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 №7**

з дисципліни «Системне програмування» на тему

«Модульне програмування. Використання процедур»

Виконав: Перевірив:

студент II курсу ФІОТ доц. Павлов В. Г. групи ІМ-22 Басараб Станіслав Анатолійович  
номер у списку групи: 2

Київ 2024

**Мета роботи**: Вивчення прийомів модульного програмування, методів звернення до процедур і передачі в них параметрів.

**Порядок виконання роботи**

1. Вивчити методи звернення до процедур і передачі в них параметрів.

2. Для свого варіанту індивідуального завдання до лабораторної роботи 6 розробити програму на мові Асемблер, в якій використовувати три процедури з різними способами передачі параметрів:

• через регістри;

• через стек;

• за допомогою директив EXTRN та PUBLIC.

3. Для цього чисельник дробу зі свого варіанту індивідуального завдання до лабораторної роботи 6 розділити на два доданка, з яких для першого застосувати передачу параметрів і результату через регістри, а для другого – через стек. Для знаменника використовувати метод оголошення загальних змінних директивами public і extern. Виведення результату виконати в основній програмі.

4. Розрахунки (п. 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, які повинні охоплювати різноманітні сполучення вхідних даних, на які програма повинна надавати вірну відповідь. Проміжні і остаточні результати контрольних розрахунків привести в звіті по лабораторній роботі. Точність розрахунків така ж, як і у лаб. роботі 6.

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. У протоколі по лабораторній роботі для першого і другого способів передачі параметрів поруч з відповідними командами у лістингу відобразити в графічному вигляді стани стека при зверненні до процедур, виконання у них команд та повернення з процедур до основної програми.

8. Зробити висновки по лабораторній роботі.

**Виконання роботи**

Формула для розрахунку за 2 варіантом: (-2 \* c - d \* 82) / tg(a / 4 - b).

Контрольні розрахунки

1. Знаменник і чисельник > 0

a = -8,8  
b = 6,3  
c = 4,5

d = -1,3

(-2 \* 4,5 - (-1,3) \* 82) / tg(-8,8 / 4 – 6,3) = (-9 + 106,6) / tg(-2,2 - 6,3) =  
= 97,6 / tg(-8,5) = 97,6 / 1,326364327785607 = 73,5846086594814

Остаточний результат = 73,5846086594814

1. Чисельник < 0, а знаменник > 0

a = 9,2

b = 7,4

c = 9,6

d = 1,1

(-2 \* 9,6 - 1,1 \* 82) / tg(9,2 / 4 - 7,4) = (-19,2 - 90,2) / tg(2,3 - 7,4) =  
= -109,4 / tg(-5,1) = -109,4 / 2,4493894155845926 =  
= -44.66419235092908

Остаточний результат = -44.66419235092908

1. Чисельник > 0, а знаменник < 0

a = 1,1

b = 1,2

c = 9,7

d = -5,5

(-2 \* 9,7 – (-5,5) \* 82) / tg(1,1 / 4 – 1,2) = (-19,4 + 451) / tg(0,275 - 1,2) = 431,6 / tg(-0,925) = 431,6 / -1,3269771683242817 =   
= -325,2505094304134

Остаточний результат = -325,2505094304134

1. Знаменник і чисельник < 0

a = -7,3

b = 8,5

c = -1,3

d = 3,4

(-2 \* (-1,3) – 3,4 \* 82) / tg(-7,3 / 4 – 8,5) = (2,6 - 278,8) / tg(-1,825 - 8,5) = -276,2 / tg(-10,325) = -276,2 / -1,2607330156384895 =   
= 219,07889820757998

Остаточний результат = 219,07889820757998

1. Знаменник = 0

a = 11,5

b = 2,875

c = 9,2

d = 3,1

(-2 \* 9,2 – 3,1 \* 82) / tg(11,5 / 4 – 2,875) =  
= (-18,4 – 254,2) / tg(2,875 – 2,875) = -272,6 / tg(0) = -272,6 / 0

Оскільки формула містить тригонометричну функцію, що має свою область визначення, а саме tg(x) – множина всіх дійсних чисел, де x != pi/2 + pi\*n, створимо перевірку області тільки для n = 0 (адже якщо перевіряти усю множину періодів функції тангенса, то реалізація такої перевірки буде досить не точною). Також оскільки pi – ірраціональне число, візьмемо його наближене значення з шістьома знаками після коми pi = 3,141592

