|  |  |
| --- | --- |
| **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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

**по лабораторной работе № \_\_1\_\_**

**Вариант №2**

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

**Название лабораторной работы:** Изучение среды и отладчика ассемблера



Студент гр. ИУ6-44Б **\_\_\_\_\_\_\_\_\_28.02.2023 Аль Сабунчи Т.О.\_\_**

(Подпись, дата) (И.О. Фамилия)

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

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

**Цель работы**

Изучение процессов создания, запуска и отладки программ на ассемблере Nasm под управлением операционной системы Linux, а также особенностей описания и внутреннего представления данных.

**Задание**

1) Для хранения всех программ лабораторных работ на компьютере в операционной системе Linux создать специальные каталог и подкаталог;

2) Вызвать текстовый редактор и ввести заготовку 32-х или 64-х разрядной программы на ассемблере. Внимательно изучить структуру программы и способ программирования системных вызовов для выполнения операций ввода-вывода и завершения программы. Зафиксировать текст программы с комментариями в отчете;

3) Выполнить трансляцию программы с листингом, а также компоновку. После чего запустить программу на выполнение;

4) Запустить отладчик edb. Средствами графического интерфейса отладчика открыть в нем исполняемую программу, после этого выполнить программу по шагам, контролируя содержимое регистров и оперативной памяти;

5) Добавить в заготовку несколько команд для вычисления результата следующего выражения: X=A+5-B. Сохранить программу с тем же именем, затем выполнить ее трансляцию, компоновку и загрузку в отладчик. Зафиксировать изменение программы в отчете;

6) Проследить в отладчике выполнение программы и зафиксировать в отчете результаты выполнения каждой добавленной команды (изменение регистров, флагов и полей данных);

7) Ввести следующие строки в разделы описания инициированных и неинициализированных данных и определите с помощью отладчика внутренние представление этих данных в памяти.

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);

8) Определить в памяти следующие данные:

а) целое число 25 размером 2 байта со знаком;

б) двойное слово, содержащее число -35;

в) символьную строку, содержащую ваше имя.

Зафиксировать в отчете описание и внутреннее представление этих данных и дайте пояснение;

9) Определить несколькими способами в программе числа, которые во внутреннем представлении (в отладчике) будут выглядеть как 25 00 и 00 25. Проверить правильность, введя соответствующие строки в программу. Зафиксировать результаты в отчете;

10) Добавить в программу переменную F1=65535 размером слово и переменную F2= 65535 размером двойное слово. Вставить в программу команды сложения этих чисел с 1:

add [F1],1 add [F2],1;

**Ход работы**

*Часть 1.* Запуск приложения

Текст 32х разрядной программы:

section .data ; сегмент инициализированных переменных

ExitMsg db "Press Enter to Exit",10 ; выводимое сообщение

lenExit equ $-ExitMsg

section .bss ; сегмент неинициализированных переменных

InBuf resb 10 ; буфер для вводимой строки

lenIn equ $-InBuf

section .text ; сегмент кода

global \_start

\_start:

; write

mov eax, 4 ; системная функция 4 (write)

mov ebx, 1 ; дескриптор файла stdout=1

mov ecx, ExitMsg ; адрес выводимой строки

mov edx, lenExit ; длина выводимой строки

int 80h ; вызов системной функции

; read

mov eax, 3 ; системная функция 3 (read)

mov ebx, 0 ; дескриптор файла stdin=0

mov ecx, InBuf ; адрес буфера ввода

mov edx, lenIn ; размер буфера

int 80h ; вызов системной функции

; exit

mov eax, 1 ; системная функция 1 (exit)

xor ebx, ebx ; код возврата 0

int 80h ; вызов системной функции

Затем была проведена трансляция и листинг. После чего программа была запущена (рис.1).

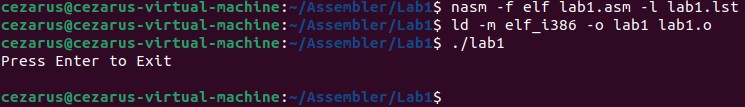


Рисунок 1 – Трансляция, листинг и запуск приложения

По средствам терминала было запущено приложение EDB, и исполняемая программа открыта непосредственно в нем (рис.2).

