
	Instytut Informatyki Politechniki Śląskiej Zespół Mikroinformatyki i Teorii Automatów Cyfrowych Projekt JA			
Rok akademicki	Rodzaj studiów*: SSI/NSI/NSM	Grupa	Sekcja	
20022/2023	SSI	2	1	
Prowadzący: OA/JP/KT/GD/BSz/GB	AO			
<h2 style="text-align: center;"><i>Raport końcowy</i></h2>				
Temat: <h1 style="text-align: center;">Filtrowanie bilinearne</h1>				
Autor:		Mikołaj Krasucki		

Opis implementacji

1. Opis tematu

Tematem projektu jest filtrowanie bilinearne (dwuliniowe), mające na celu poprawę jakości tekstur, przez ich wygładzenie.

2. Opis metody

Metoda rozwiązania projektu polega na policzeniu jasności piksela, jako sumy ważonej sąsiednich pikseli. Waga danego piksela jest uzależniona od stopnia pokrycia przez filtrowany piksel.

Wartość obliczonego piksela w punkcie (x, y) będzie wynosić:

$$Q(x, y) = Q_{11}uv + Q_{12}u(1 - v) + Q_{21}(1 - u)v + Q_{22}(1 - u)(1 - v)$$

Gdzie :

$$Q_{11} = (x_1, y_1), Q_{12} = (x_1, y_2), Q_{21} = (x_2, y_1), Q_{22} = (x_2, y_2)$$

$$x_1 \leq x \leq x_2,$$

$$y_1 \leq y \leq y_2.$$

, oraz

$$u = \frac{x - x_1}{x_2 - x_1}, \text{ zaś } v = \frac{y - y_1}{y_2 - y_1}.$$

Wartość obliczonego piksela w punkcie (x, y) jest obliczana na podstawie dwuliniowej interpolacji, tzn. złożeniu dwóch interpolacji, czyli stworzeniu nowego piksela z pikseli sąsiadujących, w taki sposób aby był jak najlepiej dopasowany do transformowanego obrazu.

3. Założenia projektu

- Projekt realizowany w środowisku Visual Studio
- Projekt dotyczy architektury procesorów Intel X86/64
- Projekt poprawia jakość zdjęcia
- Program liczy czas wykonywania operacji w C++ oraz w asemblerze X64
- Stworzenie aplikacji interfejsu użytkownika, zawierający okienkowy interfejs komunikacyjny z użytkownikiem umożliwiający parametryzację aplikacji i elementy sterujące procesem wykonania
- Użytkownik w UI (user interface) ma możliwość wyboru wywołania biblioteki w języku C++ oraz asemblerze

- Użytkownik ma możliwość wybrania ilości wątków, w których będzie wykonywanych funkcja poprawiająca jakość tekstur
- Użytkownik ma możliwość załadowania pliku .bmp do programu
- Biblioteka DLL zawiera funkcje biblioteczne wywołane dynamiczne z poziomu aplikacji głównej napisana w C++ oraz w asemblerze X64

Opis parametrów wejściowych dla programu

Do funkcji przekazywane są dane w następujący sposób :

- `mov r10, rcx` ;Wskaźnik na 1 piksel wczytujący
- `mov r11, rdx` ;Wskaźnik na 1 piksel zapisujący
- `mov rcx, r8` ;Numpixels (licznik pętli)
- `mov rdx, r9` ;Szerokosc od której program ma zacząć wykonywanie
- `mov eax, [rsp+8*11]` ;Wysokosc od której program ma zacząć wykonywanie
- `mov r12, rax`
- `mov eax, [rsp+8*12]` ;Aktualna pozycja na osi x
- `mov r13, rax`
- `mov eax, [rsp+8*13]` ; Aktualna pozycja na osi y
- `mov r14, rax`
- `mov eax, [rsp+8*14]` ;Padding
- `mov r15, rax`

