МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

КАФЕДРА ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №5

з предмету “Системне програмування”

Виконала:  
Студентка 2 курсу  
ФІОТ гр. ІО-43  
Даніленко Наталія

Київ – 2016

**Лабораторна робота №5**

**Програмування множення чисел підвищеної розрядності**

**Мета:** Навчитися програмувати на асемблері множення чисел підвищеної розрядності, а також закріпити навички програмування власних процедур у модульному проекті.

**Завдання:**

1. Створити у середовищі MS Visual Studio проект з ім’ям Lab5.

2. Написати вихідний текст програми згідно варіанту завдання. У проекті мають бути три модуля на асемблері:

- головний модуль: файл main5.asm. Цей модуль створити та написати заново, частково використавши текст модуля main4.asm попередньої роботи №4;

- другий модуль: використати module попередніх робіт№3, 4;

- третій модуль: модуль longop попередньої роботи №4 доповнити новим кодом відповідно завданню.

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 – кодовані значення чисел згідно варіанту завдання.

7. Проаналізувати та прокоментувати результати, вихідний текст та дизасембльований машинний код програми.

**Варіант завдання:** 4

n = 30 + 4\*2 = 38

**Текст програми:**

.586

.model flat, stdcall

include D:\masm32\include\kernel32.inc

include D:\masm32\include\user32.inc

include longopp.inc

include modulee.inc

includelib D:\masm32\lib\kernel32.lib

includelib D:\masm32\lib\user32.lib

.const

caption1 db "n!", 0

caption2 db "(n!)^2", 0

.data

ct dd 38

fact\_result db 1, 31 dup (0)

buf db 32 dup (0)

fact\_sqr\_result db 64 dup (0)

fact\_text db 64 dup (0)

fact\_sqr\_text db 128 dup (0)

.code

main:

@cycle: ;цикл обчислення факторіала

mov esi, offset fact\_result

mov edi, offset buf

mov ecx, 8

rep movsd

mov eax, 0

mov edi, offset fact\_result

mov ecx, 8

rep stosd

push offset buf

push ct

push offset fact\_result

push 32

call Mul\_N\_32\_LONGOP

dec ct

cmp ct, 1

jg @cycle

push offset fact\_text

push offset fact\_result

push 256

call StrHex\_MY

invoke MessageBoxA, 0, ADDR fact\_text, ADDR caption1, 0

mov eax, 0

mov edi, offset fact\_sqr\_result

mov ecx, 16

stosd

push offset fact\_result

push offset fact\_result

push offset fact\_sqr\_result

push 32

call Mul\_N\_N\_LONGOP

push offset fact\_sqr\_text

push offset fact\_sqr\_result

push 512

call StrHex\_MY

invoke MessageBoxA, 0, ADDR fact\_sqr\_text, ADDR caption2, 0

invoke ExitProcess, 0

end main

.586

.model flat, c

.code

;процедура StrHex\_MY записує текст шістнадцятькового коду

;перший параметр - адреса буфера результату (рядка символів)

;другий параметр - адреса числа

;третій параметр - розрядність числа у бітах (має бути кратна 8)

StrHex\_MY proc

push ebp

mov ebp,esp

mov ecx, [ebp+8] ;кількість бітів числа

cmp ecx, 0

jle @exitp

shr ecx, 3 ;кількість байтів числа

mov esi, [ebp+12] ;адреса числа

mov ebx, [ebp+16] ;адреса буфера результату

@cycle:

mov dl, byte ptr[esi+ecx-1] ;байт числа - це дві hex-цифри

mov al, dl

shr al, 4 ;старша цифра

call HexSymbol\_MY

mov byte ptr[ebx], al

mov al, dl ;молодша цифра

call HexSymbol\_MY

mov byte ptr[ebx+1], al

mov eax, ecx

cmp eax, 4

jle @next

dec eax

and eax, 3 ;проміжок розділює групи по вісім цифр

cmp al, 0

jne @next

mov byte ptr[ebx+2], 32 ;код символа проміжку

inc ebx

@next:

add ebx, 2

dec ecx

jnz @cycle

mov byte ptr[ebx], 0 ;рядок закінчується нулем

@exitp:

pop ebp

ret 12

StrHex\_MY endp

;ця процедура обчислює код hex-цифри

;параметр - значення AL

;результат -> AL

HexSymbol\_MY proc

and al, 0Fh

add al, 48 ;так можна тільки для цифр 0-9

cmp al, 58

jl @exitp

add al, 7 ;для цифр A,B,C,D,E,F

@exitp:

ret

HexSymbol\_MY endp

End

.586

.model flat, c

.data

cote dd ?

