МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Чувашский государственный университет имени И.Н. Ульянова»

Факультет информатики и вычислительной техники

Кафедра автоматики и управления в технических системах

Лабораторная работа №3

По дисциплине “ Микропроцессорные средства и системы”

на тему:

«Основы языка Ассемблера»

Вариант 7

Выполнил: ст. гр. ИВТ-11-21

Еремеев Сергей А.

Проверил: Григорьев А. В.

Чебоксары 2024

**Цели работы**: изучить теоретические основы языка Ассемблера, научиться писать программы, реализующие ввод, вывод и обработку числовой информации с использованием команд языка Ассемблера.

**Теоретическая часть**

Ассемблер (от англ. assembler — сборщик) — компьютерная программа, компилятор исходного текста программы, написанной на языке Ассемблера, в программу на машинном языке.

Общие положения о программировании в Windows, используемые в языке Ассемблера для ОС Windows:

* Программирование в Windows основывается на использовании функций API (Application Program Interface, т.е. интерфейс программного приложения). Их количество достигает двух тысяч. Программа на ассемблерев значительной степени будет состоять из таких вызовов. Все взаимодействие с внешними устройствами и ресурсами операционной системы будет происходить посредством таких функций.
* Главным элементом программы в среде Windows является окно. Для каждого окна определяется своя процедура обработки сообщений.
* Окно может содержать элементы управления: кнопки, списки, окна редактирования и др. Эти элементы, по сути, также являются окнами, но обладающими особыми свойствами. События, происходящие с этими элементами (и самим окном), приводят к приходу сообщений в процедуру окна.
* Операционная система Windows использует линейную модель памяти, соответственно всю память можно рассматривать как один сегмент. Для программиста на языке Ассемблера это означает, что адрес любой ячейки памяти будет определяться содержимым одного 32-битного регистра.

**Задание**

1. Изучить теоретические основы языка Ассемблера.

2. Запустить программу fasmw.exe.

3. Открыть в программе файл lab3.asm.

4. Внимательно изучить текст программы lab3.asm по имеющимся в

ней комментариям.

5. Запустить программу на выполнение (клавиша F9).

6. Модифицировать текст программы в соответствии с вариантом

задания.

Вариант 7: Реализовать вычисление следующего выражения:

**Выполнение**

Основной код программы для вывода и ввода значений взят из файла примера lab3.asm, код же для реализации поставленной задачи выделен полужирным шрифтом, а также все внесенные изменения. Комментарии выделены дополнительно курсивом.

**Код программы**:

format PE GUI 4.0 *;специальный формат исполнимых файлов*

*;для Windows — формат PE (Portable Executable)*

entry start

include 'win32a.inc'*; подключаем библиотеку win32a.inc*

**ID\_A = 104 ; *дескриптор***

**ID\_B = 103**

**ID\_C = 101**

**ID\_D = 102;**

section '.data' data readable writeable *; блок описания*

*;используемых в программе данных*

**a rb 10 *; зарезервируем 10 байт для ввода a***

**b rb 10 *; зарезервируем 10 байт для ввода b***

**c rb 10 *; зарезервируем 10 байт для ввода c***

**d rb 10 *; зарезервируем 10 байт для ввода d***

**caption db 'Answer',0 *;заголовок для окна с ответом***

**message db 'Y := ',0 *; шаблон текста ответа***

sys dd 10 *; переменная для перевода из цифрового*

*;формата в текстовый в десятичной системе счисления*

section '.code' code readable executable *; блок*

*;описания кода программы*

start:

xor eax,eax *;ЕАХ=0*

invoke DialogBoxParam,eax,37,HWND\_DESKTOP,DialogProc1,0 *;открываем окно используя процедуру DialogProc*

or eax,eax ;*проверим eax=0*

jz exit ;*если да(нажата кнопка*

*;cancel), то переход к метке exit*

**lea esi, [c] *;загрузим в регистр esi адрес***

***;введенной строки с***

**call StrToInt *;преобразование введенной***

***;строки в число в регистре EAX***

**mov ebx, eax *;сохраним значение c в ebx***

**lea esi, [d] *;загрузим в регистр esi адрес ;введенной строки d***

**call StrToInt ;*Преобразование введенной ;строки в число в регистре EAX***

**mul ebx *; eax = c\*d***

**mov edi, eax ; *edi = c\*d***

**lea esi, [a] ;*загрузим в регистр esi адрес***

***;введенной строки a***

**call StrToInt ;*преобразование введенной***

***;строки в число в регистре EAX***

**mov ebx, eax *;сохраним значение a в ebx***

**lea esi, [b] ;*загрузим в регистр esi адрес***

***;введенной строки b***

**call StrToInt ;*преобразование введенной***

***;строки в число в регистре EAX***

**mov esi, eax ; esi = b**

**mov eax, edi; eax = c \* d**

**div ebx ; eax = c\*d / a**

**sub ebx, esi ; ebx = a - b**

**add eax, ebx;**

**lea esi, [message+5] ;*ESI=адрес строки***

***;message+5 (для вывода результата в шаблон ответа)***

**call IntToStr ;*вызов процедуры перевода***

***;целого числа из регистра EAX в строку по адресу из регистра ESI***

invoke MessageBox,HWND\_DESKTOP,message,caption,MB\_OK ;*api функция вывода сообщения на экран*

