Name:Akshay S Gadhave
Roll No:221076
SY Comp A3

Assignment 5

AIM: Thread management using pthread library. Implement matrix multiplication using multithreading. Application should have pthread_create, pthread_join, pthread_exit. In the program, every thread must return the value and must be collected in pthread_join in the main function. Final sum of row column multiplication must be done by main thread (main function).

Theory :

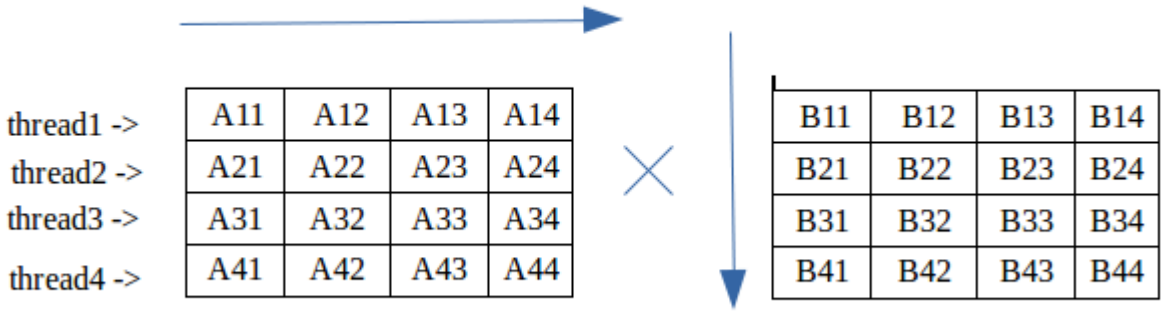
Multiplication of Matrix using threads

Multiplication of matrix does take time surely. Time complexity of matrix multiplication is O(n^3) using normal matrix multiplication. And **Strassen algorithm** improves it and its time complexity is O(n^(2.8074)).

But, Is there any way to improve the performance of matrix multiplication using the normal method.
Multi-threading can be done to improve it. In multi-threading, instead of utilizing a single core of your processor, we utilizes all or more core to solve the problem.

We create different threads, each thread evaluating some part of matrix multiplication.
Depending upon the number of cores your processor has, you can create the number of threads required. Although you can create as many threads as you need, a better way is to create each thread for one core.

n second approach,we create a seperate thread for each element in resultant matrix. Using **pthread_exit()** we return computed value from each thread which is collected by **pthread_join()**. This approach does not make use of any global variables.



C language Code:

```
#include<stdio.h>
#include<pthread.h>
#include<unistd.h>
#include<stdlib.h>
#define MAX 4


//Each thread computes single element
void *mult(void* arg)
{
   int *data = (int *)arg;
   int k = 0, i = 0;

   int x = data[0];
   for (i = 1; i <= x; i++)
         k += data[i]*data[i+x];

   int *p = (int*)malloc(sizeof(int));
       *p = k;
```

```
    pthread_exit(p);
}


int main()
{

    int matA[MAX][MAX];
    int matB[MAX][MAX];


    int r1=MAX,c1=MAX,r2=MAX,c2=MAX,i,j,k;


    // Accepting values in matA
    printf("Enter First Matrix of 4 * 4\n");
    for (i = 0; i < r1; i++)
            for (j = 0; j < c1; j++)
                 scanf("%d",&matA[i][j]);

    //Accepting values in matB
    printf("Enter Second Matrix of 34* 4\n");
    for (i = 0; i < r1; i++)
            for (j = 0; j < c1; j++)
                scanf("%d",&matB[i][j]);




    int max = r1*c2;


    //declaring array of threads of size r1*c2
    pthread_t *threads;
    threads = (pthread_t*)malloc(max*sizeof(pthread_t));
```

