

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

дисциплина: Архитектура компьютера

Студент: Трофимов Владислав Алексеевич

Группа: НКАбд-06-25

МОСКВА

2025 г.

Содержание

1 Цель работы	4
2 Задание	5
3 Теоретическое введение.....	6
4 Выполнение лабораторной работы	7
4.1 Основы работы с Midnight Commander.....	7
4.2 Работа в NASM.....	9
4.3 Подключение внешнего файла.....	11
4.4 Задание для самостоятельной работы	14
5 Выводы	19
Список литературы	19

Список иллюстраций

Рис. 4.1: Открытие Midnight Commander

Рис. 4.2: Интерфейс Midnight Commander

Рис. 4.3: Открытый каталог arh-pc

Рис. 4.4: Создание рабочего подкаталога

Рис. 4.5: Создание файла в Midnight Commander

Рис. 4.6: Редактирование файла в Midnight Commander

Рис. 4.7: Проверка сохранения сделанных изменений

Рис. 4.8: Трансляция, компоновка и последующий запуск программы

Рис. 4.9: Копирование файла в рабочий каталог

Рис. 4.10: Создание копии файла в Midnight Commander

Рис. 4.11: Изменение программы

Рис. 4.12: Запуск измененной программы

Рис. 4.13: Редактирую файл

Рис. 4.14: запуск программы с изменненой подпрограммой

Рис. 4.15: Редактирование копии

Рис. 4.16: Запуск программы

Рис. 4.17: Редактирование копии

Рис. 4.18: Запуск своей программы

1 Цель работы

Целью данной лабораторной работы является приобретение практических навыков работы в Midnight Commander, освоение инструкций языка ассемблера mov и int.

2 Задание

1. Основы работы с mc
2. Структура программы на языке ассемблера NASM
3. Подключение внешнего файла
4. Выполнение заданий для самостоятельной работы

3 Теоретическое введение

Midnight Commander (или просто mc) — это программа, которая позволяет просматривать структуру каталогов и выполнять основные операции по управлению файловой системой, т.е. mc является файловым менеджером. Midnight Commander позволяет сделать работу с файлами более удобной и наглядной.

Программа на языке ассемблера NASM, как правило, состоит из трёх секций: секция кода программы (SECTION .text), секция инициализированных (известных во время компиляции) данных (SECTION .data) и секция неинициализированных данных (тех, под которые во время компиляции только отводится память, а значение присваивается в ходе выполнения программы) (SECTION .bss).

Для объявления инициализированных данных в секции .data используются директивы DB, DW, DD, DQ и DT, которые резервируют память и указывают, какие значения должны храниться в этой памяти:

- DB (define byte) — определяет переменную размером в 1 байт;
- DW (define word) — определяет переменную размером в 2 байта (слово);
- DD (define double word) — определяет переменную размером в 4 байта (двойное слово);
- DQ (define quad word) — определяет переменную размером в 8 байт (четверёхбайтное слово);
- DT (define ten bytes) — определяет переменную размером в 10 байт.

Директивы используются для объявления простых переменных и для объявления массивов. Для определения строк принято использовать директиву DB в связи с особенностями хранения данных в оперативной памяти.

Инструкция языка ассемблера mov предназначена для дублирования данных источника в приёмнике. В общем виде эта инструкция записывается в виде `mov dst,src` Здесь операнд `dst` — приёмник, а `src` — источник. В качестве операнда могут выступать регистры (register), ячейки памяти (memory) и

непосредственные значения (`const`).

Инструкция языка ассемблера `int` предназначена для вызова прерывания с указанным номером. В общем виде она записывается в виде `int n`. Здесь `n` — номер прерывания, принадлежащий диапазону 0–255. При программировании в Linux с использованием вызовов ядра `sys_calls` $n=80h$ (принято задавать в шестнадцатеричной системе счисления).

4 Выполнение лабораторной работы

4.1 Основы работы с Midnight Commander

Введя соответствующую команду в терминале (рис. 4.1), я открываю `Midnight Commander` (рис. 4.2).

