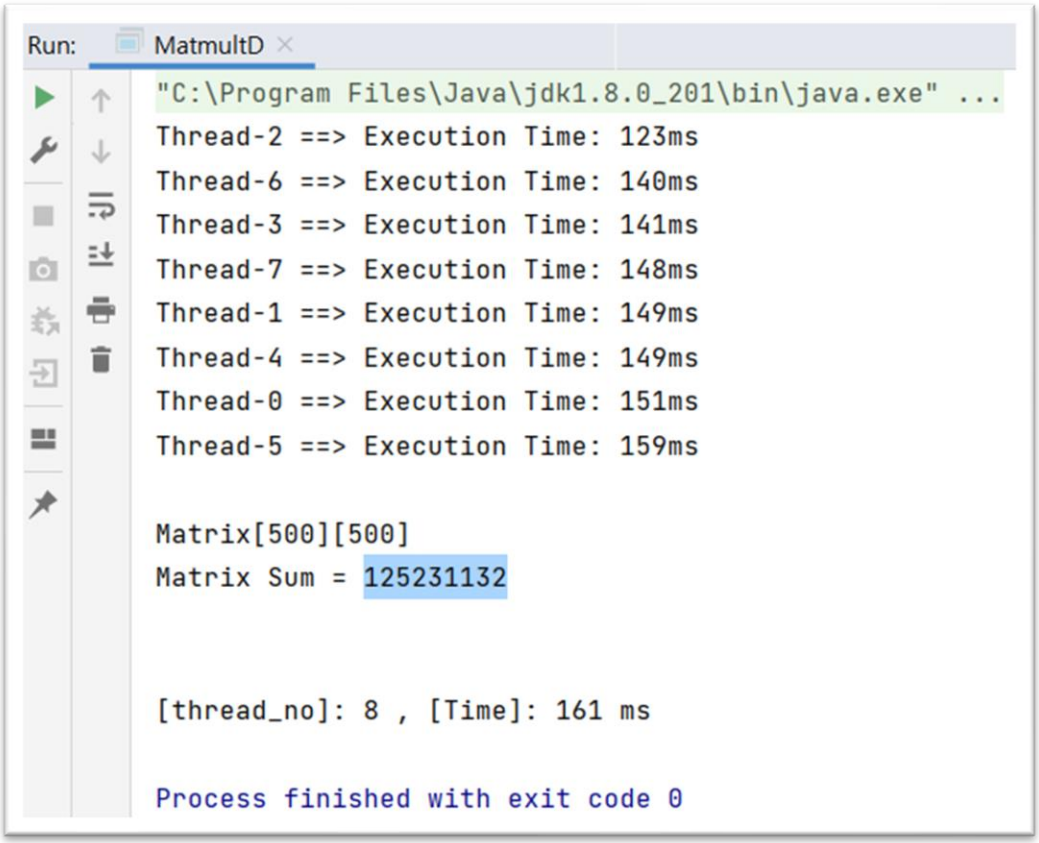
[Results]

