

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

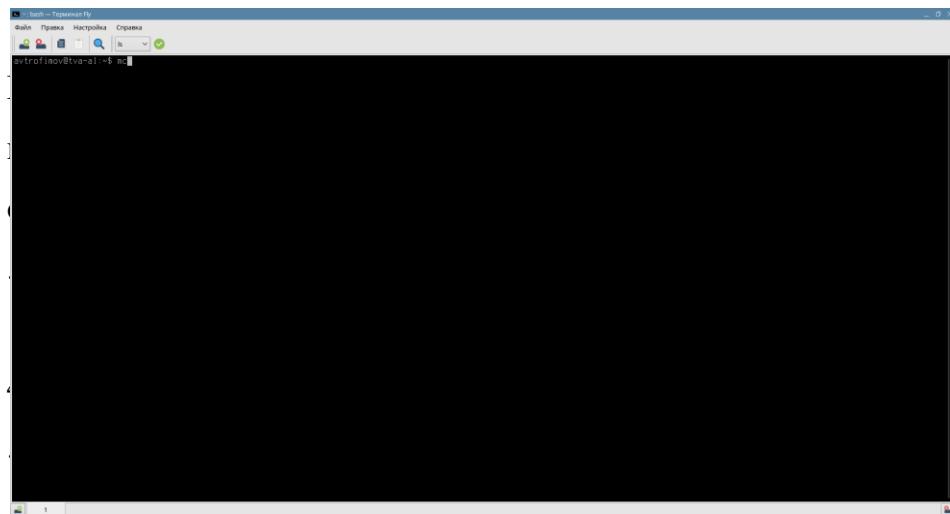
эта инструкция записывается в виде `mov dst,src` Здесь операнд `dst` — приёмник, а `src` — источник. В качестве операнда могут выступать регистры (`register`), ячейки памяти (`memory`) и непосредственные значения (`const`).

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).

