МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ 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 №5

По дисциплине “ Микропроцессорные средства и системы”

на тему:

«Реализация логических элементов средствами языка Ассемблера»

Вариант 7

Выполнил: ст. гр. ИВТ-11-21

Еремеев Сергей А.

Проверил: Григорьев А. В.

Чебоксары 2024

**Цель работы**: рассмотреть возможности языка Ассемблера в аспекте изучения логических элементов, реализующих элементарные функции алгебры логики.

**Теоретическая часть**

К логическим операциям относятся: логическое умножение (AND), логическое сложение (OR), исключающее «ИЛИ» (XOR) и отрицание (NOT). Все эти инструкции изменяют регистр признаков.

Команда AND выполняет логическое умножение двух операндов — ol и о2. Результат сохраняется в операнде ol. Типы операндов такие же, как у команды ADD: операнды могут быть 8, 16 или 32-битными регистрами, адресами памяти или непосредственными значениями. Формат команды:

AND ol, о2.

Следующий пример вычисляет логическое «И» логической единицы и логического нуля (1 AND 0).

mov а1,1 ; AL = one

mov bl,0 ; BL = zero

and al,bl ; AL = AL and BL = 0.

Команда OR выполняет логическое сложение двух операндов — ol и о2. Результат сохраняется в операнде ol. Типы операндов такие же, как у команды AND. Формат команды:

OR ol, о2.

Команда XOR вычисляет «исключающее ИЛИ» операндов ol и о2. Результат сохраняется в ol. Типы операндов такие же, как у предыдущих инструкций. Формат команды:

XOR ol, о2.

Команда NOT используется для инверсии отдельных битов единственного операнда, который может быть регистром или памятью. Соответственно команда может быть записана в трех различных форматах: NOT Регистр/память8, NOT Регистр/памятьl6, NOT Регистр/память32. Например:

mov al,00000010b ;AL = 2

mov bl,al ;BL = 2

not al ;после этой операции al= 11111101b.

**Постановка задачи**

1. Изучить теоретические основы языка Ассемблера

2. Запустить программу fasmw.exe.

3. Открыть в программе файл lab5.asm.

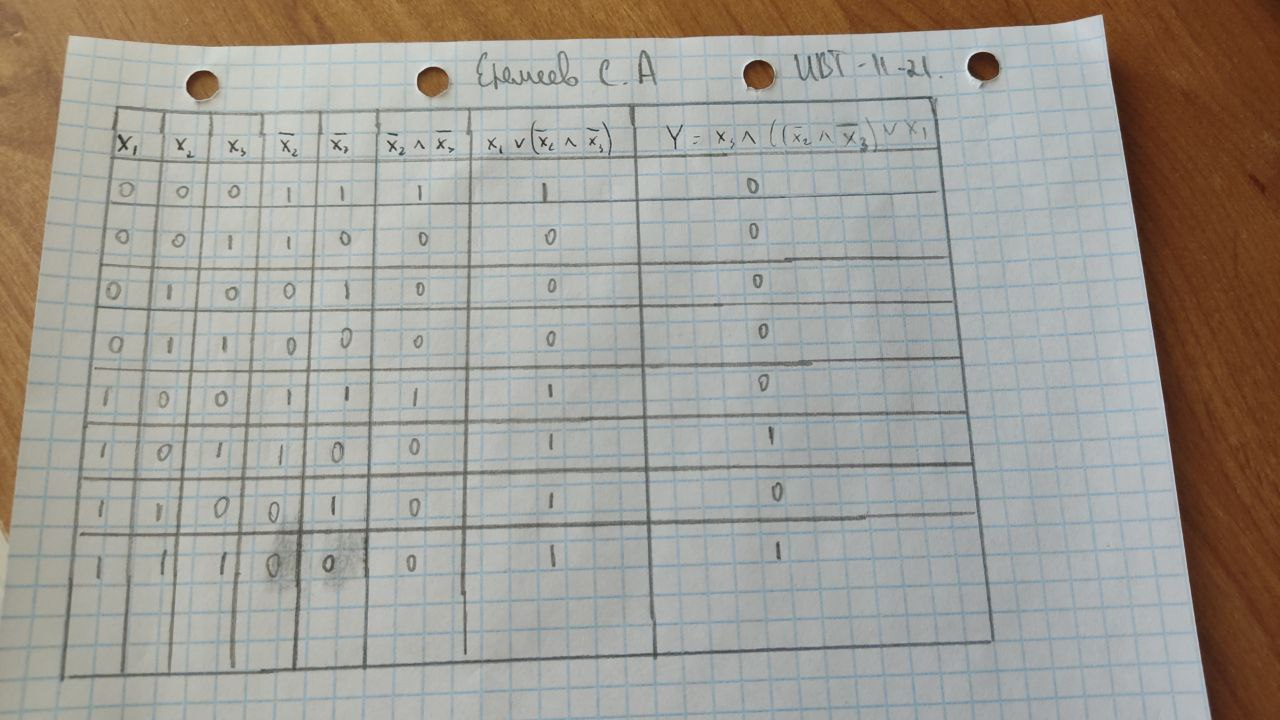
4. Внимательно изучить текст программы lab5.asm по имеющимся в ней комментариям.

