МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ РАДИОЭЛЕКТРОНИКИ

Кафедра СТ

Отчет

по лабораторной работе № 2

по дисциплине: «Программирование(системное)»

Выполнил: Проверил:

ст. гр. АКТСИу-17-1 Коваленко А. И.

Черкашин В.А.

Харьков 2018

1 Изучение языка Ассемблер

* 1. Цель работы

Изучение основных команд и директив языка программирования  
Ассемблер.

* 1. Задания

Написать программу, которая вычисляет значение функции (длиной слово)



если заданы значения коэффициентов a, b, c, d, y и значение x . Коэффициенты  
a, b, c, d, y – являются элементами массива. Реализовать ввод и вывод значений  
с помощью консоли. На консоль должна выводиться информация о студенте,  
выполняющем лабораторную работу и выбранный вариант с заданием и  
ответом. Промежуточные результаты вычислений переменных сохранять в  
сегменте данных в качестве глобальных переменных.

Вывести с помощью API функции «MessageBox» в десятичной и  
шестнадцатеричной системах счисления, получаемые при расчетах:  
- значения промежуточных результатов вычислений;  
- адреса промежуточных результатов в сегменте данных;  
- размер промежуточных результатов в байтах

**Задание1**Выполнить задание (пп.1.2.), используя прямое обращение к  
глобальным переменным в сегменте данных.

**Задание2**Выполнить задание (пп.1.2), используя адреса основных и  
промежуточных переменных, сохраненных в сегменте данных. При  
выполнении задания использовать косвенную адресацию с помощью регистров общего назначения

**Задание3**Выполнить задание (пп.1.2), используя стек (команды Ассемблера  
PUSH и POP). Все значения переменных занести в стек и использовать их,  
извлекая из стека

**Задание4**Выполнить задание (пп.1.4.1.1), используя стек. С помощью команды  
Ассемблера PUSH занести значения переменных в стек. Далее копировать  
значения из стека с помощью регистра EBP (указателя базы кадра стека). К  
глобальным переменным в сегменте данных, также как и в задании 2,  
обращаться по адресам, используя косвенную адресацию с помощью регистров общего назначения

* 1. Листинг задания 1

;Черкашин В.А. Лаба 2 задание 1 Вариант 11

.486

.model flat, STDCALL

option casemap:none

include \masm32\include\windows.inc

include \masm32\include\user32.inc

include \masm32\include\kernel32.inc

include \masm32\include\msvcrt.inc

includelib \masm32\lib\user32.lib

includelib \masm32\lib\kernel32.lib

includelib \masm32\lib\msvcrt.lib

;=========================================

.data

; Формула f(x)=a\*x^2 - a\*b\*d/y+d

X dw ?

A dw ?

B dw ?

D dw ?

Y dw ?

XmX SWORD ?

AmXmX SWORD ?

AmB SWORD ?

AmBmD SWORD ?

YpD SWORD ?

Z SWORD ?

F SWORD ?

addrXmX dd ?

addrAmXmX dd ?

addrAmB dd ?

addrAmBmD dd ?

addrYpD dd ?

addrZ dd ?

outHandle dd ?

inHandle dd ?

ConsoleTitle db "Лаба 2", 0

TitleMB db "Промежуточные результаты",0

TextBuf db 50 dup (?)

format db "Промежуточные результаты",0dh,0ah,"XmX=%d",0dh,0ah, "AmXmX=%d",0dh,0ah, "AmBmD=%d",0dh,0ah, "YpD=%d",0dh,0ah,"Z=%d",0

format1 db "Адресса промежуточных результатов",0dh,0ah,"addrXmX=%d",0dh,0ah, "addrAmXmX=%d",0dh,0ah, "addrAmBmD=%d",0dh,0ah, "addrYpD=%d",0dh,0ah,"addrZ=%d",0

format2 db "Размер промежуточных результатов в байтах",0dh,0ah,"sizeXmX=%d",0dh,0ah, "sizeAmXmX=%d",0dh,0ah, "sizeAmBmD=%d",0dh,0ah, "sizeYpD=%d",0dh,0ah,"sizeZ=%d",0

result db "Конечный результат %d", 0

namberW dd ?

namberR dd ?

buf db "Введите X",0

buf1 db "Введите A",0

buf2 db "Введите B",0

buf3 db "Введите D",0

buf4 db "Введите Y",0

StartText db "Выполнил студент группы АКТСИу 17-2 Черкашин В.А Вариант №11",0dh,0ah,0

Task db "Формула f(x)=a\*x^2 - a\*b\*d/y+d, где X=4, A=-8, B=6, D=8, Y=12",0dh,0ah,0

NumberBuf db 16 dup (?)

