**Тема**: "Знакомство с программой-отладчиком Turbo Debugger"

**Цель работы**: Целью работы является изучение среды Turbo Debugger, выполнение простейших команд микропроцессора в среде Turbo Debugger.

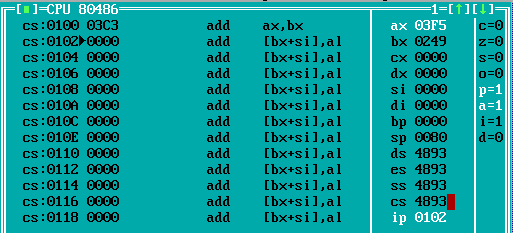
**Вариант №13**

A=1AC(428₁₀), B=249(585₁₀)

# Ход работы.

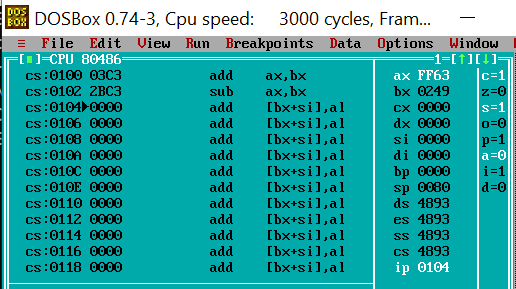
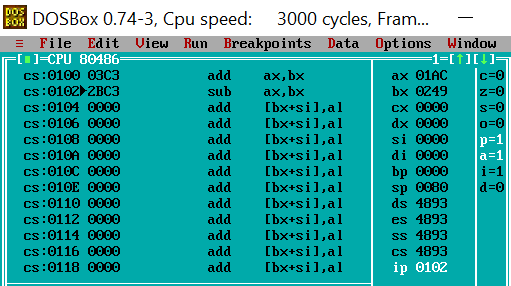
## Задание 1.

## C:\Users\mvideo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\изображение.png



Чтобы выполнить операцию А+В я заношу величины в соответствующие регистры ax и bx, затем прописываю адрес 100h в первую строку команд с помощью комбинации CTRL + G, затем записываю команду «add ax, bx». После пишу тот же адрес в строку ip в окне регистров. В конце нажимаю клавишу F7, и в регистр ax записывается результат сложения. Флаг аf становится равным 1, т.к. произошел заем из единицы из одного полубайта в другой. Флаг pf равен 1, из-за того, что двоичное представление результата содержит четное количество единиц и нулей.

## Задание 2.



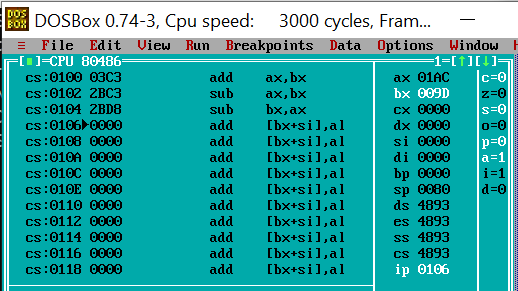
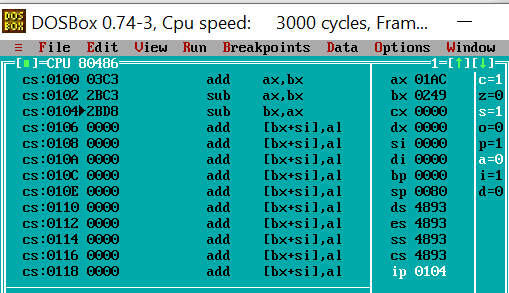
Для выполнения операции А-В выполняю теже действия, но теперь вписываю команду «sub ax, bx». Теперь дополнительного переноса не происходит, из-за чего флаг аf = 0. Флаг pf равен 1, из-за того, что двоичное представление результата содержит четное количество единиц и нулей. Флаг sf=1, потому что число получило отрицательное значение. Флаг cf=1, потому что при сложении беззнаковых чисел произошло переполнение.

1AC – 249 = FF63

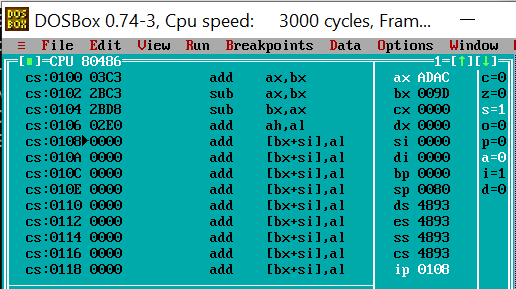
428 – 585 = -157

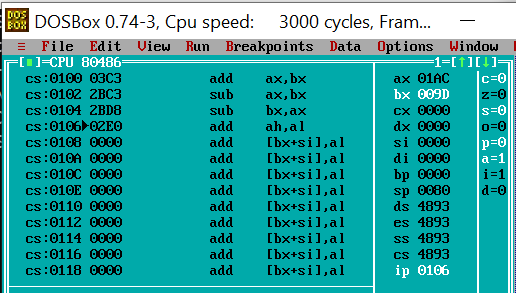
Видно, что результат явно не соответствует действительности. Это происходит из-за того, что компьютер вычитает большее число из меньшего по-иному. Число -249 представляется в дополнительном коде, затем к этому результату добавляется 1AC и получается FF63, что совпадает с результатом на скриншоте.

