

# **Отчёт по лабораторной работе №5**

**Дисциплина: архитектура компьютера**

Логинов Георгий Евгеньевич

# 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 тс . . . . .                            | 9         |
| 4.2      | Структура программы на языке ассемблера NASM . . . . .  | 11        |
| 4.3      | Подключение внешнего файла . . . . .                    | 13        |
| 4.4      | Выполнение заданий для самостоятельной работы . . . . . | 17        |
| <b>5</b> | <b>Выводы</b>   | <b>22</b> |
|          | <b>Список литературы</b>                                | <b>23</b> |

## Список иллюстраций

|      |   |    |
|------|---|----|
| 4.1  | Открытый Midnight Commander . . . . .                       | 9  |
| 4.2  | Перемещение между директориями . . . . .                    | 10 |
| 4.3  | Создание каталога . . . . .                                 | 10 |
| 4.4  | Создание файла в новой директории . . . . .                 | 11 |
| 4.5  | Открытие файла для редактирования . . . . .                 | 11 |
| 4.6  | Редактирование файла . . . . .                              | 12 |
| 4.7  | Открытие файла для просмотра . . . . .                      | 12 |
| 4.8  | Компиляция файла и передача на обработку компоновщику . . . | 13 |
| 4.9  | Исполнение файла . . . . .                                  | 13 |
| 4.10 | Скачанный файл отображается в файловом менеджере . . . . .  | 14 |
| 4.11 | Копирование файла . . . . .                                 | 14 |
| 4.12 | Копирование файла с изменением его имени . . . . .          | 15 |
| 4.13 | Редактирование файла . . . . .                              | 15 |
| 4.14 | Исполнение файла . . . . .                                  | 16 |
| 4.15 | Редактирование файла . . . . .                              | 16 |
| 4.16 | Исполнение файла . . . . .                                  | 16 |
| 4.17 | Копирование файла с изменением его имени . . . . .          | 17 |
| 4.18 | Редактирование файла . . . . .                              | 18 |
| 4.19 | Исполнение файла . . . . .                                  | 18 |
| 4.20 | Копирование файла . . . . .                                 | 20 |
| 4.21 | Редактирование файла . . . . .                              | 20 |
| 4.22 | Копирование файла . . . . .                                 | 21 |

## 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 mc

Открываю Midnight Commander, введя в терминал mc (рис. 4.1).

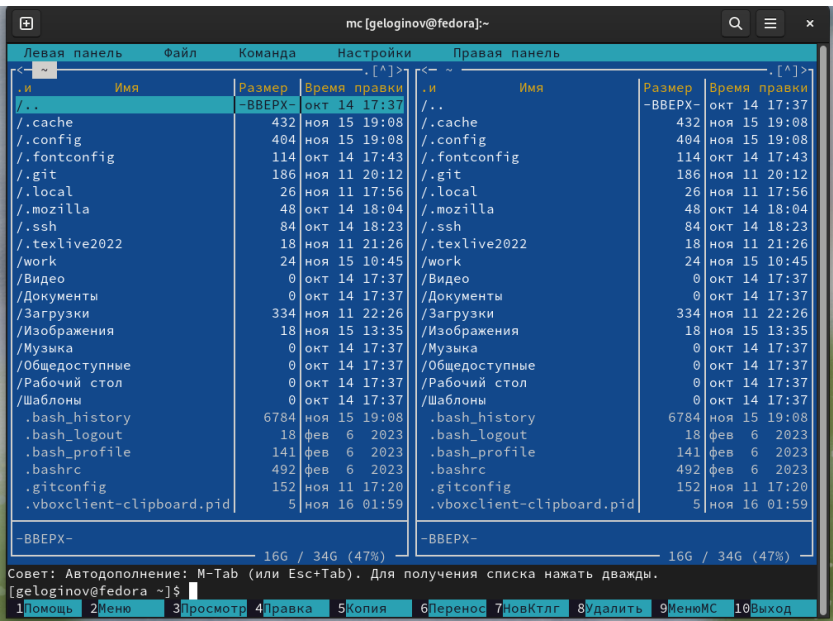


Рис. 4.1: Открытый Midnight Commander

Перехожу в каталог ~/work/arch-рс используя файловый менеджер mc (рис. 4.2).

