

Отчет по лабораторной работе №5

Дисциплина: архитектура компьютера

Панина Жанна Валерьевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Основы работы с Midnight Commander	9
4.2	Работа в NASM	11
4.3	Подключение внешнего файла	13
4.4	Задание для самостоятельной работы	15
5	Выводы	20
	Список литературы	21

Список иллюстраций

4.1	Интерфейс Midnight Commander	9
4.2	Открытый каталог arch-rc	10
4.3	Создание рабочего подкаталога	10
4.4	Создание файла в Midnight Commander	11
4.5	Редактирование файла в Midnight Commander	11
4.6	Проверка сохранения сделанных изменений	12
4.7	Трансляция, компоновка и последующий запуск программы . . .	12
4.8	Копирование файла в рабочий каталог	13
4.9	Создание копии файла в Midnight Commander	14
4.10	Изменение программы	14
4.11	Запуск измененной программы	15
4.12	Запуск измененной программы с другой подпрограммой	15
4.13	Редактирование копии	16
4.14	Запуск своей программы	16
4.15	Редактирование копии	18
4.16	Запуск своей программы	18

Список таблиц

1 Цель работы

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

2 Задание

1. Основы работы с mc
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 (четырёхбайтное 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

4.1 Основы работы с Midnight Commander

Открываю Midnight Commander (рис. -fig. 4.1).

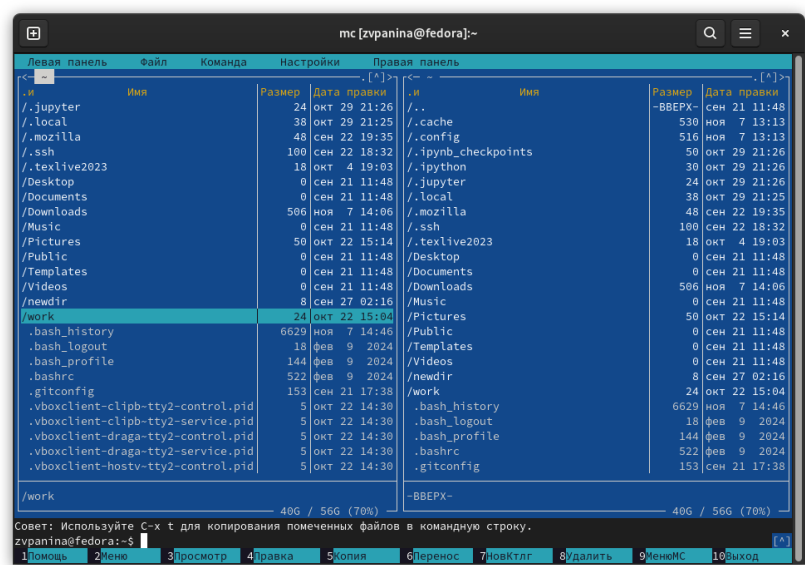


Рис. 4.1: Интерфейс Midnight Commander

Перехожу в созданный каталог в предыдущей лабораторной работе (рис. - fig. 4.2).

