

### Введение в язык Ассемблер

**Цель:** изучение основ программирования на языке ассемблера для процессоров Intel семейства IA-32.

Рекомендуемая литература:

1	Ирвин К. Р. Язык ассемблера для процессоров Intel / К. Р. Ирвин. – М.: Вильямс, 2005. – 912с.
2	Калашников О. А. Ассемблер – это просто. Учимся программировать/ О. А. Калашников – СПб.: БХВ-Петербург, 2011. – 336с.

1. **Ассемблер:** язык программирования (язык ассемблера), транслятор с языка ассемблера.

Язык ассемблера — это машинно-ориентированный язык программирования. Главная особенность — команды ассемблера в большинстве своем соответствуют инструкциям процессора.

2. **Ассемблер:** транслятор – это программа, преобразующая исходный текст на языке ассемблера в машинный код.

3. **Ассемблер:** язык разрабатывается для семейства процессоров: Motorola 680 (MC680), SPARC, IBM370, IA-32.

Примеры операторов ассемблера:

- add – сложить;
- call – вызвать.

#### Синтаксис

Для работы с процессорами x86 используются два типа синтаксиса ассемблера — это синтаксис AT&T и синтаксис Intel. Эти синтаксисы представляют одни и те же команды совершенно по-разному.

Intel	AT&T	Некоторые особенности синтаксиса AT&T
<b>mov eax, 10h</b>	<b>mov<sup>l</sup> \$0x10, %eax</b>	- <b>l</b> - суффикс имени инструкции; - <команда> <источник>, <приёмник>; - знак доллара в начале числовой константы

4. **Ассемблер:** набор инструкций процессора, CISC: Motorola 680; RISC: SPARC, IA-32.

5. **Ассемблер:** MASM (Microsoft Macro Assembler), TASM (Turbo Assembler, Borland), NASM (Netwide Assembler, Windows/Linux), Asm86, GNU Assembler (Software Foundation).

История развития языков ассемблера:

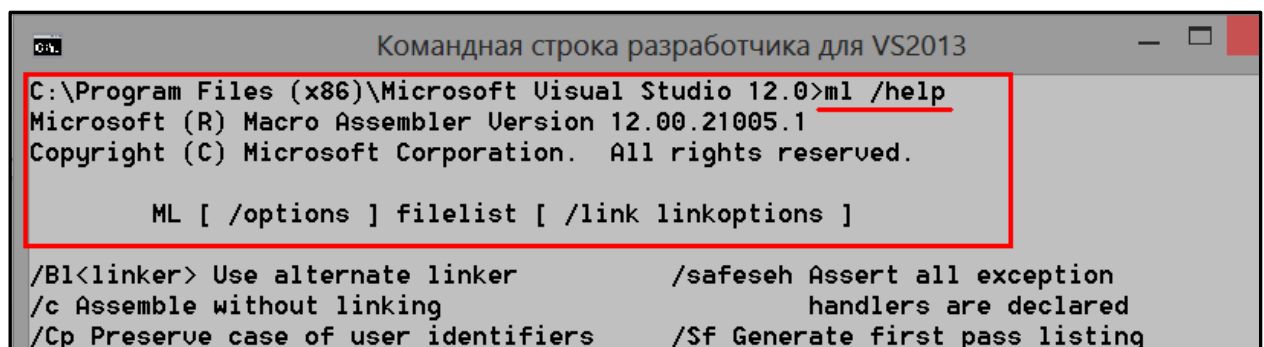
Де-факто стандартом языка ассемблера стала 5-я версия компилятора MASM (Macro Assembler) фирмы Microsoft.

В начале 90-х годов появился TASM (Turbo Assembler) фирмы Borland, который имеет полную совместимость с компилятором MASM.

Также наиболее популярными являются (синтаксис отличается):

- NASM (Netwide Assembler) – существуют версии для Windows и Linux.
- Asm86 и GNU Assembler, распространяемые Фондом программ с открытым исходным кодом (Free Software Foundation).

6. **Ассемблер:** в курсе рассматривается ассемблер для архитектуры IA-32, набор инструкций RISC, MASM 12.0



```
Командная строка разработчика для VS2013
C:\Program Files (x86)\Microsoft Visual Studio 12.0>ml /help
Microsoft (R) Macro Assembler Version 12.00.21005.1
Copyright (C) Microsoft Corporation. All rights reserved.

ML [ /options ] filelist [ /link linkoptions ]

/Bl<linker> Use alternate linker           /safeseh Assert all exception
/c Assemble without linking               handlers are declared
/Cp Preserve case of user identifiers     /Sf Generate first pass listing
```

