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 №4

По дисциплине “ Микропроцессорные средства и системы”

на тему:

«Использование условных операторов в языке Ассемблера»

Вариант 1

Выполнил: ст. гр. ИВТ-11-21

Еремеев С. А.

Проверил: Григорьев А. В.

Чебоксары 2024

**Цели работы**: изучить теоретические основы организации регистра флага микропроцессора, команды условных и безусловных переходов языка Ассемблера, научиться писать программы, реализующие ввод, вывод и обработку числовой информации в соответствии с поставленными условиями на языке Ассемблера.

**Теоретическая часть**

Управляющие конструкции. Команды СМР и TEST используются для сравнения двух операндов. Операндами могут быть как регистры, так и адреса памяти, размер операнда — 8, 16 или 32 бита.

СМР o1, о2.

Команда СМР — это сокращение от «compare», «сравнить». Она работает подобно SUB: операнд о2 вычитается из ol. Результат нигде не сохраняется, команда просто изменяет регистр признаков. Команда СМР может использоваться как для сравнения целых беззнаковых чисел, так и для сравнения чисел со знаком.

Команда TEST работает подобно СМР, но вместо вычитания она вычисляет поразрядное «И» операндов. Результат инструкции — измененные флаги регистра признаков. Можно использовать TEST для проверки значений отдельных битов в массиве битов.

Самый простой способ изменить последовательность выполнения команд заключается в использовании команды jmp - команды безусловного перехода. Она перезаписывает указатель команд (регистр IP или CS), что заставляет процессор «переключиться» на выполнение команды по указанному адресу. Формат команды таков:

JMP [тип\_перехода] операнд.

Команда JMP — аналог конструкции GOTO, которая используется в высокоуровневых языках программирования. Название команды объясняет ее действие, а именно «jump», «переход». Команде нужно передать один обязательный операнд — адрес в памяти, с которого процессор должен продолжить выполнение программы. Операнд может быть указан явно (непосредственное значение адреса) или быть регистром общего назначения, в который загружен требуемый адрес. Но новичкам я никогда не рекомендовал бы это делать: язык Ассемблера, подобно языкам программирования высокого уровня, позволяет обозначить адрес назначения при помощи метки.

В зависимости от «расстояния» переходы бывают трех типов: короткие (short), ближние (near) и дальние (far). Тип перехода задается необязательным параметром инструкции jmp. Если тип не задан, по умолчанию используется тип near.

Другой способ изменения последовательности выполнения команд заключается в использовании команды условного перехода.В языке Ассемблера имеется множество команд условного перехода. Имена этих команд различаются в зависимости от условия перехода. Условие состоит из значений одного или нескольких флагов в регистре признаков. Работают эти команды одинаково: если условие истинно, выполняется переход на указанную метку, если нет, то процессор продолжит выполнять программу со следующей команды.

Общий формат команд условного перехода следующий:

Jx метка\_назначения.

**Задание**

Постановка задачи

1. Изучить теоретические основы языка Ассемблера.

2. Запустить программу fasmw.exe.

3. Открыть в программе файл lab4.asm.

4. Внимательно изучить текст программы lab4.asm по имеющимся в ней

комментариям.

5. Запустить программу на выполнение.

6. Модифицировать текст программы в соответствии с вариантом

задания.

Вариант 1: 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\_x = 101 ; дескриптор для x

section '.data' data readable writeable ; блок описания

;используемых в программе данных

x rb 10 ; зарезервируем 10 байт для ввода x

caption db 'Ответ',0 ;заголовок для окна с ответом

error db 'Ошибка ввода!!!',0

message db 'F(x)= ',0 ; шаблон текста ответа

sys dd 10 ; для перевода из цифрового формата в

;текстовый в десятичной системе счисления

chislo dd ?

section '.code' code readable executable ; блок

;описания кода программы

start:

xor eax,eax ;EAX=0

invoke DialogBoxParam,eax,37,HWND\_DESKTOP,DialogProc,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, [x] ;загрузим в регистр esi адрес

;введенной строки x

call StrToInt ;преобразование введенной

;строки x в число в регистре EAX

or ecx,ecx ;ecx=0?

