**Міністерство освіти і науки України Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра обчислювальної техніки**

Лабораторна робота №6

з дисципліни «Системне програмування» на тему

“Обчислення арифметичних виразів і трансцендентних функцій. Використання команд співпроцесора ix87”

Виконав: Перевірив:

Студент ІІ курсу ФІОТ доц. Павлов В. Г.

групи ІМ-12

Сутулов Нікіта Олегович

номер у списку групи (варіант): 20

номер залікової книжки: 1229

Київ 2023

**Мета роботи:** Вивчення команд Асемблера для арифметики з плаваючою комою і здобуття навичок виконання розрахунків з елементами масивів.

Завдання за варіантом:

номер у списку групи: 20, тож 20 варіант: .

**Контрольні приклади:**

1. Чисельник і знаменник більші за нуль.

a = 4.1, b = 5.4, c = 3.3, d = 1.5.

.

Таким чином:

Результат =

2. Чисельник та знаменник менші за нуль.

a = -5.9, b = -7.4, c = 4.3, d = -0.2.

.

Таким чином:

Результат =

3. Знаменник дорівнює нулю.

a = 19.6, b = 20.6, c = -40.5, d = 19.2.

.

4. Чисельник більший за нуль, а знаменник менший за нуль.

a = 3.9, b = 4.3, c = -5.4, d = -2.5.

.

Таким чином:

Результат =

5. Чисельник менший за нуль, а знаменник більший за нуль.

a = 2.8, b = 7.9, c = -8.8, d = 0.1.

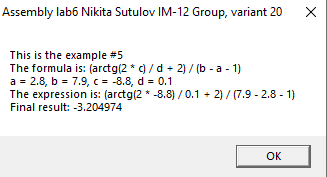
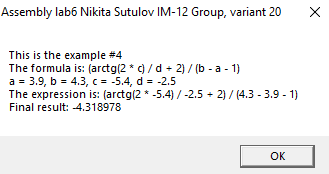
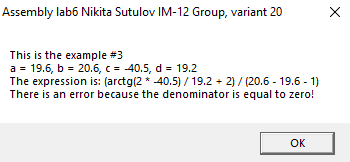
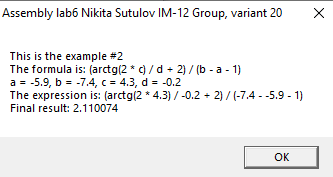
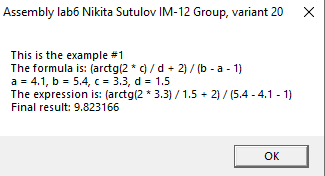
.

Таким чином:

Результат =

У формулі використовується функція arctg, яка визначена для всієї множини дійсних чисел, а тому додаткова перевірка відповідності аргумента функції її області визначення, а разом із нею й відповідний контрольний приклад, де цю область визначення порушено, НЕ ПОТРІБНІ.

**Демонстрація роботи програми**



В результаті порівняння контрольних розрахунків з результатами виконання програми видно, що дані збігаються, однак не повністю, а з деякою точністю, так як при контрольних розрахунках виконано округлення до 11 знаків після коми, а під час виконання програми комп’ютер здійснює округлення до 6 знаків після коми.

Отже програма працює коректно в усіх сценаріях:

* чисельник та знаменник більше за 0;
* чисельник та знаменник менше за 0;
* чисельник більше за 0, а знаменник – менше;
* чисельник менше за 0, а знаменник – більше;
* і також той випадок, коли знаменник дорівнює 0, опрацьовується коректно (виводиться повідомлення з помилкою). Тут варто також зауважити, що в цьому випадку не відбувається блокування розрахунків для наступних прикладів (якщо в них знаменник не дорівнює нулю), тож у них результат виконання програми коректний.