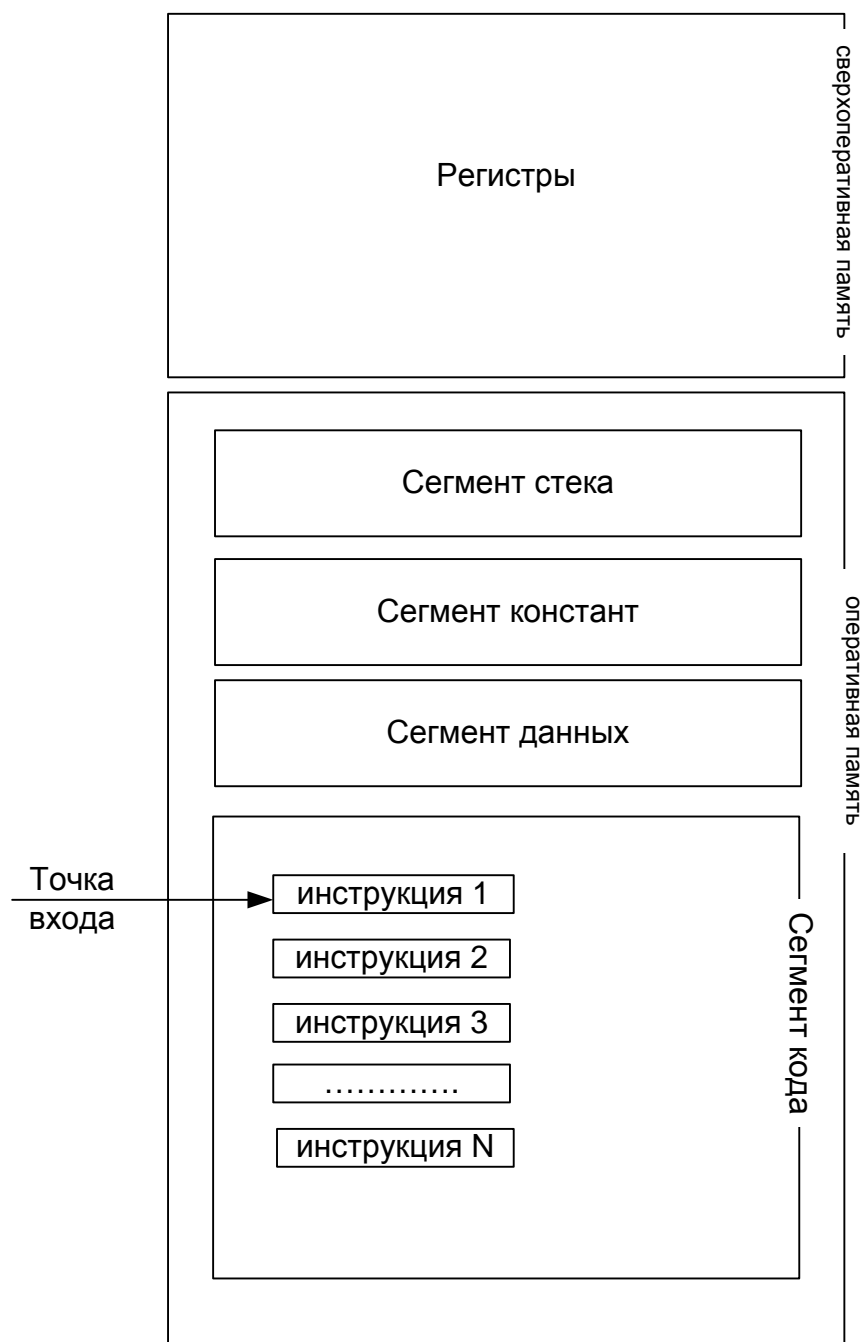
7. **Ассемблер:** архитектура аппаратно-программного обеспечения компьютера (Э. Таненбаум).



Э. Таненбаум – профессор Амстердамского свободного университета, возглавляет группу разработчиков компьютерных систем, автор книг по компьютерным наукам, преподаватель.

