

# **Лабораторная работа №6.**

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

Боровиков Даниил Александрович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>5</b>
<b>3</b>	<b>Самостоятельная работа</b>	<b>12</b>
<b>4</b>	<b>Выводы</b>	<b>16</b>

## Список иллюстраций

2.1	Midnight Commander . . . . .	5
2.2	Переход в каталог ~/work/arch-pc . . . . .	6
2.3	Переход в каталог ~/work/arch-pc/lab06 . . . . .	6
2.4	Создание файла lab6-1.asm . . . . .	7
2.5	lab6-1.asm в редакторе nano . . . . .	7
2.6	Ввод текста программы с последующим сохранением . . . . .	8
2.7	Проверка сохранения файла . . . . .	9
2.8	Запуск программы . . . . .	9
2.9	копирование файла in_out.asm . . . . .	10
2.10	Копирование файла lab6-1.asm . . . . .	10
2.11	Включаем функции в программу lab6-2.asm . . . . .	11
2.12	Проверка работы исполняемого файла . . . . .	11
2.13	Файл lab6-2.asm с вводом и выводом на одной строке . . . . .	11
3.1	Код программы без использования подпрограмм . . . . .	13
3.2	Запуск программы без использования подпрограмм . . . . .	14
3.3	Код программы с использованием подпрограмм . . . . .	15
3.4	Запуск программы с использованием подпрограмм . . . . .	15

# 1 Цель работы

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

## 2 Выполнение лабораторной работы

Откроем Midnight Commander(рис. 2.1)

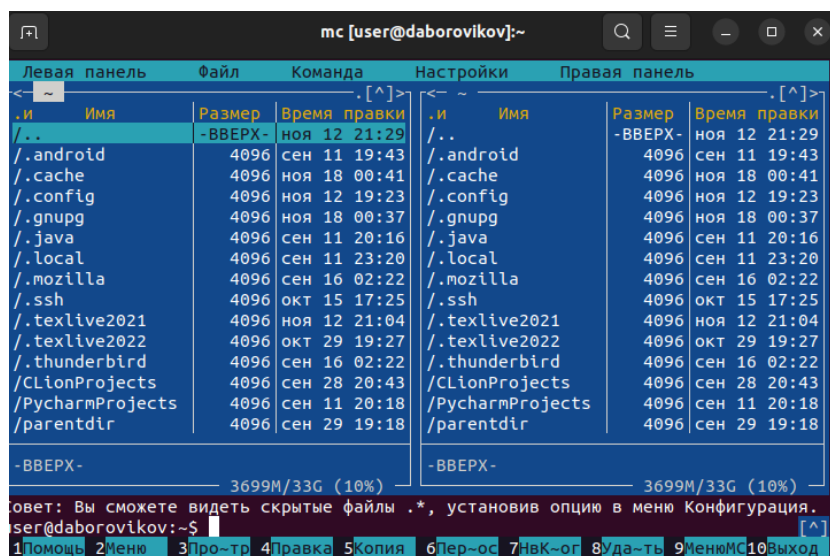


Рис. 2.1: Midnight Commander

Перейдем в каталог ~/work/arch-рс созданный при выполнении лабораторной работы №5(рис. 2.2)

