

Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Луцкая Алиса Витальевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Основы работы с mc	9
4.2	Структура программы на языке ассемблера NASM	11
4.3	Подключение внешнего файла	13
4.4	Выполнение заданий для самостоятельной работы	16
5	Выводы	20
6	Список литературы	21

Список иллюстраций

4.1	Открытый тс	9
4.2	Перемещение между директориями	10
4.3	Создание каталога	10
4.4	Создание файла	10
4.5	Проверка создание файла	11
4.6	Открытие файла для редактирования и ввод кода	11
4.7	Проверка файла	12
4.8	Компиляция файла и передача на обработку компоновщику	12
4.9	Исполнение файла	12
4.10	Скачанный файл	13
4.11	Копирование файла	13
4.12	Копирование файла	14
4.13	Редактирование файла	14
4.14	Компиляция файла и передача на обработку компоновщику	15
4.15	Исполнение файла	15
4.16	Отредактированный файл	15
4.17	Исполнение файла	15
4.18	Копирование файла	16
4.19	Редактирование файла	17
4.20	Исполнение файла	17
4.21	Копирование файла	18
4.22	Редактирование файла	18
4.23	Исполнение файла	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 mc
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).

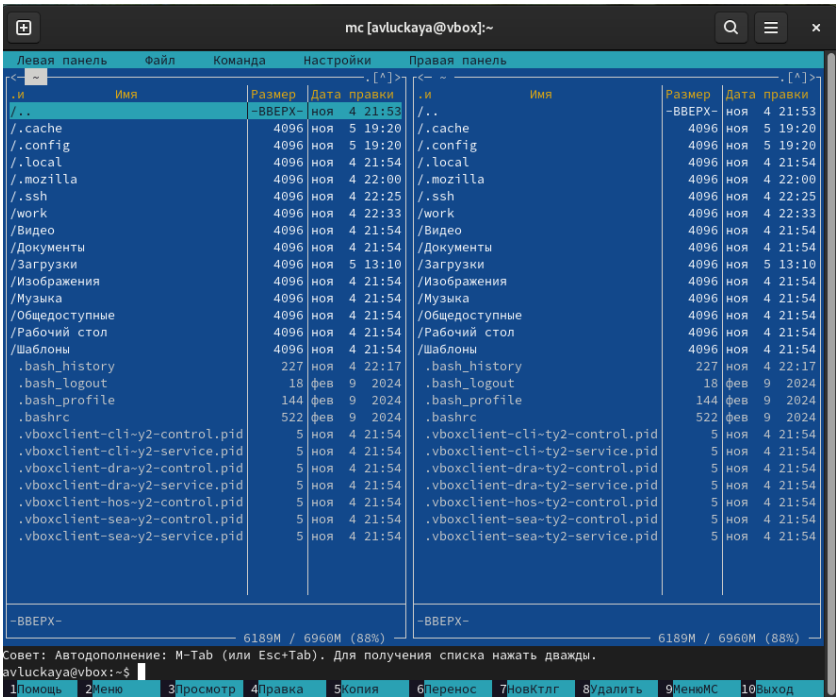


Рис. 4.1: Открытый mc

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

Левая панель	Файл	Команда	Настройки	Правая панель
< ~ /work/arch-pc .[^]>				
.и	Имя	Размер	Дата правки	
/..		-ВВЕРХ-	ноя 4 22:33	
/lab04		4096	ноя 4 22:49	

Рис. 4.2: Перемещение между директориями

С помощью функциональной клавиши F7 создаю каталог lab05 и перехожу в него (рис. 4.3).

Левая панель	Файл	Команда	Настройки	Правая панель
< ~ /work/arch-pc/lab05 .[^]>				
.и	Имя	Размер	Дата правки	
/..		-ВВЕРХ-	ноя 5 19:25	

Рис. 4.3: Создание каталога

В строке ввода прописываю команду touch lab5-1.asm, чтобы создать файл, в котором буду работать (рис. 4.4).

Совет: Макросы % работают даже в командной строке.				
avluckaya@vbox:~/work/arch-pc/lab05\$ touch lab5-1.asm				
1 Помощь	2 Меню	3 Просмотр	4 Правка	5 Копия

Рис. 4.4: Создание файла

Проверяю корректность создания файла (рис. 4.5).