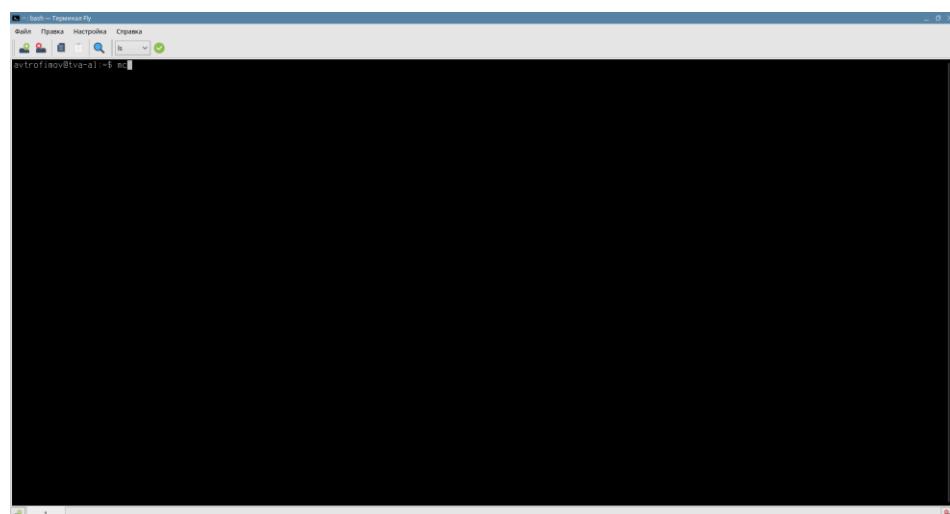


Рис. 4.1: Открытие `Midnight Commander`

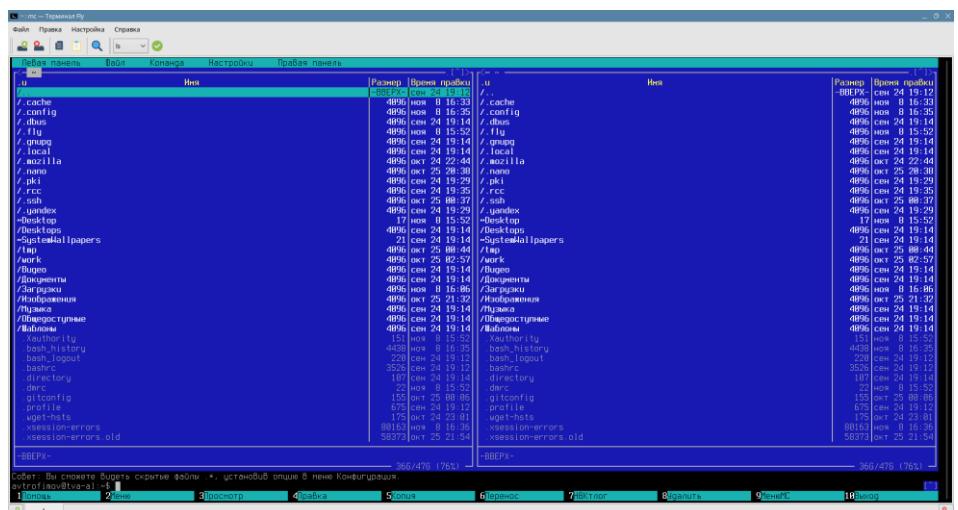


Рис. 4.2: Интерфейс Midnight Commander

Перехожу в созданный каталог в предыдущей лабораторной работе (рис. 4.3).

