

Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Намруев Максим Саналович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Подключение внешнего файла	12
5	Задание для самостоятельной работы	16
6	Выводы	20

Список иллюстраций

4.1	Открытие Midnight Commander	8
4.2	Переход в каталог	8
4.3	Создание нового каталога	9
4.4	Создание файла	9
4.5	Открытие редактора	10
4.6	Ввод текста	10
4.7	Просмотр файла	11
4.8	Запуск файла	12
4.9	Скачивание файла	13
4.10	Копирование файла	13
4.11	Копирование файла	14
4.12	Исправление файла	14
4.13	Создание исполняемого файла	15
4.14	Запуск новой программы	15
5.1	Создание копии	16
5.2	Редактирование программы	17
5.3	Проверка программы	17
5.4	Создание новой копии	18
5.5	Редактирование нового файла	18
5.6	Проверка работы программы	19

Список таблиц

1 Цель работы

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

2 Задание

1. Основы работы с Midnight Commander
2. Структура программы на языке ассемблера NASM
3. Подключение внешнего файла

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).

4 Выполнение лабораторной работы

Открываю Midnight Commander с помощью команды `mc`. (рис. [4.1]).

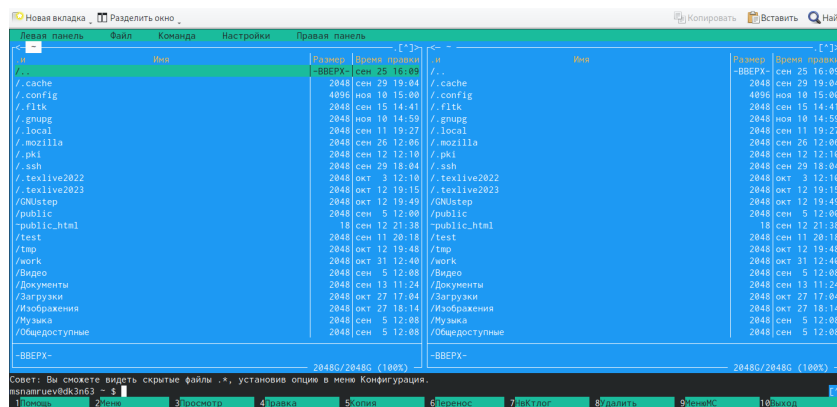


Рис. 4.1: Открытие Midnight Commander

Пользуясь клавишами вверх, вниз и `enter` перехожу в каталог `~/work/arch-rc`, созданный при выполнении прошлой лабораторной работы.(рис. [4.2]).

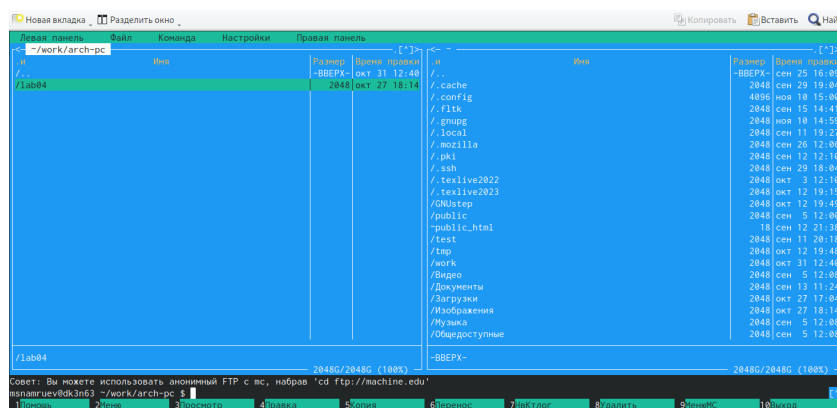


Рис. 4.2: Переход в каталог

С помощью функциональной клавиши f7 создаю папку lab05 и перехожу в созданный каталог.(рис. [4.3]).

