Name : Dahlia Shehata Mahmoud

ID : 28

Lab Assignment 2: Threads

Problem Statement:

- It is required to implement multi-threaded matrix multiplication using 2 methods:
(1) a thread computes each row in the output matrix
(2)a thread computes each element in the output matrix.
- Compare the two implementations according to the following:
(1) the number of thread created
(2) the execution time taken.

Code Organization:

The code is divided between header files and C files.
There exists 3 main C files:

1. **file_processing.c:** responsible for reading matrices from files and allocate memory for them in order to be stored and for writing the multiplication result in the output file.
2. **Rows_Threading.c:** responsible to create array of threads of length equal to the number of the output matrix rows, calculate each threaded row using matrix multiplication and finally join the threads together in case of success else report error to the main function.
3. **Elements_Threading.c:**responsible to create array of threads of length equal to the number of the output matrix elements (r1*c2), calculate each threaded element using matrix multiplication and finally join the threads together in case of success else report error to the main function.
4. **main.c:** uses the 3 above files to read files ,store them , allocate memory for the output array,execute the 2 matrix multiplication methods and calculate the time stamp taken by each one (in seconds and microseconds) as well as the number of thread used .

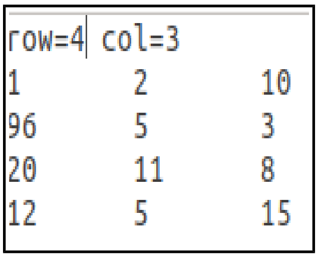
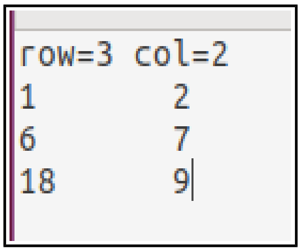
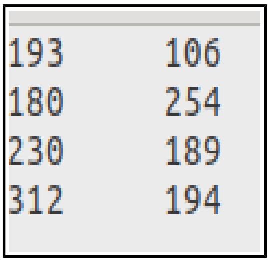
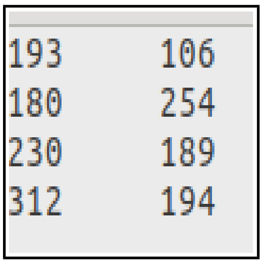
Main Functions:

Function Name	Responsible for
void elementMultiplication (void* thread_id)	When the element is the current thread ,this function computes the multiplication value for each element
int threadedElements()	Creates array of threads of size number of elements in the output array and calculate each element and then join threads
void rowMultiplication (void*thread_id)	When the row is the current thread , this function computes the multiplication value for each row
int threadedRows ()	Creates array of threads of size number of rows in the output array and calculate each row and then join threads
long longreadFile (char* fileName,int file_id)**	Read the specified input file or “a.txt/b.txt” and store matrices in memory
bool writeToFile(char*fileName)	Write the output array in file
void setup()	Set the defaults values for files
char* modifyFileName (char* fileName,char*extension)	Generates the the 2 output filenames from the name specified by user
bool checkExtension (char*fileName)	Handles “.txt”extension for formatting purposes
void displayData (struct timeval start, struct timeval stop,int threads,int id);	Display data in console : number of threads used and tome taken by each method

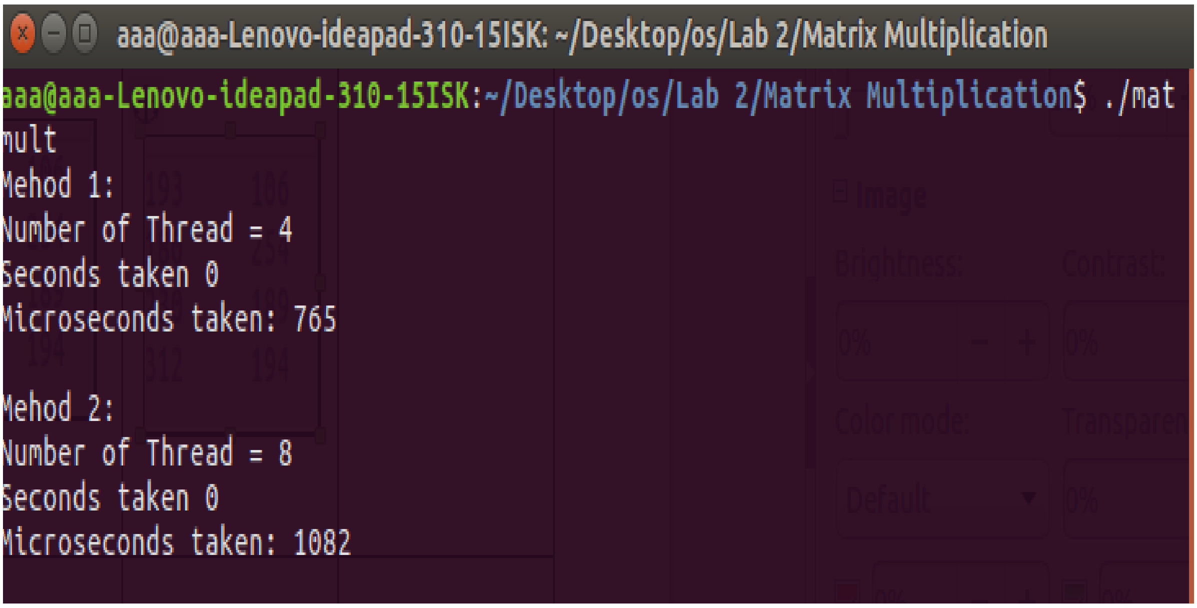
How to compile and run code:

- in the terminal reach the directory of the project using “cd”
- type “make” to generate “matmult.out”
- in the terminal type ./matmult to multiply matrices in “a.txt” and “b.txt” and get the output in “c_1.txt”and “c_2.txt”
- n the terminal type ./matmult Mat1 Mat2 MatOut to multiply matrices in “Mat1” and “Mat2” and get the output in “MatOut_1.txt”and “MatOut_2.txt”.