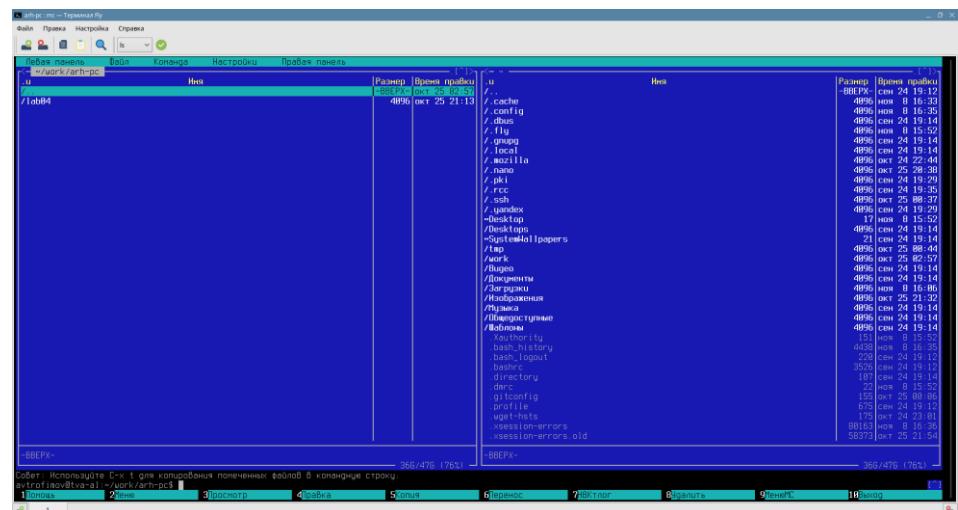


Рис. 4.3: Открытый каталог arch-pc

С помощью функциональной клавиши, я создаю подкаталог lab05, в котором буду работать (рис. 4.4).

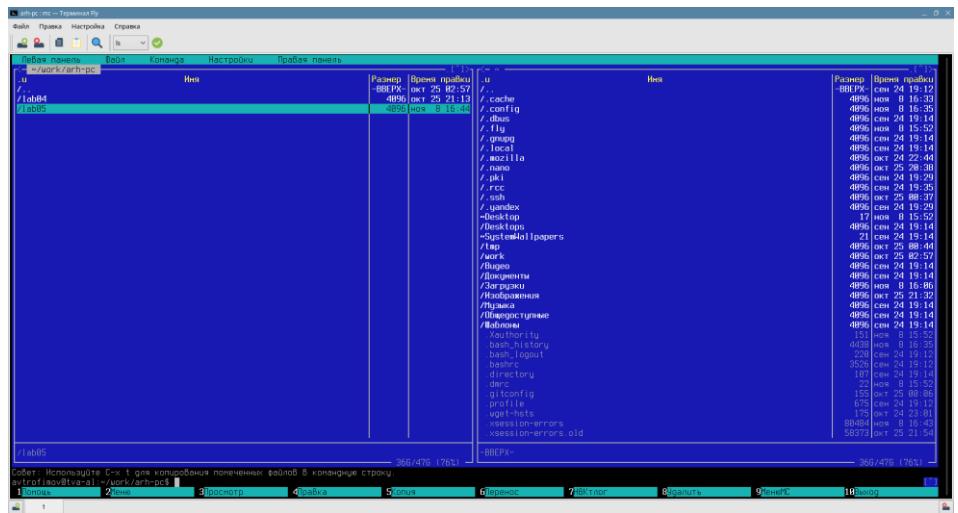


Рис. 4.4: Создание рабочего подкаталога

В строке ввода ввожу команду touch и создаю файл (рис. 4.5).

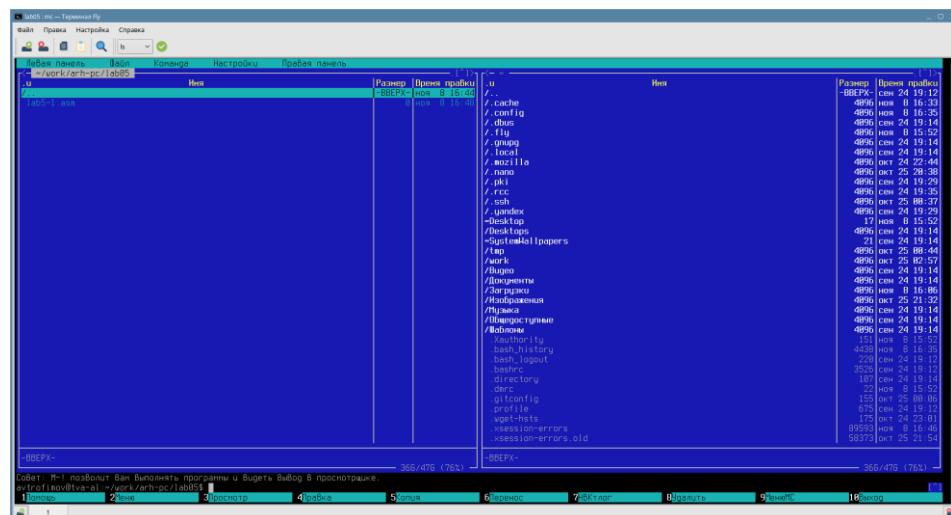


Рис. 4.5: Создание файла в Midnight Commander

4.2 Работа в NASM

С помощью F4 открываю только что созданный файл и вношу код с листинга (рис. 4.6).

