Optymalizacja kodu na różne architektury

Adrian Madej 20.05.2024

1. Procesor

1.1. Sprawdzenie parametrów

Najpierw sprawdzamy model procesora, a następnie szukamy wybranych parametrów w Internecie.

1.2. Obliczenia

Znając architekturę procesora odszukujemy standard stosowany w moim procesorze - FP32.

Na tej podstawie, zgodnie ze wzorem z PlotAll.m z poprzedniego sprawozdania jesteśmy w stanie obliczyć:

$$GFLOPS = nflops\_per\_cycle \times nprocessors \times GHz\_of\_processors$$

1.3 Parametry

Uzyskane parametry zbieramy w tabelę

Parametr	Wartość
Producent	AMD Ryzen
Model	5 5600H
Mikroarchitektura	Cezanne-H (Zen 3)
Ilość rdzeni	6
Ilość wątków	12
Częstotliwość bazowa	3.3 GHz
Częstotliwość turbo	4.2 GHz
Cache	16 MB
FP32	32
GFLOPS	633,6
GFLOPS/Rdzeń	105,6

Tabela 1. Parametry procesora

2. Optymalizacje

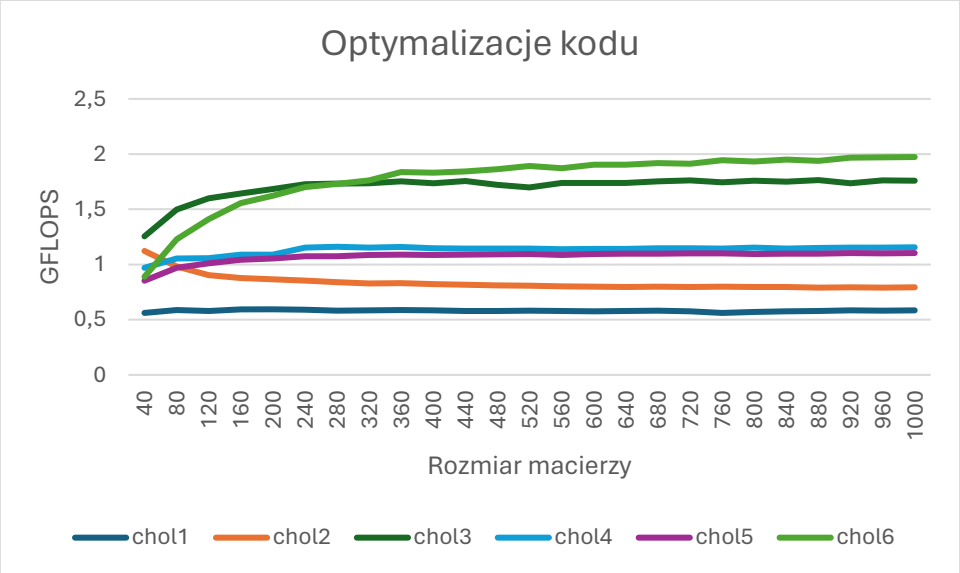
2.1 Optymalizacje kodu

- Chol1 – brak optymalizacji
- Chol2 – zastosowanie rejestrów
- Chol3 – ręczne rozwinięcie pętli o 8 iteracji
- Chol4 – zastosowanie wektorów __m128d
- Chol5 – ręczne rozwinięcie pętli do 16 iteracji
- Chol6 – zastosowanie wektorów __m256d