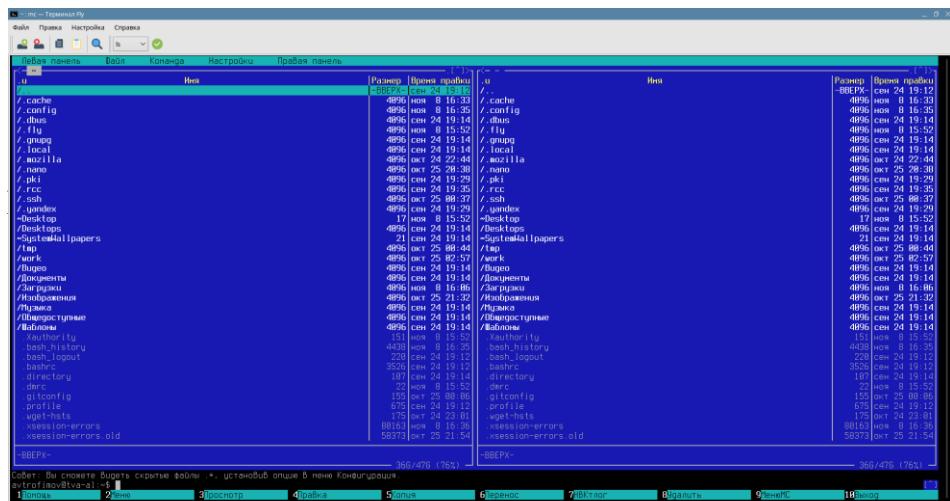
4 Выполнение лабораторной работы

4.1 Основы работы с Midnight Commander

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

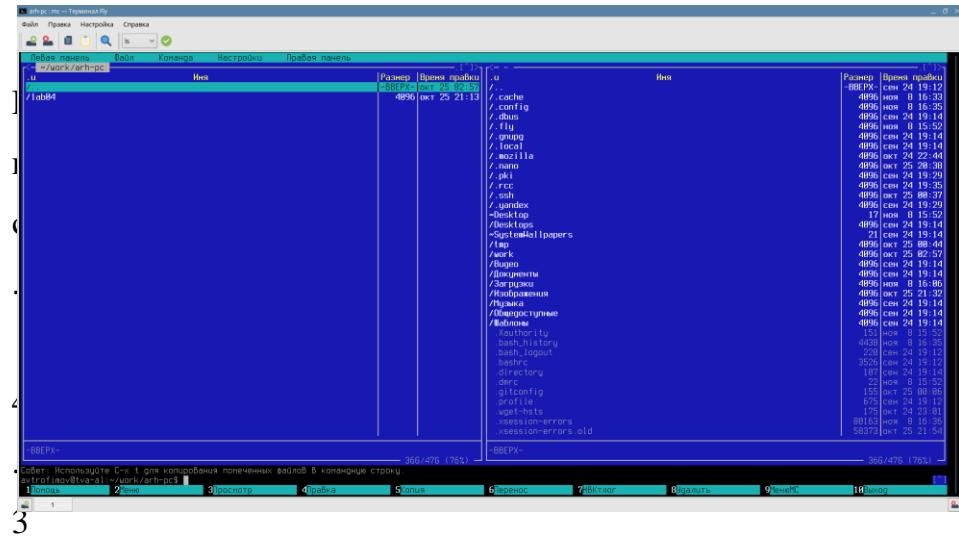


: Открытие `Midnight Commander`



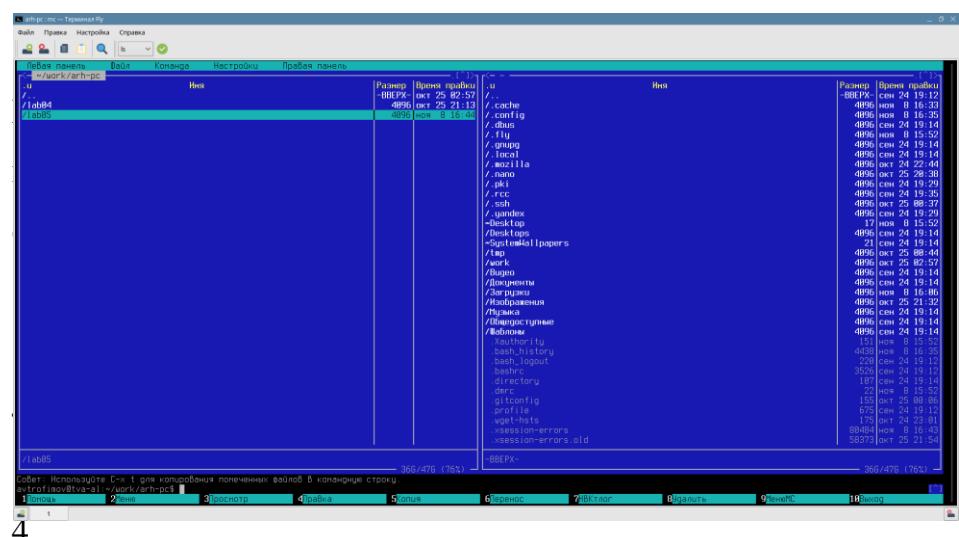
ис. 4.2: Интерфейс Midnight Commander

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



: Открытый каталог arch-pc

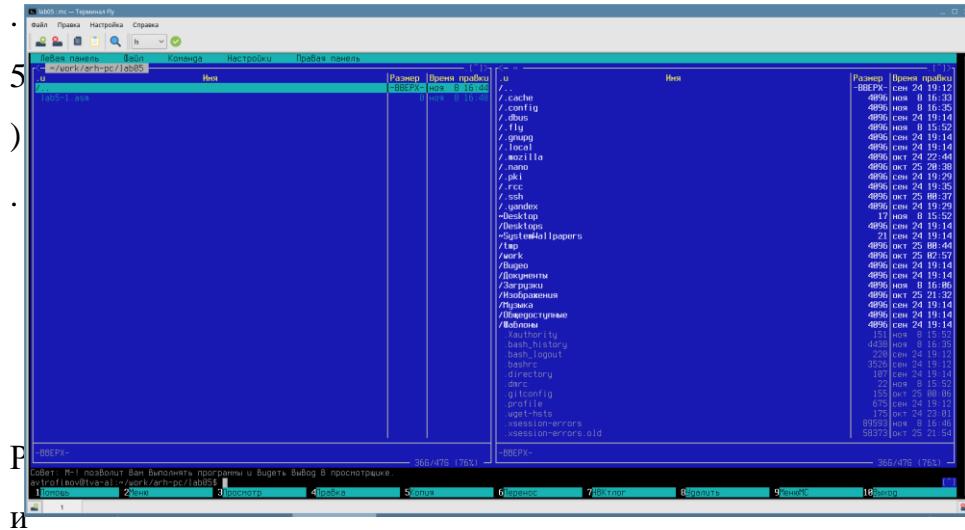
С помощью функциональной клавиши, я создаю подкаталог lab05, в котором буду работать (рис. 4.4).



