

Отчёт по лабораторной работе №5

Дисциплина: Архитектура компьютера

Осина Виктория Александровна

Содержание

1	Цель работы	5
2	Задания	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Создание каталога и файла, с которыми будет вестись дальнейшая работа	9
4.2	Освоение программы вывода сообщения на экран и ввода строки с клавиатуры	10
4.3	Подключение внешнего файла in_out.asm	14
4.4	Выполнение заданий для самостоятельной работы	18
5	Выводы	23
6	Список литературы	24

Список иллюстраций

4.1	Открытие Midnight Commander	9
4.2	Открытый Midnight Commander	9
4.3	Переход в каталог	10
4.4	Создание каталога и переход в него	10
4.5	Создание файла lab5-1.asm	10
4.6	Открытие файла для редактирования	11
4.7	Ввод текста программы	12
4.8	Сохранение изменений	12
4.9	Открытие файла для просмотра	13
4.10	Трансляция текста	13
4.11	Компановка объектного файла и запуск исполняемого файла	13
4.12	Открытие двух каталогов в панелях mc	14
4.13	Копирование in_out.asm	15
4.14	Файл теперь находится в нужном каталоге	15
4.15	Создание копии файла	16
4.16	Файл находится в нужном каталоге	16
4.17	Исправление текста программы	17
4.18	Создание исполняемого файла и проверка его работы	17
4.19	Замена программы sprintLF на sprint	18
4.20	Создание исполняемого файла и проверка работы файла	18
4.21	Создание копии файла	19
4.22	Изменение текста программы	20
4.23	Получение исполняемого файла и проверка его работы	20
4.24	Создание копии файла	21
4.25	Файл находится в нужном каталоге	21
4.26	Изменение текста программы	22
4.27	Создание исполняемого файла	22
4.28	Проверка работы исполняемого файла	22

Список таблиц

1 Цель работы

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

2 Задания

1. Создание каталога и файла, с которыми будет вестись дальнейшая работа
2. Освоение программы вывода сообщения на экран и ввода строки с клавиатуры
3. Подключение внешнего файла `in_out.asm`
4. Выполнение заданий для самостоятельной работы

3 Теоретическое введение

Midnight Commander (или просто `mc`) – это программа, которая позволяет просматривать

позволяет сделать работу с файлами более удобной и наглядной. Для активации оболочки Midnight Commander достаточно ввести в командной строке `mc` и нажать клавишу Enter. В Midnight Commander используются функциональные клавиши F1 — F10, к которым привязаны часто выполняемые операции. 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`

простых переменных и для объявления массивов. Для определения строк принято использовать директиву `DB` в связи с особенностями хранения данных в оперативной памяти. Для объявления неинициированных данных в секции `.bss` используются директивы `resb`, `resw`, `resd` и другие, которые сообщают ассемблеру, что необходимо зарезервировать заданное количество ячеек памяти. Инструкция языка ассемблера `mov` предназначена для дублирования данных источника в приёмнике. В общем виде эта инструкция записывается в виде `> mov dst,src`

Здесь операнд `dst` — приёмник, а `src` — источник. В качестве операнда могут выступать регистры (`register`), ячейки памяти (`memory`) и непосредственные

значения (const). Переслать значение из одной ячейки памяти в другую нельзя, для этого необходимо использовать две инструкции mov: >mov eax, x >mov y, eax

Также необходимо учитывать то, что размер операндов приемника и источника должны совпадать. 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. После вызова инструкции int 80h выполняется системный вызов какой-либо функции ядра Linux. При этом происходит передача управления ядру операционной системы. Чтобы узнать, какую именно системную функцию нужно выполнить, ядро извлекает номер системного вызова из регистра eax. Поэтому перед вызовом прерывания необходимо поместить в этот регистр нужный номер. Кроме того, многим системным функциям требуется передавать какие-либо параметры. По принятым в ОС Linux правилам эти параметры помещаются в порядке следования в остальные регистры процессора: ebx, ecx, edx. Если системная функция должна вернуть значение, то она помещает его в регистр eax.

4 Выполнение лабораторной работы

4.1 Создание каталога и файла, с которыми будет вестись дальнейшая работа

Открываю Midnight Commander (рис. 4.1) и (рис. 4.2).

```
[vaosina@fedora ~]$ mc
```

