Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Системне програмування

Лабораторна робота №6

«Програмування операцій ділення чисел»

Виконав:
студент групи ІО-24
Довгань М. С.

Перевірив:
Порєв В. М.

Київ - 2024

Тема: Програмування операцій ділення чисел.

Мета: навчитися програмувати на Асемблері ділення чисел та перетворення з двійкової у десяткову систему числення.

Завдання:

1. Створити у середовищі MS Visual Studio проект з ім'ям **Lab6**.
2. Написати вихідний текст програми згідно варіанту завдання. У проекті мають бути три модуля на асемблері:
   - головний модуль: файл **main6.asm**. Цей модуль створити та написати заново, частково використавши текст модуля main5.asm попередньої роботи №5;
   - другий модуль: модуль **module** попередньої роботи №5;
   - третій модуль: модуль **longop** попередньої роботи №5.
3. Додати у модулі процедури, які потрібні для виконання завдання. Обгрунтувати розподіл процедур по модулям.
4. У цьому проекті кожний модуль може окремо компілюватися.
5. Скомпілювати вихідний текст і отримати виконуваний файл програми.
6. Перевірити роботу програми. Налагодити програму.
7. Отримати результати – кодовані значення чисел згідно варіанту завдання.
8. Проаналізувати та прокоментувати результати та вихідний текст.

Індивідуальний варіант завдання:

1. Потрібно запрограмувати на асемблері вивід значення факторіалу n! у десятковому коді. Використати програмний код обчислення факторіалу лабораторної роботи №4. $n = 30 + 2 \times H$, де H – це номер студента у журналі. Значить, моє значення n наступне: $n = 30 + 2 \times 9 = 48$.
2. Перетворення у десятковий код запрограмувати діленням числа підвищеної розрядності на 10. Ділення на 10 виконати двома способами – діленням "у стовпчик" та діленням груп бітів. Ділення групами бітів запрограмувати або як 8-бітове, або як 16-бітове, або як 32-бітове відповідно до варіанту. Варіант ділення групами бітів обирається відповідно залишку ділення номеру студента у журналі (H) на 3, тобто: $9 \% 3 = 0$, отже, необхідно ділити групами по 8 бітів.
3. Програмний код ділення "у стовпчик" та ділення групами бітів оформити у вигляді процедур. Процедури повинні містити такі параметри: адреса ділимого, розрядність ділимого, значення дільника (B).

4. Запрограмувати на асемблері також обчислення формули, яку вибрати з таблиці, наведеної нижче. Номер варіанту формули згідно номеру в журналі. Узагальнено формула має вигляд y = F(x, m).Значення x, y повинні бути 32-бітовими цілими зі знаком. Значення m – ціле без знаку. Результат (y) повинен записуватися у регістр EAX. Значення цього результату має виводитися у десятковому коді у відповідному вікні виводу.
5. При програмування виражень для формул відповідно обраному варіанту проаналізувати можливість різної послідовності виконання операцій і обрати таку послідовність, яка забезпечує найвищу точність результату.
Моя індивідуальна формула:

9	$y = \frac{5}{x+1} 2^m$

Виконання завдання:

Роздруківка коду програми:

module.inc:

```
EXTERN Func : proc
```

longop.inc:

```
EXTERN Div_Column_LONGOP : proc
EXTERN Calc_N32_LONGOP : proc
EXTERN Div_LONGOP : proc
EXTERN Str_Dec_MY : proc
```

module.asm:

```
.586
.model flat, c

.code
Func proc
    push ebp
    mov ebp, esp
    mov ecx, [ebp+8] ; m
    mov ebx, [ebp+12] ; x
    add ebx, 1 ; додаємо 1 до x
    mov eax, 5

    @divide:
        xor edx, edx
        idiv ebx
```

```
        neg ebx
        jmp @multiply

    @multiply:
        shl eax, 1
        dec ecx
        cmp ecx, 0
        je @final
        jmp @multiply

    @final:
        pop ebp
        ret

Func endp

end
```