1. Порушення області визначення tg(pi/2)

a = 3,141592

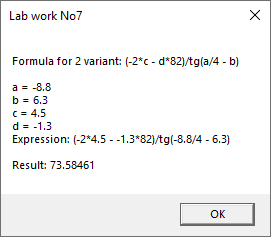
b = -0,785398

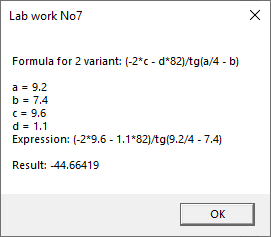
c = 5,23

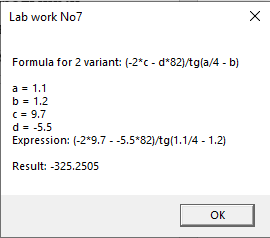
d = 41,2

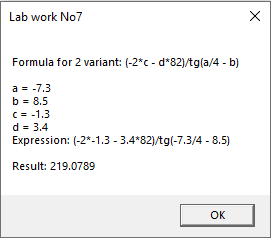
(-2 \* 5,23 – 41,2 \* 82) / tg(3,141592 / 4 – (-0,785398)) =  
= (-10,46 – 3378,4) / tg(0,785398 + 0,785398) = -3388,86 / tg(1,570796) або ж -3388,86 / tg(pi/2)

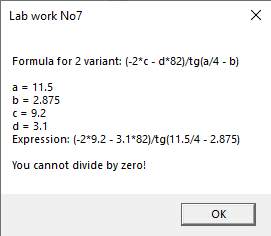
Робота програми

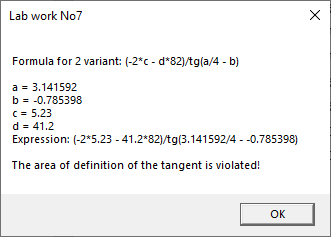












Порівнюючи результати виконання програми із контрольними розрахунками можна сказати, що програма працює коректно. Проте на відміну від контрольних розрахунків, комп’ютер виконує округлення до 6 знаків після коми. Тож перевірена робота програми у таких випадках:

1. Знаменник і чисельник > 0
2. Чисельник < 0, а знаменник > 0
3. Чисельник > 0, а знаменник < 0
4. Знаменник і чисельник < 0
5. Знаменник = 0
6. Порушення області визначення tg(pi/2)

Лістинг програми  
(стани стеку виділено жовтим кольором)

7-2-IM-22-Basarab.asm:

.386

.model flat, stdcall

option casemap:none

include \masm32\include\masm32rt.inc

public SBasarabFourNum, SBasarabMasivForBukvaA, SBasarabMasivForBukvaB

extern stbasarabCollectDown:proto

stbasarabCheckTg macro

fld SBasarabMasivForBukvaA[esi \* 8]

fdiv SBasarabFourNum

fld SBasarabMasivForBukvaB[esi \* 8]

fsub

fstp SBasarabTanArgB

invoke FloatToStr, SBasarabMasivForBukvaA[esi \* 8], addr SBasarabThisAStringB

invoke FloatToStr, SBasarabMasivForBukvaB[esi \* 8], addr SBasarabThisBStringB

invoke FloatToStr, SBasarabMasivForBukvaC[esi \* 8], addr SBasarabThisCStringB

invoke FloatToStr, SBasarabMasivForBukvaD[esi \* 8], addr SBasarabThisDStringB

; if arg == 0

fld SBasarabTanArgB ; load tg arg

fldz ; load zero

fcom ; comparison

fnstsw ax ; save to ax

sahf ; check flag

jz basarab\_divide\_by\_zero

fld SBasarabTanArgB

fstp SBasarabReminder

mov eax, dword ptr [SBasarabReminder]

mov ebx, dword ptr [SBasarabPiNa2Num]

cmp eax, ebx

je basarab\_invalid\_arg

endm

.data

SBasarabWinCaption db "Lab work No7", 0

SBasarabMainWinContent db "Formula for 2 variant: (-2\*c - d\*82)/tg(a/4 - b)", 10, 10,

"a = %s", 10,

"b = %s", 10,

"c = %s", 10,

"d = %s", 10,

"Expression: (-2\*%s - %s\*82)/tg(%s/4 - %s)", 10, 10,

"%s", 0

SBasarabMasivForBukvaA dq -8.8, 9.2, 1.1, -7.3, 11.5, 3.141592

SBasarabMasivForBukvaB dq 6.3, 7.4, 1.2, 8.5, 2.875, -0.785398

SBasarabMasivForBukvaC dq 4.5, 9.6, 9.7, -1.3, 9.2, 5.23

SBasarabMasivForBukvaD dq -1.3, 1.1, -5.5, 3.4, 3.1, 41.2

SBasarabResultWinContent db "Result: %s", 0

SBasarabDivideByZeroWinContent db "You cannot divide by zero!", 0

SBasarabTgScopeWinContent db "The area of definition of the tangent is violated!", 0

SBasarabFourNum dq 4.0

SBasarabMinusTwoNum dq -2.0

SBasarabEightyTwoNum dq -82.0

SBasarabPiNum dq 3.141592

SBasarabPiNa2Num dq 1.570796

.data?

