

# **Отчет по лабораторной работе №5**

**Дисциплина: архитектура компьютера**

Ицков Андрей Станиславович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	Основы работы с Midnight Commander . . . . .	9
4.2	Работа в NASM . . . . .	11
4.3	Подключение внешнего файла . . . . .	13
4.4	Задание для самостоятельной работы . . . . .	16
<b>5</b>	<b>Выводы</b>	<b>20</b>
	<b>Список литературы</b>	<b>21</b>

## Список иллюстраций

4.1	Открытие Midnight Commander . . . . .	9
4.2	Создание рабочего подкаталога . . . . .	10
4.3	Создание файла в Midnight Commander . . . . .	10
4.4	Редактирование файла в Midnight Commander . . . . .	11
4.5	Проверка сохранения сделанных изменений . . . . .	12
4.6	Трансляция, компоновка и последующий запуск программы . . .	12
4.7	Копирование файла в рабочий каталог . . . . .	13
4.8	Создание копии файла в Midnight Commander . . . . .	14
4.9	Изменение программы . . . . .	14
4.10	Запуск измененной программы . . . . .	15
4.11	Запуск измененной программы с другой подпрограммой . . . . .	15
4.12	Редактирование программы . . . . .	16
4.13	Запуск программы . . . . .	17
4.14	Редактирование программы . . . . .	18
4.15	Запуск программы . . . . .	19

