

Лабораторная работа №4

**Основы работы с Midnight Commander (mc). Структура программы
на языке ассемблера NASM. Системные вызовы в ОС GNU Linux**

Софич Андрей Геннадьевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Основы работы с NASM.	8
4.2	Структура программы на языке ассемблера NASM.	10
4.3	Подключение внешнего файла.	13
4.4	Выполнение заданий для самостоятельной работы.	16
5	Выводы	21
	Список литературы	22

Список иллюстраций

4.1	Открытый Midnight Commander	8
4.2	Перемещение в каталог	9
4.3	Создание каталога	9
4.4	Создание файла	10
4.5	Файл в редакторе	10
4.6	Редактирование файла	11
4.7	Открытие файла для проверки	12
4.8	Создание объектного файла	12
4.9	Компоновка	12
4.10	Запуск программы	12
4.11	Файл in_out.asm	13
4.12	Копирование файла в нужную директорию	13
4.13	Копирование файла с изменением названия	14
4.14	Редактирование файла, для использования in_put.asm	14
4.15	Создание объектного файла	14
4.16	Компоновка файла и запуск программы	15
4.17	Редактирование файла	15
4.18	Исполнение файла	15
4.19	Копирование файла lab5-1.asm	16
4.20	Редактирование программы	17
4.21	Исполнение файла	18
4.22	Копирование файла	19
4.23	Исполнение файла	20

Список таблиц

1 Цель работы

Приобретение практических навыков работы в Midnight Commander. Освоение инструкций языка ассемблера `mov` и `int`

2 Задание

Основы работы с NASM

Структура программы на языке ассемблера NASM

Подключение внешнего файла

Выполнение заданий для самостоятельной работы

3 Теоретическое введение

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 (учетверённое слово); • DT (define ten bytes) — определяет переменную размером в 10 байт.

4 Выполнение лабораторной работы

4.1 Основы работы с NASM.

Открываю Midnight Commander, используя команду mc(рис. [4.1]).

Левая панель				Правая панель			
Файл				Файл			
Команда				Команда			
Настройки				Настройки			
Правая панель				Правая панель			
[^]				[^]			
Имя	Размер	Время правки		Имя	Размер	Время правки	
..	-ВВЕРХ-	окт 21 14:29		..	-ВВЕРХ-	окт 21 14:29	
/.cache	432	ноя 5 10:58		/.cache	432	ноя 5 10:58	
/.config	406	ноя 5 10:58		/.config	406	ноя 5 10:58	
/.local	32	окт 21 17:42		/.local	32	окт 21 17:42	
/.mozilla	48	окт 21 14:30		/.mozilla	48	окт 21 14:30	
/.ssh	84	окт 21 17:48		/.ssh	84	окт 21 17:48	
/.texlive2023	18	окт 21 17:51		/.texlive2023	18	окт 21 17:51	
/study_20~arch-pc	296	окт 21 17:48		/study_20~arch-pc	296	окт 21 17:48	
/Видео	0	окт 21 14:29		/Видео	0	окт 21 14:29	
/Документы	0	окт 21 14:29		/Документы	0	окт 21 14:29	
/Загрузки	276	окт 21 17:38		/Загрузки	276	окт 21 17:38	
/Изображения	0	окт 21 14:29		/Изображения	0	окт 21 14:29	
/Музыка	0	окт 21 14:29		/Музыка	0	окт 21 14:29	
/Общедоступные	0	окт 21 14:29		/Общедоступные	0	окт 21 14:29	
/Рабочий стол	0	окт 21 14:29		/Рабочий стол	0	окт 21 14:29	
-ВВЕРХ-				-ВВЕРХ-			
7980M / 20G (39%)				7980M / 20G (39%)			

Рис. 4.1: Открытый Midnight Commander

Перехожу в каталог work, созданный при выполнении предыдущей лабораторной работы(рис. [4.2]).

Левая панель				Правая панель			
Файл		Команда		Настройки		Правая панель	
<- ~/work/arch-pc/lab04				<- /			
.и	Имя	Размер	Время правки	.и	Имя	Размер	Время правки
/..		-BBERX-	ноя 5 11:19	/afs		0	янв 19 2023
				~bin		7	янв 19 2023
				/boot		4096	окт 21 14:22
				/dev		3940	окт 28 18:08
				/etc		4610	ноя 5 10:58
				/home		24	окт 21 14:29
				~lib		7	янв 19 2023
				~lib64		9	янв 19 2023
				/lost+found		0	апр 14 2023
				/media		0	янв 19 2023
				/mnt		0	янв 19 2023
				/opt		0	янв 19 2023
				/proc		0	окт 28 17:22
				/root		140	окт 21 14:28
				/run		1320	ноя 5 11:07
				~sbin		8	янв 19 2023
				/srv		0	янв 19 2023
				/sys		0	окт 28 17:22
				/tmp		580	ноя 5 11:16
				/usr		100	апр 14 2023
-BBERX-				/afs			
7807M / 20G (38%)				7807M / 20G (38%)			

Рис. 4.2: Перемещение в каталог

Создаю новый каталог lab05(рис. [4.3]).

