

Отчёт по лабораторной работе №5

Дисциплина: Архитектура Компьютера

Курилко-Рюмин Евгений

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
5	Выводы	20
	Список литературы	21

Список иллюстраций

4.1	Midnight Commander	8
4.2	Перемещение	9
4.3	Создание и переход в каталога	9
4.4	Создание файла	10
4.5	Редактирование файла	10
4.6	Компиляция, обработка и запуск исполняемого файла	11
4.7	Исполнение файла	11
4.8	Скачанный файл	12
4.9	Копирование файла	13
4.10	Копирование файла	14
4.11	Редактирование файла	15
4.12	Компиляция, обработка и запуск исполняемого файла	15
4.13	Редактирование файла	16
4.14	Запуск исполняемого файла	16
4.15	Создание копии файла	17
4.16	Редактирование файла	17
4.17	Исполнение файла	18
4.18	Создание копии файла	18
4.19	Отправка файлов	19
4.20	Исполнение файла	19

1 Цель работы

Целью данной работы является приобретение практического опыта работы с Midnight Commander, освоение инструкций языка ассемблера `mov` и `int`.

2 Задание

1. Общее ознакомление с Midnight Commander
2. Основы работы с Midnight Commander
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: 1) DB (define byte) — определяет переменную размером в 1 байт; 2) DW (define word) — определяет переменную размером в 2 байта (слово); 3) DD (define double word) — определяет переменную размером в 4 байта (двойное слово); 4) DQ (define quad word) — определяет переменную размером в 8 байт (четверённое слово); 5) 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. 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. 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`. Также необходимо учитывать то, что размер операндов приемника и источника должны совпадать. Инструкция языка ассемблера `int` предназначена для вызова прерывания с указанным номером. 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 тот 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) Основы работы с Midnight Commander

Открываю Midnight Commander, введя в терминале команду mc. (рис.1).

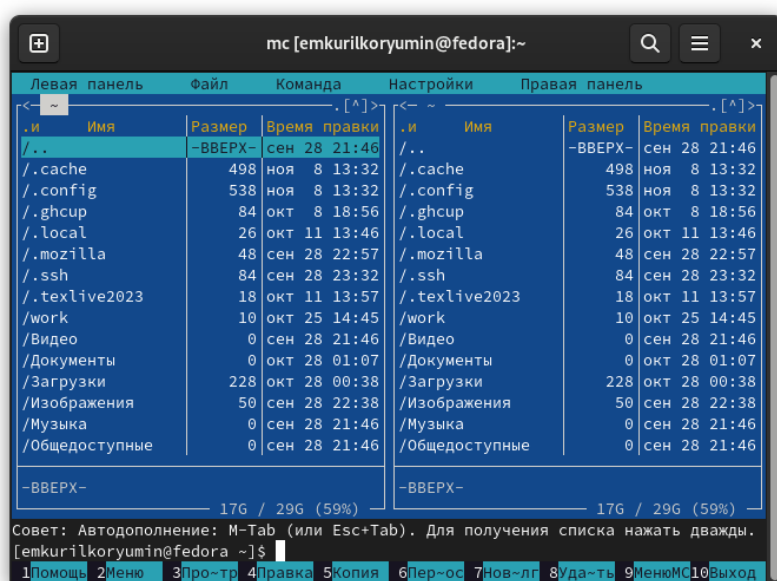


Рис. 4.1: Midnight Commander

Перехожу в каталог ~/work/arch-рс, используя файловый менеджер mc. (рис.2).

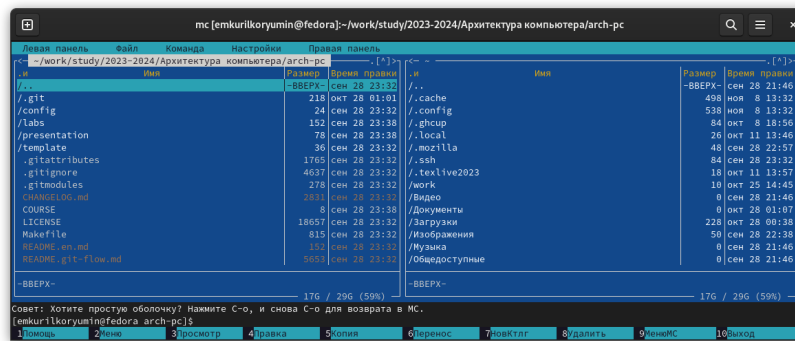


Рис. 4.2: Перемещение

С помощью функциональной клавиши F7 создаю каталог lab05 и перехожу в созданный каталог. (рис.3).

