

Отчёт по лабораторной работе №55

Дисциплина: архитектура компьютеров

Симонова Виктория Игоревна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Выполнение заданий для самостоятельной работы	15
5	Выводы	19
	Список литературы	20

Список иллюстраций

4.1	1 открытый mc	9
4.2	Создание каталога	10
4.3	Создание файла	10
4.4	Редактирование файла	10
4.5	Просмотр файла	11
4.6	Трансляция файла	11
4.7	Компановка файла	11
4.8	Ввожу имя пользователя	11
4.9	Файл скопирован	12
4.10	Копирование файла lab5	13
4.11	Исправляю файл	13
4.12	Создание и запуск исполняемого файла	14
4.13	Изменение файла	14
4.14	Запуск изменённого исполняемого файла	14
4.15	Копирую файл lab5	15
4.16	Вношу изменения в копию	16
4.17	Создаю исполняемый файл	16
4.18	Запускаю исполняемый файл	16
4.19	Копирую файл lab5-2.asm	17
4.20	Изменяю файл lab5-2-1	17
4.21	Запуск файла	18

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 тс
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 (четырёх- рённое 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

##Работа с Midnight Commander Открываю Midnight Commander, введя в терминал mc (рис. [4.1]).

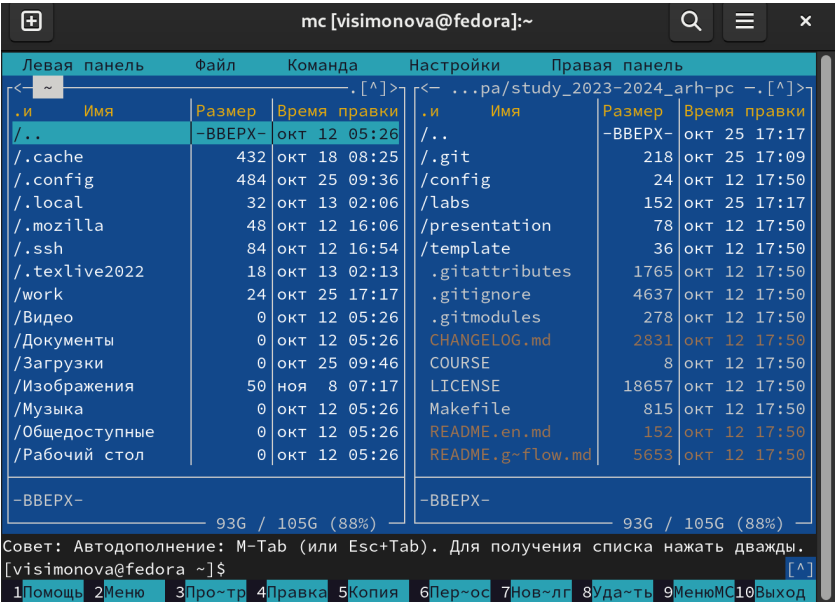


Рис. 4.1: 1 открытый mc

Создаю папку lab05 в каталоге ~/work/arch-pc (рис. [4.2]).

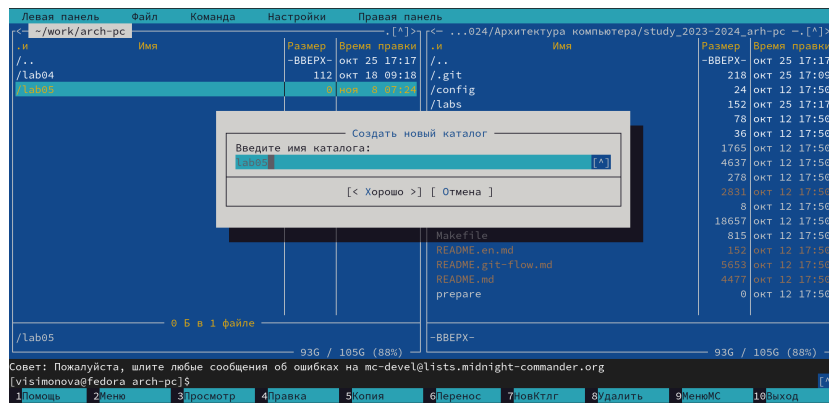


Рис. 4.2: Создание каталога

Перехожу в созданный каталог и с помощью команды touch файл lab5-1.asm (рис. [4.3]).



Рис. 4.3: Создание файла

Открываю файл lab5-1.asm для редактирования и ввожу текст программы и ввожу текст программы для запроса строки у пользователя (рис. [4.4]).

