1. ЛАБОРАТОРНАЯ РАБОТА № 1 «СИСТЕМА КОМАНД МИКРОПРОЦЕССОРА X86»

Цель работы: изучение системы команд и способов адресации микропроцессоров с архитектурой x86.

1. Создание проекта

Запустите Visual Studio 2019, выберите «Пустой проект C++» (Рис. 1). Выберите расположение и имя проекта.

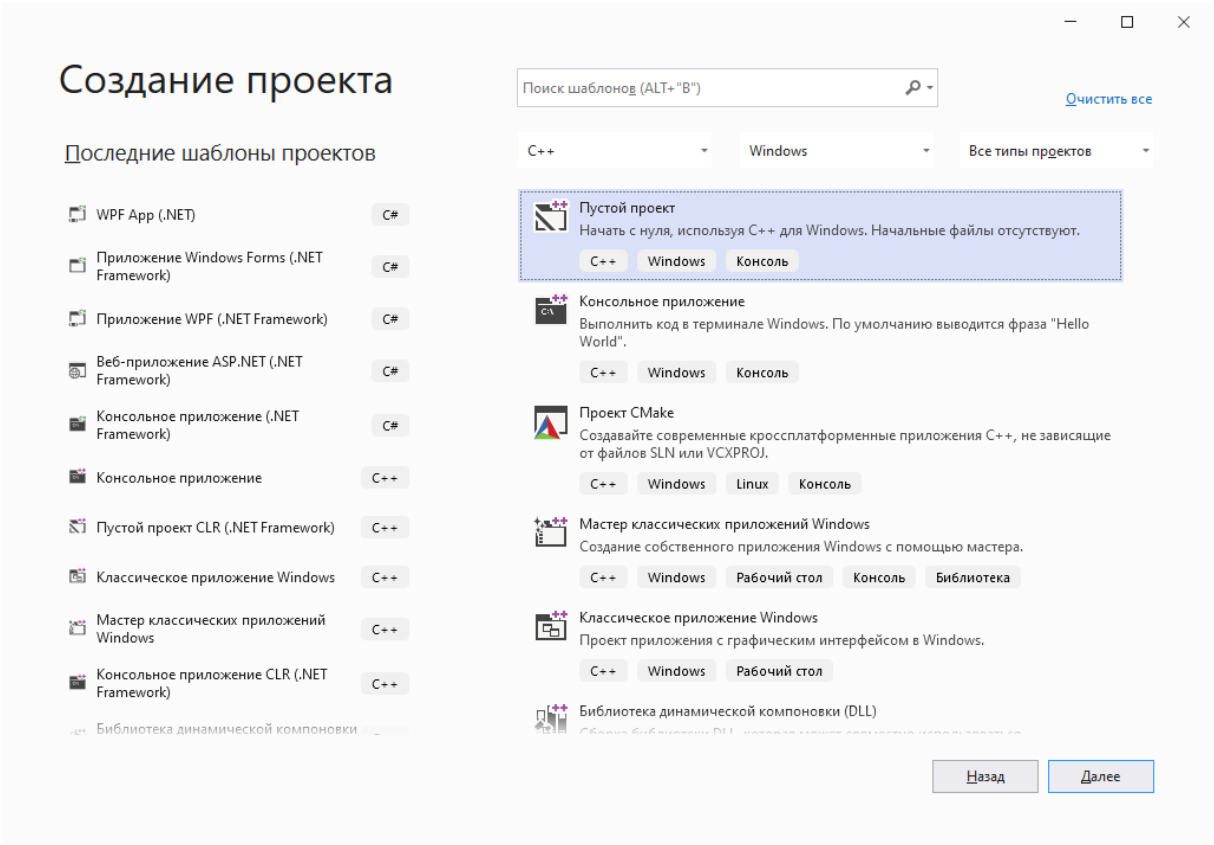

Рис. 1. Создание проекта

Добавьте файл исходного кода в проект. Для этого правой кнопкой мыши нажмите на названии проекта и выберите пункт Добавить->Создать новый элемент (Рис. 2). Выберите «Файл C++», но в имени файла поменяйте расширение на *.asm (рис. 3).