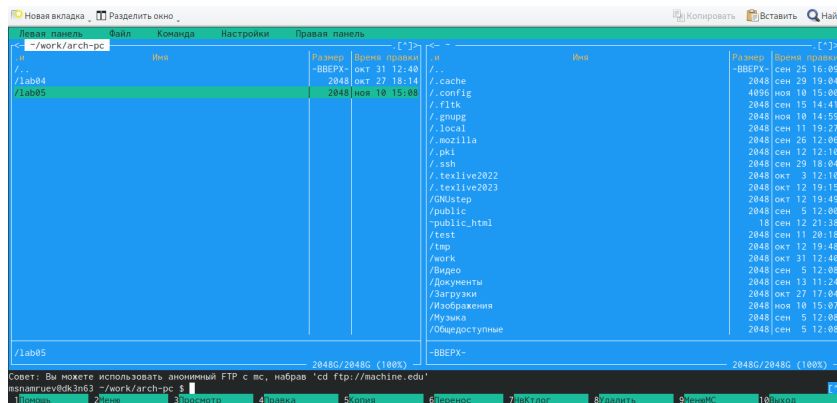


Рис. 4.3: Создание нового каталога

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

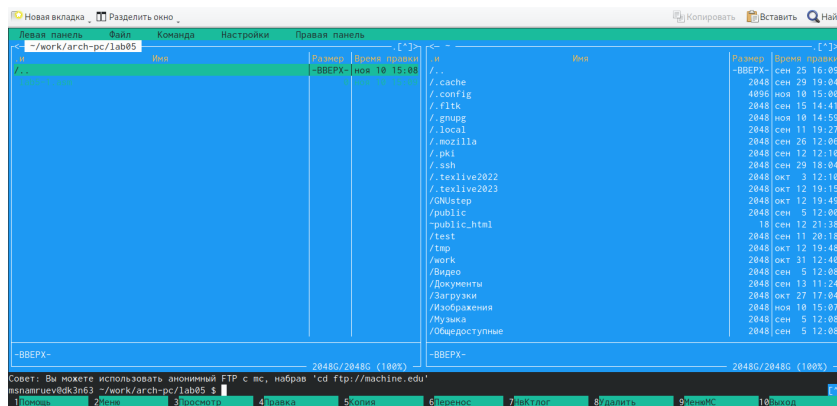


Рис. 4.4: Создание файла

С помощью функциональной клавиши F4 открываю файл lab05-1.asm для редактирования во встроенном редакторе.(рис. [4.5]).



Рис. 4.5: Открытие редактора

Ввожу текст программы из листинга 5.1, сохраняю изменения и закрываю файл.
(рис. [4.6]).

