Міністерство освіти і науки України

Національний університет «Львівська політехніка»



**Лабораторна робота №4**

з дисципліни “Системне програмування”

на тему “Обчислення елементарних функцій на математичному співпроцесорі”

Варіант 19

Виконав: ст. гр. КІ-38

Черкасов В. С.

Прийняв:

Асистент Козак Н. Б.

Львів – 2020

**Мета:** познайомитись з принципами роботи математичного співпроцесора та оволодіти навиками використання вбудованих елементарних математичних функцій та реалізацій розгалужень

**Теоретичні відомості**

**Команди порівняння чисел**

У центральному процесорі команди умовних переходів виконуються відповідно до значень окремих бітів регістра прапорців процесора. У арифметичному співпроцесорі існують спеціальні команди порівнянь, за наслідками виконання яких, встановлюються біти кодів умов в регістрі стану:

**FCOM** – порівняння;

**FICOM** – порівняння цілих чисел;

**FCOMP** - порівняння дійсних чисел і вилучення зі стеку;

**FICOMP** - порівняння цілих чисел і вилучення зі стеку;

**FCOMPP** - порівняння і подвійне вилучення зі стеку (ST(0), ST(1));

**FTST** - порівняння операнда з нулем;

**FXAM** - аналіз операнда на тип числа (скінчене число, денормалізоване число, нуль, некінечність, ...).

Команда FCOM віднімає вміст операнда, розміщеного в оперативній пам’яті, від значення у вершині стеку ST(0). Результат віднімання нікуди не записується і покажчик вершини стеку ST не змінюється.

Позначимо операнд команди порівняння як “x”. Нижче, приведемо значення бітів кодів умови після виконання команди “FCOMx”:

C3 = 0, C0 = 0 ST(0)> x

C3 = 0, C0 = 1 ST(0)< x

C3 = 1, C0 = 0 ST(0)= x

C3 = 1, C0 = 1 ST(0) і x непорівнювані

Остання комбінація виникає при спробі порівняння не чисел, невизначеностей або нескінченості, а також в деяких інших випадках.

Операндами команди FICOM є 16- або 32-розрядні цілі числа, а в решті - аналогічна команді FCOM.

Команди FCOMP і FICOMP аналогічні, відповідно, командам FCOM і FICOM, за винятком того, що після виконання операнд вилучається зі стеку.

Команда FCOMPP виконує ті ж дії, що і FCOM, але вона після виконання вилучає зі стеку обидва операнди, що брали участь в порівнянні.

Команда FTST призначена для порівняння операнду з нулем. Після її виконання коди умов встановлюються згідно з наведеним нижче:

C3 = 0, C0 = 0 ST(0)> 0

C3 = 0, C0 = 1 ST(0)< 0

C3 = 1, C0 = 0 ST(0)= 0

C3 = 1, C0 = 1 ST(0) і 0 непорівнювані

## Трансцендентні команди

Трансцендентні команди призначені для обчислення наступних функцій:

* + тригонометричні (sin, cos, tg...)
  + зворотні тригонометричні (arcsin, arccos...)
  + показникові (xy , 2x , 10x , ex )
  + гіперболічні (sh, ch, th...)
  + зворотні гіперболічні (arsh, arch, arcth...)

Ось список всіх трансцендентних команд математичного співпроцесора:

**FPTAN** - обчислення часткового тангенса;

**FPATAN** - обчислення часткового арктангенса;

**FYL2X** - обчислення y\*log2(x);

**FYL2XP1** - обчислення y\*log2(x+1);

**F2XM1** обчислення 2x-1;

**FCOS** - обчислення cos(x);

**FSIN** - обчислення sin(x);

**FSINCOS** - обчислення sin(x) і cos(x) одночасно.

## Команди керування

Команди, керування, призначені для роботи з нечисловими регістрами співпроцесора. Деякі команди мають альтернативні варіанти. Мнемоніки цих команд можуть починатися з FN або F. Перший варіант відповідає командам “Без очікування”. Для таких команд процесор не перевіряє, чи зайнятий співпроцесор виконанням команди, тобто біт зайнятості B не перевіряється. Особливі випадки також ігноруються.

Варіанти команд “З очікуванням” діють так само, як і звичайні команди співпроцесора.

Ось список команд керування для співпроцесора:

**FNSTCW (FSTCW)** - записати управляюче слово (записує вміст регістра управління в оперативну пам’ять).

**FLDCW** - завантажити управляюче слово (завантажує регістр управління з оперативної пам’яті і, як правило, використовується для зміни режиму роботи співпроцесора).

**FNSTSW (FSTSW)** - записати слово стану (записує вміст регістра стану в оперативну пам’ять).

**FNSTSW AX (FSTSW AX)** - записати слово стану в AX (записує вміст регістра стану в регістр AX центрального процесора, де можливий аналіз вмісту за допомогою команд умовних переходів).

**FNCLEX (FCLEX)** - скинути особливі випадки (скидає прапорці особливих випадків в регістрі стану співпроцесора, також скидаються біти ES і B).

