Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Системне програмування**

**Лабораторна робота №4**

«Програмування множення чисел

підвищеної розрядності»

Виконав:

студент групи ІО-24

Довгань М. С.

Перевірив:

Порєв В. М.

Київ - 2024

**Тема:** Програмування множення чисел підвищеної розрядності.

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

**Завдання:**

1. Створити у середовищі MS Visual Studio проект з ім’ям Lab4.

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

- головний модуль: файл main4.asm. Цей модуль створити та написати заново, частково використавши текст модуля main3.asm попередньої роботи №3;

- другий модуль: використати module попередніх робіт №2, 3;

- третій модуль: модуль longop попередньої роботи №3 доповнити новим кодом відповідно завданню.

3. У цьому проекті кожний модуль може окремо компілюватися.

4. Скомпілювати вихідний текст і отримати виконуваний файл програми.

5. Перевірити роботу програми. Налагодити програму.

6. Отримати результати – кодовані значення чисел згідно варіанту завдання.

7. Проаналізувати та прокоментувати вихідний текст та результати.

**Індивідуальний варіант завдання:**

Обчислення значення n, яке визначається за формулою: n = 30 + 2 × H, де H - номер у журналі. Тобто, n = 30 + 2 × 9 = 48. Отже, мені необхідно запрограмувати на асемблері:

* обчислення факторіалу 48!;
* обчислення квадрату факторіалу 48! × 48!;
* обчислення тесту множення двійкових кодів 111…1 × 111…1 розрядністю операндів N × N. Розрядність кожного операнду (N) має бути тою самою, яка була обрана для представлення означення 48!. Шістнадцяткові коди таких операндів мають вигляд FFF…F × FFF…F.
* Обчислення тесту множення двійкових кодів 111…1 × 111…1 розрядністю операндів N × 32.
* Обчислення тесту множення двійкових кодів 0101…0101 × 10…01 розрядністю операндів N × N. Шістнадцятковий код таких операндів має вигляд 55…5 × 80…01.

**Виконання завдання:**

**Роздруківка коду програми:**

module.inc:

EXTERN StrHex\_MY : proc

longop.inc:

EXTERN Factorial : proc

EXTERN Mul\_NN\_LONGOP : proc

EXTERN Mul\_N32\_LONGOP : proc

module,asm:

.586

.model flat, c

.code

;процедура StrHex\_MY запису текст шістнадцяткового 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

longop.asm:

.586

.model flat, c

.data

count dd 0h

factorialValue dd 1h

resFactorial dd 18 dup(0h)

oldResFactorial dd 18 dup(0h)

.code

Factorial proc

push ebp

mov ebp, esp

mov ecx, 1

add oldResFactorial, 1

clc

@cycle:

cmp ecx, dword ptr[ebp+12]

jg @exit

mov factorialValue, ecx

push offset oldResFactorial

push factorialValue

push 18

push offset resFactorial

call Mul\_N32\_LONGOP

mov ecx, 0

@copy:

cmp ecx, 9

je @next

mov eax, [resFactorial+4\*ecx]

mov [oldResFactorial+4\*ecx], eax

mov [resFactorial+4\*ecx], 0

inc ecx

jmp @copy

@next:

mov ecx, factorialValue

inc ecx

jmp @cycle

@exit:

mov edi, [ebp+8]

mov ecx, 0

@cycle2:

cmp ecx, 9

je @done

mov eax, [oldResFactorial+4\*ecx]

mov [edi+4\*ecx], eax

mov [oldResFactorial+4\*ecx], 0h

inc ecx

jmp @cycle2

@done:

pop ebp

ret 8

Factorial endp

Mul\_N32\_LONGOP proc

push ebp

mov ebp, esp

mov edi, [ebp+8]

mov ebx, [ebp+16]

mov esi, [ebp+20]

xor ecx, ecx

xor eax, eax

xor edx, edx

clc

@cycle:

cmp ecx, dword ptr[ebp+12]

je @exit

mov eax, dword ptr[esi+4\*ecx]

mul ebx

add [edi+4\*ecx], eax

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

inc ecx

jmp @cycle

@exit:

xor ecx, ecx

pop ebp

ret 16

Mul\_N32\_LONGOP endp

Mul\_NN\_LONGOP proc

push ebp

mov ebp, esp

mov edi, [ebp+8]

mov ecx, 0

mov count, 0

xor edx, edx

clc

@maincycle:

mov eax, count

cmp eax, dword ptr[ebp+12]

je @exit

mov esi, [ebp+16]

mov ebx, dword ptr[esi+4\*eax]

