РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

дисциплина: Архитектура компьютера

Студент: Бешкуров Михаил НК-101

Ли Тимофей НФИ-101

МОСКВА

2018г.

Цель работы

Знакомство с методами отладки при помощи EDB и его основными возможностями. Приобретение навыков написания программ с использованием подпрограмм.

Ход работы

1. Написали программу со следующим алгоритмом:
– ввести символ с клавиатуры;
– преобразовать полученный код в десятичную символьную запись;
– вывести символ и его код.
Перевод числа в десятичную символьную запись оформили в виде подпрограммы.

Код программы

```
SECTION .data

    hello: DB 'Input string',10
    hellolen: EQU $-hello

SECTION .bss
    buf2: RESB 80

SECTION .text
    GLOBAL _start
convert:
    xor ecx,ecx
    xor ebx,ebx
    mov eax,[buf2]
    mov bl,10
    .divide:
        xor edx,edx
        div ebx
        add dl,'0'
        push dx
        inc ecx
        cmp eax,0
    jnz .divide

    .reverse:
    pop ax
    mov [esi],eax
    inc esi
    loop .reverse
ret

_start:
    mov eax,4
    mov ebx,1
    mov ecx,hello
    mov edx,hellolen
    int 80h

    mov eax,3
    mov ebx,0
    mov ecx,buf2
    mov edx,1
    int 80h

    mov esi,buf2
```

```
    call convert

    mov eax,4
    mov ebx,1
    mov ecx,buf2
    mov edx,8
    int 80h
    mov eax,1
    mov ebx,0
    int 80h
```

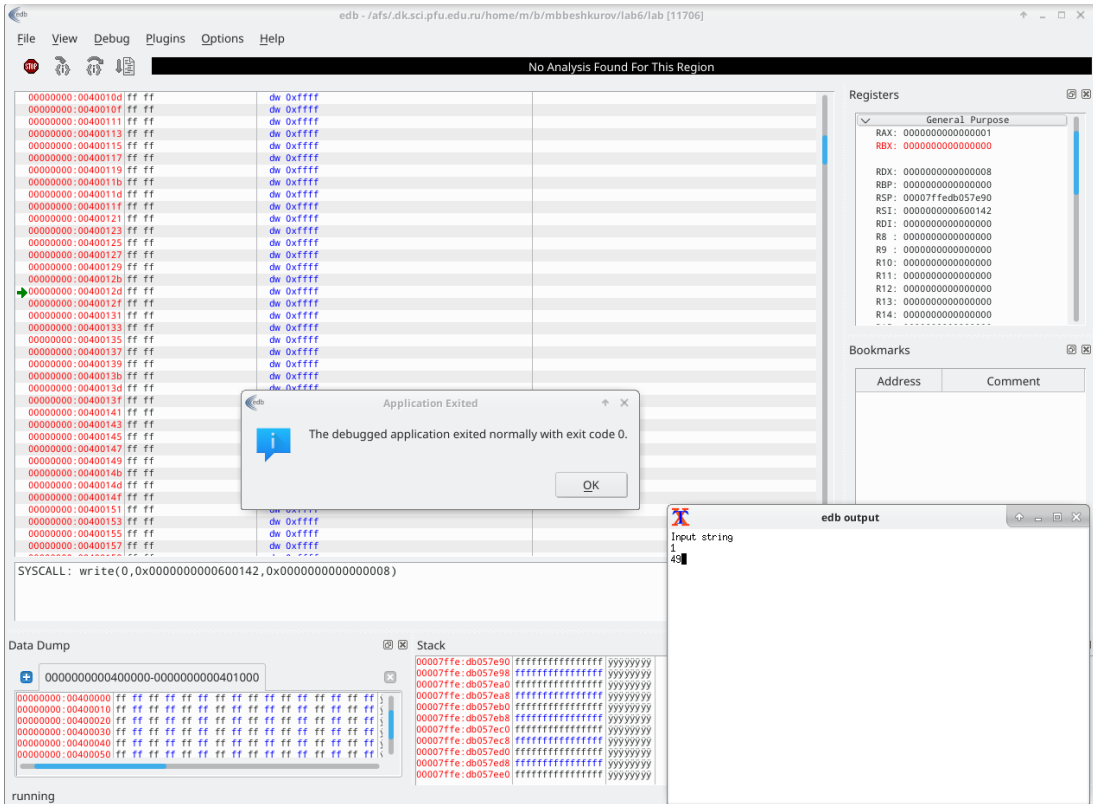
Проверили работу программы

```
mbbeshkurov@dk4n12 ~/lab6 $ nasm -f elf64 -o lab.o lab6.asm
mbbeshkurov@dk4n12 ~/lab6 $ ld -o lab lab.o
mbbeshkurov@dk4n12 ~/lab6 $  ls
lab  lab6.asm  lab.o
mbbeshkurov@dk4n12 ~/lab6 $ ./lab
Input string
1
49mbbeshkurov@dk4n12 ~/lab6 $
```

