

● Ważone dodawanie dwóch obrazów .bmp

Projekt z języków assemblerowych



„AGENDA”

- Założenia
- Czym jest i jak działa ważone dodawanie obrazów
- Opis interfejsu graficznego aplikacji
- Rola wątków w programie
- Implementacja w języku asemblera
- Wyniki pomiarów



Założenia

Teoria

Założenia

1. Stworzenie aplikacji w języku C# umożliwiającej:
 - a. załadowanie dwóch plików grafiki bitmapowej,
 - b. ustawienie określonej liczby wątków,
 - c. wyboru biblioteki dostarczającej funkcję do obróbki obrazu.

Aplikacja będzie odpowiedzialna również za podział obrazu na tyle części ile będzie wątków oraz wywołanie dla każdej z nich funkcji łączenia.

2. Napisanie biblioteki w języku C# udostępniająca funkcję umożliwiającą nakładanie obrazów przy wykorzystaniu algorytmu ważonego dodawania. Wynik działania będzie zapisany w tablicy bitmapy podanej jako image1.

Prototyp funkcji

```
void ManualBlend(int** bitmaps, int* coords, int alpha )
```

3. Napisanie biblioteki w języku asemblera udostępniającą i realizującą tą samą funkcjonalność co w/w biblioteka.

2

Czym jest i jak działa ważne dodawanie obrazów

Teoria

- Ważone dodawanie obrazów

○
$$Image(x,y) = W * Image1(x,y) + (1 - W) * Image2(x,y), \quad 0 \leq W \leq 1$$

$W = 0,3$



$W = 0,7$



Efekt przejścia pomiędzy 2 obrazami płynnie zmieniając $W: 0 \rightarrow 1$


3

Graficzny interfejs użytkownika


BlendApp

Ustawienia programu

BlendApp (c) Mateusz Szostok

 BlendApp

Ustawienia programu

Liczba wątków w programie  Rekomendowana liczba wątków : 7

Wybór algorytmu

☒ Biblioteka - C#
☐ Biblioteka - ASM

Kanał alpha $0 < \text{155} < 255$

Wybierz obrazy

Obraz 1 ...

Obraz 2 ...