@secondcycle:

cmp ecx, dword ptr[ebp+12]

je @Done

mov esi, [ebp+20]

mov eax, dword ptr[esi+4\*ecx]

mul ebx

add ecx, count

clc

add [edi+4\*ecx], eax

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

jnc @next

mov eax, ecx

@cf:

inc eax

add dword ptr[edi+4\*eax+4], 1

jc @cf

@next:

sub ecx, count

inc ecx

jmp @secondcycle

@Done:

xor ecx, ecx

add count, 1

jmp @maincycle

@exit:

mov count, 0

pop ebp

ret 16

Mul\_NN\_LONGOP endp

end

main4.asm:

.586

.model flat, stdcall

include module.inc

include longop.inc

include \masm32\include\user32.inc

include \masm32\include\kernel32.inc

.data

mainWindowTitle db "Лабораторна робота №4", 0

mainWindowText db "Здоровенькі були!", 13, 10, 13, 10,

"Лабораторну роботу виконав:", 13, 10,

"студент групи ІО-24,", 13, 10,

"Довгань М. С.", 0

computeFactorialN db "Обчислення факторіалу (n!):", 0

computeFactorialNN db "Обчислення квадрату факторіалу (n! \* n!):", 0

computeFirstMultiplicationNN db "Обчислення множення розрядністю N \* N:", 0

computeMultiplicationN32 db "Обчислення множення розрядністю N \* 32:", 0

computeSecondMultiplicationNN db "Обчислення множення розрядністю N \* N:", 0

lastWindowTitle db "Програма завершила роботу", 0

lastWindowText db "Дякую за увагу!", 0

firstFactorialN dd 48

resFirstFactorialN dd 9 dup(0h), 0

secondFactorialN dd 48, 0

resSecondFactorial1N dd 9 dup(0h)

resSecondFactorial2N dd 9 dup(0h)

resSecondFactorial dd 18 dup(0h)

firstValueNN dd 9 dup(0FFFFFFFFh)

secondValueNN dd 9 dup(0FFFFFFFFh)

firstResNN dd 18 dup(0h)

firstValueN32 dd 9 dup(0FFFFFFFFh)

secondValueN32 dd 0FFFFFFFFh

resN32 dd 10 dup(0h)

thirdValueNN dd 9 dup(55555555h)

fourthValueNN dd 00000001h, 00000000h, 00000000h, 00000000h, 00000000h,

00000000h, 00000000h, 00000000h, 80000000h

secondResNN dd 18 dup(0h)

textbufFirstFactorial db 9 dup(?)

textbufSecondFactorial db 18 dup(?)

textbufFirstNN db 18 dup(?)

textbufN32 db 10 dup(?)

textbufSecondNN db 18 dup(?)

.code

main4:

invoke MessageBoxA, 0, ADDR mainWindowText, ADDR mainWindowTitle, 0

;---------- Обчислення факторіалу n! ----------

push firstFactorialN

push offset resFirstFactorialN

call Factorial

push offset textbufFirstFactorial

push offset resFirstFactorialN

push 288

call StrHex\_MY

invoke MessageBoxA, 0, ADDR textbufFirstFactorial, ADDR computeFactorialN, 0

;---------- Обчислення квадрату факторіалу n! \* n! ----------

push secondFactorialN

push offset resSecondFactorial1N

call Factorial

push secondFactorialN

push offset resSecondFactorial2N

call Factorial

push offset resSecondFactorial1N

push offset resSecondFactorial2N

push 9

push offset resSecondFactorial

call Mul\_NN\_LONGOP

push offset textbufSecondFactorial

push offset resSecondFactorial

push 576

call StrHex\_MY

invoke MessageBoxA, 0, ADDR textbufSecondFactorial, ADDR computeFactorialNN, 0

;---------- Обчислення множення двійкових кодів розрядністю операндів N \* N ----------

