Sprawozdanie

Zadanie domowe 1

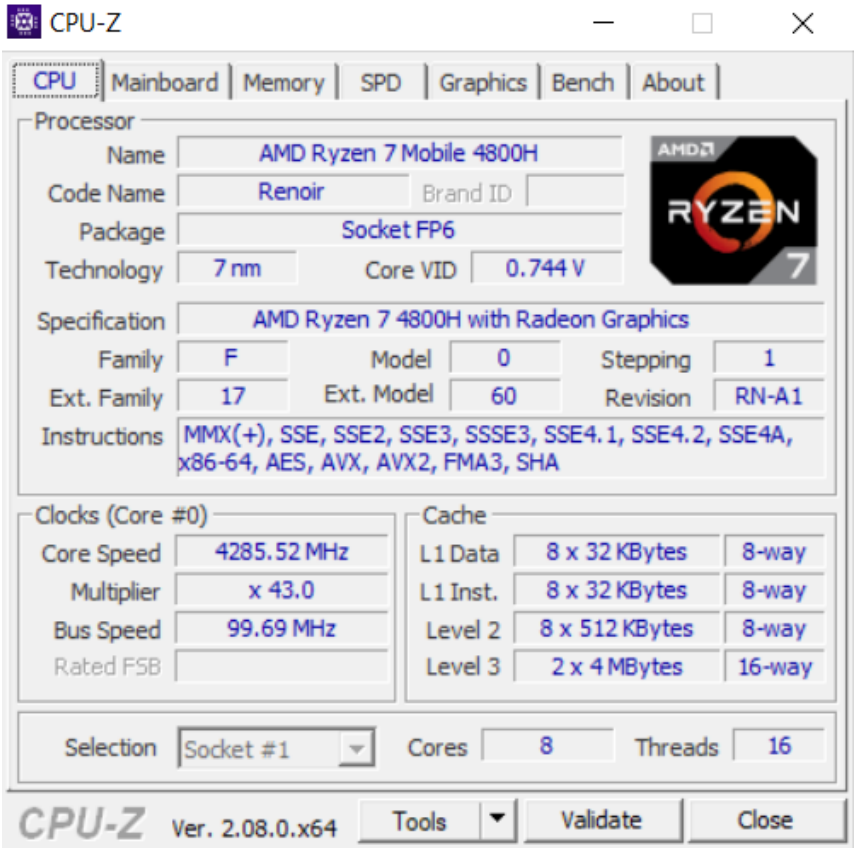
„How to optimize Gemm”

Autor: Adrian Żerebiec

1 – Procesor komputera

a) Parametry

Z pomocą programu CPU_Z odczytujemy następujące wartości:



Architektura	Renoir (Zen2)
Rdzenie	8
Wątki	16
Częstotliwość bazowa	2.9 GHz
Częstotliwość maksymalna	4.2 GHz
GFLOPS/Rdzeń	46.4
GFLOPS	371.2

b) Wyznaczanie GFLOPS/Rdzeń

Aby wyznaczyć wartość wykorzystałem wzór znajdujący się w pliku PlotAll.py, czyli

nflops_per_cycle * nprocessors * GHz_of_processor = 16 * 1 * 2,9 = 46,4

Wartość nflops_per_cycle dostępna jest w Internecie:

Microarchitecture	Instruction set architecture	FP64	FP32	FP16
AMD Zen 2[19] (Ryzen 3000 series, Threadripper 3000 series, Epyc Rome)) AMD Zen 3 (Ryzen 5000 series, Epyc Milan)	AVX2 & FMA (256-bit)	16	32	0

Do obliczeń wybieramy wartość FP64, gdyż jest on używany do klasyfikacji.

2 – Optymalizacja

a) Optymalizacja opisana w „How to Optimize Gemm”

Jako N w poniższym opisie traktujemy jako N=1 lub N=4.

