FALL – SEMESTER **Name: Nidhi Singh**

Course Code: MCSE503P **Reg. No:22MAI0015**

Course-Title: – Computer Architecture and Organization

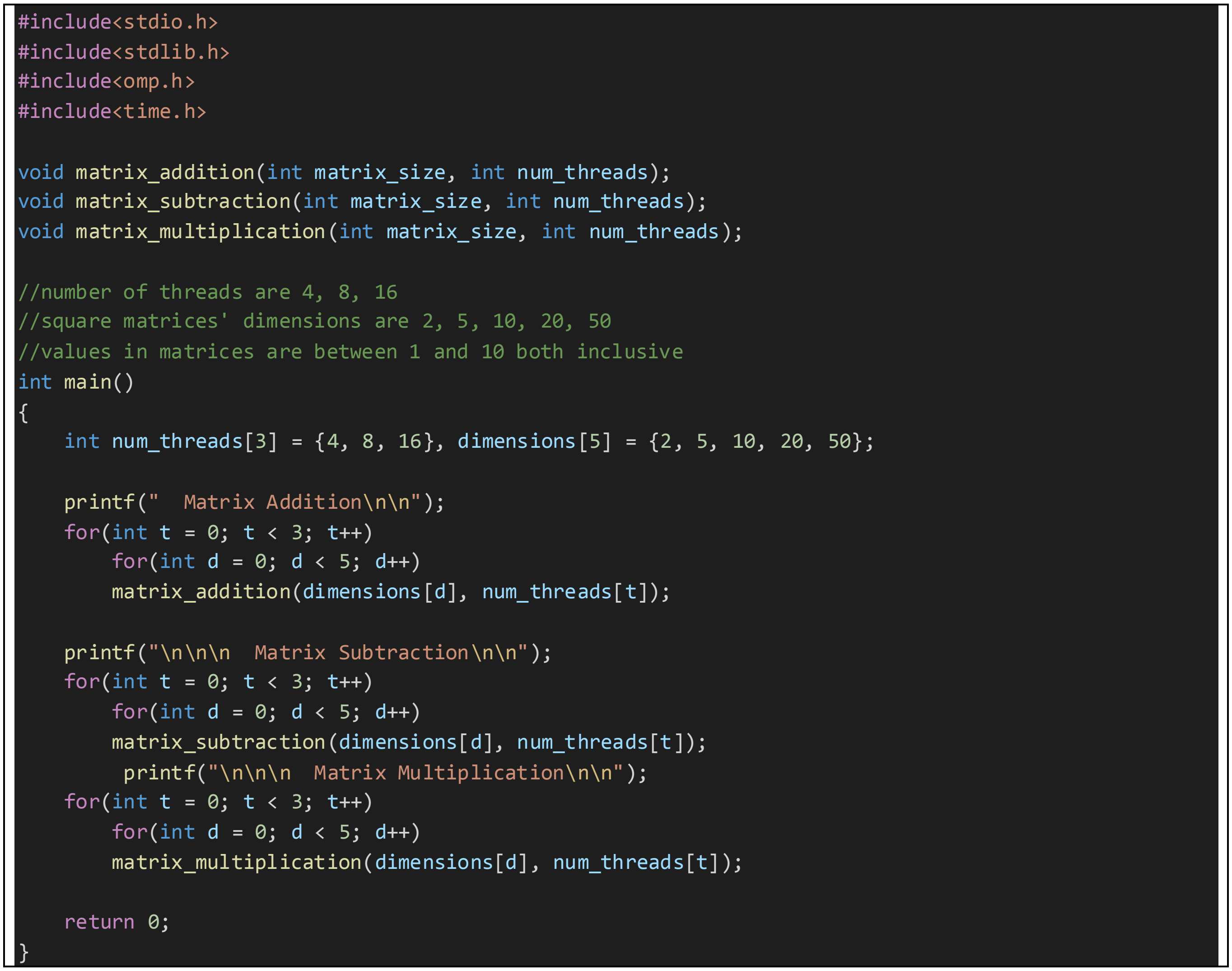
DIGITAL ASSIGNMENT - IV

(LAB) **Slot-** L53+L54

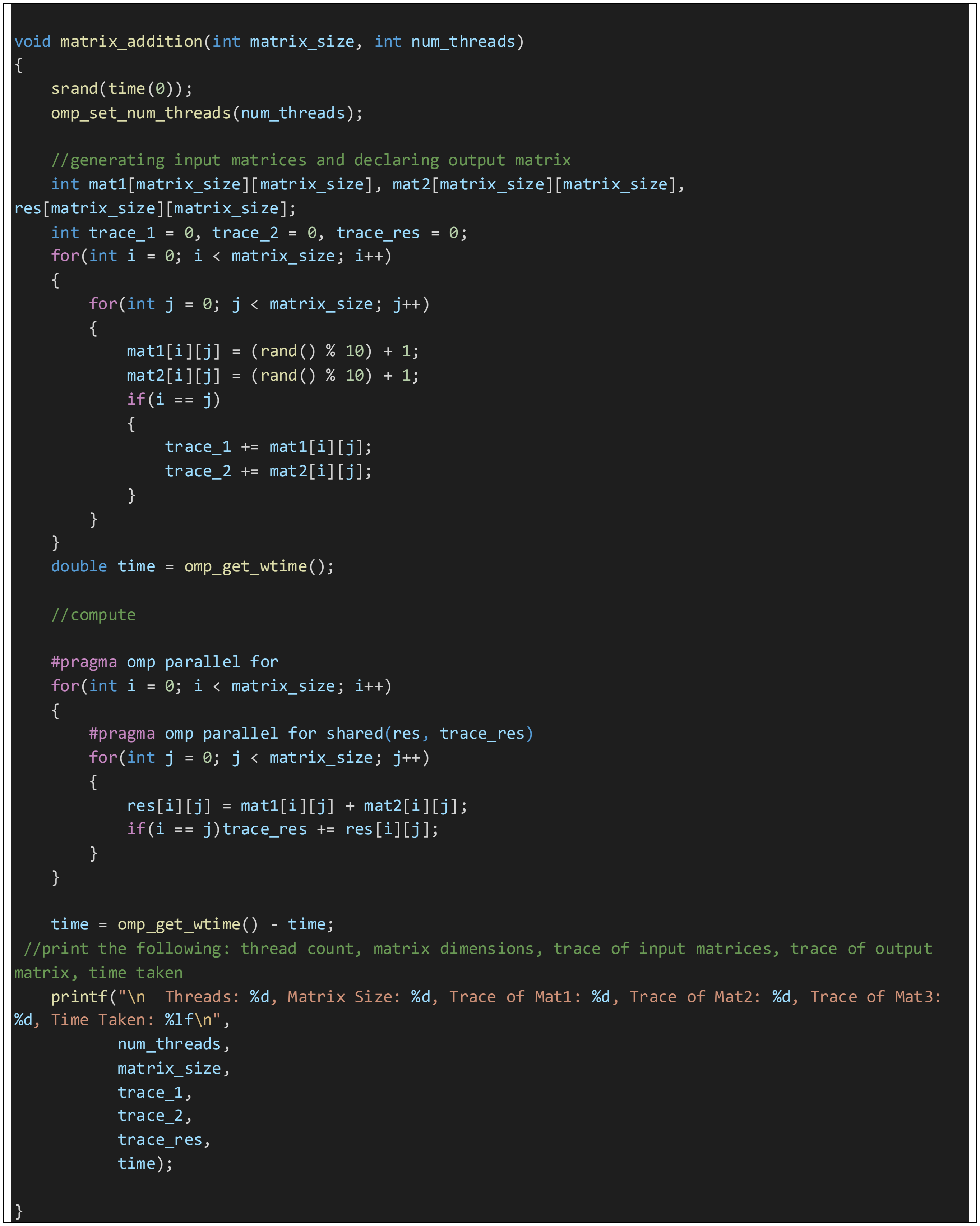
Faculty Name: M.Narayana Moorthi

Write an Open MP program using C for the following 2-D array and perform the following task. Matrix addition, subtraction and multiplication. Find the trace of input and resultant matrices. Also compute the time to perform the above task with different size of matrices (like 2 x 2, 5 x 5, 10 x 10, 20 x 20, 50 x 50) with different thread counts. Assign the matrix elements using random function.

