WSTAWKI ASEMBLEROWE

Badanie wpływu wstawek na czas wykonania operacji arytmetycznych

MATEUSZ TESAREWICZ 272909
WDWK LABY, PN 18:55, GRUPA 8
13.05.2024

1 Wprowadzenie

W dzisiejszym sprawozdaniu przedstawiamy wyniki eksperymentu mającego na celu zbadanie wpływu wstawek asemblerowych na czas wykonania operacji arytmetycznych w programie napisanym w języku C. Celem eksperymentu było zrozumienie, czy użycie wstawek asemblerowych może przyspieszyć operacje arytmetyczne w programach napisanych w języku C. Przeprowadziliśmy serię testów, porównując czasy wykonania operacji z i bez użycia wstawek asemblerowych, aby ocenić różnicę w wydajności.

2 Plan przeprowadzenia badania

Poprawnie zaplanowanie przebiegu badania jest bardzo kluczowe, aby uniknąć ewentualnych komplikacji lub co gorsza, niepoprawnych wyników przeprowadzonego badania. Badaniu poddamy podstawowe operacje arytmetyczne, czyli dodawanie, odejmowanie, mnożenie i dzielenie liczb całkowitych oraz liczb zmiennoprzecinkowych.

2.1 Środowisko i specyfikacja sprzętu

Badania zostaną przeprowadzone na komputerze o podanej specyfikacji:
Procesor : AMD Ryzen 7 5700X 8-Core Processor 3.40 GHz
Pamięć RAM: 16 GB 3200 MHz
Typ systemu: 64-bitowy system operacyjny, procesor x64

2.2 Pomiar czasu

Czas zostanie zmierzony za pomocą biblioteki <time.h>. Użyjemy funkcji clock(), które zwracają ilość ticków procesora od momentu włączenia programu. Obliczmy różnicę przed i po wykonaniu fragmentu kodu i przekonwertujemy to na milisekundy poprzez podzielenie wyniku przez stałą TICKS_PER_SECOND * 1000.

2.3 Przebieg eksperymentu

Zaimplementowane programy posłużą nam do zbadania czasu wykonania operacji arytmetycznych dla liczb typu int i liczb typu float. Aby wynik był zauważalny, każdą operację wykonamy w pętli 10mln, 20mln i 50mln razy. Jako końcowy wynik eksperymentu weźmiemy średnią ze 100 testów dla każdej długości pętli w celu uniknięcia rozbieżności wyników. Dla każdego testu będą losowane nowe liczby przy użyciu funkcji rand().

