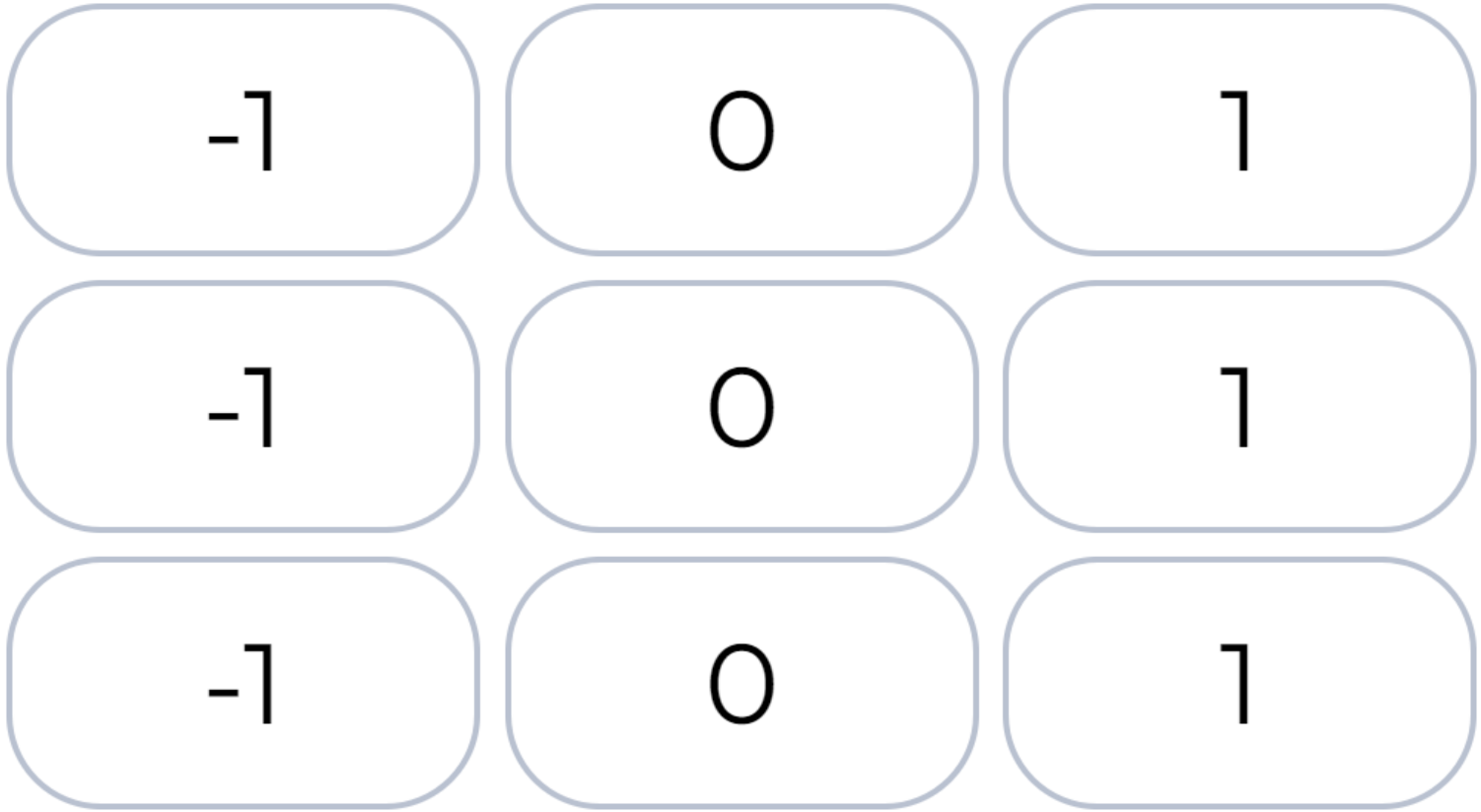
PROJEKT JA

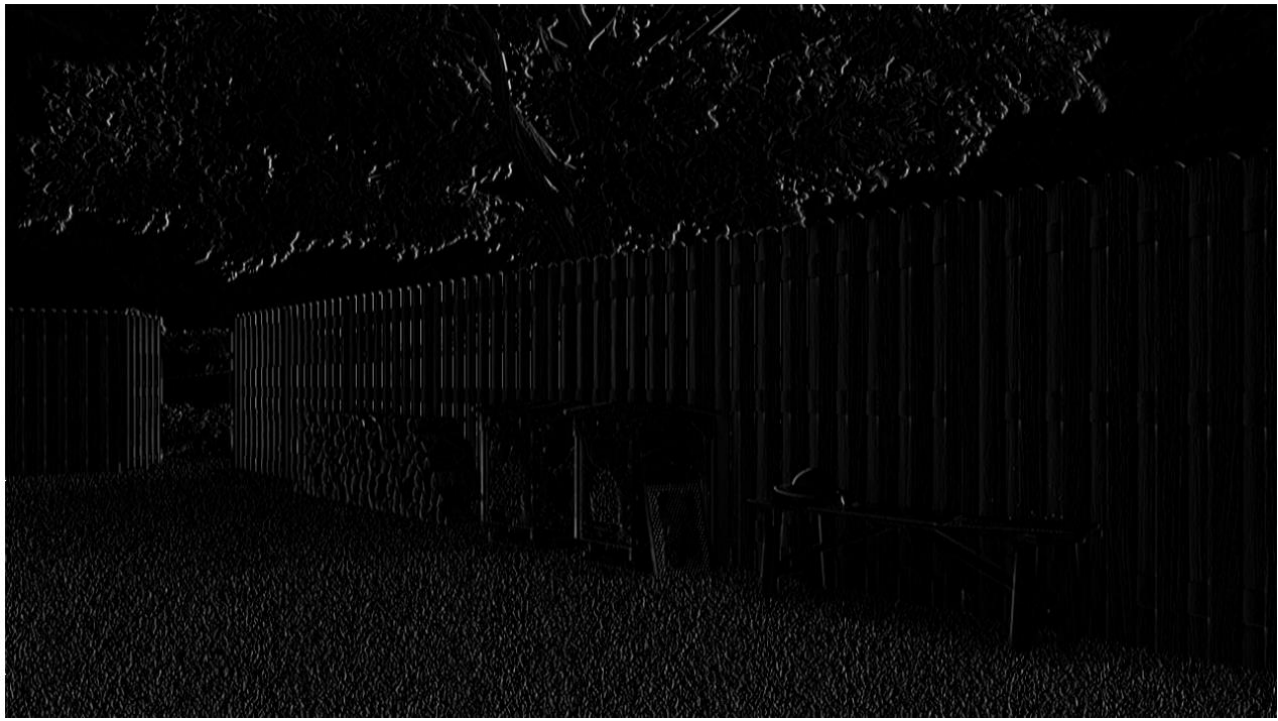
PORÓWNANIE CZASÓW PRZETWARZANIA OBRAZU W C++ ORAZ ASM

Mateusz Kucharczyk
matekuc688@student.polsl.pl

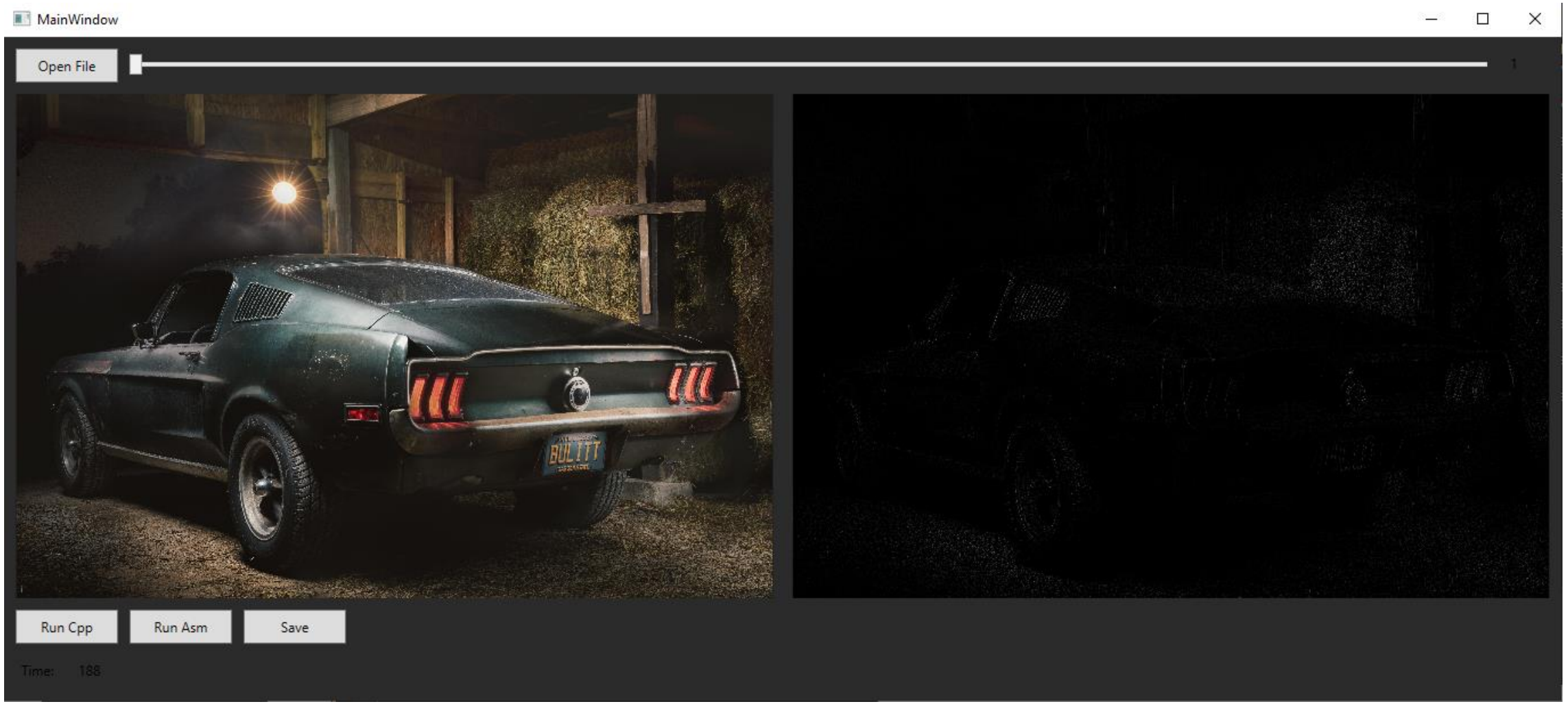


FILTR PIONOWY

PRZYKŁAD



GUI



ZAWIERA

- Przycisk „Open File” – do otwierania obrazów bmp
- Przycisk „Run Cpp” – do uruchomienia przetwarzania za pomocą Cpp
- Przycisk „Run Asm” – do uruchomienia przetwarzania za pomocą Asm
- Przycisk „Save” – do zapisywania wynikowego obrazu
- Slider – do ustawiania ilości wątków
- Obraz wejściowy i przetworzony
