

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»

Институт информатики и кибернетики

Кафедра радиотехники

Отчет по лабораторной работе
”Основы программирования на языке ассемблера
процессоров фирмы INTEL”

Студент:	Согонов Е.А.
Преподаватель:	Корнилин Д.В.
Группа:	6364-120304D

Самара 2022

СОДЕРЖАНИЕ

1	Начало работы с Visual Studio	3
1.1	Создание исходных файлов	3
1.2	Добавление ассемблерной вставки	4
1.3	Исследование с помощью disassembly	5
1.4	Организация взаимодействия между программами на ассемблере и Си	7

1. Начало работы с Visual Studio

В данной лабораторной работе рассматриваются вопросы составления и отладки программ на языке ассемблера, а также совместного использования ассемблера и Си. Для начала работы необходимо создать директорию с проектом, в которой будут размещены необходимые файлы с текстом исходной программы, служебные файлы и, наконец, будет сформирован исполняемый файл. Были произведены все шаги, описанные в методических указаниях, после чего были исследованы предложенные примеры программ, результаты описаны далее.

1.1. Создание исходных файлов

Для начала, был создан файл с исходным кодом на языке Си следующего содержания:

```
#include "windows.h" //подключение библиотек
#include "stdio.h"
char Out[256]; //массив, хранящий текст
int X; //переменная, необходимая в следующем примере

int APIENTRY WinMain(HINSTANCE hInstance,
HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
// функция, выводящая на экран сообщение с результатом
{
    sprintf(Out, "Моя программа");
    MessageBox(NULL, Out, "TEST", MB_OK);
    return 1;
}
```

Наша программа с помощью функции `sprintf` записывает в переменную `Out` текст «Моя программа» и выводит этот текст на экран с помощью функции `MessageBox`. Результат выполнения на рисунке 1.