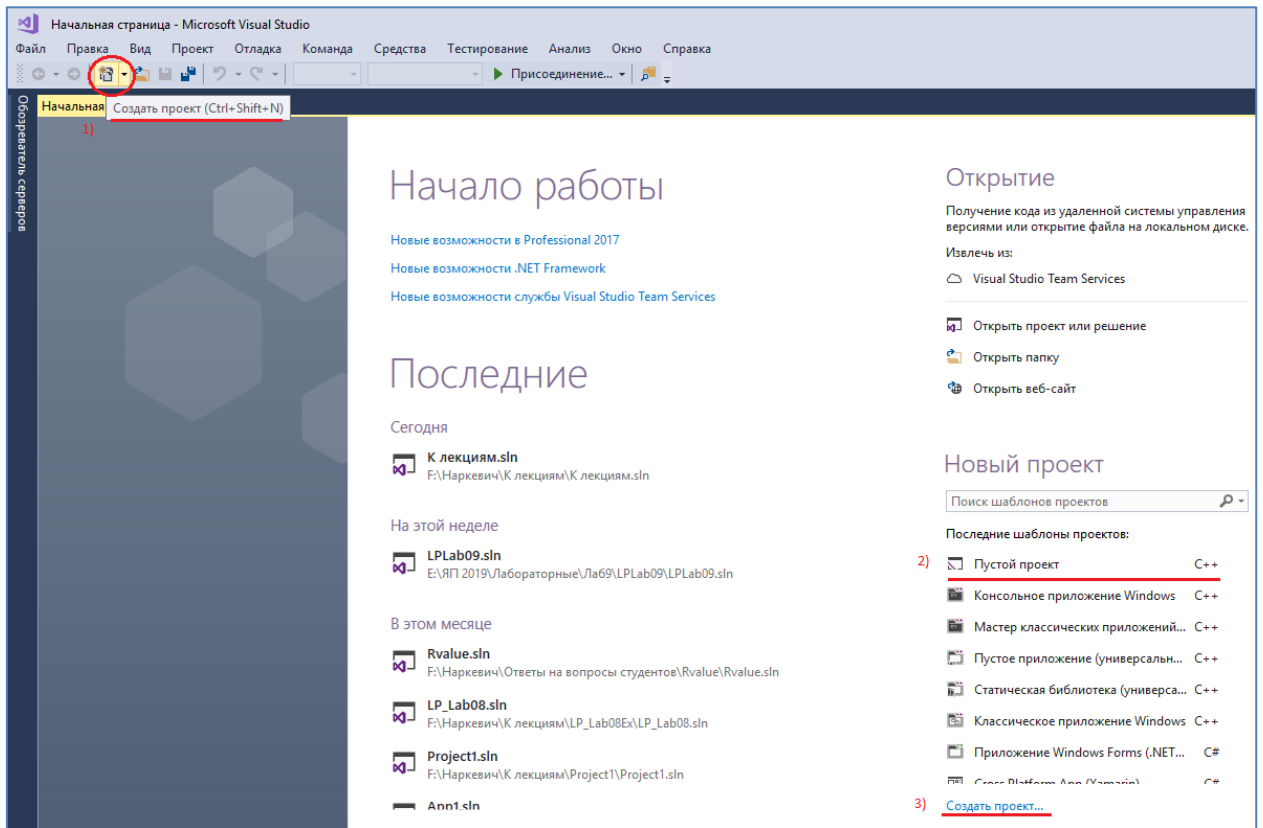


## 8. Ассемблер: представление компьютера для разработчика

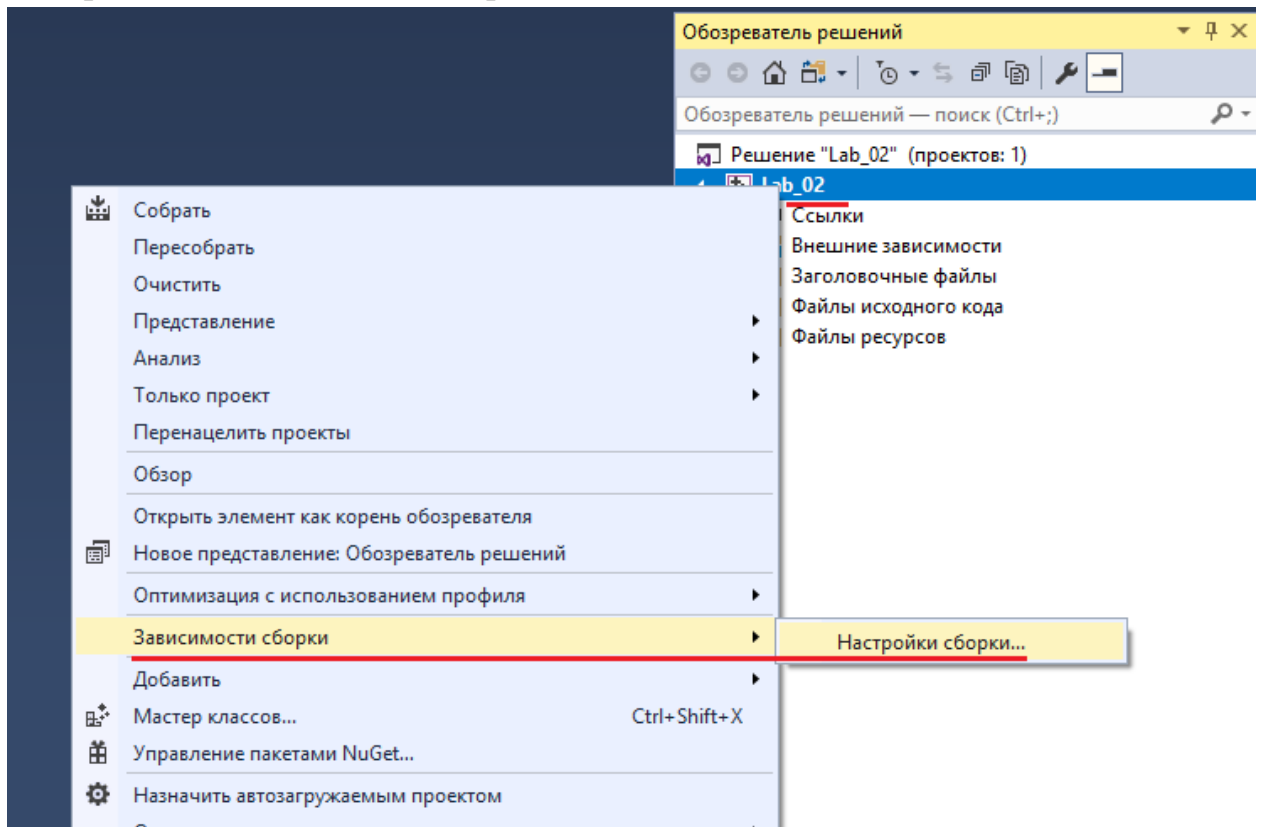


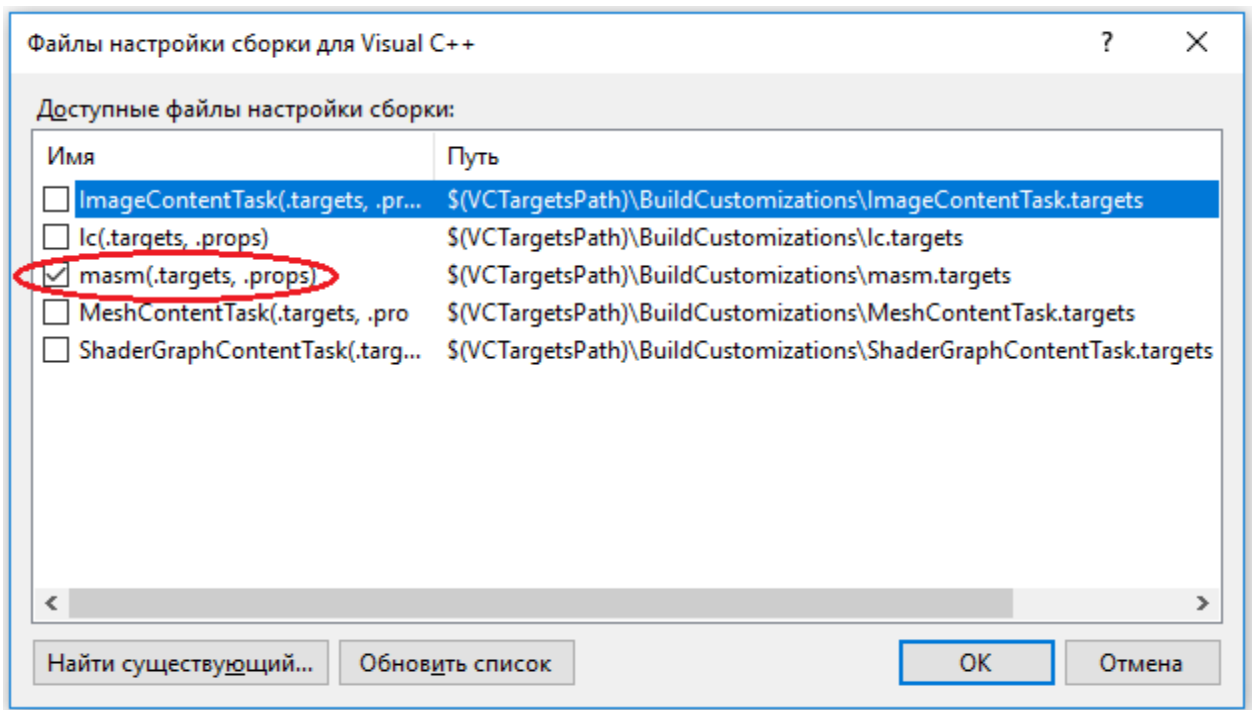
## 9. Создание MASM-проекта в Visual Studio 2017.