: Создание рабочего подкатаолога

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

4



с. 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).

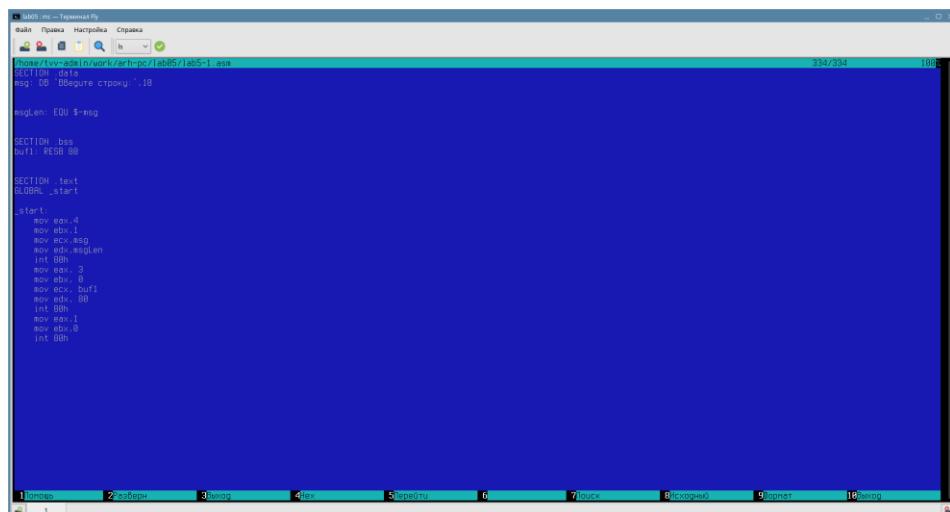
A screenshot of the Linux desktop environment showing the Midnight Commander file manager. A terminal window is open at the bottom with the command 'nasm -f elf32 lab5-1.asm'. The code listed in the terminal is:

```
section .text
        GLOBAL _start
_start:
    mov    ebx, 4
    mov    ebx, 1
    mov    ebx, esp
    mov    ebx, espLen
    int    80h
    mov    ebx, 3
    mov    ebx, 0
    mov    ebx, buf
    mov    ebx, 80
    int    80h
    mov    ebx, 1
    mov    ebx, 0
    int    80h
```

The status bar at the bottom of the terminal shows the path '/home/argh/pc/lab5' and the command '366/476 (76%)'.

