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

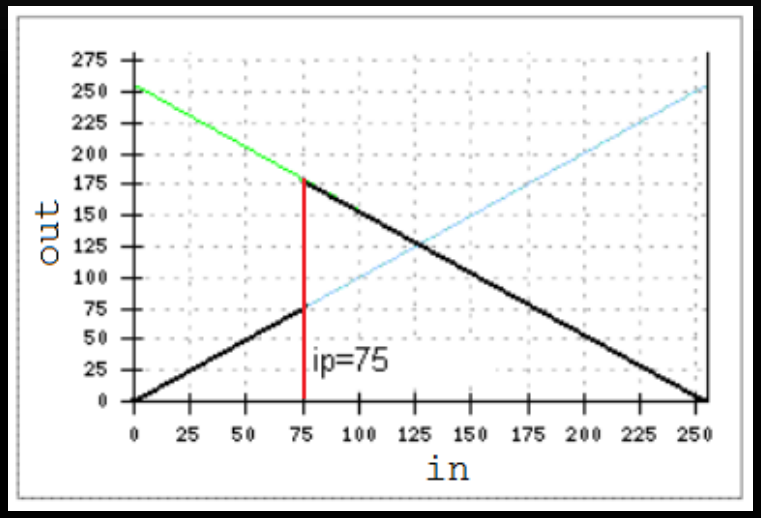
ALGORYTM SOLARYZACJI OBRAZU

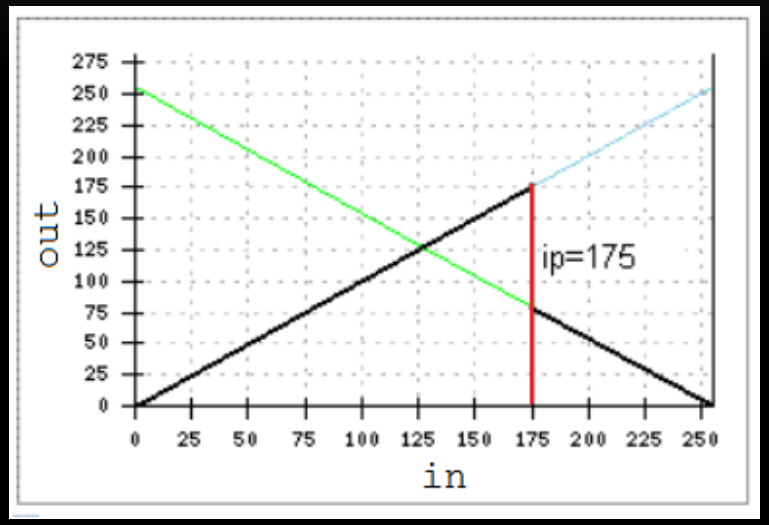
Solaryzacja jest to zjawisko całkowitego lub częściowego odwrócenia obrazu negatywowego w pozytywowy. Stopień ”odwrócenia” obrazu zależy od zadanego progu jasności powyżej którego następuje odwrócenie wartości składowych RGB piksela. Wartość progu i_p może oscylować w granicach od 0 do 255.

Algorytm można zapisać jako równanie:

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

WYKRESY FUNKCJI SOLARYZACJI

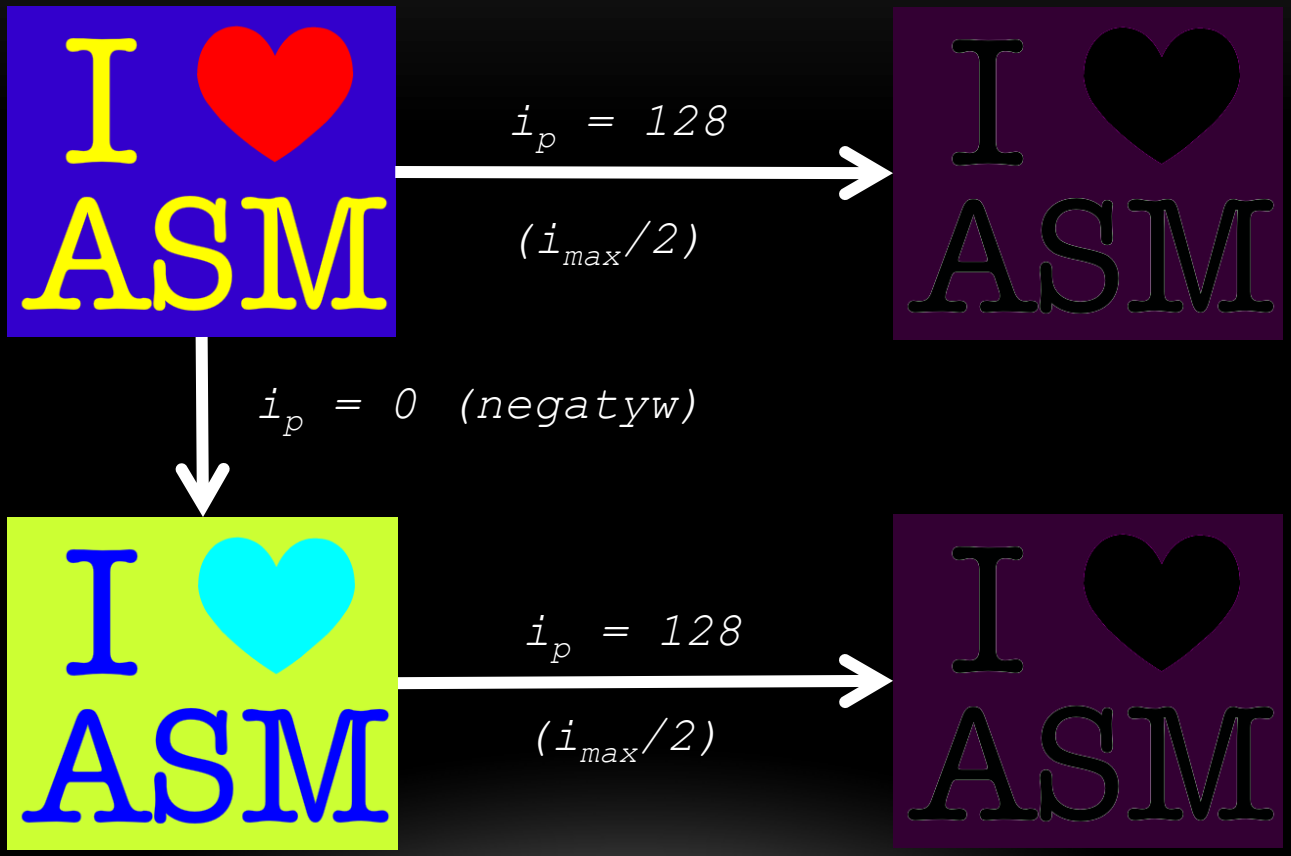




$i_p = 0 \Rightarrow$ negatyw

$i_p = 255 \Rightarrow$ 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 ; wszytskie 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     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 niezmienionych 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 = D717FCC510F6BD16FBC602E8B5FFE4B5
XMM3 = 00000000000000000000000000000000
XMM4 = 00000000000000000000000000000000
XMM5 = 80808080808080808080808080808080
XMM6 = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
XMM7 = E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4
```

ALGORYTM W ASM

```
120 ; ALGORYTM
121 ; MODYFIKACJA DANYCH
122     vpxor xmm1, xmm1, xmm6 ; wszytskie 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  ≤ 1ms 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 niezmienionych 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 = 00000000000000000000000000000000
XMM4 = 00000000000000000000000000000000
XMM5 = 80808080808080808080808080808080
XMM6 = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
XMM7 = E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4
```

ALGORYTM W ASM

```
120 ; ALGORYTM
121  ; MODYFIKACJA DANYCH
122     vpxor xmm1, xmm1, xmm6 ; wszytskie 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  ≤ 2ms elapsed
128  ; USTALENIE KTÓRE DANE POZOSTAWIAMY BEZ ZMIAN
129     vpxor xmm4, xmm2, xmm6 ; których niezmienionych 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 = 00000000000000000000000000000000
XMM5 = 80808080808080808080808080808080
XMM6 = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
XMM7 = E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4
```

ALGORYTM W ASM

```
120 ; ALGORYTM
121  ; MODYFIKACJA DANYCH
122     vpxor xmm1, xmm1, xmm6 ; wszytskie 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 niezmienionych danych potrzebujemy (FF) do xmm4 ≤ 1ms elapsed
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 = FF0000FF0000FF0000FF0000FF00FFFF
XMM5 = 8080808080808080808080808080808080
XMM6 = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
XMM7 = E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4
```

ALGORYTM W ASM

```
120 ; ALGORYTM
121  ; MODYFIKACJA DANYCH
122     vpxor xmm1, xmm1, xmm6 ; wszytskie 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 niezmienionych danych potrzebujemy (FF) do xmm4
130     vpand xmm4, xmm0, xmm4 ; niezmienione dane docelowe do xmm4 ≤ 2ms elapsed
131  ; POŁĄCZENIE DOCELOWYCH ZMIENIONYCH I NIEZMIENIONYCH DANYCH
132     vpor xmm1, xmm3, xmm4 ; wynik do xmm1
```

```
XMM0 = 57977C4590763D967B468268357F6435
XMM1 = A86883BA6F89C26984B97D97CA809BCA
XMM2 = 00FFFF00FFFF00FFFF00FFFF00FF0000
XMM3 = 006883006F8900698400 7D9700800000
XMM4 = 5700004500003D000046000035006435
XMM5 = 8080808080808080808080808080808080
XMM6 = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
XMM7 = E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4E4
```

ALGORYTM W ASM

```
120 ; ALGORYTM
121  ; MODYFIKACJA DANYCH
122     vpxor xmm1, xmm1, xmm6 ; wszytskie 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 niezmienionych 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 ≤ 1ms elapsed
```

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

DZIĘKUJĘ ZA UWAGĘ