**FNINIT (FINIT)** - ініціалізувати співпроцесор (ініціалізує регістр стану, регістр управління, і регістр тегів таким чином:

*регістр управління* - проектна нескінченість, округлення до найближчого, розширена точність, всі особливі випадки замасковані;

*регістр стану* - B=0 (біт зайнятості скинутий), код умови не визначений, ST=ES=0, прапорці особливих випадків встановлені в нуль;

*регістр тегів* - усі поля регістру тегів містять значення 11 (порожній регістр));

**FNSTENV (FSTENV)** - записати оточення (записує в пам’ять вміст всіх регістрів, окрім числових, у визначеному форматі. Команда корисна при обробці особливих випадків);

**FLDENV** - завантажити оточення (завантажує регістри, збережені командою FNSTENV);

**FNSAVE (FSAVE)** - записати повний стан (діє аналогічно команді FNSTENV, але додатково зберігає вміст числових регістрів);

**FRSTOR** - відновити повний стан (діє аналогічно команді FLDENV, але додатково відновлює вміст числових регістрів);

**FINCSTP** - збільшити покажчик стека SP на 1;

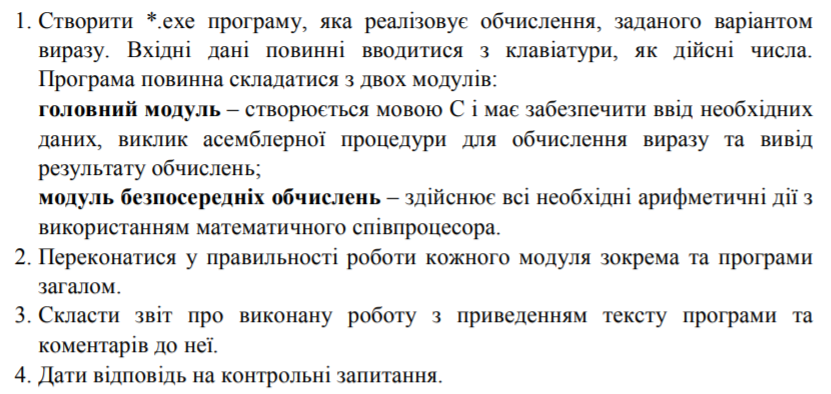
**FDECSTP** - зменшити покажчик стека SP на 1;

**FFREE** - звільнити регістр (визначає числовий регістр ST, вказаний як операнд, як порожній, записуючи у відповідне поле регістра тегів значення 11);

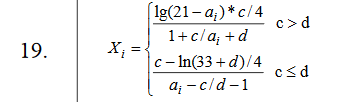
**FNOP** - порожня команда, немає операції (не робить жодних дій);

**FSETPM** - встановлює захищений режим роботи (переводить співпроцесор в захищений режим роботи).

**Завдання:**



**Варіант завдання:**

****

**source.cpp**

#include <stdio.h>

extern "C" double calc(double, float, float);

int main()

{

double arr[5] = { 11.0, 10.0, 2.0, 11.1, 2.0 };

float c = 0;

float d = 0;

double res = 0;

printf("Enter numbers: \n");

printf("C = ");

scanf\_s("%f", &c);

printf("D = ");

scanf\_s("%f", &d);

if (c <= d)

{

if (c == 0)

{

printf("C cannot be zero-value!\r\n");

return 0;

}

}

printf("\r\nCalculation result: \r\n");

for (int i = 0; i < 5; ++i)

{

res = calc(arr[i], c, d);

printf("X%d = %lf\r\n", i, res);

}

return 0;

}

**calc.asm**

.386

.model flat, c

.data

const\_1 dd 1.0

const\_4 dd 4.0

const\_21 dd 21.0

const\_33 dd 33.0

.code

calc proc

push ebp

mov ebp, esp

finit

fld dword ptr [ebp+16]

fcomp dword ptr [ebp+20]

fstsw ax

sahf

jbe next

; знаменник

fld dword ptr [ebp+16]

fdiv qword ptr [ebp+8]

fld const\_1

fadd st, st(1)

fadd dword ptr [ebp+20]

; чисельник

fldlg2

fld const\_21

fsub qword ptr [ebp+8]

fyl2x

fmul dword ptr [ebp+16]

fdiv const\_4

fdiv st, st(1)

jmp finish

next:

; знаменник

fld dword ptr [ebp+16]

fdiv dword ptr [ebp+20]

fld qword ptr [ebp+8]

fsub st, st(1)

fsub const\_1

; чисельник

fldln2

fld const\_33

fadd dword ptr [ebp+20]

fyl2x

fdiv const\_4

fld dword ptr [ebp+16]

fsub st, st(1)

fdiv st, st(2)

jmp finish

finish:

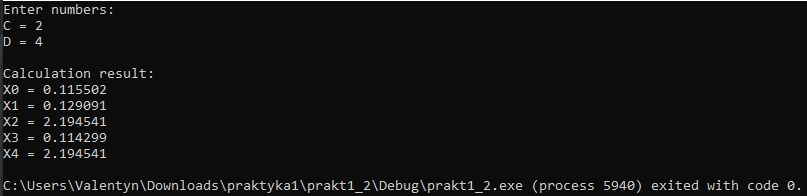
pop ebp

ret

calc endp

end

**Результат виконання:**



**Висновок:** на даній лабораторній роботі я ознайомився з принципами роботи математичного співпроцесора та навчився використовувати вбудовані математичні функції для обчислення виразів, а також навчився створювати розгалуження у програмі