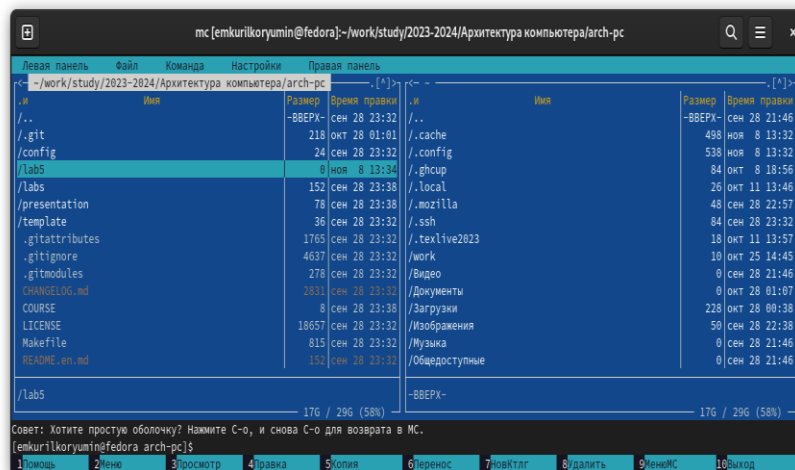


Рис. 4.3: Создание и переход в каталога

В строке ввода прописываю команду `touch lab5-1.asm`, дабы создать соответствующий файл. (рис.4).

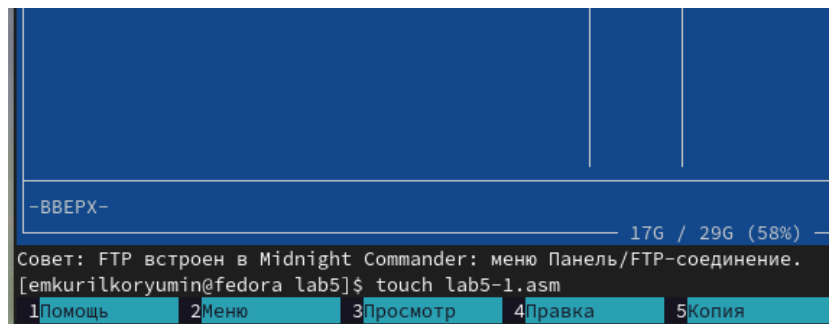


Рис. 4.4: Создание файла

С помощью функциональной клавиши F4 открываю созданный файл в режиме правки. Ввожу в файл код программы для запроса строки у пользователя, затем сохраняю изменения и выхожу из файла. (рис.5).

