

# **Лабораторная работа №5**

**Дисциплина: Архитектура компьютера**

Савостин Олег

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	Основы работы с ms . . . . .	9
4.2	Подключение внешнего файла. . . . .	12
4.3	Выполнение заданий для самостоятельной работы . . . . .	15
<b>5</b>	<b>Выводы</b>	<b>20</b>
	<b>Список литературы</b>	<b>21</b>

# Список иллюстраций

4.1	Запуск программы Midnight Commander . . . . .	9
4.2	Переход в нужный каталог . . . . .	10
4.3	Создание нового каталога . . . . .	10
4.4	Наличие нового каталога в папке. . . . .	10
4.5	Создание файла с помощью командной строки . . . . .	11
4.6	Файл lab5-1.asm . . . . .	11
4.7	Новое содержимое созданного файла . . . . .	12
4.8	Свойство программы . . . . .	12
4.9	in_out.asm в нужном каталоге . . . . .	12
4.10	Копирование файла lab5-2.asm . . . . .	13
4.11	Файл lab5-2.asm . . . . .	13
4.12	Содержимое файла lab5-2.asm . . . . .	14
4.13	Работа файла lab5-2 . . . . .	14
4.14	Изменение sprintLF на sprint . . . . .	15
4.15	Результаты. . . . .	15
4.16	Копия файла . . . . .	15
4.17	Редактированный текст . . . . .	16
4.18	Проверка на правильность выполнения на правильность . . . . .	16
4.19	Создание копии файла lab5-2.asm . . . . .	18
4.20	Содержимое копии файла lab5-2-2.asm . . . . .	18
4.21	Проверка на правильность выполнения работы. . . . .	19

## **Список таблиц**

# 1 Цель работы

Целью данной работы является приобретение практических навыков работы в Midnight Commander. Освоение инструкций языка ассемблера `mov` и `int`.

## **2 Задание**

1. Основы работы с тс
2. Подключение внешнего файла
3. 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)

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)

Инструкция языка ассемблера `int` предназначена для вызова прерывания с указанным номером. В общем виде она записывается в виде `int n` Здесь `n` — номер прерывания, принадлежащий диапазону 0–255. Простейший диалог с пользователем требует наличия двух функций — вывода текста на экран и ввода текста с клавиатуры. Простейший способ вывести строку на экран — использовать системный вызов `write`. Этот системный вызов имеет номер 4, поэтому перед вызовом инструкции `int` необходимо поместить значение 4 в регистр `eax`. Первым аргументом `write`, помещаемым в регистр `ebx`, задаётся дескриптор файла.

Для вывода на экран в качестве дескриптора файла нужно указать 1 (это означает «стандартный вывод», т. е. вывод на экран). Вторым аргументом задаётся адрес выводимой строки (помещаем его в регистр `ecx`, например, инструкцией `mov ecx, msg`). Строка может иметь любую длину. Последним аргументом (т.е. в регистре `edx`) должна задаваться максимальная длина выводимой строки.



# 4 Выполнение лабораторной работы

## 4.1 Основы работы с mc

Сперва, я открываю терминал и открываю Midnight Commander с помощью команды “mc” (рис. 4.1)