Sample runs:

Input file 1	Input File 2	Output file 1	Output File 2
a.txt	**b.txt**	**C_1.txt**	**C_2.txt**
 row=4 col=3 1 2 10 96 5 3 20 11 8 12 5 15	 row=3 col=2 1 2 6 7 18 9	 193 106 180 254 230 189 312 194	 193 106 180 254 230 189 312 194

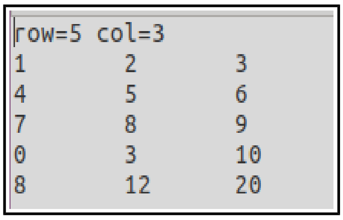
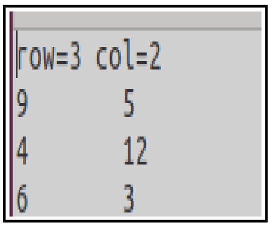
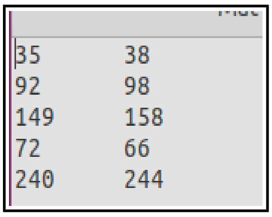
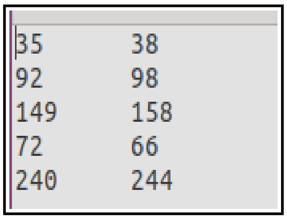
Console



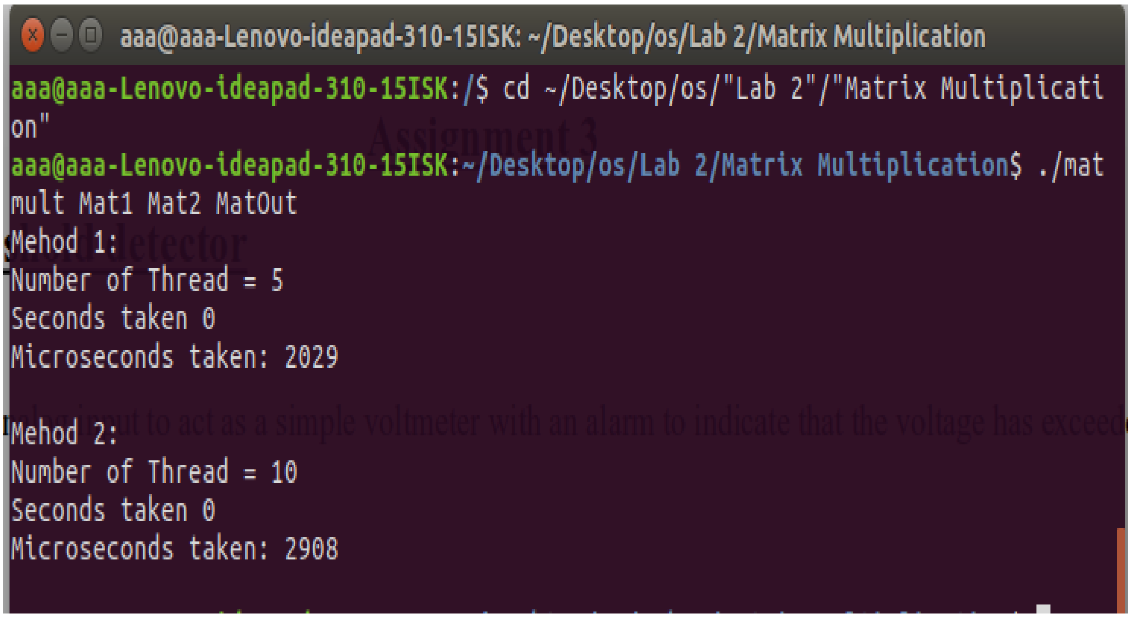
aaa@aaa-Lenovo-ideapad-310-15ISK: ~/Desktop/os/Lab 2/Matrix Multiplication

```
aaa@aaa-Lenovo-ideapad-310-15ISK:~/Desktop/os/Lab 2/Matrix Multiplication$ ./mat
mult
Mehod 1:
Number of Thread = 4
Seconds taken 0
Microseconds taken: 765

Mehod 2:
Number of Thread = 8
Seconds taken 0
Microseconds taken: 1082
```

Input file 1	Input File 2	Output file 1	Output File 2
Mat1  row=5 col=3 1 2 3 4 5 6 7 8 9 0 3 10 8 12 20	**Mat2**  row=3 col=2 9 5 4 12 6 3	**MatOut_1**  35 38 92 98 149 158 72 66 240 244	**MatOut_2**  35 38 92 98 149 158 72 66 240 244