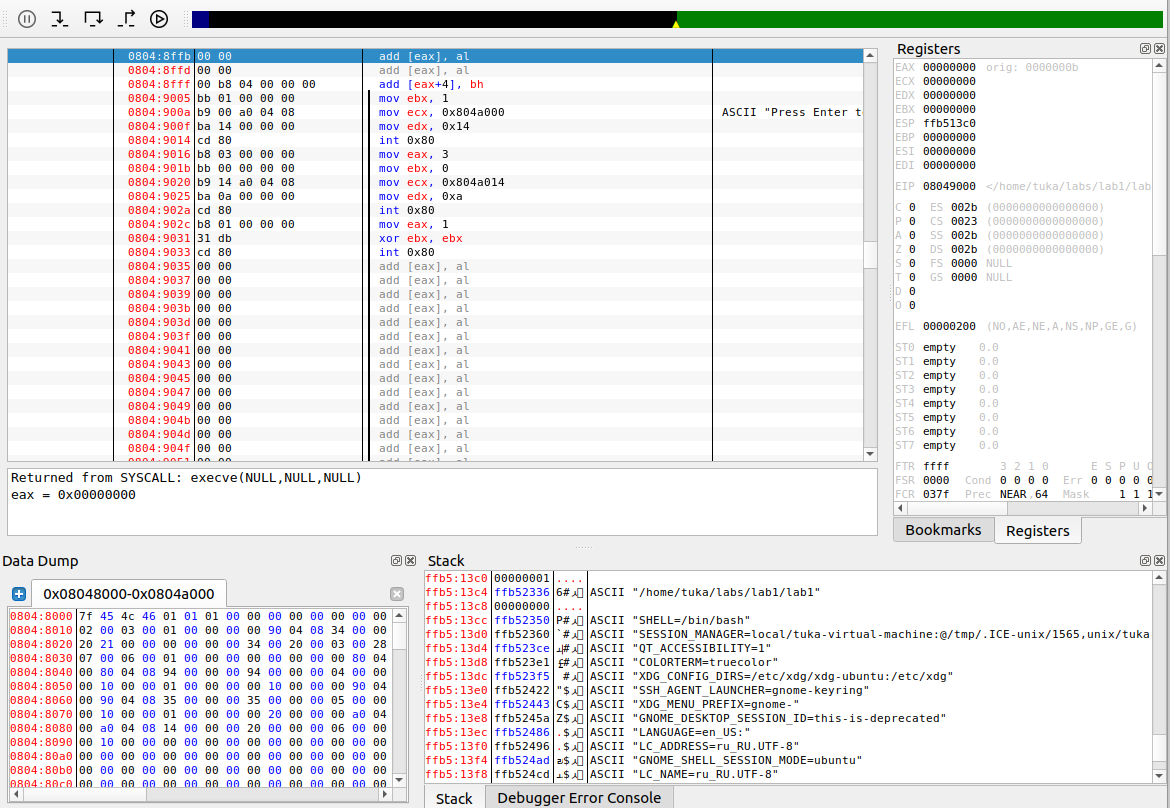


Рисунок 2 – EDB

Левая часть приложения показывает шаги выполнения исполняемой программы, раздел Register отслеживает состояние регистров, Data Dump отвечает за отображения состояние памяти компьютера.

*Часть 2.* Написания первой программы

Текст 32-х разрядной программы (рис.3).

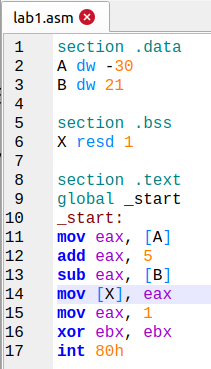


Рисунок 3 - Текст программы (1)

Выполняем трансляцию и компоновку программы. Запускаем ее в EDB и пошагово выполняем (рис.4-8). Фиксируем каждый шаг.

