Міністерство освіти і науки України

Одеський національний політехнічний університет

Iнститут компьютерних систем

Кафедра компьютерних інтеллектуальних систем і мереж

Лабораторна робота №4

З дисципліни: “Архiтектура комп’ютерiв”

По темi: “Організація переходів. Режим Ideal (поліпшений)”

Зробив

Ст. Баженов М.I.

Групи АМ-181

Перевiрили:

Шапорiн Р.О.

Одеса 2020

**Мета:** Ознайомитися з командами безумовної передачі управління JMP, передачі параметрів у підпрограми.

**Завдання:**

Завдання до лабораторної роботи.

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.





Вивід результату коли “А” та “В” однакові







Вивід результату коли “А” > “В” та перевірка якщо “А” = 0







Вивід результату коли “А” < “В” та перевірка якщо “B” = 0

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

strA db 'Enter A: ', 0

strB db 'Enter B: ', 0

resStr db 'Y = %d', 0 ; output with C libraries

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

spaceStr db ' %d', 0 ;just space

infinity db 'infinity', 0 ; when /0

point db ',', 0

A dd ?

B dd ?

RES dd ?

FstSt dd ?

ScndSt dd ?

ThrdSt dd ?

temp dd ?

NULL = 0

section '.code' readable executable

Start:

push strA ; entering text for A

call [printf]

push A ;entering A

push spaceStr

call [scanf]

push strB ; entering text B

call [printf]

push B ;entering B

push spaceStr

call [scanf]

;calculating V.2 A>B Y = (A - B) / A + 1 | a=b Y = 2 | A<B Y = ( A \* A \* A + 1) / B

; if (A == B)

mov ecx, [A]

cmp ecx, [B]

jne GoNext ;if No - calculating

mov [RES], 2

push [RES]

push resStr ; if Yes - Y = 2 and exit the programm

call [printf]

jmp exit

GoNext:

mov ecx, [A]

sub ecx, [B]

and ecx, 8000h ;check if A-B > 0

jne AsmallerB ;if A-B < 0 (A < B)

je BsmallerA ;if A-B > 0 (A > B)

AsmallerB: ;A<B A Y = ( A \* A - B) / B

; A \* A \* A FstSt

mov ecx, [A]

imul ecx, [A]

mov [FstSt], ecx

mov ecx, [FstSt]

imul ecx, [A]

mov [FstSt], ecx

; FstSt + 1 ScndSt

mov eax, [FstSt]

add eax, 1

mov [ScndSt], eax

; ScndSt / B

cmp [B], 0

jne Go1 ;if No - calculating

push infinity ; if Yes - exit the programm

call [printf]

jmp exit

Go1:

mov eax, [ScndSt]

mov ecx, [B]

mov edx, 0

div ecx

mov [RES], eax

mov [ThrdSt], edx

push [RES] ;output Y

push resStr

call [printf]

push point

call [printf]

mov ebx, 0

lp: ; output ostatka

mov eax, [ThrdSt]

mov ecx, [B]

imul eax, 10

mov edx, 0

div ecx

mov [ThrdSt], edx

push eax

push emptyStr

call [printf]

add ebx, 1

cmp ebx, 3

jne lp

jmp exit

BsmallerA: ;A> B Y = (a - B) / A - 3

;B / A (FstSt)

cmp [A], 0

jne Go ;if No - calculating

push infinity ; if Yes - exit the programm

call [printf]

jmp exit

Go:

mov ecx, [A]

sub ecx, [B]

mov [temp], ecx

mov eax, [temp]

mov ecx, [A]

mov edx, 0

div ecx

mov [FstSt], eax ;B mod A DO NOT TOUCH EDX !!!

mov [ScndSt], edx

; FrstSt -3

mov ecx, [FstSt]

sub ecx, 3

mov [RES], ecx

push [RES] ;output Y

push resStr

call [printf]

push point

call [printf]

mov ebx, 0

lp1:

mov eax, [ScndSt]

mov ecx, [A]

imul eax, 10

mov edx, 0

div ecx

mov [ScndSt], edx

push eax

push emptyStr

call [printf]

add ebx, 1

cmp ebx, 3

jne lp1

jmp exit

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 командам безумовної передачі управління JMP, передачі параметрів у підпрограми та логічним операціям “==”, “<”, “>” 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”