The image shows a terminal window with a file manager interface. The title bar indicates the user is 'mc [andreysofich@fedora]: ~/work/arch-pc/lab04'. The interface is divided into four main sections: 'Левая панель' (Left panel), 'Файл' (File), 'Команда' (Command), and 'Правая панель' (Right panel). The 'Левая панель' shows the current directory as '~/.work/arch-pc/lab04'. The 'Файл' section lists files and directories with columns for 'Имя' (Name), 'Размер' (Size), and 'Время правки' (Modification time). The 'Команда' section shows the command prompt '~/.work/arch-pc/lab04'. The 'Правая панель' shows a list of files and directories with columns for 'Имя' (Name), 'Размер' (Size), and 'Время правки' (Modification time). A dialog box is open in the center, titled 'Создать новый каталог' (Create new directory). It prompts the user to 'Введите имя каталога:' (Enter directory name:). The input field contains 'lab05'. Below the input field are two buttons: '< Хорошо >' (OK) and '[Отмена]' (Cancel). The background of the terminal window is dark blue, and the text is white. The dialog box has a light gray background and a black border.

Рис. 4.3: Создание каталога

В новом каталоге создаю файл lab5-1.asm,в котором я буду работать далее, используя команду touch(рис. [4.4]).

```
[andreysofich@fedora lab05]$ touch lab5-1.asm
```

Рис. 4.4: Создание файла

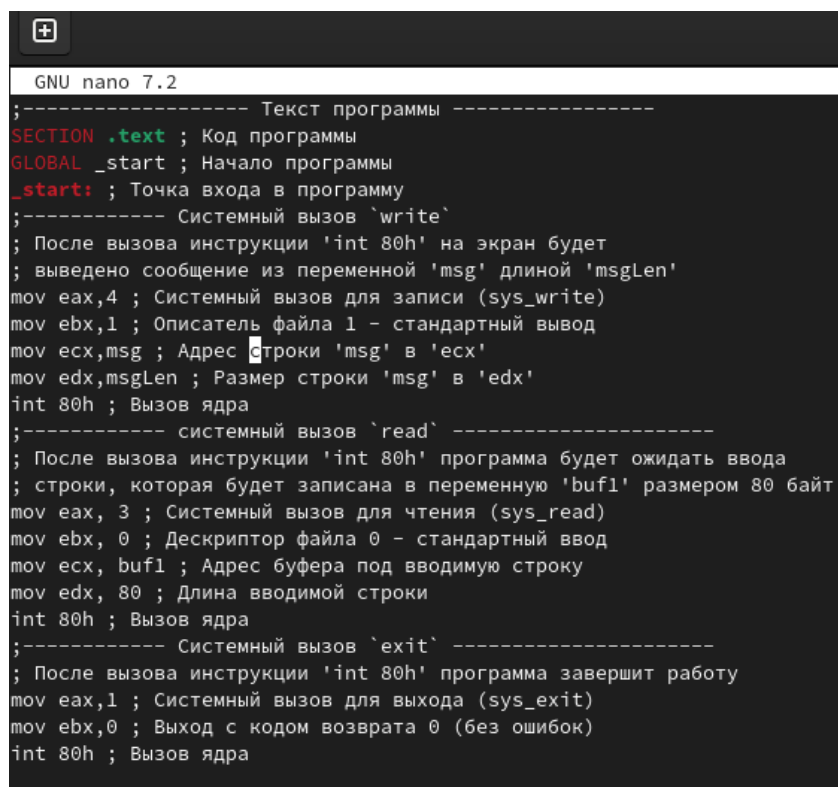
4.2 Структура программы на языке ассемблера NASM.

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



Рис. 4.5: Файл в редакторе

Ввожу в файл код программы для запроса строки(рис. [4.6]). Далее выхожу из редактора,сохраняя изменения.



```
GNU nano 7.2
;----- 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 ; 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. 4.6: Редактирование файла

Используя клавишу F4, открываю файл для просмотра, чтобы проверить, сохранилась ли в нем написанная программа (рис. [4.7]).

