

Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Шабалина Елизавета Андреевна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Основы работы с тс	8
4.2	Структура программы на языке ассемблера NASM	10
4.3	Подключение внешнего файла	13
4.4	Выполнение заданий для самостоятельной работы	15
5	Выводы	19

Список иллюстраций

4.1	Открытый тс	8
4.2	Перемещение между директориями	9
4.3	Создание каталога	9
4.4	Создание файла	10
4.5	Открытие файла для редактирования	11
4.6	Редактирование файла	11
4.7	Открытие файла для просмотра	12
4.8	Исполнение файла	13
4.9	Копирование файла	13
4.10	Копирование файла	14
4.11	Исполнение файла	15
4.12	Редактирование файла	16
4.13	Исполнение файла	16
4.14	Исполнение файла	16
4.15	Редактирование файла	17
4.16	Исполнение файла	17
4.17	Исполнение файла	18

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 (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 mc

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

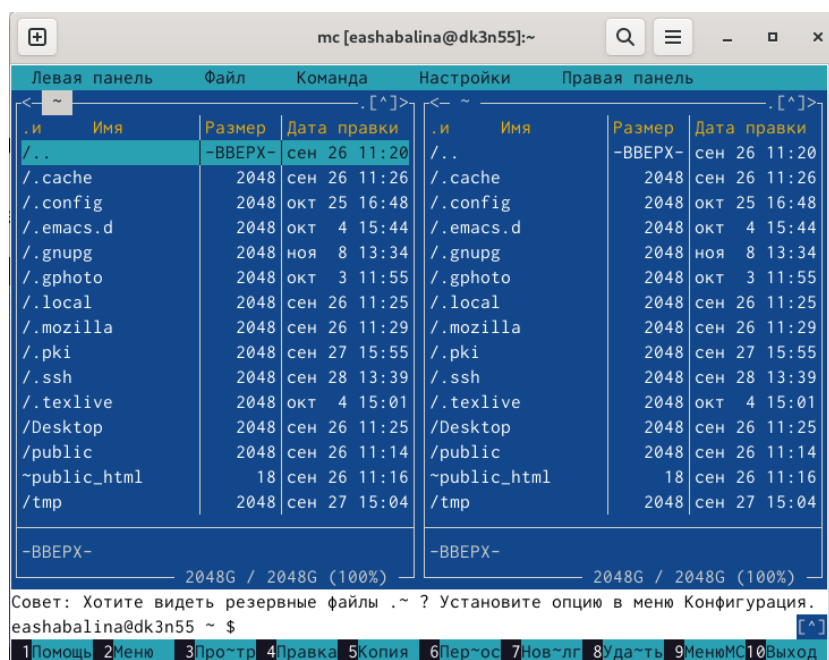


Рис. 4.1: Открытый mc

Перехожу в каталог ~/work/study/2023-2024/Архитектура Компьютера/arch-рс, используя файловый менеджер mc (рис. 4.2)

