МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ІКНІ

Кафедра **ПЗ**

ЗВІТ

до лабораторної роботи №6

на тему: «Програмування арифметичного співпроцесора мікропроцесорів x86»
з дисципліни: «Архітектура комп'ютера»

Лектор : доцент каф. ПЗ

Крук О.Г

Виконала: ст.гр.ПЗ-23

Кохман О.В.

Прийняв: доцент каф. ПЗ

Крук О.Г

«____»_____2022 р.

Σ___ _ _ _ _ _ _ _ _.

Львів – 2022

Тема: програмування арифметичного співпроцесора мікропроцесорів x86.

Мета: розвинути навики складання програми для арифметичного співпроцесора мовою асемблера для обчислення математичного виразу, відтранслювати і виконати в режимі відлагодження програму, складену відповідно до свого варіанту, обчислити заданий вираз в програмі мовою С та порівняти результати.

Індивідуальне завдання

1. Складіть програму обчислення виразу за допомогою команд співпроцесора для WINDOWS.
2. Перевірте результат роботи асемблерної програми, порівнявши його з результатом програми мовою Сі.
3. У звіті наведіть текст програми, копії вікон з результатами.
4. Зробіть висновки про виконану роботу.

Варіант 11:

11	$\dfrac{\frac{5.5}{d} + tg(c * a) - \sqrt{53 * c + 6.4}}{7.8 - \frac{c}{4.4} + 17 * d}$	a = 3.3 c=8.1 d=6.2

Теоретичні відомості

Арифметичний процесор або співпроцесор - це цифровий пристрій, призначений для апаратного виконання арифметичних операцій над дійсними числами або числами з рухомою/плаваючою комою. Наявність співпроцесора дозволяє значно прискорити роботу програм, що виконують обчислення з високою точністю, тригонометричні розрахунки та опрацювання інформації, яка повинна бути подана у вигляді дійсних чисел. В перших моделях мікропроцесорів Intel співпроцесора не було, він виготовлявся у вигляді окремої інтегральної мікросхеми і входив в склад комп'ютерів як опція. Починаючи з моделі i486DX співпроцесор розміщується на тому ж кристалі, що і основний процесор.

Співпроцесор має вісім 80-розрядних регістрів даних R0 … R7 для зберігання чисел з плаваючою комою, організованих у вигляді кільцевого стека. Номер регістра, який на даний момент перебуває на вершині стека, вказується в 3-бітовому полі TOP, що міститься в слові стану співпроцесора. При написанні програм, в яких використовуються команди з плаваючою комою, до вершини стека можна звернутися за допомогою операнда ST(0) (або просто ST). В командах можна також використовувати відносні до вершини стека операнди ST(1) ... ST(7). Абсолютні імена регістрів типу R0, R1, ... R7 використовувати не можна.

При виконанні команд з плаваючою комою їх операнди зберігаються в десятибайтових регістрах у розширеному форматі з подвійною точністю. При збереженні результату арифметичної операції в пам'яті співпроцесор автоматично перетворює його з розширеного формату в ціле або довге ціле число, а також в коротке або довге дійсне число.

Основний процесор і співпроцесор можуть обмінюватися значеннями з плаваючою комою тільки через оперативну пам'ять. Тому перед викликом команди співпроцесора її операнд завжди повинен міститися в пам'яті. При цьому співпроцесор завантажує число з пам'яті в свій стек регістрів, виконує над ним арифметичну операцію і результат зберігає в оперативну пам'ять.

Мнемоніки команд з плаваючою комою завжди починаються з літери F/f, щоб їх можна було відрізнити від інших команд основного процесора. Друга літера в мнемоніці (зазвичай це B/b або I/i) визначає спосіб інтерпретації операнда, що міститься в пам'яті. Літера B свідчить про те, що оператор поданий в двійково-десятковому коді (Binary-Coded Decimal, або BCD). Літера I говорить про те, що оператор поданий у вигляді цілочислового значення. Якщо ці літери не вказані, то вважається, що оператор міститься в пам'яті в одному з форматів чисел із плаваючою комою. До прикладу, команда FBLD оперує з двійково-десятковими числами (BCD-числами), команда FILD - з цілими числами, а FLD - з дійсними, поданими в форматі з плаваючою комою.

У командах з плаваючою комою можна вказати максимум два операнди, причому один з них - це ім'я одного з регістрів даних. Безпосередньо задані операнди не використовуються. Як операнди не можна також використовувати імена регістрів загального призначення основного процесора, таких як AX або EBX. Не дозволені також операції типу "пам’ять-пам’ять".

Команда fabs обчислює абсолютне значення в регістрі стека ST(0), результат зберігається на місці аргументу.

Команда fsin / fcos обчислює sin / cos кута в радіанах (360° = 2л радіан), заданого в регістрі вершини стека ST; аргумент не повинен перевищувати 2^{63}. Результат зберігається в регістрі вершини стека ST, на місці аргументу.

