*МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ*

*НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ*

*“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”*

*Лабораторна робота №5*

*з предмету “Системне програмування”*

*Виконав:*

*Студент 2-го курсу ФІОТ*

*групи ІО-22*

*Коломієць Олег*

*Київ 2014*

Лабораторна робота №5

Програмування множення чисел підвищеної розрядності

Мета: Навчитися програмувати на асемблері множення чисел підвищеної розрядності, а також закріпити навички програмування власних процедур у модульному проекті. *Роздруківка тексту програми*

.586

.model flat, c

.data

counter dd 0

counterin dd 0

.code

Add\_LONGOP proc

push ebp

mov ebp, esp

mov esi, [ebp + 16]; ESI = адреса A

mov ebx, [ebp + 12]; EBX = адреса B

mov edi, [ebp + 8]; EDI = адреса результату

mov ecx, 20; ECX = потрібна кількість повторень

mov edx, 0

clc; обнулює біт CF регістру EFLAGS

cycle :

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

adc eax, dword ptr[ebx + 4 \* edx]; додавання групи з 32 бітів

mov dword ptr[edi + 4 \* edx], eax

inc edx

dec ecx; лічильник зменшуємо на 1

jnz cycle

pop ebp

ret 12

Add\_LONGOP endp

Sub\_LONGOP proc

push ebp

mov ebp, esp

mov esi, [ebp + 16]; ESI = адреса A

mov ebx, [ebp + 12]; EBX = адреса B

mov edi, [ebp + 8]; EDI = адреса результату

mov ecx, 120; ECX = потрібна кількість повторень

mov edx, 0

clc; обнулює біт CF регістру EFLAGS

cycle :

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

sbb eax, dword ptr[ebx + 4 \* edx]; віднімання групи з 32 бітів

mov dword ptr[edi + 4 \* edx], eax

inc edx

dec ecx; лічильник зменшуємо на 1

jnz cycle

pop ebp

ret 12

Sub\_LONGOP endp

Mul\_N32\_LONGOP proc

push ebp

mov ebp, esp

mov esi, [ebp + 16]

mov ebx, [ebp + 12]

mov edi, [ebp + 8]

mov ecx, 0

@cycle:

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

mul ebx

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

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

inc ecx

cmp ecx, 7

jb @cycle

pop ebp; відновлення стеку

ret 12

Mul\_N32\_LONGOP endp

Mul\_NN\_LONGOP proc

push ebp

mov ebp, esp

mov esi, [ebp + 16]; A

mov edi, [ebp + 8]; dest

mov ebp, [ebp + 12]; B

mov counter, 0

@cycle:

mov counterin, -7

@innerCycle :

mov ecx, counter

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

mov ecx, counterin

mov ebx, dword ptr[ebp + 20 + 4 \* ecx]

mul ebx

mov ebx, counter

add ebx, counter

add ebx, counter

add ebx, counter

add ebx, edi

adc dword ptr[20 + 4 \* ecx + ebx], eax

adc dword ptr[24 + 4 \* ecx + ebx], edx

inc counterin

jnz @innerCycle

inc counter

cmp counter, 7

jb @cycle

pop ebp; відновлення стеку

ret 12

Mul\_NN\_LONGOP endp

end

.586

.model flat, c

.code

;процедура StrHex\_MY записує текст шістнадцятькового 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

.586

.model flat, stdcall

option casemap : none

include \masm32\include\kernel32.inc

include \masm32\include\user32.inc

include \masm32\include\windows.inc

includelib \masm32\lib\kernel32.lib

includelib \masm32\lib\user32.lib

include module.inc

include longop.inc

.data

ValueA dd 00000001h, 00000000h, 00000000h, 00000000h, 00000000h, 00000000h, 00000000h

factor dd 00000001h

temp dd 7 dup(0)

ValueB dd 14 dup(0)

temp2 dd 14 dup(0)

Caption db "Результат", 0

n32TextBuf db 224 dup(? )

nnTextBuf db 448 dup(? )

counter dd 0

.code

start :

@cycle1 :

push offset ValueA

push factor

push offset temp

call Mul\_N32\_LONGOP

inc factor

inc counter

cmp counter, 48

mov ecx, 7

@swap:

mov ebx, dword ptr[temp + 4 \* ecx - 4]

mov dword ptr[ValueA + 4 \* ecx - 4], ebx

mov dword ptr[temp + 4 \* ecx - 4], 0

dec ecx

jnz @swap

jb @cycle1

push offset n32TextBuf

push offset ValueA

push 224

call StrHex\_MY

invoke MessageBoxA, 0, ADDR n32TextBuf, ADDR Caption, MB\_ICONINFORMATION

push offset ValueA

push offset ValueA

push offset ValueB

call Mul\_NN\_LONGOP

push offset nnTextBuf

push offset ValueB

push 448

call StrHex\_MY

invoke MessageBoxA, 0, ADDR nnTextBuf, ADDR Caption, MB\_ICONINFORMATION

invoke ExitProcess, 0

end start