Редактирование файла в Midnight Commander

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



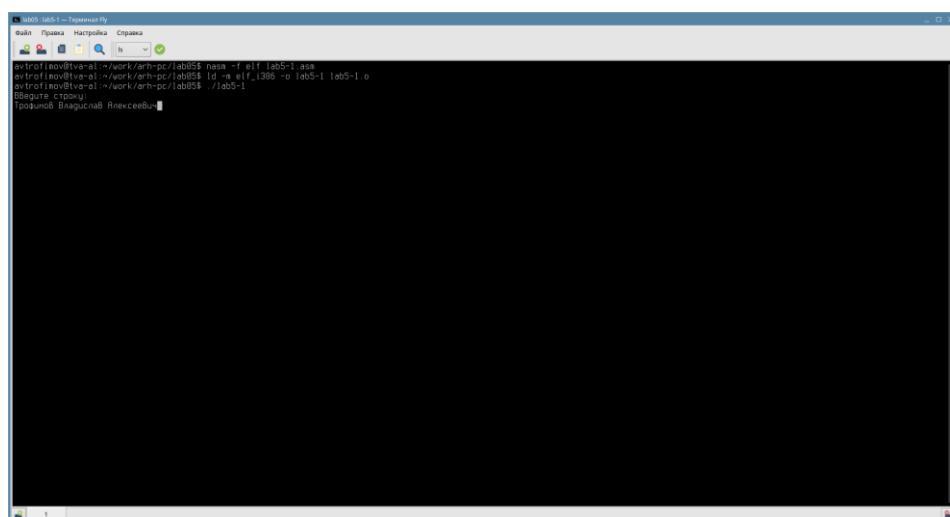
```
lab5.mn — Терминалы
файл Правка Настройка Справка
file:///vyy-admin/work/orh-pc/lab55/lab5-1.asm
SECTION .data
msg db 'Введите строку: ',0
msglen EQU $-msg

SECTION .bss
buf1 RESB 60

SECTION .text
GLOBAL start
start:
    mov eax,4
    mov ebx,1
    mov ecx,0A
    mov edx,0E4D4E4D
    int 80h
    mov eax,3
    mov ebx,0
    mov ecx,0
    mov edx,0
    int 80h
    mov eax,1
    mov ebx,0
    int 80h
```

Проверка сохранения сделанных изменений

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

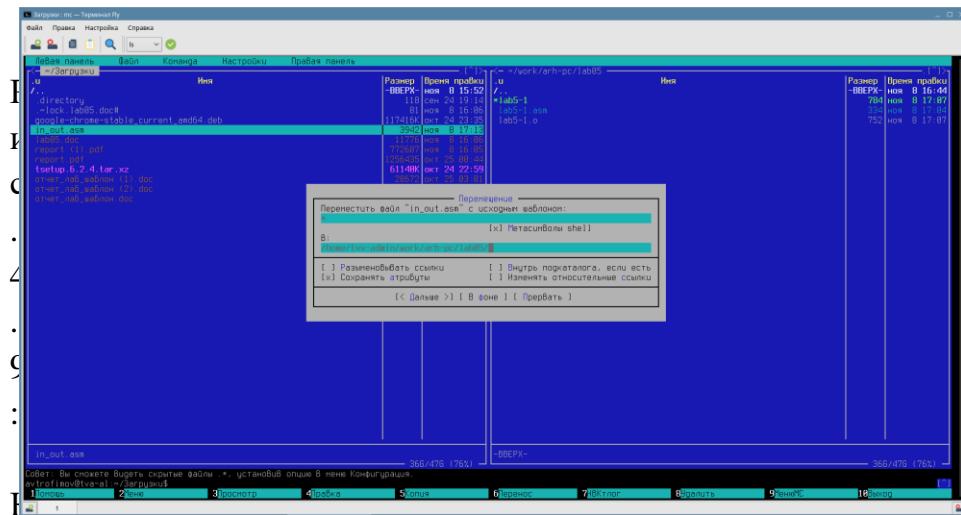


```
lab5.lab5-1 — Терминалы
файл Правка Настройка Справка
file:///vyy-admin/work/orh-pc/lab55/lab5-1.asm
$ ./lab55 nasa -f elf lab5-1.asm
$ ./lab55 ld -m elf_i386 -o lab5-1 lab5-1.o
$ ./lab5-1
DEBUG: ссылаюсь на логин Алексея
$
```

