|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

**ФАКУЛЬТЕТ \_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**КАФЕДРА \_\_\_\_КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**НАПРАВЛЕНИЕ ПОДГОТОВКИ \_\_09.03.01 Информатика и Вычислительная техника \_\_\_\_\_**

**Отчет**

**по лабораторной работе № \_1\_**

**Дисциплина: \_**Машинно-зависимые языки и основы компиляции**\_\_\_\_**

**Название лабораторной работы: \_**Изучение среды и отладчика \_\_\_\_

ассемблера**\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент гр. **\_**ИУ6-42б**\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_**И.С. Марчук**\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Москва, 2021

**Задание:**

1) Запустите RADAsm, создайте файл проекта по шаблону консольного приложения. Внимательно изучите структуру программы и зафиксируйте текст с комментариями в отчете. 19

2) Запустите шаблон на выполнение и просмотрите все полученные сообщения. Убедитесь, что текст программы и настройки среды не содержат ошибок.

3) Добавьте директивы определения данных и команды сложения и вычитания, описанные в разделе 3 настоящих методических указаний. Найдите в отладчике внутреннее представление исходных данных, зафиксируйте его в отчете и поясните. Проследите в отладчике выполнение набранной вами программы и зафиксируйте в отчете результаты выполнения каждой добавленной команды (изменение регистров, флагов и полей данных).

4) Введите следующие строки в раздел описания инициированных данных и определите с помощью отладчика внутренние представление этих данных в памяти. Результаты проанализируйте и занесите в отчет. val1 BYTE 255 chart WORD 256 lue3 SWORD -128 alu BYTE ? v5 BYTE 10h BYTE 100101B beta BYTE 23,23h,0ch sdk BYTE ″Hello″,0 min SWORD -32767 ar DWORD 12345678h valar BYTE 5 DUP (1, 2, 8)

5) Определите в памяти следующие данные:

а) целое число 25 размером 2 байта со знаком;

б) двойное слово, содержащее число -35; 20

в) символьную строку, содержащую ваше имя (русскими буквами и латинскими буквами).

Зафиксируйте в отчете внутреннее представление этих данных и дайте пояснение.

6) Определите несколькими способами в программе числа, которые во внутреннем представлении (в отладчике) будут выглядеть как 25 00 и 00 25. Проверьте правильность ваших предположений, введя соответствующие строки в программу. Зафиксируйте результаты в отчете.

7) Замените директивы описания знаковых данных на беззнаковые: A DWORD -30 B DWORD 21 X DWORD ? Запустите программу и прокомментируйте результат.

8) Добавьте в программу переменную F1=65535 размером слово и переменную F2= 65535 размером двойное слово. Вставьте в программу команды сложения этих чисел с 1: add F1,1 add F2,1 Проанализируйте и прокомментируйте в отчете полученный результат (обратите внимание на флаги).

**Цель работы:**

Изучение процессов создания, запуска и отладки программ на ассемблере в среде программирования RADAsm с использованием 32-разрядного отладчика OlleDBG.