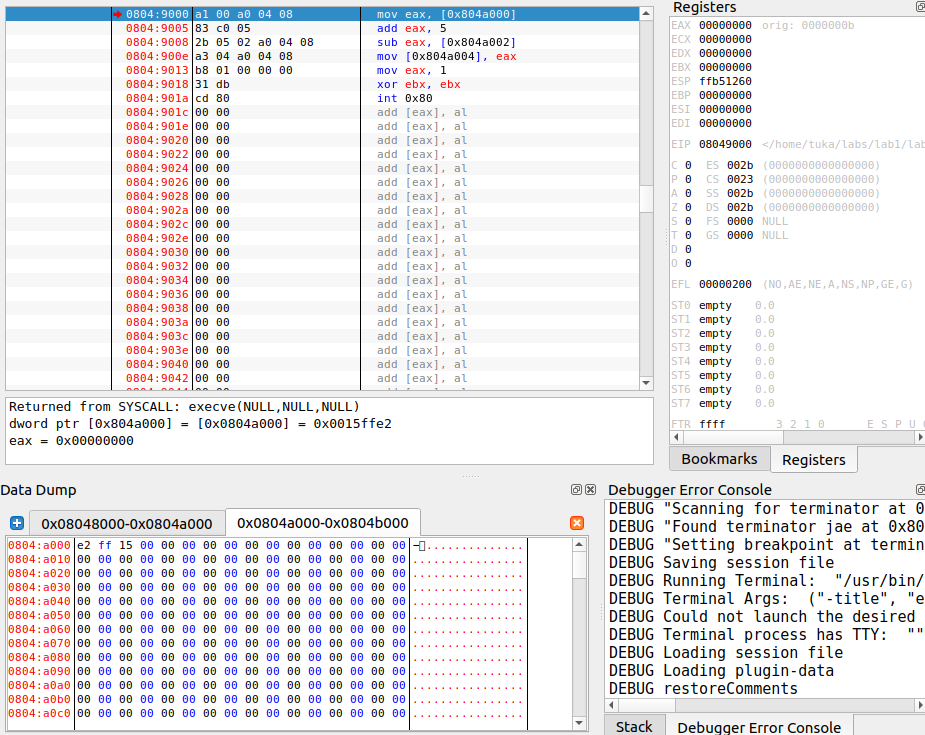


Рисунок 4 - Начало выполнения программы

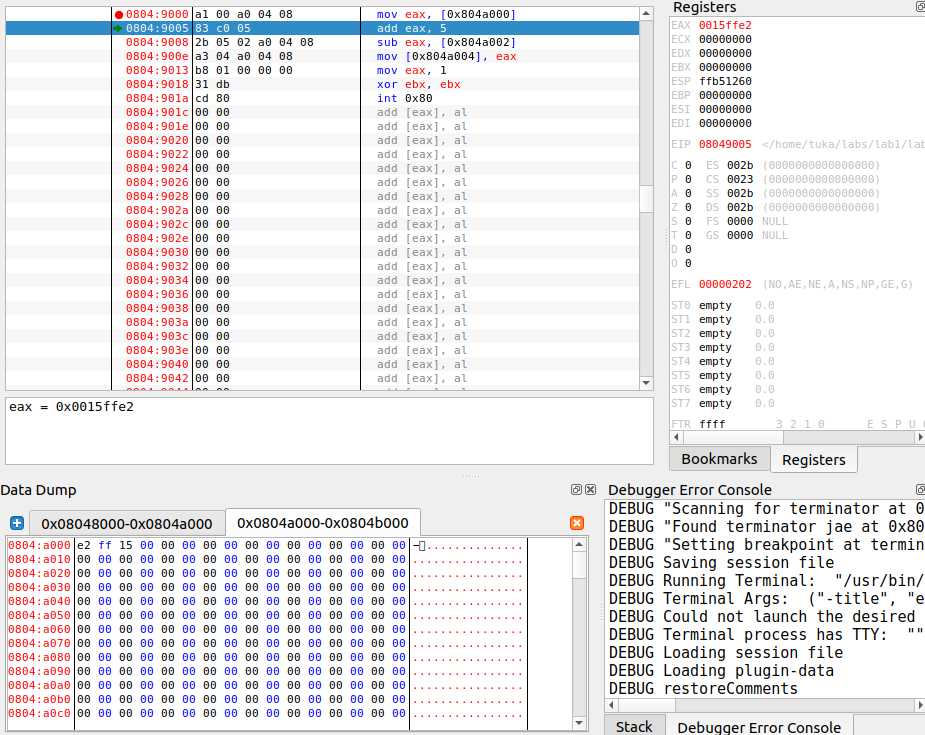


Рисунок 5 – Команда MOV

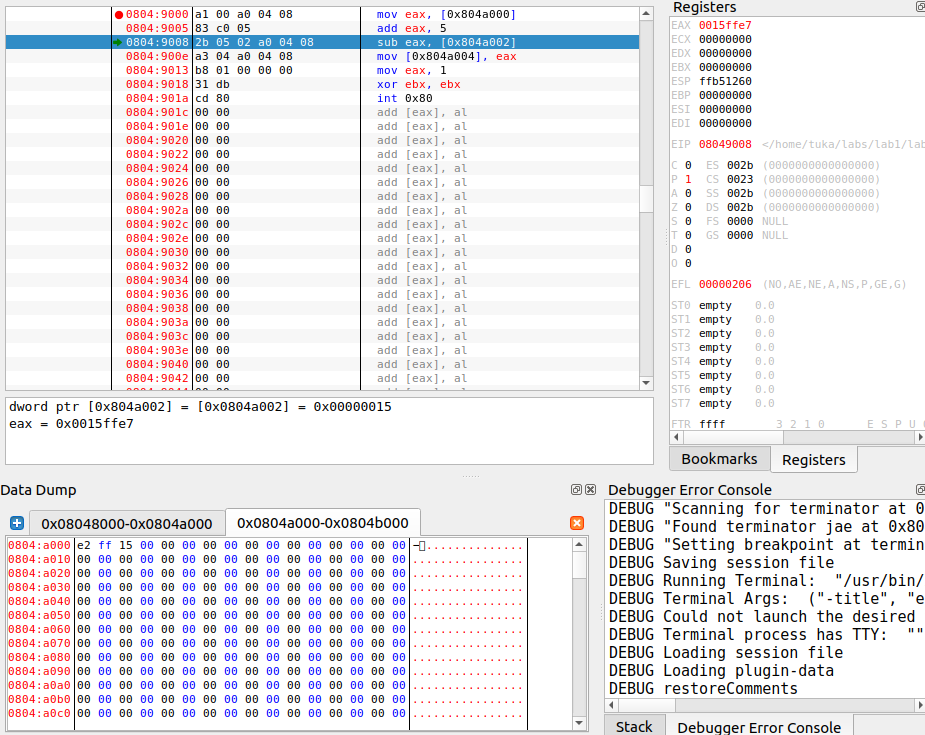


Рисунок 6 - Команда ADD

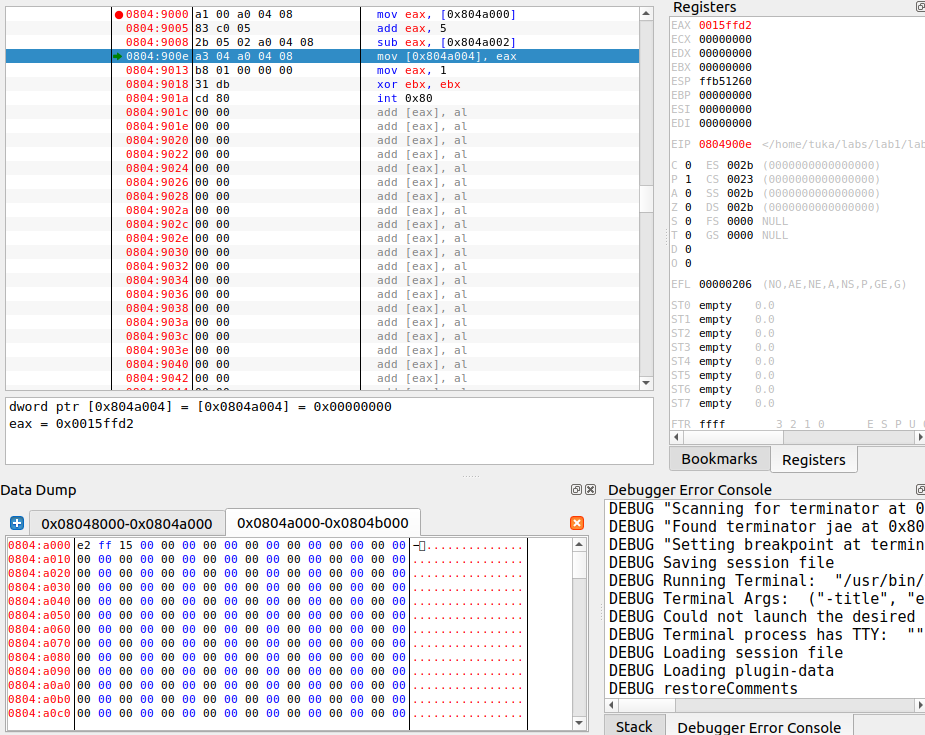


Рисунок 7 - Команда SUB

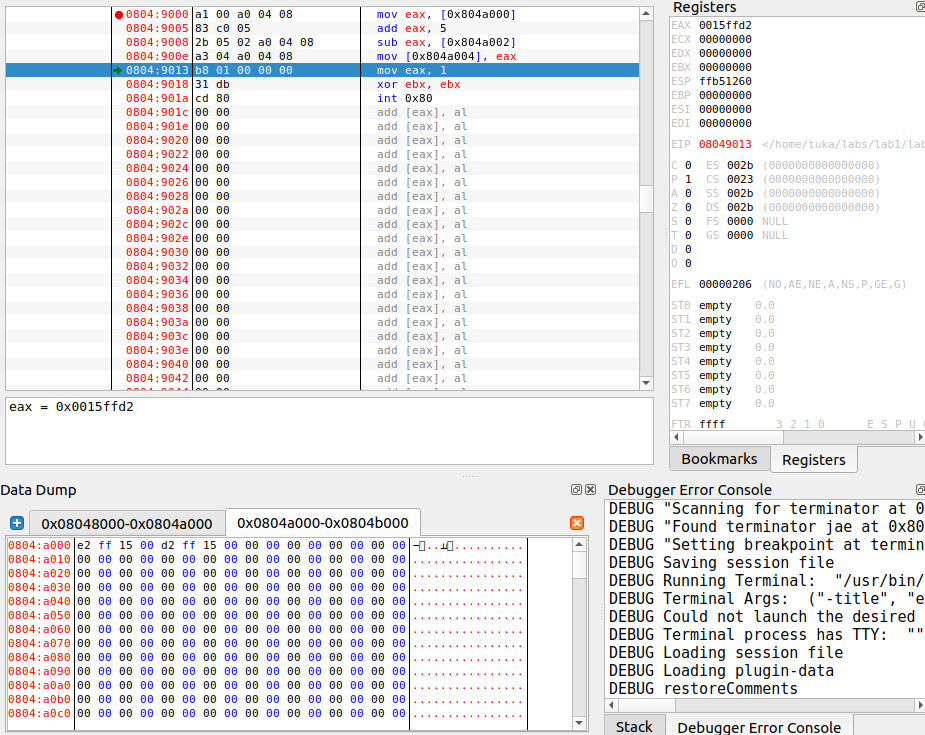


Рисунок 8 - Команда MOV

Можно сделать вывод, что программа верно отработала все шаги.

*Часть 3.* Внутреннее представление данных

Прописываем строчки текста в раздел инициализированных и неинициализированных данный (в «data» отдел и «bss» отдел).