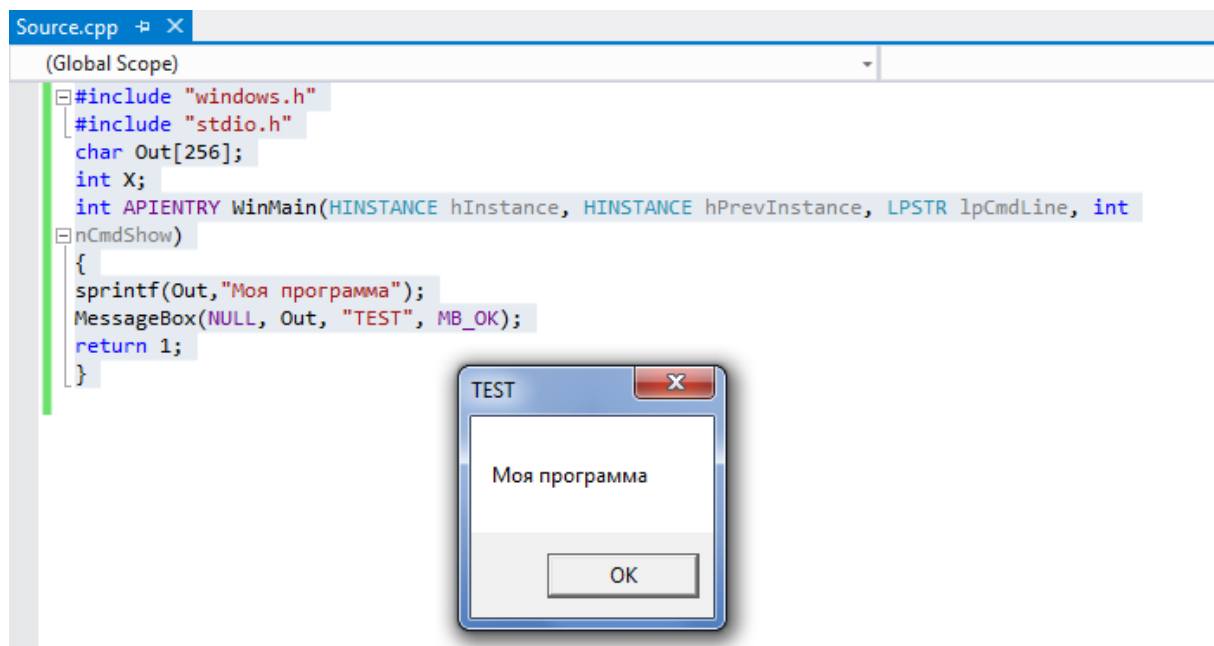


Рисунок 1 – Результат выполнения первой программы

1.2. Добавление ассемблерной вставки

Следующим этапом было добавление вставки в программу функции на языке ассемблера. Код принимает следующий вид:

```

#include "windows.h"
#include "stdio.h"

char Out[256];
int X;
int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
LPSTR lpCmdLine, int nCmdShow)
{
    //подключение ассемблерной вставки
    {
        mov eax, 2           ;копирует 2 в регистр eax
        mov ebx, 3           ;аналогично
        add eax, ebx         ;eax + ebx = 2+3=5; результат записан в eax
        mov X, eax           ;копирует в переменную X из регистра eax
    }
    sprintf(Out, "число - %d", X);    //создаем строку, которая будет
    //записана в массив out
    MessageBox(NULL, Out, "TEST", MB_OK);
    return 1;
}

```

Наша программа с помощью функции на языке ассемблера складывает два числа, записывает результат в переменную, которую уже Си выведет на экран с помощью функции MessageBox. Результат выполнения на рисунке 2

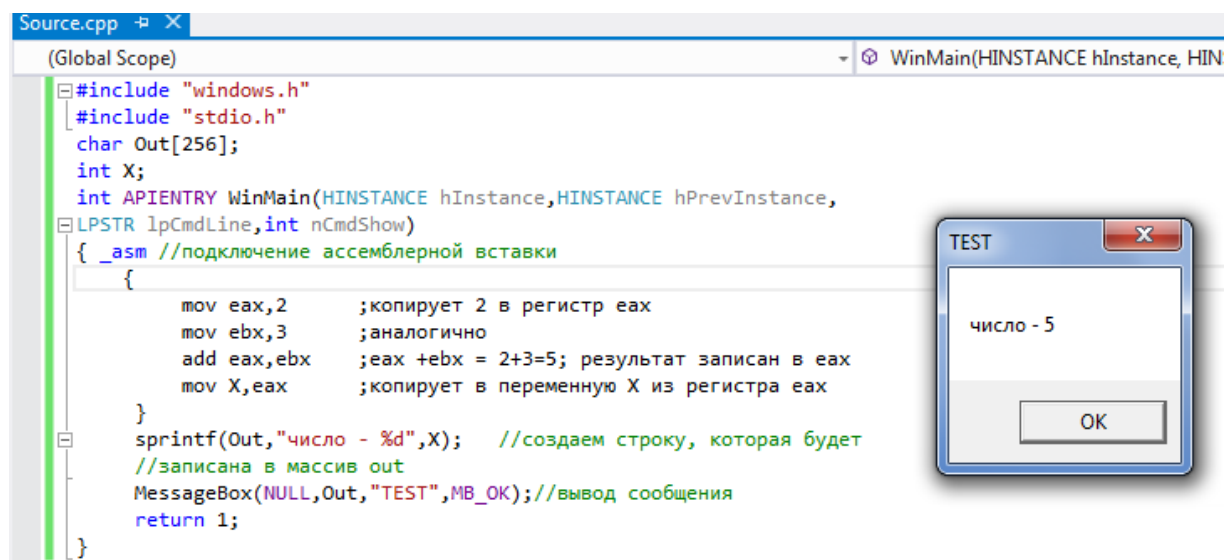


Рисунок 2 – Результат выполнения первой программы

1.3. Исследование с помощью disassembly

Далее программа была изменена, теперь она представляет собой простейший шифратор текста. Код принимает следующий вид:

```
#include "windows.h"
#include "stdio.h"
char In[256], Out[256];
int N;
BYTE Crypt=0x67;
int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
LPSTR lpCmdLine, int nCmdShow)
{
    sprintf(In, "Моя строка");
    N=strlen(In);
    _asm
    {
        mov ecx, N ;запись в регистр ecx переменной N
        lea edi, Out ;копирование в edi(регистра приемника) адреса массива out
        lea esi, In ;копирование в esi(регистра источника) адреса массива out

        L: lodsb ;Объявление метки L, считывание одного байта из esi в al
        xor al, Crypt ;сравнение байта в al и crypt
        jnz label ;переход к метке label если результат хог не 0
        mov al, 0x20 ;запись в регистр al 20h
        label: stosb ;объявление метки label, запись строки байт в edi
        loop L ;цикл- вернуться к метке L
    }
    MessageBox(NULL, Out, "TEST", MB_OK);
    return 1;
}
```

