МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Лабораторна робота №9
з предмету “Системне програмування”

Виконав:
Студент 2-го курсу ФІОТ
групи ІО-22
Бас Андрій

Київ 2014

Лабораторна робота №9

Використання у проекті C++ модулів на асемблері

Мета: Навчитися створювати програми на C++ з використанням модулів на асемблері

Роздруківка тексту програми

```
Longop.h
extern "C"
{
	void Add_LONGOP(long bits, long *pA, long *pB, long *dest);
	void Sub_LONGOP(long bits, long *pA, long *pB, long *dest);
	void Mul_N32_LONGOP(long counter, long N, long *pA, long *res);
	void Mul_NN_LONGOP(long aLen, long *pA, long bLen, long *pB, long *res);
	void Div_N32_LONGOP(long aLen, long *pA, long nDivider, long *res);
}

Module.h
extern "C"
{
	void StrHex_MY(long bits, long *src, char *dest);
	void StrDec_MY(long bits, long *src, char *dest);
}

Lab09.cpp
void lab09Func(HWND hWnd)
{
	const long len = 1;
	long oA[len] = { 0x00010001 };
	long oB[len] = { 0x00001001 };
	long oC[len * 2] = { 0x00010001, 0x02100022 };
	long oD[len * 2] = { 0x00010001, 0x20010003 };

	long oAB[len * 2] = { 0, 0 };
	long res1[len * 2] = { 0, 0 };
	long res2[len * 2] = { 0, 0 };

	char TextBuf[len * 2 * 32];
	Mul_NN_LONGOP(len, oA, len, oB, oAB);
	StrHex_MY(len * 2 * 32, oAB, TextBuf);
	MessageBox(hWnd, TextBuf, "Результат A * B", MB_OK);

	Add_LONGOP(len * 2, oAB, oC, res1);
	StrHex_MY(len * 2 * 32, res1, TextBuf);
	MessageBox(hWnd, TextBuf, "Результат A * B + C", MB_OK);

	Sub_LONGOP(len * 2, res1, oD, res2);
	StrHex_MY(len * 2 * 32, res2, TextBuf);
	MessageBox(hWnd, TextBuf, "Результат A * B + C - D", MB_OK);
}

.586
.model flat, c
.code
;**************************
;**
;**
;**************************

Add_LONGOP proc counter:DWORD, pA:DWORD, pB:DWORD, res:DWORD
```

```
	mov edi, res   ; address of RESULT
	mov ebx, pB  ; address of operand B
	mov esi, pA  ; address of operand A
	mov ecx, counter  ; counter, required number of repetitions

	clc         ; обнулюємо біт CF регістру EFLAGS, куди записується переповнення

	mov edx, 0  ; лічильник, що відповідає за зсув

	@cycle:

		mov eax, dword ptr [esi + edx] ; take next 32 bits of A
		adc eax, dword ptr [ebx + edx] ; add them with next 32 bits of B
		mov dword ptr [edi + edx], eax ; write result on appropriate position

		inc edx
		inc edx
		inc edx
		inc edx

		dec ecx
		jnz @cycle

	@exitp:
		ret
Add_LONGOP endp


;***************************
;**
;**
;***************************
Sub_LONGOP proc counter:DWORD, pA:DWORD, pB:DWORD, res:DWORD

	mov edi, res   ; address of RESULT
	mov ebx, pB  ; address of operand B
	mov esi, pA  ; address of operand A
	mov ecx, counter  ; counter, required number of repetitions

	clc         ; put zero to flag CF from EFLAGS, where borrow is put

	mov edx, 0  ; лічильник, що відповідає за зсув

	@cycle:

		mov eax, dword ptr [esi + edx] ; take next 32 bits of A
		sbb eax, dword ptr [ebx + edx] ; sub them from next 32 bits of B
		mov dword ptr [edi + edx], eax ; write result on appropriate position

		inc edx
		inc edx
		inc edx
		inc edx

		dec ecx
		jnz @cycle

	@exitp:
		ret
Sub_LONGOP endp

;***************************
;**
;** 1 param : len of A
;** 2 param : pointer to A
```

```
;** 3 param : 32-bit multiplier
;** 4 param : pointer to RESULT
;**
;**************************
Mul_N32_LONGOP proc counter:DWORD, N:DWORD, pA:DWORD, res:DWORD

	mov edi, res		; address of RESULT
	mov ebx, N			; N (32-bit multiplier)
	mov esi, pA			; address of operand A
	mov ecx, counter	; counter, how many dd's to multiply
					; simply, just lenght of the array (operand A)

	clc		; обнулюємо біт CF регістру EFLAGS, куди записується переповнення

	; put all zero's to RESULT
	mov ebp, ecx
	dec ebp
	@zero:
		mov dword ptr [edi + ebp * 4], 0
		dec ebp
		jge @zero