## **Список таблиц**

# 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 инициированных (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 инициированных 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 (четырёхбайтное 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.

```
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.

**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 я открываю Midnight Commander (рис. 4.1).

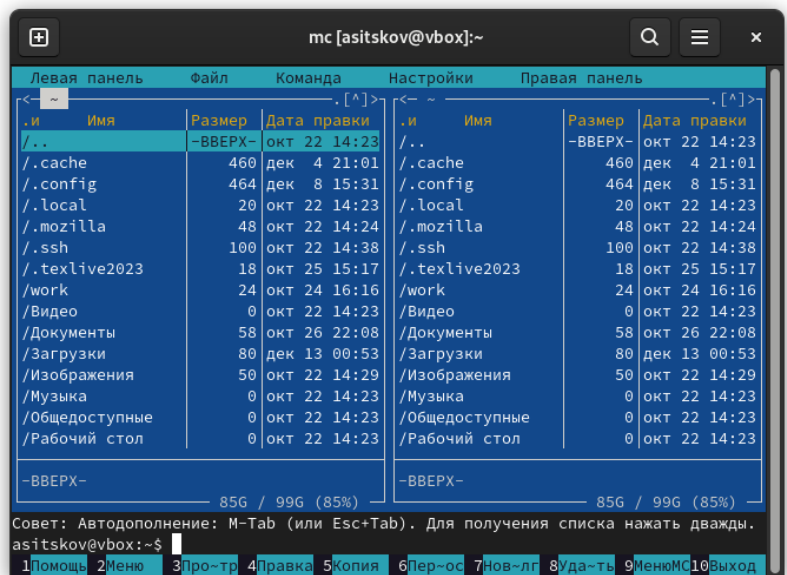


Рис. 4.1: Открывание Midnight Commander

Перехожу в созданный каталог в предыдущей лабораторной работе и создаю подкаталог lab05, в котором буду работать (рис. 4.2).

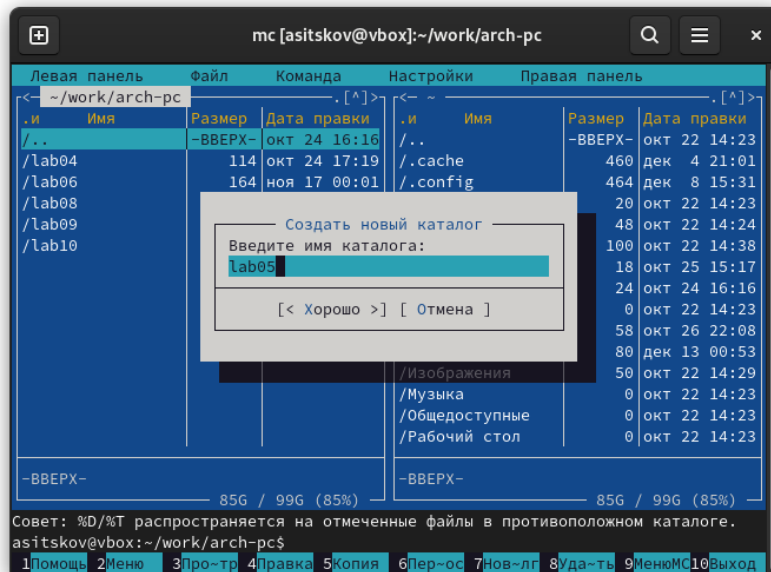


Рис. 4.2: Создание рабочего подкаталога

В строке ввода вводжу команду `touch` и создаю файл (рис. 4.3).

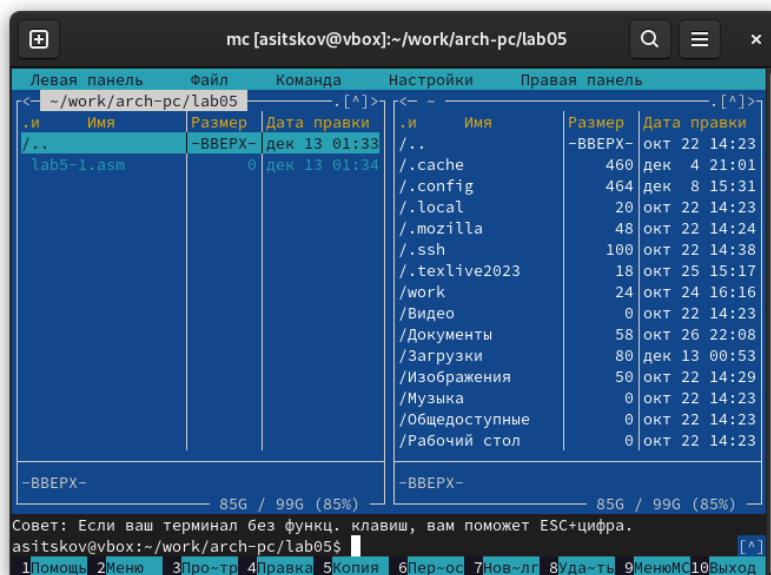
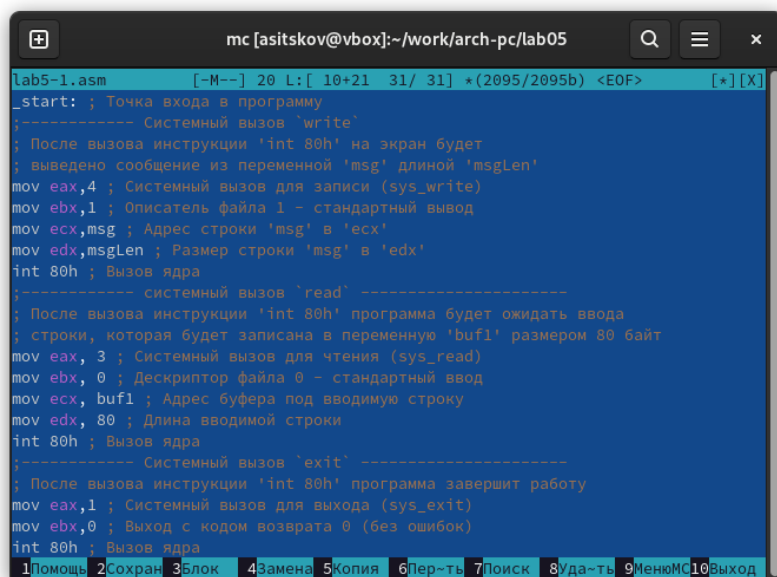


Рис. 4.3: Создание файла в Midnight Commander

## 4.2 Работа в NASM

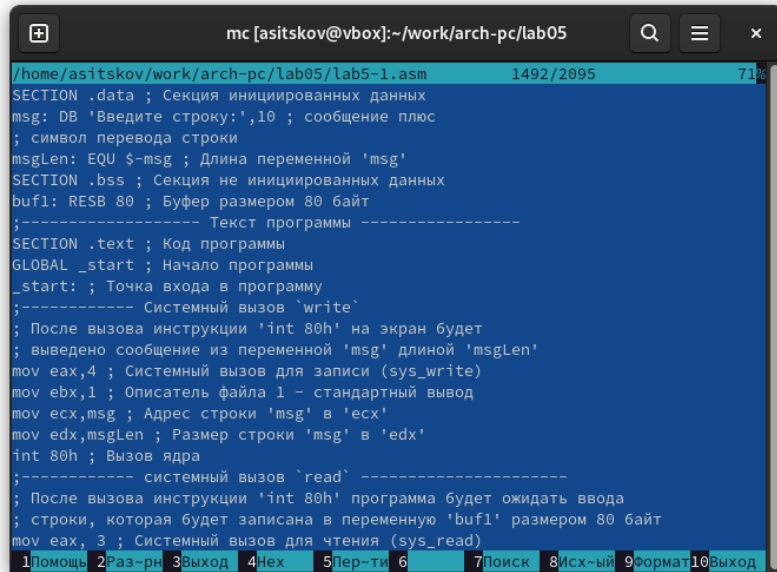
С помощью F4 открываю только что созданный файл и вношу код с листинга (рис. 4.4).