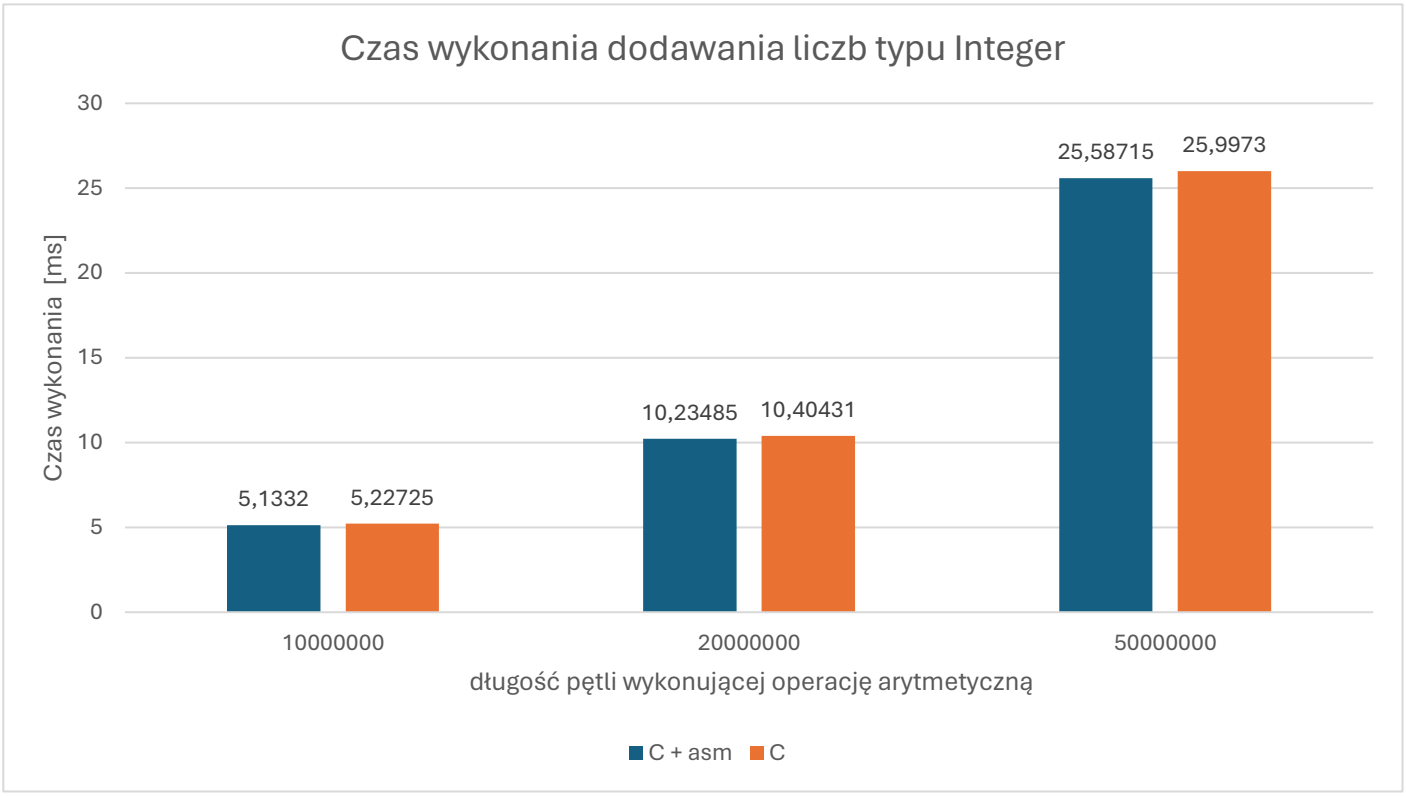
3 Wynik badań dla liczb Integer

3.1 Tabela z wynikami

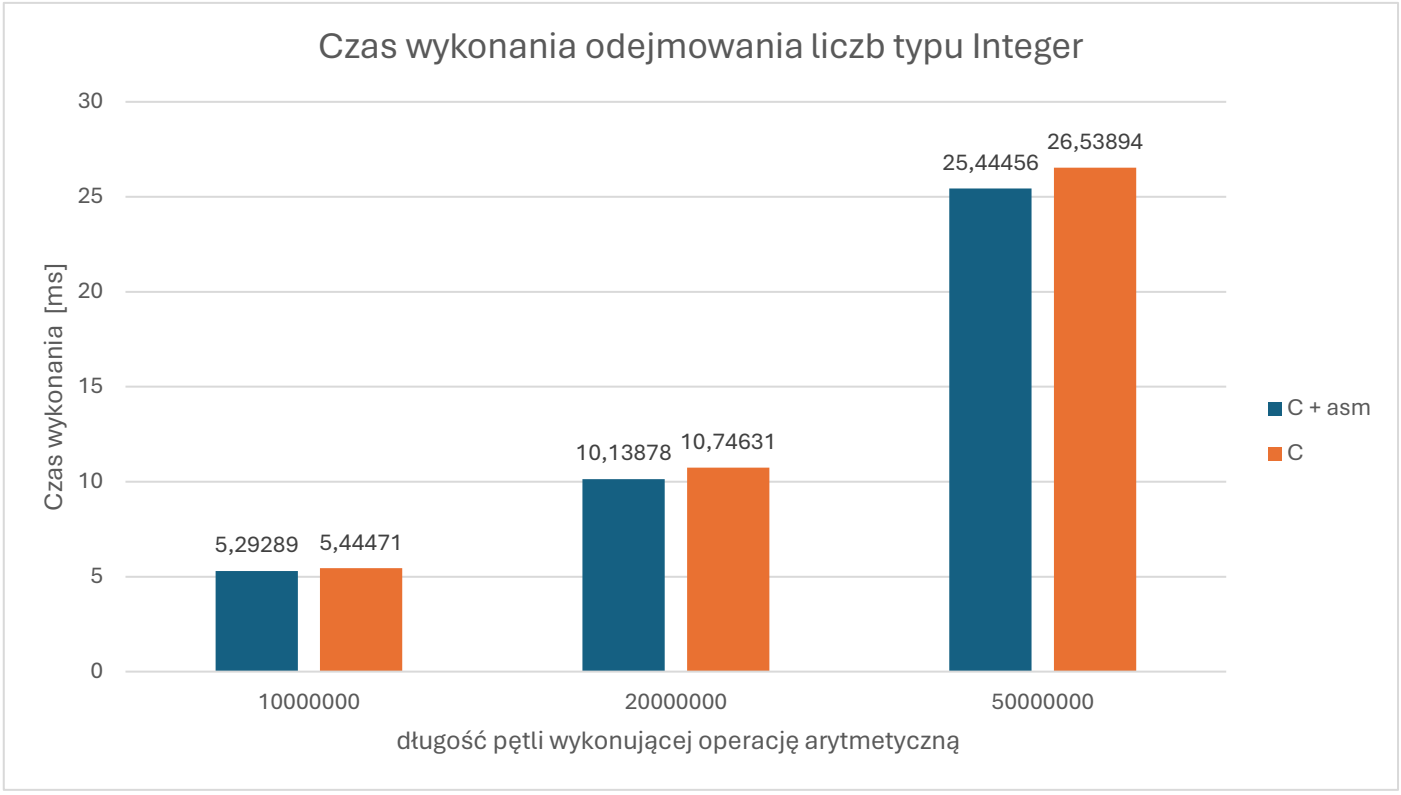
Tabela z wynikami wszystkich operacji arytmetycznych dla liczb typu Integer oraz ze wszystkimi długościami pętli wykonujących te operacje.