	mov ebp, 0  ; лічильник, що відповідає за зсув

	@cycle:

		add ebp, 4	; increment ebp
				; ATTENTION : DO NOT PUT THIS OPERATION AFTER next 4
commands, because small children will cry

		mov eax, dword ptr [esi + ebp - 4]		; take next 32 bits of
A
		mul ebx		;
multiply them with N (stored in ebx)
		add dword ptr [edi + ebp - 4], eax	; add lower  32-bits from eax
(without carry)
		adc dword ptr [edi + ebp], edx  ; add higher 32-bits from edx (with carry)

		dec ecx
		jnz @cycle

	ret
Mul_N32_LONGOP endp



;**************************
;**
;** ATTENTION !!!! multiplicatioin is writen to result, but added to it ))) enjoy
;**
;** 1 param : len of A
;** 2 param : pointer to A
;** 3 param : len of B
;** 4 param : pointer to B
;** 5 param : pointer to RESULT
;**
;**************************
Mul_NN_LONGOP proc aLen:DWORD, pA:DWORD, bLen:DWORD, pB:DWORD, res:DWORD

	mov edi, res		; address of RESULT
	mov ebx, pB			; address of operand B
	mov ecx, bLen		; length of B
	mov maxCounter2, ecx; save length of B to maxCounter2

	mov esi, pA  ; address of operand A
```

```
	mov ecx, aLen  ; length of A
	mov maxCounter1, ecx ; save length of A to maxCounter2

	mov counter1, 0h ; put zero to counter1
	@outer:
		mov eax, counter1
		cmp eax, maxCounter1    ; check outer counter
		jge @exitp              ; exit on condition counter1 >= maxCounter1
		mov counter2, 0h           ; prepare counter2

		@inner:
			mov ecx, counter1                       ; get index of counter1
			mov eax, dword ptr [esi + 4 * ecx]     ; take next 32-bits from A

			mov ecx, counter2                                ; get index of
counter2
			mul dword ptr [ebx + 4 * ecx]          ; take next 32-bits from B

			add ecx, counter1                                ; get index where to
put result, ecx = counter1 + counter2
			; save result
			add dword ptr [edi + 4 * ecx    ], eax
			adc dword ptr [edi + 4 * ecx + 4], edx

			inc counter2            ; increment counter2
			mov eax, counter2
			cmp eax, maxCounter2    ; check inner counter
			jl @inner               ; continue inner loop if counter2 < maxCounter2

			inc counter1  ; increment counter1
			jmp @outer          ; jump to @outer

	@exitp:
		ret

Mul_NN_LONGOP endp


;**************************
;**
;** procedure divides array A by 32-bit value N, fraction written to edx
;**
;** 1 param : length of A
;** 2 param : pointer to A
;** 3 param : divider N
;** 4 param : pointer to RESULT
;**
;**************************
Div_N32_LONGOP proc aLen:DWORD, pA:DWORD, nDivider:DWORD, res:DWORD

	mov esi, res               ; address of RESULT
	mov ecx, nDivider                       ; divider (32-bit value)
	mov divider, ecx           ; save divider

	mov ebx, pA                       ; address of dividend A (array of 32-bits)
	mov ecx, aLen              ; length of A

	xor edx, edx               ; put zero to edx, it is used by div operation below
	dec ecx
	@cycle:

		mov eax, dword ptr [ebx + 4 * ecx]   ; get next 32-bits to divide
		div divider                                          ; divide them
		mov dword ptr [esi + 4 * ecx], eax   ; save result
```

```
              dec ecx
              jge @cycle

       @exitp:
              ret
Div_N32_LONGOP endp

Copy_LONGOP proc dest:DWORD, src:DWORD, len:DWORD

       mov edi, dest       ; address of DEST
       mov ebx, src        ; address of SRC
       mov ecx, len        ; length of arrays

       dec ecx
       @copy_my:
              mov eax, [ebx + 4 * ecx]
              mov [edi + 4 * ecx], eax
              dec ecx
              jge @copy_my

       ret
Copy_LONGOP endp

.data
       ; vars for Mul_NN
       counter1 dd 0h
       counter2 dd 0h
       maxCounter1 dd 0h
       maxCounter2 dd 0h

       ; vars for Div_N32
       divider dd 1
end
```