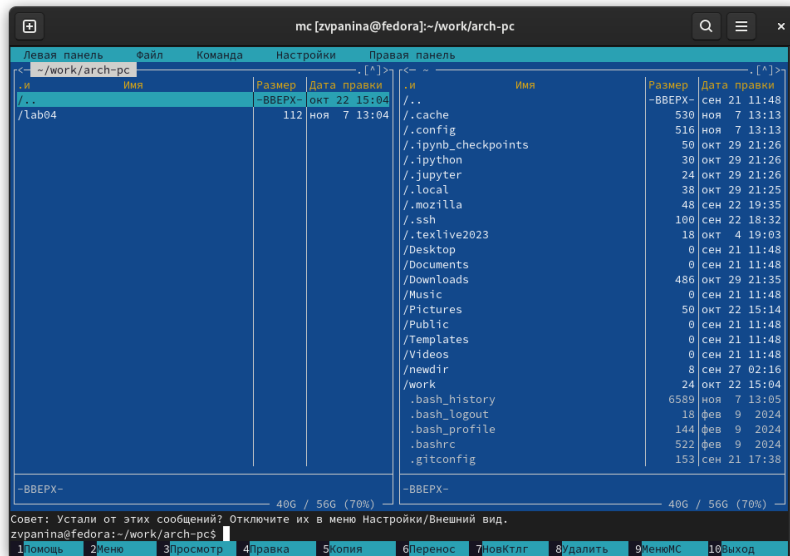


Рис. 4.2: Открытый каталог arch-pc

С помощью функциональной клавиши, я создаю подкаталог lab05, в котором буду работать (рис. -fig. 4.3).

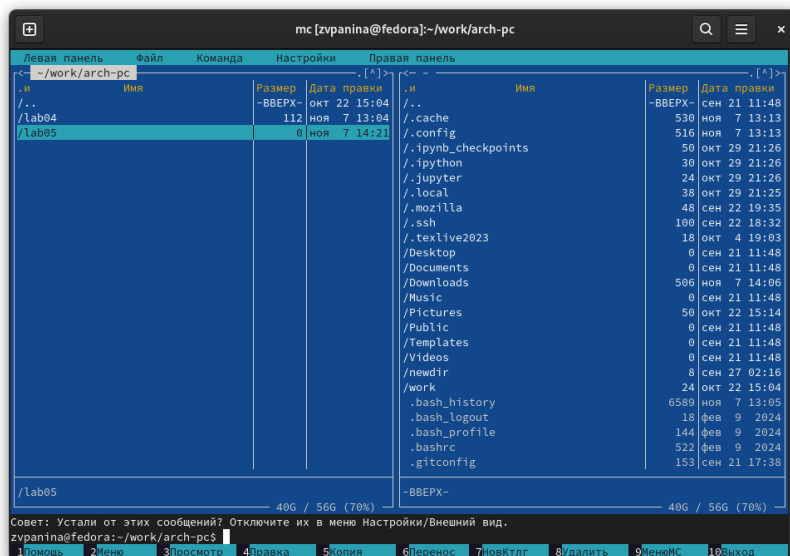


Рис. 4.3: Создание рабочего подкаталога

В строке ввода ввожу команду touch и создаю файл (рис. -fig. 4.4).

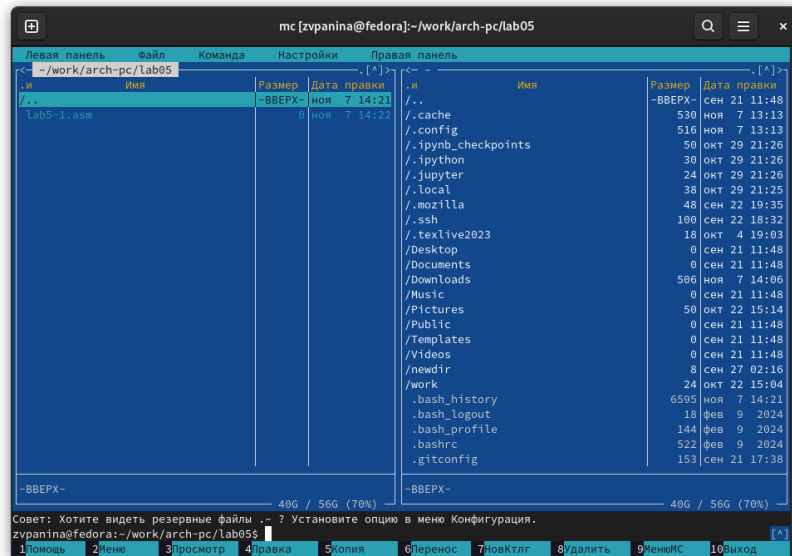


Рис. 4.4: Создание файла в Midnight Commander

4.2 Работа в NASM

С помощью F4 открываю только что созданный файл и вношу код с листинга (рис. -fig. 4.5).