Создаем пустой проект:

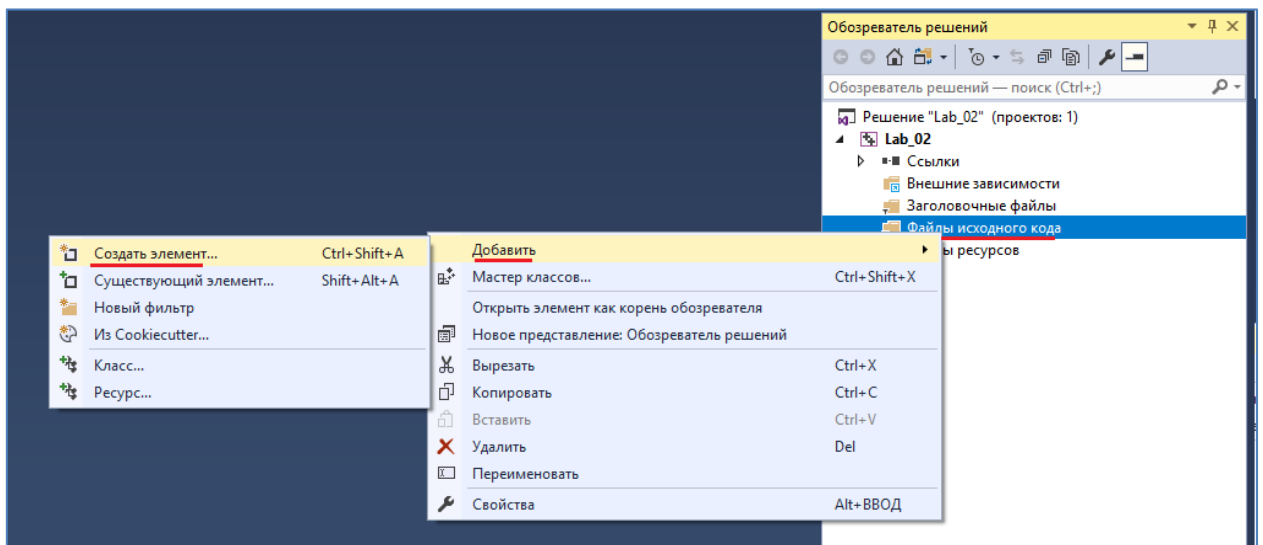


Настраиваем зависимости сборки:

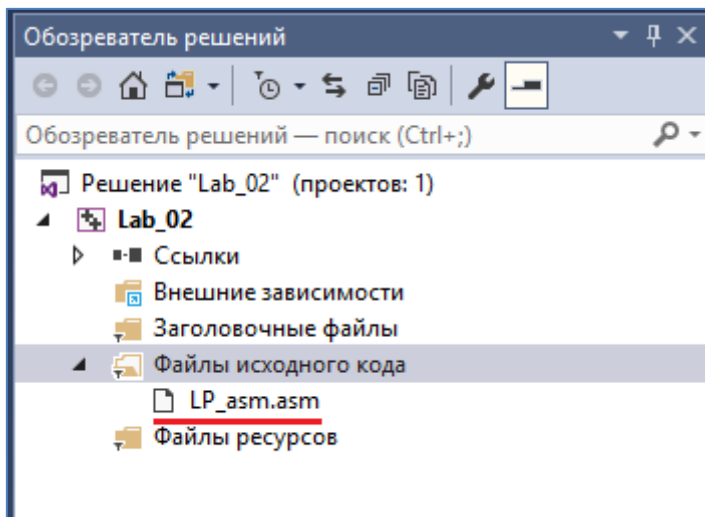
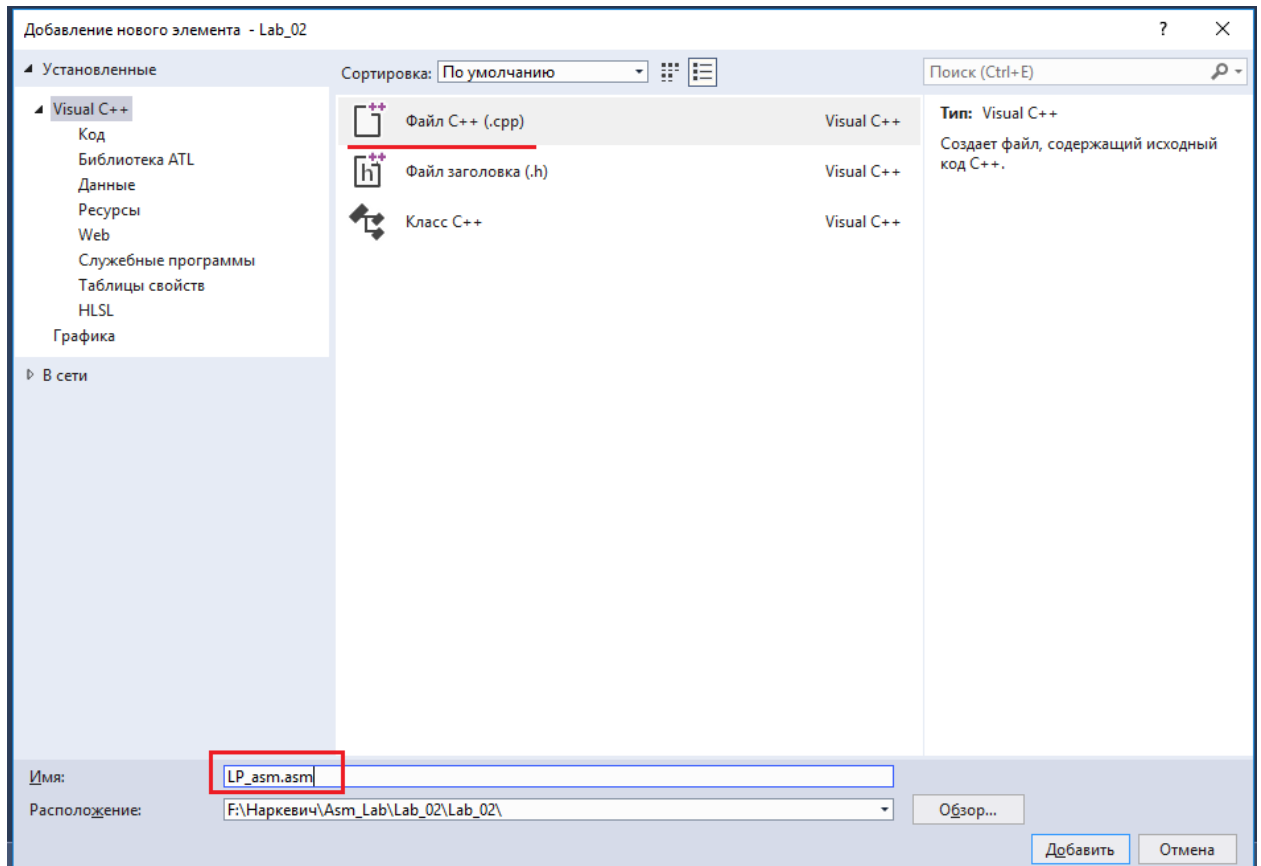