- Informacje o czasie wykonania lub błędzie

KOD CPP

```cpp
void passImageToCpp(unsigned char* inputArray, unsigned char* outputArray, int width, int height, int start, int stop) {
    int filter[3][3] = { {-1, 0, 1}, {-1, 0, 1}, {-1, 0, 1} };
    int realInputWidth = ((width + 1) * 3) & ~3;
    int realOutputWidth = ((width - 1) * 3) & ~3;

    for (int y = start; y < stop; y++) {
        for (int x = 0; x < width - 2; x++) {
            int sum = 0;
            for (int y1 = 0; y1 < 3; y1++) {
                for (int x1 = 0; x1 < 3; x1++) {
                    int B = inputArray[(y + y1) * realInputWidth + (x + x1) * 3 + 0];
                    int G = inputArray[(y + y1) * realInputWidth + (x + x1) * 3 + 1];
                    int R = inputArray[(y + y1) * realInputWidth + (x + x1) * 3 + 2];
                    int S = (R + G + B) / 9;
                    sum += S * filter[y1][x1];
                }
            }
            if (sum < 0) { sum = 0; }
            outputArray[y * realOutputWidth + x * 3 + 0] = (unsigned char) sum;
            outputArray[y * realOutputWidth + x * 3 + 1] = (unsigned char) sum;
            outputArray[y * realOutputWidth + x * 3 + 2] = (unsigned char) sum;
        }
    }
}
```