```
lab5-1.asm      [-M--] 20 L:[ 10+21  31/ 31] *(2095/2095b) <EOF>  [*][X]
_start: ; Точка входа в программу
;----- Системный вызов 'write' -----
; После вызова инструкции 'int 80h' на экран будет
; выведено сообщение из переменной 'msg' длиной 'msgLen'
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла 1 - стандартный вывод
mov ecx,msg ; Адрес строки 'msg' в 'ecx'
mov edx,msgLen ; Размер строки 'msg' в 'edx'
int 80h ; Вызов ядра
;----- системный вызов 'read' -----
; После вызова инструкции 'int 80h' программа будет ожидать ввода
; строки, которая будет записана в переменную 'buf1' размером 80 байт
mov eax, 3 ; Системный вызов для чтения (sys_read)
mov ebx, 0 ;Descriptor файла 0 - стандартный ввод
mov ecx, buf1 ; Адрес буфера под вводимую строку
mov edx, 80 ; Длина вводимой строки
int 80h ; Вызов ядра
;----- Системный вызов 'exit' -----
; После вызова инструкции 'int 80h' программа завершит работу
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход
```

Рис. 4.4: Редактирование файла в Midnight Commander

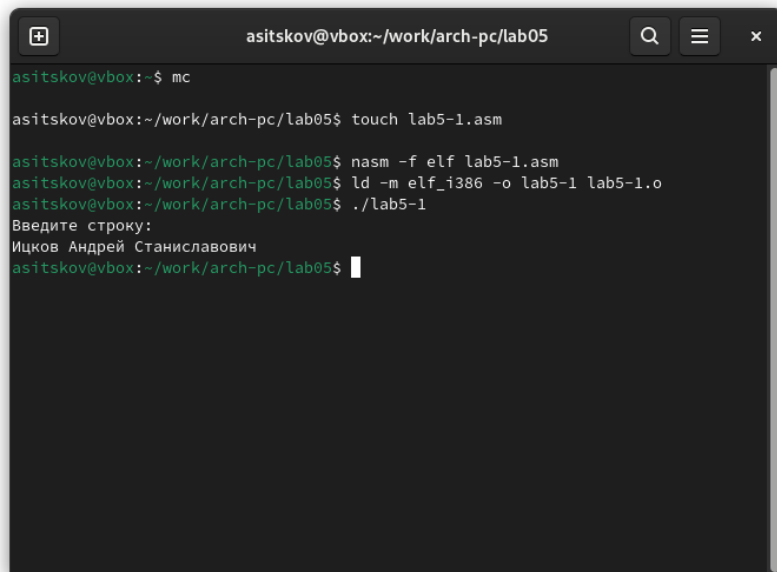
Проверяю сохраненные изменения с помощью клавиши F3 (рис. 4.5).



```
mc [asitskov@vbox]:~/work/arch-pc/lab05
/home/asitskov/work/arch-pc/lab05/lab5-1.asm 1492/2095 71%
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не иницированных данных
buf1: RESB 80 ; Буфер размером 80 байт
;----- Текст программы -----
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
;----- Системный вызов 'write' -----
; После вызова инструкции 'int 80h' на экран будет
; выведено сообщение из переменной 'msg' длиной 'msgLen'
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла 1 - стандартный вывод
mov ecx,msg ; Адрес строки 'msg' в 'ecx'
mov edx,msgLen ; Размер строки 'msg' в 'edx'
int 80h ; Вызов ядра
;----- системный вызов 'read' -----
; После вызова инструкции 'int 80h' программа будет ожидать ввода
; строки, которая будет записана в переменную 'buf1' размером 80 байт
mov eax,3 ; Системный вызов для чтения (sys_read)
1Помощь 2Раз-рн 3Выход 4Нех 5Пер-ти 6 7Поиск 8Исх-ый 9Формат10Выход
```

Рис. 4.5: Проверка сохранения сделанных изменений

Транслирую и компоную измененный файл, запускаю (рис. 4.6).