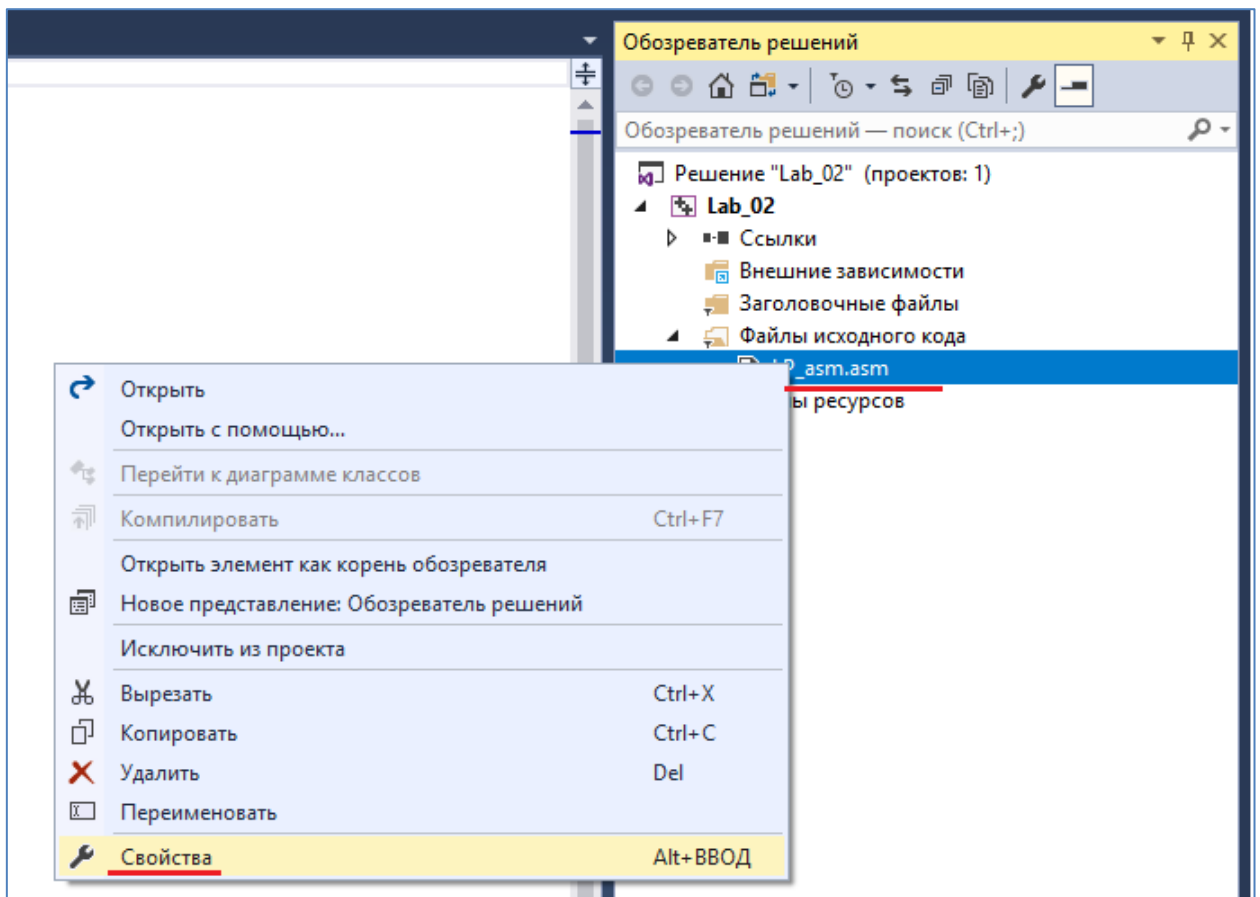


Добавляем новый элемент (создаем новый):