section .data ; сегмент инициализированных переменных

val1 db 255

chart dw 256

lue3 dw -128

v5 db 10h

db 100101B

beta db 23,23h,0ch

sdk db "Hello",10

min dw -32767

ar dd 12345678h

valar times 5 db 8

section .bss ; сегмент неинициализированных переменных

alu resw 10

f1 resb 5

section .text

global \_start

\_start:

; exit

mov eax, 1; системная функция 1 (exit)

xor ebx, ebx; код возврата 0

int 80h; вызов системной функции

По средствам использования приложения EDB, можно определить внутреннее представление данных, в разделе “Data Dump” (рис.9).

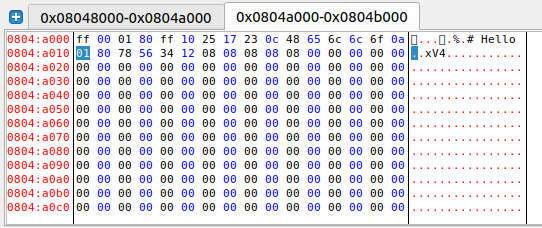


Рисунок 9 - Внутреннее представление данных

Данные:

1) val1 db 255 – первый байт;

2) chart dw 256 – следующие 2 байта;

3) lue3 dw -128 – следующие 2 байта;

4) v5 db 10h – следующий байт;

5) beta db 23, 23h, 0Ch – следующие 3 байта;

6) sdk db “Hello”, 10 – следующие 6 байтов;

7) min dw -32767 – следующие 2 байта;

8) ar dd 12345678h – следующие 4 байта;

9) valar times 5 db 8 – следующие 5 байтов;

10) alu resw 10 – следующие 20 байтов;

11) f1 resb 5 – следующие 5 байтов.