Do programu wczytywana jest bitmapa w następujący sposób :

```
BMPFile::BMPFile(const char* fileName)
{
    this->fileName = fileName;

    std::fstream file(fileName, std::fstream::in | std::fstream::binary);

    if (file) {

        file.seekg(0, file.end);
        dataSize = file.tellg();
        file.seekg(0, file.beg);

        data = new byte[dataSize];
        file.read((char*)data, dataSize);

        fileHeader = PBITMAPFILEHEADER(data);
        infoHeader = PBITMAPINFOHEADER(data + sizeof(BITMAPFILEHEADER));
        pixels = data + fileHeader->bfOffBits;
    }

    file.close();
}
```

Program przyjmuje od usera także :

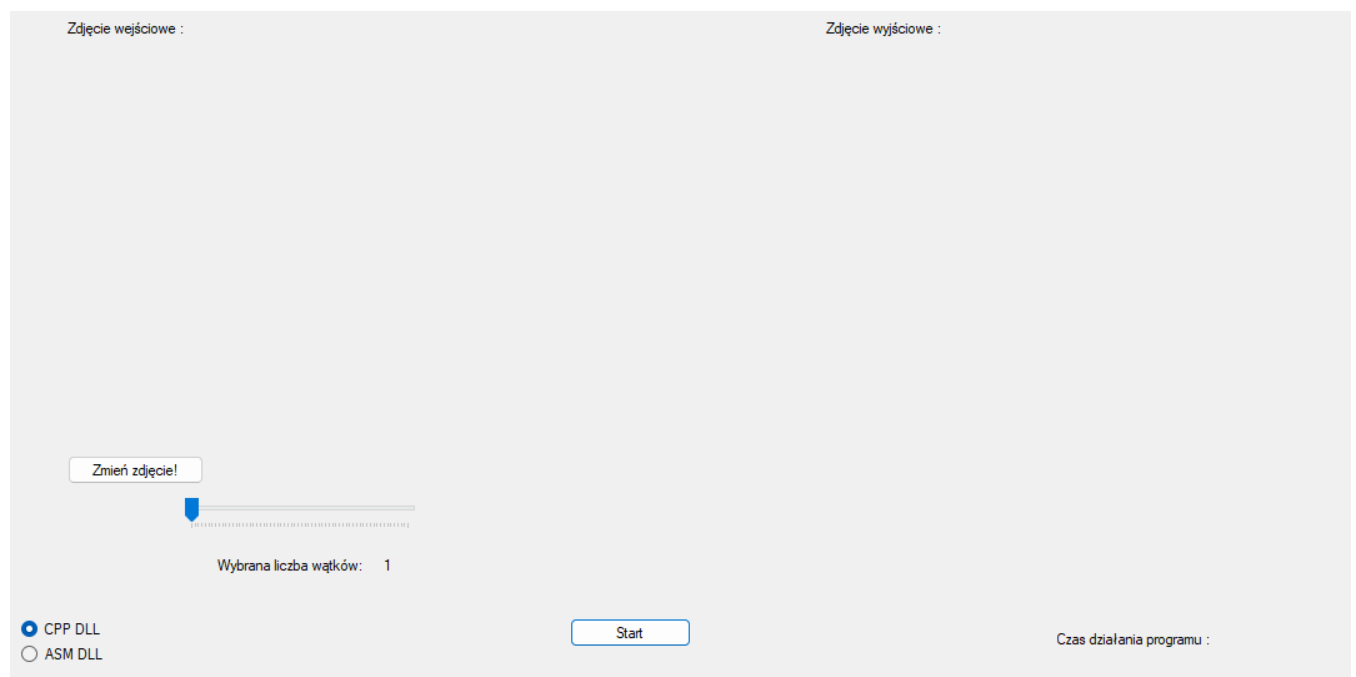
- Ilość wątków
- Biblioteki (Asm / Cpp)
- Nazwę zdjęcia