Integer				
Wykonanie operacji arytmetycznej 10 milionów razy				
operacja arytm → / program↓	Dodawanie [ms]	Odejmowanie [ms]	Mnożenie [ms]	Dzielenie [ms]
C + asm	5,1332	5,29289	5,26207	12,91432
C	5,22725	5,44471	5,17584	12,92302
Wykonanie operacji arytmetycznej 20 milionów razy				
operacja arytm → / program↓	Dodawanie [ms]	Odejmowanie [ms]	Mnożenie [ms]	Dzielenie [ms]
C + asm	10,23485	10,13878	10,41327	25,82567
C	10,40431	10,74631	10,33614	25,8275
Wykonanie operacji arytmetycznej 50 milionów razy				
operacja arytm → / program↓	Dodawanie [ms]	Odejmowanie [ms]	Mnożenie [ms]	Dzielenie [ms]
C + asm	25,58715	25,44456	26,44868	64,76732
C	25,9973	26,53894	25,87576	65,46518

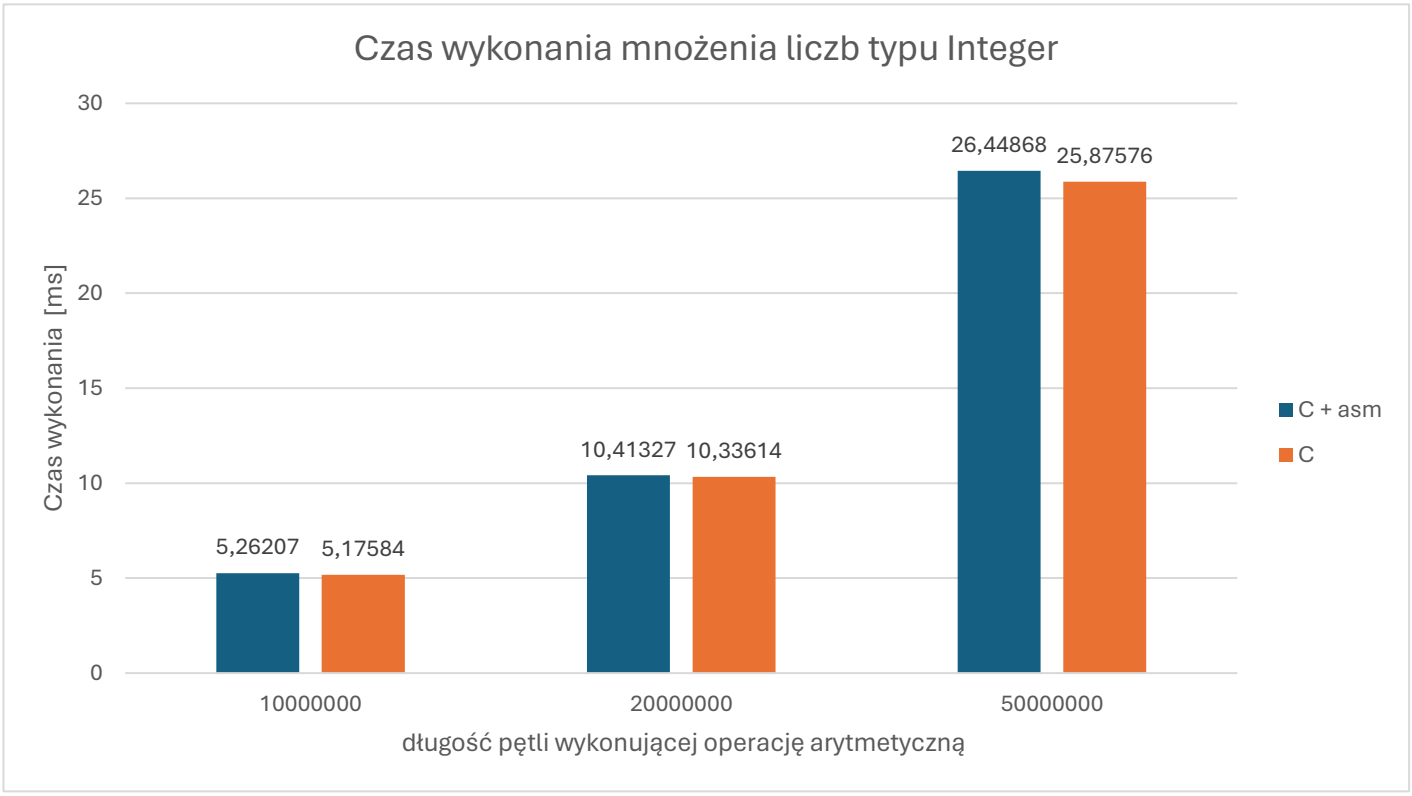
3.2 Wykres czasu dodawania liczb typu Integer