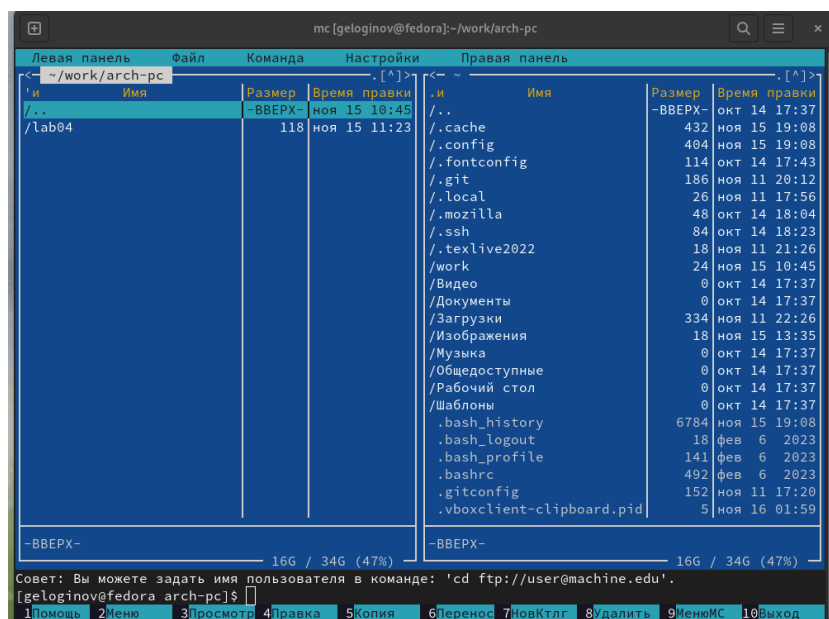


Рис. 4.2: Перемещение между директориями

С помощью функциональной клавиши F7 создаю каталог lab05 (рис. 4.3).

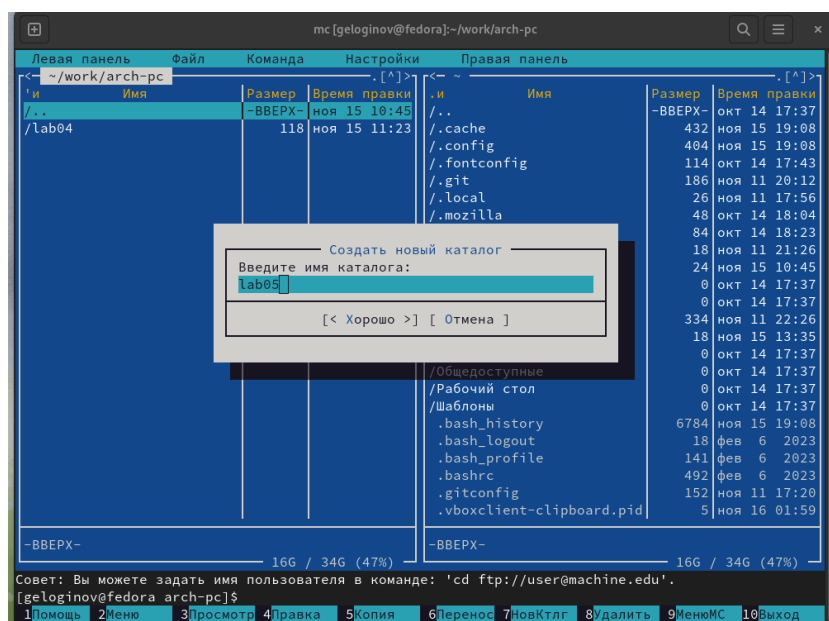


Рис. 4.3: Создание каталога

Перехожу в созданный каталог и прописываю команду touch lab5-1.asm в стро-

ке ввода, чтобы создать файл, в котором буду работать (рис. 4.4).



