Отчёт по лабораторной работе №5

Дисциплина: Архитектура компьютера

София Андреевна Кудякова

Содержание

Список иллюстраций

1 Цель работы

Цель данной лабораторной работы - научиться работать с Midnight Commander, а также освоить инструкции языка ассамблера mov и int.

2 Задания

1. Основы работы с mc
2. Структура программы на языке ассамблера 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 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. Синтаксис директив определения данных следующий: DB [,] [,]. Для объявления неинициированных данных в секции .bss используются директивы resb, resw, resd и другие, которые сообщают ассемблеру, что необходимо зарезервировать заданное ко-

личество ячеек 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 intпредназначена 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 mc

Ввожу команду mc, с помощью которой открывается Midnight Commander. (рис. 4.1).

Рис. 4.1: Команда mc

Убеждаюсь, что команда сработала, так как открылся mc. (рис. 4.2).

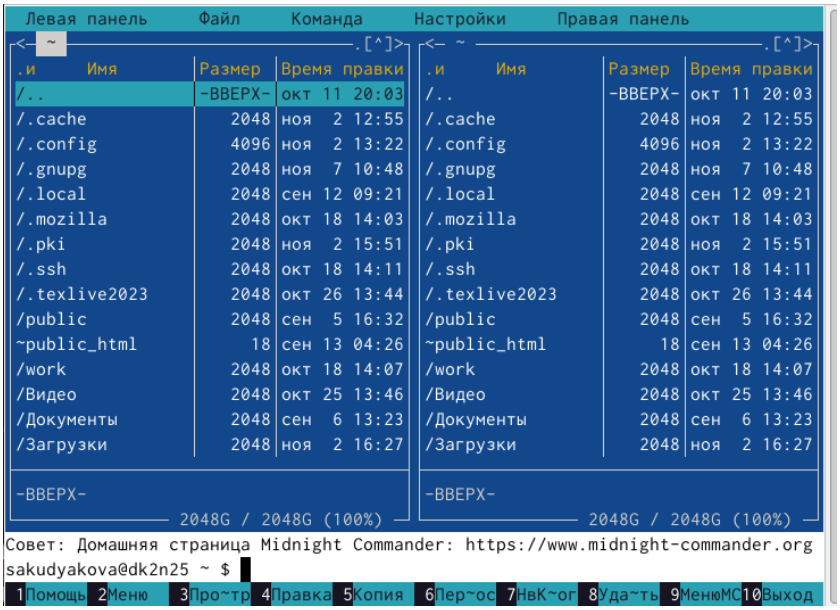

Рис. 4.2: Открытый mc

Перехожу в каталог ~/work/arch-pc и с помощью функциональной клавиши F7 создаю папку lab05 и перехожу в созданный каталог. (рис. 4.3).

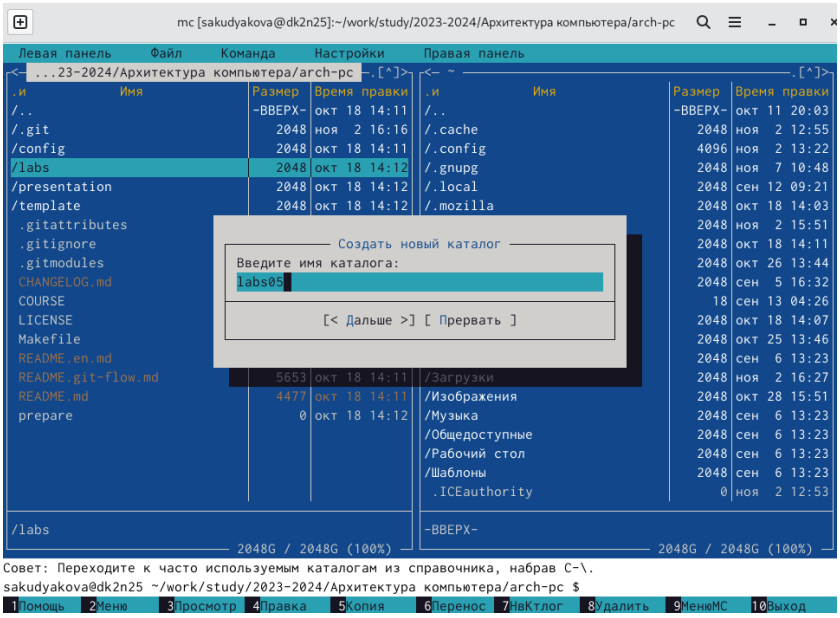

Рис. 4.3: Создание каталога

Пользуясь строкой ввода и командой touch создаю файл lab5-1.asm. (рис. 4.4).

```
Совет: Используйте C-x t для копирования помеченных файлов в командную строку.
sakudyakova@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs05 $ touch lab5-1.asm
 1Помощь  2Меню  3Просмотр  4Правка  5Копия  6Перенос  7НвКтлог  8Удалить  9МенюМС  10Выход
```

Рис. 4.4: Создание файла

4.2 Структура программы на языке ассамблера NASM

С помощью клавиши F4 открываю созданный файл в редакторе nano. (рис. 4.5).