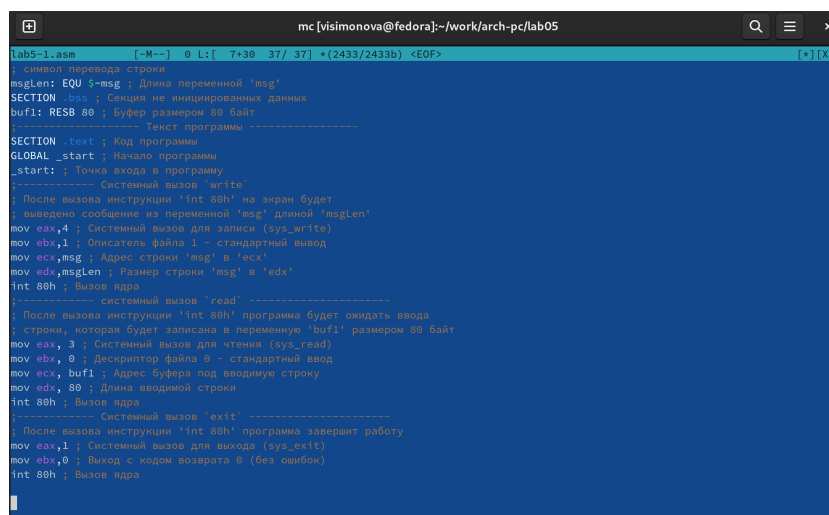
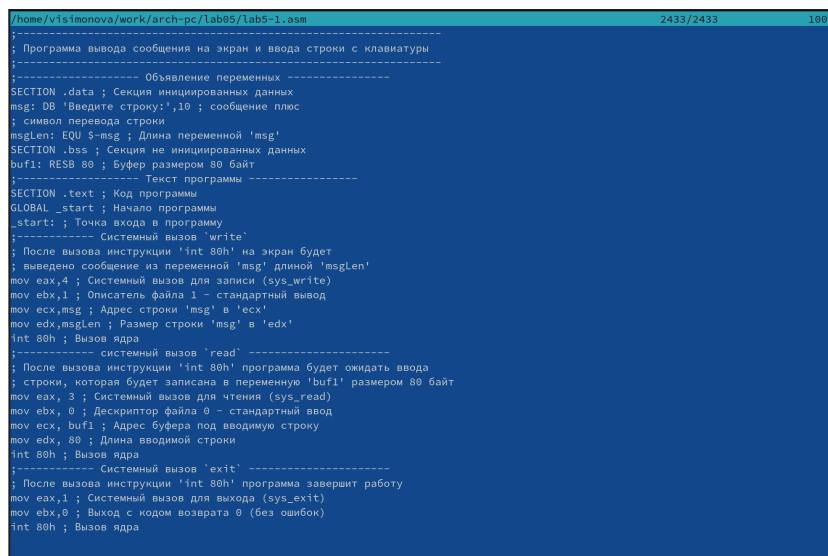


Рис. 4.4: Редактирование файла

Открываю файл lab5-1.asm для просмотра, чтобы проверить содержание текста программы в файле (рис. [4.5]).



```
;/home/visimonova/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 ; 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. 4.5: Просмотр файла

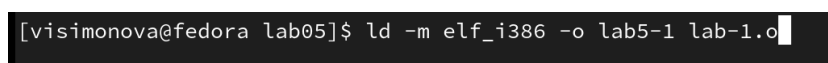
Транслирую файл lab5-1.asm (рис. [4.6]).



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

Рис. 4.6: Трансляция файла

Компанирую файл lab5-1.asm (рис. [4.7]).



```
[visimonova@fedora lab05] $ ld -m elf_i386 -o lab5-1 lab-1.o
```

Рис. 4.7: Компилирование файла

Запускаю исполняемый файл и ввожу имя пользователя с клавиатуры (рис. [4.8]).