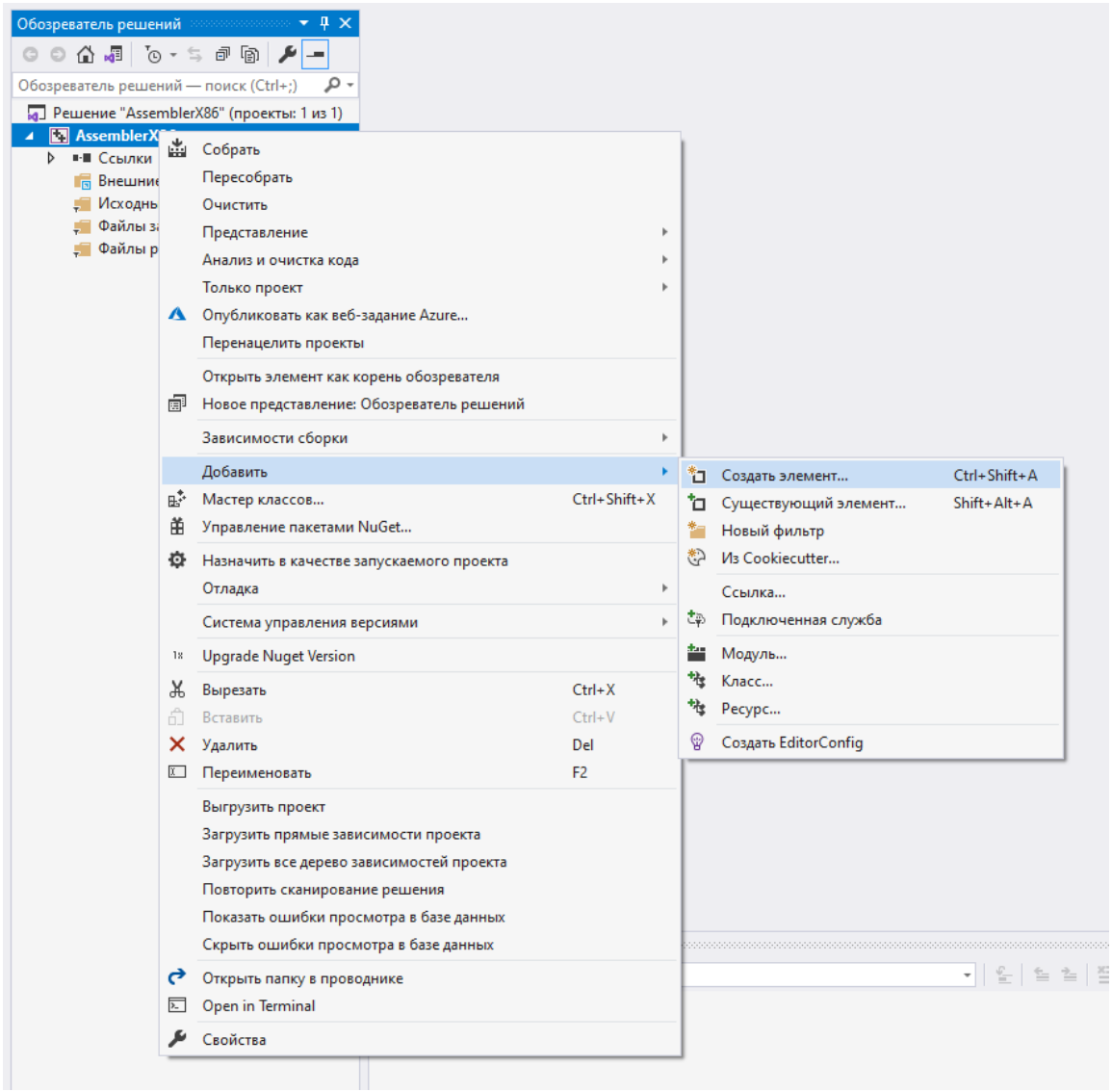

Рис. 2. Добавление нового файла в проект

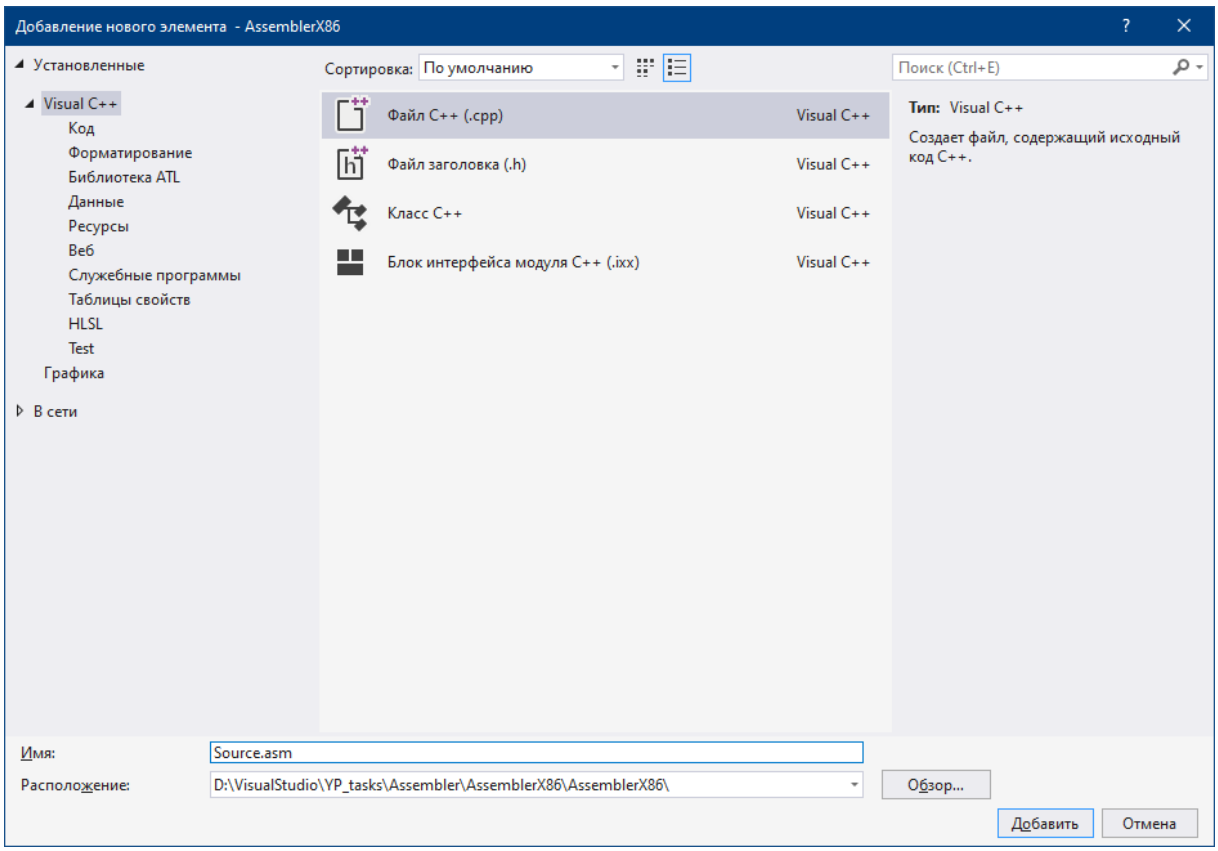

Рис. 3. Создание файла ассемблера

В созданный файл добавьте следующий код:

```
.686
.model flat,stdcall

.stack 100h

.data

.code
ExitProcess PROTO STDCALL :DWORD
Start:

exit:
Invoke ExitProcess,1
End Start
```

Далее необходимо сообщить среде разработки, что данный файл является программой на языке ассемблера, и для корректного включения его в проект требуется использовать Microsoft Macro Assembler.

Нажмите правой кнопкой мыши на имени проекта, выберите Зависимости сборки -> Настройки сборки (рис. 4). В открывшемся окне поставьте галочку в строке masm (рис. 5).