Console



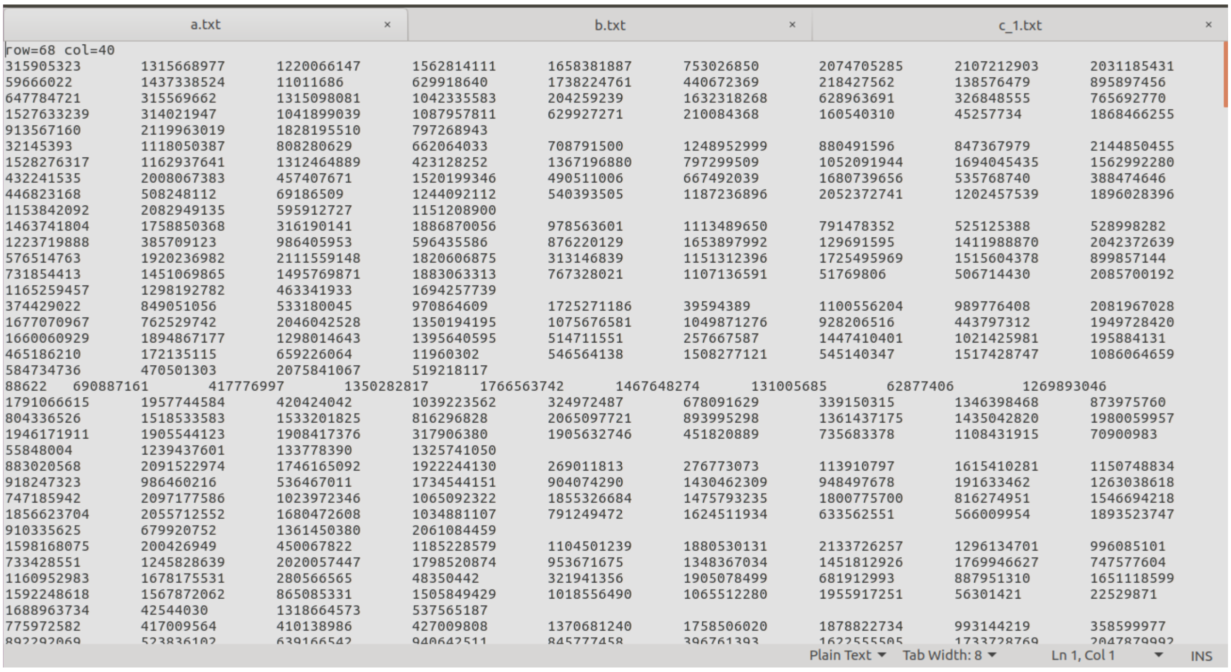
aaa@aaa-Lenovo-ideapad-310-15ISK: ~/Desktop/os/Lab 2/Matrix Multiplication

```
aaa@aaa-Lenovo-ideapad-310-15ISK:/$ cd ~/Desktop/os/"Lab 2"/"Matrix Multiplicati
on"
aaa@aaa-Lenovo-ideapad-310-15ISK:~/Desktop/os/Lab 2/Matrix Multiplication$ ./mat
mult Mat1 Mat2 MatOut
Mehod 1:
Number of Thread = 5
Seconds taken 0
Microseconds taken: 2029

Mehod 2:
Number of Thread = 10
Seconds taken 0
Microseconds taken: 2908
```

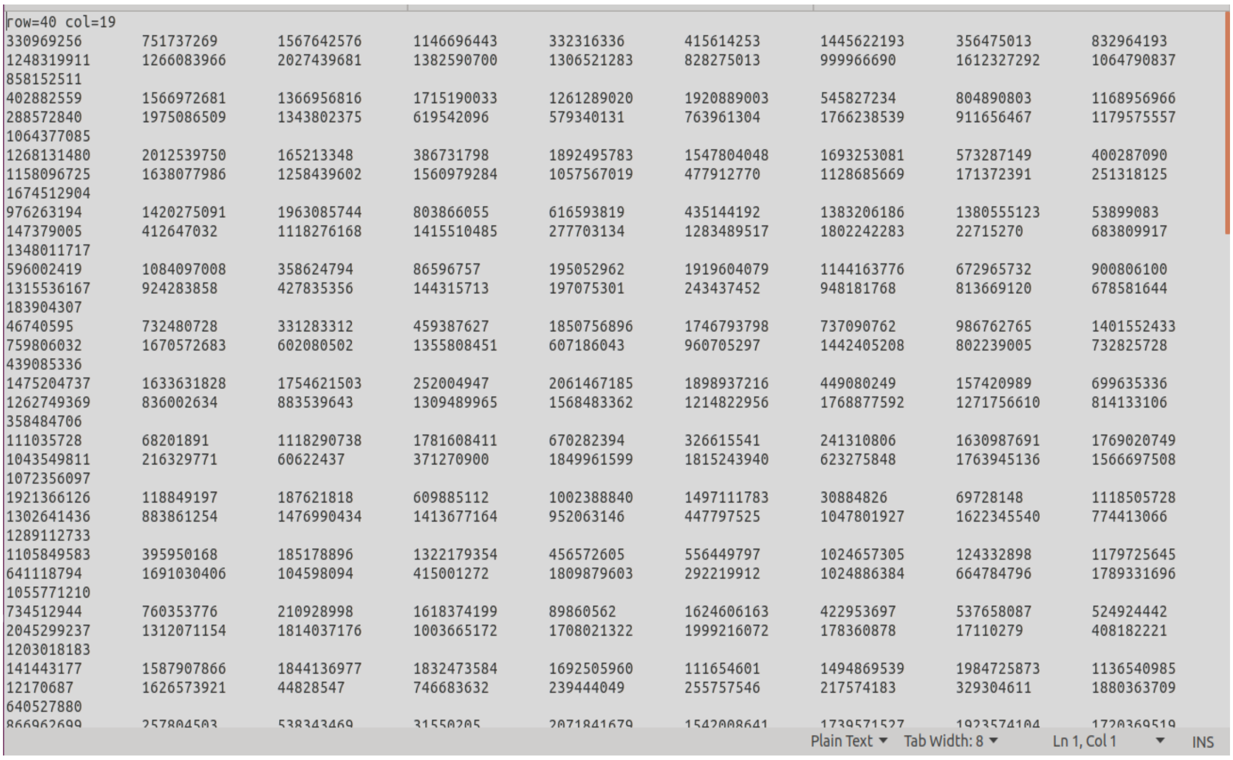
Change directory to the the one containing project

a.txt:



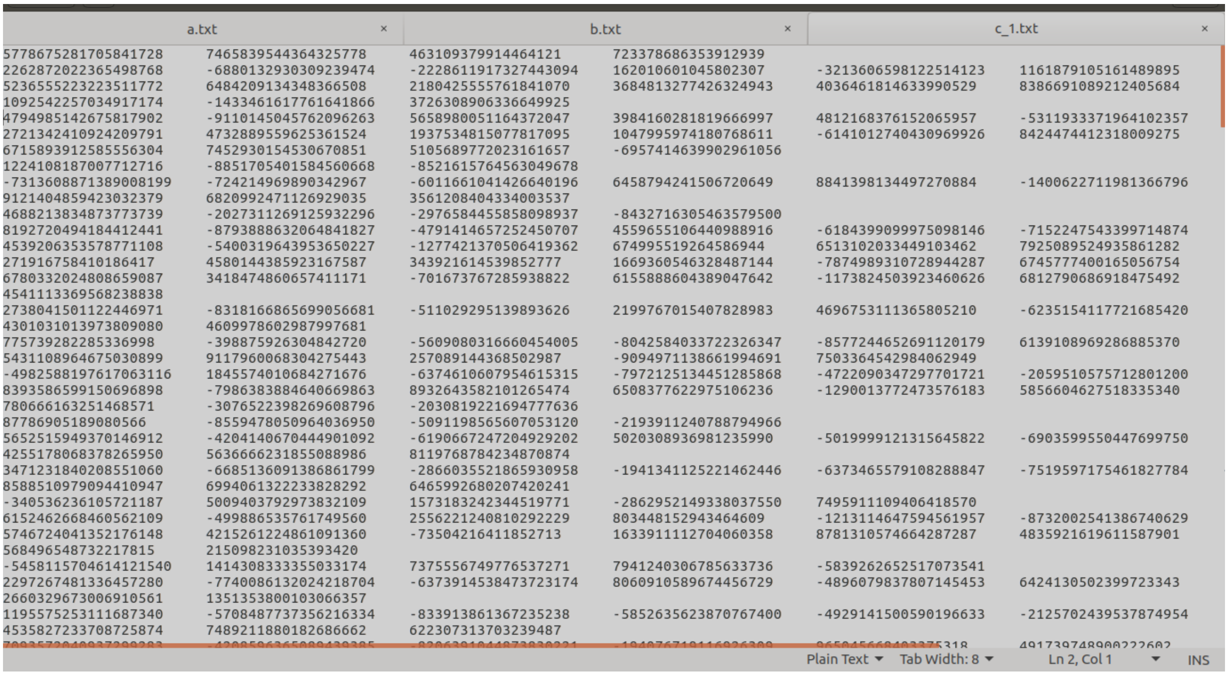
```
row=68 col=40
315905323     1315668977    1220066147    1562814111    1658381887    753026850     2074705285    2107212903    2031185431
59666022      1437338524    11011686      629918640     1738224761    440672369     218427562     138576479     895897456
647784721     315569662     1315098081    1042335583    204259239     1632318268    628963691     326848555     765692770
1527633239    314021947     1041899039    1087957811    629927271     210084368     160540310     45257734      1868466255
913567160     2119963019    1828195510    797268943
32145393      1118050387    808280629     662064033     708791500     1248952999    880491596     847367979     2144850455
1528276317    1162937641    1312464889    423128252     1367196880    797299509     1052091944    1694045435    1562992280
432241535     2008067383    457407671     1520199346    490511006     667492039     1680739656    535768740     388474646
446823168     508248112     69186509      1244092112    540393505     1187236896    2052372741    1202457539    1896028396
1153842092    2082949135    595912727     1151208900
1463741804    1758850368    316190141     1886870056    978563601     1113489650    791478352     525125388     528998282
1223719888    385709123     986405953     596435586     876220129     1653897992    129691595     1411988870    2042372639
576514763     1920236982    2111559148    1820606875    313146839     1151312396    1725495969    1515604378    899857144
731854413     1451069865    1495769871    1883063313    767328021     1107136591    51769806      506714430     2085700192
1165259457    1298192782    463341933     1694257739
374429022     849051056     533180045     970864609     1725271186    39594389      1100556204    989776408     2081967028
1677070967    762529742     2046042528    1350194195    1075676581    1049871276    928206516     443797312     1949728420
1660060929    1894867177    1298014643    1395640595    514711551     257667587     1447410401    1021425981    195884131
465186210     172135115     659226064     11960302      546564138     1508277121    545140347     1517428747    1086064659
584734736     470501303     2075841067    519218117
88622   690887161       417776997       1350282817      1766563742      1467648274      131005685       62877406        1269893046
1791066615    1957744584    420424042     1039223562    324972487     678091629     339150315     1346398468    873975760
804336526     1518533583    1533201825    816296828     2065097721    893995298     1361437175    1435042820    1980059957
1946171911    1905544123    1908417376    317906380     1905632746    451820889     735683378     1108431915    70900983
55848004      1239437601    133778390     1325741050
883020568     2091522974    1746165092    1922244130    269011813     276773073     113910797     1615410281    1150748834
918247323     986460216     536467011     1734544151    904074290     1430462309    948497678     191633462     1263038618
747185942     2097177586    1023972346    1065092322    1855326684    1475793235    1800775700    816274951     1546694218
1856623704    2055712552    1680472608    1034881107    791249472     1624511934    633562551     566009954     1893523747
910335625     679920752     1361450380    2061084459
1598168075    200426949     450067822     1185228579    1104501239    1880530131    2133726257    1296134701    996085101
733428551     1245828639    2020057447    1798520874    953671675     1348367034    1451812926    1769946627    747577604
1160952983    1678175531    280566565     48350442      321941356     1905078499    681912993     887951310     1651118599
1592248618    1567872062    865085331     1505849429    1018556490    1065512280    1955917251    56301421      22529871
1688963734    42544030      1318664573    537565187
775972582     417009564     410138986     427009808     1370681240    1758506020    1878822734    993144219     358599977
892292069     523836102     639166542     940642511     845777458     396761393     1622555505    1733728769    2047879992
```

b.txt:



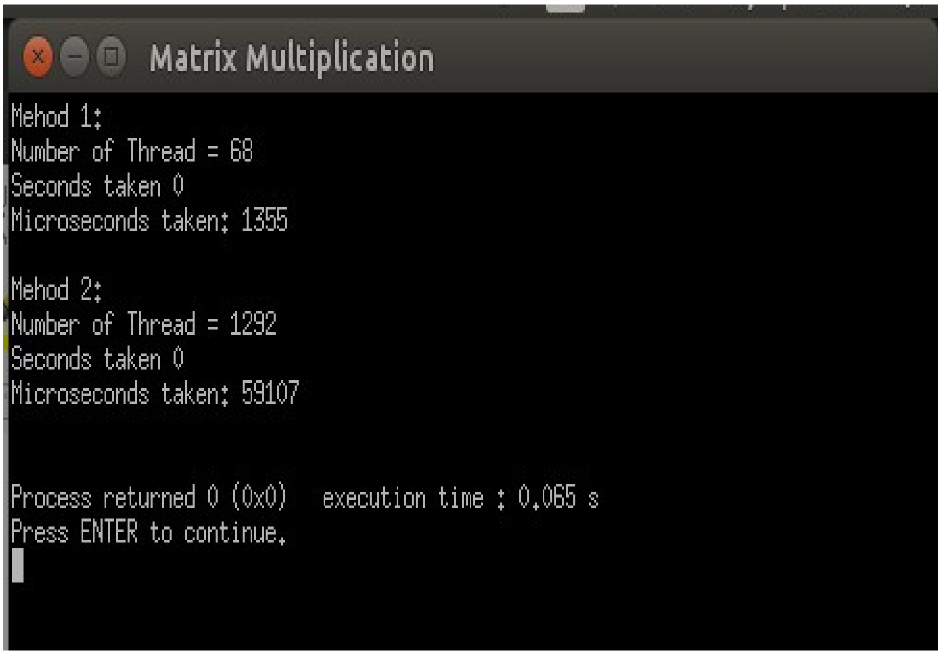
```
row=40 col=19
330969256     751737269     1567642576    1146696443    332316336     415614253     1445622193    356475013     832964193
1248319911    1266083966    2027439681    1382590700    1306521283    828275013     999966690     1612327292    1064790837
858152511
402882559     1566972681    1366956816    1715190033    1261289020    1920889003    545827234     804890803     1168956966
288572840     1975086509    1343802375    619542096     579340131     763961304     1766238539    911656467     1179575557
1064377085
1268131480    2012539750    165213348     386731798     1892495783    1547804048    1693253081    573287149     400287090
1158096725    1638077986    1258439602    1560979284    1057567019    477912770     1128685669    171372391     251318125
1674512904
976263194     1420275091    1963085744    803866055     616593819     435144192     1383206186    1380555123    53899083
147379005     412647032     1118276168    1415510485    277703134     1283489517    1802242283    22715270      683809917
1348011717
596002419     1084097008    358624794     86596757      195052962     1919604079    1144163776    672965732     900806100
1315536167    924283858     427835356     144315713     197075301     243437452     948181768     813669120     678581644
183904307
46740595      732480728     331283312     459387627     1850756896    1746793798    737090762     986762765     1401552433
759806032     1670572683    602080502     1355808451    607186043     960705297     1442405208    802239005     732825728
439085336
1475204737    1633631828    1754621503    252004947     2061467185    1898937216    449080249     157420989     699635336
1262749369    836002634     883539643     1309489965    1568483362    1214822956    1768877592    1271756610    814133106
358484706
111035728     68201891      1118290738    1781608411    670282394     326615541     241310806     1630987691    1769020749
1043549811    216329771     60622437      371270900     1849961599    1815243940    623275848     1763945136    1566697508
1072356097
1921366126    118849197     187621818     609885112     1002388840    1497111783    30884826      69728148      1118505728
1302641436    883861254     1476990434    1413677164    952063146     447797525     1047801927    1622345540    774413066
1289112733
1105849583    395950168     185178896     1322179354    456572605     556449797     1024657305    124332898     1179725645
641118794     1691030406    104598094     415001272     1809879603    292219912     1024886384    664784796     1789331696
1055771210
734512944     760353776     210928998     1618374199    89860562      1624606163    422953697     537658087     524924442
2045299237    1312071154    1814037176    1003665172    1708021322    1999216072    178360878     17110279      408182221
1203018183
141443177     1587907866    1844136977    1832473584    1692505960    111654601     1494869539    1984725873    1136540985
12170687      1626573921    44828547      746683632     239444049     255757546     217574183     329304611     1880363709
640527880
866962699     257804503     538343469     31550205      2071841679    1542008641    1739571527    1923574104    1720369519
```