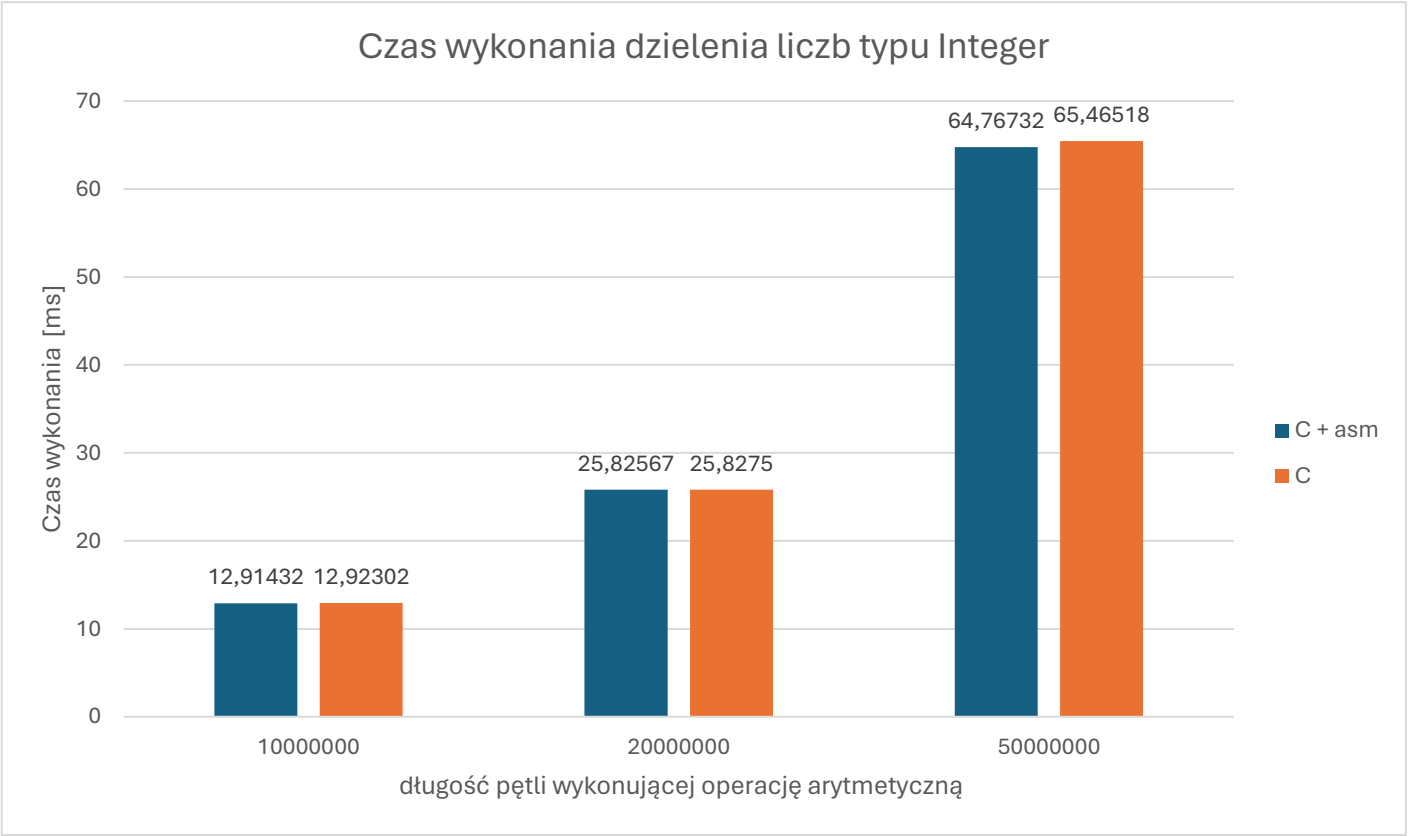
3.3 Wykres czasu odejmowania liczb typu Integer