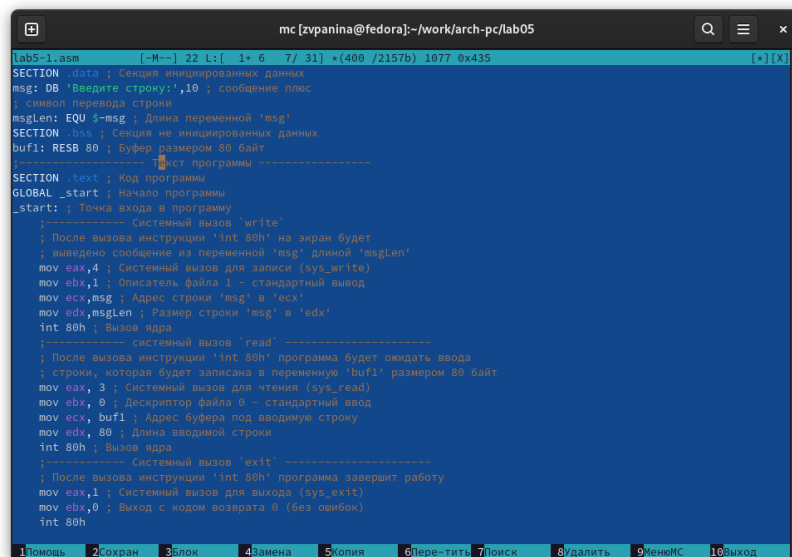
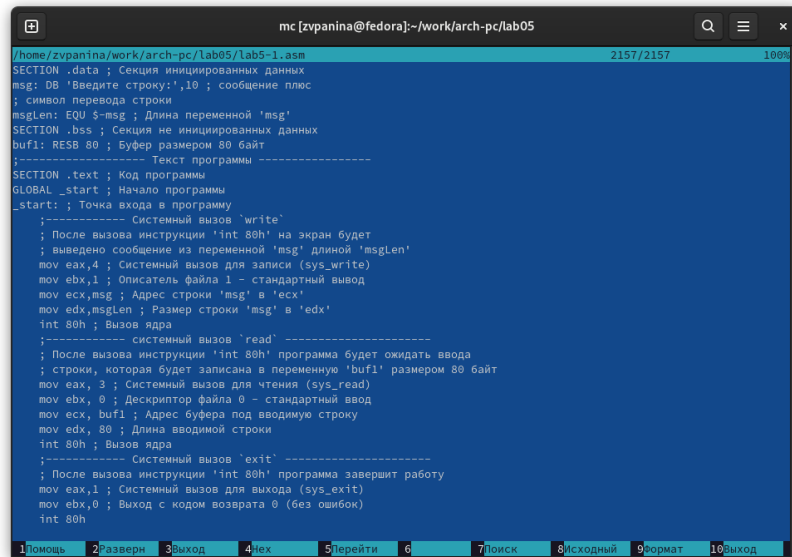


