**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 групи ІМ-22 доц. Павлов В. Г.

Тимофеєв Даниіл Костянтинович

номер в списку групи: 23

Київ 2024

**Мета роботи:**

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

**Порядок виконання роботи:**

1. Вивчити арифметичні команди з плаваючою комою [1].
2. Розробити програму на мові Асемблер, в якій згідно з індивідуальним варіантом завдання (табл. 1) виконуються обчислення значення арифметичного вираження із застосуванням команд співпроцесора ix87 з подальшим виведенням результату\* у віконному інтерфейсі.
3. Для всіх варіантів передбачити завдання значень вхідних змінних у форматі **double (DQ)**, проміжних результатів обчислень – у форматі **long double (DT)**, а кінцевих - знову у форматі **double**.
4. Розрахунки (п. 2, 3) повторити в програмі для 5 значень змінних\*\*, причому всі вхідні значення задати дійсними числами у вигляді одновимірних масивів.
5. Для перевірки правильності виконання розрахунків і результатів, що виводяться, заздалегідь виконати контрольні розрахунки. Проміжні і остаточні результати контрольних розрахунків з точністю не гірше точності розрахунків у комп'ютеру привести в звіті.
6. Виконати відладку програми шляхом порівняння розрахованих програмою результатів з контрольними прикладами. Лістинг розробленої програми і скріншоти розрахунків по всіх контрольних прикладах привести в звіті по лабораторній роботі.
7. Зробити висновки по лабораторній роботі.

**Пункт 5.** Для перевірки правильності виконання розрахунків і результатів, що виводяться, заздалегідь виконати контрольні розрахунки. Проміжні і остаточні результати контрольних розрахунків з точністю не гірше точності розрахунків у комп'ютеру привести в звіті.

**Контрольні розрахунки**

Номер у списку групи = 23, тож 23 - 21 = 2 варіант для цієї лабораторної роботи. Формула (-2\*c - d\*82)/tg(a/4 - b)**.** Розгляньмо декілька випадків:

* чисельник і знаменник більші за нуль;
* чисельник більше за нуль, але знаменник менший за нуль;
* чисельник менше за нуль, а знаменник більший за нуль;
* чисельник і знаменник менші за нуль;
* знаменник дорівнює нулю;
* порушення області визначення (аргумент = Pi/2 +Pi\*k, k = 0 та k = 1)

1. Чисельник і знаменник більші за нуль:

числа: a = 20.4, b =0.5, c = -5.9, d = -1.7

формула: (-2\*(-5.9) - (-1.7) \* 82)/tg(20.4/4 - 0.5) = (11.8 + 1.7 \* 82)/tg(5.1 - 0.5) = (11.8 + 139.4)/tg(4.6) = 151.2/tg(4.6) = 151.2/8.860174895648074 = 17.065125889813547

1. Чисельник більше за нуль, але знаменник менший за нуль

числа: a = 22.8, b = 3.2, c = -7.8, d = -4.5

формула: (-2 \* (-7.8) - (-4.5) \* 82)/ tg(22.8/4 - 3.2) = (15.6 + 4.5\*82)/tg(5.7 - 3.2) = (15.6 + 369)/tg(2.5) = 384.6/tg(2.5) = 384.6/(-0.747022297238660) = -514.844070145776641

1. Чисельник менше за нуль, а знаменник більший за нуль

числа: a = 44.4, b = 7.2, c = -3.4, d = 5.7

формула: (-2 \* (-3.4) - 5.7\*82)/tg(44.4/4 - 7.2) = (6.8 - 467.4)/tg(11.1 - 7.2) = (-460.6)/tg(3.9) = (-460.6)/0.947424649935893 = -486.160033973325824

1. Чисельник і знаменник менші за нуль

числа: a = 13.8, b = -5.2, c = -7.4, d = 10.7

формула: (-2 \* (-7.4) - 10.7 \* 82)/tg(13.8/4 - (-5.2)) = (14.8 - 877.4)/tg(3.45 + 5.2) = (-862.6)/tg(8.65) = (-862.6)/(-0.978982019750091) = 881.119349076706840

1. Знаменник дорівнює нулю

числа: a = 198.8, b = 49.7, c = 48.4, d = 0.3

формула: (-2 \* 48.4 - 0.3 \* 82)/tg(198.8/4 - 49.7) = (-96.8 - 24.6)/tg(49.7 - 49.7 ) = (-121.4)/tg(0) = (-121.4)/0. На 0 ділити не можемо => у програмі кидаємо помилку

Розрахунки з порушенням області визначення тангенса:

Для розрахунків і в коді було розглянуто порушення області визначення при **k = 0** та **k = 1.** Числа було округлено до 10 знаків після коми, тобто числа, які будуть підставлені в код і в розрахунки нижче: **Pi** = 3.1415926536; **2\*Pi** = 6.2831853072; **3\*Pi/2** = 4.7123889804; **Pi/2** = 1.5707963268. У коді помилка буде кидатися, якщо аргумент тангенса буде = **Pi/2** або **(-Pi/2)**; **3\*Pi/2** або (-**3\*Pi/2)**.