Для выполнения операции В-А я меняю команду на «sub bx,ax», из-за чего результат вычитания переходит из регистра ax в bx. И ответ получается 9D



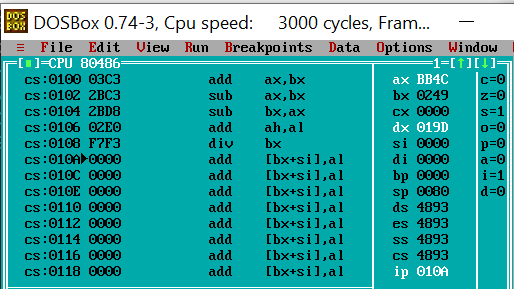
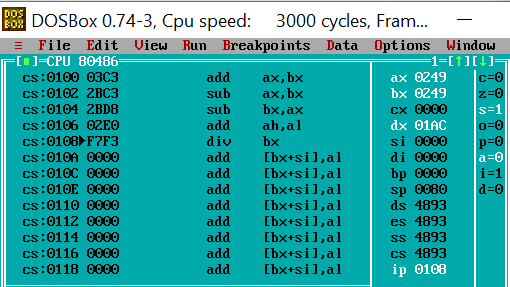
## Задание 3.

  
Повторяю шаги из задания и ввожу команду «add ah, al», которая складывает старшие байты числа А(01) и младшие (AC) и записывает результат в ah, вследствие чего число в регистре ax меняется с 1AC на ADAC.



sf = 1 (отрицательное значение).

## Задание 4.

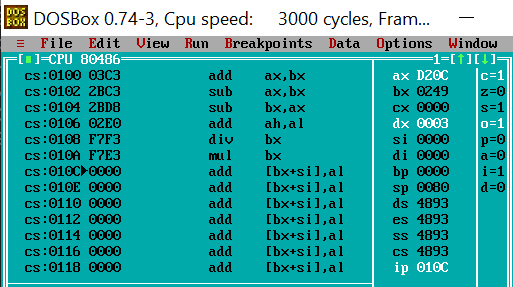
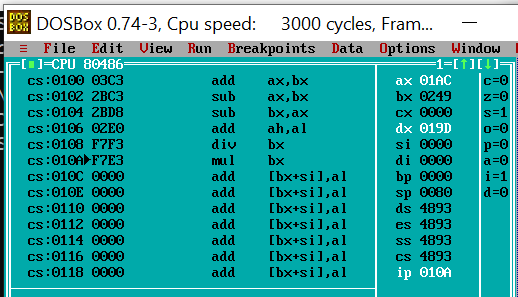


Пишу команду «div bx». После выполнения деление регистр AX будет содержать частное BB4C, а регистр DX — остаток 19D

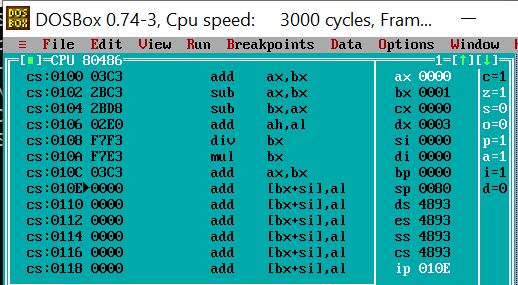
## 

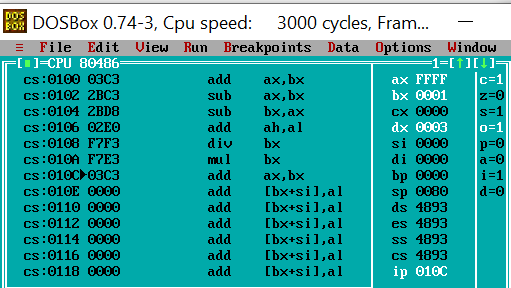
## Задание 5.

Пишу команду «mul bx». т.к. результат получился больше, а именно получилось 32 битовое число(11 1101 0010 0000 1100), то у нас активируется флаг переполнения (of = 1) Результат данной операции записывается в регистрах DX (старшие 2 байта) равен 3 и AX (младшие 2 байта) равен D20C.



