**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-21 доцент

Лесько Дмитро Миколайович Павлов В.Г.

Номер у списку групи: 14

**Київ 2024**

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

**Виконання роботи:**

Як і сказано у завданні, я розробив код для обчислення значень рівняння, використовуючи наші змінні.

Далі розширив програму, додавши масив різних чисел (значення наших змінних) та додавши цикл, що проходиться по масиву цих чисел і видає нам відповіді. Для виводу значень і обрахування використовуються дійсні числа.

Згадавши досвід роботи з макросами з минулої лабораторної роботи, я взяв макрос для виведення повідомлень і використав його у своєму коді.

За основу була взята моя лабораторна робота №5. Як я тоді й казав, завдяки виділенням обчислення знаменника і чисельника в окремі макроси я зміг швидко переробити обчислення під нові умови і вдало все обрахувати.

**Детальні розрахунки контрольних прикладів:**

**Наша формула: ( 4 \* lg(c) - b/2 + 23)/(d - a + 1)**

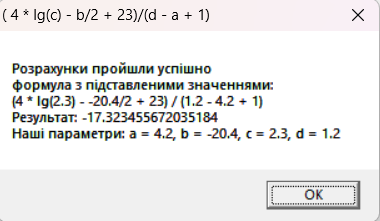
**Перший випадок**:

а = 4.2, б = -20.4, с = 2.3, d = 1.2

Розрахунок:

(4\*lg(2.3)+20.4/2+23)/(1.2-4.2+1)= (4\*0.3617278360175928+10.2+23)/(-2) = 34.64691134407037/-2 = -17.32345567203518

Скріншот з програми:



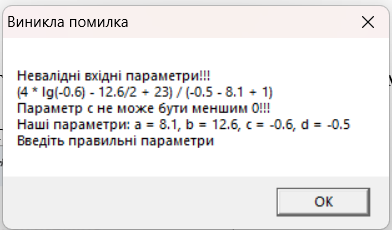
**Другий випадок**:

а = 8.1, б = 12.6, с = -0.6, d = -0.5

Розрахунок:

Цей випадок (приклад) демонструє вдалу валідацію вхідних параметрів нашою програмою. У цьому випадку підставлене число в логарифм менше 0, що порушує його область визначення, тому ми виводимо відповідне повідомлення

Скріншот з програми:



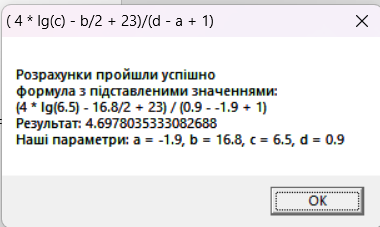
**Третій випадок**:

а = -1.9, б = 16.8, с = 6.5, d = 0.9

Розрахунок:

(4\*lg(6.5)-16.8/2+23)/(0.9+1.9+1)= (4\* 0.8129133566428555-8.4+23)/(3.8) = 17.85165342657142/3.8 = 4.697803533308269

Скріншот з програми:



\*Результат не збігається останніми числами, через округлення калькулятором

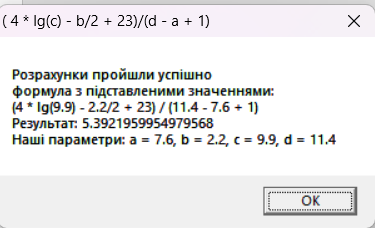
**Четвертий випадок:**

а = 7.6, б = 2.2, с = 9.9, d = 11.4

Розрахунок:

(4\*lg(9.9)-2.2/2+23)/(11.4-7.6+1)= (4\* 0.9956351945975499-1.1+23)/(4.8) = 25.8825407783902/4.8 = 5.392195995497958

Скріншот з програми:



**П’ятий випадок(з нулем у знаменнику):**

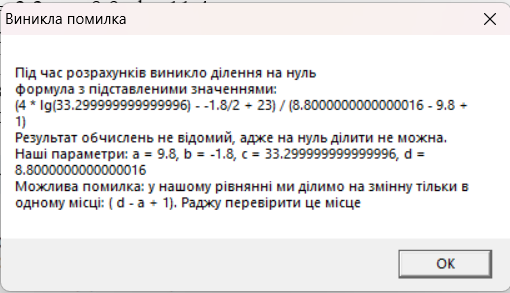
а = 7.6, б = 2.2, с = 9.9, d = 11.4

Розрахунок:

(4\*lg(33.3)+1.8/2+23)/(8.8-9.8+1)= 29.98977693402528/ 0 = Infinity?

У цьому випадку ми отримуємо 0 у знаменнику, тому програма не йде далі, а виводить повідомлення про виниклу помилку.