Рис. 4.1: Открытие Midnight Commander

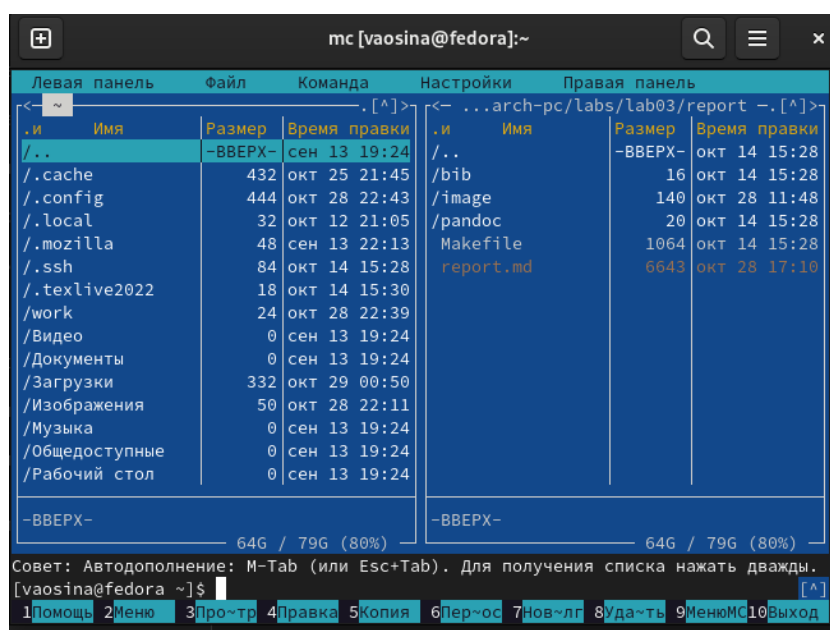


Рис. 4.2: Открытый Midnight Commander

Пользуясь клавишами **⌘**, **⌘** и Enter перехожу в каталог ~/work/arch-pc, созданный при выполнении лабораторной работы №4 (рис. 4.3).

Имя	Размер	Время правки
..	-ВВЕРХ-	окт 28 15:56
/arch-pc	10	окт 28 22:39
/study	18	окт 14 15:26

Имя	Размер	Время правки
..	-ВВЕРХ-	окт 14 15:28
/bib	16	окт 14 15:28
/image	140	окт 28 11:48
/pandoc	20	окт 14 15:28
Makefile	1064	окт 14 15:28
report.ed	6643	окт 28 17:10

Рис. 4.3: Переход в каталог

С помощью функциональной клавиши F7 создаю каталог lab05 и перехожу в него (рис. 4.4).

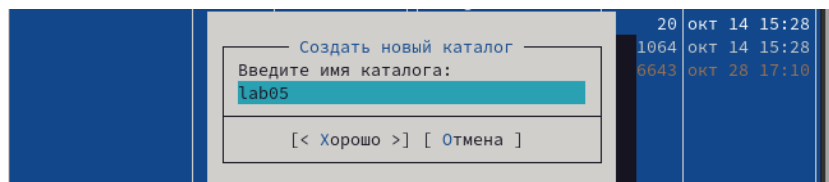


Рис. 4.4: Создание каталога и переход в него

Пользуясь строкой ввода и командой touch создаю файл lab5-1.asm (рис. 4.5).

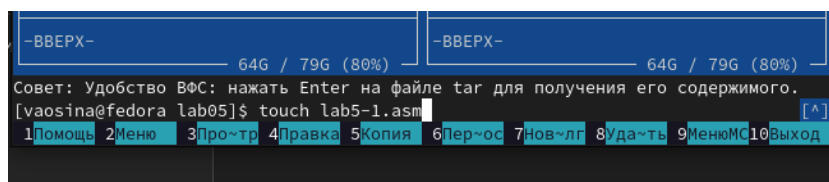


Рис. 4.5: Создание файла lab5-1.asm

4.2 Освоение программы вывода сообщения на экран и ввода строки с клавиатуры

С помощью функциональной клавиши F4 открываю файл lab5-1.asm для редактирования во встроенном редакторе (рис. 4.6).