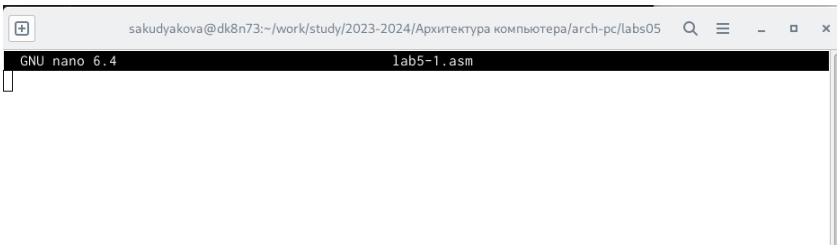

Рис. 4.5: Открытие файла в редакторе

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

```
/afs/.dk.sci.pfu.edu.ru/home/s/a/sakudyakova/work/study/2023-2024/Архитектура компьюте
SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
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,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра
```

Рис. 4.6: Ввод кода программы

С помощью клавиши F3 открываю файл для просмотра и убеждаюсь, что внесенный код программы был сохранен. (рис. 4.7).

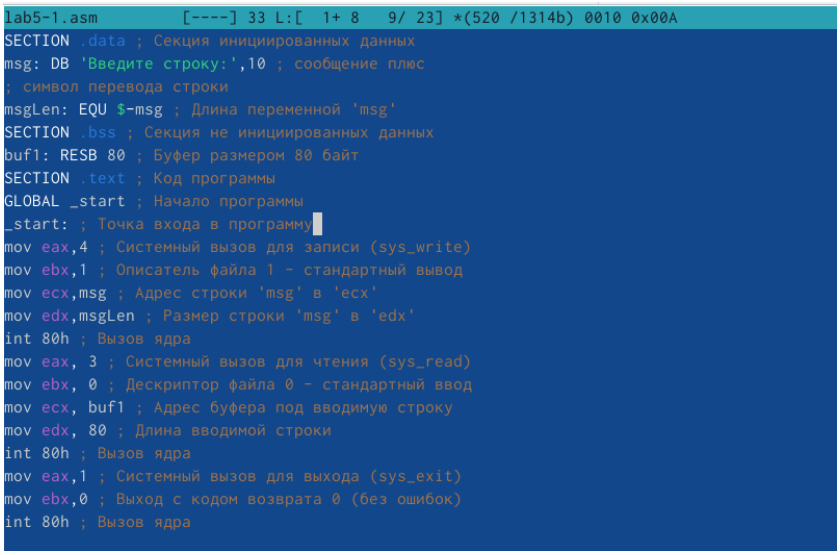

Рис. 4.7: Просмотр файла

Транслируйте текст программы lab5-1.asm в объектный файл. Выполняю компоновку объектного файла и запускаю получившийся исполняемый файл. Программа выводит строку ‘Введите строку:’ и ожидает ввода с клавиатуры. На запрос введите свои ФИО.(рис. 4.8).

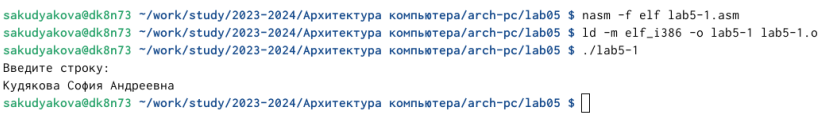

Рис. 4.8: Исполнение файла

4.3 Подключение внешнего файла

Скачиваю файл in_out.asm со страницы курса в ТУИС. Он сохранился в каталог “Загрузки”. (рис. 4.9).