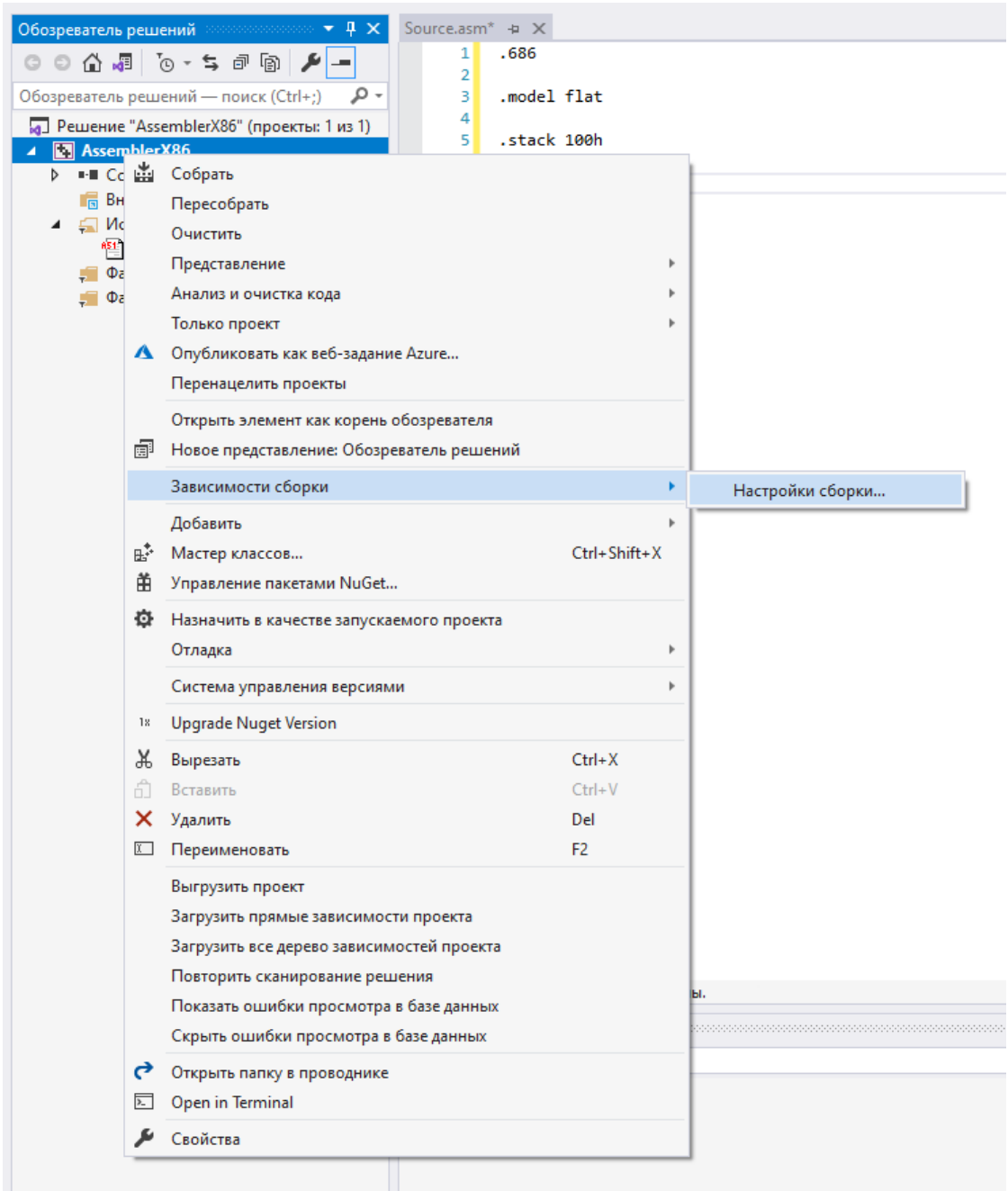

Рис. 4. Настройки сборки

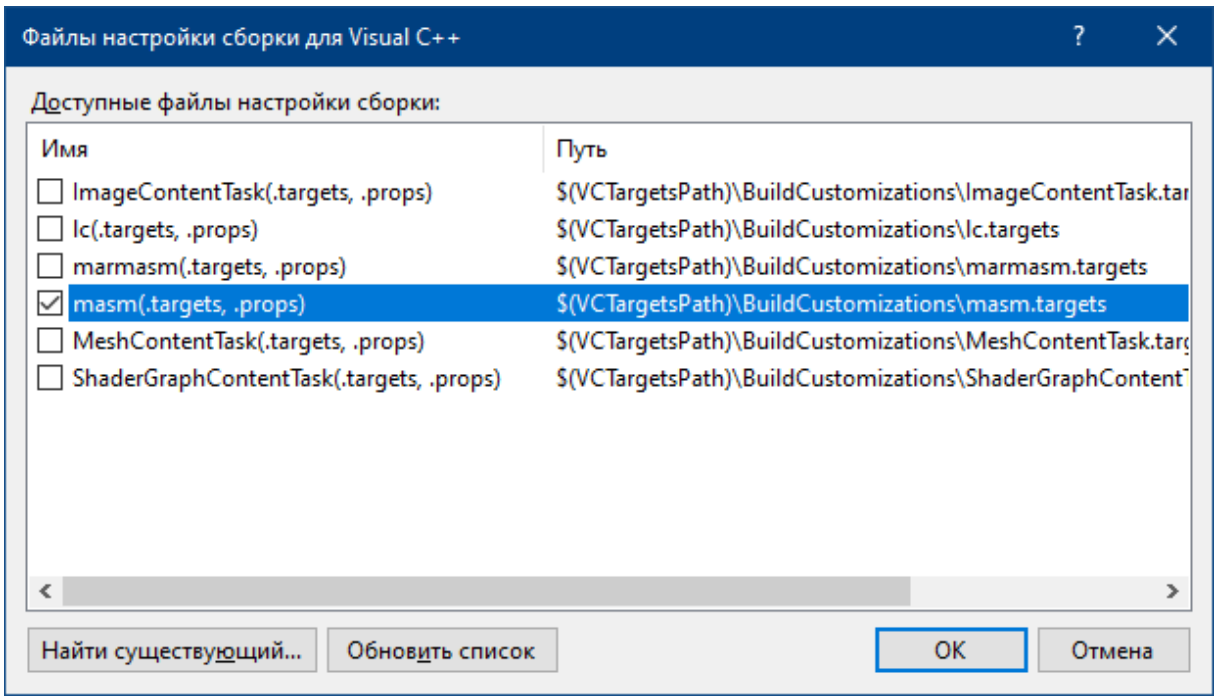

Рис. 5. Использование MASM

Теперь нужно проверить, что для файла на языке ассемблера установился соответствующий инструмент сборки. Правой кнопкой мыши нажмите на файле *.asm и выберите пункт Свойства (рис. 6). В открывшемся окне в пункте «Тип элемента» выберите «Microsoft Macro Assembler» (рис. 7).