1. Порушення області визначення. Аргумент = (-Pi/2), k = 0.

числа: a = 2\*Pi, b = Pi, c = -36.7, d = 0.9

формула: (-2 \* (-36.7) - 0.9 \* 82)/tg(2\*Pi/4 - Pi) = (73.4 - 73.8)/tg(-Pi/2) = (-0.4)/tg(-Pi/2) => тангенс не визначений у (-Pi/2), кидаємо помилку в програмі.

1. Порушення області визначення. Аргумент = (3\*Pi/2), k = 1.

числа: a = 2\*Pi, b = -Pi, c = -48.9, d = 1.5

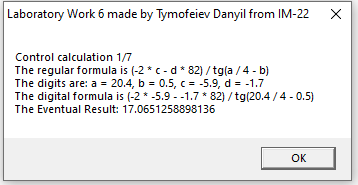
формула: (-2 \* (-48.9) - 1.5 \* 82)/tg(2\*Pi/4 + Pi) = (97.8 - 123)/tg(3\*Pi/2) = (-25.2)/tg(3\*Pi/2) => тангенс не визначений у (3\*Pi/2), кидаємо помилку в програмі.

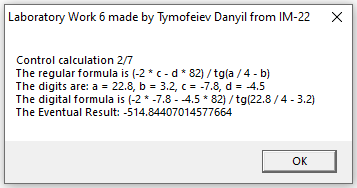
Що ж, було враховано розрахунки, коли тангенс не визначений, а саме при k = 0 аргумент = (-Pi/2); при k = 1 аргумент = (3\*Pi/2). Ще було продемонстровано розрахунки для всіх варіантів знаків чисельника та знаменника.

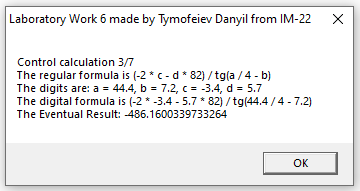
**Пункт 6.** Виконати відладку програми шляхом порівняння розрахованих програмою результатів з контрольними прикладами. Лістинг розробленої програми і скріншоти розрахунків по всіх контрольних прикладах привести в звіті по лабораторній роботі.

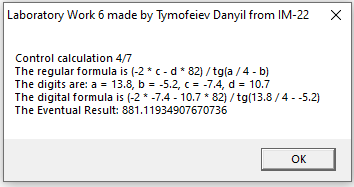
**Скріншоти виконання роботи**

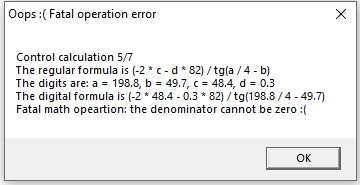
Порядок обрахунків такий самий, що і в контрольних розрахунках:

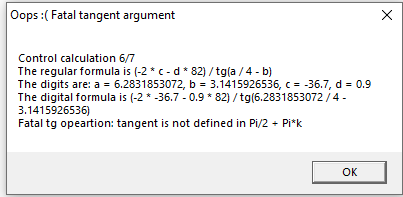


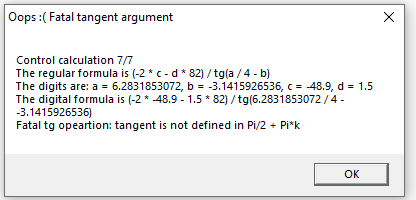












Що ж, можна побачити, що результати контрольних обрахунків і виконання програми збігаються, але з дуже і дуже маленьким відхиленням: це пов'язано з тим, що під час контрольних обрахунків було виконано округлення до 15 знаків після коми, а у програмі трішки менше. Варто зазначити, що для виводу результату в програмі було використано FloatToString2 з більшою точністю, але відрізняється вивід від розрахунків на нуль цілих та декілька трильйоних десь приблизно на 10^(-12), що свідчить про дуже високу точність виконання програми та високий збіг з теоретичними розрахунками. Також програма коректно виявляє ділення на 0 та порушення області визначення тангенса.