Рис. 4.5: Редактирование файла в Midnight Commander

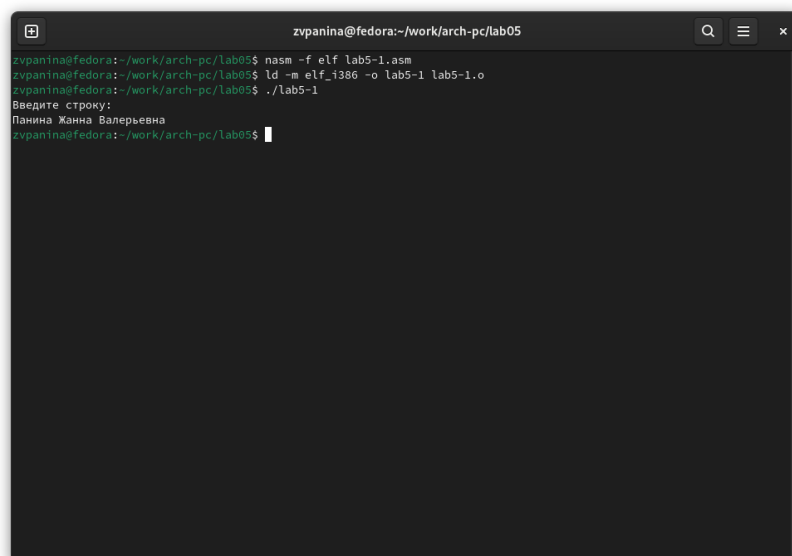
Проверяю сохраненные изменения с помощью клавиши F3 (рис. -fig. 4.6).



```
mc [zvpanina@fedora]:~/work/arch-pc/lab05
/home/zvpanina/work/arch-pc/lab05/lab5-1.asm 2157/2157 100%
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 ; Описание 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
1Помощь 2Разверн 3Выход 4Мен 5Перейти 6 7Поиск 8Исходный 9Формат 10Выход
```

Рис. 4.6: Проверка сохранения сделанных изменений

Транслирую и компоную измененный файл, запускаю (рис. -fig. 4.7).



```
zvpanina@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
zvpanina@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
zvpanina@fedora:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Панина Жанна Валерьевна
zvpanina@fedora:~/work/arch-pc/lab05$
```

Рис. 4.7: Трансляция, компоновка и последующий запуск программы

4.3 Подключение внешнего файла

Скачанный с ТУИС файл сохраняю в общую папку на своем компьютере, на виртуальной машине в интерфейсе Midnight Commander перехожу в директорию общей папки, копирую файл в рабочий подкаталог. (рис. -fig. 4.8).

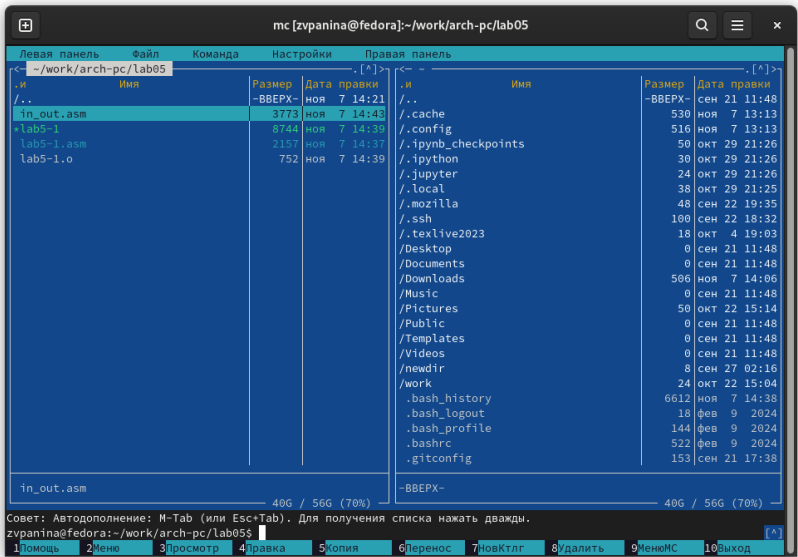


Рис. 4.8: Копирование файла в рабочий каталог

Создаю копию файла для последующей работы с ним (рис. -fig. 4.9).

