

Лабораторная работа №5

**Основы работы с Midnight Commander (mc). Структура программы
на языке ассемблера NASM. Системные вызовы в ОС GNU Linux**

Самарханова Полина Тимуровна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Основы работы с NASM.	8
4.2	Структура программы на языке ассемблера NASM.	9
4.3	Подключение внешнего файла.	12
4.4	Выполнение заданий для самостоятельной работы.	15
5	Выводы	19
	Список литературы	20

Список иллюстраций

4.1	Открытый Midnight Commander	8
4.2	Перемещение в каталог	9
4.3	Создание каталога	9
4.4	Создание файла	9
4.5	Файл в редакторе	10
4.6	Редактирование файла	10
4.7	Открытие файла для проверки	11
4.8	Создание объектного файла	11
4.9	Компоновка	11
4.10	Запуск программы	12
4.11	Файл in_out.asm	12
4.12	Копирование файла в нужную директорию	13
4.13	Редактирование файла, для использования in_put.asm	14
4.14	Создание объектного файла	14
4.15	Компоновка файла и запуск программы	14
4.16	Редактирование файла	15
4.17	Исполнение файла	15
4.18	Копирование файла lab5-1.asm	16
4.19	Редактирование программы	16
4.20	Исполнение файла	17
4.21	Копирование файла	17
4.22	Редактирование программы	18
4.23	Исполнение файла	18

Список таблиц

1 Цель работы

Приобретение практических навыков работы в Midnight Commander. Освоение инструкций языка ассемблера `mov` и `int`

2 Задание

Основы работы с NASM

Структура программы на языке ассемблера NASM

Подключение внешнего файла

Выполнение заданий для самостоятельной работы

3 Теоретическое введение

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 (учетверённое слово); • DT (define ten bytes) — определяет переменную размером в 10 байт.

4 Выполнение лабораторной работы

4.1 Основы работы с NASM.

Я открыла Midnight Commander, используя команду `mc` (рис. [4.1]).

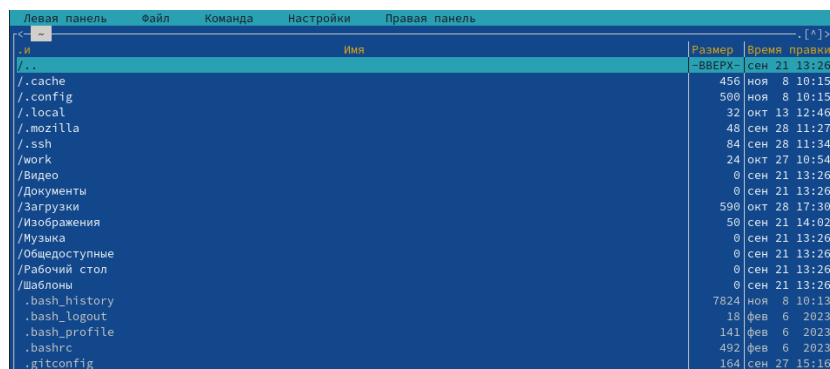


Рис. 4.1: Открытый Midnight Commander

Перешла в каталог `work`, созданный при выполнении предыдущей лабораторной работы (рис. [4.2]).

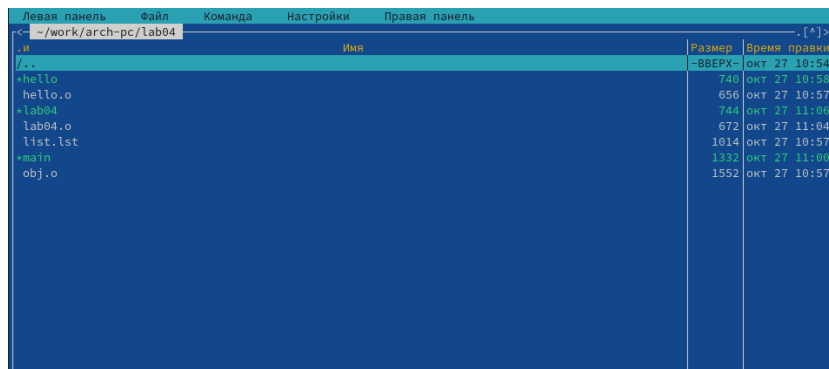


Рис. 4.2: Перемещение в каталог

Далее создала новый каталог lab05(рис. [4.3]).