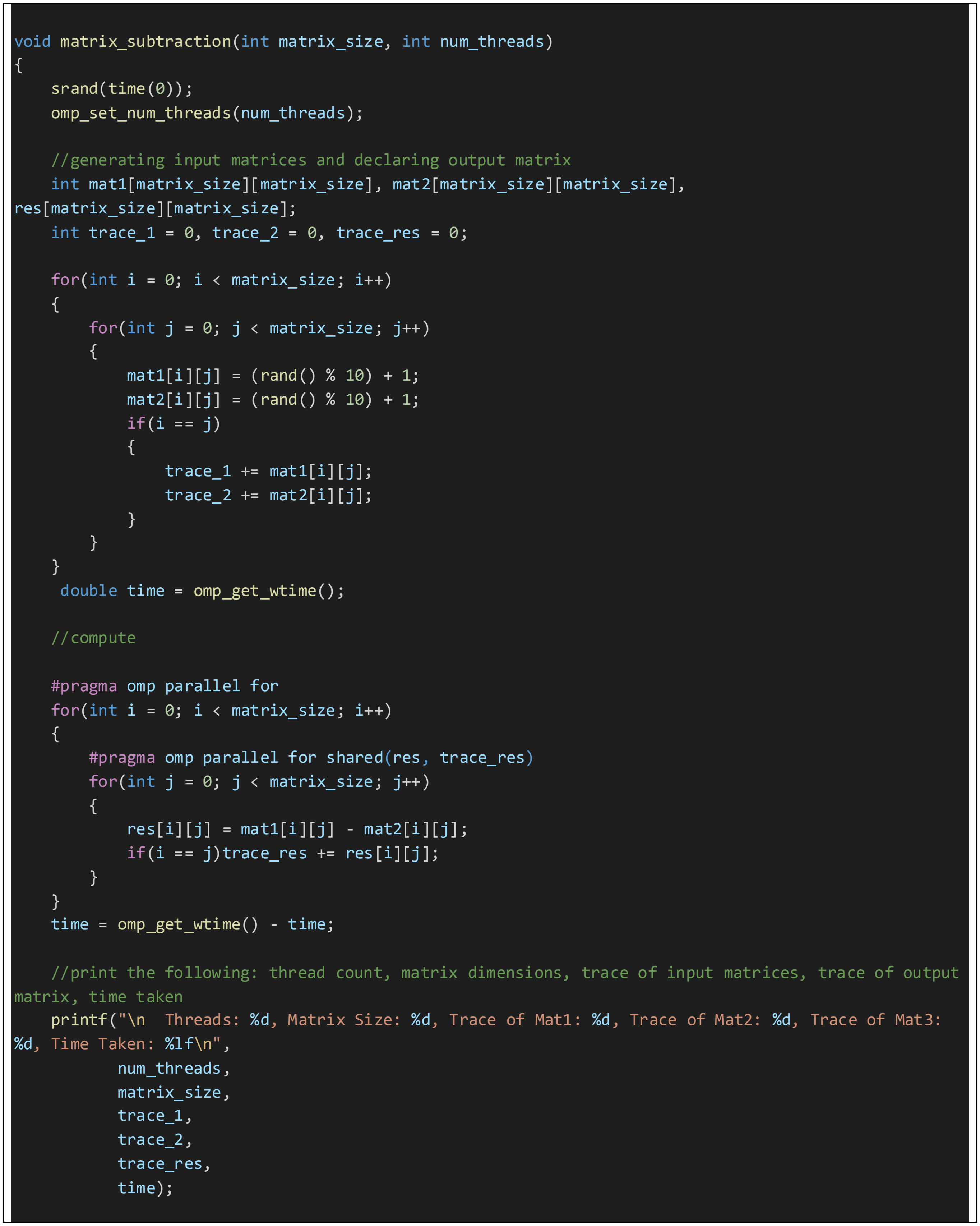
CODE :-



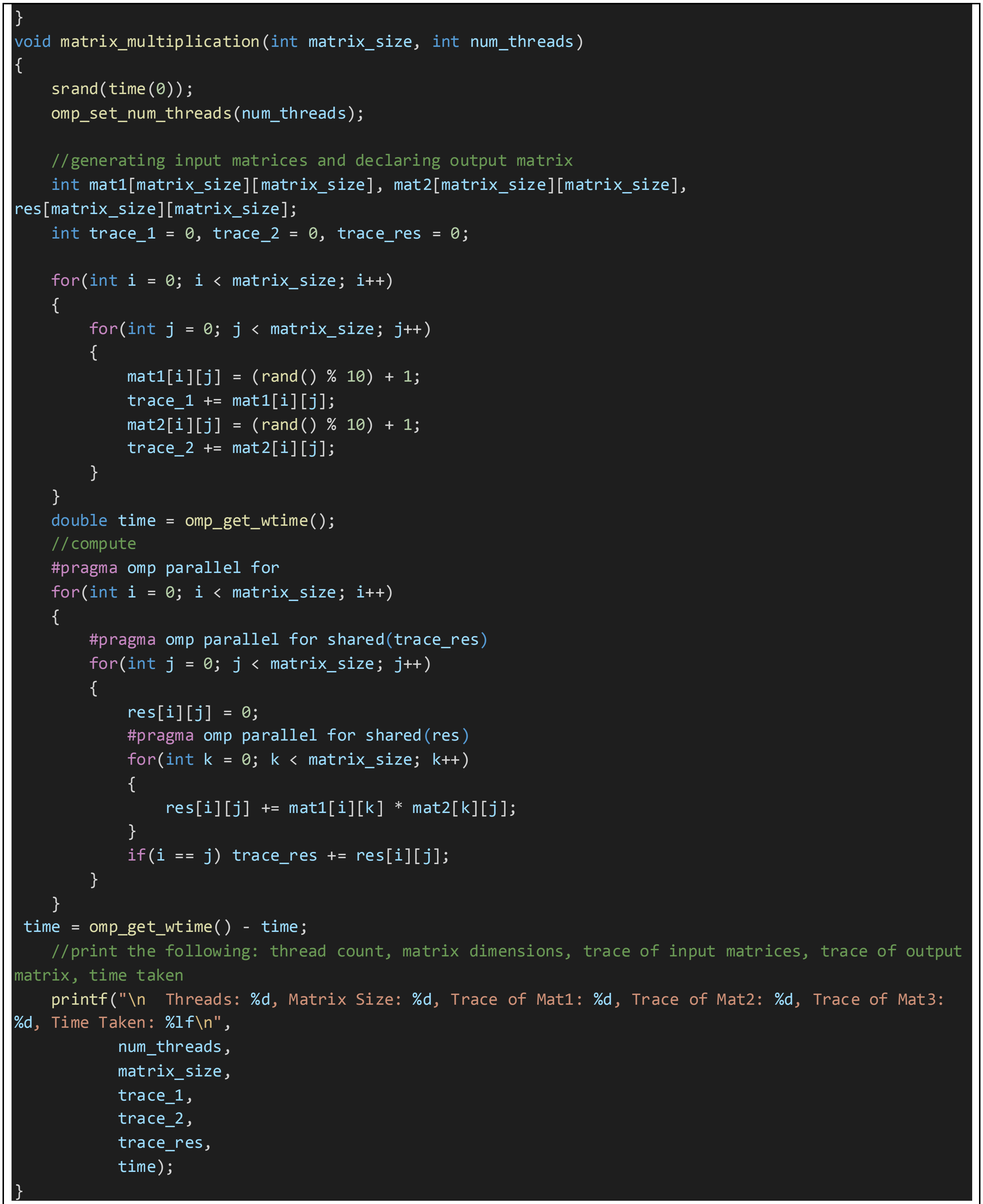
```
#include<stdio.h>
#include<stdlib.h>
#include<omp.h>
#include<time.h>

void matrix_addition(int matrix_size, int num_threads);
void matrix_subtraction(int matrix_size, int num_threads);
void matrix_multiplication(int matrix_size, int num_threads);

//number of threads are 4, 8, 16
//square matrices' dimensions are 2, 5, 10, 20, 50
//values in matrices are between 1 and 10 both inclusive
int main()
{
    int num_threads[3] = {4, 8, 16}, dimensions[5] = {2, 5, 10, 20, 50};

    printf("  Matrix Addition\n\n");
    for(int t = 0; t < 3; t++)
        for(int d = 0; d < 5; d++)
        matrix_addition(dimensions[d], num_threads[t]);

    printf("\n\n\n  Matrix Subtraction\n\n");
    for(int t = 0; t < 3; t++)
        for(int d = 0; d < 5; d++)
        matrix_subtraction(dimensions[d], num_threads[t]);
         printf("\n\n\n  Matrix Multiplication\n\n");
    for(int t = 0; t < 3; t++)
        for(int d = 0; d < 5; d++)
        matrix_multiplication(dimensions[d], num_threads[t]);

    return 0;
}
```