**Лістинг розробленої програми**

.686

.model flat, stdcall

option casemap:none

include \masm32\include\masm32rt.inc

include \masm32\include\dialogs.inc

.data?

TymofeievSummaryBuffer db 256 dup (?)

TymofeievSummaryBufferForTempValues db 256 dup (?)

TymofeievMediumResult dq ?

TymofeievEventualResult dq ?

TymofeievTemporaryAState dq ?

TymofeievRoutineAState dq ?

TymofeievTemporaryDState dq ?

TymofeievRoutineDState dq ?

TymofeievTemporaryCState dq ?

TymofeievRoutineCState dq ?

TymofeievTanDominatorRes dt ?

TymofeievNumeratorRes dt ?

TymofeievTanArgument dt ?

TymofeievARoutineString db 32 dup (?)

TymofeievBRoutineString db 32 dup (?)

TymofeievCRoutineString db 32 dup (?)

TymofeievDRoutineString db 32 dup (?)

TymofeievEventualResultBuffer dq ?

TymofeievEventualResultBufferStr db 256 dup(?)

.data

ControlCalculationsCount dd 1

ArrayStartPosition dd 0

Pi dq 3.1415926536

TwoPi dq 6.2831853072

TymofeievArrayForAValues dq 20.4, 22.8, 44.4, 13.8, 198.8, 6.2831853072, 6.2831853072

TymofeievArrayForBValues dq 0.5, 3.2, 7.2, -5.2, 49.7, 3.1415926536, -3.1415926536

TymofeievArrayForCValues dq -5.9, -7.8, -3.4, -7.4, 48.4, -36.7, -48.9

TymofeievArrayForDValues dq -1.7, -4.5, 5.7, 10.7, 0.3, 0.9, 1.5

PiOverTwo dq 1.5707963268

PiOverTwoMinus dq -1.5707963268

ThreePiOverTwo dq 4.7123889804

ThreePiOverTwoMinus dq -4.7123889804

TymofeievCheckZero dq 0.0

TymofeievNominatorTwo dq 2.0

TymofeievForTanFour dq 4.0

TymofeievEightyTwo dq 82.0

TymofeievValidLabCaption db "Laboratory Work 6 made by Tymofeiev Danyil from IM-22", 0

TymofeievFatalIOperationCaption db "Oops :( Fatal operation error", 0

TymofeievFatalIOperationBody db "Fatal math opeartion: the denominator cannot be zero :(", 0

TymofeievFatalITanArgCaption db "Oops :( Fatal tangent argument", 0

TymofeivUndefinedTanErrorMsg db "Tangent is undefined!", 0

TymofeivVariantMessgBoxInfo db "Control calculation %d/7", 13,

"The regular formula is (-2 \* c - d \* 82) / tg(a / 4 - b)", 13,

"The digits are: a = %s, b = %s, c = %s, d = %s", 13,

"The digital formula is (-2 \* %s - %s \* 82) / tg(%s / 4 - %s)", 13,

"The Eventual Result: %s", 0

TymofeivVariantInvalidMessgBoxInfo db "Control calculation %d/7", 13,

"The regular formula is (-2 \* c - d \* 82) / tg(a / 4 - b)", 13,

"The digits are: a = %s, b = %s, c = %s, d = %s", 13,

"The digital formula is (-2 \* %s - %s \* 82) / tg(%s / 4 - %s)", 13,

"Fatal math opeartion: the denominator cannot be zero :(", 0

TymofeivVariantUndefinedTanMessgBoxInfo db "Control calculation %d/7", 13,

"The regular formula is (-2 \* c - d \* 82) / tg(a / 4 - b)", 13,

"The digits are: a = %s, b = %s, c = %s, d = %s", 13,

"The digital formula is (-2 \* %s - %s \* 82) / tg(%s / 4 - %s)", 13,

"Fatal tg opeartion: tangent is not defined in Pi/2 + Pi\*k", 0

.code

TymofeievOutputNormalMSGBoxLoop PROC uses ebx edi

invoke FloatToStr2, [TymofeievArrayForAValues + edi \* 8], offset TymofeievARoutineString

invoke FloatToStr2, [TymofeievArrayForBValues + edi \* 8], offset TymofeievBRoutineString

invoke FloatToStr, [TymofeievArrayForCValues + edi \* 8], offset TymofeievCRoutineString

invoke FloatToStr, [TymofeievArrayForDValues + edi \* 8], offset TymofeievDRoutineString