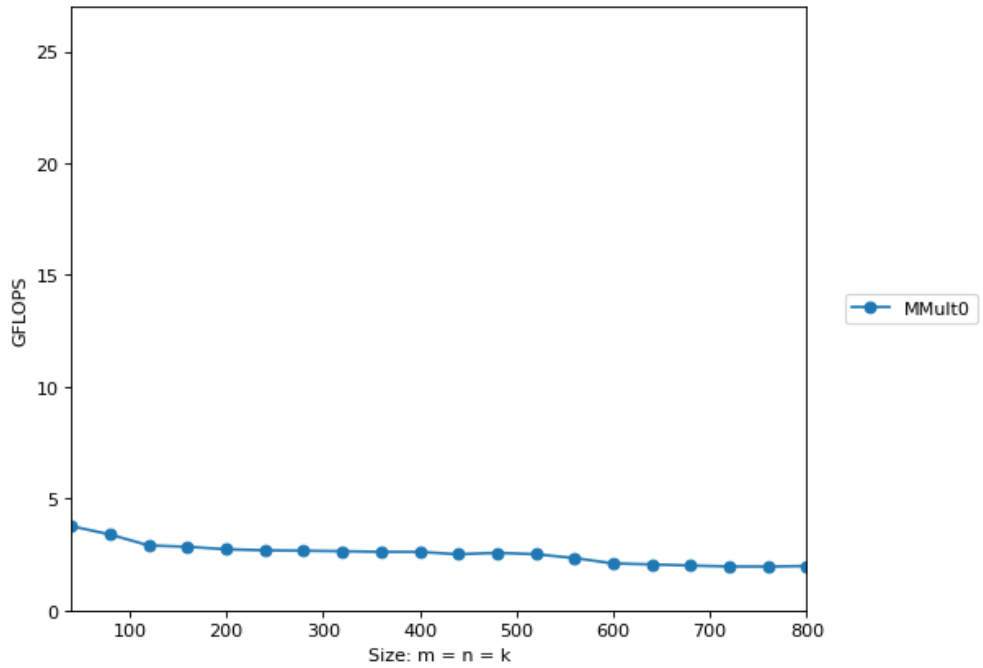
1. **MMult1** - Dodanie makra X oraz funkcji AddDot do programu
2. **MMult2** - Wykonywanie kroku co 4 dla zmiennej j
3. **MMult_Nx4_3** - Przeniesienie wywoływania funkcji AddDot do AddDotNx4
4. **MMult_Nx4_4 -** Rozwinięcie funkcji AddDot w miejscach wywoływania
5. **MMult_Nx4_5** - Jedna pętla do rozwinięć z poprzedniej optymalizacji
6. **MMult_Nx4_6** – Gromadzenie elementów C w rejestrach i rejestr dla A
7. **MMult_Nx4_7** – użycie wskaźników do elementów w B
8. **MMult_Nx4_8** -
   dla N=1: zmiana kroku pętli z optymalizacji 5. na 4
   dla N=4: przechowywanie w rejestrze wierszy kx4 macierzy B
9. **MMult_Nx4_9** -
   dla N=1: użycie niebezpośredniego adresowania, aby zmniejszyć liczbę wskaźników
   dla N=4: zmiana kolejności wykonywanych operacji, aby jednocześnie wykonywać obliczenia dla dwóch wierszy bloków C 4x4
10. **MMult_4x4_10** – Wykorzystanie wektorów __m128d
11. **MMult_4x4_11** - dodanie funkcji InnerKernel, podział na mniejsze problemy
12. **MMult_4x4_12** - dodanie funkcji PackMatrixA
13. **MMult_4x4_13** - Uproszczenie odwoływania do elementów z A
14. **MMult_4x4_14** - dodanie funkcji PackMatrixB, zmiana kroku funkcji PackMatrixA na 4
15. **MMult_4x4_15** - Warunkowe wykonanie dodanie funkcji PackMatrixB

b) Optymalizacja dostosowana do procesora

Program uruchamiałem z flagą dostosowaną do architektury mojego procesora, czyli – march=znver3

3 – Wyniki

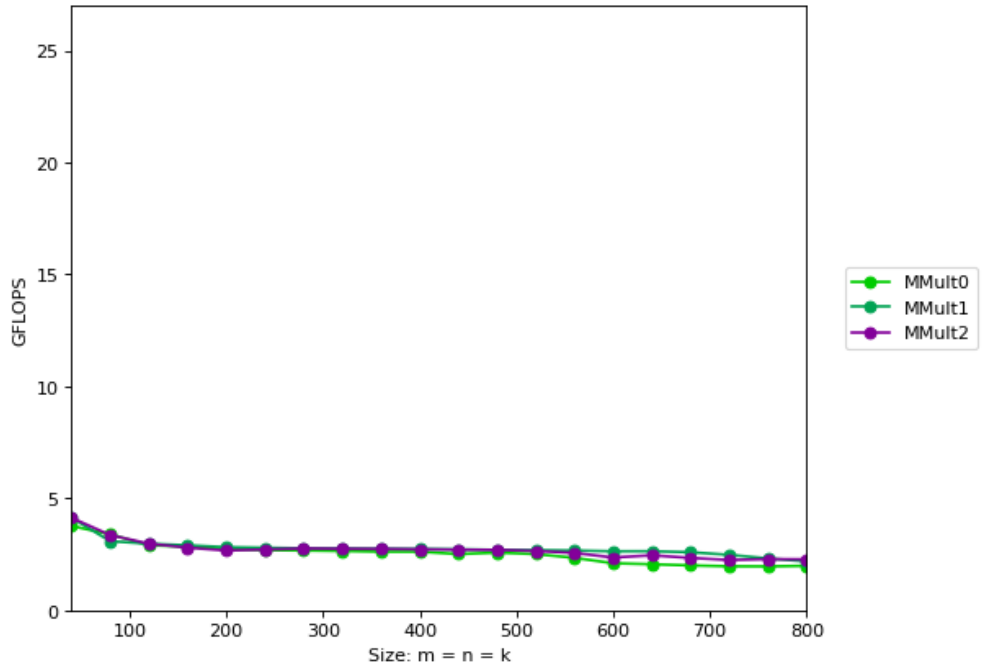
a) Bez optymalizacji



Wykres 1 – wartość początkowa

Widzimy, że obecne GFLOPS jest dalekie od oczekiwanego.

