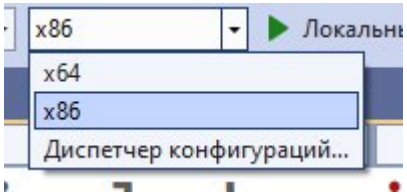
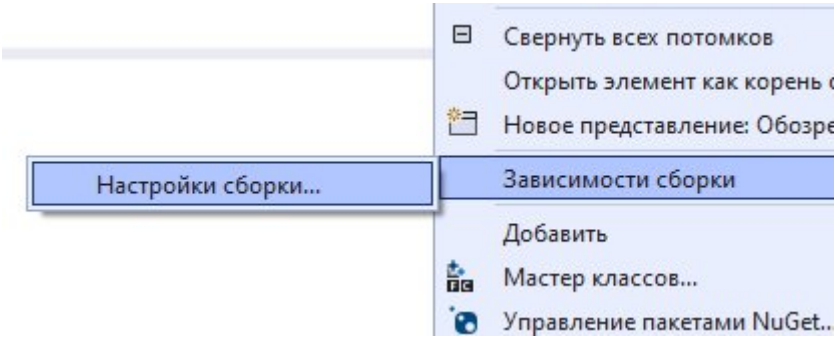
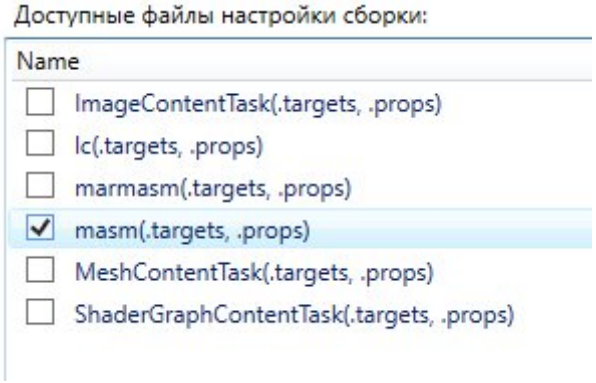
Метки и переходы

Запуск кода

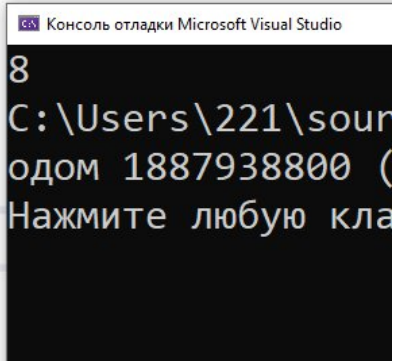

x86
Локальны
x64
x86
Диспетчер конфигураций...


Свернуть всех потомков
Открыть элемент как корень
Новое представление: Обозре
Настройки сборки...
Зависимости сборки
Добавить
Мастер классов...
Управление пакетами NuGet...


Доступные файлы настройки сборки:
Name
ImageContentTask(.targets, .props)
lc(.targets, .props)
marmasm(.targets, .props)
masm(.targets, .props)
MeshContentTask(.targets, .props)
ShaderGraphContentTask(.targets, .props)

Запуск кода

```
#include <iostream>
using namespace std;
int main()
{
    int a = 5, b = 3, sum;
    __asm {
        mov eax, a;
        mov ebx, b;
        add eax, ebx;
        mov sum, eax;
    }
    cout << sum;
}
```



Что такое метки?

Метка в ассемблере — это имя, которое указывает на конкретное место в программе, на которое можно сослаться в коде. Метки обычно располагаются в начале строки и заканчиваются двоеточием :. Они могут быть использованы как для безусловных, так и для условных переходов.

Что такое метки?

Пример:

```
start:
   ; Код, который будет выполнен первым
   MOV AX, 5
   ; Далее выполняется переход на метку end
   JMP end

end:
   ; Завершающий код
   MOV BX, 10
```

Переходы

Переходы — это команды, которые изменяют порядок выполнения программы, перенаправляя ее выполнение на указанную метку. В Assembler существует несколько видов переходов, каждый из которых используется в различных ситуациях.

Безусловный переход (JMP)

Безусловный переход JMP позволяет передать управление на любую метку в программе, не проверяя никаких условий.:

```
start:
   MOV AX, 5
   JMP end


end:
   MOV BX, 10
```

Условные переходы

Условные переходы изменяют порядок выполнения программы в зависимости от различных условий и выполняют переход, если условие верно.

Условные переходы

JE (Jump if Equal) — Переход, если операнды равны.
JNE (Jump if Not Equal) — Переход, если операнды не равны.
JG (Jump if Greater) — Переход, если первый операнд больше.
JL (Jump if Less) — Переход, если первый операнд меньше.
JZ (Jump if Zero) — Переход, если операнд равен нулю.
JNZ (Jump if Not Zero) — Переход, если операнд не равен нулю.

Условные переходы

Пример:

```
MOV AX, 5
MOV BX, 5
CMP AX, BX   ; Сравниваем значения AX и BX
JE equal     ; Если равны, выполняем переход на метку equal

MOV CX, 10
JMP end

equal:
MOV CX, 20

end:
```

Применение меток и переходов

1. Циклы: Использование меток и переходов позволяет создавать циклы, которые повторяют блоки кода до тех пор, пока не будет выполнено определенное условие.

Циклы

Пример цикла с использованием меток и условного перехода:

```
MOV AX, 0
loop_start:
   INC AX
   CMP AX, 10
   JL loop_start  ; Пока AX < 10, продолжаем цикл
```

Применение меток и переходов

2. Условные операторы: Метки и переходы позволяют реализовать условные конструкции, аналогичные if и else в высокоуровневых языках.

Условные операторы

```
MOV AX, 5
CMP AX, 10
JGE greater_than_10
; Другие операции
JMP end

greater_than_10:
; Действия, если AX >= 10

end:
```

Применение меток и переходов

3. Выход из программы: Переходы могут использоваться для завершения выполнения программы. Например, можно сделать переход на метку, которая завершит программу или выполнит очистку ресурсов.

Применение меток и переходов

4. Обработка ошибок: Если в процессе выполнения программы возникает ошибка или неожиданная ситуация, можно использовать метки и переходы для перехода в блок обработки ошибок.

Задание

Написать программу, выводящую 1, если число введенное пользователем положительное и 0,если отрицательное. Использовать условные операторы и метки.