

Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Курушин Георгий Романович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Основы работы с Midnight Commander	9
4.2	Работа в NASM	12
4.3	Подключение внешнего файла	13
4.4	Задание для самостоятельной работы	16
5	Выводы	21
	Список литературы	22

Список иллюстраций

4.1	Открытие Midnight Commander	9
4.2	Интерфейс Midnight Commander	10
4.3	Открытый каталог arch-rc	10
4.4	Создание рабочего подкаталога	11
4.5	Создание файла в Midnight Commander	11
4.6	Редактирование файла в Midnight Commander	12
4.7	Проверка сохранения сделанных изменений	12
4.8	Трансляция, компоновка и последующий запуск программы . . .	13
4.9	Копирование файла в рабочий каталог	14
4.10	Создание копии файла в Midnight Commander	14
4.11	Изменение программы	15
4.12	Запуск измененной программы	15
4.13	Запуск измененной программы с другой подпрограммой	16
4.14	Редактирование копии	16
4.15	Запуск своей программы	17
4.16	Редактирование копии	19
4.17	Запуск своей 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 тс
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 (четырёх- рённое 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 (рис. -fig. 4.1), я открываю Midnight Commander (рис. -fig. 4.2).

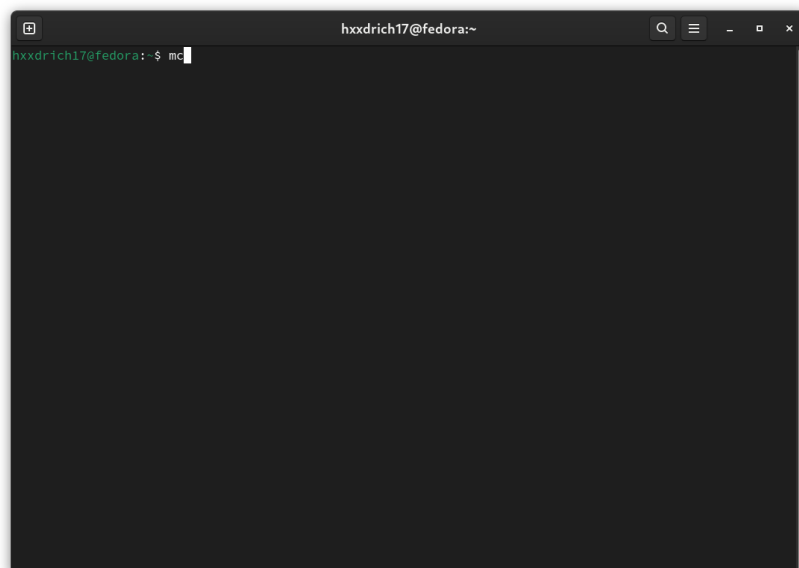


Рис. 4.1: Открытие Midnight Commander

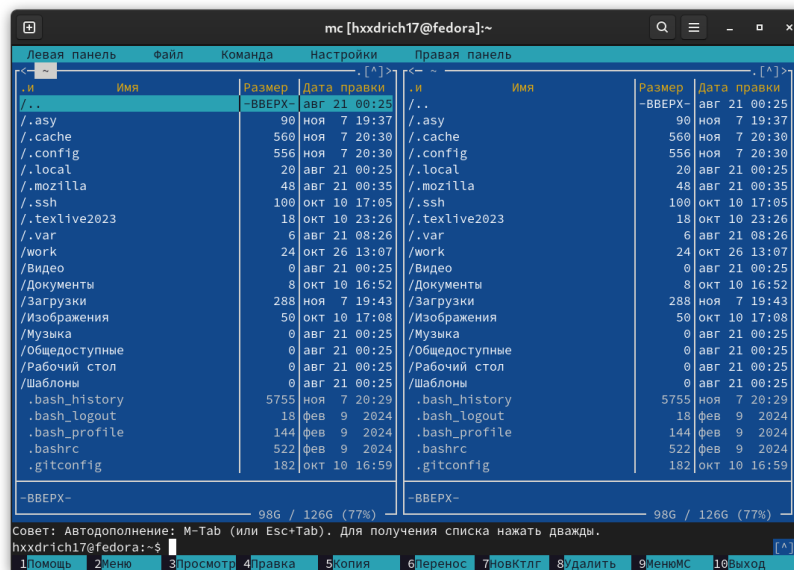


Рис. 4.2: Интерфейс Midnight Commander

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

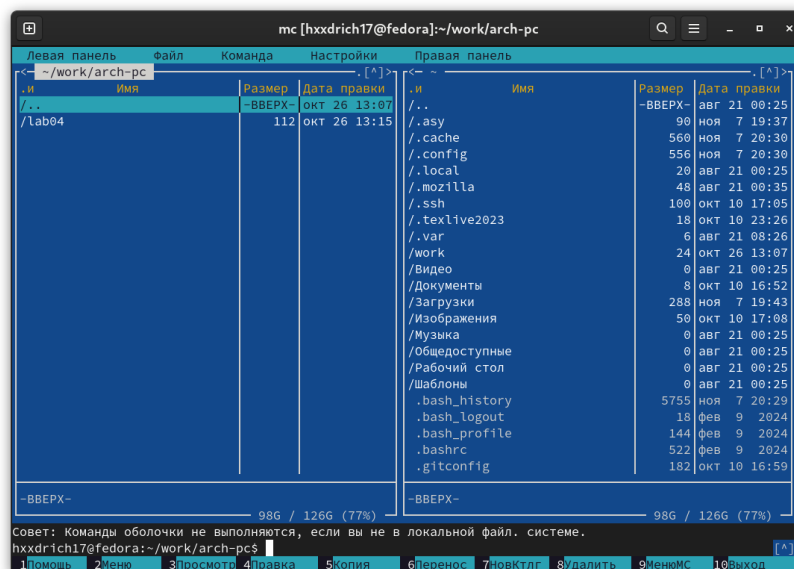


Рис. 4.3: Открытый каталог arch-pc

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

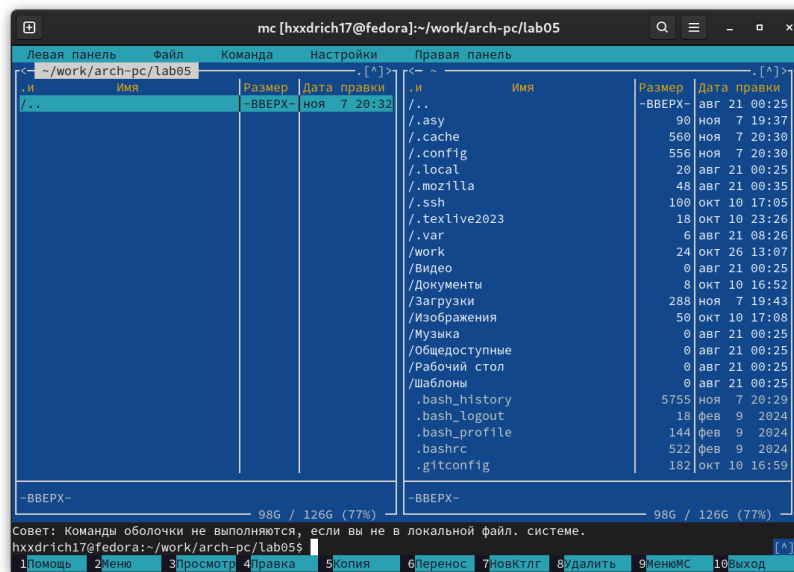


Рис. 4.4: Создание рабочего подкаталога

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

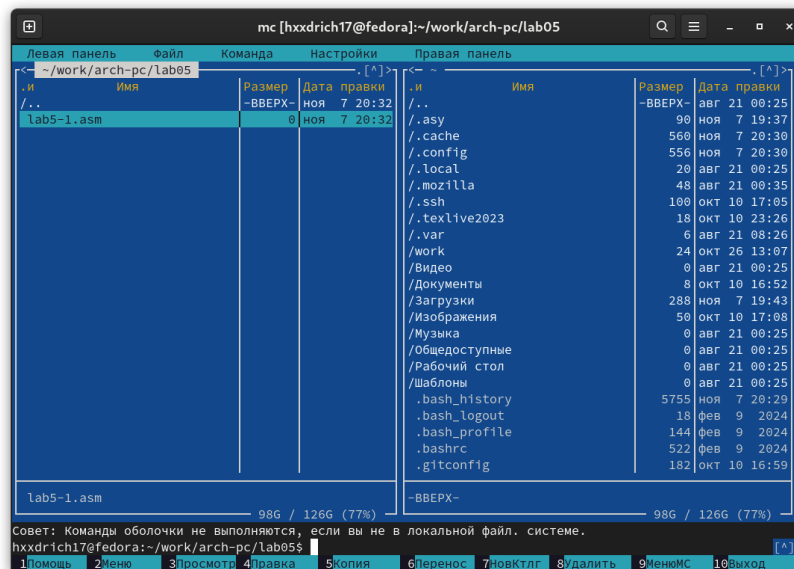
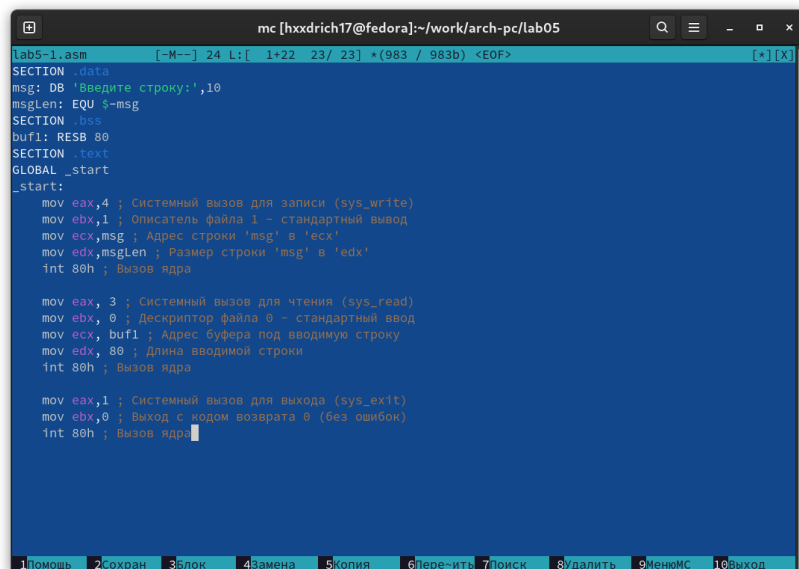


Рис. 4.5: Создание файла в Midnight Commander

4.2 Работа в NASM

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

A screenshot of the Midnight Commander file manager. The title bar shows the user 'hxxdrich17@fedora' and the current directory '~/work/arch-pc/lab05'. The