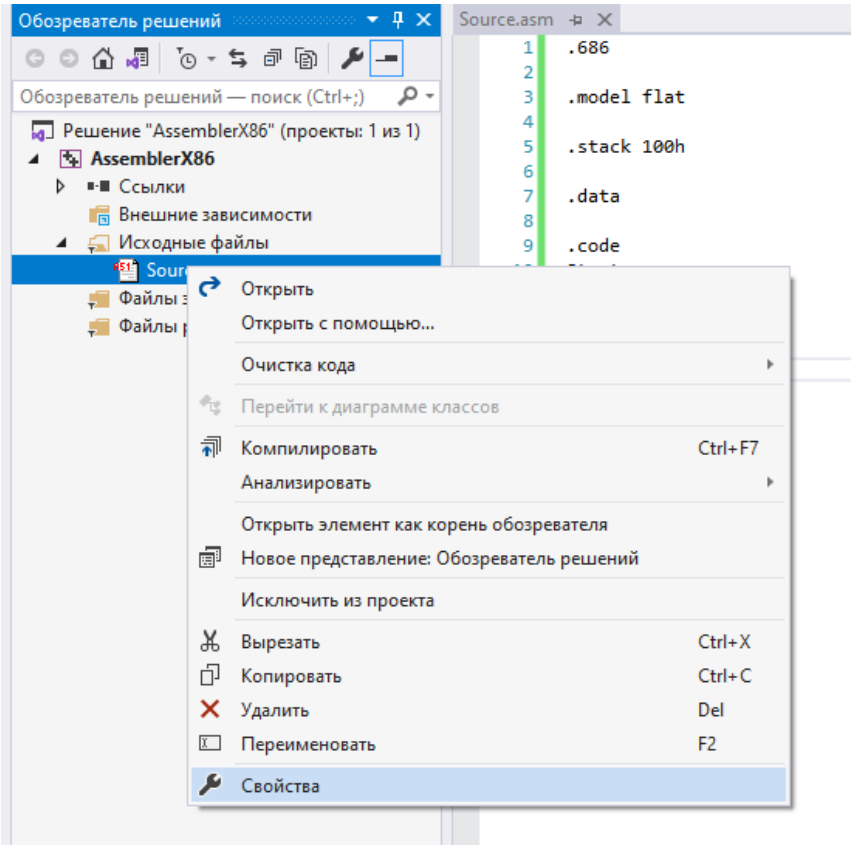

Рис. 6. Свойства файла *.asm

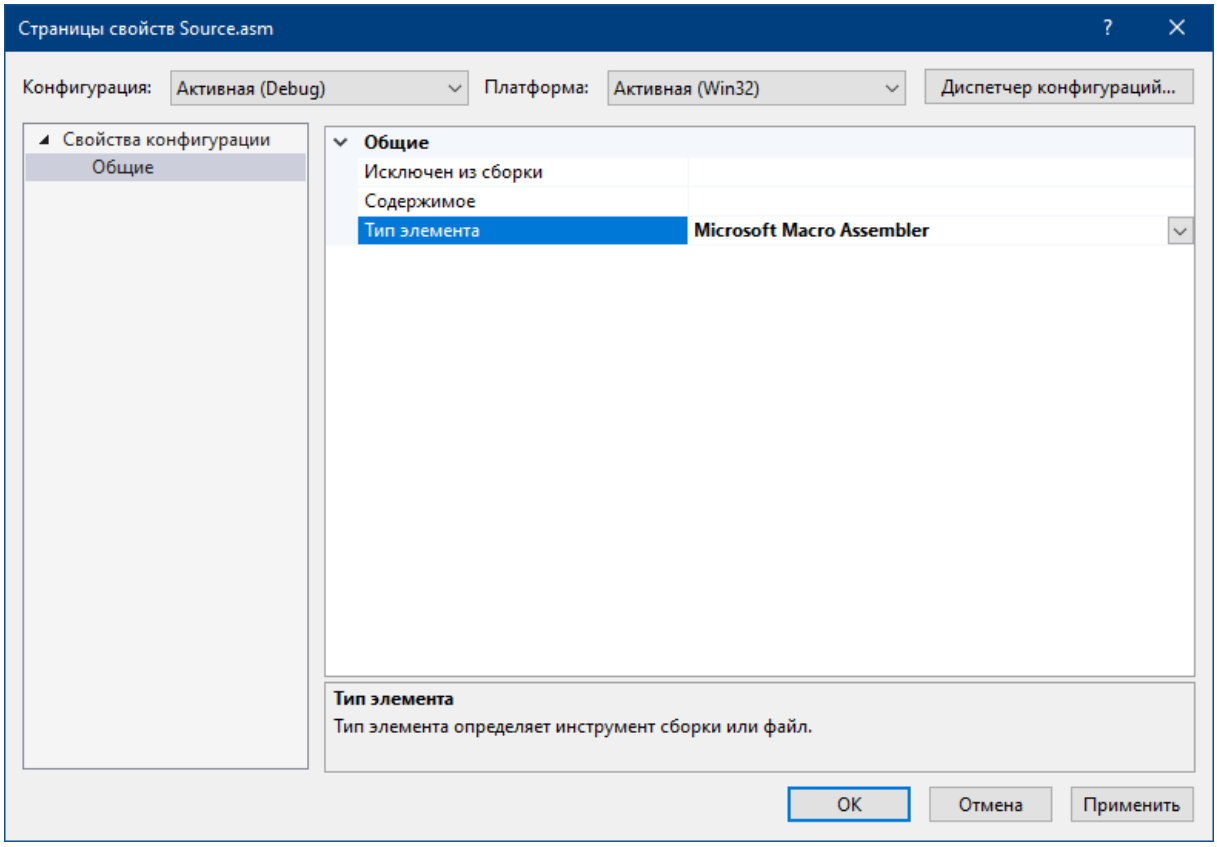

Рис. 7. Выбор типа элемента

Переменные объявляются в сегменте .data. Типы данных ассемблера приведены в таблице 1.

Таблица 1. Типы данных ассемблера

Тип	Директива	Количество байт	
Байт	DB	1	Символ, целое
Слово	DW	2	Целое, вещественное
Двойное слово	DD	4	Целое, вещественное
8 байт	DQ	8	Целое, вещественное
10 байт	DT	10	Вещественное

Формат объявления переменной:

[имя] [тип] [значение]

Если нужно объявить переменную, но не инициализировать ее, то вместо поля [значение] ставится символ «?».

Например,

X dw 5 ; объявление переменной размером 2 байта X = 5

X dw ? ; объявление переменной без инициализации

M1 dd 0,1,2,3,4,5,6,7,8,9 ; объявление массива M1

a dd 20 dup (0) ; объявление массива a, состоящего из 20 элементов, начальные значения которых равны 0.

Для отладки программы на ассемблере можно использовать стандартные средства Visual Studio.

Чтобы посмотреть значения переменных при выполнении программы, используйте окно «Контрольные значения» (рис. 8).

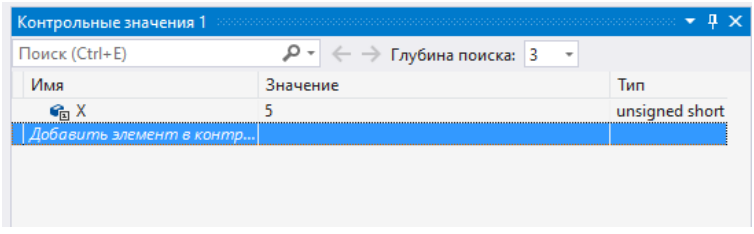

Рис. 8. Просмотр значений переменных

В ассемблере вычисления в основном осуществляются с регистрами, значения регистров можно посмотреть в окне «Регистры». Чтобы открыть данное окно запустите программу в режиме пошагового выполнения (F11) в меню «Отладка» выберите Окна -> Регистры.

