parallelisation (thread-wise) of LAB 2 :

To execute, make sure you have the NASM and GCC compilers installed :

Sudo apt-get install nasm

Then make sure you have the image called “test.raw” in the same directory as the source files.

Then run the following :

nasm -f elf64 treshold_lab1.ass && gcc -o3 -g treshold_lab1.o ccode_lab1.c -lpthread -o res1 && ./res1

results :

```
Number of threads : 1
        execution time in C : 0.002909
        execution time in SIMD : 0.000225
        execution time in SIMD AVX : 0.000302
Number of threads : 2
        execution time in C : 0.001328
        execution time in SIMD : 0.000128
        execution time in SIMD AVX : 0.000234
Number of threads : 4
        execution time in C : 0.002684
        execution time in SIMD : 0.000087
        execution time in SIMD AVX : 0.000088
Number of threads : 8
        execution time in C : 0.000640
        execution time in SIMD : 0.000137
        execution time in SIMD AVX : 0.000127
Number of threads : 16
        execution time in C : 0.000975
        execution time in SIMD : 0.000617
        execution time in SIMD AVX : 0.000379
Number of threads : 32
        execution time in C : 0.000858
        execution time in SIMD : 0.000684
        execution time in SIMD AVX : 0.001088
Number of threads : 64
        execution time in C : 0.002667
        execution time in SIMD : 0.002261
        execution time in SIMD AVX : 0.001650
```

parallelisation (thread-wise) of LAB 3 :

run using the following code :

```
nasm -f elf64 assembly_lab2.ass && gcc -o3 -g assembly_lab2.o
ccode_lab2.c -lpthread -o res2 && ./res2
```

results :

```
Number of threads : 1
        3X3     execution time in C : 0.065308
                execution time in SIMD : 0.000600
        5X5     execution time in C : 0.159490
                execution time in SIMD : 0.000535
Number of threads : 2
        3X3     execution time in C : 0.035923
                execution time in SIMD : 0.000229
        5X5     execution time in C : 0.085099
                execution time in SIMD : 0.000270
Number of threads : 4
        3X3     execution time in C : 0.016770
                execution time in SIMD : 0.000111
        5X5     execution time in C : 0.041721
                execution time in SIMD : 0.000398
Number of threads : 8
        3X3     execution time in C : 0.007936
                execution time in SIMD : 0.000270
        5X5     execution time in C : 0.022375
                execution time in SIMD : 0.000231
Number of threads : 16
        3X3     execution time in C : 0.004544
                execution time in SIMD : 0.000611
        5X5     execution time in C : 0.008778
                execution time in SIMD : 0.000611
Number of threads : 32
        3X3     execution time in C : 0.002662
                execution time in SIMD : 0.000872
        5X5     execution time in C : 0.004340
                execution time in SIMD : 0.000428
Number of threads : 64
        3X3     execution time in C : 0.001666
                execution time in SIMD : 0.000853
        5X5     execution time in C : 0.004658
                execution time in SIMD : 0.000951
```

important conclusion :
in SIMD, when using a number of threads equal to that of the number of the cores (of the system on which the implementation is running), we get the best performance. The reason for that is that each core has it's own SIMD registers xmm0—xmm16. So if we have more threads than cores, these registers have to be shared (for each thread switch they have to be pushed onto the stack and then back in), and so this kills performance. If we have the same number of cores than threads however, no thread switching is needed so the maximum performance is achieved.

in C however, more we increase the number of threads, better performance we will get (until a certain limit of course) and this is basically because when C is converted into assembly by GCC, it mainly uses the stack to save it's data and local variables.