Рис. 4.4: Создание файла в новой директории

## 4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе mcedit (рис. 4.5).

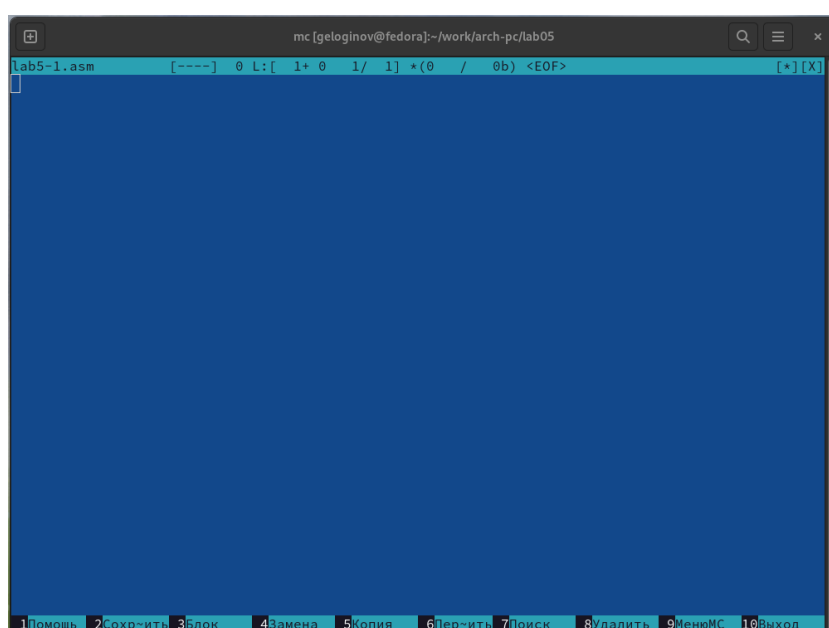
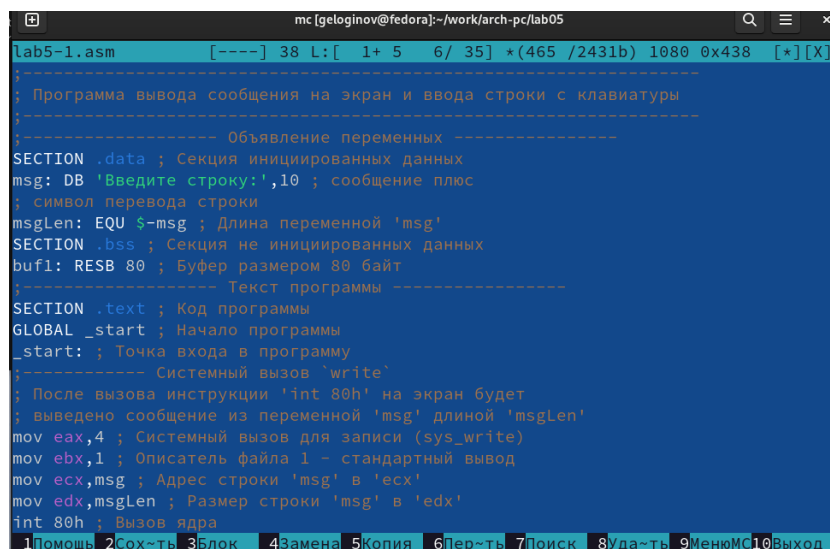