Połącz



4

Rola wątków w programie

BlendApp

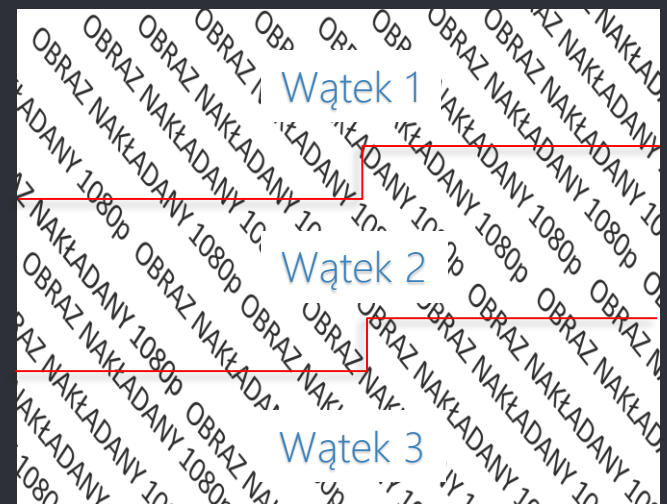
● Wątki a obraz

○ Obrazy dzielone są na tyle części ile użytkownik wybierze wątków

Liczba wątków : 3



Obraz bazowy



Obraz nakładany

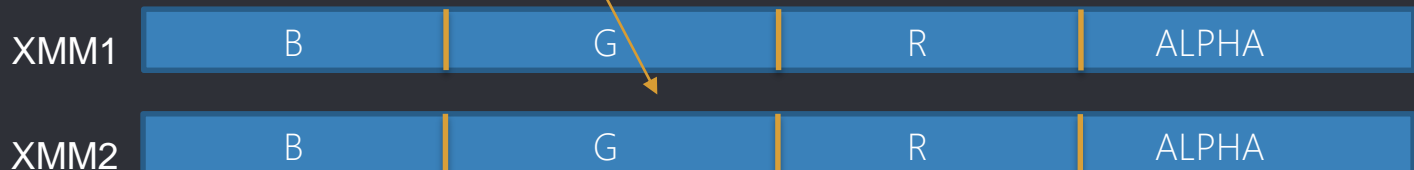
5

Implementacja w języku asemblera

Bibliotek ASM – główny algorytm

Bibliotek ASM – główny algorytm

```
198 ;===== WAŻNE NAKŁADANIE OBRAZÓW =====
199 mov ecx, start ; zainicjalizowanie licznika pętli
200 blendLoop:
201 MOVDQU xmm1, [ebx + ecx*SIZEOF DWORD] ; załadowanie 4 wartości (Move unaligned double quad words)
202 MOVDQU xmm2, [edx + ecx*SIZEOF DWORD] ; załadowanie 4 wartości (Move unaligned double quad words)
203
204 CVTDQ2PS xmm1, xmm1 ; konwersja z double word na single-precision float
205 CVTDQ2PS xmm2, xmm2 ; konwersja z double word na single-precision float
206
207 MULPS xmm1, xmm6 ; mnożenie równoległe alphaBottom*bitmapBottom(x,y)
208 MULPS xmm2, xmm5 ; mnożenie równoległe alphaTop*bitmapTop(x,y)
209
210 ADDPS xmm1, xmm2 ; dodanie wartości rgb o odpowiednich wagach
211
212 CVTTPS2DQ xmm1, xmm1 ; konwersja z single-precision na double words z obcięciem
213 CVTTPS2DQ xmm2, xmm2 ; konwersja z single-precision na double words z obcięciem
214
215 MOVDQU [rgba], xmm1 ; pobranie wartości do struktury w celu wyłuskania parametru 'a'
216 mov [rgba.Pixel].a, 255 ; przywrócenie domyślnej wartości parametru 'a'
217 MOVDQU xmm1, [rgba] ; załadowanie poprawnych wartości do rejestru
218
219 MOVDQU [ebx + ecx*SIZEOF DWORD], xmm1 ; nadpisanie wartości w komórkach bitmapy bazowej
220
221 add ecx, bytesPerPixel ; zwiększenie licznika o jeden piksel który został już obliczony
222 cmp ecx, stop ; sprawdzenie czy należy już skończyć
223 jne blendLoop ; jeśli ecx != stop to wróć na początek pętli, w przeciwnym wypadku
224 ; zakończ obliczenia
```



Bibliotek ASM – główny algorytm

```
198 ;===== WAŻNE NAKŁADANIE OBRAZÓW =====
199 mov ecx, start ; zainicjalizowanie licznika pętli
200 blendLoop:
201 MOVDQU xmm1, [ebx + ecx*SIZEOF DWORD] ; załadowanie 4 wartości (Move unaligned double quad words)
202 MOVDQU xmm2, [edx + ecx*SIZEOF DWORD] ; załadowanie 4 wartości (Move unaligned double quad words)
203
204 CVTDQ2PS xmm1, xmm1 ; konwersja z double word na single-precision float
205 CVTDQ2PS xmm2, xmm2 ; konwersja z double word na single-precision float
206
207 MULPS xmm1, xmm6 ; mnożenie równoległe alphaBottom*bitmapBottom(x,y)
208 MULPS xmm2, xmm5 ; mnożenie równoległe alphaTop*bitmapTop(x,y)
209
210 ADDPS xmm1, xmm2 ; dodanie wartości rgb o odpowiednich wagach
211
212 CVTTPS2DQ xmm1, xmm1 ; konwersja z single-precision na double words z obcięciem
213 CVTTPS2DQ xmm2, xmm2 ; konwersja z single-precision na double words z obcięciem
214
215 MOVDQU [rgba], xmm1 ; pobranie wartości do struktury w celu wyłuskania parametru 'a'
216 mov [rgba.Pixel].a, 255 ; przywrócenie domyślnej wartości parametru 'a'
217 MOVDQU xmm1, [rgba] ; załadowanie poprawnych wartości do rejestru
218
219 MOVDQU [ebx + ecx*SIZEOF DWORD], xmm1 ; nadpisanie wartości w komórkach bitmapy bazowej
220
221 add ecx, bytesPerPixel ; zwiększenie licznika o jeden piksel który został już obliczony
222 cmp ecx, stop ; sprawdzenie czy należy już skończyć
223 jne blendLoop ; jeśli ecx != stop to wróć na początek pętli, w przeciwnym wypadku
224 ; zakończ obliczenia
```

