**LAB # 7 :-**

**OBJECT : TO UNDERSTAND MULTITHREADING.**

**Write the same code for matrix multiplication and divide the code into ten threads. Also note the timestamp at the start and end of the program. Give your conclusion.**

**Matrix Multiplication using Traditional Method:**

**Program :**

**MatrixMultiplication.java**

import java.lang.\*;

public class MatrixMultiplication{

public static void main(String args[]){

System.out.print("Current Time in milliseconds =");

System.out.println(System.currentTimeMillis());

double time1=System.currentTimeMillis();

int a[][]={{1,2,3},{4,6,7},{8,9,10}};

int b[][]={{1,2,3},{4,6,7},{8,9,10}};

int c[][]=new int[3][3];

for(int i=0;i<3;i++){

for(int j=0;j<3;j++){

c[i][j]=0;

for(int k=0;k<3;k++){

c[i][j]+=a[i][k]\*b[k][j];

}

System.out.print(c[i][j]+" ");

}

System.out.println();

}

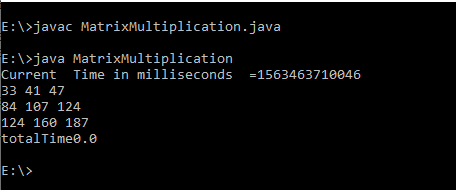
double time2=System.currentTimeMillis();

double totalTime=time2-time1;

System.out.println("totalTime"+totalTime);

}}

**OUTPUT :**

****

**Matrix Multiplication using Thread:**

**Program :**

**ParentThread.java**

public class ParentThread{

public static final int NUM\_OF\_THREADS = 9;

public static void main(String args[]){

int row;

int col;

int A[][] = {{1,2,3},{5,5,6},{8,7,8}};

int B[][]= {{1,2,3},{5,5,6},{8,7,8}};

int C[][] = new int[3][3];

int threadcount = 0;

System.out.print("Current Time in milliseconds =");

System.out.println(System.currentTimeMillis());

double time1=System.currentTimeMillis();

Thread[] thrd = new Thread[NUM\_OF\_THREADS];

try{

for (row = 0 ; row < 3; row++){

for (col = 0 ; col < 3; col++ )

{

thrd[threadcount] = new Thread(new WorkerTh(row, col, A, B, C));

thrd[threadcount].start();

thrd[threadcount].join();

threadcount++;

}}}

catch (InterruptedException ie){}

System.out.println("A Matrix :");

for (row = 0 ; row < 3; row++){

for (col = 0 ; col < 2; col++ ){

System.out.print(" "+A[row][col]);

}

System.out.println();

}

System.out.println(" B Matrix :");

for (row = 0 ; row < 2; row++){

for (col = 0 ; col < 3; col++ ){

System. out . print ( " "+B [row] [col]) ;

}

System.out.println();

}

System.out.println(" Resulting C Matrix :");

for (row = 0 ; row < 3; row++){

for (col = 0 ; col < 3; col++){

System.out.print(" "+C[row][col]);

}

System.out.println();

}

double time2=System.currentTimeMillis();

double totalTime=time2-time1;

System.out.println("totalTimen+totalTime");

}

}

class WorkerTh implements Runnable

{

private int row;

private int col;

private int A[][];

private int B[][];

private int C[][];

public WorkerTh(int row, int col, int A[][], int B[][], int C[][] )

{

this.row = row;

this.col = col;

this.A = A;

this.B = B;

this.C = C;

}

@Override

public void run()

{

for(int k = 0; k < B.length; k++)

{

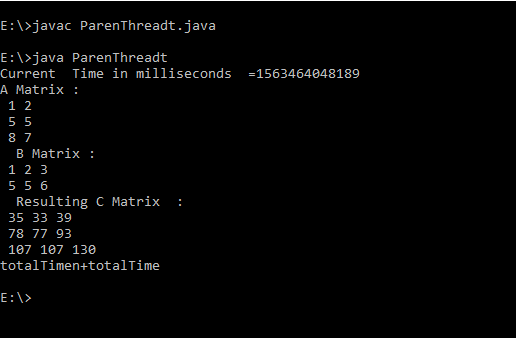
C[row][col] += A[row][k] \* B[k][col];

}

}

}

**OUTPUT :**

****

**Conclusion: time taken to calculate the matrix multiplication using threads is more than the traditional one.**