№1

#include <iostream>

using namespace std;

int main()

{

int a = 21, b = 47;

cout << a<< ' ' << b<< endl;

\_\_asm

{

mov eax, a

mov ebx, b

mov b, eax

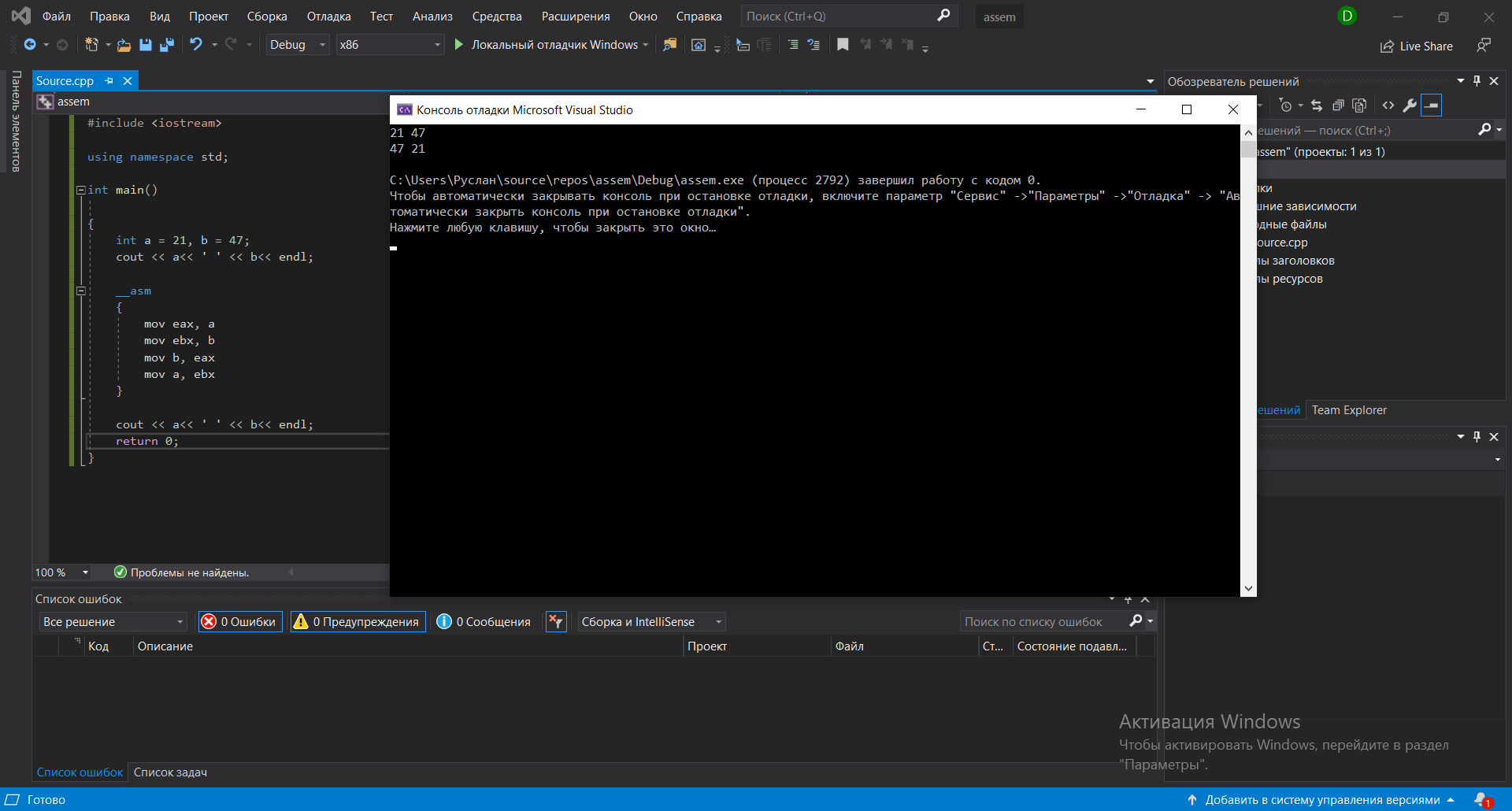
mov a, ebx

}

cout << a<< ' ' << b<< endl;

return 0;

