

# **Лабораторная работа №5**

**Дисциплина: Архитектура компьютера**

Малюга Валерия Васильевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
3.1	Основы работы с Midnight Commander . . . . .	6
3.2	Структура программы на языке ассемблера NASM . . . . .	7
3.3	Описание инструкции mov . . . . .	8
3.4	Описание инструкций int . . . . .	8
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	Основы работы с Midnight Commander . . . . .	9
4.2	Подключение внешнего файла in_out.asm . . . . .	12
4.3	Задание для самостоятельной работы . . . . .	14
<b>5</b>	<b>Выводы</b>	<b>18</b>

## Список иллюстраций

4.1	Создание папки lab05 . . . . .	9
4.2	Создание файла lab5-1.asm . . . . .	10
4.3	Ввод текста программы из листинга . . . . .	10
4.4	Проверка содержимого программы . . . . .	11
4.5	Трансляция, компоновка объектного файла и запуск программы .	11
4.6	Копирование файла . . . . .	12
4.7	Копирование файла . . . . .	12
4.8	Редактирование файла . . . . .	13
4.9	Исполнение файла . . . . .	13
4.10	Отредактированный файл . . . . .	14
4.11	Вывод программ . . . . .	14
4.12	Копирование файла . . . . .	15
4.13	Редактирование файла . . . . .	15
4.14	Исполнение файла . . . . .	16
4.15	Копирование файла . . . . .	16
4.16	Редактирование файла . . . . .	17
4.17	Исполнение файла . . . . .	17

# 1 Цель работы

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

### 3.1 Основы работы с Midnight Commander

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. В Midnight Commander используются функциональные клавиши *F1* — *F10*, к которым привязаны часто выполняемые операции (табл. 3.1)

Таблица 3.1: Функциональные клавиши Midnight Commander

Функциональные клавиши	Выполняемое действие
F1	вызов контекстно-зависимой подсказки
F2	вызов меню, созданного пользователем
F3	просмотр файла, на который указывает подсветка в активной панели
F4	вызов встроенного редактора для файла, на который указывает подсветка в активной панели
F5	копирование файла (группы файлов из каталога), отображаемого в активной панели, в каталог, отображаемый на второй панели

Функциональные клавиши	Выполняемое действие
F6	перенос файла (группы файлов из каталога), отображаемого в активной панели, в каталог, отображаемый на второй панели
F7	создание подкаталога в каталоге, отображаемом в активной панели
F8	удаление файла (подкаталога) или группы отмеченных файлов
F9	вызов основного меню программы
F10	выход из программы

## 3.2 Структура программы на языке ассемблера NASM

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 размеров 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 (учетверённое 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. Для определения строк принято использовать директиву DB в связи с особенностями хранения данных в оперативной памяти.

### 3.3 Описание инструкции mov

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).

### 3.4 Описание инструкций int

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

Открыла Midnight Commander с помощью команды `mc`. Пользуясь клавишами `⌘`, `⌘` и `Enter` перешла в каталог `~/work/arch-рс`, созданный при выполнении лабораторной работы №4. С помощью функциональной клавиши `F7` создала папку `lab05` (рис. 4.1). Перешла в созданный каталог и, пользуясь строкой ввода и командой `touch`, создала файл `lab5-1.asm` (рис. 4.2)

