Silesian University of Technology

Faculty of Automatic Control, Electronics and Computer Science

Department of Graphics, Computer Vision and Digital Systems

Języki asemblerowe projekt

Rok akademicki	Rodzaj studiów*:	Numer ćwiczenia:	Grupa	Sekcja
2022/2023	**SSI**	**-**	**5**	**2**
Data i godzina planowana ćwiczenia: dd/mm/rrrr - gg:mm	-	**Prowadzący**:	**JP**	
Data i godzina wykonania ćwiczenia: dd/mm/rrrr - gg:mm	-			

Raport końcowy

Temat projektu:

Desaturacja zdjęcia

Skład sekcji:	1. Filip Miera

1. Opis implementacji algorytmu

Program ma za zadanie zmienić wybrane i wgrane do aplikacji zdjęcie kolorowe na zdjęcie szare. Algorytm, który został zaimplementowany w tym programie, wymnaża każdy kolor pixela przez odpowiednią wartość. Odpowiednio dla czerwonego: 0.2627, zielonego: 0.6780, niebieskiego: 0.0593. Algorytm może działać wielowątkowo, poprzez podzielenie zdjęcia na odpowiednią ilość wierszy i przydzielenie ich każdemu wątkowi. Ilość wątków jest automatycznie ustawiana na wszystkie możliwe wątki procesora sprzętu użytkownika, lecz on sam również może zmienić tę liczbę w programie. Algorytm został zaimplementowany w C++ oraz w asemblerze co daje nam możliwość porównania ich czasów wykonywania. Oba algorytmy są bibliotekami DLL które są dołączane dynamicznie do programu (w zależności którą bibliotekę wybierze użytkownik) w trakcie jego działania.

2. Opis parametrów wejściowych programu

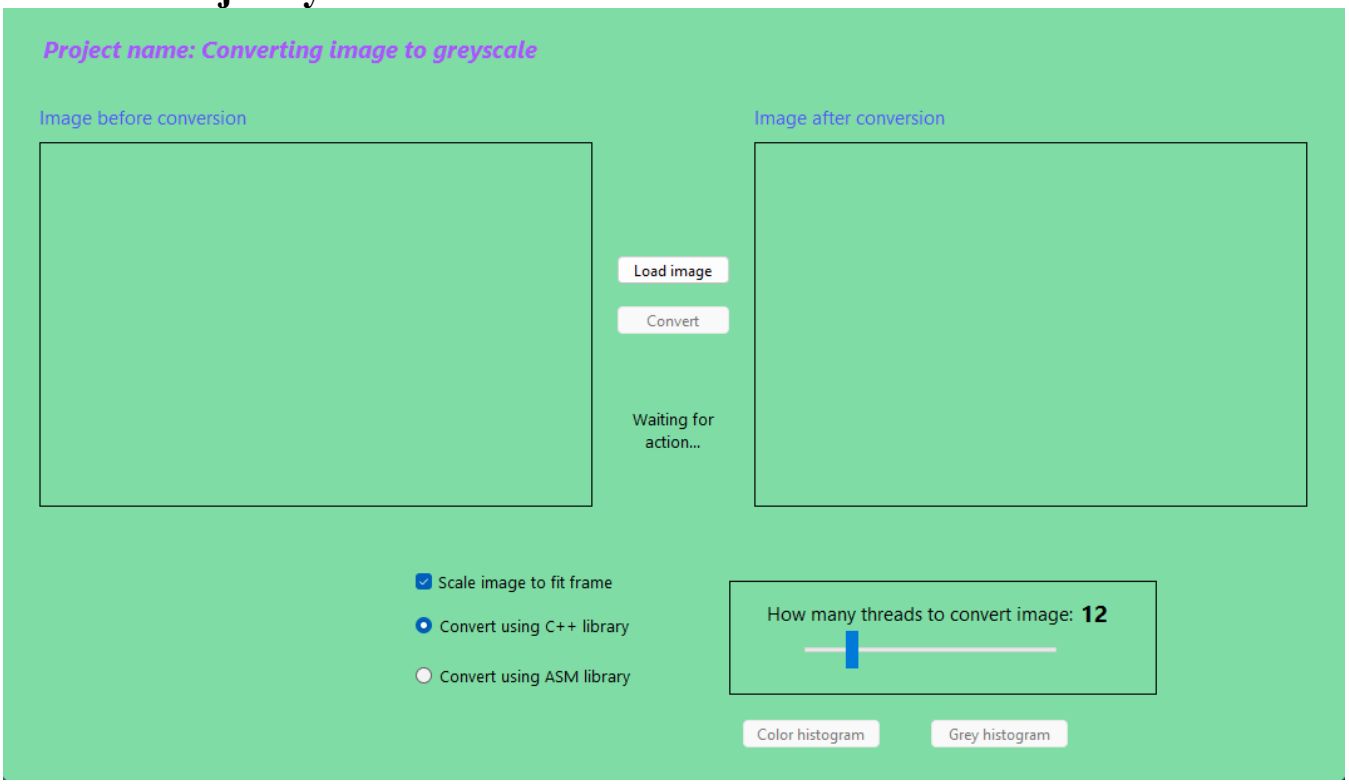
Parametrem wejściowym w programie jest obraz z rozszerzeniem .png, .jpg lub .jpeg, dodatkowymi parametrami są również: ilość wątków przeznaczonych do konwersji zdjęcia, wybór biblioteki DLL oraz czy przeskalować zdjęcie do ramki programu. Wybór ilości wątków jest możliwa za pomocą slidera w zakresie 1-64, domyślna ilość wątków jest ustalana na podstawie maksymalnej ilości wątków w procesorze użytkownika. Wybór biblioteki z której chce się skorzystać został przedstawiony jako 2 radio butony z odpowiednio przypisanymi bibliotekami. Przeskalowywanie zdjęcia można wybierać za pomocą checkboxa.