Рис. 4.5: Открытие файла для редактирования

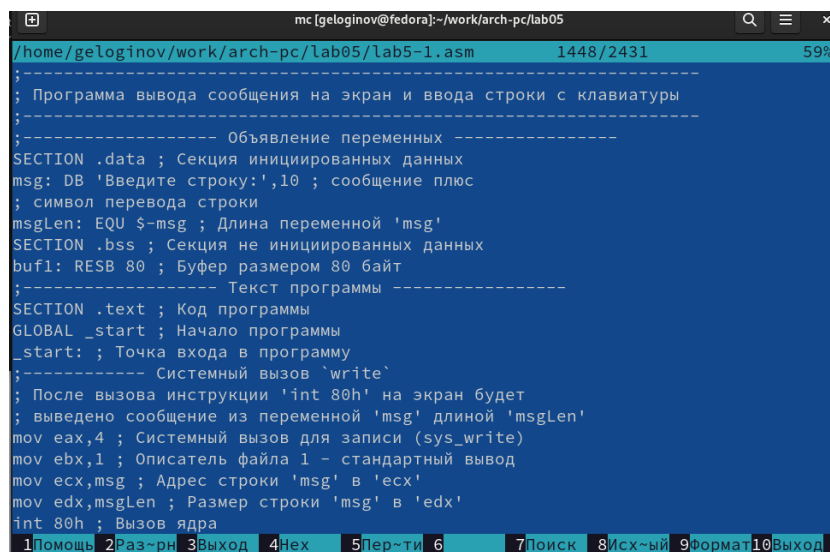
Ввожу в файл код программы для запроса строки у пользователя (рис. 4.6). Далее выхожу из файла (F10), сохраняя изменения (F2).



```
lab5-1.asm [----] 38 L: [ 1+ 5 6/ 35] *(465 /2431b) 1080 0x438 [*][X]
;
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;
;----- Объявление переменных -----
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициализированных 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
1Помощь 2Сох-ть 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход
```

Рис. 4.6: Редактирование файла

С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы (рис. 4.7).



```
/home/geloginov/work/arch-pc/lab05/lab5-1.asm 1448/2431 59%
;
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;
;----- Объявление переменных -----
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициализированных 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
1Помощь 2Раз-рн 3Выход 4Нех 5Пер-ти 6 7Поиск 8Исх-ый 9Формат10Выход
```

Рис. 4.7: Открытие файла для просмотра

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-1.asm`. Создался объектный файл `lab5-1.o`. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-1 lab5-1.o` создался испол-

няемый файл lab5-1. (рис. 4.8).

```
[geloginov@fedora lab05]$ nasm -f elf lab5-1.asm  
[geloginov@fedora lab05]$ ld -m elf_i386 -o lab5-1 lab5-1.o  
[geloginov@fedora lab05]$
```

Рис. 4.8: Компиляция файла и передача на обработку компоновщику

Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ожидает ввода с клавиатуры, я ввожу свои ФИО, на этом программа завершает свою работу (рис. 4.9).

```
[geloginov@fedora lab05]$ ./lab5-1  
Введите строку:  
Логинев Георгий Евгеньевич
```

Рис. 4.9: Исполнение файла

### 4.3 Подключение внешнего файла

Скачиваю файл in\_out.asm со страницы курса в ТУИС. Он сохранился в каталог “Загрузки” (рис. 4.10).

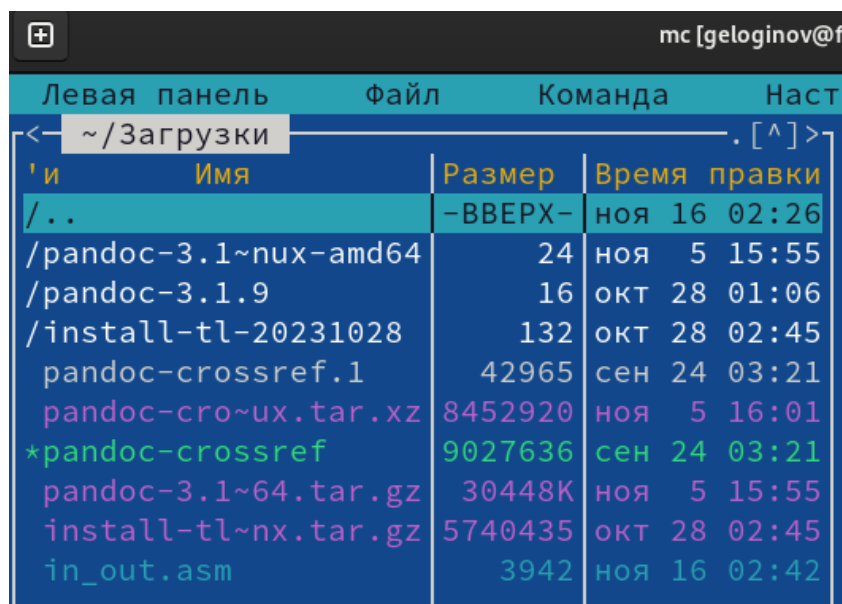


Рис. 4.10: Скачанный файл отображается в файловом менеджере

С помощью функциональной клавиши F5 копирую файл in\_out.asm из каталога Загрузки в созданный каталог lab05 (рис. 4.11).