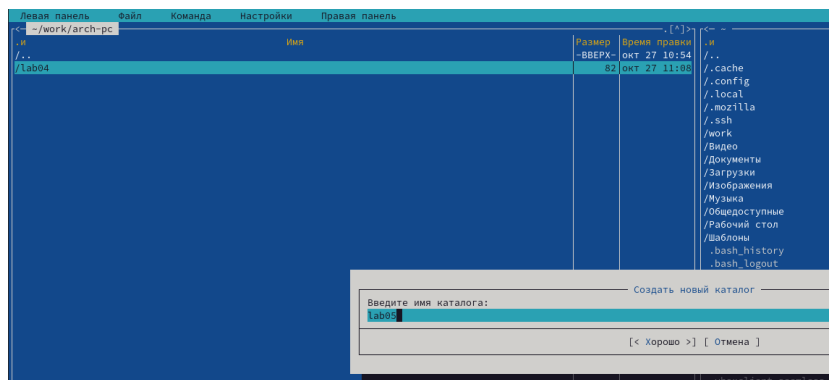


Рис. 4.3: Создание каталога

В новом каталоге создала файл lab5-1.asm, в котором я буду работать далее, используя команду touch(рис. [4.4]).

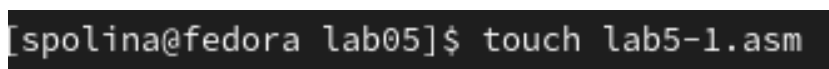


Рис. 4.4: Создание файла

4.2 Структура программы на языке ассемблера NASM.

С помощью клавиши F4 я открыла созданный файл в редакторе nano(рис. [4.5]).

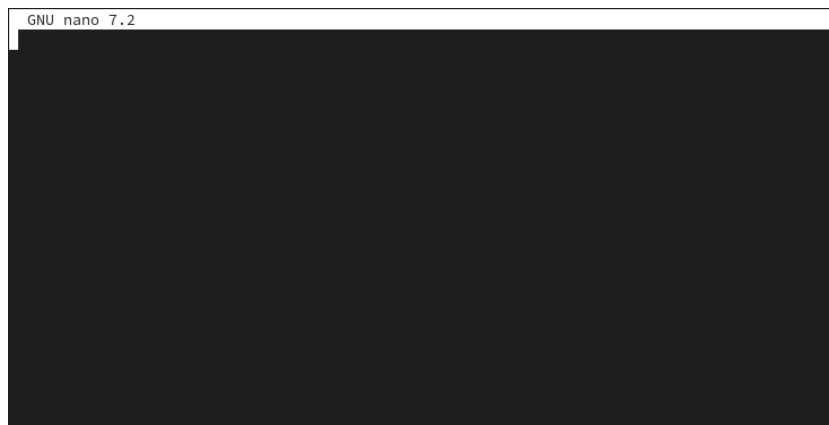


Рис. 4.5: Файл в редакторе

Я ввела в файл код программы для запроса строки(рис. [4.6]). Далее вышла из редактора,сохраняя изменения.