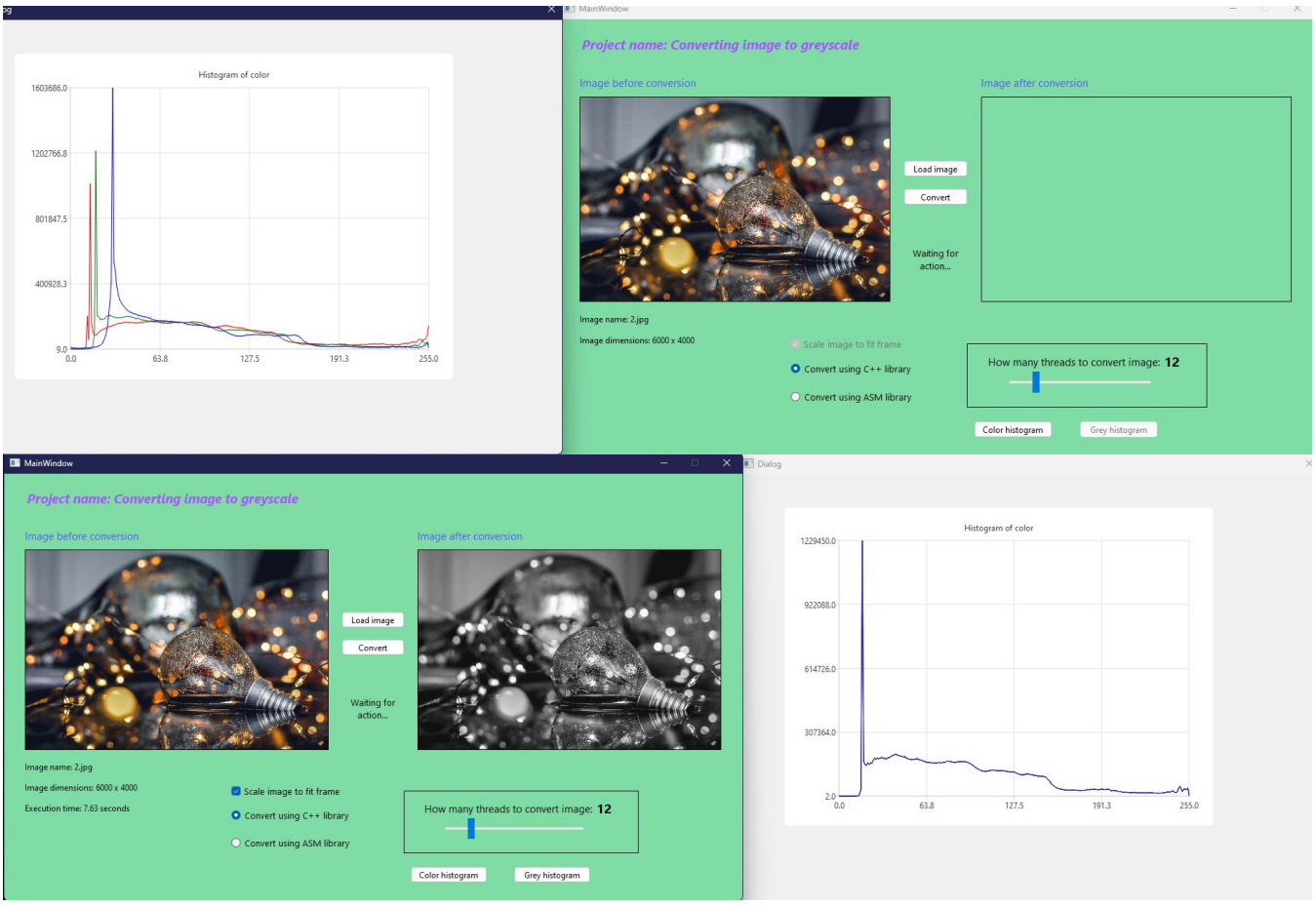
3. Opis kodu źródłowego biblioteki DLL z implementacją algorytmu w asemblerze

```
.data
; tworzenie stalych ktore zostana uzyte do otrzymania odcieniu szarosci
RED_MULTIPLIER real4 0.2627
GREEN_MULTIPLIER real4 0.6780
BLUE_MULTIPLIER real4 0.0593

.code
DesaturationFunction proc ;poczatek pogramu desaturacji

movd xmm3, RED_MULTIPLIER ;ladowanie wartosci zmiennych do rejestrow xmm3-5
movd xmm4, GREEN_MULTIPLIER
movd xmm5, BLUE_MULTIPLIER

mulss xmm0, xmm3 ;mnozenie pierwszego przekazanego do funkcji parametru przez odpowiednia wartosc
mulss xmm1, xmm4 ;mnozenie drugiego przekazanego do funkcji parametru przez odpowiednia wartosc
mulss xmm2, xmm5 ;mnozenie trzeciego przekazanego do funkcji parametru przez odpowiednia wartosc

addss xmm0, xmm1 ;sumowanie otrzymanych wartosci aby,
addss xmm0, xmm2 ;otrzymac odpowiedni kolor pixela

ret
DesaturationFunction endp
end ;koniec pogramu desaturacji
```