Opis wybranego frgmentu asemblerowego

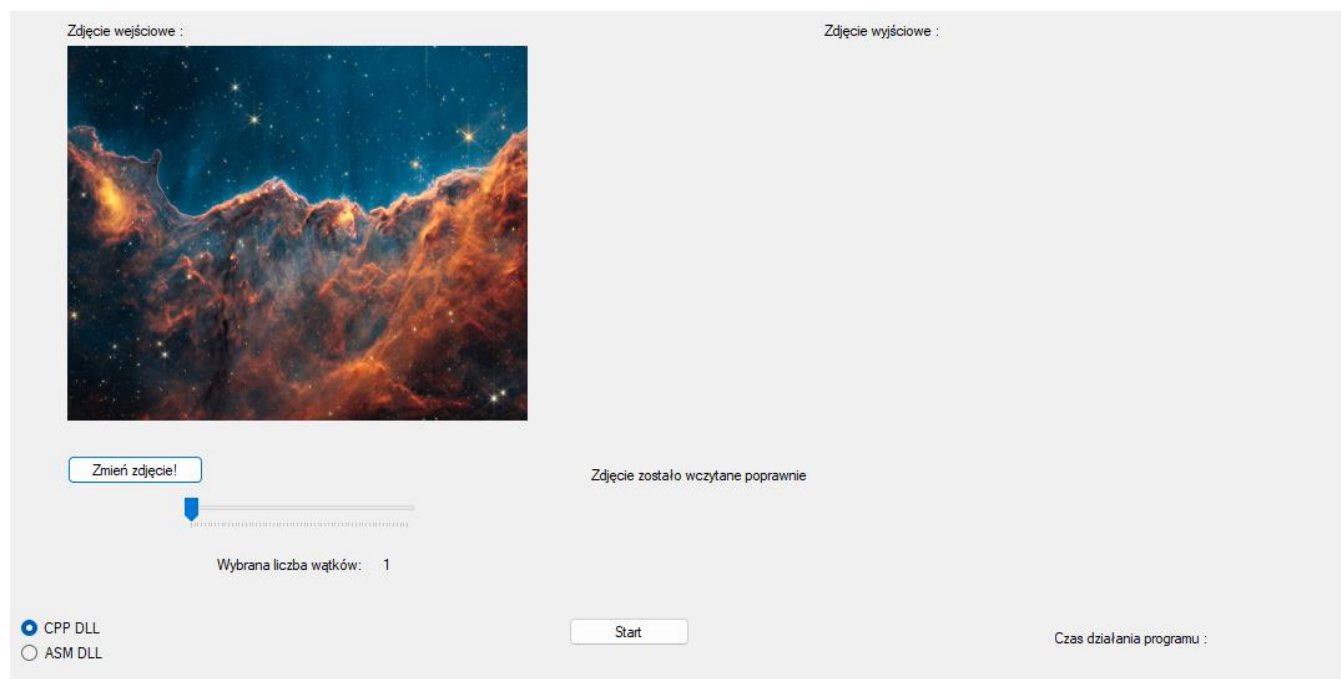
```
dec rdx                ;Szer -1 (potrzebne do warunkow if)
cmp r13, 0             ;Jezeli x = 0
je next               ;Skok przy spelnieniu warunku
cmp r13, rdx           ;Jezeli x = szerokosci
je next               ;Skok przy spelnieniu warunku
cmp r14, 0             ;Jezeli y = 0
je next               ;Skok przy spelnieniu warunku
cmp r14, r12           ;Jezeli y = wysokosci
je next               ;Skok przy spelnieniu warunku
```

Wybrałem ten fragment kodu asemblerowego, ze względu na skomplikowane przełożenie języka cpp na język asemblerowy. Próbowałem przekształcić jak najbardziej kod z języka cpp na kod asemblerowy, co okazało się złudne. W programie zapewniłem, że ramka wybranego obrazu nie wykona się, lecz mogło to zostać wykonane innym sposobem, np. przekształceniem pętli.