.code

Mul\_N\_32\_LONGOP proc

local ct:DWORD

push ebp

mov ebp,esp

mov esi, [ebp+28] ;ESI = адреса множеного

mov ebx, [ebp+24] ;EBX = множник

mov edi, [ebp+20] ;EDI = адреса результату

mov ecx, [ebp+16] ;кількість байтів

shr ecx, 2 ;кількість подвійних слів

mov ct, ecx

xor ecx, ecx

cycle\_p: ;для парних доданків

mov eax, [esi+4\*ecx]

mul ebx

mov [edi+4\*ecx], eax

mov [edi+4\*ecx+4], edx

inc ecx

inc ecx

cmp ecx, ct

jl cycle\_p

xor ecx, ecx

inc ecx

cycle\_np: ;для непарних доданків

mov eax, [esi+4\*ecx]

mul ebx

add [edi+4\*ecx], eax

adc [edi+4\*ecx+4], edx

adc byte ptr [edi+4\*ecx+8], 0

inc ecx

inc ecx

cmp ecx, ct

jl cycle\_np

pop ebp ;відновлення стеку

ret 16

Mul\_N\_32\_LONGOP endp

Mul\_N\_N\_LONGOP proc

local ct:DWORD

push ebp

mov ebp,esp

mov esi, [ebp+28] ;ESI = адреса множеного

mov ebx, [ebp+24] ;EBX = адреса множника

mov edi, [ebp+20] ;EDI = адреса результату

mov ecx, [ebp+16] ; кількість байтів

shr ecx, 2

mov cote, ecx

xor ecx, ecx

cycle:

push ebx

push ecx

push edi

shl ecx, 2

add ebx, ecx

add edi, ecx

shr ecx, 2

xor ecx, ecx

cycle\_p: ;для парних доданків

mov eax, [esi+4\*ecx]

mul dword ptr [ebx]

add [edi+4\*ecx], eax

adc [edi+4\*ecx+4], edx

adc byte ptr [edi+4\*ecx+8], 0

inc ecx

inc ecx

cmp ecx, cote

jl cycle\_p

xor ecx, ecx

inc ecx

cycle\_np: ;для непарних доданків

mov eax, [esi+4\*ecx]

mul dword ptr [ebx]