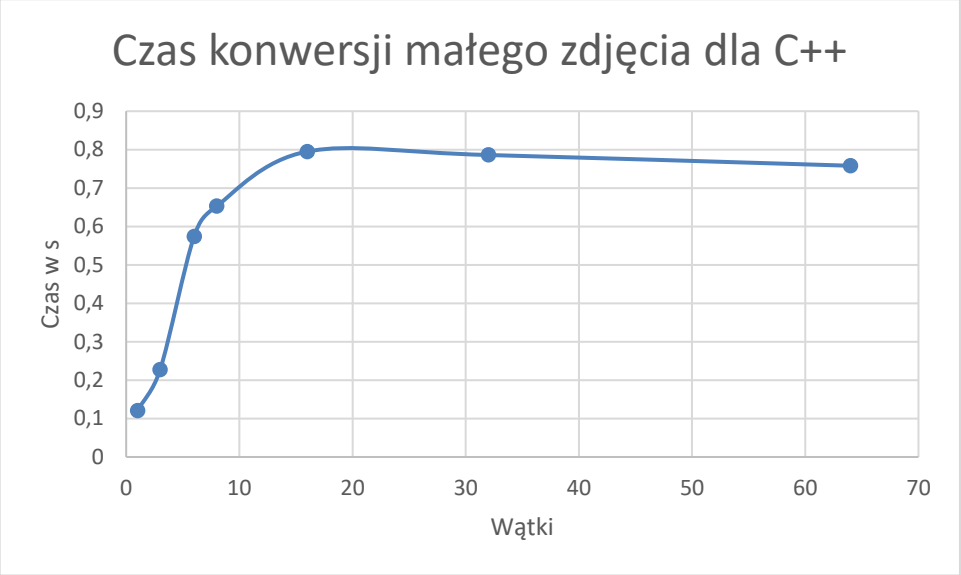
Pierwszym krokiem, było utworzenie stałych które posłużą do przemnożenia odpowiednich kolorów przez te stałe, aby otrzymać kolor szary pixela. Następnie te stałe zostały załadowane do rejestrów XMM3-5. Jako iż przekazywałem wartości pixela do programu jako wartości float'owe to zmienne zapisały się do rejestrów XMM1 – czerwony, XMM2 – zielony, XMM3 – niebieski. Więc kolejnym krokiem było przemożenie wartości w każdym rejestrze przez dobrane wartości, oraz zsumowanie ich.