XMM1 B * alphaBottom G * alphaBottom R * alphaBottom A * alphaBottom

XMM2 B * alphaBottom G * alphaBottom R * alphaBottom A * alphaBottom

Bibliotek ASM – główny algorytm

```

198 ;===== WAŻNE NAKŁADANIE OBRAZÓW =====
199 mov ecx, start ; zainicjalizowanie licznika pętli
200 blendLoop:
201 MOVDQU xmm1, [ebx + ecx*SIZEOF DWORD] ; załadowanie 4 wartości (Move unaligned double quad words)
202 MOVDQU xmm2, [edx + ecx*SIZEOF DWORD] ; załadowanie 4 wartości (Move unaligned double quad words)
203
204 CVTDQ2PS xmm1, xmm1 ; konwersja z double word na single-precision float
205 CVTDQ2PS xmm2, xmm2 ; konwersja z double word na single-precision float
206
207 MULPS xmm1, xmm6 ; mnożenie równoległe alphaBottom*bitmapBottom(x,y)
208 MULPS xmm2, xmm5 ; mnożenie równoległe alphaTop*bitmapTop(x,y)
209
210 ADDPS xmm1, xmm2 ; dodanie wartości rgb o odpowiednich wagach
211
212 CVTTPS2DQ xmm1, xmm1 ; konwersja z single-precision na double words z obcięciem
213 CVTTPS2DQ xmm2, xmm2 ; konwersja z single-precision na double words z obcięciem
214
215 MOVDQU [rgba], xmm1 ; pobranie wartości do struktury w celu wyłuskania parametru 'a'
216 mov [rgba.Pixel].a, 255 ; przywrócenie domyślnej wartości parametru 'a'
217 MOVDQU xmm1, [rgba] ; załadowanie poprawnych wartości do rejestru
218
219 MOVDQU [ebx + ecx*SIZEOF DWORD], xmm1 ; nadpisanie wartości w komórkach bitmapy bazowej
220
221 add ecx, bytesPerPixel ; zwiększenie licznika o jeden piksel który został już obliczony
222 cmp ecx, stop ; sprawdzenie czy należy już skończyć
223 jne blendLoop ; jeśli ecx != stop to wróć na początek pętli, w przeciwnym wypadku
224 ; zakończ obliczenia

```

XMM1 B_xmm1 + B_xmm2 G_xmm1 + G_xmm1 R_xmm1 + R_xmm2 A_xmm1 + A_xmm2

Bibliotek ASM – główny algorytm

```
198 ;===== WAŻONE NAKŁADANIE OBRAZÓW =====
199 mov ecx, start ; zainicjalizowanie licznika pętli
200 blendLoop:
201 MOVDQU xmm1, [ebx + ecx*SIZEOF DWORD] ; załadowanie 4 wartości (Move unaligned double quad words)
202 MOVDQU xmm2, [edx + ecx*SIZEOF DWORD] ; załadowanie 4 wartości (Move unaligned double quad words)
203
204 CVTDQ2PS xmm1, xmm1 ; konwersja z double word na single-precision float
205 CVTDQ2PS xmm2, xmm2 ; konwersja z double word na single-precision float
206
207 MULPS xmm1, xmm6 ; mnożenie równoległe alphaBottom*bitmapBottom(x,y)
208 MULPS xmm2, xmm5 ; mnożenie równoległe alphaTop*bitmapTop(x,y)
209
210 ADDPS xmm1, xmm2 ; dodanie wartości rgb o odpowiednich wagach
211
212 CVTTPS2DQ xmm1, xmm1 ; konwersja z single-precision na double words z obcięciem
213 CVTTPS2DQ xmm2, xmm2 ; konwersja z single-precision na double words z obcięciem
214
215 MOVDQU [rgba], xmm1 ; pobranie wartości do struktury w celu wyłuskania parametru 'a'
216 mov [rgba.Pixel].a, 255 ; przywrócenie domyślnej wartości parametru 'a'
217 MOVDQU xmm1, [rgba] ; załadowanie poprawnych wartości do rejestru
218
219 MOVDQU [ebx + ecx*SIZEOF DWORD], xmm1 ; nadpisanie wartości w komórkach bitmapy bazowej
220
221 add ecx, bytesPerPixel ; zwiększenie licznika o jeden piksel który został już obliczony
222 cmp ecx, stop ; sprawdzenie czy należy już skończyć
223 jne blendLoop ; jeśli ecx != stop to wróć na początek pętli, w przeciwnym wypadku
224 ; zakończ obliczenia
```