;==========================================

.code

start:

;==Запрос Консоли, Установка Title, Получения handle

invoke AllocConsole

invoke GetStdHandle, STD\_OUTPUT\_HANDLE

MOV outHandle, EAX

invoke GetStdHandle,

STD\_INPUT\_HANDLE

MOV inHandle, EAX

;===Перекодируем строк =================

invoke CharToOem,

ADDR ConsoleTitle,

ADDR ConsoleTitle

invoke SetConsoleTitle, ADDR ConsoleTitle

invoke CharToOem,

ADDR StartText,

ADDR StartText

invoke CharToOem,

ADDR Task,

ADDR Task

invoke CharToOem,

ADDR buf,

ADDR buf

invoke CharToOem,

ADDR buf1,

ADDR buf1

invoke CharToOem,

ADDR buf2,

ADDR buf2

invoke CharToOem,

ADDR buf3,

ADDR buf3

invoke CharToOem,

ADDR buf4,

ADDR buf4

;===Заносим значения=====================

invoke WriteConsoleA,

outHandle,

ADDR StartText,

SIZEOF StartText,

ADDR namberW,

NULL

invoke WriteConsoleA,

outHandle,

ADDR Task,

SIZEOF Task,

ADDR namberW,

NULL

invoke WriteConsoleA,

outHandle,

ADDR buf,

SIZEOF buf,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV X, AX

invoke WriteConsoleA,

outHandle,

ADDR buf1,

SIZEOF buf1,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV A, AX

invoke WriteConsoleA,

outHandle,

ADDR buf2,

SIZEOF buf2,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV B, AX

invoke WriteConsoleA,

outHandle,

ADDR buf3,

SIZEOF buf3,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV D, AX

invoke WriteConsoleA,

outHandle,

ADDR buf4,

SIZEOF buf4,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV Y, AX

;=========================================

MOV AX, X

MUL X

MOV XmX , AX

;============================================

MOV AX, A

MUL XmX

MOV AmXmX, AX

;============================================

MOV AX, A

MUL B

MOV AmB, AX

;============================================

MOV AX, D

MUL AmB

MOV AmBmD, AX

;============================================

MOV AX, Y

ADD AX, D

MOV YpD, AX

;============================================

MOV AX,AmBmD ;Делимое

MOV BL, byte ptr YpD ;Делитель

IDIV BL ;AL=0Ah (частное) AH=06h (остаток)

CBW ; Конвертнули

MOV Z, AX

;============================================

MOV AX, AmXmX

SUB AX, Z

MOV F, AX

;============================================

invoke wsprintf, addr TextBuf,addr result, F

invoke CharToOem,

ADDR TextBuf,

ADDR TextBuf

invoke WriteConsoleA,

outHandle,

ADDR TextBuf,

SIZEOF TextBuf,

ADDR namberW,

NULL

invoke wsprintf, addr TextBuf,addr format, XmX, AmXmX, AmBmD, YpD, Z

invoke MessageBox, 0,

ADDR TextBuf,

ADDR TitleMB,

MB\_OK

lea eax,XmX

mov addrXmX,eax

lea eax,AmXmX

mov addrAmXmX ,eax

lea eax,AmB

mov addrAmB ,eax

lea eax,AmBmD

mov addrAmBmD ,eax

lea eax,YpD

mov addrYpD ,eax

lea eax,Z

mov addrZ ,eax

invoke wsprintf, addr TextBuf,addr format1, addrXmX, addrAmXmX, addrAmBmD, addrYpD, addrZ

invoke MessageBox, 0,

ADDR TextBuf,

ADDR TitleMB,

MB\_OK

invoke wsprintf, addr TextBuf,addr format2, sizeof XmX, sizeof AmXmX, sizeof AmBmD, sizeof YpD, sizeof Z