## Задание 6.

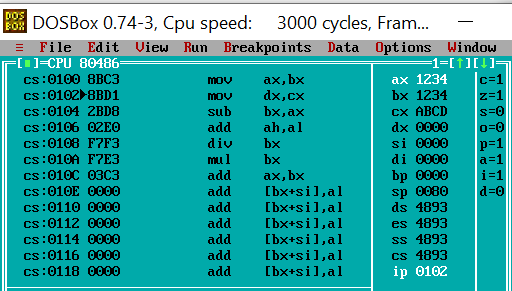
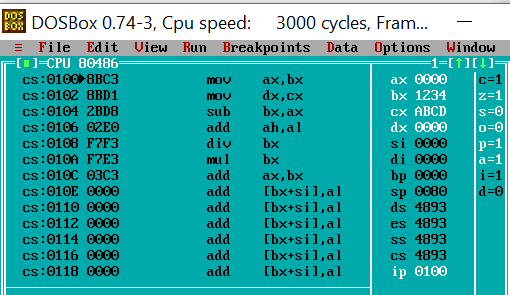




При выполнении сложения 0FFFFh и 0001h видимый результат в регистре ax составляет 0000. По флагам мы видим, что активны сразу 4: переполнения, нуля, чётности и переноса. c=1 потому что в (n+1) значащем бите появляется единица; z=1 получение нулевого результата; p=1 – чётное число единиц; a=1 – произошёл перенос из младшего полубайта в старший.

## Задание 7.

## C:\Users\mvideo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\изображение (17).pngC:\Users\mvideo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\изображение (18).png



Сначала с помощью команд «mov ax,bx» и «mov dx, cx» записываю числа в нужные регистры, затем с помощью команды «mov ah,dl» записывается в старший байт регистра ах младший байт регистра dx.

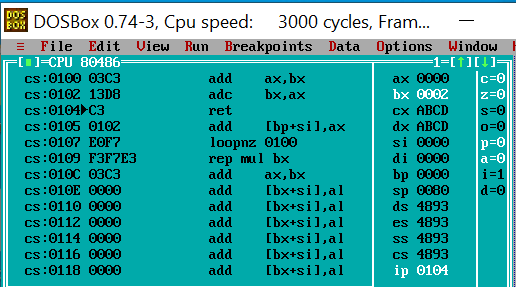
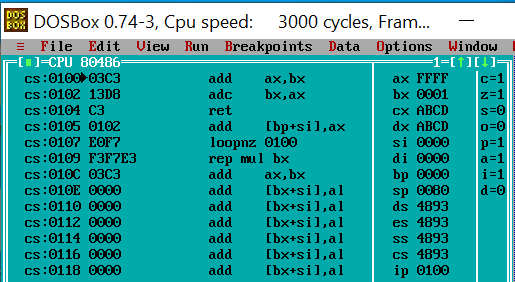
## Задание 8.

## 

При последовательном выполнении команд FFFF + 1 и FF00 + 1, флаг переноса (cf) сначала становится равным 1, а затем возвращается к 0. Это происходит из-за того, что при первой команде переполнение происходит, а при второй – нет.

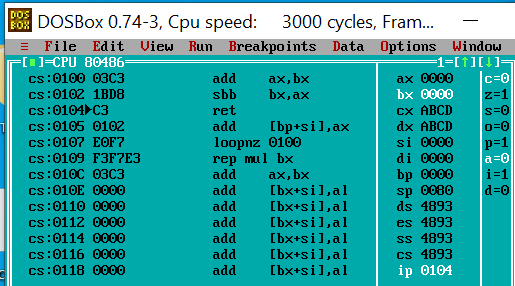
## Задание 9.

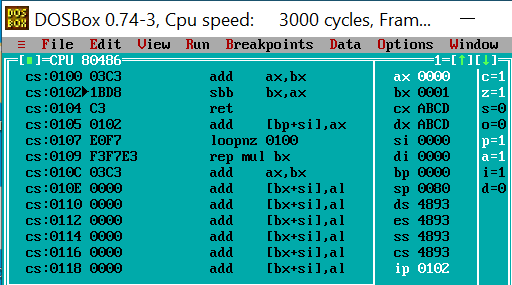
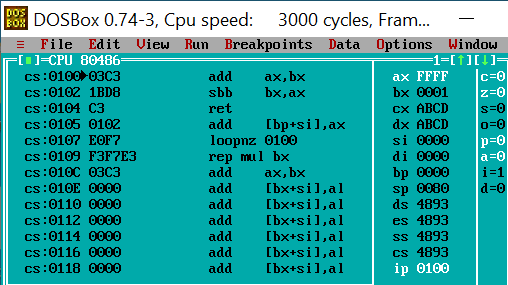
## C:\Users\mvideo\AppData\Local\Microsoft\Windows\INetCache\Content.Word\изображение (22).png



После первой операции в регистре ax виден результат 0000, но на самом деле результат равен 10000. Флаг переноса становится равным 1. После второй операции случается сложение 0000, 0001 и значение флага переноса, т. е. 1, из-за чего результат равен 2.