XMM1

B

G

R

255

6

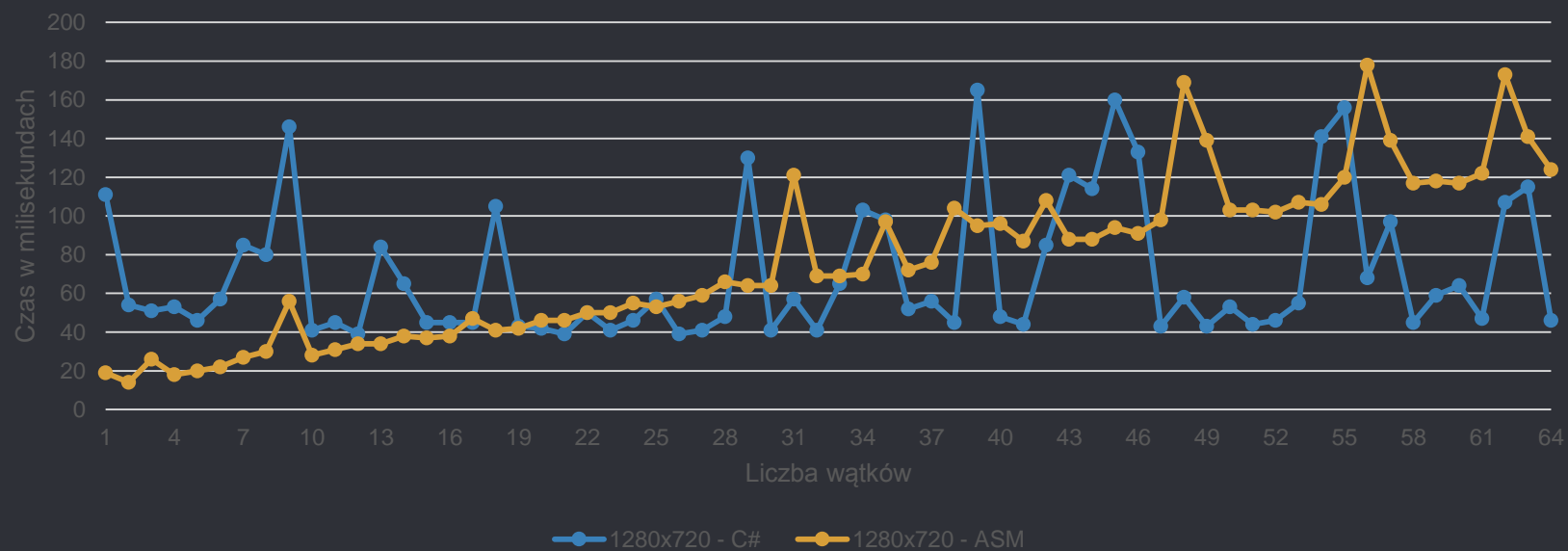
Wyniki pomiarów

Wykresy dla różnych rozdzielczości

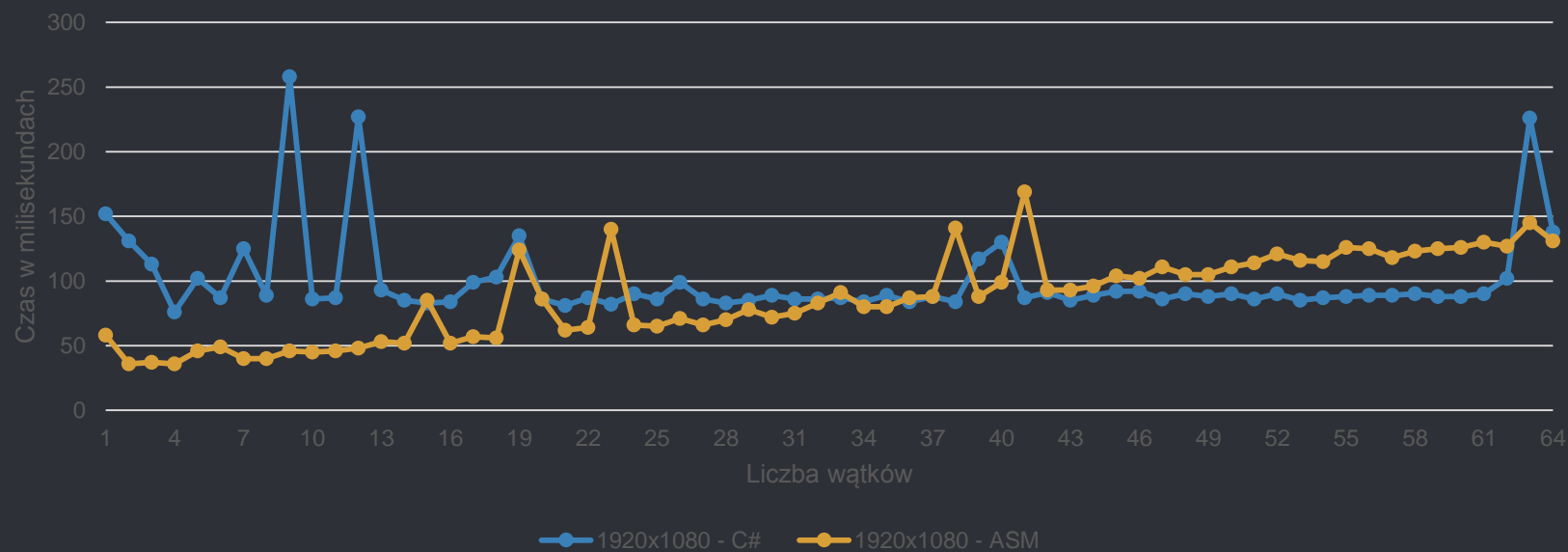


Intel® Core™2 Duo Processor T5500
(2M Cache, 1.66 GHz, 667 MHz FSB)