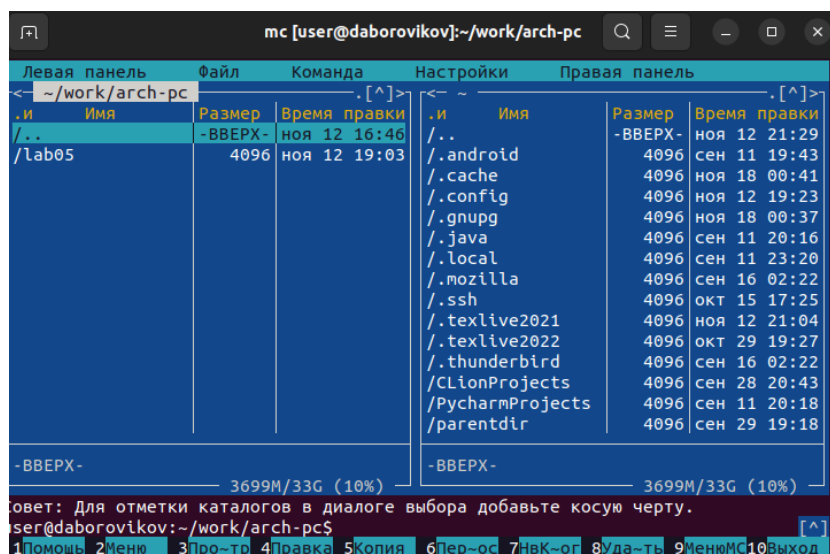


Рис. 2.2: Переход в каталог ~/work/arch-pc

С помощью функциональной клавиши F7 создадим папку lab06 и перейдем в созданный каталог.(рис. 2.3)

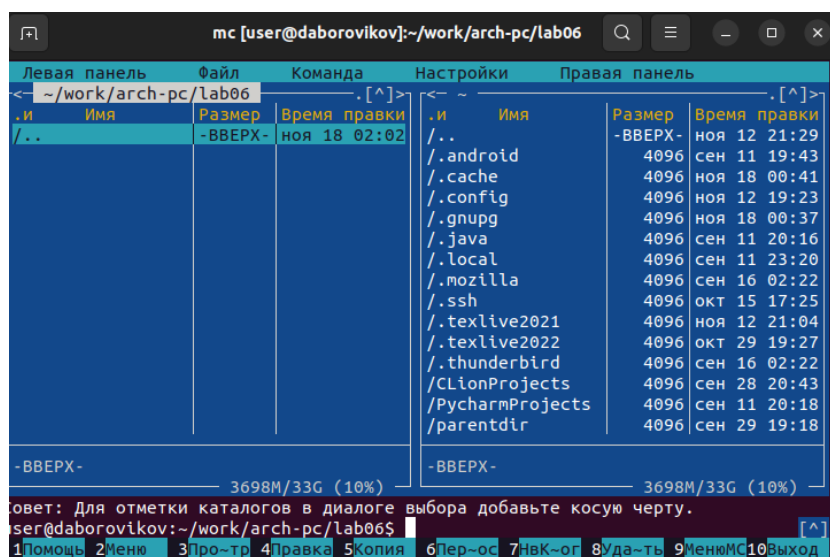


Рис. 2.3: Переход в каталог ~/work/arch-pc/lab06

Пользуясь строкой ввода и командой touch создадим файл lab6-1.asm(рис. 2.4)

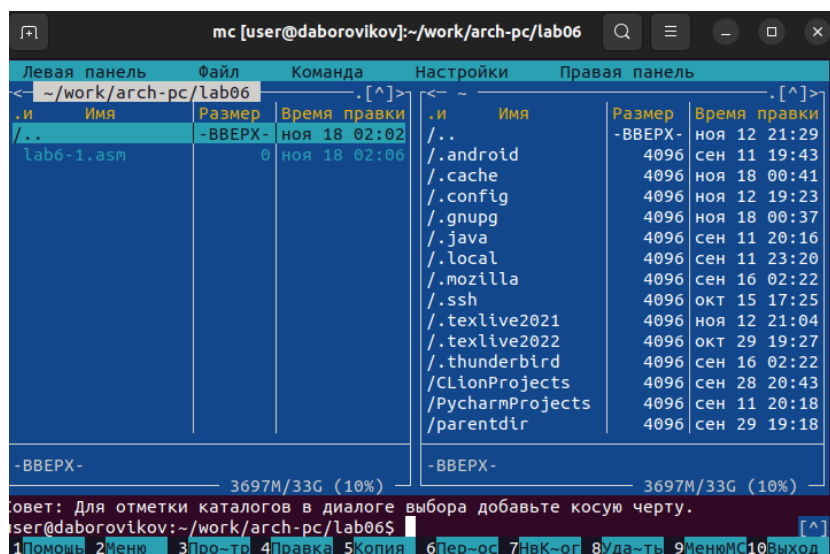


Рис. 2.4: Создание файла lab6-1.asm

С помощью функциональной клавиши F4 откроем файл lab6-1.asm для редактирования во встроенном редакторе. В качестве редактора выберем nano(рис. 2.5)