main window displays the contents of the file 'lab5-1.asm'. The code includes section declarations for .data, .bss, and .text, followed by assembly instructions for system calls like sys_write, sys_read, and sys_exit. A status bar at the bottom contains numbered shortcuts from 1 to 10.

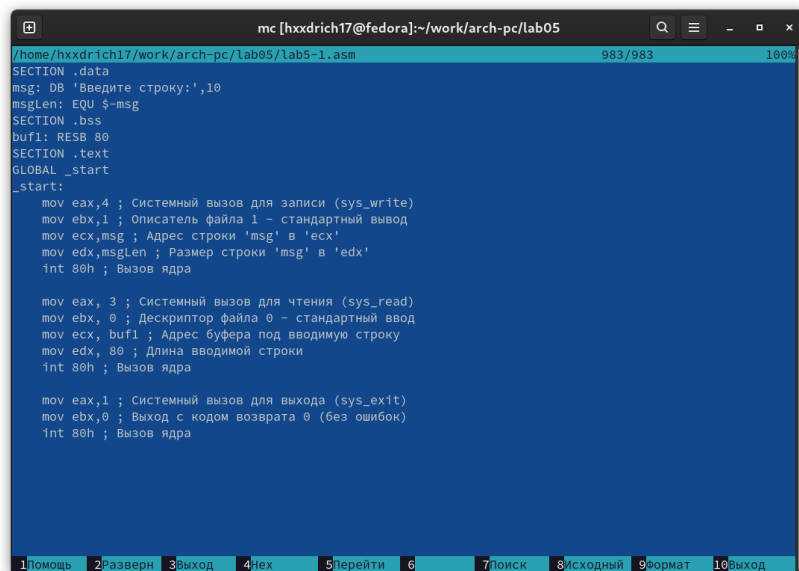
```
lab5-1.asm [-M--] 24 L:[ 1+22 23/ 23] *(983 / 983b) <EOF> [*][X]
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
    mov eax,4 ; Системный вызов для записи (sys_write)
    mov ebx,1 ; Описатель файла 1 - стандартный вывод
    mov ecx,msg ; Адрес строки 'msg' в 'ecx'
    mov edx,msgLen ; Размер строки 'msg' в 'edx'
    int 80h ; Вызов ядра

    mov eax,3 ; Системный вызов для чтения (sys_read)
    mov ebx,0 ; Дескриптор файла 0 - стандартный ввод
    mov ecx,buf1 ; Адрес буфера под вводимую строку
    mov edx,80 ; Длина вводимой строки
    int 80h ; Вызов ядра

    mov eax,1 ; Системный вызов для выхода (sys_exit)
    mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
    int 80h ; Вызов ядра
```