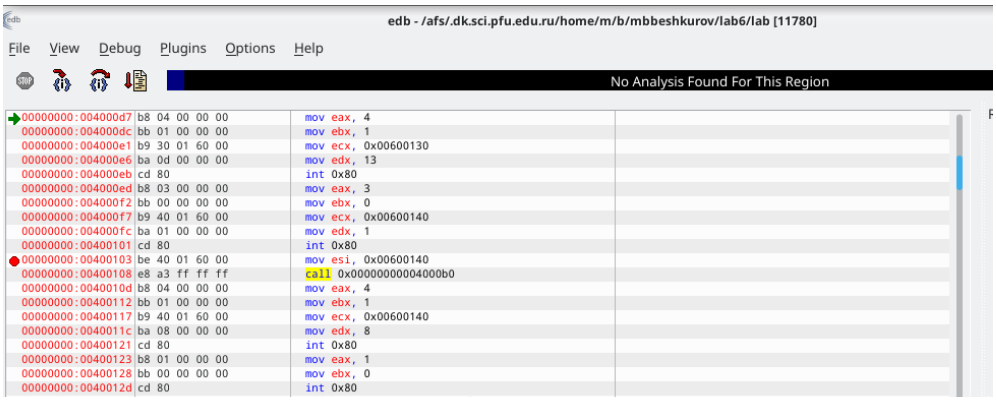
2. Загрузили программу в отладчик. Сделали это с помощью командной строки и команды edb --run lab.

```
mbbeshkurov@dk4n12 ~/lab6 $ edb --run lab
Starting edb version: 0.9.20
Please Report Bugs & Requests At: http://bugs.codef00.com/
comparing versions: [4096] [2324]

** (edb:11600): WARNING **: Invalid borders specified for theme pixmap:
        /usr/share/themes/Breeze/gtk-2.0/../assets/line-h.png,
borders don't fit within the image

** (edb:11600): WARNING **: invalid source position for vertical gradient

** (edb:11600): WARNING **: invalid source position for vertical gradient
```

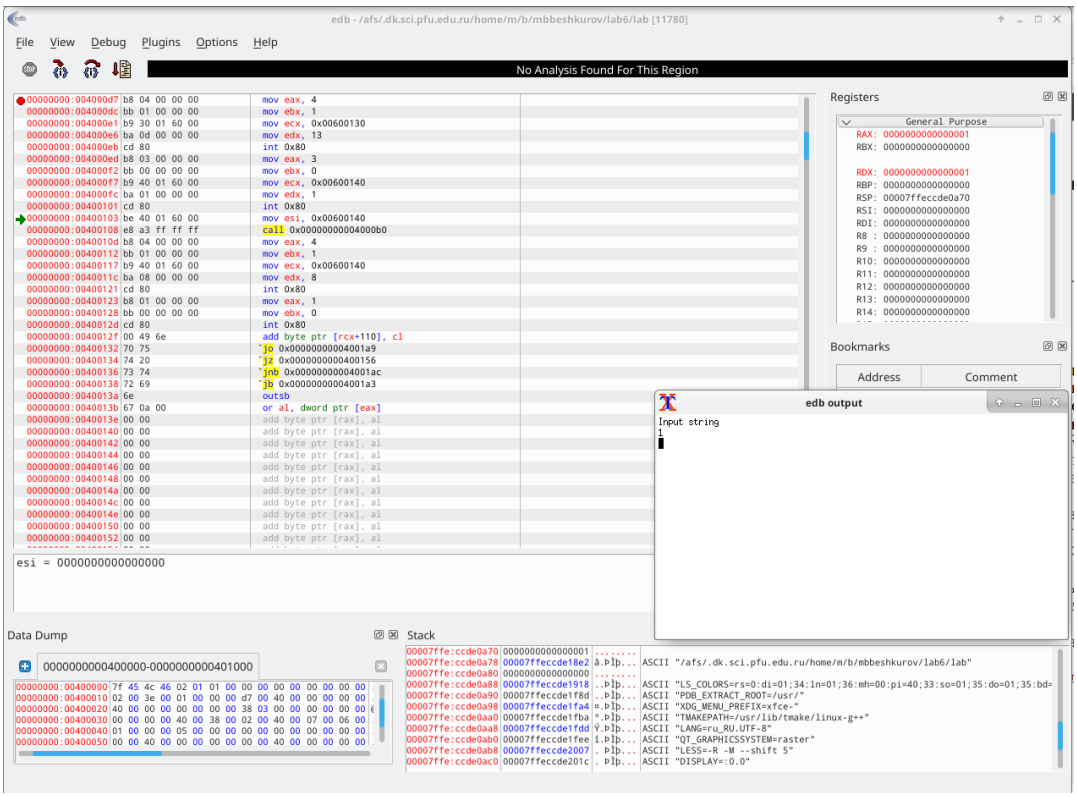
3. Выполнили программу по шагам, нажимая кнопку Step Over панели инструментов или клавишу F7 (находясь в основном окне отладчика), до конца.



