

ALGORYTM SOLARYZACJI OBRAZU

Bartłomiej Krasoń

ZAŁOŻENIA PROJEKTU

- Wykorzystanie C/C++ jako języku wysokiego poziomu (aplikacja konsolowa)
- Elementy obliczeniowe programu realizowane dwoma sposobami:
 - Algorytm w języku C/C++
 - Algorytm w asemblerze x64
- Zmierzenie i porównanie czasów wykonania dla dwóch sposobów
- Wykorzystanie wielowątkowości
- Wykorzystanie rozkazów typu SIMD w części asemblerowej

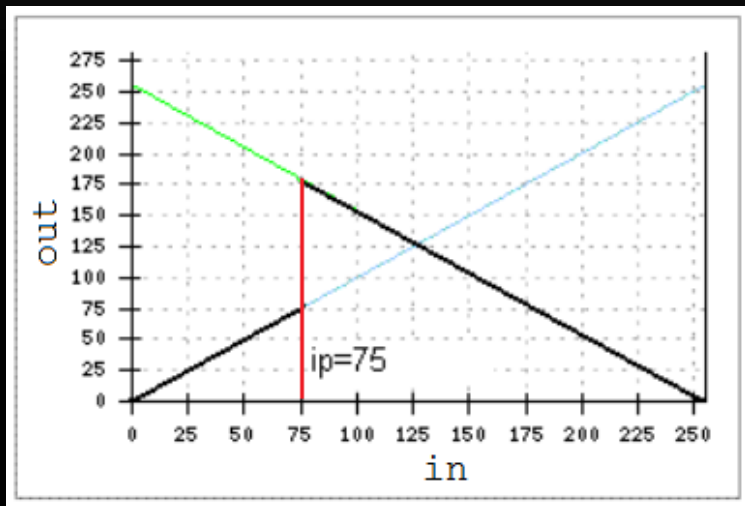
ALGORYTM SOLARYZACJI OBRAZU

Solaryzacja jest to zjawisko całkowitego lub częściowego odwrócenia obrazu negatywowego w pozytywowy. Stopień "odwrócenia" obrazu zależy od zadanego progu jasności powyżej którego następuje odwrócenie wartości składowych RGB piksela. Wartość progu i_p może oscylować w granicach od 0 do 255.

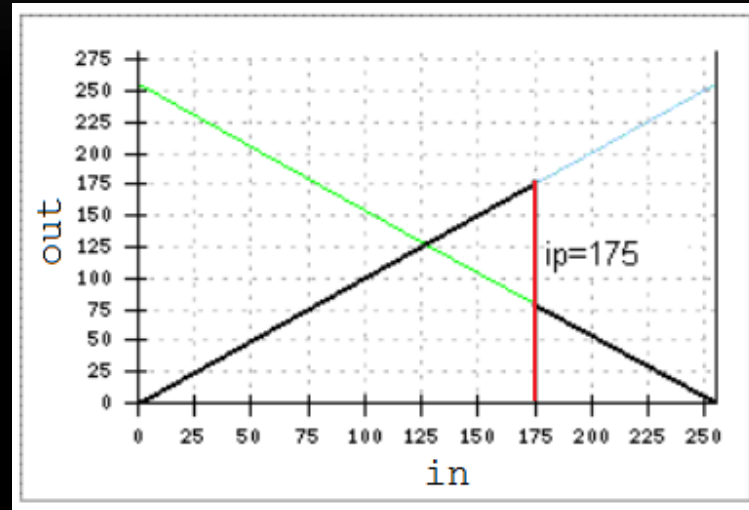
Algorytm można zapisać jako równanie:

$$i_{out} = \begin{cases} i_{in} & , \text{ jeżeli } i < i_p \\ i_{max} - i_{in} & , \text{ jeżeli } i \geq i_p \end{cases}$$

