**Assembler. Установка интерпретатора и запуск первой программы через DOSBox**

В данной статье разбирается способ установки интерпретатора и запуск файла EXE через DOSBox. Есть много учебников по Ассемблер. В изучении очень помогает сайт av-assembler.ru. А теперь перейдём к основной теме.

Для начала давайте установим наш старенький интерпретатор.  
[**Ссылка**](https://vk.com/topic-205404099_47749107)

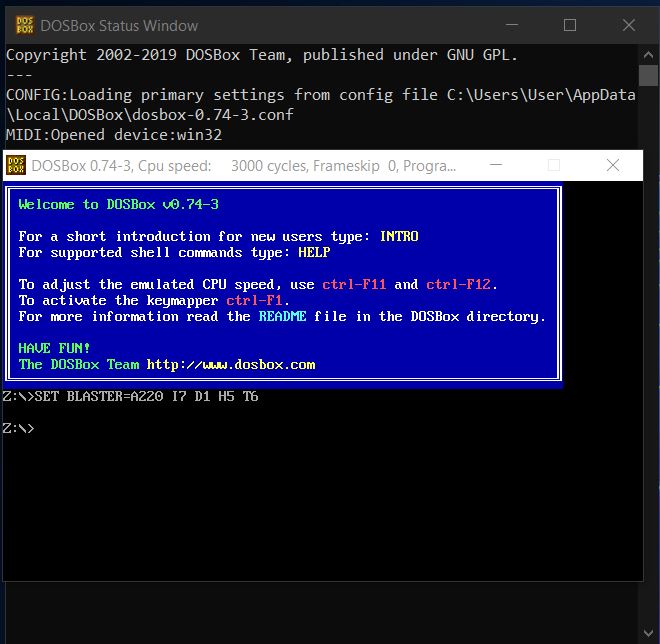
После распаковки файлов, советую сохранить их в папке Asm на диск C, чтобы иметь меньше расхождений с представленным тут материалом. Если вы разместите директорию в другое место, изменится лишь путь до файлов, когда вы будете использовать команду mount.

Для запуска интерпретатора нам так же потребуется эмулятор DOSBox. Он и оживит все наши компоненты. Скачаем и установим его!  
[**Ссылка**](https://www.dosbox.com/download.php?main=1)

В папке Asm я специально оставил файл code.asm. Именно на нём мы и потренируемся запускать нашу программу. Советую сохранить его копию, ибо там хранится весь код, который в 99% случаев будет присутствовать в каждом вашем проекте.

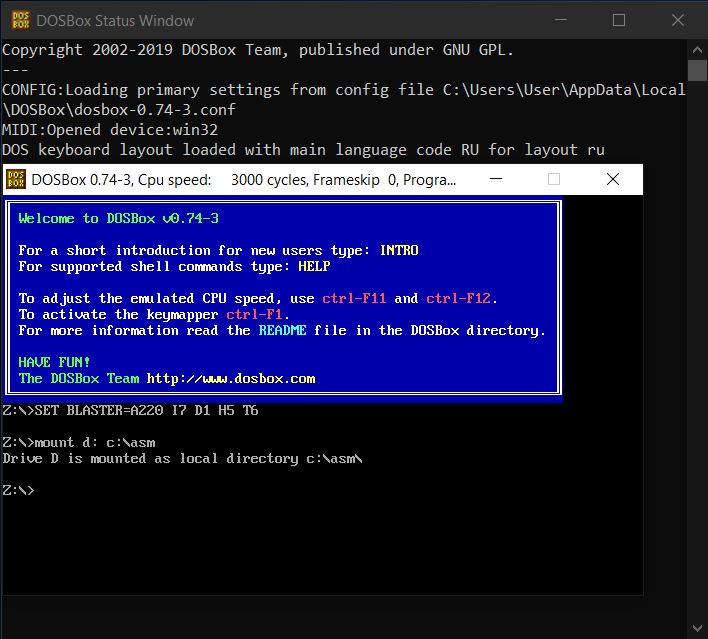
code.asm

Итак. Запускаем наш DOSBox и видим следующее:



Для простоты сопоставим имя пути, по которому лежит наша папка Asm. Чтобы это сделать, пропишем следующую команду:

mount d: c:\asm

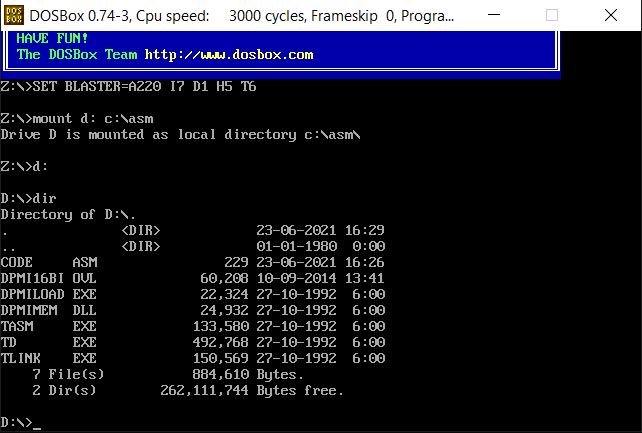


Здесь вместо d: мы можем использовать любую другую букву. Например, назвать i или s. А C это наш реальный диск. Мы прописываем путь до наших файлов ассемблера.

