CSE-467 - Parallel and Distributed Computing

Assignment No 1

Objective: Find the impact of programming habits on data access time in various languages.

All programming languages from low level assembly to high-level C/C++/Java/Python/Go allow programmers to move data from RAM to registers without knowing the existence of cache memory or even the registers. And programmers also ignore how the programming languages store multidimensional arrays as linear array in RAM. In this assignment you need to explore the impact of coding on memory access time in various languages while cache hit and miss occurred and how the locality of reference play its role to increase cache hits.

Submit a word file containing all codes, output snapshots and the comparison table

----

Input: Matrix: assigned with random values

Processing: determine the memory access time of matrix stored in two-dimensional arrays or lists in different languages.

T1 = get clock cycle/system time

Perform M1 = M2 + M3

T2 = get clock cycle/system time

Time elapsed = T2 – T1

Output: Time elapsed - Tables show the comparison

Execution Time					
	C/C++	Java	Matlab	Python-list	Python-NumPy
Without loop M1=M2+M3	-	-	0.000681 secs	0.00100159645080 5664 sec	0.00100517272 94921875 sec
Row Major loops (R-C)	0.000878947 seconds / 1017.000000 Clock Cycles	0.004582219 seconds	0.012095 secs	0.17294764518737 793 sec	0.51291561126 70898 sec
Column Major loops (C-R)	0.00183034 seconds / 1875.000000 Clock Cycles	0.00172495 seconds	0.008265 secs	0.18007230758666 992 sec	0.52595496177 67334 sec