```
[visimonova@fedora lab05] $ ./lab5-1
Введите строку:
Симонова Виктория Игоревна
[visimonova@fedora lab05] $
```

Рис. 4.8: Ввожу имя пользователя

##Подключение внешнего файла

Скачиваю файл in_out.asm из ТУИС файл,он попадает в загрузки. Копирую его в каталог lab05, тк он должен будет имрользоваться в прогамме, которая находится в этом каталоге. (рис. [4.9]).

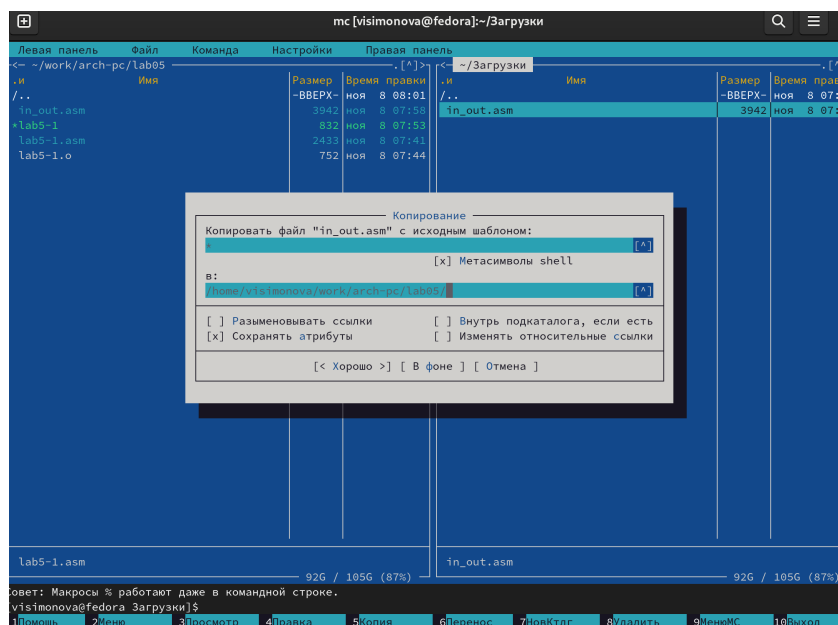


Рис. 4.9: Файл скопирован

Создаю копию файла lab5-1.asm с именем lab5-2.asm (рис. [4.10]).

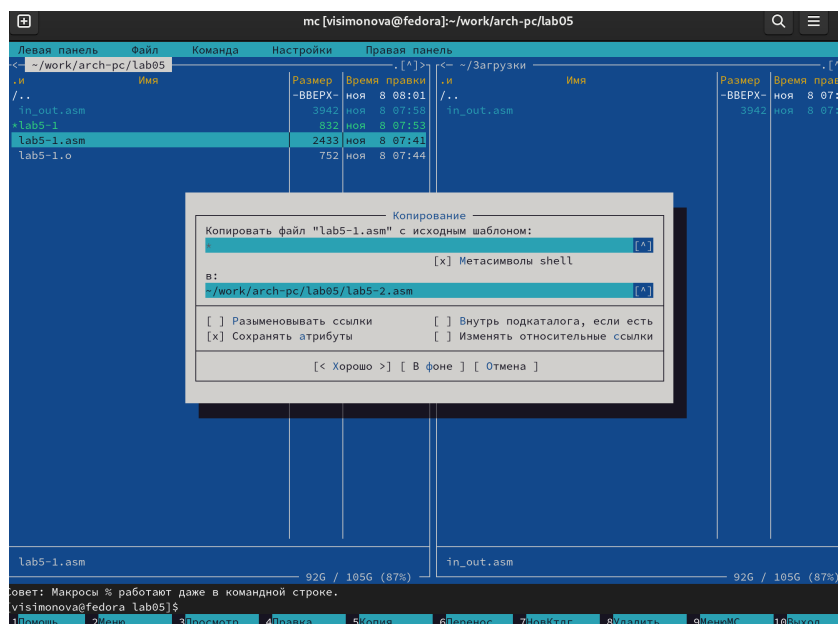


Рис. 4.10: Копирование файла lab5

Изменяю содержимое файла lab5-2.asm, используя подпрограмму из внешнего файла (рис. [4.11]).

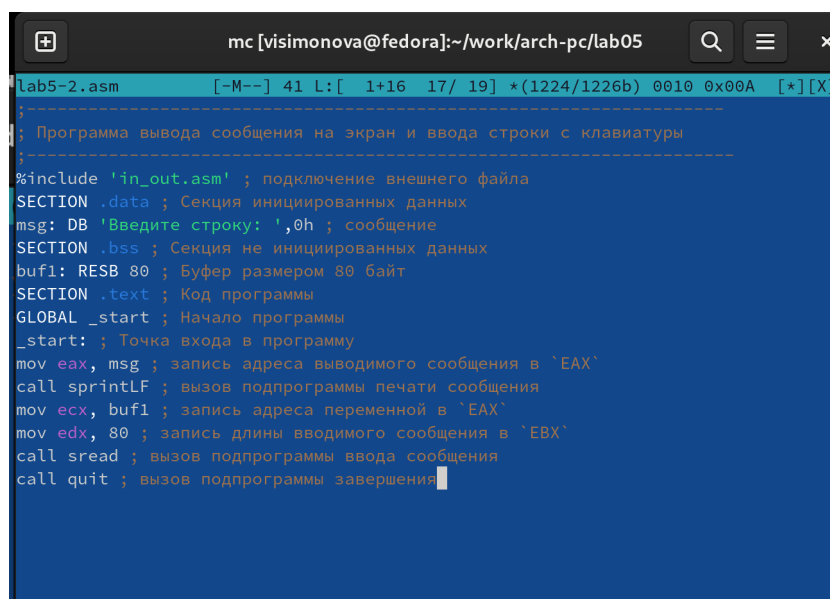


Рис. 4.11: Исправляю файл

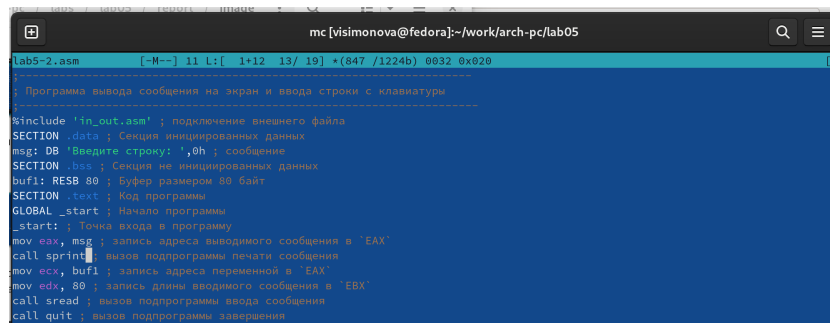
Транслирую данный файл, затем передаю на работу компановщику и запускаю

исполняемый файл, атем ввожу свои ФИО (рис. [4.12]).