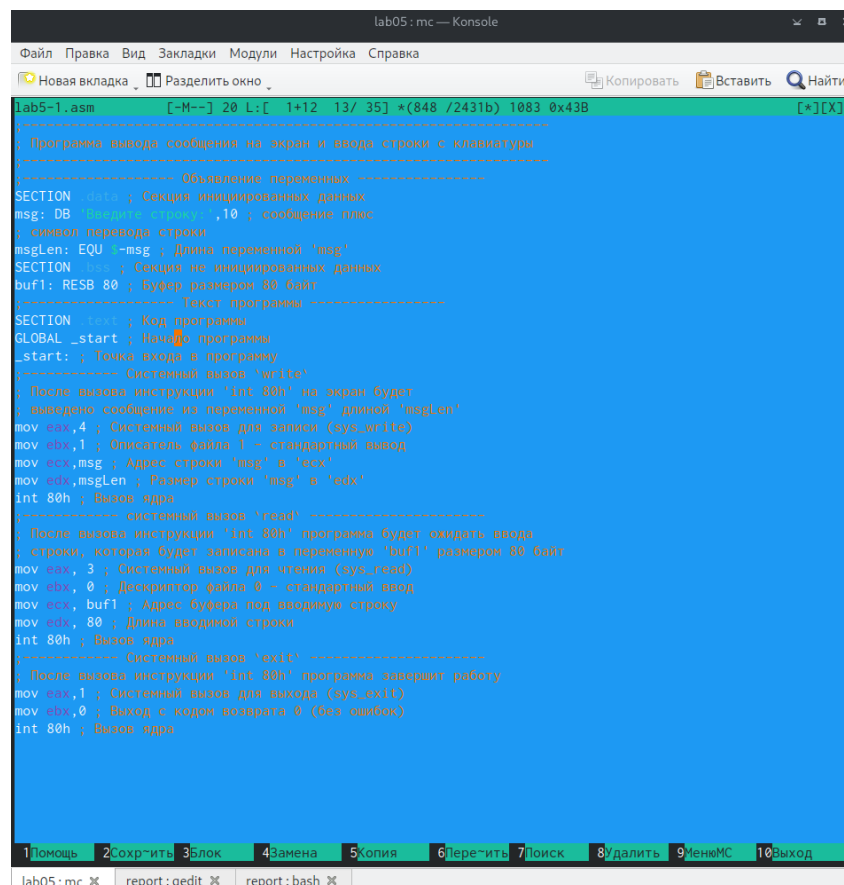
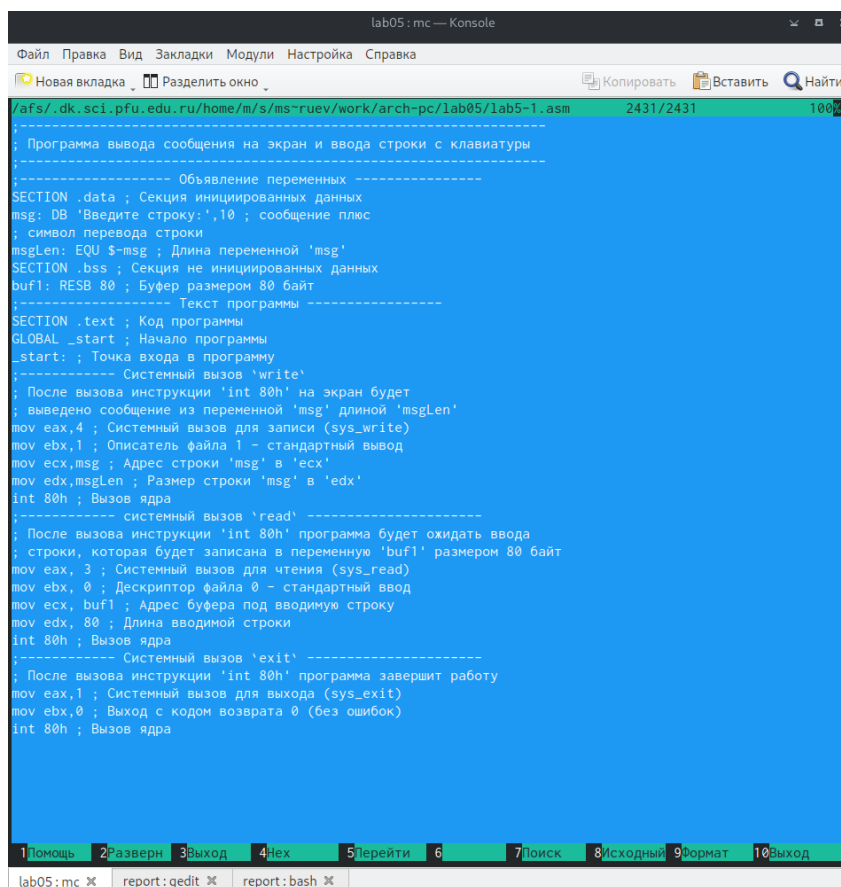


Рис. 4.6: Ввод текста

С помощью функциональной клавиши F3 открываю файл lab5-1.asm для просмотра. Убеждаюсь что файл содержит текст программы.(рис. [4.7]).