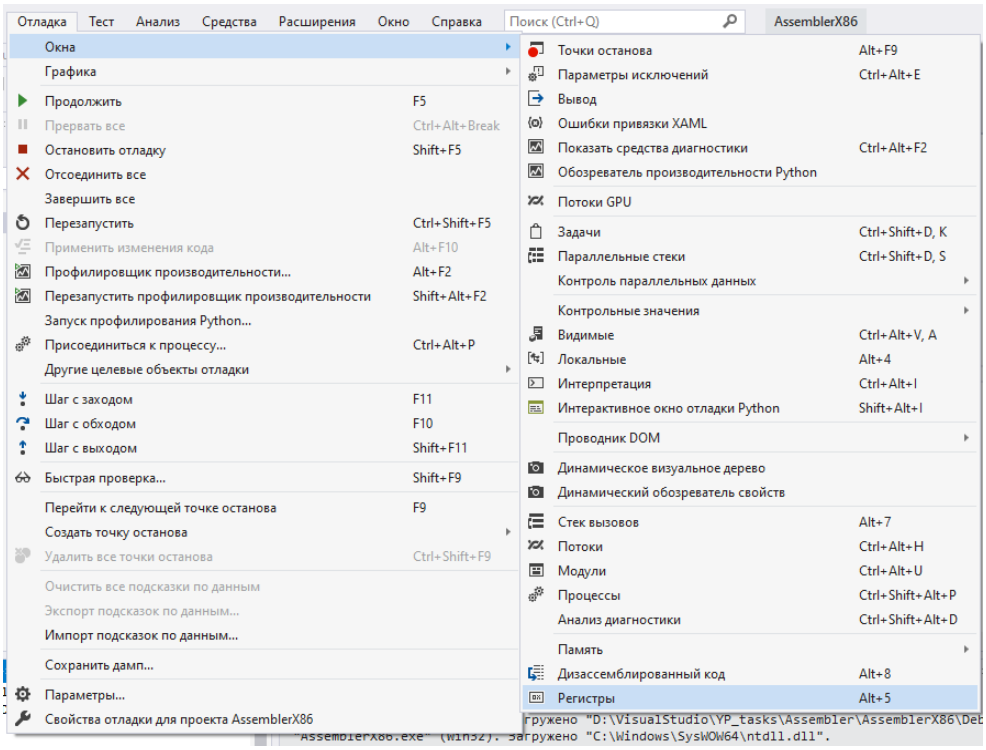

Рис. 9. Окно Регистры

В открывшемся окне на пустом поле нажмите правой кнопкой мыши, появится список доступных регистров. Для данной работы вам понадобятся: «ЦП», «Сегменты ЦП», «Флаги». Отметьте данные пункты в выпадающем списке. После выбора регистров их значения будут отображаться в окне (рис. 10).

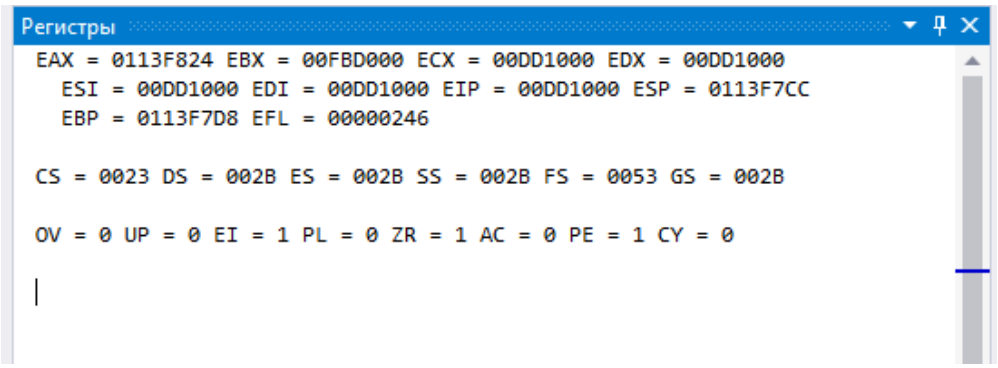

Рис. 10. Просмотр регистров

В Visual Studio названия флагов несколько отличаются от общепринятых:

Номер бита	11	10	09	07	06	04	02	00
Флаги	**of**	**df**	**if**	**sf**	**zf**	**af**	**pf**	**cf**
Флаги в **Visual Studio**	**OV**	**UP**	**EI**	**PL**	**ZR**	**AC**	**PE**	**CY**

Код программы пишется в разделе .code после метки Start.

2. Выполнение индивидуального задания

Напишите программу на Ассемблере, которая реализует выражение в соответствии с вариантом (таблица 1). Определите, происходит ли переполнение при вычислении выражения. Команды ассемблера см. в Приложении 1 или в [2].

Таблица 1 – Варианты заданий

Вариант	Исходные данные	Задание
1	X = 100 Y = 21 Z = 4	Вычислить M=(X'*3+Y') or Z', где X', Y', Z' – получены в результате циклического сдвига вправо на 3 разряда X, Y, Z
2	X = 103 Y = 12	Вычислить M=(X'-4*Y') xor Y', где X', Y' – получены в результате вычитания X, Y из 0.
3	X = 58 Y = 23 Z = 11	Вычислить M=((X+Y+Z) & X)-((X+Y+Z) & Y') , где Y' – получено в результате обмена местами старших и младших бит Y
4	X = 17 Y = 5 Z = 44	Вычислить M=(Z-X-Y)/2+(X & Y)
5	X = 121 Y = 35 Z = 4	Вычислить M=Z*5+(X' & Y'), где X', Y' – получены в результате инвертирования младших бит X, Y
6	X = 18 Y = 33 Z = 8	Вычислить M=X*4+ Z'*4 - Y*4, где Z' – получено в результате циклического сдвига влево на 5 разрядов через перенос Z
7	X = 15 Y = 79 Z = 81	Вычислить M = ((X+Y)/4) or (Z-Y-X)
8	X = 8 Y = -7 Z = -81	Вычислить M=(Z'+X*Y)xor(X + Y) , где Z' – получено в результате циклического сдвига на 3 бита влево Z
9	X = 87 Y = 60 Z = -2	Вычислить M=(X'-Y)+(Z & Y') , где X', Y' – получено в результате циклического сдвига на 2 бита вправо X и Y соответственно

10	X = -13 Y = 26 Z = 15	Вычислить M=(Z'*5-X+1)+(X or Y) , где Z' – получено в результате сдвига на 3 бита влево через перенос Z
11	X = -20 Y = 54 Z = -5	Вычислить M=(Y' & X xor Z') / 2 , где Z', Y' – получены в результате инверсии младших 4 бит Z и Y соответственно
12	X = 11 Y = 5 Z = 76	Вычислить M=(Z'xor(X+Y))and(X - Y) , где Z' – получено в результате инверсии всех бит Z
13	X = 23 Y = 6 Z = 16	Вычислить M=(Z*X'-Y)or(X and Y') , где X', Y' – получены в результате сдвига влево на 3 бита X и Y соответственно
14	X = 15 Y = -10 Z = 65	Вычислить M=(Z+X'*Y')xor(X' + Y') , где X', Y' – получены в результате сдвига вправо через перенос на 5 бит X и Y соответственно
15	X = 9 Y = 44 Z = 12	Вычислить M=(Z'or X' and Y)+(X' / Z) , где Z', X' – получены в результате инверсии 4 старших бит X и Y соответственно
16	X = 8 Y = -5 Z = 14	Вычислить M=(Z+X+Y)or(X' + Y' + Z') , где X', Y', Z' – получены в результате сдвига вправо через перенос на 6 бит X, Y, Z соответственно
17	X = -15 Y = 5 Z = 3	Вычислить M=(Z/2+X'*Y)+(X' and Z') , где X', Z' – получены в результате инверсии 4 младших бит X и Z соответственно
18	X = 16 Y = 57 Z = 68	Вычислить M=(Y'/2+X'+Z')+((X and Z) orY) , где X', Z',Y' – получены в результате увеличения на 1 X, Z и Y соответственно
19	X = 3 Y = 0 Z = 12	Вычислить M=((Y/2)or(X*Z')+(X * Y') , где Y', Z' – получены в результате инверсии Y и Z соответственно
20	X = 67 Y = 2 Z = 13	Вычислить M=(X-Z'/Y)+((X' xor Z')and Y') , где X', Z', Y' – получены в результате сдвига вправо на 5 бит X, Z и Y соответственно

