

Шаблон отчёта по лабораторной работе

Простейший вариант

Сокирка Анна Константиновна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Основы работы с mc	9
4.2	Структура программы на языке ассемблера NASM	10
4.3	Подключение внешнего файла	12
4.4	Выполнение заданий для самостоятельной работы	16
5	Выводы	20
6	Список литературы	21

Список иллюстраций

4.1 Рисунок 1	9
4.2 Рисунок 2	9
4.3 Рисунок 3	10
4.4 Рисунок 4	10
4.5 Рисунок 5	11
4.6 Рисунок 6	11
4.7 Рисунок 7	12
4.8 Рисунок 8	12
4.9 Рисунок 9	12
4.10 Рисунок 10	13
4.11 Рисунок 11	13
4.12 Рисунок 12	14
4.13 Рисунок 13	14
4.14 Рисунок 14	15
4.15 Рисунок 15	15
4.16 Рисунок 16	16
4.17 Рисунок 17	17
4.18 Рисунок 18	17
4.19 Рисунок 19	18
4.20 Рисунок 20	18
4.21 Рисунок 21	19
4.22 Рисунок 22	19

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 (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

4.1 Основы работы с mc

Открою Midnight Commander (рис. 4.1).

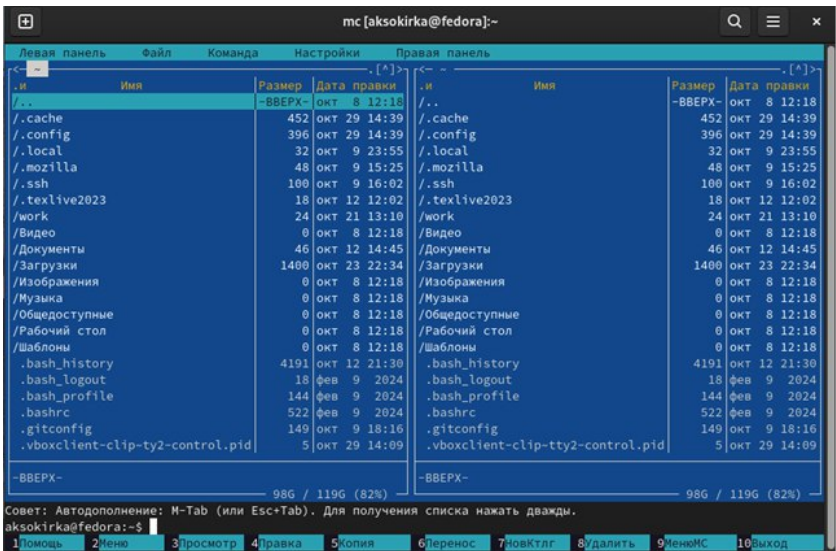


Рис. 4.1: Рисунок 1

Перейду в каталог ~/work/arch-рс созданный при выполнении лабораторной работы №4 (рис. 4.2).

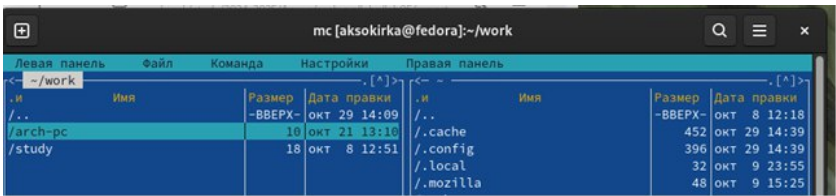


Рис. 4.2: Рисунок 2

С помощью функциональной клавиши F7 создам папку lab05 и перейду в созданный каталог (рис. 4.3).