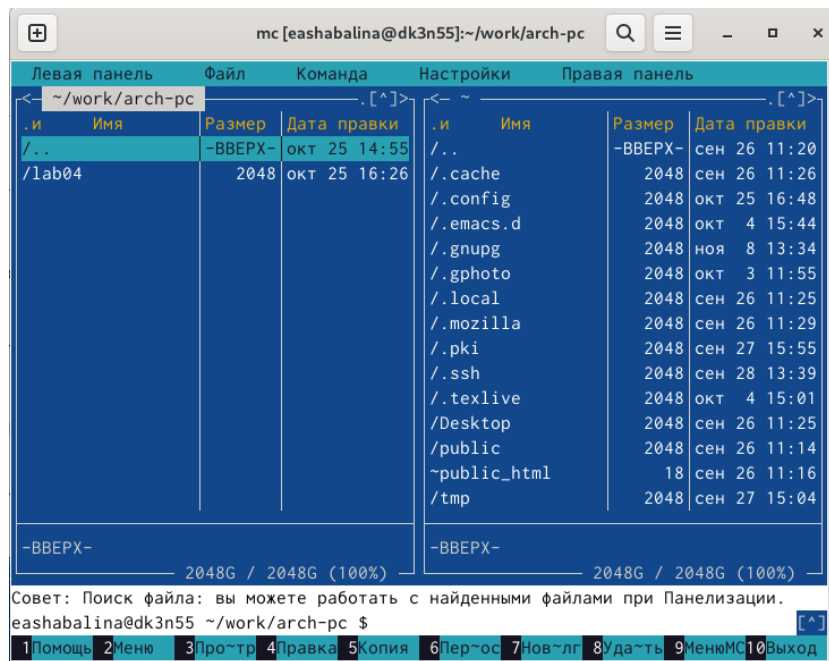


Рис. 4.2: Перемещение между директориями

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

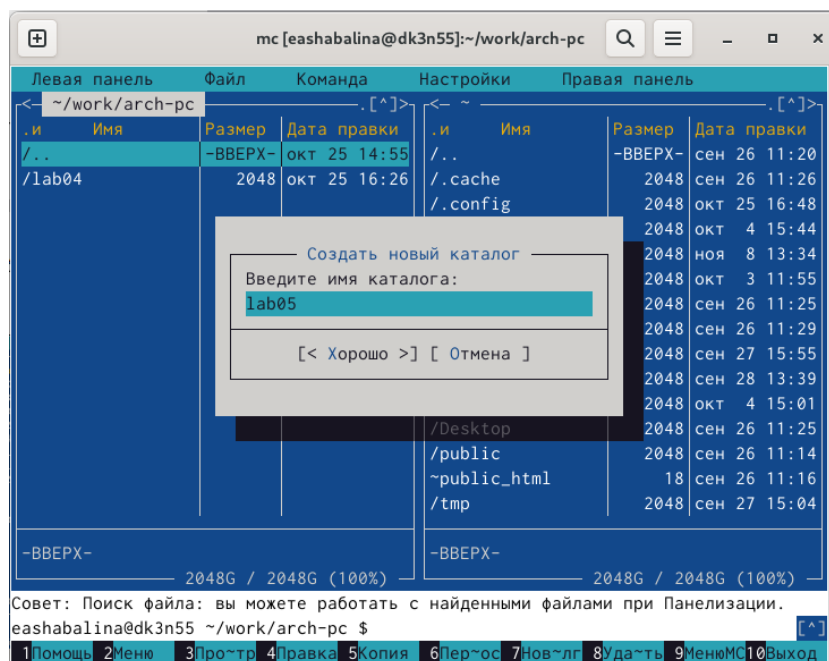


Рис. 4.3: Создание каталога

В строке ввода прописываю команду touch lab5-1.asm, чтобы создать файл, в

котором буду работать (рис. 4.4).