The screenshot shows a terminal window titled "lab05.mc — Терминалы" with the path "D:\work\erh-pc\erh-pc\lab05\lab05-1.asm". The assembly code is as follows:

```
SECTION .data
msg DB "DEBUG строку", 10

msglen EQU $-msg

SECTION .text
GLOBAL _start
_start:
    mov eax,4
    mov ebx,1
    mov ecx,msg
    mov edx,msglen
    int 80h
    mov eax,3
    mov ebx,0
    mov ecx,buf1
    mov edx,80
    int 80h
    mov eax,1
    mov ebx,0
    int 80h
```

Рис. 4.6: Редактирование файла в Midnight Commander

Проверяю сохраненные изменения с помощью клавиши F3 (рис. 4.7).

The screenshot shows the same terminal window after saving changes. The status bar at the bottom right indicates "334/334" and "100%". The assembly code is identical to the one in Figure 4.6.

Рис. 4.7: Проверка сохранения сделанных изменений

Транслирую и компоную измененный файл, запускаю (рис. 4.8).

```

lab5_lab5_1 - Терминал
Файл Правка Настройка Справка
nasm -f elf lab5-1.asm
ld -a elf_i386 -o lab5-1 lab5-1.o
./lab5-1
Введите строки:
Трофимов Владислав Алексеевич

```

Рис. 4.8: Трансляция, компоновка и последующий запуск программы

4.3 Подключение внешнего файла

Скачанный с ТУИС файл сохраняю в общую папку на своем компьютере, в интерфейсе Midnight Commander перехожу в директорию общей папки, копирую файл в рабочий подкаталог (рис. 4.9).

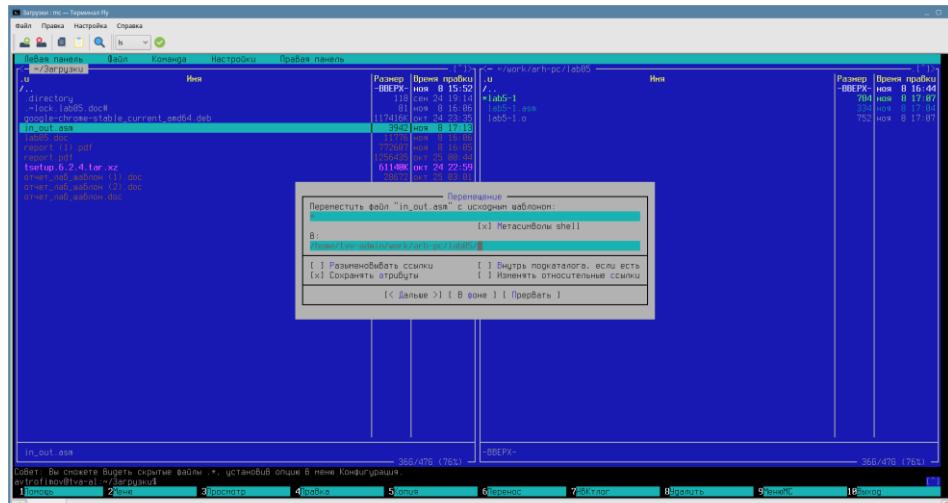


Рис. 4.9: Копирование файла в рабочий каталог

Создаю копию файла для последующей работы с ним (рис. 4.10).