**1) Текст программы с комментариями:**

**.586** ; подключение набора команд Реntium

**.MODEL flat, stdcall** ; модель памяти и

; конвенция о передаче параметров

**OPTION CASEMAP:NONE** ; опция различия строчных

; и прописных букв

**Include kernel32.inc** ; подключение описаний процедур и

**Include masm32.inc** ; констант

**IncludeLib kernel32.lib** ; подключение библиотек

**IncludeLib masm32.lib**

**.CONST** ; начало раздела констант

**MsgExit DB "Press Enter to Exit",0AH,0DH,0**

**.DATA** ;раздел инициализированных переменных

**.DATA?** ;раздел неинициализированных переменных

**inbuf DB 100 DUP (?)**

**.CODE** ; начало сегмента кода

**Start:**

**add F1,1**

**add F2,1**

**mov EAX,A**

**add EAX,5**

**sub EAX,B**

**mov X,EAX**

**XOR EAX,EAX**

**Invoke StdOut,ADDR MsgExit** ; вывод сообщения

**Invoke StdIn,ADDR inbuf,LengthOf inbuf** ; ввод строки

**Invoke ExitProcess,0** ; завершение программы

**End Start** ; конец модуля

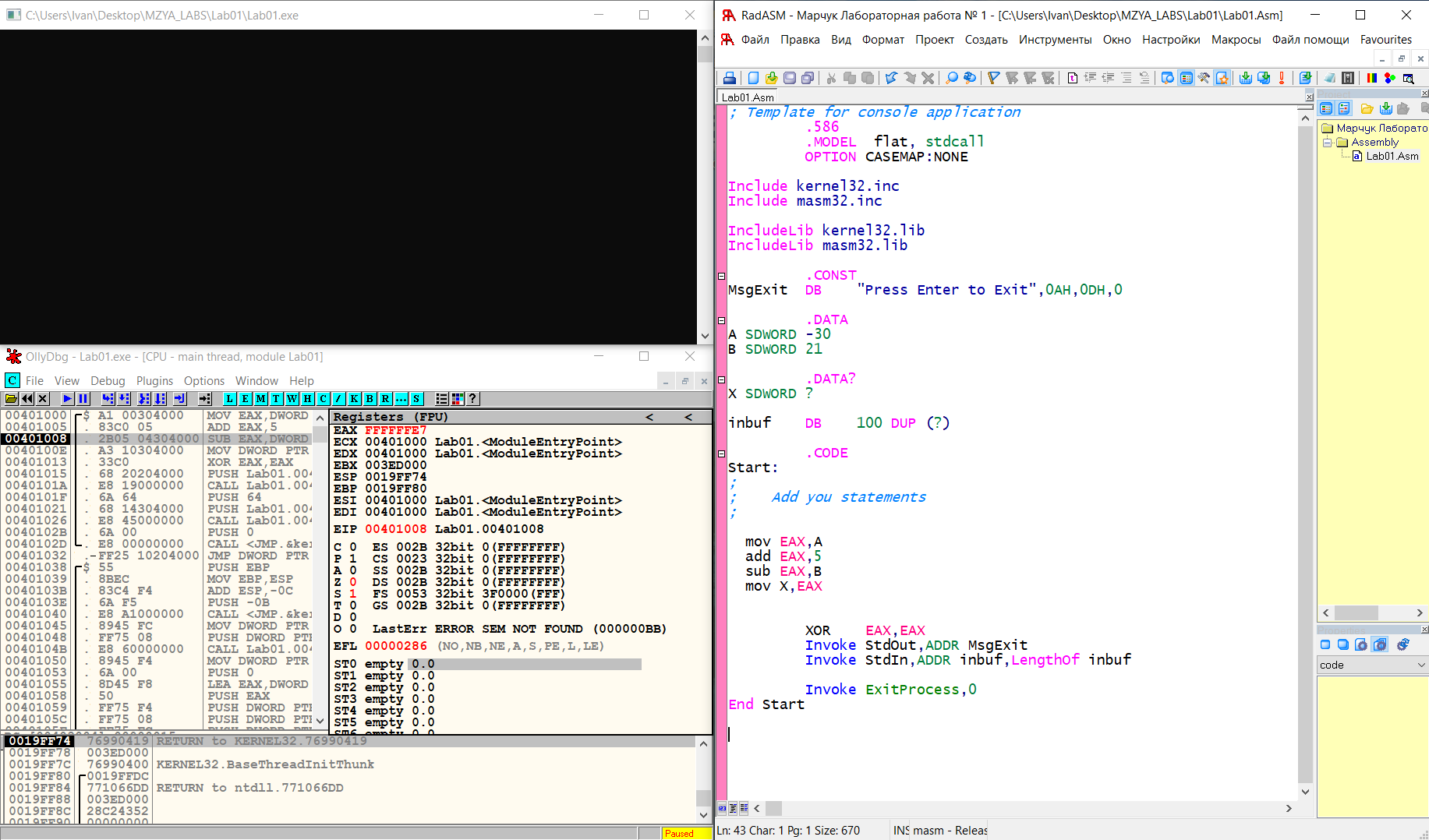


Рисунок 1 – Среда отладки с написанным кодом программы и запущенным отладчиком

**2) Запуск программы на выполнение:**

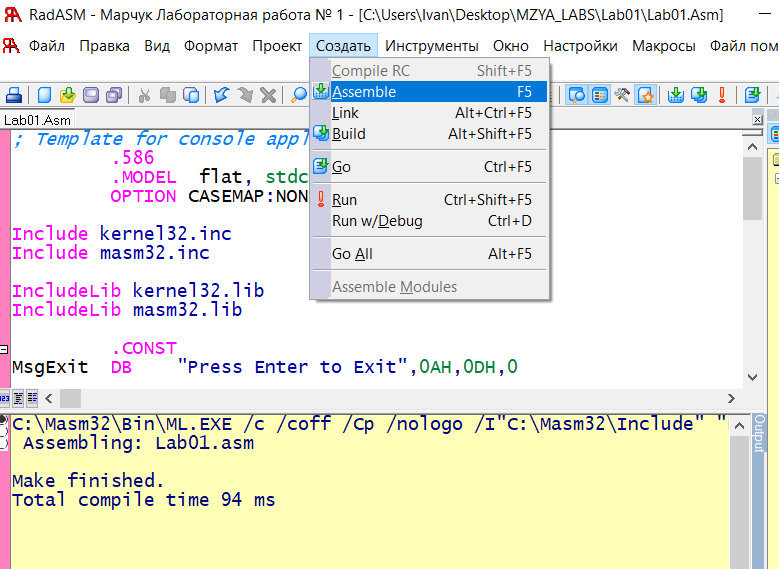


Рисунок 2 – Ассемблирование программы

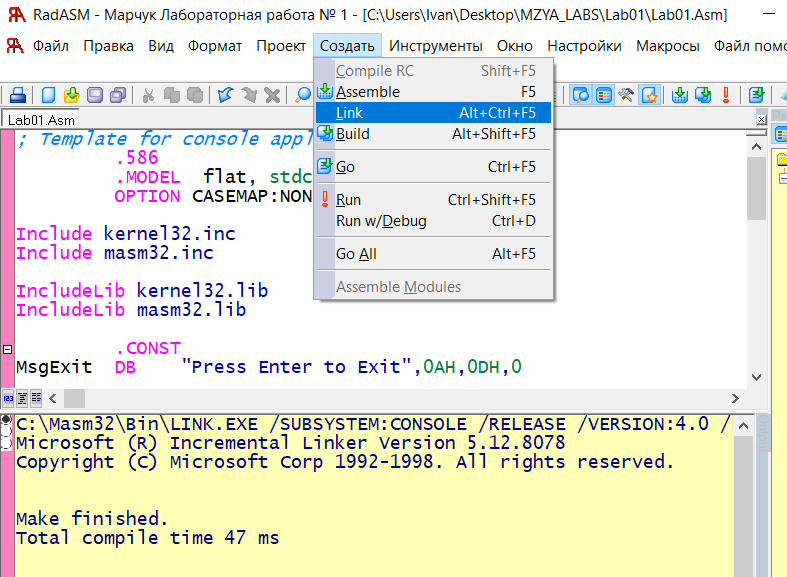


Рисунок 3 – Связывание кода с библиотеками

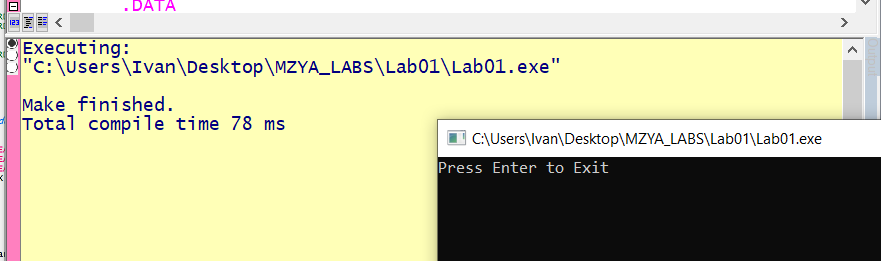


Рисунок 4 – Запущенная на выполнение программа

**3) Добавление директив определения данных и команды сложения и вычитания, а также их представление и выполнение:**

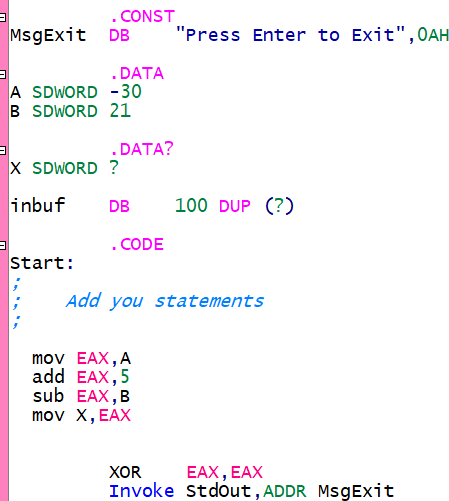


