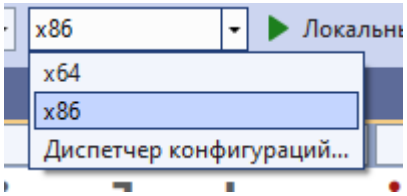
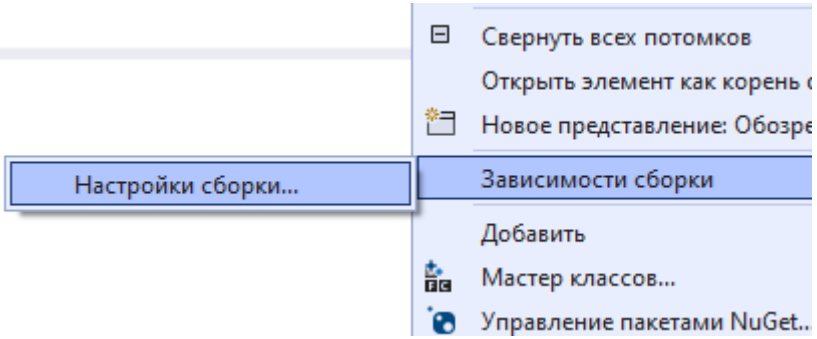
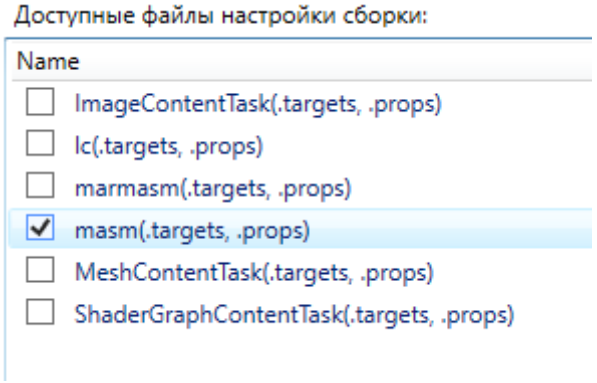
Арифметические операции

Запуск кода

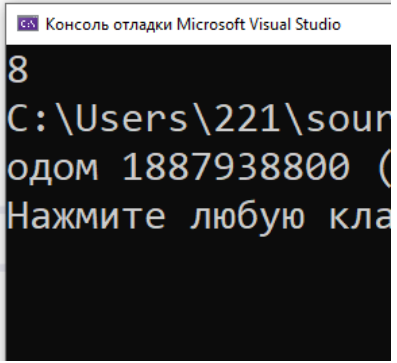

x86
Локальны
x64
x86
Диспетчер конфигураций...


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;
}
```



ADD

Инструкция add выполняет сложение двух операндов

```
add destination, source
//destination = destination + source
```

ADD

Пример:

```
__asm{
    mov eax, 6
    mov ebx, 10
    add eax, ebx  ; eax = eax + ebx
}
```

SUB

Интсрукция sub находит разницу (вычитание) двух операндов

```
sub destination, source
//destination = destination - source
```

SUB

Пример:

```
__asm{
    mov eax, 6
    mov ebx, 10
    sub eax, ebx  ; eax = eax - ebx
}
```

INC

Увеличивает на единицу значение операнда

```
inc ecx; EAX++
```

MUL и IMUL

Инструкции mul и imul умножает два целых числа. imul умножает числа со знаком, а mul - беззнаковые числа. Инструкция imul имеет следующую общую форму:

```
imul dest, source
//dest = dest * source
```

MUL и IMUL

Первый операнд - всегда регистр. Второй операнд может представлять регистр, переменную или константу. Оба операнда должны совпадать по размеру и могут быть 16-, 32- и 64-разрядными,

MUL и IMUL

Пример:

```
__asm{
    mov eax, 3
    imul eax, 5 ; EAX = EAX * 5 = 3 * 5 = 15
}
```

MUL и IMUL

Инструкции mul и imul поддерживают также форму с одним операндом:

```
mul/imul operand8   ; результат в AX
mul/imul operand16  ; результат в DX:AX
mul/imul operand32  ; результат в EDX:EAX
mul/imul operand64  ; результат в RDX:RAX
```

DIV и IDIV

idiv делит два числа со знаком, а div - беззнаковые числа. Эти инструкции принимают следующие формы:

```
div reg8
div reg16
div reg32
```

DIV и IDIV

Если операнд 32-разрядный, частное в EAX, а остаток в EDX

Инструкция idiv имеет аналогичное действие, только в качестве операндов принимает числа со знаком.

DIV и IDIV

Пример деления

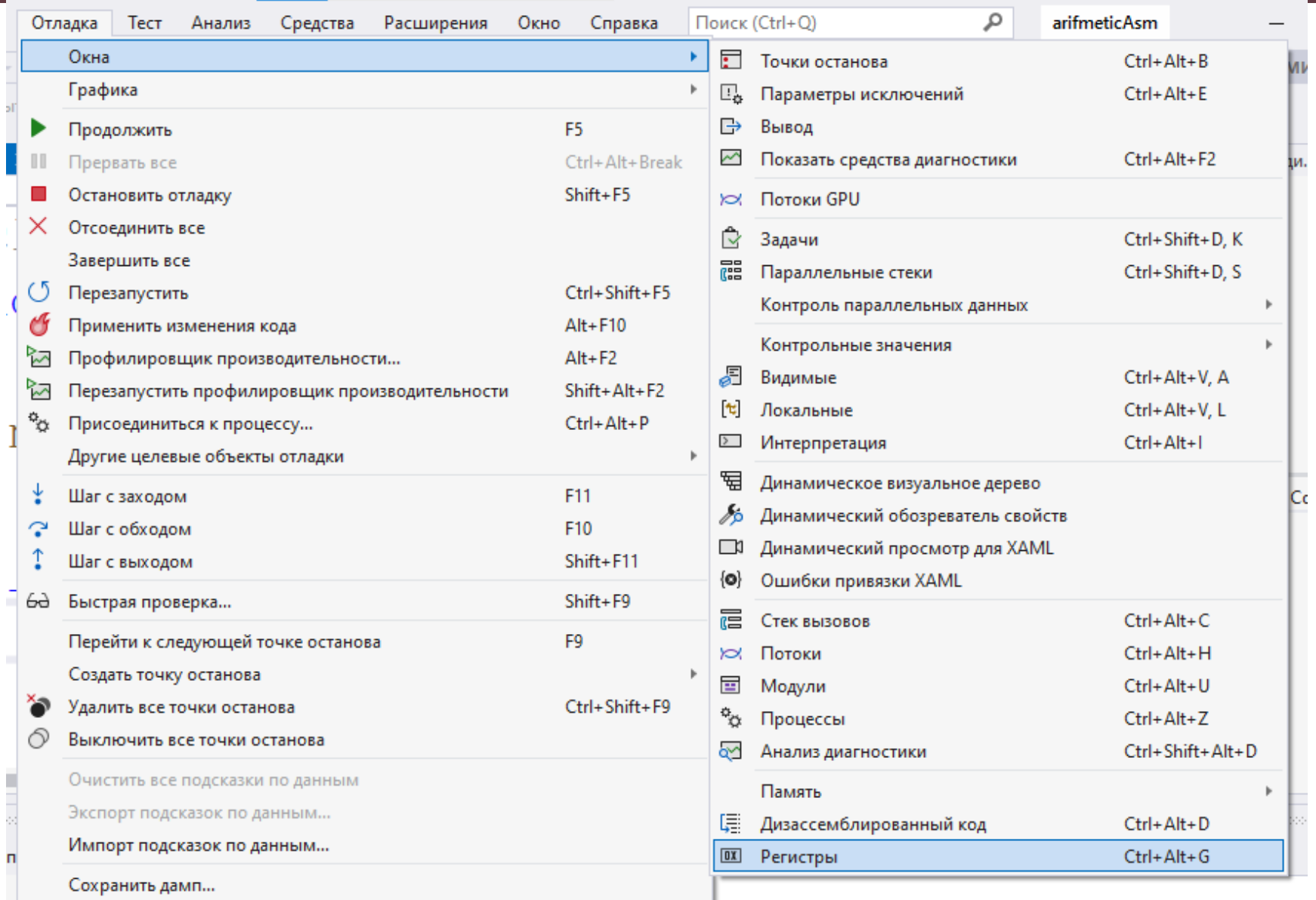
```
__asm{
    xor eax, eax
    mov eax, 22  ; 32-разрядный регистр
    mov ebx, 5   ; 32-разрядный регистр
    div ebx      ; EAX =4 (результат), EDX = 2 (остаток)
}
```

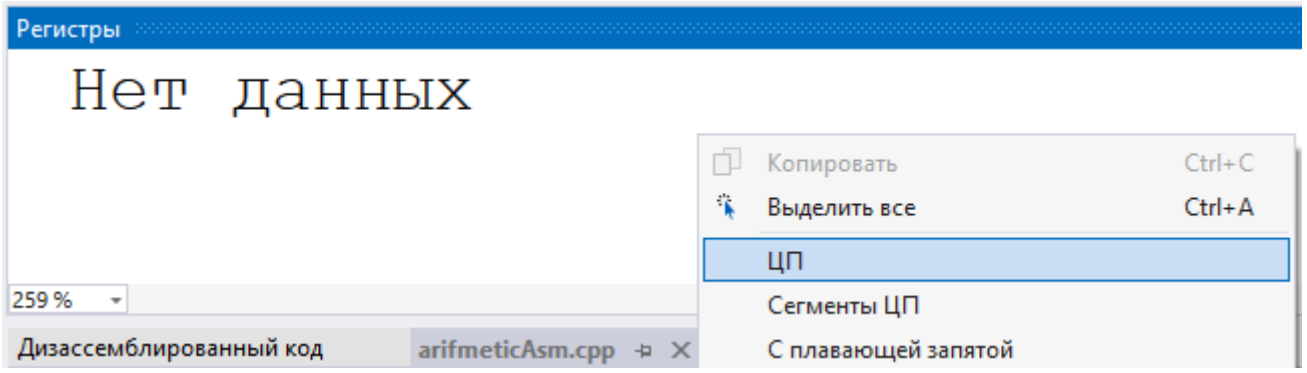
Просмотр состояний регистров

Бывает полезным во время выполнения ознакомится с содержимым регистров. Это можно сделать в режиме отладки.

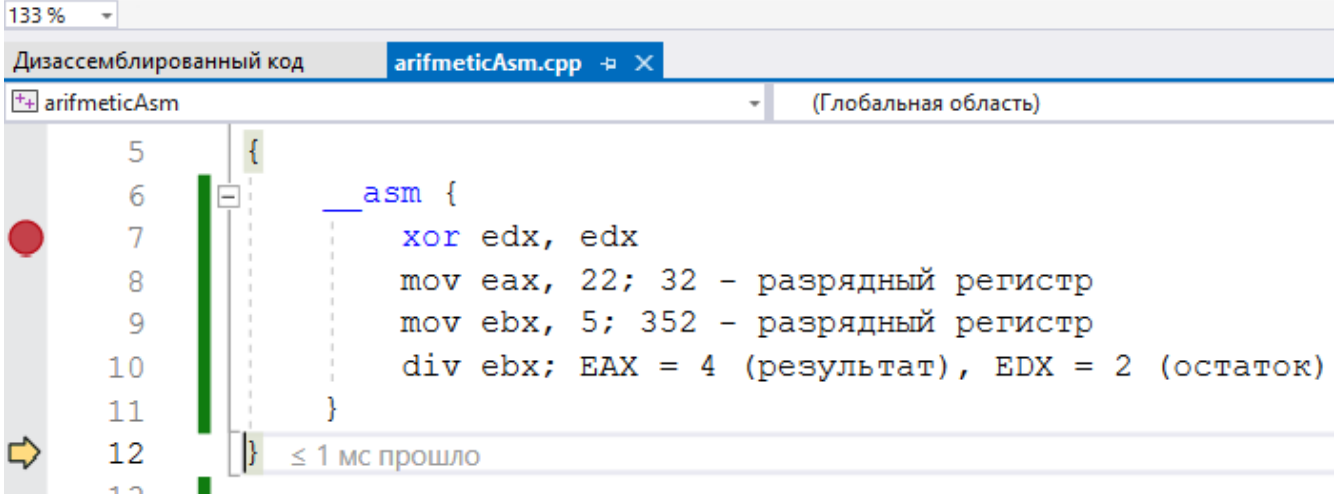