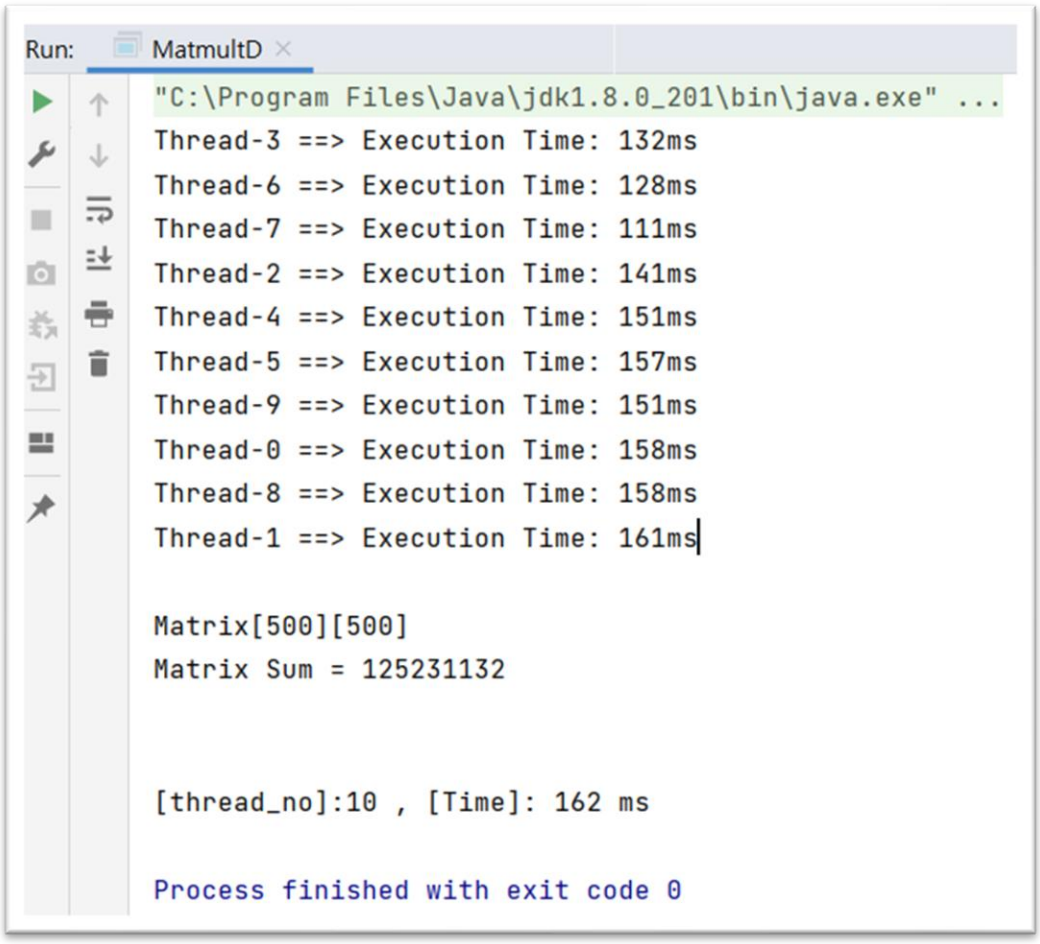


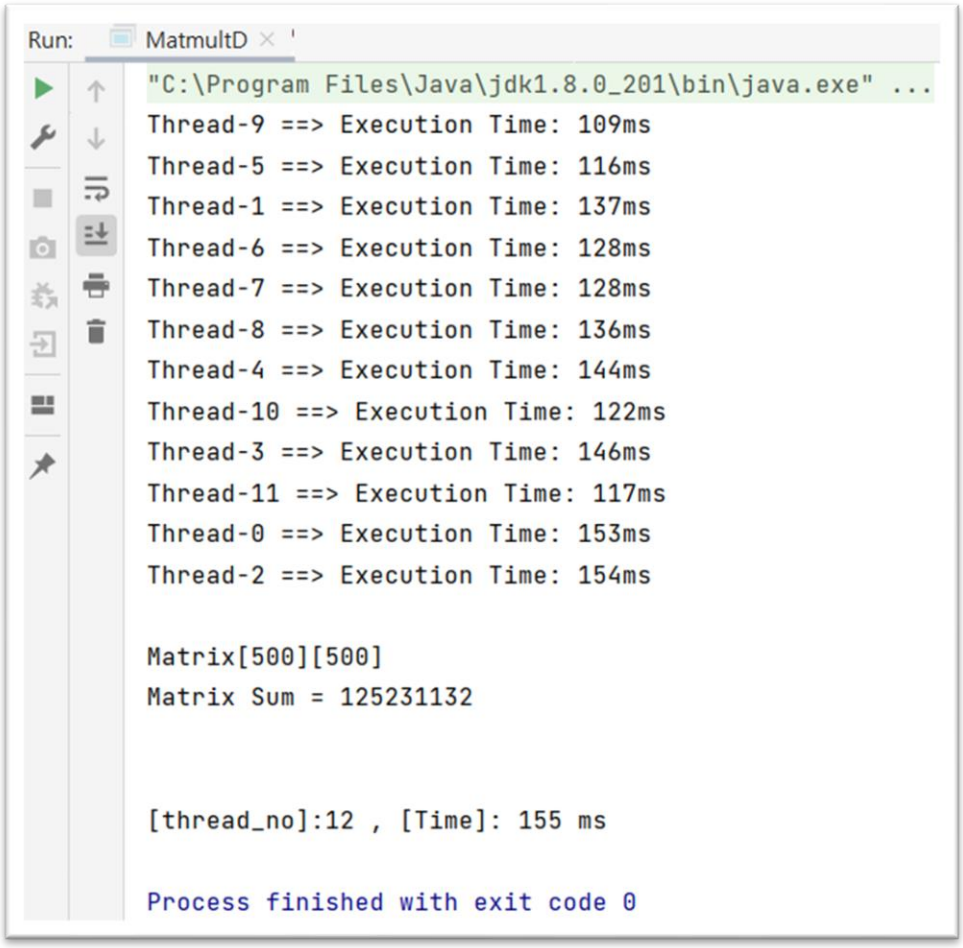

Run: MatmultD
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Thread-1 ==> Execution Time: 267ms
Thread-0 ==> Execution Time: 268ms
Matrix[500][500]
Matrix Sum = 125231132
[thread_no]: 2 , [Time]: 268 ms
Process finished with exit code 0