SBasarabWinContentB db 256 dup (?)

SBasarabResultContentB db 256 dup (?)

SBasarabThisAStringB db 128 dup (?)

SBasarabThisBStringB db 128 dup (?)

SBasarabThisCStringB db 128 dup (?)

SBasarabThisDStringB db 128 dup (?)

SBasarabResultStringB db 128 dup (?)

SBasarabReminder dq ?

SBasarabTanArgB dt ?

SBasarabUpB dt ?

SBasarabDownB dt ?

SBasarabMultiplicationNum1 dt ?

SBasarabMultiplicationNum2 dt ?

SBasarabResultB dq ?

.code

stbasarabCollectFirstPart proc

; -2 \* c

fld qword ptr [ebx]

fld qword ptr [edx]

fmul

fstp SBasarabMultiplicationNum1

mov eax, dword ptr [SBasarabMultiplicationNum1]

ret**(4)**

stbasarabCollectFirstPart endp

stbasarabCollectSecondPart proc

push ebp**(8)**

mov ebp, esp

mov ebx, [ebp+8]

mov edx, [ebp+12]

; -82 \* d

fld qword ptr [ebx]

fld qword ptr [edx]

fmul

pop ebp**(9)**

ret 8**(10)**

stbasarabCollectSecondPart endp

basarabLab6:

mov esi, 0

cycl\_for\_counting:

finit

stbasarabCheckTg ; check for 0 and pi/2

call stbasarabCollectDown**(1)** ; collect denominator

fstp st(0)

fstp SBasarabDownB

mov ebx, offset SBasarabMinusTwoNum

lea edx, SBasarabMasivForBukvaC[esi \* 8]

call stbasarabCollectFirstPart**(3)**

mov dword ptr [SBasarabMultiplicationNum1], eax

lea eax, SBasarabMasivForBukvaD[esi \* 8]

push offset SBasarabEightyTwoNum**(5)**

push eax**(6)**

call stbasarabCollectSecondPart**(7)**

fstp SBasarabMultiplicationNum2

; -2 \* c + (-82) + d

fld SBasarabMultiplicationNum1

fld SBasarabMultiplicationNum2

fadd

fstp SBasarabUpB

; (-2 \* c - d \* 82) / tg(a / 4 - b)

fld SBasarabUpB

fld SBasarabDownB

fdiv

fstp SBasarabResultB

invoke FloatToStr, SBasarabResultB, addr SBasarabResultStringB

invoke wsprintf, addr SBasarabResultContentB, addr SBasarabResultWinContent, addr SBasarabResultStringB

invoke wsprintf, addr SBasarabWinContentB, addr SBasarabMainWinContent,

addr SBasarabThisAStringB, addr SBasarabThisBStringB, addr SBasarabThisCStringB, addr SBasarabThisDStringB,

addr SBasarabThisCStringB, addr SBasarabThisDStringB, addr SBasarabThisAStringB, addr SBasarabThisBStringB,

addr SBasarabResultContentB

basarab\_return:

invoke MessageBox, 0, addr SBasarabWinContentB, addr SBasarabWinCaption, 0

inc esi

.if esi == 6

invoke ExitProcess, 0

.endif

jmp cycl\_for\_counting

basarab\_divide\_by\_zero:

invoke wsprintf, addr SBasarabWinContentB, addr SBasarabMainWinContent,

addr SBasarabThisAStringB, addr SBasarabThisBStringB, addr SBasarabThisCStringB, addr SBasarabThisDStringB,

addr SBasarabThisCStringB, addr SBasarabThisDStringB, addr SBasarabThisAStringB, addr SBasarabThisBStringB,

addr SBasarabDivideByZeroWinContent

jmp basarab\_return

basarab\_invalid\_arg:

invoke wsprintf, addr SBasarabWinContentB, addr SBasarabMainWinContent,

addr SBasarabThisAStringB, addr SBasarabThisBStringB, addr SBasarabThisCStringB, addr SBasarabThisDStringB,

addr SBasarabThisCStringB, addr SBasarabThisDStringB, addr SBasarabThisAStringB, addr SBasarabThisBStringB,

addr SBasarabTgScopeWinContent

jmp basarab\_return

end basarabLab6

7-2-IM-22-Basarab-PUBLIC.asm:

.386

.model flat, stdcall

option casemap:none

public stbasarabCollectDown

extern SBasarabFourNum:qword, SBasarabMasivForBukvaA:qword, SBasarabMasivForBukvaB:qword