invoke MessageBox, 0,

ADDR TextBuf,

ADDR TitleMB,

MB\_OK

invoke ExitProcess, 0

end start

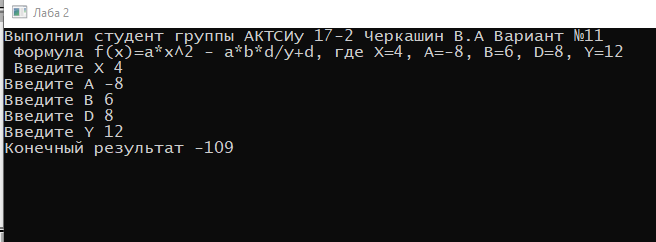
****

Рис 1 – Результаты вывода первого задания

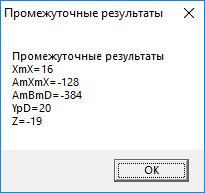


Рис 2 – Промежуточные результаты

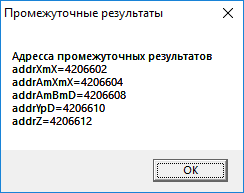


Рис 3 – Адреса промежуточных результатов

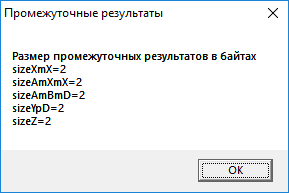


Рис 4 – Размер промежуточных результатов в байтах

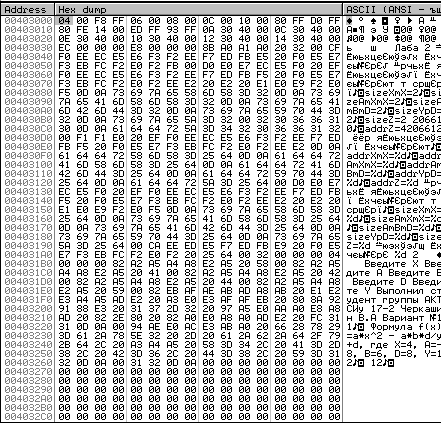


Рис 5 – Дамп сегмента данных задания 1

1.3.2 Листинг программы 2

;Черкашин В.А. Лаба 2 задание 2 Вариант 11

.486

.model flat, STDCALL

option casemap:none

include \masm32\include\windows.inc

include \masm32\include\user32.inc

include \masm32\include\kernel32.inc

include \masm32\include\msvcrt.inc

includelib \masm32\lib\user32.lib

includelib \masm32\lib\kernel32.lib

includelib \masm32\lib\msvcrt.lib

;=========================================

.data

; Формула f(x)=a\*x^2 - a\*b\*d/y+d

X dw ?

A dw ?

B dw ?

D dw ?

Y dw ?

addrX dd ?

addrA dd ?

addrB dd ?

addrD dd ?

addrY dd ?

XmX SWORD ?

AmXmX SWORD ?

AmB SWORD ?

AmBmD SWORD ?

YpD SWORD ?

Z SWORD ?

F SWORD ?

addrF DD ?

addrXmX dd ?

addrAmXmX dd ?

addrAmB dd ?

addrAmBmD dd ?

addrYpD dd ?

addrZ dd ?

outHandle dd ?

inHandle dd ?

ConsoleTitle db "Лаба 2", 0

TitleMB db "Промежуточные результаты",0

TextBuf db 50 dup (?)

format db "Промежуточные результаты",0dh,0ah,"XmX=%d",0dh,0ah, "AmXmX=%d",0dh,0ah, "AmBmD=%d",0dh,0ah, "YpD=%d",0dh,0ah,"Z=%d",0

format1 db "Адресса промежуточных результатов",0dh,0ah,"addrXmX=%d",0dh,0ah, "addrAmXmX=%d",0dh,0ah, "addrAmBmD=%d",0dh,0ah, "addrYpD=%d",0dh,0ah,"addrZ=%d",0

format2 db "Размер промежуточных результатов в байтах",0dh,0ah,"sizeXmX=%d",0dh,0ah, "sizeAmXmX=%d",0dh,0ah, "sizeAmBmD=%d",0dh,0ah, "sizeYpD=%d",0dh,0ah,"sizeZ=%d",0

result db "Конечный результат %d", 0

namberW dd ?

namberR dd ?

buf db "Введите X",0

buf1 db "Введите A",0

buf2 db "Введите B",0

buf3 db "Введите D",0

buf4 db "Введите Y",0

StartText db "Выполнил студент группы АКТСИу 17-2 Черкашин В.А Вариант №11 Косвенная адресация",0dh,0ah,0

Task db "Формула f(x)=a\*x^2 - a\*b\*d/y+d, где X=4, A=-8, B=6, D=8, Y=12",0dh,0ah,0

NumberBuf db 16 dup (?)

;==========================================

.code

start:

;==Запрос Консоли, Установка Title, Получения handle

invoke AllocConsole

invoke GetStdHandle, STD\_OUTPUT\_HANDLE

MOV outHandle, EAX

invoke GetStdHandle,

STD\_INPUT\_HANDLE

MOV inHandle, EAX

;===Перекодируем строк =================

invoke CharToOem,

ADDR ConsoleTitle,

ADDR ConsoleTitle

invoke SetConsoleTitle, ADDR ConsoleTitle

invoke CharToOem,

ADDR StartText,

ADDR StartText

invoke CharToOem,

ADDR Task,

ADDR Task

invoke CharToOem,

ADDR buf,

ADDR buf

invoke CharToOem,

ADDR buf1,

ADDR buf1

invoke CharToOem,

ADDR buf2,

ADDR buf2

invoke CharToOem,

ADDR buf3,

ADDR buf3

invoke CharToOem,

ADDR buf4,

ADDR buf4

;===Заносим значения=====================

invoke WriteConsoleA,

outHandle,

ADDR StartText,

SIZEOF StartText,

ADDR namberW,

NULL

invoke WriteConsoleA,

outHandle,

ADDR Task,

SIZEOF Task,

ADDR namberW,

NULL

invoke WriteConsoleA,

outHandle,

ADDR buf,

SIZEOF buf,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV X, AX

invoke WriteConsoleA,

outHandle,

ADDR buf1,

SIZEOF buf1,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV A, AX

invoke WriteConsoleA,

outHandle,

ADDR buf2,

SIZEOF buf2,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV B, AX

invoke WriteConsoleA,

outHandle,

ADDR buf3,

SIZEOF buf3,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV D, AX

invoke WriteConsoleA,

outHandle,

ADDR buf4,

SIZEOF buf4,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV Y, AX

;====Инициализация переменных адресами=======

lea eax, X

mov addrX, eax

lea eax, A

mov addrA, eax

lea eax, B

mov addrB, eax

lea eax, D

mov addrD, eax

lea eax, Y

mov addrY, eax

lea eax, F

MOV addrF,eax

lea eax,XmX

mov addrXmX,eax

lea eax,AmXmX

mov addrAmXmX ,eax

lea eax,AmB

mov addrAmB ,eax

lea eax,AmBmD

mov addrAmBmD ,eax

lea eax,YpD

mov addrYpD ,eax

lea eax,Z

mov addrZ ,eax

;==============(XmX)==========================

MOV EBX, addrX

MOV EAX, 0

MOV AX, [EBX]