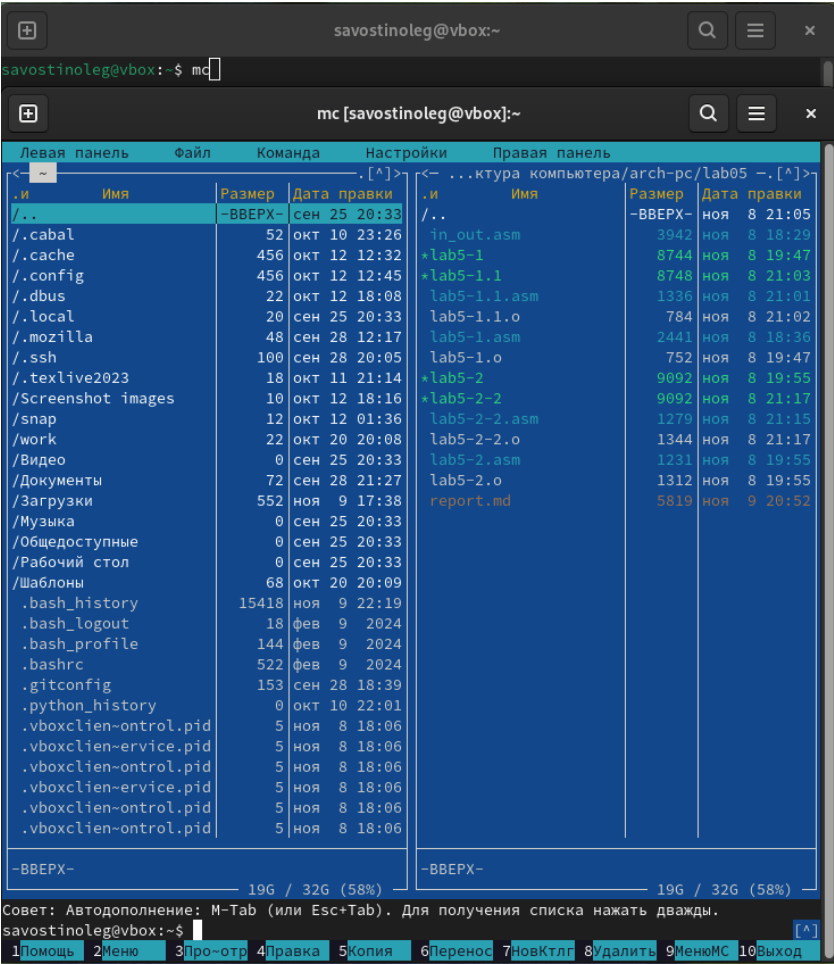


Рис. 4.1: Запуск программы Midnight Commander

Затем, я перехожу в каталог arch-pc и создаю новую папку “lab05” с помощью клавиши F7 (рис. 4.2)(рис. 4.3)(рис. 4.4)

Левая панель	Файл	Команда	Настройки
< ...-2025/Архитектура компьютера/arch-pc -.[^]>			
.и	Имя	Размер	Дата правки
/..		-ВВЕРХ-	сен 28 21:08
/.git		218	окт 20 20:13
/config		24	сен 28 20:05
/labs		152	окт 11 21:03
/presentation		78	окт 11 21:03
/template		36	сен 28 20:05
.gitattributes		1765	сен 28 20:05
.gitignore		4637	сен 28 20:05
.gitmodules		278	сен 28 20:05

Рис. 4.2: Переход в нужный каталог

Создать новый каталог

Введите имя каталога:

lab06 [^]

[< Хорошо >]

[Отмена]

Рис. 4.3: Создание нового каталога

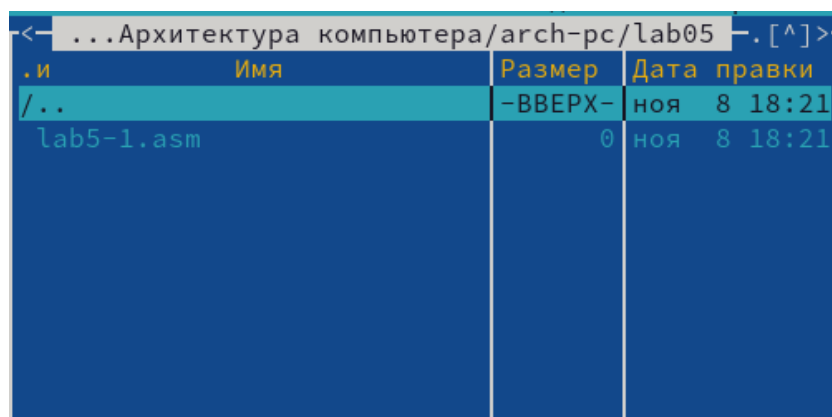
Левая панель	Файл	Команда	Настройки
< ...-2025/Архитектура компьютера/arch-pc -.[^]>			
.и	Имя	Размер	Дата правки
/..		-ВВЕРХ-	сен 28 21:08
/.git		218	окт 20 20:13
/config		24	сен 28 20:05
/lab05		0	ноя 8 18:21
/labs		152	окт 11 21:03
/presentation		78	окт 11 21:03
/template		36	сен 28 20:05
.gitattributes		1765	сен 28 20:05
.gitignore		4637	сен 28 20:05