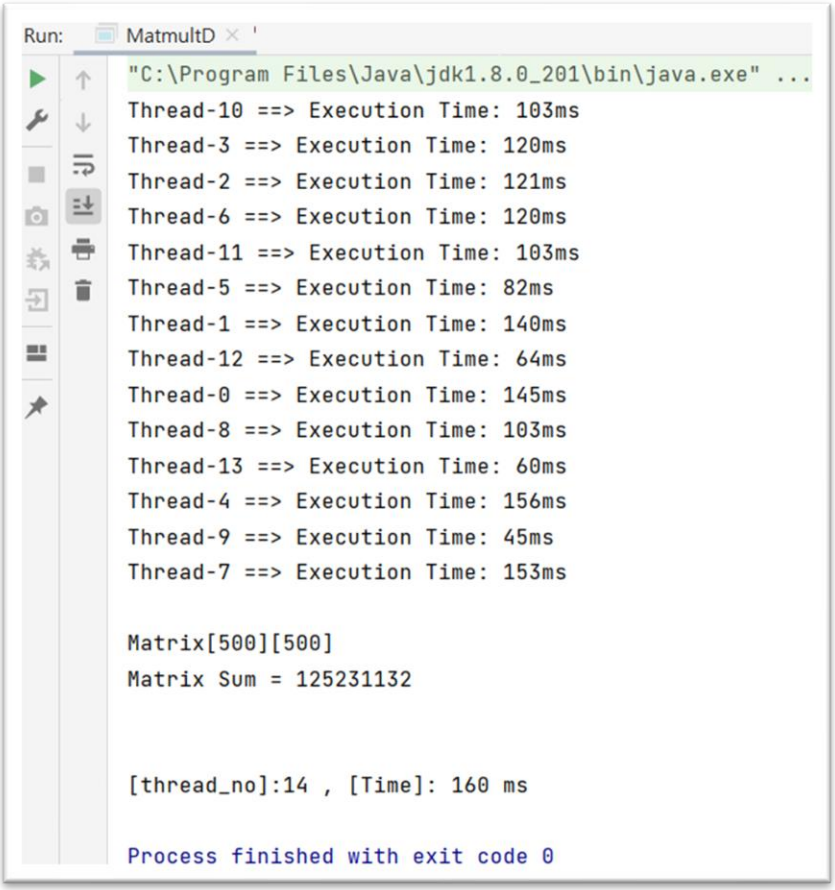

Run: MatmultD
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Thread-1 ==> Execution Time: 149ms
Thread-2 ==> Execution Time: 148ms
Thread-3 ==> Execution Time: 172ms
Thread-0 ==> Execution Time: 175ms
Matrix[500][500]
Matrix Sum = 125231132
[thread_no]: 4 , [Time]: 176 ms
Process finished with exit code 0