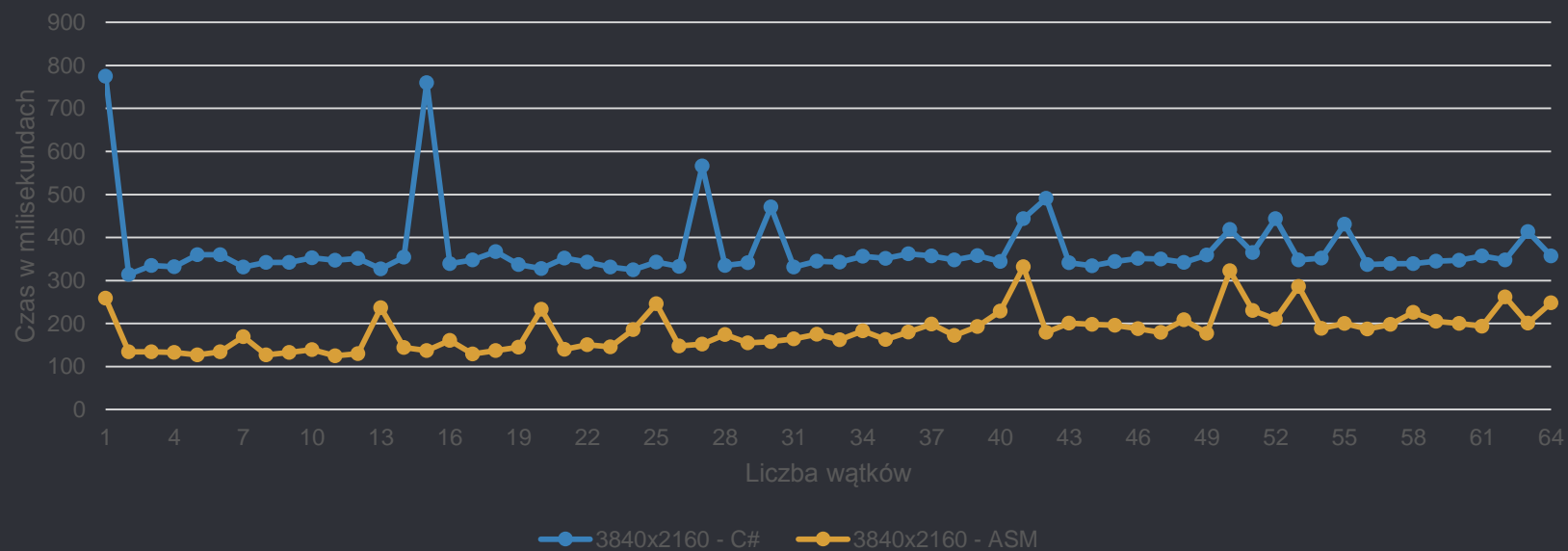
Wyniki dla rozdzielczości HD




Wyniki dla rozdzielczości FHD



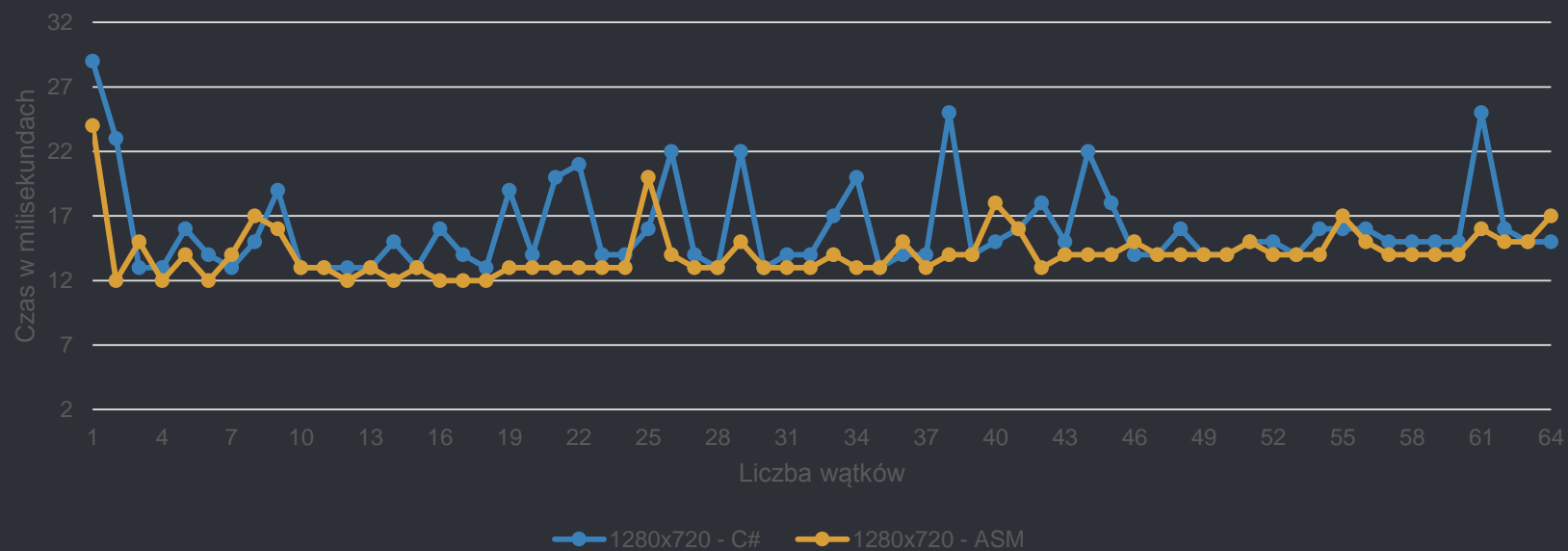