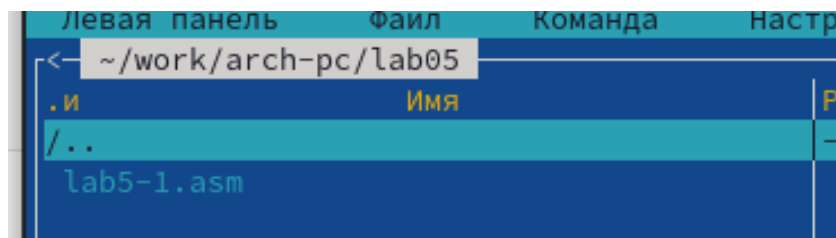


Рис. 4.5: Проверка создание файла

4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе nano, ввожу в файл код программы для запроса строки. Далее выхожу из файла (Ctrl+X), сохраняя изменения (Y, Enter). (рис. 4.6).

```

+ avluckyaya@vbox:~
GNU nano 7.2 /home/avluckyaya/work/arch-pc/lab05/lab5-1.asm
;----- 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
;----- 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,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра

^G Справка      ^O Записать    ^W Поиск       ^K Вырезать    ^T Выполнить   ^G Позиция
^X Выход        ^R ЧитФайл    ^N Замена     ^U Вставить    ^J Вывод       ^/_ К строке

```

Рис. 4.6: Открытие файла для редактирования и ввод кода

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

```

mc [avluckaya@vbox]:~/work/arch-pc/lab05

/home/avluckaya/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. 4.7: Проверка файла

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` (рис. 4.8). Создался исполняемый файл `lab5-1`.

```

avluckaya@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
avluckaya@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o

```

Рис. 4.8: Компиляция файла и передача на обработку компоновщику

Запускаю исполняемый файл. Программа выводит строку “Введите строку:”, я ввожу свои ФИО, после этого программа заканчивает свою работу(рис. 4.9).