Трансляция, компоновка и последующий запуск программы

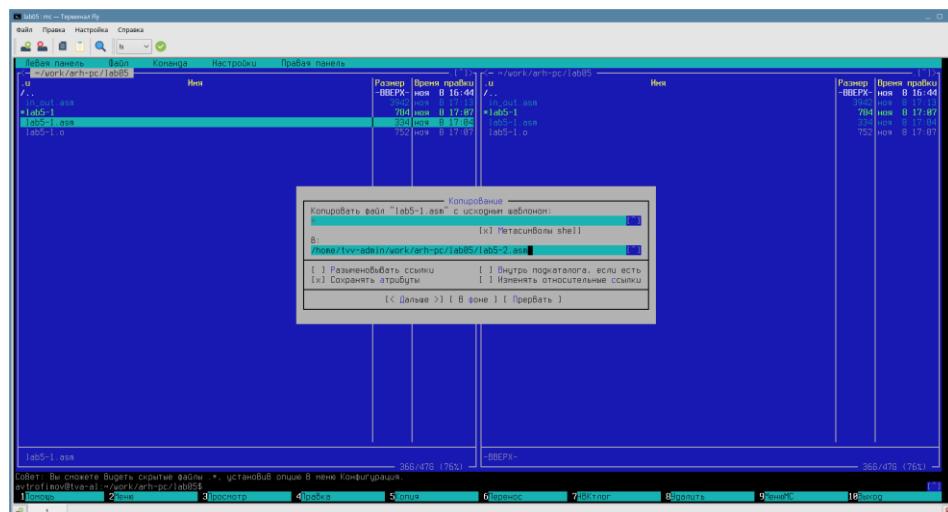
4.3 Подключение внешнего файла

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).



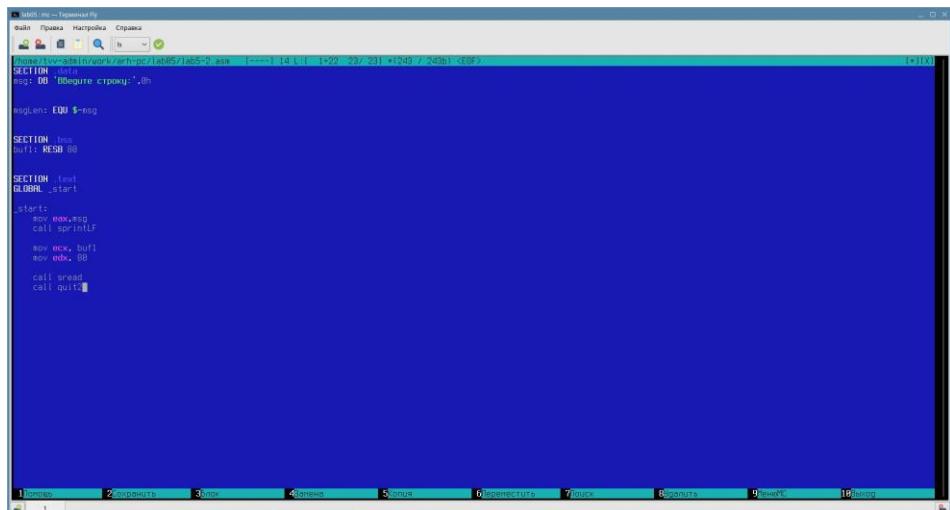
копирование файла в рабочий каталог

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



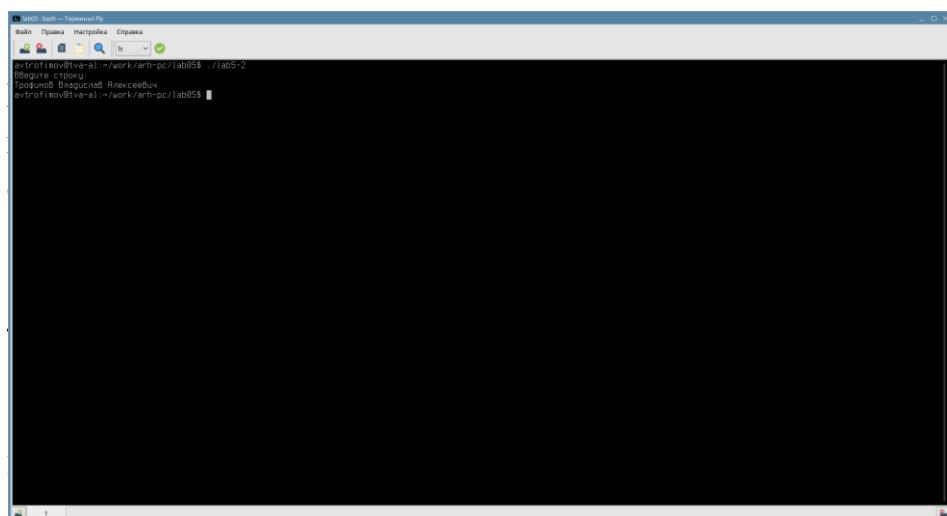
Создание копии файла в Midnight Commander