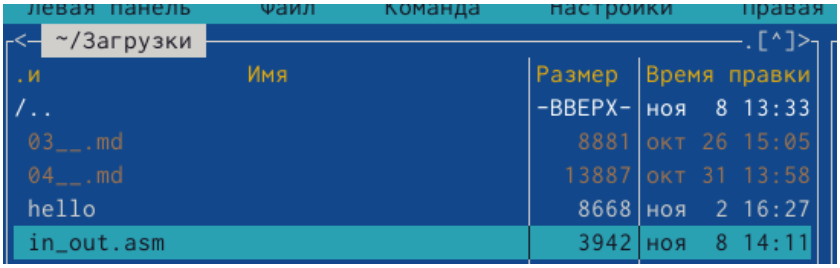

Рис. 4.9: Скачивание файла

Копирую файл in_out.asm из каталога “Загрузки” в каталог lab05 с помощью функциональной клавиши F5. (рис. 4.10).

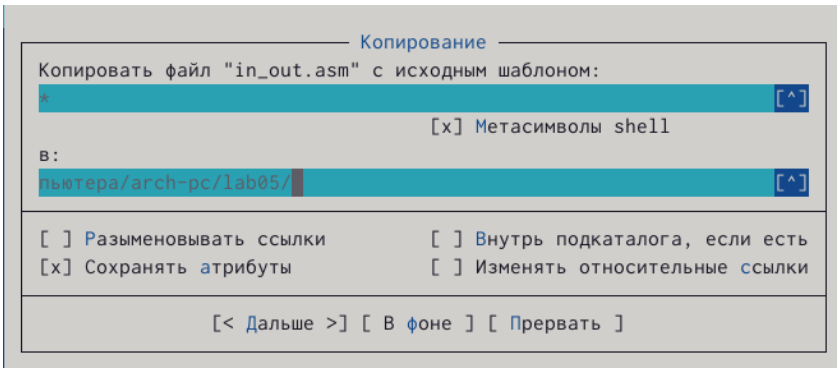

Рис. 4.10: Копирования файла

С помощью функциональной клавиши F6 создаю копию файла lab5-1.asm с именем lab5-2.asm. Выделяю файл lab5-1.asm, нажимаю клавишу F6, ввожу имя файла lab5-2.asm и нажимю Enter. (рис. 4.11).

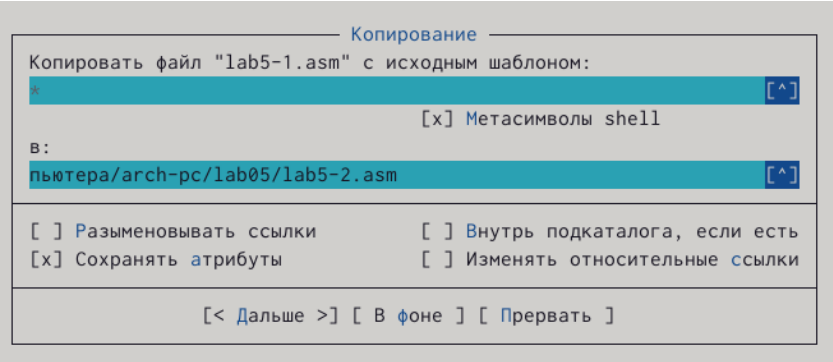

Рис. 4.11: Копирование файла

Исправляю текст программы в файле lab5-2.asm с использованием подпрограмм из внешнего файла in_out.asm. (рис. 4.12).

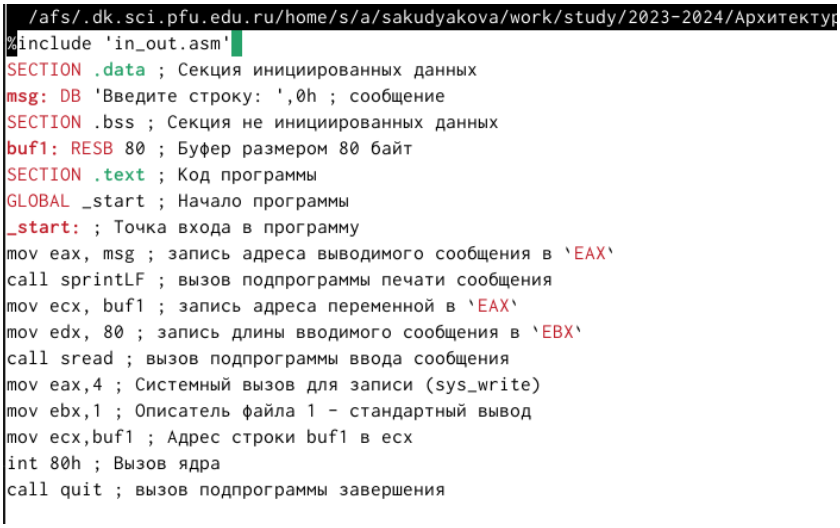

Рис. 4.12: Редактирование файла

Транслируйте текст программы lab5-2.asm в объектный файл. Выполняю компоновку объектного файла и запускаю получившийся исполняемый файл. Программа выводит строку ‘Введите строку:’ и ожидает ввода с клавиатуры. (рис. 4.13).

```
sakudyakova@dk8n73 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ nasm -f elf lab5-2.asm
sakudyakova@dk8n73 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
sakudyakova@dk8n73 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ ./lab5-2
Введите строку:
Кудякова София Андреевна
sakudyakova@dk8n73 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $
```