invoke FloatToStr2, TymofeievEventualResultBuffer, offset TymofeievEventualResultBufferStr

invoke wsprintf, offset TymofeievSummaryBuffer, offset TymofeivVariantMessgBoxInfo,

ebx,

offset TymofeievARoutineString, offset TymofeievBRoutineString, offset TymofeievCRoutineString, offset TymofeievDRoutineString,

offset TymofeievCRoutineString, offset TymofeievDRoutineString, offset TymofeievARoutineString, offset TymofeievBRoutineString,

offset TymofeievEventualResultBufferStr

invoke MessageBox, NULL, offset TymofeievSummaryBuffer, offset TymofeievValidLabCaption, MB\_OK

ret

TymofeievOutputNormalMSGBoxLoop ENDP

TymofeievTangentProcedure PROC

; Prologue of the procedure

push ebp

mov ebp, esp

sub esp, 8 ; Reserving space for local variables

; Loading argument into) and calculating tangent

; Load the argument into the register

fptan ; Calculate the tangent

; Stack correction and procedure completion

mov esp, ebp ; Recovering the stack pointer

pop ebp ; Recovering the base stack pointer

ret ; Taking back control

TymofeievTangentProcedure ENDP

TymofeievHandleErrorMessage PROC uses ebx edi

invoke FloatToStr2, [TymofeievArrayForAValues + edi \* 8], offset TymofeievARoutineString

invoke FloatToStr2, [TymofeievArrayForBValues + edi \* 8], offset TymofeievBRoutineString

invoke FloatToStr, [TymofeievArrayForCValues + edi \* 8], offset TymofeievCRoutineString

invoke FloatToStr, [TymofeievArrayForDValues + edi \* 8], offset TymofeievDRoutineString

; Formatting an Error Message

invoke wsprintf, offset TymofeievSummaryBuffer, offset TymofeivVariantInvalidMessgBoxInfo,

ebx,

offset TymofeievARoutineString, offset TymofeievBRoutineString, offset TymofeievCRoutineString, offset TymofeievDRoutineString,

offset TymofeievCRoutineString, offset TymofeievDRoutineString, offset TymofeievARoutineString, offset TymofeievBRoutineString

; Error message output

invoke MessageBox, NULL, offset TymofeievSummaryBuffer, offset TymofeievFatalIOperationCaption, MB\_OK

inc edi

TymofeievHandleErrorMessage ENDP

start:

mov edi, 0

mov ebx, 1

TymofeievCountTheFormulaLoop:

cmp edi, 7

je TymofeievEscapeLoop

FINIT

; Compute (a / 4 - b) and store it in TymofeievTanArgument

FLD [TymofeievArrayForAValues + edi \* 8]

FDIV TymofeievForTanFour

FSUB [TymofeievArrayForBValues + edi \* 8]

FSTP TymofeievTanArgument

; Check if (a / 4 - b) equals PiOverTwoMinus

FLD TymofeievTanArgument

FCOMP PiOverTwoMinus

FSTSW ax

SAHF

JE TymofeievUndefinedTanArgumentMSGBoxLoop

; Check if (a / 4 - b) equals PiOverTwo

FLD TymofeievTanArgument

FCOMP PiOverTwo

FSTSW ax

SAHF

JE TymofeievUndefinedTanArgumentMSGBoxLoop

; Check if (a / 4 - b) equals ThreePiOverTwo

FLD TymofeievTanArgument

FCOMP ThreePiOverTwo

FSTSW ax

SAHF

JE TymofeievUndefinedTanArgumentMSGBoxLoop

; Check if (a / 4 - b) equals ThreePiOverTwoMinus

FLD TymofeievTanArgument

FCOMP ThreePiOverTwoMinus

FSTSW ax

SAHF

JE TymofeievUndefinedTanArgumentMSGBoxLoop

; Calculate the tangent only if (a / 4 - b) is not equal to (PiOverTwo | PiOverTwoMinus) OR (ThreePiOverTwo | ThreePiOverTwoMinus)

FLD TymofeievTanArgument

FPTAN

FSTP st(0) ; Discard the tangent result, we need only the division result

FSTP TymofeievTanDominatorRes

; Check if the dominator with tg is 0

FLD TymofeievTanDominatorRes

FCOMP TymofeievCheckZero

FSTSW ax

SAHF

JE TymofeievOutputInvalidMSGBoxLoop

; Compute numerator (-2 \* c - d \* 82)

FLD [TymofeievArrayForCValues + edi \* 8]

FMUL TymofeievNominatorTwo

FCHS ; Multiply by -2

FLD [TymofeievArrayForDValues + edi \* 8]

FMUL TymofeievEightyTwo

FSUB

FSTP TymofeievNumeratorRes

; Divide numerator with digits by denominator with tg

FLD TymofeievNumeratorRes

FLD TymofeievTanDominatorRes

FDIV

; SAVE!

