НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №8

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

includelib \masm32\lib\kernel32.lib

includelib \masm32\lib\user32.lib

include \masm32\include\comdlg32.inc

includelib \masm32\lib\comdlg32.lib

include \masm32\include\windows.inc

include \masm32\modules\module.inc

include \masm32\modules\longop.inc

.data

Caption db "---",0

Var dd ?

Sor dd ?

last dd 544 dup(0)

Text dd 544 dup(0)

divreshex db 10 dup(0)

divresdec db 10 dup(0)

hFile dd 0

pRes dd 0

szFileName db 256 dup(0)

szTextBuf db " ",13,10, 0

regeax db 10 dup(0)

regebx db 10 dup(0)

regecx db 10 dup(0)

regedx db 10 dup(0)

regesi db 10 dup(0)

.code

MySaveFileName proc

LOCAL ofn : OPENFILENAME

invoke RtlZeroMemory, ADDR ofn, SIZEOF ofn ; спочатку усі поля обнулюємо

mov ofn.lStructSize, SIZEOF ofn

mov ofn.lpstrFile, OFFSET szFileName

mov ofn.nMaxFile, SIZEOF szFileName

invoke GetSaveFileName,ADDR ofn ; виклик вікна File Save As

ret

MySaveFileName endp

main:

call MySaveFileName

cmp eax, 0 ; перевірка: якщо у вікні було натиснуто кнопку Cancel, то EAX=0

je exit

invoke CreateFile, ADDR szFileName, ;

GENERIC\_WRITE, ;

FILE\_SHARE\_WRITE, ; сторення файлу

0, CREATE\_ALWAYS, ;

FILE\_ATTRIBUTE\_NORMAL, ;

0 ;

cmp eax, INVALID\_HANDLE\_VALUE ;

je exit ;доступ до файлу неможливий

mov hFile, eax ; прив'язка до хендлера

xor esi, esi

mov esi, 1h

invoke GlobalAlloc, GPTR, 3000 ;

mov Var, eax

add eax, 545 ;

mov Sor, eax ; створення динамічних масивів

cycle:

;

xor eax, eax ;

innercycle: ;

; обнулення всих масивів,

mov dword ptr[Text + eax], 0 ; які використовуються

add eax, 4 ; в програмі

cmp eax, 545 ;

jl innercycle ;

mov dword ptr[regesi], esi ;

push Sor ; обчислення факторіалу

push esi ;

call Factorial ;

push Var ;

push Sor ; переведення в десяткову систему

push 544 ;

call StrDec ;

push offset Text ;

push Var ;

push 544 ; переведення в текст

call StrHex\_MY ;

mov eax, dword ptr[regeax] ;

mov ebx, dword ptr[regebx] ; відновлення регістрів

mov ecx, dword ptr[regecx] ;

mov edx, dword ptr[regedx] ;

invoke lstrlen, ADDR Text ;

invoke WriteFile, hFile, ADDR Text, ; запис у файл

eax, ADDR pRes, 0 ;

invoke lstrlen, ADDR szTextBuf ;

invoke WriteFile, hFile, ADDR szTextBuf, ;

eax, ADDR pRes, 0 ;

mov dword ptr[regeax], eax ;

mov dword ptr[regecx], ecx ;

mov dword ptr[regebx], ebx ; збереження регістрів

mov dword ptr[regedx], edx ;

mov esi, dword ptr[regesi] ;

invoke RtlZeroMemory, Var, 3000

inc esi ;

cmp esi, 51 ; перехід

jl cycle ;

invoke GlobalFree, Var ; вивільнення динамічної пам'яті

invoke CloseHandle, hFile ; закриття файлу

exit:

invoke ExitProcess, 0 ; вихід з програми.