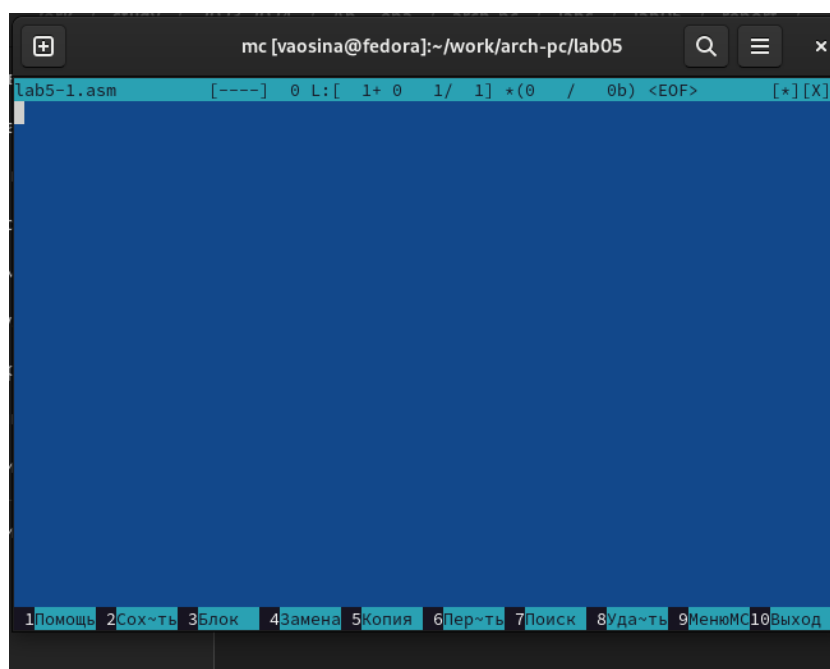
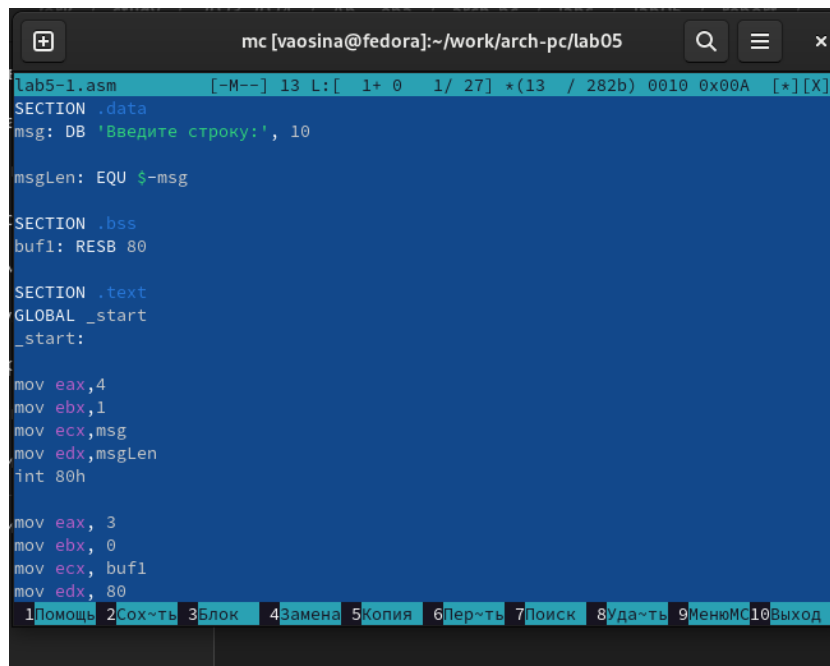


Рис. 4.6: Открытие файла для редактирования

Ввожу текст программы, сохраняю изменения и закрываю файл. (рис. 4.7) и (рис. 4.8).



```
lab5-1.asm [-M--] 13 L: [ 1+ 0 1/ 27] *(13 / 282b) 0010 0x00A [*][X]
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

1Помощь 2Сох~ть 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюМС 10Выход
```

Рис. 4.7: Ввод текста программы

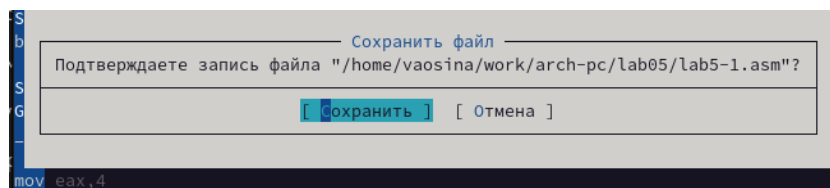
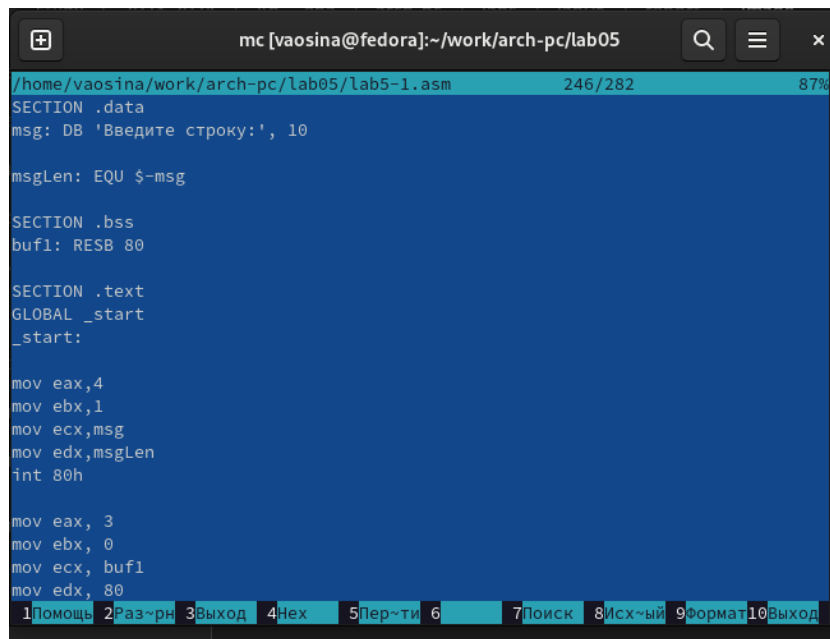


Рис. 4.8: Сохранение изменений

С помощью функциональной клавиши F3 открываю файл lab5-1.asm для просмотра. Убеждаюсь, что файл содержит текст программы. (рис. 4.9).