4. Поместили в программу точку останова на инструкции, следующей после ввода символа с клавиатуры, щелкнув правой кнопкой по нужной строке дизассемблированного кода и выбрав пункт Add Breakpoint всплывающего меню. Выполнили программу до точки останова, нажав клавишу F9.

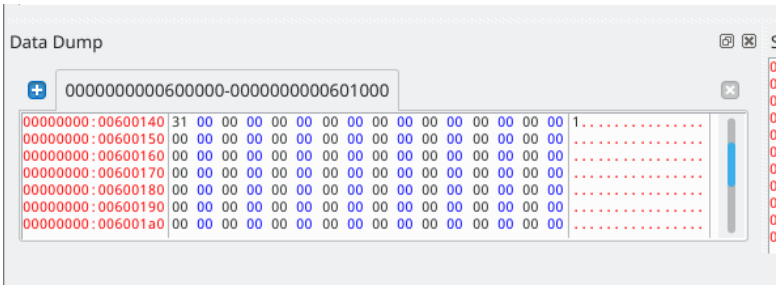


Поствили точку останова.

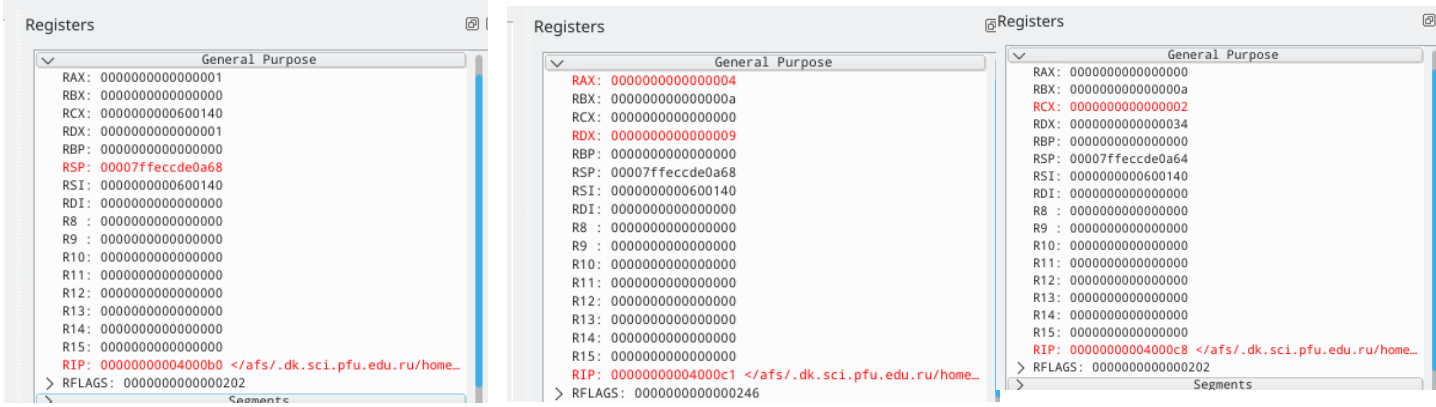


Выполнили программу до точки останова.