KOD ASM - INICJALIZACJA

```
10      .CODE
11          passImageToAsm PROC
12              PUSH RBX
13
14              ;;;;;;;;;;;; GET ARGUMENTS ;;;;;;;;;;;;
15              ADD RSP, 030H
16              MOV RBX, 000000000FFFFFFFFH
17              POP R10
18              AND R10, RBX
19              POP R11
20              AND R11, RBX
21              SUB RSP, 040H
22              MOV R12, RCX
23              MOV R13, RDX
24
25              ;;;; CALCULATE INPUT, OUTPUT STRIDE ;;;;
26              MOV RBX, 0FFFFFFFFFFFFFFCH
27              MOV RAX, 3
28              INC R8
29              MUL R8
30              AND RAX, RBX
31              MOV R14, RAX
32              SUB R8, 2
33              MOV RAX, 3
34              MUL R8
35              AND RAX, RBX
36              MOV R15, RAX
```

```
38              ;;;;;;; INITIALIZE YMM REGISTERS ;;;;;;;
39              VMOVUPS YMM15, YMMWORD PTR [SHUF1]
40              VMOVUPS YMM14, YMMWORD PTR [SHUFB]
41              VMOVUPS YMM13, YMMWORD PTR [SHUFG]
42              VMOVUPS YMM12, YMMWORD PTR [SHUFR]
43              VMOVUPS YMM11, YMMWORD PTR [QQQQQ]
44              VMOVUPS YMM10, YMMWORD PTR [SHUFF]
45              VMOVUPS YMM9,  YMMWORD PTR [SHUFO]
46              VPXOR   YMM0, YMM0, YMM0
47              VPXOR   YMM3, YMM3, YMM3
48              VPXOR   YMM6, YMM0, YMM6
49
50              ;;;;;;; CALCULATE OUTPUT POINTER ;;;;;;;
51              MOV RAX, R10
52              MUL R15
53              ADD R13, RAX
54
55              ;;;; CALCULATE INPUT BEGIN POINTER ;;;;;
56              MOV RAX, R10
57              MUL R14
58              ADD RAX, R12
59              MOV R10, RAX
60
61              ;;;;; CALCULATE INPUT END POINTER ;;;;;;
62              MOV RAX, R11
63              MUL R14
64              ADD RAX, R12
65              MOV R11, RAX
66
67              ;;;;;;;;; CALCULATE END OF ROW ;;;;;;;;;
68              MOV RAX, 3
69              MUL R8
70              SUB RAX, 12
71              ADD RAX, R10
```

KOD ASM – GŁÓWNA PĘTLA

```
72          L1:
73              ;;;;; SET POINTER TO BEGIN OF ROW ;;;;;;
74              MOV RBX, R10
75              MOV RCX, R13
76              L2:
77                  ;;;;;;;;;;;; LOAD FIRST ROW ;;;;;;;;;;;;
78                  MOVUPS   XMM0, XMMWORD PTR [RBX]
79                  VPERMD   YMM0, YMM15, YMM0
80                  VPSHUFB  YMM2, YMM0,  YMM14
81                  VPSHUFB  YMM1, YMM0,  YMM13
82                  VPSHUFB  YMM0, YMM0,  YMM12
83
84                  ;;;;;;;;;;; LOAD SECOND ROW ;;;;;;;;;;;;
85                  MOVUPS   XMM3, XMMWORD PTR [RBX + R14]
86                  VPERMD   YMM3, YMM15, YMM3
87                  VPSHUFB  YMM5, YMM3,  YMM14
88                  VPSHUFB  YMM4, YMM3,  YMM13
89                  VPSHUFB  YMM3, YMM3,  YMM12
90
91                  ;;;;;;;;;;;; LOAD THIRD ROW ;;;;;;;;;;;;
92                  MOVUPS   XMM6, XMMWORD PTR [RBX + R14 * 2]
93                  VPERMD   YMM6, YMM15, YMM6
94                  VPSHUFB  YMM8, YMM6,  YMM14
95                  VPSHUFB  YMM7, YMM6,  YMM13
96                  VPSHUFB  YMM6, YMM6,  YMM12
97
98                  ;;;;;;;;;;;;;;;; SUM ;;;;;;;;;;;;;;;;;
99                  VPADDD   YMM0, YMM0, YMM1
100                 VPADDD   YMM0, YMM0, YMM2
101                 VPADDD   YMM0, YMM0, YMM3
102                 VPADDD   YMM0, YMM0, YMM4
103                 VPADDD   YMM0, YMM0, YMM5
104                 VPADDD   YMM0, YMM0, YMM6
105                 VPADDD   YMM0, YMM0, YMM7
106                 VPADDD   YMM0, YMM0, YMM8
```