}



№2

#include <iostream>

using namespace std;

int main()

{

int a = 3, b = 12, result;

\_\_asm

{

mov eax, b

cdq

idiv a

neg eax

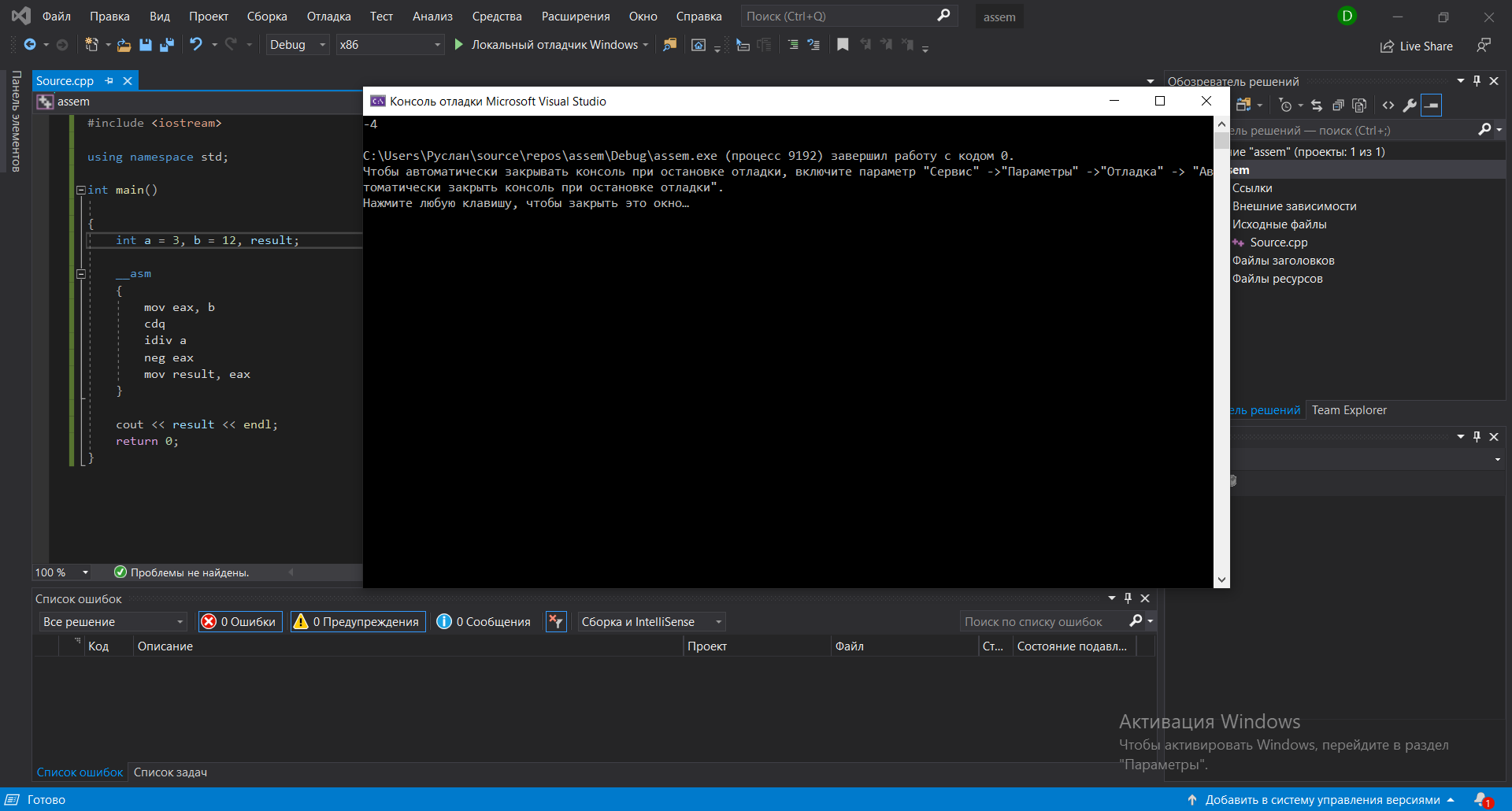
mov result, eax

}

cout << result << endl;

return 0;