Рисунок 5 – Объявленные в теле программы переменные

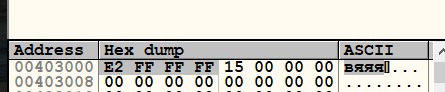


Рисунок 6 – Переменные в памяти

Переменная “A”, объявленная в коде (рисунок 5), находящаяся в памяти представлена на рисунке 6, и выделена серым цветом. Рядом с ней находится переменная “B” (21 в шестнадцатеричном виде равен 15);

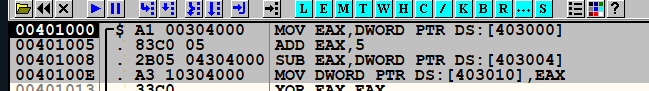


Рисунок 7 – Дизассемблирование команд

Вот список команд, представленных на рисунке 7, с замененными именами:

mov EAX,A

add EAX,5

sub EAX,B

mov X,EAX

Результаты выполнения каждой из команд по отдельности

**4) Добавление дополнительных переменных:**

Вид переменных в коде представлен на рисунке 8:

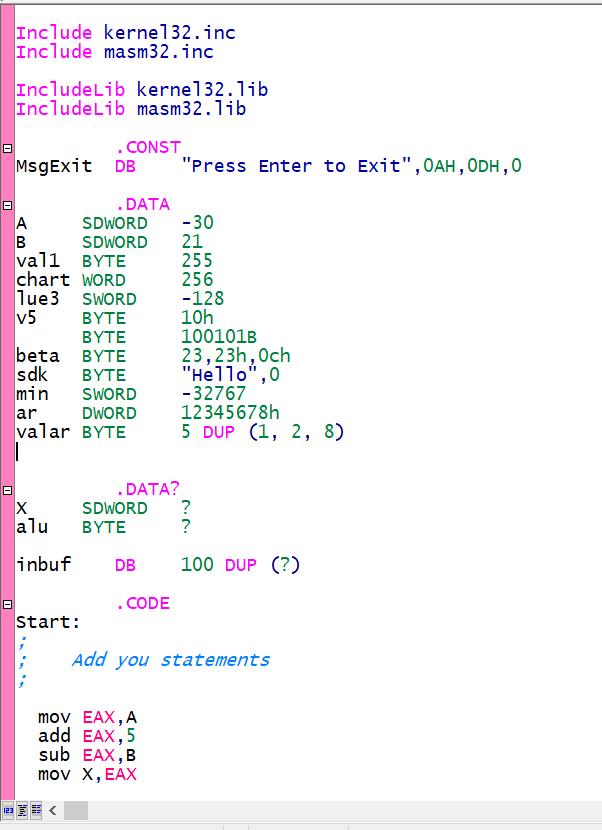


Рисунок 8 – Вид переменных в коде

Переменные в выделенной памяти (рисунок 9):

