

Лабораторная работа №4

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

Селиванов Вячеслав Алексеевич

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	13
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	16
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]).

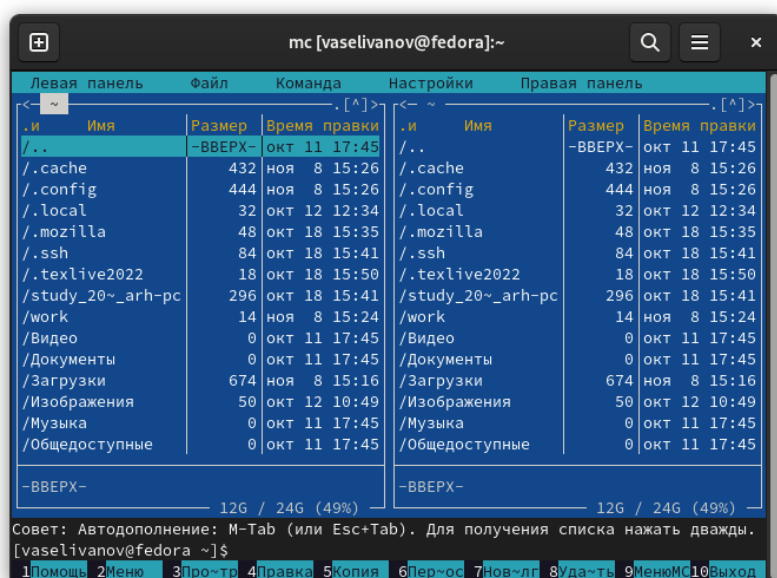


Рис. 4.1: Открытый Midnight Commander

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

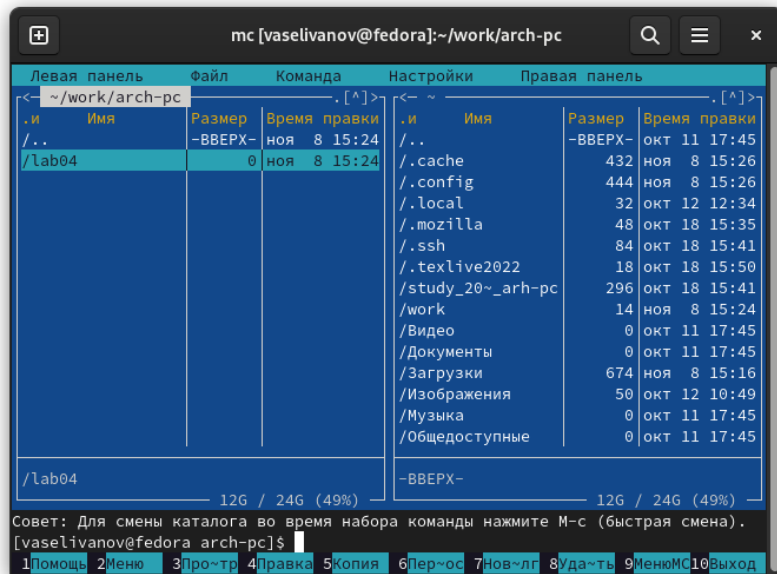


Рис. 4.2: Перемещение в каталог

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

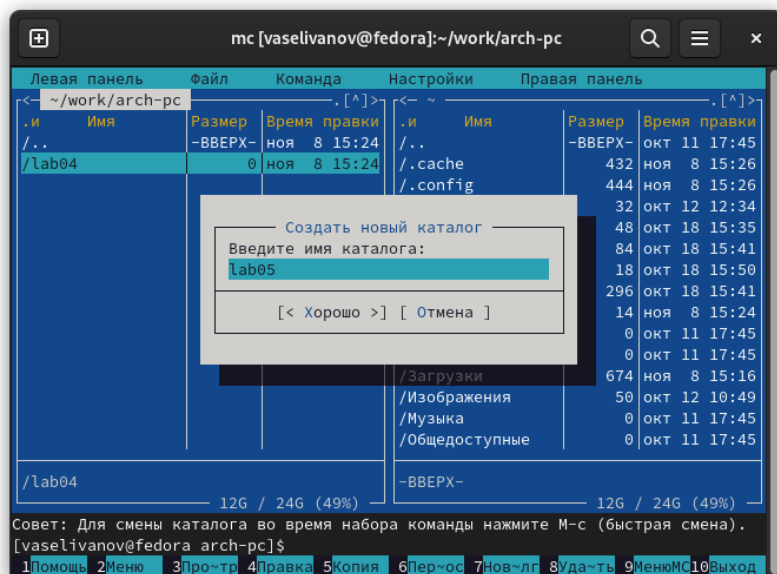


Рис. 4.3: Создание каталога

В новом каталоге создаю файл lab5-1.asm, в котором я буду работать далее,

используя команду touch(рис. [4.4]).

```
vaselivanov@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]).

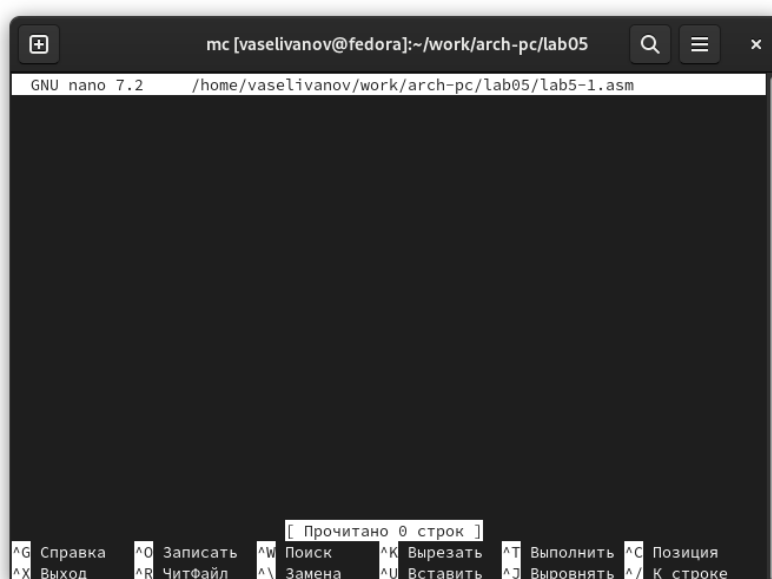


Рис. 4.5: Файл в редакторе

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

```

GNU nano 7.2
;-----
; 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
;----- 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]).

```

lab5-1.asm      [-M--]  0 L:[ 1+ 0  1/ 36] *(0  /2432b) 0059 0x0
;-----
; 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
;----- 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: Открытие файла для проверки

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

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

Рис. 4.8: Создание объектного файла

```
[vaselivanov@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]).

```
Селиванов [vaselivanov@fedora lab05]$ ./lab5-1
Введите строку:
Селиванов Вячеслав Алексеевич
```

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]).