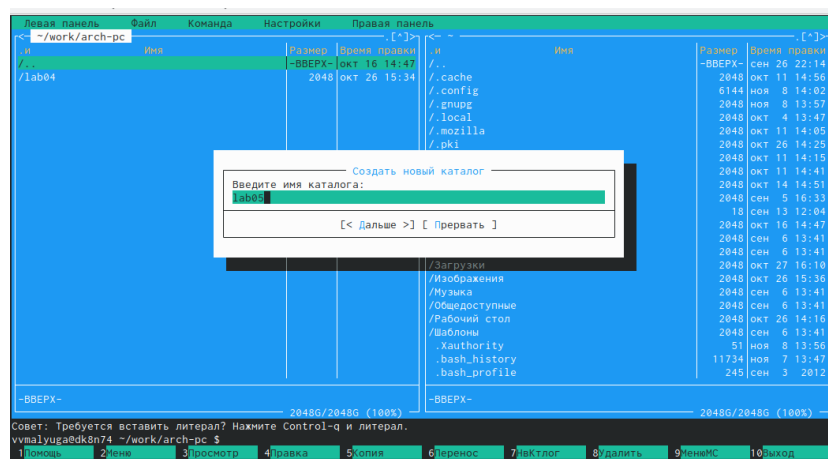


Рис. 4.1: Создание папки lab05

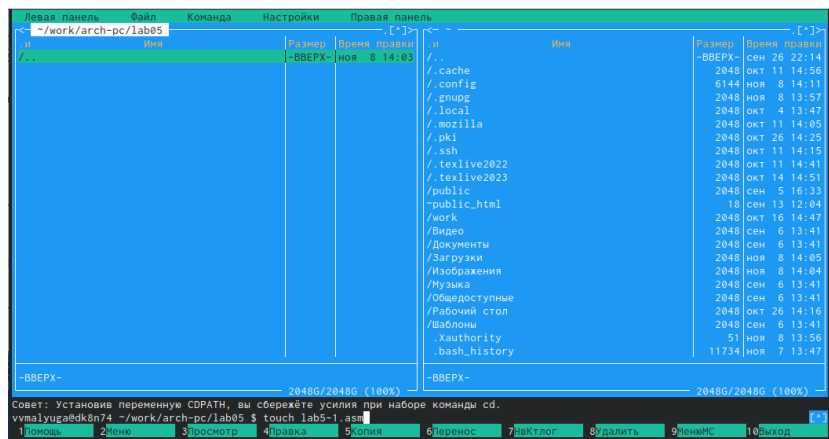


Рис. 4.2: Создание файла lab5-1.asm

С помощью функциональной клавиши F4 откройте файл lab5-1.asm для редактирования во встроенном редакторе. Введите текст программы из листинга 5.1, сохранила изменения и закрыла файл (рис. 4.3). С помощью функциональной клавиши F3 открыла файл lab5-1.asm для просмотра. Убедилась, что файл содержит текст программы (рис. 4.4).

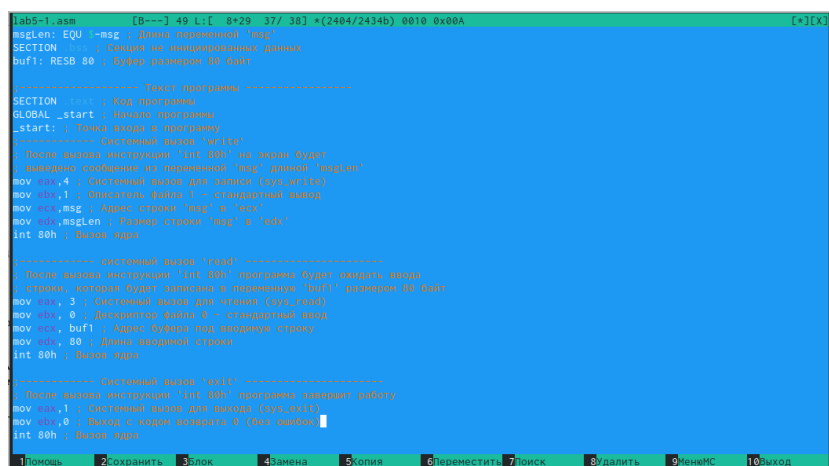


Рис. 4.3: Ввод текста программы из листинга

```

/home/vvmalyuga/work/arch-pc/lab05/lab5-1.asm
-----
; 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 ; Описание 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 ; Дескриптор 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