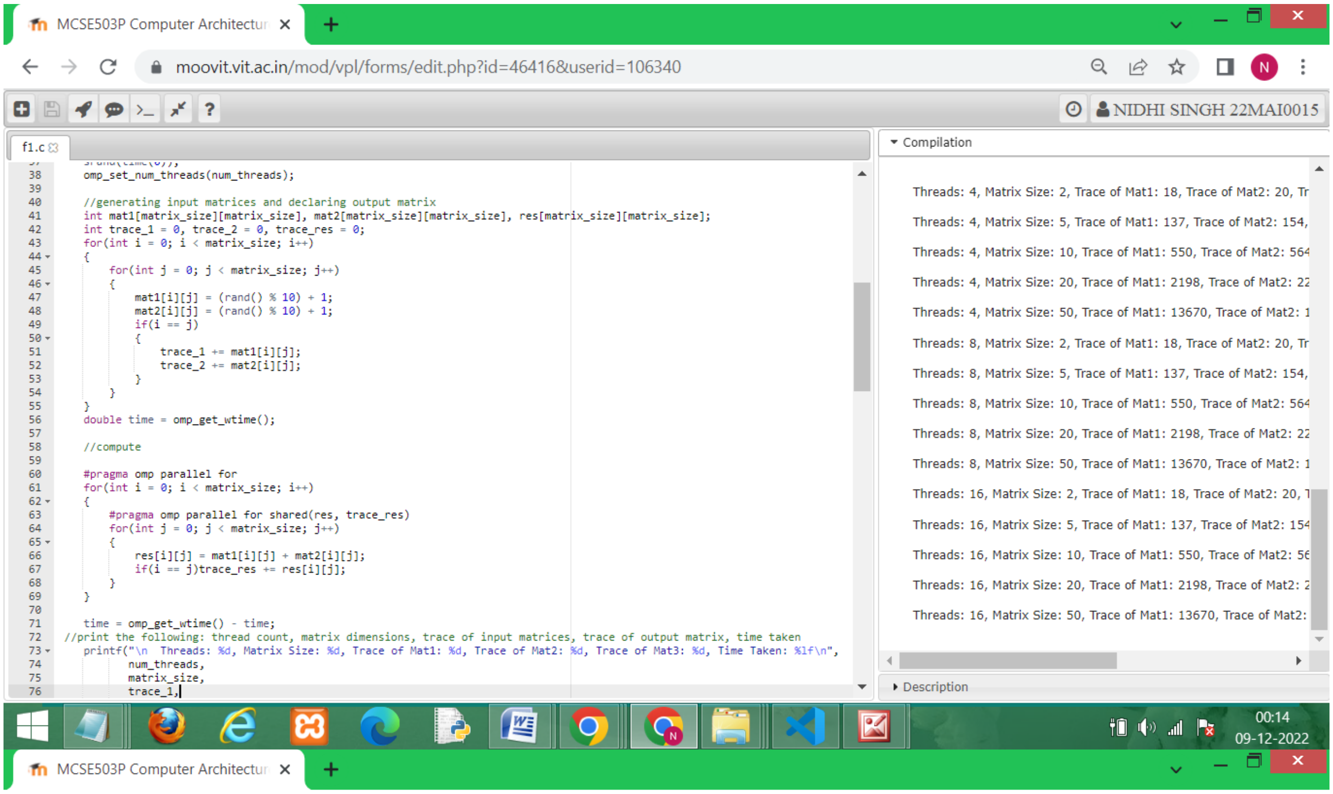
```
void matrix_addition(int matrix_size, int num_threads)
{
    srand(time(0));
    omp_set_num_threads(num_threads);

    //generating input matrices and declaring output matrix
    int mat1[matrix_size][matrix_size], mat2[matrix_size][matrix_size],
res[matrix_size][matrix_size];
    int trace_1 = 0, trace_2 = 0, trace_res = 0;
    for(int i = 0; i < matrix_size; i++)
    {
        for(int j = 0; j < matrix_size; j++)
        {
            mat1[i][j] = (rand() % 10) + 1;
            mat2[i][j] = (rand() % 10) + 1;
            if(i == j)
            {
                trace_1 += mat1[i][j];
                trace_2 += mat2[i][j];
            }
        }
    }
    double time = omp_get_wtime();

    //compute

    #pragma omp parallel for
    for(int i = 0; i < matrix_size; i++)
    {
        #pragma omp parallel for shared(res, trace_res)
        for(int j = 0; j < matrix_size; j++)
        {
            res[i][j] = mat1[i][j] + mat2[i][j];
            if(i == j)trace_res += res[i][j];
        }
    }

    time = omp_get_wtime() - time;
 //print the following: thread count, matrix dimensions, trace of input matrices, trace of output
matrix, time taken
    printf("\n  Threads: %d, Matrix Size: %d, Trace of Mat1: %d, Trace of Mat2: %d, Trace of Mat3:
%d, Time Taken: %lf\n",
           num_threads,
           matrix_size,
           trace_1,
           trace_2,
           trace_res,
           time);

}
```



```
void matrix_subtraction(int matrix_size, int num_threads)
{
    srand(time(0));
    omp_set_num_threads(num_threads);

    //generating input matrices and declaring output matrix
    int mat1[matrix_size][matrix_size], mat2[matrix_size][matrix_size],
res[matrix_size][matrix_size];
    int trace_1 = 0, trace_2 = 0, trace_res = 0;

    for(int i = 0; i < matrix_size; i++)
    {
        for(int j = 0; j < matrix_size; j++)
        {
            mat1[i][j] = (rand() % 10) + 1;
            mat2[i][j] = (rand() % 10) + 1;
            if(i == j)
            {
                trace_1 += mat1[i][j];
                trace_2 += mat2[i][j];
            }
        }
    }
     double time = omp_get_wtime();

    //compute

    #pragma omp parallel for
    for(int i = 0; i < matrix_size; i++)
    {
        #pragma omp parallel for shared(res, trace_res)
        for(int j = 0; j < matrix_size; j++)
        {
            res[i][j] = mat1[i][j] - mat2[i][j];
            if(i == j)trace_res += res[i][j];
        }
    }
    time = omp_get_wtime() - time;

    //print the following: thread count, matrix dimensions, trace of input matrices, trace of output
matrix, time taken
    printf("\n  Threads: %d, Matrix Size: %d, Trace of Mat1: %d, Trace of Mat2: %d, Trace of Mat3:
%d, Time Taken: %lf\n",
           num_threads,
           matrix_size,
           trace_1,
           trace_2,
           trace_res,
           time);
```

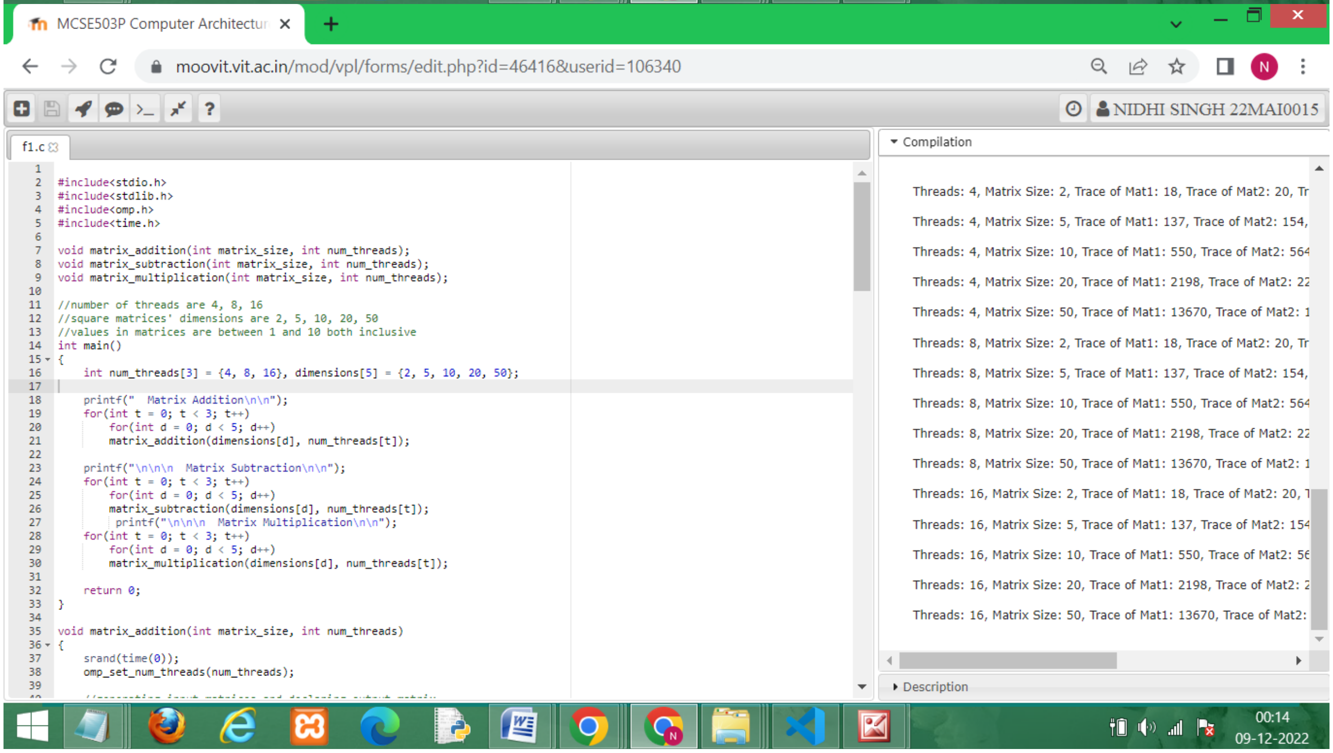


```
}
void matrix_multiplication(int matrix_size, int num_threads)
{
    srand(time(0));
    omp_set_num_threads(num_threads);

    //generating input matrices and declaring output matrix
    int mat1[matrix_size][matrix_size], mat2[matrix_size][matrix_size],
res[matrix_size][matrix_size];
    int trace_1 = 0, trace_2 = 0, trace_res = 0;

    for(int i = 0; i < matrix_size; i++)
    {
        for(int j = 0; j < matrix_size; j++)
        {
            mat1[i][j] = (rand() % 10) + 1;
            trace_1 += mat1[i][j];
            mat2[i][j] = (rand() % 10) + 1;
            trace_2 += mat2[i][j];
        }
    }
    double time = omp_get_wtime();
    //compute
    #pragma omp parallel for
    for(int i = 0; i < matrix_size; i++)
    {
        #pragma omp parallel for shared(trace_res)
        for(int j = 0; j < matrix_size; j++)
        {
            res[i][j] = 0;
            #pragma omp parallel for shared(res)
            for(int k = 0; k < matrix_size; k++)
            {
                res[i][j] += mat1[i][k] * mat2[k][j];
            }
            if(i == j) trace_res += res[i][j];
        }
    }
 time = omp_get_wtime() - time;
    //print the following: thread count, matrix dimensions, trace of input matrices, trace of output
matrix, time taken
    printf("\n  Threads: %d, Matrix Size: %d, Trace of Mat1: %d, Trace of Mat2: %d, Trace of Mat3:
%d, Time Taken: %lf\n",
           num_threads,
           matrix_size,
           trace_1,
           trace_2,
           trace_res,
           time);
}
```


MCSE503P Computer Architectur
moovit.vit.ac.in/mod/vpl/forms/edit.php?id=46416&userid=106340
NIDHI SINGH 22MAI0015
f1.c
Compilation
Description
00:14
09-12-2022