.code

stbasarabCollectDown proc

; a / 4

fld SBasarabMasivForBukvaA[esi \* 8]

fdiv SBasarabFourNum

; a / 4 - b

fld SBasarabMasivForBukvaB[esi \* 8]

fsub

; tg(a / 4 - b)

fptan

ret**(2)**

stbasarabCollectDown endp

end

7-2-IM-22-Basarab.bat:

@echo off

ML /c /coff "7-2-IM-22-Basarab.asm"

ML /c /coff "7-2-IM-22-Basarab-PUBLIC.asm"

LINK /subsystem:windows "7-2-IM-22-Basarab.obj" "7-2-IM-22-Basarab-PUBLIC.obj"

7-2-IM-22-Basarab.exe

**Cтани стеку**

|  |  |
| --- | --- |
| Стан 1. call stbasarabCollectDown | |
| max |  |
| … | … |
| **EIP** | Адреса повернення з процедури в основну програму |
|  |  |
| min |  |

|  |  |
| --- | --- |
| Стан 2. ret (відбулось повернення в основну програму) | |
| max |  |
| … | … |
|  |  |
|  |  |
| min |  |

|  |  |
| --- | --- |
| Стан 3. call stbasarabCollectFirstPart | |
| max |  |
| … | … |
| **EIP** | Адреса повернення з процедури в основну програму |
|  |  |
| min |  |

|  |  |
| --- | --- |
| Стан 4. ret (відбулось повернення в основну програму) | |
| max |  |
| … | … |
|  |  |
|  |  |
| min |  |

|  |  |
| --- | --- |
| Стан 5. push offset SBasarabEightyTwoNum | |
| max |  |
| … | … |
| **SBasarabEightyTwoNum** | Дійсне число для здійснення обрахунків |
|  |  |
| min |  |

|  |  |
| --- | --- |
| Стан 6. push eax | |
| max |  |
| … | … |
| **SBasarabEightyTwoNum** | Дійсне число для здійснення обрахунків |
| **EAX** | Адреса дійсного числа з масиву чисел d |
|  |  |
| min |  |

|  |  |
| --- | --- |
| Стан 7. call stbasarabCollectSecondPart | |
| max |  |
| … | … |
| **SBasarabEightyTwoNum** | Дійсне число для здійснення обрахунків |
| **EAX** | Адреса дійсного числа з масиву чисел d |
| **EIP** | Адреса повернення з процедури в основну програму |
|  |  |
| min |  |

|  |  |
| --- | --- |
| Стан 8. push ebp | |
| max |  |
| … | … |
| **SBasarabEightyTwoNum** | Дійсне число для здійснення обрахунків |
| **EAX** | Адреса дійсного числа з масиву чисел d |
| **EIP** | Адреса повернення з процедури в основну програму |
| **EBP** | Поточне значення регістра EBP |
|  |  |
| min |  |

|  |  |
| --- | --- |
| Стан 9. pop ebp (повернення значення EBP зі стеку) | |
| max |  |
| … | … |
| **SBasarabEightyTwoNum** | Дійсне число для здійснення обрахунків |
| **EAX** | Адреса дійсного числа з масиву чисел d |
| **EIP** | Адреса повернення з процедури в основну програму |
|  |  |
| min |  |

|  |  |
| --- | --- |
| Стан 10. ret 8 (повернення з процедури в основну програми, а також зсув стеку на 8 байтів) | |
| max |  |
| … | … |
|  |  |
| min |  |

**Висновки**

Отже, під час виконання лабораторної роботи було вивчено прийоми модульного програмування. Здобуто навички зведення до процедур і передачі в них параметрів. Написано програму на мові асемблера для обчислення значення арифметичного виразу із застосуванням команд співпроцесора. У програмі використано три процедури з різними способами передачі параметрів: через регістри, стек та за допомогою директив PUBLIC та EXTERN. Заздалегідь виконано контрольні розрахунки для шести випадків:

1. Знаменник і чисельник > 0
2. Чисельник < 0, а знаменник > 0
3. Чисельник > 0, а знаменник < 0
4. Знаменник і чисельник < 0
5. Знаменник = 0
6. Порушення області визначення tg(pi/2)

Між контрольними розрахунками і результатами роботи програми є незначні відмінності, які полягають в тому, що комп’ютер здійснює округлення до шести знаків після коми. А так в цілому результати співпадають, що говорить про правильність роботи процедур і в цілому програми. Додатково було досліджено стани стеку. Під час передачі параметрів у процедуру через стек, вони доступні через регістр EBP, що ставиться у пролозі процедури. На початку його значення рівне регістру ESP. Додатково при виклику процедури за допомогою call, у стек додається регістр EIP, що містить адресу повернення в основну програму. Саме після виконання команди ret, ми повертаємо цю адресу зі стеку для повернення в основну програму.