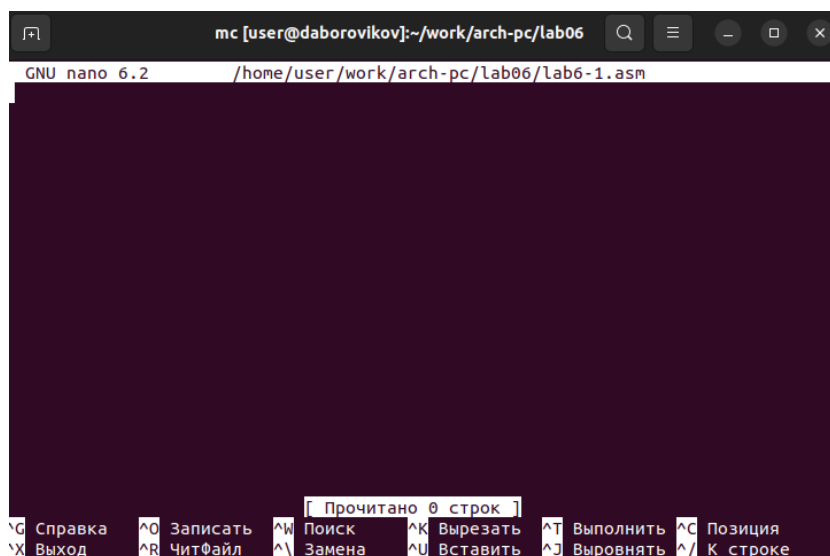
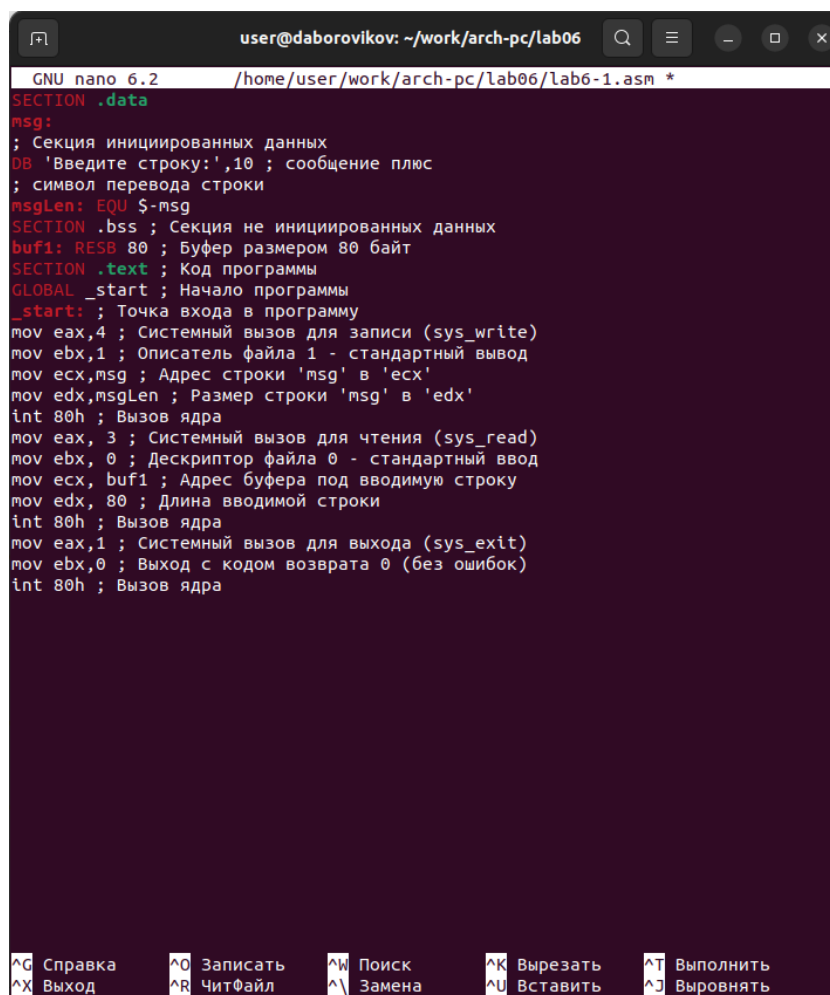


Рис. 2.5: lab6-1.asm в редакторе nano

Введем текст программы из листинга 6.1, создадим его и закроем файл(рис. 2.6)



