



VIT[®]
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)

FALL – SEMESTER

Course Code: MCSE503P

Course-Title: – Computer Architecture and Organization

DIGITAL ASSIGNMENT - IV

(LAB)

Name: Nidhi Singh

Reg. No:22MAI0015

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 Architecture x +

moovit.vit.ac.in/mod/vpl/forms/edit.php?id=46416&userid=106340

NIDHI SINGH 22MAI0015

f1.c

```

38  omp_set_num_threads(num_threads);
39
40  //generating input matrices and declaring output matrix
41  int mat1[matrix_size][matrix_size], mat2[matrix_size][matrix_size];
42  int trace_1 = 0, trace_2 = 0, trace_res = 0;
43  for(int i = 0; i < matrix_size; i++)
44  {
45      for(int j = 0; j < matrix_size; j++)
46      {
47          mat1[i][j] = (rand() % 10) + 1;
48          mat2[i][j] = (rand() % 10) + 1;
49          if(i == j)
50          {
51              trace_1 += mat1[i][j];
52              trace_2 += mat2[i][j];
53          }
54      }
55  }
56  double time = omp_get_wtime();
57
58  //compute
59
60  #pragma omp parallel for
61  for(int i = 0; i < matrix_size; i++)
62  {
63      #pragma omp parallel for shared(res, trace_res)
64      for(int j = 0; j < matrix_size; j++)
65      {
66          res[i][j] = mat1[i][j] + mat2[i][j];
67          if(i == j) trace_res += res[i][j];
68      }
69  }
70
71  time = omp_get_wtime() - time;
72  //print the following: thread count, matrix dimensions, trace of input matrices, trace of output matrix, time taken
73  printf("\n Threads: %d, Matrix Size: %d, Trace of Mat1: %d, Trace of Mat2: %d, Trace of Mat3: %d, Time Taken: %lf\n",
74         num_threads,
75         matrix_size,
76         trace_1,

```

Compilation

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, 1
 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:

Description

MCSE503P Computer Architecture x +

moovit.vit.ac.in/mod/vpl/forms/edit.php?id=46416&userid=106340

NIDHI SINGH 22MAI0015

f1.c

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<omp.h>
4  #include<time.h>
5
6
7  void matrix_addition(int matrix_size, int num_threads);
8  void matrix_subtraction(int matrix_size, int num_threads);
9  void matrix_multiplication(int matrix_size, int num_threads);
10
11 //number of threads are 4, 8, 16
12 //square matrices' dimensions are 2, 5, 10, 20, 50
13 //values in matrices are between 1 and 10 both inclusive
14 int main()
15 {
16     int num_threads[3] = {4, 8, 16}, dimensions[5] = {2, 5, 10, 20, 50};
17
18     printf(" Matrix Addition\n\n");
19     for(int t = 0; t < 3; t++)
20     {
21         for(int d = 0; d < 5; d++)
22             matrix_addition(dimensions[d], num_threads[t]);
23
24     }
25     printf("\n\n Matrix Subtraction\n\n");
26     for(int t = 0; t < 3; t++)
27     {
28         for(int d = 0; d < 5; d++)
29             matrix_subtraction(dimensions[d], num_threads[t]);
30
31     }
32     printf("\n\n Matrix Multiplication\n\n");
33     for(int t = 0; t < 3; t++)
34     {
35         for(int d = 0; d < 5; d++)
36             matrix_multiplication(dimensions[d], num_threads[t]);
37
38     }
39     return 0;
40 }
41
42 void matrix_addition(int matrix_size, int num_threads)
43 {
44     srand(time(0));
45     omp_set_num_threads(num_threads);
46
47     //generating input matrices and declaring output matrix

```

Compilation

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, 1
 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:

Description

MCSE503P Computer Architecture x +

moovit.vit.ac.in/mod/vpl/forms/edit.php?id=46416&userid=106340

NIDHI SINGH 22MAI0015

f1.c