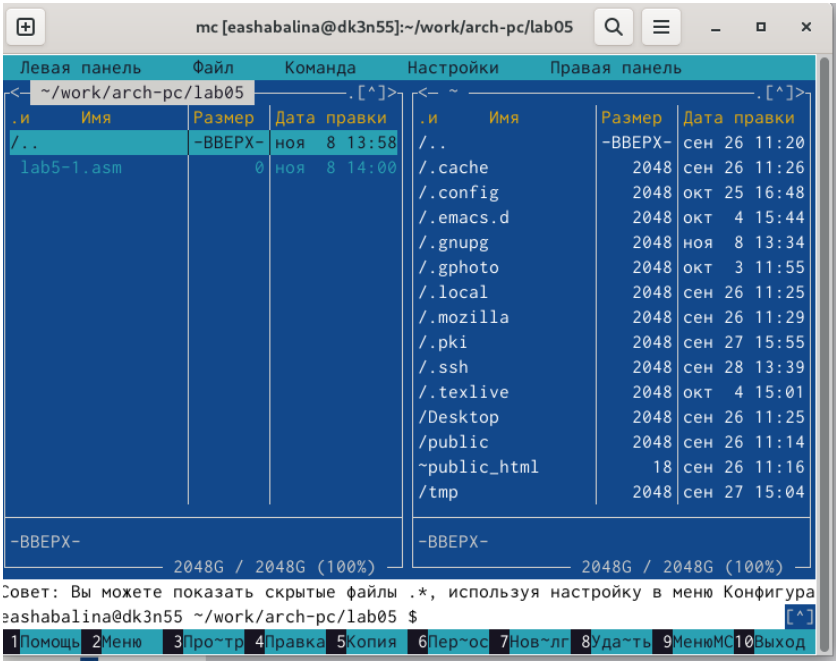


Рис. 4.4: Создание файла

4.2 Структура программы на языке ассемблера NASM

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

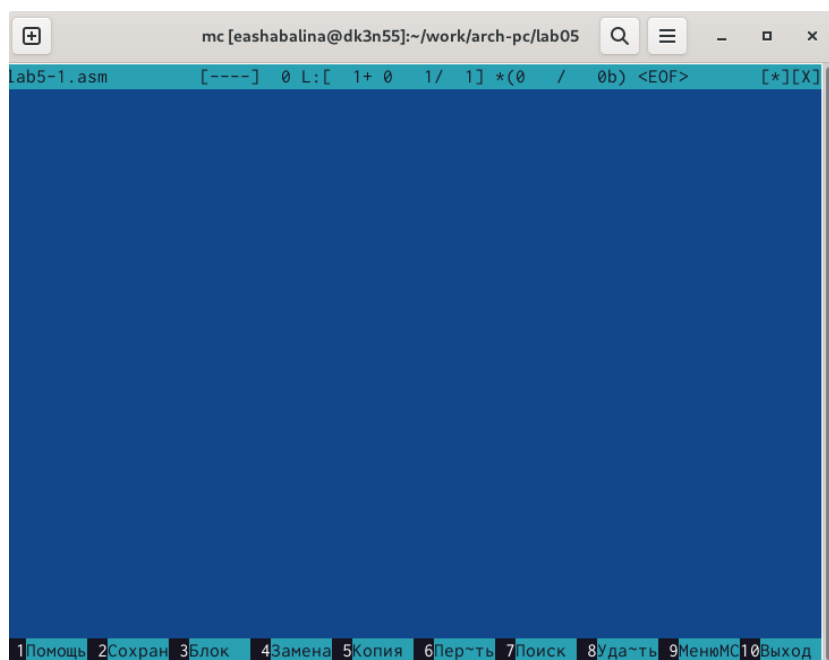


