

Task 3: Loop Ordering and Matrix Multiplication

1. Which 2 orderings perform best for these 1000-by-1000 matrices? Write your answer in the form "[Ordering1], [Ordering2]" (e.g. "ijk, ijk").

“jki” and “kji” are the 2 orders that performs best for these 1000*1000 matrices.

Reasoning:

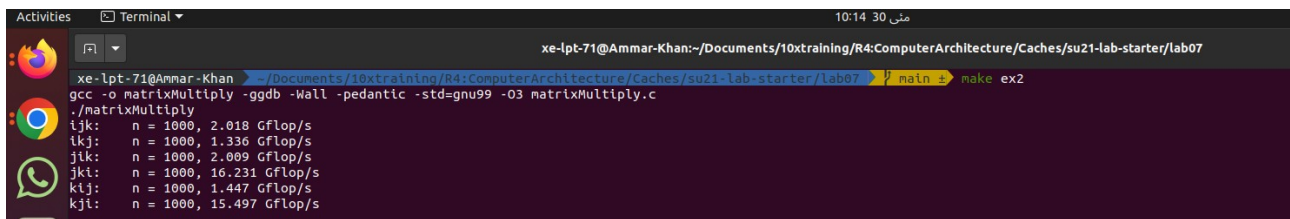
From the snapshot below we can see that for a 1000*1000 matrix the “Gflops/s” *Giga-floating point-operations per second* for the order “jki” and “kji” is greater than the other 4 possible combinations, also for *the innermost loop (the one that increments i)* the stride for **A and C is 1 but for B is 0**. Here the stride is the *minimum distance that needs to be covered in memory to move from one element to the next along that dimension* since the stride is minimum for these two cases that’s why these two orders performs the best for 1000*1000 matrices.

2. Which 2 orderings perform the worst?

“ikj” and “kij” are the 2 orders that performs the worst.

Reasoning:

If we examine the *innermost loop (the one that increments j)* the stride for the **A is 0 but for B and C it is n**, also the Gflops/s for the order “ikj” and “kij” is the minimum than the other 4 possible combinations, as shown below



```
xe-lpt-71@Ammar-Khan:~/Documents/10xtraining/R4:ComputerArchitecture/Caches/su21-lab-starter/lab07
gcc -o matrixMultiply -ggdb -Wall -pedantic -std=gnu99 -O3 matrixMultiply.c
./matrixMultiply
ijk:  n = 1000, 2.018 Gflop/s
ikj:  n = 1000, 1.336 Gflop/s
jik:  n = 1000, 2.009 Gflop/s
jki:  n = 1000, 16.231 Gflop/s
kij:  n = 1000, 1.447 Gflop/s
kji:  n = 1000, 15.497 Gflop/s
```