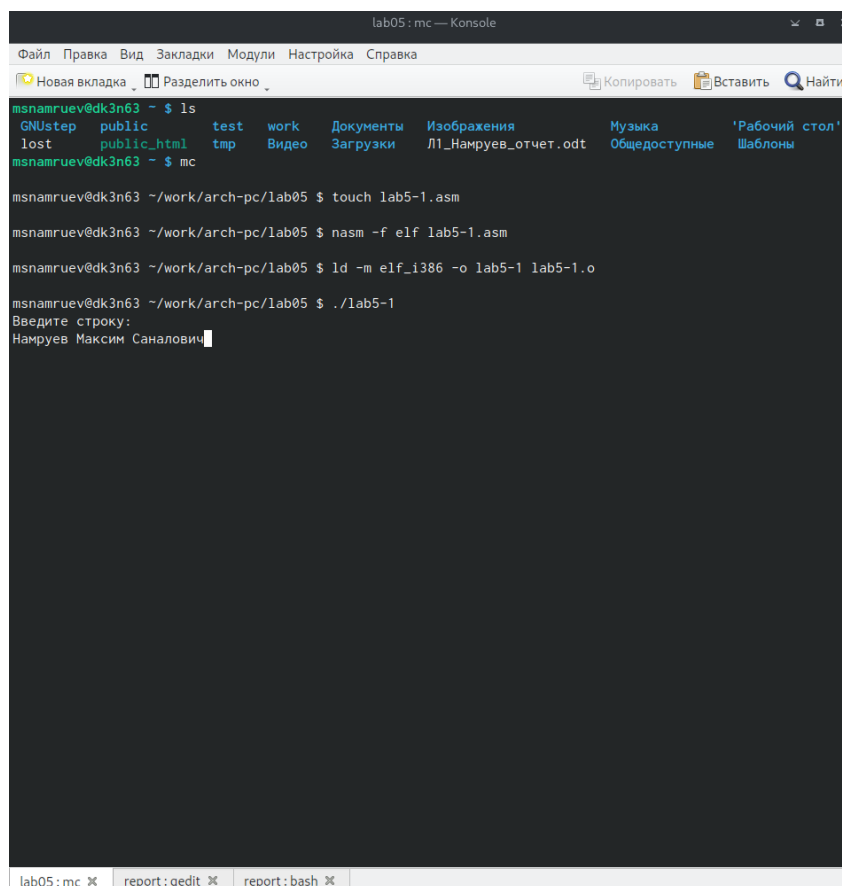
```
lab05: mc — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
Новая вкладка  Разделить окно  Копировать  Вставить  Найти
/afs/.dk.sci.pfu.edu.ru/home/m/s/ms~ruev/work/arch-pc/lab05/lab5-1.asm  2431/2431  100%

; Программа вывода сообщения на экран и ввода строки с клавиатуры
;----- Объявление переменных -----
SECTION .data ; Секция иницированных данных
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 ;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

1Помощь 2Разверн 3Выход 4Hex 5Терейти 6 7Поиск 8Исходный 9Формат 10Выход
lab05: mc  report:gedit  report:bash
```

Рис. 4.7: Просмотр файла

Далее оттранслирую текст программы lab5-1.asm в объектный файл, выполняю компоновку объектного файла и запускаю получившийся исполняемый файл. Ввожу своё ФИО.(рис. [4.8]).



```
lab05: mc — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
Новая вкладка  Разделить окно  Копировать  Вставить  Найти
msnamruev@dk3n63 ~ $ ls
GNUstep  public  test  work  Документы  Изображения  Музыка  'Рабочий стол'
lost     public_html  tmp  Видео  Загрузки  Л1_Намуев_отчет.odt  Общедоступные  Шаблоны
msnamruev@dk3n63 ~ $ mc

msnamruev@dk3n63 ~/work/arch-pc/lab05 $ touch lab5-1.asm

msnamruev@dk3n63 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1.asm

msnamruev@dk3n63 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1 lab5-1.o

msnamruev@dk3n63 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:
Намуев Максим Саналович
```

Рис. 4.8: Запуск файла

4.1 Подключение внешнего файла

Скачиваю файл in_out.asm со страницы курса в ТУИС.(рис. [4.9]).

