

Отчёт по лабораторной работе №5

Дисциплина: архитектура компьютера

Мирзоян Ян Игоревич

Содержание

1	Цель работы	6
2	Задание	7
3	Теоретическое введение	8
4	Выполнение лабораторной работы	9
4.1	Основы работы с тс	9
4.2	Структура программы на языке ассемблера NASM	11
4.3	Подключение внешнего файла	12
4.4	Выполнение заданий для самостоятельной работы	16
5	Выводы	21
	Список литературы	22

Список иллюстраций

4.1	Открываю Midnight Commander с помощью команды <code>mc</code>	9
4.2	Перехожу в каталог <code>arch-rc</code> используя файловый менеджер <code>mc</code> , с помощью F7 создаю каталог <code>lab05</code>	10
4.3	Перехожу в созданный каталог и прописываю команду <code>touch lab5-1.asm</code> в строке ввода, чтобы создать файл, в котором буду работать	10
4.4	С помощью функциональной клавиши F4 открываю созданный файл, Ввожу в файл код программы для запроса строки у пользователя	11
4.5	С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы	11
4.6	Транслирую текст программы файла в объектный файл командой <code>nasm -f elf lab5-1.asm</code> . Создался объектный файл <code>lab5-1.o</code> . Выполняю компоновку объектного файла с помощью команды <code>ld -m elf_i386 -o lab5-1 lab5-1.o</code> создался исполняемый файл <code>lab5-1</code>	12
4.7	Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ожидает ввода с клавиатуры, я ввожу свои ФИО, на этом программа завершает свою работу	12
4.8	Скачиваю файл <code>in_out.asm</code> со страницы курса в ТУИС. Он сохранился в каталог “Загрузки”	12
4.9	С помощью функциональной клавиши F5 копирую файл <code>in_out.asm</code> из каталога Загрузки в созданный каталог <code>lab05</code>	13
4.10	С помощью функциональной клавиши F5 копирую файл <code>lab5-1</code> в тот же каталог, но с другим именем, для этого в появившемся окне <code>mc</code> прописываю имя для копии файла	13
4.11	Изменяю содержимое файла <code>lab5-2.asm</code> во встроенном редакторе <code>mcedit</code>	14
4.12	Транслирую текст программы файла в объектный файл командой <code>nasm -f elf lab5-2.asm</code> . Создался объектный файл <code>lab5-2.o</code> . Выполняю компоновку объектного файла с помощью команды <code>ld -m elf_i386 -o lab5-2 lab5-2.o</code> Создался исполняемый файл <code>lab5-2</code> . Запускаю исполняемый файл	14
4.13	Открываю файл <code>lab5-2.asm</code> для редактирования в <code>mcedit</code> функциональной клавишей F4. Изменяю в нем подпрограмму <code>sprintLF</code> на <code>sprint</code>	15
4.14	Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл	15
4.15	Создаю копию файла <code>lab5-1.asm</code> с именем <code>lab5-1-1.asm</code> с помощью функциональной клавиши F5	16

4.16	С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку	16
4.17	Создаю объектный файл lab5-1-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-1-1, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свои ФИО, далее программа выводит введенные мною данные	18
4.18	Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5	18
4.19	С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку	19
4.20	Создаю объектный файл lab5-2-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-2-1, запускаю полученный исполняемый файл. Программа запрашивает ввод без переноса на новую строку, ввожу свои ФИО, далее программа выводит введенные мною данные	20

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 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.

4 Выполнение лабораторной работы

4.1 Основы работы с mc

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 4.1).