```
asitskov@vbox:~/work/arch-pc/lab05
asitskov@vbox:~$ mc
asitskov@vbox:~/work/arch-pc/lab05$ touch lab5-1.asm
asitskov@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
asitskov@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
asitskov@vbox:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Ицков Андрей Станиславович
asitskov@vbox:~/work/arch-pc/lab05$
```

Рис. 4.6: Трансляция, компоновка и последующий запуск программы

## 4.3 Подключение внешнего файла

Скачанный с ТУИС файл сохраняю в общую папку на своем компьютере, на виртуальной машине в интерфейсе Midnight Commander перехожу в директорию общей папки, копирую файл в рабочий подкаталог. (рис. 4.7).

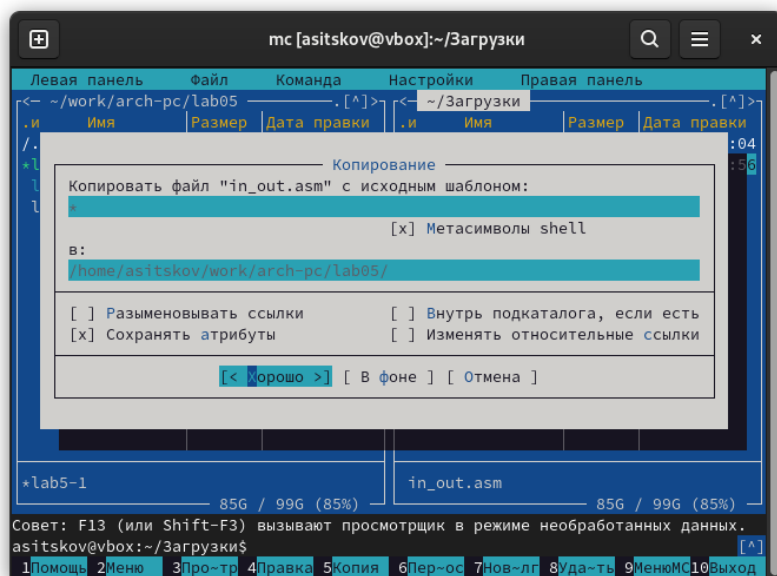


Рис. 4.7: Копирование файла в рабочий каталог

Создаю копию файла для последующей работы с ним (рис. 4.8).

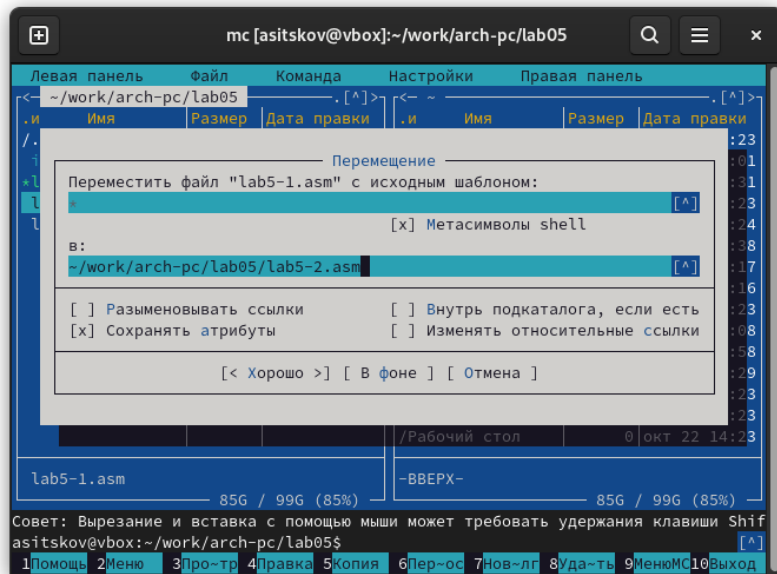


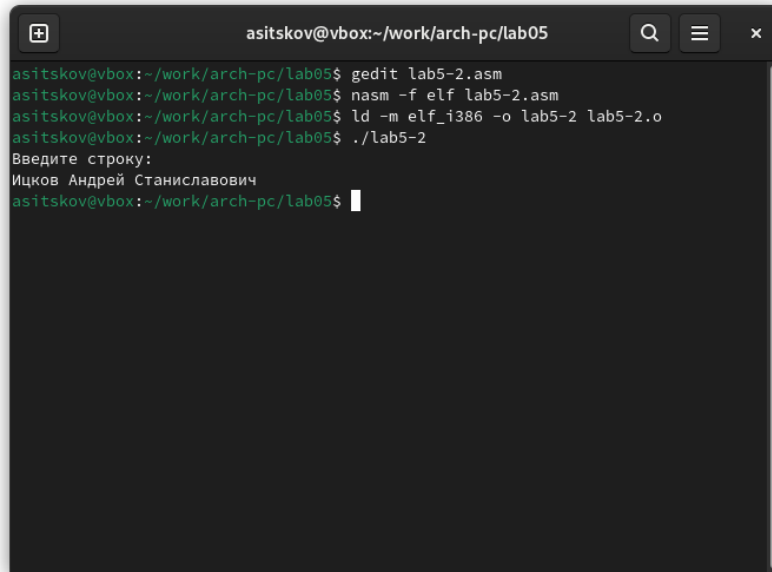
Рис. 4.8: Создание копии файла в Midnight Commander

В копии файла подключаю подпрограмм из подключенного файла (рис. 4.9).