Для этого необходимо запустить отладку → Окна → Регистры

В окне с регистрами нажать ПКМ → выбрать необходимые регистры (нам понадобится ЦП)

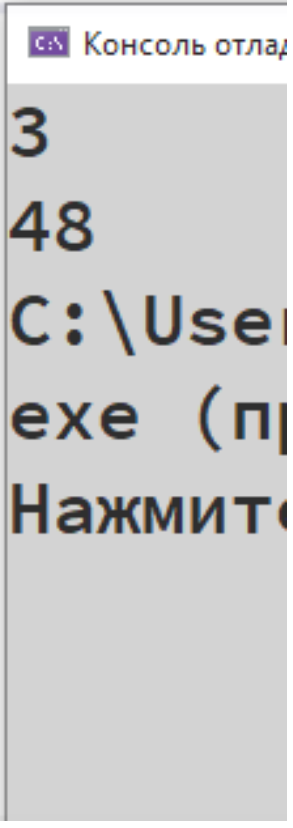

Отладка
Тест
Анализ
Средства
Расширения
Окно
Справка
Поиск (Ctrl+Q)
arifmeticAsm
Окна
Графика
Продолжить F5
Прервать все Ctrl+Alt+Break
Остановить отладку Shift+F5
Отсоединить все
Завершить все
Перезапустить Ctrl+Shift+F5
Применить изменения кода Alt+F10
Профилировщик производительности... Alt+F2
Перезапустить профилировщик производительности Shift+Alt+F2
Присоединиться к процессу... Ctrl+Alt+P
Другие целевые объекты отладки
Шаг с заходом F11
Шаг с обходом F10
Шаг с выходом Shift+F11
Быстрая проверка... Shift+F9
Перейти к следующей точке останова F9
Создать точку останова
Удалить все точки останова Ctrl+Shift+F9
Выключить все точки останова
Очистить все подсказки по данным
Экспорт подсказок по данным...
Импорт подсказок по данным...
Сохранить дамп...
Точки останова Ctrl+Alt+B