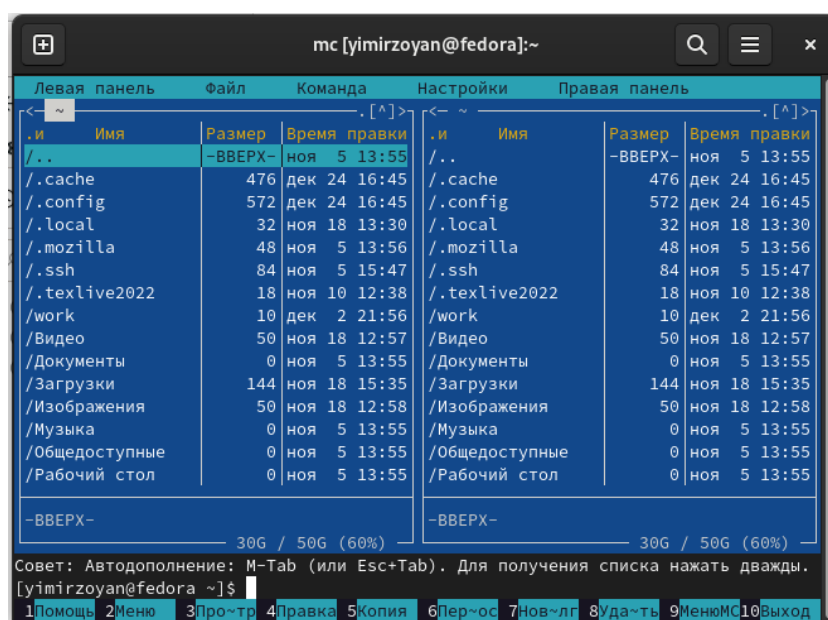


Рис. 4.1: Открываю Midnight Commander с помощью команды mc

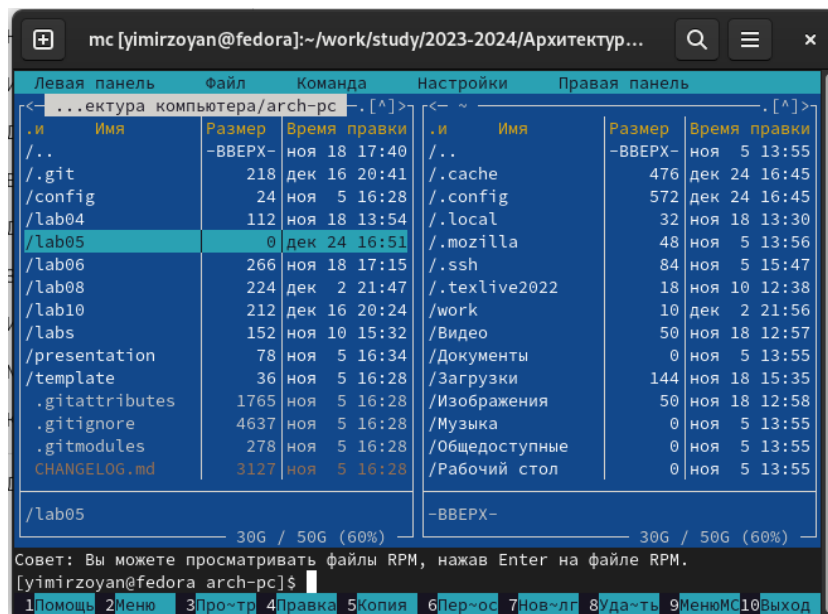


Рис. 4.2: Перехожу в каталог arch-рс используя файловы менеджер mc, с помощью F7 создаю каталог lab05

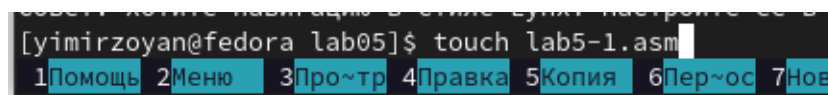
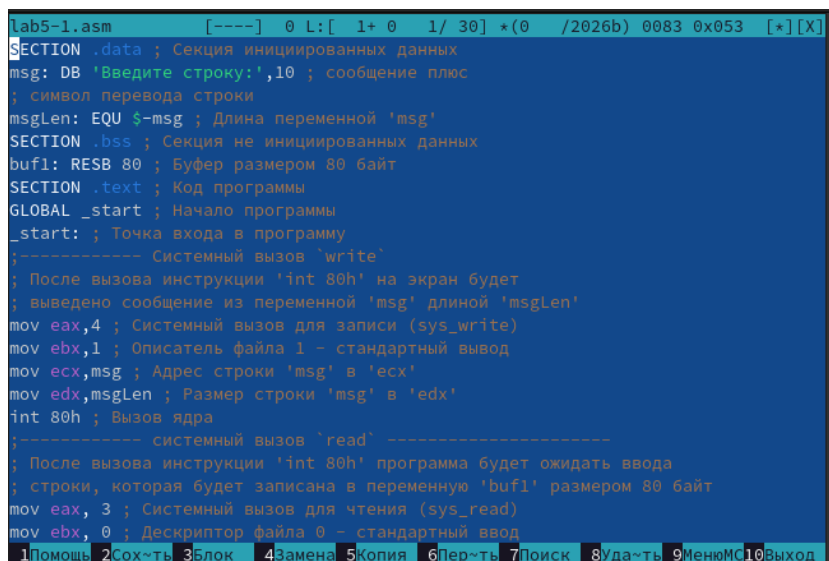