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



```
lab5_2.asm -- Терминал Ру
файл Правка Настройка Справка
/home/lyv-alek/work/avr-pc/lab05/lab5_2.asm  (->-->) 14 L ( 1+92+23+23) +1908 - 2490 <000>
lmc: 00 0000 строка: 1.0n
msglen EQU $-msg
SECTION data
dbff: MSG DB
SECTION Text
GLOBAL start
start:
    mov msg,esp
    call sprintLF
    mov msg, buf1
    mov msg, buf1
    call read
    call quit
: Изменение программы
```

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

```
file Права Настройка Справка
[2020-05-20 10:00:00] [arm-pc] [lat05/lat05-2.8.1] [->] 15 L: (-1410 197,251 *199 / 2656) 0018 B:00R
[2]include in.out arm
SECTION .text
msg DB 'Широкие строки',0h

msglen EQU $-msg

SECTION .bss
buf1 RESB 90

SECTION .text
GLOBAL _start

_start:
    mov rax,msg
    call sprint
    mov rcx,buf1
    mov rdx,80
    call wread
    call quit
```

Редактирую файл

```
avtrol@novikve-ai ~$ /work/arm-pc/lab05/nasm -f elf lab05-2.asm
avtrol@novikve-ai ~$ /work/arm-pc/lab05/nasm -f elf lab05-2.asm > lab05-2.o
avtrol@novikve-ai ~$ /work/arm-pc/lab05/ld -Ttext=0x8000 -o lab05-2 lab05-2.o
Внедрите строки троеком в фрагмент lab05-2.o
avtrol@novikve-ai ~$ /work/arm-pc/lab05/lab05
```

апуск программы с изменненой подпрограммой

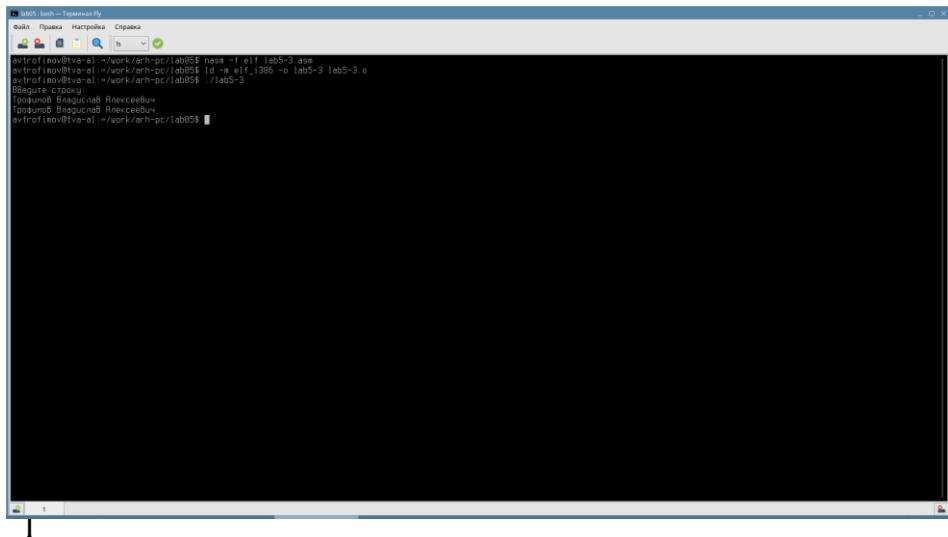
4.4 Задание для самостоятельной работы

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

```
lab5.m - Терминал ПУ
Файл Правка Настройка Справка
autotool@x86-64:~/work/arm-pc/lab05$ nasm -f elf lab5-3.asm
autotool@x86-64:~/work/arm-pc/lab05$ ld -m elf_i386 -o lab5-3 lab5-3.o
autotool@x86-64:~/work/arm-pc/lab05$ ./lab5-3
Ведите строку:
Геннадий Геннадьевич
Геннадий Геннадьевич
autotool@x86-64:~/work/arm-pc/lab05$
```