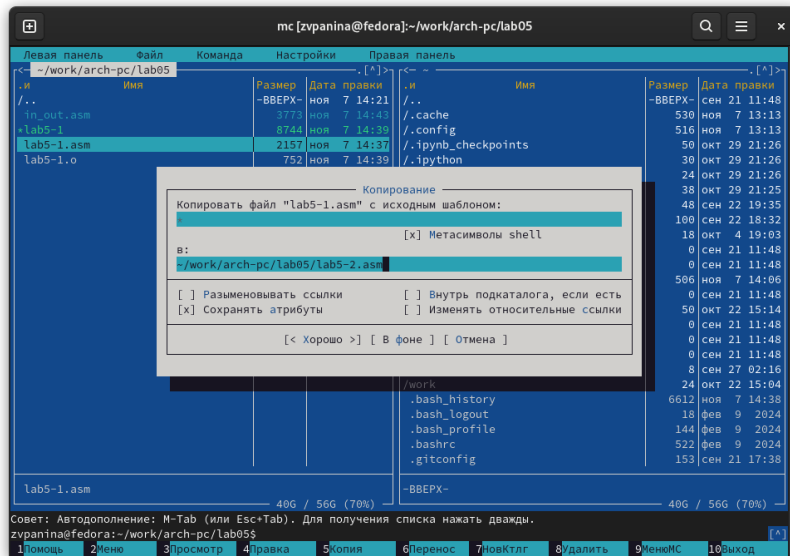


Рис. 4.9: Создание копии файла в Midnight Commander

В копии файла подключаю подпрограмм из подключенного файла (рис. -fig. 4.10).

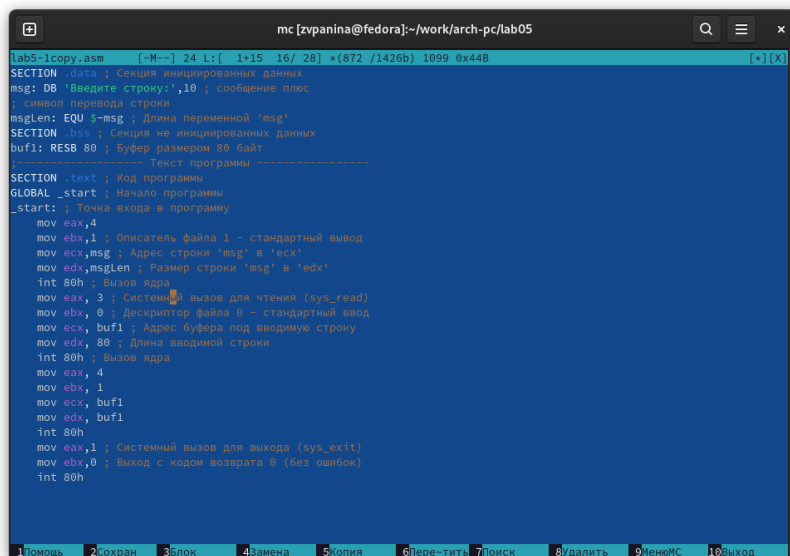
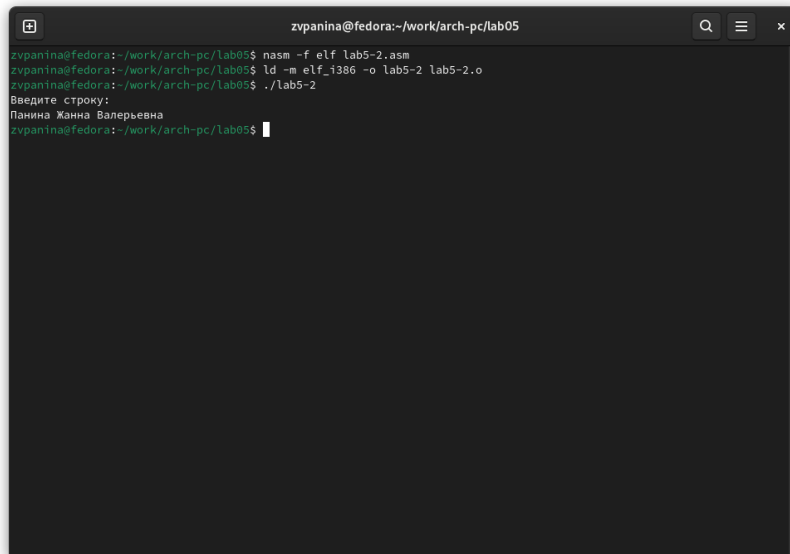


Рис. 4.10: Изменение программы

Транслирую, компоную и запускаю программу с подключенным файлом (рис. -fig. 4.11).