Результат дизассемблирования на рисунке 3

Disassembly Source.cpp

Address: _WinMain(HINSTANCE __*, HINSTANCE __*, char __*, int)

Viewing Options

```

    lea edi,Out ;копирование в edi(регистра приемника) адреса массива out
00F6104F lea      edi,[Out (0F6E260h)]
    lea esi,In ;копирование в esi(регистра источника) адреса массива out
00F61055 lea      esi,[In (0F6E158h)]

    L: lodsb ;Объявление метки L, считывание одного байта из esi в al
00F6105B lodsb   byte ptr [esi]
    xor al,Crypt ;сравнение байта в al и crypt
00F6105C xor     al,byte ptr [Crypt (0F6E000h)]
    jnz label ;переход к метке label если результат хог не 0
00F61062 jne     label (0F61066h)
    mov al,0x20 ;запись в регистр al 20h
00F61064 mov     al,20h
    label: stosb ;объявление метки label, запись строки байт в edi
00F61066 stosb   byte ptr es:[edi]
    loop L ;цикл- вернуться к метке L
00F61067 loop    L (0F6105Bh)
}
MessageBox(NULL,Out,"TEST",MB_OK);
00F61069 push     0
00F6106B push     offset ___xi_z+84h (0F64164h)
00F61070 push     offset Out (0F6E260h)
00F61075 push     0
00F61077 call     +7m@ MessageBox (0F6108Ah)

```

Рисунок 3 – Результат дизассемблирования

Disassembly Source.cpp

(Global Scope) WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)

```

#include "windows.h"
#include "stdio.h"
char In[256],Out[256];
int N;
BYTE Crypt=0x67;
int APIENTRY WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,
LPSTR lpCmdLine,int nCmdShow)
{
    sprintf(In,"Моя строка");
    N=strlen(In);
    _asm
    {
        mov ecx,N ;запись в регистр ecx переменной N
        lea edi,Out ;копирование в edi(регистра приемника) адреса массива out
        lea esi,In ;копирование в esi(регистра источника) адреса массива out

        L: lodsb ;Объявление метки L, считывание одного байта из esi в al
        xor al,Crypt ;сравнение байта в al и crypt
        jnz label ;переход к метке label если результат хог не 0
        mov al,0x20 ;запись в регистр al 20h
        label: stosb ;объявление метки label, запись строки байт в edi
        loop L ;цикл- вернуться к метке L
    }
    MessageBox(NULL,Out,"TEST",MB_OK);
    return 1;
}

```

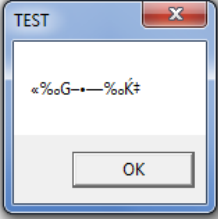


Рисунок 4 – Результат выполнения

1.4. Организация взаимодействия между программами на ассемблере и Си

Кроме использования ассемблерных вставок, можно создавать отдельные модули, написанные на языке ассемблера, и представляющие собой самостоятельные подпрограммы. Для этого необходимо создать отдельный файл с расширением .asm, и настроить компилятор. В результате действий, описанных в указаниях к лабораторной работе, были получены следующие результаты.

Настроен компилятор для автоматического ассемблирования файлов *.asm при сборке с помощью следующих выражений, как показано на рисунке 5

```
ml /c /Zi %(FileName).asm --command line
```

```
%(FileName).obj --outputs
```