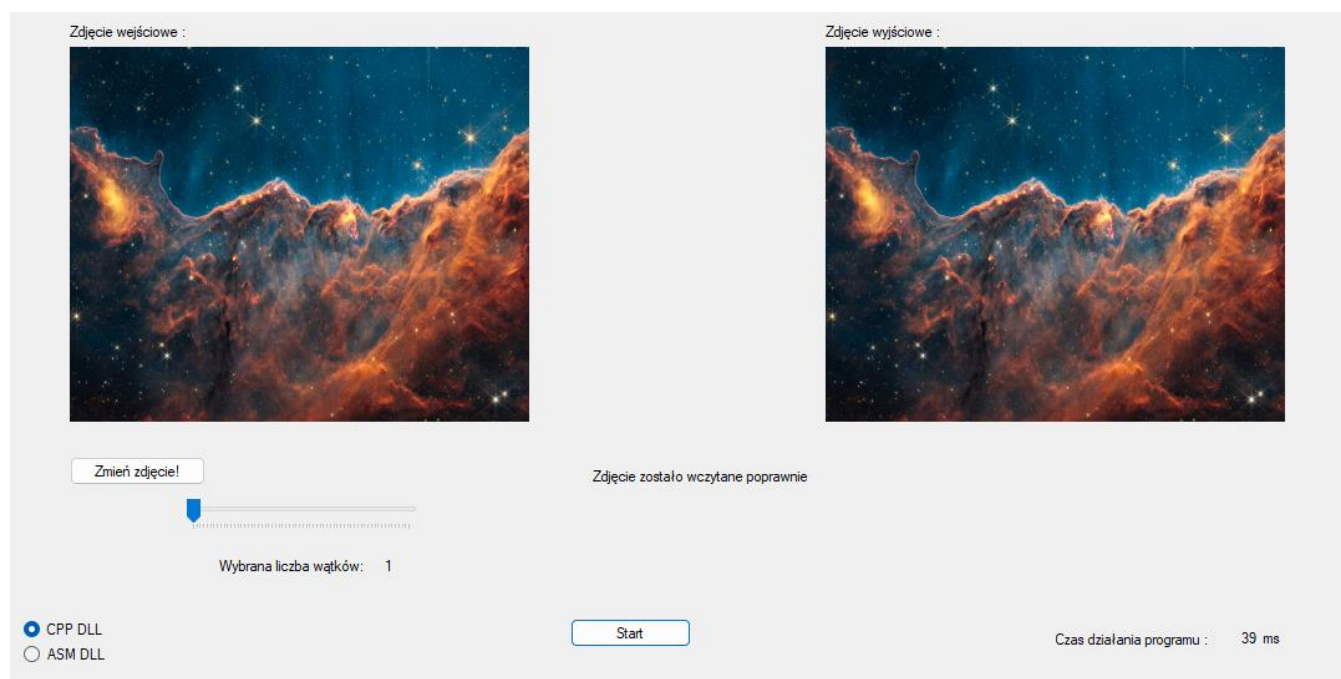
Zrzuty ekranu wyglądu UI