MUL word ptr [EBX]

MOV EBX, addrXmX

MOV [EBX] , AX

;============================================

MOV EBX, addrA

MOV EAX, 0

MOV AX, [EBX]

MOV EBX, addrXmX

MUL word ptr [EBX]

MOV EBX, addrAmXmX

MOV [EBX], AX

;============================================

MOV EBX, addrA

MOV EAX, 0

MOV AX, [EBX]

MOV EBX, addrB

MUL word ptr [EBX]

MOV EBX, addrAmB

MOV [EBX], AX

;============================================

MOV EBX, addrD

MOV EAX, 0

MOV AX, [EBX]

MOV EBX, addrAmB

MUL word ptr [EBX]

MOV EBX, addrAmBmD

MOV [EBX], AX

;============================================

MOV EBX, addrY

MOV ECX, addrD

MOV EAX, 0

MOV AX, [EBX]

ADD AX, [ECX]

MOV EBX, addrYpD

MOV [EBX], AX

;============================================

MOV EBX, addrAmBmD

MOV AX, sword ptr [EBX]

MOV EBX, addrYpD

IDIV sbyte ptr [EBX]

CBW

MOV EBX, addrZ

MOV [EBX], AX

;============================================

MOV EBX, addrAmXmX

MOV AX, word ptr [EBX]

MOV EBX, addrZ

SUB AX, word ptr [EBX]

MOV EBX, addrF

MOV [EBX], AX

;============================================

invoke wsprintf, addr TextBuf,addr result, F

invoke CharToOem,

ADDR TextBuf,

ADDR TextBuf

invoke WriteConsoleA,

outHandle,

ADDR TextBuf,

SIZEOF TextBuf,

ADDR namberW,

NULL

invoke wsprintf, addr TextBuf,addr format, XmX, AmXmX, AmBmD, YpD, Z

invoke MessageBox, 0,

ADDR TextBuf,

ADDR TitleMB,

MB\_OK

invoke wsprintf, addr TextBuf,addr format1, addrXmX, addrAmXmX, addrAmBmD, addrYpD, addrZ

invoke MessageBox, 0,

ADDR TextBuf,

ADDR TitleMB,

MB\_OK

invoke wsprintf, addr TextBuf,addr format2, sizeof XmX, sizeof AmXmX, sizeof AmBmD, sizeof YpD, sizeof Z

invoke MessageBox, 0,

ADDR TextBuf,

ADDR TitleMB,

MB\_OK

invoke ExitProcess, 0

end start

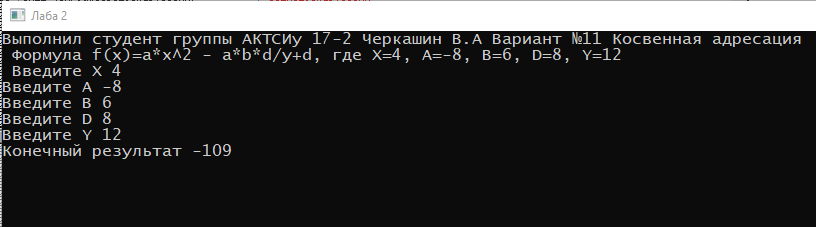


Рис 6 – Вывод на консоль программы 2

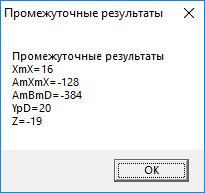


Рис 7 – Промежуточные результаты

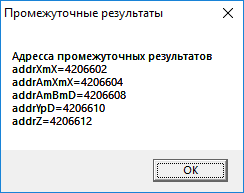


Рис 8 – Адреса промежуточных результатов

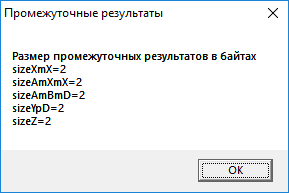


Рис 9 – Размер промежуточных результатов в байтах

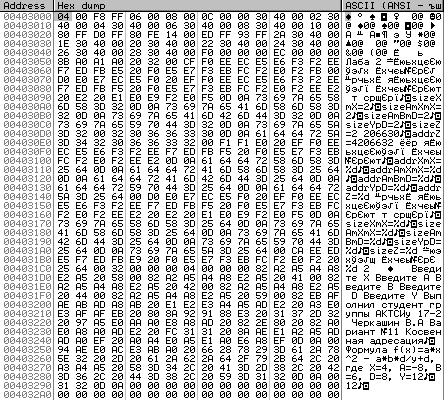


Рис 10 – Дамп сегмента данных программы 2

1.3.3 Листинг программы 3

;Черкашин В.А. Лаба 2 задание 3 Вариант 11

.486

.model flat, STDCALL

option casemap:none

include \masm32\include\windows.inc

include \masm32\include\user32.inc

include \masm32\include\kernel32.inc

include \masm32\include\msvcrt.inc

includelib \masm32\lib\user32.lib

includelib \masm32\lib\kernel32.lib

includelib \masm32\lib\msvcrt.lib

;=========================================

.data

; Формула f(x)=a\*x^2 - a\*b\*d/y+d

X dw ?