```
[visimonova@fedora lab05]$ nasm -f elf lab5-2.asm
[visimonova@fedora lab05]$ ld -m elf_i386 -o lab5-2 lab5-2.o
[visimonova@fedora lab05]$ ./lab5-2
Введите строку:
Симонова Виктория Игоревна
[visimonova@fedora lab05]$
```

Рис. 4.12: Создание и запуск исполняемого файла

Изменяю в файле lab5-2.asm подпрограмму sprintLF на sprint.(рис. [4.13]).



```
lab5-2.asm  [-N--] 11 L: [ 1+12 13/ 19] *(847 /1224b) 0032 0x020 [+*]
; 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 "ECX"
mov edx, 80 ; запись длины входного сообщения в "EDX"
call read ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения
```

Рис. 4.13: Изменение файла

Транслирую данный файл, затем переда. на работу компановщику и запускаю исполняемый файл,затем ввожу свои ФИО (рис. [4.14]).

```
nasm: /home/visimonova/work/arch-pc/lab05: 316 каталог
[visimonova@fedora lab05]$ nasm -f elf lab5-2.asm
[visimonova@fedora lab05]$ ld -m elf_i386 -o lab5-2-2 lab5-2.o
[visimonova@fedora lab05]$ ./lab5-2-2
Введите строку: Симонова Виктория Игоревна
[visimonova@fedora lab05]$
```

Рис. 4.14: Запуск изменённого исполняемого файла

Разница между исполняемым файлом lab5-2 и файлом 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, именно в этом заключается различие между sprintLF и sprint.

4.1 Выполнение заданий для самостоятельной работы

Создаю копию файла lab6-1.asm с именем lab6-1-1.asm (рис. [4.15]).

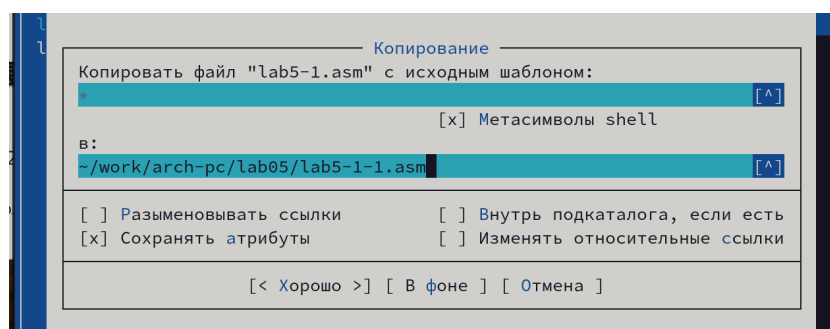


Рис. 4.15: Копирую файл lab5

Открываю данный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку с ФИО (рис. [4.16]).