Prepared By :
Muhammad Ali Haider
23047

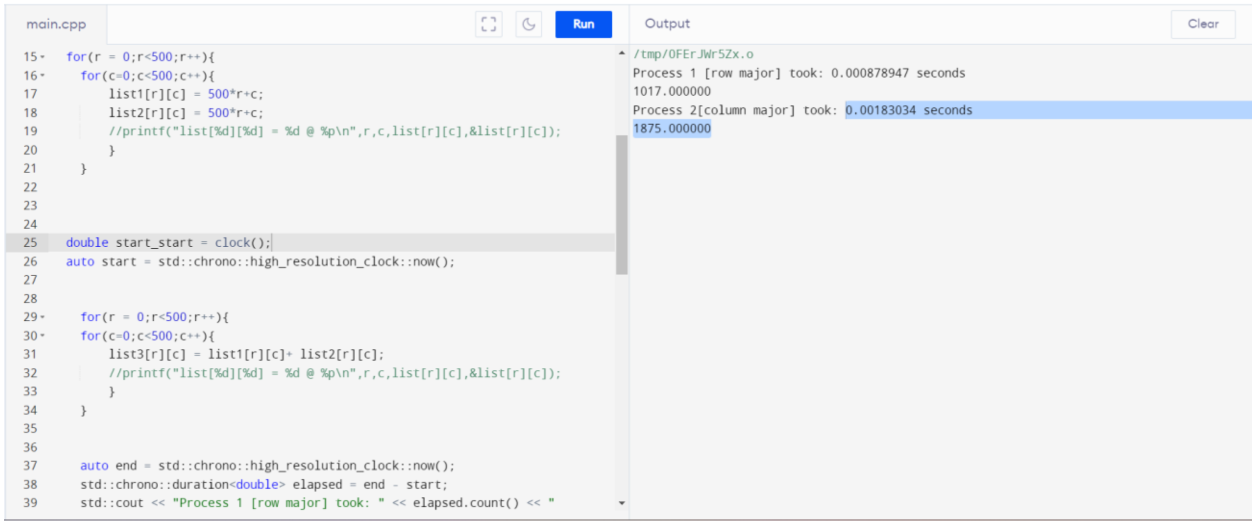
c/c++

Code:

```
#include <iostream>

#include <chrono>

#include <time.h>

using namespace std;

int main() {

  int list1[500][500];

  int list2[500][500];

  int list3[500][500];

  int r,c;

  for(r = 0;r<500;r++){

            for(c=0;c<500;c++){

            list1[r][c] = 500*r+c;

        list2[r][c] = 500*r+c;

        //printf("list[%d][%d] = %d @ %p\n",r,c,list[r][c],&list[r][c]);

            }

            }

  double start_start = clock();

  auto start = std::chrono::high_resolution_clock::now();

    for(r = 0;r<500;r++){

            for(c=0;c<500;c++){

            list3[r][c] = list1[r][c]+ list2[r][c];

        //printf("list[%d][%d] = %d @ %p\n",r,c,list[r][c],&list[r][c]);
            }
            }
    auto end = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double> elapsed = end - start;
    std::cout << "Process 1 [row major] took: " << elapsed.count() << " seconds" << std::endl;

    double end_end =  clock();
    double x = (double) (end_end - start_start);
    printf("%lf \n" , x);

    int list4[500][500];
            int list5[500][500];

     for(r = 0;r<500;r++){
            for(c=0;c<500;c++){
            list4[r][c] = 500*r+c;
        list5[r][c] = 500*r+c;
        //printf("list[%d][%d] = %d @ %p\n",r,c,list[r][c],&list[r][c]);
            }
            }
            start_start = clock();
            start = std::chrono::high_resolution_clock::now();
    for(c = 0;c<500;c++){
            for(r=0;r<500;r++){
            list3[r][c] = list4[r][c]+ list5[r][c];
        //printf("list[%d][%d] = %d @ %p\n",r,c,list[r][c],&list[r][c]);
            }
            }
        end = std::chrono::high_resolution_clock::now();
        std::chrono::duration<double> elapsed1 = end - start;
        std::cout << "Process 2[column major] took: " << elapsed1.count() << " seconds" << std::endl;
        end_end =  clock();
        double x1 = (double) (end_end - start_start);
        printf("%lf \n" , x1);
    }
```

Output

- **Row major and Column major :**


main.cpp

```
for(r = 0;r<500;r++){
  for(c=0;c<500;c++){
      list1[r][c] = 500*r+c;
      list2[r][c] = 500*r+c;
      //printf("list[%d][%d] = %d @ %p\n",r,c,list[r][c],&list[r][c]);
      }
  }



double start_start = clock();
auto start = std::chrono::high_resolution_clock::now();


  for(r = 0;r<500;r++){
  for(c=0;c<500;c++){
      list3[r][c] = list1[r][c]+ list2[r][c];
      //printf("list[%d][%d] = %d @ %p\n",r,c,list[r][c],&list[r][c]);
      }
  }


  auto end = std::chrono::high_resolution_clock::now();
  std::chrono::duration<double> elapsed = end - start;
  std::cout << "Process 1 [row major] took: " << elapsed.count() << "
```

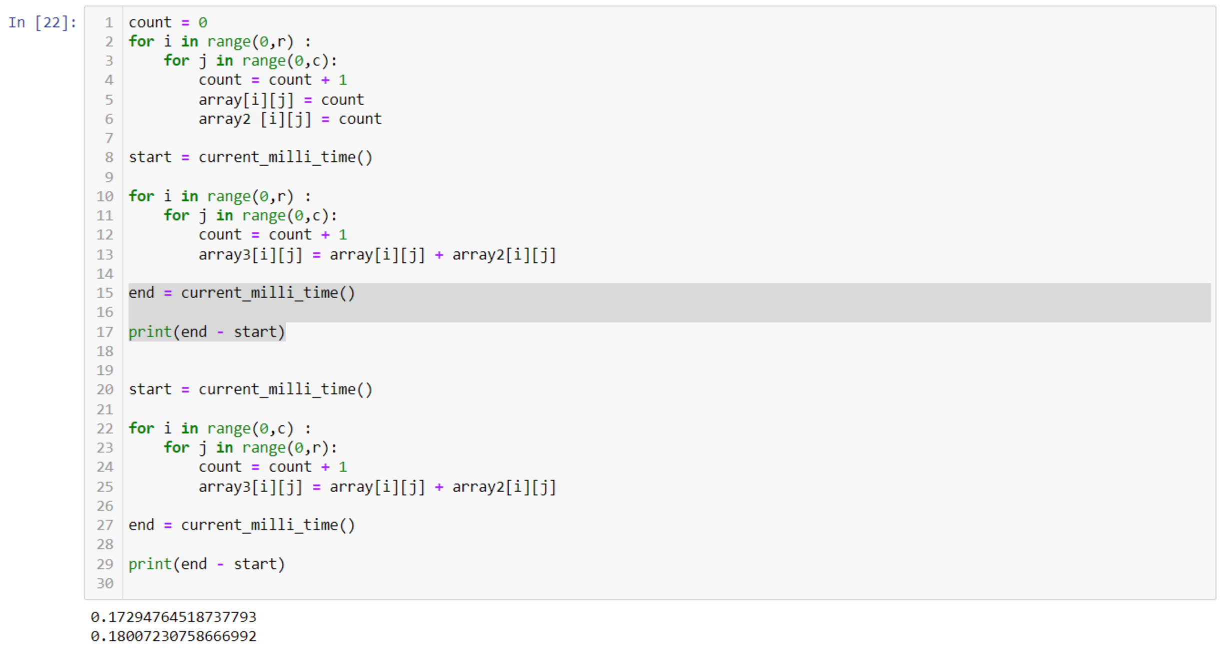
Output

```
/tmp/0FErJWr5Zx.o
Process 1 [row major] took: 0.000878947 seconds
1017.000000
Process 2[column major] took: 0.00183034 seconds
1875.000000
```