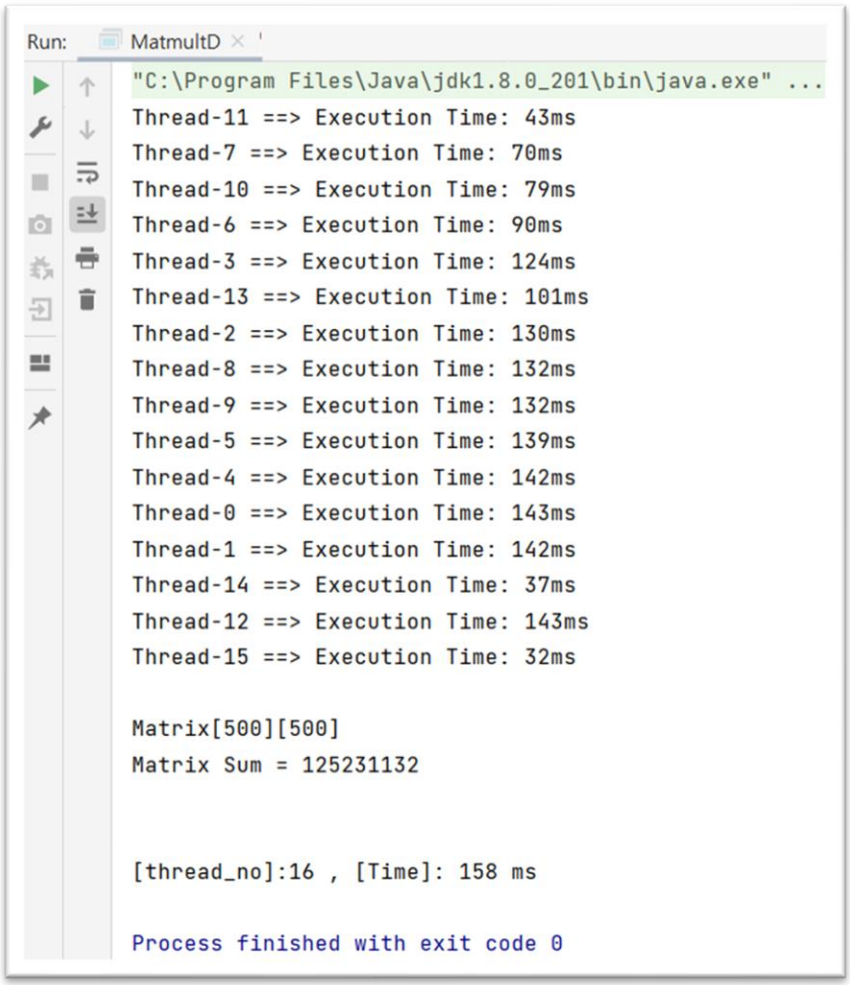

Run: MatmultD
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Thread-0 ==> Execution Time: 119ms
Thread-1 ==> Execution Time: 141ms
Thread-3 ==> Execution Time: 144ms
Thread-2 ==> Execution Time: 154ms
Thread-5 ==> Execution Time: 153ms
Thread-4 ==> Execution Time: 161ms
Matrix[500][500]
Matrix Sum = 125231132
[thread_no]: 6 , [Time]: 162 ms