Скріншот з програми:



У такому випадку після виведення повідомлення ми пропускаємо цей цикл і йдемо далі рахувати числа.

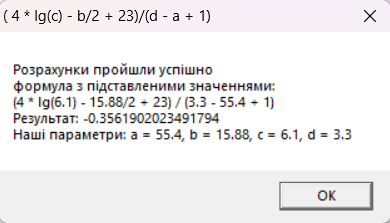
**Шостий випадок:**

а = 55.4, б = 15.88, с = 6.1, d = 3.3

Розрахунок:

(4\*lg(6.1)-15.88/2+23)/(3.3-55.4+1)= (4\* 0.7853298350107669-7.94+23)/( -51.1) = 18.20131934004307/-51.1 = -0.3561902023491794

Скріншот з програми:



**Лістенінг:**

.386

.model flat, stdcall

option casemap :none

include \masm32\include\windows.inc

include \masm32\macros\macros.asm

include \masm32\include\kernel32.inc

include \masm32\include\masm32.inc

include \masm32\include\user32.inc

includelib \masm32\lib\masm32.lib

includelib \masm32\lib\user32.lib

includelib \masm32\lib\kernel32.lib

.data?

equationBuffer dd 512 dup(?)

answer dq 8 dup(?)

divider dt 1 dup(?)

shared dt 1 dup(?)

Acurrent db 32 dup(?)

Bcurrent db 32 dup(?)

Ccurrent db 32 dup(?)

Dcurrent db 32 dup(?)

result db 64 dup(?)

.data

zero\_cause\_msg\_box\_title db "Виникла помилка", 0

msg\_box\_title db "( 4 \* lg(c) - b/2 + 23)/(d - a + 1)", 0

zero\_cause\_output\_msg\_box db "Під час розрахунків виникло ділення на нуль", 10, 13,

"формула з підставленими значеннями: ", 10, 13,

"(4 \* lg(%s) - %s/2 + 23) / (%s - %s + 1)", 10, 13,

"Результат обчислень не відомий, адже на нуль ділити не можна.", 10, 13,

"Наші параметри: а = %s, b = %s, c = %s, d = %s", 10, 13,

"Можлива помилка: у нашому рівнянні ми ділимо на змінну тільки в одному місці: ( d - a + 1). Раджу перевірити це місце", 0

default\_output\_msg db "Розрахунки пройшли успішно", 10, 13,

"формула з підставленими значеннями: ", 10, 13,

"(4 \* lg(%s) - %s/2 + 23) / (%s - %s + 1)", 10, 13,

"Результат: %s ", 10, 13,

"Наші параметри: а = %s, b = %s, c = %s, d = %s", 0

c\_less\_than\_zero\_msg db "Невалідні вхідні параметри!!!", 10, 13,

"(4 \* lg(%s) - %s/2 + 23) / (%s - %s + 1)", 10, 13,

"Параметр с не може бути меншим 0!!!", 10, 13,

"Наші параметри: а = %s, b = %s, c = %s, d = %s", 10, 13,

"Введіть правильні параметри", 0

Avalue dq 4.2, 8.1, -1.9, 7.6, 9.8, 55.4

Bvalue dq -20.4, 12.6, 16.8, 2.2, -1.8, 15.88

Cvalue dq 2.3, -0.6, 6.5, 9.9, 33.3, 6.1

Dvalue dq 1.2, -0.5, 0.9, 11.4, 8.8, 3.3

arraySize equ ($ - Avalue) / 32

twentyThree dq 23.0

ten dq 10.0

four dq 4.0

two dq 2.0

one dq 1.0

zero dq 0.0

.code

call\_msg\_window MACRO title:REQ, msg:REQ

invoke MessageBox, 0, addr msg, addr title, 0

ENDM

count\_divider MACRO index:REQ

; Calculate the denominator (d - a + 1)

fld [Avalue + index \* 8] ; a

fld [Dvalue + index \* 8] ; d

fsub st(0), st(1) ; d - a

fld one

fxch st(1)

fadd st(0), st(1) ; (d - a) + 1

fstp divider

ENDM

count\_shared MACRO index:REQ

; Load values from memory into FPU stack

fld [one]

fld qword ptr [Cvalue + index \* 8] ; Load c

fyl2x ; Calculate logarithm base 2 of c: log2(c)

fldl2t ; log2(10)

fdiv ; Divide log2(c) by log2(10): log2(c) / log2(10) = lg(c)

fld qword ptr [four] ; Load 4

fmul ; Multiply: 4 \* log10(c)

