Data oddania: 19.04.2018r.

Sprawozdanie: Laboratorium nr 4, Badanie złożoności czasowej SIMD i SISD.

Prowadzący: mgr inż. Tomasz Serafin

Autor: Wojciech Wójcik

Indeks: 235621

Termin: Czwartek TP godz. 7:30

1. Opis zadania

Celem laboratoriów było zaimplementowanie operacji dodawania, odejmowania, mnożenia i dzielenia metodą SIMD i SISD dla 2048,4096 i 8192 liczb łącząc język C i assembler. Następnie należało zmierzyć czas wykonywania operacji i przeanalizować wyniki.

2. Opis wykonania

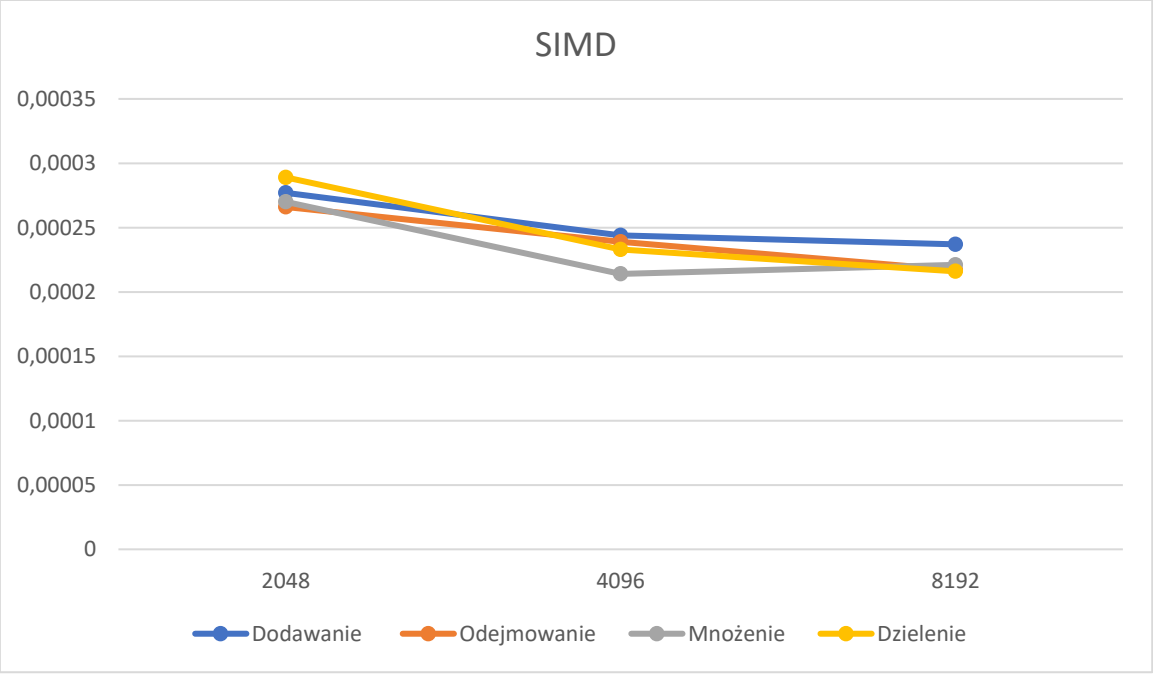
W celu przeprowadzenia obliczeń SIMD została stworzona struktura w języku c 4 liczb typu float:

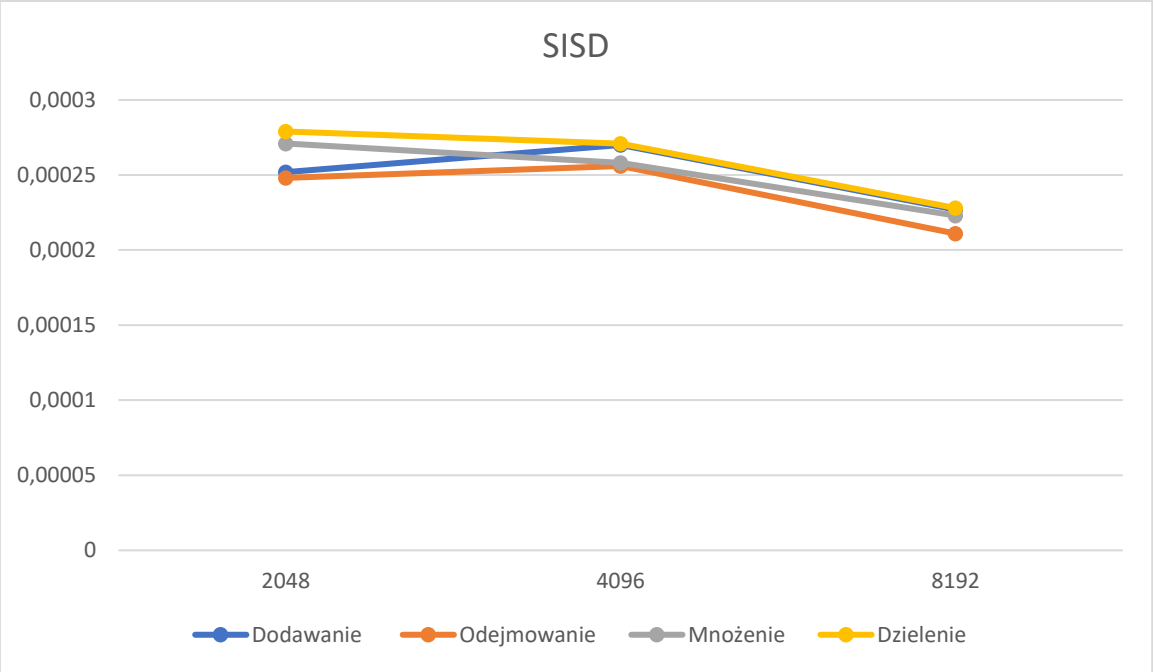
```
struct vec {
    float a;
    float b;
    float c;
    float d;
};
```

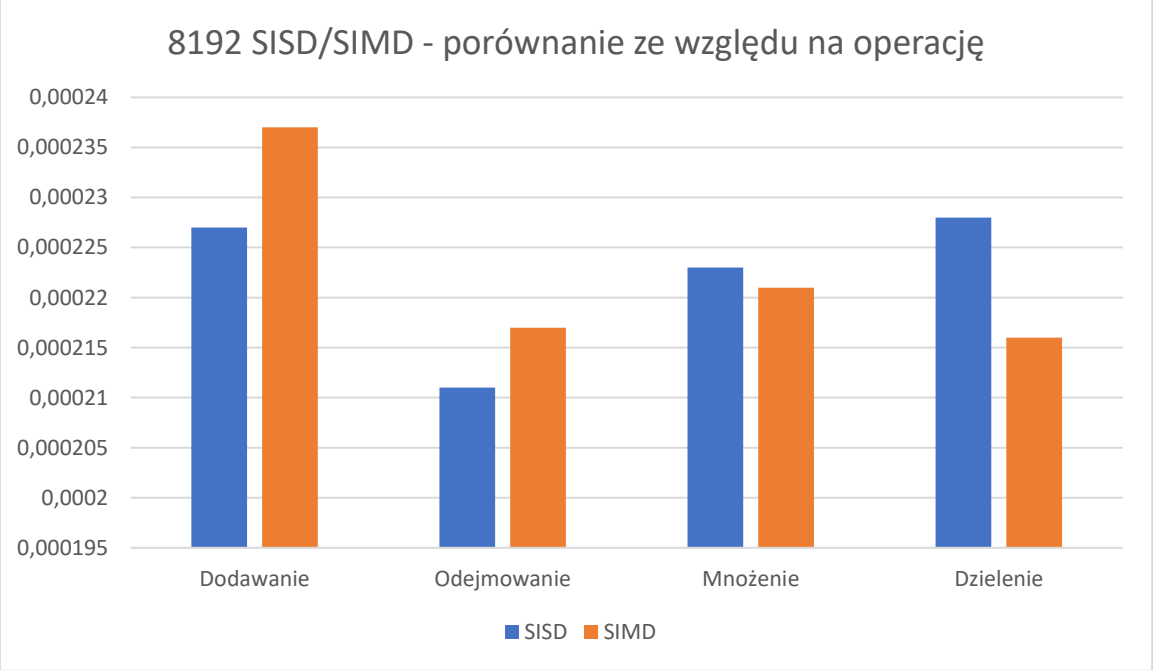
Do zliczania czasu została wykorzystania funkcja clock() z biblioteki time.h. Czas jest przechowywany w zmiennych typu double i wyświetlany w wynikach końcowych w milisekundach. Pomiary zostały wykonane na linux mint sylvia 64bit w trybie konsolowym, by zminimalizować wpływ innych aplikacji na wyniki(głównie gui). Każda funkcja SISD i SIMD zawiera 3 argumenty: wektor a i b, oraz wskaźnik na wektor wyniku działania. Do funkcjach SISD również zostały zastosowane wektory, by każde wywołanie zawierało tyle samo argumentów i żeby było ich tyle samo – w funkcjach SISD jest wykonywane działanie na 4 liczbach po kolei z użyciem FPU.