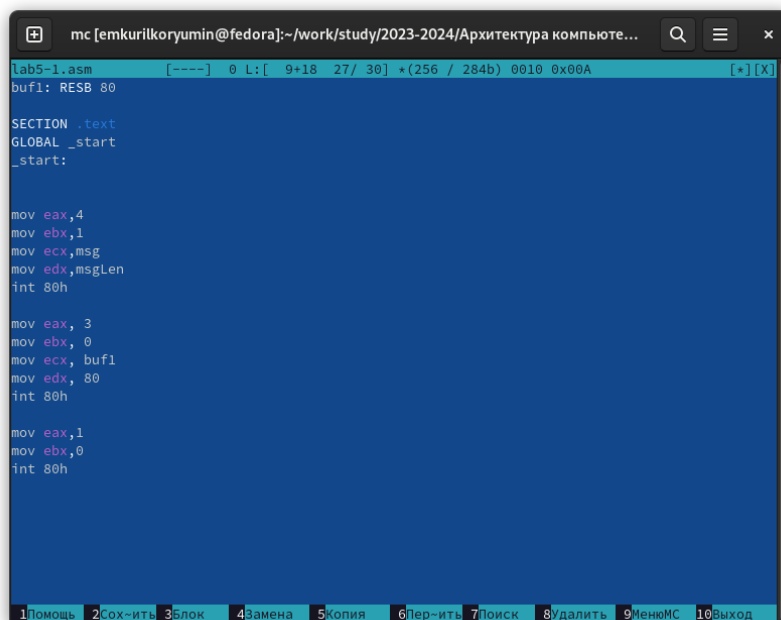


Рис. 4.5: Редактирование файла

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`. Запускаю исполняемый файл. Программа выводит “Введите

строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу. (рис.6).

```
[emkurilkoryumin@fedora lab5]$ nasm -f elf lab5-1.asm
[emkurilkoryumin@fedora lab5]$ ld -m elf_i386 -o lab5-1.o
ld: error: no input files

[emkurilkoryumin@fedora ~]$ ./lab5-1
bash: ./lab5-1: Нет такого файла или каталога

[emkurilkoryumin@fedora ~]$ Курилко-Рюмин Евгений
bash: Курилко-Рюмин: команда не найдена...

[emkurilkoryumin@fedora lab5]$ ld -m elf_i386 -o lab5-1 lab5-1.o

[emkurilkoryumin@fedora lab5]$ ./lab5-1
Введите строку:
Курилко-Рюмин Евгений
```

Рис. 4.6: Компиляция, обработка и запуск исполняемого файла

Запускаю исполняемый файл. Программа выводит строку “Введите строку:” и ждет ввода с клавиатуры, я ввожу свои ФИО, на этом программа заканчивает свою работу (рис.7).

```
[emkurilkoryumin@fedora lab5]$ ./lab5-1
Введите строку:
Курилко-Рюмин Евгений
```

Рис. 4.7: Исполнение файла

4.2) Подключение внешнего файла

Скачиваю файл in_out.asm со страницы курса в ТУИС. (рис.8)