5. Запустить программу на выполнение.

6. Модифицировать текст программы для вычисления значения логической функции Y по введенным значениям аргументов в соответствии с вариантом задания, предварительно построив таблицу истинности.

Вариант 1.

**Выполнение**



Основной код программы для вывода и ввода значений взят из файла примера lab5.asm, код же для реализации поставленной задачи выделен полужирным шрифтом, а также все внесенные изменения.

**Код программы**:

; Пример выполнения лабораторной работы №5 по ассемблеру

; \_\_\_ \_\_\_

; Y = X3 /\((X2 /\ X3) \/ X1)

;

format PE GUI 4.0

entry start

include 'win32a.inc'

ID\_x1 = 101

ID\_x2 = 102

ID\_x3 = 103

section '.data' data readable writeable

x1 rb 2

x2 rb 2

x3 rb 2

caption\_er db 'Ошибка!',0

message\_er db 'Введены не верные параметры x1, x2, x3',0

caption db 'Ответ',0

message db 'Y('

x\_1 db 0,','

x\_2 db 0,','

x\_3 db 0,')='

y db 0,0

section '.code' code readable executable

start:

xor eax,eax

invoke DialogBoxParam,eax,37,HWND\_DESKTOP,DialogProc,0

or eax,eax

jz exit

mov al,[x1]

mov [x\_1],al

sub al,30h

jz ok\_1

cmp al,1

jz ok\_1

jmp error

ok\_1: mov bl,[x2]

mov [x\_2],bl

sub bl,30h

jz ok\_2

cmp bl,1

jz ok\_2

jmp error

ok\_2: mov ah,[x3]

mov [x\_3],ah

sub ah,30h

jz ok\_3

cmp ah,1

jz ok\_3

jmp error

ok\_3:

;al=x1, bl=x2, ah=x3

;вычисление логической функции Y по заданной в задании функции

mov bh,ah ;bh=ah (bh=x3)

and bl,ah ;bl=bl/\ah (bl=(x2/\x3))

or bl,al ;bl=bl/al (bl=(x2/\x3)/x1)

mov bh,bh ;bh=bh/\bl (bh=x3/\((x2/\x3)/x1))

add bh,30h

mov [y],bh

invoke MessageBox,HWND\_DESKTOP,message,caption,MB\_OK

jmp exit

error: invoke MessageBox,HWND\_DESKTOP,message\_er,caption\_er,MB\_OK

jmp exit

exit:

invoke ExitProcess,0

proc DialogProc,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\_x1,x1,2 ; api функция получения 2-х байт текста в переменную x1 из объекта имеющего идентификатор ID\_x1

invoke GetDlgItemText,[hwnddlg],ID\_x2,x2,2 ; api функция получения 2-х байт текста в переменную x2 из объекта имеющего идентификатор ID\_x2

invoke GetDlgItemText,[hwnddlg],ID\_x3,x3,2 ; api функция получения 2-х байт текста в переменную x3 из объекта имеющего идентификатор ID\_x3