end main

.586

.model flat, c

.data

level1 db 432 dup(0)

level2 db 432 dup(0)

counter db 40 dup(0)

variable db 40 dup(0)

cf db 1 dup(0)

cf2 db 1 dup(0)

temp db 1 dup(0)

regeax db 10 dup(0)

regebx db 10 dup(0)

regecx db 10 dup(0)

regedx db 10 dup(0)

.code

Add\_Long proc

push ebp

mov ebp,esp

mov esi, [ebp+16]

mov ebx, [ebp+12]

mov edi, [ebp+8]

mov edx, 0

mov ecx, 384

addAB:

mov eax, dword ptr[esi+edx]

adc eax, dword ptr[ebx+edx]

mov dword ptr [edi+edx], eax

inc edx

inc edx

inc edx

inc edx

dec ecx

dec ecx

dec ecx

dec ecx

jnz addAB

pop ebp

ret 12

Add\_Long endp

Sub\_Long proc

push ebp

mov ebp,esp

mov esi, [ebp+16]

mov ebx, [ebp+12]

mov edi, [ebp+8]

mov ecx, 736

mov edx, 0

subAB:

mov eax, dword ptr[esi+edx]

sbb eax, dword ptr[ebx+edx]

mov dword ptr [edi+edx], eax

inc edx

inc edx

inc edx

inc edx

dec ecx

dec ecx

dec ecx

dec ecx

jnz subAB

pop ebp

ret 12

Sub\_Long endp

Factorial proc

push ebp

mov ebp,esp

mov ebx, [ebp+12] ;адреса результату

mov edi, [ebp+8] ;факторіал числа

add dword ptr[ebx], 00000001h

mov ecx, edi

mov dword ptr[counter], 00000001h

factcycle:

mov dword ptr [variable], ecx

mov ecx, 56

mov esi, 0

level1cycle:

mov eax, dword ptr[ebx + esi]

mul dword ptr [counter]

mov dword ptr[level1 + esi], eax

mov dword ptr[level1 + 4 + esi], edx

add esi, 8

sub ecx, 8

jnz level1cycle

mov ecx, 48

mov esi, 0

level2cycle:

mov eax, dword ptr[ebx + 4 + esi]

mul dword ptr[counter]

mov dword ptr[level2 + 4 + esi], eax

mov dword ptr[level2 + 8 + esi], edx

add esi, 8

sub ecx, 8

jnz level2cycle

mov ecx, 56

mov esi, 0

sumlevelscycle:

mov eax, dword ptr[level1 + esi]

add cf, 0ffh

adc eax, dword ptr[level2 + esi]

mov cf, 0

adc cf, 0

mov [ebx + esi], eax

add esi, 4

sub ecx, 4

jnz sumlevelscycle

inc dword ptr [counter]

mov ecx, dword ptr [variable]

dec ecx

jnz factcycle

; mov ecx, dword ptr[counter]

; mov dword ptr[ebx], ecx

pop ebp

ret 12

Factorial endp

Multlong proc

push ebp

mov ebp,esp

mov esi, [ebp+16] ; a

mov ebx, [ebp+12] ; b

mov edi, [ebp+8] ; res

clc

mov ecx, 28

mov dword ptr[counter], 0

level0cycle:

mov eax, dword ptr[ebx]

mov dword ptr[variable], ecx

mov ecx, dword ptr[counter]

mul dword ptr[esi + ecx]

clc

add cf, 11111111b

adc dword ptr[edi + ecx], eax

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

mov cf, 00000000b

adc cf, 0

add dword ptr [counter], 4

mov ecx, dword ptr[variable]

sub ecx, 4

jnz level0cycle

clc

; mov cf, 00000000b

mov ecx, 28

mov dword ptr[counter], 0

level1cycle:

mov eax, dword ptr[ebx + 4]

mov dword ptr[variable], ecx

mov ecx, dword ptr[counter]

mul dword ptr[esi + ecx]

clc

add cf, 11111111b

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

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

mov cf, 00000000b

adc cf, 0

add dword ptr [counter], 4

mov ecx, dword ptr[variable]