3. Литература

1. Справочник по ассемблеру макросов / MSDN. – url: https://docs.microsoft.com/ru-ru/cpp/assembler/masm/microsoft-macro-assembler-reference?view=msvc-160

2. Базовая система команд x86 / club155. – url: http://www.club155.ru/x86cmdcpu

3. Микропроцессоры семейства i80x86. Учебное пособие / П.С. Епифанов, Н.А. Краев, А.В. Частиков // ВятГУ. – 2000.

4. Содержание отчета

1. Текст программы с комментариями
2. Верификация программы: результаты расчета заданного выражения, скриншоты, показывающие содержимое регистров и значения переменных после каждого действия программы.

Приложение 1. Машинные коды команд

КОМАНДА	БАЙТЫ КОДА КОМАНДЫ			СИМВОЛИЧЕСКОЕ ОПИСАНИЕ И/ИЛИ СОДЕРЖАНИЕ ОПЕРАЦИИ
	B1	B2	B3-B6	
КОМАНДЫ ОБЩЕГО НАЗНАЧЕНИЯ				
MOV				ПЕРЕСЫЛКА
r,r/m	1000101W	md reg r/m	(disp8/16)	r<-r/m
r/m,r	1000100W	md reg r/m	(disp8/16)	r/m<-r
r/m,d	1100011W	md 000 r/m	(disp8/16)d8/16	r/m<-d
r,d	1011Wreg	data L	(data H)	r<-d
ac,m	1010000W	disp L	disp H	ac<-m; прям. адр.
m,ac	1010001W	disp L	(disp H)	m<-ac; прям. адр.
sr,r/m	10001110	md Osr r/m	(disp8/16)	sr<-r/m; запись в сегментный регистр
r/m,sr	10001100	md Osr r/m	(disp8/16)	r/m<-sr; чтение из сегментного регистра
PUSH r/m	11111111	md 110 r/m	(disp8/16)	ЗАПИСЬ в стек
R	01010reg			
Sr	000sr110			
POP r/m	10001111	md 000 r/m	(disp8/16)	ЧТЕНИЕ из стека
R	01011reg			
Sr	000sr111			
XCHG				ОБМЕН
AX,r	10010reg			AX<->r
r,AX	10010reg			r<->AX
r,r/m	1000011W	md reg r/m	(disp8/16)	r<->r/m
m,r	1000011W	md reg r/m	(disp8/16)	m<->r
XLAT	11010111			ПРЕОБРАЗОВАНИЕ байта из AL
КОМАНДЫ ВВОДА-ВЫВОДА				
IN ac,port	1110010W	port8		ВВОД из фиксированного порта
ac,DX	1110110W			ВВОД из порта с косвенной адресацией
OUT ac,port	1110011W	port8		ВЫВОД в фиксированный порт
ac,DX	1110111W			ВЫВОД в порт при косвенной адресации
КОМАНДЫ ПЕРЕСЫЛКИ АДРЕСА				
LEA r16,m	10100101	md reg r/m	(disp8/16)	ЗАГРУЗКА эффективного адреса r<-EA
LDS r16,m32	11000101	md reg r/m	(disp8/16)	ЗАГРУЗКА указателя адреса r,DS<-m32
LES r16,m32	11000100	md reg r/m	(disp8/16)	ЗАГРУЗКА указателя адреса r,ES<-m32
КОМАНДЫ ПЕРЕСЫЛКИ ФЛАГОВ				
LAHF	10011111			ПЕРЕСЫЛКА младшего байта F в AH
SAHF	10011110			ПЕРЕСЫЛКА AH в регистр F

КОМАНДА	БАЙТЫ КОДА КОМАНДЫ			СИМВОЛИЧЕСКОЕ ОПИСАНИЕ И/ИЛИ СОДЕРЖАНИЕ ОПЕРАЦИИ
	B1	B2	B3-B6	
PUSHF	10011100			ЗАПИСЬ регистра флагов в стек
POPF	10011101			ЗАГРУЗКА регистра флагов из стека
КОМАНДЫ СЛОЖЕНИЯ				
ADD				СЛОЖЕНИЕ
r,r/m	0000001W	md reg r/m	(disp8/16)	r<-r+r/m
r/m,r	0000000W	md reg r/m	(disp8/16)	r/m<-r+r/m
r/m,d	1000000W	md 000 r/m	(disp8/16)d8/16	r/m<-r/m+d
r16/m16,d8	10000011	md 000 r/m	data L	r/m<-r/m+d
ac,d	0000010W	data L	(data H)	ac<-ac+d
ADC				СЛОЖЕНИЕ с переносом
r,r/m	0001001W	md reg r/m	(disp8/16)	r<-r+r/m+CF
r/m,r	0001000W	md reg r/m	(disp8/16)	r/m<-r+r/m+CF
r16/m16,d8	10000011	md 010 r/m	data L	r/m<-r/m+d+CF
ac,d	0001010W	data L	(data H)	ac<-ca+d+CF
INC				ИНКРЕМЕНТ
r/m	1111111W	md 000 r/m	(disp8/16)	r/m<-r/m+1
r	01000reg			r<-r+1
AAA	00110111			ASCII-КОРРЕКЦИЯ для сложения
DAA	00100111			ДЕСЯТИЧНАЯ КОРРЕКЦИЯ для сложения
КОМАНДЫ ВЫЧИТАНИЯ				
SUB				ВЫЧИТАНИЕ
r,r/m	0010101W	md reg r/m	(disp8/16)	r<-r-r/m
r/m,m	0010100W	md reg r/m	(diso8/16)	r/m<-r/m-r
r/m,d	1000000W	md 101 r/m	(disp8/16)d8/16	r/m<-r/m-d
r16/m16,d8	10000011	md 101 r/m	data L	r/m<-r/m-d
ac,d	0010110W	data L	(data H)	ac<-ca-d
SBB				ВЫЧИТАНИЕ с заемом
r,r/m	0001101W	md reg r/m	(disp8/16)	r<-r-r/m-CF
r/m,r	0001100W	md reg r/m	(disp8/16)	r/m<-r/m-r-CF
r/m,d	1000000W	md 011 r/m	(disp8/16)d8/16	r/m<-r/m-d-CF
r16/m16,d8	10000011	md 011 r/m	data L	r/m<-r/m-d-CF
ac,d	0001110W	data L	(data H)	ac<-ac-d-CF
DEC				ДЕКРЕМЕНТ
r/m	1111111W	md 001 r/m	(disp8/16)	r/m<-r/m-1
r	01001reg			r<-r-1
NEG				
r/m	1111011W	md 011 r/m	(disp8/16)	СМЕНА знака числа
CMP				СРАВНЕНИЕ
r,r/m	0011101W	md reg r/m	(disp8/16)	r-r/m
r/m,r	0011100W	md reg r/m	(disp8/16)	r/m-r
r/m,d	1000000W	md 111 r/m	(disp8/16)	r/m-d
r16/m16,d8	10000011	md 111 r/m	data L	r/m-d
ac,d	0011110W	data L	(data H)	ac-d