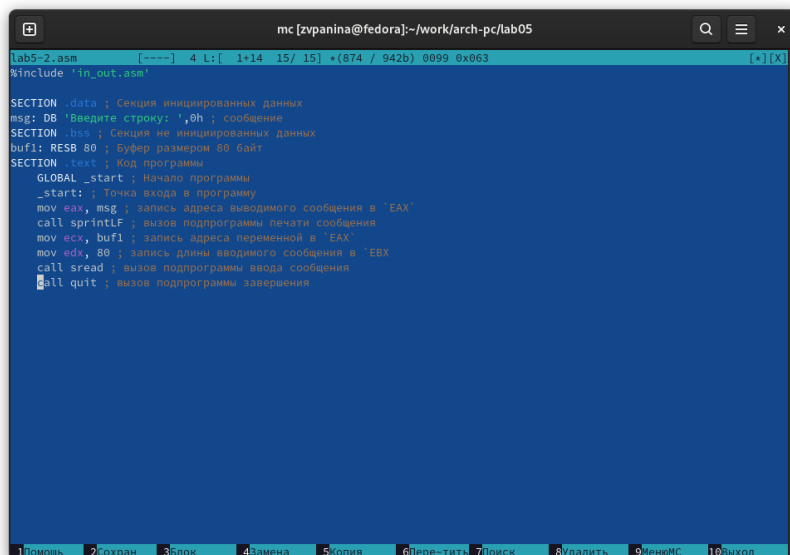


```
zvpanina@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
zvpanina@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
zvpanina@fedora:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Панина Жанна Валерьевна
zvpanina@fedora:~/work/arch-pc/lab05$
```

Рис. 4.11: Запуск измененной программы

Редактирую файл и заменяю в нем подпрограмму `sprintf` на `sprint`. Разница подпрограмм в том, что вторая вызывает ввод на той же строке (рис. -fig. 4.12).

4.4 Задание для самостоятельной работы

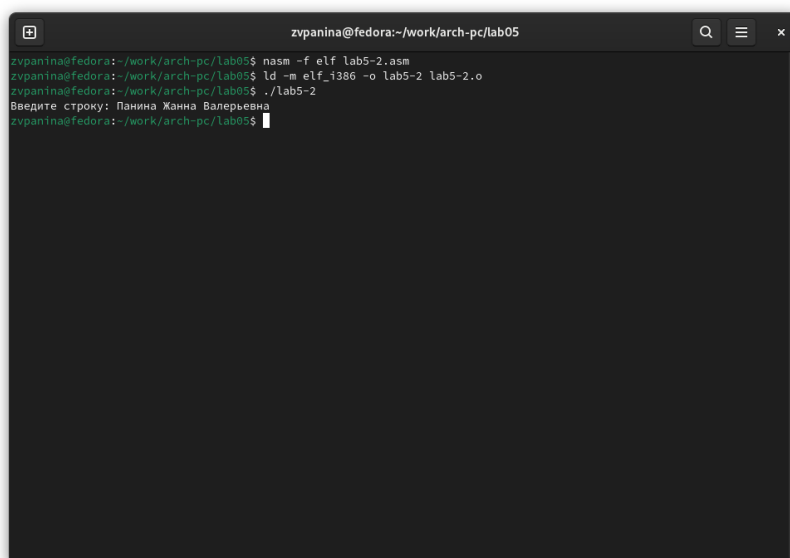


```
lab5-2.asm [----] 4 L: [ 1+14 15/ 15] x(874 / 942b) 0099 0x063 [*)(X]
#include "in_out.asm"

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 read ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения
```