WYKRESY FUNKCJI SOLARYZACJI

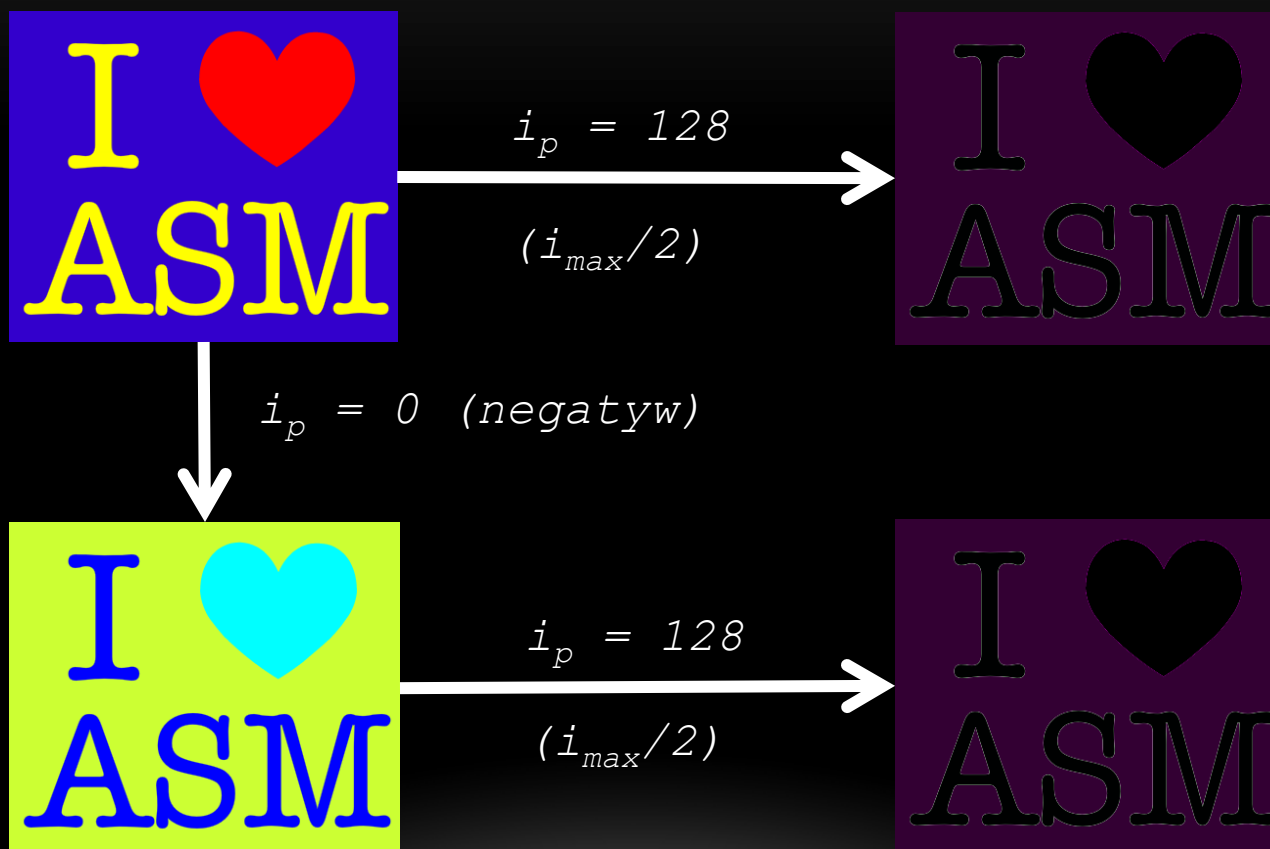


$i_p = 0 \Rightarrow \text{negatyw}$



$i_p = 255 \Rightarrow \text{oryginał}$

PRZYKŁAD DZIAŁANIA ALGORYTMU



ALGORYTM W C++

```
void Algorithm::solarize(BYTE *_data, BYTE *_result, const BYTE limit) {  
    if (int(*(_data)) >= limit) {  
        *_result = 255 - *_data;  
    } else {  
        *_result = *_data;  
    }  
}
```