Python- list

with loop

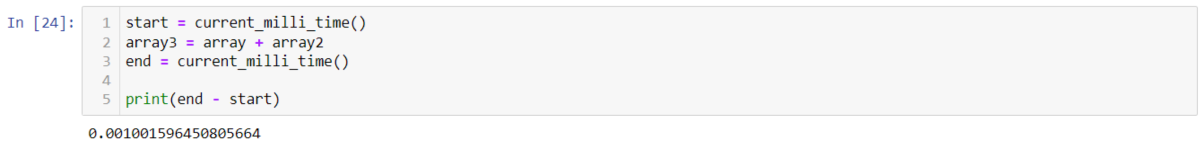
- Row Major and Column Major :



```
In [22]:
count = 0
for i in range(0,r) :
    for j in range(0,c):
        count = count + 1
        array[i][j] = count
        array2 [i][j] = count

start = current_milli_time()

for i in range(0,r) :
    for j in range(0,c):
        count = count + 1
        array3[i][j] = array[i][j] + array2[i][j]

end = current_milli_time()

print(end - start)


start = current_milli_time()

for i in range(0,c) :
    for j in range(0,r):
        count = count + 1
        array3[i][j] = array[i][j] + array2[i][j]

end = current_milli_time()

print(end - start)

```

```
0.17294764518737793
0.18007230758666992
```

Without loop



```
In [24]:
start = current_milli_time()
array3 = array + array2
end = current_milli_time()

print(end - start)
```