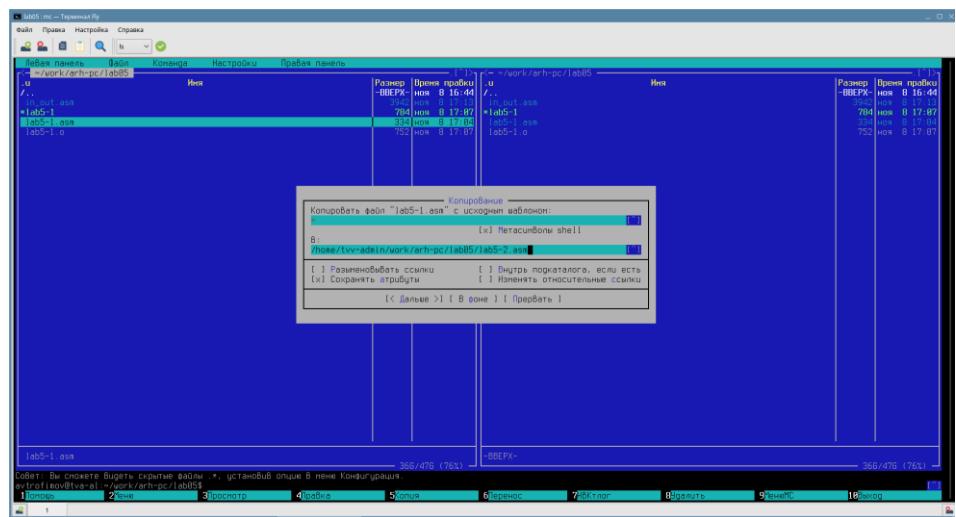


Рис. 4.10: Создание копии файла в Midnight Commander

В копии файла подключаю подпрограмм из подключенного файла `in_out.asm` (рис. 4.11).

```

file: /v/v-s08/n/work/eth-pc/lab05/lab5-1.asm (----) 14.1 | 1+2 23/251 +1943 / (40B) <ELF>
SECTION .data
msg: DB 'Debug строка:', 0h
msglen EQU $-msg

SECTION .text
GLOBAL _start
_start:
    movw $msg, %bx
    call sprintLF
    mov $exit, %ax
    movw %bx, %dx
    call quit

```

Рис. 4.11: Изменение программы

Транслирую, компоную и запускаю программу с подключенным файлом (рис. 4.12).

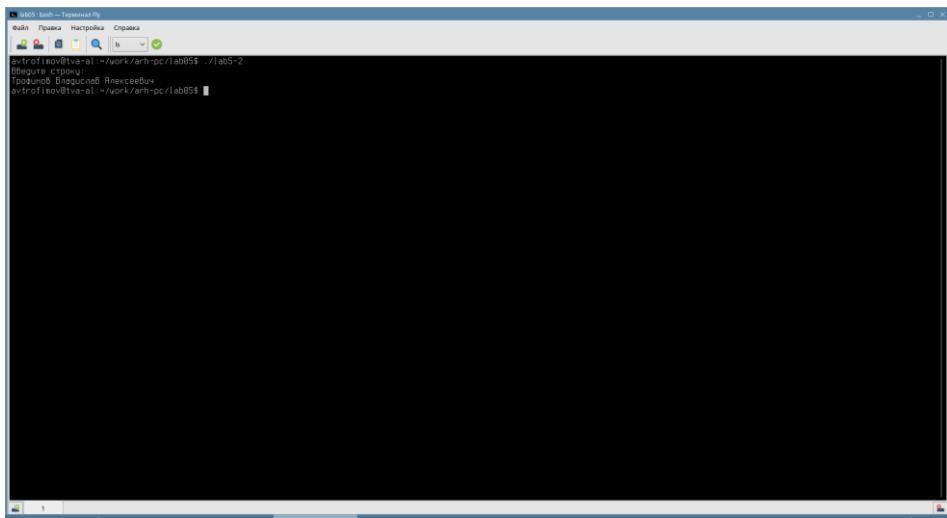


Рис. 4.12: Запуск измененной программы

Редактирую файл и заменяю в нем подпрограмму sprintLF на sprint. Разница подпрограмм в том, что вторая вызывает ввод на той же строке (рис. 4.13) и (рис. 4.14).

```
laba5 mc - Терминал Pu
Файл Правка Настройка Справка
Свигите строки
Помощь Владислав Алексееву
oktrojnikov@eva-01 ~ /work/zhn-pc/lab05$ ./lab05-2
SECTION .data
msg: DB 'Введите строку!',0h
msglen EQU $-msg

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:
    mov eax,msg
    call write
    mov ecx,buf1
    mov edx,80
    call read
    call quit
```