3.4 Wykres czasu mnożenia liczb typu Integer



3.5 Wykres czasu dzielenia liczb typu Integer



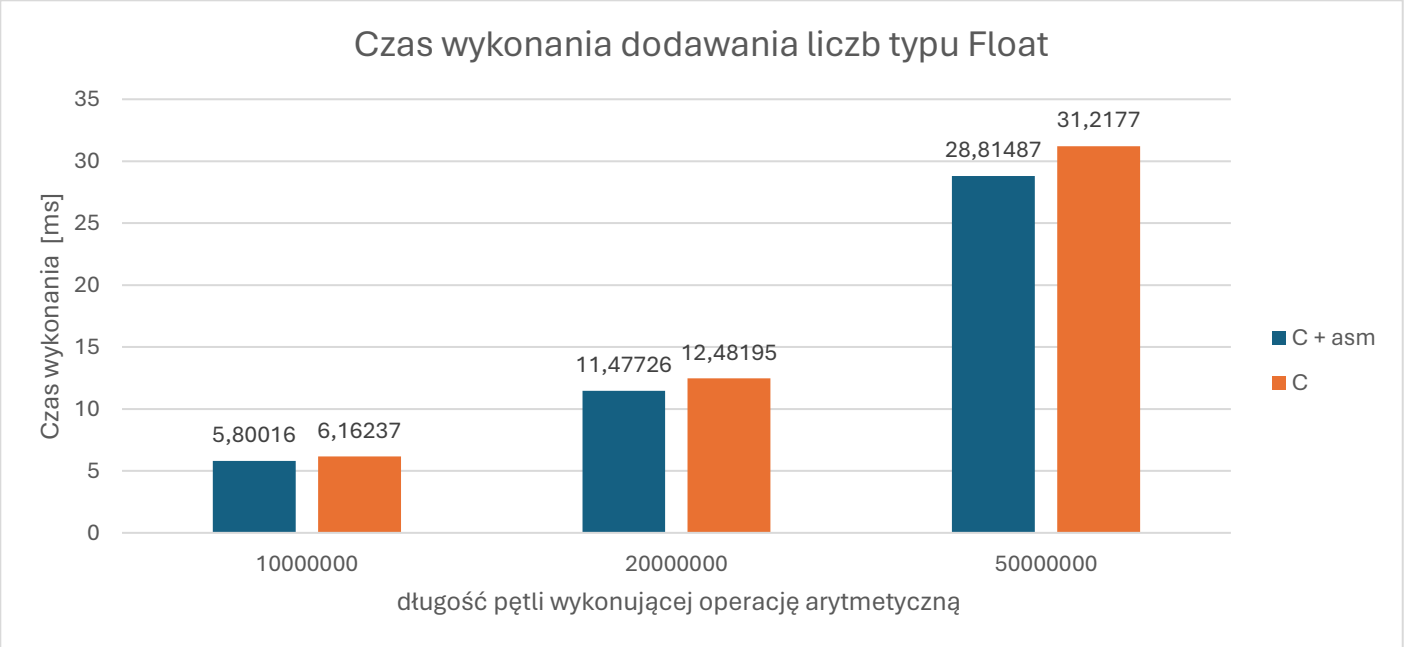
4 Wyniki badań dla liczb typu Float

4.1 Tabela z wynikami

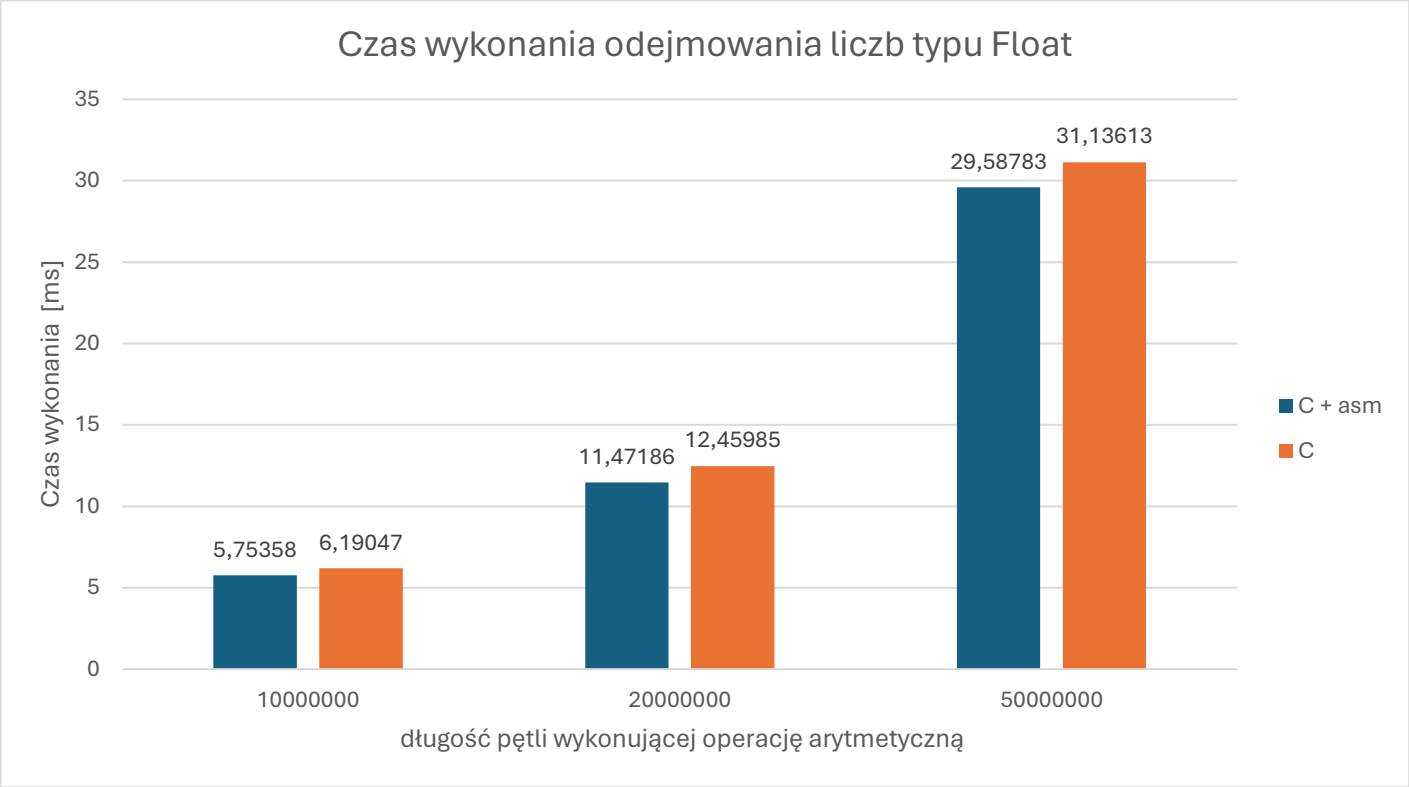
wszystkich operacji arytmetycznych dla liczb typu Float oraz ze wszystkimi długościami pętli wykonujących te operacje.

