Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Системне програмування**

**Лабораторна робота №5**

«Програмування множення чисел підвищеної розрядності»

Виконав:

студент групи ІО-82

Шендріков Є. О.

Залікова № 8227

Перевірив Порєв В. М.

Київ – 2020

**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 дизасемблерний машинний код програми.

**Варіант завдання**

Для кожного студента своє значення n, яке визначається за формулою: **n = 30 + 2×H**, де H – це номер студента у журналі. Потрібно запрограмувати на асемблері:

* обчислення факторіалу ***n*!**
* обчислення квадрату факторіалу ***n*! × *n*!**
* обчислення тесту множення 111…1×111…1 розрядністю операндів N×N. Розрядність (N) обирається відповідно значенню ***n*!** згідно варіанту.
* обчислення тесту множення 111…1 × 10…01 розрядністю операндів N×N.
* обчислення тесту множення 111…1 × 111…1 розрядністю операндів N×32. Точні цілі значення результатів надати у шістнадцятковій системі числення. Оцінка може бути підвищена у таких випадках:
* розроблена процедура множення N×32, яка записує результат у операнд підвищеної розрядності, і ця процедура використана при обчисленні факторіалу ***n*!**
* розроблена 64-бітова програма, у якій множення 64-бітовими порціями, зокрема, замість процедури розрядності N×32 – процедура N×64

**n = 30 + 2 \* 25 = 80**

**Текст програми**

*main5.asm*

.model flat, stdcall

include \masm32\include\kernel32.inc

include \masm32\include\user32.inc

include module.inc

include longop.inc

includelib \masm32\lib\kernel32.lib

includelib \masm32\lib\user32.lib

option casemap :none

.data

Capt db "80!" ,0

Capt1 db "80! x 80!" ,0

Capttest1 db "test N\*N (111..1\*111..1)" ,0

Capttest2 db "test N\*32 (111..1\*111..1)" ,0

Capttest3 db "test N\*N (111..1\*11..100..0)" ,0

textBuf dd 100 dup(?)

textBuf1 dd 100 dup(?)

textBuftest1 dd 100 dup(?)

textBuftest2 dd 100 dup(?)

textBuftest3 dd 100 dup(?)

var dd 25 dup(0); 25\*32

bigger\_var dd 50 dup(0)

x dd 80

test1 dd 25 dup(4294967295)

test1res dd 50 dup(0)

test2 dd 25 dup(4294967295)

test31 dd 25 dup(4294967295)

test32 dd 1h, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 80000000h

test3res dd 50 dup(0)

.code

main:

mov [var], 1

@factorial:

push offset var

push x

call Mul\_Nx32\_LONGOP

dec x

jne @factorial

push offset textBuf

push offset var

push 400

call StrHex\_MY

invoke MessageBoxA, 0, ADDR textBuf, ADDR Capt, 0

push offset var

push offset var

push offset bigger\_var

call Mul\_NxN\_LONGOP

push offset textBuf1

push offset bigger\_var

push 796

call StrHex\_MY

invoke MessageBoxA, 0, ADDR textBuf1, ADDR Capt1, 0

push offset test1

push offset test1

push offset test1res

call Mul\_NxN\_LONGOP

push offset textBuftest1

push offset test1res

push 1600

call StrHex\_MY

invoke MessageBoxA, 0, ADDR textBuftest1, ADDR Capttest1, 0

mov dword ptr [test2 + 16] , 0

push offset test2

push 4294967295

call Mul\_Nx32\_LONGOP

push offset textBuftest2

push offset test2

push 160

call StrHex\_MY

invoke MessageBoxA, 0, ADDR textBuftest2, ADDR Capttest2, 0

push offset test31

push offset test32

push offset test3res

call Mul\_NxN\_LONGOP

push offset textBuftest3

push offset test3res

push 1600

call StrHex\_MY

invoke MessageBoxA, 0, ADDR textBuftest3, ADDR Capttest3, 0

invoke ExitProcess,0

end main

*longop.asm*

.586

.model flat, c

.data

x dd 1

n dd 0

a dd 25

b dd 25

.code

Mul\_Nx32\_LONGOP proc

push ebp

mov ebp, esp

mov edi, [ebp + 12]

