Termin zajęć PONIEDZIAŁEK 18:55	**Wprowadzenie do wysokowydajnych komputerów**	
Osoby wykonujące ćwiczenie: Jan Napieralski,		
Tytuł ćwiczenia: Porównanie wydajności kodu C i wstawek assemblerowych		
Data wykonania ćwiczenia	06-05-2024	**Ocena:**
Data oddania sprawozdania	13-05-2024	

Wstęp teoretyczny

Zadaniem było przeprowadzenie eksperymentu mającego na celu porównanie i ocenę w zmianie wydajności kodu napisanego w języku C oraz języku C z użyciem wstawek assemblerowych. Testowane miałyby być podstawowe operacje na dwóch liczbach stało i zmiennoprzecinkowych

Przebieg eksperymentu

Całość eksperymentu została przeprowadzona na komputerze o podanej specyfikacji:

- procesor: Intel(R) Core(TM) i9-10900 CPU @ 2.80GHz 2.81 GHz

- pamięć RAM: 32,0 GB DDR4

- karta graficzna: NVIDIA GeForce RTX 2060 SUPER

- system operacyjny: Ubuntu 24.04 w architekturze x86

W celu uniknięcia wpływu niepotrzebnego obciążenia komputera podczas testów, wyłączone zostały wszystkie aplikacje i niepotrzebne procesy

W programie w języku C przygotowanym do eksperymentu, 100 razy generuję dwie losowe liczby stałoprzecinkowe reprezentowane jako typ int oraz 100 dwie losowe liczby zmiennoprzecinkowe typu float.

Dla tak wygenerowanych liczb, przeprowadzam analizę podstawowych operacji arytmetycznych: dodawania, odejmowania, mnożenia i dzielenia wykonując je po 1000000, 10000000, 100000000, 1000000000 i 10000000000 razy. Następnie z wyników, robię średnią arytmetyczną

Pomiary kodu bez oraz z wstawkami assemblerowymi są liczone z identycznym narzutem, tą samą pętlą for. Nie zostały też użyte żadne modyfikacje kodu, dla przykładu deklaracje zmiennych ze słowem kluczowym „register” oraz żadne flagi kompilatora mogące optymalizować kod. Program skompilowano za pomocą komendy „gcc -o main main.c”. Zweryfikowano również poleceniem „objdump” czy kompilator w procesie optymalizacji nie odrzuca lub modyfikuje kodu wykonywującego operacje arytmetyczne, co mogłoby się stać biorąc pod uwagę to, że nigdzie w programie nie jest używany wynik takich eksperymentowych operacji

Do pomiaru czasu została wykorzystana funkcja clock z time.h

Wyniki pomiarów

wydajność operacji na liczbach stałoprzecinkowych (int) w C				
liczba operacji	dodawanie [ms]	odejmowanie [ms]	mnożenie [ms]	dzielenie [ms]
1,000,000	1.02395	1.026	1.88622	1.65575
10,000,000	10.08294	10.07865	18.44688	16.23749
100,000,000	100.48183	100.50685	185.87413	162.18506
1,000,000,000	1004.18363	1004.36998	1861.83932	1621.31815

wydajność operacji na liczbach stałoprzecinkowych (int) w C z wstawkami assemblerowymi				
liczba operacji	dodawanie [ms]	odejmowanie [ms]	mnożenie [ms]	dzielenie [ms]
1,000,000	1.0648	1.02138	2.01594	1.64737
10,000,000	10.08659	10.07265	19.84199	16.22336
100,000,000	100.52984	100.45401	198.44784	162.34919
1,000,000,000	1004.80312	1004.0786	1988.70586	1621.46356