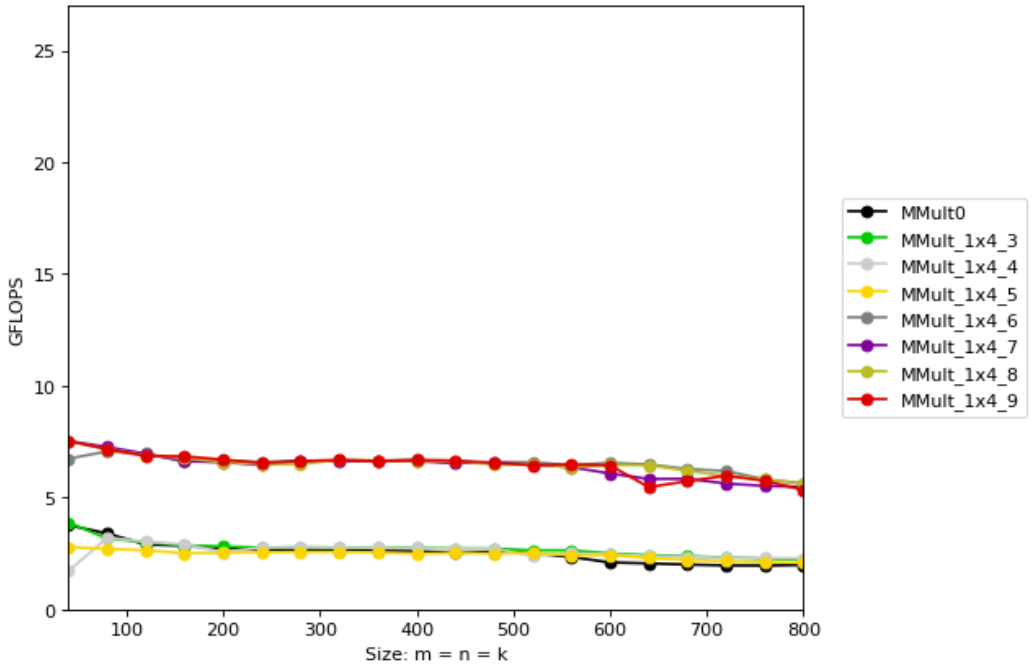
b) Przygotowanie do optymalizacji



Wykres 2 – przygotowanie do optymalizacji

Pierwsze optymalizacje nie wnoszą nic do wyniku i różni się on nieznacznie.