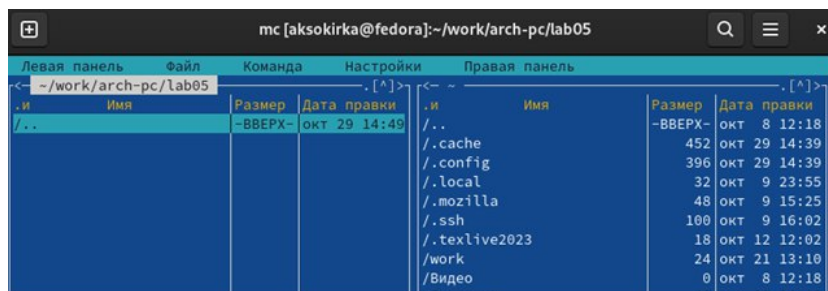


Рис. 4.3: Рисунок 3

Пользуясь строкой ввода и командой touch создам файл lab5-1.asm (рис. 4.4).

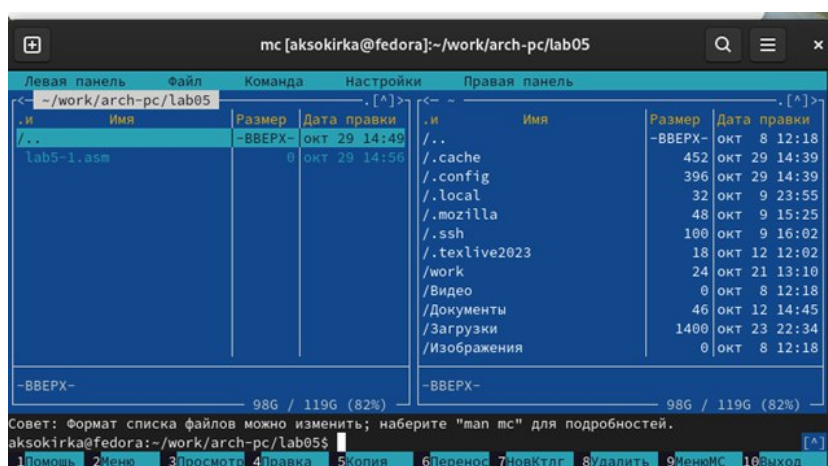


Рис. 4.4: Рисунок 4

4.2 Структура программы на языке ассемблера NASM

С помощью функциональной клавиши F4 откройте файл lab5-1.asm для редактирования во встроенном редакторе (рис. 4.5).

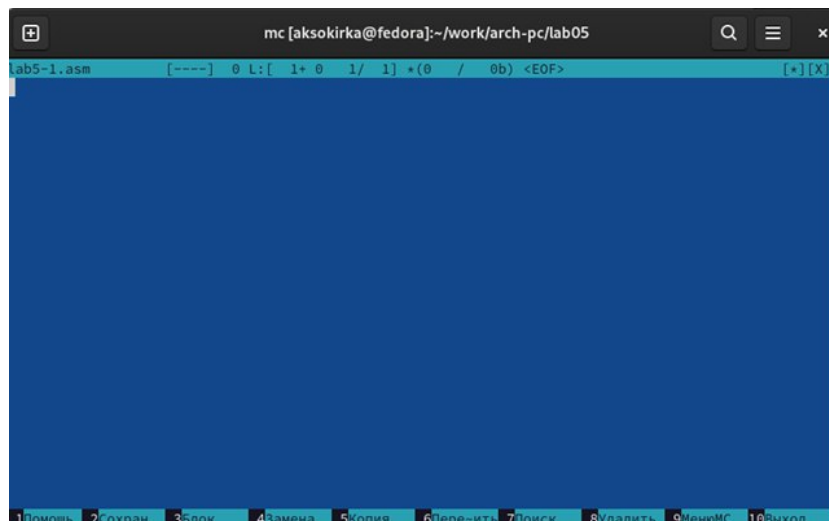


Рис. 4.5: Рисунок 5

Ввожу текст программы из листинга, сохраняю изменения и закрываю файл (рис. 4.6).