```
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data ; Секция иницированных данных
3 msg: DB 'Введите строку: ',0h ; сообщение
4 SECTION .bss ; Секция не иницированных данных
5 buf1: RESB 80 ; Буфер размером 80 байт
6 SECTION .text ; Код программы
7 GLOBAL _start ; Начало программы
8 _start: ; Точка входа в программу
9 mov eax, msg ; запись адреса выводимого сообщения в 'EAX'
10 call sprintf ; вызов подпрограммы печати сообщения
11 mov ecx, buf1 ; запись адреса переменной в 'EAX'
12 mov edx, 80 ; запись длины вводимого сообщения в 'EBX'
13 call read ; вызов подпрограммы ввода сообщения
14 call quit ; вызов подпрограммы завершения
```

Рис. 4.9: Изменение программы

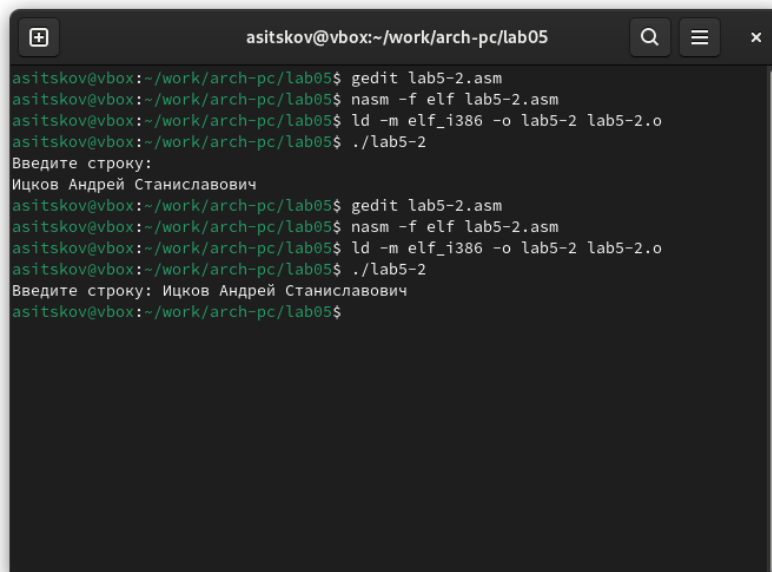
Транслирую, компоную и запускаю программу с подключенным файлом (рис. 4.10).



```
asitskov@vbox:~/work/arch-pc/lab05
asitskov@vbox:~/work/arch-pc/lab05$ gedit lab5-2.asm
asitskov@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
asitskov@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
asitskov@vbox:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Ицков Андрей Станиславович
asitskov@vbox:~/work/arch-pc/lab05$
```

Рис. 4.10: 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. 4.11).