Рис. 4.4: Наличие нового каталога в папке.

Теперь с помощью командной строки создаю файл lab5-1.asm с touch (рис. 4.5)(рис. 4.6)

```
c:/lab05$ touch lab5-1.asm
```

Рис. 4.5: Создание файла с помощью командной строки



Имя	Размер	Дата правки
..	-ВВЕРХ-	ноя 8 18:21
lab5-1.asm	0	ноя 8 18:21
..	-ВВЕРХ-	ноя 8 18:21

Рис. 4.6: Файл lab5-1.asm

Открываю мною созданный файл с помощью текстового редактора и вставляю туда текст, предоставленный в документе Архитектура ЭВМ (рис. 4.7). Транслирую текст в объектный файл, компилирую его и затем запускаю исполняемый файл. Программа вводит текст 'Введите строку:', после чего я введу свои ФИ. (рис. 4.8)

```

/home/savostinoleg/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05/lab5-1.asm 2079/2441 85%
;-----
; 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 ; Дескриптор файла 0 - стандартный ввод
mov ecx,buf1 ; Адрес буфера под вводимую строку
mov edx,80 ; Длина вводимой строки
int 80h ; Вызов ядра

```

Рис. 4.7: Новое содержимое созданного файла

```

savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-1.asm
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-1
Введите строку:
Савостин Олег

```

Рис. 4.8: Свойство программы

## 4.2 Подключение внешнего файла.

Сначала, я устанавливаю файл in\_out.asm со страницы курса в ТУИС и переношу его в каталог, в котором он будет использован (рис. 4.9)

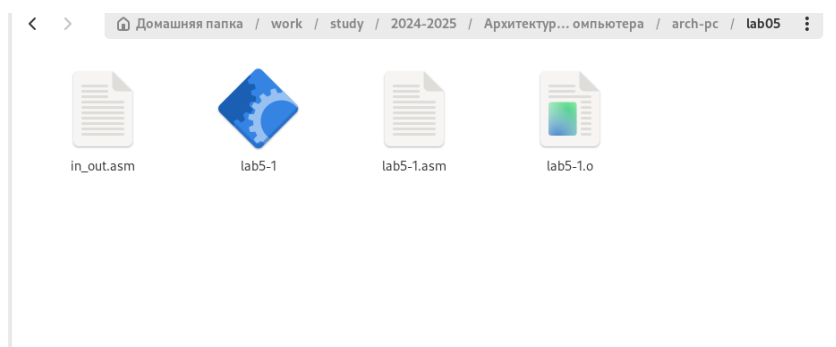


Рис. 4.9: in\_out.asm в нужном каталоге

Теперь, с помощью функциональной клавиши F6 я создаю копию файла lab5-1.asm с именем lab5-2.asm. Выделяю файл lab5-1.asm, нажимаю клавишу F6 и ввожу название lab5-2.asm (рис. 4.10)