**Лістинг програми**

.386

.model flat, stdcall

option casemap :none

include \masm32\include\masm32rt.inc

.data

; arrays

SutulovAArray dq 4.1, -5.9, 19.6, 3.9, 2.8

SutulovBArray dq 5.4, -7.4, 20.6, 4.3, 7.9

SutulovCArray dq 3.3, 4.3, -40.5, -5.4, -8.8

SutulovDArray dq 1.5, -0.2, 19.2, -2.5, 0.1

; window caption

SutulovWindowCaption db "Assembly lab6 Nikita Sutulov IM-12 Group, variant 20", 0

; window text template

SutulovExampleForm db "This is the example #%d", 0

SutulovFormula db "The formula is: (arctg(2 \* c) / d + 2) / (b - a - 1)", 0

SutulovCurrentValuesForm db "a = %s, b = %s, c = %s, d = %s", 0

SutulovExpressionForm db "The expression is: (arctg(2 \* %s) / %s + 2) / (%s - %s - 1)", 0

SutulovFinalResultForm db "Final result: %s", 0

SutulovErrorMessage db "There is an error because the denominator is equal to zero!", 0

SutulovFinalForm db "%s", 10 ,13,

"%s", 10, 13,

"%s", 10, 13,

"%s", 10, 13,

"%s", 0

SutulovErrorForm db "%s", 10, 13,

"%s", 10, 13,

"%s", 10, 13,

"%s", 0

SutulovErrorMessageForm db "%s", 0

SutulovConstZero dq 0.0

SutulovConstOne dq 1.0

SutulovConstTwo dq 2.0

.data?

; buffers

SutulovExampleBuffer db 64 dup (?)

SutulovCurrentValuesBuffer db 64 dup (?)

SutulovExpressionBuffer db 128 dup (?)

SutulovFinalResultBuffer db 64 dup (?)

SutulovFinalBuffer db 256 dup (?)

SutulovDenominator dt ?

SutulovNumerator dt ?

SutulovArctanResult dt ?

SutulovArctanArgument dt ?

SutulovFinalResult dq ?

SutulovCurrentAString db 16 dup (?)

SutulovCurrentBString db 16 dup (?)

SutulovCurrentCString db 16 dup (?)

SutulovCurrentDString db 16 dup (?)

SutulovFinalResultString db 16 dup (?)

.code

main:

mov edi, 0 ; counter for indexes

mov esi, 1 ; counter for example numbers

loopToIterateThroughArrays:

cmp edi, 5

je exitLoop

finit

; calculating denominator: b - a - 1

fld SutulovBArray[edi \* 8]

fld SutulovAArray[edi \* 8]

fsub

fld SutulovConstOne

fsub

; checking denominator for zero:

fcom SutulovConstZero

fstsw ax

sahf

je denominatorZero

; saving denominator to 10-bytes buffer

fstp SutulovDenominator

; calculating arctan argument: 2 \* c

fld SutulovConstTwo

fld SutulovCArray[edi \* 8]

fmul

; saving arctan argument to 10-bytes buffer

fstp SutulovArctanArgument

; calculating arctg(2 \* c)

fld SutulovArctanArgument

fld SutulovConstOne

fpatan

; saving arctan calculation result to 10-bytes buffer

fstp SutulovArctanResult

; calculating numerator: (arctg(2 \* c) / d + 2)

fld SutulovArctanResult

fld SutulovDArray[edi \* 8]

fdiv ; d can't be 0 according to the task, so no zero checks needed here

fld SutulovConstTwo

fadd

; saving numerator to 10-bytes buffer

fstp SutulovNumerator

; dividing numerator by denominator

fld SutulovNumerator

fld SutulovDenominator

fdiv

; saving the final result to 8-bytes buffer

fstp SutulovFinalResult

normalWindow:

; converting float numbers to strings for them to be shown correctly

invoke FloatToStr, SutulovAArray[edi \* 8], offset SutulovCurrentAString

invoke FloatToStr, SutulovBArray[edi \* 8], offset SutulovCurrentBString

invoke FloatToStr, SutulovCArray[edi \* 8], offset SutulovCurrentCString

invoke FloatToStr, SutulovDArray[edi \* 8], offset SutulovCurrentDString

invoke FloatToStr, SutulovFinalResult, offset SutulovFinalResultString

; making the window text

invoke wsprintf, offset SutulovExampleBuffer,

offset SutulovExampleForm, esi

invoke wsprintf, offset SutulovCurrentValuesBuffer,

offset SutulovCurrentValuesForm,

offset SutulovCurrentAString, offset SutulovCurrentBString,

offset SutulovCurrentCString, offset SutulovCurrentDString

invoke wsprintf, offset SutulovExpressionBuffer,

offset SutulovExpressionForm,

offset SutulovCurrentCString, offset SutulovCurrentDString,

offset SutulovCurrentBString, offset SutulovCurrentAString

invoke wsprintf, offset SutulovFinalResultBuffer,

offset SutulovFinalResultForm, offset SutulovFinalResultString

invoke wsprintf, offset SutulovFinalBuffer, offset SutulovFinalForm,

offset SutulovExampleBuffer, offset SutulovFormula,

offset SutulovCurrentValuesBuffer, offset SutulovExpressionBuffer,

offset SutulovFinalResultBuffer

invoke MessageBox, 0, offset SutulovFinalBuffer, offset SutulovWindowCaption, 0

inc edi

inc esi

jmp loopToIterateThroughArrays

denominatorZero:

; converting float numbers to strings for them to be shown correctly

invoke FloatToStr, SutulovAArray[edi \* 8], offset SutulovCurrentAString

invoke FloatToStr, SutulovBArray[edi \* 8], offset SutulovCurrentBString

invoke FloatToStr, SutulovCArray[edi \* 8], offset SutulovCurrentCString

invoke FloatToStr, SutulovDArray[edi \* 8], offset SutulovCurrentDString

invoke wsprintf, offset SutulovExampleBuffer,

offset SutulovExampleForm, esi

invoke wsprintf, offset SutulovCurrentValuesBuffer,

offset SutulovCurrentValuesForm,

offset SutulovCurrentAString, offset SutulovCurrentBString,

offset SutulovCurrentCString, offset SutulovCurrentDString

invoke wsprintf, offset SutulovExpressionBuffer,

offset SutulovExpressionForm,

offset SutulovCurrentCString, offset SutulovCurrentDString,

offset SutulovCurrentBString, offset SutulovCurrentAString

invoke wsprintf, offset SutulovFinalResultBuffer,

offset SutulovErrorMessageForm, offset SutulovErrorMessage

invoke wsprintf, offset SutulovFinalBuffer, offset SutulovErrorForm,

offset SutulovExampleBuffer,

offset SutulovCurrentValuesBuffer,

offset SutulovExpressionBuffer,

offset SutulovFinalResultBuffer

invoke MessageBox, 0, offset SutulovFinalBuffer, offset SutulovWindowCaption, 0

inc edi

inc esi

jmp loopToIterateThroughArrays

exitLoop:

invoke ExitProcess, 0

end main

**Висновки**

У результаті порівняння контрольних розрахунків із результатами виконання програми для всіх передбачених лабораторною роботою сценаріїв (перевірка аргумента функції на відповідність її області визначення, а також відповідний приклад, у якому цю область визначення порушено, реалізовано не було, оскільки в моєму варіанті використовується функція arctg, яка визначена для всієї множини дійсних чисел, а тому такої перевірки не потребує):

* чисельник та знаменник більше за 0;
* чисельник та знаменник менше за 0;
* чисельник більше за 0, а знаменник – менше;
* чисельник менше за 0, а знаменник – більше;
* знаменник дорівнює 0,

бачимо, що результати обчислень співпадають з деякою точністю (похибка виникає через те, що в ході виконання програми округлення здійснювалось із меншою точністю, ніж у контрольних розрахунках), і випадок, коли знаменник дорівнює 0, обробляється коректно (і в цьому випадку не блокується виконання розрахунків для наступних прикладів, якщо вони відповідають умові, що знаменник не дорівнює нулю). Таким чином, роблю висновок, що програму написано коректно.

У ході виконання цієї лабораторної роботи я познайомився та вивчив команди для арифметики з плаваючою комою в мові асемблера MASM32 (додавання fadd, віднімання fsub, множення fmul, ділення fdiv, порівняння fcom, обчислення арктангенса fpatan тощо), а також здобув навички здійснення розрахунків з елементами одновимірних масивів.