Рис. 4.6: Редактирование файла в Midnight Commander

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

A screenshot of the Midnight Commander file manager, similar to the previous one, but showing the file status after a save operation. The status bar at the bottom now includes '2Заверн' (Save) and '3Выход' (Exit), and the file size is shown as 983/983 bytes (100%).

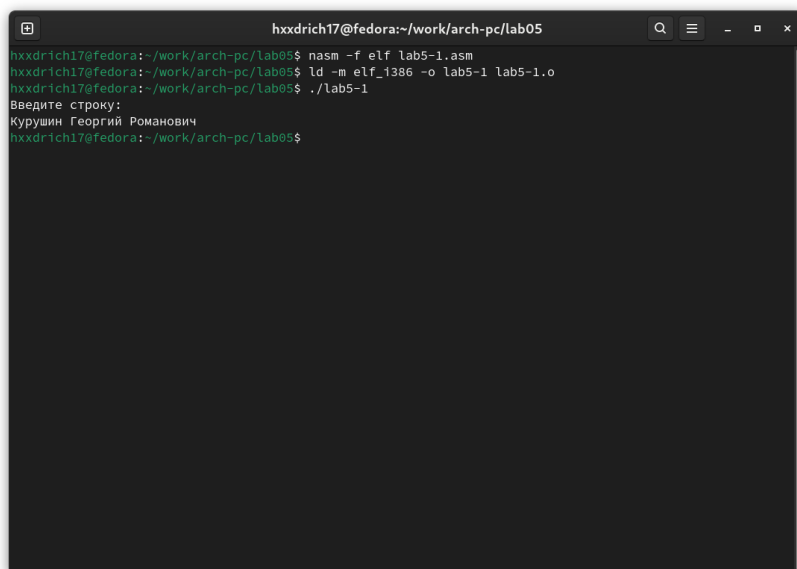
```
/home/hxxdrich17/work/arch-pc/lab05/lab5-1.asm 983/983 100%
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
    mov eax,4 ; Системный вызов для записи (sys_write)
    mov ebx,1 ; Описатель файла 1 - стандартный вывод
    mov ecx,msg ; Адрес строки 'msg' в 'ecx'
    mov edx,msgLen ; Размер строки 'msg' в 'edx'
    int 80h ; Вызов ядра

    mov eax,3 ; Системный вызов для чтения (sys_read)
    mov ebx,0 ; Дескриптор файла 0 - стандартный ввод
    mov ecx,buf1 ; Адрес буфера под вводимую строку
    mov edx,80 ; Длина вводимой строки
    int 80h ; Вызов ядра

    mov eax,1 ; Системный вызов для выхода (sys_exit)
    mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
    int 80h ; Вызов ядра
```

Рис. 4.7: Проверка сохранения сделанных изменений

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

A terminal window titled 'hxxdrich17@fedora:~/work/arch-pc/lab05'. The terminal shows the following commands and output:

```
hxxdrich17@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
hxxdrich17@fedora:~/work/arch-pc/lab05$ ld -m elf_1386 -o lab5-1 lab5-1.o
hxxdrich17@fedora:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Курушин Георгий Романович
hxxdrich17@fedora:~/work/arch-pc/lab05$
```

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, на виртуальной машине в интерфейсе Midnight Commander перехожу в директорию общей папки, копирую файл в рабочий подкаталог. (рис. -fig. 4.9).