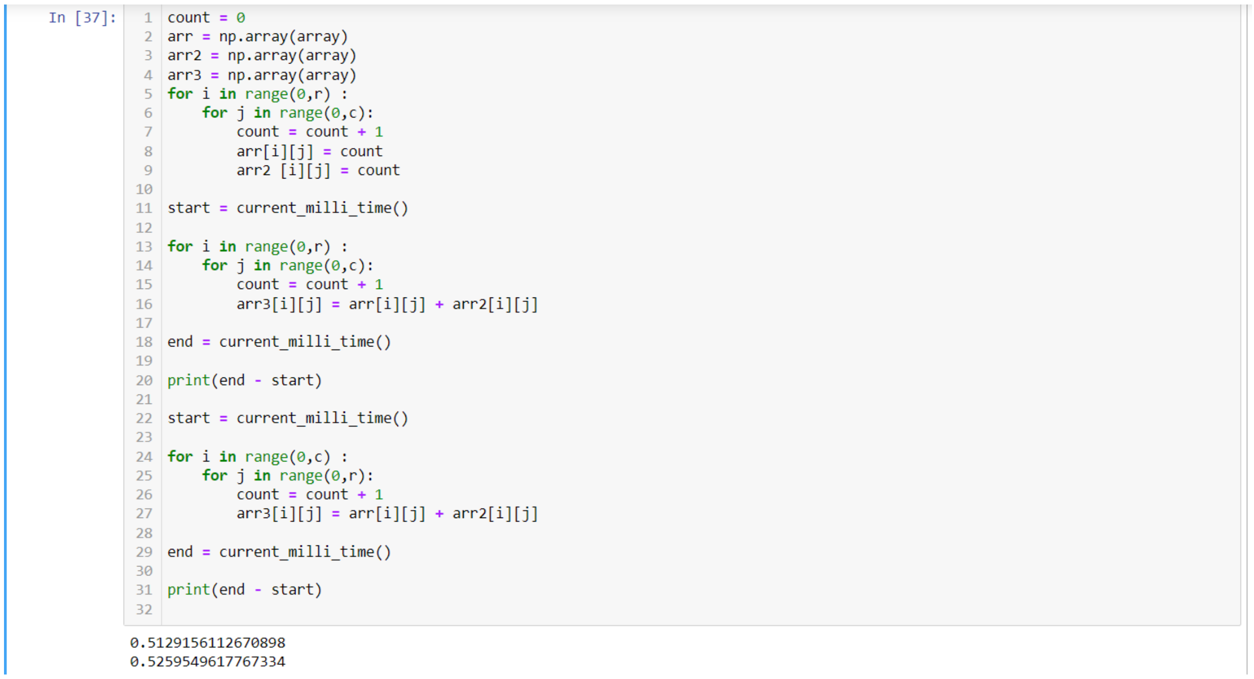
```
0.001001596450805664
```

Prepared By :
Muhammad Ali Haider
23047

Python-Numpy

With loop

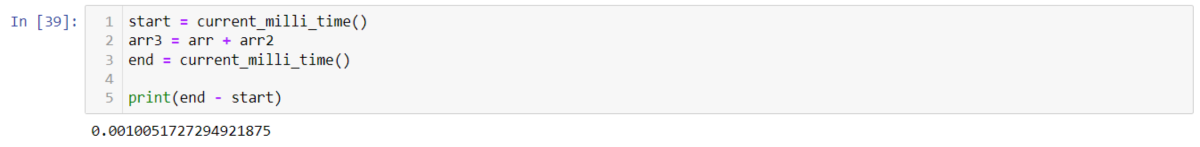
- Row Major and Column Major:



```
In [37]:
count = 0
arr = np.array(array)
arr2 = np.array(array)
arr3 = np.array(array)
for i in range(0,r) :
    for j in range(0,c):
        count = count + 1
        arr[i][j] = count
        arr2 [i][j] = count

start = current_milli_time()

for i in range(0,r) :
    for j in range(0,c):
        count = count + 1
        arr3[i][j] = arr[i][j] + arr2[i][j]

end = current_milli_time()

print(end - start)

start = current_milli_time()

for i in range(0,c) :
    for j in range(0,r):
        count = count + 1
        arr3[i][j] = arr[i][j] + arr2[i][j]

end = current_milli_time()

print(end - start)
```

```
0.5129156112670898
0.5259549617767334
```

Without loop



```
In [39]:
start = current_milli_time()
arr3 = arr + arr2
end = current_milli_time()

print(end - start)
```

```
0.0010051727294921875
```