mov ebx, [ebp + 8]

mov x, ebx

mov n, 25; кількість двійних слів яка потрібна

xor ebx, ebx; очистити регістри

xor ecx, ecx

; множимо кожні групи по 32 біта великого числа на 32 біта

@multiply32:

mov eax, dword ptr[edi + ecx]

mul x

mov dword ptr[edi + ecx], eax

clc

adc dword ptr[edi + ecx], ebx

mov ebx, edx

add ecx, 4

dec n

jnz @multiply32

pop ebp

ret 8

Mul\_Nx32\_LONGOP endp

Mul\_NxN\_LONGOP proc

push ebp

mov ebp, esp

mov esi, dword ptr[ebp + 16]; перше н-розрядне число

mov edi, dword ptr[ebp + 12]; друге

mov ebx, dword ptr[ebp + 8]; результат

xor ecx, ecx

xor edx, edx

mov b, 25

@main\_cycle:

mov eax, dword ptr[esi + edx]

push edx

push ebx

mov ebx, ecx

sub ebx, edx

mul dword ptr[edi + ebx]

pop ebx

add dword ptr[ebx + ecx], eax

adc dword ptr[ebx + ecx + 4], edx

jnc @first

xor eax, eax

mov eax, ecx

@second:

add eax, 4

add dword ptr[ebx + eax + 4], 1

jc @second

@first:

; first - немає переносу, коли більше ніж 2 двійних слова при сумуванні по блоках

; second - є перенос

pop edx

add ecx, 4

dec a

jnz @main\_cycle

add edx, 4

mov ecx, edx

mov a, 25

dec b

jnz @main\_cycle

pop ebp

ret 12

Mul\_NxN\_LONGOP endp

end

*module.asm*

.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

;ця процедура записує 8 символів HEX коду числа

;перший параметр - 32-бітове число

;другий параметр - адреса буфера тексту

DwordToStrHex proc

push ebp

mov ebp,esp

mov ebx,[ebp+8] ;другий параметр

mov edx,[ebp+12] ;перший параметр

xor eax,eax

mov edi,7

@next:

mov al,dl

and al,0Fh ;виділяємо одну шістнадцяткову цифру

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

cmp ax,58

jl @store

add ax,7 ;для цифр A,B,C,D,E,F

@store:

mov [ebx+edi],al

shr edx,4

dec edi

cmp edi,0

jge @next

pop ebp

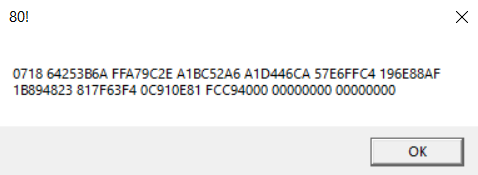
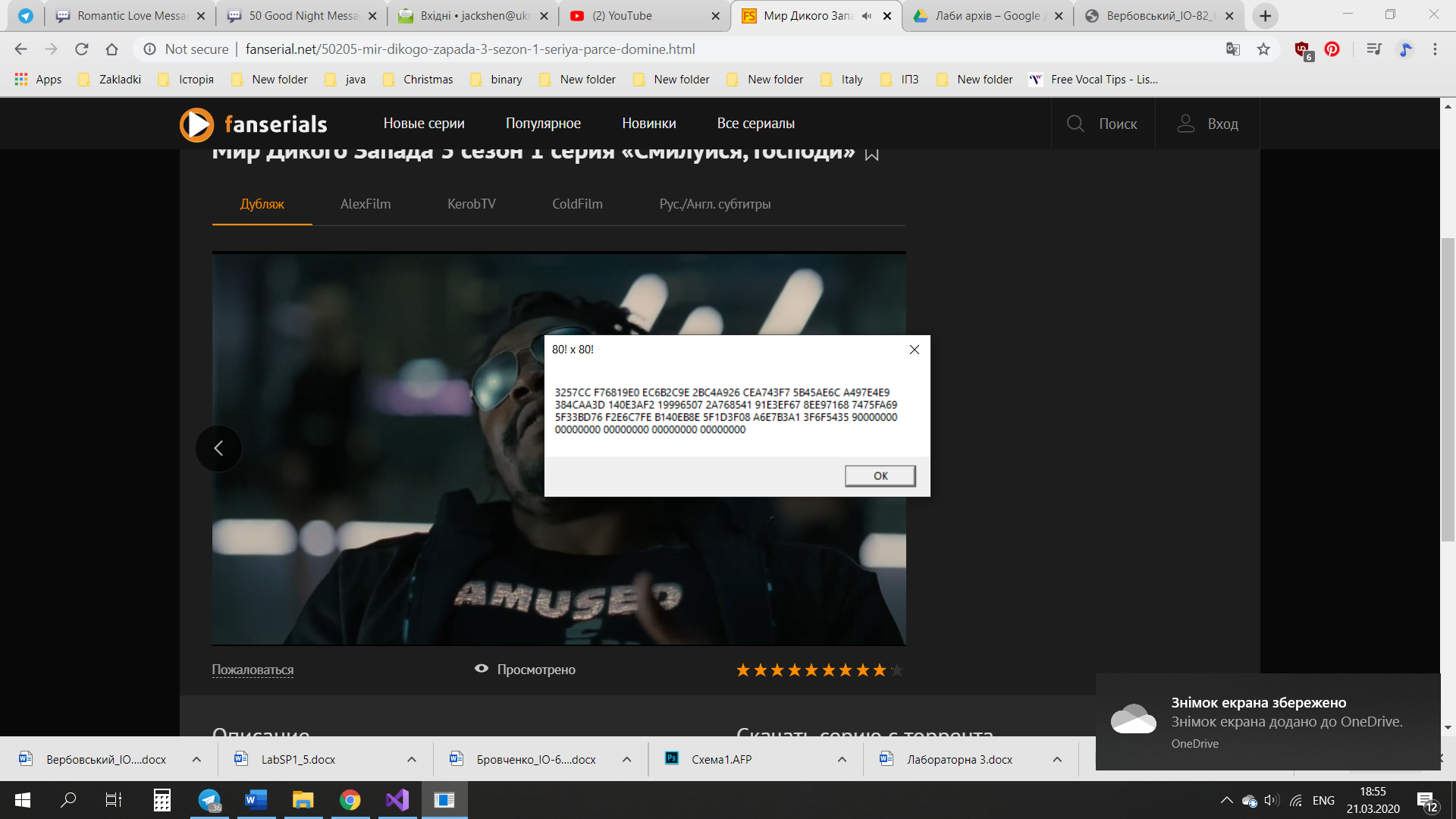
ret 8

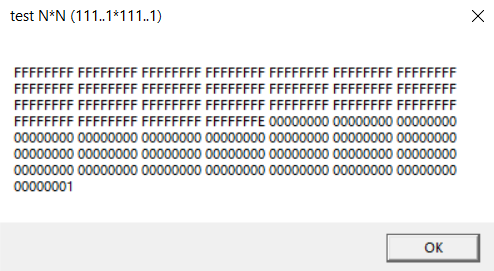
DwordToStrHex endp

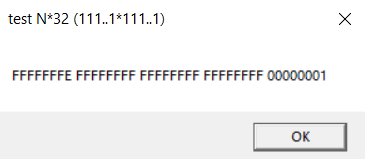
end

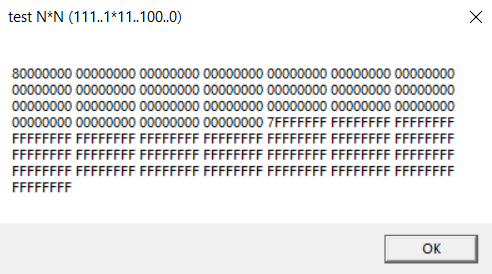
**Аналіз результатів**

Дана програма виконує операції множення з числами підвищеної розрядності.

**Результати роботи програми**







**Висновок**

Під час виконання лабораторної роботи були покращені навички написання власних модулів, роботи з циклами, а також були закріпленні основні навички в операціях множення чисел з підвищеною розрядністю.