Міністерство освіти і науки України

Одеський національний політехнічний університет

Iнститут компьютерних систем

Кафедра компьютерних інтеллектуальних систем і мереж

Лабораторна робота №3

З дисципліни: “Архiтектура комп’ютерiв”

По темi: “Спрощений опис сегментів. Обчислення цілочисельних арифметичних виразів.”

Зробив

Ст. Борщов М.I.

Групи АМ-181

Перевiрили:

Шапорiн Р.О.

Одеса 2020

**Мета:** Ознайомитися з описом сегментів та арифметичними командами.

**Завдання:**

1. Вибрати свій варіант арифметичного виразу.



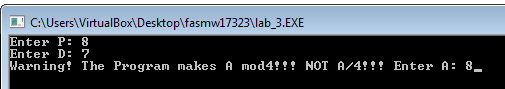
2. Написати модуль мовою Асемблера для обчислення значення виразу.

3. Написати на базовій алгоритмічній мові програму коректного вводу вихідних даних (з контролем

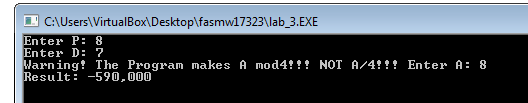
припустимого діапазону) і виводу отриманого результату.

4. Вмонтувати виклик модуля мовою Асемблера в програму на базовій алгоритмічній мові.

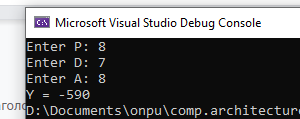
5. Зробити тестові перевірки, зробити аналіз результатів.



ввод чисел



результат



перевірка обчислень у прогаррмі на С++

6. Занесіть до протоколу:

- складений арифметичний вираз;

- вихідні дані й отримані результати обчислень у шістнадцятирічній формі, а також їхнє місце розташування

в оперативній пам'яті і їхні еквіваленти в десятковій формі (для можливості перевірки обчислень);

- текст програми і її опис.

Код програми

format PE console

entry Start

include 'INCLUDE\win32a.inc'

section '.data' data readable writable

strP db 'Enter P: ', 0 ;text for P

strD db 'Enter D: ', 0 ;text for P

strA db 'Warning! The Program makes A mod4!!! NOT A/4!!! Enter A: ', 0 ;text for P

resStr db 'Result: %d', 0 ; output with C libraries

resMod db '/%d' , 0 ;for drobi

spaceStr db ' %d', 0 ;just space

emptyStr db '%d',0 ;empty Str for numbers

infinity db 'infinity', 0 ; when /0

point db ',', 0

P dd ?

D dd ?

A dd ?

RES dd ?

FstSt dd ?

ScndSt dd ?

ThrdSt dd ?

FthSt dd ?

FfthSt dd ?

NULL = 0

section '.code' readable executable

Start:

push strP ; entering text for P

call [printf]

push P ;entering P

push spaceStr

call [scanf]

push strD ; entering text D

call [printf]

push D ;entering D

push spaceStr

call [scanf]

push strA ; entering text A

call [printf]

push A ;entering A

push spaceStr

call [scanf]

;calculating V.3 Y = ( -2\*P - D\*82 ) / ( A/4 -1 )

; A/4 (FirstStep) FstSt

cmp [A],4

jne Go ;if No - calculating

push infinity ; if Yes - exit the programm

call [printf]

jmp exit

Go:

mov eax, [A]

mov ecx, 4

mov edx, 0

div ecx

mov [FstSt], eax

;( A/4 - 1 ) Second Step ScndSt

mov ecx, [FstSt]

sub ecx, 1

mov [ScndSt], ecx

; -2\*P Third step ThrdSt

mov ecx, [P]

imul ecx, -2

mov [ThrdSt], ecx

; D \* 82 Fourth Step FthSt

mov ecx, [D]

imul ecx, 82

mov [FthSt], ecx

;( -2\*P - D\*82 ) fifth Step FfthSt

mov ecx, [ThrdSt]

sub ecx, [FthSt]

mov [FfthSt], ecx

; Result RES

mov eax, [FfthSt]

mov ecx, [ScndSt]

mov edx, 0

cmp [ScndSt],0 ;D == 0?

jne NotNullDiv1 ;if No - calculating

push infinity ; if Yes - exit the programm

call [printf]

jmp exit

NotNullDiv1:

div ecx

mov [RES], edx

push eax

push resStr

call [printf]

push point

call [printf]

mov ebx, 0

lp:

mov eax, [RES]

mov ecx, [D]

imul eax, 10

mov edx, 0

div ecx

mov [RES], edx

push eax

push emptyStr

call [printf]

add ebx, 1

cmp ebx, 3

jne lp

jmp exit

jmp exit

exit:

call [getch]

push NULL

call [ExitProcess]

section '.idata' import data readable

library kernel, 'kernel32.dll',\

msvcrt, 'msvcrt.dll'

import kernel,\ ; metki for comfortable usage

ExitProcess, 'ExitProcess'

import msvcrt,\

printf, 'printf',\

scanf, 'scanf', \

getch, '\_getch'

**Висновок:** У цій лабораторній роботі я навчився сегментувати код програми та вікористовувати арифметичні команди на ЯП ASSEMBLER. Для більш зручного вводу та виводу інформації додатково навчився користуваватися можливостями бібліотеки “win32a.inc”, яка йде разом з програмою для кодування “Flat Assembler”