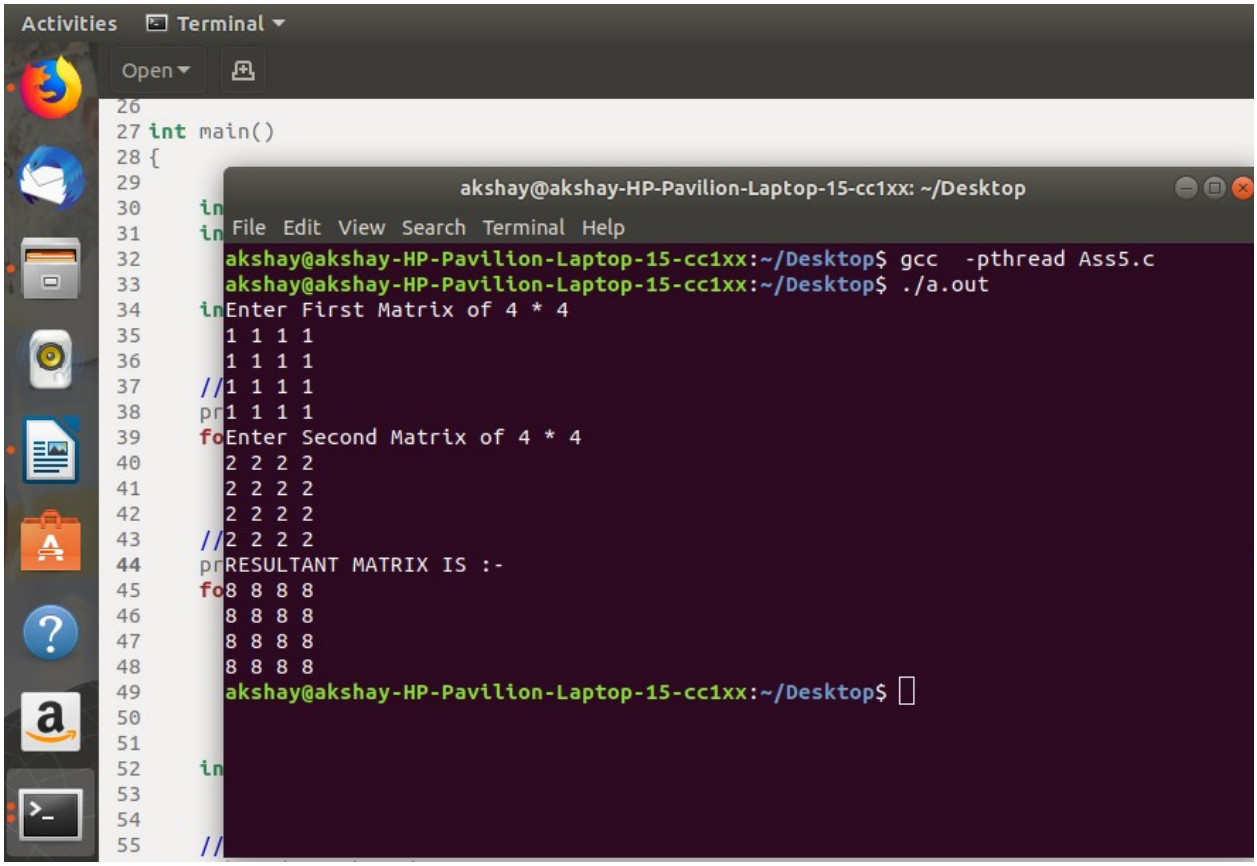
```
int count = 0;
int* data = NULL;
for (i = 0; i < r1; i++)
    for (j = 0; j < c2; j++)
          {

            //storing row and column elements in data
        data = (int *)malloc((20)*sizeof(int));
        data[0] = c1;

        for (k = 0; k < c1; k++)
            data[k+1] = matA[i][k];

        for (k = 0; k < r2; k++)
            data[k+c1+1] = matB[k][j];

         //creating threads
            pthread_create(&threads[count++], NULL,
                        mult, (void*)(data));

                }

printf("RESULTANT MATRIX IS :- \n");
for (i = 0; i < max; i++)
{
  void *k;

  //Joining all threads and collecting return value
  pthread_join(threads[i], &k);


     int *p = (int *)k;
  printf("%d ",*p);
  if ((i + 1) % c2 == 0)
     printf("\n");
}
```

```
  return 0;
}
```

Compile in linux using following code:

```
gcc -pthread program_name.cpp
```

Output:


Activities
Terminal
Open
int main()
{
akshay@akshay-HP-Pavilion-Laptop-15-cc1xx: ~/Desktop
File Edit View Search Terminal Help
akshay@akshay-HP-Pavilion-Laptop-15-cc1xx:~/Desktop$ gcc -pthread Ass5.c
akshay@akshay-HP-Pavilion-Laptop-15-cc1xx:~/Desktop$ ./a.out
Enter First Matrix of 4 * 4
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
Enter Second Matrix of 4 * 4
2 2 2 2
2 2 2 2
2 2 2 2
2 2 2 2
RESULTANT MATRIX IS :-
8 8 8 8
8 8 8 8
8 8 8 8
8 8 8 8
akshay@akshay-HP-Pavilion-Laptop-15-cc1xx:~/Desktop$