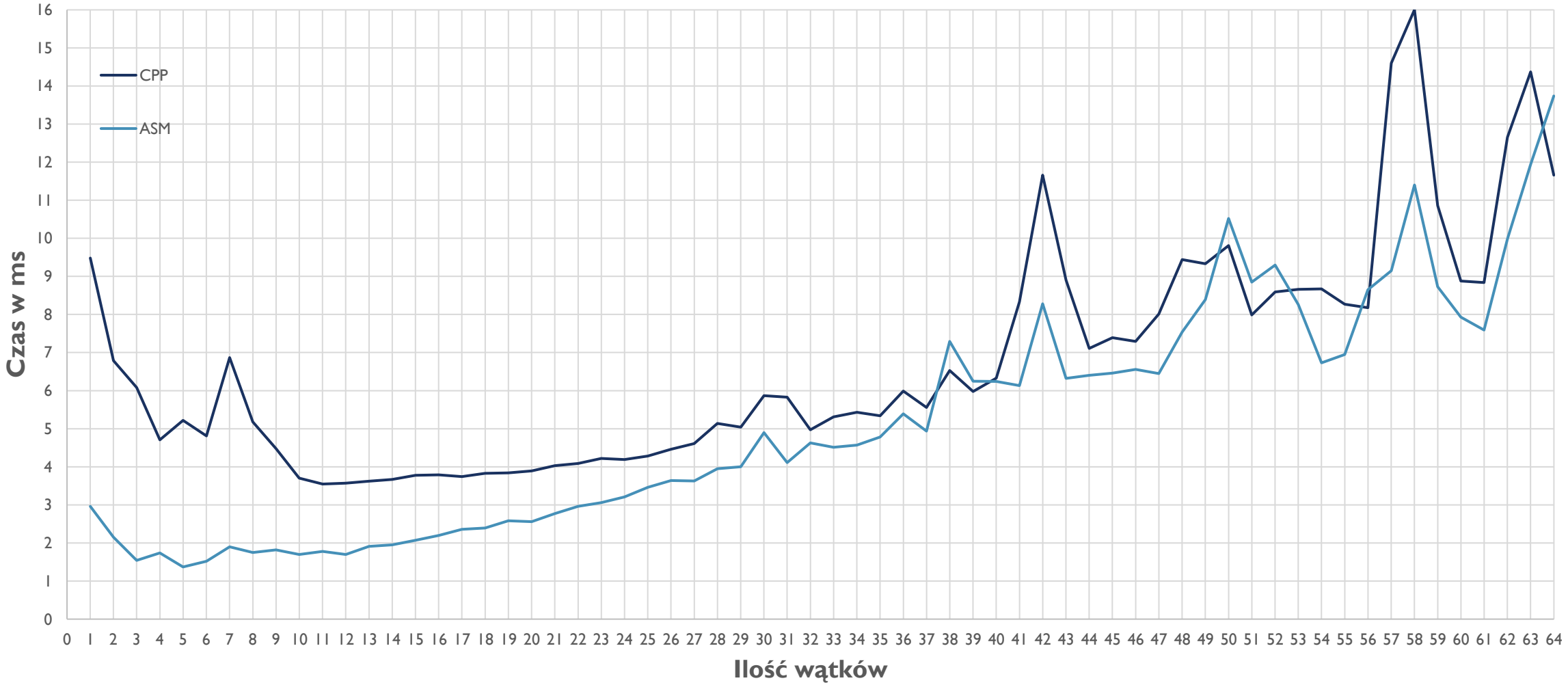
```
108                 ;;;;;;;;;;;;; DIVIDE BY 9 ;;;;;;;;;;;;;
109                 VPSLLD   YMM1, YMM0, 3
110                 VPSUBD   YMM0, YMM1, YMM0
111                 VPSLLD   YMM1, YMM0, 6
112                 VPADDD   YMM0, YMM0, YMM1
113                 VPADDD   YMM0, YMM0, YMM11
114                 VPSRLD   YMM0, YMM0, 12
115
116                 ;;;;;;;;;;;;; APPLY FILTER ;;;;;;;;;;;;;
117                 VPERMD   YMM1, YMM10, YMM0
118                 VPSUBD   YMM0, YMM1,  YMM0
119
120                 ;;;;;;;;;;;;;;;; RELU ;;;;;;;;;;;;;;;;
121                 VPXOR    YMM8, YMM8, YMM8
122                 VPMAXSD  YMM0, YMM0, YMM8
123
124                 ;;;;;;;;;;;;;;;; STORE ;;;;;;;;;;;;;;;;
125                 VPSHUFB  YMM0, YMM0, YMM9
126                 MOVLPD   QWORD PTR [RCX], XMM0
127                 PEXTRB   BYTE PTR [RCX + 8], XMM0, 8
128
129                 ;;;;;;;;;; NEXT SLICE IN ROW ;;;;;;;;;;
130                 ADD RBX, 9
131                 ADD RCX, 9
132                 CMP RBX, RAX
133                 JL L2
134             ;;;;;;;;;;;;;;; NEXT ROW ;;;;;;;;;;;;;;;
135             ADD R10, R14
136             ADD R13, R15
137             ADD RAX, R14
138             CMP R10, R11
139             JNZ L1
```