```
74 num_threads,
75 matrix_size,
76 trace_1,
77 trace_2,
78 trace_res,
79 time);
80
81 }
82
83 void matrix_subtraction(int matrix_size, int num_threads)
84 {
85     srand(time(0));
86     omp_set_num_threads(num_threads);
87
88     //generating input matrices and declaring output matrix
89     int mat1[matrix_size][matrix_size], mat2[matrix_size][matrix_size];
90     int trace_1 = 0, trace_2 = 0, trace_res = 0;
91
92     for(int i = 0; i < matrix_size; i++)
93     {
94         for(int j = 0; j < matrix_size; j++)
95         {
96             mat1[i][j] = (rand() % 10) + 1;
97             mat2[i][j] = (rand() % 10) + 1;
98             if(i == j)
99             {
100                 trace_1 += mat1[i][j];
101                 trace_2 += mat2[i][j];
102             }
103         }
104     }
105     double time = omp_get_wtime();
106
107     //compute
108
109     #pragma omp parallel for
110     for(int i = 0; i < matrix_size; i++)
111     {
112         #pragma omp parallel for shared(res, trace_res)
```

Compilation

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, 1

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:

Description

00:15 09-12-2022

MCSE503P Computer Architecture x +

moovit.vit.ac.in/mod/vpl/forms/edit.php?id=46416&userid=106340

NIDHI SINGH 22MAI0015

f1.c

```
112 #pragma omp parallel for shared(res, trace_res)
113 for(int j = 0; j < matrix_size; j++)
114 {
115     res[i][j] = mat1[i][j] - mat2[i][j];
116     if(i == j) trace_res += res[i][j];
117 }
118 }
119 time = omp_get_wtime() - time;
120
121 //print the following: thread count, matrix dimensions, trace of input matrices, trace of output matrix, time taken
122 printf("\n Threads: %d, Matrix Size: %d, Trace of Mat1: %d, Trace of Mat2: %d, Trace of Mat3: %d, Time Taken: %lf\n",
123 num_threads,
124 matrix_size,
125 trace_1,
126 trace_2,
127 trace_res,
128 time);
129
130 }
131 void matrix_multiplication(int matrix_size, int num_threads)
132 {
133     srand(time(0));
134     omp_set_num_threads(num_threads);
135
136     //generating input matrices and declaring output matrix
137     int mat1[matrix_size][matrix_size], mat2[matrix_size][matrix_size], res[matrix_size][matrix_size];
138     int trace_1 = 0, trace_2 = 0, trace_res = 0;
139
140     for(int i = 0; i < matrix_size; i++)
141     {
142         for(int j = 0; j < matrix_size; j++)
143         {
144             mat1[i][j] = (rand() % 10) + 1;
145             trace_1 += mat1[i][j];
146             mat2[i][j] = (rand() % 10) + 1;
147             trace_2 += mat2[i][j];
148         }
149     }
150     double time = omp_get_wtime();
151     ++
```

Compilation

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, 1

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:

Description

00:15 09-12-2022

```
f1.c
142 for(int j = 0; j < matrix_size; j++)
143 {
144     mat1[i][j] = (rand() % 10) + 1;
145     trace_1 += mat1[i][j];
146     mat2[i][j] = (rand() % 10) + 1;
147     trace_2 += mat2[i][j];
148 }
149 }
150 double time = omp_get_wtime();
151 //compute
152
153 #pragma omp parallel for
154 for(int i = 0; i < matrix_size; i++)
155 {
156     #pragma omp parallel for shared(trace_res)
157     for(int j = 0; j < matrix_size; j++)
158     {
159         res[i][j] = 0;
160         #pragma omp parallel for shared(res)
161         for(int k = 0; k < matrix_size; k++)
162         {
163             res[i][j] += mat1[i][k] * mat2[k][j];
164             if(i == j) trace_res += res[i][j];
165         }
166     }
167 }
168 time = omp_get_wtime() - time;
169
170 //print the following: thread count, matrix dimensions, trace of input matrices, trace of output matrix, time taken
171 printf("\n Threads: %d, Matrix Size: %d, Trace of Mat1: %d, Trace of Mat2: %d, Trace of Mat3: %d, Time Taken: %lf\n",
172        num_threads,
173        matrix_size,
174        trace_1,
175        trace_2,
176        trace_res,
177        time);
178
179 }
180 }
```

Compilation

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, 1

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:

Description

Output:-

```
f1.c
155 for(int i =
156 {
157     #pragma
158     for(int
159     {
160         res[
161         #pre
162         for(
163         {
164             }
165         }
166         if(i
167     }
168 }
169 time = omp_get_
170
171 //print the
172 printf("\n
173 num_t
174 matri
175 trace
176 trace
177 trace
178 time)
179
180 }
```

Compilation

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

Description

MCSE503P Computer Architecture x +

moovit.vit.ac.in/mod/vpl/forms/edit.php?id=46416&userid=106340

NIDHI SINGH 22MAI0015

f1.c