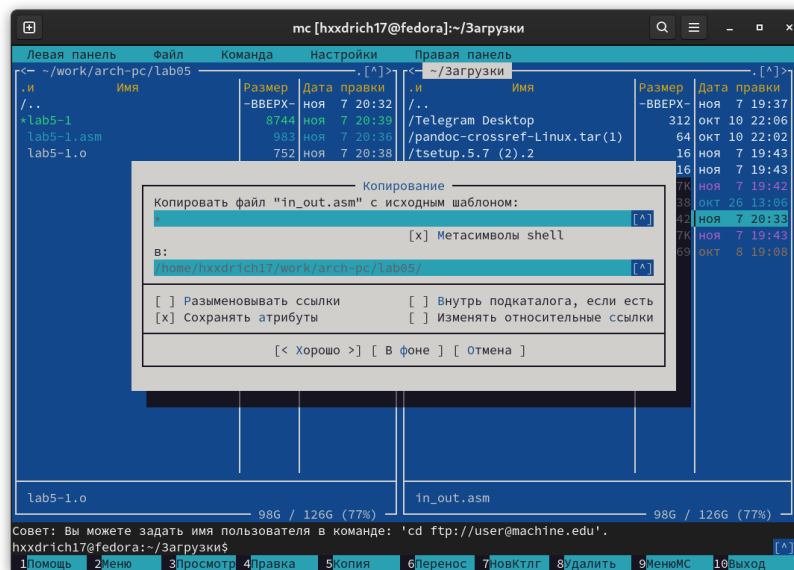


Рис. 4.9: Копирование файла в рабочий каталог

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

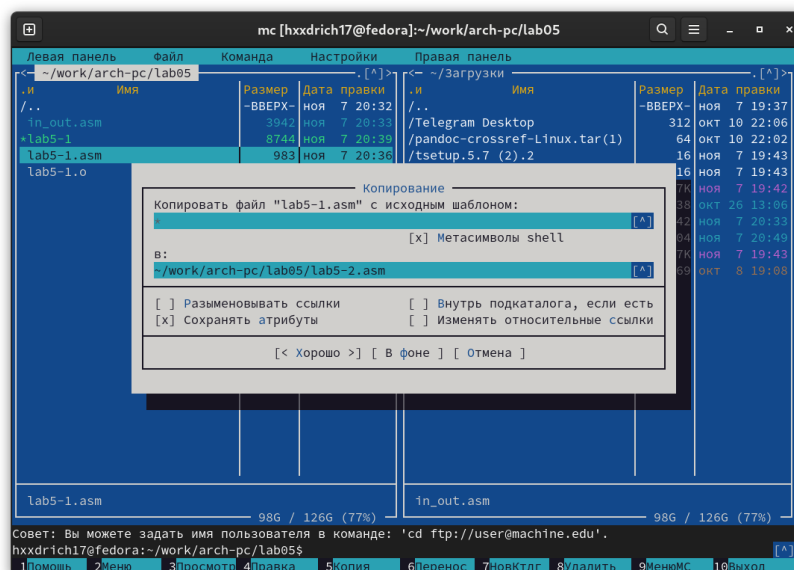
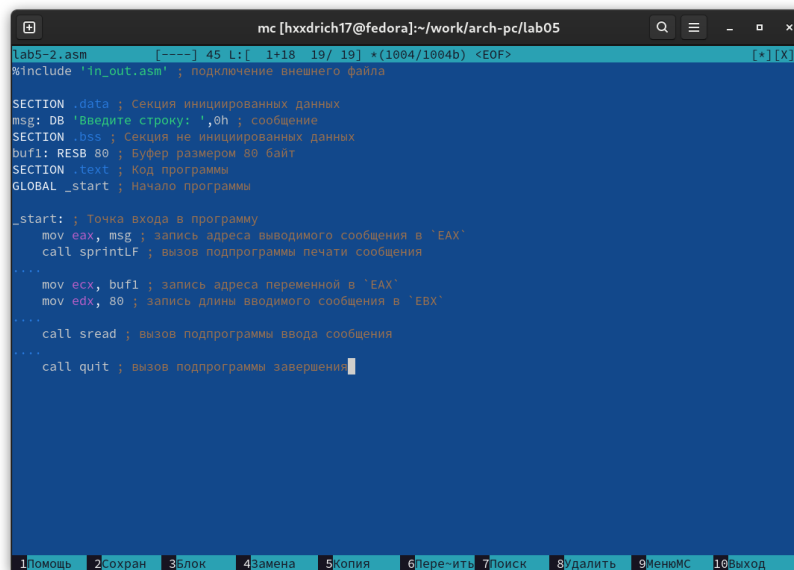


Рис. 4.10: Создание копии файла в Midnight Commander

В копии файла подключаю подпрограмм из подключенного файла (рис. -fig. 4.11).