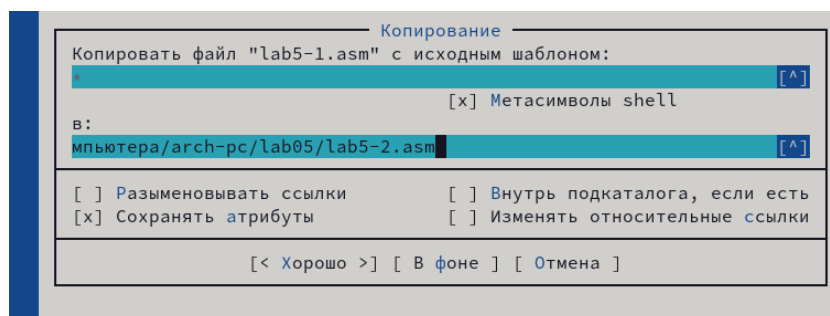


Рис. 4.10: Копирование файла lab5-2.asm

...-2025/Архитектура компьютера/arch-pc/lab05			
Имя	Размер	Дата правки	
./..	-ВВЕРХ-	ноя 8	18:21
in_out.asm	3942	ноя 8	18:29
*lab5-1	8744	ноя 8	18:41
lab5-1.asm	2441	ноя 8	18:36
lab5-1.o	752	ноя 8	18:40
lab5-2.asm	2441	ноя 8	18:36

Рис. 4.11: Файл lab5-2.asm

Изменяю содержимое файла на текст, предоставленный на странице курса в ТУИС (рис. 4.12)

```
lab5-2.asm [----] 0 L: [ 1+25 26/ 26] *(1233/1233b) <EOF>
;-----
; 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 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
```

Рис. 4.12: Содержимое файла lab5-2.asm

Создаю файл и проверяю его работу(рис. 4.13)

```
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-2.asm
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-2
Введите строку:
Савостин Олег
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$
```

Рис. 4.13: Работа файла lab5-2

Теперь, изменяю sprintf на printf(рис. 4.14). Разница состоит в том, что теперь вводимый мною текст теперь находится на одной строке с “Введите строку:”, когда в прошлый раз текст переходил на новую строку (рис. 4.15)

```

;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----

%include 'in_out.asm' ; 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

call quit ; вызов подпрограммы завершения

```

Рис. 4.14: Изменение sprintLF на sprint

```

savostinolegavbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-2.asm
savostinolegavbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
savostinolegavbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-2
Введите строку: Савостин Олег
savostinolegavbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$

```

Рис. 4.15: Результаты.

## 4.3 Выполнение заданий для самостоятельной работы

Создаю копию файла lab5-1.asm - lab5-1.1.asm(рис. 4.16)

Копирование

Копировать файл "lab5-1.asm" с исходным шаблоном:

[x] Метасимволы shell

в:

пьютера/arch-pc/lab05/lab5-1.1.asm

[ ] Разыменовывать ссылки

[x] Сохранять атрибуты

[ ] Внутрь подкаталога, если есть

[ ] Изменять относительные ссылки

[< Хорошо >]

[ В фоне ]

[ Отмена ]

Рис. 4.16: Копия файла

Редактирую содержимое файла чтобы он работал по алгоритму (1): • вывести приглашение типа “Введите строку.”; • ввести строку с клавиатуры; • вывести введенную строку на экран.(рис. 4.17)

```
lab5-1.1.asm      [----]  0 L:[ 1+26 27/ 27] *(1336/1336b) <EOF>      [*]
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
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start
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
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. 4.17: Редактированный текст

Теперь проверяю на правильность выполнения изменения файла. Всё сделано верно.(рис. 4.18)

```
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-1.1.asm
lab5-1.1.asm:8: warning: label alone on a line without a colon might be in error [-w+label-orphan]
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 -o lab5-1
.1 lab5-1.1.o
ld: не распознан режим эмуляции: elf_i386
Поддерживаемые эмуляции: elf_x86_64 elf32_x86_64 elf_i386 elf_iamcu i386pep i386pe elf64bpf
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -m elf_i386 -o lab5-1
.1 lab5-1.1.o
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-1.1
Введите строку:
Савостин Олег
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ls
lab5-1 lab5-1.1 lab5-1.1.asm lab5-1.1.o lab5-1.asm lab5-1.o lab5-2 lab5-2.asm lab5-2.o
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$
```

Рис. 4.18: Проверка на правильность выполнения на правильность

Текст кода в Рис.16 : 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
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start
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
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