FSTP TymofeievEventualResultBuffer

call TymofeievOutputNormalMSGBoxLoop

inc edi ; Increment calculation counter

inc ebx ; Increment window title counter

jmp TymofeievCountTheFormulaLoop

TymofeievOutputInvalidMSGBoxLoop:

invoke FloatToStr2, [TymofeievArrayForAValues + edi \* 8], offset TymofeievARoutineString

invoke FloatToStr2, [TymofeievArrayForBValues + edi \* 8], offset TymofeievBRoutineString

invoke FloatToStr, [TymofeievArrayForCValues + edi \* 8], offset TymofeievCRoutineString

invoke FloatToStr, [TymofeievArrayForDValues + edi \* 8], offset TymofeievDRoutineString

invoke wsprintf, offset TymofeievSummaryBuffer, offset TymofeivVariantInvalidMessgBoxInfo,

ebx,

offset TymofeievARoutineString, offset TymofeievBRoutineString, offset TymofeievCRoutineString, offset TymofeievDRoutineString,

offset TymofeievCRoutineString, offset TymofeievDRoutineString, offset TymofeievARoutineString, offset TymofeievBRoutineString

invoke MessageBox, NULL, offset TymofeievSummaryBuffer, offset TymofeievFatalIOperationCaption, MB\_OK

inc edi

inc ebx ; Increment window title counter

jmp TymofeievCountTheFormulaLoop

TymofeievUndefinedTanArgumentMSGBoxLoop:

invoke FloatToStr2, [TymofeievArrayForAValues + edi \* 8], offset TymofeievARoutineString

invoke FloatToStr2, [TymofeievArrayForBValues + edi \* 8], offset TymofeievBRoutineString

invoke FloatToStr, [TymofeievArrayForCValues + edi \* 8], offset TymofeievCRoutineString

invoke FloatToStr, [TymofeievArrayForDValues + edi \* 8], offset TymofeievDRoutineString

invoke wsprintf, offset TymofeievSummaryBuffer, offset TymofeivVariantUndefinedTanMessgBoxInfo,

ebx,

offset TymofeievARoutineString, offset TymofeievBRoutineString, offset TymofeievCRoutineString, offset TymofeievDRoutineString,

offset TymofeievCRoutineString, offset TymofeievDRoutineString, offset TymofeievARoutineString, offset TymofeievBRoutineString

invoke MessageBox, NULL, offset TymofeievSummaryBuffer, offset TymofeievFatalITanArgCaption, MB\_OK

inc edi

inc ebx ; Increment window title counter

jmp TymofeievCountTheFormulaLoop

TymofeievEscapeLoop:

invoke ExitProcess, 0

end start

**Висновки**

Отже, під час виконання лабораторної роботи я вивчив команди Асемблера для арифметики з плаваючою комою та практично ознайомився з виконанням розрахунків з елементами масивів. Було розроблено програму, яка обробляє 7 випадків: чисельник і знаменник більші за нуль; чисельник більше за нуль, але знаменник менший за нуль; чисельник менше за нуль, а знаменник більший за нуль; чисельник і знаменник менші за нуль; знаменник дорівнює нулю; порушення області визначення, коли аргумент = (-Pi/2), k = 0 та аргумент = (3\*Pi/2), k = 1. Обмеження для області визначення розглядав тільки для k = 0 та k = 1. Щодо порівняння результатів, то вони дуже точні, однак треба зазначити, що для теоретичних розрахунків було округлення до 15 знаків, але для програми округлення бралося трішки менше, однак ці відхилення дуже маленькі приблизно 10^(-12), що свідчить про дуже високу точність виконання програми та високий збіг з теоретичними розрахунками. Також програма правильно обробляє випадок ділення на нуль => кидає помилку. Ба більше, програма ще й обробляє випадки порушення області визначення тангенса: у коді помилка буде кидатися, якщо аргумент тангенса буде = **Pi/2** або **(-Pi/2)** при k = 0; **3\*Pi/2** або (-**3\*Pi/2)** при k = 1. Варто зазначити, що помилки не блокують вивід подальших прикладів.