A screenshot of a code editor window titled 'mc [vaosina@fedora]:~/work/arch-pc/lab05'. The editor displays assembly code for 'lab5-1.asm'. The code includes sections for .data, .bss, and .text. In the .text section, it defines a global _start function. The function sets up registers (eax, ebx, ecx, edx) and uses the int 80h instruction to exit. At the bottom of the editor, there is a menu bar with options: 1Помощь, 2Раз~рн, 3Выход, 4Чех, 5Пер~ти, 6, 7Поиск, 8Исх~ый, 9Формат, 10Выход.

```
//home/vaosina/work/arch-pc/lab05/lab5-1.asm 246/282 87%
SECTION .data
msg: DB 'Введите строку:', 10

msgLen: EQU $-msg

SECTION .bss
buf1: RESB 80

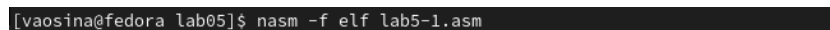
SECTION .text
GLOBAL _start
_start:

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
```

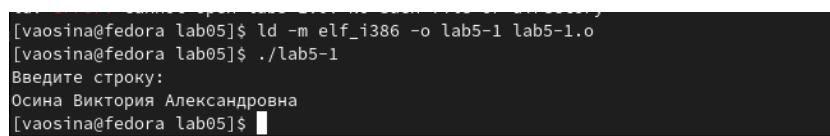
Рис. 4.9: Открытие файла для просмотра

Транслирую текст программы lab5-1.asm в объектный файл. Выполняю компоновку объектного файла и запускаю получившийся исполняемый файл. Программа выводит строку 'Введите строку:' и ожидает ввода с клавиатуры. На запрос ввожу свои ФИО (рис. 4.10) и (рис. 4.11).

A terminal window showing the command to translate the assembly file into an object file using the nasm assembler.

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

Рис. 4.10: Трансляция текста

A terminal window showing the linking of the object file into an executable using the ld linker, followed by the execution of the program. The program prompts for input, and the user enters their name: 'Осина Виктория Александровна'.

```
[vaosina@fedora lab05]$ ld -m elf_i386 -o lab5-1 lab5-1.o
[vaosina@fedora lab05]$ ./lab5-1
Введите строку:
Осина Виктория Александровна
[vaosina@fedora lab05]$
```

Рис. 4.11: Компоновка объектного файла и запуск исполняемого файла

4.3 Подключение внешнего файла in_out.asm

Скачиваю файл in_out.asm. В одной из панелей mc открываю каталог с файлом lab5-1.asm. В другой панели каталог со скаченным файлом in_out.asm (для перемещения между панелями использую Tab) (рис. 4.12).

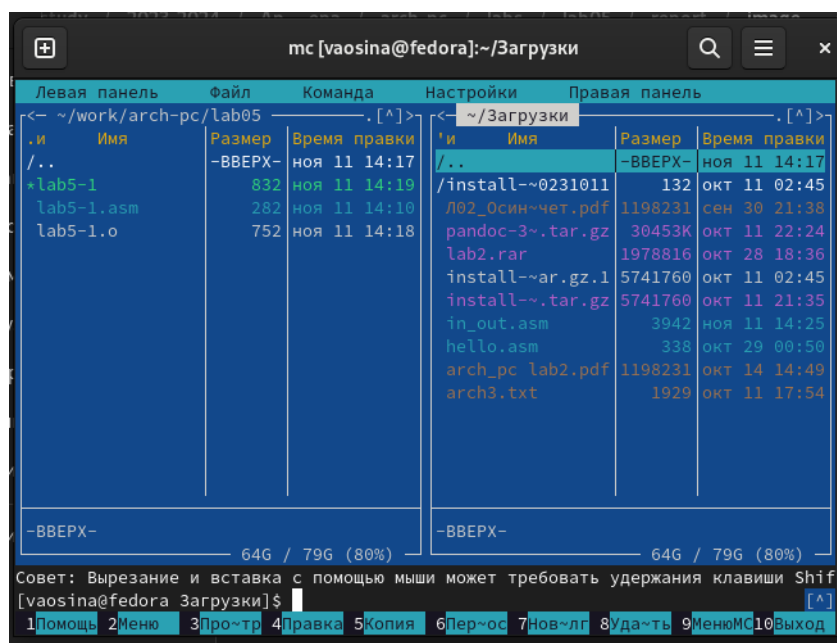


Рис. 4.12: Открытие двух каталогов в панелях mc

Копирую файл in_out.asm в каталог с файлом lab5-1.asm с помощью функциональной клавиши F5 (рис. 4.13).