push offset firstValueNN

push offset secondValueNN

push 9

push offset firstResNN

call Mul\_NN\_LONGOP

push offset textbufFirstNN

push offset firstResNN

push 576

call StrHex\_MY

invoke MessageBoxA, 0, ADDR textbufFirstNN, ADDR computeFirstMultiplicationNN, 0

;----------- Обчислення множення двійкових кодів розрядністю операндів N \* 32 ----------

push offset firstValueN32

push secondValueN32

push 9

push offset resN32

call Mul\_N32\_LONGOP

push offset textbufN32

push offset resN32

push 320

call StrHex\_MY

invoke MessageBoxA, 0, ADDR textbufN32, ADDR computeMultiplicationN32, 0

;---------- Обчислення множення двійкових кодів розрядністю операндів N \* N ----------

push offset thirdValueNN

push offset fourthValueNN

push 9

push offset secondResNN

call Mul\_NN\_LONGOP

push offset textbufSecondNN

push offset secondResNN

push 576

call StrHex\_MY

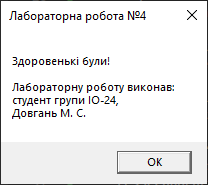
invoke MessageBoxA, 0, ADDR textbufSecondNN, ADDR computeSecondMultiplicationNN, 0

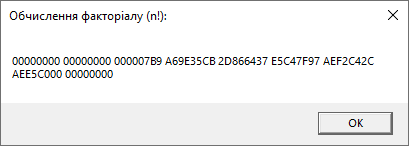
invoke MessageBoxA, 0, ADDR lastWindowText, ADDR lastWindowTitle, 0

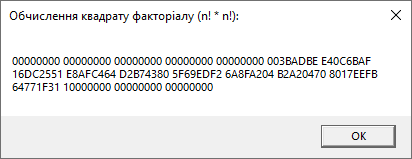
invoke ExitProcess, 0

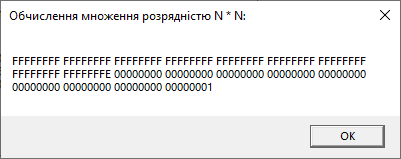
end main4

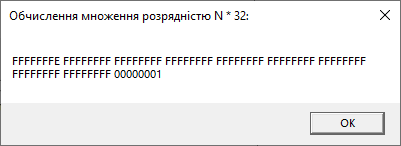
**Результати виконання програми:**

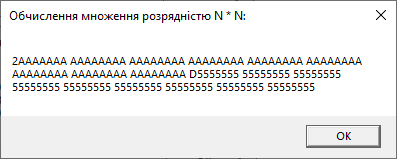


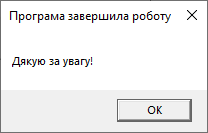












**Аналіз виконання програми:**

Створена мною програма виконує завдання лабораторної роботи, відповідно до мого індивідуального варіанту завдання та обчисленого значення n. Усі значення, включно із n заносяться до програми у форматі dd (define double word), тобто подвійне слово, по чотири байти. При початковому запуску програми для користувача з’являється стартове вікно-привітання, із привітанням та вказаною інформацією щодо лабораторної роботи та її автора. Потім програма обчислює факторіал заданого нами значення n використовуючи дві процедури - Factorial та StrHex\_My і виводить його результат у нове вікно. За схожим принципом виконується обчислення квадрату факторіалу n - ми викликаємо процеси Factorial, NN\_Longop та StrHex\_My і виводимо вже нове значення у нове вікно, яке підписане, для більшої зручності користувача. Далі у нас виконується обчислення двійкових кодів розрядністю операндів N \* N викликаючи ті ж процеси, із виводом результату в окреме вікно та обчислення множення двійкових кодів. У кінці програма видає останнє вікно, в якому йдеться, що вона видала всі значення та завершує роботу.

**Висновок:** під час виконання даної лабораторної роботи я навчився програмувати на асемблері множення підвищеної розрядності, а також закріпив навички програмування власних процедур у модульному проекті.