НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №5

з дісципліни **«**Системне програмування**»**

Виконав:

студент 2 курсу

ФІОТ гр. ІО-21

Кузьменко Володимир

Перевірив:

Порєв В.М.

Київ 2014 р.

.586

.model flat, stdcall

option casemap :none

include \masm32\include\kernel32.inc

include \masm32\include\user32.inc

option casemap :none

include \masm32\include\windows.inc

include module.inc

include Longop.inc

includelib \masm32\lib\kernel32.lib

includelib \masm32\lib\user32.lib

.const

.data

TextBuf db 192 dup(?)

counter db ?

counter2 db ?

Caption db 'Result', 0

ValueA dd 80010001h, 80020001h, 80030001h, 80040001h, 80050002h

ValueA1 dd 1,0,0,0,0

Value dd 2

Result dd 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

Result1 dd 1,0,0,0,0,0,0,0,0,0

Result2 dd 0,0,0,0,0,0,0,0,0,0

.code

start:

mov counter,62

cycle1:

push offset Result1

push offset Value

push offset Result2

call N32

push offset Result2

push offset Result1

call Copy

inc Value

dec counter

jnz cycle1

push offset TextBuf

push offset Result2

push 320

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

xor eax,eax

push offset Result2

push offset Result2

push offset Result

call NN

push offset TextBuf

push offset Result

push 640

call StrHex\_MY

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

invoke MessageBox, 0, ADDR TextBuf, ADDR Caption, MB\_ICONINFORMATION

invoke ExitProcess,0

end start

.586

.model flat, c

.data

src dd 80010001h

counter dd 0

counter2 dd 0

shift1 dd 0

shift2 dd 0

B1 dd 0,0,0,0,0

B2 dd 0,0,0,0,0

A1 dd 0,0,0,0,0

A2 dd 0,0,0,0,0

Result1 dd 0,0,0,0,0,0

Result2 dd 2,2,2,2,2,2

Result dd 0,0,0,0,0,0

cf db 1 dup(0)

.code

AddLONGOP 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, 6; ECX = потрібна кількість повторень

clc ; обнулює біт CF регістру EFLAGS

mov edx, 0

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; модифікація зсуву

dec ecx ; лічильник зменшуємо на 1

jnz cycle

pop ebp ;відновлення стеку

ret 12

AddLONGOP endp

N32 proc

push ebp

mov ebp,esp

mov esi, [ebp+16]

mov ebx, [ebp+12]

mov edi,[ebp+8]

mov counter,8

mov ecx,0

mov edx,0

mov eax,0

cycle:

mov eax, dword ptr[esi+8\*ecx]

mul dword ptr[ebx]

mov dword ptr[edi+ecx\*8],eax

mov dword ptr[edi+ecx\*8+4],edx

inc ecx

dec counter

jnz cycle

clc;

mov edx,0

mov eax,0

mov counter,7

mov ecx,0

cycle1:

mov eax, dword ptr[esi+8\*ecx+4]

mul dword ptr[ebx]

add cf, 0ffh

adc eax,dword ptr[edi+8\*ecx+4]

mov dword ptr[edi+8\*ecx+4],eax

adc edx,dword ptr[edi+8\*ecx+8]

mov dword ptr[edi+8\*ecx+8],edx

mov cf, 0

adc cf, 0

inc ecx

dec counter

jnz cycle1

pop ebp

ret 12

N32 endp

Copy proc

push ebp

mov ebp,esp

mov ebx, [ebp+12]

mov edi,[ebp+8]

mov ecx,16

mov edx, 0

cycle:

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

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

inc edx

dec ecx ;

jnz cycle

pop ebp

ret 12

Copy endp

NN proc

push ebp

mov ebp,esp

mov esi, [ebp+16]

mov ebx, [ebp+12]

mov edi,[ebp+8]

clc

mov edx,0

mov ecx,0

mov counter,10

level1:

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

mul dword ptr[ebx]

add cf, 0ffh

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

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

mov cf, 0

adc cf, 0

inc ecx

dec counter

jnz level1

clc

mov edx,0

mov ecx,0

mov counter,10

level2:

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

mul dword ptr[ebx+4]

add cf, 0ffh

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

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

mov cf, 0

adc cf, 0

inc ecx

dec counter

jnz level2

clc

mov edx,0

mov ecx,0

mov counter,10

level3:

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

mul dword ptr[ebx+8]

add cf, 0ffh

adc dword ptr[edi+4\*ecx+8],eax

adc dword ptr[edi+4\*ecx+4+8],edx

mov cf, 0

adc cf, 0

inc ecx

dec counter

jnz level3

clc

mov edx,0

mov ecx,0

mov counter,10

level4:

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

mul dword ptr[ebx+12]

add cf, 0ffh

adc dword ptr[edi+4\*ecx+12],eax

adc dword ptr[edi+4\*ecx+4+12],edx

mov cf, 0

adc cf, 0

inc ecx

dec counter

jnz level4

clc

mov edx,0

mov ecx,0

mov counter,10

level5:

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

mul dword ptr[ebx+16]

add cf, 0ffh

adc dword ptr[edi+4\*ecx+16],eax

adc dword ptr[edi+4\*ecx+4+16],edx

mov cf, 0

adc cf, 0

inc ecx

dec counter

jnz level5

clc

mov edx,0

mov ecx,0

mov counter,10

level6:

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

mul dword ptr[ebx+20]

add cf, 0ffh

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

adc dword ptr[edi+4\*ecx+4+20],edx

mov cf, 0

adc cf, 0

inc ecx

dec counter

jnz level6

clc

mov edx,0

mov ecx,0

mov counter,10

level7:

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

mul dword ptr[ebx+24]

add cf, 0ffh

adc dword ptr[edi+4\*ecx+24],eax

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

mov cf, 0

adc cf, 0

inc ecx

dec counter

jnz level7

clc

mov edx,0

mov ecx,0

mov counter,10

level8:

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

mul dword ptr[ebx+28]

add cf, 0ffh

adc dword ptr[edi+4\*ecx+28],eax

adc dword ptr[edi+4\*ecx+4+28],edx

mov cf, 0

adc cf, 0

inc ecx

dec counter

jnz level8

clc

mov edx,0

mov ecx,0

mov counter,10

level9:

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

mul dword ptr[ebx+32]

add cf, 0ffh

adc dword ptr[edi+4\*ecx+32],eax

adc dword ptr[edi+4\*ecx+4+32],edx

mov cf, 0

adc cf, 0

inc ecx

dec counter

jnz level9

clc

mov edx,0

mov ecx,0

mov counter,10

level10:

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

mul dword ptr[ebx+36]

add cf, 0ffh

adc dword ptr[edi+4\*ecx+36],eax

adc dword ptr[edi+4\*ecx+4+36],edx

mov cf, 0

adc cf, 0

inc ecx

dec counter

jnz level10

pop ebp

ret 12

NN 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