ALGORYTM W ASM

```
120 ; ALGORYTM
121 ; MODYFIKACJA DANYCH
122 vpxor xmm1, xmm1, xmm6 ; wszystkie bajty: 255-val do xmm1
123 vpxor xmm2, xmm2, xmm5 ; konwersja danych na unsigned_int (w celu porównania z progiem) ; 1ms elapsed
124 ; USTALENIE KTÓRE DANE ODWRACAMY
125 pcmptb xmm2, xmm7 ; których negacji potrzebujemy do xmm2
126 ; wstawia FF tam gdzie BYTE z xmm2 jest większy od BYTE na xmm7, inaczej wstawia 00
127 vpand xmm3, xmm1, xmm2 ; negacje docelowe do xmm3
128 ; USTALENIE KTÓRE DANE POZOSTAWIAMY BEZ ZMIAN
129 vpxor xmm4, xmm2, xmm6 ; których niezmiennych danych potrzebujemy (FF) do xmm4
130 vpand xmm4, xmm0, xmm4 ; niezmiennione dane docelowe do xmm4
131 ; POŁĄCZENIE DOCELOWYCH ZMIENIONYCH I NIEZMIENIONYCH DANYCH
132 vpor xmm1, xmm3, xmm4 ; wynik do xmm1
```

```
XMM0 = 57977C4590763D967B468268357F6435
XMM1 = A86883BA6F89C26984B97D97CA809BCA
XMM2 = D717FCC510F6BD16FBC602E8B5FFE4B5
XMM3 = 00000000000000000000000000000000
XMM4 = 00000000000000000000000000000000
XMM5 = 80808080808080808080808080808080
XMM6 = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
XMM7 = E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4
```

ALGORYTM W ASM

```
120 ; ALGORYTM
121 ; MODYFIKACJA DANYCH
122 vpxor xmm1, xmm1, xmm6 ; wszystkie bajty: 255-val do xmm1
123 vpxor xmm2, xmm2, xmm5 ; konwersja danych na unsigned_int (w celu porównania z progiem)
124 ; USTALENIE KTÓRE DANE ODWRACAMY
125 pcmpltb xmm2, xmm7 ; których negacji potrzebujemy do xmm2 ; 1 ms elapsed
126 ; wstawia FF tam gdzie BYTE z xmm2 jest większy od BYTE na xmm7, inaczej wstawia 00
127 vpand xmm3, xmm1, xmm2 ; negacje docelowe do xmm3
128 ; USTALENIE KTÓRE DANE POZOSTAWIAMY BEZ ZMIAN
129 vpxor xmm4, xmm2, xmm6 ; których niezmiennych danych potrzebujemy (FF) do xmm4
130 vpand xmm4, xmm0, xmm4 ; niezmiennione dane docelowe do xmm4
131 ; POŁĄCZENIE DOCELOWYCH ZMIENIONYCH I NIEZMIENIONYCH DANYCH
132 vpor xmm1, xmm3, xmm4 ; wynik do xmm1
```

```
XMM0 = 57977C4590763D967B468268357F6435
XMM1 = A86883BA6F89C26984B97D97CA809BCA
XMM2 = 00FFFF00FFFF00FFFF00FFFF00FF0000
XMM3 = 00000000000000000000000000000000
XMM4 = 00000000000000000000000000000000
XMM5 = 80808080808080808080808080808080
XMM6 = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
XMM7 = E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4
```