Float				
Wykonanie operacji arytmetycznej 10 milionów razy				
operacja arytm → / program↓	Dodawanie [ms]	Odejmowanie [ms]	Mnożenie [ms]	Dzielenie [ms]
C + asm	5,80016	5,75358	5,73821	12,92394
C	6,16237	6,19047	6,17349	12,95265
Wykonanie operacji arytmetycznej 20 milionów razy				
operacja arytm → / program↓	Dodawanie [ms]	Odejmowanie [ms]	Mnożenie [ms]	Dzielenie [ms]
C + asm	11,47726	11,47186	11,47243	25,82617
C	12,48195	12,45985	12,43206	25,90075
Wykonanie operacji arytmetycznej 50 milionów razy				
operacja arytm → / program↓	Dodawanie [ms]	Odejmowanie [ms]	Mnożenie [ms]	Dzielenie [ms]
C + asm	28,81487	29,58783	28,69318	64,57025
C	31,2177	31,13613	31,14169	64,96012

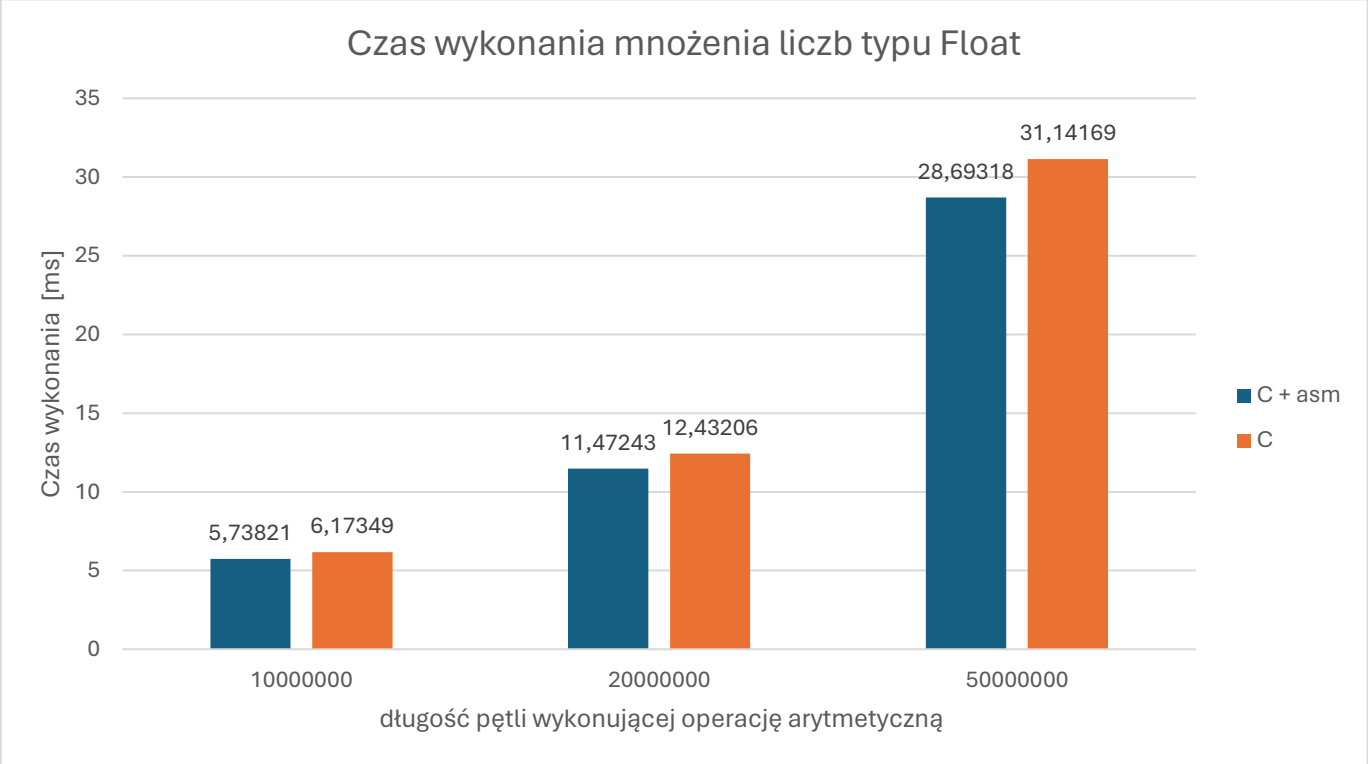
4.2 Wykres czasu dodawania liczb typu Float



