ELEC-H-473 — Exercise 02

Max/Min SIMD functions and morphological image filtering (grey level images)

1. Implementation in SIMD

Implement in C and C + SIMD assembly the Min/Max filtering for the image.

The neighbourhood should be fixed to 3X3.

If you have time you can try to implement 5x5 and 7x7 (it is the extension of the 3x3). This could give you extra points ...

Start from the program that you have developed in the previous lab. Use lecture notes to understand the algorithm. Before programming think on how you would solve this problem on paper. The idea is to look for the vector operation that could perform required computation.

Some useful SIMD operations (google for the others):

- `pand xmm0, xmm1`
  - Logical AND operation between 2 registers
- `pcmpeqb xmm0, xmm1`
  - Compares 2 values in mode BYTE. If TRUE, all bits of the destination register are set to 1, 0 otherwise
- `pminub xmm0, xmm1 (ou pmaxub)`
  - Performs the comparison in en mode BYTE between xmm0, xmm1. Puts minumun of tw in `xmm0`.

Help

Few indications:

- Time measurement: *"Retrieves the current value of the performance counter, which is a high resolution (<1us) time stamp that can be used for time-interval measurements."*

  Parameters
  lpPerformanceCount [out]
  A pointer to a variable that receives the current performance-counter value, in counts.

  Return value
  If the function succeeds, the return value is nonzero.
  If the function fails, the return value is zero. To get extended error information, call GetLastError. On systems that run Windows XP or later, the function will always succeed and will thus never return zero.

```
#include <windows.h>

double PCFreq = 0.0;
__int64 CounterStart = 0;

void StartCounter()
{
    LARGE_INTEGER li;
    if(!QueryPerformanceFrequency(&li))
     cout << "QueryPerformanceFrequency failed!\n";

    PCFreq = double(li.QuadPart)/1000.0;

    QueryPerformanceCounter(&li);
    CounterStart = li.QuadPart;
}
double GetCounter()
{
    LARGE_INTEGER li;
    QueryPerformanceCounter(&li);
    return double(li.QuadPart-CounterStart)/PCFreq;
}

int main()
{
    StartCounter();
    Sleep(1000);
    cout << GetCounter() <<"\n";
    return 0;
}
{
…
_asm{
// Code assembleur SIMD y compris
…
label1 :
…
sub ecx , 1 ;
jnz label1
}
…
emms;
}
```