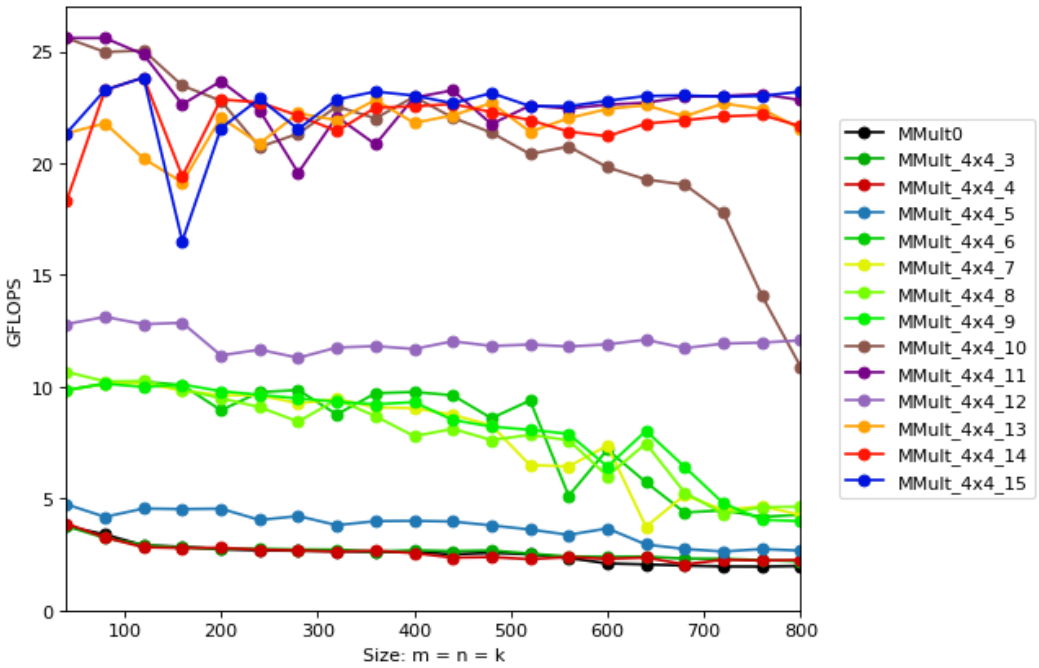
c) Optymalizacja 1x4



Wykres 3 – optymalizacje 1x4

Jak widzimy, wynik poprawiły się mniej więcej dwukrotnie w porównaniu z początkowym wskazaniem.

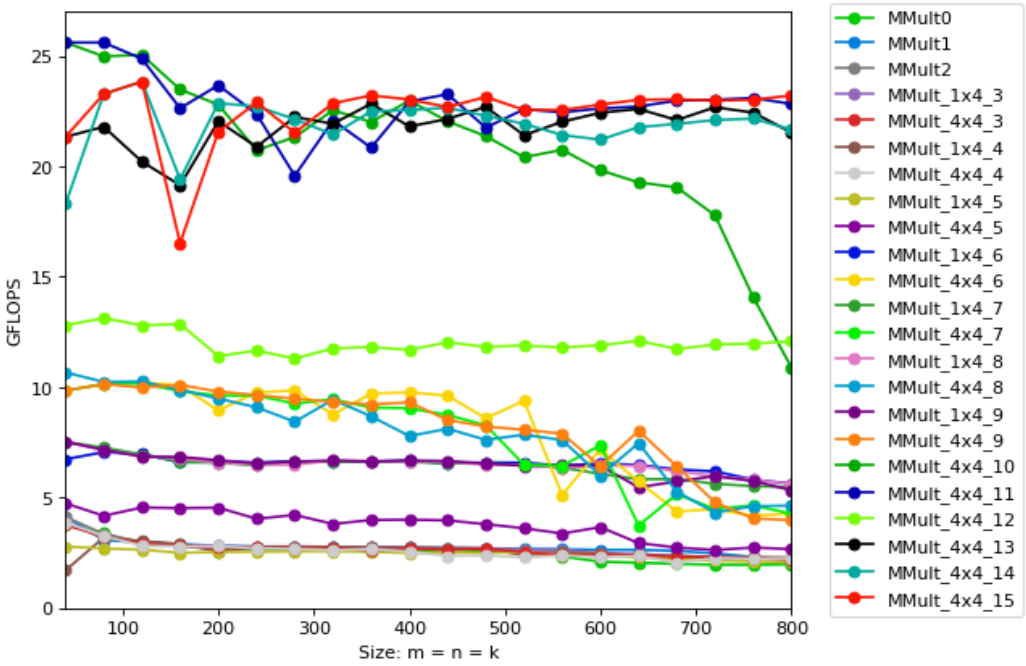
d) Optymalizacja 4x4



Wykres 4 – optymalizacje 4x4

Optymalizacje 4x4 znacznie poprawiają wyniki i są one kilka razy lepsze niż początkowe.

e) Końcowe wyniki



Wykres 5 – wszystkie wyniki optymalizacji

Wprowadzone przez nas optymalizacje, jak widać zdecydowanie poprawiły wyniki.

4 - Podsumowanie

a) Analiza wyników

1. Najlepszy uzyskany wynik to 25,6 GFLOPS, czyli około 55% maksymalnej teoretycznej wartości, czyli 46,4 GFLOPS. Wpłynąć mogły na to inne procesy działające w tym czasie w systemie, co nie pozwoliło wykorzystać całej mocy obliczeniowej.
2. Najbardziej wydajne wersje otrzymaliśmy po optymalizacji: MMult_4x4_13, MMult_4x4_14, MMult_4x4_15
3. Największe spadki wydajności otrzymaliśmy w przypadkach: MMult_4x4_10 dla dużych macierzy, MMult_4x4_15 dla macierzy n=160, MMult_4x4_14 dla n=160
4. Najbardziej przydatne optymalizacje to: MMult_1x4_6, MMult_4x4_10, MMult_4x4_11

b) Wnioski

1. Warto umieszczać dane w rejestrach, gdyż może to znacznie wpłynąć na wydajność programu
2. Stosowanie operacji wektorowych __128d poprawia wydajność programu
3. Czasami dodanie optymalizacji może wprowadzić kompilator w błąd jak w przypadku MMult_1x4_8
4. Są przypadki, kiedy kompilator sam wprowadza optymalizację np. w przypadku MMult_1x4_9