1. Krótki opis celu implementacji algorytmu
   Algorytm ma na celu wykrywanie krawędzi pionowych. Najpierw obraz jest zamieniany na szary a następnie aplikowany jest filtr 3x3

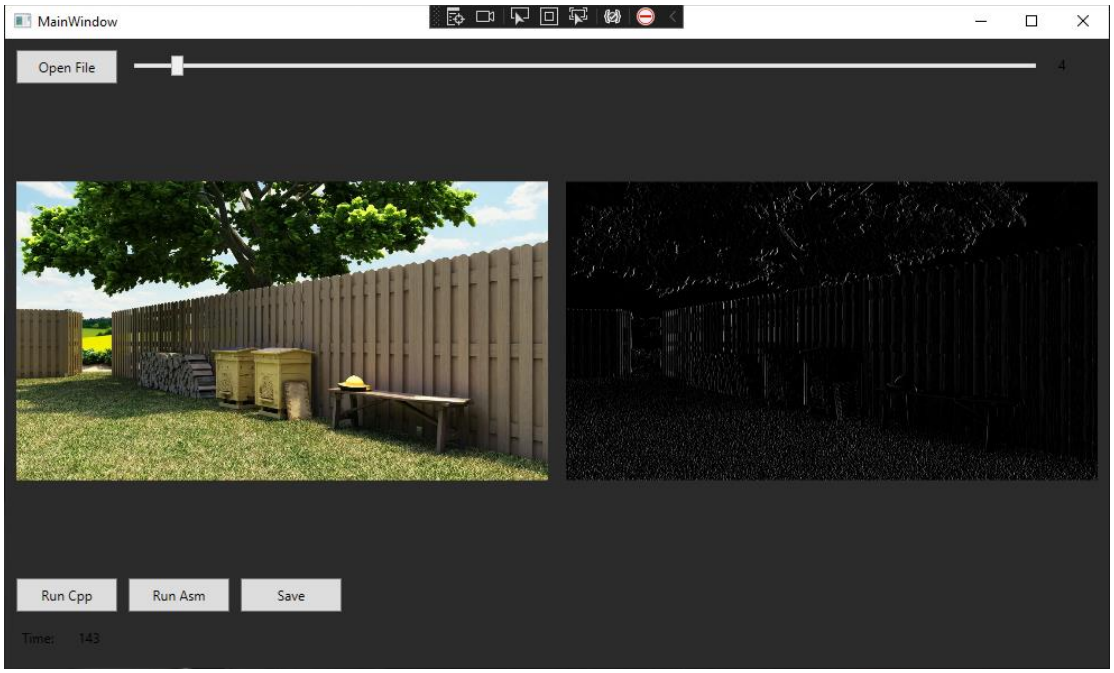
-1	0	1
-1	0	1
-1	0	1

2. Opis parametrów wejściowych
   Funkcja cpp oraz asm posiada dokładnie takie same parametry wejściowe
   (byte* InputArray, byte* OutputArray, int width, int height, int start, int stop)
   InputArray – wskaźnik na tablicę wejściową z pixelami
   OutputArray – wskaźnik na tablicę wyjściową dla obrazu mniejszego o 2x2 pixele
   width – ilość pixeli w wierszu
   height – ilość pixeli w kolumnie
   start – numer wiersza w którym funkcja zaczyna
   stop – numer wiersza w którym funkcja kończy (pobierze jeszcze dwa następne wiersze)
3. Opis wybranego fragmentu asemblerowego kodu