Rysunek 1. Start programu



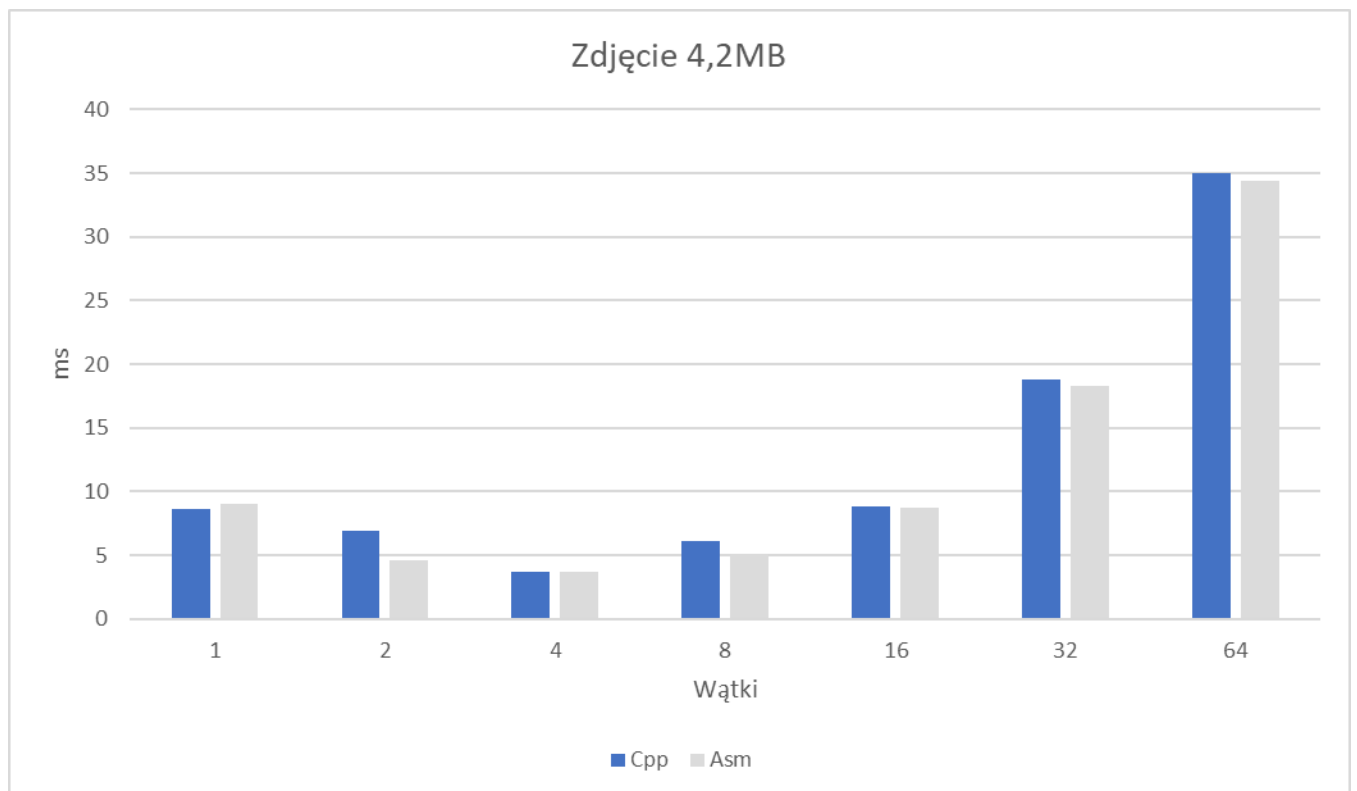