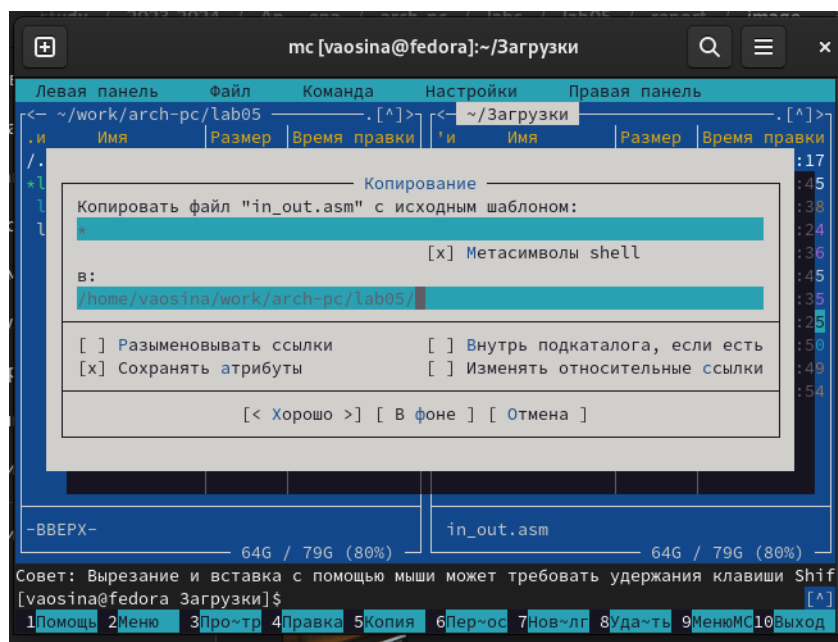


Рис. 4.13: Копирование in_out.asm

Убеждаюсь, что файл in_out.asm теперь находится в нужном каталоге (рис. 4.14).

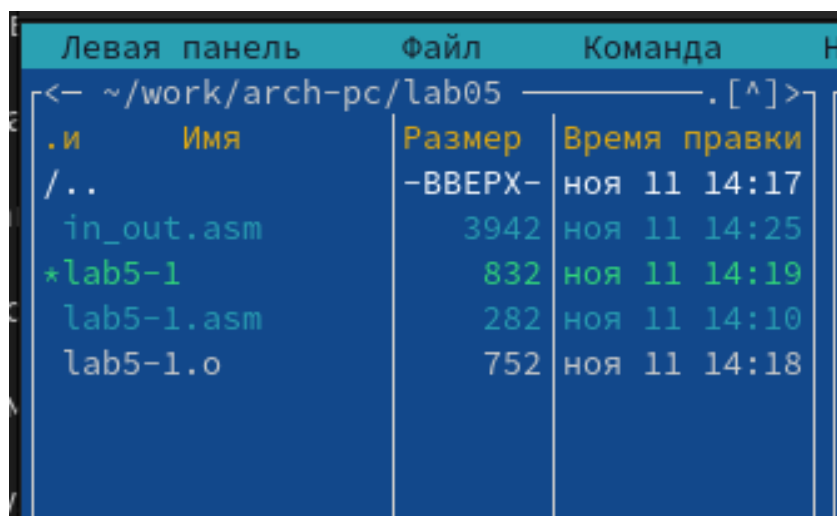


Рис. 4.14: Файл теперь находится в нужном каталоге

С помощью функциональной клавиши F5 создаю копию файла lab5-1.asm с именем lab5-2.asm и убеждаюсь в том, что скопированный файл есть в каталоге

(рис. 4.15) (рис. 4.16).

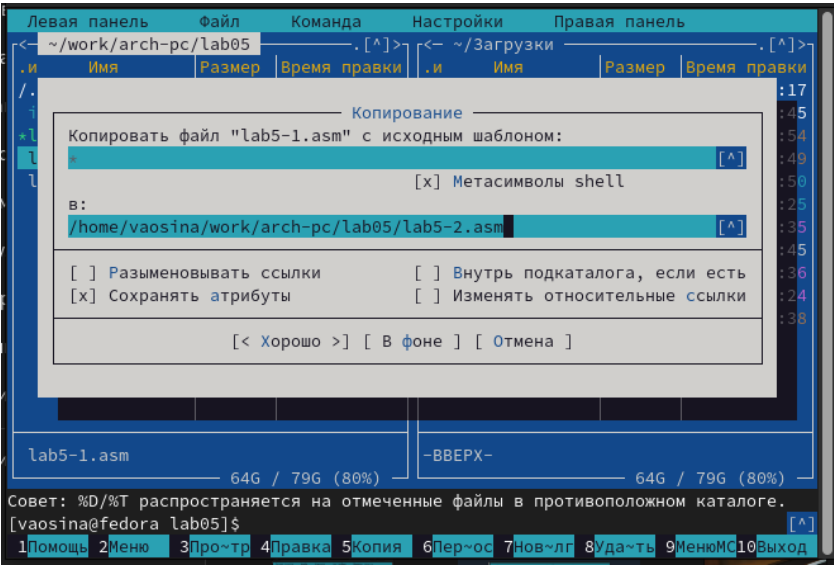
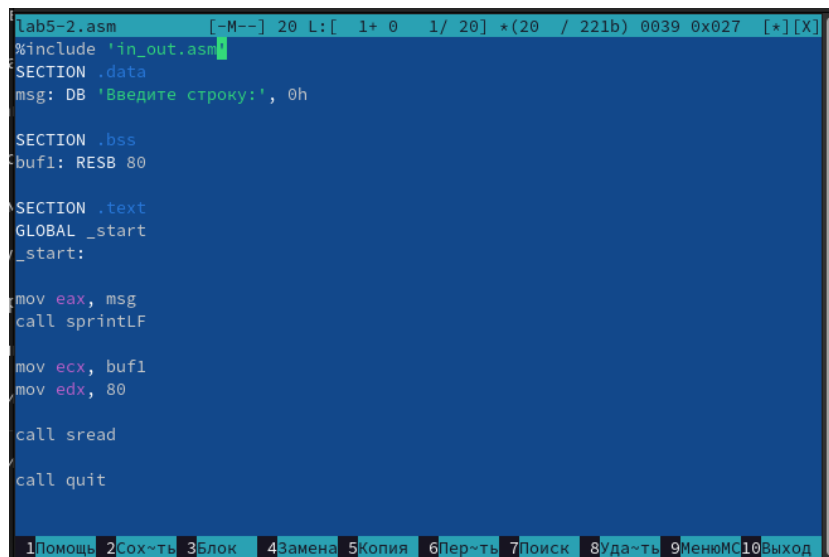