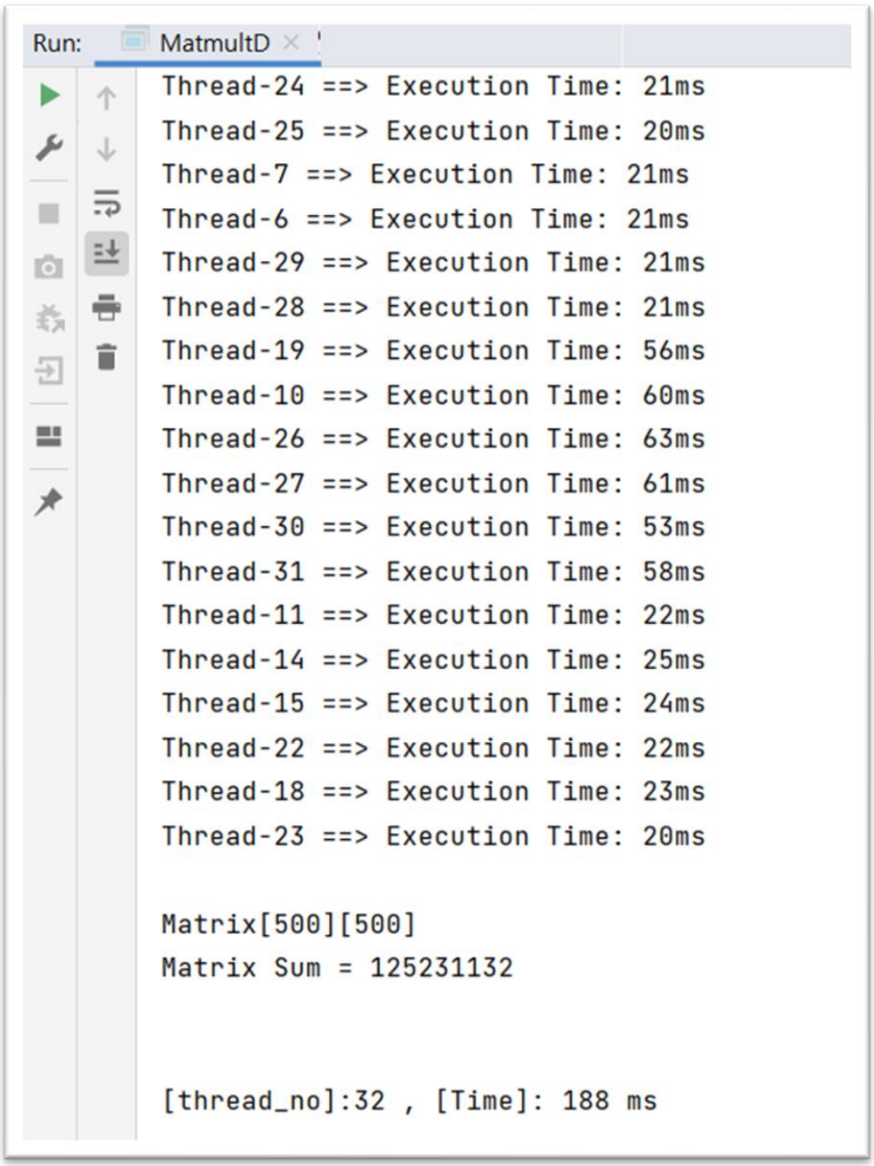

Run:
MatmultD
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Thread-2 ==> Execution Time: 123ms
Thread-6 ==> Execution Time: 140ms
Thread-3 ==> Execution Time: 141ms
Thread-7 ==> Execution Time: 148ms
Thread-1 ==> Execution Time: 149ms
Thread-4 ==> Execution Time: 149ms
Thread-0 ==> Execution Time: 151ms
Thread-5 ==> Execution Time: 159ms
Matrix[500][500]
Matrix Sum = 125231132
[thread_no]: 8 , [Time]: 161 ms
Process finished with exit code 0