add [edi+4\*ecx], eax

adc [edi+4\*ecx+4], edx

adc byte ptr [edi+4\*ecx+8], 0

inc ecx

inc ecx

cmp ecx, cote

jl cycle\_np

pop edi

pop ecx

pop ebx

inc ecx

cmp ecx, cote

jl cycle

pop ebp ;відновлення стеку

ret 16

Mul\_N\_N\_LONGOP endp

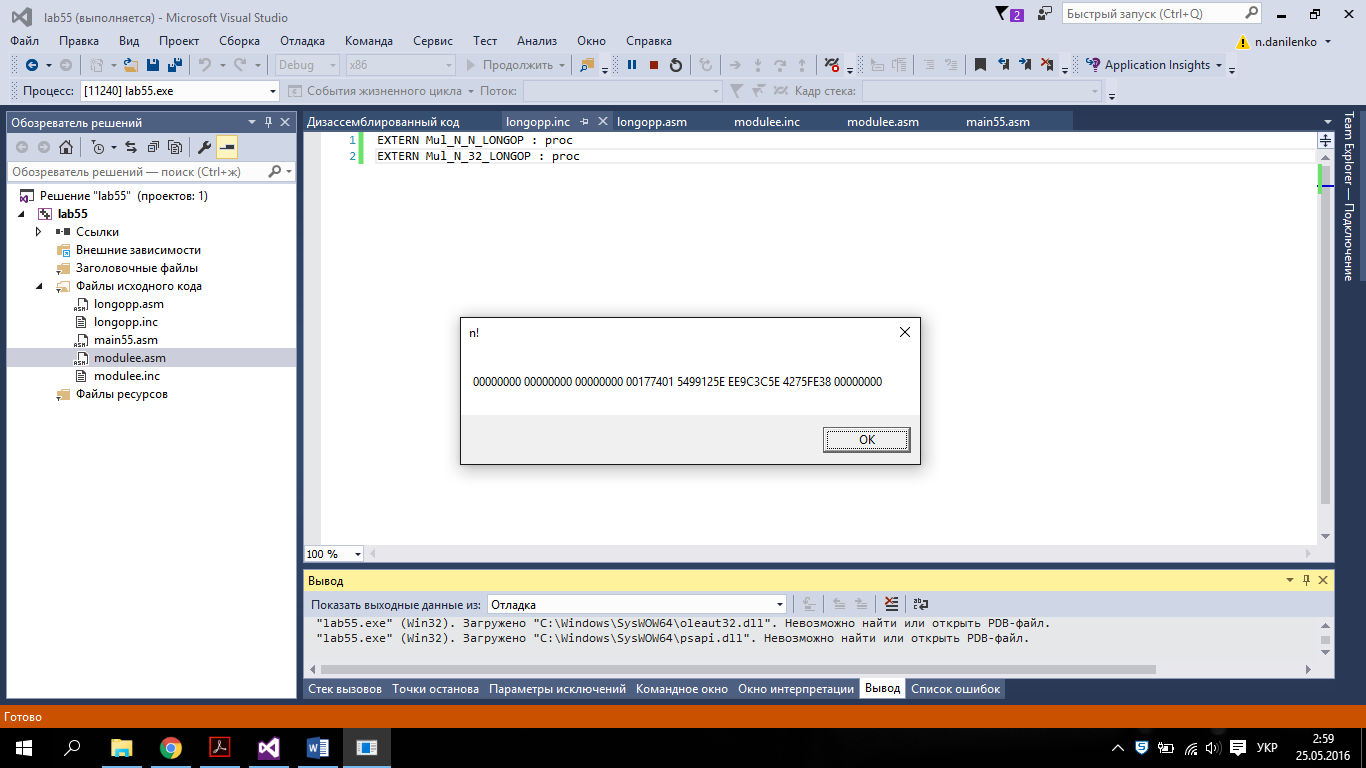
End

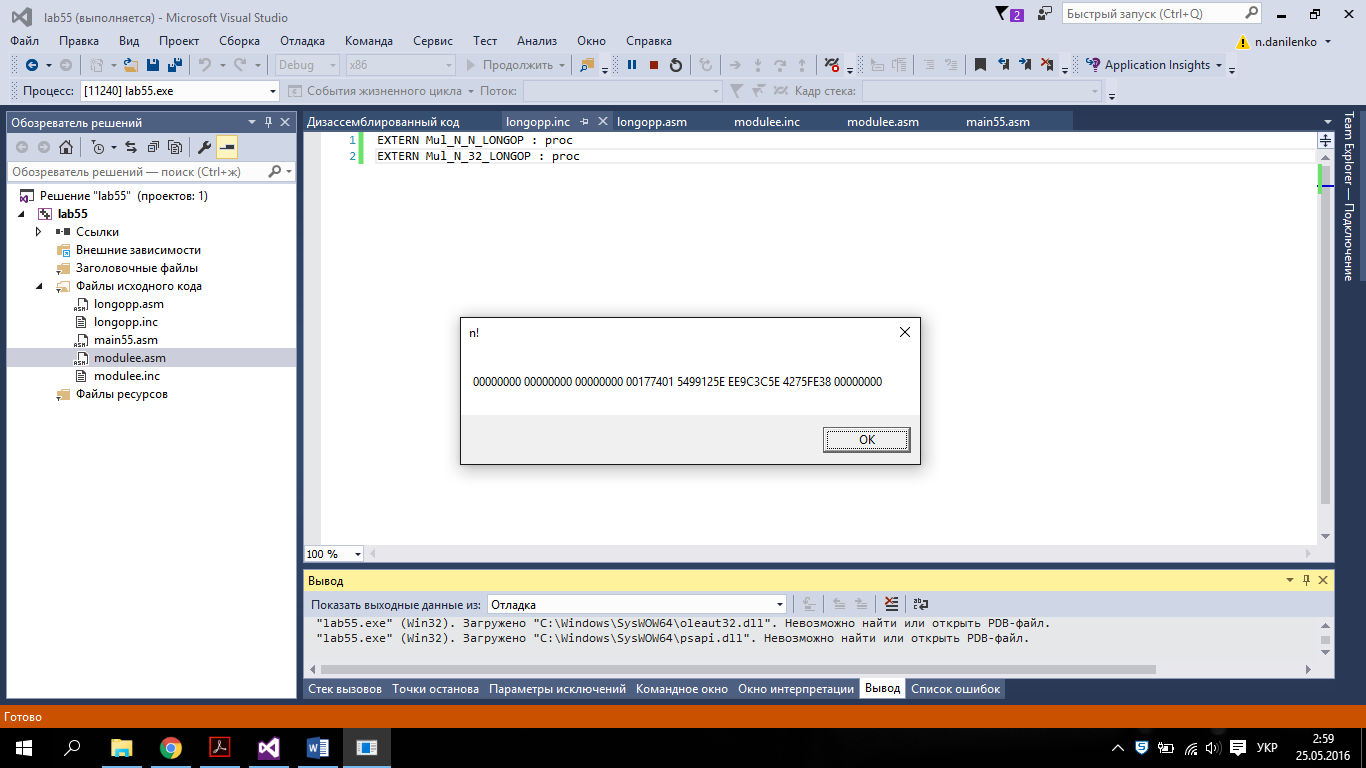
EXTERN StrHex\_MY : proc

EXTERN Mul\_N\_N\_LONGOP : proc

EXTERN Mul\_N\_32\_LONGOP : proc

**Результати виконання програми:**





**Аналіз результатів:**

Програма виконує обчислення n! та n!^2 за допомогою операцій множення на 32-бітове число (обчислення факторіалу) та множення на n-бітове число (піднесення факторіалу до квадрату).

**Висновок:**

Була розроблена програма, що обчислює n! та n!^2. Правильність виконання програми підтверджують результати наведені вище.