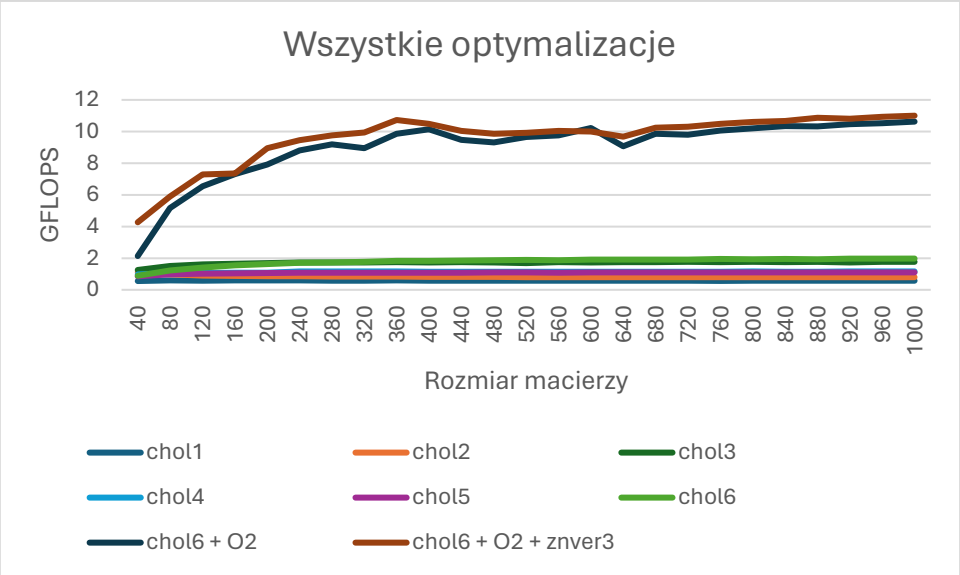
2.2 Kompilacja z dodatkowymi flagami

- Chol6 + O2 – dodanie flagi O2
- Chol6 + O2 + znver3 – dodanie flagi march = znver3