exit:

invoke ExitProcess,0 *; api функция для*

*;завершения программы*

proc IntToStr *; eax - число, esi - адрес строки для*

*;вывода*

pushad

mov ebx,10

xor ecx,ecx

*;------ загоняем в стек, начиная с младших разрядов*

start1:

cmp eax,0

je end1

xor edx,edx

div ebx

or dl,30h

push edx

inc ecx

jmp start1

end1:

*;------ читаем из стека в буфер*

start3:

cmp ecx,0

je end3

pop eax

mov [esi],al

inc esi

dec ecx

jmp start3

end3:

popad

ret

endp

;*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

proc StrToInt *;(esi-строка):eax-число,esi-остаток*

*;строки,ecx-прочитано цифр enter*

xor eax,eax ; *Результат*

xor ecx,ecx ; *Счетчик числа символов*

;mov [sys],10 ; *Определяет систему счисления*

.l:

mul [sys]

;or edx,edx

;jnz .exit

; *считываем символ из строки*

movzx edx,byte [esi]

sub dl,30h

jc .err

cmp dl,9h

*; если dl меньше 9h, то переход дальше*

jle .next

sub dl,7h

.next:

cmp dl,byte [sys]

jnc .err

add eax,edx

jc .err2

; *Увеличиваем ecx (число символов на единицу)*

inc ecx

; *Смещаем указатель на следующий символ*

inc esi

; *Повторяем все со следующим символом*

jmp .l

.err2:

sub eax,edx

.err:

xor edx,edx

.exit:

div [sys]

ret

endp

proc DialogProc1,hwnddlg,msg,wparam,lparam

push ebx esi edi

cmp [msg],WM\_INITDIALOG

je processed

cmp [msg],WM\_COMMAND

je wmcommand

cmp [msg],WM\_CLOSE

je wmclose

xor eax,eax

jmp finish

wmcommand:

cmp [wparam],BN\_CLICKED shl 16 + IDCANCEL

je wmclose

cmp [wparam],BN\_CLICKED shl 16 + IDOK

jne processed

**invoke GetDlgItemText,[hwnddlg],ID\_A,a,3 *; api***

***;функция получения 3-х байт текста в переменную c из***

***;объекта имеющего идентификатор ID\_A***

**invoke GetDlgItemText,[hwnddlg],ID\_B,b,3 *; api***

***;функция получения 3-х байт текста в переменную d из***

***;объекта имеющего идентификатор ID\_B***

**invoke GetDlgItemText,[hwnddlg],ID\_C,c,3 *; api***

***;функция получения 3-х байт текста в переменную c из***

***;объекта имеющего идентификатор ID\_C***

**invoke GetDlgItemText,[hwnddlg],ID\_D,d,3 *; api***

***;функция получения 3-х байт текста в переменную d из объекта***

***;имеющего идентификатор ID\_D***

topmost\_ok:

invoke EndDialog,[hwnddlg],1 ; *api функция завершения*

*;диалога с введенными аргументами из формы (параметр 1)*

jmp processed

wmclose:

invoke EndDialog,[hwnddlg],0 ; *api функция аварийного*

*;завершения диалога нажатием на кнопку cancel (параметр 0)*

processed:

mov eax,1

finish:

pop edi esi ebx

ret

endp

section '.idata' import data readable writeable

library kernel,'KERNEL32.DLL',\

user,'USER32.DLL'

import kernel,\

GetModuleHandle,'GetModuleHandleA',\

ExitProcess,'ExitProcess'

import user,\

DialogBoxParam,'DialogBoxParamA',\

CheckRadioButton,'CheckRadioButton',\

GetDlgItemText,'GetDlgItemTextA',\

IsDlgButtonChecked,'IsDlgButtonChecked',\

MessageBox,'MessageBoxA',\

EndDialog,'EndDialog'

section '.rsrc' resource data readable

directory RT\_DIALOG,dialogs

resource dialogs,\

37,LANG\_RUSSIAN,demonstration \

dialog demonstration,'Laboratornay rabota 1',70,70,190,100,WS\_CAPTION ; *задаем заголовок и размеры окна*