A dw ?

B dw ?

D dw ?

Y dw ?

XmX SWORD ?

AmXmX SWORD ?

AmB SWORD ?

AmBmD SWORD ?

YpD SWORD ?

Z SWORD ?

F SWORD ?

addrXmX dd ?

addrAmXmX dd ?

addrAmB dd ?

addrAmBmD dd ?

addrYpD dd ?

addrZ dd ?

outHandle dd ?

inHandle dd ?

ConsoleTitle db "Лаба 2", 0

TitleMB db "Промежуточные результаты",0

TextBuf db 50 dup (?)

format db "Промежуточные результаты",0dh,0ah,"XmX=%d",0dh,0ah, "AmXmX=%d",0dh,0ah, "AmBmD=%d",0dh,0ah, "YpD=%d",0dh,0ah,"Z=%d",0

format1 db "Адресса промежуточных результатов",0dh,0ah,"addrXmX=%d",0dh,0ah, "addrAmXmX=%d",0dh,0ah, "addrAmBmD=%d",0dh,0ah, "addrYpD=%d",0dh,0ah,"addrZ=%d",0

format2 db "Размер промежуточных результатов в байтах",0dh,0ah,"sizeXmX=%d",0dh,0ah, "sizeAmXmX=%d",0dh,0ah, "sizeAmBmD=%d",0dh,0ah, "sizeYpD=%d",0dh,0ah,"sizeZ=%d",0

result db "Конечный результат %d", 0

namberW dd ?

namberR dd ?

buf db "Введите X",0

buf1 db "Введите A",0

buf2 db "Введите B",0

buf3 db "Введите D",0

buf4 db "Введите Y",0

StartText db "Выполнил студент группы АКТСИу 17-2 Черкашин В.А Вариант №11 Используя стек",0dh,0ah,0

Task db "Формула f(x)=a\*x^2 - a\*b\*d/y+d, где X=4, A=-8, B=6, D=8, Y=12",0dh,0ah,0

NumberBuf db 16 dup (?)

;==========================================

.code

start:

;==Запрос Консоли, Установка Title, Получения handle

invoke AllocConsole

invoke GetStdHandle, STD\_OUTPUT\_HANDLE

MOV outHandle, EAX

invoke GetStdHandle,

STD\_INPUT\_HANDLE

MOV inHandle, EAX

;===Перекодируем строк =================

invoke CharToOem,

ADDR ConsoleTitle,

ADDR ConsoleTitle

invoke SetConsoleTitle, ADDR ConsoleTitle

invoke CharToOem,

ADDR StartText,

ADDR StartText

invoke CharToOem,

ADDR Task,

ADDR Task

invoke CharToOem,

ADDR buf,

ADDR buf

invoke CharToOem,

ADDR buf1,

ADDR buf1

invoke CharToOem,

ADDR buf2,

ADDR buf2

invoke CharToOem,

ADDR buf3,

ADDR buf3

invoke CharToOem,

ADDR buf4,

ADDR buf4

;===Заносим значения=====================

invoke WriteConsoleA,

outHandle,

ADDR StartText,

SIZEOF StartText,

ADDR namberW,

NULL

invoke WriteConsoleA,

outHandle,

ADDR Task,

SIZEOF Task,

ADDR namberW,

NULL

invoke WriteConsoleA,

outHandle,

ADDR buf,

SIZEOF buf,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV X, AX

invoke WriteConsoleA,

outHandle,

ADDR buf1,

SIZEOF buf1,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV A, AX

invoke WriteConsoleA,

outHandle,

ADDR buf2,

SIZEOF buf2,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV B, AX

invoke WriteConsoleA,

outHandle,

ADDR buf3,

SIZEOF buf3,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV D, AX

invoke WriteConsoleA,

outHandle,

ADDR buf4,

SIZEOF buf4,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV Y, AX

;=============================================

PUSH X

PUSH X

POP AX

POP BX

MUL BX

MOV XmX , AX

;============================================

PUSH XmX

PUSH A

POP AX

POP BX

MUL BX

MOV AmXmX, AX

;============================================

PUSH A

PUSH B

POP AX

POP BX

MUL BX

MOV AmB, AX

;============================================

PUSH D

PUSH AmB

POP AX

POP BX

MUL BX

MOV AmBmD, AX

;============================================

PUSH D

PUSH Y

POP AX

POP BX

ADD AX, BX

MOV YpD, AX

;============================================

PUSH AmBmD

PUSH YpD

POP BX

POP AX

IDIV BL ;AL=0Ah (частное) AH=06h (остаток)