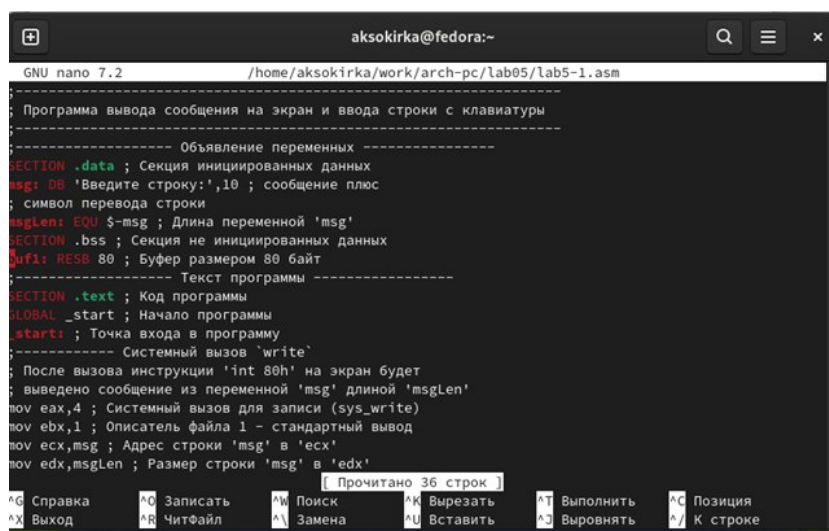


Рис. 4.6: Рисунок 6

С помощью функциональной клавиши F3 открою файл lab5-1.asm для просмотра. Убеждаюсь, что файл содержит текст программы (рис. 4.7).

```

mc [aksokirka@fedora]:~/work/arch-pc/lab05
/home/aksokirka/work/arch-pc/lab05/lab5-1.asm 1522/2433 62%
;-----
; 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' -----
1Помощь 2Разверн 3Выход 4Нех 5Перейти 6 7Поиск 8Исходный 9Формат 10Выход

```

Рис. 4.7: Рисунок 7

Оттранслирую текст программы lab5-1.asm в объектный файл. Выполню компоновку объектного файла и запущу получившийся исполняемый файл. Программа выводит строку 'Введите строку:' и ожидает ввода с клавиатуры. На запрос введу мои ФИО (рис. 4.8).

```

aksokirka@fedora:~$ mc
aksokirka@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
aksokirka@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
aksokirka@fedora:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Сокирка Анна Константиновна

```

Рис. 4.8: Рисунок 8

4.3 Подключение внешнего файла

Скачиваю файл in_out.asm со страницы курса в ТУИС (рис. 4.9).

5242396606182056885.jpg	6210	ОКТ 22 15:07	/evolution	14	ОКТ 8 12:18
5242396606182056888.jpg	42448	ОКТ 22 15:07	/goa-1.0	0	ОКТ 8 12:18
524239660618-56894(1).jpg	10795	ОКТ 22 15:07	/gtk-3.0	18	ОКТ 29 14:09
in_out.asm	3942	ОКТ 30 15:52	/gtk-4.0	0	ОКТ 12 10:43

Рис. 4.9: Рисунок 9

С помощью функциональной клавиши F5 копирую файл in_out.asm из каталога Загрузки в созданный каталог lab05-1.asm (рис. 4.10).