ALGORYTM W ASM

```
120 ; ALGORYTM
121 ; MODYFIKACJA DANYCH
122     vpxor xmm1, xmm1, xmm6 ; wszystkie bajty: 255-val do xmm1
123     vpxor xmm2, xmm2, xmm5 ; konwersja danych na unsigned_int (w celu porównania z progiem)
124 ; USTALENIE KTÓRE DANE ODWRACAMY
125     pcmpgtb xmm2, xmm7 ; których negacji potrzebujemy do xmm2
126     ; wstawia FF tam gdzie BYTE z xmm2 jest większy od BYTE na xmm7, inaczej wstawia 00
127     vpand xmm3, xmm1, xmm2 ; negacje docelowe do xmm3 a 2mo niapad
128 ; USTALENIE KTÓRE DANE POZOSTAWIAMY BEZ ZMIAN
129     vpxor xmm4, xmm2, xmm6 ; których niezmiennych danych potrzebujemy (FF) do xmm4
130     vpand xmm4, xmm0, xmm4 ; niezmiennione dane docelowe do xmm4
131 ; POŁĄCZENIE DOCELOWYCH ZMIENIONYCH I NIEZMIENIONYCH DANYCH
132     vpor xmm1, xmm3, xmm4 ; wynik do xmm1
```

XMM0 = 57977C4590763D967B468268357F6435
XMM1 = A86883BA6F89C26984B97D97CA809BCA
XMM2 = 00FFFF00FFFF00FFFF00FFFF00FF0000
XMM3 = 006883006F89006984007D9700800000
XMM4 = 00000000000000000000000000000000
XMM5 = 80808080808080808080808080808080
XMM6 = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
XMM7 = E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4

ALGORYTM W ASM

```
120 ; ALGORYTM
121 ; MODYFIKACJA DANYCH
122 vpxor xmm1, xmm1, xmm6 ; wszystkie bajty: 255-val do xmm1
123 vpxor xmm2, xmm2, xmm5 ; konwersja danych na unsigned_int (w celu porównania z progiem)
124 ; USTALENIE KTÓRE DANE ODWRACAMY
125 pcmpltb xmm2, xmm7 ; których negacji potrzebujemy do xmm2
126 ; wstawia FF tam gdzie BYTE z xmm2 jest większy od BYTE na xmm7, inaczej wstawia 00
127 vpand xmm3, xmm1, xmm2 ; negacje docelowe do xmm3
128 ; USTALENIE KTÓRE DANE POZOSTAWIAMY BEZ ZMIAN
129 vpxor xmm4, xmm2, xmm6 ; których niezmiennych danych potrzebujemy (FF) do xmm4
130 vpand xmm4, xmm0, xmm4 ; niezmiennione dane docelowe do xmm4
131 ; POŁĄCZENIE DOCELOWYCH ZMIENIONYCH I NIEZMIENIONYCH DANYCH
132 vpor xmm1, xmm3, xmm4 ; wynik do xmm1
```

XMM0 = 57977C4590763D967B468268357F6435
XMM1 = A86883BA6F89C26984B97D97CA809BCA
XMM2 = 00FFFF00FFFF00FFFF00FFFF00FF0000
XMM3 = 006883006F89006984007D9700800000
XMM4 = FF0000FF0000FF0000FF0000FF00FFFF
XMM5 = 80808080808080808080808080808080
XMM6 = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
XMM7 = E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4

ALGORYTM W ASM

```
120 ; ALGORYTM
121 ; MODYFIKACJA DANYCH
122     vpxor xmm1, xmm1, xmm6 ; wszystkie bajty: 255-val do xmm1
123     vpxor xmm2, xmm2, xmm5 ; konwersja danych na unsigned_int (w celu porównania z progiem)
124 ; USTALENIE KTÓRE DANE ODWRACAMY
125     pcmpgtb xmm2, xmm7 ; których negacji potrzebujemy do xmm2
126         ; wstawia FF tam gdzie BYTE z xmm2 jest większy od BYTE na xmm7, inaczej wstawia 00
127     vpand xmm3, xmm1, xmm2 ; negacje docelowe do xmm3
128 ; USTALENIE KTÓRE DANE POZOSTAWIAMY BEZ ZMIAN
129     vpxor xmm4, xmm2, xmm6 ; których niezmiennych danych potrzebujemy (FF) do xmm4
130     vpand xmm4, xmm0, xmm4 ; niezmienione dane docelowe do xmm4
131 ; POŁĄCZENIE DOCELOWYCH ZMIENIONYCH I NIEZMIENIONYCH DANYCH
132     vpor xmm1, xmm3, xmm4 ; wynik do xmm1
```