CBW ; Конвертнули

MOV Z, AX

;============================================

PUSH AmXmX

PUSH Z

POP BX

POP AX

SUB AX, BX

MOV F, AX

;============================================

invoke wsprintf, addr TextBuf,addr result, F

invoke CharToOem,

ADDR TextBuf,

ADDR TextBuf

invoke WriteConsoleA,

outHandle,

ADDR TextBuf,

SIZEOF TextBuf,

ADDR namberW,

NULL

invoke wsprintf, addr TextBuf,addr format, XmX, AmXmX, AmBmD, YpD, Z

invoke MessageBox, 0,

ADDR TextBuf,

ADDR TitleMB,

MB\_OK

lea eax,XmX

mov addrXmX,eax

lea eax,AmXmX

mov addrAmXmX ,eax

lea eax,AmB

mov addrAmB ,eax

lea eax,AmBmD

mov addrAmBmD ,eax

lea eax,YpD

mov addrYpD ,eax

lea eax,Z

mov addrZ ,eax

invoke wsprintf, addr TextBuf,addr format1, addrXmX, addrAmXmX, addrAmBmD, addrYpD, addrZ

invoke MessageBox, 0,

ADDR TextBuf,

ADDR TitleMB,

MB\_OK

invoke wsprintf, addr TextBuf,addr format2, sizeof XmX, sizeof AmXmX, sizeof AmBmD, sizeof YpD, sizeof Z

invoke MessageBox, 0,

ADDR TextBuf,

ADDR TitleMB,

MB\_OK

invoke ExitProcess, 0

end start

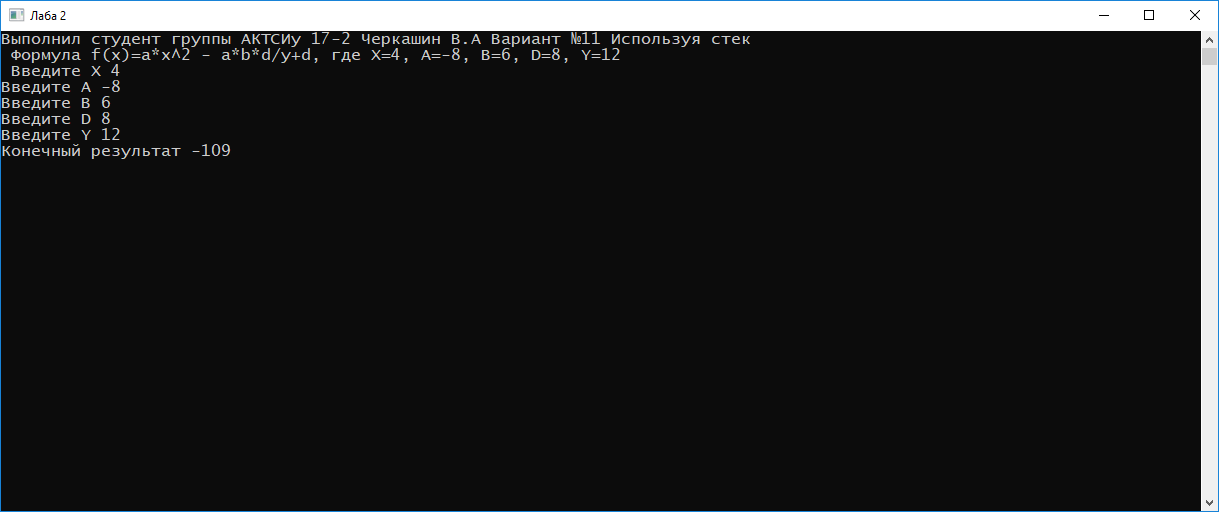


Рис 11 – Вывод на консоль программы 3

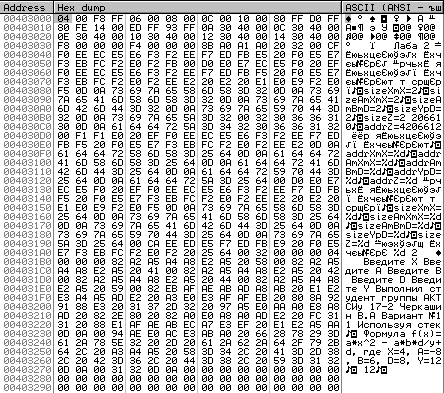


Рис 12 – Дамп сегмента данных для 3 программы

MB тот же что и в программе 2

1.3.4 Листинг программы 4

;Черкашин В.А. Лаба 2 задание 4 Вариант 11

.486

.model flat, STDCALL

option casemap:none

include \masm32\include\windows.inc

include \masm32\include\user32.inc

include \masm32\include\kernel32.inc

include \masm32\include\msvcrt.inc

includelib \masm32\lib\user32.lib

includelib \masm32\lib\kernel32.lib

includelib \masm32\lib\msvcrt.lib

;=========================================

.data

; Формула f(x)=a\*x^2 - a\*b\*d/y+d

X dw ?

