Міністерство освіти і науки України

Національний університет “Львівська політехніка”



**Звіт**

З лабораторної роботи №2

З дисципліни: «Системне програмування»

На тему: «Змішане прошрамування на мовах С та Ассемблер»

Виконав: ст.гр. КІ-38

Рудавський І.Я.

Прийняв: Козак Н.Б.

**Львів 2020**

**Мета:** оволодіти навиками створення програм, частини яких написані різними мовами програмування. Засвоїти правила взаємодії між програмними модулями різних.

**Завдання:**

Створити програму, яка реалізовує обчислення, заданого виразу, згідно варіанту.

Програма повинна складатися з кількох модулів, передача параметрів між якими здійснюється через стек.

Основний модуль – створюється мовою С.

Він повинен забезпечувати:

ввід даних з клавіатури;

виклик підпрограми обчислення виразу;

використати взаємовиклики С – ASM – С. Для виводу результатів обчислень забезпечити виклик стандартної функції printf() із модуля ASM.

Модуль безпосередніх обчислень – здійснює всі обчислення виразу і створюється мовою Assembler;

Відлагодити та протестувати програму. Результати роботи програми продемонструвати викладачу.

Скласти звіт про виконану роботу з приведенням тексту програми та коментарів до неї, а також результатів її роботи.

А, В, С, D, E, F - знакові операнди, довжиною в байтах, згідно з індексу,

значення К подано у 16-му форматі.

**Варіант завдання:**

****

**Завдання 1:**

**Код програми:**

#include <stdio.h>

extern "C" int calc(short C, int A, long D);

extern "C"

{

};

void main(void)

{

int nRes, nNumber2, nNumber3, nNumber4;

nNumber1 = 1;

printf("C = "); scanf\_s("%d", &nNumber2);

printf("A = "); scanf\_s("%d", &nNumber3);

printf("D = "); scanf\_s("%d", &nNumber4);

nRes = calc(nNumber2, nNumber3, nNumber4);

printf("Result is: %d", nRes);

}

func.asm

; використовується процесор з ядром і586 і вище

.586

.model flat

;flat - вказує на те, що всі сегменти розміщені фізично в одному сегменті

; Узгодження, які використовує лінкер для мови С передбачають:

; 1. аргументи функціям передаються через стек;

; 2. викликаюча функція очищає стек від параметрів;

; 3. аргументи завантажуються в стек перед викликом функції справа на ліво;

; 4. імена ідентифікаторів мають префікс "підкреслення" (\_);

; 5. імена ідентифікаторів є регістрозалежними (є різниця між великими і малими буквами);

; 6. функції повертають значення в регістропару eax:edx.

PUBLIC \_calc

.data

K EQU 4569600Fh

Temp1 dd 00h

Temp2 dd 00h

.code

\_calc PROC

; в стеку на даний момент є 32-ох розрядна ближня адреса пвернення з процедури

; збережемо в стек 32-ох розрядне значення ebp

push ebp

mov ebp,esp

xor eax,eax

xor ebx,ebx

xor ecx,ecx

xor edx,edx

mov eax,[ebp+16]

mov ebx,2

idiv bx

mov Temp1,eax

mov eax,[ebp+8]

mov bx,4

imul bx

mov Temp2,eax

mov eax,[ebp+12]

add eax,Temp2

sub eax,Temp1

add eax,K

pop ebp

ret

\_calc endp

end

**Завдання 2:**

#include <stdio.h>

extern "C" int calc(short C, int A, long D);

extern "C"

{

};

void main(void)

{

int nRes, nNumber2, nNumber3, nNumber4;

nNumber1 = 1;

printf("C = "); scanf\_s("%d", &nNumber2);

printf("A = "); scanf\_s("%d", &nNumber3);

printf("D = "); scanf\_s("%d", &nNumber4);

}

.586

.model flat

Printf proto arg1:ptr byte, printlist vararg

.data

K EQU 4569600Fh

Temp1 dd 00h

Temp2 dd 00h

Msg db ‘Result is: %d’, 0

.code

\_calc PROC

push ebp

mov ebp,esp

xor eax,eax

xor ebx,ebx

xor ecx,ecx

xor edx,edx

mov eax,[ebp+16]

mov ebx,2

idiv bx

mov Temp1,eax

mov eax,[ebp+8]

mov bx,4

imul bx

mov Temp2,eax

mov eax,[ebp+12]

add eax,Temp2

sub eax,Temp1

add eax,K

pop ebp

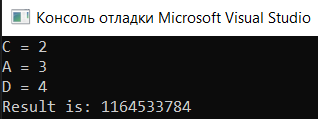
invoke printf, addr Msg, eax

ret

\_calc endp

end

**Результати виконання програми:**

****

**Перевірка результату:**

3+(4\*2)- 4/2 + 4569600F = 9 + 4569600F = 1164533784

**Висновок:** на даній лабораторній роботі я опанував роботу з програмною моделлю 32 розрядних процесорів Intel та оволодів навиками створення програм, використовуючи 32 розрядний асемблер.