sub ecx, 4

jnz level1cycle

clc

mov ecx, 28

mov dword ptr[counter], 0

level2cycle:

mov eax, dword ptr[ebx + 8]

mov dword ptr[variable], ecx

mov ecx, dword ptr[counter]

mul dword ptr[esi + ecx]

clc

add cf, 11111111b

adc dword ptr[edi + 8 + ecx], eax

adc dword ptr[edi + 12 + ecx], edx

mov cf, 00000000b

adc cf, 0

add dword ptr [counter], 4

mov ecx, dword ptr[variable]

sub ecx, 4

jnz level2cycle

clc

mov ecx, 28

mov dword ptr[counter], 0

level3cycle:

mov eax, dword ptr[ebx + 12]

mov dword ptr[variable], ecx

mov ecx, dword ptr[counter]

mul dword ptr[esi + ecx]

clc

add cf, 11111111b

adc dword ptr[edi + 12 + ecx], eax

adc dword ptr[edi + 16 + ecx], edx

mov cf, 00000000b

adc cf, 0

add dword ptr [counter], 4

mov ecx, dword ptr[variable]

sub ecx, 4

jnz level3cycle

clc

mov ecx, 28

mov dword ptr[counter], 0

level4cycle:

mov eax, dword ptr[ebx + 16]

mov dword ptr[variable], ecx

mov ecx, dword ptr[counter]

mul dword ptr[esi + ecx]

clc

add cf, 11111111b

adc dword ptr[edi + 16 + ecx], eax

adc dword ptr[edi + 20 + ecx], edx

mov cf, 00000000b

adc cf, 0

add dword ptr [counter], 4

mov ecx, dword ptr[variable]

sub ecx, 4

jnz level4cycle

clc

mov ecx, 28

mov dword ptr[counter], 0

level5cycle:

mov eax, dword ptr[ebx + 20]

mov dword ptr[variable], ecx

mov ecx, dword ptr[counter]

mul dword ptr[esi + ecx]

clc

add cf, 11111111b

adc dword ptr[edi + 20 + ecx], eax

adc dword ptr[edi + 24 + ecx], edx

mov cf, 00000000b

adc cf, 0

add dword ptr [counter], 4

mov ecx, dword ptr[variable]

sub ecx, 4

jnz level5cycle

clc

mov ecx, 28

mov dword ptr[counter], 0

level6cycle:

mov eax, dword ptr[ebx + 24]

mov dword ptr[variable], ecx

mov ecx, dword ptr[counter]

mul dword ptr[esi + ecx]

clc

add cf, 11111111b

adc dword ptr[edi + 24 + ecx], eax

adc dword ptr[edi + 28 + ecx], edx

mov cf, 00000000b

adc cf, 0

add dword ptr [counter], 4

mov ecx, dword ptr[variable]

sub ecx, 4

jnz level6cycle

pop ebp

ret 12

Multlong endp

module\_changer proc

push ebp

mov ebp,esp

mov edi, [ebp+16] ; sourse/res

mov ebx, [ebp+12] ; m

mov edx, [ebp+8] ; n

; not dword ptr [edi]

mov ecx, 544

mov esi, 0

cycle:

mov dword ptr[edi + esi], 0ffffffffh

add esi, 4

sub ecx, 4

jnz cycle

mov ecx, ebx

cmp ecx, 0

jz over

mov esi, edx

cycle2:

mov ebx, esi ; esi – це номер біту

mov edx, ebx

shr ebx, 3 ; номер байту

and edx, 07h ; позиція потрібного біту у байті

mov eax, 1

mov dword ptr [temp], ecx

mov ecx, edx

shl eax, cl ; маска 0..010..0 за умовчанням

mov ecx, dword ptr [temp]

not eax ; маска 1..101..1 для AND

and dword ptr [edi + ebx], eax

add esi, 1

sub ecx, 1

jnz cycle2

over:

pop ebp

ret 12

module\_changer endp

module\_div10 proc

push ebp

mov ebp,esp

mov esi, [ebp+20] ; sourse

mov edi, [ebp+16] ; n-bit

mov ebx, [ebp+12] ; res

mov edx, [ebp+8] ; last

mov eax, edi

shr eax, 3

mov dword ptr[counter], 0h

xor ecx, ecx

mov cl, byte ptr[esi + eax - 1]

shr cl, 4

sub edi, 4

again:

cmp cl, 10

jl little

sub cl, 10

mov dword ptr[regeax], eax

xor eax, eax

mov dword ptr[regecx], ecx

xor ecx, ecx

mov edx, edi

mov ecx, edx

shr edx, 3

and ecx, 07h

mov al, 1

shl al, cl

; not al

or byte ptr[ebx + edx], al

mov eax, dword ptr[regeax]

mov edx, [ebp+8]

mov ecx, dword ptr[regecx]

little:

dec edi

cmp edi, 0

jl exit

shl cl, 1

mov dword ptr[regeax], eax

xor eax, eax

mov dword ptr[regecx], ecx

xor ecx, ecx

mov edx, edi ; Nbit – це номер біту

mov ecx, edx

shr edx, 3 ; номер байту

and ecx, 07h ; бітова позиція = вирізаємо 3 молодші біти

mov al, 1

shl al, cl ; AL = маска вирізання біту Nbit

mov ah, byte ptr [esi + edx]

and ah, al ; результат у регістрі AH

mov edx, [ebp+8]

mov ecx, dword ptr[regecx]

cmp ah, 0

je here

add cl, 01h

here:

mov eax, dword ptr[regeax]

jmp again

exit:

mov byte ptr[edx], cl

pop ebp

ret 16

module\_div10 endp

end

.586

.model flat, c

.data

level1 db 432 dup(0)

level2 db 432 dup(0)

counter db 40 dup(0)

variable db 40 dup(0)

cf db 1 dup(0)

cf2 db 1 dup(0)

temp db 1 dup(0)

regeax db 10 dup(0)

regebx db 10 dup(0)

regecx db 10 dup(0)

regedx db 10 dup(0)

.code

Add\_Long proc

push ebp

mov ebp,esp

mov esi, [ebp+16]

mov ebx, [ebp+12]

mov edi, [ebp+8]

mov edx, 0

mov ecx, 384

addAB:

mov eax, dword ptr[esi+edx]

adc eax, dword ptr[ebx+edx]

mov dword ptr [edi+edx], eax

inc edx

inc edx

inc edx

inc edx

dec ecx

dec ecx

dec ecx

dec ecx

jnz addAB

pop ebp

ret 12

Add\_Long endp

Sub\_Long proc

push ebp

mov ebp,esp

mov esi, [ebp+16]

mov ebx, [ebp+12]

mov edi, [ebp+8]

mov ecx, 736

mov edx, 0

subAB:

mov eax, dword ptr[esi+edx]

sbb eax, dword ptr[ebx+edx]

mov dword ptr [edi+edx], eax

inc edx

inc edx

inc edx

inc edx

dec ecx

dec ecx

dec ecx

dec ecx

jnz subAB

pop ebp

ret 12

Sub\_Long endp

Factorial proc

push ebp

mov ebp,esp

mov ebx, [ebp+12] ;адреса результату

mov edi, [ebp+8] ;факторіал числа

add dword ptr[ebx], 00000001h

mov ecx, edi

mov dword ptr[counter], 00000001h

factcycle:

mov dword ptr [variable], ecx

mov ecx, 56

mov esi, 0

level1cycle:

mov eax, dword ptr[ebx + esi]

mul dword ptr [counter]