Prepared By :
Muhammad Ali Haider
23047

Java

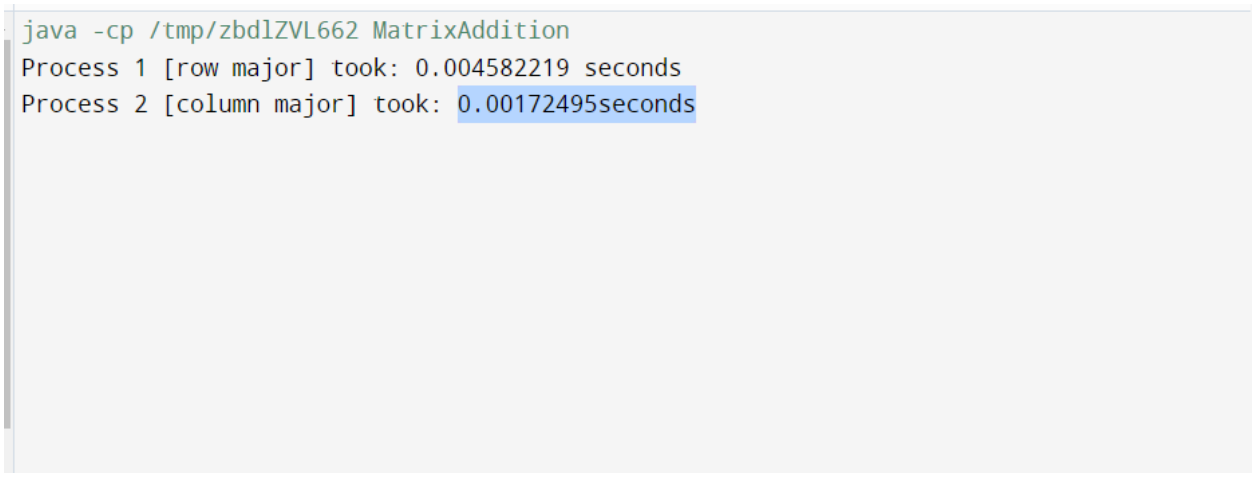
code

```
import java.util.Date;


public class MatrixAddition {
    public static void main(String[] args) {
        int[][] list1 = new int[500][500];
        int[][] list2 = new int[500][500];
        int[][] list3 = new int[500][500];

        for (int r = 0; r < 500; r++) {
            for (int c = 0; c < 500; c++) {
                list1[r][c] = 500 * r + c;
                list2[r][c] = 500 * r + c;
            }
        }

        long start = System.nanoTime();

        for (int r = 0; r < 500; r++) {
            for (int c = 0; c < 500; c++) {
                list3[r][c] = list1[r][c] + list2[r][c];
            }
        }
```

```
long end = System.nanoTime();
double elapsed = end - start;
System.out.println("Process 1 [row major] took: " + (elapsed/1000000000) + " seconds");

int[][] list4 = new int[500][500];
int[][] list5 = new int[500][500];
// System.out.println( System.nanoTime());
for (int r = 0; r < 500; r++) {
   for (int c = 0; c < 500; c++) {
      list4[r][c] = 500 * r + c;
      list5[r][c] = 500 * r + c;
   }
}

start = System.nanoTime();

for (int c = 0; c < 500; c++) {
   for (int r = 0; r < 500; r++) {
      list3[r][c] = list4[r][c] + list5[r][c];
   }
}

end = System.nanoTime();
elapsed = end - start;

// System.out.println( System.nanoTime());
```

```
        System.out.println("Process 2 [column major] took: " + (elapsed/1000000000) +
"seconds");
    }
}
```

Output Java

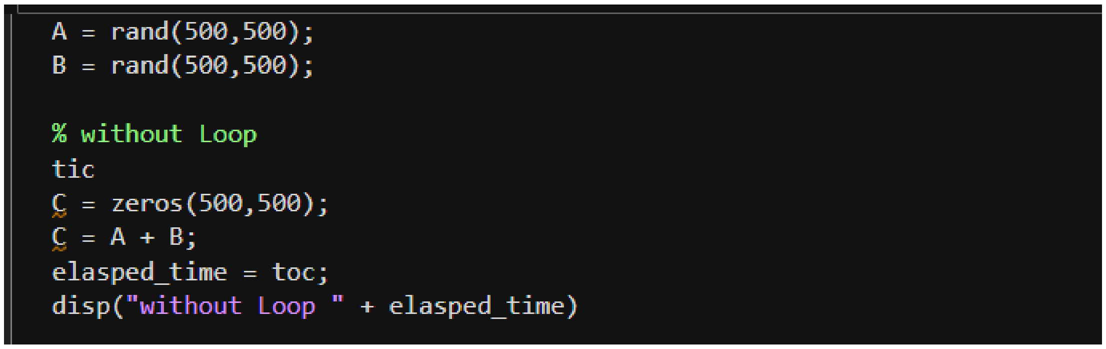


```
java -cp /tmp/zbdlZVL662 MatrixAddition
Process 1 [row major] took: 0.004582219 seconds
Process 2 [column major] took: 0.00172495seconds
```