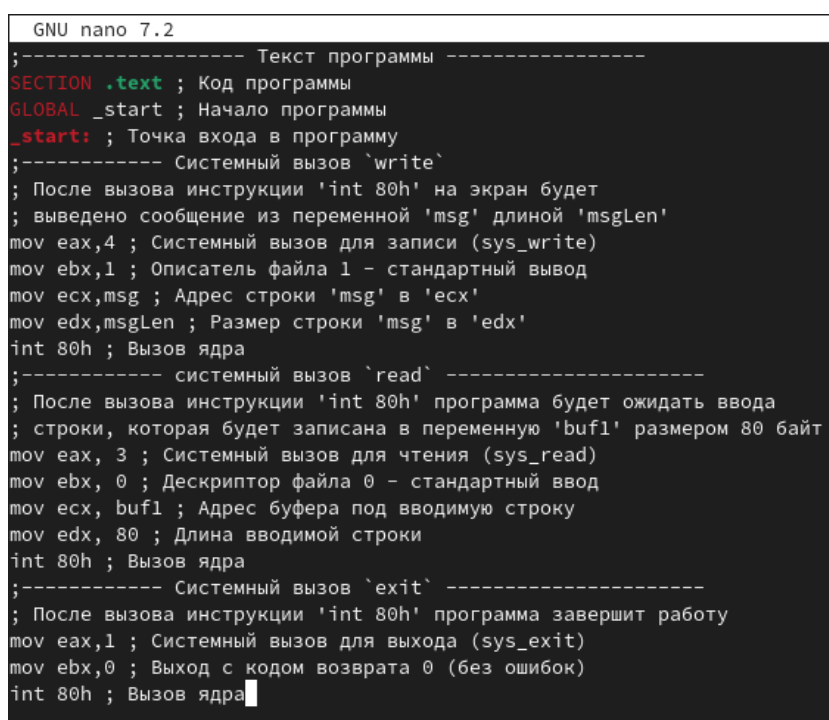


Рис. 4.6: Редактирование файла

Используя клавишу F4,я открыла файл для просмотра,чтобы проверить,сохранилась ли в нем написанная программа(рис. [4.7]).

```

/home/spolina/work/arch-pc/lab05/lab5-1.asm
;----- 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` -----
; 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 ; Вызов ядра

```

Рис. 4.7: Открытие файла для проверки

После этого я создала для текста программы объектный файл. Выполняю его компоновку(рис. [4.8]) (рис. [4.9]). После чего создался исполняемый файл lab5-1.

```

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

```

Рис. 4.8: Создание объектного файла

```

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

```

Рис. 4.9: Компоновка

Я запустила исполняемый файл. Программа выводит строку и ждет ввода с клавиатуры, после ввода своего ФИО программа завершает работу(рис. [4.10]).

```
[spolina@fedora lab05]$ ./lab5-1
Введите строку:
Самарханова Полина Тимуровна
```

Рис. 4.10: Запуск программы

4.3 Подключение внешнего файла.

Скачиваю файл in_out.asm со страницы ТУИС. Файл сохранился в 'Загрузки' (рис. [4.11]).

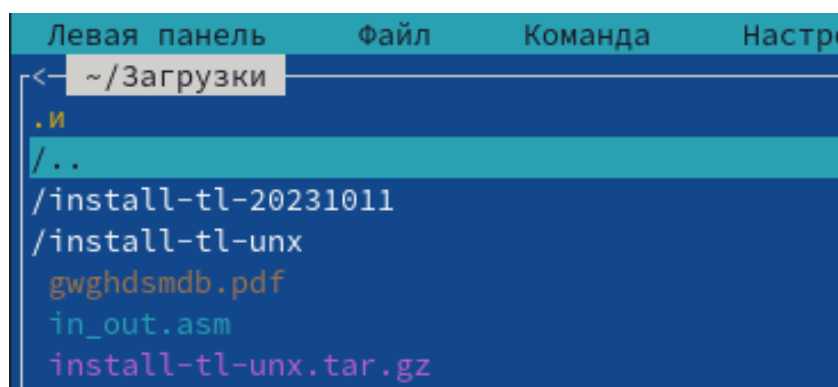


Рис. 4.11: Файл in_out.asm

Далее я копирую данный файл в каталог lab05, используя клавишу F5 (рис. [4.12]).

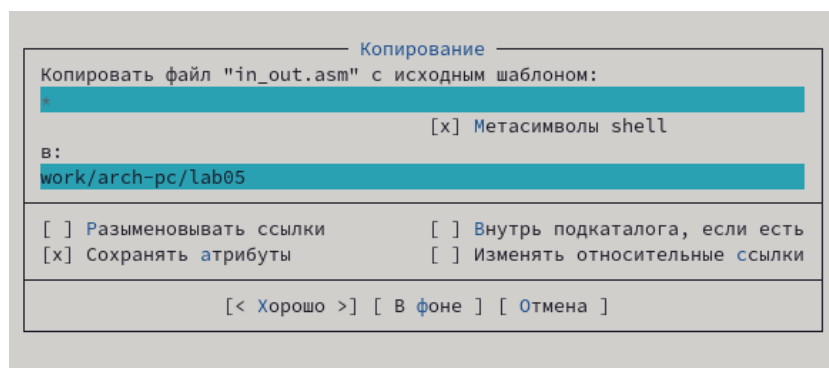
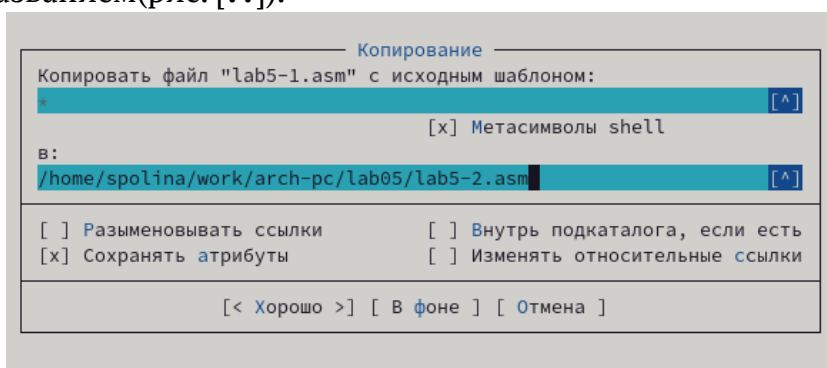


