WSTAWKI ASEMBLEROWE

Badanie wpływu rejestru xmm na czas wykonania operacji arytmetycznych

MATEUSZ TESAREWICZ 272909
WDWK LABY, PN 18:55, GRUPA 8
22.05.2024

1 Wprowadzenie

W dzisiejszym sprawozdaniu przedstawiamy wyniki eksperymentu mającego na celu zbadanie wpływu użycia rejestru xmm we wstawce asemblerowej na czas wykonania operacji arytmetycznych w programie napisanym w języku C. Celem eksperymentu było zrozumienie, czy użycie rejestru xmm może przyspieszyć operacje arytmetyczne w programach napisanych w języku C. Przeprowadziliśmy serię testów, porównując czasy wykonania operacji z i bez użycia rejestu xmm, 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 32 bitowych. Rejestry xmm mogą wykonywać operacje arytmetyczne na 8 liczbach naraz, ponieważ pojemność jednego rejestru to 128 bitów, więc pomieści 4 liczby typu int.

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. Aby wynik był zauważalny, każdą serię operacji wykonamy w pętli 2,5mln razy. Użyjemy tylko jednej długości pętli, ponieważ czasy rosną liniowo wraz z długością pętli. Jako końcowy wynik eksperymentu weźmiemy średnią ze 100 testów w celu uniknięcia rozbieżności wyników. Dla każdego programu i testu użyjemy zestawu tych samych liczb. Seria operacji arytmetycznych oznacza 4 operacje pod rząd dla każdego przebiegu pętli. Daje to łącznie 10 milionów operacji arytmetycznych.

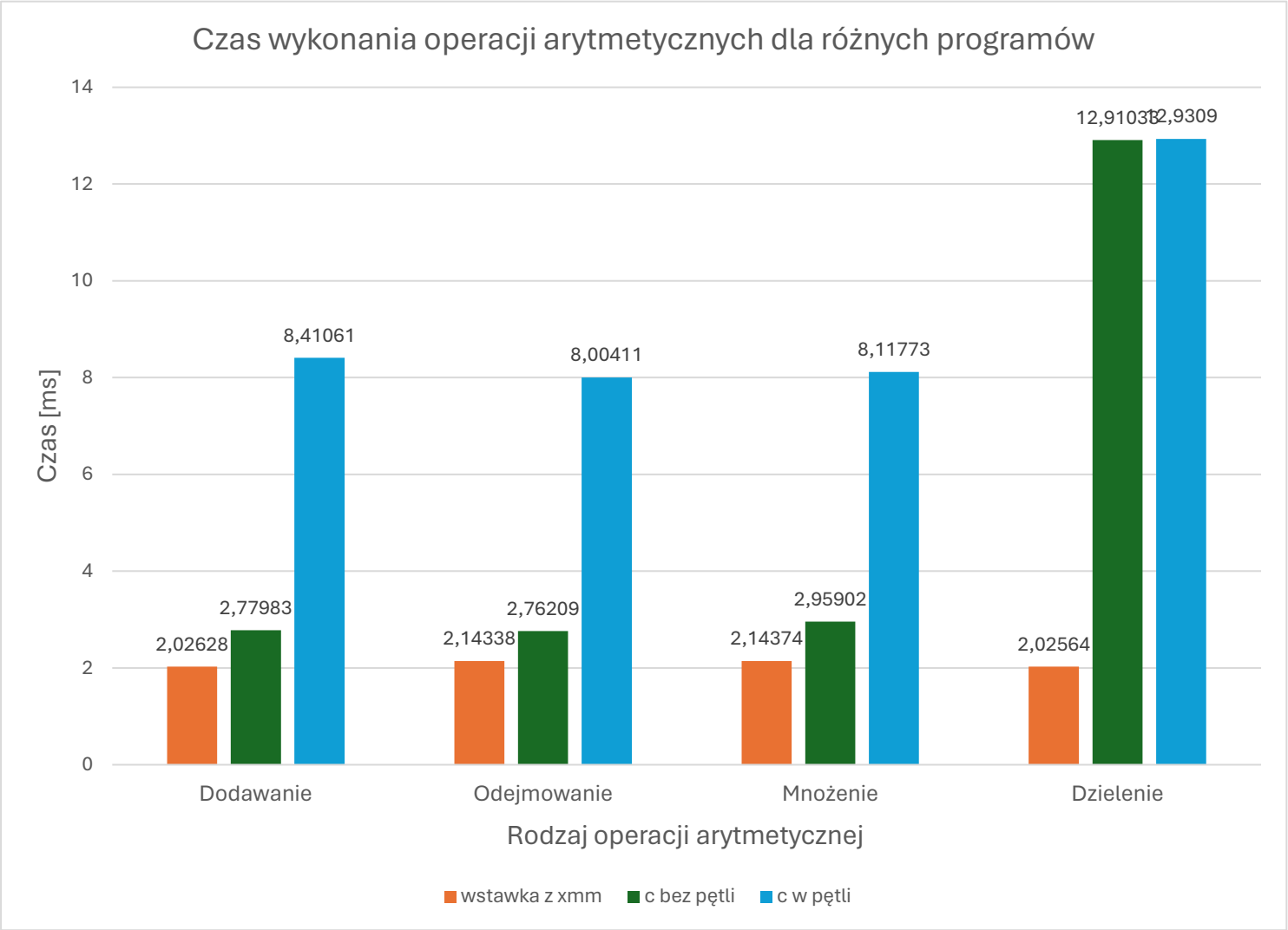
3 Wyniki badania

3.1 Tabela z wynikami

Tabela zawiera czasy wykonania 4 operacji arytmetycznych w pętli 2,5 mln razy dla 3 programów. C bez pętli oznacza, że w pętli 2,5 mln były wykonywane 4 operacje dla 4 par liczb. C z pętlą oznacza, że w pętli 2,5 mln była kolejna pętla, która wykonywała 4 razy 1 operację dla 1 pary liczb. W razie niezrozumienia proszę zobaczyć kod.

Integer 32bit				
Wykonanie operacji arytmetycznej 2,5mln * 4 razy				
Typ operacji → / Typ programu ↓	Dodawanie [ms]	Odejmowanie [ms]	Mnożenie [ms]	Dzielenie [ms]
wstawka z xmm	2,02628	2,14338	2,14374	2,02564
c bez pętli	2,77983	2,76209	2,95902	12,91033
c w pętli	8,41061	8,00411	8,11773	12,9309

3.2 Wykres z wynikami



4 Wnioski

Podsumowując wyniki przeprowadzonych badań można stwierdzić, że używanie rejestrów xmm do wykonania wielu operacji arytmetycznych zmniejsza czas wykonywania operacji dodawania, odejmowania, mnożenia i dzielenia liczb całkowitych. Dla pierwszych trzech czas udało się skrócić o około 25%, natomiast największy sukces osiągnęliśmy przy dzieleniu, gdzie czas został skrócony o 85%! Porównując 2 programy bez użycia rejestrów xmm możemy wywnioskować, że o wiele lepiej wykonywać operacje

arytmetyczne beż użycia pętli, pozwoli nam to zmniejszyć czas ponad 2 krotnie. Jedynym wyjątkiem jest dzielenie, dla którego użycie dodatkowej pętli nie wydłuża czasu.