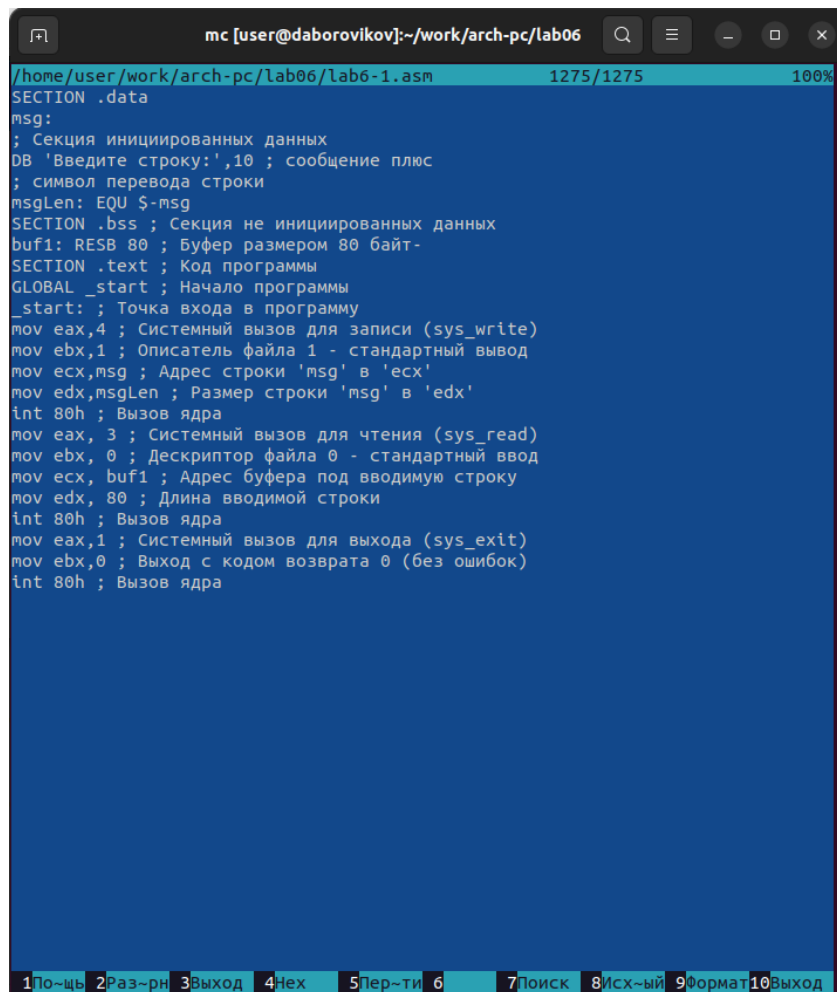
```
GNU nano 6.2 /home/user/work/arch-pc/lab06/lab6-1.asm *
SECTION .data
msg:
; Секция иницированных данных
DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg
SECTION .bss ; Секция не иницированных данных
buf1: RESB 80 ; Буфер размером 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
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,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра

^G Справка      ^O Записать
^X Выход        ^R ЧитФайл
^_              ^W Поиск
               ^K Вырезать
               ^U Вставить
               ^T Выполнить
               ^J Выровнять
```

Рис. 2.6: Ввод текста программы с последующим сохранением

С помощью функциональной клавиши F3 откроем файл lab6-1.asm для просмотра и убедимся, что файл содержит текст программы.(рис. 2.7)



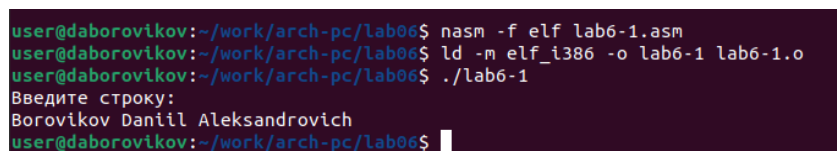


```
mc [user@daborovikov]:~/work/arch-pc/lab06
/home/user/work/arch-pc/lab06/lab6-1.asm 1275/1275 100%
SECTION .data
msg:
; Секция иницированных данных
DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg
SECTION .bss ; Секция не иницированных данных
buf1: RESB 80 ; Буфер размером 80 байт-
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
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,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра

1По-щ 2Раз-рн 3Выход 4Нех 5Пер-ти 6 7Поиск 8Исх-ый 9Формат10Выход
```