Рис. 4.15: Создание копии файла

Левая панель	Файл	Команда	На
<-	~/work/arch-pc/lab05	. [^]>	<-
.и	Имя	Размер	Время правки
/..	-ВВЕРХ-	ноя 11 14:17	/
in_out.asm	3942	ноя 11 14:25	/
*lab5-1	832	ноя 11 14:19	
lab5-1.asm	282	ноя 11 14:10	
lab5-1.o	752	ноя 11 14:18	
lab5-2.asm	282	ноя 11 14:10	

Рис. 4.16: Файл находится в нужном каталоге

Исправляю текст программы в файле lab5-2.asm с использованием подпрограмм из внешнего файла in_out.asm (использую подпрограммы sprintLF, sread и quit) (рис. 4.17).



```
lab5-2.asm [-M--] 20 L: [ 1+ 0 1/ 20] *(20 / 221b) 0039 0x027 [*][X]
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите строку:', 0h

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, msg
call sprintLF

mov ecx, buf1
mov edx, 80

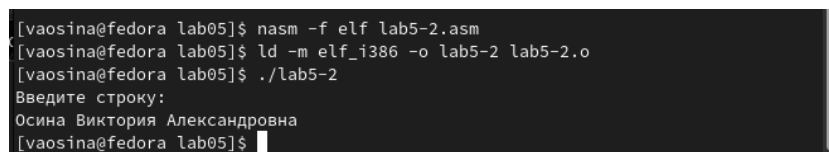
call sread

call quit
```

1Помощь 2Сох-ть 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюМС 10Выход

Рис. 4.17: Исправление текста программы

Создаю исполняемый файл и проверяю его работу. (рис. 4.18).



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

Рис. 4.18: Создание исполняемого файла и проверка его работы

В файле lab5-2.asm заменяю подпрограмму sprintLF на sprint. Создаю исполняемый файл и проверяю его работу (рис. 4.19) и (рис. 4.20).

```
lab5-2.asm      [-M--] 11 L:[ 1+12 13/ 20] *(169 / 219b) 0010 0x00A  [*][X]
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите строку:', 0h

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, msg
call sprint

mov ecx, buf1
mov edx, 80

call sread

call quit
```

Рис. 4.19: Замена программы sprintLF на sprint

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

Рис. 4.20: Создание исполняемого файла и проверка работы файла

Разница в том, что подкомманда sprintLF при выводе на экран добавляет к сообщению символ перевода строки, а sprint не добавляет, поэтому во втором случае ввод осуществляется на той же строке, где было выведено сообщение.

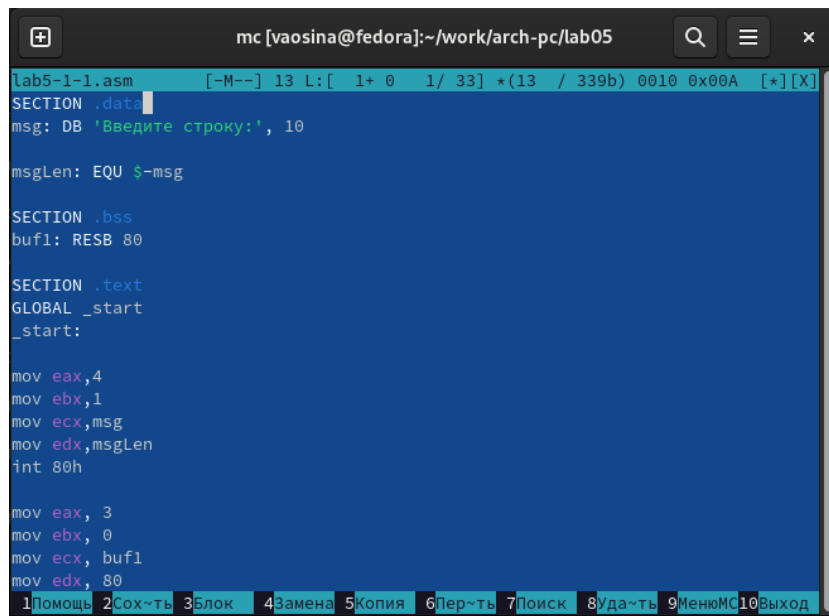
4.4 Выполнение заданий для самостоятельной работы

Создаю копию файла lab5-1.asm и называю её lab5-1-1.asm (рис. 4.21).