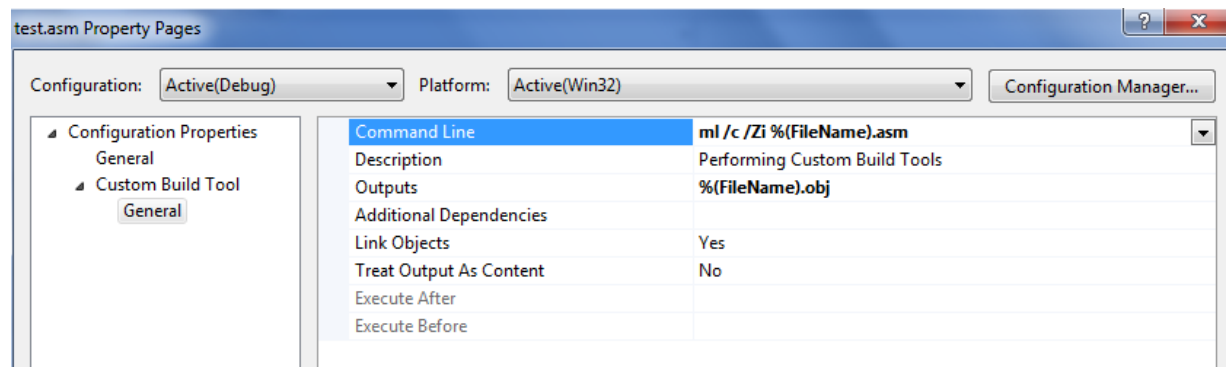


Рисунок 5 – Настройка компилятора

Создан файл с кодом на языке ассемблера:

```
.486
.model flat, c
.code
Func PROC
push ebp
mov ebp, esp
mov ecx, [ebp+16]
mov edi, [ebp+12]
mov esi, [ebp+8]
L: lodsb
mov [edi+ecx-1], al
loop L
mov esi, offset Text
add edi, [ebp+16]
mov ecx, 22
rep movsb
pop ebp
ret
Func ENDP
Text DB ' - Перевернутая строка'
END
```

И изменен код программы на языке Си

```
#include "windows.h"
#include "stdio.h"
extern "C" void Funct(char* x, char* y, int Len);
char In[256], Out[256];
int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
LPSTR lpCmdLine, int nCmdShow)
{
    sprintf(In, "Моя строка");
    Funct(In, Out, strlen(In));
    MessageBox(NULL, Out, "TEST", MB_OK);
    return 1;
}
```

Результат выполнения программы на рисунке 6

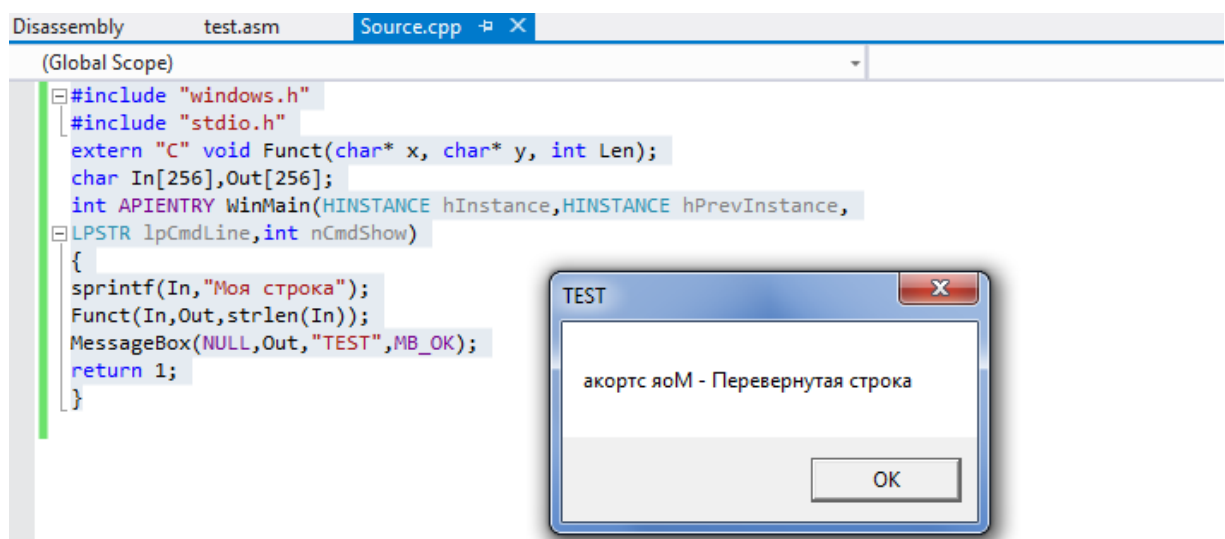


Рисунок 6 – Результат выполнения программы