Run: MatmultD
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Thread-3 ==> Execution Time: 132ms
Thread-6 ==> Execution Time: 128ms
Thread-7 ==> Execution Time: 111ms
Thread-2 ==> Execution Time: 141ms
Thread-4 ==> Execution Time: 151ms
Thread-5 ==> Execution Time: 157ms
Thread-9 ==> Execution Time: 151ms
Thread-0 ==> Execution Time: 158ms
Thread-8 ==> Execution Time: 158ms
Thread-1 ==> Execution Time: 161ms
Matrix[500][500]
Matrix Sum = 125231132
[thread_no]:10 , [Time]: 162 ms
Process finished with exit code 0


Run: MatmultD
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Thread-9 ==> Execution Time: 109ms
Thread-5 ==> Execution Time: 116ms
Thread-1 ==> Execution Time: 137ms
Thread-6 ==> Execution Time: 128ms
Thread-7 ==> Execution Time: 128ms
Thread-8 ==> Execution Time: 136ms
Thread-4 ==> Execution Time: 144ms
Thread-10 ==> Execution Time: 122ms
Thread-3 ==> Execution Time: 146ms
Thread-11 ==> Execution Time: 117ms
Thread-0 ==> Execution Time: 153ms
Thread-2 ==> Execution Time: 154ms
Matrix[500][500]
Matrix Sum = 125231132
[thread_no]:12 , [Time]: 155 ms
Process finished with exit code 0