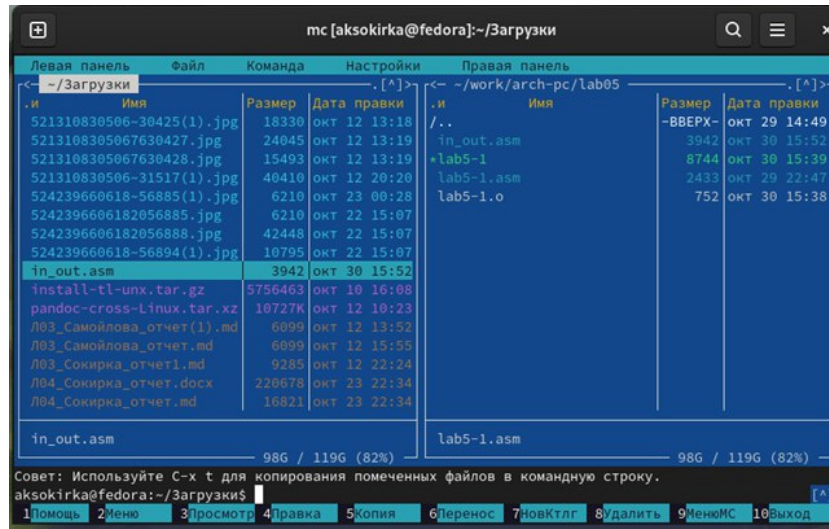


Рис. 4.10: Рисунок 10

Создам копию файла lab5-1.asm с именем lab5-2.asm. Выделю файл lab5-1.asm, введу имя файла lab5-2.asm (рис. 4.11).