## Задание 10.





После 1 операции в ax остаётся 0000, флаг переноса равен 1, а в bx – 1. После 2 операции (sbb bx, ax) получается последовательное вычитание 0001, 0000 и значение флага переноса, т.е. 1, из-за чего результат равен 0.

**Вывод:** В ходе выполнения первой лабораторной работы я познакомился с рабочим окном Turbo Debugger, простейшими командами, некоторыми флагами и регистрами.

**Контрольные вопросы**

1)  Отладка (debugging) — один из важнейших этапов разработки

программного обеспечения (английский термин bug означает "ошибка в

программе"). В процессе отладки путем детального анализа в компьютерных

программах выявляются и устраняются возможные логические ошибки,

которые не обнаруживаются на стадии компиляции.

2) Отладчики (debugger) — это вспомогательные программы (утилиты),

включаемые в набор инструментальных средств программиста для

выполнения отладки других программ. Отладчики предоставляют

программисту возможность выполнять программу по шагам, следить за

изменениями данных и проверять выполнение условий. В зависимости от

уровня языка, которым оперирует отладчик, можно выделить два их типа.

Отладчики исходного кода дают программисту возможность видеть

текст программы на языке высокого уровня (например, Си), проверять

значения отдельных переменных и агрегатов данных (таких, как массивы),

используя их имена.

Отладчики машинного уровня отслеживают реально выполняемые

машинные команды, отображаемые в виде команд ассемблера. Они позволяют также просматривать содержимое ячеек памяти и регистров микропроцессора.

3) Регистры — это небольшие (несколько байт) именованные области

памяти микропроцессора, используемые для временного хранения двоичных

данных, к которым необходимо обеспечить быстрый доступ.

Машинное слово есть фрагмент данных фиксированного размера, обрабатываемый как единое целое с помощью набора команд или аппаратного обеспечения процессора. Количество бит в машинном слове — размер слова (он же ширина или длина слова) — является важной характеристикой любой конкретной архитектуры процессора или компьютерной архитектуры.

4) add результат в адресе первого операнда.

5) После выполнения команды ADD результат вычисления помещается в регистр AX, где ранее был один из операндов.

6) sub результат – регистр левого операнда.

7) дополнительный код (флаг знака - s)

8) В микропроцессоров Intel используются двухбайтовые машинные слова. Каждый регистр общего назначения (AX, BX, CX и DX) может хранить одно машинное слово. Однако имеется возможность оперировать с отдельными байтами этих регистров. В этом случае каждый регистр рассматривается состоящим из старшего (High) и младшего (Low) байтов. Обозначения отдельных байтов из регистров состоят из двух букв. Первая задает имя регистра (A, B, C или D), а вторая указывает, какой это байт регистра. Для обозначения старшего байта используется буква H, а младшего — L. Таким образом, регистр AX можно рассматривать, состоящим из двух однобайтовых регистров AH и AL.

9) mul, помещает результат в регистры  dx – старшие биты и в ax – младшие биты.

10) div, ax – частное dx – остаток. div bx – ax:bx / bx.

11) При выполнении сложения беззнаковых чисел суть переполнения (в двоичном представлении) состоит в том, что в результате сложения двух единиц в старшем разряде возникает единица, выходящая за разрядную сетку результирующего регистра. Эта единица в регистр помещена быть не может, и при записи в регистр "отсекается".

12) Флаг - это бит, принимающий значение 1 ("флаг установлен"), если выполнено некоторое условие, и значение 0 ("флаг сброшен") в противном

случае.

13) adc – к результату прибавляется флаг переноса.

14) sbb – от результата отнимается флаг переноса.

15) Если при сложении беззнаковых чисел происходит переполнение (возникает единица переноса за пределы разрядной сетки регистра), то единичка переноса записывается в Carry Flag. флаг нуля обозначает явл. ли результат нулевым или нет.

16) Флаг переполнения устанавливается в той ситуации, когда получен ошибочный результат. Данный флаг знака позволяет узнать знак числа. Если результате выполнения команды получено отрицательное значение, то флаг знака принимает значение SF=1 (Sign Flag).