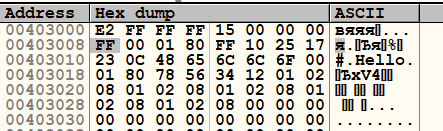


Рисунок 9 – Переменные в памяти

На рисунке 9, выделена переменная val1, находящаяся в сегменте данных DS:[403008] и занимающая 1 байт.

А также по адресу DS:[403010] находятся два байта переменной chart.

**5) Определите в памяти следующие данные:**

**а) целое число 25 размером 2 байта со знаком;**

**б) двойное слово, содержащее число -35; 20**

**в) символьную строку, содержащую ваше имя (русскими буквами и латинскими буквами).**

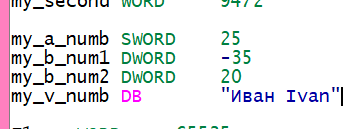
****

Рисунок 10 **–** Переменные в коде программы

На рисунке 10 видно текст объявления переменных. Где:

– пункту a) соответствует переменная my\_a\_numb

– пункту б) соответствуют переменные my\_b\_num1; my\_b\_num2

– пункту в) соответствует переменная my\_v\_numb

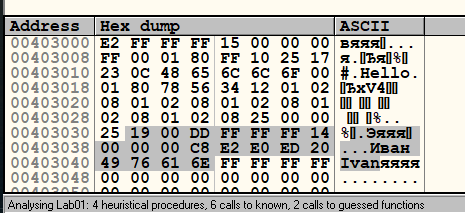


Рисунок 11 **–** Переменные в коде программы

На рисунке 11 мы видим эти переменные в памяти, правда в перевернутом виде. Переведем числа из 16-ого вида в 10-й

– 00 19 = > 25

– FF FF FF DD => -35

– 00 00 00 14 => 20

Дальше идет строка, записанная уже в прямом виде, как видно из рисунка 13, это связано с тем, что в строке каждый символ строки записывается отдельно и кодирует собой только один элемент строки в отличие от чисел кодирующихся в 16-ной форме.

**6) Определите несколькими способами в программе числа, которые во внутреннем представлении (в отладчике) будут выглядеть как 25 00 и 00 25:**

Так как байты в памяти записываются наоборот, возьмем шестнадцатиричные числа 0025 и 2500.

Переведем в десятичную: 37 и 9472. Оба числа будем записывать в двухбайтовые переменные, потому что в задании приведена запись состоящая из 4 цифр. Такой тип переменных можно реализовать тремя способами.

* my\_first WORD 37 ; 00 25

my\_second WORD 9472 ; 25 00

* my\_firs1 SWORD 37 ; 00 25

my\_secon1 SWORD 9472 ; 25 00

* my\_firs2 DW 37 ; 00 25

my\_secon2 DW 9472 ; 25 00

Здесь, WORD и DW - беззнаковые двухбайтовые, SWORD - двухбайтовые со знаком.

Переменную my\_first находим под адресом DS:[40302D] (25 00),

за ней находим переменную my\_second под адресом DS:[40302F] (00 25).

Переменную my\_firs1 находим под адресом DS:[403031] (25 00),

за ней находим переменную my\_secon1 под адресом DS:[403033] (00 25).

Переменную my\_firs2 находим под адресом DS:[403035] (25 00),

за ней находим переменную my\_secon2 под адресом DS:[403037] (00 25).