Рис. 2.7: Проверка сохранения файла

Оттранслируем текст программы lab6-1.asm в объектный файл. Выполним компоновку объектного файла и запустим получившийся исполняемый файл. Программа выводит строку ‘Введите строку:’ и ожидает ввода с клавиатуры. На запрос вводим ФИО.(рис. 2.8)



```
user@daborovikov:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
user@daborovikov:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
user@daborovikov:~/work/arch-pc/lab06$ ./lab6-1
Введите строку:
Borovikov Daniil Aleksandrovich
user@daborovikov:~/work/arch-pc/lab06$
```

Рис. 2.8: Запуск программы

Скопируем файл in\_out.asm в каталог с файлом lab6-1.asm с помощью функци-

ональной клавиши F5(рис. 2.9)

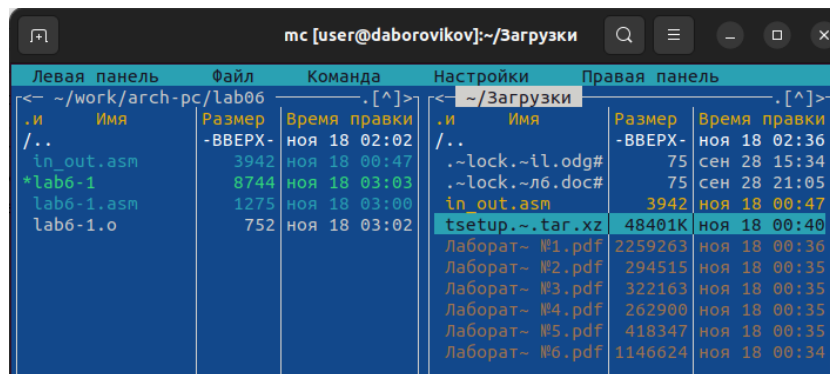


Рис. 2.9: копирование файла in\_out.asm

С помощью функциональной клавиши F6 создадим копию файла lab6-1.asm с именем lab6-2.asm.(рис. 2.10)

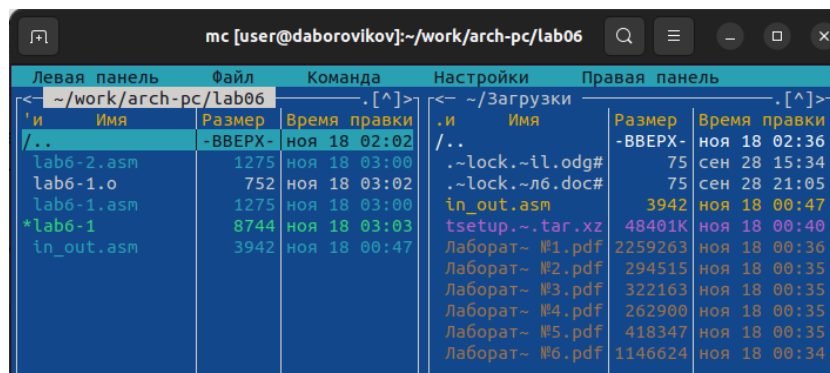
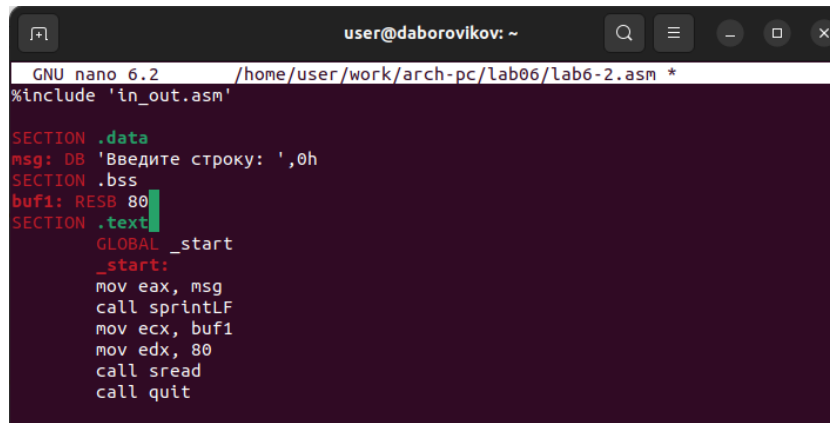