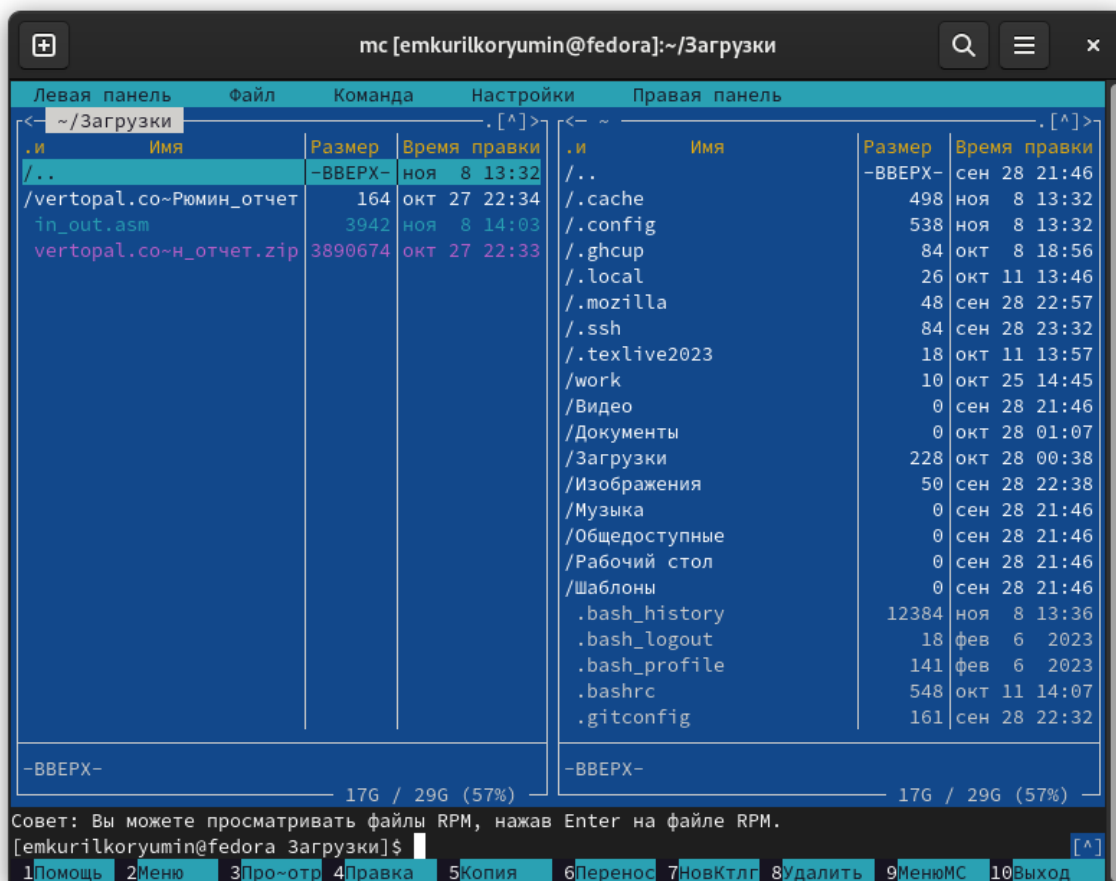


Рис. 4.8: Скачанный файл

С помощью функциональной клавиши F5 копирую файл in_out.asm из каталога “Загрузки”, куда он скачался, в созданный каталог lab05. (рис.9).

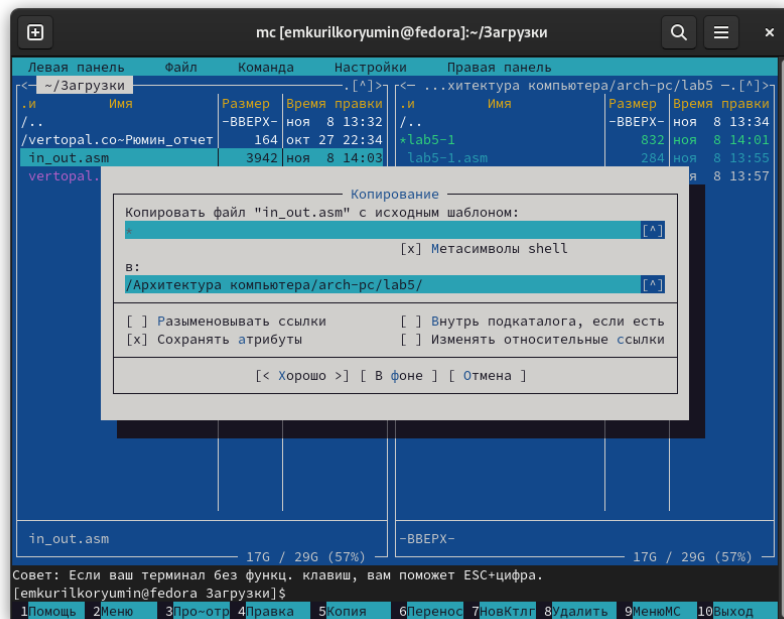


Рис. 4.9: Копирование файла

Далее копирую файл lab5-1 в тот же каталог, но с другим именем, для этого в окне mc прописываю путь к каталогу и новое имя файла. (рис.10).