```

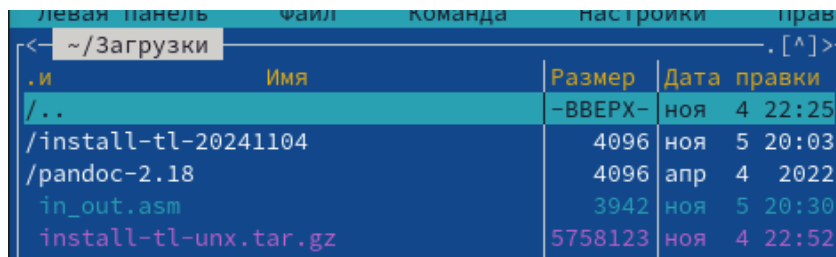
avluckaya@vbox:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Луцкая Алиса Витальевна

```

Рис. 4.9: Исполнение файла

4.3 Подключение внешнего файла

Скачиваю файл `in_out.asm` со страницы курса в ТУИС. Он находится в каталоге “Загрузки” (рис. 4.10).



Имя	Размер	Дата	правки
../	-ВВЕРХ-	ноя 4 22:25	
/install-tl-20241104	4096	ноя 5 20:03	
/pandoc-2.18	4096	апр 4 2022	
in_out.asm	3942	ноя 5 20:30	
install-tl-unx.tar.gz	5758123	ноя 4 22:52	

Рис. 4.10: Скачанный файл

С помощью F5 копирую файл `in_out.asm` из каталога Загрузки в созданный каталог `lab05` (рис. 4.11).

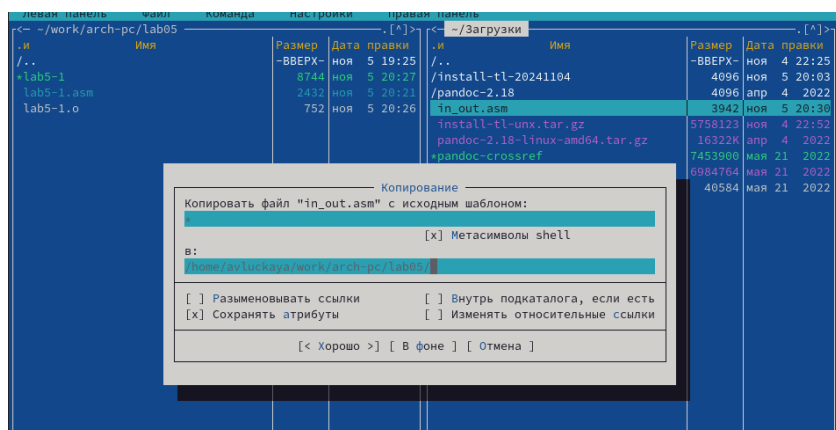


Рис. 4.11: Копирование файла

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).

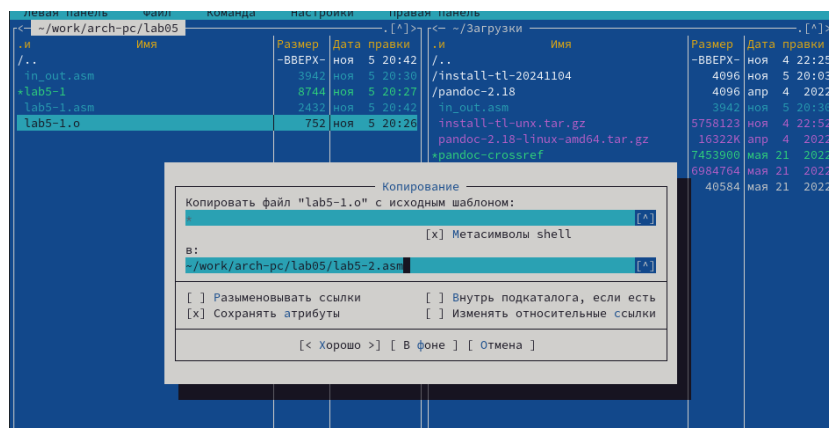


Рис. 4.12: Копирование файла

Изменяю содержимое файла lab5-2.asm во встроенном редакторе nano (рис. 4.13), чтобы в программе использовались подпрограммы из внешнего файла in_out.asm.

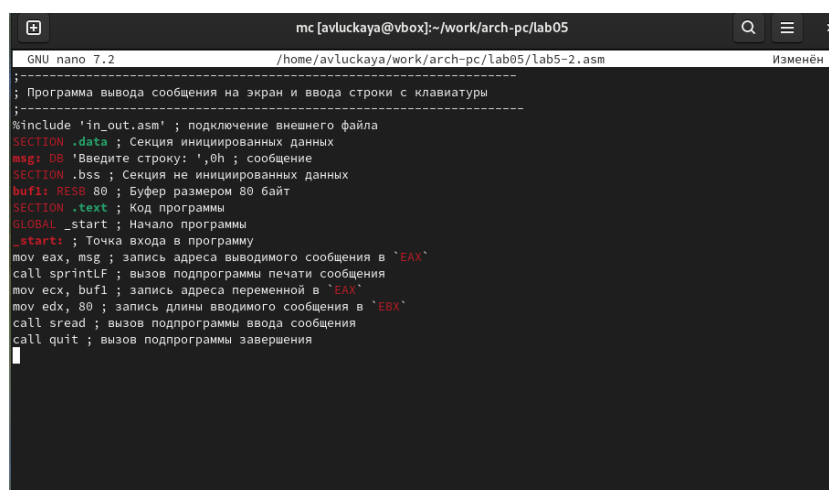


Рис. 4.13: Редактирование файла

Транслирую текст программы файла в объектный файл командой `nasm -f elf lab6-2.asm`. Создался объектный файл lab6-2.o. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab6-2 lab6-2.o`. Создался исполняемый файл lab6-2.(рис. 4.14).

```

avluckaya@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
avluckaya@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o

```

Рис. 4.14: Компиляция файла и передача на обработку компоновщику

Запускаю исполняемый файл (рис. 4.15).

```

avluckaya@vbox:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Луцкая Алиса Витальевна

```

Рис. 4.15: Исполнение файла

Открываю файл lab5-2.asm для редактирования в nano функциональной клавишей F4. Изменяю в нем подпрограмму sprintLF на sprint. Сохраняю изменения и открываю файл для просмотра, проверяю сохранение действий (рис. 4.16).

```

mc [avluckaya@vbox]:~/work/arch-pc/lab05
/home/avluckaya/work/arch-pc/lab05/lab5-2.asm
-----
; 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: Отредактированный файл

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл (рис. 4.17).

```

avluckaya@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
avluckaya@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-2 lab5-2.o
avluckaya@vbox:~/work/arch-pc/lab05$ ./lab5-2-2
Введите строку: Луцкая Алиса Витальевна

```

Рис. 4.17: 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, в этом и заключается различие между подпрограммами `sprintf` и `sprint`.

4.4 Выполнение заданий для самостоятельной работы

1. С помощью функциональной клавиши F5 создаю копию файла `lab5-1.asm` с именем `lab5-1-1.asm` (рис. 4.18).