Рис. 4.5: Открытие файла для редактирования

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

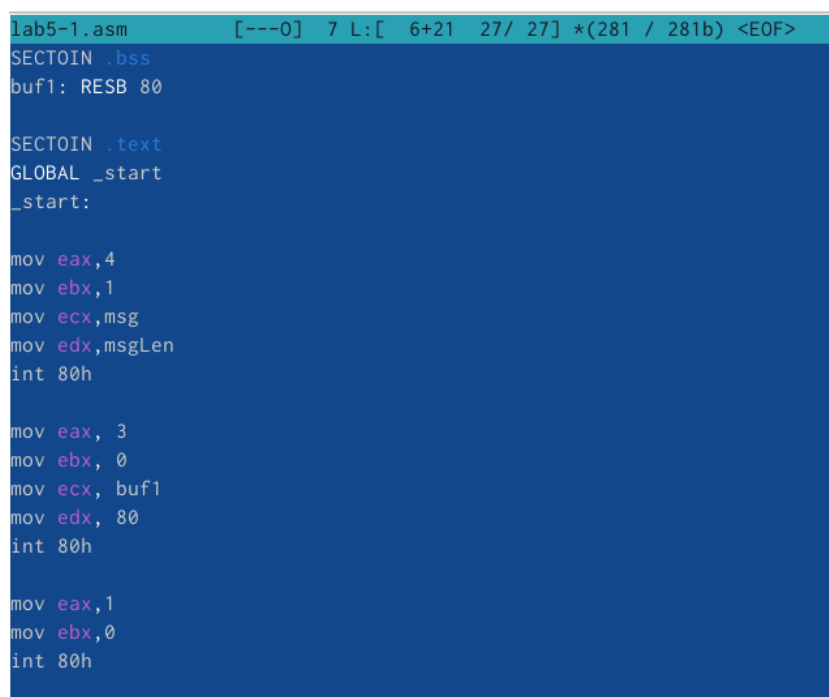
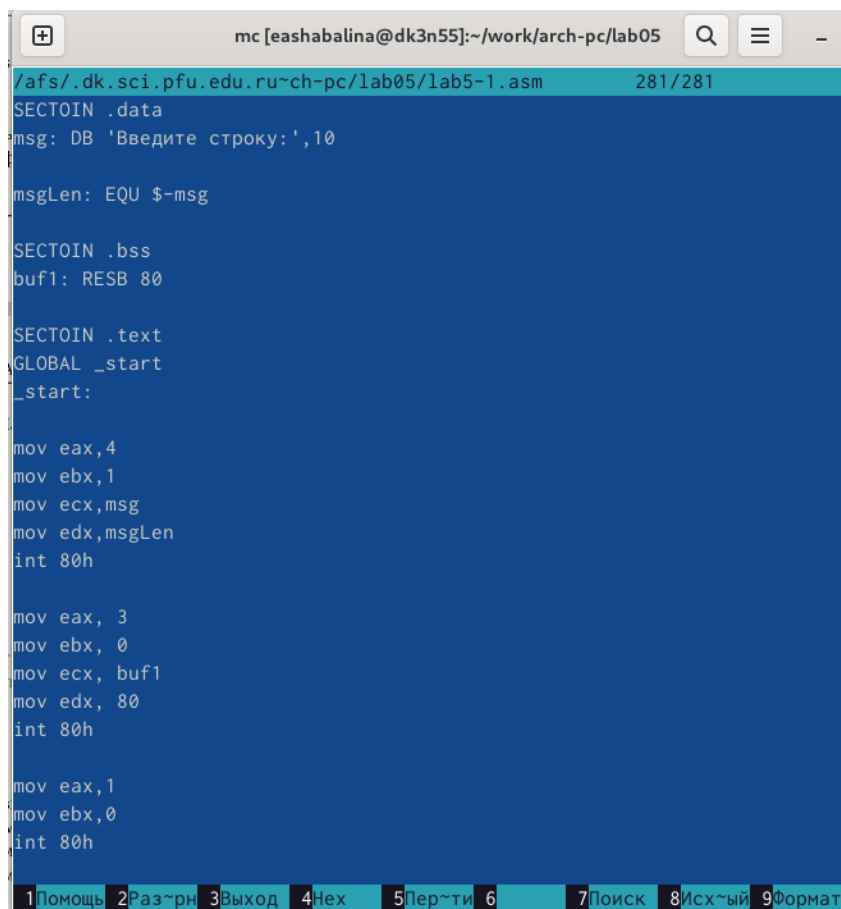