jnz norm ;если нет, то ввод данных

;корректный

invoke MessageBox,HWND\_DESKTOP,error,caption,MB\_OK ;вывод

;сообщения об ошибке

jmp exit

; вычислим ответ

norm: cmp eax,0 ;сравним наше число x с 0

jl var1 ; Если EAX<0 то переход к метке var1

cmp eax,3 ;сравним наше число x с 3

jg var1 ; Если EAX>3 то переход к метке var1

mov ebx,eax ; ebx = eax (ebx = x)

mul ebx ; eax = eax \* ebx (eax = x \* x)

jmp vivod ; переходим к выводу:результата

var1: mov eax,4 ; eax = 4

vivod: lea esi, [message+6] ;ESI=адрес строки

;message+6 (для вывода результата в шаблон ответа)

call IntToStr

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

exit:

invoke ExitProcess,0

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

.l:

mul [sys]

; 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 .errs

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

jmp .exit

.errs:

xor ecx,ecx

ret

.exit:

div [sys]

ret

endp

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\_x,x,10

topmost\_ok:

invoke EndDialog,[hwnddlg],1

jmp processed

wmclose:

invoke EndDialog,[hwnddlg],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 2',100,100,110,60,WS\_CAPTION ; задаем заголовок,

;координаты верхнего левого угла окна на экране, ширина

;окна, высота окна

dialogitem 'STATIC','X=',1,10,10,70,8,WS\_VISIBLE ; задаем заголовок строки ее координаты, устанавливаем

;атрибут "видимая"

dialogitem 'EDIT','',ID\_x,22,8,15,13,WS\_VISIBLE+WS\_BORDER+WS\_TABSTOP ;

dialogitem 'BUTTON','OK',IDOK,10,35,45,15,WS\_VISIBLE+WS\_TABSTOP+BS\_DEFPUSHBUTTON ;

dialogitem 'BUTTON','C&ancel',IDCANCEL,60,35,45,15,WS\_VISIBLE+WS\_TABSTOP ;

enddialog

**Результат выполнения программы:**

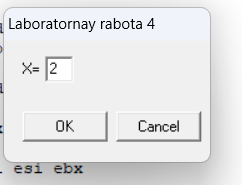


Рисунок 1 – Ввод значения x в диалоговое окно

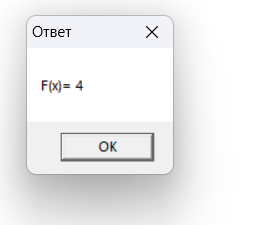


Рисунок 2 – Вывод результата работы программы

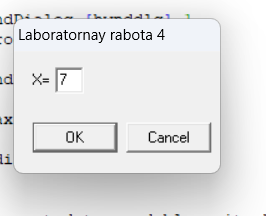


Рисунок 3 – Ввод значения x

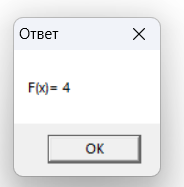


Рисунок 4 – Результат выполнения программы

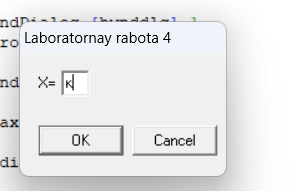


Рисунок 5 – Ввод случайного символа вместо значения x

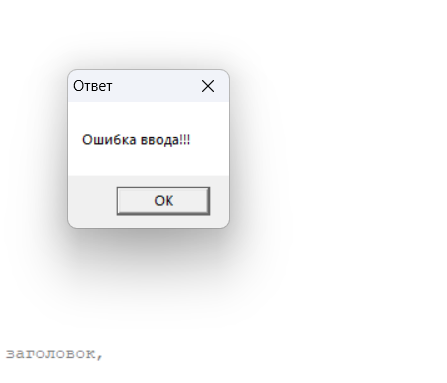


Рисунок 6 – Результат при неправильном вводе

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

**Вывод**: изучила теоретические основы организации регистра флага микропроцессора, команды условных и безусловных переходов языка Ассемблера, научилась писать программы, реализующие ввод, вывод и обработку числовой информации в соответствии с поставленными условиями на языке Ассемблера.