Рис. 4.13: Исполнение файла

В файле lab5-2.asm меняю подпрограмму sprintLF на sprin. (рис. 4.14).

```
...home/s/a/sakudyakova/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05/
%include 'in_out.asm' ; подключение внешнего файла
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
call quit ; вызов подпрограммы завершения
```

Рис. 4.14: Изменение подпрограммы

Затем транслирую файл, выполняю компоновку объектного файла и заупскаю новый исполняемый файл. (рис. 4.15).

```
sakudyakova@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $  nasm -f elf lab5-2.asm
sakudyakova@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
sakudyakova@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $ ./lab5-2
Введите строку: Кудякова София Андреевна
sakudyakova@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab05 $
```

Рис. 4.15: Исполнение файла

Разница между подпрограммами sprintLF и sprint заключается в том, что в первом случае(sprintLF) ввод запрашивается с новой строки, а во втором случае(sprint) - с той же.

4.4 Выполнение заданий для самостоятельной работы

1. Копирую файл lab5-1.asm и создаю нвый с именем lab5-1-1.asm. (рис. 4.16).

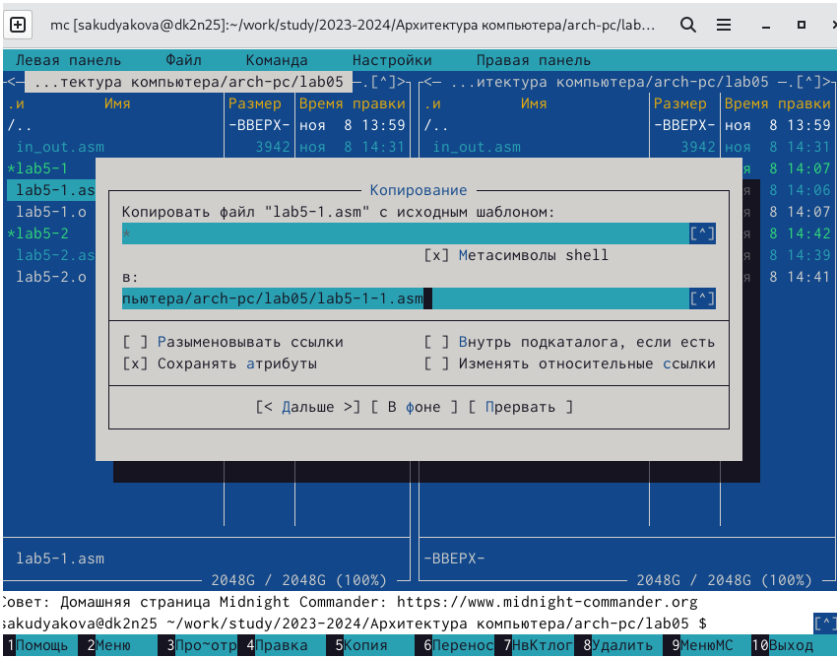

Рис. 4.16: Копирование файла

Открываю файл с помощью F4 для редактирования. Изменяю программу так, чтобы она выводила вывод приглашения, запрос ввода и вводимую пользователем строку. (рис. 4.17).