A dw ?

B dw ?

D dw ?

Y dw ?

addrX dd ?

addrA dd ?

addrB dd ?

addrD dd ?

addrY dd ?

XmX SWORD ?

AmXmX SWORD ?

AmB SWORD ?

AmBmD SWORD ?

YpD SWORD ?

Z SWORD ?

F SWORD ?

addrF DD ?

addrXmX dd ?

addrAmXmX dd ?

addrAmB dd ?

addrAmBmD dd ?

addrYpD dd ?

addrZ dd ?

outHandle dd ?

inHandle dd ?

ConsoleTitle db "Лаба 2", 0

TitleMB db "Промежуточные результаты",0

TextBuf db 50 dup (?)

format db "Промежуточные результаты",0dh,0ah,"XmX=%d",0dh,0ah, "AmXmX=%d",0dh,0ah, "AmBmD=%d",0dh,0ah, "YpD=%d",0dh,0ah,"Z=%d",0

format1 db "Адресса промежуточных результатов",0dh,0ah,"addrXmX=%d",0dh,0ah, "addrAmXmX=%d",0dh,0ah, "addrAmBmD=%d",0dh,0ah, "addrYpD=%d",0dh,0ah,"addrZ=%d",0

format2 db "Размер промежуточных результатов в байтах",0dh,0ah,"sizeXmX=%d",0dh,0ah, "sizeAmXmX=%d",0dh,0ah, "sizeAmBmD=%d",0dh,0ah, "sizeYpD=%d",0dh,0ah,"sizeZ=%d",0

formatH db "Промежуточные результаты H",0dh,0ah,"XmX=%0X",0dh,0ah, "AmXmX=%0X",0dh,0ah, "AmBmD=%0X",0dh,0ah, "YpD=%0X",0dh,0ah,"Z=%0X",0

result db "Конечный результат %d", 0

namberW dd ?

namberR dd ?

buf db "Введите X",0

buf1 db "Введите A",0

buf2 db "Введите B",0

buf3 db "Введите D",0

buf4 db "Введите Y",0

StartText db "Выполнил студент группы АКТСИу 17-2 Черкашин В.А Вариант №11 Стек + Регистр EBP",0dh,0ah,0

Task db "Формула f(x)=a\*x^2 - a\*b\*d/y+d, где X=4, A=-8, B=6, D=8, Y=12",0dh,0ah,0

NumberBuf db 16 dup (?)

;==========================================

.code

start:

;==Запрос Консоли, Установка Title, Получения handle

invoke AllocConsole

invoke GetStdHandle, STD\_OUTPUT\_HANDLE

MOV outHandle, EAX

invoke GetStdHandle,

STD\_INPUT\_HANDLE

MOV inHandle, EAX

;===Перекодируем строк =================

invoke CharToOem,

ADDR ConsoleTitle,

ADDR ConsoleTitle

invoke SetConsoleTitle, ADDR ConsoleTitle

invoke CharToOem,

ADDR StartText,

ADDR StartText

invoke CharToOem,

ADDR Task,

ADDR Task

invoke CharToOem,

ADDR buf,

ADDR buf

invoke CharToOem,

ADDR buf1,

ADDR buf1

invoke CharToOem,

ADDR buf2,

ADDR buf2

invoke CharToOem,

ADDR buf3,

ADDR buf3

invoke CharToOem,

ADDR buf4,

ADDR buf4

;===Заносим значения=====================

invoke WriteConsoleA,

outHandle,

ADDR StartText,

SIZEOF StartText,

ADDR namberW,

NULL

invoke WriteConsoleA,

outHandle,

ADDR Task,

SIZEOF Task,

ADDR namberW,

NULL

invoke WriteConsoleA,

outHandle,

ADDR buf,

SIZEOF buf,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV X, AX

invoke WriteConsoleA,

outHandle,

ADDR buf1,

SIZEOF buf1,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV A, AX

invoke WriteConsoleA,

outHandle,

ADDR buf2,

SIZEOF buf2,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV B, AX

invoke WriteConsoleA,

outHandle,

ADDR buf3,

SIZEOF buf3,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV D, AX

invoke WriteConsoleA,

outHandle,

ADDR buf4,

SIZEOF buf4,

ADDR namberW,

NULL

invoke ReadConsole,

inHandle,

ADDR NumberBuf,

SIZEOF NumberBuf,

ADDR namberR,

NULL

invoke crt\_atoi,

addr NumberBuf

MOV Y, AX

;====Инициализация переменных адресами=======

lea eax, X

mov addrX, eax

lea eax, A

mov addrA, eax

lea eax, B

mov addrB, eax

lea eax, D

mov addrD, eax

lea eax, Y

mov addrY, eax

lea eax, F

MOV addrF,eax

lea eax,XmX

mov addrXmX,eax

lea eax,AmXmX

mov addrAmXmX ,eax

lea eax,AmB

mov addrAmB ,eax

lea eax,AmBmD

mov addrAmBmD ,eax

lea eax,YpD

mov addrYpD ,eax

lea eax,Z

mov addrZ ,eax

;=============================================

PUSH X ;+8

PUSH A ;+6

PUSH B ;+4

PUSH D ;+2

PUSH Y ;EBP

MOV EBP, ESP

;==============(XmX)==========================

MOV AX, [EBP+8]

