Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №7

з дисципліни «Системне програмування»

на тему:

**«Програмування операцій ділення чисел»**

Виконала:

студентка 2-го курсу

групи ІО-32

Руденко Т.А.

Номер залікової книжки: 3224

Київ - 2015 р.

Мета: Навчитися програмувати на асемблері ділення чисел, вивчити

перетворення з двійкової у десяткову систему числення

Варіант 25

Лістинг:

.data

Caption1 db "Div 1" ,0

Caption2 db "Div 2" ,0

Caption3 db "Factorial" ,0

Caption4 db "Function" ,0

textBuf1 dd 50 dup(?)

intP dd 10 dup(?)

fractP dd ?

number dd 800020h,1080001h,52410401h,0A880001h,0FFFFCCCCh

decCode dd 50 dup(?)

factorial dd 16 dup(?)

solution dd ?

.code

main:

push offset number

push offset intP

push offset fractP

push 20

call DIV1\_LONGOP

push offset textBuf1

push offset intP

push 160

call StrHex\_MY

invoke MessageBoxA, 0, ADDR textBuf1, ADDR Caption1, 40h

push offset number

push offset intP

push offset fractP

push 5

call DIV2\_LONGOP

push offset textBuf1

push offset intP

push 160

call StrHex\_MY

invoke MessageBoxA, 0, ADDR textBuf1, ADDR Caption2, 40h

push offset factorial

push 30

push 50

call FACTORIAL\_LONGOP

;push offset factorial

;push offset decCode

;push 20

;push 50

;call StrToDec\_LONGOP

;invoke MessageBoxA, 0, ADDR decCode, ADDR Caption3, 40h

mov eax, 45

mov ecx, 1

call getSolution\_LONGOP

mov solution, eax

push offset solution

push offset decCode

push 8

push 8

call StrToDec\_LONGOP

invoke MessageBoxA, 0, ADDR decCode, ADDR Caption4, 40h

invoke ExitProcess, 0

end main

.code

Mul\_N\_x\_32\_LONGOP proc

push ebp

mov ebp, esp

mov esi, [ebp + 16]

mov edi, [ebp + 12]

mov ebx, [ebp + 8]

mov x, ebx

mov ecx, 5

xor ebx, ebx

@cycle1:

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

mul x

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

mov dword ptr[esi + 8 \* ebx + 4], edx

inc ebx

dec ecx

jnz @cycle1

mov ecx, 5

xor ebx, ebx

@cycle2:

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

mul x

clc

adc eax, dword ptr[esi + 8 \* ebx + 4]

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

clc

adc edx, dword ptr[esi + 8 \* ebx + 8]

mov dword ptr[esi + 8 \* ebx + 8], edx

inc ebx

dec ecx

jnz @cycle2

pop ebp

ret 12

Mul\_N\_x\_32\_LONGOP endp

DIV1\_LONGOP proc

push ebp

mov ebp, esp

mov esi, [ebp + 20]

mov edi, [ebp + 16]

mov edx, [ebp + 12]

mov eax, [ebp + 8]

mov x, eax

mov ecx, x

dec x

mov ebx, x

mov maska, 10h

mov counter, 5

mov shift, 4

mov flag, 1

mov al, byte ptr[esi + ebx]

and al, 0F0h

shr al, 4

@cycle:

mov ebx, ecx

dec ebx

push ecx

mov ecx, counter

@cycleInner:

mov ah, byte ptr[esi + ebx]

and ah, maska

push ecx

mov cl, shift

shr ah, cl

pop ecx

cmp flag, 0

jne @nShift

shl al, 1

add al, ah

@nShift:

mov flag, 0

cmp al, TEN

jae @ABOVE

jmp @EXIT

@ABOVE:

push edx

mov dl, maska

add dl, byte ptr[edi + ebx]

mov byte ptr[edi + ebx], dl

sub al, TEN

pop edx

@EXIT:

shr maska, 1

dec shift

dec ecx

jnz @cycleInner

mov flag, 0

mov shift, 7

mov maska, 80h

mov counter, 8

pop ecx

dec ecx

jnz @cycle

mov eax, dword ptr[edi + 4]

pop ebp

ret 16

DIV1\_LONGOP endp

DIV2\_LONGOP proc

push ebp

mov ebp, esp

mov esi, [ebp + 20]

mov edi, [ebp + 16]

mov ebx, [ebp + 12]

mov eax, [ebp + 8]

mov x, eax

push ebx

xor edx, edx

mov ecx, x

dec x

mov ebx,x

@cycle :

push ecx

mov ecx, 10

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

div ecx

mov fractionalPart, dl

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

dec ebx

pop ecx

dec ecx

jnz @cycle

pop ebx

mov al, fractionalPart

mov byte ptr[ebx], al

pop ebp

ret 16

DIV2\_LONGOP endp

StrToDec\_LONGOP proc

push ebp

mov ebp, esp

mov esi, [ebp + 20]

mov edi, [ebp + 16]

mov eax, [ebp + 12]

mov x1, eax

mov eax, [ebp + 8]

mov x2, eax

xor ecx, ecx

xor ebx, ebx

@cycle:

push ecx

push edi

push esi

push offset buf

push offset decCode

push x1

call DIV2\_LONGOP

pop edi

mov ebx, b

mov al, byte ptr[decCode]

add al, 48

mov byte ptr[edi + ebx], al

xor ecx, ecx

@cycleInner:

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

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

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

inc ecx

cmp ecx, x1

jl @cycleInner

pop ecx

inc ecx

inc b

cmp ecx, x2

jl @cycle

mov ebx, x2

mov eax, x2

xor edx, edx

div two

mov x2, eax

dec ebx

xor ecx, ecx

@cycle1:

mov al, byte ptr[edi + ecx]

mov ah, byte ptr[edi + ebx]

mov byte ptr[edi + ecx], ah

mov byte ptr[edi + ebx], al

dec ebx

inc ecx

cmp ecx, x2

jl @cycle1

pop ebp

ret 16

StrToDec\_LONGOP endp

FACTORIAL\_LONGOP proc

push ebp

mov ebp, esp

mov esi, [ebp + 16]

mov edi, [ebp + 12]

mov ebx, [ebp + 8]

@cycle:

push edi

push ebx

push esi

push offset bufForFactorial

push x3

call Mul\_N\_x\_32\_LONGOP

pop ebx

pop edi

inc x3

mov ecx, 0

@cycleIn:

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

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

inc ecx

cmp ecx, ebx

jl @cycleIn

dec edi

jnz @cycle

pop ebp

ret 8

FACTORIAL\_LONGOP endp

getSolution\_LONGOP proc

mov ebx, 11

xor edx, edx

div ebx

shr eax, cl

ret

getSolution\_LONGOP endp

end