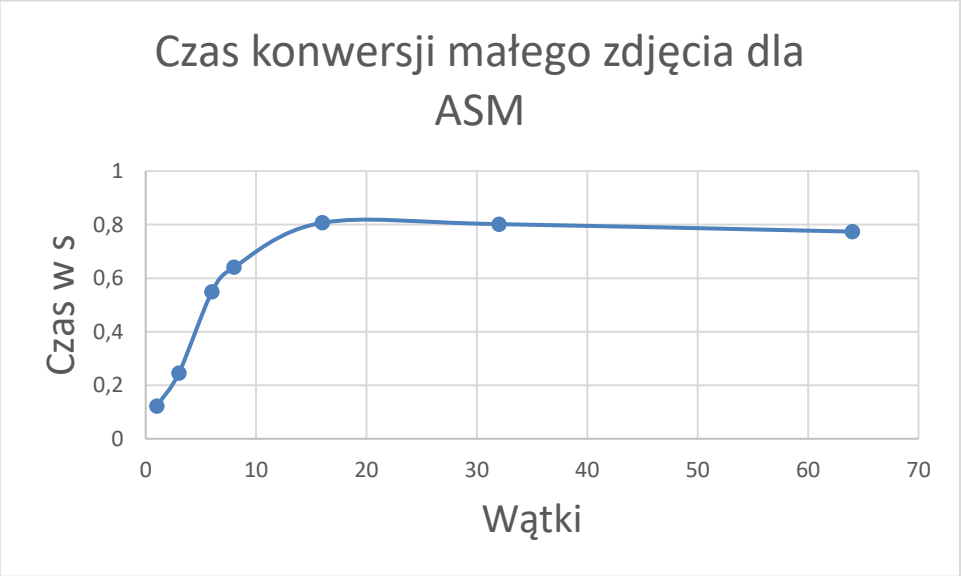
4. Interfejs użytkownika

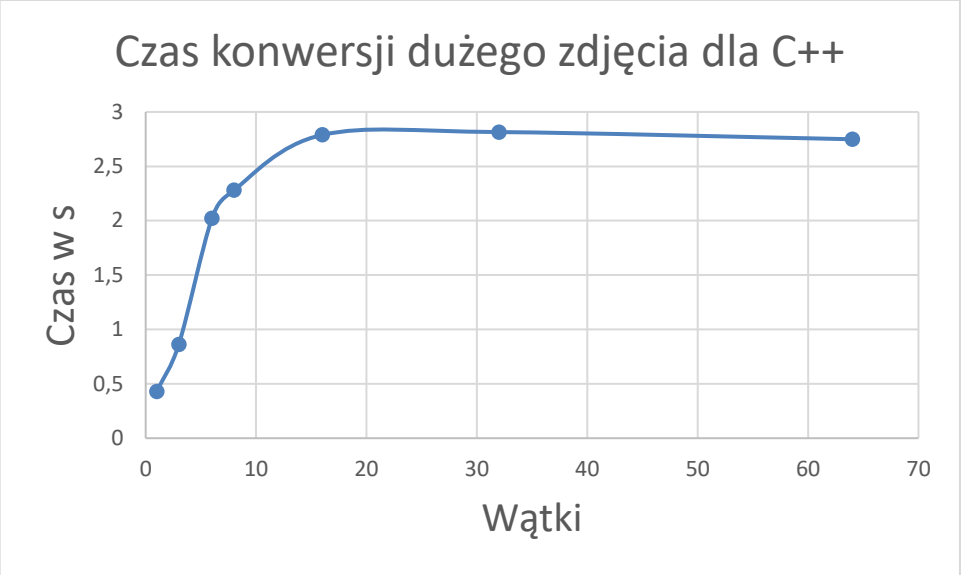


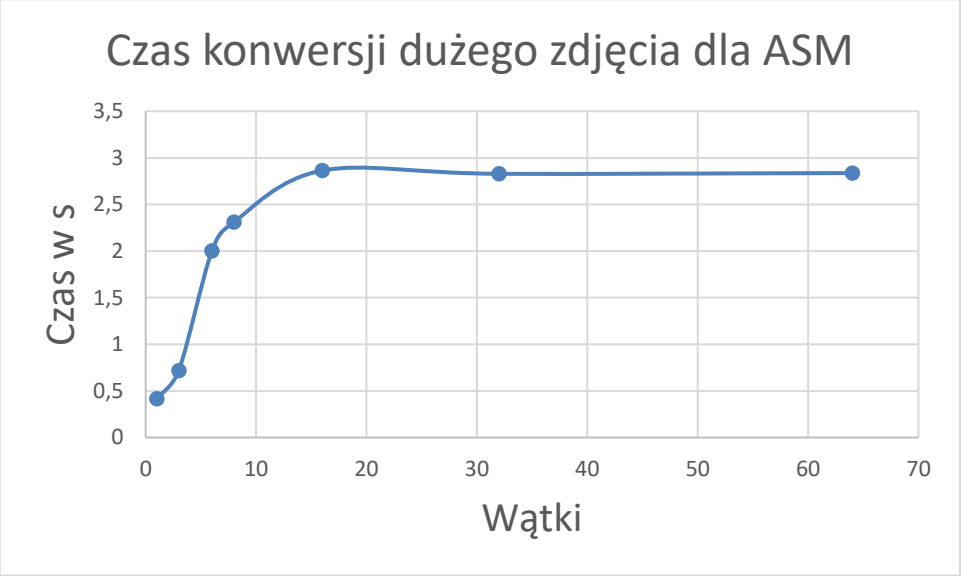


5. Raport szybkości działania









6. Opis testowania i uruchamiania programu

W programie została zablokowana możliwość wgrania pliku o niewłaściwym rozszerzeniu, program akceptuje tylko rozszerzenia jpg, jpeg, oraz png. Wybór ilości wątków działa jako slider więc użytkownik nie ma możliwości podania błędnych danych. Przycisk konwersji jest zablokowanych dopóki użytkownik nie doda zdjęcia do programu. Aby zobaczyć histogram zdjęcia kolorowego, trzeba wgrać zdjęcie do programu, a histogram zdjęcia wyszarzonego to trzeba przekonwertować zdjęcie.

7. Wnioski

Projekt pozwolił zapoznać się lepiej z zastosowaniem asemblera w praktyce oraz porównać czasy wykonania się tych samych algorytmów w języku C++ oraz asemblera za pomocą bibliotek DLL. Oba algorytmy zostały napisane w sposób jak najbardziej zbliżony do siebie co pozwoliło również zaobserwować różnice wynikające z implementacji. Przy porównywaniu czasów obu algorytmów, nie widać znaczniej różnicy w czasach wykonywania ponieważ, konwersja zaimplementowana w bibliotekach DLL w C++ oraz assemblerze jest bardzo prosta.