KOD ASM – RETURN I DATA

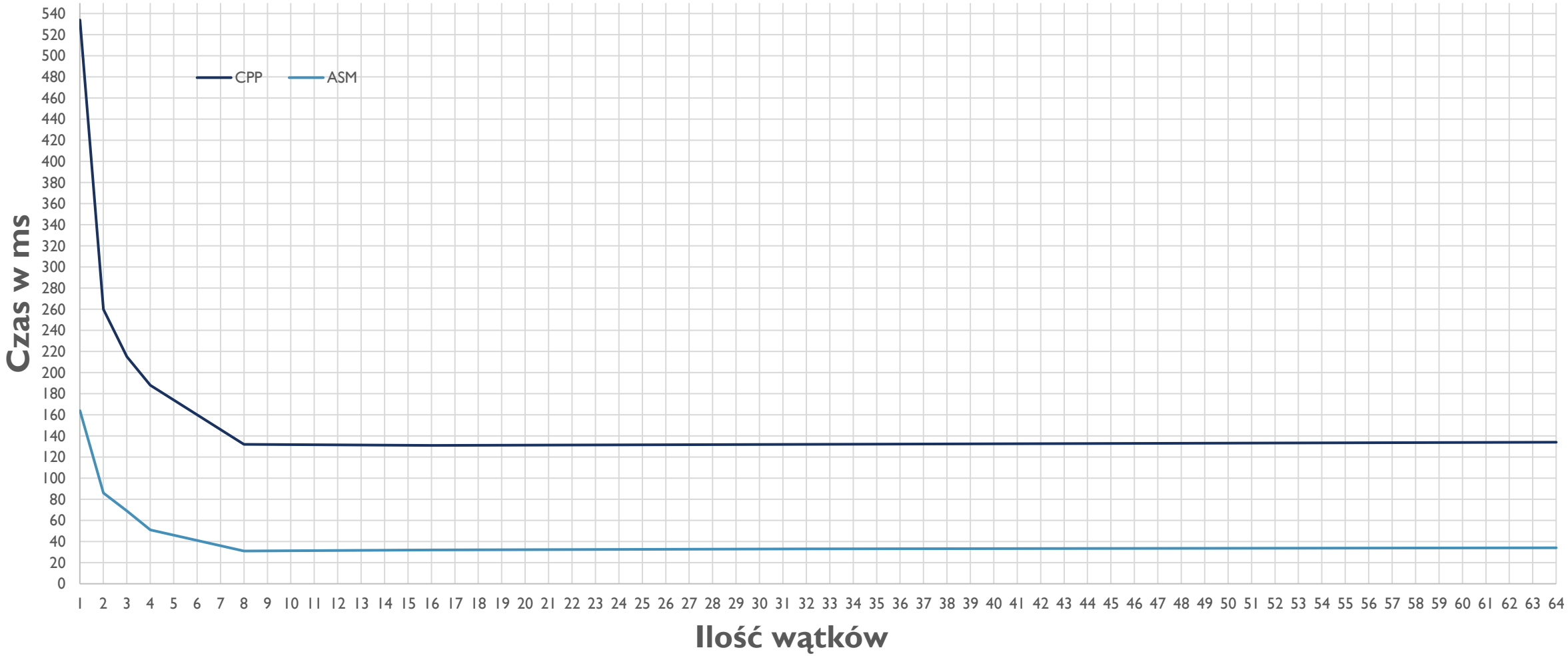
```
141                 POP  RBX
142                 RET
143             passImageToAsm ENDP
144
145         END
```

```
1       .DATA
2           SHUF1 DB    0,   0,   0,   0,   1,   0,   0,   0,   2,   0,   0,   0,   7,   0,   0,   0,   3,   0,   0,   0,   7,   0,   0,   0,   7,   0,   0,   0,   7,   0,   0,   0
3           SHUFB DB    0,  15,  15,  15,   3,  15,  15,  15,   6,  15,  15,  15,   9,  15,  15,  15,   0,  15,  15,  15,  15,  15,  15,  15,  15,  15,  15,  15,  15,  15,  15,  15
4           SHUFG DB    1,  15,  15,  15,   4,  15,  15,  15,   7,  15,  15,  15,  10,  15,  15,  15,   1,  15,  15,  15,  15,  15,  15,  15,  15,  15,  15,  15,  15,  15,  15,  15
5           SHUFR DB    2,  15,  15,  15,   5,  15,  15,  15,   8,  15,  15,  15,  11,  15,  15,  15,   2,  15,  15,  15,  15,  15,  15,  15,  15,  15,  15,  15,  15,  15,  15,  15
6           QQQQQ DB  199,   1,   0,   0, 199,   1,   0,   0, 199,   1,   0,   0, 199,   1,   0,   0, 199,   1,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0
7           SHUFF DB    2,   0,   0,   0,   3,   0,   0,   0,   4,   0,   0,   0,   7,   0,   0,   0,   7,   0,   0,   0,   7,   0,   0,   0,   7,   0,   0,   0,   7,   0,   0,   0