XMM0 = 57977C4590763D967B468268357F6435
XMM1 = A86883BA6F89C26984B97D97CA809BCA
XMM2 = 00FFFF00FFFF00FFFF00FFFF00FF0000
XMM3 = 006883006F89006984007D9700800000
XMM4 = 5700004500003D000046000035006435
XMM5 = 80808080808080808080808080808080
XMM6 = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
XMM7 = E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4

ALGORYTM W ASM

```
120 ; ALGORYTM
121 ; MODYFIKACJA DANYCH
122 vpxor xmm1, xmm1, xmm6 ; wszystkie bajty: 255-val do xmm1
123 vpxor xmm2, xmm2, xmm5 ; konwersja danych na unsigned_int (w celu porównania z progiem)
124 ; USTALENIE KTÓRE DANE ODWRACAMY
125 pcmpgtb xmm2, xmm7 ; których negacji potrzebujemy do xmm2
126 ; wstawia FF tam gdzie BYTE z xmm2 jest większy od BYTE na xmm7, inaczej wstawia 00
127 vpand xmm3, xmm1, xmm2 ; negacje docelowe do xmm3
128 ; USTALENIE KTÓRE DANE POZOSTAWIAMY BEZ ZMIAN
129 vpxor xmm4, xmm2, xmm6 ; których niezmiennych danych potrzebujemy (FF) do xmm4
130 vpand xmm4, xmm0, xmm4 ; niezmiennione dane docelowe do xmm4
131 ; POŁĄCZENIE DOCELOWYCH ZMIENIONYCH I NIEZMIENIONYCH DANYCH
132 vpor xmm1, xmm3, xmm4 ; wynik do xmm1 a 1ms elapsed
```

XMM0 = 57977C4590763D967B468268357F6435
XMM1 = 576883456F893D6984467D9735806435
XMM2 = 00FFFF00FFFF00FFFF00FFFF00FF0000
XMM3 = 006883006F89006984007D9700800000
XMM4 = 5700004500003D000046000035006435
XMM5 = 80808080808080808080808080808080
XMM6 = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
XMM7 = E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4