Левая панель		Файл	Команда
< ~/work/arch-pc/lab05			. [^]>
.и	Имя	Размер	Время правки
/..		-ВВЕРХ-	ноя 11 14:17
in_out.asm		3942	ноя 11 14:25
*lab5-1		832	ноя 11 14:19
lab5-1-1.asm		339	ноя 11 16:41
lab5-1.asm		282	ноя 11 14:10
lab5-1.o		752	ноя 11 14:18
*lab5-2		1396	ноя 11 16:37
lab5-2.asm		219	ноя 11 16:36
lab5-2.o		1312	ноя 11 16:37

Рис. 4.21: Создание копии файла

Вношу изменения в программу (без использования внешнего файла in_out.asm), так чтобы она работала по следующему алгоритму: * вывести приглашение типа “Введите строку:”; * ввести строку с клавиатуры; * вывести введённую строку на экран (рис. 4.22).



```
lab5-1-1.asm [-M--] 13 L: [ 1+ 0 1/ 33] *(13 / 339b) 0010 0x00A [*][X]
SECTION .data
msg: DB 'Введите строку:', 10

msgLen: EQU $-msg

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

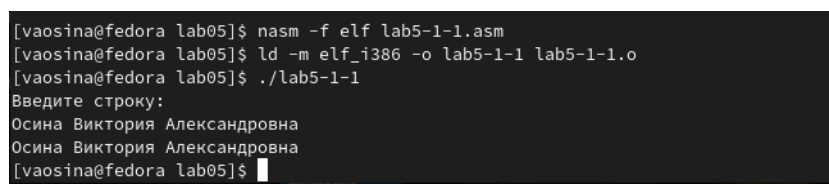
mov eax, 4
mov ebx, 1
mov ecx, msg
mov edx, msgLen
int 80h

mov eax, 3
mov ebx, 0
mov ecx, buf1
mov edx, 80

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9Меню 10Выход
```

Рис. 4.22: Изменение текста программы

Получаю исполняемый файл и проверяю его работу. На приглашение ввести строку ввожу свои ФИО (рис. 4.23).



```
[vaosina@fedora lab05]$ nasm -f elf lab5-1-1.asm
[vaosina@fedora lab05]$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
[vaosina@fedora lab05]$ ./lab5-1-1
Введите строку:
Осина Виктория Александровна
[vaosina@fedora lab05]$
```

Рис. 4.23: Получение исполняемого файла и проверка его работы

Создаю копию файла lab5-2.asm и называю её lab5-2-2.asm (рис. 4.24) и (рис. 4.25).