MUL sword ptr [EBP+8]

MOV EBX, addrXmX

MOV [EBX], AX

;============================================

MOV AX, [EBP+6]

MOV EBX, addrXmX

MUL word ptr [EBX]

MOV EBX, addrAmXmX

MOV [EBX], AX

;============================================

MOV AX, [EBP+6]

MUL word ptr [EBP+4]

MOV EBX, addrAmB

MOV [EBX], AX

;============================================

MOV AX, [EBP+2]

MOV EBX, addrAmB

MUL word ptr [EBX]

MOV EBX, addrAmBmD

MOV [EBX], AX

;============================================

MOV AX, [EBP]

ADD AX, [EBP+2]

MOV EBX, addrYpD

MOV [EBX], AX

;============================================

MOV EBX, addrAmBmD

MOV AX, sword ptr [EBX]

MOV EBX, addrYpD

IDIV sbyte ptr [EBX]

CBW

MOV EBX, addrZ

MOV [EBX], AX

;============================================

MOV EBX, addrAmXmX

MOV AX, word ptr [EBX]

MOV EBX, addrZ

SUB AX, word ptr [EBX]

MOV EBX, addrF

MOV [EBX], AX

;============================================

invoke wsprintf, addr TextBuf,addr result, F

invoke CharToOem,

ADDR TextBuf,

ADDR TextBuf

invoke WriteConsoleA,

outHandle,

ADDR TextBuf,

SIZEOF TextBuf,

ADDR namberW,

NULL

invoke wsprintf, addr TextBuf,addr format, XmX, AmXmX, AmBmD, YpD, Z

invoke MessageBox, 0,

ADDR TextBuf,

ADDR TitleMB,

MB\_OK

invoke wsprintf, addr TextBuf,addr formatH, XmX, AmXmX, AmBmD, YpD, Z

invoke MessageBox, 0,

ADDR TextBuf,

ADDR TitleMB,

MB\_OK

invoke wsprintf, addr TextBuf,addr format1, addrXmX, addrAmXmX, addrAmBmD, addrYpD, addrZ

invoke MessageBox, 0,

ADDR TextBuf,

ADDR TitleMB,

MB\_OK

invoke wsprintf, addr TextBuf,addr format2, sizeof XmX, sizeof AmXmX, sizeof AmBmD, sizeof YpD, sizeof Z

invoke MessageBox, 0,

ADDR TextBuf,

ADDR TitleMB,

MB\_OK

invoke ExitProcess, 0

end start

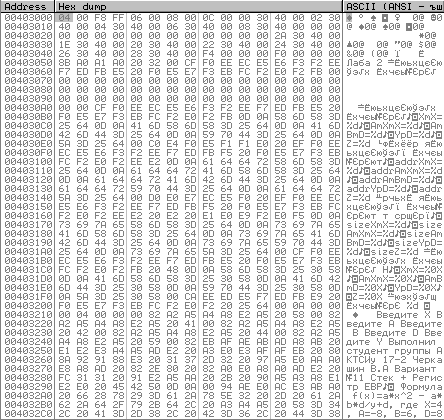


Рис 13 – Дамп программы 4

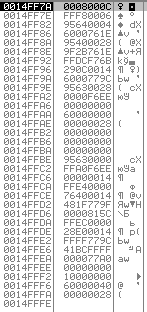


Рис 14 – сегмент стека для программы 4

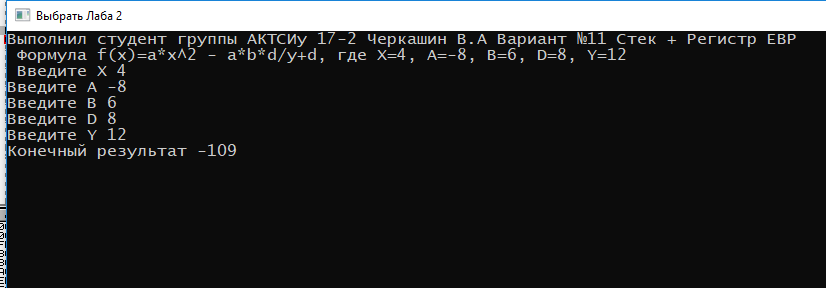


Рис 15 – Вывод программы 4

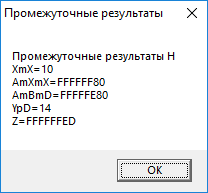


Рис 16 – Промежуточные результаты в H формате

Окна MB такие же как в программе 2

Выводы: В ходе лабораторной работы были изучен принцип адресации памяти на языке ассемблер, и разработаны программы с использование прямой, косвенной адресации, был изучен оператор [выражение] и [Регистр] , так же были изучены арифметические мнемоники на языке ассемблера.