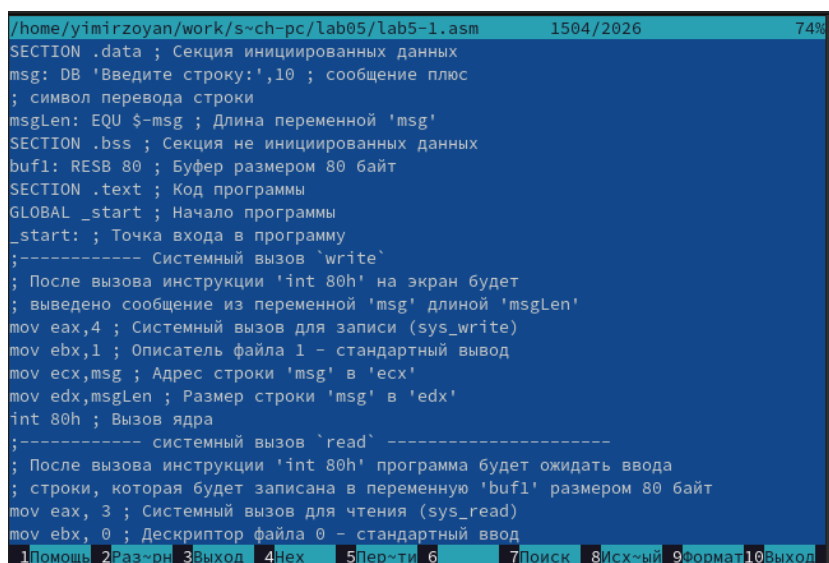
Рис. 4.3: Перехожу в созданный каталог и прописываю команду touch lab5-1.asm в строке ввода, чтобы создать файл, в котором буду работать

4.2 Структура программы на языке ассемблера NASM



```
lab5-1.asm [----] 0 L: [ 1+ 0 1/ 30] *(0 /2026b) 0083 0x053 [*][X]
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
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
1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9Меню10Выход
```

Рис. 4.4: С помощью функциональной клавиши F4 открываю созданный файл, ввожу в файл код программы для запроса строки у пользователя



```
/home/yimirzoyan/work/s-ch-pc/lab05/lab5-1.asm 1504/2026 74%
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
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
1Помощь 2Развернуть 3Выход 4Нех 5Перейти 6Поиск 7Исходный 8Формат10Выход
```

Рис. 4.5: С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы

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

Рис. 4.6: Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-1.asm`. Создался объектный файл `lab5-1.o`. Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-1 lab5-1.o` создался исполняемый файл `lab5-1`.

```
[yimirzoyan@fedora lab05]$ ./lab5-1
Введите строку:
Мирзоян Ян Игоревич
[yimirzoyan@fedora lab05]$
```

Рис. 4.7: Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ожидает ввода с клавиатуры, я ввожу свои ФИО, на этом программа завершает свою работу

4.3 Подключение внешнего файла

< ~ /Загрузки .[^]>				< ~ .[^]>			
.и	Имя	Размер	Время правки	.и	Имя	Размер	Время правки
./..	-ВВЕРХ-		дек 24 17:12	./..	-ВВЕРХ-		ноя 5 13:55
/install~0231109		132	ноя 9 21:00	/.cache		476	дек 24 16:45
/pandoc-2.18		16	апр 4 2022	/.config		572	дек 24 16:45
in_out.asm		3942	ноя 18 15:35	/.local		32	ноя 18 13:30
*pandoc-crossref		7453900	мая 21 2022	/.mozilla		48	ноя 5 13:56
pandoc-cssref.1		40584	мая 21 2022	/.ssh		84	ноя 5 15:47
				/.texlive2022		18	ноя 10 12:38
				/work		10	дек 2 21:56
				/Видео		50	ноя 18 12:57