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 3',70,70,120,65,WS\_CAPTION ; задаем заголовок и размеры окна приложения

dialogitem 'STATIC','x1:',1,10,10,170,8,WS\_VISIBLE ; задаем заголовок строки ее координаты, устанавливаем атрибут "видимая"

dialogitem 'EDIT','',ID\_x1,20,9,10,13,WS\_VISIBLE+WS\_BORDER+WS\_TABSTOP ;

dialogitem 'STATIC','x2:',1,10,20,170,8,WS\_VISIBLE ;

dialogitem 'EDIT','',ID\_x2,20,19,10,13,WS\_VISIBLE+WS\_BORDER+WS\_TABSTOP ;

dialogitem 'STATIC','x3:',1,10,30,170,8,WS\_VISIBLE ;

dialogitem 'EDIT','',ID\_x3,20,29,10,13,WS\_VISIBLE+WS\_BORDER+WS\_TABSTOP ;

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

dialogitem 'BUTTON','C&ancel',IDCANCEL,65,45,45,15,WS\_VISIBLE+WS\_TABSTOP ;

enddialog

**Результат выполнения программы:**

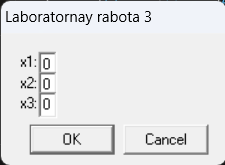


Рисунок 1 – Ввод значения в диалоговое окно

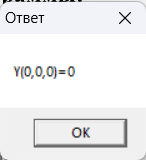


Рисунок 2 – Вывод результата работы программы

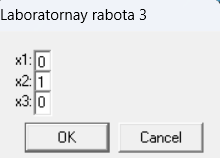


Рисунок 3 – Ввод значения в диалоговое окно

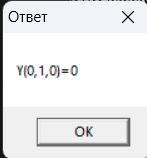


Рисунок 4 – Результат выполнения программы

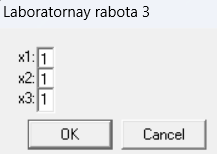


Рисунок 5 – Ввод значения в диалоговое окно

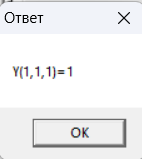


Рисунок 6 – Результат выполнения программы

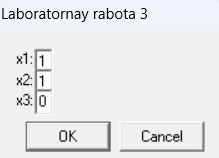


Рисунок 7 – Ввод значения в диалоговое окно

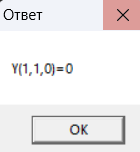


Рисунок 8 – Результат выполнения программы

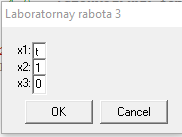


Рисунок 9 – Ввод случайного символа вместо значения

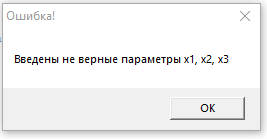


Рисунок 10 – Результат при неправильном вводе

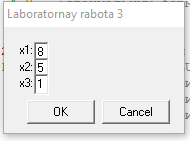


Рисунок 11 – Ввод вместо двоичных десятичных чисел

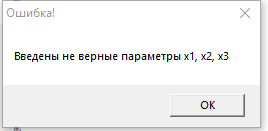


Рисунок 12 – Результат при вводе вместо двоичных десятичных чисел

При нажатии на кнопку ОК без введенных значений программа как и в случае неправильного ввода чисел возвращает сообщение об ошибке ввода. При нажатии на кнопку Cancel программа прекращает свою работу.

**Вывод**: рассмотрела возможности языка Ассемблера в аспекте изучения логических элементов, реализующих элементарные функции алгебры логики.