;*приложения*

**dialogitem 'STATIC','Vvedite a:',1,10,10,170,8,WS\_VISIBLE**

**dialogitem 'EDIT','',ID\_A,110,9,15,13,WS\_VISIBLE+WS\_BORDER+WS\_TABSTOP**

**dialogitem 'STATIC','Vvedite b:',1,10,20,170,8,WS\_VISIBLE**

**dialogitem 'EDIT','',ID\_B,110,19,15,13,WS\_VISIBLE+WS\_BORDER+WS\_TABSTOP**

**dialogitem 'STATIC','Vvedite c:',1,10,30,170,8,WS\_VISIBLE *;задаем заголовок строки ее координаты, устанавливаем атрибут***

**;*"видимая"***

**dialogitem 'EDIT','',ID\_C,110,29,15,13,WS\_VISIBLE+WS\_BORDER+WS\_TABSTOP**

**dialogitem 'STATIC','Vvedite d:',1,10,40,170,8,WS\_VISIBLE**

**dialogitem 'EDIT','',ID\_D,110,39,15,13,WS\_VISIBLE+WS\_BORDER+WS\_TABSTOP**

**dialogitem** 'BUTTON','OK',IDOK,85,75,45,15,WS\_VISIBLE+WS\_TABSTOP+BS\_DEFPUSHBUTTON ;

dialogitem 'BUTTON','C&ancel',IDCANCEL,135,75,45,15,WS\_VISIBLE+WS\_TABSTOP

enddialog

**Результат выполнения программы:**

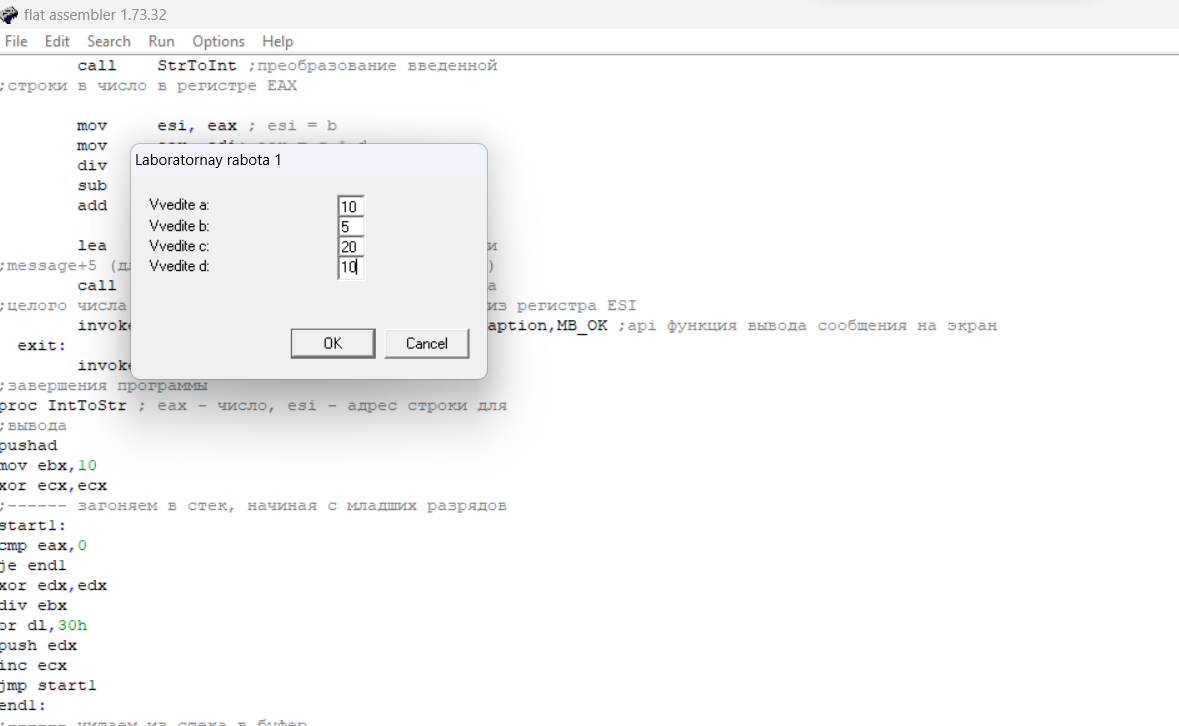


Рисунок 1 – Ввод значений в диалоговое окно

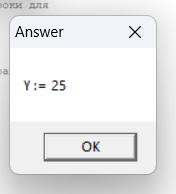


Рисунок 2 – Вывод результата работы программы

При нажатии на кнопку ОК без введенных значений программа прекращает свою работу. При нажатии на кнопку Cancel программа прекращает свою работу.

**Вывод**: изучила теоретические основы языка Ассемблера, научилась писать программы, реализующие ввод, вывод и обработку числовой информации с использованием команд языка Ассемблера.