}



№3

#include <iostream>

using namespace std;

int main()

{

int a = 2, b = 3, x = 2, c = 5, result;

\_\_asm

{

mov eax, 2

imul a

imul c

mov ecx, eax

mov eax, b

idiv x

sub ecx, eax

sub ecx, 12

mov eax, ecx

cdq

mov ecx, x

imul c

add ecx, a

idiv ecx

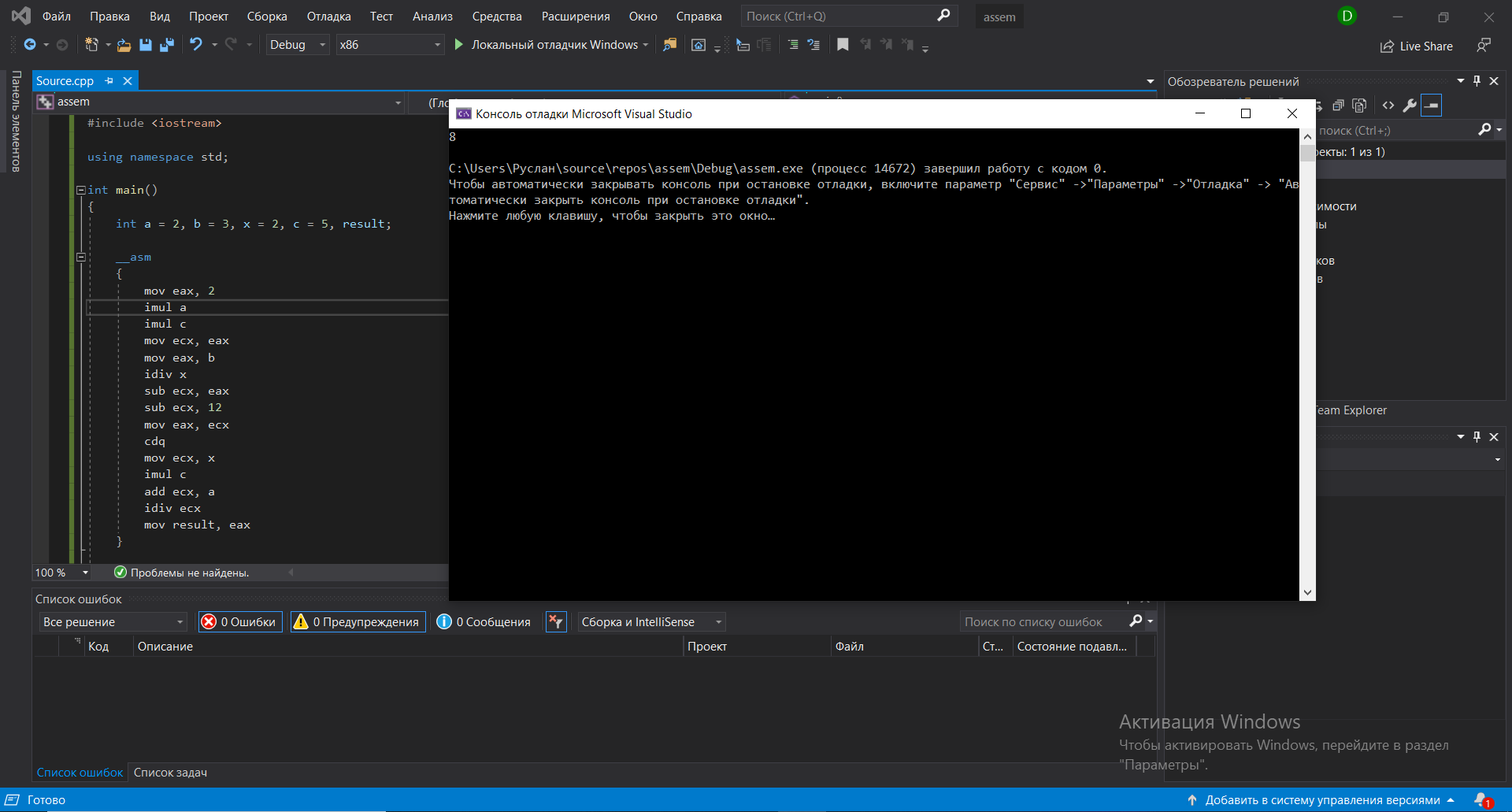
mov result, eax

}

cout << result << endl;

return 0;