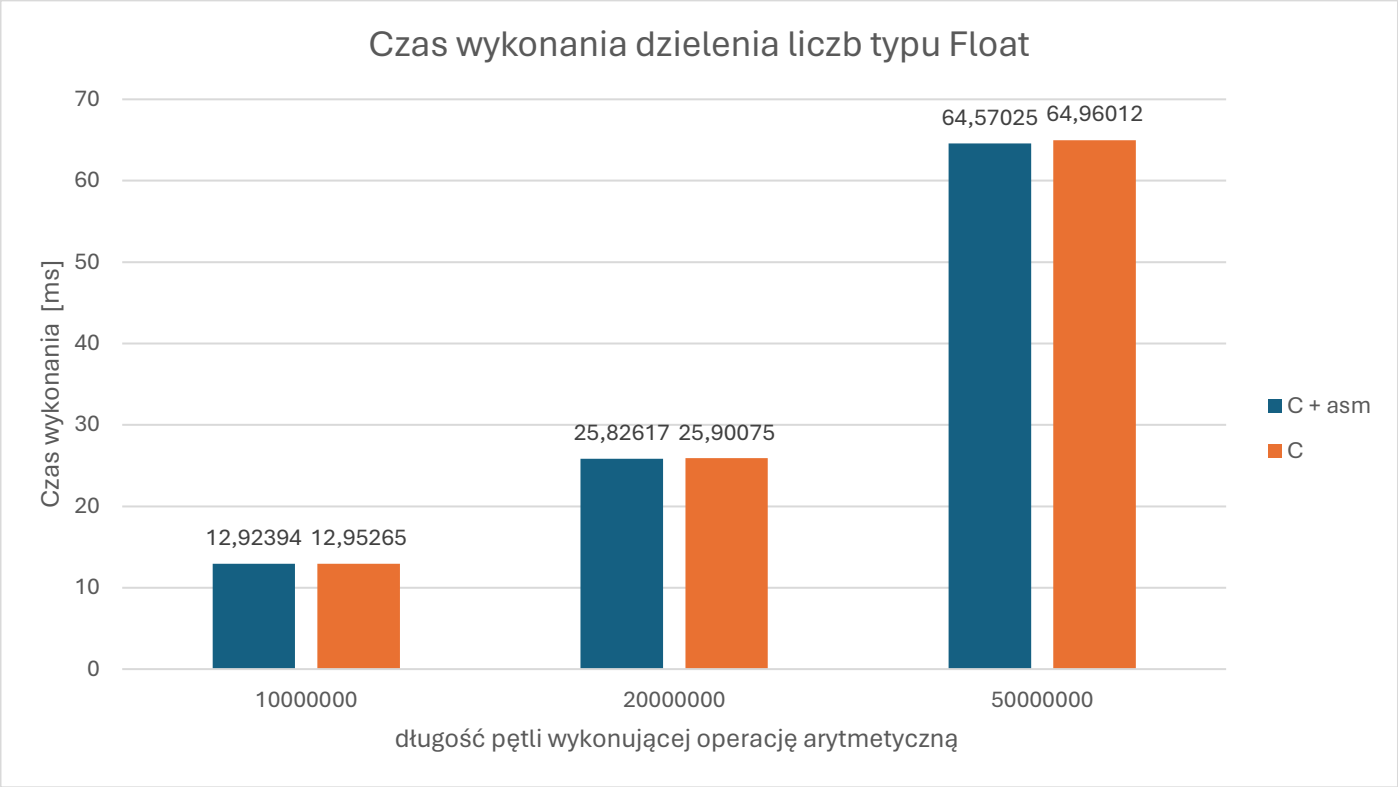
4.3 Wykres czasu odejmowania liczb typu Float



4.4 Wykres czasu mnożenia liczb typu Float



4.5 Wykres czasu dzielenia liczb typu Float



5 Wnioski

Podsumowując wyniki przeprowadzonych badań można stwierdzić, że wstawki asemblerowe minimalnie zmniejszają czas wykonywania operacji dodawania i odejmowania liczb całkowitych. Należy jednak uważać przy implementacji algorytmu mnożenia, ponieważ w badaniu wstawka asemblerowa wydłużyła ten czas. Dzielenie nie uległo zmianie.

Dla operacji arytmetycznych dla liczb zmiennoprzecinkowych pojedynczej precyzji wstawki asemblerowe polepszyły działanie wszystkich operacji arytmetycznych i jest to bardziej zauważalne niż w przypadku liczb całkowitych. Jedynie różnica dla operacji dzielenia jest prawie niezauważalna.