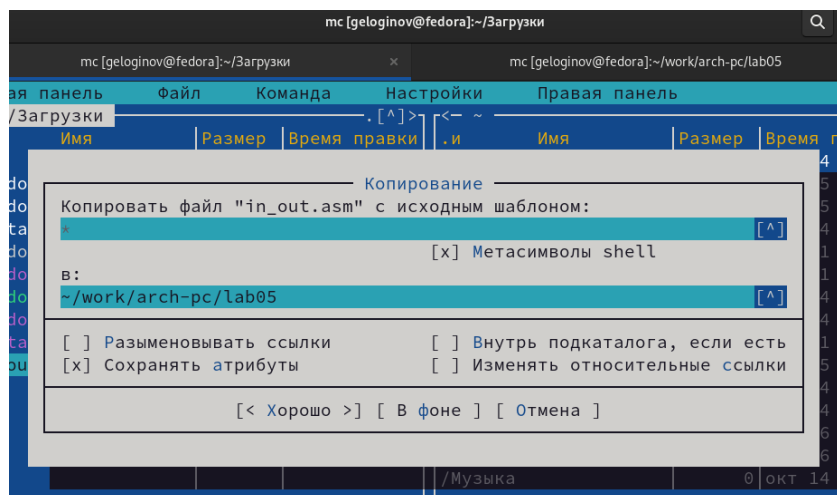


Рис. 4.11: Копирование файла

С помощью функциональной клавиши F5 копирую файл lab5-1 в тот же каталог, но с другим именем, для этого в появившемся окне mc прописываю имя для копии файла(рис. 4.12).

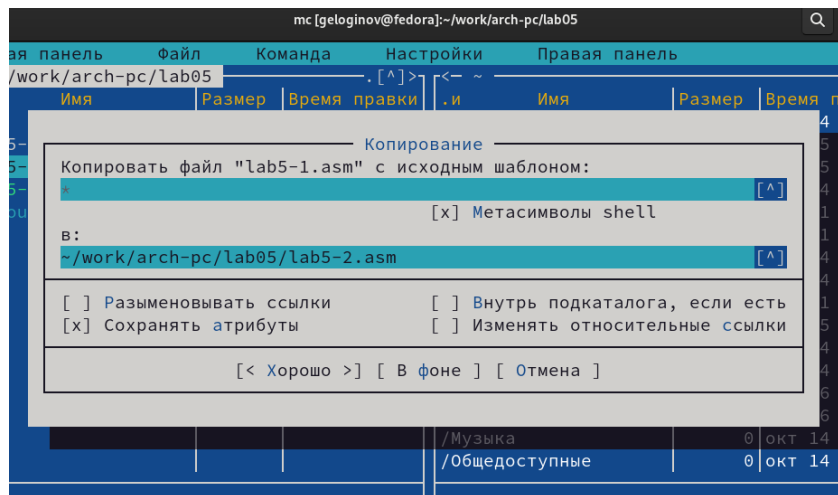


Рис. 4.12: Копирование файла с изменением его имени

Изменяю содержимое файла lab5-2.asm во встроенном редакторе mcedit (рис. 4.13), чтобы в программе использовались подпрограммы из внешнего файла in\_out.asm.