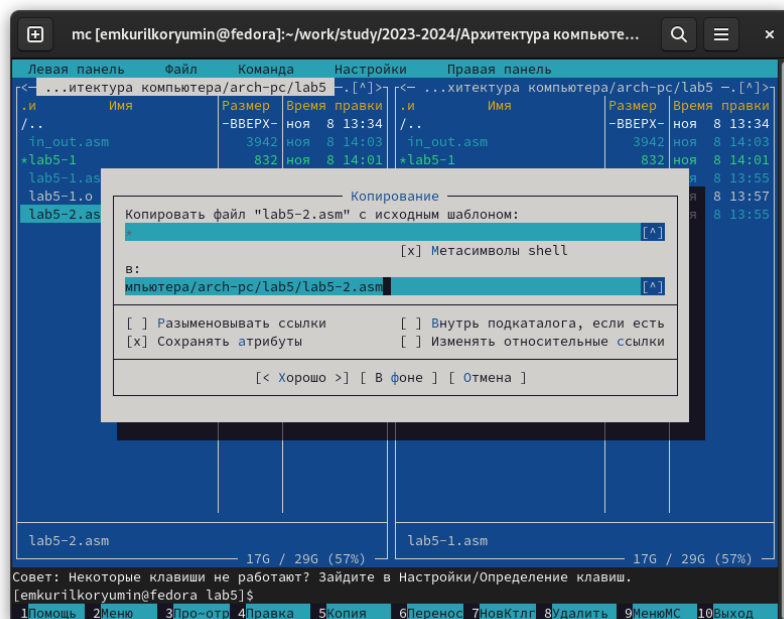
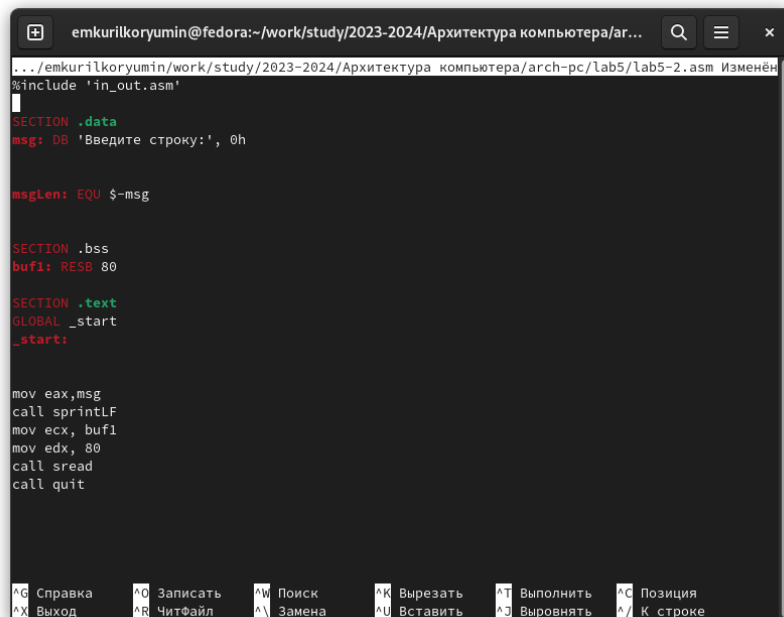


Рис. 4.10: Копирование файла

Изменяю содержимое файла lab5-2.asm во встроенном редакторе, чтобы в программе также использовался подключенный внешний файл in_out.asm. (рис.11).



```
emkurilkoryumin@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab5/lab5-2.asm Изменён
.../emkurilkoryumin/work/study/2023-2024/Архитектура компьютера/arch-pc/lab5/lab5-2.asm
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите строку:', 0h

msgLen: EQU $-msg

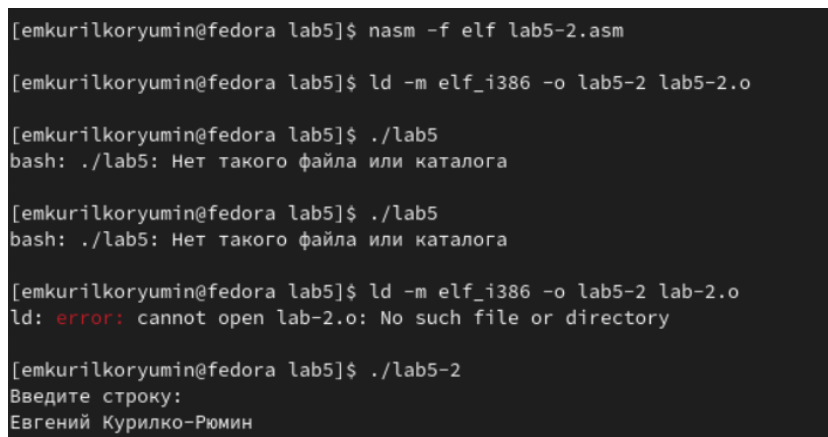
SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax,msg
call sprintLF
mov ecx, buf1
mov edx, 80
call sread
call quit
```

Рис. 4.11: Редактирование файла

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 его. (рис.12).