Рис. 4.8: Скачиваю файл `in_out.asm` со страницы курса в ТУИС. Он сохранился в каталог “Загрузки”

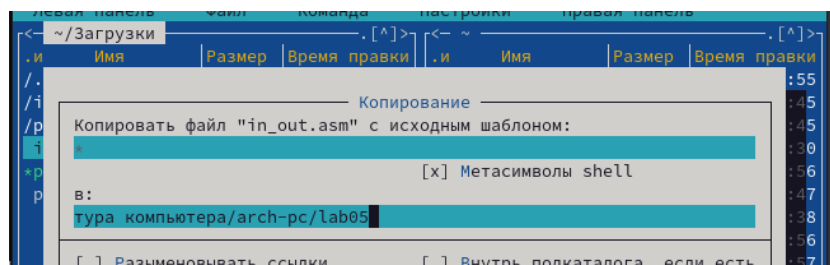


Рис. 4.9: С помощью функциональной клавиши F5 копирую файл in_out.asm из каталога Загрузки в созданный каталог lab05

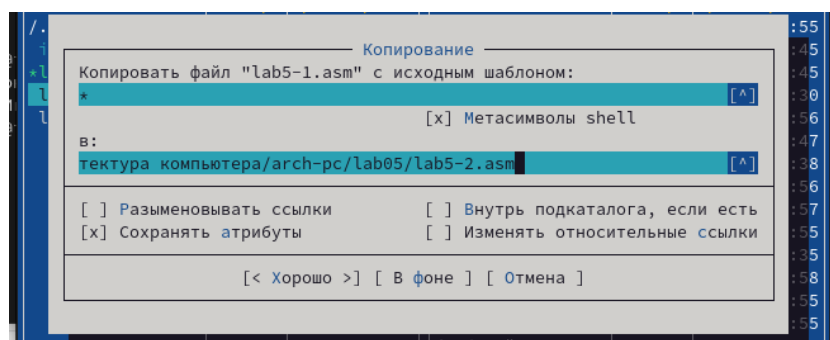


Рис. 4.10: С помощью функциональной клавиши F5 копирую файл lab5-1 в тот же каталог, но с другим именем, для этого в появившемся окне mc прописываю имя для копии файла