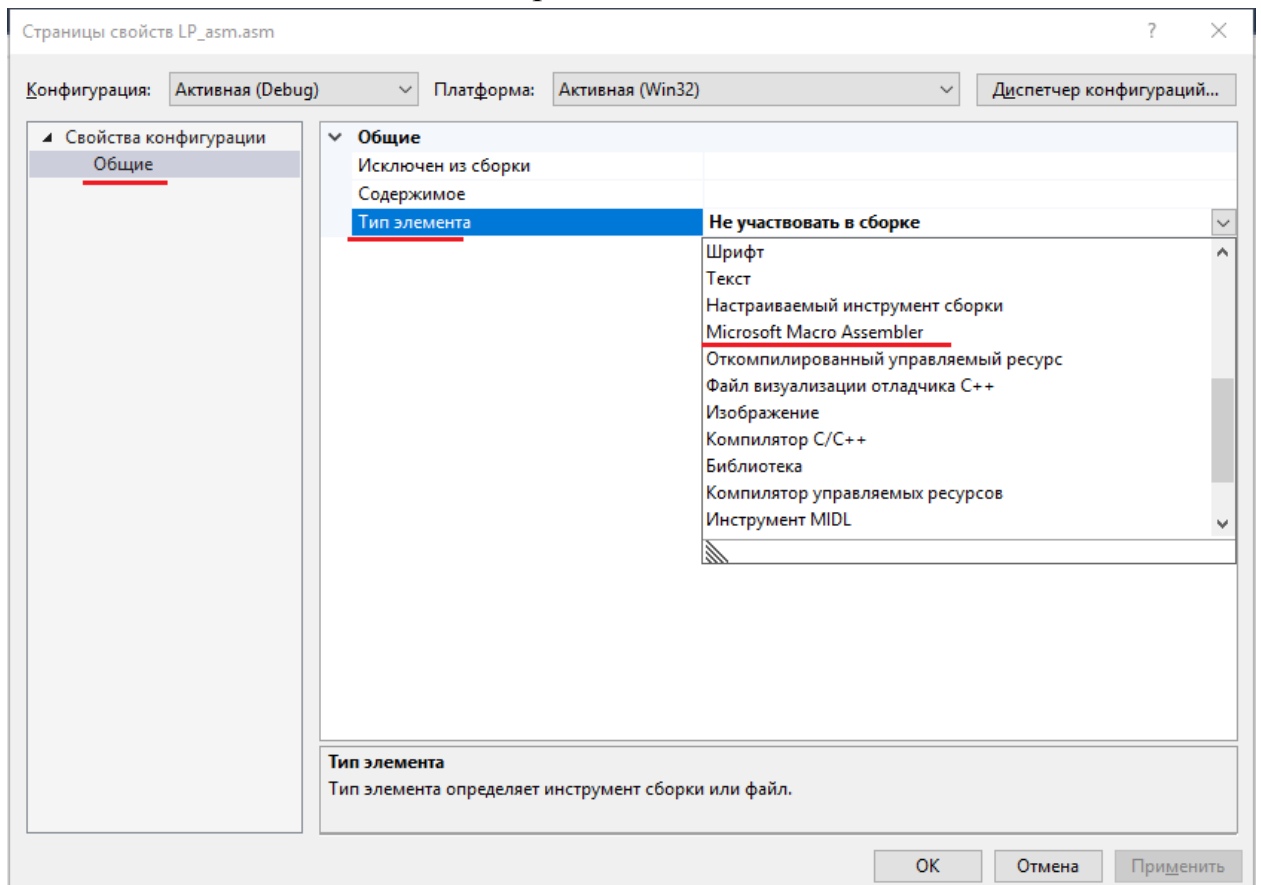


Указываем имя исходного файла на языке ассемблера (.asm)



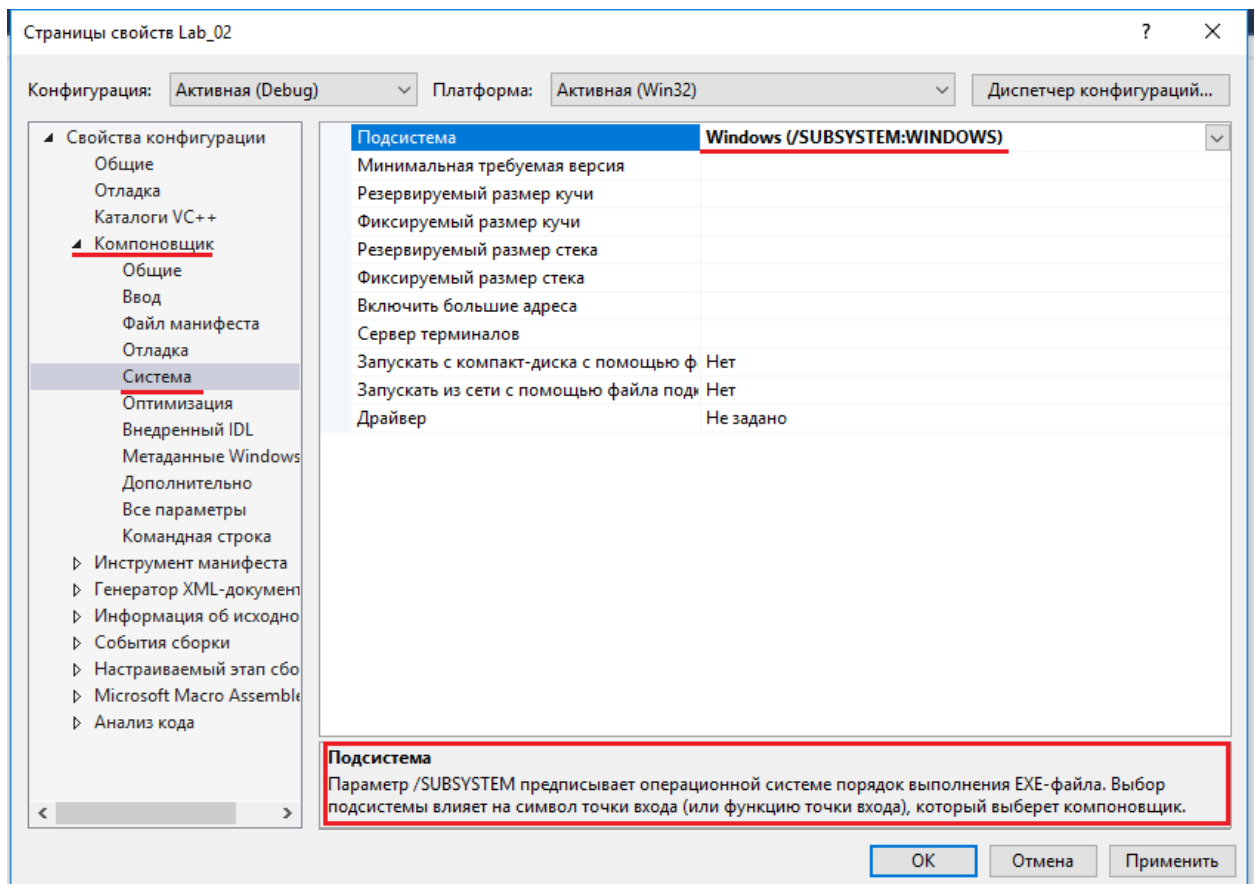
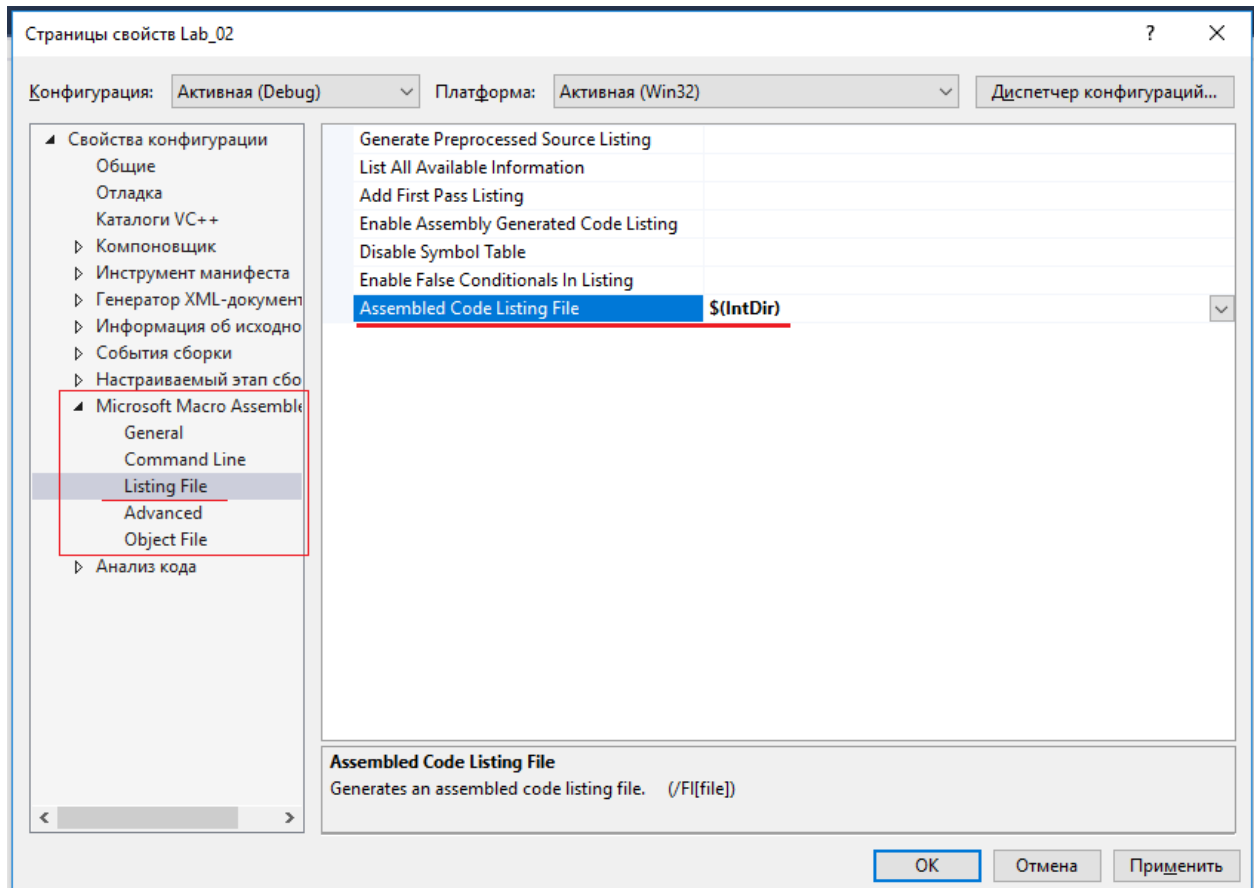