Теперь, откроем смонтированный диск:

d:

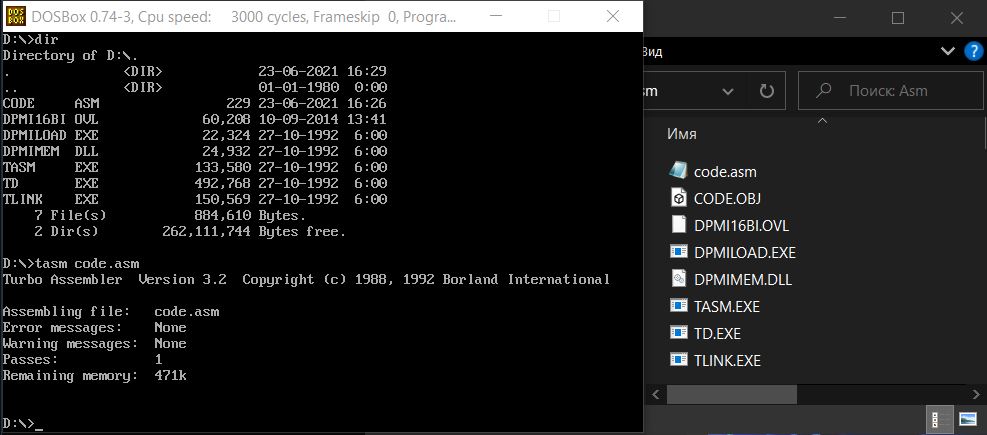
Прописав команду dir, мы сможем увидеть все файлы, которые там хранятся. Здесь можно заметить и наш файл CODE с расширением ASM, а также дату его создания.



И только теперь мы начинаем запускать наш файл! Бедные программисты 20 века, как они только терпели всё это? Пропишем следующую команду:

tasm code.asm

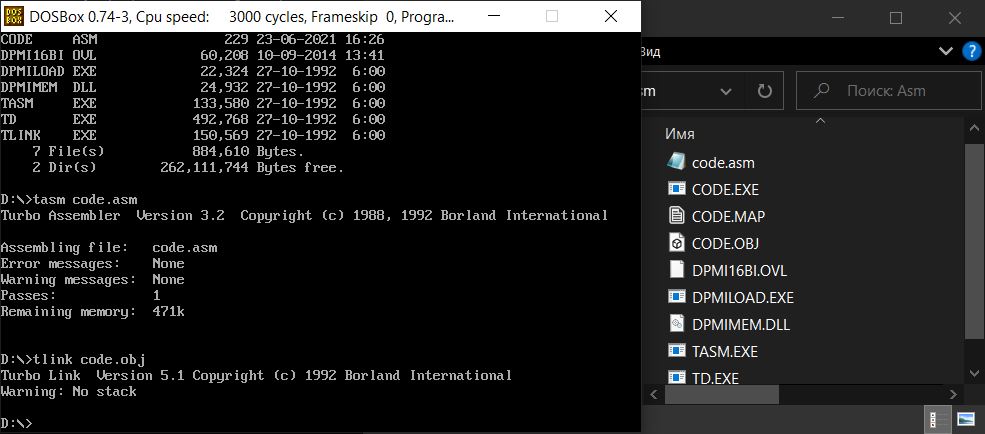
После мы увидим следующее сообщение, а наша директория пополнится новым файлом с расширением OBJ.



Теперь пропишем ещё одну команду:

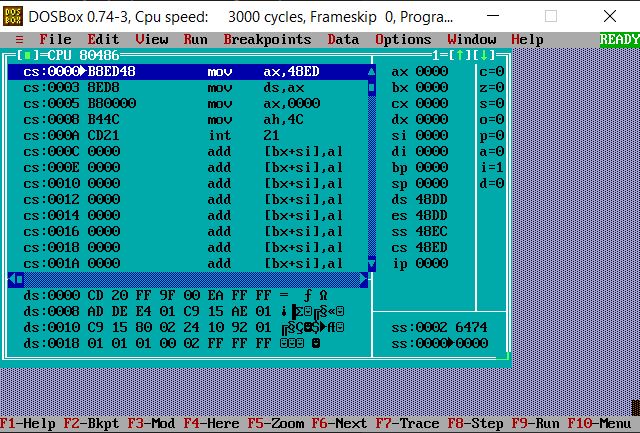
tlink code.obj

В нашей папке появилась ещё пара файлов – CODE.MAP и  CODE.EXE. Последний как раз и есть исполняемый файл нашего кода assembler.