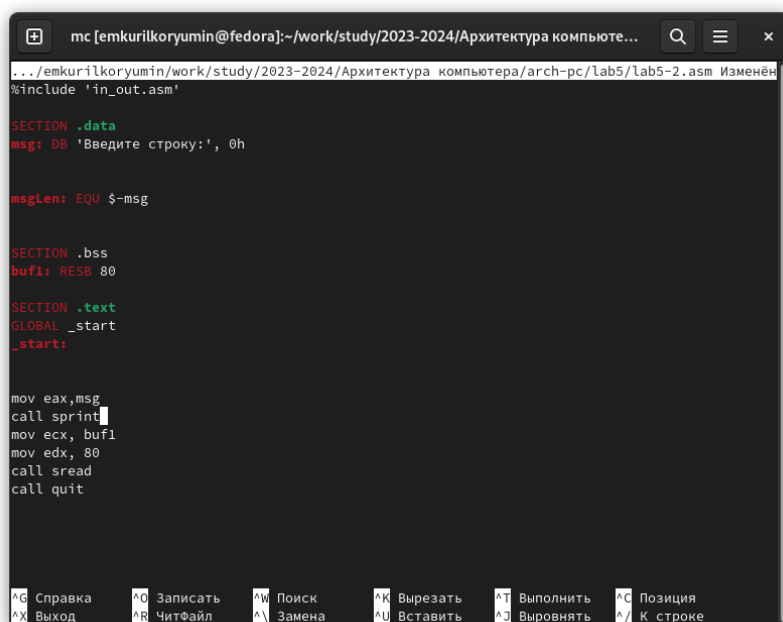


```
[emkurilkoryumin@fedora lab5]$ nasm -f elf lab5-2.asm
[emkurilkoryumin@fedora lab5]$ ld -m elf_i386 -o lab5-2 lab5-2.o
[emkurilkoryumin@fedora lab5]$ ./lab5
bash: ./lab5: Нет такого файла или каталога
[emkurilkoryumin@fedora lab5]$ ./lab5
bash: ./lab5: Нет такого файла или каталога
[emkurilkoryumin@fedora lab5]$ ld -m elf_i386 -o lab5-2 lab-2.o
ld: error: cannot open lab-2.o: No such file or directory
[emkurilkoryumin@fedora lab5]$ ./lab5-2
Введите строку:
Евгений Курилко-Рюмин
```

Рис. 4.12: Компиляция, обработка и запуск исполняемого файла

Открываю файл `lab5-2.asm` в режиме редактирования, меняю в нем `sprintLF` на

sprint. (рис.13).



```
mc [emkurilkoryumin@fedora]:~/work/study/2023-2024/Архитектура компьюте...
.../emkurilkoryumin/work/study/2023-2024/Архитектура компьютера/arch-pc/lab5/lab5-2.asm Изменён
#include 'in_out.asm'

SECTION .data
msg: DB 'Введите строку:', 0h

msgLen: EQU $-msg

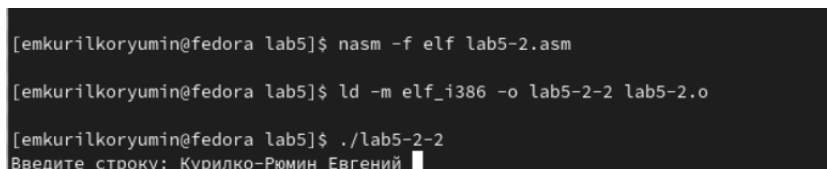
SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax,msg
call sprint
mov ecx, buf1
mov edx, 80
call sread
call quit
```