Изменяем код программы для определения целого числа, двойного слова со знаком, символьной строки. Текст получившейся программы:

A dw +25

B dd -35

C db "Tuka",0

D db "Тука",0

val1 db 255

chart dw 256

lue3 dw -128

v5 db 10h

db 100101B

beta db 23,23h,0ch

sdk db "Hello",10

min dw -32767

ar dd 12345678h

valar times 5 db 8

section .bss ; сегмент неинициализированных переменных

alu resw 10

f1 resb 5

section .text

global \_start

\_start:

; exit

mov eax, 1; системная функция 1 (exit)

xor ebx, ebx; код возврата 0

int 80h; вызов системной функции

В EDB можно увидеть изменение распределения памяти в разделе «Data Dump» (рис.10).



Рисунок 10 – Data Dump

Получаем следующее распределение памяти:

1) A dw 25 – целое число, занимающее первые 2 байта;

2) B dd -35 – число со знаком, следующие 4 байта;

3) C db “Tuka ”, 0 – символьная строка (латинские буквы), следующие 5 байт;

4) D db “Тука ”, 0 – символьная строка (русские буквы). Следующие 9 байт, учитывая маркер окончания строки.

*Часть 4.* Определение чисел

Определение чисел в программе может происходить несколькими способами. Сначала создадим переменную для числа «00 25» и посмотрим на Data Dump в EDB (рис.11). Затем сделаем точно также, но уже для числа «25 00» (рис.12).

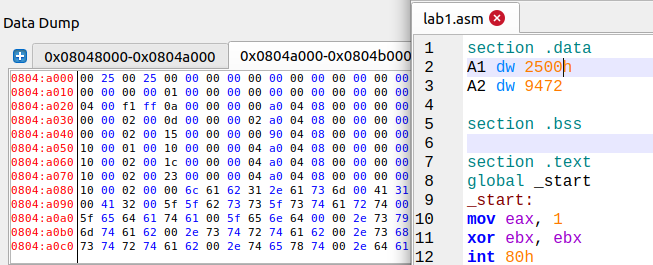


Рисунок 11 - Число "00 25"

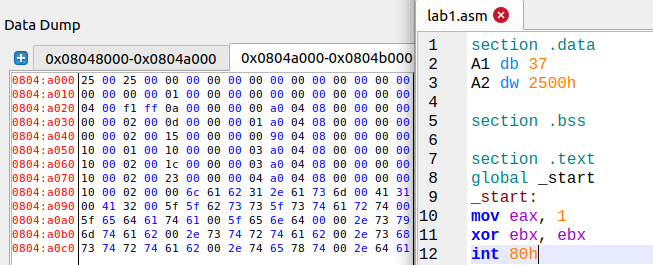


Рисунок 12 - Число "25 00"

На рисунке 11 продемонстрированы два способа задачи числа «00 25». В первом случае A1 = 2500h для смены байтов местами в памяти компьютера. Второй способ A2 работает на переводе числа в системах счисления.

Для рисунка 12 и числа «25 00» аналогичная схема.

*Часть 5.* Переполнение при симметрии

Для проверки переполнения приписываем следующий код (рис. 13).

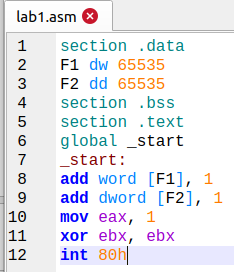


Рисунок 13 - Текст программы (2)

При пошаговом выполнении программы можно заметить изменение флагов регистра (рис.14-16).