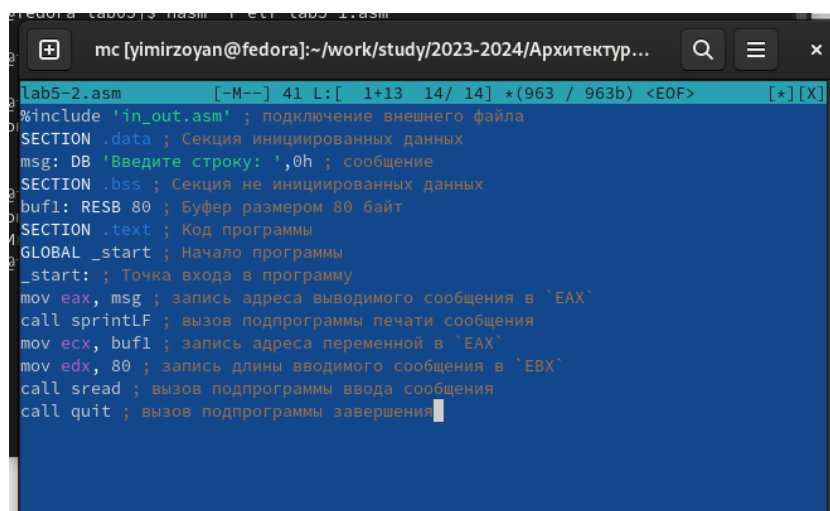


Рис. 4.11: Изменяю содержимое файла lab5-2.asm во встроенном редакторе mcedit

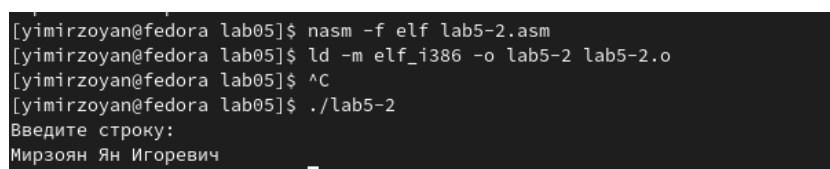


Рис. 4.12: Транслирую текст программы файла в объектный файл командой nasm -f elf lab5-2.asm. Создался объектный файл lab5-2.o. Выполняю компоновку объектного файла с помощью команды ld -m elf_i386 -o lab5-2 lab5-2.o Создался исполняемый файл lab5-2. Запускаю исполняемый файл

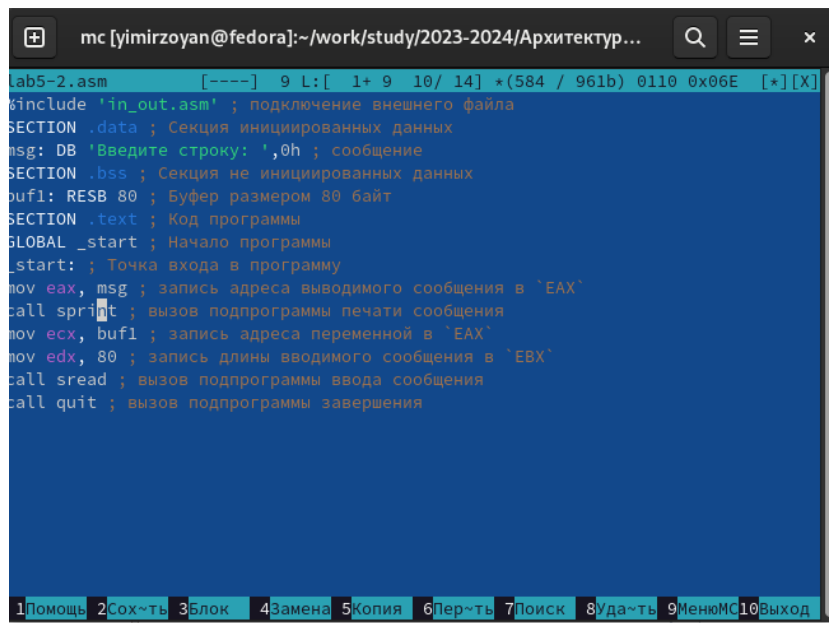


Рис. 4.13: Открываю файл lab5-2.asm для редактирования в mcedit функциональной клавишей F4. Изменяю в нем подпрограмму sprintLF на sprint

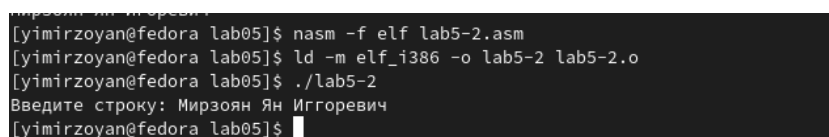


Рис. 4.14: Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл

Разница между первым исполняемым файлом lab5-2 и вторым lab5-2-2 в том, что запуск первого запрашивает ввод с новой строки, а программа, которая выполняется при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами sprintLF и sprint.

4.4 Выполнение заданий для самостоятельной работы

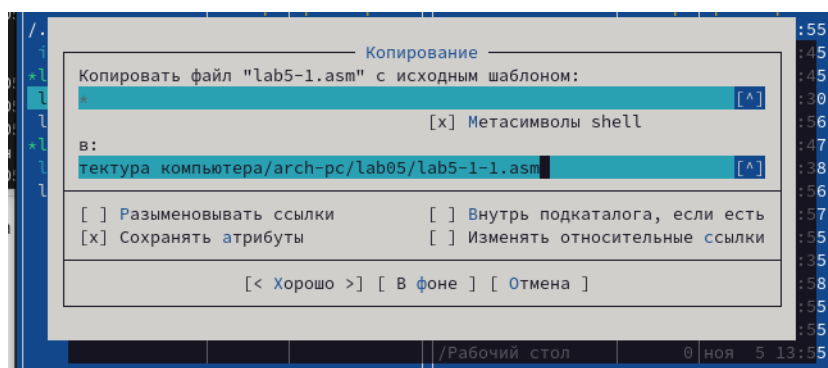


Рис. 4.15: Создаю копию файла lab5-1.asm с именем lab5-1-1.asm с помощью функциональной клавиши F5