PORÓWNANIE CZASÓW

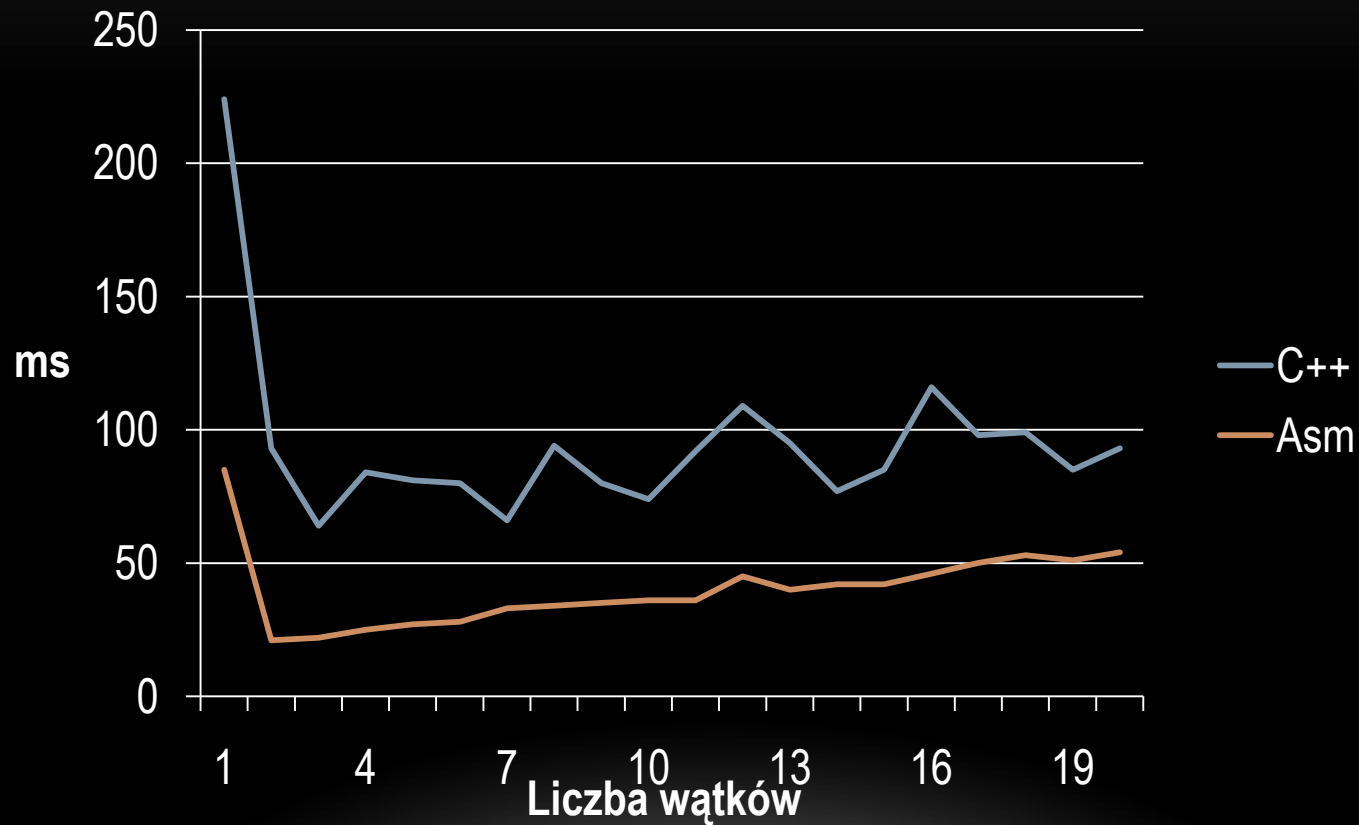
C++ (Release)

```
Enter file path:
img/city.bmp
Enter limit from range of <0 : 255>
128
1 threads - duration: 224ms.
2 threads - duration: 93ms.
3 threads - duration: 64ms.
4 threads - duration: 84ms.
5 threads - duration: 81ms.
6 threads - duration: 80ms.
7 threads - duration: 66ms.
8 threads - duration: 94ms.
9 threads - duration: 80ms.
10 threads - duration: 74ms.
11 threads - duration: 92ms.
12 threads - duration: 109ms.
13 threads - duration: 95ms.
14 threads - duration: 77ms.
15 threads - duration: 85ms.
16 threads - duration: 116ms.
17 threads - duration: 98ms.
18 threads - duration: 99ms.
19 threads - duration: 85ms.
20 threads - duration: 93ms.
Press any key to continue . . .
```

Asembler

```
Enter file path:
img/city.bmp
Enter limit from range of <0 : 255>
128
1 threads - duration: 85ms.
2 threads - duration: 21ms.
3 threads - duration: 22ms.
4 threads - duration: 25ms.
5 threads - duration: 27ms.
6 threads - duration: 28ms.
7 threads - duration: 33ms.
8 threads - duration: 34ms.
9 threads - duration: 35ms.
10 threads - duration: 36ms.
11 threads - duration: 36ms.
12 threads - duration: 45ms.
13 threads - duration: 40ms.
14 threads - duration: 42ms.
15 threads - duration: 42ms.
16 threads - duration: 46ms.
17 threads - duration: 50ms.
18 threads - duration: 53ms.
19 threads - duration: 51ms.
20 threads - duration: 54ms.
Press any key to continue . . .
```

PORÓWNANIE CZASÓW - WYKRES



DZIĘKUJĘ ZA UWAGĘ