longop.asm:

```
.586
.model flat, c

.data
    count dd 0h
    x dd 1
    n dd 0
    counter_in dd 5
    counter_out dd 5

    num10 db 10
    inner dd 0
    num7 db 7
    minn db 0
    spacee db 3
    fpart db 0

.code
Div_Column_LONGOP proc
    xor ebx, ebx
    xor ecx, ecx
    dec edx
    cmp byte ptr[esi + edx], 0
    jnz @cycleout
    inc bl

    @cycleout:
        mov ch, byte ptr[esi + edx]

    @cycleinner:
        shl cl, 1
```

```
        shl bh, 1
        shl ch, 1
        jnc @zero
        inc bh

    @zero:
        cmp bh, num10
        jc @less
        inc cl
        sub bh, num10

    @less:
        inc inner
        cmp inner, 8
        jnz @cycleinner
        mov byte ptr[esi + edx], cl
        mov inner, 0
        sub edx, 1
        jnc @cycleout
        ret

Div_Column_LONGOP endp

Div_LONGOP proc
    push ebp
    mov ebp, esp
    mov esi, [ebp + 20]
    mov edi, [ebp + 16]
    mov ebx, [ebp + 12]
    mov eax, [ebp + 8]
    mov x, eax
    push ebx
    xor edx, edx
    mov ecx, x
    dec x
    mov ebx, x

    @cycle:
        push ecx
        mov ecx, 10
        mov eax, dword ptr[esi + 4*ebx]
        div ecx
        mov fpart, dl
        mov dword ptr[edi + 4*ebx], eax
        dec ebx
        pop ecx
        dec ecx
        jnz @cycle
        pop ebx
        mov al, fpart
```

```
        mov byte ptr[ebx], al
        pop ebp
        ret 16

Div_LONGOP endp

Str_Dec_MY proc
    push ebp
    mov ebp,esp
    mov edx, [ebp+8]
    shr edx, 3
    mov esi, [ebp+12]
    mov edi, [ebp+16]
    mov eax, edx
    shl eax, 2
    mov cl, byte ptr[esi + edx - 1]
    and cl, 128
    cmp cl, 128
    jnz @plus
    mov minn, 1
    push edx

    @minus:
        not byte ptr[esi + edx - 1]
        sub edx, 1
        jnz @minus
        inc byte ptr[esi + edx]
        pop edx

    @plus:
    @cycle:
        push edx
        call Div_Column_LONGOP

        pop edx
        add bh, 48
        mov byte ptr[edi + eax], bh
        dec eax
        cmp bl, 0
        jz @cycle
        dec edx
        jnz @cycle
        cmp minn, 1
        jc @nomin
        mov byte ptr[edi + eax + 1], 45
        dec eax

    @nomin:
        inc eax
```

```
    @space:
        mov byte ptr[edi + eax], 32
        sub eax, 1
        jnc @space
        pop ebp
        ret 12

Str_Dec_MY endp

Calc_N32_LONGOP proc
    push ebp
    mov ebp, esp
    mov edi, [ebp+12]
    mov ebx, [ebp+8]
    mov x, ebx
    mov n, 11
    xor ebx, ebx
    xor ecx, ecx

    @mult32:
        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 @mult32
        pop ebp
        ret 8

Calc_N32_LONGOP endp

end
```

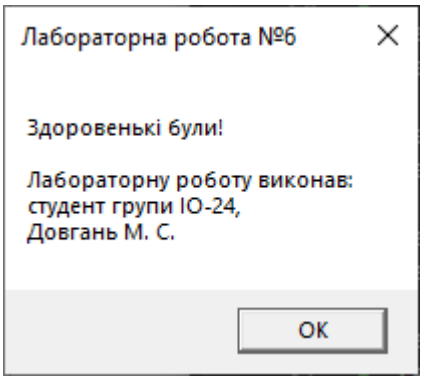
main6.asm:

```
.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 №6", 0
mainWindowText db "Здоровенькі були!", 13, 10, 13, 10,
```

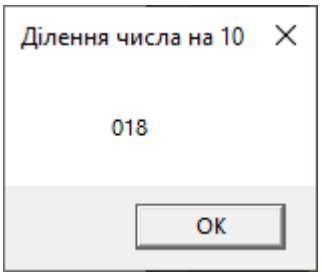
```
                          "Лабораторну роботу виконав:", 13, 10,
                        "студент групи ІО-24,", 13, 10,
                          "Довгань М. С.", 0

divideTitle db "Ділення числа на 10", 0
resIntText dd 12 dup(?)

factorialTitle db "Значення факторіалу n!", 0
resFactorialText dd 80 dup(?)

resFunctionTitle db "Результат обчислення функції Y", 0
resFunctionText dd 30 dup(?)

lastWindowTitle db "Програма завершила роботу", 0
lastWindowText db "Дякую за увагу!", 0

var dd 13 dup(0)
factorialValue dd 48
functionResult dd 0

testValue dd 180 ; число для перевірки ділення на 10
intP dd 0
fractP dd 0

.code
start:
    invoke MessageBoxA, 0, ADDR mainWindowText, ADDR mainWindowTitle, 0

    ;---------- Ділення числа на 10 ----------

    push offset testValue
    push offset intP
    push offset fractP
    push 32
    call Div_LONGOP

    push offset resIntText
    push offset intP
    push 32
    call Str_Dec_MY

    invoke MessageBoxA, 0, ADDR resIntText, ADDR divideTitle, 0

    ;---------- Обчислення значення факторіалу ----------

    mov [var], 1

    @fact:
        push offset var
        push factorialValue
```

```
        call Calc_N32_LONGOP

        dec factorialValue
        jne @fact

    push offset resFactorialText
    push offset var
    push 450
    call Str_Dec_MY

    invoke MessageBoxA, 0, ADDR resFactorialText, ADDR factorialTitle, 0

    ;---------- Обчислення функції Y ----------

    push 0
    push 5
    call Func

    mov functionResult, eax
    push offset resFunctionText
    push offset functionResult
    push 32
    call Str_Dec_MY

    invoke MessageBoxA, 0, ADDR resFunctionText, ADDR resFunctionTitle, 0

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

    invoke ExitProcess, 0

end start
```