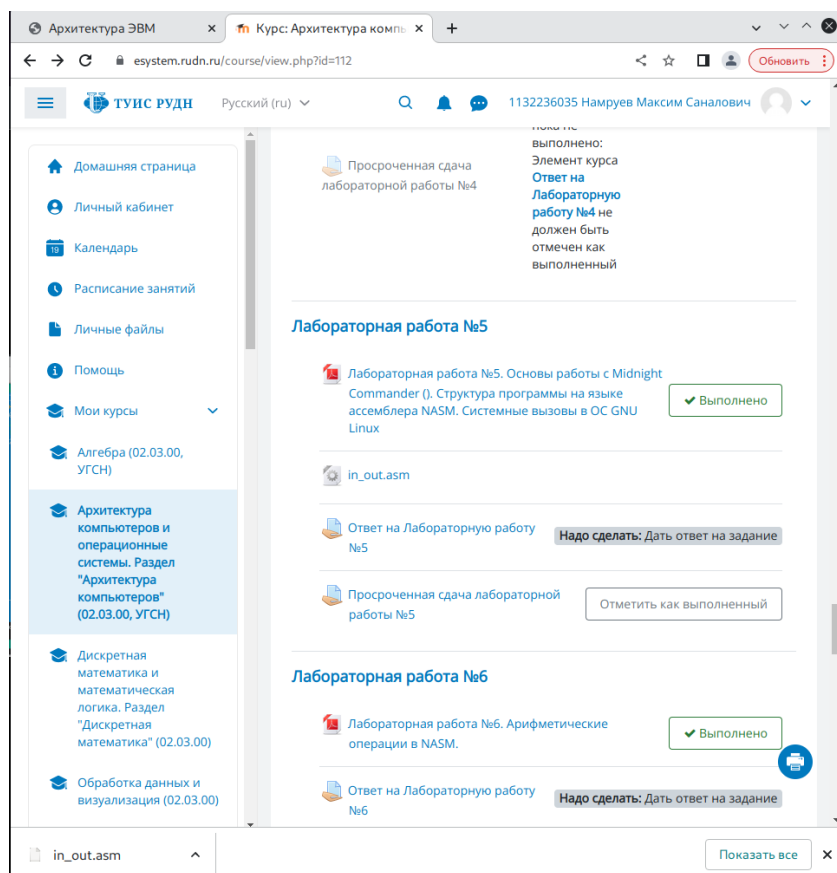


Рис. 4.9: Скачивание файла

В одной панели mc открываю каталог с файлом lab5-1.asm. В другом открываю каталог со скаченным файлом in_out.asm. Копирую файл in_out.asm в каталог с файлом lab5-1.asm с помощью функциональной клавиши F5. (рис. [4.10]).

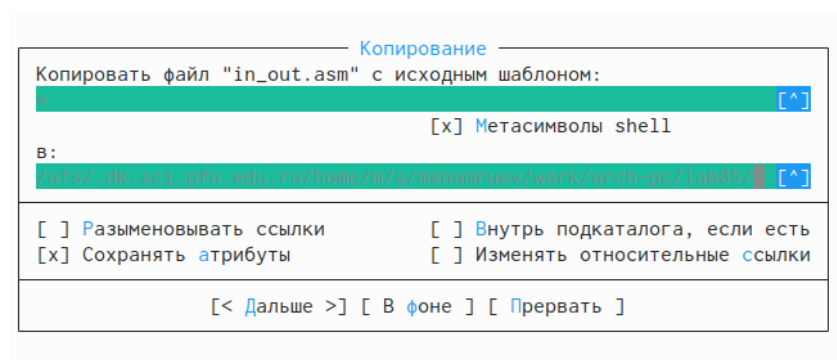


Рис. 4.10: Копирование файла

С помощью функциональной клавиши f6 создаю копию файла lab5-1.asm с именем lab5-2.asm.(рис. [4.11]).