Устанавливаем тип элемента сборки:





Указываем местоположение листинга:



## 10. Простейшее приложение

```
.586                ; система команд (процессор Pentium)
.model flat,stdcall  ; модель памяти, соглашение о вызовах
includelib kernel32.lib ; компоновщик: компоновать с kernel32.lib
ExitProcess PROTO    :DWORD ; прототип функции

.stack 4096          ; сегмент стека объемом 4096

.const               ; сегмент констант

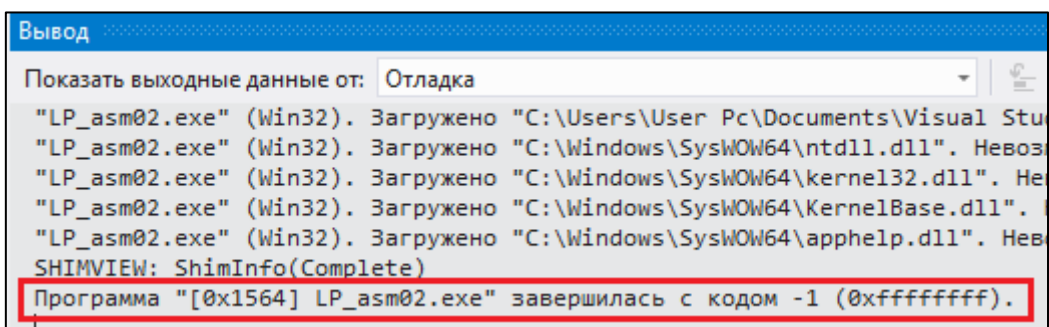
.data                ; сегмент данных

.code                ; сегмент кода

main PROC            ; начало процедуры

    push -1           ; код возврата процесса (параметр ExitProcess )
    call ExitProcess  ; так должен заканчиваться любой процесс Windows
main ENDP             ; конец процедуры

end main              ; конец модуля, main - точка входа
```



Выход

Показать выходные данные от: Отладка

"LP\_asm02.exe" (Win32). Загружено "C:\Users\User Pc\Documents\Visual Stu

"LP\_asm02.exe" (Win32). Загружено "C:\Windows\SysWOW64\ntdll.dll". Невоз

"LP\_asm02.exe" (Win32). Загружено "C:\Windows\SysWOW64\kernel32.dll". Не

"LP\_asm02.exe" (Win32). Загружено "C:\Windows\SysWOW64\KernelBase.dll".

"LP\_asm02.exe" (Win32). Загружено "C:\Windows\SysWOW64\apphelp.dll". Нев

SHIMVIEW: ShimInfo(Complete)

Программа "[0x1564] LP\_asm02.exe" завершилась с кодом -1 (0xffffffff).

## Простое приложение:

```
.586P ; система команд(процессор Pentium)
.MODEL FLAT, STDCALL ; модель памяти, соглашение о вызовах
includelib kernel32.lib ; компоновщику: компоновать с kernel32

ExitProcess PROTO : DWORD ; прототип функции для завершения процесса Windows
MessageBoxA PROTO : DWORD, : DWORD, : DWORD, : DWORD ; прототип API-функции MessageBoxA

.STACK 4096 ; выделение стека объемом 4 мегабайта

.CONST ; сегмент констант

.DATA ; сегмент данных
MB_OK EQU 0 ; EQU определяет константу
STR1 DB "Моя первая программа", 0 ; строка, первый элемент данные + нулевой бит
STR2 DB "Привет всем!", 0 ; строка, первый элемент данные + нулевой бит
HW DD ? ; двойное слово длиной 4 байта, неинициализировано

.CODE ; сегмент кода

main PROC ; точка входа main
START : ; метка
    PUSH MB_OK
    PUSH OFFSET STR1
    PUSH OFFSET STR2
    PUSH HW
    CALL MessageBoxA ; вызов функции

    push - 1 ; код возврата процесса Windows(параметр ExitProcess)
    call ExitProcess ; так завершается любой процесс Windows
main ENDP ; конец процедуры

end main ; конец модуля main
```

Программа на ассемблере должна включать один главный, или основной, модуль, с которого начинается ее выполнение.