```

/home/andreysofich/work/arch-pc/lab04/lab05/lab5-1.asm 1909/2432 78%
;-----
; 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
;----- 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

```

Рис. 4.7: Открытие файла для проверки

Создаю для текста программы объектный файл. Выполняю его компоновку(рис. [4.8]) (рис. [4.9]). После чего создался исполняемый файл lab5-1.

```

[andreysofich@fedora lab05]$ nasm -f elf lab5-1.asm

```

Рис. 4.8: Создание объектного файла

```

[andreysofich@fedora lab05]$ ld -m elf_i386 -o lab5-1 lab5-1.o

```

Рис. 4.9: Компоновка

Запускаю исполняемый файл. Программа выводит строку и ждет ввода с клавиатуры, после ввода своего ФИО программа завершает работу(рис. [4.10]).

```

[andreysofich@fedora lab05]$ ./lab5-1
Введите строку:
Софич Андрей Геннадьевич

```

Рис. 4.10: Запуск программы

4.3 Подключение внешнего файла.

Скачиваю файл `in_out.asm` со страницы ТУИС. Файл сохранился в 'Загрузки' (рис. [4.11]).

<code>/..</code>	<code>-ВВЕРХ-</code>	ноя 5 11:20
<code>/install-tl-unx</code>	100	окт 21 17:38
<code>/pandoc-3.1.8</code>	16	сен 9 20:46
<code>in_out.asm</code>	3942	ноя 5 11:31

Рис. 4.11: Файл `in_out.asm`

Копирую данный файл в каталог `lab05`, используя клавишу `F5` (рис. [4.12]).

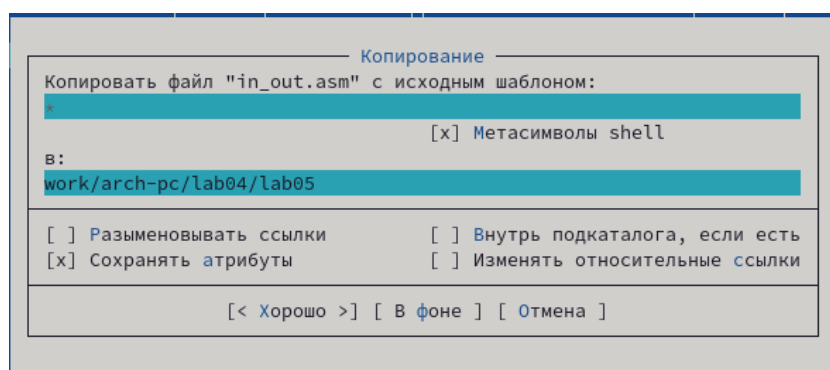


Рис. 4.12: Копирование файла в нужную директорию

С помощью той же утилиты `F5` копирую файл `lab5-1.asm`, но уже с другим названием (рис. [4.13]).

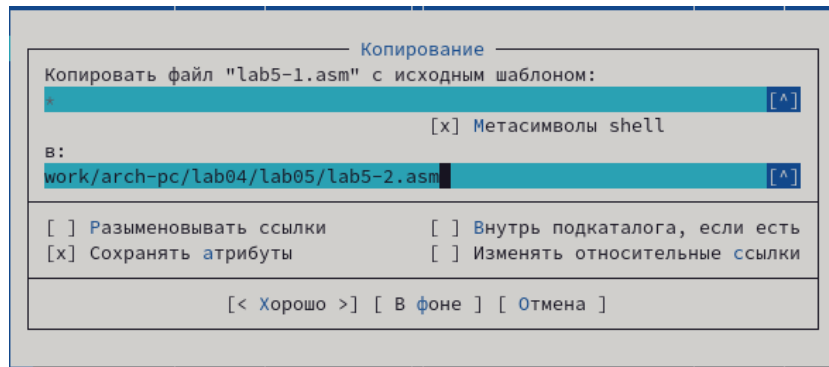


Рис. 4.13: Копирование файла с изменением названия

Меняю содержимое файла lab5-2.asm в редакторе nano, чтобы в программе использовались подпрограммы из внешнего файла in_out.asm(рис. [4.14]).

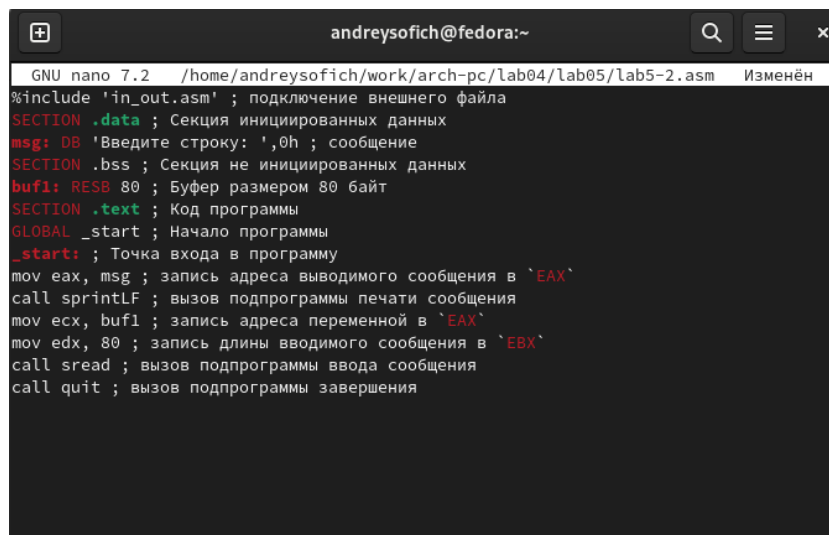


Рис. 4.14: Редактирование файла, для использования in_put.asm

Создаю объектный файл для lab5-2.asm(рис. [4.15]).

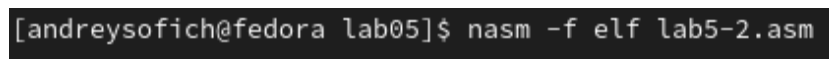


Рис. 4.15: Создание объектного файла

Компоную данный файл, после чего создается исполняемый файл. Запускаю

его и проверяю, работает ли данная программа (рис. [4.16]).