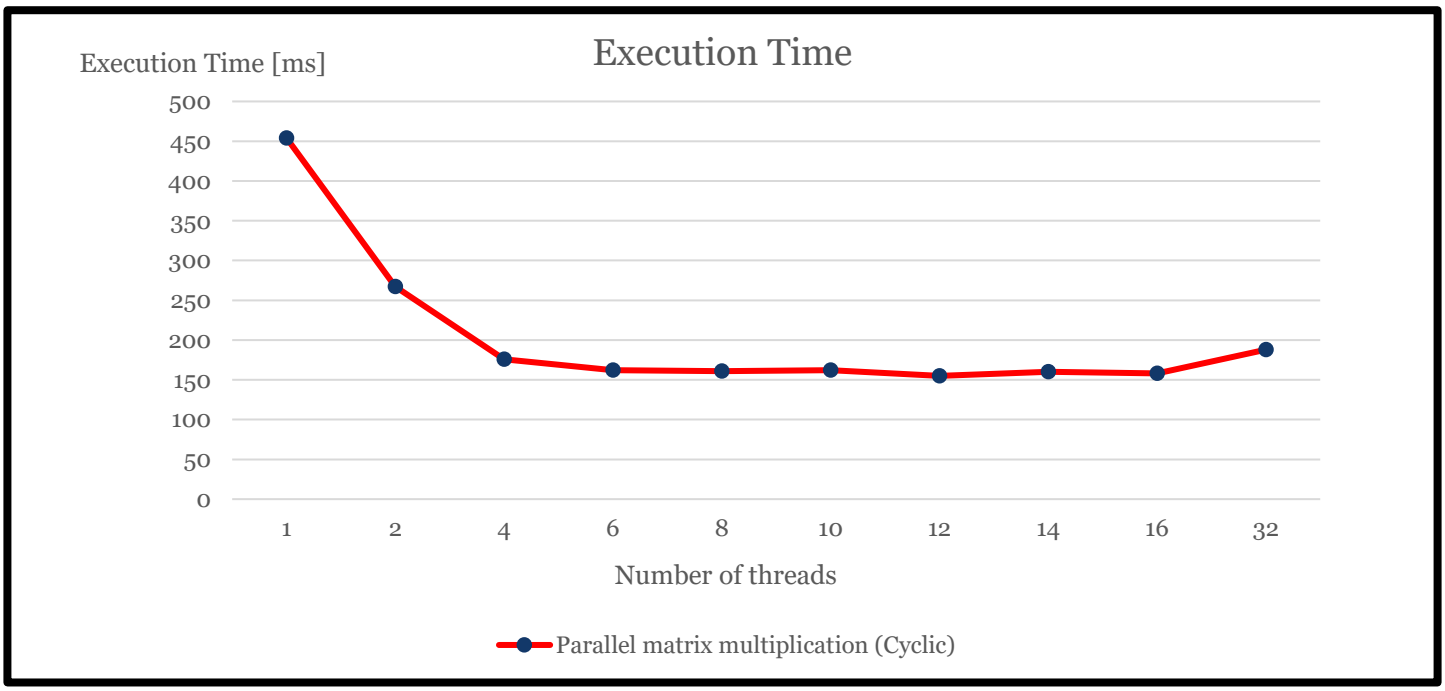

Run: MatmultD
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Thread-10 ==> Execution Time: 103ms
Thread-3 ==> Execution Time: 120ms
Thread-2 ==> Execution Time: 121ms
Thread-6 ==> Execution Time: 120ms
Thread-11 ==> Execution Time: 103ms
Thread-5 ==> Execution Time: 82ms
Thread-1 ==> Execution Time: 140ms
Thread-12 ==> Execution Time: 64ms
Thread-0 ==> Execution Time: 145ms
Thread-8 ==> Execution Time: 103ms
Thread-13 ==> Execution Time: 60ms
Thread-4 ==> Execution Time: 156ms
Thread-9 ==> Execution Time: 45ms
Thread-7 ==> Execution Time: 153ms
Matrix[500][500]
Matrix Sum = 125231132
[thread_no]:14 , [Time]: 160 ms
Process finished with exit code 0


Run: MatmultD
"C:\Program Files\Java\jdk1.8.0_201\bin\java.exe" ...
Thread-11 ==> Execution Time: 43ms
Thread-7 ==> Execution Time: 70ms
Thread-10 ==> Execution Time: 79ms
Thread-6 ==> Execution Time: 90ms
Thread-3 ==> Execution Time: 124ms
Thread-13 ==> Execution Time: 101ms
Thread-2 ==> Execution Time: 130ms
Thread-8 ==> Execution Time: 132ms
Thread-9 ==> Execution Time: 132ms
Thread-5 ==> Execution Time: 139ms
Thread-4 ==> Execution Time: 142ms
Thread-0 ==> Execution Time: 143ms
Thread-1 ==> Execution Time: 142ms
Thread-14 ==> Execution Time: 37ms
Thread-12 ==> Execution Time: 143ms
Thread-15 ==> Execution Time: 32ms
Matrix[500][500]
Matrix Sum = 125231132
[thread_no]:16 , [Time]: 158 ms
Process finished with exit code 0


Run:
MatmultD
Thread-24 ==> Execution Time: 21ms
Thread-25 ==> Execution Time: 20ms
Thread-7 ==> Execution Time: 21ms
Thread-6 ==> Execution Time: 21ms
Thread-29 ==> Execution Time: 21ms
Thread-28 ==> Execution Time: 21ms
Thread-19 ==> Execution Time: 56ms
Thread-10 ==> Execution Time: 60ms
Thread-26 ==> Execution Time: 63ms
Thread-27 ==> Execution Time: 61ms
Thread-30 ==> Execution Time: 53ms
Thread-31 ==> Execution Time: 58ms
Thread-11 ==> Execution Time: 22ms
Thread-14 ==> Execution Time: 25ms
Thread-15 ==> Execution Time: 24ms
Thread-22 ==> Execution Time: 22ms
Thread-18 ==> Execution Time: 23ms
Thread-23 ==> Execution Time: 20ms
Matrix[500][500]
Matrix Sum = 125231132
[thread_no]:32 , [Time]: 188 ms