```

Рис. 4.4: Проверка содержимого программы

Оттранслировала текст программы lab5-1.asm в объектный файл. Выполнила компоновку объектного файла и запустила получившийся исполняемый файл. Программа выводит строку 'Введите строку:' и ожидает ввода с клавиатуры. На запрос ввела свои ФИО (рис. 4.5).

```

vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $ ls
lab5-1.asm
vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1.asm
vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $ ls
lab5-1.asm lab5-1.o
vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $ ld -m elf_i386 lab5-1.o -o lab5-1
vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $ ls
lab5-1 lab5-1.asm lab5-1.o
vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:
Малюга Валерия Васильевна
vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $

```

Рис. 4.5: Трансляция, компоновка объектного файла и запуск программы

## 4.2 Подключение внешнего файла in\_out.asm

Скачала файл in\_out.asm со страницы курса в ТУИСе. С помощью функциональной клавиши F5 скопировала файл in\_out.asm из каталога Загрузки в созданный каталог lab05 (рис. 4.6).

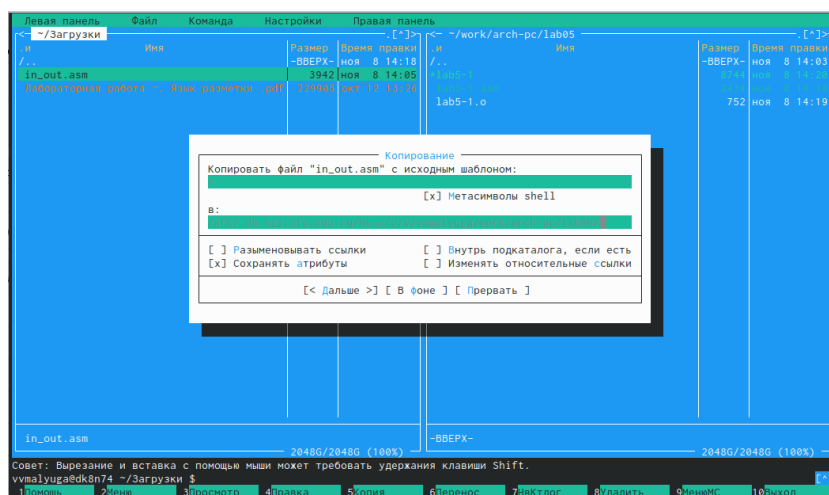


Рис. 4.6: Копирование файла

С помощью функциональной клавиши F5 скопировала файл lab5-1 в тот же каталог, но с другим именем, для этого в появившемся окне прописала имя для копии файла (рис. 4.7).

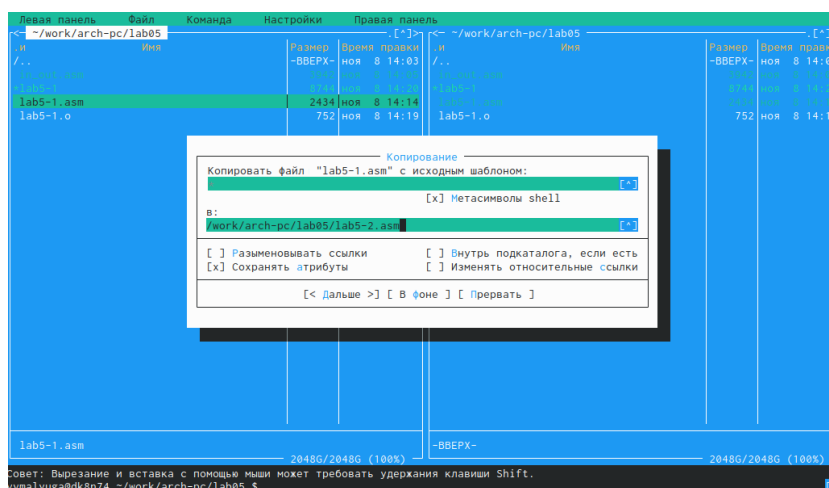
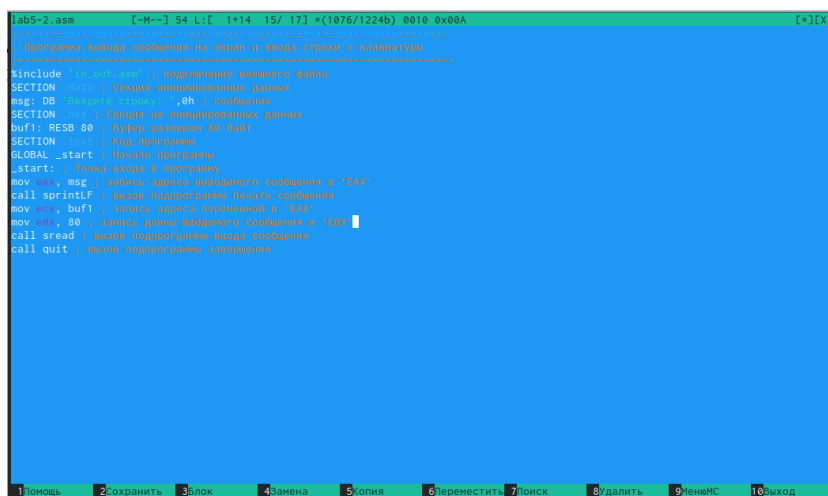