Рис. 4.12: Запуск измененной программы с другой подпрограммой

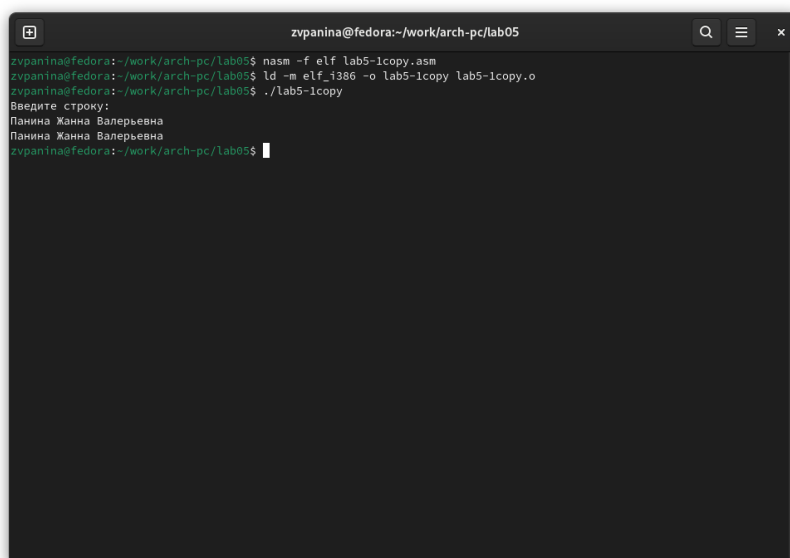
Создаю копию lab5-1.asm, редактирую так, чтобы в конце выводилась введенная мною строка с клавиатуры (рис. -fig. 4.13).



```
zvpanina@fedora:~/work/arch-pc/lab05
zvpanina@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
zvpanina@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
zvpanina@fedora:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Панина Жанна Валерьевна
zvpanina@fedora:~/work/arch-pc/lab05$
```

Рис. 4.13: Редактирование копии

Транслирую, компоную и запускаю свою программу (рис. -fig. 4.14).



```
zvpanina@fedora:~/work/arch-pc/lab05
zvpanina@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-1copy.asm
zvpanina@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1copy lab5-1copy.o
zvpanina@fedora:~/work/arch-pc/lab05$ ./lab5-1copy
Введите строку:
Панина Жанна Валерьевна
Панина Жанна Валерьевна
zvpanina@fedora:~/work/arch-pc/lab05$
```

Рис. 4.14: Запуск своей программы

Код прикладываю


```
SECTION .data
```

```
msg: DB 'Введите строку:',10
```

```
msgLen: EQU $-msg
```

```
SECTION .bss
```

```
buf1: RESB 80
```

```
SECTION .text
```

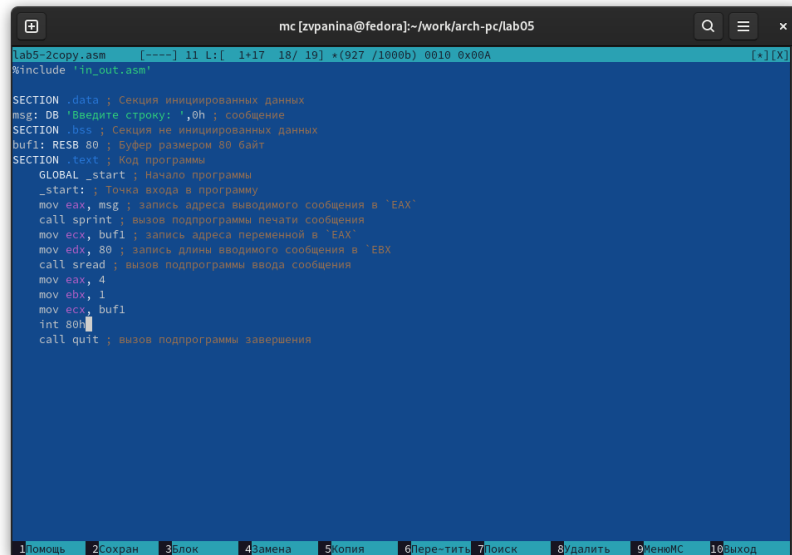
```
GLOBAL _start
```

```
_start:
```

```
    mov     eax, 4
    mov     ebx, 1
    mov     ecx, msg
    mov     edx, msgLen
    int     80h
    mov     eax, 3
    mov     ebx, 0
    mov     ecx, buf1
    mov     edx, 80
    int     80h
    mov     eax, 4
    mov     ebx, 1
    mov     ecx, buf1
    mov     edx, buf1
    int     80h
    mov     eax, 1
```

```
mov     ebx, 0
int     80h
```