```

130 }
131 void matrix_multiplication(int matrix_size, int num_threads)
132 {
133     srand(time(0));
134     omp_set_num_threads(num_threads);
135
136     //generating input matrices and declaring output matrix
137     int mat1[matrix_size][matrix_size], mat2[matrix_size][matrix_size],
138     int trace_1 = 0, trace_2 = 0, trace_res = 0;
139
140     for(int i = 0; i < matrix_size; i++)
141     {
142         for(int j = 0; j < matrix_size; j++)
143         {
144             mat1[i][j] = (rand() % 10) + 1;
145             trace_1 += mat1[i][j];
146             mat2[i][j] = (rand() % 10) + 1;
147             trace_2 += mat2[i][j];
148         }
149     }
150     double time = omp_get_wtime();
151
152     //compute
153
154     #pragma omp parallel for
155     for(int i = 0; i < matrix_size; i++)
156     {
157         #pragma omp parallel for shared(trace_res)
158         for(int j = 0; j < matrix_size; j++)
159         {
160             res[i][j] = 0;
161             #pragma omp parallel for shared(res)
162             for(int k = 0; k < matrix_size; k++)
163             {
164                 res[i][j] += mat1[i][k] * mat2[k][j];
165             }
166             if(i == j) trace_res += res[i][j];
167         }
168     }

```

Compilation

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

Description

00:12
09-12-2022

MCSE503P Computer Architecture x +

moovit.vit.ac.in/mod/vpl/forms/edit.php?id=46416&userid=106340

NIDHI SINGH 22MAI0015

f1.c

```

130 }
131 void matrix_multiplication(int matrix_size, int num_threads)
132 {
133     srand(time(0));
134     omp_set_num_threads(num_threads);
135
136     //generating input matrices and declaring output matrix
137     int mat1[matrix_size][matrix_size], mat2[matrix_size][matrix_size],
138     int trace_1 = 0, trace_2 = 0, trace_res = 0;
139
140     for(int i = 0; i < matrix_size; i++)
141     {
142         for(int j = 0; j < matrix_size; j++)
143         {
144             mat1[i][j] = (rand() % 10) + 1;
145             trace_1 += mat1[i][j];
146             mat2[i][j] = (rand() % 10) + 1;
147             trace_2 += mat2[i][j];
148         }
149     }
150     double time = omp_get_wtime();
151
152     //compute
153
154     #pragma omp parallel for
155     for(int i = 0; i < matrix_size; i++)
156     {
157         #pragma omp parallel for shared(trace_res)
158         for(int j = 0; j < matrix_size; j++)
159         {
160             res[i][j] = 0;
161             #pragma omp parallel for shared(res)
162             for(int k = 0; k < matrix_size; k++)
163             {
164                 res[i][j] += mat1[i][k] * mat2[k][j];
165             }
166             if(i == j) trace_res += res[i][j];
167         }
168     }

```

Compilation

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

Description

00:12
09-12-2022

MCSE503P Computer Architecture x +

moovit.vit.ac.in/mod/vpl/forms/edit.php?id=46416&userid=106340

NIDHI SINGH 22MAI0015

f1.c

```
130 }
131 void matrix_multiplication(int matrix_size, int num_threads)
132 {
133     srand(time(0));
134     omp_set_num_threads(num_threads);
135
136     //generating input matrices and declaring output matrix
137     int mat1[matrix_size][matrix_size], mat2[matrix_size][matrix_size],
138     int trace_1 = 0, trace_2 = 0, trace_res = 0;
139
140     for(int i = 0; i < matrix_size; i++)
141     {
142         for(int j = 0; j < matrix_size; j++)
143         {
144             mat1[i][j] = (rand() % 10) + 1;
145             trace_1 += mat1[i][j];
146             mat2[i][j] = (rand() % 10) + 1;
147             trace_2 += mat2[i][j];
148         }
149     }
150     double time = omp_get_wtime();
151     //compute
152
153     #pragma omp parallel for
154     for(int i = 0; i < matrix_size; i++)
155     {
156         #pragma omp parallel for shared(trace_res)
157         for(int j = 0; j < matrix_size; j++)
158         {
159             res[i][j] = 0;
160             #pragma omp parallel for shared(res)
161             for(int k = 0; k < matrix_size; k++)
162             {
163                 res[i][j] += mat1[i][k] * mat2[k][j];
164             }
165             if(i == j) trace_res += res[i][j];
166         }
167     }
168 }
```

Compilation

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

Description

00:11 09-12-2022