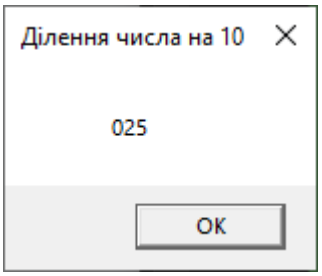
Результати виконання програми:

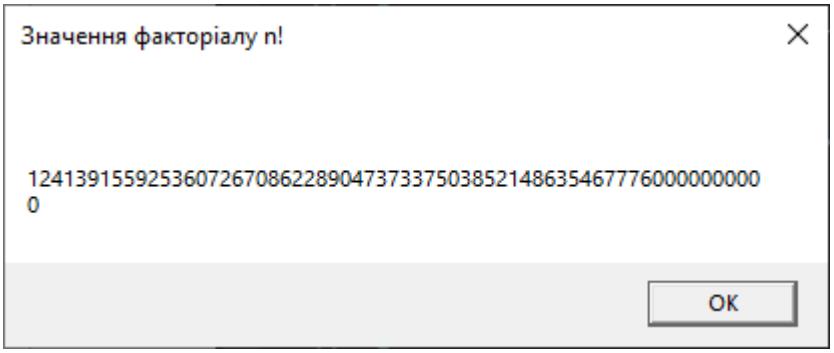


Ділення числа 180 на 10:

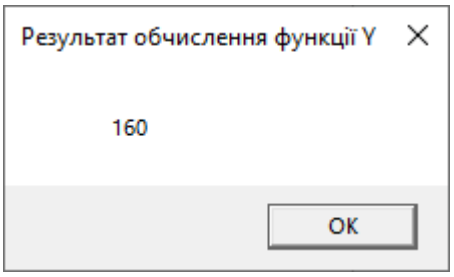


Ділення числа 250 на 10:

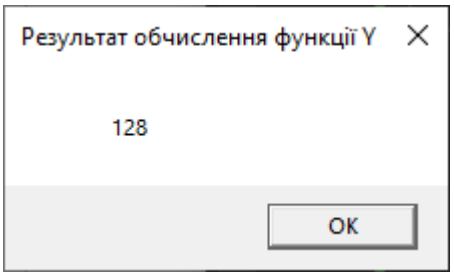


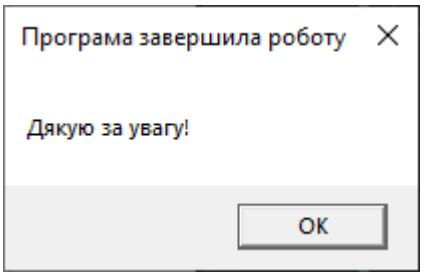


X = 0, M = 5:



X = 4, M = 7:





Аналіз виконання програми:
Створена мною програма виконує завдання лабораторної роботи, відповідно до мого індивідуального варіанту завдання. Ділене число записано безпосередньо до програми у файлі main6.asm (розділ .data) у вигляді dd (define double word), тобто 4 байти. Число, яке треба піднести до факторіалу записано там же, під назвою factorialValue з тим же типом dd. Числа X та M для обчислення функції Y записані вже в розділі .code. При початковому запуску програми у користувача з’являється стартове вікно-привітання, яке містить в собі привітання, а також інформацію про номер лабораторної роботи та її автора. Потім у нас викликається вікно із даними, щодо ділення заданого числа на 10 з обчисленим результатом. Після цього обчислюється та виводиться в окремому вікні значення факторіалу n в десятковій системі числення. І, перед кінцем, в нас виводиться також результат обчислення функції Y, згідно заданого варіанту завдання, також у десятковій системі числення. У кінці користувачеві виводиться останнє вікно, яке повідомляє, що програма виконала свою задачу та завершує роботу.

Висновок: під час виконання даної лабораторної роботи я ознайомився та навчився програмувати на Асемблері ділення чисел і перетворення з двійкової у десяткову систему числення.