Создаю копию lab5-2.asm, редактирую так, чтобы в конце выводилась введенная мною строка с клавиатуры (рис. -fig. 4.15).

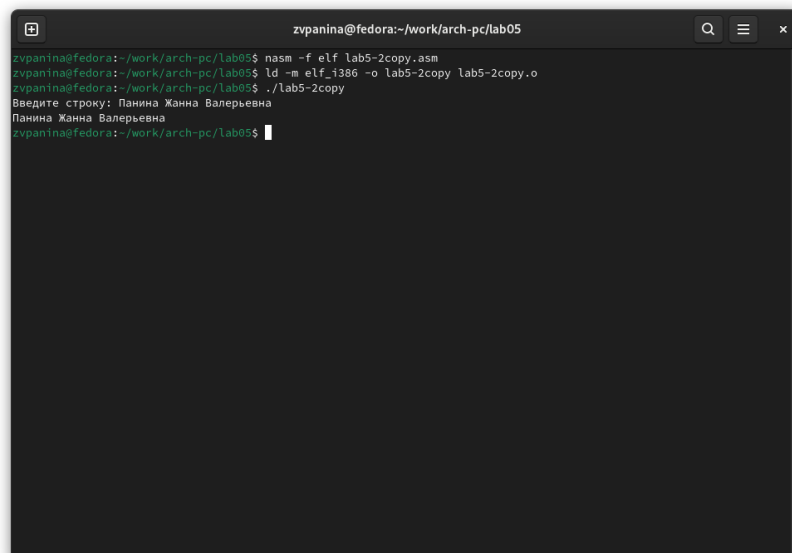


```
lab5-2copy.asm  [----] 11 L: 1+17 18/ 19] *(927 /1000b) 0010 0x00A [X]
%include 'in_out.asm'

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. 4.15: Редактирование копии

Транслирую, компоную и запускаю свою программу (рис. -fig. 4.16).



```
zspanina@fedora:~/work/arch-pc/lab05
zspanina@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2copy.asm
zspanina@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2copy lab5-2copy.o
zspanina@fedora:~/work/arch-pc/lab05$ ./lab5-2copy
Введите строку: Панина Жанна Валерьевна
Панина Жанна Валерьевна
zspanina@fedora:~/work/arch-pc/lab05$
```

Рис. 4.16: Запуск своей программы

Код прикладываю:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg: DB 'Введите строку: ', 0h
```

```
msgLen: EQU $-msg
```

```
SECTION .bss
```

```
buf1: RESB 80
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
mov eax, msg
```

```
call sprint
```

```
mov ecx, buf1
```

```
mov edx, 80
```

```
call sread
```

```
mov eax, 4
```

```
mov ebx, 1
```

```
mov ecx, buf1
```

```
int 80h
```

```
call quit
```

5 Выводы

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

Список литературы

1. Курс на ТУИС
2. Лабораторная работа №5
3. Программирование на языке ассемблера NASM Столяров А. В.