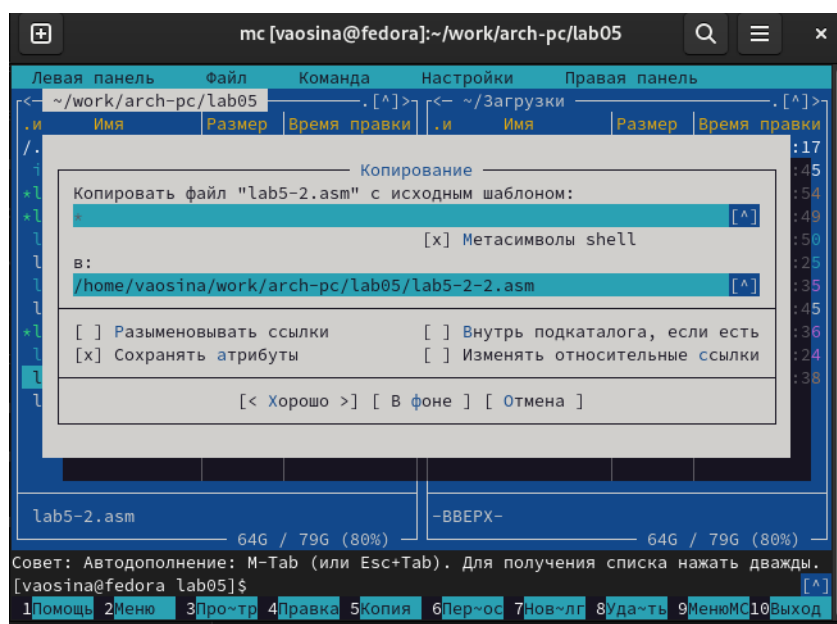


Рис. 4.24: Создание копии файла

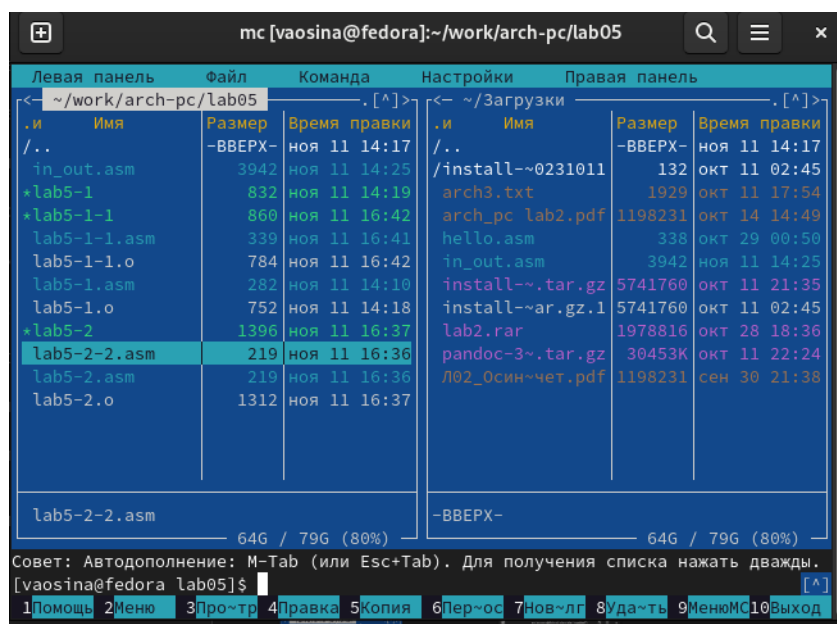
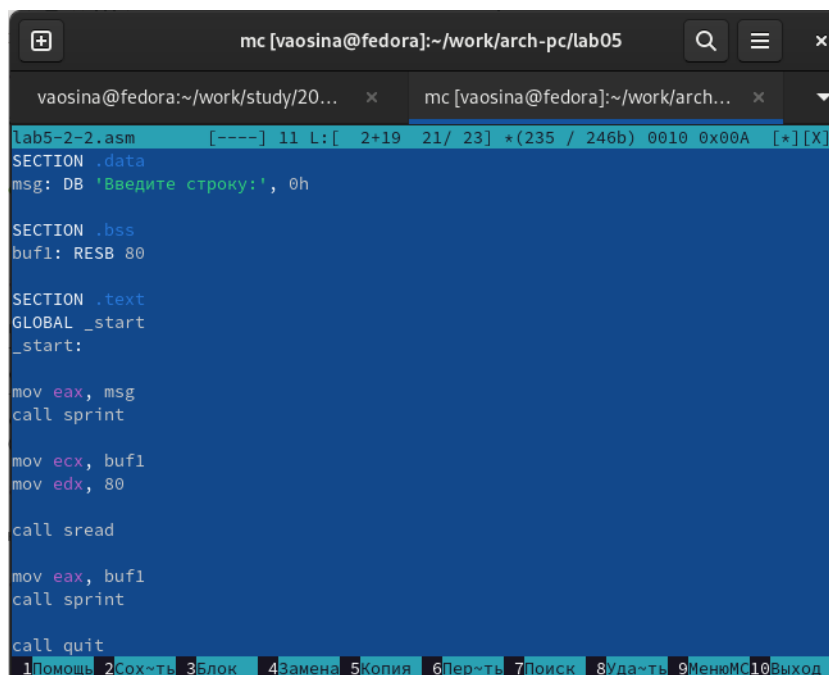


Рис. 4.25: Файл находится в нужном каталоге

Исправляю текст программы с использованием подпрограмм из внешнего файла `in_out.asm`, так чтобы она работала по следующему алгоритму: * вывести пригла-

шение типа “Введите строку:”; * ввести строку с клавиатуры; * вывести введенную строку на экран. (рис. 4.26).



```
lab5-2-2.asm [----] 11 L: [ 2+19 21/ 23] *(235 / 246b) 0010 0x00A [*][X]
SECTION .data
msg: DB 'Введите строку:', 0h

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, msg
call sprint

mov ecx, buf1
mov edx, 80

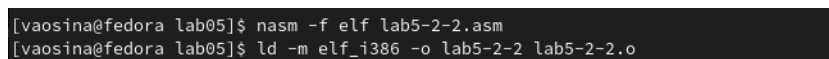
call sread

mov eax, buf1
call sprint

call quit
1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9Меню 10Выход
```

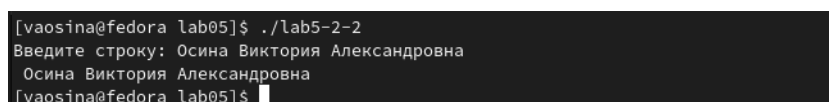
Рис. 4.26: Изменение текста программы

Создаю исполняемый файл и проверяю его работу.(рис. 4.27) и (рис. 4.28).



```
[vaosina@fedora lab05]$ nasm -f elf lab5-2-2.asm
[vaosina@fedora lab05]$ ld -m elf_i386 -o lab5-2-2 lab5-2-2.o
```

Рис. 4.27: Создание исполняемого файла



```
[vaosina@fedora lab05]$ ./lab5-2-2
Введите строку: Осина Виктория Александровна
Осина Виктория Александровна
[vaosina@fedora lab05]$
```

Рис. 4.28: Проверка работы исполняемого файла

5 Выводы

Я приобрела практические навыки работы в Midnight Commander и освоила инструкции языка ассемблера `mov` и `int`.

6 Список литературы

1. Архитектура ЭВМ