fld qword ptr [twentyThree] ; Load 23

fadd ; Add 23: 4 \* log10(c) + 23

fld qword ptr [Bvalue + index \* 8] ; Load b

fld qword ptr [two] ; Load 2

fdiv ; Divide b by 2: b / 2

fsub ; Subtract b / 2 from 4 \* log10(c) + 23

fstp shared ; Store the result in shared

ENDM

main:

mov esi, 0

valueLoop:

cmp esi, arraySize

jge endLoop

finit

invoke FloatToStr2, [Avalue + esi \* 8], addr Acurrent

invoke FloatToStr2, [Bvalue + esi \* 8], addr Bcurrent

invoke FloatToStr2, [Cvalue + esi \* 8], addr Ccurrent

invoke FloatToStr2, [Dvalue + esi \* 8], addr Dcurrent

fld qword ptr [Cvalue + esi \* 8] ; Load the current value of C

fldz ; Load 0

fcom ; Compare C with 0

fstsw ax ; Store the FPU status word in AX

sahf ; Set the flags in EFLAGS register

; Test if the C0 (greater than) flag is set

test ah, 00000001b ; Check if bit 0 (C0 flag) is set

jnz continueWithCode ; If the C0 flag is set, jump to continueWithCode

; If C is less than or equal to 0, jump to errorNegativeC

jmp errorNegativeC

continueWithCode:

count\_divider esi ; Calculate the denominator

count\_shared esi ; Calculate the numerator

fld qword ptr [divider] ; Load the divider value

ftst ; Compare the divider value with 0

fnstsw ax ; Store the FPU status word in AX

sahf ; Set the flags in EFLAGS register

test ah, 01000000b ; Check if bit 7 (C3 flag) is set

jnz zeroDenominatorError ; If the C3 flag is not set, jump to error handling

; Perform the division and store the result

fld shared

fld divider

fdiv

fstp answer

; Format the output message

invoke FloatToStr2, answer, addr result

invoke wsprintf, addr equationBuffer, addr default\_output\_msg,

addr Ccurrent, addr Bcurrent, addr Dcurrent, addr Acurrent,

addr result,

addr Acurrent, addr Bcurrent, addr Ccurrent, addr Dcurrent

call\_msg\_window msg\_box\_title, equationBuffer

jmp loopEnd

errorNegativeC:

; Display error message for C, when it is lower than 0

invoke wsprintf, addr equationBuffer, addr c\_less\_than\_zero\_msg,

addr Ccurrent, addr Bcurrent, addr Dcurrent, addr Acurrent,

addr Acurrent, addr Bcurrent, addr Ccurrent, addr Dcurrent

call\_msg\_window zero\_cause\_msg\_box\_title, equationBuffer

jmp loopEnd

zeroDenominatorError:

; Display error message for division by zero

invoke wsprintf, addr equationBuffer, addr zero\_cause\_output\_msg\_box,

addr Ccurrent, addr Bcurrent, addr Dcurrent, addr Acurrent,

addr Acurrent, addr Bcurrent, addr Ccurrent, addr Dcurrent

call\_msg\_window zero\_cause\_msg\_box\_title, equationBuffer

loopEnd:

add esi, 1

jmp valueLoop

endLoop:

invoke ExitProcess, NULL

end main

Як бачимо, у своєму коді я використав один макрос з минулої лабораторної роботи (№4), та два з лабораторної роботи №5, для правильності роботи я змінив формулу та методи обчислення, а також написав два нових для окремого обчислення значення числівника та знаменника, тобто діленого та дільника. Я вирішив, що це виглядає краще, і ніяк не впливає на результат. Також для проходження по усім значенням ми використовуємо цикл, який корегується відповідно довжині масиву. Відповідно до результату обчислень виводиться потрібне повідомлення. Загалом програма працює правильно і після перевірки правильності обчислень за допомогою ручного підрахунку контрольних випадків, я ще більше у цьому впевнився.

Для зручності я помітив коментарями відповідні етапи обрахунку по формулі.

**Висновок**

Виконавши цю лабораторну роботу я вивчив команди для обчислень з числами з плаваючою формулою. Розглянув різні методи обчислень з рівзних сторін (виведення формул логарифму) мови Assembler, та розробив власну програму, яка їх використовую. Також повторив використання макросів та інших корисних бібліотек. Провівши контрольні розрахунки і порівнявши їх з отриманими результатами з програми, Я впевнився, що вона працює правильно та вдало рахує результат по вказаній у варіанті формулі. Також, протягом розробки, у мене виникали проблеми з точністю та вірністю розрахунків, тому я додатково «дебажив» і впевнювався у вірності значень чисельника та знаменника, то впевнений, що там встановлюються правильні значення. Всі результати співпадають. Була також проблема зі стеком та обчислень логарифму за основою десяти. Для цього я вивів формулу та додатково розібрався у правильності його роботи.