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

Лабораторна робота №10
з дисципліни «Системне програмування»

Виконав:
студент групи ІО-32
Попенко Р.Л.
Перевірив:
Порєв В.М.

Київ, 2015 р.

Завдання:

1. Створити у середовищі MS Visual Studio проект C++ з ім’ям Lab10.
2. Написати вихідний текст програми згідно варіанту завдання. У проекті мають бути такі файли вихідного тексту:
- головний файл: lab10.cpp
- файли двох модулів на асемблері: module.asm та longop.asm.
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 машинний код програми.

№ варіанту	Вираження	Розрядність
24	$A^3 + B^2 - C$	320

Код програми

Lab10.cpp (частина)

```
void MyWork1(HWND hWnd)
{
	long oA[8]={0x80000001,0x80000001,0x80000001,0x80000001,
			0x80000001,0x80000001,0x80000001,0x80000001};
	long oB[8]={0x80010001,0x80020001,0x80030001,0x80040001,
			0x80050001,0x80060001,0x80070001,0x80080001};
	long oC[8]={0x00010001,0x00020001,0x00030001,0x00040001,
			0x00050001,0x00060001,0x00070001,0x00080001};

	long resultA2[16];
	long resultA3[16];
	long resultB2[16];
	long resultA3plusB2[16];
	long resultA3plusB2subC[16];
	char TextBuf[512];

	Mult_LONGOP(resultA2, oA, oA);
	Mult_LONGOP(resultA3, resultA2, oA);
	Mult_LONGOP(resultB2, oB, oB);
	Add_LONGOP(resultA3plusB2, resultB2, resultA3);
	Add_LONGOP(resultA3plusB2subC, oC, resultA3plusB2);
	StrHex_MY(320, resultA3plusB2subC, TextBuf);
	MessageBox(hWnd, TextBuf, "Результат A^3 + B^2 - C в 16", MB_OK);
	StrToDec_LONGOP(520, TextBuf, resultA3plusB2subC);
	MessageBox(hWnd, TextBuf, "Результат A^3 + B^2 - C в 10", MB_OK);
}
```

LONGOP.ASM

```
.586
.model flat, c
.data

	x dd 0h
	x1 dd 0h
	x2 dd 0h
	a dd 0
	b dd 0
	r dd 0
	buffer dd
1h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h,0h
	decCode db ?
	buf dd 80 dup(0)
```

```
	two dd 2
	fractionalPart db ?

.code

;Процедура Add_LONGOP: результат = A + B
;dest - адреса результату
;pA - адреса A
;pB - адреса B

Add_LONGOP proc dest:DWORD, pB:DWORD, pA:DWORD

	mov esi, pA
	mov ebx, pB
	mov edi, dest
	mov ecx, 10

	mov edx,0
	clc
	cycle:
		mov eax, dword ptr[esi + 4*edx]
		adc eax, dword ptr[ebx + 4*edx]
		mov dword ptr[edi + 4*edx], eax
		inc edx
		dec ecx
	jnz cycle

	ret

Add_LONGOP endp


;Процедура Sub_LONGOP: результат = A - B
;dest - адреса результату
;pA - адреса A
;pB - адреса B

Sub_LONGOP proc dest:DWORD, pB:DWORD, pA:DWORD

	mov esi, pA
	mov ebx, pB
	mov edi, dest
	mov ecx, 10

	mov edx,0
	clc
	cycle:
		mov eax, dword ptr[esi + 4*edx]
		sbb eax, dword ptr[ebx + 4*edx]
		mov dword ptr[edi + 4*edx], eax
		inc edx
		dec ecx
	jnz cycle

	ret

Sub_LONGOP endp


Mult_LONGOP proc dest:DWORD, pB:DWORD, pA:DWORD

	mov esi, pA
	mov edi, pB
	mov ebx, dest

	mov b, 0
	mov a, 0
	mov r, 0
```

```
	mov ecx, 10
	@cycle:
		push ecx
		mov ecx, 10
		@cycleInner:
			push ecx
			mov ecx, a
			mov eax, dword ptr[esi + 4 * ecx]
			mov ecx, b
			mul dword ptr[edi + 4 * ecx]
			mov ecx, r
			add eax, dword ptr[ebx + 4 * ecx]
			mov dword ptr[ebx + 4 * ecx], eax
			mov eax, dword ptr[ebx + 4 * ecx]
			adc edx, dword ptr[ebx + 4 * ecx + 4]
			mov dword ptr[ebx + 4 * ecx + 4], edx
			mov eax, dword ptr[ebx + 4 * ecx + 4]
			inc a
			inc r
			pop ecx
			dec ecx
			jnz @cycleInner
		inc b
		xor eax, eax
		mov a, eax
		mov eax, b
		mov r, eax
		pop ecx
		dec ecx
	jnz @cycle

	ret

Mult_LONGOP endp


StrToDec_LONGOP proc bons:DWORD, dest:DWORD, src:DWORD

	mov esi, src
	mov edi, dest
	mov eax, 10
	mov x1, eax
	mov eax, bons
	mov x2, eax

	mov b, 0

	xor ecx, ecx
	xor ebx, ebx
	@cycle:
		push ecx
		push edi

		push esi
		push offset buf
		push offset decCode
		call Div_LONGOP

		pop edi
		mov ebx, b
		mov al, byte ptr[decCode]
		add al, 48
		mov byte ptr[edi + ebx], al

		xor ecx, ecx
		@cycleInner:
			mov eax, dword ptr[buf + 4 * ecx]
			mov dword ptr[esi + 4 * ecx], eax
```

```
                mov dword ptr[buf + 4 * ecx], 0
                inc ecx
                cmp ecx, x1
                jl @cycleInner

            pop ecx
            inc ecx
            inc b
            cmp ecx, x2
            jl @cycle


        mov ebx, x2
        mov eax, x2
        xor edx, edx
        div two
        mov x2, eax
        dec ebx
        xor ecx, ecx
        @cycle1:

            mov al, byte ptr[edi + ecx]
            mov ah, byte ptr[edi + ebx]
            mov byte ptr[edi + ecx], ah
            mov byte ptr[edi + ebx], al

            dec ebx
            inc ecx
            cmp ecx, x2
            jl @cycle1

        ret

StrToDec_LONGOP endp


Div_LONGOP proc

        push ebp
        mov ebp, esp

        mov esi, [ebp + 16] ;number
        mov edi, [ebp + 12]  ;integer
        mov ebx, [ebp + 8] ;fractional
        mov eax, 10
        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 fractionalPart, dl
            mov dword ptr[edi + 4 * ebx], eax
            dec ebx
            pop ecx
            dec ecx

            jnz @cycle

        pop ebx
        mov al, fractionalPart
```