Рис. 4.13: Редактирую файл

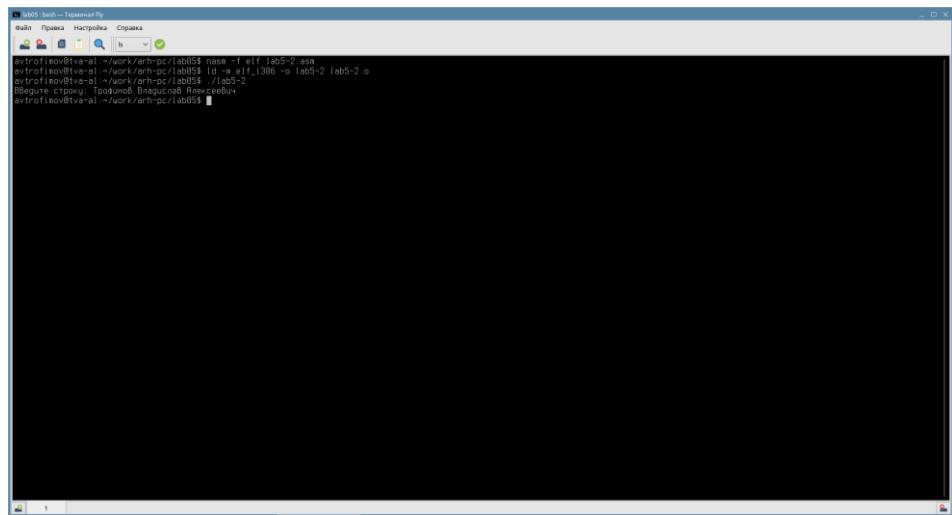


Рис. 4.14: запуск программы с изменненой подпрограммой

4.4 Задание для самостоятельной работы

Создаю копию lab5-1.asm, редактирую так, чтобы в конце выводилась введеная мною строка с клавиатуры (рис. 4.15).

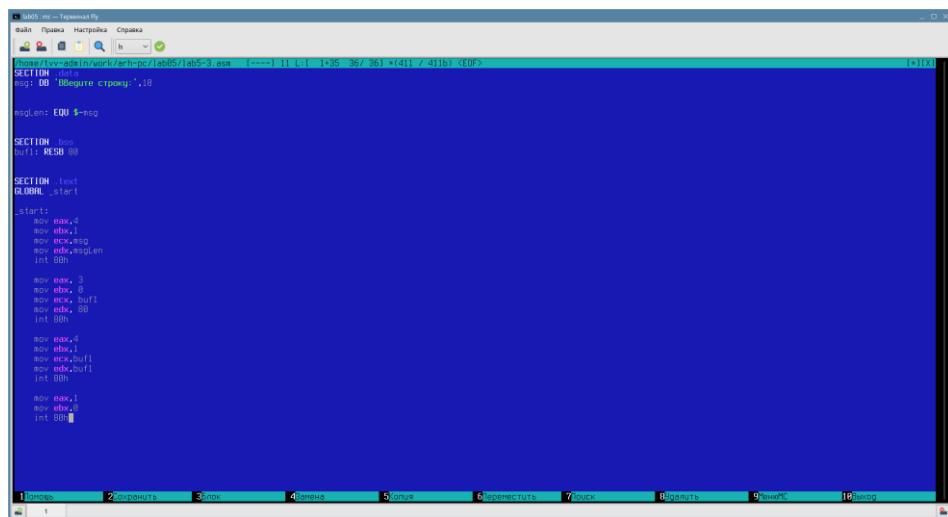


Рис. 4.15: Редактирование копии

Транслирую, компоную и запускаю свою программу (рис. 4.16).