Wyniki dla rozdzielczości 3840x2160



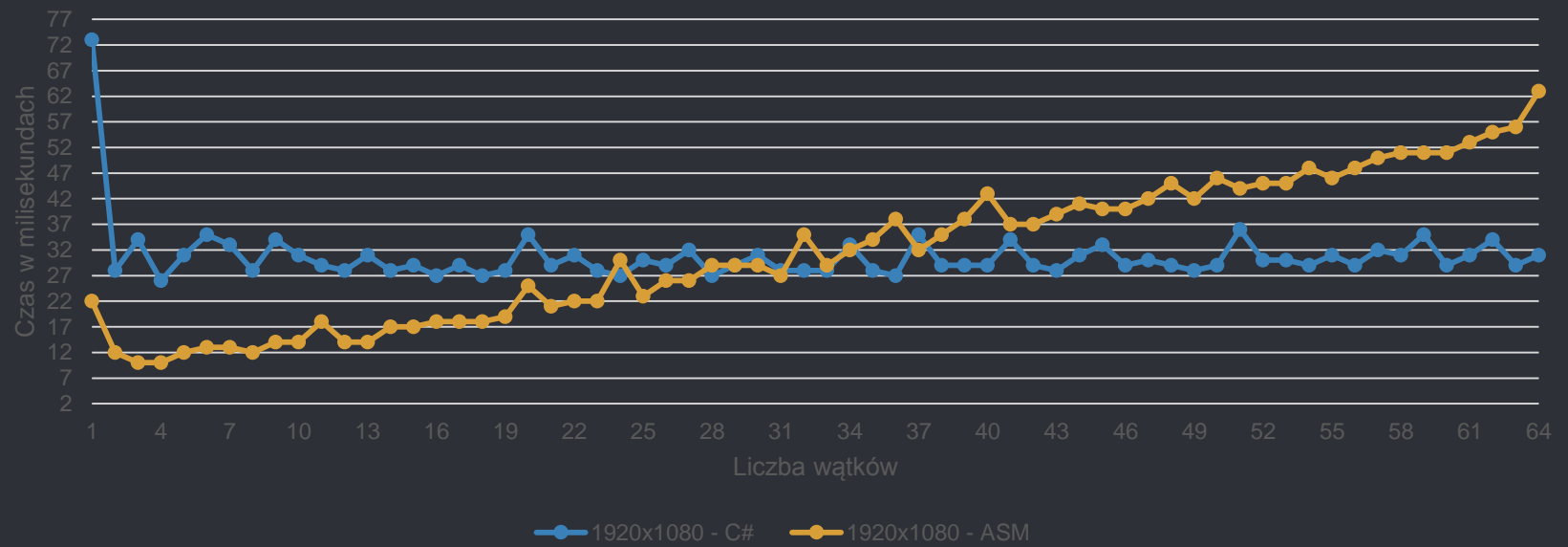


Intel® Core™ i5-5200U Processor
(3M Cache, up to 2.70 GHz)