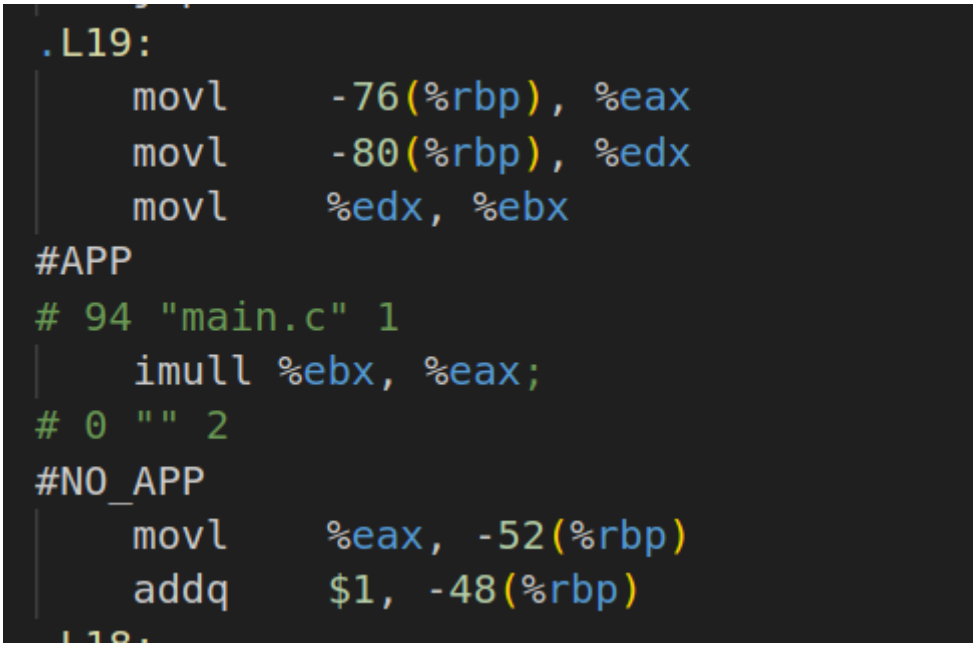
wydajność operacji na liczbach zmiennoprzecinkowych (float) w C				
liczba operacji	dodawanie [ms]	odejmowanie [ms]	mnożenie [ms]	dzielenie [ms]
1,000,000	1.0346	1.93181	1.05669	1.0547
10,000,000	10.30225	19.66734	10.18613	10.38804
100,000,000	99.65614	198.68275	101.43747	102.79036
1,000,000,000	997.80494	1986.15292	1013.73331	1026.24323

wydajność operacji na liczbach zmiennoprzecinkowych (float) w C z wstawkami assemblerowymi				
liczba operacji	dodawanie [ms]	odejmowanie [ms]	mnożenie [ms]	dzielenie [ms]
1,000,000	1.03452	1.9157	1.05259	1.03487
10,000,000	10.30329	19.50848	10.19216	10.26859
100,000,000	99.93012	198.36574	101.56858	101.75543
1,000,000,000	996.47166	1982.18017	1014.6594	1017.9576

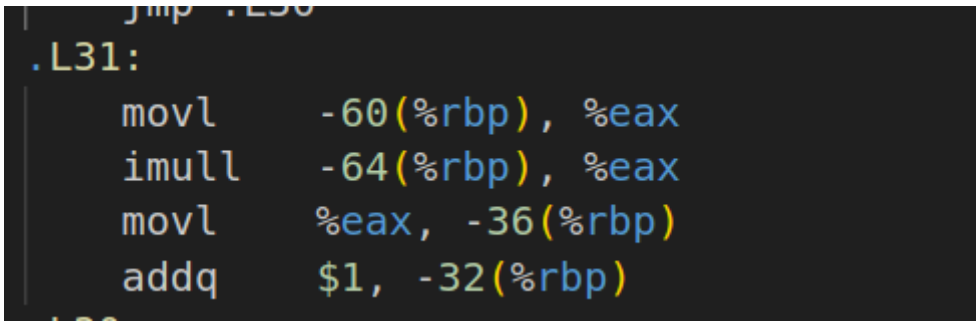
Wnioski

Dla wszystkich testowanych operacji arytmetycznych na liczbach stało i zmiennoprzecinkowych różnice wydajności pomiędzy kodem napisanym w C bez oraz z wstawkami assemblerowymi były bardzo małe, na granicy niepewności pomiarowej. Można na tej podstawie przyjąć, że wstawka nie wpłynęła na wydajność kodu.

Wyjątkiem od tej zaobserwowanej reguły była operacja mnożenia dla intów. Po zbadaniu wygenerowanego kodu assemblera z programu, zaobserwowałem drobną różnicę, która wpłynęła na niewielkie spowolnienie kodu ze wstawką



Kod mnożenia ze wstawką assemblera w C



Kod mnożenia bez wstawki w C

Operacja mnożenia dla kodu bez wstawki zostaje wykonana “od razu”, to znaczy ze stosu ładujemy do akumulatora wartość zmiennej a, by potem rozkazem imull domnożyć wartość b do akumulatora. W przypadku kodu ze wstawką assemblerową, wygenerowany kod najpierw ładuje do rejestru eax i ebx zmienne a i b, by dopiero potem pomnożyć je ze sobą. Tracimy tutaj niepotrzebnie chwilę na załadowanie wartości zmiennej b do rejestru, stąd różnice w czasie na korzyść zwykłego kodu w C

Jak widać wstawki assemblerowe nie są uniwersalnym sposobem zwiększenia wydajności kodu. Powinny być używane po dokładnym zbadaniu kodu assemblera generowanego przez język programowania i dopiero po wyczerpaniu możliwości optymalizacji kompilatora i kodu rozważane jako potencjalne usprawnienie. Wszystko sprowadza się do pytania “czy mój kod assemblera jest wydajniejszy od wygenerowanego”.

Wstawki mają też swoje ograniczenia, co pokazał przykład mnożenia, bo w przypadku niektórych operacji tracimy czas na załadowaniu wartości do rejestrów, co jest wymagane w języku C