|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Лабораторные работы № 1 | | |
| по дисциплине «Языки программирования и методы трансляции» | | |
| **Основные элементы языка ассемблера** | | |
|  | | |
|  | Бригада 2 | Глазунов Кирилл |
| Вариант 2 | Дудкина Мария |
| Группа ПМ-21 | Осиновский Артём |
|  |  |
|  |  |
|  |  |
| Преподаватель | Бортников Никита Евгеньевич |
|  |  |
| Новосибирск, 2025 | | |

1. **Цель**

Изучить и приобрести практические навыки работы с основными командами языка Ассемблера, функциями ввода-вывода, регистрами и символьными данными.

1. **Задание**

По предложенному преподавателем варианту разработать программу на языке Ассемблера, решающую поставленную задачу:

1) ввод с клавиатуры числа в заданной системе счисления (система счисления А);

2) вывод введенного числа в десятичной системе счисления;

3) вычисление значения полинома вида ax2 + bx + c (в предположении, что результат вычислений не приводит к переполнению регистров);

4) вывод результата в заданной системе счисления (система счисления Б);

5) вывод результата в десятичной системе счисления.

Все промежуточные данные должны сохраняться в памяти. При выводе результата не использовать функцию wsprintfA. **Приглашение к вводу числа, а также вывод результатов на экран должны сопровождаться понятными пользователю текстовыми сообщениями.**

**Вариант 2:**

Система счисления А: 8

Система счисления Б: 16

Полином:

1. **Алгоритмы**

**1) Перевод числа из восьмеричной системы в десятичную**

1. Считать строку с клавиатуры
2. Проверить первый символ: если это минус - запомнить, что число отрицательное, и не учитывать этот символ дальше.
3. Установить результат равным нулю.
4. Для каждого следующего символа:
   * Проверить, что он от '0' до '7'. Если нет - выдать сообщение об ошибке и завершить работу.
   * Преобразовать символ в цифру. (вычитанием ‘0’)
   * Умножить накопленный результат на восемь и прибавить эту цифру.
5. Если число было с минусом - изменить знак результата на отрицательный.
6. Полученное значение считать числом в десятичной системе.

**2) Перевод числа из десятичной системы в шестнадцатеричную**

1. Если число равно нулю - результатом будет строка «0».
2. Если число отрицательное - запомнить знак и заменить число на его модуль.
3. Взять указатель на конец буфера вывода.
4. Пока число не равно нулю:
   * Разделить число на шестнадцать, запомнить остаток.
   * Найти символ цифры по остатку (0…F).
   * Сдвинуть указатель на один байт назад и записать туда символ цифры.
5. Если исходное число было отрицательным - добавить минус в начало результата.
6. Полученную строку считать числом в шестнадцатеричной системе.
7. **Текст программы**

; ЛР1, вариант 2 (A=8, Б=16; a=3, b=-12, c=3)

; Ввод числа в восьмеричной системе, вывод в десятичной.

; Вычисление F(x)=3\*x^2-12\*x+3. Вывод F в 16-ричной и в десятичной.