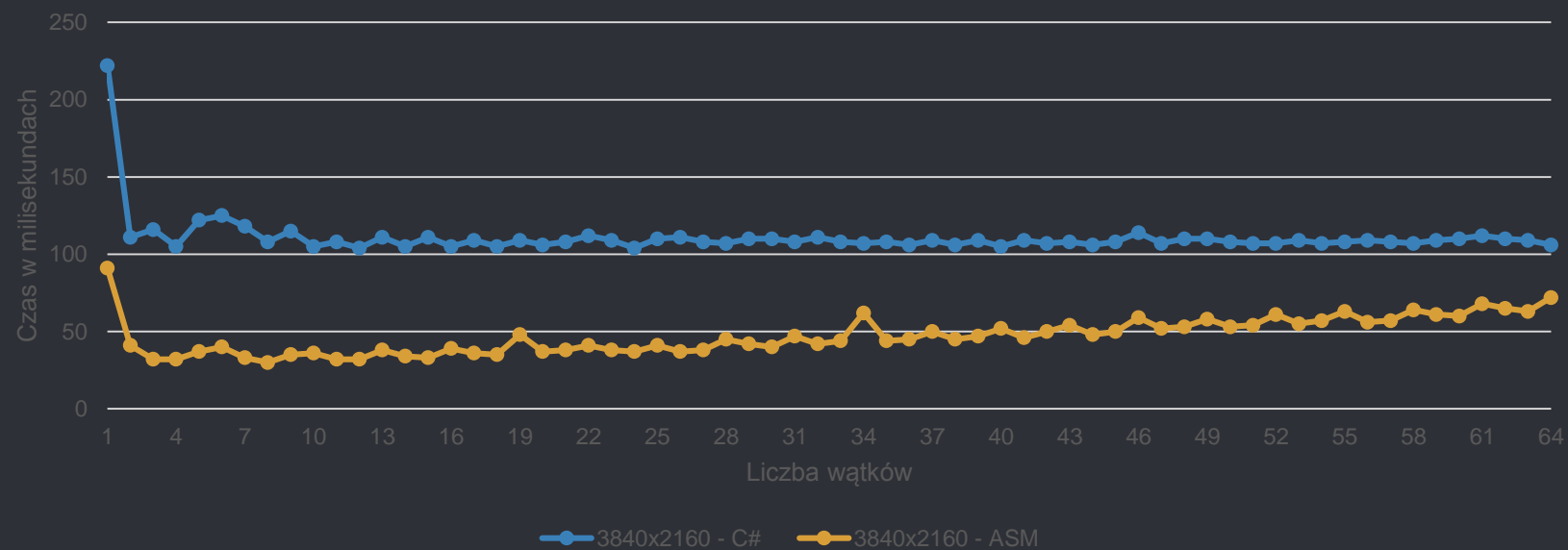
Wyniki dla rozdzielczości HD



Wyniki dla rozdzielczości FHD



Wyniki dla rozdzielczości 3840x2160



- KONIEC