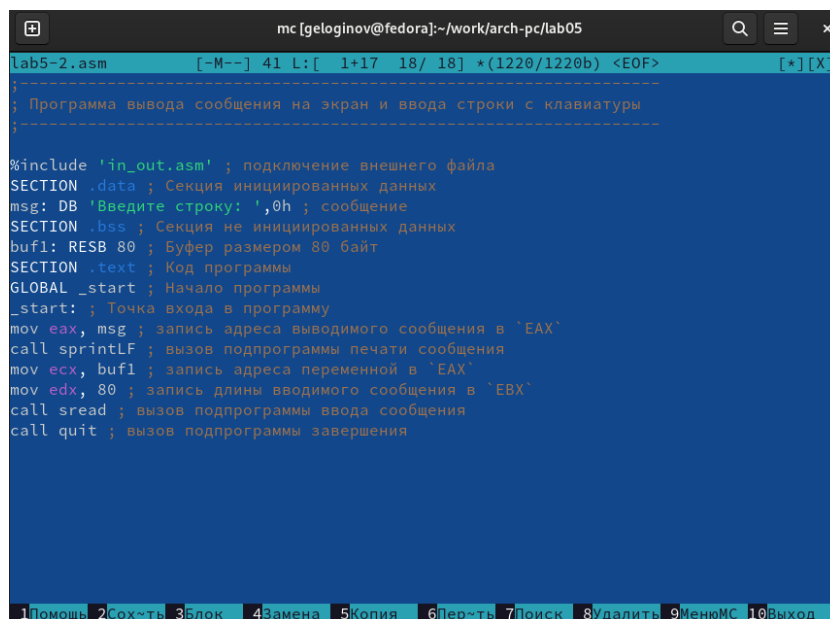


Рис. 4.13: Редактирование файла

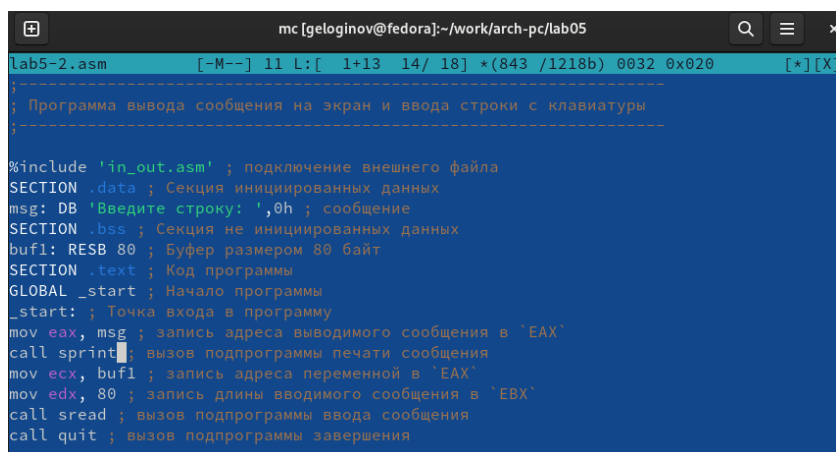
Транслирую текст программы файла в объектный файл командой `nasm -f elf`

lab5-2.asm. Создался объектный файл lab5-2.o. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-2 lab5-2.o` Создался исполняемый файл lab5-2. Запускаю исполняемый файл (рис. 4.14).

```
[geloginov@fedora lab05]$ nasm -f elf lab5-2.asm
[geloginov@fedora lab05]$ ld -m elf_i386 -o lab5-2 lab5-2.o
[geloginov@fedora lab05]$ ./lab5-2
Введите строку:
Логинов Георгий Евгеньевич
```

Рис. 4.14: Исполнение файла

Открываю файл lab5-2.asm для редактирования в mcedit функциональной клавишей F4. Изменяю в нем подпрограмму `sprintLF` на `sprint`. (рис. 4.15).



```
lab5-2.asm  [-M--] 11 L:[ 1+13 14/ 18] *(843 /1218b) 0032 0x020  [*] [X]
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----

%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 sprintLF ; 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. 4.15: Редактирование файла

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл (рис. 4.16).