3. Wyniki

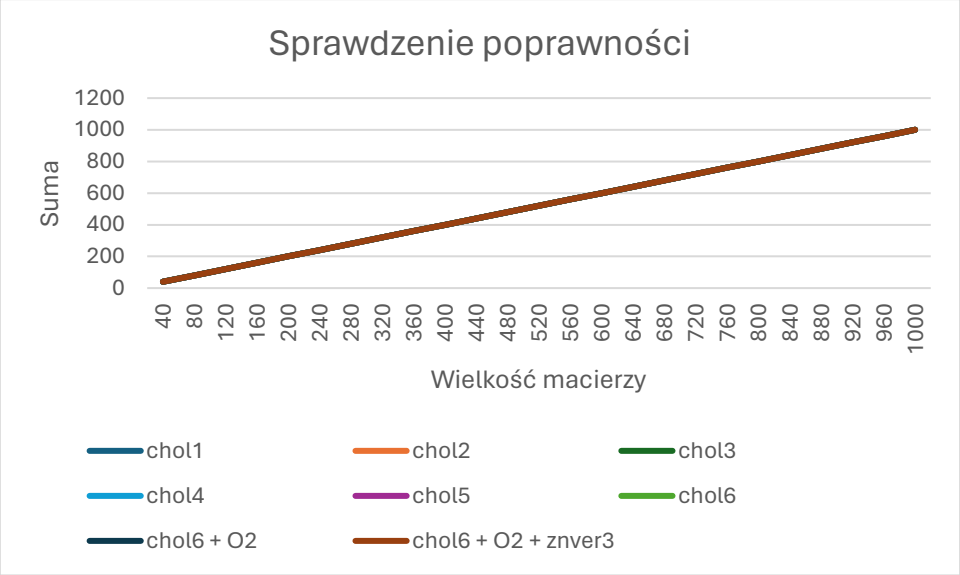


Wykres 1. Optymalizacje kodu

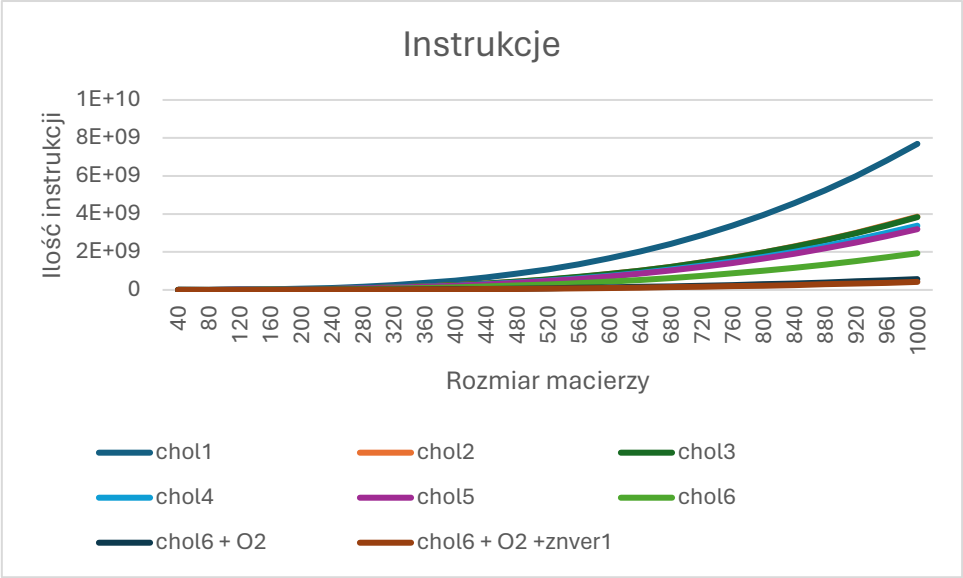


Wykres 2. Wszystkie optymalizacje

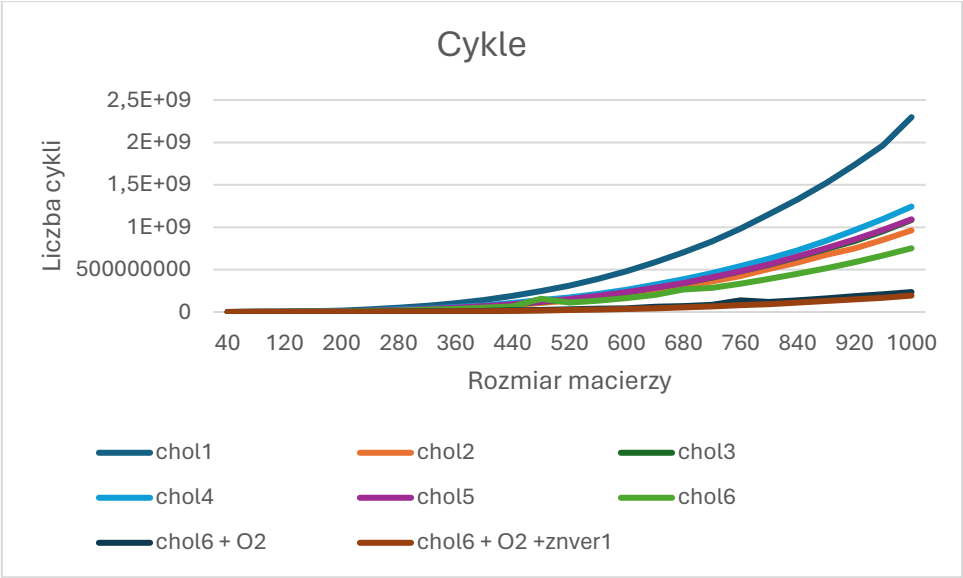
W celu weryfikacji poprawności sprawdzamy czy macierze po wykonaniu faktoryzacji są identyczne, tzn. czy ich elementy są takie same. W trakcie testów nie znaleziono różnic w wynikach co świadczy o poprawności optymalizacji kodu. Na wykresie przedstawiono sumę elementów macierzy dla danego rozmiaru; jak widać sumy pokrywają się.



Wykres 3. Sprawdzenie poprawności



Wykres 4. Ilość instrukcji



Wykres 5. Liczba cykli

4. Wnioski

4.1 Analiza wyników

- Najwyższy wynik GFLOPS uzyskany w trakcie obliczeń był równy 11 co stanowi 10,4% teoretycznej wartości; mogły mieć na to wpływ m. in. inne procesy w systemie, które ograniczyły dostępność mocy obliczeniowej procesora.
- Rozwinięcie pętli do 8 iteracji przyniosło zauważalną poprawę.
- Najwyższe wyniki GFLOPS oraz najmniejszą ilość instrukcji i cykli uzyskaliśmy przy zastosowaniu wektorów __m256d.
- Zastosowanie flagi optymalizacyjnej O2 przy kompilacji znacznie przyspieszyło działanie kodu.

4.2 Wnioski ogólne

- Umieszczanie zmiennych w rejestrze może znacznie przyspieszyć działanie programu
- Ręczne rozwinięcie pętli może znacząco zwiększyć wydajność
- Zastosowanie wektorów __256d znacznie przyspiesza wydajność
- Flaga 02 znacznie przyspiesza wykonywanie programu, zmniejsza ilość instrukcji oraz cyklów

5. Źródła

- Gitub - Home · flame/how-to-optimize-gemm Wiki (github.com)
- Specyfikacja procesora - AMD Ryzen™ 5 5600H Drivers & Support | AMD
- FLOPS - FLOPS - Wikipedia
- Co oznacza FLOPS - Gflops real world meaning | Overclockers UK Forums
- Faktoryzacje Choleskiego - Cholesky Factorization on SIMD multi-core architectures (hal.science), lectures.dvi (puc-rio.br)