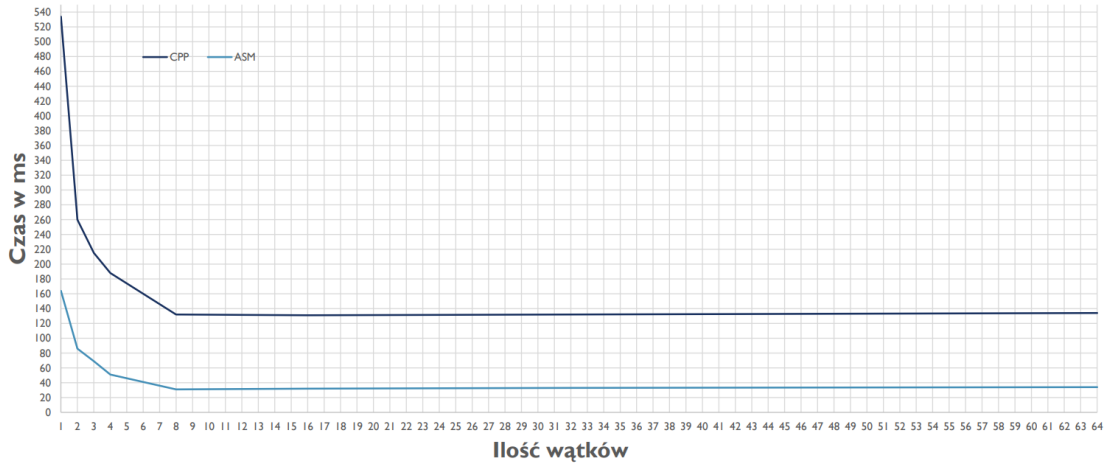
```
;;;;;;;;;;;; LOAD FIRST ROW ;;;;;;;;;;;;
; wczytanie 16 bajtów (5 pixeli) obrazka z wiersza pierwszego, RBX – wskaźnik na początek wiersza tablicy wejściowej
MOVUPS   XMM0, XMMWORD PTR [RBX]
; przemieszczenie czwartego DOUBLE WORDA na pozycję pierwszego DOUBLE WORDA lewej połówki YMM
VPERMD   YMM0, YMM15, YMM0
; wyizolowanie kanału Blue (5x DOUBLE WORD)
VPSHUFB  YMM2, YMM0,  YMM14
; wyizolowanie kanału Green (5x DOUBLE WORD)
VPSHUFB  YMM1, YMM0,  YMM13
; wyizolowanie kanału Red (5x DOUBLE WORD)
VPSHUFB  YMM0, YMM0,  YMM12

;;;;;;;;;;;; LOAD SECOND ROW ;;;;;;;;;;;;
; wczytanie 16 bajtów (5 pixeli) obrazka z wiersza drugiego, RBX – wskaźnik na początek wiersza tablicy wejściowej, R14 stride
MOVUPS   XMM3, XMMWORD PTR [RBX + R14]
; przemieszczenie czwartego DOUBLE WORDA na pozycję pierwszego DOUBLE WORDA lewej połówki YMM
VPERMD   YMM3, YMM15, YMM3
; wyizolowanie kanału Blue (5x DOUBLE WORD)
VPSHUFB  YMM5, YMM3,  YMM14
; wyizolowanie kanału Green (5x DOUBLE WORD)
VPSHUFB  YMM4, YMM3,  YMM13
; wyizolowanie kanału Red (5x DOUBLE WORD)
VPSHUFB  YMM3, YMM3,  YMM12

;;;;;;;;;;;; LOAD THIRD ROW ;;;;;;;;;;;;
; wczytanie 16 bajtów (5 pixeli) obrazka z wiersza trzeciego, RBX – wskaźnik na początek wiersza tablicy wejściowej, R14 stride
MOVUPS   XMM6, XMMWORD PTR [RBX + R14 * 2]
; przemieszczenie czwartego DOUBLE WORDA na pozycję pierwszego DOUBLE WORDA lewej połówki YMM
VPERMD   YMM6, YMM15, YMM6
; wyizolowanie kanału Blue (5x DOUBLE WORD)
VPSHUFB  YMM8, YMM6,  YMM14
; wyizolowanie kanału Green (5x DOUBLE WORD)
VPSHUFB  YMM7, YMM6,  YMM13
; wyizolowanie kanału Red (5x DOUBLE WORD)
VPSHUFB  YMM6, YMM6,  YMM12

;;;;;;;;;;;;;;;; SUM ;;;;;;;;;;;;;;;;
; dodanie DOUBLE WORDÓW
VPADDD   YMM0, YMM0, YMM1
; dodanie DOUBLE WORDÓW
VPADDD   YMM0, YMM0, YMM2
; dodanie DOUBLE WORDÓW
VPADDD   YMM0, YMM0, YMM3
; dodanie DOUBLE WORDÓW
VPADDD   YMM0, YMM0, YMM4
```

```
; dodanie DOUBLE WORDÓW
VPADDD   YMM0, YMM0, YMM5
; dodanie DOUBLE WORDÓW