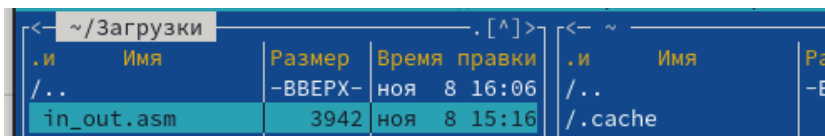


Рис. 4.11: Файл in_out.asm

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

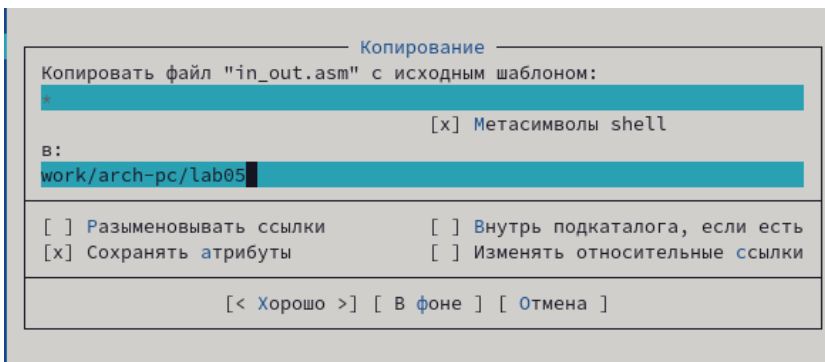


Рис. 4.12: Копирование файла в нужную директорию

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

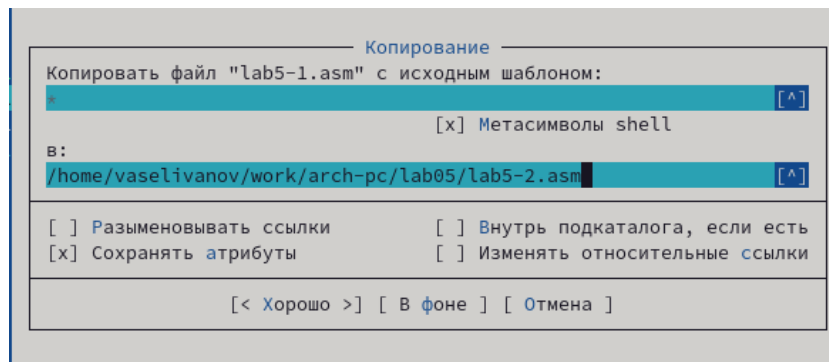


Рис. 4.13: Копирование файла с изменением названия

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

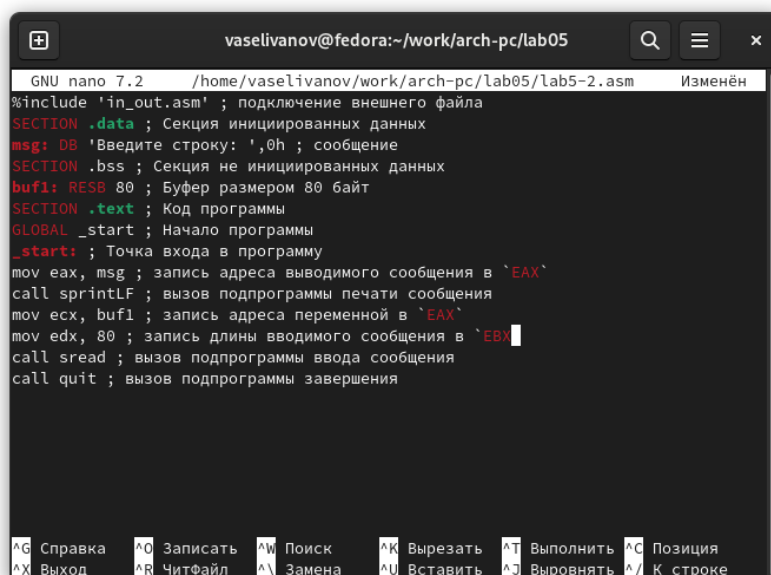


Рис. 4.14: Редактирование файла, для использования in_put.asm

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

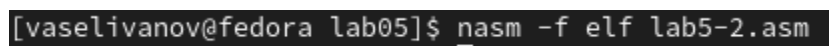


Рис. 4.15: Создание объектного файла

Компонуя данный файл, после чего создается исполняемый файл. Запускаю его и проверяю, работает ли данная программа (рис. [4.16]).

```
[vaselivanov@fedora lab05]$ nasm -f elf lab5-2.asm
[vaselivanov@fedora lab05]$ ld -m elf_i386 -o lab5-2 lab5-2.o
[vaselivanov@fedora lab05]$ ./lab5-2
Введите строку:
Селиванов Вячеслав Алексеевич
```

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]).

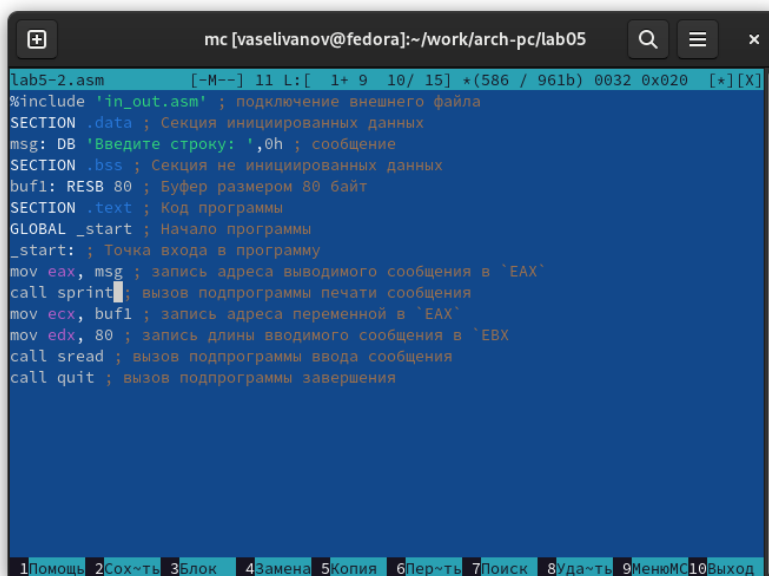


Рис. 4.17: Редактирование файла

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

```
[vaselivanov@fedora lab05]$ nasm -f elf lab5-2.asm
[vaselivanov@fedora lab05]$ ld -m elf_i386 -o lab5-2-2 lab5-2.o
[vaselivanov@fedora lab05]$ ./lab5-2-2
Введите строку: Селиванов Вячеслав Алексеевич
[vaselivanov@fedora lab05]$
```

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.asm` именем `lab5-1-1.asm` с помощью клавиши F5(рис. [4.19]).