mov dword ptr[level1 + esi], eax

mov dword ptr[level1 + 4 + esi], edx

add esi, 8

sub ecx, 8

jnz level1cycle

mov ecx, 48

mov esi, 0

level2cycle:

mov eax, dword ptr[ebx + 4 + esi]

mul dword ptr[counter]

mov dword ptr[level2 + 4 + esi], eax

mov dword ptr[level2 + 8 + esi], edx

add esi, 8

sub ecx, 8

jnz level2cycle

mov ecx, 56

mov esi, 0

sumlevelscycle:

mov eax, dword ptr[level1 + esi]

add cf, 0ffh

adc eax, dword ptr[level2 + esi]

mov cf, 0

adc cf, 0

mov [ebx + esi], eax

add esi, 4

sub ecx, 4

jnz sumlevelscycle

inc dword ptr [counter]

mov ecx, dword ptr [variable]

dec ecx

jnz factcycle

; mov ecx, dword ptr[counter]

; mov dword ptr[ebx], ecx

pop ebp

ret 12

Factorial endp

Multlong proc

push ebp

mov ebp,esp

mov esi, [ebp+16] ; a

mov ebx, [ebp+12] ; b

mov edi, [ebp+8] ; res

clc

mov ecx, 28

mov dword ptr[counter], 0

level0cycle:

mov eax, dword ptr[ebx]

mov dword ptr[variable], ecx

mov ecx, dword ptr[counter]

mul dword ptr[esi + ecx]

clc

add cf, 11111111b

adc dword ptr[edi + ecx], eax

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

mov cf, 00000000b

adc cf, 0

add dword ptr [counter], 4

mov ecx, dword ptr[variable]

sub ecx, 4

jnz level0cycle

clc

; mov cf, 00000000b

mov ecx, 28

mov dword ptr[counter], 0

level1cycle:

mov eax, dword ptr[ebx + 4]

mov dword ptr[variable], ecx

mov ecx, dword ptr[counter]

mul dword ptr[esi + ecx]

clc

add cf, 11111111b

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

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

mov cf, 00000000b

adc cf, 0

add dword ptr [counter], 4

mov ecx, dword ptr[variable]

sub ecx, 4

jnz level1cycle

clc

mov ecx, 28

mov dword ptr[counter], 0

level2cycle:

mov eax, dword ptr[ebx + 8]

mov dword ptr[variable], ecx

mov ecx, dword ptr[counter]

mul dword ptr[esi + ecx]

clc

add cf, 11111111b

adc dword ptr[edi + 8 + ecx], eax

adc dword ptr[edi + 12 + ecx], edx

mov cf, 00000000b

adc cf, 0

add dword ptr [counter], 4

mov ecx, dword ptr[variable]

sub ecx, 4

jnz level2cycle

clc

mov ecx, 28

mov dword ptr[counter], 0

level3cycle:

mov eax, dword ptr[ebx + 12]

mov dword ptr[variable], ecx

mov ecx, dword ptr[counter]

mul dword ptr[esi + ecx]

clc

add cf, 11111111b

adc dword ptr[edi + 12 + ecx], eax

adc dword ptr[edi + 16 + ecx], edx

mov cf, 00000000b

adc cf, 0

add dword ptr [counter], 4

mov ecx, dword ptr[variable]

sub ecx, 4

jnz level3cycle

clc

mov ecx, 28

mov dword ptr[counter], 0

level4cycle:

mov eax, dword ptr[ebx + 16]

mov dword ptr[variable], ecx

mov ecx, dword ptr[counter]

mul dword ptr[esi + ecx]

clc

add cf, 11111111b

adc dword ptr[edi + 16 + ecx], eax

adc dword ptr[edi + 20 + ecx], edx

mov cf, 00000000b

adc cf, 0

add dword ptr [counter], 4

mov ecx, dword ptr[variable]

sub ecx, 4

jnz level4cycle

clc

mov ecx, 28

mov dword ptr[counter], 0

level5cycle:

mov eax, dword ptr[ebx + 20]

mov dword ptr[variable], ecx

mov ecx, dword ptr[counter]

mul dword ptr[esi + ecx]

clc

add cf, 11111111b

adc dword ptr[edi + 20 + ecx], eax

adc dword ptr[edi + 24 + ecx], edx

mov cf, 00000000b

adc cf, 0

add dword ptr [counter], 4

mov ecx, dword ptr[variable]

sub ecx, 4

jnz level5cycle

clc

mov ecx, 28

mov dword ptr[counter], 0

level6cycle:

mov eax, dword ptr[ebx + 24]

mov dword ptr[variable], ecx

mov ecx, dword ptr[counter]

mul dword ptr[esi + ecx]

clc

add cf, 11111111b

adc dword ptr[edi + 24 + ecx], eax

adc dword ptr[edi + 28 + ecx], edx

mov cf, 00000000b

adc cf, 0

add dword ptr [counter], 4

mov ecx, dword ptr[variable]

sub ecx, 4

jnz level6cycle

pop ebp

ret 12

Multlong endp

module\_changer proc

push ebp

mov ebp,esp

mov edi, [ebp+16] ; sourse/res

mov ebx, [ebp+12] ; m

mov edx, [ebp+8] ; n

; not dword ptr [edi]

mov ecx, 544

mov esi, 0

cycle:

mov dword ptr[edi + esi], 0ffffffffh

add esi, 4

sub ecx, 4

jnz cycle

mov ecx, ebx

cmp ecx, 0

jz over

mov esi, edx

cycle2:

mov ebx, esi ; esi – це номер біту

mov edx, ebx

shr ebx, 3 ; номер байту

and edx, 07h ; позиція потрібного біту у байті

mov eax, 1

mov dword ptr [temp], ecx

mov ecx, edx

shl eax, cl ; маска 0..010..0 за умовчанням

mov ecx, dword ptr [temp]

not eax ; маска 1..101..1 для AND

and dword ptr [edi + ebx], eax

add esi, 1

sub ecx, 1

jnz cycle2

over:

pop ebp

ret 12

module\_changer endp

module\_div10 proc

push ebp

mov ebp,esp

mov esi, [ebp+20] ; sourse

mov edi, [ebp+16] ; n-bit

mov ebx, [ebp+12] ; res

mov edx, [ebp+8] ; last

mov eax, edi

shr eax, 3

mov dword ptr[counter], 0h

xor ecx, ecx

mov cl, byte ptr[esi + eax - 1]

shr cl, 4

sub edi, 4

again:

cmp cl, 10

jl little

sub cl, 10

mov dword ptr[regeax], eax

xor eax, eax

mov dword ptr[regecx], ecx

xor ecx, ecx

mov edx, edi

mov ecx, edx

shr edx, 3

and ecx, 07h

mov al, 1

shl al, cl

; not al

or byte ptr[ebx + edx], al

mov eax, dword ptr[regeax]

mov edx, [ebp+8]

mov ecx, dword ptr[regecx]

little:

dec edi

cmp edi, 0

jl exit

shl cl, 1

mov dword ptr[regeax], eax

xor eax, eax

mov dword ptr[regecx], ecx

xor ecx, ecx

mov edx, edi ; Nbit – це номер біту

mov ecx, edx

shr edx, 3 ; номер байту

and ecx, 07h ; бітова позиція = вирізаємо 3 молодші біти

mov al, 1

shl al, cl ; AL = маска вирізання біту Nbit

mov ah, byte ptr [esi + edx]

and ah, al ; результат у регістрі AH

mov edx, [ebp+8]

mov ecx, dword ptr[regecx]

cmp ah, 0

je here

add cl, 01h

here:

mov eax, dword ptr[regeax]

jmp again

exit:

mov byte ptr[edx], cl

pop ebp

ret 16

module\_div10 endp

end