Рис. 4.6: Редактирование файла

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

The image shows a window titled 'mc [eashabalina@dk3n55]:~/work/arch-pc/lab05'. The file being viewed is '/afs/.dk.sci.pfu.edu.ru~ch-pc/lab05/lab5-1.asm' at line 281/281. The code is as follows:

```
SECT0IN .data
msg: DB 'Введите строку:',10

msgLen: EQU $-msg

SECT0IN .bss
buf1: RESB 80

SECT0IN .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,1
mov ebx,0
int 80h
```

At the bottom, there is a menu bar with options: 1Помощь, 2Разрешения, 3Выход, 4Нех, 5Перейти, 6, 7Поиск, 8Исходный, 9Формат.

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 исполняемый файл `lab5-1`.

Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу (рис. 4.8).

```
eashabalina@dk3n55 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:
Шабалина Елизавета Андреевна
```

Рис. 4.8: Исполнение файла

4.3 Подключение внешнего файла

Скачиваю файл `in_out.asm` со страницы курса в ТУИС. Он сохранился в каталог “Загрузки”.

С помощью функциональной клавиши F5 копирую файл `in_out.asm` из каталога Загрузки в созданный каталог `lab05` (рис. 4.9).

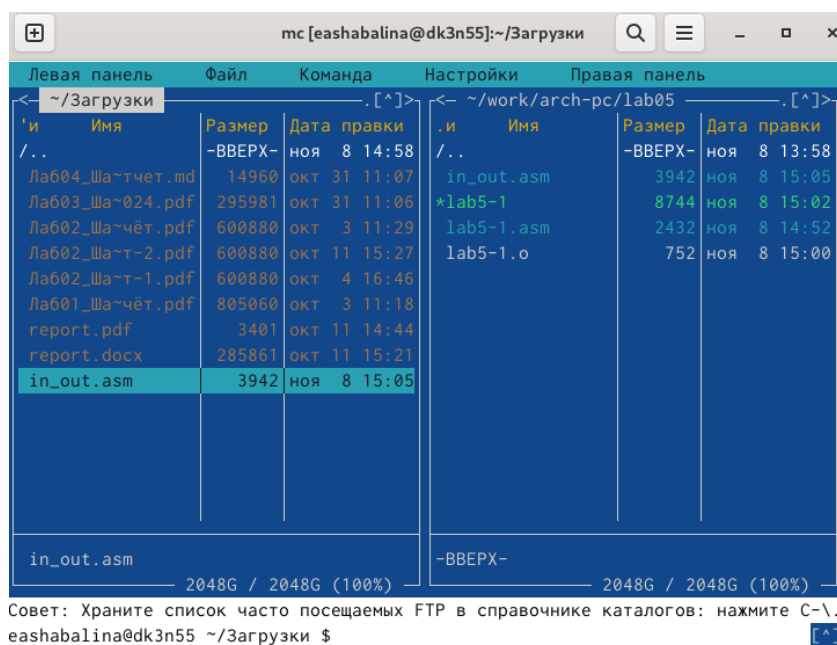


Рис. 4.9: Копирование файла

С помощью функциональной клавиши F5 копирую файл `lab5-1` в тот же каталог, но с другим именем (`lab5-2`), для этого в появившемся окне `mc` прописываю имя для копии файла (рис. 4.10).

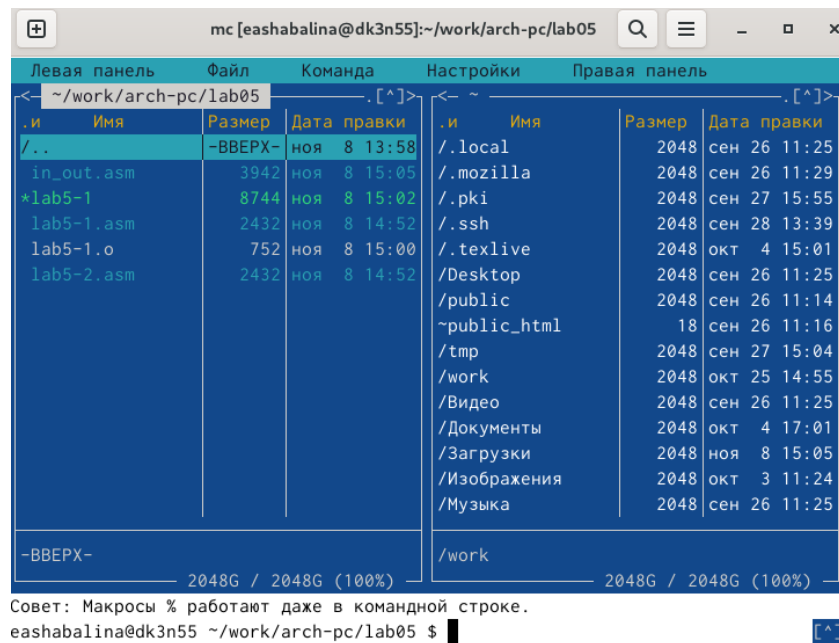


Рис. 4.10: Копирование файла

Изменяю содержимое файла lab5-2.asm в редакторе, чтобы в программе использовались подпрограммы из внешнего файла in_out.asm.

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. 4.11).

Открываю файл lab5-2.asm для редактирования функциональной клавишей F4. Изменяю в нем подпрограмму `sprintLF` на `sprint`. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий.

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл.

```

eashabalina@dk3n55 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm

eashabalina@dk3n55 ~/work/arch-pc/lab05 $ mc

eashabalina@dk3n55 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o

eashabalina@dk3n55 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку:
Шабалина Елизавета Андреевна

eashabalina@dk3n55 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm

eashabalina@dk3n55 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o

eashabalina@dk3n55 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку: Шабалина Елизавета Андреевна

