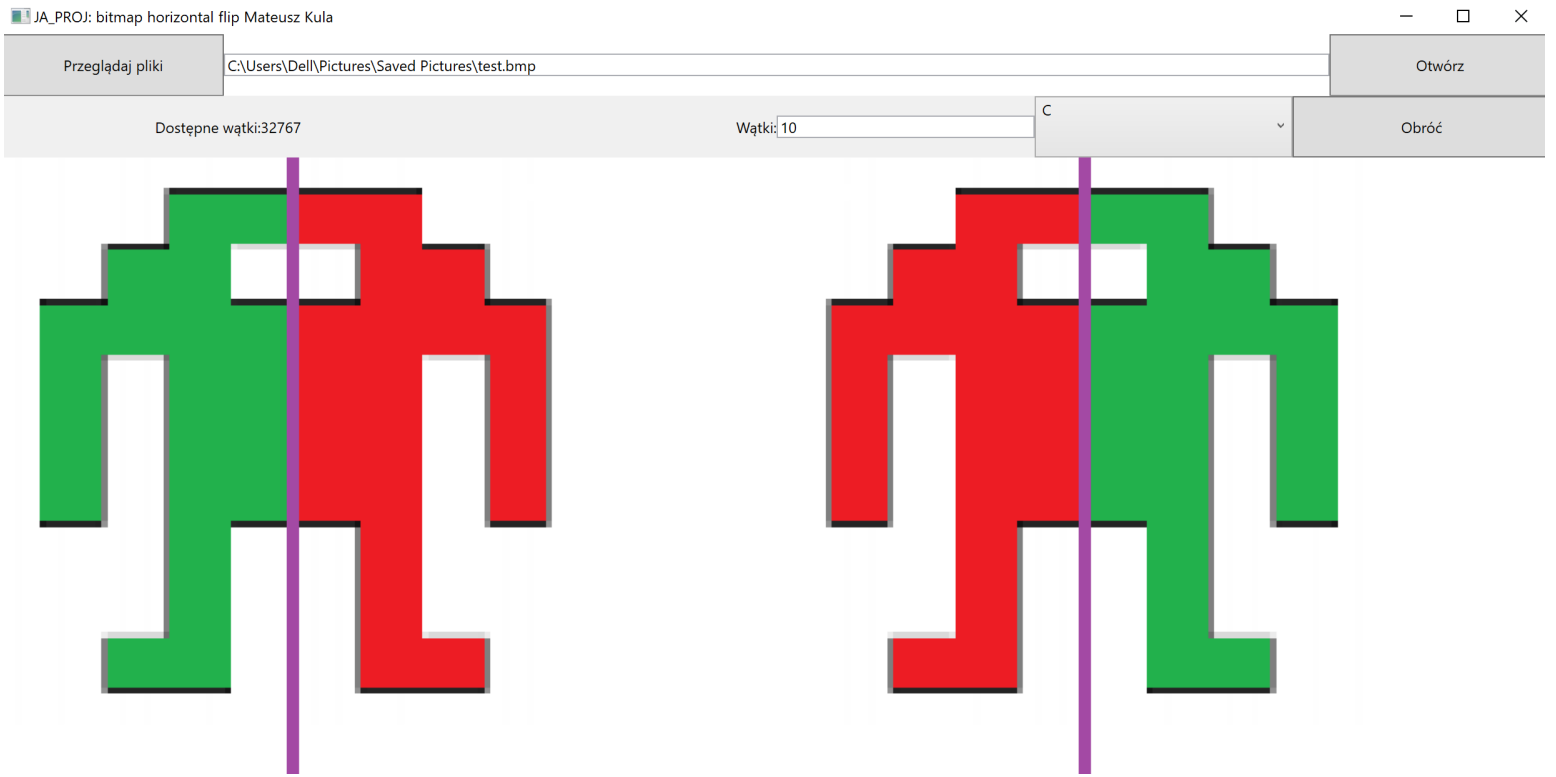


Obracanie bitmapy

Języki Asemblerowe

Mateusz Kula

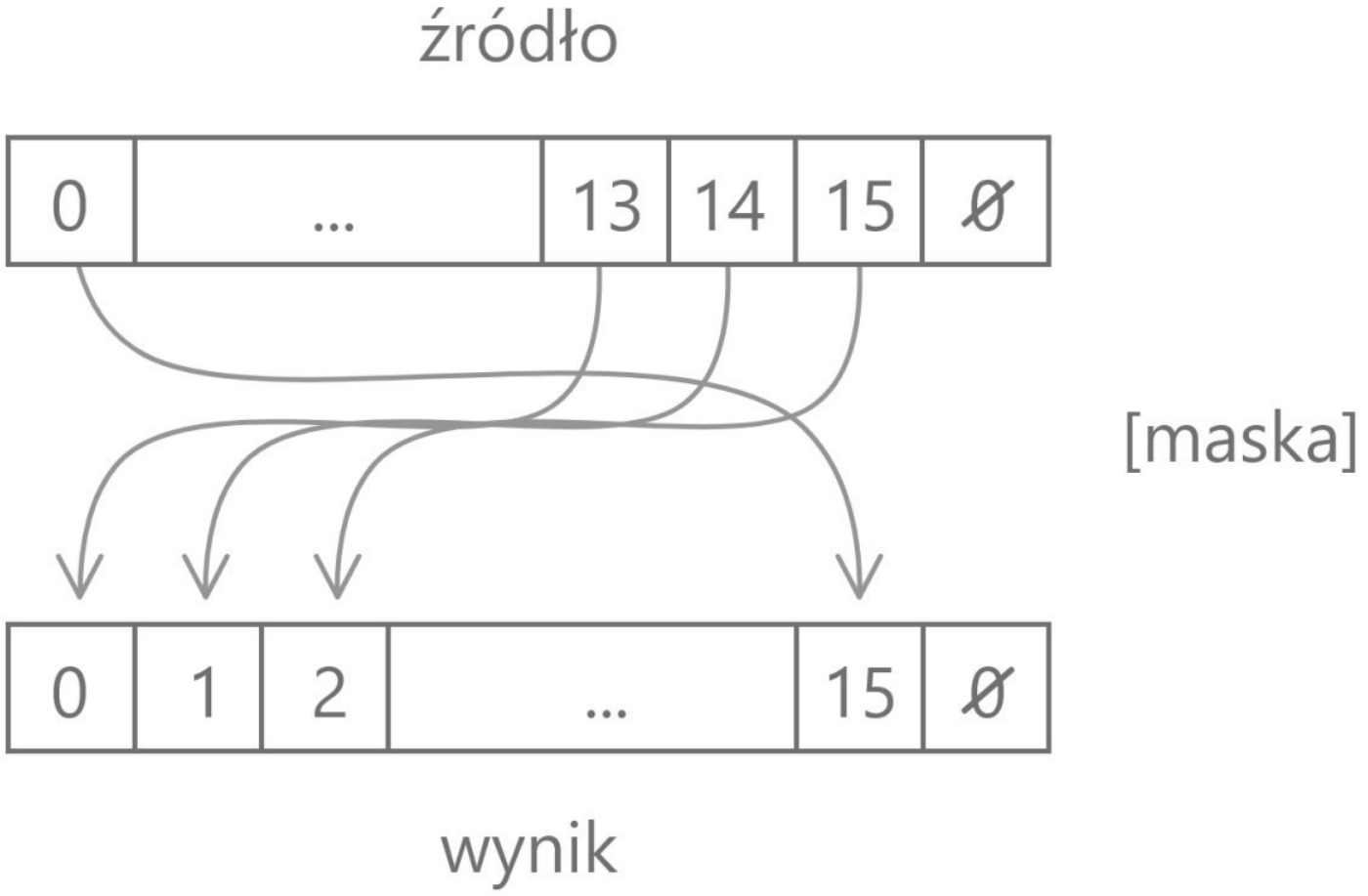
Założenia projektowe

- Program z GUI w C# + WPF, który wczyta bitmapę
- Dllki w C i Asemblerze które przetworzą bitmapę
- Wykorzystanie operacji wektorowych
- Przetwarzanie bitmapy polega na obróceniu jej wokół osi pionowej