c_1.txt:



```
5778675281705841728     7465839544364325778     463109379914464121      723378686353912939
2262872022365498768     -6880132930309239474    -2228611917327443094    162010601045802307      -3213606598122514123    1161879105161489895
5236555223223511772     6484209134348366508     2180425555761841070     3684813277426324943     4036461814633990529     8386691089212405684
1092542257034917174     -1433461617761641866    3726308906336649925
4794985142675817902     -9110145045762096263    5658980051164372047     3984160281819666997     4812168376152065957     -5311933371964102357
2721342410924209791     4732889559625361524     1937534815077817095     1047995974180768611     -6141012740430969926    8424474412318009275
6715893912585556304     7452930154530670851     5105689772023161657     -6957414639902961056
1224108187007712716     -8851705401584560668    -8521615764563049678
-7313608871389008199    -724214969890342967     -6011661041426640196    6458794241506720649     8841398134497270884     -1400622711981366796
9121404859423032379     6820992471126929035     3561208404334003537
4688213834873773739     -2027311269125932296    -2976584455858098937    -8432716305463579500
8192720494184412441     -8793888632064841827    -4791414657252450707    4559655106440988916     -6184399099975098146    -7152247543399714874
4539206353578771108     -5400319643953650227    -1277421370506419362    674995519264586944      6513102033449103462     7925089524935861282
271916758410186417      4580144385923167587     343921614539852777      1669360546328487144     -7874989310728944287    6745777400165056754
6780332024808659087     3418474860657411171     -701673767285938822     6155888604389047642     -1173824503923460626    6812790686918475492
4541113369568238838
2738041501122446971     -8318166865699056681    -511029295139893626     2199767015407828983     4696753111365805210     -6235154117721685420
4301031013973809080     4609978602987997681
775739282285336998      -398875926304842720     -5609080316660454005    -8042584033722326347    -8577244652691120179    6139108969286885370
5431108964675030899     9117960068304275443     257089144368502987      -9094971138661994691    7503364542984062949
-4982588197617063116    1845574010684271676     -6374610607954615315    -7972125134451285868    -4722090347297701721    -2059510575712801200
8393586599150696898     -7986383884640669863    8932643582101265474     6508377622975106236     -1290013772473576183    5856604627518335340
780666163251468571      -3076522398269608796    -2030819221694777636
87786905189080566       -8559478050964036950    -5091198565607053120    -2193911240788794966
5652515949370146912     -4204140670444901092    -6190667247204929202    5020308936981235990     -5019999121315645822    -6903599550447699750
4255178068378265950     5636666231855088986     8119768784234870874
3471231840208551060     -6685136091386861799    -2866035521865930958    -1941341125221462446    -6373465579108288847    -7519597175461827784
8588510979094410947     6994061322233828292     6465992680207420241
-340536236105721187     5009403792973832109     1573183242344519771     -2862952149338037550    7495911109406418570
6152462668460562109     -499886535761749560     2556221240810292229     803448152943464609      -1213114647594561957    -8732002541386740629
5746724041352176148     4215261224861091360     -73504216411852713      1633911112704060358     8781310574664287287     4835921619611587901
568496548732217815      215098231035393420
-5458115704614121540    1414308333355033174     7375556749776537271     7941240306785633736     -5839262652517073541
2297267481336457280     -7740086132024218704    -6373914538473723174    8060910589674456729     -4896079837807145453    6424130502399723343
2660329673006910561     1351353800103066357
1195575253111687340     -5708487737356216334    -833913861367235238     -5852635623870767400    -4929141500590196633    -2125702439537874954
4535827233708725874     7489211880182686662     622307313703239487
7093572040937299283     -4208596365089439385    -8206391044873830221    -194076719116926309     965045668403375318      491739748900222602
```

Console:

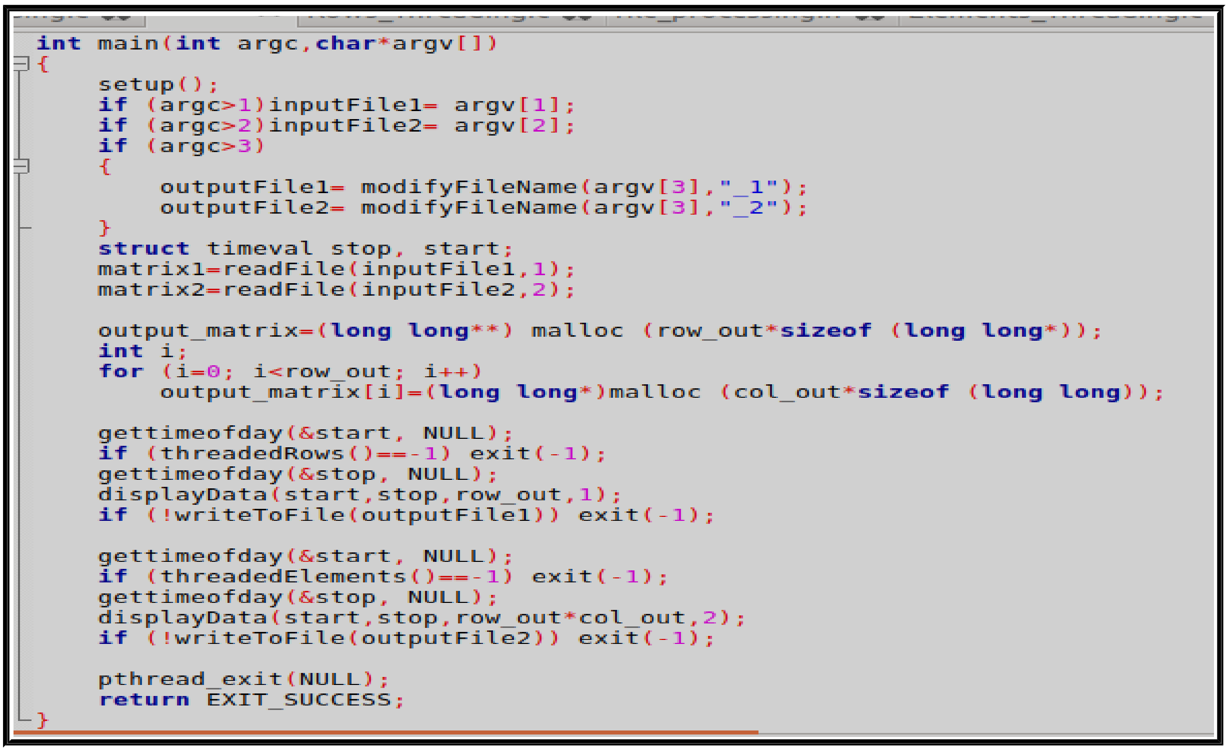


```
Mehod 1:
Number of Thread = 68
Seconds taken 0
Microseconds taken: 1355

Mehod 2:
Number of Thread = 1292
Seconds taken 0
Microseconds taken: 59107


Process returned 0 (0x0)   execution time : 0.065 s
Press ENTER to continue.
```

Code Snippets:



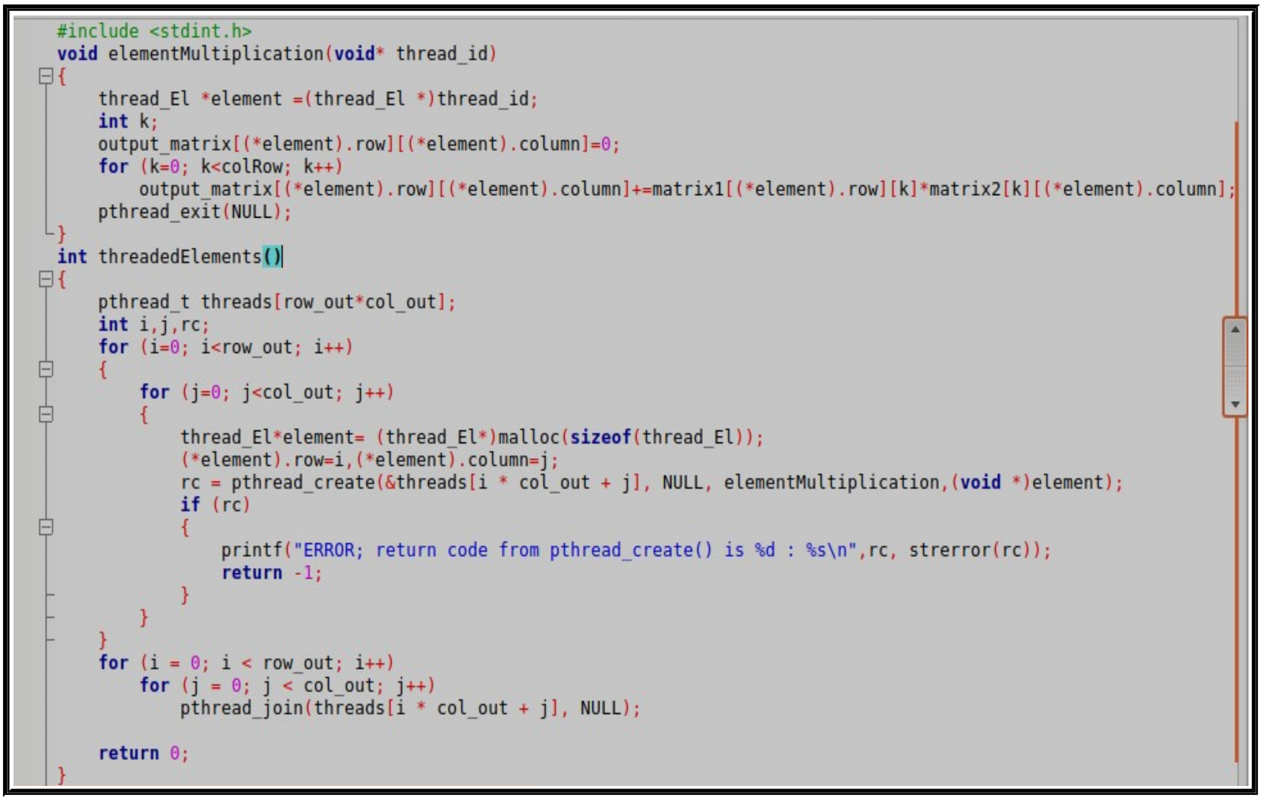
```c
int main(int argc,char*argv[])
{
    setup();
    if (argc>1)inputFile1= argv[1];
    if (argc>2)inputFile2= argv[2];
    if (argc>3)
    {
        outputFile1= modifyFileName(argv[3],"_1");
        outputFile2= modifyFileName(argv[3],"_2");
    }
    struct timeval stop, start;
    matrix1=readFile(inputFile1,1);
    matrix2=readFile(inputFile2,2);

    output_matrix=(long long**) malloc (row_out*sizeof (long long*));
    int i;
    for (i=0; i<row_out; i++)
        output_matrix[i]=(long long*)malloc (col_out*sizeof (long long));

    gettimeofday(&start, NULL);
    if (threadedRows()==-1) exit(-1);
    gettimeofday(&stop, NULL);
    displayData(start,stop,row_out,1);
    if (!writeToFile(outputFile1)) exit(-1);

    gettimeofday(&start, NULL);
    if (threadedElements()==-1) exit(-1);
    gettimeofday(&stop, NULL);
    displayData(start,stop,row_out*col_out,2);
    if (!writeToFile(outputFile2)) exit(-1);

    pthread_exit(NULL);
    return EXIT_SUCCESS;
}
```