```

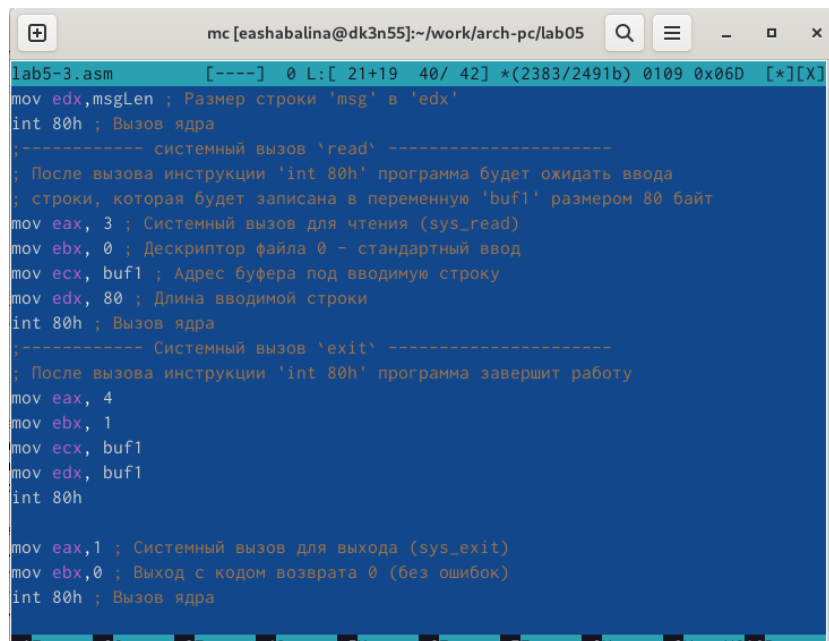
Рис. 4.11: Исполнение файла

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 lab5-3.asm с помощью функциональной клавиши F5.

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. 4.12).



```
lab5-3.asm [----] 0 L: [ 21+19 40/ 42] *(2383/2491b) 0109 0x06D [*][X]
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, 4
mov ebx, 1
mov ecx, buf1
mov edx, buf1
int 80h

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

Рис. 4.12: Редактирование файла

2. Создаю объектный файл lab5-3.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-3, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свою фамилию, далее программа выводит введенные мною данные (рис. 4.13) (рис. 4.14).

```
eashabalina@dk3n55 ~/work/arch-pc/lab05 $ nasm -f elf lab5-3.asm
eashabalina@dk3n55 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-3 lab5-3.o
eashabalina@dk3n55 ~/work/arch-pc/lab05 $ ./lab5-3.asm
```

Рис. 4.13: Исполнение файла

Введите строку:

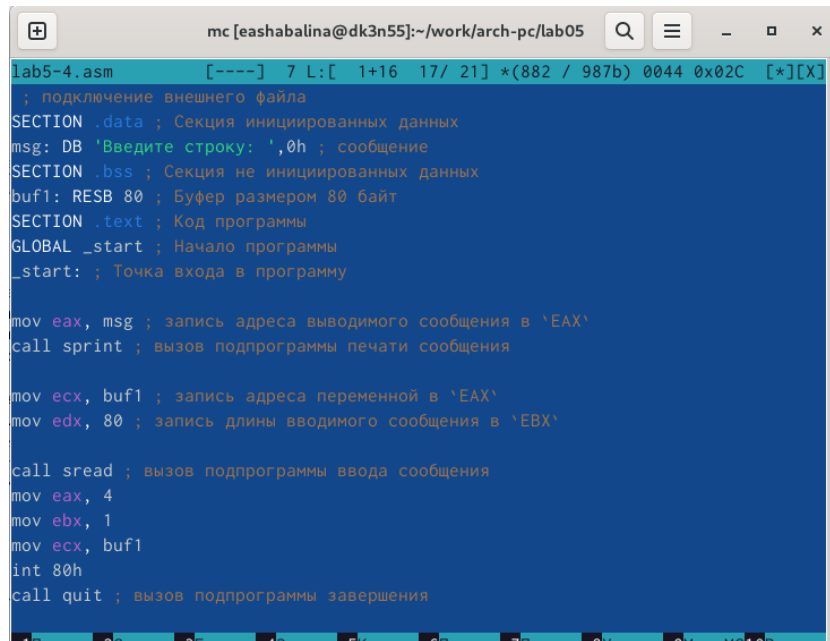
Шабалина

Шабалина

Рис. 4.14: Исполнение файла

3. Создаю копию файла lab5-2.asm с именем lab5-4.asm с помощью функциональной клавиши F5.

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. 4.15).



```
lab5-4.asm [----] 7 L: [ 1+16 17/ 21] *(882 / 987b) 0044 0x02C [*][X]
; 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
```

Рис. 4.15: Редактирование файла

4. Создаю объектный файл lab5-4.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-4, запускаю полученный исполняемый файл. Программа запрашивает ввод без переноса на новую строку, ввожу свою фамилию, далее программа выводит введенные мною данные (рис. 4.16) (рис. 4.17).

Введите строку: Шабалина

Рис. 4.16: Исполнение файла

Шабалина



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`.