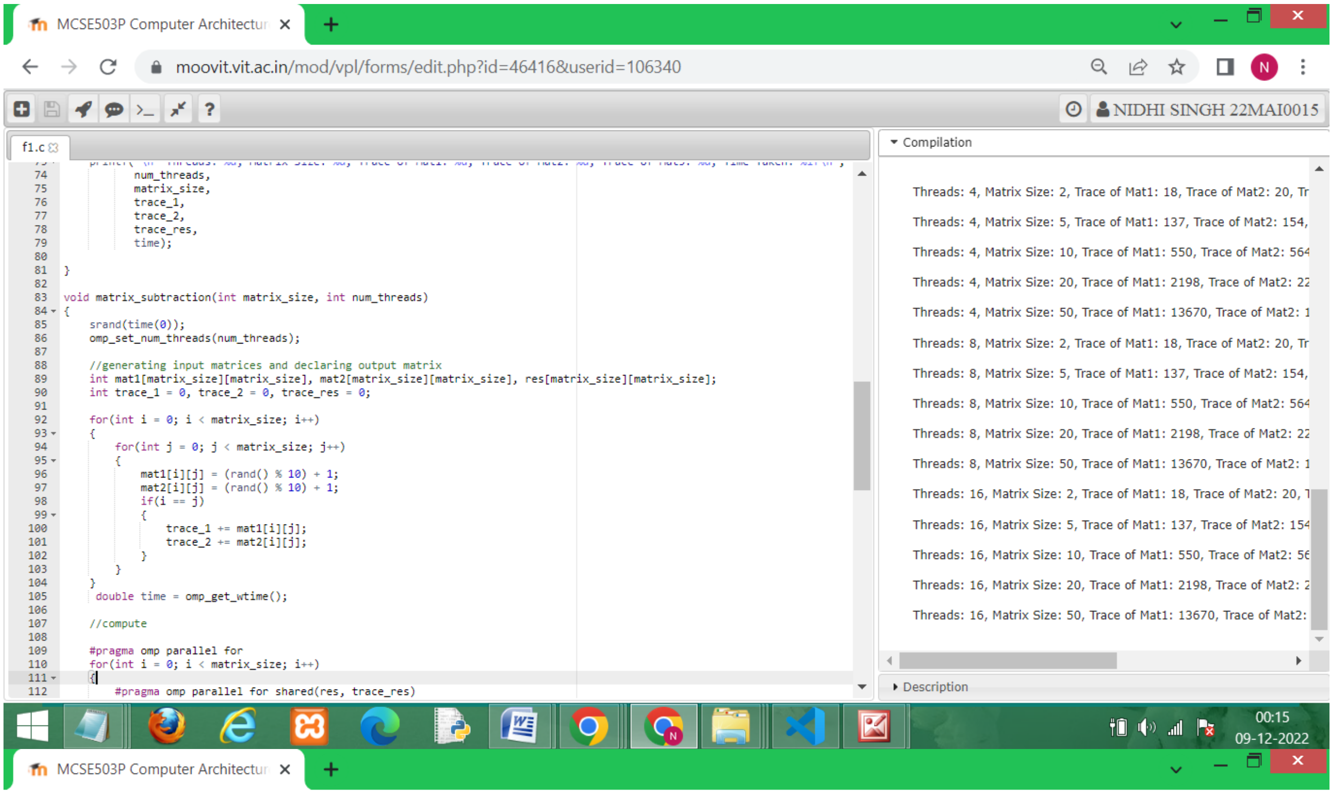
```
    omp_set_num_threads(num_threads);

    //generating input matrices and declaring output matrix
    int mat1[matrix_size][matrix_size], mat2[matrix_size][matrix_size], res[matrix_size][matrix_size];
    int trace_1 = 0, trace_2 = 0, trace_res = 0;
    for(int i = 0; i < matrix_size; i++)
    {
        for(int j = 0; j < matrix_size; j++)
        {
            mat1[i][j] = (rand() % 10) + 1;
            mat2[i][j] = (rand() % 10) + 1;
            if(i == j)
            {
                trace_1 += mat1[i][j];
                trace_2 += mat2[i][j];
            }
        }
    }
    double time = omp_get_wtime();

    //compute

    #pragma omp parallel for
    for(int i = 0; i < matrix_size; i++)
    {
        #pragma omp parallel for shared(res, trace_res)
        for(int j = 0; j < matrix_size; j++)
        {
            res[i][j] = mat1[i][j] + mat2[i][j];
            if(i == j)trace_res += res[i][j];
        }
    }

    time = omp_get_wtime() - time;
//print the following: thread count, matrix dimensions, trace of input matrices, trace of output matrix, time taken
    printf("\n  Threads: %d, Matrix Size: %d, Trace of Mat1: %d, Trace of Mat2: %d, Trace of Mat3: %d, Time Taken: %lf\n",
           num_threads,
           matrix_size,
           trace_1,
```

```
Threads: 4, Matrix Size: 2, Trace of Mat1: 18, Trace of Mat2: 20, Tr

Threads: 4, Matrix Size: 5, Trace of Mat1: 137, Trace of Mat2: 154,

Threads: 4, Matrix Size: 10, Trace of Mat1: 550, Trace of Mat2: 564

Threads: 4, Matrix Size: 20, Trace of Mat1: 2198, Trace of Mat2: 22

Threads: 4, Matrix Size: 50, Trace of Mat1: 13670, Trace of Mat2: 1

Threads: 8, Matrix Size: 2, Trace of Mat1: 18, Trace of Mat2: 20, Tr

Threads: 8, Matrix Size: 5, Trace of Mat1: 137, Trace of Mat2: 154,

Threads: 8, Matrix Size: 10, Trace of Mat1: 550, Trace of Mat2: 564

Threads: 8, Matrix Size: 20, Trace of Mat1: 2198, Trace of Mat2: 22

Threads: 8, Matrix Size: 50, Trace of Mat1: 13670, Trace of Mat2: 1

Threads: 16, Matrix Size: 2, Trace of Mat1: 18, Trace of Mat2: 20, T

Threads: 16, Matrix Size: 5, Trace of Mat1: 137, Trace of Mat2: 154

Threads: 16, Matrix Size: 10, Trace of Mat1: 550, Trace of Mat2: 56

Threads: 16, Matrix Size: 20, Trace of Mat1: 2198, Trace of Mat2: 2

Threads: 16, Matrix Size: 50, Trace of Mat1: 13670, Trace of Mat2:
```


MCSE503P Computer Architectur
moovit.vit.ac.in/mod/vpl/forms/edit.php?id=46416&userid=106340
NIDHI SINGH 22MAI0015
f1.c
Compilation
Description
00:14
09-12-2022

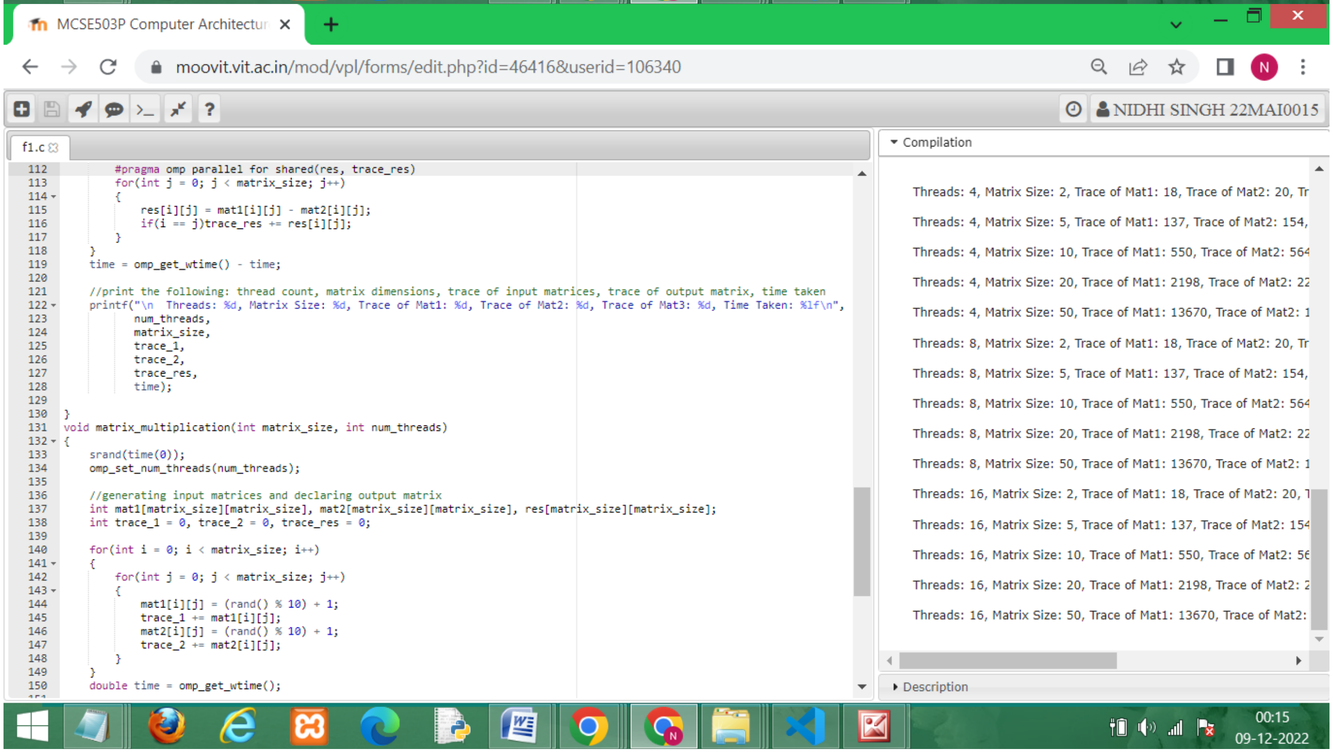
```

#include<stdio.h>
#include<stdlib.h>
#include<omp.h>
#include<time.h>

void matrix_addition(int matrix_size, int num_threads);
void matrix_subtraction(int matrix_size, int num_threads);
void matrix_multiplication(int matrix_size, int num_threads);

//number of threads are 4, 8, 16
//square matrices' dimensions are 2, 5, 10, 20, 50
//values in matrices are between 1 and 10 both inclusive
int main()
{
    int num_threads[3] = {4, 8, 16}, dimensions[5] = {2, 5, 10, 20, 50};

    printf("  Matrix Addition\n\n");
    for(int t = 0; t < 3; t++)
        for(int d = 0; d < 5; d++)
        matrix_addition(dimensions[d], num_threads[t]);

    printf("\n\n\n  Matrix Subtraction\n\n");
    for(int t = 0; t < 3; t++)
        for(int d = 0; d < 5; d++)
        matrix_subtraction(dimensions[d], num_threads[t]);
         printf("\n\n\n  Matrix Multiplication\n\n");
    for(int t = 0; t < 3; t++)
        for(int d = 0; d < 5; d++)
        matrix_multiplication(dimensions[d], num_threads[t]);

    return 0;
}

void matrix_addition(int matrix_size, int num_threads)
{
    srand(time(0));
    omp_set_num_threads(num_threads);