Редактирование копии

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



6: 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
```

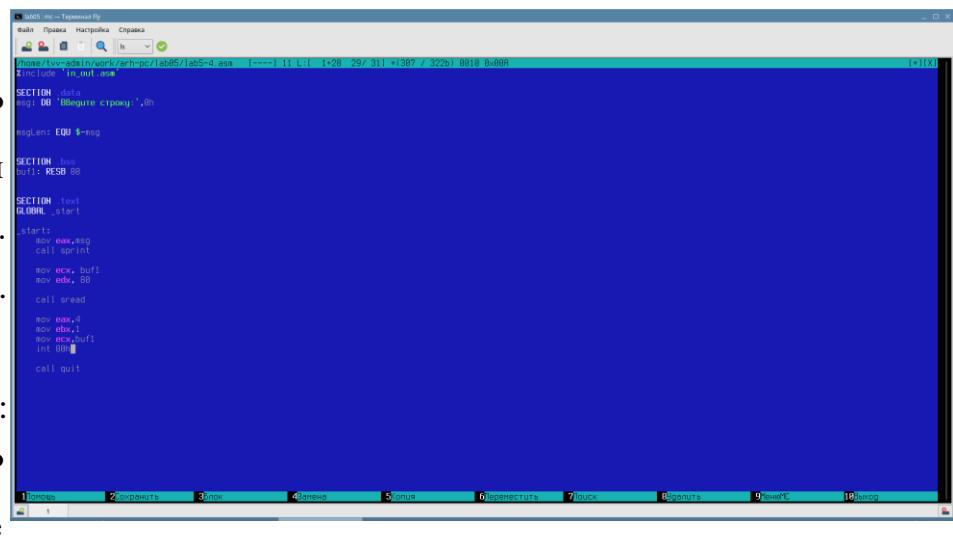
```
    mov edx, 80
```

```
    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).



The screenshot shows a terminal window titled "0005.asm - Терминал Ry" with the following assembly code:

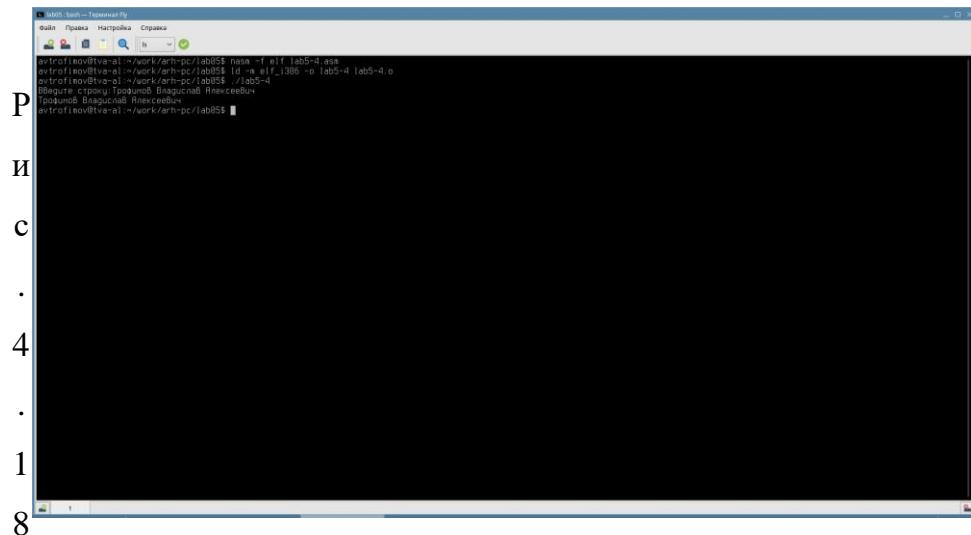
```
P  
И  
C.  
4.  
1  
7:  
P  
е
```

```
SECTION data  
msg1 DB 'Выведите строку:',0h  
msglen EQU $-msg1  
  
SECTION bss  
buff: RESB 80  
  
SECTION text  
GLOBAL _start  
  
.start  
    mov eax,4  
    mov ebx,1  
    mov ecx,buf1  
    mov edx,buf1  
    int 80h  
  
    mov eax,1  
    mov ebx,0  
    int 80h
```

The assembly code includes sections for data, bss, and text, with labels for start, msg1, msglen, and various print/int80h instructions.

дактирование копии

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



: Запуск своей 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
```

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.