Рис. 4.12: Копирование файла в нужную директорию

С помощью той же утилиты F5 копировала файл lab5-1.asm, но уже с другим названием(рис. [??]).



После чего поменяла содержимое файла lab5-2.asm в редакторе nano, чтобы в программе использовались подпрограммы из внешнего файла in_out.asm(рис. [4.13]).

```

GNU nano 7.2
;-----;
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----;
#include 'in_out.asm' ; подключение внешнего файла
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 sread ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения

```

Рис. 4.13: Редактирование файла, для использования in_put.asm

Я создала объектный файл для lab5-2.asm(рис. [4.14]).

```

[spolina@fedora lab05]$ nasm -f elf lab5-2.asm

```

Рис. 4.14: Создание объектного файла

Я компоную данный файл, после чего создается исполняемый файл. Запустила его и проверила, работает ли данная программа(рис. [4.15]).

```

[spolina@fedora lab05]$ ld -m elf_i386 -o lab5-2 lab5-2.o
[spolina@fedora lab05]$ ./lab5-2
Введите строку:
Самарханова Полина Тимуровна

```

Рис. 4.15: Компоновка файла и запуск программы

Далее я открыла файл lab5-2.asm для редактирования в nano, используя F4. Изменила в нем подпрограмму sprintf на sprint, сохранила изменения и открыла файл для проверки(рис. [4.16]).

```

lab5-2.asm      [----] 11 L: [ 1+12 13/ 19] *(847 /1224b
;
; Программа вывода сообщения на экран и ввода строки с клави
;
-----
%include 'in_out.asm' ; подключение внешнего файла
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
call quit ; вызов подпрограммы завершения

```

Рис. 4.16: Редактирование файла

После я делаю компоновку объектного файла и запускаю новый исполняемый файл(рис. [4.17]).

```

Введите строку: 11111
[spolina@fedora lab05]$ ld -m elf_i386 -o lab5-2-2 lab5-2.o
[spolina@fedora lab05]$ mc

[spolina@fedora lab05]$ ./lab5-2-2
Введите строку: Самарханова Полина Тимуровна
[spolina@fedora lab05]$

```

Рис. 4.17: Исполнение файла

Вся разница заключается в том, что запуск с подпрограммой `sprintLF` запрашивает ввод с новой строки, а исполняемый файл с подпрограммой `sprint` просит ввод без переноса на новую строку.

4.4 Выполнение заданий для самостоятельной работы.

Я создаю копию файла `lab5-1.asm` с именем `lab5-1-1.asm` с помощью клавиши F5(рис. [4.18]).