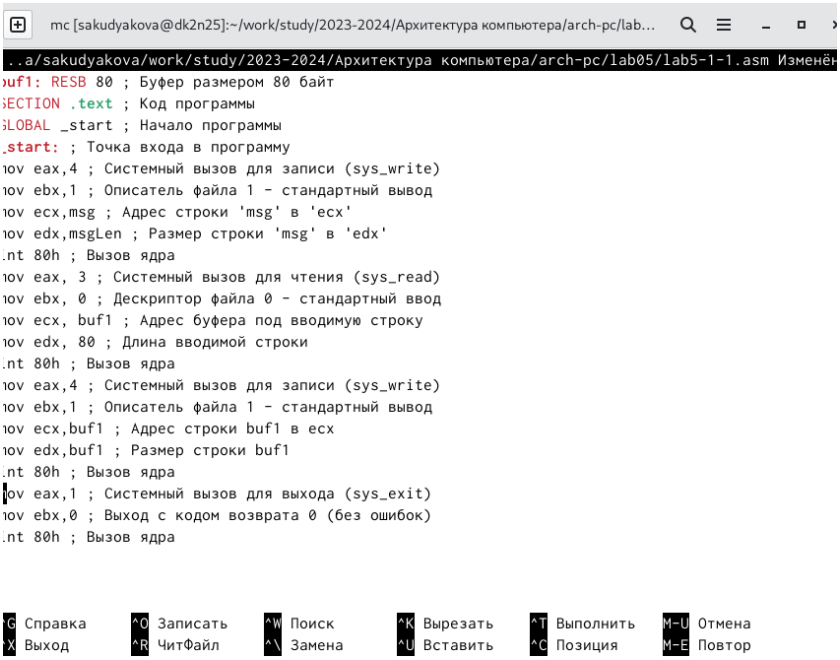

Рис. 4.17: Редактирование файла

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. 4.18).

Рис. 4.18: Исполнение файла

Код программы из п.1:

```
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. Копирую файл lab5-2.asm и создаю нвый с именем lab5-2-1.asm. (рис. 4.19).

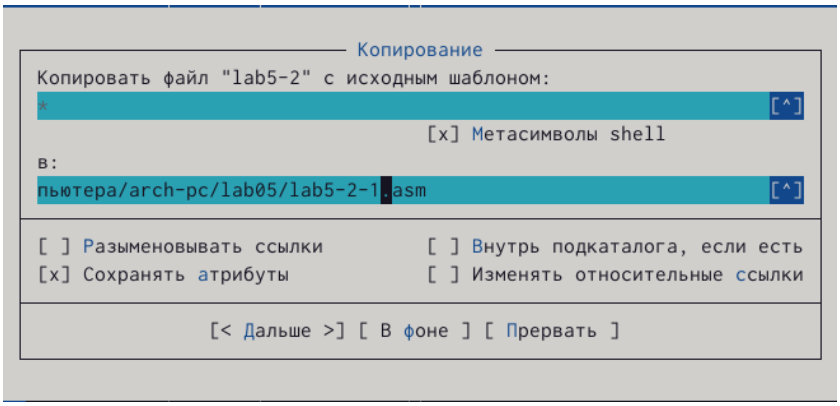

Рис. 4.19: Копирование файла

Открываю файл с помощью F4 для редактирования. Изменяю программу так,

чтобы она выводила вывод приглашения, запрос ввода и вводимую пользователем строку. (рис. 4.20).

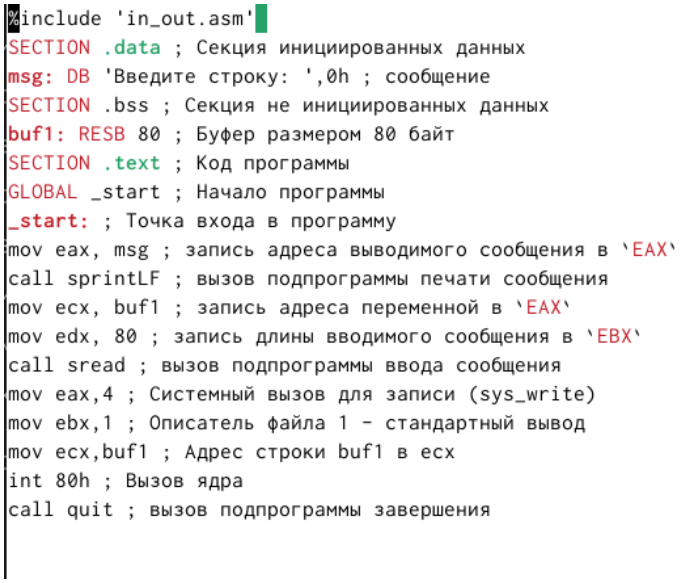

Рис. 4.20: Редактирование файла

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, ввожу свои ФИО, далее программа выводит введенные мною данные. (рис. 4.21).

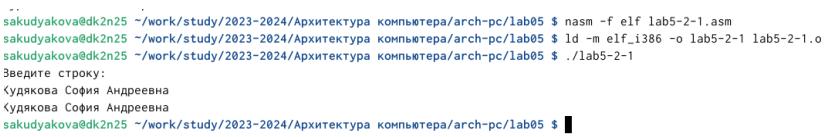

Рис. 4.21: Редактирование файла

Код программы из п.3:

```
%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 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
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

В ходе данной лабораторной работы я научилась работать с Midnight Commander, а также освоила инструкции языка ассамблера mov и int.

Список литературы

Архитектура ЭВМ