```
[andreysofich@fedora lab05]$ ld -m elf_i386 -o lab5-2 lab5-2.o
[andreysofich@fedora lab05]$ ./lab5-2
Введите строку:
Софич Андрей Геннадьевич
```

Рис. 4.16: Компоновка файла и запуск программы

Открываю файл lab5-2.asm для редактирования в nano, используя F4. Изменяю в нем подпрограмму `sprintLF` на `sprint`, сохраняю изменения и открываю файл для проверки (рис. [4.17]).

```
lab5-2.asm [-M--] 11 L: [ 1+ 9 10/ 15] *(586 / 962b) 0032 0x020 [*][X]
#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. 4.17: Редактирование файла

Выполняю компоновку объектного файла и запускаю новый исполняемый файл (рис. [4.18]).

```
[andreysofich@fedora lab05]$ nasm -f elf lab5-2.asm
[andreysofich@fedora ~]$ mc
[andreysofich@fedora lab05]$ ld -m elf_i386 -o lab5-2 lab5-2.o
[andreysofich@fedora ~]$ mc
[andreysofich@fedora lab05]$ ./lab5-2
Введите строку: Софич Андрей Геннадьевич
```

Рис. 4.18: Исполнение файла

Вся разница заключается в том, что запуск с подпрограммой `sprintLF` запрашивает ввод с новой строки, а исполняемый файл с подпрограммой `sprint` просит

ввод без переноса на новую строку.

4.4 Выполнение заданий для самостоятельной работы.

Создаю копию файла lab5-1.asmс именем lab5-1-1.asm с помощью клавиши F5(рис. [4.19]).

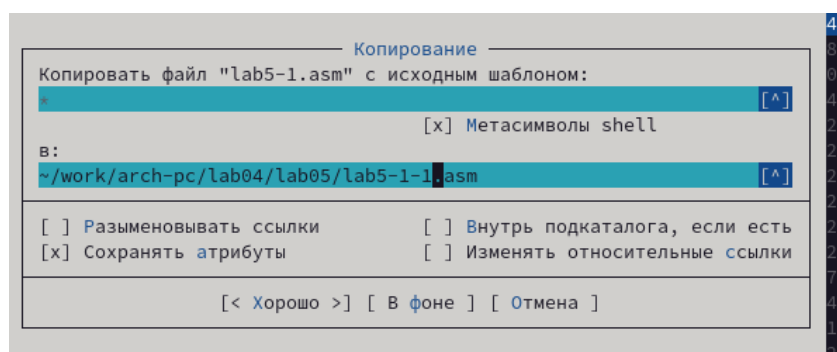


Рис. 4.19: Копирование файла lab5-1.asm

Используя клавишу F4 открываю данный файл в папо и редактирую файл так,чтобы кроме вывода приглашения и запроса ввода, она выводила строку,которую пользователь ввел с клавиатуры(рис. [4.20]).


```

lab5-1-1.asm      [-M--] 19 L:[ 6+25 31/ 39] *(2098/2470b) 1081 0x439
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не иницированных 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 ; 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 записи
mov ebx,1 ; Стандартный вывод
mov ecx,buf1 ; Адрес строки buf1 в ecx
mov edx,buf1 ; Размер 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. 4.20: Редактирование программы