```
[geloginov@fedora lab05]$ nasm -f elf lab5-2.asm
[geloginov@fedora lab05]$ ld -m elf_i386 -o lab5-2-2 lab5-2.o
[geloginov@fedora lab05]$ ./lab5-2-2
Введите строку: Логинов Георгий Евгеньевич
```

Рис. 4.16: Исполнение файла

Разница между первым исполняемым файлом lab5-2 и вторым lab5-2-2 в том,



что запуск первого запрашивает ввод с новой строки, а программа, которая исполняется при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами `sprintLF` и `sprint`.

## 4.4 Выполнение заданий для самостоятельной работы

1. Создаю копию файла `lab5-1.asm` с именем `lab5-1-1.asm` с помощью функциональной клавиши F5 (рис. 4.17).

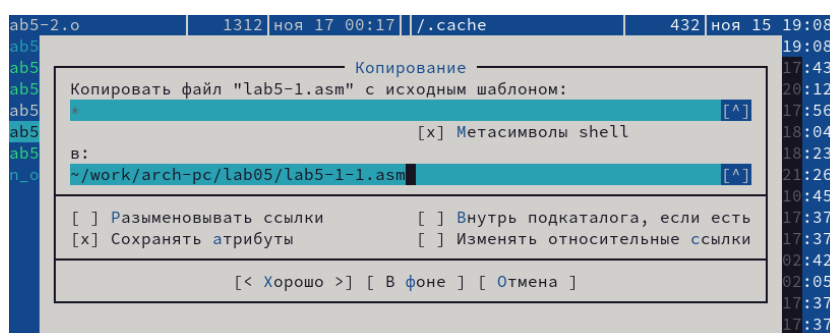
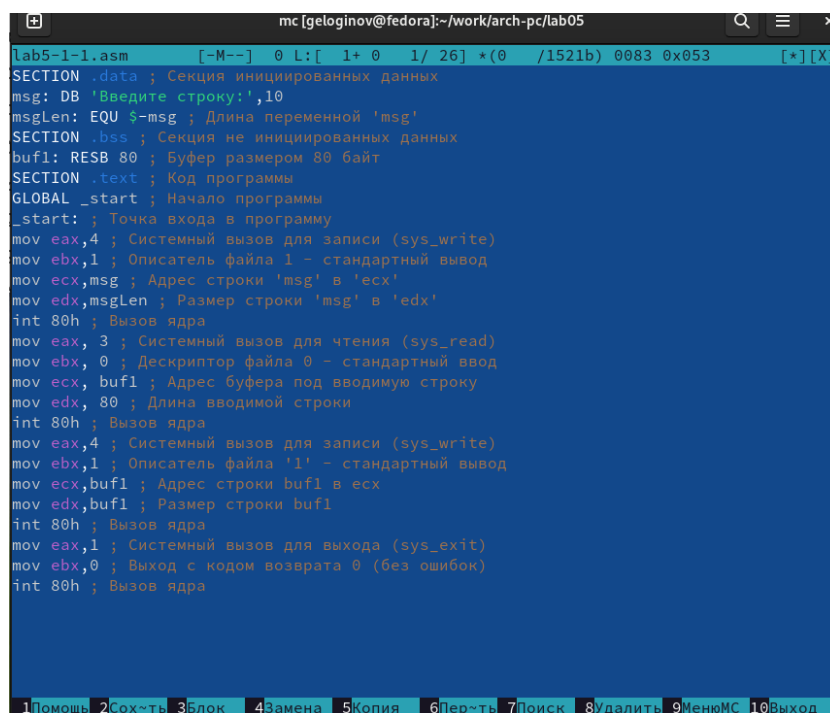


Рис. 4.17: Копирование файла с изменением его имени

С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. 4.18).

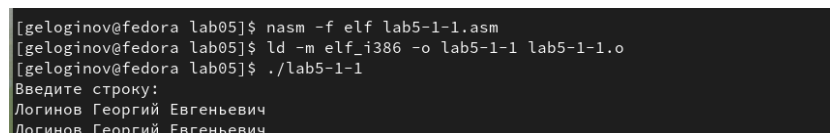


```
lab5-1-1.asm [-M--] 0 L:[ 1+ 0 1/ 26] *(0 /1521b) 0083 0x053 [*][X]
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не иницированных данных
buf1: RESB 80 ; Буфер размером 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
mov eax,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,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
mov ecx,buf1 ; Адрес строки buf1 в ecx
mov edx,buf1 ; Размер строки buf1
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

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9МенюМС 10Выход
```

Рис. 4.18: Редактирование файла

2. Создаю объектный файл lab5-1-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-1-1, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свои ФИО, далее программа выводит введенные мною данные(рис. 4.19).