Команда fptan обчислює наближене значення tg вмісту вершини стека ST(0). Значення аргументу в співпроцесорах 8087-80287 обмежене проміжком 0 - π/4; в наступних поколіннях співпроцесорів верхня межа розширена до 2^{63}. Якщо значення аргументу виходить за допустимі межі, то операція не виконується і встановлюється прапорець C2; це означає, що абсолютне значення аргументу слід спочатку зменшити на величину, кратну 2π, командою fprem1. Після записування результату в ST(0) (на місце аргументу) в стек додатково завантажується число 1.0. У підсумку отримуємо: ST = 1.0, а тангенс поміщається в ST(1), що дозволяло за

відсутності команд fsin та fcos прискорити обчислення функцій sin та cos через tg.
Команда fcom виконує порівняння даних з рухомою комою. Здійснюється порівняння дійсних чисел, одне з яких завжди перебуває на вершині стека, а інше - в зазначеному регістрі або в пам'яті. Якщо операнд в команді не заданий, порівнюються значення вмісту вершини стека ST та ST(1).

Код програми

Назва файлу: main.asm

```
.386
.model flat, stdcall
.stack
_data segment
    num1 real4 3.3 ; a
    num2 real4 8.1 ; c
    num3 real4 6.2 ; d
    num4 real4 5.5 ; 5.5
    num5 real4 53.0 ; 53
    num6 real4 6.4 ; 6.4
    num7 real4 7.8 ; 7.8
    num8 real4 4.4 ; 4.4
    num9 real4 17.0 ; 17
    numerator real4 ?
    denumerator real4 ?
    result real4 ?
_data ends
_text segment
start:
    finit
    fld num3
    fdiv num4

    fld num2
    fmul num1
    fptan

    faddp

    fld num5
    fmul num2
    fadd num6
    fsqrt

    fsubp
    fst numerator

    fld num7
    fld num2
    fdiv num8

    fsubp

    fld num9
    fmul num3

    faddp
    fst denumerator
```

```
    fld numerator
    fld denumerator

    fdivp
    fst result
    ret
_text ends
end start
```

Назва файлу: main.c

```c
#include <stdio.h>
#include <math.h>
int main() {
	double a = 3.3, c = 8.1, d = 6.2;
	double numerator, denumerator, result;
	numerator = (5.5 / d) + tan(c * a) - sqrt(53 * c + 6.4);
	denumerator = 7.8 - (c / 4.4) + 17 * d;
	result = numerator / denumerator;
	printf("result = %lf\n", result);
	return 0;
}
```

Протокол роботи

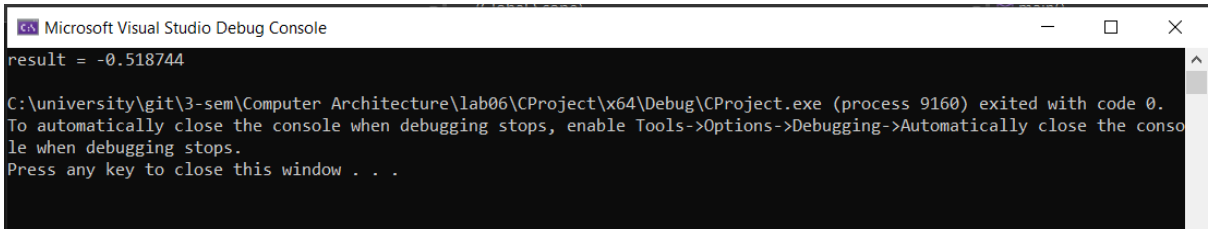

Рис. 1 Результат виконання програми на мові С.

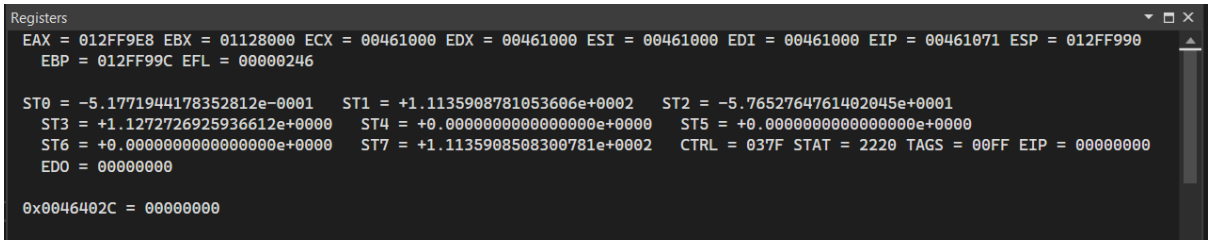

Рис. 2 Вікно регістрів.

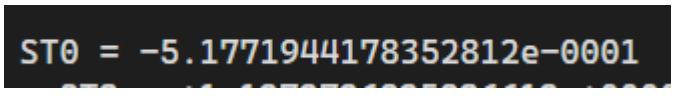

Рис. 3 Значення змінної result на мові асемблера.

Висновки

На цій лабораторній роботі я дізналась як складати програму для арифметичного співпроцесора мовою асемблера. Також я реалізувала

власну програму, яка обчислює математичний вираз згідно з варіантом і порівняла результат роботи програми на асемблері із результатом роботи попередньо написаної програми на С.