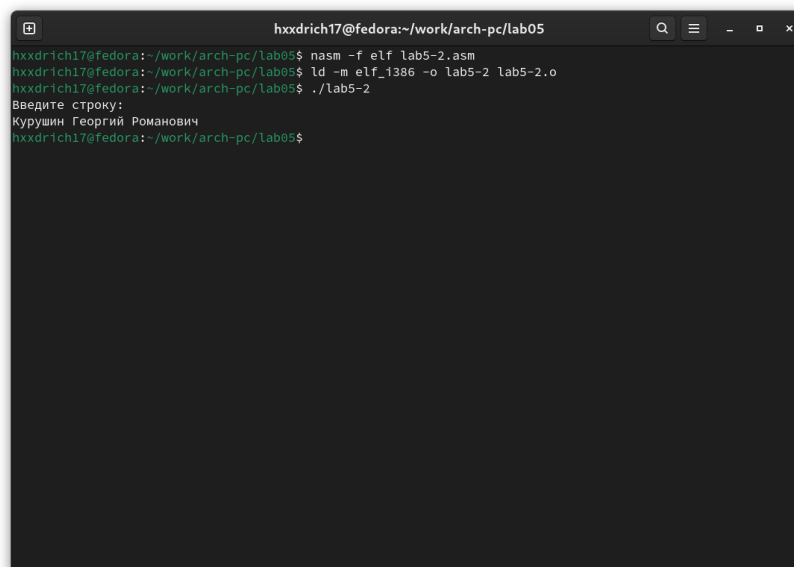
```
lab5-2.asm [----] 45 L: [ 1+18 19/ 19] *(1004/1004b) <EOF> [*] [X]
#include 'in_out.asm' ; подключение внешнего файла

SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не инициализированных данных
buf1: RESB 80 ; Буфер размером 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы

_start: ; Точка входа в программу
    mov eax, msg ; запись адреса выводимого сообщения в 'EAX'
    call sprintf ; вызов подпрограммы печати сообщения
    ....
    mov ecx, buf1 ; запись адреса переменной в 'EAX'
    mov edx, 80 ; запись длины вводимого сообщения в 'EBX'
    ....
    call sread ; вызов подпрограммы ввода сообщения
    ....
    call quit ; вызов подпрограммы завершения
```

Рис. 4.11: Изменение программы

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



```
hxxdrich17@fedora:~/work/arch-pc/lab05
hxxdrich17@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
hxxdrich17@fedora:~/work/arch-pc/lab05$ ld -m elf_1386 -o lab5-2 lab5-2.o
hxxdrich17@fedora:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Курушин Георгий Романович
hxxdrich17@fedora:~/work/arch-pc/lab05$
```

Рис. 4.12: Запуск измененной программы

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

4.4 Задание для самостоятельной работы

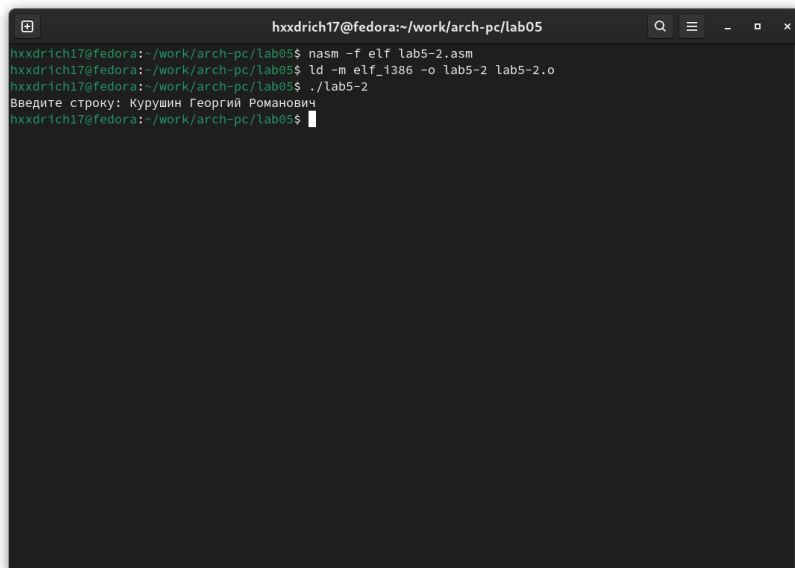


Рис. 4.13: Запуск изменной программы с другой подпрограммой

Создаю копию lab5-1.asm, редактирую так, чтобы в конце выводилась введенная мною строка с клавиатуры (рис. -fig. 4.14).