Prepared By :
Muhammad Ali Haider
23047

Matlab

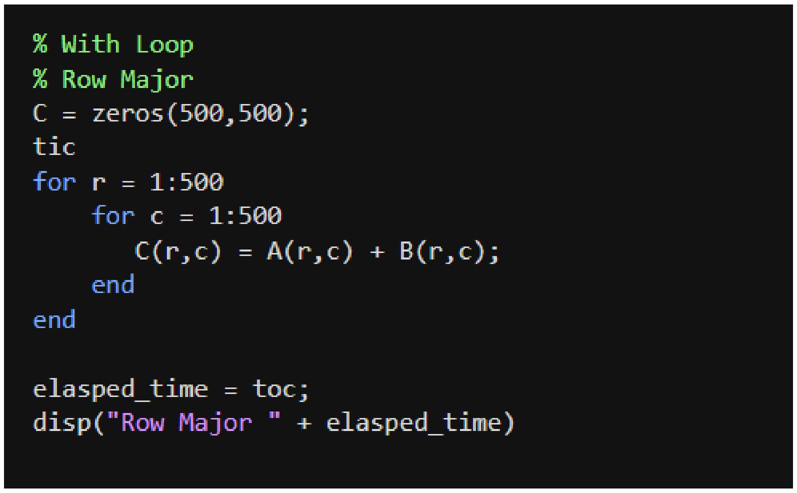
without Loop



```
A = rand(500,500);
B = rand(500,500);

% without Loop
tic
C = zeros(500,500);
C = A + B;
elasped_time = toc;
disp("without Loop " + elasped_time)
```

With loop

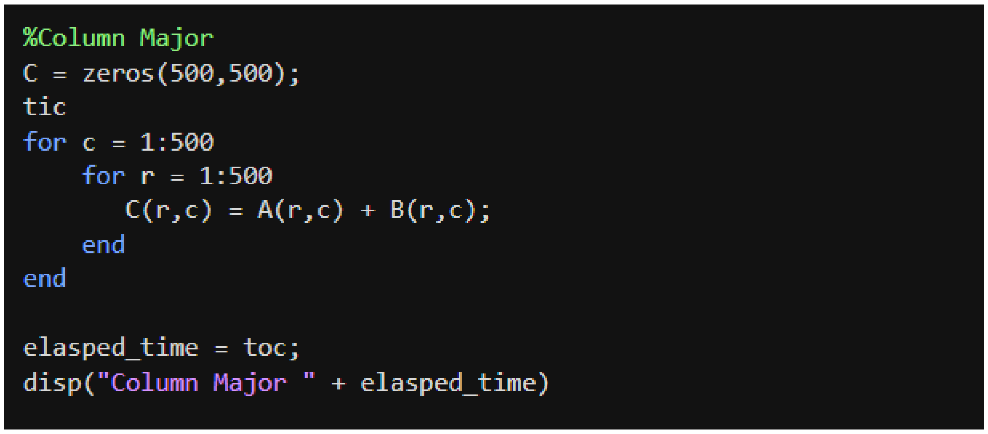
- Row Major



```
% With Loop
% Row Major
C = zeros(500,500);
tic
for r = 1:500
    for c = 1:500
       C(r,c) = A(r,c) + B(r,c);
    end
end

elasped_time = toc;
disp("Row Major " + elasped_time)
```

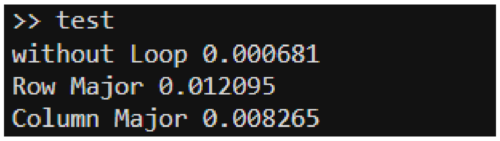
Prepared By :
Muhammad Ali Haider
23047

- Column Major



```
%Column Major
C = zeros(500,500);
tic
for c = 1:500
    for r = 1:500
       C(r,c) = A(r,c) + B(r,c);
    end
end

elasped_time = toc;
disp("Column Major " + elasped_time)
```

Output



```
>> test
without Loop 0.000681
Row Major 0.012095
Column Major 0.008265
```

Prepared By :
Muhammad Ali Haider
23047