```

```
Threads: 4, Matrix Size: 2, Trace of Mat1: 18, Trace of Mat2: 20, Tr

Threads: 4, Matrix Size: 5, Trace of Mat1: 137, Trace of Mat2: 154,

Threads: 4, Matrix Size: 10, Trace of Mat1: 550, Trace of Mat2: 564

Threads: 4, Matrix Size: 20, Trace of Mat1: 2198, Trace of Mat2: 22

Threads: 4, Matrix Size: 50, Trace of Mat1: 13670, Trace of Mat2: 1

Threads: 8, Matrix Size: 2, Trace of Mat1: 18, Trace of Mat2: 20, Tr

Threads: 8, Matrix Size: 5, Trace of Mat1: 137, Trace of Mat2: 154,

Threads: 8, Matrix Size: 10, Trace of Mat1: 550, Trace of Mat2: 564

Threads: 8, Matrix Size: 20, Trace of Mat1: 2198, Trace of Mat2: 22

Threads: 8, Matrix Size: 50, Trace of Mat1: 13670, Trace of Mat2: 1

Threads: 16, Matrix Size: 2, Trace of Mat1: 18, Trace of Mat2: 20, T

Threads: 16, Matrix Size: 5, Trace of Mat1: 137, Trace of Mat2: 154

Threads: 16, Matrix Size: 10, Trace of Mat1: 550, Trace of Mat2: 56

Threads: 16, Matrix Size: 20, Trace of Mat1: 2198, Trace of Mat2: 2

Threads: 16, Matrix Size: 50, Trace of Mat1: 13670, Trace of Mat2:
```


MCSE503P Computer Architectur
moovit.vit.ac.in/mod/vpl/forms/edit.php?id=46416&userid=106340
NIDHI SINGH 22MAI0015
f1.c
Compilation
Description
00:15
09-12-2022

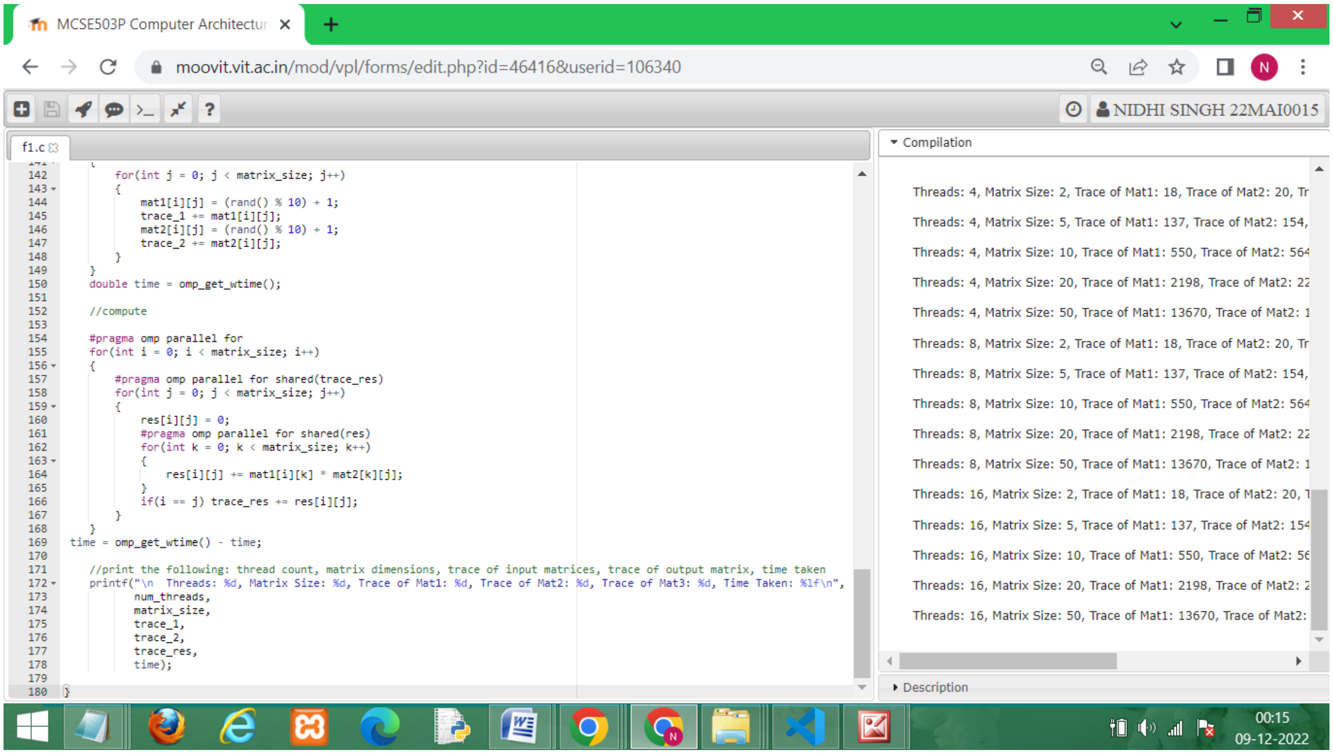
```c
            num_threads,
            matrix_size,
            trace_1,
            trace_2,
            trace_res,
            time);

}

void matrix_subtraction(int matrix_size, int num_threads)
{
    srand(time(0));
    omp_set_num_threads(num_threads);

    //generating input matrices and declaring output matrix
    int mat1[matrix_size][matrix_size], mat2[matrix_size][matrix_size], res[matrix_size][matrix_size];
    int trace_1 = 0, trace_2 = 0, trace_res = 0;

    for(int i = 0; i < matrix_size; i++)
    {
        for(int j = 0; j < matrix_size; j++)
        {
            mat1[i][j] = (rand() % 10) + 1;
            mat2[i][j] = (rand() % 10) + 1;
            if(i == j)
            {
                trace_1 += mat1[i][j];
                trace_2 += mat2[i][j];
            }
        }
    }
     double time = omp_get_wtime();

    //compute

    #pragma omp parallel for
    for(int i = 0; i < matrix_size; i++)
    {
        #pragma omp parallel for shared(res, trace_res)
```

```
Threads: 4, Matrix Size: 2, Trace of Mat1: 18, Trace of Mat2: 20, Tr
Threads: 4, Matrix Size: 5, Trace of Mat1: 137, Trace of Mat2: 154,
Threads: 4, Matrix Size: 10, Trace of Mat1: 550, Trace of Mat2: 564
Threads: 4, Matrix Size: 20, Trace of Mat1: 2198, Trace of Mat2: 22
Threads: 4, Matrix Size: 50, Trace of Mat1: 13670, Trace of Mat2: 1
Threads: 8, Matrix Size: 2, Trace of Mat1: 18, Trace of Mat2: 20, Tr
Threads: 8, Matrix Size: 5, Trace of Mat1: 137, Trace of Mat2: 154,
Threads: 8, Matrix Size: 10, Trace of Mat1: 550, Trace of Mat2: 564
Threads: 8, Matrix Size: 20, Trace of Mat1: 2198, Trace of Mat2: 22
Threads: 8, Matrix Size: 50, Trace of Mat1: 13670, Trace of Mat2: 1
Threads: 16, Matrix Size: 2, Trace of Mat1: 18, Trace of Mat2: 20, T
Threads: 16, Matrix Size: 5, Trace of Mat1: 137, Trace of Mat2: 154
Threads: 16, Matrix Size: 10, Trace of Mat1: 550, Trace of Mat2: 56
Threads: 16, Matrix Size: 20, Trace of Mat1: 2198, Trace of Mat2: 2
Threads: 16, Matrix Size: 50, Trace of Mat1: 13670, Trace of Mat2:
```


MCSE503P Computer Architectur
moovit.vit.ac.in/mod/vpl/forms/edit.php?id=46416&userid=106340
NIDHI SINGH 22MAI0015
f1.c
Compilation
Description
00:15
09-12-2022

```c
        #pragma omp parallel for shared(res, trace_res)
        for(int j = 0; j < matrix_size; j++)
        {
            res[i][j] = mat1[i][j] - mat2[i][j];
            if(i == j)trace_res += res[i][j];
        }
    }
    time = omp_get_wtime() - time;

    //print the following: thread count, matrix dimensions, trace of input matrices, trace of output matrix, time taken
    printf("\n  Threads: %d, Matrix Size: %d, Trace of Mat1: %d, Trace of Mat2: %d, Trace of Mat3: %d, Time Taken: %lf\n",
            num_threads,
            matrix_size,
            trace_1,
            trace_2,
            trace_res,
            time);

}
void matrix_multiplication(int matrix_size, int num_threads)
{
    srand(time(0));
    omp_set_num_threads(num_threads);

    //generating input matrices and declaring output matrix
    int mat1[matrix_size][matrix_size], mat2[matrix_size][matrix_size], res[matrix_size][matrix_size];
    int trace_1 = 0, trace_2 = 0, trace_res = 0;

    for(int i = 0; i < matrix_size; i++)
    {
        for(int j = 0; j < matrix_size; j++)
        {
            mat1[i][j] = (rand() % 10) + 1;
            trace_1 += mat1[i][j];
            mat2[i][j] = (rand() % 10) + 1;
            trace_2 += mat2[i][j];
        }
    }
    double time = omp_get_wtime();
```