Wykorzystane instrukcje Asm

- MOVQ xmm1, xmm2/m64
- MOVLHPS xmm1, xmm2
- MOVDQA xmm1, xmm2
- VPSHUFB xmm1, xmm2, xmm3

VPSHUFB



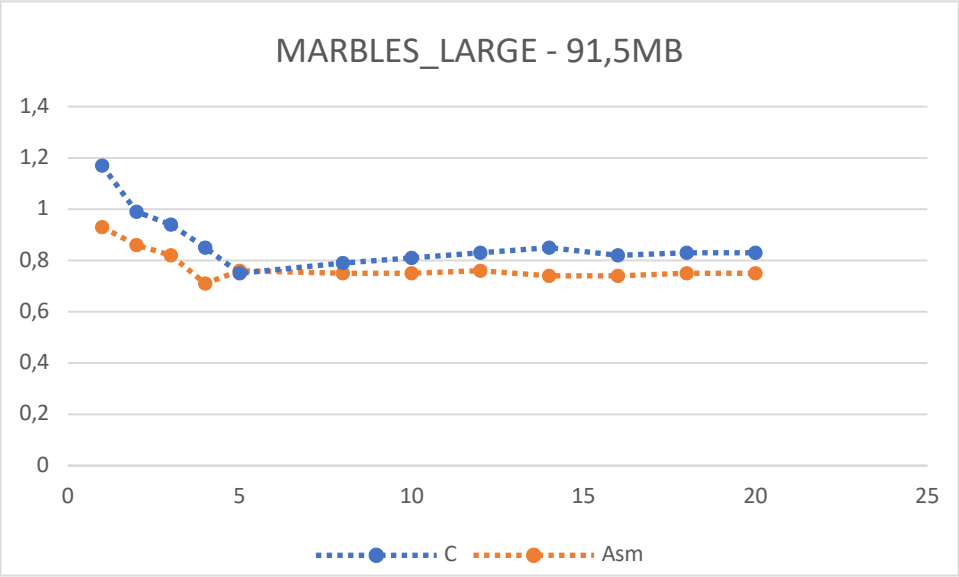
Implementacja Asmebler

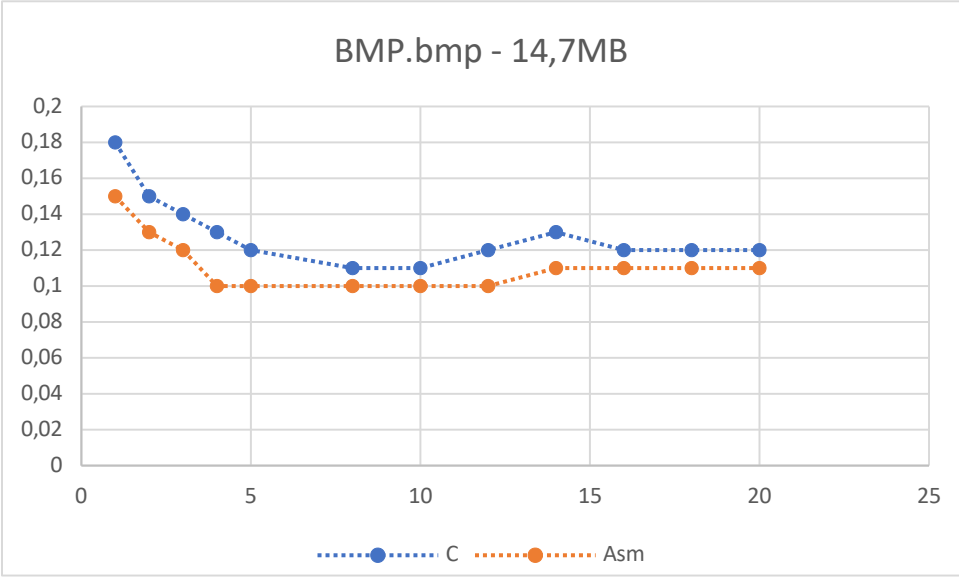
```
_fori: ;petla dla czytania po 5 pikseli- 15 bajtow

    cmp r14,r13  ;i+5<szer
    jae _endfori

    movdqa xmm2,[r9]    ;czyta ostatnie 5 pikseli

    vpshufb xmm1,xmm2,xmm15 ;zapis do xmm1 danych z xmm2 tak jak wskazuje maska w xmm15

    movdqa [r10],xmm1   ;zapisuje 5 pikseli

    add r10,15  ;przesuwa wskaźnik zapisu o +15
    sub r9,15   ;przesuwa wskaźnik poboru o -15

    add r14,5   ;przesuwa licznik o 5 pikseli
    jmp _fori

_endfori:
```