Это показано на рисунке 12:

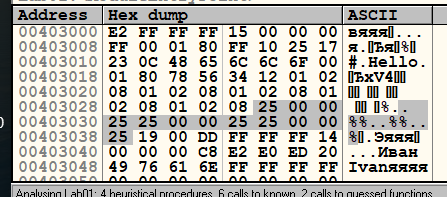


Рисунок 12 – Переменные в памяти

**7) Замените директивы описания знаковых данных на беззнаковые**После замены сегмент данных стал выглядеть так (рисунок 13):

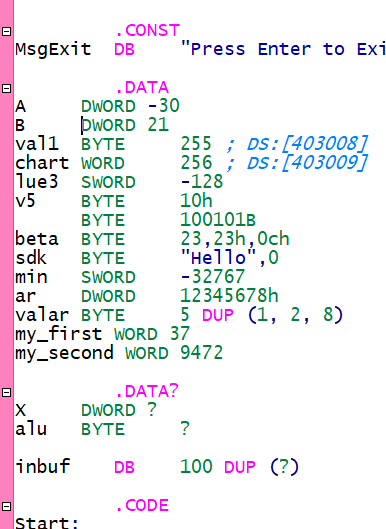


Рисунок 13 – Переменные в коде

В представлении данных в памяти ничего не поменялось (рисунок 14):

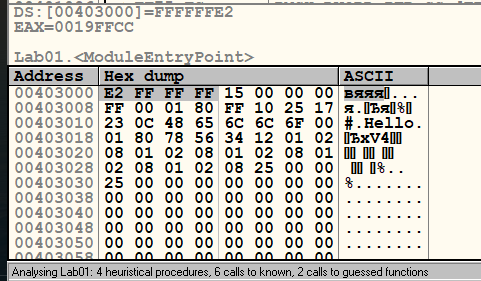


Рисунок 14 – Скан памяти

Они все также занимают 4 байта и поскольку числа отрицательные, в старших разрядах стоят единицы.

Сами вычисления и результат вычислений – одинаковы: 0xFFFFFFD2 = -46, что можно увидеть в регистре EAX на рисунке 15:

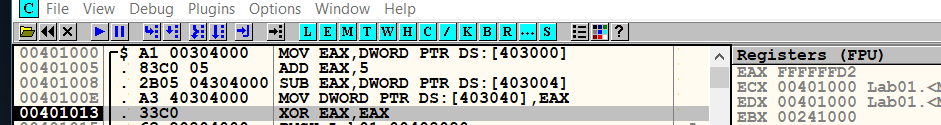


Рисунок 15 – Результат вычислений в регистре EAX

**8) Добавьте в программу переменную F1=65535 размером слово и переменную F2= 65535 размером двойное слово. Вставьте в программу команды сложения этих чисел с 1:**

- F1 WORD 65535

- F2 DWORD 65535

Вот их представление в памяти (рисунок 16):

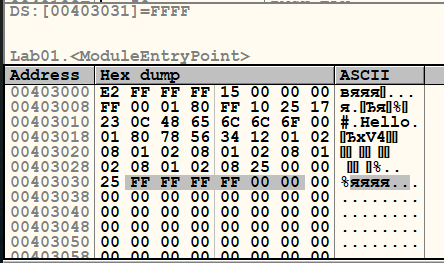


Рисунок 16 – Два числа, лежащие в памяти (первые два байта – F1, следующие 4 байта -- F2)

После прибавления к первому числу оно перешло в нули, так как старшая единица не поместилась в разряды (FF FF+1 -> 1 00 00), что видно из рисунка 17