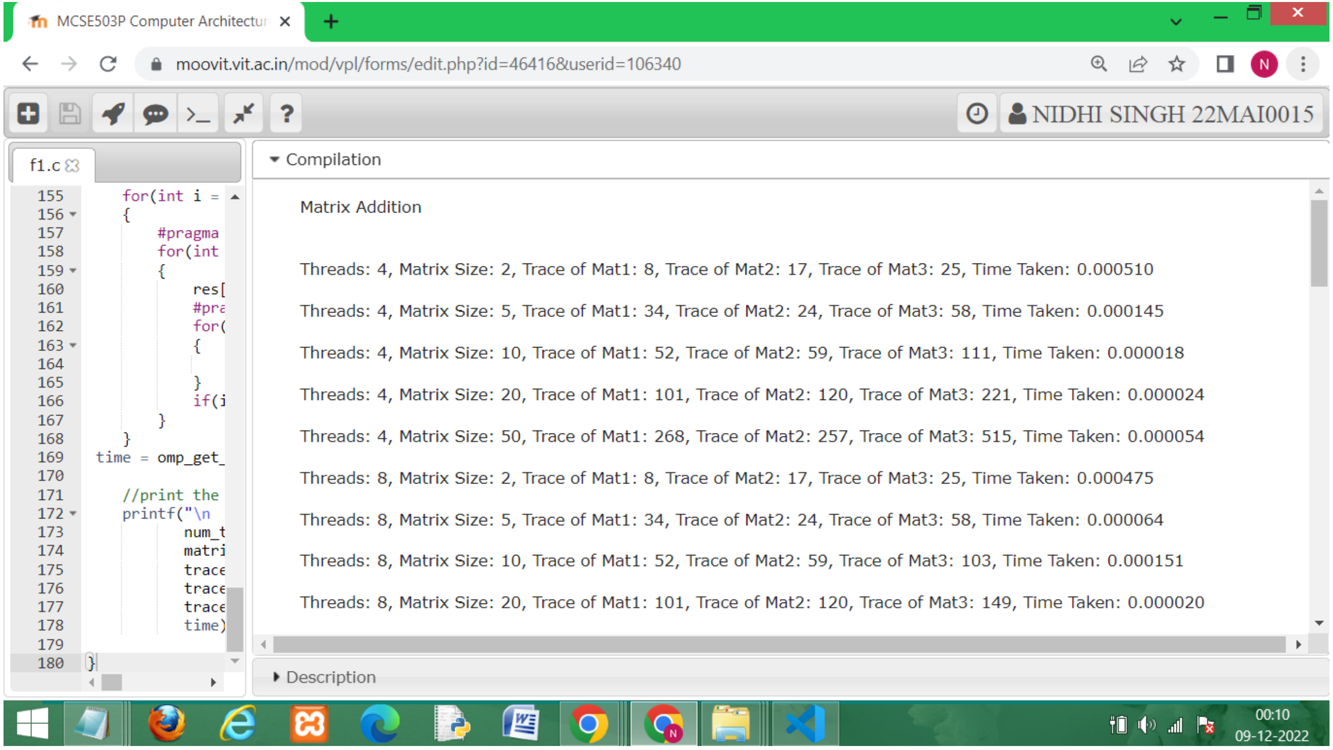
```
Threads: 4, Matrix Size: 2, Trace of Mat1: 18, Trace of Mat2: 20, Tr
Threads: 4, Matrix Size: 5, Trace of Mat1: 137, Trace of Mat2: 154,
Threads: 4, Matrix Size: 10, Trace of Mat1: 550, Trace of Mat2: 564
Threads: 4, Matrix Size: 20, Trace of Mat1: 2198, Trace of Mat2: 22
Threads: 4, Matrix Size: 50, Trace of Mat1: 13670, Trace of Mat2: 1
Threads: 8, Matrix Size: 2, Trace of Mat1: 18, Trace of Mat2: 20, Tr
Threads: 8, Matrix Size: 5, Trace of Mat1: 137, Trace of Mat2: 154,
Threads: 8, Matrix Size: 10, Trace of Mat1: 550, Trace of Mat2: 564
Threads: 8, Matrix Size: 20, Trace of Mat1: 2198, Trace of Mat2: 22
Threads: 8, Matrix Size: 50, Trace of Mat1: 13670, Trace of Mat2: 1
Threads: 16, Matrix Size: 2, Trace of Mat1: 18, Trace of Mat2: 20, T
Threads: 16, Matrix Size: 5, Trace of Mat1: 137, Trace of Mat2: 154
Threads: 16, Matrix Size: 10, Trace of Mat1: 550, Trace of Mat2: 56
Threads: 16, Matrix Size: 20, Trace of Mat1: 2198, Trace of Mat2: 2
Threads: 16, Matrix Size: 50, Trace of Mat1: 13670, Trace of Mat2:
```


MCSE503P Computer Architectur
moovit.vit.ac.in/mod/vpl/forms/edit.php?id=46416&userid=106340
NIDHI SINGH 22MAI0015
f1.c
Compilation
Description
00:15
09-12-2022

```c
        for(int j = 0; j < matrix_size; j++)
        {
            mat1[i][j] = (rand() % 10) + 1;
            trace_1 += mat1[i][j];
            mat2[i][j] = (rand() % 10) + 1;
            trace_2 += mat2[i][j];
        }
    }
    double time = omp_get_wtime();

    //compute

    #pragma omp parallel for
    for(int i = 0; i < matrix_size; i++)
    {
        #pragma omp parallel for shared(trace_res)
        for(int j = 0; j < matrix_size; j++)
        {
            res[i][j] = 0;
            #pragma omp parallel for shared(res)
            for(int k = 0; k < matrix_size; k++)
            {
                res[i][j] += mat1[i][k] * mat2[k][j];
            }
            if(i == j) trace_res += res[i][j];
        }
    }
  time = omp_get_wtime() - time;

    //print the following: thread count, matrix dimensions, trace of input matrices, trace of output matrix, time taken
    printf("\n  Threads: %d, Matrix Size: %d, Trace of Mat1: %d, Trace of Mat2: %d, Trace of Mat3: %d, Time Taken: %lf\n",
            num_threads,
            matrix_size,
            trace_1,
            trace_2,
            trace_res,
            time);

}
```

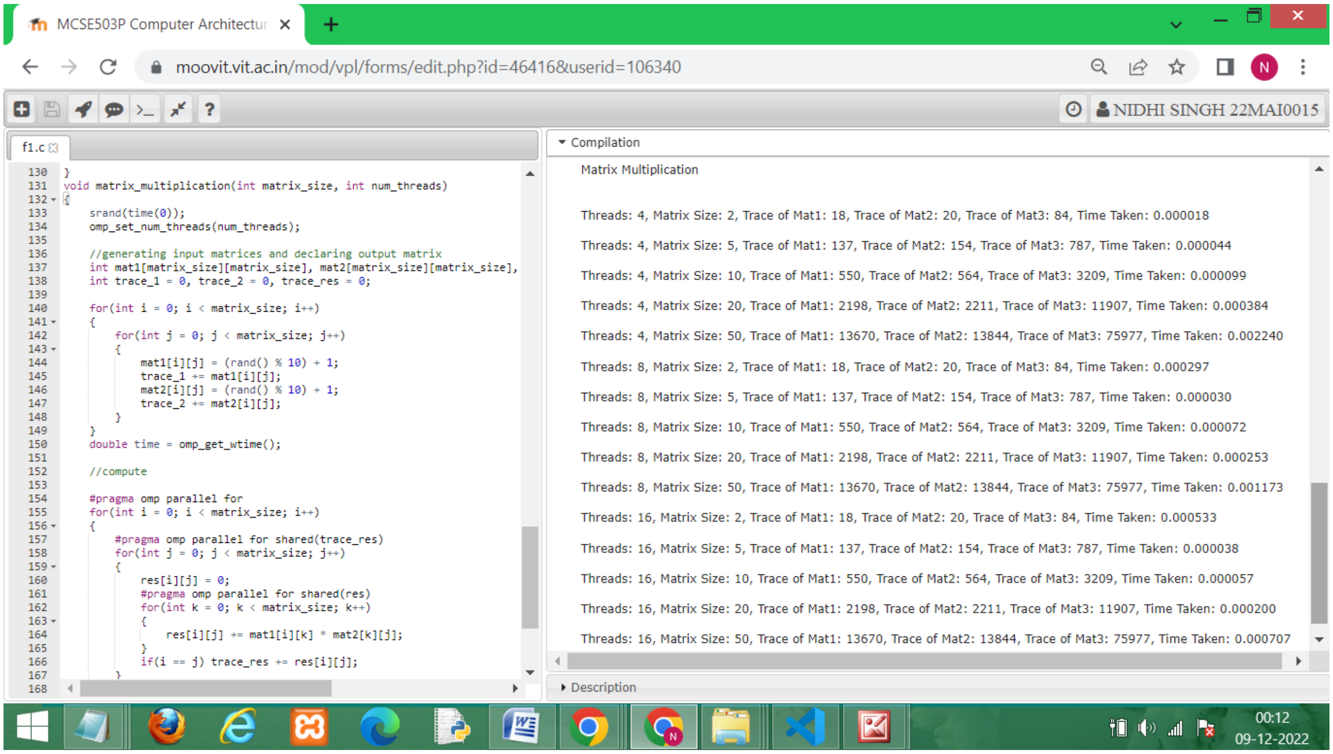
```
Threads: 4, Matrix Size: 2, Trace of Mat1: 18, Trace of Mat2: 20, Tr

Threads: 4, Matrix Size: 5, Trace of Mat1: 137, Trace of Mat2: 154,

Threads: 4, Matrix Size: 10, Trace of Mat1: 550, Trace of Mat2: 564

Threads: 4, Matrix Size: 20, Trace of Mat1: 2198, Trace of Mat2: 22

Threads: 4, Matrix Size: 50, Trace of Mat1: 13670, Trace of Mat2: 1

Threads: 8, Matrix Size: 2, Trace of Mat1: 18, Trace of Mat2: 20, Tr

Threads: 8, Matrix Size: 5, Trace of Mat1: 137, Trace of Mat2: 154,

Threads: 8, Matrix Size: 10, Trace of Mat1: 550, Trace of Mat2: 564

Threads: 8, Matrix Size: 20, Trace of Mat1: 2198, Trace of Mat2: 22

Threads: 8, Matrix Size: 50, Trace of Mat1: 13670, Trace of Mat2: 1

Threads: 16, Matrix Size: 2, Trace of Mat1: 18, Trace of Mat2: 20, T

Threads: 16, Matrix Size: 5, Trace of Mat1: 137, Trace of Mat2: 154

Threads: 16, Matrix Size: 10, Trace of Mat1: 550, Trace of Mat2: 56

Threads: 16, Matrix Size: 20, Trace of Mat1: 2198, Trace of Mat2: 2

Threads: 16, Matrix Size: 50, Trace of Mat1: 13670, Trace of Mat2:
```