Рис. 4.7: Копирование файла

Изменила содержимое файла lab5-2.asm во встроенном редакторе (рис. 4.8), чтобы в программе использовались подпрограммы из внешнего файла in\_out.asm.

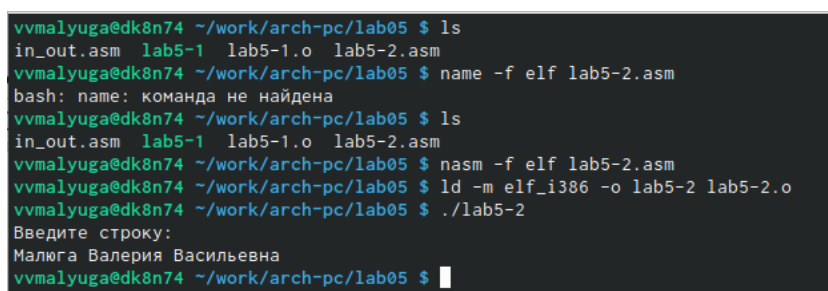


```

lab5-2.asm [M--] 54 L: [I+14 15/ 17] *(1076/1224b) 0010 0x00A [*][X]
; Программа ввода сообщения на экран и ввода строки с клавиатуры
;-----
%include "in_out.asm" ; подключение внешнего файла
SECTION .text ; Секция инициализированных данных
msg: DB "Введите сообщение: ", 0h ; сообщение
SECTION .data ; Секция не инициализированных данных
buf1: RESB 80 ; буфер размером 80 байт
SECTION .code ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
mov eax, msg ; записываем адрес входного сообщения в 'EAX'
call sprintf ; вызов подпрограммы печати сообщения
mov ebx, buf1 ; записываем адрес переменной в 'EBX'
mov edx, 80 ; записываем длину входного сообщения в 'EDX'
call read ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения
  
```

Рис. 4.8: Редактирование файла

Оттранслировала текст программы файла в объектный файл командой `nasm -f elf lab5-2.asm`. Создался объектный файл lab5-2.o. Выполнила компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-2 lab5-2.o` Создался исполняемый файл lab5-2. Запустила исполняемый файл (рис. [4.9]).



```

vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $ ls
in_out.asm  lab5-1  lab5-1.o  lab5-2.asm
vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $ name -f elf lab5-2.asm
bash: name: команда не найдена
vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $ ls
in_out.asm  lab5-1  lab5-1.o  lab5-2.asm
vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку:
Малюга Валерия Васильевна
vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $
  
```

Рис. 4.9: Исполнение файла

Открыла файл lab5-2.asm для редактирования функциональной клавишей F4. Изменила в нем подпрограмму `sprintf` на `sprint`. Сохранила изменения и открыла файл для просмотра, чтобы проверить сохранение действий (рис. 4.10).

```

lab5-2.asm      [BM--] 10 L: [ 1+12 13/ 17] *(846 /1222b) 0116 0x074
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
#include "io.h" ; 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