Рис. 4.13: Редактирование файла

Сохраняю изменения и транслирую файл, выполняю компоновку созданного объектного файла, в конце концов запускаю новый исполняемый файл. (рис.14).



```
[emkurilkoryumin@fedora lab5]$ nasm -f elf lab5-2.asm
[emkurilkoryumin@fedora lab5]$ ld -m elf_i386 -o lab5-2-2 lab5-2.o
[emkurilkoryumin@fedora lab5]$ ./lab5-2-2
Введите строку: Курилко-Рюмин Евгений
```

Рис. 4.14: Запуск исполняемого файла

Разница между первой и второй версией исполняемого файла состоит в том, что запуск первой запрашивает ввод с новой строки, а программа, которая исполняется при запуске второй, запрашивает ввод без переноса на новую строку, ибо в этом и заключается различие между подпрограммами `sprintLF` и `sprint`.

4.3) Выполнение заданий для самостоятельной работы

1. Создаю копию файла `lab5-1.asm` с именем `lab5-1-a.asm`. (рис.15).

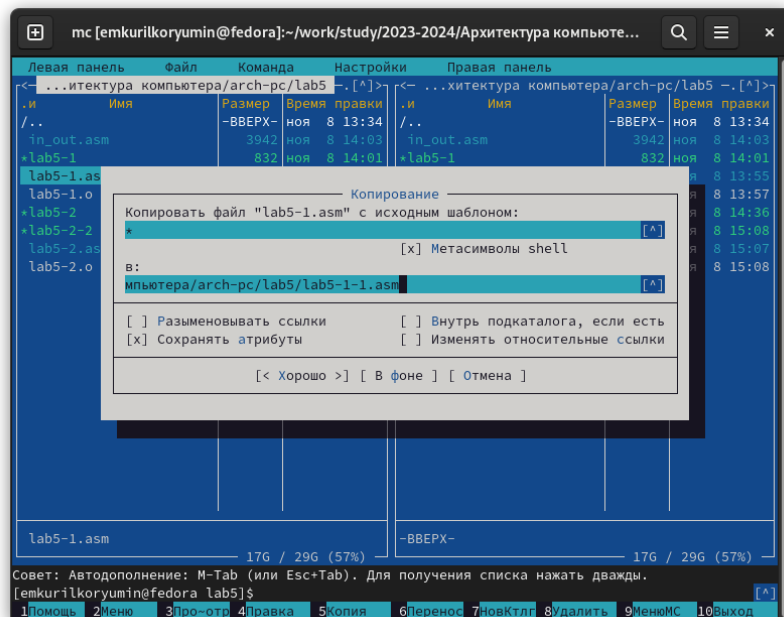


Рис. 4.15: Создание копии файла

Затем открываю созданный файл в режиме правки. Изменяю программу так, чтобы кроме вывода “Введите строку” и запроса ввода, она выводила вводимую пользователем строку. (рис.16)

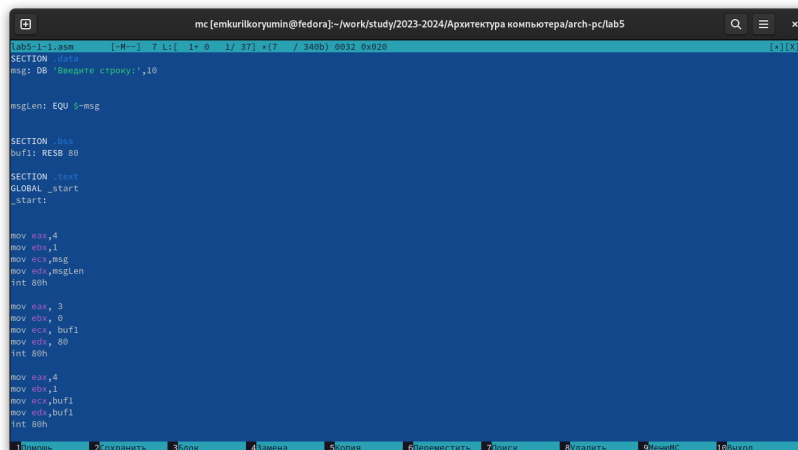


Рис. 4.16: Редактирование файла

2. Создаю объектный файл lab5-1-а.о, отдаю его на обработку компоновщику,

получаю исполняемый файл lab5-1-а, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свои ФИО, далее программа выводит введенные мною данные.(рис.17)

```
[emkurilkoryumin@fedora lab5]$ nasm -f elf lab5-1-1.asm
[emkurilkoryumin@fedora lab5]$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
[emkurilkoryumin@fedora lab5]$ ./lab5-1-1
Введите строку:
Курилко-Рюмин Евгений
Курилко-Рюмин Евгений
```