3. Wyniki

Wyniki pomiarów zostały zaprezentowane na wykresach:




SISD
0,0003
0,00025
0,0002
0,00015
0,0001
0,00005
0
2048
4096
8192
Dodawanie
Odejmowanie
Mnożenie
Dzielenie


8192 SISD/SIMD - porównanie ze względu na operację
0,00024
0,000235
0,00023
0,000225
0,00022
0,000215
0,00021
0,000205
0,0002
0,000195
Dodawanie
Odejmowanie
Mnożenie
Dzielenie
SISD
SIMD

4. Wnioski

Podczas testowania i tworzenia programu przekonałem się, jak dużo czasu zajmuje wywołanie funkcji, gdyż pierwotnie czas był mierzony przez wywołanie funkcji. Drugą rzeczą, którą zauważyłem podczas testowania jest dużo większy rozrzut wyników podczas testowania aplikacji na systemie z GUI, czy innymi aplikacjami w tle. Jest to bardziej zauważalne, gdyż sam czas wywołania funkcji jest strasznie mały, i każde przerwanie procesora by wykonać inną operacje jest jeszcze bardziej widoczne.
Testy zostały przeprowadzone również na maszynie wirtualnej jak i serwerze VPS. Oba przyniosły wyniki o 0.0001ms większe od tych uruchomionych bezpośrednio na linuxie w trybie konsolowym. Ciekawym spostrzeżeniem jest to, że serwer VPS osiągnął najgorsze czasy, mimo większej wydajności przypadającej na rdzeń. Jest to spowodowane prawdopodobnie większym obciążeniem maszyny fizycznej oraz prawdopodobnie większym obciążeniem przez innych jej użytkowników, niż podczas mierzenia na domowej wirtualnej maszynie, gdzie działa tylko jeden system-gość.

Otrzymane wyniki nie potwierdzają większej szybkości uniwersalnej(dla wielu działań) jakiegokolwiek sposobu. W przypadku podstawowych (prostszych) działań – dodawanie i odejmowanie widać przewagę metody SIMD, jednak wyniki dla dzielenia i mnożenia (bardziej skomplikowanych działań) szybsza okazuje się metoda SISD. Można zauważyć że różnice są strasznie małe i trzeba wziąć pod uwagę błąd systematyczny. Jednak bardziej skomplikowane operacje w dobie dzisiejszych szybkich procesorów wektorowo mogą być wolniejsze – po prostu szybsze jest wykonanie tych instrukcji jednostce FPU.

Ważną rzeczą jest również przewaga układu FPU, który potrafi dodać liczbę prosto z pamięci, bez ładowania jej do rejestru, co jest niemożliwe w przypadku SSE. Podczas jednej operacji wykorzystując FPU przenosimy do rejestrów 128 bitów mniej, co ma na pewno wpływ na czas wykonywania obliczeń. Ta cecha FPU pomaga nadrobić czas, który traci przez wykonywanie obliczeń po kolei.

Warto zauważyć, że na wynik miał wpływ błąd systematyczny. Można go zauważyć uruchamiając program z tymi samymi ustawieniami kilka razy – wyniki zawsze będą się różnić i nie mamy na to wpływu. Błąd systematyczny jest spowodowany brakiem bezpośredniego dostępu do CPU. W trakcie działania programu może on być przerywany przez system i inne programy oraz usługi. By zminimalizować ten błąd obliczenia zostały wykonane na systemie bez GUI oraz przez nieuruchamianie jakichkolwiek innych operacji w tym czasie (oprócz systemowych).

Ciekawą rzeczą jest średni czas operacji, który maleje razem ze wzrostem ilości przetworzonych liczb. Jest to wspólna cecha wszystkich wyników. Jest to spowodowane tym w jakim sposób CPU zachowuje się pod obciążeniem i podczas spoczynku. Gdy system pod którym działamy nie jest obciążony, to taktowanie procesora jest obniżane w celu na przykład zaoszczędzenia energii. Przy 2048 liczbach CPU może nie zdążyć podnieść swojego taktowania do maksimum, a podczas przetwarzania 8192 liczb osiąga znacznie wyższe taktowanie. Wyższe taktowanie powoduje szybsze wykonywanie obliczeń, co przekłada się na średnią.