```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Переместить 7Поиск 8Удалить

Рис. 4.10: Отредактированный файл

Разница между первым исполняемым файлом lab5-1 и вторым lab5-2 в том, что запуск первого запрашивает ввод с новой строки, а программа, которая выполняется при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами `sprintf` и `sprint` (рис. 4.11).

```

vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку: Малья Валерия Васильевна
vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:
Малья Валерия Васильевна
vvmalyuga@dk8n74 ~/work/arch-pc/lab05 $

```

Рис. 4.11: Вывод программ

## 4.3 Задание для самостоятельной работы

1. Создала копию файла lab5-1.asm с именем lab5-1-1.asm с помощью функциональной клавиши F5 (рис. [4.12]).

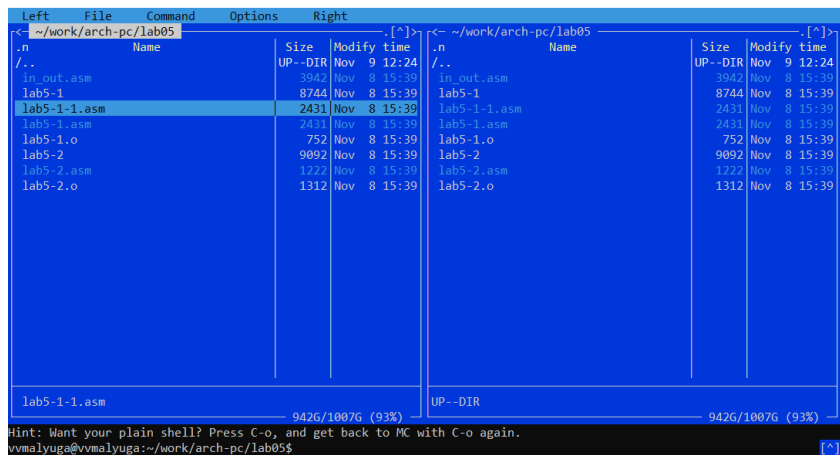


Рис. 4.12: Копирование файла

С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменила программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. [4.13]).