.386

.MODEL FLAT, STDCALL

OPTION CASEMAP:NONE

; внешние функции Windows API

EXTERN GetStdHandle@4: PROC

EXTERN WriteConsoleA@20: PROC

EXTERN CharToOemA@8: PROC

EXTERN ReadConsoleA@20: PROC

EXTERN ExitProcess@4: PROC

EXTERN lstrlenA@4: PROC

.DATA

; сообщения

STR\_INVITE DB "Введите число в восьмеричной системе: ",0

STR\_DEC DB "Число в десятичной системе: ",0

STR\_FHEX DB "F(x)=3\*x^2-12\*x+3 в шестнадцатеричной системе: ",0

STR\_FDEC DB "F(x)=3\*x^2-12\*x+3 в десятичной системе: ",0

STR\_NL DB 13,10,0

STR\_ERR DB 13,10,"Ошибка: вводите только цифры 0..7!",13,10,0

DIGITS DB "0123456789ABCDEF",0 ; таблица символов для HEX

; дескрипторы

DIN DD ? ; stdin

DOUT DD ? ; stdout

; буферы и переменные

BUF DB 200 DUP(?) ; буфер ввода

LENS DD ? ; длина строки

OUTBUF DB 64 DUP(?) ; буфер для вывода числа

OUTLEN DD ? ; длина строки в OUTBUF

X\_DEC DD ? ; введённое число в десятичной

F\_VAL DD ? ; значение полинома

Acoef DD 3

Bcoef DD -12

Ccoef DD 3

X\_SQR DD ? ; x^2

AX2 DD ? ; a\*x^2

BX1 DD ? ; b\*x

.CODE

MAIN PROC

; перекодируем все сообщения в OEM

MOV EAX, OFFSET STR\_INVITE

PUSH EAX

PUSH EAX

CALL CharToOemA@8

MOV EAX, OFFSET STR\_DEC

PUSH EAX

PUSH EAX

CALL CharToOemA@8

MOV EAX, OFFSET STR\_FHEX

PUSH EAX

PUSH EAX

CALL CharToOemA@8

MOV EAX, OFFSET STR\_FDEC

PUSH EAX

PUSH EAX

CALL CharToOemA@8

MOV EAX, OFFSET STR\_NL

PUSH EAX

PUSH EAX

CALL CharToOemA@8

MOV EAX, OFFSET STR\_ERR

PUSH EAX

PUSH EAX

CALL CharToOemA@8

; получаем дескрипторы ввода/вывода

PUSH -10

CALL GetStdHandle@4

MOV DIN,EAX

PUSH -11

CALL GetStdHandle@4

MOV DOUT,EAX

; вывод приглашения

PUSH OFFSET STR\_INVITE

CALL lstrlenA@4 ; длина строки в EAX

PUSH 0

PUSH OFFSET LENS

PUSH EAX

PUSH OFFSET STR\_INVITE

PUSH DOUT

CALL WriteConsoleA@20

; чтение строки

PUSH 0

PUSH OFFSET LENS

PUSH 200

PUSH OFFSET BUF

PUSH DIN

CALL ReadConsoleA@20

sub LENS, 2

; перевод строки (восьмеричной) в число

MOV EDI,8 ; основание системы = 8

MOV ECX,LENS ; счётчик символов

MOV ESI,OFFSET BUF ; указатель на строку

XOR EAX,EAX ; число = 0

XOR EBX,EBX ; EBX будем использовать для цифры

XOR EBP,EBP ; EBP = 0 -> знак '+', 1 -> '-'

CMP BYTE PTR [ESI], '-' ; символ -> BL

JNE CONVERT\_OCT

MOV EBP, 1 ; отметили минус

INC ESI ; пропускаем '-'

DEC ECX ; уменьшаем длину

CONVERT\_OCT:

MOV BL,[ESI]

CMP BL,'0'

JB BAD\_INPUT ; если < '0' ? ошибка

CMP BL,'7'

JA BAD\_INPUT ; если > '7' ? ошибка

SUB BL,'0' ; преобразуем символ в цифру

SHL EAX, 3 ; EAX = EAX \* 8

ADD EAX,EBX ; EAX += цифра

INC ESI ; следующий символ

LOOP CONVERT\_OCT ; повторяем ECX раз

NEXT\_OCT:

CMP EBP, 0

JE CONVERT\_OK

XOR EDX, EDX ; EDX = 0

SUB EDX, EAX ; EDX = 0 - EAX

MOV EAX, EDX ; EAX = -EAX

JMP CONVERT\_OK

; обработка ошибки

BAD\_INPUT:

; печатаем строку STR\_ERR

PUSH OFFSET STR\_ERR

CALL lstrlenA@4

PUSH 0

PUSH OFFSET LENS

PUSH EAX

PUSH OFFSET STR\_ERR

PUSH DOUT

CALL WriteConsoleA@20

; завершаем программу

PUSH 0

CALL ExitProcess@4

CONVERT\_OK:

MOV X\_DEC,EAX ; сохранили десятичное число

; подпись "Число в десятичной системе: "

PUSH OFFSET STR\_DEC

CALL lstrlenA@4

PUSH 0

PUSH OFFSET LENS

PUSH EAX

PUSH OFFSET STR\_DEC

PUSH DOUT

CALL WriteConsoleA@20

; конвертация EAX -> десятичная строка со знаком

MOV EAX, X\_DEC ; число для печати

XOR EBP, EBP ; EBP=0 -> '+', 1 -> '-'

CMP EAX, 0

JGE dec\_abs\_ready

MOV EBP, 1 ; отрицательное

XOR EDX, EDX

SUB EDX, EAX ; |EAX| = 0 - EAX

MOV EAX, EDX

dec\_abs\_ready:

LEA ESI, OUTBUF[63] ; будем писать с конца буфера

XOR ECX, ECX ; длина строки = 0

MOV EBX, 10

; частный случай: ноль

CMP EAX, 0

JNE dec\_loop

DEC ESI

MOV BYTE PTR [ESI], '0'

INC ECX

JMP dec\_sign

dec\_loop:

; EAX > 0: вытаскиваем цифры в обратном порядке

XOR EDX, EDX

DIV EBX ; EAX = EAX/10, EDX = остаток

ADD DL, '0'

DEC ESI

MOV [ESI], DL

INC ECX

CMP EAX, 0

JNZ dec\_loop

dec\_sign:

CMP EBP, 0

JZ dec\_print

DEC ESI

MOV BYTE PTR [ESI], '-'

INC ECX

dec\_print:

; Печатаем ECX байт начиная с ESI

PUSH 0

PUSH OFFSET LENS ; сюда вернётся кол-во выведенных символов

PUSH ECX ; длина

PUSH ESI ; адрес начала строки

PUSH DOUT

CALL WriteConsoleA@20

; перевод строки

PUSH OFFSET STR\_NL

CALL lstrlenA@4

PUSH 0

PUSH OFFSET LENS

PUSH EAX

PUSH OFFSET STR\_NL

PUSH DOUT

CALL WriteConsoleA@20

; ===================== ВЫЧИСЛЕНИЕ F(x) =====================

; x^2

MOV EAX, X\_DEC

IMUL EAX, EAX ; EAX = x\*x

MOV X\_SQR, EAX

; a\*x^2

MOV EAX, X\_SQR

IMUL EAX, Acoef ; EAX = a\*x^2

MOV AX2, EAX

; b\*x

MOV EAX, X\_DEC

IMUL EAX, Bcoef ; EAX = b\*x

MOV BX1, EAX

; F = a\*x^2 + b\*x + c

MOV EAX, AX2

ADD EAX, BX1

ADD EAX, Ccoef

MOV F\_VAL, EAX

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

; ---- подпись для HEX ----

PUSH OFFSET STR\_FHEX

CALL lstrlenA@4

PUSH 0

PUSH OFFSET LENS

PUSH EAX

PUSH OFFSET STR\_FHEX

PUSH DOUT

CALL WriteConsoleA@20

; ---- конвертация F\_VAL -> HEX ----

MOV EAX, F\_VAL ; число для печати

XOR EBP, EBP ; 0 = '+', 1 = '-'

CMP EAX, 0

JGE f\_hex\_abs\_ready

MOV EBP, 1 ; отрицательное

XOR EDX, EDX

SUB EDX, EAX ; |EAX| = 0 - EAX

MOV EAX, EDX

f\_hex\_abs\_ready:

LEA ESI, OUTBUF[63] ; будем писать с конца буфера

XOR ECX, ECX ; длина строки = 0

MOV EBX, 16

; частный случай: ноль

CMP EAX, 0

JNE f\_hex\_loop

DEC ESI

MOV BYTE PTR [ESI], '0'

INC ECX

JMP f\_hex\_sign

f\_hex\_loop:

XOR EDX, EDX

DIV EBX ; EAX = EAX / 16, EDX = остаток 0..15

MOV DL, [DIGITS+EDX] ; символ из таблицы

DEC ESI

MOV [ESI], DL

INC ECX

CMP EAX, 0

JNZ f\_hex\_loop

f\_hex\_sign:

CMP EBP, 0

JZ f\_hex\_print

DEC ESI

MOV BYTE PTR [ESI], '-'

INC ECX

f\_hex\_print:

PUSH 0

PUSH OFFSET LENS

PUSH ECX ; длина строки

PUSH ESI ; адрес первого символа

PUSH DOUT

CALL WriteConsoleA@20

; перевод строки

PUSH OFFSET STR\_NL

CALL lstrlenA@4

PUSH 0

PUSH OFFSET LENS

PUSH EAX

PUSH OFFSET STR\_NL

PUSH DOUT

CALL WriteConsoleA@20

; ---- подпись для десятичного F(x) ----

PUSH OFFSET STR\_FDEC

CALL lstrlenA@4

PUSH 0

PUSH OFFSET LENS

PUSH EAX

PUSH OFFSET STR\_FDEC

PUSH DOUT

CALL WriteConsoleA@20

; ---- конвертация F\_VAL -> десятичная строка ----

MOV EAX, F\_VAL ; число для печати

XOR EBP, EBP ; 0 = '+', 1 = '-'

CMP EAX, 0

JGE f\_dec\_abs\_ready

MOV EBP, 1 ; отрицательное

XOR EDX, EDX

SUB EDX, EAX ; |EAX| = 0 - EAX

MOV EAX, EDX

f\_dec\_abs\_ready:

LEA ESI, OUTBUF[63] ; пишем с конца буфера

XOR ECX, ECX ; длина строки = 0

MOV EBX, 10

; частный случай: 0

CMP EAX, 0

JNE f\_dec\_loop

DEC ESI

MOV BYTE PTR [ESI], '0'

INC ECX

JMP f\_dec\_sign

f\_dec\_loop:

XOR EDX, EDX

DIV EBX ; EAX = EAX/10, EDX = остаток

ADD DL, '0'

DEC ESI

MOV [ESI], DL

INC ECX

CMP EAX, 0

JNZ f\_dec\_loop

f\_dec\_sign:

CMP EBP, 0

JZ f\_dec\_print

DEC ESI

MOV BYTE PTR [ESI], '-'

INC ECX

f\_dec\_print:

; печатаем ECX байт, начиная с ESI

PUSH 0

PUSH OFFSET LENS

PUSH ECX

PUSH ESI

PUSH DOUT

CALL WriteConsoleA@20

; перевод строки

PUSH OFFSET STR\_NL

CALL lstrlenA@4

PUSH 0

PUSH OFFSET LENS

PUSH EAX

PUSH OFFSET STR\_NL

PUSH DOUT

CALL WriteConsoleA@20

; выход

PUSH 0

CALL ExitProcess@4

MAIN ENDP

END MAIN

1. **Тесты**

|  |  |  |
| --- | --- | --- |
| № | Входные данные | Назначение |
| 1 |  | Введено положительное число в 8-системе счисления |
| 2 |  | Введено отрицательное число в 8-системе счисления |
| 3 |  | Введён ноль |
| 4 |  | Введено не число |
| 5 |  | Введено число, не входящее в 8 систему счисления |
| 6 |  | Введено большое число, для проверки на переполнение |
| 7 |  | Проверка на переполнение |