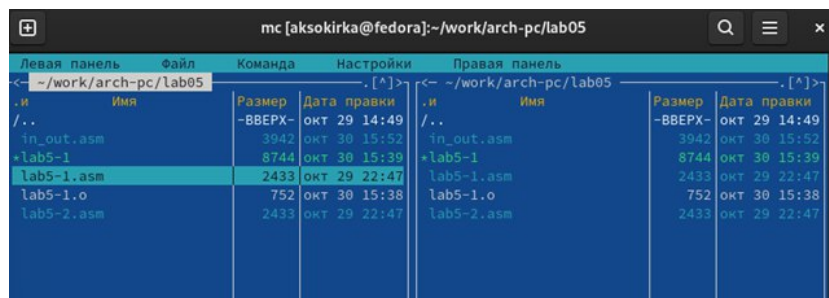


Рис. 4.11: Рисунок 11

Изменяю содержимое файла lab5-2.asm во встроенном редакторе nano, чтобы в программе использовались подпрограммы из внешнего файла in_out.asm (рис. 4.12).

```

mc [aksokirka@fedora:~/Заргузки
GNU nano 7.2 /home/aksokirka/work/arch-pc/lab05/lab5-2.asm
#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 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: Рисунок 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).

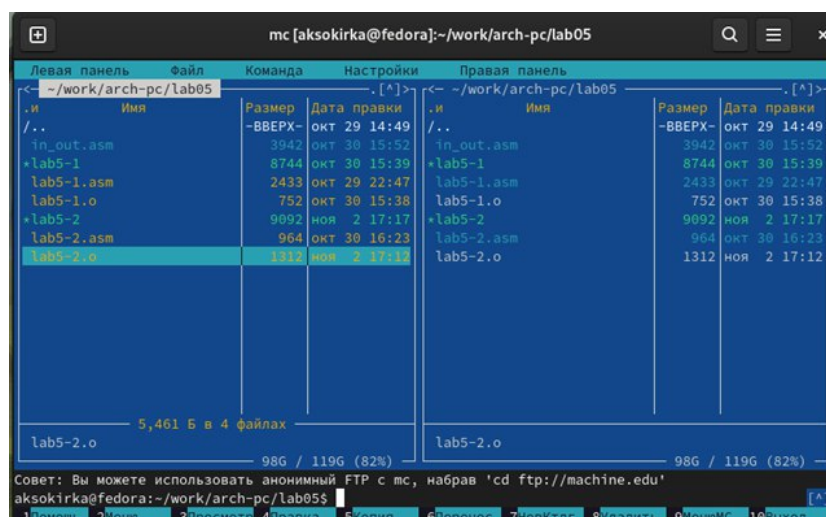


Рис. 4.13: Рисунок 13

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 исполняется при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами `sprintf` и `print` (рис. 4.14).

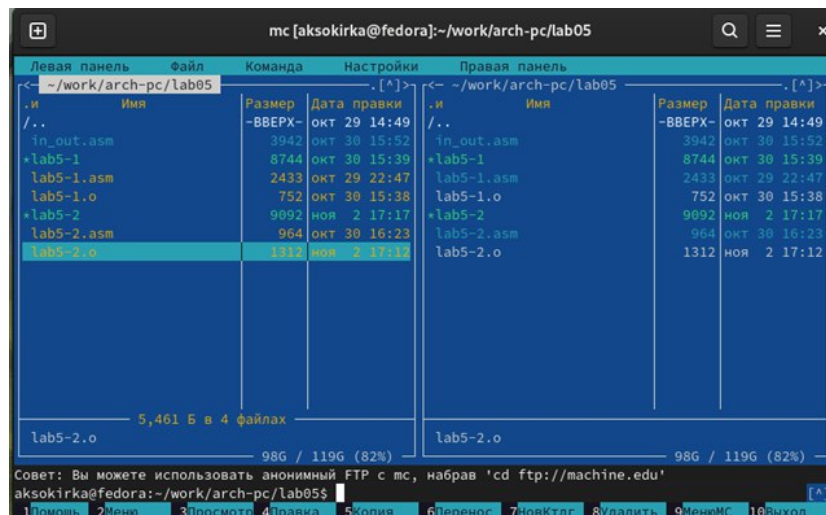


Рис. 4.14: Рисунок 14