8           SHUFO DB    0,   0,   0,   4,   4,   4,   8,   8,   8,   3,   3,   3,   3,   3,   3,   3,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0
```

PORÓWNANIE CZASÓW DLA ŚREDNIEGO OBRAZU (AUTOMATYCZNIE, ŚREDNIA Z 100 POMIARÓW)



PORÓWNANIE CZASÓW DLA OBRAZU 8K (MANUALNIE)



WYNIKI DLA ŚREDNIEGO OBRAZU

- Osiągnięto maksymalne przyspieszenie na korzyść ASM: 3.948x
- Średnie przyspieszenie wyniosło: 1.58x

WYNIKI DLA OBRAZU 8K

- Osiągnięto maksymalne przyspieszenie na korzyść ASM: 4.258x
- Średnie przyspieszenie wyniosło: 3.672x

UŻYTE WEKTOROWE INSTRUKCJE ASM

- VPMAXSD
- MOVLPD
- VPSUBD
- VPERMD
- VMOVUPS
- VPSHUFB
- VPSLLD
- PEXTRB
- MOVUPS
- VPSRLD
- VPADDD
- VPXOR

ŹRÓDŁA

- https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html
- https://www.felixcloutier.com/x86/index.html
- https://godbolt.org
- https://chat.openai.com
- https://stackoverflow.com/questions/36122766/how-to-divide-by-9-using-just-shifts-add-sub
- https://www.youtube.com/watch?v=AT5nuQQO96o
- https://www.youtube.com/watch?v=rxsBghsrvpI&list=PLKK11Ligqitg9MOX3-0tFT1Rmh3uJp7kA
- https://www.youtube.com/@WhatsACreel

DZIĘKUJE