```

SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не иницированных данных
buf1: RESB 80 ; Буфер размером 80 байт
;----- Текст программы -----
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'

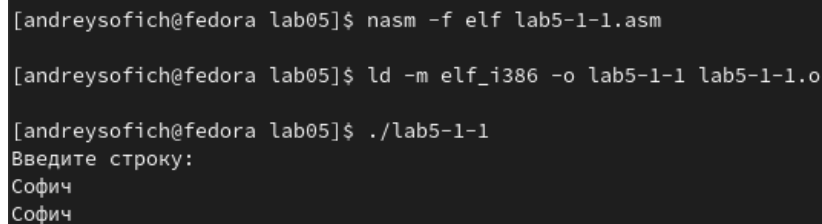
```

```

mov edx,msgLen ; Размер строки 'msg' в 'edx'
int 80h ; Вызов ядра
mov eax, 3 ; Системный вызов для чтения (sys_read)
mov ebx, 0 ;Descriptor 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 записи
mov ebx,1; Стандартный вывод
mov ecx,buf1; Адрес строки buf1 в ecx
mov edx,buf1; Размер 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

```

Создаю объектный файл lab5-1-1.o и обрабатываю его,используя компоновщик,запускаю созданный исполняемый файл, ввожу своё имя, после этого программа выводит то,что я напечатал(рис. [4.21]).



```

[andreysofich@fedora lab05]$ nasm -f elf lab5-1-1.asm
[andreysofich@fedora lab05]$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
[andreysofich@fedora lab05]$ ./lab5-1-1
Введите строку:
Софич
Софич

```

Рис. 4.21: Исполнение файла

Копирую файл lab5-2.asm,используя F5, переименовываю его в lab5-2-3.asm(рис. [4.22]).

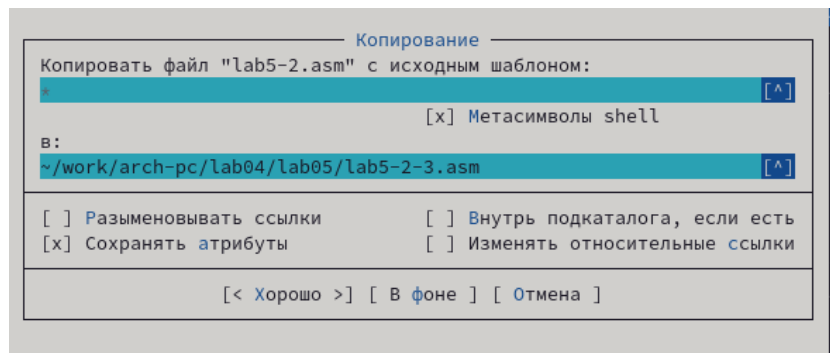


Рис. 4.22: Копирование файла

Используя клавишу F4 открываю данный файл в nano и редактирую файл так, чтобы кроме вывода приглашения и запроса ввода, она выводила строку, которую пользователь ввел с клавиатуры

```
%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 ввода сообщения
mov eax,4; Системный вызов для записи (sys-write)
mov ebx,1; Стандартный вывод
mov ecx,buf1; Адрес строки в buf1 для ecx
int 80h; Вызов ядра
```

`call quit ; вызов подпрограммы завершения`

Создаю объектный файл lab5-2-3.o и обрабатываю его,используя компоновщик,запускаю созданный исполняемый файл, ввожу своё имя, после этого программа выводит то,что я напечатал(рис. [4.23]).

```
[andreysofich@fedora lab05]$ nasm -f elf lab5-2-3.asm
[andreysofich@fedora lab05]$ ld -m elf_i386 -o lab5-2-3 lab5-2-3.o
[andreysofich@fedora lab05]$ ./lab5-2-3
Введите строку: Софич Андрей
Софич Андрей
```

Рис. 4.23: Исполнение файла

5 Выводы

При выполнении данной работы я приобрел навыки работы с Midnight Commander, а также освоил инструкции языка ассемблера mov и int.

Список литературы

::: {#Лабораторная работа №5} :::