КОМАНДА	БАЙТЫ КОДА КОМАНДЫ			СИМВОЛИЧЕСКОЕ ОПИСАНИЕ И/ИЛИ СОДЕРЖАНИЕ ОПЕРАЦИИ
	B1	B2	B3-B6	
AAS	00111111			ASCII-КОРРЕКЦИЯ для вычитания
DAS	00101111			ДЕСЯТИЧНАЯ КОРРЕКЦИЯ для вычитания
КОМАНДЫ УМНОЖЕНИЯ				
MUL r/m	1111011W	md 100 r/m	(diap8/16)	УМНОЖЕНИЕ без знака
IMUL r/m	1111011W	md 101 r/m	(disp8/16)	УМНОЖЕНИЕ со знаком
AAM	11010100	00001010		ASCII-КОРРЕКЦИЯ для умножения
КОМАНДЫ ДЕЛЕНИЯ				
DIV r/m	1111011W	md 110 r/m	(diap8/16)	ДЕЛЕНИЕ без знака
IDIV r/m	1111011W	mr 101 r/m	(disp8/16)	ДЕЛЕНИЕ со знаком
AAD	11010101	00001010		ASCII-КОРРЕКЦИЯ для деления
КОМАНДЫ РАСШИРЕНИЯ ЗНАКА				
CBW	10011000			ПРЕОБРАЗОВАНИЕ байта в слово
CWD	10011001			ПРЕОБРАЗОВАНИЕ слова в двойное слово
ЛОГИЧЕСКИЕ КОМАНДЫ				
AND				КОНЪЮНКЦИЯ (И)
r,r/m	0010001W	md reg r/m	(disp8/16)	r<-r∧r/m
r/m,r	0010000W	md reg r/m	(disp8/16)	r/m<-r∧r/m
r/m,d	1000000W	md 100 r/m	data L data H	r/m<-r/m∧d
ac,d	0010010W	data L	(data H)	ac<-ac∧d
OR				ДИЗЪЮНКЦИЯ (ИЛИ)
r,r/m	0000101W	md reg r/m	(disp8/16)	r<-r∨r/m
r/m,r	0000100W	md reg r/m	(disp8/16)	r/m<-r∨r/m
r/m,d	1000000W	md 001 r/m	(disp8/16)d8/16	r/m<-r/m∨d
ac,d	0000110W	data L	(data H)	ac<-ac∨d
XOR				СУММИРОВАНИЕ ПО МОДУЛЮ 2 (исключающее ИЛИ)
r,r/m	0011001W	md reg r/m	(disp8/16)	r<-r⊕r/m
r/m,r	0011000W	md reg r/m	(disp8/16)	r/m<-r⊕r/m
r/m,d	1000000W	md 110 r/m	(disp8/16)d8/16	r/m<-r/m⊕d
ac,d	0011010W	data L	(data H)	ac<-ac⊕d
NOT r/m	1111011W	md 010 r/m	(disp8/16)	(НЕ) ИНВЕРСИЯ r/m<-r/m
TEST				ПРОВЕРКА (неразрушающее И)
r,r/m	1000010W	md reg r/m	(disp8/16)	r∧r/m
r/m,r	1000010W	md reg r/m	(disp8/16)	r/m∧r
r/m,d	1111011W	md 000 r/m	(disp8/16)d8/16	r/m∧d
ac,d	1010100W	data L	(data H)	ac∧d
КОМАНДЫ СДВИГА				

КОМАНДА	БАЙТЫ КОДА КОМАНДЫ			СИМВОЛИЧЕСКОЕ ОПИСАНИЕ И/ИЛИ СОДЕРЖАНИЕ ОПЕРАЦИИ
	B1	B2	B3-B6	
SAL				АРИФМЕТИЧЕСКИЙ сдвиг
r/m,1	1101000W	md 100 r/m	(diap8/16)	влево
r/m,CL	1101001W	md 100 r/m	(disp8/16)	
SHL				ЛОГИЧЕСКИЙ сдвиг влево
r/m,1	1101000W	md 100 r/m	(diap8/16)	
r/m,CL	1101001W	md 100 r/m	(disp8/16)	АРИФМЕТИЧЕСКИЙ сдвиг
SAR				вправо
r/m,1	1101000W	md 111 r/m	(diap8/16)	
r/m,CL	1101001W	md 111 r/m	(disp8/16)	ЛОГИЧЕСКИЙ сдвиг
SHR r/m,1	1101000W	md 101 r/m	(diap8/16)	вправо
r/m,CL	1101001W	md 101 r/m	(disp8/16)	
КОМАНДЫ ЦИКЛИЧЕСКОГО СДВИГА				
ROL				ЦИКЛИЧЕСКИЙ сдвиг
r/m,1	1101000W	md 000 r/m	(diap8/16)	влево
r/m,CL	1101001W	md 000 r/m	(disp8/16)	
ROR				ЦИКЛИЧЕСКИЙ сдвиг
r/m,1	1101000W	md 001 r/m	(diap8/16)	вправо
r/m,CL	1101001W	md 001 r/m	(disp8/16)	
RCL				ЦИКЛИЧЕСКИЙ сдвиг
r/m,1	1101000W	md 010 r/m	(diap8/16)	влево через CF
r/m,CL	1101001W	md 010 r/m	(disp8/16)	
RCR				ЦИКЛИЧЕСКИЙ сдвиг
r/m,1	1101000W	md 011 r/m	(diap8/16)	вправо через CF
r/m,CL	1101001W	md 011 r/m	(disp8/16)	
КОМАНДЫ БЕЗУСЛОВНОЙ ПЕРЕДАЧИ УПРАВЛЕНИЯ				
CALL				
NEAR sbr	11101000	diff L	diff H	ВЫЗОВ прямой
				IP<-IP+diff
NEAR				ВЫЗОВ косвенный
r16/m16	11111111	md 010 r/m	(disp8/16)	IP<-IP+r/m
FAR sbr	10011010	IP-L IP-H	CS-L CS-H	ВЫЗОВ прямой
				IP<-IP-L,IP-H;
				CS<-CS-L,CS-H
FAR m32	11111111	md 011 r/m	(disp8/16)	IP<-m16; CS<-m16
RET (NEAR)	11000011			ВОЗВРАТ из подпрограммы
				ВОЗВРАТ с прибавлением
d(NEAR)	11000011	data L	data H	константы к SP
				ВОЗВРАТ из подпрограммы
(FAR)	11001011			ВОЗВРАТ с прибавлением
				константы к SP
d(FAR)	11001010	data L	data H	ПЕРЕХОД прямой
				короткий IP<-IP+diff
JMP				IP<-IP+diff L
SHORT label	11101011	diff L		длинный
NEAR label	11101001	diff L	diff H	IP<-IP-L,IP-H
FAR label	11101010	IP-L IP-H	CS-L CS-H	CS<-CS-L,CS-H
				ПЕРЕХОД косвенный