Output:-


MCSE503P Computer Architectur
moovit.vit.ac.in/mod/vpl/forms/edit.php?id=46416&userid=106340
NIDHI SINGH 22MAI0015
f1.c
Compilation
Description
00:10
09-12-2022

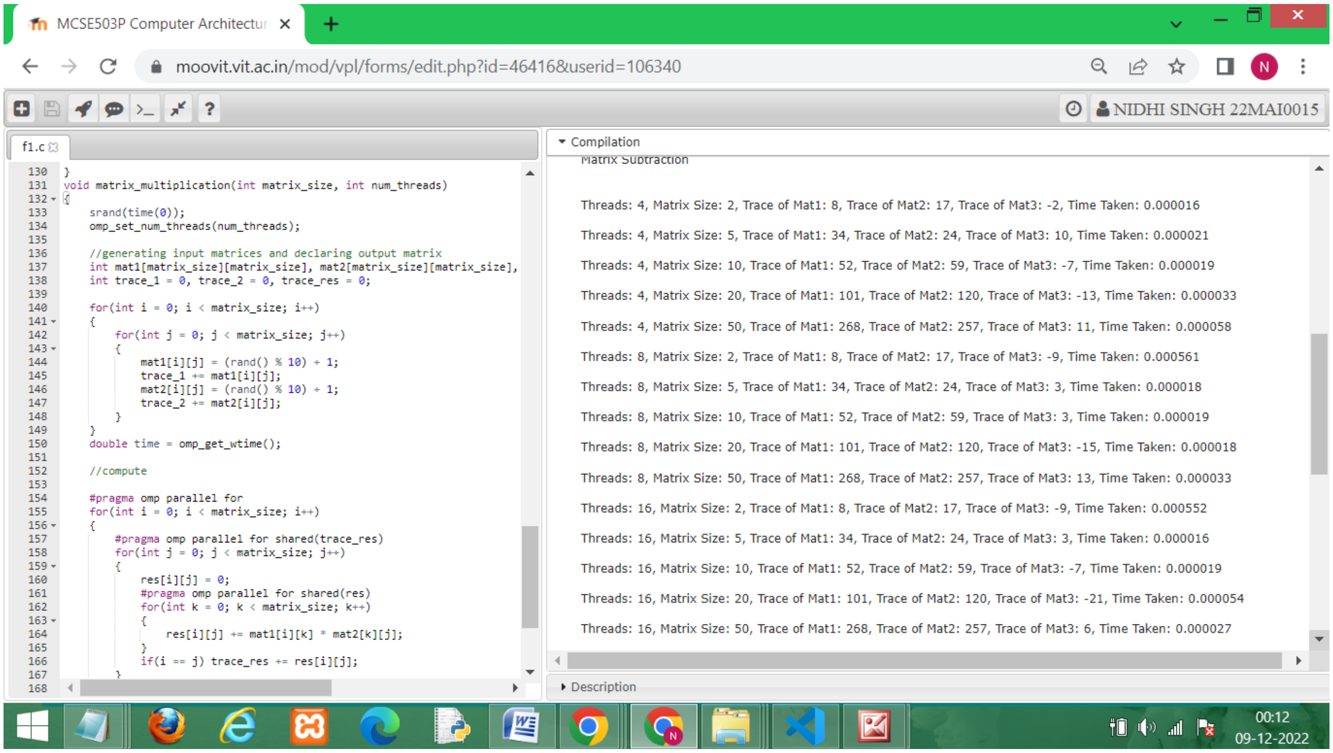
```c
    for(int i =
    {
        #pragma
        for(int
        {
            res[
            #pra
            for(
            {

            }
            if(i
        }
    }
time = omp_get_

    //print the
    printf("\n
          num_t
          matri
          trace
          trace
          trace
          time)

}
```

```
Matrix Addition


Threads: 4, Matrix Size: 2, Trace of Mat1: 8, Trace of Mat2: 17, Trace of Mat3: 25, Time Taken: 0.000510

Threads: 4, Matrix Size: 5, Trace of Mat1: 34, Trace of Mat2: 24, Trace of Mat3: 58, Time Taken: 0.000145

Threads: 4, Matrix Size: 10, Trace of Mat1: 52, Trace of Mat2: 59, Trace of Mat3: 111, Time Taken: 0.000018

Threads: 4, Matrix Size: 20, Trace of Mat1: 101, Trace of Mat2: 120, Trace of Mat3: 221, Time Taken: 0.000024

Threads: 4, Matrix Size: 50, Trace of Mat1: 268, Trace of Mat2: 257, Trace of Mat3: 515, Time Taken: 0.000054

Threads: 8, Matrix Size: 2, Trace of Mat1: 8, Trace of Mat2: 17, Trace of Mat3: 25, Time Taken: 0.000475

Threads: 8, Matrix Size: 5, Trace of Mat1: 34, Trace of Mat2: 24, Trace of Mat3: 58, Time Taken: 0.000064

Threads: 8, Matrix Size: 10, Trace of Mat1: 52, Trace of Mat2: 59, Trace of Mat3: 103, Time Taken: 0.000151

Threads: 8, Matrix Size: 20, Trace of Mat1: 101, Trace of Mat2: 120, Trace of Mat3: 149, Time Taken: 0.000020
```


MCSE503P Computer Architectur
moovit.vit.ac.in/mod/vpl/forms/edit.php?id=46416&userid=106340
NIDHI SINGH 22MAI0015
f1.c
Compilation
Description
00:12
09-12-2022

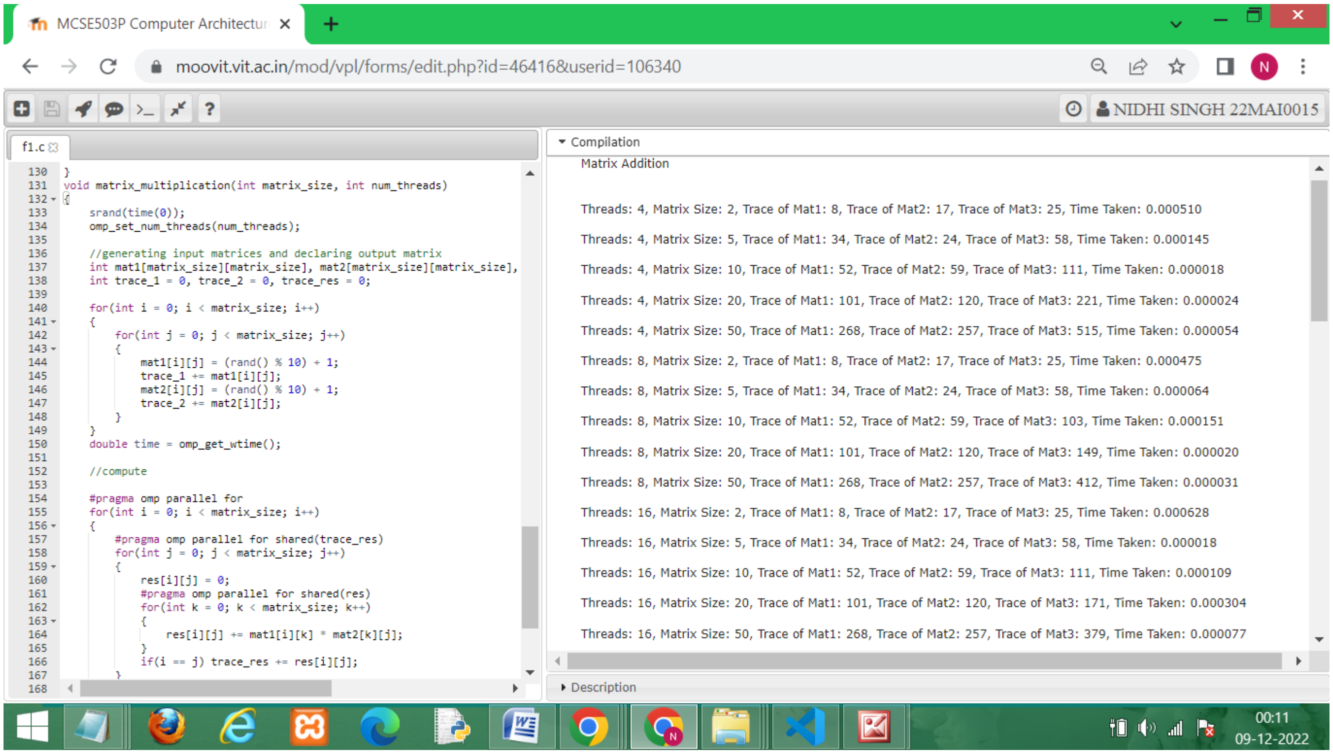
```c
}
void matrix_multiplication(int matrix_size, int num_threads)
{
    srand(time(0));
    omp_set_num_threads(num_threads);

    //generating input matrices and declaring output matrix
    int mat1[matrix_size][matrix_size], mat2[matrix_size][matrix_size],
    int trace_1 = 0, trace_2 = 0, trace_res = 0;

    for(int i = 0; i < matrix_size; i++)
    {
        for(int j = 0; j < matrix_size; j++)
        {
            mat1[i][j] = (rand() % 10) + 1;
            trace_1 += mat1[i][j];
            mat2[i][j] = (rand() % 10) + 1;
            trace_2 += mat2[i][j];
        }
    }
    double time = omp_get_wtime();

    //compute

    #pragma omp parallel for
    for(int i = 0; i < matrix_size; i++)
    {
        #pragma omp parallel for shared(trace_res)
        for(int j = 0; j < matrix_size; j++)
        {
            res[i][j] = 0;
            #pragma omp parallel for shared(res)
            for(int k = 0; k < matrix_size; k++)
            {
                res[i][j] += mat1[i][k] * mat2[k][j];
            }
            if(i == j) trace_res += res[i][j];
        }
```

```
Matrix Multiplication

Threads: 4, Matrix Size: 2, Trace of Mat1: 18, Trace of Mat2: 20, Trace of Mat3: 84, Time Taken: 0.000018

Threads: 4, Matrix Size: 5, Trace of Mat1: 137, Trace of Mat2: 154, Trace of Mat3: 787, Time Taken: 0.000044

Threads: 4, Matrix Size: 10, Trace of Mat1: 550, Trace of Mat2: 564, Trace of Mat3: 3209, Time Taken: 0.000099

Threads: 4, Matrix Size: 20, Trace of Mat1: 2198, Trace of Mat2: 2211, Trace of Mat3: 11907, Time Taken: 0.000384

Threads: 4, Matrix Size: 50, Trace of Mat1: 13670, Trace of Mat2: 13844, Trace of Mat3: 75977, Time Taken: 0.002240

Threads: 8, Matrix Size: 2, Trace of Mat1: 18, Trace of Mat2: 20, Trace of Mat3: 84, Time Taken: 0.000297

Threads: 8, Matrix Size: 5, Trace of Mat1: 137, Trace of Mat2: 154, Trace of Mat3: 787, Time Taken: 0.000030

Threads: 8, Matrix Size: 10, Trace of Mat1: 550, Trace of Mat2: 564, Trace of Mat3: 3209, Time Taken: 0.000072

Threads: 8, Matrix Size: 20, Trace of Mat1: 2198, Trace of Mat2: 2211, Trace of Mat3: 11907, Time Taken: 0.000253

Threads: 8, Matrix Size: 50, Trace of Mat1: 13670, Trace of Mat2: 13844, Trace of Mat3: 75977, Time Taken: 0.001173

Threads: 16, Matrix Size: 2, Trace of Mat1: 18, Trace of Mat2: 20, Trace of Mat3: 84, Time Taken: 0.000533

Threads: 16, Matrix Size: 5, Trace of Mat1: 137, Trace of Mat2: 154, Trace of Mat3: 787, Time Taken: 0.000038

Threads: 16, Matrix Size: 10, Trace of Mat1: 550, Trace of Mat2: 564, Trace of Mat3: 3209, Time Taken: 0.000057

Threads: 16, Matrix Size: 20, Trace of Mat1: 2198, Trace of Mat2: 2211, Trace of Mat3: 11907, Time Taken: 0.000200

Threads: 16, Matrix Size: 50, Trace of Mat1: 13670, Trace of Mat2: 13844, Trace of Mat3: 75977, Time Taken: 0.000707
```