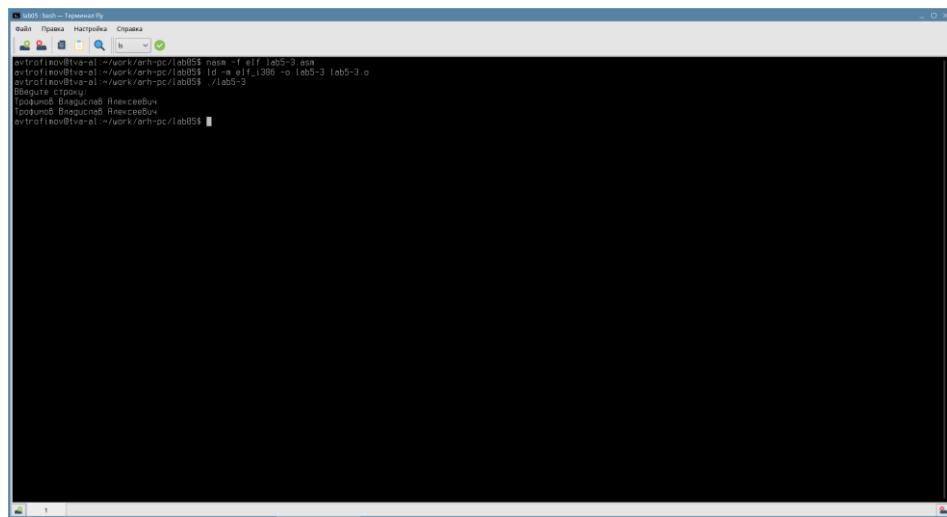


Рис. 4.16: Запуск программы

Код программы:

SECTION .data

msg: DB 'Введите строку:',10

msgLen: EQU \$-msg

SECTION .bss

buf1: RESB 80

SECTION .text

GLOBAL _start

_start:

```
mov eax,4  
mov ebx,1  
mov ecx,msg  
mov edx,msgLen  
int 80h
```

```
mov eax, 3  
mov ebx, 0  
mov ecx, buf1  
mov edx, 80  
int 80h
```

```
mov eax,4  
mov ebx,1  
mov ecx,buf1  
mov edx,buf1  
int 80h
```

```
mov eax,1  
mov ebx,0  
int 80h
```

Создаю копию lab5-2.asm, редактирую так, чтобы в конце выводилась введеная мною строка с клавиатуры (рис. 4.17).

```
lab05.asm -- Терминал Ру
Файл Правка Настройка Справка
h
zinc: /vvn-admin/work/arch-pc/lab05/lab05-4.asm  (-====) 41 L: ( 1>20 29/ 31) * (397 / 322b) 6010 0x0000
zinc: use 16_msf.asm

SECTION .data
msg: DB 'Ведите строку:',0h

msglen EQU $-msg

SECTION .text
GLOBAL _start
_start:
    mov eax,msg
    call readint
    mov ebx,buf1
    mov edx,00
    call read
    mov ebx,1
    mov ebx,buf1
    int 0h
    call quit
```

Рис. 4.17: Редактирование копии

Транслирую, компоную и запускаю свою программу (рис. 4.18).

```
lab05.asm -- Терминал Ру
Файл Правка Настройка Справка
h
avrot@avrot-OptiPlex-5090:~/work/arch-pc/lab05$ nasm -f elf lab05-4.asm
avrot@avrot-OptiPlex-5090:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab05-4 lab05-4.o
avrot@avrot-OptiPlex-5090:~/work/arch-pc/lab05$ ./lab05-4
Ведите строку:Программа Введена вами
Программа Введена вами
avrot@avrot-OptiPlex-5090:~/work/arch-pc/lab05$
```

Рис. 4.18: Запуск своей программы

Код программы:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg: DB 'Ведите строку:',0h
```

```
msgLen: EQU $-msg
```

```
SECTION .bss
```

```
buf1: RESB 80
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
    mov eax,msg
```

```
    call sprint
```

```
    mov ecx, buf1
```

```
    mov edx, 80
```

```
    call sread
```

```
    mov eax,4
```

```
    mov ebx,1
```

```
    mov ecx,buf1
```

```
    int 80h
```

call quit

5 Выводы

При выполнении данной лабораторной работы я приобрёл практические навыки работы в Midnight Commander, а также освоил инструкции языка ассемблера mov и int.

Список литературы

[Курс ТУИС](#)

[Лабораторная работа №5](#)

[Программирование на языке ассемблера NASM Столяров А. В.](#)