```
[geloginov@fedora lab05]$ nasm -f elf lab5-1-1.asm
[geloginov@fedora lab05]$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
[geloginov@fedora lab05]$ ./lab5-1-1
Введите строку:
Логинов Георгий Евгеньевич
Логинов Георгий Евгеньевич
```

Рис. 4.19: Исполнение файла

Код программы из первого пункта:

```
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10
msgLen: EQU $-msg ; Длина переменной 'msg'
```

```

SECTION .bss ; Секция не инициированных данных
buf1: RESB 80 ; Буфер размером 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
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,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
mov ecx,buf1 ; Адрес строки buf1 в ecx
mov edx,buf1 ; Размер строки buf1
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

```

3. Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5(рис. 4.20).

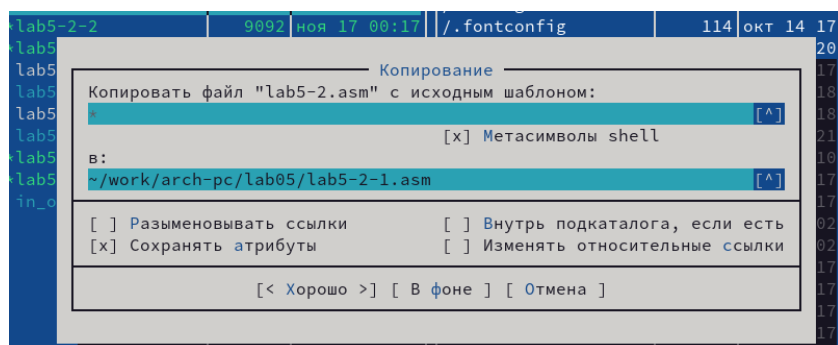


Рис. 4.20: Копирование файла

С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. 4.21).

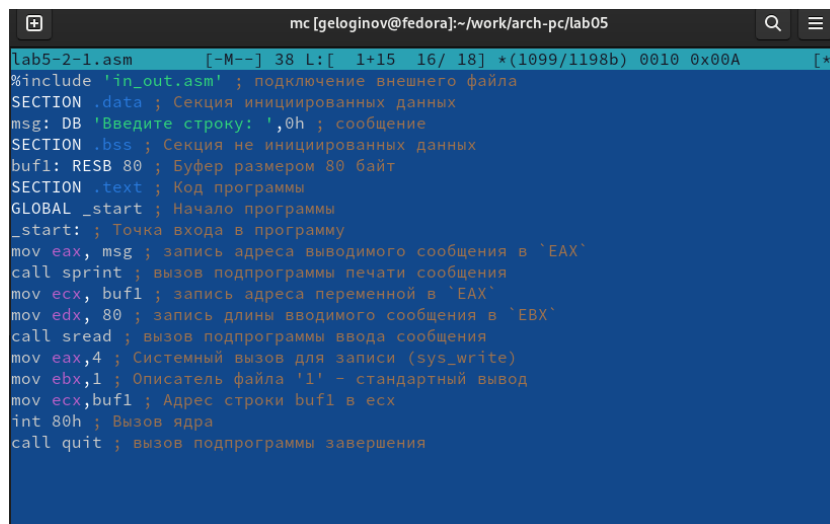


Рис. 4.21: Редактирование файла

4. Создаю объектный файл lab5-2-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-2-1, запускаю полученный исполняемый файл. Программа запрашивает ввод без переноса на новую строку, ввожу свои ФИО, далее программа выводит введенные мною данные (рис. 4.22).

```
[geloginov@fedora lab05]$ nasm -f elf lab5-2-1.asm
[geloginov@fedora lab05]$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
[geloginov@fedora lab05]$ ./lab5-2-1
Введите строку: Логинов Георгий Евгеньевич
Логинов Георгий Евгеньевич
```

Рис. 4.22: Копирование файла

Код программы из третьего пункта:

```
%include 'in_out.asm'

SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку: ',0h ; сообщение

SECTION .bss ; Секция не инициированных 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 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
mov ecx,buf1 ; Адрес строки buf1 в ecx
int 80h ; Вызов ядра
call quit ; вызов подпрограммы завершения
```

## 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. Лабораторная работа №5 ::: {#refs} :::