VPADDD   YMM0, YMM0, YMM6
; dodanie DOUBLE WORDÓW
VPADDD   YMM0, YMM0, YMM7
; dodanie DOUBLE WORDÓW
VPADDD   YMM0, YMM0, YMM8

;;;;;;;;;;;;; DIVIDE BY 9 ;;;;;;;;;;;;;;
; przesnięcie bitowe w lewo o 3 w DOUBLE WORDACH
VPSLLD   YMM1, YMM0, 3
; różnica DOUBLE WORDÓW
VPSUBD   YMM0, YMM1, YMM0
; przesnięcie bitowe w lewo o 6 w DOUBLE WORDACH
VPSLLD   YMM1, YMM0, 6
; dodanie DOUBLE WORDÓW
VPADDD   YMM0, YMM0, YMM1
; dodanie do DOUBLE WORDÓW stałej
VPADDD   YMM0, YMM0, YMM11
; przesnięcie bitowe w prawo o 12 w DOUBLE WORDACH
VPSRLD   YMM0, YMM0, 12

;;;;;;;;;;;;; APPLY FILTER ;;;;;;;;;;;;;
; przesuniecie o 2 DOUBLE WORDY w prawo aby ustawić lewą kolumnę filtru nad prawą
VPERMD   YMM1, YMM10, YMM0
; odjęcie od prawej części lewej w celu realizacji filtru pionowego
VPSUBD   YMM0, YMM1,  YMM0

;;;;;;;;;;;;;;;; RELU ;;;;;;;;;;;;;;;;
; zerowanie YMM8
VPXOR    YMM8, YMM8, YMM8
; zamienienie ujemnych DOUBLE WORDÓW na 0
VPMAXSD  YMM0, YMM0, YMM8

;;;;;;;;;;;;;;;; STORE ;;;;;;;;;;;;;;;;
; stworzenie 3 pixeli (9 bajtów)
VPSHUFB  YMM0, YMM0, YMM9
; zapisanie QUAD WORDA (8 bajtów) do tablicy wyjściowej, RCX – wskaźnik na początek wiersza tablicy wyjściowej
MOVLPD   QWORD PTR [RCX], XMM0
; zapisanie dziewiątego bajta do tablicy wyjściowej, RCX – wskaźnik na początek wiersza tablicy wyjściowej
PEXTRB   BYTE PTR [RCX + 8], XMM0, 8
```

4. Zrzuty ekranu interfejsu



5. Raport szybkości (Obraz 8k)



6. Opis testowania
   Aplikację testowano na 3 różnych obrazach BMP (24-bit) oraz na jednym wadliwym pliku. Wielokrotnie uruchamiano funkcję cpp i asm dla różnej ilości wątków. Aplikację uruchamiano poprzez plik .exe lub w Visual Studio w trybie release
7. Wnioski
   Dla obrazu 8k osiągnięto maksymalne przyspieszenie na korzyść ASM: 4.258x, natomiast średnie przyspieszenie wyniosło: 3.672x. Dużo wygodniejsze i szybsze byłoby zastosowanie bit mapy 32-bitowej przy używaniu instrukcji wektorowych. Problematyczny również okazał się brak możliwości przenoszenia BAJTÓW oraz WORDÓW między połówkami rejestrów YMM.