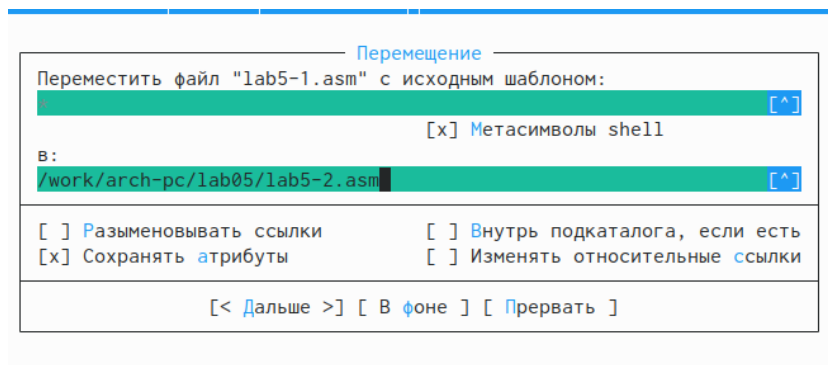


Рис. 4.11: Копирование файла

Исправляю текст программы в файле lab5-2.asm с использованием подпрограмм из внешнего файла in_out.asm. Далее создаю исполняемый файл и проверяю его работу.(рис. [4.12]).(рис. [4.13]).

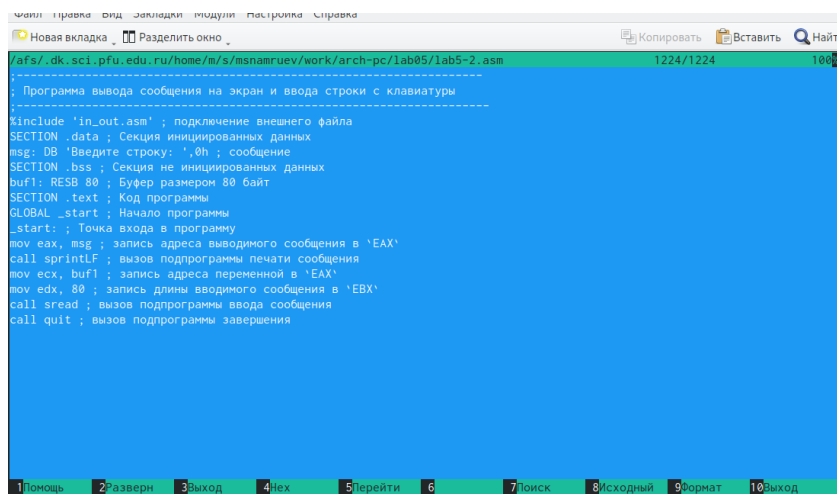


Рис. 4.12: Исправление файла

```
msnamruev@dk3n63 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
msnamruev@dk3n63 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
msnamruev@dk3n63 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку:
Насруев Максим Саналович
```

Рис. 4.13: Создание исполняемого файла

После того как я изменил `sprintLF` на `sprint` и ещё раз создал исполняемый файл, вводимое сообщение перестало переноситься на следующую строку.(рис. [4.14]).

```
msnamruev@dk8n75 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку: privet
```

Рис. 4.14: Запуск новой программы

5 Задание для самостоятельной работы

Создаю копию файла lab5-1.asm с помощью клавиши F5.(рис. [5.1]).