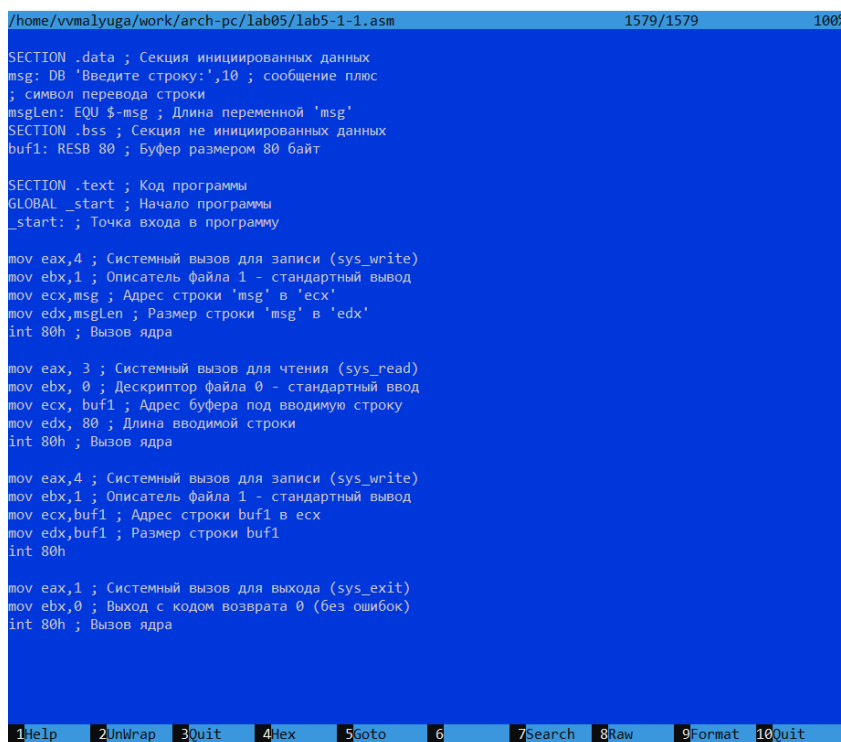


Рис. 4.13: Редактирование файла

2. Создала объектный файл lab5-1-1.o, отдала его на обработку компоновщику, получила исполняемый файл lab5-1-1. Запустила полученный исполняемый файл. Программа запросила ввод, ввела свою фамилию, далее программа вывела введенную мной фамилию (рис. [4.14]).

```
vvmalyuga@vvmalyuga:~/work/arch-pc/lab05$ ls
in_out.asm lab5-1 lab5-1-1.asm lab5-1.o lab5-2 lab5-2.asm lab5-2.o
vvmalyuga@vvmalyuga:~/work/arch-pc/lab05$ nasm -f lab5-1-1.asm
nasm: fatal: unrecognised output format 'lab5-1-1.asm' - use -hf for a list
Type nasm -h for help.
vvmalyuga@vvmalyuga:~/work/arch-pc/lab05$ nasm -f elf lab5-1-1.asm
vvmalyuga@vvmalyuga:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
vvmalyuga@vvmalyuga:~/work/arch-pc/lab05$ ./lab5-1-1
Введите строку:
Малюга
Малюга
vvmalyuga@vvmalyuga:~/work/arch-pc/lab05$
```

Рис. 4.14: Исполнение файла

3. Создала копию файла lab5-2.asm с именем lab5-2-2.asm с помощью функциональной клавиши F5 (рис. [4.15]).

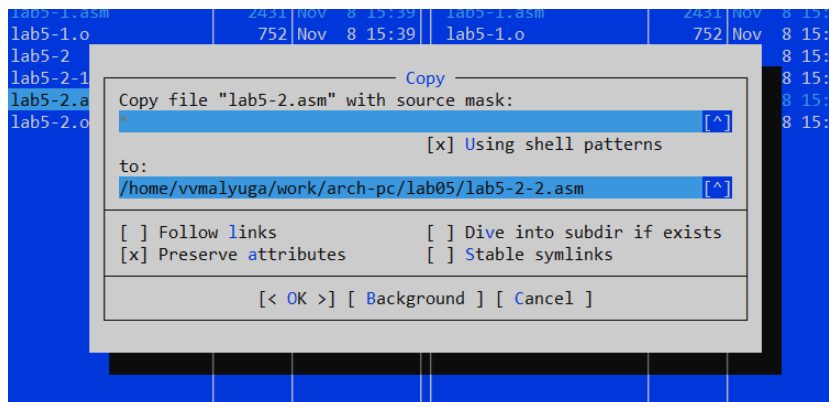


Рис. 4.15: Копирование файла

С помощью функциональной клавиши F4 открывала созданный файл для редактирования. Изменила программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. [4.16]).



```
/home/vvmalyuga/work/arch-pc/lab05/lab5-2-2.asm 1223/1223
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
%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 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
call quit ; вызов подпрограммы завершения
```

Рис. 4.16: Редактирование файла

4. Создала объектный файл lab5-2-2.o, отдала его на обработку компоновщику, получила исполняемый файл lab5-2-2. Запустила полученный исполняемый файл. Программа запросила ввод без переноса на новую строку, ввожу свою фамилию, далее программа вывела введеную мной фамилию (рис. [4.17]).

```
vvmalyuga@vvmalyuga:~/work/arch-pc/lab05$ nasm -f elf lab5-2-2.asm
vvmalyuga@vvmalyuga:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-2 lab5-2-2.o
vvmalyuga@vvmalyuga:~/work/arch-pc/lab05$ ./lab5-2-2
Введите строку: Малюга
```

Рис. 4.17: Исполнение файла

## 5 Выводы

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