КОМАНДА	БАЙТЫ КОДА КОМАНДЫ			СИМВОЛИЧЕСКОЕ ОПИСАНИЕ И/ИЛИ СОДЕРЖАНИЕ ОПЕРАЦИИ
	B1	B2	B3-B6	
NEAR r16/m16 FAR m32	11111111 11111111	md 100 r/m md 101 r/m	(disp8/16) (disp8/16)	IP<-IP+r/m IP<-m16; CS<-m16
КОМАНДЫ УСЛОВНОЙ ПЕРЕДАЧИ УПРАВЛЕНИЯ				
JA label	01110111	diff L		ВЫШЕ IP<-IP+diff L
JAE label	01110011	diff L		ВЫШЕ ИЛИ РАВНО IP<-IP+diff L
JB label	01110010	diff L		НИЖЕ IP<-IP+diff L
JBE label	01110110	diff L		НИЖЕ ИЛИ РАВНО IP<-IP+diff L
JC label	01110010	diff L		ЕСТЬ ПЕРЕНОС IP<-IP+diff L
JSXZ label	11100011	diff L		СХ РАВЕН НУЛЮ IP<-IP+diff L
JE label	01110111	diff L		РАВНО IP<-IP+diff L
*JG label	01111111	diff L diff L		БОЛЬШЕ IP<-IP+diff L
*JGE label	01111101	diff L		БОЛЬШЕ-РАВНО IP<-IP+diff L
*JL label	01111100	diff L		МЕНЬШЕ IP<-IP+diff L
*JLE label	01111110	diff L		МЕНЬШЕ-РАВНО IP<-IP+diff L
JNA label	01110110	diff L		НЕ ВЫШЕ IP<-IP+diffL
JNAE label	01110011	diff L		НЕ ВЫШЕ И НЕ РАВНО IP<-IP+diff L
JNB label	01110111	diff L		НЕ НИЖЕ IP<-IP+diffL
JNBE label	01110111	diff L		НЕ НИЖЕ И НЕ РАВНО IP<-IP+diff L
JNC label	01110011	diff L		НЕТ ПЕРЕНОСА IP<-IP+diff L
JNE label	01110101	diff L		НЕ РАВНО IP<-IP+diff L
*JNG label	01111110	diff L		НЕ БОЛЬШЕ IP<-IP+diff L
*JNGE label	01111100	diff L		НЕ БОЛЬШЕ ИЛИ РАВНО IP<-IP+diff L
*JNL label	01111101	diff L		НЕ МЕНЬШЕ IP<-IP+diff L
*JNLE label	01111111	diff L		НЕ МЕНЬШЕ И НЕ РАВНО IP<-IP+diff L
*JNO label	01110001	diff L		НЕТ ПЕРЕПОЛНЕНИЯ IP<-IP+diff L
JNP label	01111011	diff L		НЕТ ЧЕТНОСТИ IP<-IP+diff L
*JNS label	01111001	diff L		РЕЗ-Т ПОЛОЖИТЕЛЬНЫЙ IP<-IP+diff L
JNZ label	01110101	diff L		НЕ НУЛЬ IP<-IP+diffL
*JO label	01110000			ПЕРЕПОЛНЕНИЕ

КОМАНДА	БАЙТЫ КОДА КОМАНДЫ			СИМВОЛИЧЕСКОЕ ОПИСАНИЕ И/ИЛИ СОДЕРЖАНИЕ ОПЕРАЦИИ
	B1	B2	B3-B6	
JP label	01111010	diff L		IP<-IP+diff L ЕСТЬ ЧЕТНОСТЬ
JPE label	01111010	diff L		IP<-IP+diff L ПАРИТЕТ ЧЕТНЫЙ
JPO label	01111011	diff L		IP<-IP+diff L ПАРИТЕТ НЕЧЕТНЫЙ
JS label	01111000	diff L		IP<-IP+diff L РЕЗ-Т ОТРИЦАТЕЛЬНЫЙ
JZ label	01110100	diff L		IP<-IP+diff L НУЛЬ IP<-IP+diff L
Примечание: * - Команды действия над числами со знаком.				
КОМАНДЫ УПРАВЛЕНИЯ ЦИКЛАМИ				
LOOP label	11100010	diff L		ЗАЦИКЛИТЬ IP<-IP+diff L
LOOPE/LOOPZ label	11100001	diff L		ЗАЦИКЛИТЬ, ЕСЛИ РАВНО IP<-IP+diff L
LOOPNE/ LOOPNZ label	11100000	diff L		ЗАЦИКЛИТЬ, ЕСЛИ НЕ РАВНО IP<-IP+diff L
КОМАНДЫ ПРЕРЫВАНИЯ				
INT type	11001101			ПРЕРЫВАНИЕ заданного типа
INTO	11001110			ПРЕРЫВАНИЕ по переполнению
IRET	11001111			ВОЗВРАТ из прерывания IP<-IP+diff L
КОМАНДЫ УПРАВЛЕНИЯ ФЛАГАМИ				
STC	11111001			УСТАНОВКА флага переноса CF<-1
CLC	11111000			СБРОС флага переноса CF<-0
CMC	11110101			ИНВЕРСИЯ флага переноса CF<-CF
STD	11111101			УСТАНОВКА флага направления DF<-1
CLD	11111100			СБРОС флага направления DF<-0
STI	11111011			РАЗРЕШЕНИЕ прерывания IF<-1
CLI	11111010			ЗАПРЕТ прерывания IF<-0
ВНЕШНЯЯ СИНХРОНИЗАЦИЯ				
HLT	11110100			ОСТАНОВ
WAIT	00111011			ОЖИДАНИЕ
ESC OPCODE,r/m	11011XXX	md YYYr/m	(disp8/16)	ПЕРЕКЛЮЧЕНИЕ НА СОПРОЦЕССОР

КОМАНДА	БАЙТЫ КОДА КОМАНДЫ			СИМВОЛИЧЕСКОЕ ОПИСАНИЕ И/ИЛИ СОДЕРЖАНИЕ ОПЕРАЦИИ
	B1	B2	B3-B6	
LOCK NOP	11110000 10010000			БЛОКИРОВКА ШИНЫ ХОЛОСТОЙ ХОД