Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 8

З дисципліни «Системне програмування»

ВИКОНАВ:

студент 2 курсу ФІОТ

групи ІО-41

Логвинчук А. І.

ПЕРЕВІРИВ:

ст. вик. Порєв В. М.

Київ – 2016

**Тема:** Виконання операцій з плаваючою точкою та вивчення команд x87 FPU

**Мета:** Навчитися програмувати операції з плаваючою точкою на асемблері.

**Завдання:**

1. Створити у середовищі MS Visual Studio проект з ім’ям **Lab8**.

2. Написати вихідний текст програми згідно варіанту завдання. У проекті

мають бути головний файл **main8.asm** та інші модулі (за необхідності).

3. У цьому проекті кожний модуль може окремо компілюватися.

4. Скомпілювати вихідний текст і отримати виконуємий файл програми.

5. Перевірити роботу програми. Налагодити програму.

6. Отримати результати – файл числових значень згідно варіанту завдання.

7. Проаналізувати та прокоментувати результати, вихідний текст та

дизасембльований машинний код програми.

**Варіант завдання:**

Вар. 20.

**Програмний код**

**main8.asm:**

.586

.model flat, stdcall

include D:\DEV\masm32\include\kernel32.inc

include D:\DEV\masm32\include\user32.inc

include module.inc

include longop.inc

includelib D:\DEV\masm32\lib\kernel32.lib

includelib D:\DEV\masm32\lib\user32.lib

.data

Text db 100 dup(0), 0

Caption db "Лабораторна робота №8", 0

Iterations dd 5

ValA dd 3.0, 1.5, 2.1, 0.4, 3.8

ValB dd 0.5, 1.2, 3.4, 0.9, 1.1

Result dd 0.0

val dd 10111110110000000000000000000111b

.code

main:

push offset Result

push offset ValA

push offset ValB

push Iterations

call MIN\_LONGOP

push offset Text

push Result

call FloatToDec32

invoke MessageBoxA, 0, ADDR Text, ADDR Caption, 0

; TEST

push offset Text

push val

call FloatToDec32

invoke MessageBoxA, 0, ADDR Text, ADDR Caption, 0

invoke ExitProcess, 0

end main

**Функція:**

MIN\_LONGOP proc

push ebp

mov ebp, esp

mov esi, [ebp+20] ; адреса результату

mov eax, [ebp+16] ; адреса масиву А

mov ebx, [ebp+12] ; адреса масиву В

mov edx, [ebp+8] ; кількість повторень

fld dword ptr[eax] ; запис А0 в стек FPU

fadd dword ptr[ebx] ; додаємо В0, результат в ST(0)

fmul st(0), st(0) ; підносимо до квадрату

fstp min ; записуємо як мінімальне значення

mov ecx, 1 ; лічильник

@loop:

fld dword ptr[eax+4\*ecx]

fadd dword ptr[ebx+4\*ecx]

fmul st(0), st(0)

fstp dword ptr[result\_local]

mov edi, result\_local

cmp min, edi

jle @continue

mov min, edi

@continue:

inc ecx

cmp ecx, edx

jl @loop

mov edi, min

mov dword ptr[esi], edi

mov esp, ebp

pop ebp

ret 16

MIN\_LONGOP endp

**Процедура переведення числа з плавачою точкою:**

FloatToDec32 proc

push ebp

mov ebp, esp

mov edi, [ebp+12] ; рядок тексту

mov esi, [ebp+8] ; результат

mov eax, esi ; визначаємо знак результату

and eax, 80000000h

cmp eax, 0

je @no\_sign

mov byte ptr[edi], 45

inc edi

@no\_sign:

mov ecx, edi

mov eax, esi

and eax, 7F800000h ; виділяємо біти порядку

shr eax, 23 ; зсуваємо в біти в початок регістру, щоб отримати значення порядку

cmp eax, 0 ; якщо рівне - signed zero exception

jne @next

mov byte ptr[edi], 48

jmp @endproc

@next:

cmp eax, 0FFh ; якщо рівне - infinity exception

jne @next2

mov byte ptr[edi], 105

mov byte ptr[edi+1], 110

mov byte ptr[edi+2], 102

mov byte ptr[edi+3], 105 ; print "infinity"

mov byte ptr[edi+4], 110

mov byte ptr[edi+5], 105

mov byte ptr[edi+6], 116

mov byte ptr[edi+7], 121

jmp @endproc

@next2:

sub eax, 7Fh ; e-127

cmp eax, 0

jge @next3

mov byte ptr[edi], 48

inc ecx ; edi+1

mov ebx, esi

and ebx, 7FFFFFh ; вирізаємо біти мантиси

add ebx, 800000h ; додаємо цілу частину мантиси

mov edx, 0FFFFFFFFh ; -1

imul edx ; знаходимо абсолютне значення порядку

mov edx, ecx

mov ecx, eax

shr ebx, cl ; ділимо мантису на 2^E

mov ecx, edx

jmp @fraction

@next3:

jg @next4

mov byte ptr[edi], 49 ; 1,

inc ecx ; edi + 1

mov ebx, esi

and ebx, 7FFFFFh ; вирізаємо біти мантиси

jmp @fraction

@next4:

push ecx

mov ecx, 23

sub ecx, eax

push ecx

mov eax, esi

and eax, 7FFFFFh ; вирізаємо біти мантиси

add eax, 800000h ; додаємо цілу частину

xor ebx, ebx

mov ebx, 1

shl ebx, cl ; множимо на 2^(23-E)

mov edx, ebx

@mask: ; готуємо маску

inc cl

shl ebx, 1

add ebx, edx

cmp cl, 24

jne @mask

mov edx, eax

and edx, ebx

mov ebx, eax

sub ebx, edx

pop ecx

shr edx, cl ; виділяємо цілу частину

mov eax, 23

sub eax, ecx

mov ecx, eax

shl ebx, cl

mov eax, edx

pop ecx

push ebx

mov ebx, 10

@integer\_part: ; визначаємо цілу частину

xor edx, edx

div ebx

add edx, 48

mov byte ptr[ecx], dl

inc ecx

cmp eax, 0

jne @integer\_part

mov eax, ecx

dec eax

@swap: ; переставляємо цифри

xor edx, edx

mov dh, byte ptr[eax]

mov dl, byte ptr[edi]

mov byte ptr[eax], dl

mov byte ptr[edi], dh

inc edi

dec eax

cmp edi, eax

jl @swap

pop ebx

@fraction: ; визначаємо дробову чатстину

mov byte ptr[ecx], 44

inc ecx

mov ax, 6

@loop:

shl ebx, 1

mov edx, ebx

shl edx, 2

add ebx, edx

mov edx, ebx

and edx, 0FF800000h

shr edx, 23

add dl, 48

mov [ecx], dl

and ebx, 7FFFFFh

inc ecx

dec ax

cmp ax, 0

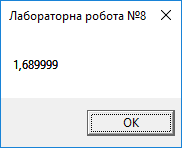
jne @loop

@endproc:

pop ebp

ret 8

FloatToDec32 endp

**Результати роботи програми:**

**Висновок**

Під час виконання роботи я навчився працювати з командами блоку x87 FPU, написав процедури для знаходження найменшого значення функції, заданої на двох масивах чисел, а також написав процедури переведення числа з плаваючою точкою у рядок тексту.