}



№4

#include <iostream>

using namespace std;

typedef unsigned \_\_int32 DWORD;

int main()

{

short int a = 128, result;

\_\_asm

{

mov ax, a

xchg al, ah

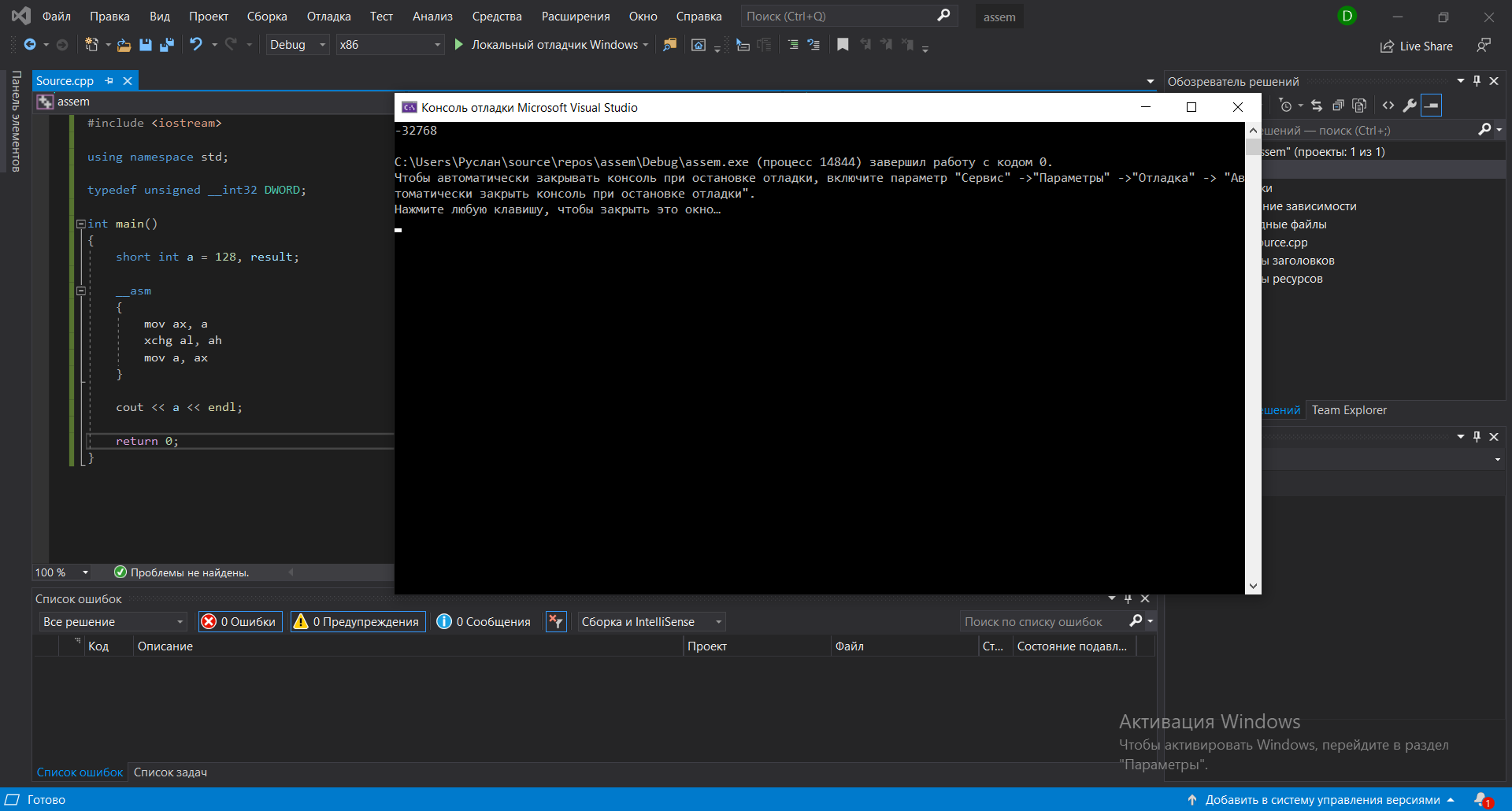
mov a, ax

}

cout << a << endl;

return 0;