Рис. 4.17: Исполнение файла

3. Создаю копию файла lab5-2.asm с именем lab5-2-а.asm. (рис.18)

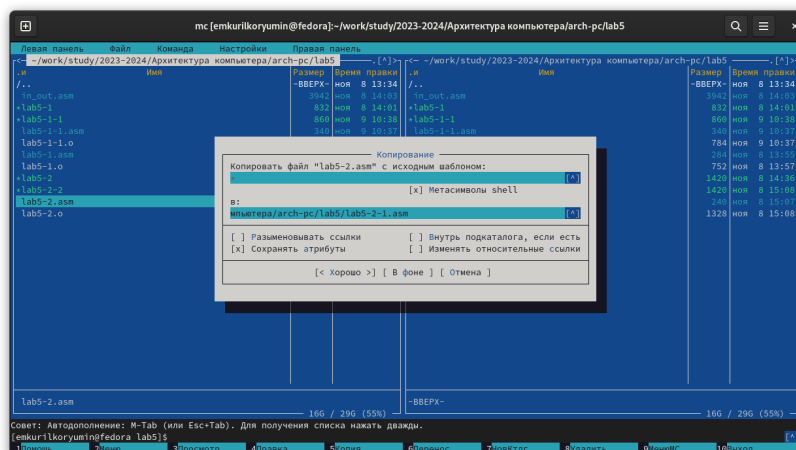


Рис. 4.18: Создание копии файла

Открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку. (рис.19)

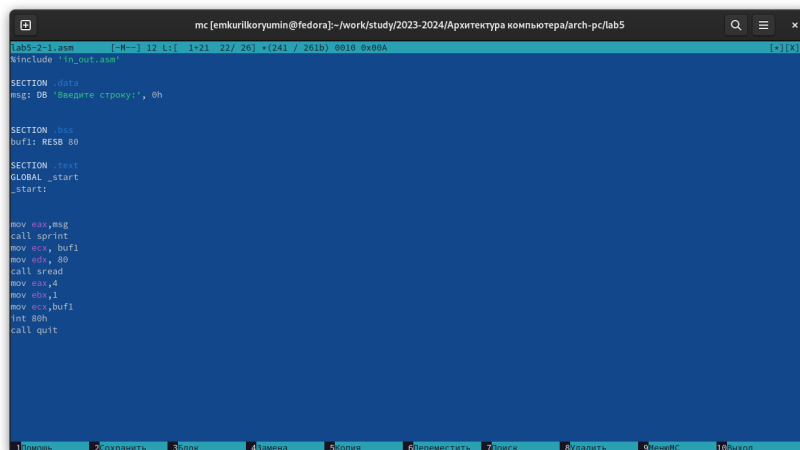


Рис. 4.19: Отправка файлов

4. Создаю объектный файл lab5-2-а.о, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-2-а, запускаю полученный исполняемый файл. Затем программа запрашивает ввод без переноса на новую строку, ввожу свои ФИО, далее снова выводятся введенные мною данные. (рис.20)

```

[emkurilkoryumin@fedora lab5]$ nasm -f elf lab5-2-1.asm

[emkurilkoryumin@fedora lab5]$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o

[emkurilkoryumin@fedora lab5]$ ./lab5-2-1
Введите строку:Курилко-Рюмин Евгений
Курилко-Рюмин Евгений
  
```

Рис. 4.20: Исполнение файла

5 Выводы

При выполнении лабораторной работы я приобрёл практический опыт работы с Midnight Commander, освоил инструкции языка ассемблера.

Список литературы

Архитектура ЭВМ