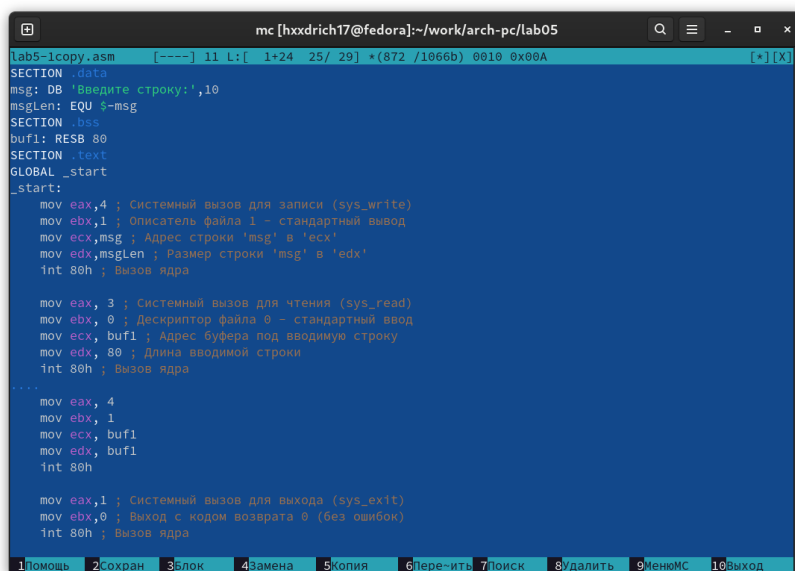
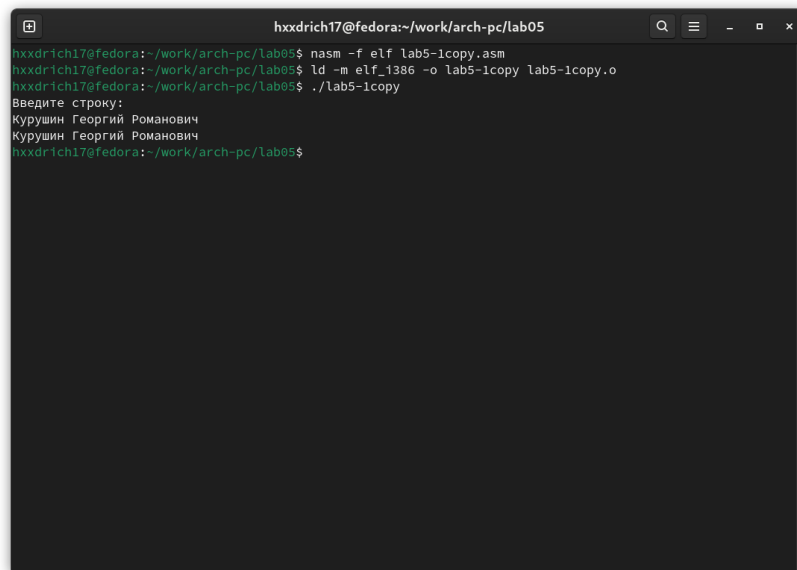


Рис. 4.14: Редактирование копии

Транслирую, компоную и запускаю свою программу (рис. -fig. 4.15).



```
hxxdrich17@fedora:~/work/arch-pc/lab05
hxxdrich17@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-1copy.asm
hxxdrich17@fedora:~/work/arch-pc/lab05$ ld -m elf_1386 -o lab5-1copy lab5-1copy.o
hxxdrich17@fedora:~/work/arch-pc/lab05$ ./lab5-1copy
Введите строку:
Курушин Георгий Романович
Курушин Георгий Романович
hxxdrich17@fedora:~/work/arch-pc/lab05$
```