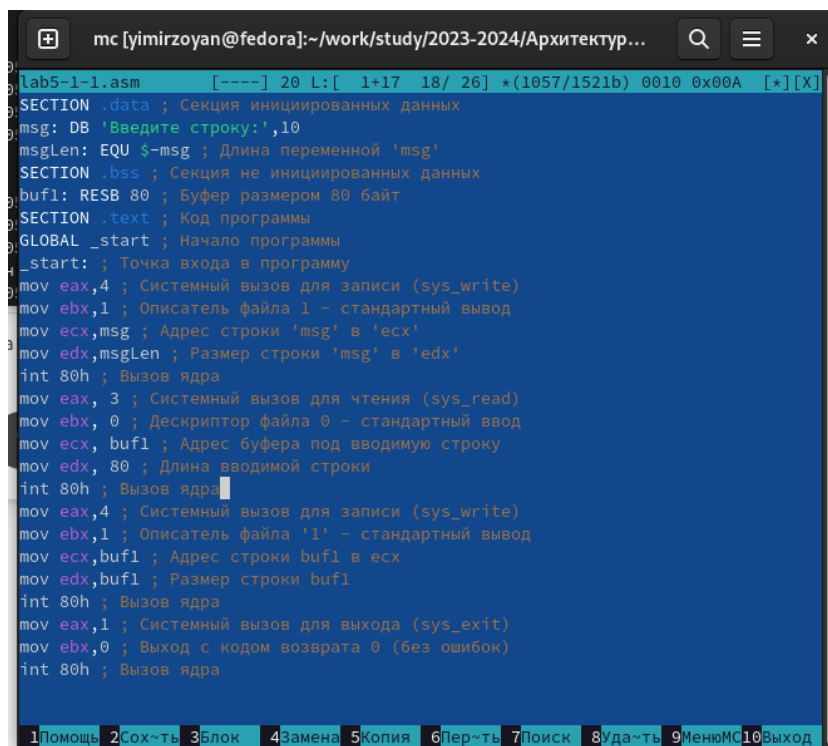


Рис. 4.16: С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку


```

SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку:',10
msgLen: EQU $-msg ; Длина переменной 'msg'
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,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 для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
mov ecx,buf1 ; Адрес строки buf1 в ecx
mov edx,buf1 ; Размер строки buf1
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

```

```
[yimirzoyan@fedora lab05]$ nasm -f elf lab5-1-1.asm
[yimirzoyan@fedora lab05]$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
[yimirzoyan@fedora lab05]$ ./lab5-1-1
Введите строку:
Мирзоян Ян Игоревич
Мирзоян Ян Игоревич
[yimirzoyan@fedora lab05]$
```

Рис. 4.17: Создаю объектный файл lab5-1-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-1-1, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свои ФИО, далее программа выводит введенные мною данные