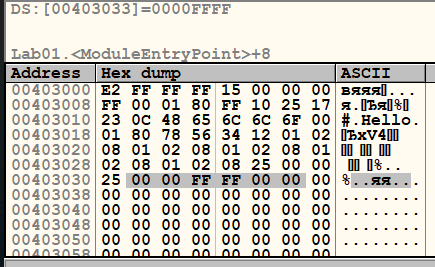


Рисунок 17 – Результат прибавления единицы к F1

А вот второе число полностью поместилось, что видно из рисунка 18

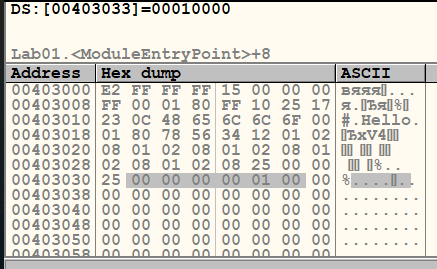


Рисунок 18 – Результат прибавления единицы к F2

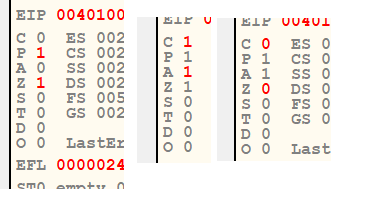


Рисунок 19 – Флаги до первого сложения, после первого сложения и после второго сложения

На рисунке 19 особенно интересен случай посередине, то есть после сложения F1 и 1. Здесь была помещена единица в регистр переполнения, из-за переполнения описанного выше. Также при сложении имеем единицу во флаге переноса тетрады и в случае переполнения единицу во флаге признака нуля. Что и требовалось.

**Контрольные вопросы:**

1) Дайте определение ассемблеру. К какой группе языков он относится?

Язык программирования низкого уровня. Команды ассемблера есть интерпретация машинных команд в удобочитаемый формат.

2) Как создать заготовку программы на ассемблере? Из каких частей она состоит?

При создании нового проекта необходимо выбрать заготовку в меню, которая состоит из:

- Раздела подключения библиотек и описаний процедур,

- Раздела констант,

- Раздела инициализированных переменных,

- Раздела неинициализированных переменных,

- Раздела кода.

3) Как запустить программу на ассемблере на выполнение? Что происходит с программой на каждом этапе обработки?

В RADASM32 для того, чтобы получить исполняемый файл из исходного кода и запустить его, необходимо выполнить в среде следующее:

- Assemble (транслировать или, точнее, ассемблировать),

- Link (компоновать),

- Run (выполнить)/Run w-Debug (выполнить в подключенном отладчике).

4) Назовите основные режимы работы отладчика. Как осуществить пошаговое выполнение программы и просмотреть результаты выполнения машинных команд.

Для выполнения команды отладчика используют следующие комбинации клавиш:

– F7 - выполнить шаг с заходом в тело процедуры;

– F8 - выполнить шаг, не заходя в тело процедуры.

5) В каком виде отладчик показывает положительные и отрицательные целые числа? Как будут представлены в памяти числа: **A Word 5,-5** ? Смотри рисунок 20.

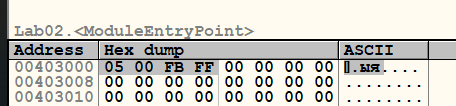


Рисунок 20 – Числа в памяти

Как те же числа будут выглядеть после загрузки в регистр AX?

Точно также, не востребованные разряды не изменяются.

6) Что такое «разрядная сетка»? Как ограничения разрядной сетки влияют на представление чисел в памяти компьютера?

Количество разрядов, используемых в ЭВМ для представления чисел.

7) Каким образом в ассемблере программируются выражения? Составьте фрагмент программы для вычисления С=A+B, где A, В и С – целые числа формата BYTE

Код переменных на языке ассемблера:

– mov EAX, A

– add EAX, B

– mov C, EAX

**Вывод:**

Я научился писать и отлаживать программы в среде RADAsm. И создал с помощью полученных знаний и в соответствии с методическими указаниями программу. Так же я создал переменные и научился выполнять простейшие математические операции. Я научился анализировать память, и понял структуру переменных, созданных мною и хранящихся в ней.