Рис. 4.15: Запуск своей программы

Код прикладываю

```
SECTION .data

msg: DB 'Введите строку:',10
msgLen: EQU $-msg

SECTION .bss
buf1: RESB 80

SECTION .text

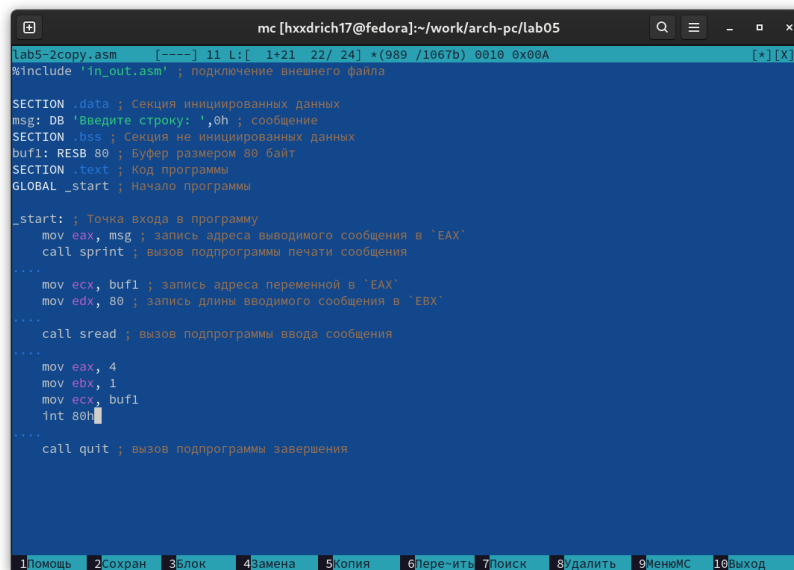
GLOBAL _start

_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 введенная мною строка с клавиатуры (рис. -fig. 4.16).



```
mc [hxxdrich17@fedora:~/work/arch-pc/lab05]
lab5-2copy.asm [----] 11 L:[ 1+21 22/ 24] *(989 /1067b) 0010 0x00A [*] [X]
#include 'in_out.asm' ; подключение внешнего файла

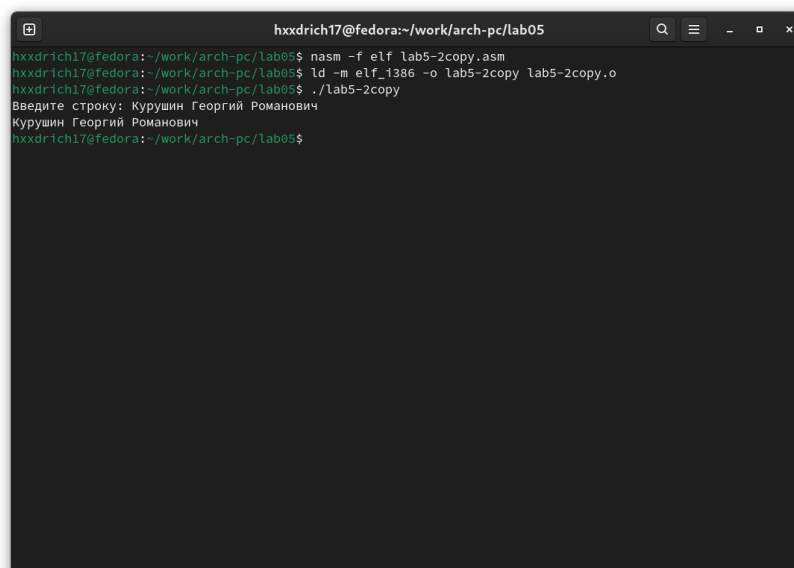
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не иницированных данных
buf1: RESB 80 ; Буфер размером 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы

_start: ; Точка входа в программу
    mov eax, msg ; запись адреса выводимого сообщения в 'EAX'
    call sprint ; вызов подпрограммы печати сообщения
    ....
    mov ecx, buf1 ; запись адреса переменной в 'EAX'
    mov edx, 80 ; запись длины вводимого сообщения в 'EBX'
    ....
    call sread ; вызов подпрограммы ввода сообщения
    ....
    mov eax, 4
    mov ebx, 1
    mov ecx, buf1
    int 80h
    ....
    call quit ; вызов подпрограммы завершения

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пере-ить 7Поиск 8Удалить 9МенюМС 10Выход
```

Рис. 4.16: Редактирование копии

Транслирую, компоную и запускаю свою программу (рис. -fig. 4.17).



```
hxxdrich17@fedora:~/work/arch-pc/lab05
hxxdrich17@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2copy.asm
hxxdrich17@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2copy lab5-2copy.o
hxxdrich17@fedora:~/work/arch-pc/lab05$ ./lab5-2copy
Введите строку: Курушин Георгий Романович
Курушин Георгий Романович
hxxdrich17@fedora:~/work/arch-pc/lab05$
```

Рис. 4.17: Запуск своей программы

Код прикладываю:

```
%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

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