```

lab5-1-1.asm  [-M--] 7 L:[ 1+34 35/ 42] *(2124/2487b) 0010 0x00A [*][X]
;
;----- 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
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)

```

Рис. 4.16: Вношу изменения в копию

Транслирую данный файл, затем передаю на работу компановщику (рис. [4.17]).

```

[visimonova@fedora lab05]$ nasm -f elf lab5-1-1.asm
[visimonova@fedora lab05]$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
[visimonova@fedora lab05]$ ./lab5-1-1

```

Рис. 4.17: Создаю исполняемый файл

Запускаю исполняемый файл, затем ввожу свои ФИО (рис. [4.18]).

```

[visimonova@fedora lab05]$ ./lab5-1-1
Введите строку:
Симонова Виктория Игоревна
Симонова Виктория Игоревна

```

Рис. 4.18: Запускаю исполняемый файл

Создаю копию файла lab5-2.asm с именем lab5-2-1.asm (рис. [4.19]).

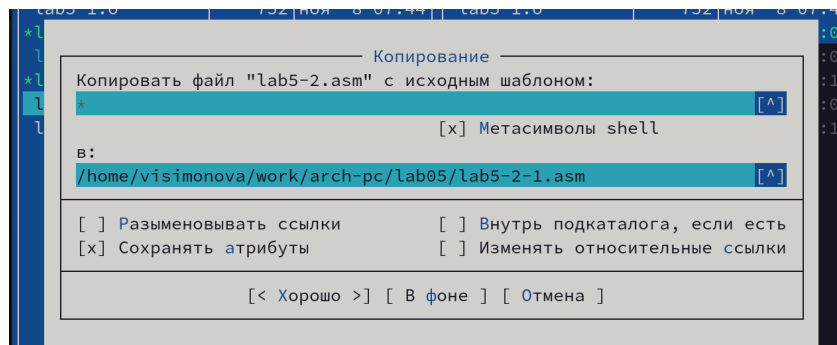


Рис. 4.19: Копирую файл lab5-2.asm

Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку (рис. [4.20]).

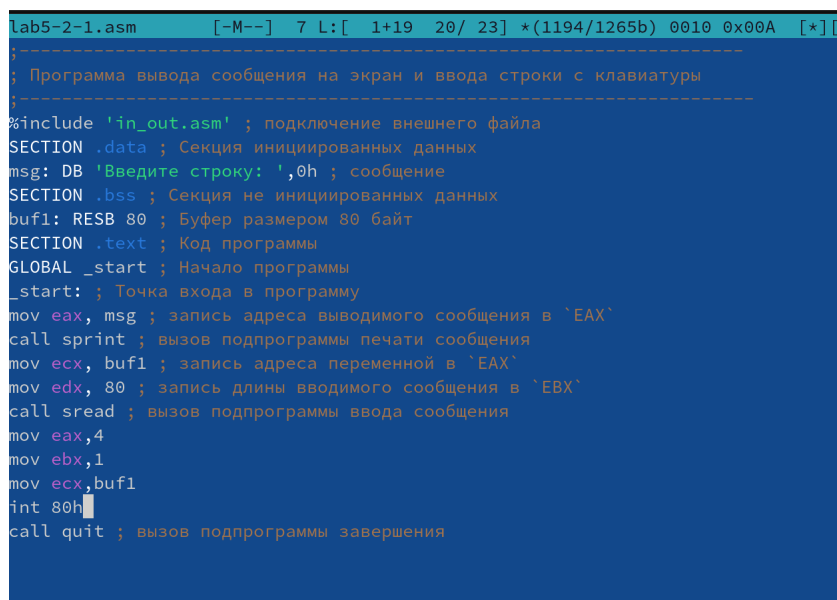


Рис. 4.20: Изменяю файл lab5-2-1

Транслирую данный файл, затем передаю на работу компановщику и запускаю исполняемый файл, затем ввожу свои ФИО (рис. [4.21]).

```
[visimonova@fedora lab05]$ nasm -f elf lab5-2-1.asm
[visimonova@fedora lab05]$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
[visimonova@fedora lab05]$ ./lab5-2-1
Введите строку: Симонова Виктория Игоревна
Симонова Виктория Игоревна
[visimonova@fedora lab05]$
```

Рис. 4.21: Запуск файла

5 Выводы

При выполнении данной лабораторной работы я приобрела практические навыки работы с Midnight Commander, освоила инструкции языка ассемблера `mov` и `int`.

Список литературы

1. Лабораторная работа №6