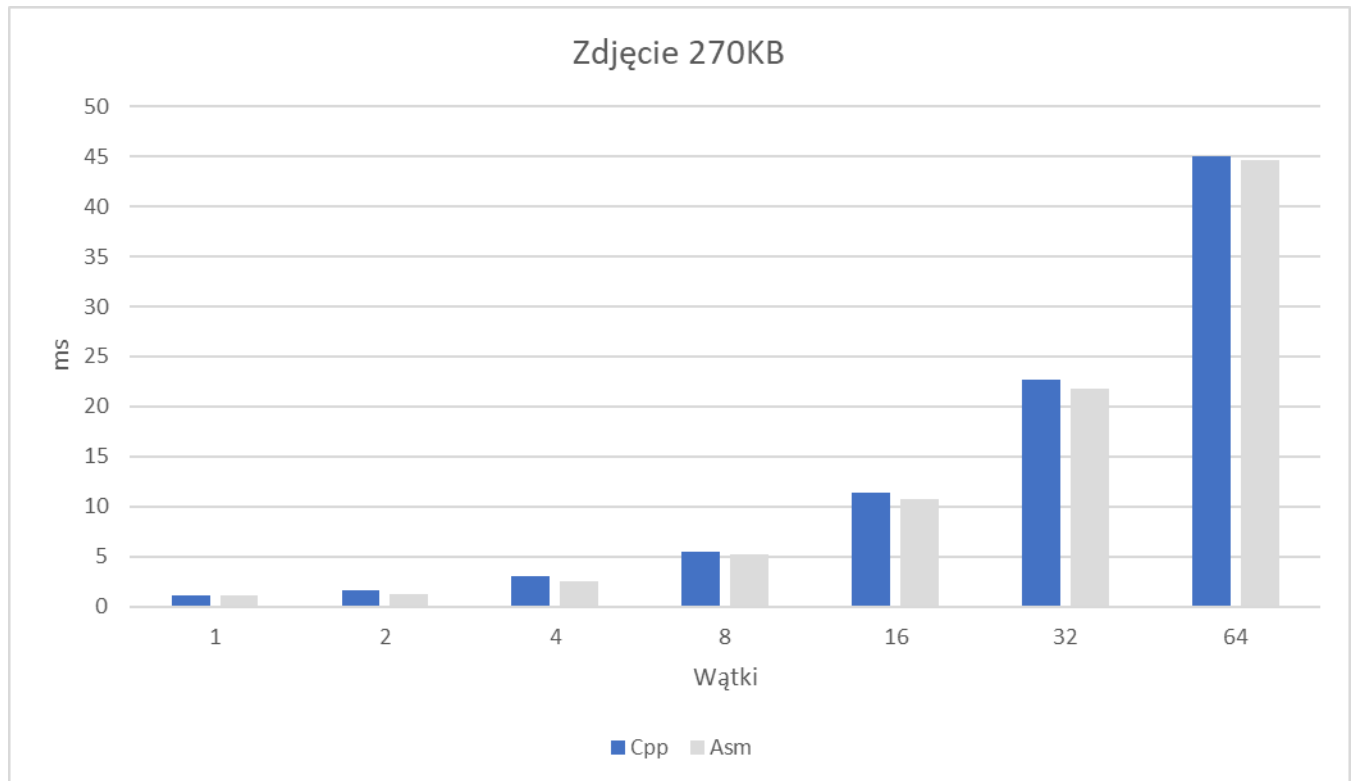
Rysunek 2. UI po wczytaniu zdjęcia