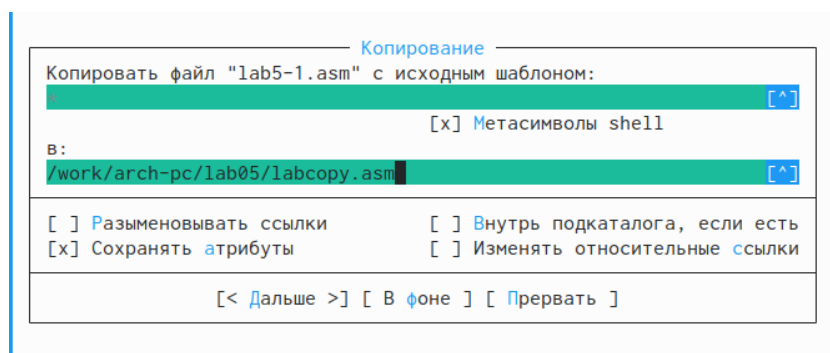


Рис. 5.1: Создание копии

Далее редактирую файл labcopy.asm так, чтобы она выводила то, что вводит пользователь.(рис. [5.2]).


```
labcopy.asm      [-M--] 10 L:[ 14+17 31/ 40] *(2081/2489b) 0010 0x00A
_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
mov ebx, 1
mov ecx, buf1
mov edx, buf1
int 80h
; ----- 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
1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перем-тить 7Поиск
```

Рис. 5.2: Редактирование программы

Создаю объектный и исполняемый файлы и запускаю полученную программу.
Ввожу свою фамилию и программа выводит мне её обратно.(рис. [5.3]).

```
msnamruev@dk8n75 ~/work/arch-pc/lab05 $ nasm -f elf labcopy.asm
msnamruev@dk8n75 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o labcopy labcopy.o
msnamruev@dk8n75 ~/work/arch-pc/lab05 $ ./labcopy
Введите строку:
Намруев
Намруев
msnamruev@dk8n75 ~ $
```

Рис. 5.3: Проверка программы

После этого создаю другую копию файла lab5-2.asm и называю его labcopy2.asm.(рис. [5.4]).

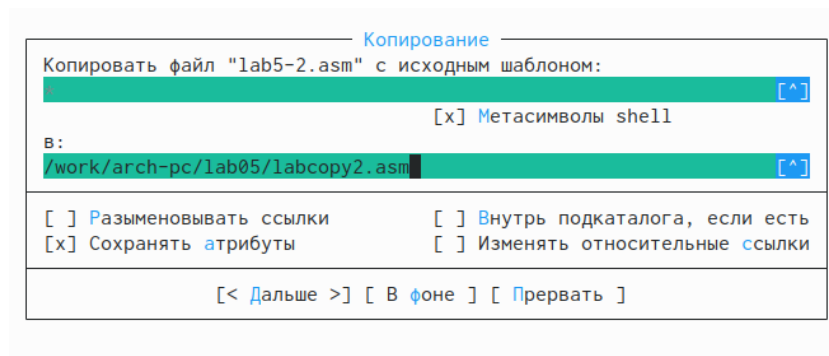


Рис. 5.4: Создание новой копии

Также редактирую файл чтобы он выводил строку, введенную пользователем.(рис. [5.5]).

```

;-----
; 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
mov eax,4
mov ebx,1
mov ecx,buf1
int 80h
call quit ; вызов подпрограммы завершения

```

Рис. 5.5: Редактирование нового файла

После создаю объектный и исполняемый файлы для labcopy2.asm и запускаю эту программу. Также ввожу свою фамилию и получаю её обратно.(рис. [5.6]).

```
msnamruev@dk8n78 ~ $ mc
msnamruev@dk8n78 ~/work/arch-pc/lab05 $ nasm -f elf labcopy2.asm
msnamruev@dk8n78 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o labcopy2 labcopy.o
msnamruev@dk8n78 ~/work/arch-pc/lab05 $ ./labcopy2
Введите строку:
Намруев
Намруев
```

Рис. 5.6: Проверка работы программы

6 Выводы

После выполнения данной лабораторной работы я научился навыкам работы в Midnight Commnader и основе инструкция ассемблера mov и int.