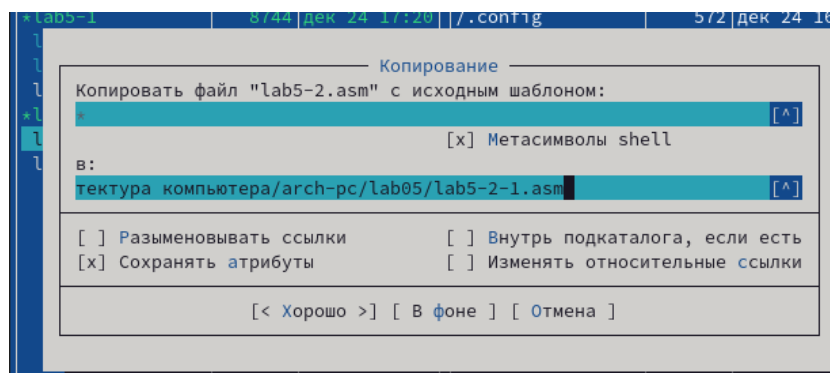
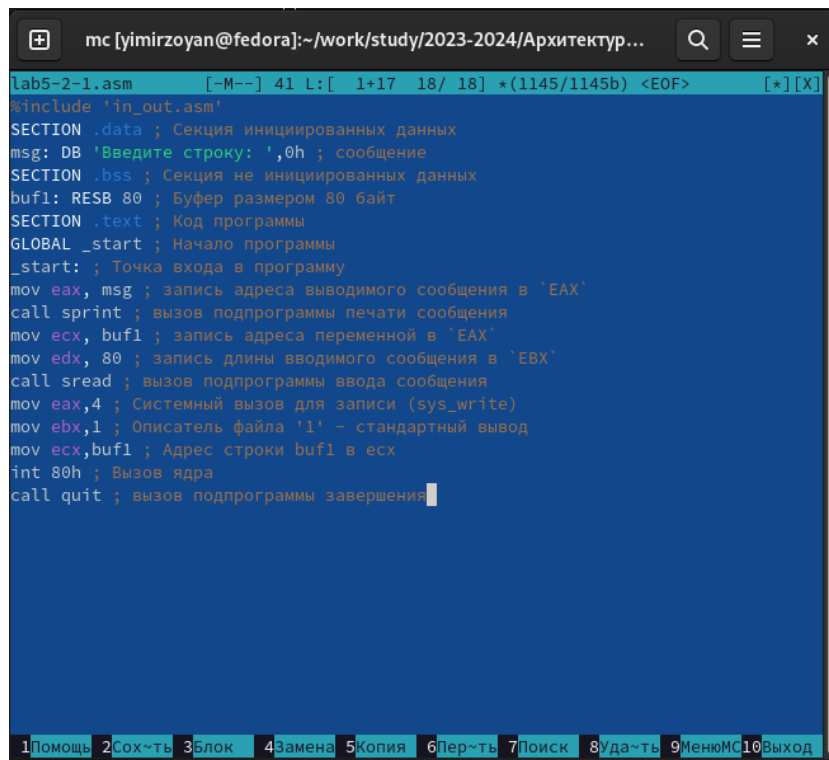


Рис. 4.18: Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5



```
lab5-2-1.asm [-M--] 41 L: [ 1+17 18/ 18] *(1145/1145b) <EOF> [*][X]
#include 'in_out.asm'
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 подпрограммы ввода сообщения
mov eax, 4 ; Системный вызов для записи (sys_write)
mov ebx, 1 ; Описатель файла '1' - стандартный вывод
mov ecx, buf1 ; Адрес строки buf1 в ecx
int 80h ; Вызов ядра
call quit ; вызов подпрограммы завершения
```

Рис. 4.19: С помощью функциональной клавиши F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку

```
%include 'in_out.asm'

SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку: ',0h ; сообщение

SECTION .bss ; Секция не инициированных данных
buf1: RESB 80 ; Буфер размером 80 байт

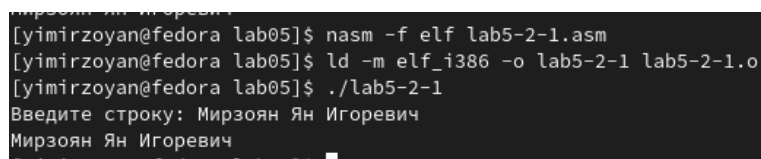
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу

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 подпрограммы ввода сообщения
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
mov ecx,buf1 ; Адрес строки buf1 в есх
int 80h ; Вызов ядра
call quit ; вызов подпрограммы завершения

```



```

[ymirzoyan@fedora lab05]$ nasm -f elf lab5-2-1.asm
[ymirzoyan@fedora lab05]$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
[ymirzoyan@fedora lab05]$ ./lab5-2-1
Введите строку: Мирзоян Ян Игоревич
Мирзоян Ян Игоревич

```

Рис. 4.20: Создаю объектный файл lab5-2-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-2-1, запускаю полученный исполняемый файл. Программа запрашивает ввод без переноса на новую строку, ввожу свои ФИО, далее программа выводит введенные мною данные

5 Выводы

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

Список литературы

[Лабораторная работа №5](https://esystem.rudn.ru/pluginfile.php/2089538/mod_resou