НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №7

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

Виконав:

студент 2 курсу

ФІОТ гр. ІО-21

Кузьменко Володимир

Перевірив:

Порєв В.М.

Київ 2014 р.

Текст програми:

.586

.model flat, stdcall

include \masm32\include\kernel32.inc

include \masm32\include\user32.inc

includelib \masm32\lib\kernel32.lib

includelib \masm32\lib\user32.lib

include \masm32\modules\module.inc

include \masm32\modules\longop.inc

.data

Caption db "---",0

Var db 544 dup(0)

Sor db 544 dup(0)

last db 544 dup(0)

Text db 544 dup(0)

divreshex db 10 dup(0)

divresdec db 10 dup(0)

.code

main:

; mov word ptr[sor], 110101110101b

; push offset Sor

; push 16

; push offset Var

; push offset last

; call module\_div10

push offset Sor

push 50

call Factorial

push offset Var

push offset Sor

push 544

call StrDec

push offset Text

push offset Var

push 544

call StrHex\_My

invoke MessageBoxA, 0, ADDR Text, ADDR Caption, 0

mov edx, 0h ; старші біти х

mov eax, 40 ; молодші біти х

mov ebx, 5 ; константа 5

idiv bx ; ділення х на константу

mov ecx, 2 ; константа 2

shr ecx, 1 ; піднесення до потрібного степеню -m

imul ax,cx ; знаходження у

mov word ptr[divreshex], ax

mov word ptr[divreshex + 2], dx

push offset Text

push offset divreshex

push 32

call StrHex\_My

invoke MessageBoxA, 0, ADDR Text, ADDR Caption, 0

invoke ExitProcess, 0

end main

Процедури:

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

StrDec proc

push ebp

mov ebp,esp

mov eax, [ebp + 16] ; result

mov ebx, [ebp + 12] ; number

mov ecx, [ebp + 8] ; n-bit

cmp ecx, 0 ; якщо розрядність 0 - на вихід

je exit ;

xor edi, edi ;

cycle1: ;

mov esi, dword ptr[ebx + edi] ; заносимо в масив діленого число

mov dword ptr[number + edi], esi ; для конвертування

add edi, 4 ;

cmp edi, 544 ;

je again ;

jmp cycle1

again:

;-----------------------------------------------------------------------------

;---------запис залишку в молодші розряди байта--------------------------------

;-----------------------------------------------------------------------------

mov dword ptr[reggecx], ecx ;

mov dword ptr[reggedx], edx ; зберігаємо стани регістрів

mov dword ptr[reggesi], esi ;

mov dword ptr[reggedi], edi ;

push offset number ;

push [ebp + 8] ;

push offset result ; викликаємо ділення

push offset last ;

call module\_div10 ;

mov ecx, dword ptr[reggecx] ;

mov edx, dword ptr[reggedx] ; відновлюємо стани регістрів

mov esi, dword ptr[reggesi] ;

mov edi, dword ptr[reggedi] ;

xor ecx, ecx ;

xor edi, edi ;

cycle2: ;

mov esi, dword ptr[result + edi] ;

cmp esi, 0 ;

je nothing ;

inc ecx ; перевіряємо результат ділення на 0

nothing: ;

add edi, 4 ;

cmp edi, 544 ;

je next1 ;

jmp cycle2 ;

next1:

mov eax, [ebp + 16] ;

mov dl, byte ptr[last] ; заносимо залишок від ділення в результат

mov esi, dword ptr[count] ;

add byte ptr[eax + esi], dl ;

cmp ecx, 0 ; якщо результат ділення 0 - на вихід

je exit ;

xor edi, edi ;

cycle: ;

mov esi, dword ptr[result + edi] ;

mov dword ptr[result + edi], 0h ; переносимо результат ділення в масив діленого

mov dword ptr[number + edi], esi ;

add edi, 4 ;

cmp edi, 544 ;

je nexth ;

jmp cycle ;

nexth:

;-----------------------------------------------------------------------------

;---------запис залишку в старші розряди байта--------------------------------

;-----------------------------------------------------------------------------

mov dword ptr[reggecx], ecx ;

mov dword ptr[reggedx], edx ; зберігаємо стани регістрів

mov dword ptr[reggesi], esi ;

mov dword ptr[reggedi], edi ;

push offset number ;

push [ebp + 8] ;

push offset result ; викликаємо ділення

push offset last ;

call module\_div10 ;

mov ecx, dword ptr[reggecx] ;

mov edx, dword ptr[reggedx] ; відновлюємо стани регістрів

mov esi, dword ptr[reggesi] ;

mov edi, dword ptr[reggedi] ;

xor ecx, ecx ;

xor edi, edi ;

cycle2h: ;

mov esi, dword ptr[result + edi] ;

cmp esi, 0 ;

je nothingh ;

inc ecx ; перевіряємо результат ділення на 0

nothingh: ;

add edi, 4 ;

cmp edi, 544 ;

je next1h ;

jmp cycle2h ;

next1h:

mov eax, [ebp + 16] ;

mov dl, byte ptr[last] ; заносимо залишок від ділення в результат

shl dl, 4 ;

mov esi, dword ptr[count] ;

add byte ptr[eax + esi], dl ;

cmp ecx, 0 ; якщо результат ділення 0 - на вихід

je exit ;

xor edi, edi ;

cycleh: ;

mov esi, dword ptr[result + edi] ;

mov dword ptr[result + edi], 0h ; переносимо результат ділення в масив діленого

mov dword ptr[number + edi], esi ;

add edi, 4 ;

cmp edi, 544 ;

je next ;

jmp cycleh ;

next: ; повторюємо процес з початку

add dword ptr[count], 01h ;

jmp again ;

exit:

pop ebp

ret 12

StrDec endp

Висновок:

Під час виконання лабораторної роботи було створено дві процедури. Перша виконує ділення чисел довгої арифметики на 10, інша процедура користується попередньою для конвертування в десяткову систему. В результаті були закріплені навички ділення чисел в мові програмування асемблер, а також навички конвертування чисел.