```

Теперь делаю подобную программу, только с использованием in\_out.asm. Сперва, я создаю копию файла lab5-2.asm и возвращаю файл in\_out.asm(рис. 4.19)

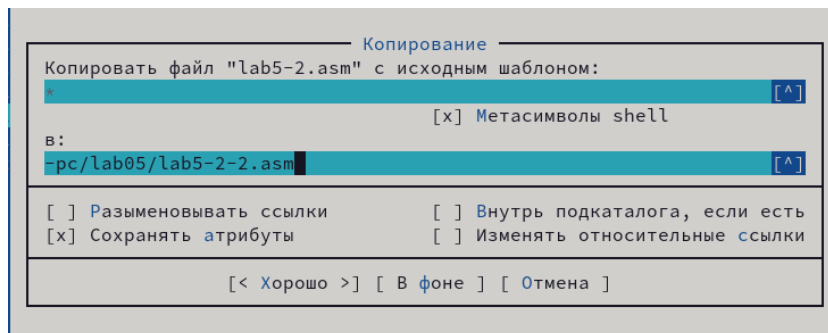


Рис. 4.19: Создание копии файла lab5-2.asm

Затем, я редактирую текст файла, чтобы он повторял подобный алгоритм (1).(рис. 4.20)

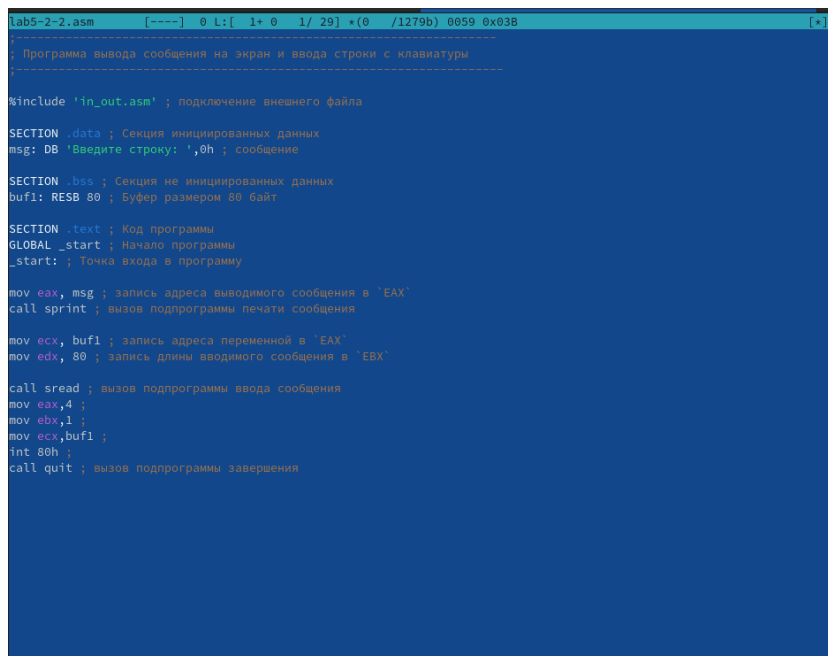


Рис. 4.20: Содержимое копии файла lab5-2.asm

Теперь, подобно предыдущему разу, я проверяю на правильность выполнения работы. Всё сделано корректно (рис. 4.21)

```
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ nasm -f elf lab5-2-2.asm
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ld -melf_i386 -o lab5-2-2 lab5-2-2.o
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$ ./lab5-2-2
Введите строку: Савостин Олег
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab05$
```

Рис. 4.21: Проверка на правильность выполнения работы.

Код из второго файла:

```
%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
mov eax,4 ;
mov ebx,1 ;
mov ecx,buf1 ;
int 80h ;
```

## 5 Выводы

При выполнении лабораторной работы я приобрел практические навыки работы в Midnight Commander и освоил инструкции языка ассемблера `mov` и `int`.

# **Список литературы**

1. Лабораторная работа №5