Модуль может содержать программные сегменты, сегменты данных и стека. Нужно указать модель памяти при помощи директивы `.MODEL`.

Плоская модель памяти flat (flat memory model). Эта модель памяти используется в операционной системе Windows. Адресация любой ячейки памяти будет определяться содержимым одного 32-битного регистра. Определяет приложение, выполняющееся, в защищенном режиме с использованием линейной (несегментированной) модели памяти.

`stdcall` – используемое соглашение о вызовах процедур.

Подключаем необходимые библиотеки: `kernel32.lib`, которая содержит функцию `ExitProcess`.

Объявляем прототип функции с использованием директивы `PROTO`

(после символа «:» указываются типы параметров).

Параметры WinAPI-функций 32-битные – это целые числа.

Все WinAPI-функции созданы по соглашению `stdcall`.

Функция MessageBoxA вызывается из системной библиотеки Windows user32.dll.

MessageBox – выводит на экран окно с сообщением и кнопкой выхода. Параметры:

- дескриптор окна, в котором будет появляться окно-сообщение;
- текст, который будет появляться в окне;
- текст в заголовке окна;
- тип окна, в частности можно определить количество кнопок выхода.

В Windows для прикладной программы отводится один большой плоский сегмент.

### Секции:

.STACK – стек. Размер стека по умолчанию – 1 Мб.

.DATA – сегмент (или секция) данных.

.CODE – сегмент кода.

Директива EQU определяет константу (подобно #define в языке СИ).

STR1 и STR2 – символьные строки, должны заканчиваться 0 байтом.

HW – неинициализированное двойное слово (4 байта = 32-бита).

START – метка.

Директива OFFSET указатель начала строки.

CALL – вызов функции.

END main – конец программы, начало которой определяет метка main.

```
.586P                                ; система команд(процессор Pentium)
.MODEL FLAT, STDCALL                 ; модель памяти, соглашение о вызовах
includelib kernel32.lib              ; компоновщику: компоновать с kernel32

ExitProcess PROTO : DWORD             ; прототип функции для завершения процесса Windows
MessageBoxA PROTO : DWORD, : DWORD, : DWORD, : DWORD ; прототип API - функции MessageBoxA

.STACK 4096                           ; выделение стека объемом 4 мегабайта

.CONST                                ; сегмент констант

.DATA                                 ; сегмент данных
MB_OK EQU 0                           ; EQU определяет константу
STR1 DB "Моя первая программа", 0     ; строка, первый элемент данные + нулевой бит
STR2 DB "Привет всем!", 0             ; строка, первый элемент данные + нулевой бит
HW DD ?                               ; двойное слово длиной 4 байта, неинициализировано

.CODE                                 ; сегмент кода

main PROC                             ; точка входа main
START :                               ; метка

    INVOKE MessageBoxA, HW, OFFSET STR2, OFFSET STR1, MB_OK

    push - 1                          ; код возврата процесса Windows(параметр ExitProcess)
    call ExitProcess                  ; так завершается любой процесс Windows
main ENDP                             ; конец процедуры

end main                             ; конец модуля main
```

Транслятор языка MASM позволяет упростить вызов функций при помощи директивы INVOKE. Встроенный макрос INVOKE используется для вызова любых функций, прототип должен быть задан. Параметры записываются точно в том порядке, в котором приведены в описании функции.

## 11. Листинг

Ассемблер может создавать листинг программы с номерами строк, адресами переменных, операторами исходного языка и таблицей перекрестных ссылок символов и переменных, используемых в программе.

D:) ► Adel ► LPPrim ► LP_asm02 ► LP_asm02 ► Debug			
Имя	Дата изменения	Тип	Размер
LP_asm02.tlog	03.04.2017 0:53	Папка с файлами	
<b>LP_asm.lst</b>	03.04.2017 0:51	MASM Listing	4 КБ
LP_asm.obj	03.04.2017 0:51	Object File	2 КБ
LP_asm02.Build.CppClean.log	03.04.2017 0:51	Log File	1 КБ
LP_asm02.log	03.04.2017 0:53	Log File	2 КБ

Microsoft (R) Macro Assembler Version 12.00.21005.1		04/03/17 00:51:04	
LP_asm.asm		Page 1 - 1	
.586P		; система команд(процессор Pentium)	
.MODEL FLAT, STDCALL		; модель памяти, соглашение о вызовах	
includelib kernel32.lib		; компоновщику: компоновать с kernel32	
ExitProcess PROTO : DWORD		; прототип функции для завершения процесса Windows	
MessageBoxA PROTO : DWORD, : DWORD, : DWORD, : DWORD			
.STACK 4096		; выделение стека объемом 4 мегабайта	
00000000 .CONST		; сегмент констант	
00000000 .DATA		; сегмент данных	
= 00000000 MB_OK EQU 0		; EQU определяет константу	
00000000 CC EE FF 20 EF STR1 DB "Моя первая программа", 0		; строка, первый элемент данные + нулевой бит	
E5 F0 E2 E0 FF			
20 EF F0 EE E3			
F0 E0 EC EC E0			
00			
00000015 CF F0 E8 E2 E5 STR2 DB "Привет всем!", 0		; строка, первый элемент данные + нулевой бит	
F2 20 E2 F1 E5			
EC 21 00			
00000022 00000000 HW DD ?			
00000000 .CODE		; сегмент кода	
00000000 main PROC		; точка входа main	
00000000 START :		; метка	

# Segments and Groups:

N a m e	Size	Length	Align	Combine	Class
CONST . . . . .	32 Bit	00000000	Para	Public	'CONST' ReadOnly
FLAT . . . . .	GROUP				
STACK . . . . .	32 Bit	00001000	Para	Stack	'STACK'
__DATA . . . . .	32 Bit	00000026	Para	Public	'DATA'
__TEXT . . . . .	32 Bit	0000001E	Para	Public	'CODE'

# Procedures, parameters, and locals:

N a m e	Type	Value	Attr
ExitProcess . . . . .	P Near	00000000	FLAT Length= 00000000 External STDCALL
MessageBoxA . . . . .	P Near	00000000	FLAT Length= 00000000 External STDCALL
main . . . . .	P Near	00000000	__TEXT Length= 0000001E Public STDCALL
START . . . . .	L Near	00000000	__TEXT