}



Контрольные вопросы

1)      Форматы записи команд Ассемблера Мнемоническая команда Ассемблера дает указание, какое действие должен выполнять данный оператор. Операнд может определять либо начальное значение данных, либо элементы, над которыми выполняются действия по соответствующей команде. При разработке программ рекомендуется использовать комментарии. Комментарий всегда начинается с символа «;». Ассемблер полагает, что все символы, которые находятся справа от этого символа являются комментарием.

2)      Количество команд и их сложность, безусловно, являются важнейшими факторами, однако не меньшую роль при выборе АСК играет ответ на вопрос о том, где могут храниться операнды и каким образом к ним осуществляется доступ. С этих позиций различают следующие виды архитектур системы команд: - стековую; - аккумуляторную; - регистровую; - с выделенным доступом к памяти. Выбор той или иной архитектуры влияет на принципиальные моменты: сколько адресов будет содержать адресная часть команд, какова будет длина этих адресов, насколько просто будет происходить доступ к операндам и какой, в конечном итоге, будет общая длина команд.

2.1)  Результат операции заносится в стек.

2.2) После выполнения команды обработки результат находится в аккумуляторе и, если он не является операндом для последующей команды, его требуется сохранить в ячейке памяти.

2.3) Регистровая архитектура допускает расположение операндов в одной из двух запоминающих сред: основной памяти или регистрах. С учетом возможного размещения операндов в рамках регистровых АСК выделяют три подвида команд обработки:

- регистр-регистр;

- регистр-память;

- память-память.

3) Предложения, составляющие программу, могут представлять собой синтаксическую конструкцию, соответствующую команде, макрокоманде, директиве или комментарию. Для того чтобы транслятор ассемблера мог распознать их, они должны формироваться по определенным синтаксическим правилам. Для этого лучше всего использовать формальное описание синтаксиса языка наподобие правил грамматики. Наиболее распространенные способы подобного описания языка программирования – синтаксические диаграммы и расширенные формы Бэкуса-Наура. Для практического использования более удобны синтаксические диаграммы. К примеру, синтаксис предложений ассемблера можно описать с помощью синтаксических диаграмм, показанных на следующих рисунках. Язык ассемблера — тип языка программирования низкого уровня. Команды языка ассемблера один в один соответствуют командам процессора и представляют собой удобную символьную форму записи (мнемокод) команд и аргументов. Язык ассемблера обеспечивает связывание частей программы и данных через метки, выполняемое при ассемблировании (для каждой метки высчитывается адрес, после чего каждое вхождение метки заменяется на этот адрес). Каждая модель процессора имеет свой набор команд и соответствующий ему язык (или диалект) ассемблера. Обычно программы или участки кода пишутся на языке ассемблера в случаях, когда разработчику критически важно оптимизировать такие параметры, как быстродействие (например, при создании драйверов) и размер кода (загрузочные секторы, программное обеспечение для микроконтроллеров и процессоров с ограниченными ресурсами, вирусы, навесные защиты). Наиболее распространенные способы подобного описания языка программирования – синтаксические диаграммы и расширенные формы Бэкуса-Наура. Для практического использования более удобны синтаксические диаграммы. К примеру, синтаксис предложений ассемблера можно описать с помощью синтаксических диаграмм

4) Оператор PTR используется совместно с атрибутами типа BYTE, WORD или DWORD для локальной отмены определенных типов (DB, DW или DD) или с атрибутами NEAR или FAR для отмены значения дистанции по умолчанию. Формат оператора следующий:

тип PTR выражение

В поле «тип» указывается новый атрибут, например BYTE. Выражение имеет ссылку на переменную или константу.