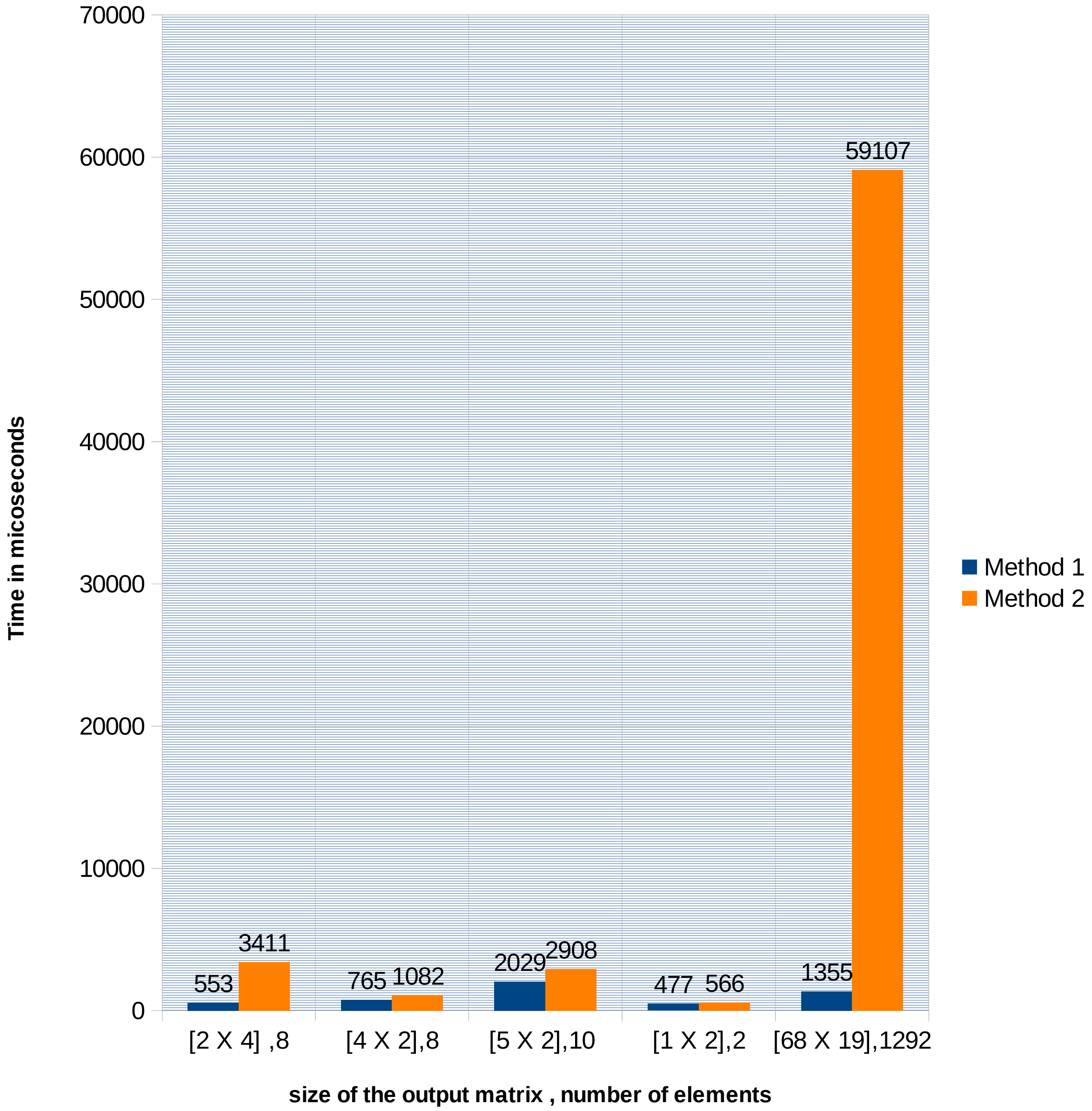


```c
#include <stdint.h>
void elementMultiplication(void* thread_id)
{
    thread_El *element =(thread_El *)thread_id;
    int k;
    output_matrix[(*element).row][(*element).column]=0;
    for (k=0; k<colRow; k++)
        output_matrix[(*element).row][(*element).column]+=matrix1[(*element).row][k]*matrix2[k][(*element).column];
    pthread_exit(NULL);
}
int threadedElements()
{
    pthread_t threads[row_out*col_out];
    int i,j,rc;
    for (i=0; i<row_out; i++)
    {
        for (j=0; j<col_out; j++)
        {
            thread_El*element= (thread_El*)malloc(sizeof(thread_El));
            (*element).row=i,(*element).column=j;
            rc = pthread_create(&threads[i * col_out + j], NULL, elementMultiplication,(void *)element);
            if (rc)
            {
                printf("ERROR; return code from pthread_create() is %d : %s\n",rc, strerror(rc));
                return -1;
            }
        }
    }
    for (i = 0; i < row_out; i++)
        for (j = 0; j < col_out; j++)
            pthread_join(threads[i * col_out + j], NULL);

    return 0;
}
```

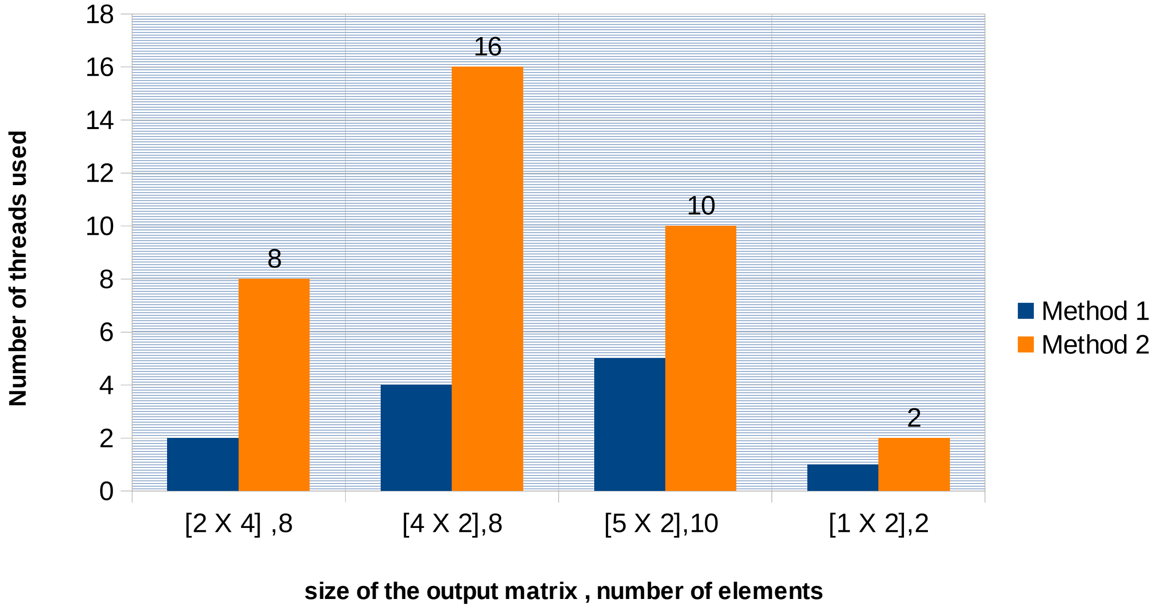
Comparison between the two methods of matrix multiplication:

1. Relation between size of different samples and time per microseconds:



- According to graph :The second Method is slower

2. **Relation between size of different samples and number of threads:**



- According to graph : the second Method uses larger number of threads

In conclusion :

Number of threads is proportional with time in microseconds. The more threads we use,the more time we take and the slower the multiplication is.

Method 1	Method 2
faster	slower
Number of threads is less Number of threads= number of rows in the output matrix	Number of the threads is larger Number of threads= number of elements in the output matrix