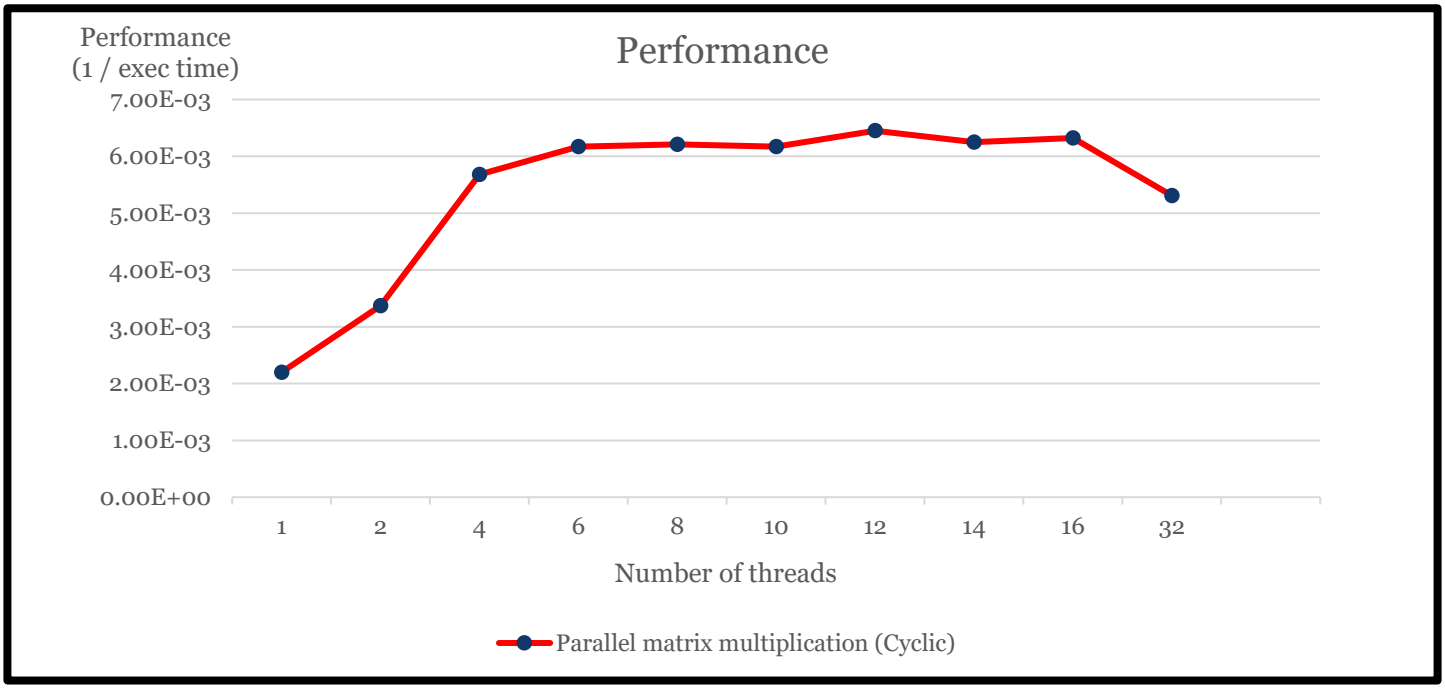
Execution Time

	1	2	4	6	8	10	12	14	16	32
Exec time	454ms	268ms	176ms	162ms	161ms	162ms	155ms	160ms	158ms	188ms



- **Performance**

	1	2	4	6	8	10	12	14	16	32
Performance (1/exec time)	2.20e-3	3.37e-3	5.68e-3	6.17e-3	6.21e-3	6.17e-3	6.45e-3	6.25e-3	6.32e-3	5.31e-3



[Explanation/Analysis on the Results]

The above program uses cyclic decomposition method. Each thread deals with allocated rows. In the above program, when number of threads is small and increases (ex. 1, 2, 4, …), program performance is improved dramatically. But, when number of threads is large and increases (ex. 8, 10, 12, …), program performance is not improved. And number of threads becomes very large (ex. 32), performance eventually decreases. It's because large number of threads makes large overhead, which makes program's performance unable to increase. Also, very large number of threads is the same as sequential computation, not parallel computation. So, the performance becomes the same as execution time when number of threads is only one. For example, if number of threads is 320, then the execution time is similar to one which has only one thread. So, we need to decide how many threads we use.