5) Регистр EAX (аккумулятор) – автоматически применяется при операциях умножения, деления и при работе с портами ввода-вывода. Его использование в арифметических, логических и некоторых других операциях позволяет увеличить скорость их выполнения. Использует для записи возвращаемого значения из процедуры.

Регистр EBX (регистр базы) – может содержать адреса элементов оперативной памяти. По умолчанию эти адреса будут представлять собой смещение в сегменте данных.

Регистр ECX (счетчик) – используется в операциях повторения, например в циклах, в строковых командах и т.д.

Регистр EDX (регистр данных) – является единственным элементом, который может хранить адреса портов ввода-вывода в командах типа IN (получить из порта) и OUT (вывести в порт). Без его помощи невозможно обратиться к портам с адресами в адресном пространстве больше 1 байта. Автоматически применяется также в операциях умножения и деления.

6) ADD и SUB

7) В знаковой операнде самый первый бит несет информацию о знаке.

0 это “+”

1 это -

8) Да

9) Делимое находится в регистре AX, а делитель - в

байте памяти или в однобайтовом регистре. После деления остаток получается в

регистре AH, а частное -в AL.

10) Целая часть частного помещается в регистр AL, AX или EAX в зависимости от заданного размера делителя (8, 16 или 32 бита). При этом остаток от целочисленного деления помещается в регистр AH, DX или EDX, соответственно.

11) Команды условных переходов — проверяют состояние одного или нескольких битов регистра признаков и при выполнении условия осуществляют передачу программного управления в другую точку кода, задаваемую операндом. Указанный класс команд не запоминает информацию для возврата. Операнд определяет адрес команды, которой должно быть передано управление.

|  |  |  |  |
| --- | --- | --- | --- |
| **Команда** | **Операнды** | **Пояснение** | **Описание** |
| JCXZ | метка | if(ECX==0)  EIP=метка; | Переход при ECX=0 |
| JC | if(CF==1)  EIP=метка; | Переход при переносе (CF=1) |
| JNC | if(CF==0)  EIP=метка; | Переход при отсутствии переноса (CF=0) |
| JS | if(SF==1)  EIP=метка; | Переход при отрицательном результате (SF=1) |
| JNS | if(SF==0)  EIP=метка; | Переход при неотрицательном результате (SF=0) |
| JE  JZ | if(ZF==1)  EIP=метка; | Переход при нулевом результате (ZF=1) |
| JNE  JNZ | if(ZF==0)  EIP=метка; | Переход при ненулевом результате (ZF=0) |
| JP  JPE | if(PF==1)  EIP=метка; | Переход по четности (PF=1) |
| JNP  JPO | if(PF==0)  EIP=метка; | Переход по нечетности (PF=0) |
| JO  JNZ | if(OF==1)  EIP=метка; | Переход при переполнении (OF=1) |
| JNO  JNZ | if(OF==0)  EIP=метка; | Переход при отсутствии переполнения (OF=0) |

12)

|  |  |
| --- | --- |
| BTR | Тестирование бита с номером из операнда 2 в операнде 1 и перенос его значения во флаг CF. Само значение бита сбрасывается в 0 |
| BTS | Тестирование бита с номером из операнда 2 в операнде 1 и перенос его значения во флаг CF. Само значение бита устанавливается в 1 |

13) Поскольку 555516 равно 01010101010101012, результатом операции x & 555516 является число, в котором все нечетные биты соответствуют нечетным битам числа x. Аналогично, результатом операции (x \gt \gt \gt  1) & 555516 является число, в котором все нечетные биты соответствуют четным битам x. Четные биты результата в обоих случаях равны нулю. Мысленно разобьем двоичную запись нашего числа x на группы по 2 бита. Результатом операции x & 555516+(x \gt \gt \gt  1) & 555516 будет такое число, что если разбить его двоичную запись на группы по два бита, значение каждой группы соответствует количеству единичных битов в соответствующей паре битов числа x.

14) С помощью команд shr и sar

15) Когда нужно переслать меньшее по длине значение в большую по длине переменную или регистр.

**Команда** CBW

**Команда** CWD

**Команда** CWDE

**Команда** CDQ

**Команда** MOVZX

**Команда** MOVSX