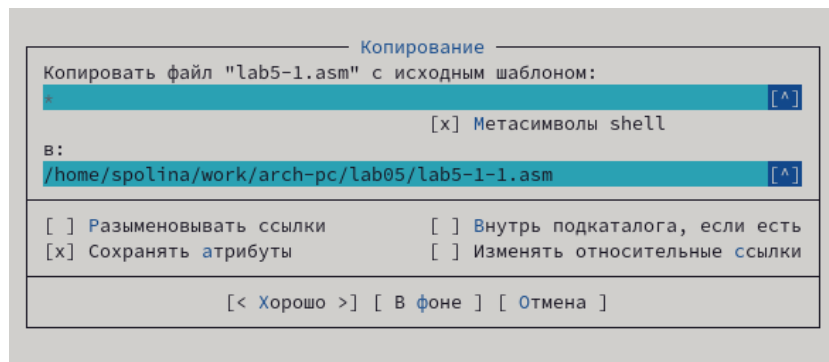


Рис. 4.18: Копирование файла lab5-1.asm

Используя клавишу F4 я открыла данный файл в nano и редактировала файл так,чтобы кроме вывода приглашения и запроса ввода, она выводила строку,которую пользователь ввел с клавиатуры(рис. [4.19]).

```
lab5-1-1.asm  [----]  0 L: [ 1+28 29/ 29] *(1557/1557b) <EOF>
SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициированных данных
buf1: RESB 80 ; Буфер размером 80 байт
;----- Текст программы -----
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 ;Descriptor 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 записи
mov ebx,1; Стандартный вывод
mov ecx,buf1; Адрес строки buf1 в ecx
mov edx,buf1; Размер строки
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
```

Рис. 4.19: Редактирование программы

Далее я создала объектный файл lab5-1-1.o и обработала его,используя

компоновщик, запустила созданный исполняемый файл, ввела своё имя. После этого программа вывела то, что я напечатал (рис. [4.20]).

```
[spolina@fedora lab05]$ nasm -f elf lab5-1-1.asm
[spolina@fedora lab05]$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
[spolina@fedora lab05]$ ./lab5-1-1
Введите строку:
Самарханова Полина Тимуровна
Самарханова Полина Тимуровна
```

Рис. 4.20: Исполнение файла

После я скопировала файл lab5-2.asm, используя F5, переименовала его в lab5-2-3.asm (рис. [4.21]).

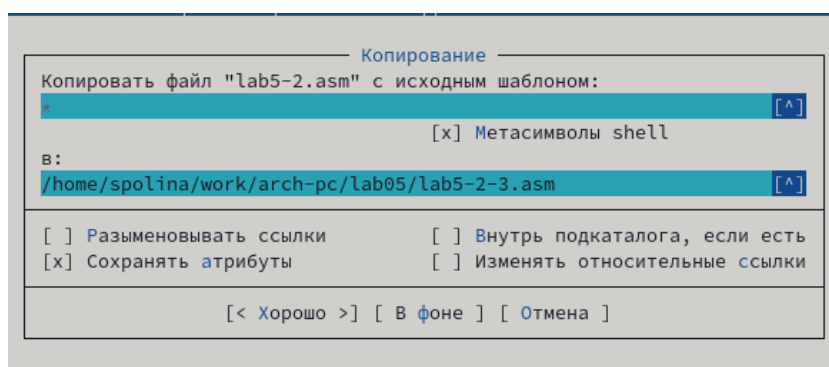


Рис. 4.21: Копирование файла

Используя клавишу F4 я открыла данный файл в nano и отредактировала файл так, чтобы кроме вывода приглашения и запроса ввода, она выводила строку, которую пользователь вводит с клавиатуры (рис. [4.22]).

```

lab5-2-3.asm      [-M--]  0 L:[ 1+18 19/ 19] *(1140/1140b) <
#include 'in_out.asm' ; подключение внешнего файла
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 ввода сообщения
mov eax,4; Системный вызов для записи (sys-write)
mov ebx,1; Стандартный вывод
mov ecx,buf1; Адрес строки в buf1 для ecx
int 80h; Вызов ядра
call quit ; вызов подпрограммы завершения

```

Рис. 4.22: Редактирование программы

И я создала объектный файл lab5-2-3.o и обработала его,используя компоновщик,запускаю созданный исполняемый файл, ввожу своё имя, после этого программа выводит то,что я напечатал(рис. [4.23]).

```

[spolina@fedora lab05]$ nasm -f elf lab5-2-3.asm
[spolina@fedora lab05]$ ld -m elf_i386 -o lab5-2-3 lab5-2-3.o
[spolina@fedora lab05]$ ./lab5-2-3
Введите строку: Самарханова Полина Тимуровна
Самарханова Полина Тимуровна

```

Рис. 4.23: Исполнение файла

5 Выводы

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

Список литературы

::: {#Лабораторная работа №5} :::