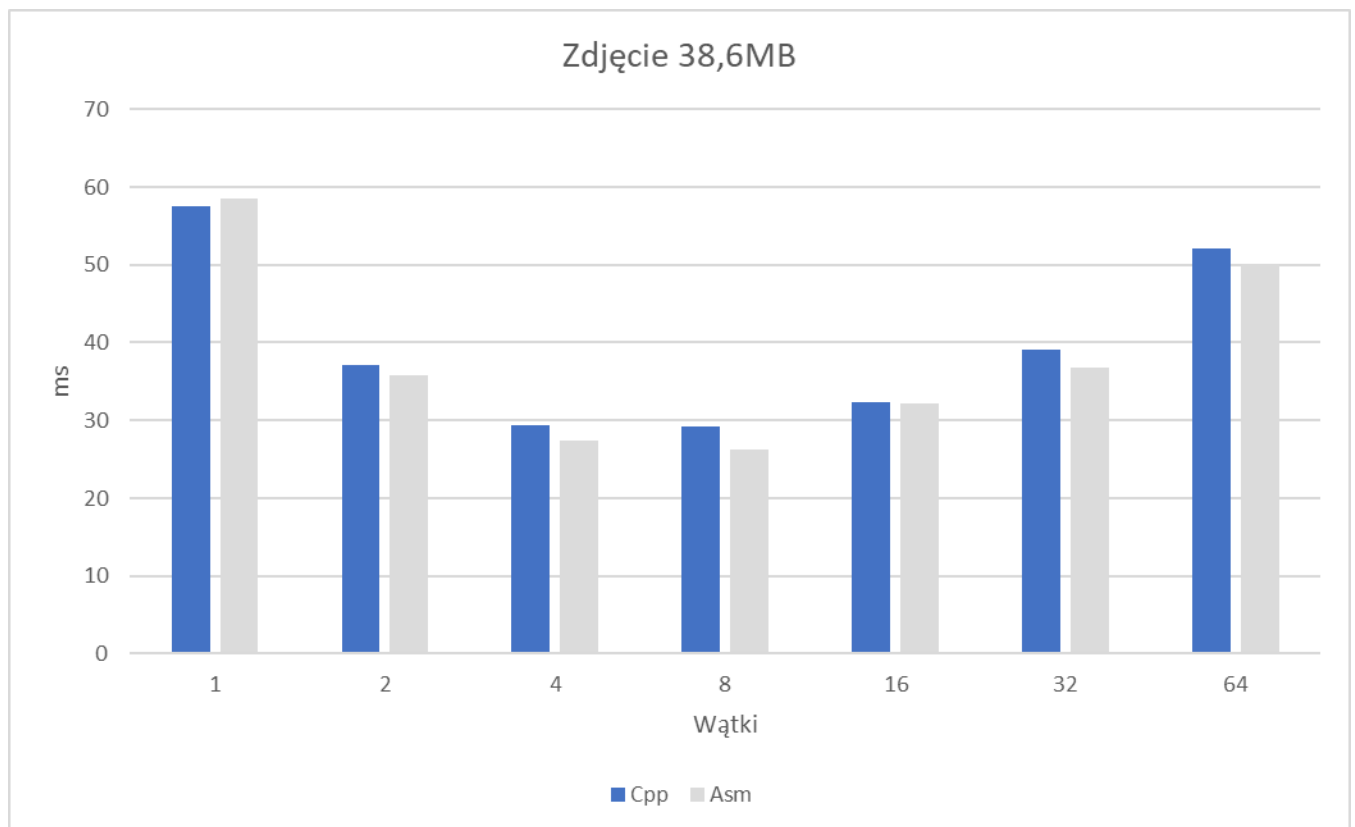


Rysunek 3. UI po wykonaniu programu

Raport szybkości działania

Dla CPU o 4 rdzeniach i 8 procesorach logicznych





Opis testowania i uruchamiania programu

Program został przetestowany dla wszystkich możliwych opcji (tj. ilości wątków, różnych formatów zdjęć, bibliotek). Program posiada zabezpieczenie przez brakiem podania zdjęcia oraz przed wczytaniem innego zdjęcia podczas trwania programu. W programie nie zapewniłem sprawdzenia poprawności formatu wczytanego zdjęcia.

UI wykonany jest przy pomocy Windows Forms.

Wnioski

Projekt z przedmiotu Języki Asemblerowe zapoznał mnie z tworzeniem biblioteki DLL oraz polepszył moje umiejętności, jeżeli chodzi o język assemblerowy. Dzięki projektowi nauczyłem się także optymalizacji kodu assemblerowego, projektowania UI przy użyciu Windows Forms, używania operacji wektorowych, wczytywania i zapisywania bitmapy oraz innych ważnych informacji, które były niezbędne przy tworzeniu projektu.

Uważam, że projekt był skomplikowany, ze względu na brak wcześniejszej znajomości tematu, lecz bardzo kształcący.

Interpretacja czasów

Czasy prezentowane przeze mnie w punkcie „Raport szybkości działania” odwzorowują idealnie jak powinny przebiegać czasy biorąc pod uwagę właściwości CPU, na którym przeprowadzane były testy.

- ❖ Dla małego zdjęcia tj. 270KB czas wzrasta przy wzroście ilości wątków, ze względu na rozdzielczość zdjęcia. Przy tak rozmiarowo małym zdjęciu wątki nie przyspieszają pracy, ani dla biblioteki Cpp, ani dla biblioteki asm.*
- ❖ Dla średniego zdjęcia tj. 4.2MB widać przyspieszenie pracy biblioteki cpp oraz biblioteki asm przy użyciu większej ilości wątków. Czas zaczyna wzrastać dla 8 wątków, co sugeruje że optymalna ilość wątków przy takim rozmiarze zdjęcia to 6.*
- ❖ Dla dużego zdjęcia tj. 36.6MB widać idealnie przyspieszenie dla wątków, co widoczne jest w punkcie „Raport szybkości działania”. Po 8 wątkach przyspieszenie zaczyna być mniejsze, co jest spowodowane właściwościami mojego CPU.*