Если он появился, значит, мы можем запустить режим отладки нашей программы, введя последнюю команду. Обратите внимание, теперь мы не указываем расширение файла, который запускаем.

td code

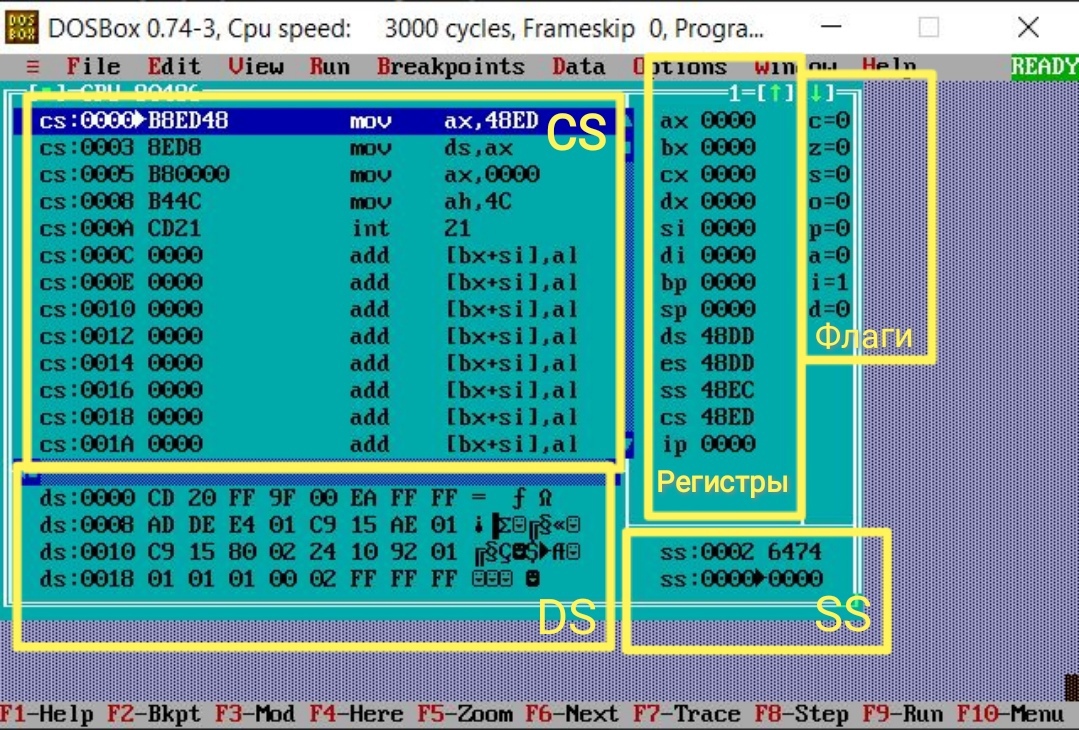


Этот старинный интерфейс насквозь пропитан духом ушедшей эпохи старых операционных систем. Тем не менее…

Нажав F7 или fn + F7 вы сможете совершить 1 шаг по коду. Синяя строка начнёт движение вниз, изменяя значения регистров и флагов. Пока это всего лишь шаблон, на котором мы потренировались запускать нашу программу в режиме дебага. Реальное “волшебство” мы увидим лишь с полноценным кодом на asm.

Небольшой пример для запуска

Давайте ознакомимся с имеющимися разделами.



**CS**

Code segment – место, где turbo debug отражает все найденные строки кода. Важное замечание – все данные отражаются в TD в виде 16-ричной системы. А значит какая-нибудь ‘12’ это на самом деле 18, а реальное 12 это ‘C’. CS аналогичен разделу “Begin end.” на Pascal или функции main.

**DS**

Data segment, отражает данные, которые TD обнаружил в d\_s. Справа мы видим их символьную (char) интерпретацию. В будущем мы сможем увидеть здесь наш “Hello, world”, интерпретируемый компилятором в числа, по таблице ASCII. Хорошей аналогией DS является раздел VAR, как в Pascal. Для простоты можно сказать, что это одно и тоже.

**SS**

Stack segment – место хранения данных нашего стека.

**Регистры**

Все эти ax, bx, cx, si, di, ss, cs и т. д. – это наши регистры, которые используются как переменные для хранения данных. Да, это очень грубое упрощение. Переменные из Pascal и регистры Assembler это не одно и тоже, но надеюсь, такая аналогия даёт более чёткую картину. Здесь мы сможем хранить данные о циклах, арифметических операциях, системных прерываниях и т. д.

**Флаги**

Все эти c, z, s, o, p и т.д. это и есть наши флаги. В них хранится промежуточная информация о том, например, было ли полученное число чётным, произошло ранее переполнение или нет. Они могут хранить результат побитого сдвига. По опыту, могу сказать, на них обращаешь внимание лишь при отладке программы, а не во время штатного исполнения.

Ещё одно замечание. Если вы измените данные исходного файла с расширением .ASM, то вам придётся совершить все ранее описанные операции вновь, ибо обновив, например, code.asm вы не меняете code.obj или code.exe.

**Маленькая шпаргалка для заметок:**

1. mount d: c:\asm – создаём виртуальный диск, где корень –папка asm
2. d: - открываем созданный диск
3. tasm code.asm – компилируем исходный код
4. tlink code.obj – создаём исполняемый файл
5. td code – запускаем debug
6. F7 – делаем шаг в программе