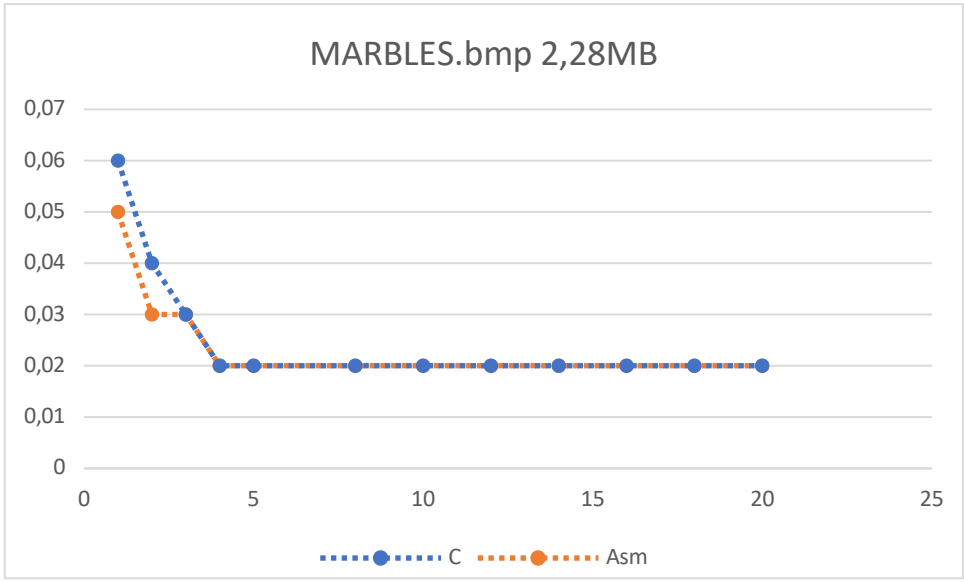
Implementacja Asmebler

```
    sub r14,5
    _forj: ;petla dla czytania po pikselu- 1 bajt

        cmp r14,r13
        jae _endforj

        movzx eax, byte ptr[r9]          ; R
        movzx ebx, byte ptr[r9-1]        ; G Czytanie piksela
        movzx ecx, byte ptr[r9-2]        ; B

        mov byte ptr[r10],al             ; R
        mov byte ptr[r10+1],bl           ; G Zapis piksela
        mov byte ptr[r10+2],cl           ; B

        add r10,3   ;przesuwa wskaźnik zapisu o +3
        sub r9,3    ;przesuwa wskaźnik poboru o -3

        inc r14     ;inkrementuje licznik o 1 piksel

        jmp _forj
_endforj:
```

Porównanie wydajności







Wnioski

- Wykorzystanie asemblera w projekcie drastycznie zwiększa ilość godzin potrzebnych na implementacje pomimo niewielkiego zysku na wydajności
- Kompilator w dużym stopniu optymalizuje kod
- Użycie dynamicznie linkowanych bibliotek pozwala na „odchudzenie” głównego projektu