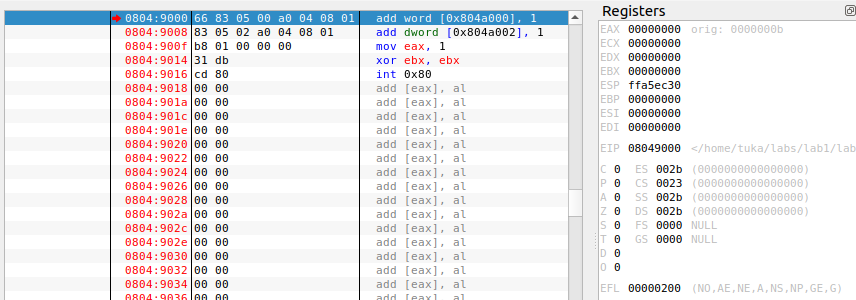


Рисунок 14 - Начальная позиция флагов

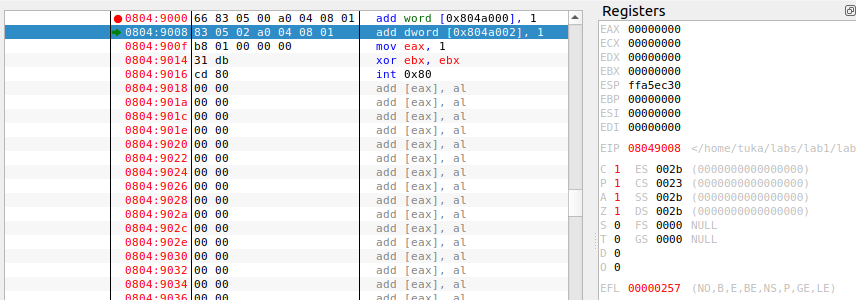


Рисунок 15 - Вторая позиция флагов

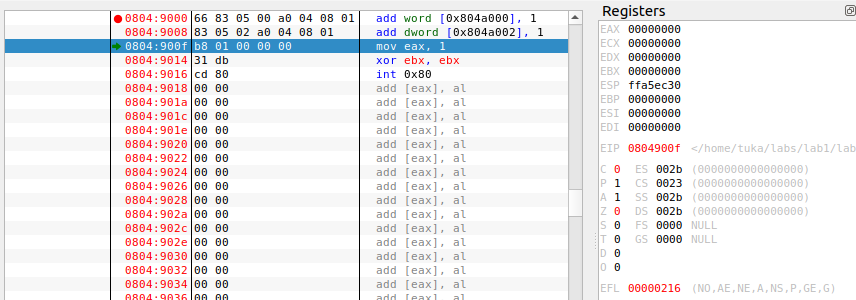


Рисунок 16 - Заключительная позиция флагов

Число «65535» – это максимальная граница для типа word, и при прибавлении единицы «F1» обнуляется, а в «CF» устанавливается флаг (флаг переноса) и в «AF» (флаг вспомогательного переноса).

При прохождении «F2» такого не случается, следовательно устанавливается исключительно флаг «AF».

**Вывод**

Во время выполнения данной лабораторной работы, были пройдены первоначальные шаги по изучению низкоуровневого языка программирования Aseembler. Были написаны и запущены несколько программ, которые показывали многие аспекты данного языка. Также была проведена работа с приложением EDB для проверки работы программы и для отслеживания внутреннего представления памяти. Прошло ознакомление с графическим интерфейсом программ EDB и SASM, а также проведены первичные математические операции на языке Assembler.

**Контрольные вопросы**

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

Assembler – низкоуровневый язык программирования, управляемый по средством написания команд, которые подбираются под процессор машины. Он относится к группе машинно-зависимых языков.

1. *Из каких частей состоит заготовка программы на ассемблере?*

Заготовка программы на Assembler стоится из частей: section .data, section .bss, section.text.

1. *Как запустить программу на ассемблере на выполнение? Что происходит с программой на каждом этапе обработки?*

Запуск программы идет по средствам ее трансляции, компоновки и выполнения. Во время трансляции программа переходит в двоичное представление, компоновка приписывает объектные коды.

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

Отладка выполняется по средствам кнопок в верхней левой части экрана приложения EDB. Результаты можно увидеть в терминале.

1. *В каком виде отладчик показывает положительные и отрицательные целые числа? Как будут представлены в памяти числа: A dw 5,-5? Как те же числа будут выглядеть после загрузки в регистр AX?*

Данные показаны в виде 16-ых цифр. A dw 5 будет выглядеть, как «05 00», A dw -5 будет «fb ff». При загрузки этих же чисел в регистр AX: A dw 5 станет «00 05», а -5 примет вид «ff fb».

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

mov AX, [A]

mov BX, [B]

add AX, BX

mov [C], AX