Открываю файл lab5-2.asm для редактирования в nano функциональной клавишей F4. Изменяю в нем подпрограмму sprintLF на sprint. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий (рис. 4.15).

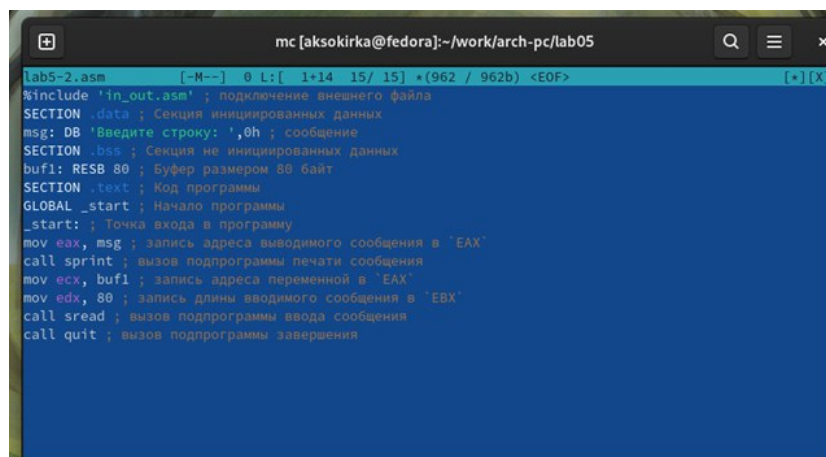


Рис. 4.15: Рисунок 15

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл (рис. 4.16).

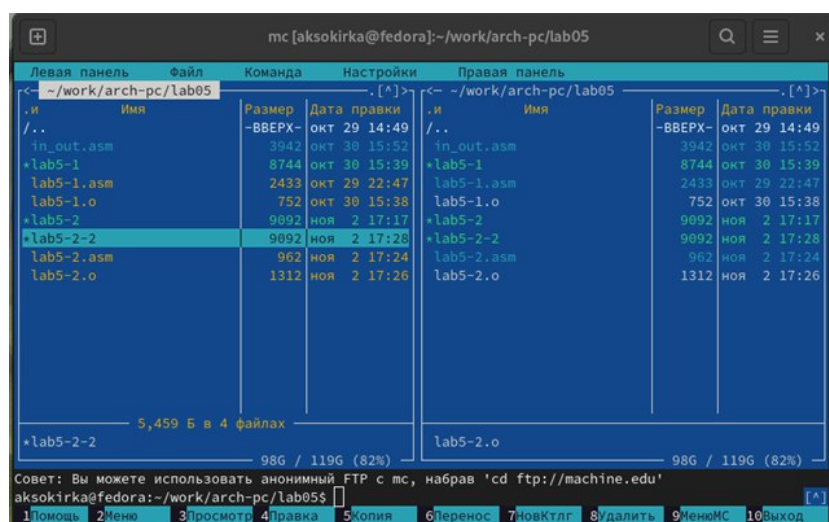


Рис. 4.16: Рисунок 16

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 исполняется 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

Создаю копию файла `lab5-1.asm` с именем `lab5-1-1.asm` с помощью функциональной клавиши F5 (рис. 4.17).

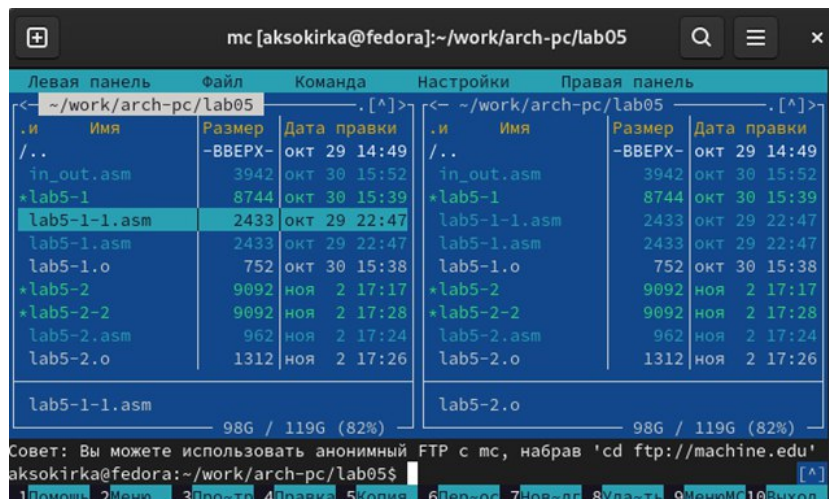


Рис. 4.17: Рисунок 17

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 (рис. 4.18).

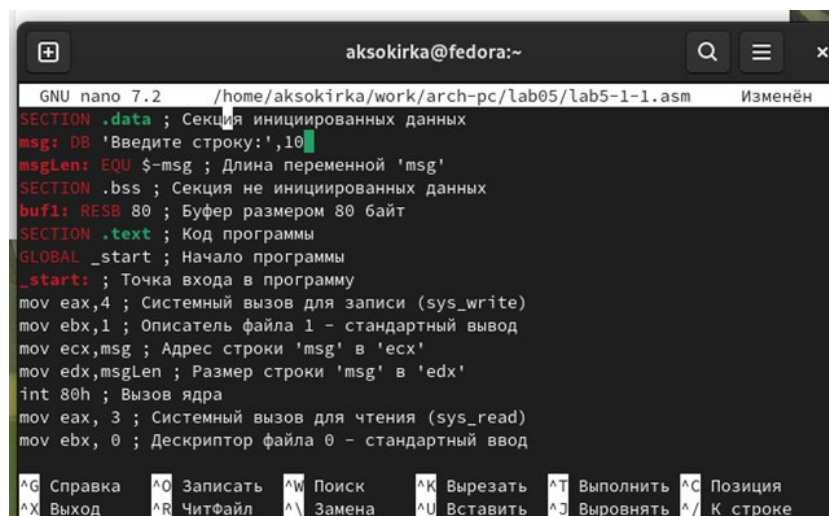


Рис. 4.18: Рисунок 18

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. 4.19).

