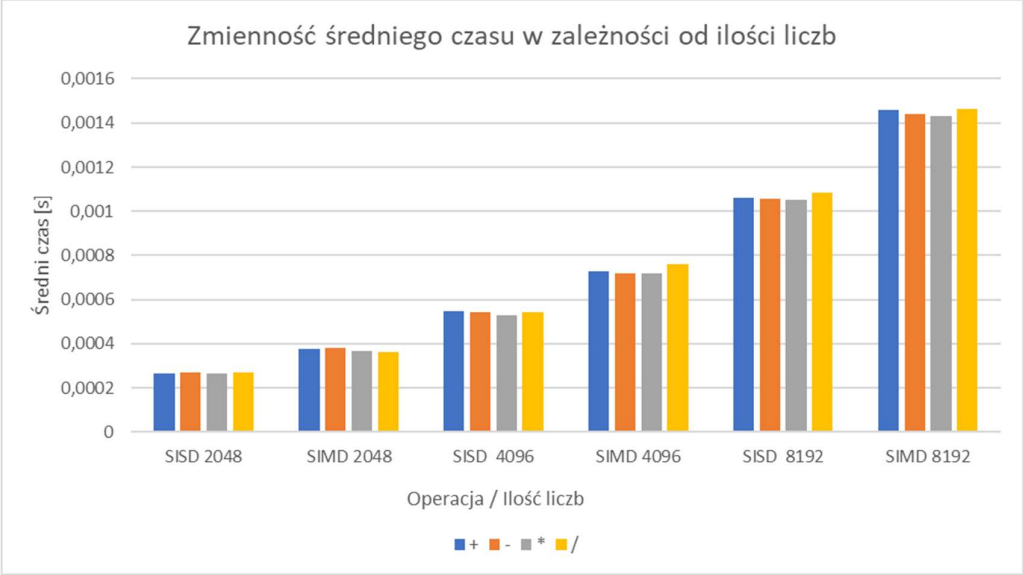
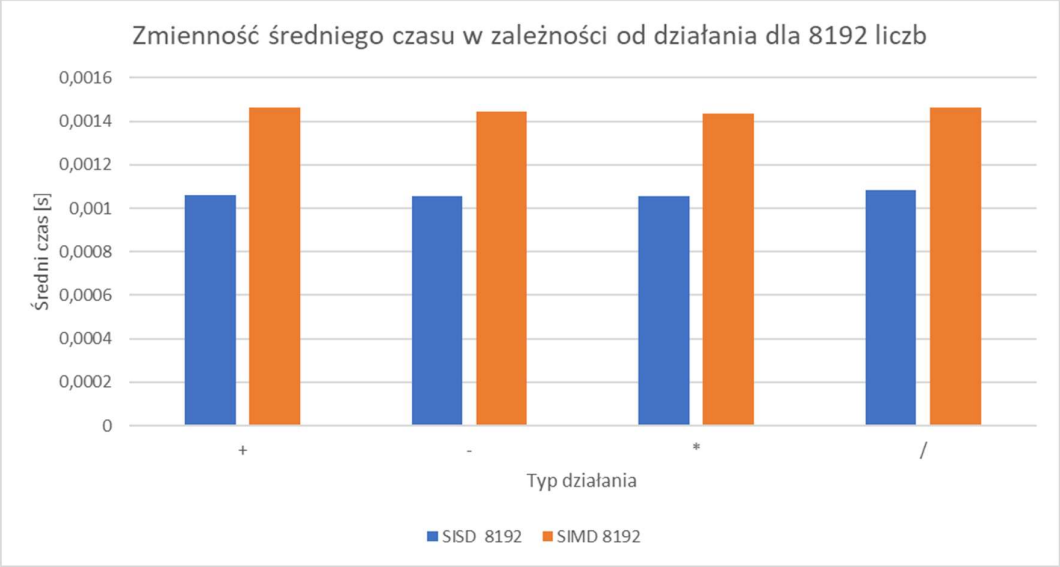
Maciej Kaźmierczyk

Lab 3 – SISD & SIMD

1. **Wykresy**
   a) Zmienność średniego czasu w zależności od ilości liczb



   b) Zmienność średniego czasu w zależności od typu działania dla 8192 liczb



2. **Przebieg pracy:**
   a) Na początku przygotowane zostały fragmenty kodu assemblera, które następnie wpisane zostały w funkcje kodu c++.
   b) Następnie przygotowane zostały funkcje pomocnicze oraz main, zajmująca się wywołaniem wszystkich operacji oraz zapisaniem wyników do plików.