MCSE503P Computer Architectur
moovit.vit.ac.in/mod/vpl/forms/edit.php?id=46416&userid=106340
NIDHI SINGH 22MAI0015
f1.c
Compilation
Description
00:12
09-12-2022

```c
}
void matrix_multiplication(int matrix_size, int num_threads)
{
    srand(time(0));
    omp_set_num_threads(num_threads);

    //generating input matrices and declaring output matrix
    int mat1[matrix_size][matrix_size], mat2[matrix_size][matrix_size],
    int trace_1 = 0, trace_2 = 0, trace_res = 0;

    for(int i = 0; i < matrix_size; i++)
    {
        for(int j = 0; j < matrix_size; j++)
        {
            mat1[i][j] = (rand() % 10) + 1;
            trace_1 += mat1[i][j];
            mat2[i][j] = (rand() % 10) + 1;
            trace_2 += mat2[i][j];
        }
    }
    double time = omp_get_wtime();

    //compute

    #pragma omp parallel for
    for(int i = 0; i < matrix_size; i++)
    {
        #pragma omp parallel for shared(trace_res)
        for(int j = 0; j < matrix_size; j++)
        {
            res[i][j] = 0;
            #pragma omp parallel for shared(res)
            for(int k = 0; k < matrix_size; k++)
            {
                res[i][j] += mat1[i][k] * mat2[k][j];
            }
            if(i == j) trace_res += res[i][j];
        }
```

```
Matrix Subtraction

Threads: 4, Matrix Size: 2, Trace of Mat1: 8, Trace of Mat2: 17, Trace of Mat3: -2, Time Taken: 0.000016

Threads: 4, Matrix Size: 5, Trace of Mat1: 34, Trace of Mat2: 24, Trace of Mat3: 10, Time Taken: 0.000021

Threads: 4, Matrix Size: 10, Trace of Mat1: 52, Trace of Mat2: 59, Trace of Mat3: -7, Time Taken: 0.000019

Threads: 4, Matrix Size: 20, Trace of Mat1: 101, Trace of Mat2: 120, Trace of Mat3: -13, Time Taken: 0.000033

Threads: 4, Matrix Size: 50, Trace of Mat1: 268, Trace of Mat2: 257, Trace of Mat3: 11, Time Taken: 0.000058

Threads: 8, Matrix Size: 2, Trace of Mat1: 8, Trace of Mat2: 17, Trace of Mat3: -9, Time Taken: 0.000561

Threads: 8, Matrix Size: 5, Trace of Mat1: 34, Trace of Mat2: 24, Trace of Mat3: 3, Time Taken: 0.000018

Threads: 8, Matrix Size: 10, Trace of Mat1: 52, Trace of Mat2: 59, Trace of Mat3: 3, Time Taken: 0.000019

Threads: 8, Matrix Size: 20, Trace of Mat1: 101, Trace of Mat2: 120, Trace of Mat3: -15, Time Taken: 0.000018

Threads: 8, Matrix Size: 50, Trace of Mat1: 268, Trace of Mat2: 257, Trace of Mat3: 13, Time Taken: 0.000033

Threads: 16, Matrix Size: 2, Trace of Mat1: 8, Trace of Mat2: 17, Trace of Mat3: -9, Time Taken: 0.000552

Threads: 16, Matrix Size: 5, Trace of Mat1: 34, Trace of Mat2: 24, Trace of Mat3: 3, Time Taken: 0.000016

Threads: 16, Matrix Size: 10, Trace of Mat1: 52, Trace of Mat2: 59, Trace of Mat3: -7, Time Taken: 0.000019

Threads: 16, Matrix Size: 20, Trace of Mat1: 101, Trace of Mat2: 120, Trace of Mat3: -21, Time Taken: 0.000054

Threads: 16, Matrix Size: 50, Trace of Mat1: 268, Trace of Mat2: 257, Trace of Mat3: 6, Time Taken: 0.000027
```


MCSE503P Computer Architectur
moovit.vit.ac.in/mod/vpl/forms/edit.php?id=46416&userid=106340
NIDHI SINGH 22MAI0015
f1.c
Compilation
Description
00:11
09-12-2022

```c
}
void matrix_multiplication(int matrix_size, int num_threads)
{
    srand(time(0));
    omp_set_num_threads(num_threads);

    //generating input matrices and declaring output matrix
    int mat1[matrix_size][matrix_size], mat2[matrix_size][matrix_size],
    int trace_1 = 0, trace_2 = 0, trace_res = 0;

    for(int i = 0; i < matrix_size; i++)
    {
        for(int j = 0; j < matrix_size; j++)
        {
            mat1[i][j] = (rand() % 10) + 1;
            trace_1 += mat1[i][j];
            mat2[i][j] = (rand() % 10) + 1;
            trace_2 += mat2[i][j];
        }
    }
    double time = omp_get_wtime();

    //compute

    #pragma omp parallel for
    for(int i = 0; i < matrix_size; i++)
    {
        #pragma omp parallel for shared(trace_res)
        for(int j = 0; j < matrix_size; j++)
        {
            res[i][j] = 0;
            #pragma omp parallel for shared(res)
            for(int k = 0; k < matrix_size; k++)
            {
                res[i][j] += mat1[i][k] * mat2[k][j];
            }
            if(i == j) trace_res += res[i][j];
        }
```

```
Matrix Addition

Threads: 4, Matrix Size: 2, Trace of Mat1: 8, Trace of Mat2: 17, Trace of Mat3: 25, Time Taken: 0.000510

Threads: 4, Matrix Size: 5, Trace of Mat1: 34, Trace of Mat2: 24, Trace of Mat3: 58, Time Taken: 0.000145

Threads: 4, Matrix Size: 10, Trace of Mat1: 52, Trace of Mat2: 59, Trace of Mat3: 111, Time Taken: 0.000018

Threads: 4, Matrix Size: 20, Trace of Mat1: 101, Trace of Mat2: 120, Trace of Mat3: 221, Time Taken: 0.000024

Threads: 4, Matrix Size: 50, Trace of Mat1: 268, Trace of Mat2: 257, Trace of Mat3: 515, Time Taken: 0.000054

Threads: 8, Matrix Size: 2, Trace of Mat1: 8, Trace of Mat2: 17, Trace of Mat3: 25, Time Taken: 0.000475

Threads: 8, Matrix Size: 5, Trace of Mat1: 34, Trace of Mat2: 24, Trace of Mat3: 58, Time Taken: 0.000064

Threads: 8, Matrix Size: 10, Trace of Mat1: 52, Trace of Mat2: 59, Trace of Mat3: 103, Time Taken: 0.000151

Threads: 8, Matrix Size: 20, Trace of Mat1: 101, Trace of Mat2: 120, Trace of Mat3: 149, Time Taken: 0.000020

Threads: 8, Matrix Size: 50, Trace of Mat1: 268, Trace of Mat2: 257, Trace of Mat3: 412, Time Taken: 0.000031

Threads: 16, Matrix Size: 2, Trace of Mat1: 8, Trace of Mat2: 17, Trace of Mat3: 25, Time Taken: 0.000628

Threads: 16, Matrix Size: 5, Trace of Mat1: 34, Trace of Mat2: 24, Trace of Mat3: 58, Time Taken: 0.000018

Threads: 16, Matrix Size: 10, Trace of Mat1: 52, Trace of Mat2: 59, Trace of Mat3: 111, Time Taken: 0.000109

Threads: 16, Matrix Size: 20, Trace of Mat1: 101, Trace of Mat2: 120, Trace of Mat3: 171, Time Taken: 0.000304

Threads: 16, Matrix Size: 50, Trace of Mat1: 268, Trace of Mat2: 257, Trace of Mat3: 379, Time Taken: 0.000077
```