Параметры исключений Ctrl+Alt+E
Вывод
Показать средства диагностики Ctrl+Alt+F2
Потоки GPU
Задачи Ctrl+Shift+D, K
Параллельные стеки Ctrl+Shift+D, S
Контроль параллельных данных
Контрольные значения
Видимые Ctrl+Alt+V, A
Локальные Ctrl+Alt+V, L
Интерпретация Ctrl+Alt+I
Динамическое визуальное дерево
Динамический обозреватель свойств
Динамический просмотр для XAML
Ошибки привязки XAML
Стек вызовов Ctrl+Alt+C
Потоки Ctrl+Alt+H
Модули Ctrl+Alt+U
Процессы Ctrl+Alt+Z
Анализ диагностики Ctrl+Shift+Alt+D
Память
Дизассемблированный код Ctrl+Alt+D
Регистры Ctrl+Alt+G




```
EAX = 00000004 EBX = 00000005 ECX = 00000000 EDX = 00000002 ESI =
  ESP = 00B5FC48 EBP = 00B5FD14 EFL = 00000212
```



```
//(x + 3)2  * 4
int x;
cin >> x;
__asm {
    mov eax, x
    add eax, 3
    mov ebx, 2
    mul  ebx
    mov ebx, 4
    mul  ebx
    mov x, eax
}
cout << x;
```



Задание

Написать программу вычисляющую выражение

1. $(x + 3)^2 / 4$
2. $(x^2 * 8) — (x * 4)$, x вводит пользователь