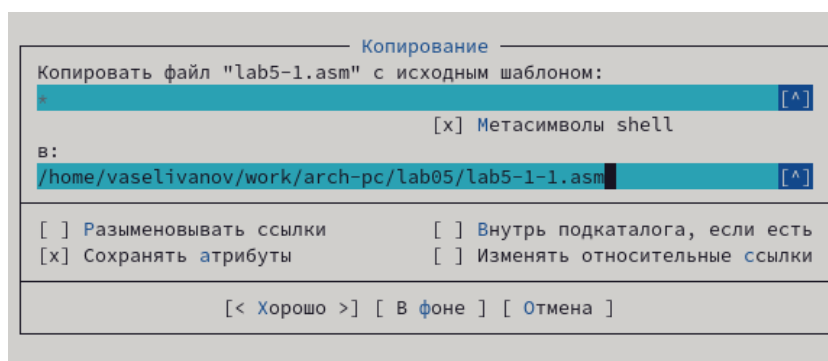
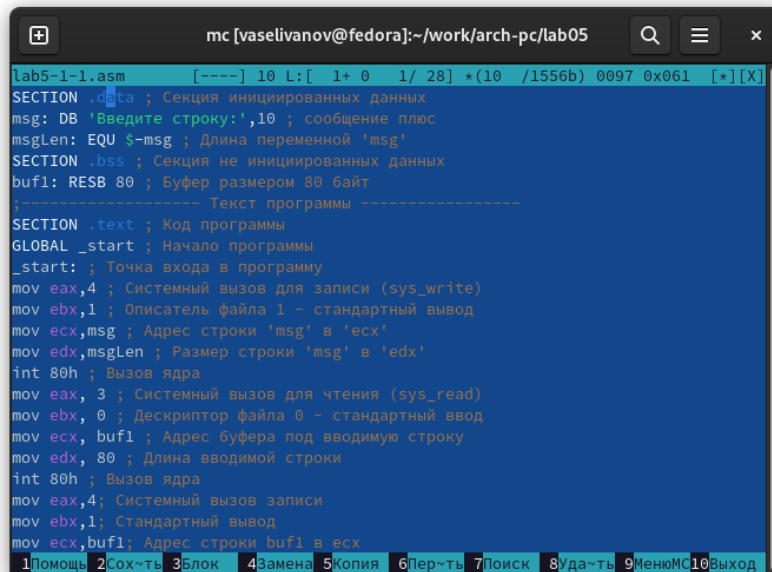


Рис. 4.19: Копирование файла `lab5-1.asm`

Используя клавишу F4 открываю данный файл в nano 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 [----] 10 L: [ 1+ 0 1/ 28] *(10 /1556b) 0097 0x061 [*][X]
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 ; 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
1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9Меню 10Выход
```

Рис. 4.20: Редактирование программы

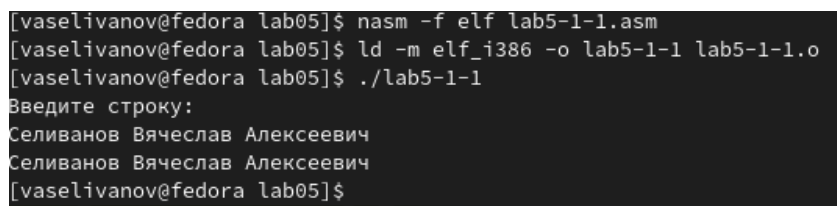
```
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не иницированных данных
buf1: RESB 80 ; Буфер размером 80 байт
;----- Текст программы -----
SECTION .text ; Код программы
GLOBAL _start ; Начало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

```

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



```

[vaselivanov@fedora lab05]$ nasm -f elf lab5-1-1.asm
[vaselivanov@fedora lab05]$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
[vaselivanov@fedora lab05]$ ./lab5-1-1
Введите строку:
Селиванов Вячеслав Алексеевич
Селиванов Вячеслав Алексеевич
[vaselivanov@fedora lab05]$

```

Рис. 4.21: Исполнение файла

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

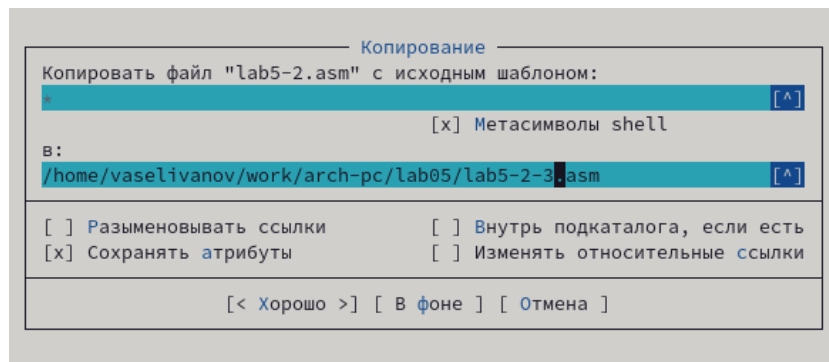


Рис. 4.22: Копирование файла

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]).

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

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} :::