Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Системне програмування**

**Лабораторна робота №5**

«Програмування побітових операцій»

Виконав:

студент групи ІО-24

Довгань М. С.

Перевірив:

Порєв В. М.

Київ - 2024

**Тема:** Програмування побітових операцій.

**Мета:** навчитися програмувати на Асемблері побітові операції, вивчити основні команди обробки бітів.

**Завдання:**

1. Створити у середовищі Microsoft 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;

- другий модуль: використати **module** попередніх робіт;

- 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 виконуваний 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.

**Індивідуальний варіант завдання:**

Потрібно запрограмувати процедуру, яка обробляє дані підвищеної розрядності. У процедури мають бути такі параметри: адреса джерела даних, адреса результату, розрядність, а також параметри N, M. Параметри N та M повинні бути довільними цілими.

Зсув M молодших бітів на N позицій вправо. Решта бітів нерухомі. Розрядність (біт) 672.

**Виконання завдання:**

**Роздруківка коду програми:**

module.inc:

EXTERN StrHex\_MY : proc

longop.inc:

EXTERN Shr\_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

@maincycle:

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 @maincycle

@exit:

mov edi, [ebp+8]

mov ecx, 0

@secondcycle:

cmp ecx, 9

je @done

mov eax, [oldResFactorial+4\*ecx]

mov [edi+4\*ecx], eax

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

inc ecx

jmp @secondcycle

@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

ReadOneBit proc

push ebp

mov ebp, esp

xor ah, ah

xor cl, cl

push ecx

push edx

push edi

xor ecx, ecx

xor ebx, ebx

mov ebx, [ebp+8]

mov edi, [ebp+12]

mov ecx, ebx

shr ebx, 3

and ecx, 07h

mov al, 1

shl al, cl

mov ah, byte ptr [edi+ebx]

and ah, al

shl ebx, 3

pop edi

pop edx

pop ecx

pop ebp

ret 8

ReadOneBit endp

OneBitRightShift proc

push ebp

mov ebp, esp

xor ah, ah

push edx

push ecx

mov edx, [ebp+12]

mov ecx, [ebp+8]

push edx

push ecx

call ReadOneBit

shr al, 1

shr ebx, 3

cmp ah, 0

jz @setzero

or byte ptr [edx+ebx], al

jmp @exit

@setzero:

not al

and byte ptr [edx+ebx], al

@exit:

shl ebx, 3

xor eax, eax

pop ecx

pop edx

pop ebp

ret 8

OneBitRightShift endp

Shr\_LONGOP proc

push ebp

mov ebp, esp

xor ecx, ecx

xor edx, edx

push esi

mov esi, [ebp+16]

mov edx, [ebp+12]

mov edi, [ebp+8]

@cycle:

cmp edx, 0

jle @exit

xor ecx, ecx

@inner\_cycle:

push edi

push ecx

call OneBitRightShift

inc ecx

cmp ecx, [ebp+16]

jl @inner\_cycle

cmp esi, 0

jl @exit

push edi

push esi

call ReadOneBit

shr ebx, 3

not al

and byte ptr[edi+ebx], al

shl ebx, 3

dec esi

dec edx

jmp @cycle

@exit:

pop esi

pop ebp

ret 16

Shr\_LONGOP endp

end

main5.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 №5", 0

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

"Лабораторну роботу виконав:", 13, 10,

"студент групи ІО-24,", 13, 10,

"Довгань М. С.", 0

Value dd 21 dup(0FFFFFFFFh)

M dd 138

N dd 138

windowTitle db "Зсув M молодших бітів на N позицій вправо", 0

lastWindowTitle db "Програма завершила роботу", 0

lastWindowText db "Дякую за увагу!", 0

textbuf dd 1 dup(?)

.code

main5:

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

push M

push N

push offset Value

call Shr\_LONGOP

push offset textbuf

push offset Value

push 672

call StrHex\_MY

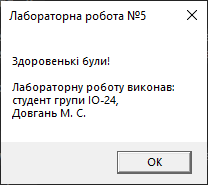
invoke MessageBoxA, 0, ADDR textbuf, ADDR windowTitle, 0

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

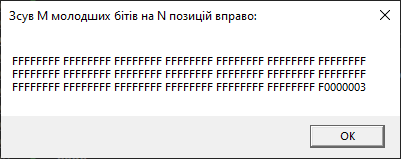
invoke ExitProcess, 0

end main5

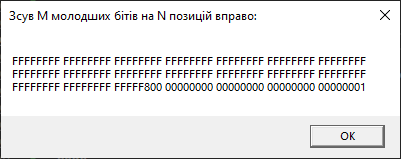
**Результати виконання програми:**

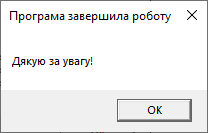


Для значень M = 27, N = 26:



Для значень M = 138, N = 138:





**Аналіз виконання програми:**

Створена мною програма виконує завдання лабораторної роботи, відповідно до мого індивідуального варіанту завдання. Значення M та N заносяться безпосередньо до програми у файлі main5.asm (розділ .data) у вигляді dd (define double word), тобто 4 байти. При початковому запуску програми запускається стартове вікно-привітання із користувачем, в якому міститься привітання, номер лабораторної роботи та її автора. Потім програма виконує зсув вказаного користувачем значення M молодших бітів на також вказане значення користувачем N позицій вправо, завдяки процесу Shr\_Longop, при цьому решта бітів залишаються нерухомими, далі переводимо отримане значення в шістнадцяткову систему числення за допомогою процесу StrHex\_My. Даний результат записується та виводиться в окремому вікні, заголовок якого містить у собі назву проведеної операції користувачем, а виведений текст є отриманим результатом. Після цього користувачу виводиться останнє вікно, яке повідомляє користувача, що всі значення були обчислені та програма завершує роботу.

**Висновок:** під час виконання даної лабораторної роботи я ознайомився та вивчив основні команди обробки бітів та навчився програмувати на Асемблері побітові операції.