```

aksokirka@fedora:~$
aksokirka@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-1-1.asm

aksokirka@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o

aksokirka@fedora:~/work/arch-pc/lab05$ ./lab5-1-1
Введите строку:
Сокирка Анна Константиновна
Сокирка Анна Константиновна

```

Рис. 4.19: Рисунок 19

Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью функциональной клавиши F5 (рис. 4.20).

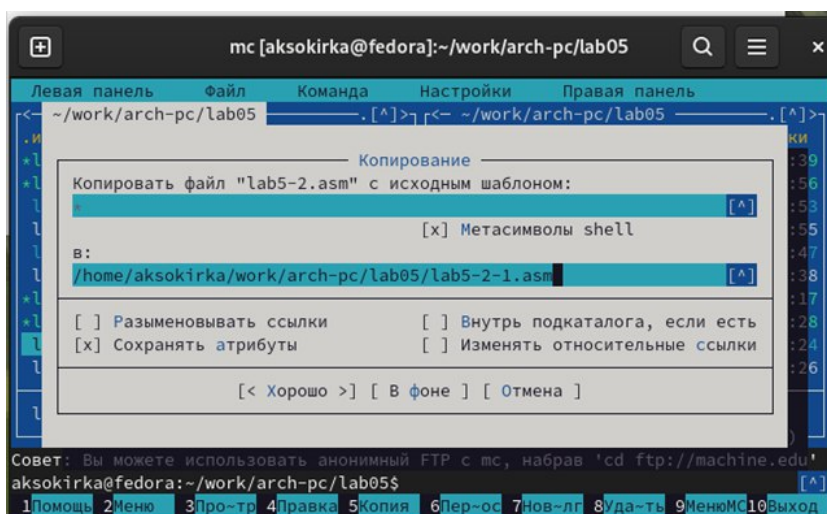
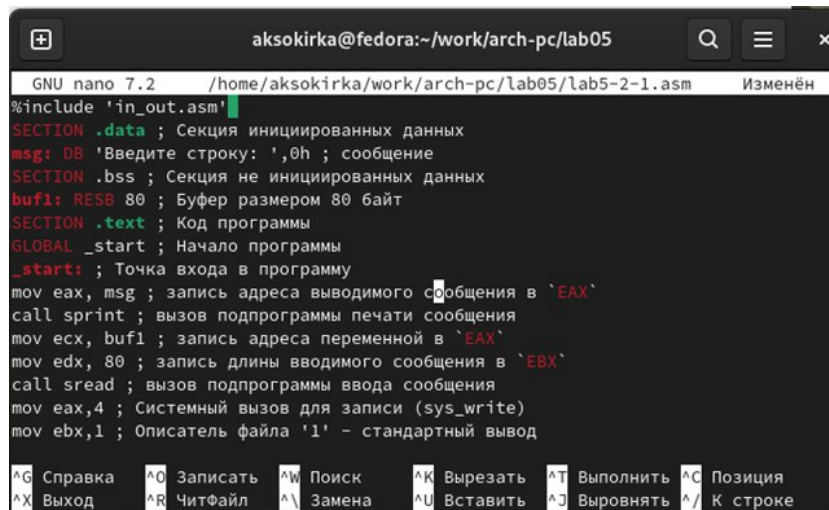


Рис. 4.20: Рисунок 20

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 (рис. 4.21).

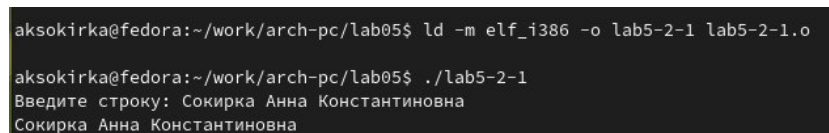


```
aksokirka@fedora:~/work/arch-pc/lab05
GNU nano 7.2 /home/aksokirka/work/arch-pc/lab05/lab5-2-1.asm
%include 'in_out.asm'
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не иницированных данных
buf1: RESB 80 ; Буфер размером 80 байт
SECTION .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 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Вывернуть ^_ К строке
```

Рис. 4.21: Рисунок 21

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 (рис. 4.22).



```
aksokirka@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
aksokirka@fedora:~/work/arch-pc/lab05$ ./lab5-2-1
Введите строку: Сокирка Анна Константиновна
Сокирка Анна Константиновна
```

Рис. 4.22: Рисунок 22

5 Выводы

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

6 Список литературы

https://esystem.rudn.ru/pluginfile.php/2089085/mod_resource/content/0/Лабораторная%20работа%20№5.%20Основы%20работы%20с%20Midnight%20Commander%20%28%20тура%20программы%20на%20языке%20ассемблера%20NASM.%20Системные%20вызовы%20в%20ОС%20GNU%20Linux.pdf