3. **Napotkane problemy:**
    a) Jedynym napotkanym problemem, był brak możliwości skrócenia kodu assemblerowego, np. z wykorzystaniem funkcji podmieniającej string odpowiedzialny za rodzaj działania, w celu skrócenia powtarzalności kodu.

4. **Kluczowe fragmenty kodu:**
    a) **Kod assemblerowy do działań SISD** - działania typu SISD są wykonywane poprzez wstawianie pierwszego argumentu na stos (x), a następnie dodawanie do niego drugiego argumentu (y). Wynik jest pobierany ze stosu i umieszczany w pierwszym argumencie.

```
void add_SISD (float x, float y) {
    asm(
        "FLDS %1 \n"
        "FADDS %2  \n"
        "FSTPS %0 \n"
        : "=m"(x)
        : "m"(x), "m"(y));
}
```

    b) **Kod assemblerowy do działań SIMD** – działania typu SIMD polegają na umieszczaniu argumentów (wektorów) w rejestrach *xmm*. Następnie za pomocą specjalnych instrukcji wykonywane są wybrane operacje arytmetyczne na tych wektorach.

```
void add_SIMD(vector x, vector y) {
    asm(
        "movups %1, %%xmm0 \n"
        "movups %2, %%xmm1 \n"
        "addps %%xmm1, %%xmm0 \n"
        "movups %%xmm0, %0 \n"
        : "=g"(x)
        : "g"(x), "g"(y));
}
```

    c) Kod do pomiaru czasu – w celu pomiaru czasu wykorzystywane są dwie pętle: zewnętrzna odpowiada za wykonanie pomiaru 10 krotnie, natomiast wewnętrzna za wykonanie działania na *n* liczbach.

```
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < n; j++) {
        clock_t start = clock();
        result(randVector(), randVector());
        clock_t end = clock() - start;
        time += ((double)end) / CLOCKS_PER_SEC;
    }
}
```

5. **Uruchomienie programu:**
    a) W celu uruchomienia programu wystraczy skorzystać z polecenia *make*.
    b) W pliku makefile zmodyfikowana została tylko nazwa pliku i usunięte zostały flagi do debugowania.

6. **Zagadnienia teoretyczne:**
    Stworzona przez Michaela Flynna klasyfikacja architektur komputerowych opiera się na liczbie przetwarzanych strumieni instrukcji i danych. Zawierają się w niej dwie grupy wykorzystywane podczas laboratoriów:
    a) **SISD -** single instruction, single data – jedna instrukcja i jeden strumień danych
    b) **SIMD -** single instruction, multiple data – jedna instrukcja I wiele strumieni danych. Jest używane do przetwarzania obrazu, grafiki trójwymiarowej, wideo oraz obliczeń naukowych.

7. **Wnioski:**
    Patrząc na wykresy opisujące zużycie czasowe dla obu architektur, można zauważyć, iż mimo że operacje typu SIMD wykonywane są na 4-krotnie większych liczbach, to nie zajmują 4 razy więcej czasu. Często są nieznacznie wolniejsze, nawet przy dużych ilościach liczb, a typ obliczeń (+-*/) nie wpływa widocznie na opóźnienie. Można zatem wnioskować, że przesyłanie danych takim sposobem jest znacznie bardziej opłacalne. Dzieląc wartości na ilość wykorzystywanych bitów, można wyliczyć, że korzystanie z SIMD jest **wydajniejsze o około 34%** (dla danych liczb).