5. Вывели в окне дампа памяти содержимое входного буфера, щелкнув в подокне Data Dump правой кнопкой мыши и выбрав пункт Goto Address всплывающего меню.



6. Зашли в процедуру перевода числа в десятичную запись. Выполнили 2 прохода цикла по F7 (Step Into), контролируя значения регистров.



Меняются регистры RAX, RCX, RDX, RSP, RIP.

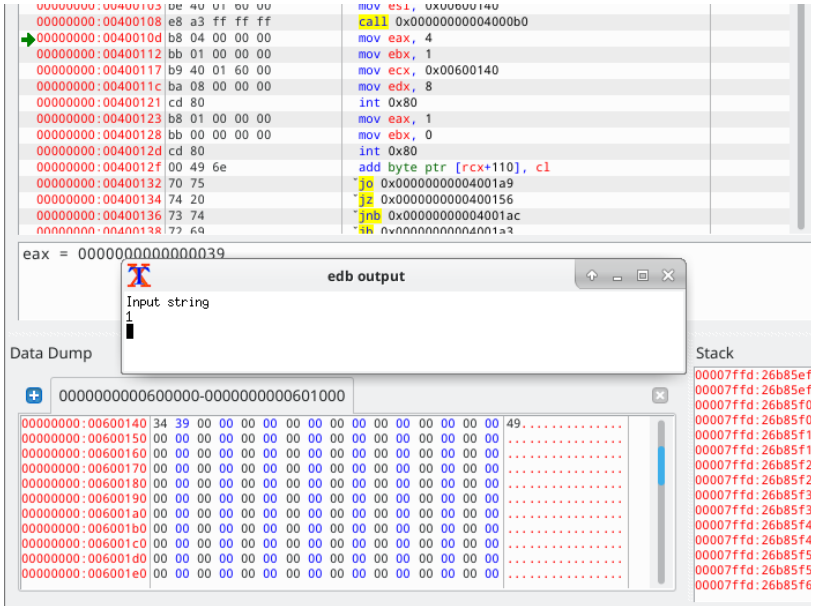
7. Остальные проходы цикла выполнили по F8 (Step Over).

Команда Step Over (F8) приводит к выполнению программы до тех пор, пока не будет достигнута следующая строка ее кода. В отличие от Step Into, вызов процедуры считается единой инструкцией.

8. Определили физический адрес выходного буфера в ОЗУ.

```
mov ecx, 0x00600140
```

9. Вывели ячейки памяти, соответствующие выходному буферу, в подокне Data Dump.



Вывод

Ознакомились с методами отладки при помощи EDB и его основными возможностями. Приобрели навыки написания программ с использованием подпрограмм.

Контрольные вопросы.

1. Какие языковые средства используются в ассемблере для оформления и активизации подпрограмм?

Инструкция call. Она заносит адрес следующей инструкции в стек и загружает адрес подпрограммы в регистр eip, осуществляя переход.
Инструкция ret. Она извлекает из стека адрес, занесенный туда инструкцией call, и заносит его в eip. Это приводит к тому, что вызывающая программа возобновит выполнение с инструкции, следующей за инструкцией call.

2. Опишите структуру подпрограммы на языке ассемблера

Подпрограмма включает в себя инструкцию call, вызывающую подпрограмму, и инструкцию ret, которая возвращает права управления основной программе.

3. Объясните механизм вызова подпрограмм

Сначала в программе выполняется call, она заносит адрес следующей инструкции в стек и загружает адрес подпрограммы в eip, осуществляя переход. После этого подпрограмма выполняется как любой другой код. Когда подпрограмма заканчивает работу, она вызывает инструкцию ret, которая извлекает из стека адрес, занесенный туда инструкцией call, и заносит его в eip. Это приводит к тому, что вызывающая программа возобновит выполнение с инструкции, следующей за инструкцией call.

4. Как используется стек для обеспечения взаимодействия между вызывающей и вызываемой процедурами?

В конце работы программа вызывает инструкцию ret, которая извлекает из стека адрес, занесенный туда call, и заносит его в eip. Это приводит к тому, что вызывающая программа возобновит выполнение с инструкции, следующей за инструкцией call.

5. Каково назначение операнда в команде ret?

Необязательный операнд число определяет количество байтов стека, которые будут вытолкнуты после выталкивания адреса возврата.