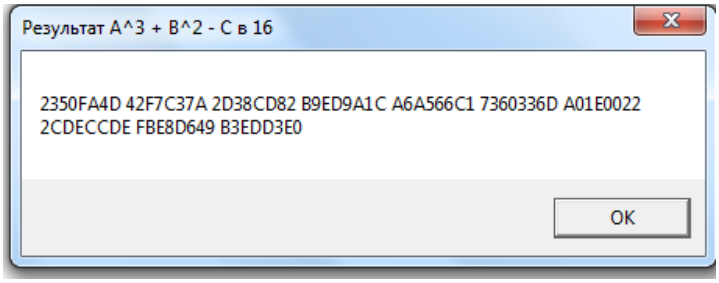
```
        mov byte ptr[ebx], al

        pop ebp
        ret 12

Div_LONGOP endp


End
```

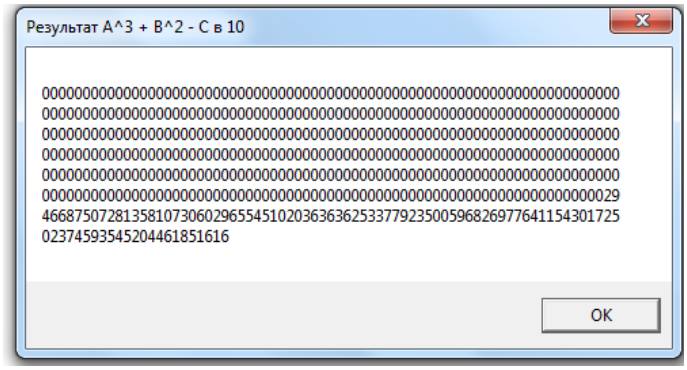
Module.ASM

```
.586
.model flat, c
.code

;Процедура StrHex_MY записує шістнадцятковий код числових даних
;bits - розрядність даних у бітах
;src - адреса даних
;dest - адреса буфера результату (рядка символів)

StrHex_MY proc bits:DWORD, src:DWORD, dest:DWORD

mov ecx, bits
cmp ecx, 0
jle @exitp
shr ecx, 3 ;кількість байтів числа
mov esi, src
mov ebx, dest
@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:

ret

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
```

Результати роботи програми:





Висновок: у даній лабораторній роботі я навчився створювати програми на C++ з використанням модулів на асемблері