```
asitskov@vbox:~/work/arch-pc/lab05
asitskov@vbox:~/work/arch-pc/lab05$ gedit lab5-2.asm
asitskov@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
asitskov@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
asitskov@vbox:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Ицков Андрей Станиславович
asitskov@vbox:~/work/arch-pc/lab05$ gedit lab5-2.asm
asitskov@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
asitskov@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
asitskov@vbox:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Ицков Андрей Станиславович
asitskov@vbox:~/work/arch-pc/lab05$
```

Рис. 4.11: Запуск изменной программы с другой подпрограммой

## 4.4 Задание для самостоятельной работы

Изменяю программу lab5-1.asm так, чтобы в конце выводилась введенная мною строка с клавиатуры (рис. 4.12).

```
1 SECTION .data
2
3 msg: DB 'Введите строку:',10
4 msgLen: EQU $-msg
5
6 SECTION .bss
7 buf1: RESB 80
8
9 SECTION .text
10
11 GLOBAL _start
12
13 _start:
14     mov     eax, 4
15     mov     ebx, 1
16     mov     ecx, msg
17     mov     edx, msgLen
18     int     80h
19     mov     eax, 3
20     mov     ebx, 0
21     mov     ecx, buf1
22     mov     edx, 80
23     int     80h
24     mov     eax, 4
25     mov     ebx, 1
26     mov     ecx, buf1
27     mov     edx, buf1
28     int     80h
29     mov     eax, 1
30     mov     ebx, 0
31     int     80h
```

Рис. 4.12: Редактирование программы

Код программы:

**SECTION** .data

msg: DB 'Введите строку:',10

msgLen: EQU \$-msg

**SECTION** .bss

buf1: RESB 80

**SECTION** .text

**GLOBAL** \_start

\_start:

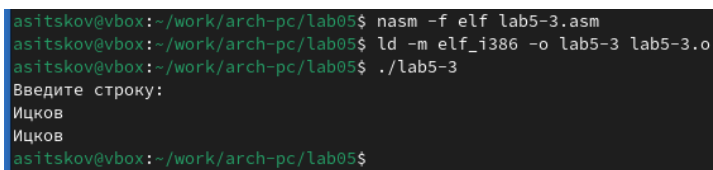


```

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

```

Транслирую, компоную и запускаю программу (рис. 4.13).



```

asitskov@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-3.asm
asitskov@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-3 lab5-3.o
asitskov@vbox:~/work/arch-pc/lab05$ ./lab5-3
Введите строку:
Ицков
Ицков
asitskov@vbox:~/work/arch-pc/lab05$

```

Рис. 4.13: Запуск программы

Изменяю программу lab5-2.asm так, чтобы в конце выводилась введенная мною строка с клавиатуры (рис. 4.14).

```

1 %include 'in_out.asm'
2
3 SECTION .data
4
5 msg: DB 'Введите строку: ', 0h
6 msgLen: EQU $-msg
7
8 SECTION .bss
9 buf1: RESB 80
10
11 SECTION .text
12 GLOBAL _start
13 _start:
14
15     mov eax, msg
16     call sprint
17
18     mov ecx, buf1
19     mov edx, 80
20
21     call sread
22
23     mov eax, 4
24     mov ebx, 1
25     mov ecx, buf1
26     int 80h
27
28     call quit

```

Рис. 4.14: Редактирование программы

Код программы:

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

```
SECTION .data
```

```
msg: DB 'Введите 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
```

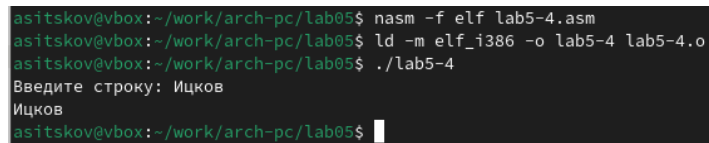
```
mov ebx, 1
```

```
mov ecx, buf1
```

```
int 80h
```

```
call quit
```

Транслирую, компоную и запускаю программу (рис. 4.15).



```
asitskov@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-4.asm
asitskov@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-4 lab5-4.o
asitskov@vbox:~/work/arch-pc/lab05$ ./lab5-4
Введите строку: Ицков
Ицков
asitskov@vbox:~/work/arch-pc/lab05$
```

Рис. 4.15: Запуск программы

## 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

1. Курс на ТУИС
2. Лабораторная работа №5
3. Программирование на языке ассемблера NASM Столяров А. В.