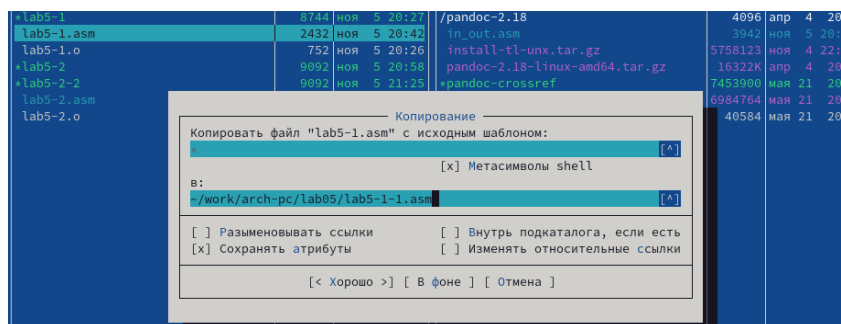


Рис. 4.18: Копирование файла

Открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. 4.19).


```
mc [avluckaya@vbox]:~/work/arch-pc/lab05
GNU nano 7.2 /home/avluckaya/work/arch-pc/lab05/lab5-1-1.asm
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,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра
```

Рис. 4.19: 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. 4.20).

```
avluckaya@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-1-1.asm
avluckaya@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
avluckaya@vbox:~/work/arch-pc/lab05$ ./lab5-1-1
Введите строку:
Луцкая Алиса Витальевна
Луцкая Алиса Витальевна
```

Рис. 4.20: Исполнение файла

3. Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5 (рис. 4.21).

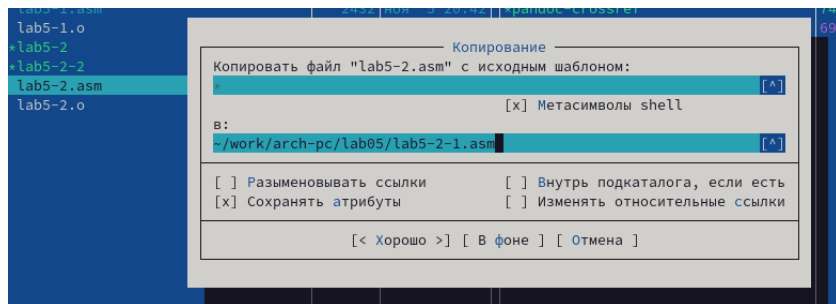


Рис. 4.21: Копирование файла

Открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. 4.22).

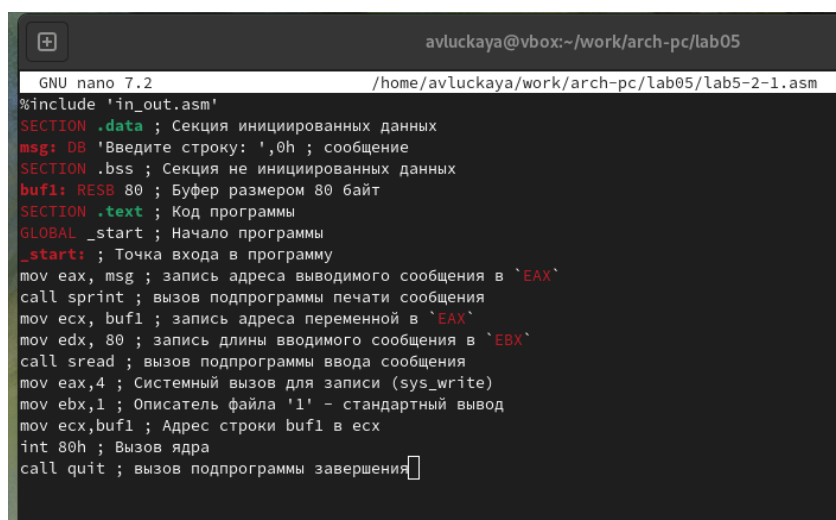


Рис. 4.22: Редактирование файла

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. 4.23).

```
avluckaya@vbox:~/work/arch-pc/lab05$ nasm -f elf lab5-2-1.asm
avluckaya@vbox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
avluckaya@vbox:~/work/arch-pc/lab05$ ./lab5-2-1
Введите строку: Луцкая Алиса Витальевна
Луцкая Алиса Витальевна
```

Рис. 4.23: 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`.

6 Список литературы

1. Лабораторная работа №5