Рис. 2.10: Копирование файла lab6-1.asm

Исправьте текст программы в файле lab6-2.asm с использованием подпрограмм из внешнего файла in\_out.asm (используйте подпрограммы sprintLF, sread и quit) в соответствии с листингом 6.2.(рис. 2.11)

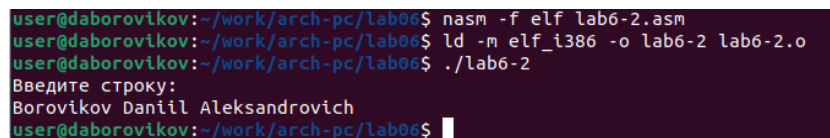


```
GNU nano 6.2 /home/user/work/arch-pc/lab06/lab6-2.asm *
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите строку: ',0h
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
    mov eax, msg
    call sprintf
    mov ecx, buf1
    mov edx, 80
    call sread
    call quit
```

Рис. 2.11: Включаем функции в программу lab6-2.asm

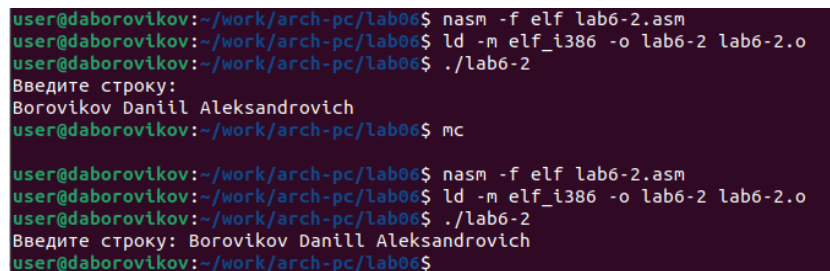
Создадим исполняемый файл и проверим его работу(рис. 2.12)



```
user@daborovikov:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
user@daborovikov:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
user@daborovikov:~/work/arch-pc/lab06$ ./lab6-2
Введите строку:
Borovikov Daniil Aleksandrovich
user@daborovikov:~/work/arch-pc/lab06$
```

Рис. 2.12: Проверка работы исполняемого файла

В файле lab6-2.asm заменим подпрограмму sprintf на sprintf. Создадим исполняемый файл и проверьте его работу. Выясним, что от первоначальной версии полученная программа отличается вводом и выводом на одной строке(рис. 2.13)



```
user@daborovikov:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
user@daborovikov:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
user@daborovikov:~/work/arch-pc/lab06$ ./lab6-2
Введите строку:
Borovikov Daniil Aleksandrovich
user@daborovikov:~/work/arch-pc/lab06$ mc

user@daborovikov:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
user@daborovikov:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
user@daborovikov:~/work/arch-pc/lab06$ ./lab6-2
Введите строку: Borovikov Daniil Aleksandrovich
user@daborovikov:~/work/arch-pc/lab06$
```

Рис. 2.13: Файл lab6-2.asm с вводом и выводом на одной строке

### 3 Самостоятельная работа

Программа работающая по алгоритму без использования подпрограмм:

вывести приглашение типа “Введите строку:”;

ввести строку с клавиатуры;

вывести введенную строку на экран.(рис. 3.1)

Листинг программы:

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

```

```

GNU nano 6.2 /home/user/work/arch-pc/lab06/lab6-3.asm
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10
msgLen: EQU $-msg ; Длина переменной 'msg'
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,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

```

Рис. 3.1: Код программы без использования подпрограмм

Получим исполняемый файл требуемой программы и проверим его работу. На приглашение ввести строку введем свою фамилию(рис. 3.2)

```
user@daborovikov:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
user@daborovikov:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
user@daborovikov:~/work/arch-pc/lab06$ ./lab6-3
Введите строку:
Borovikov Daniil Aleksandrovich
Borovikov Daniil Aleksandrovich
user@daborovikov:~/work/arch-pc/lab06$ mc
user@daborovikov:~/work/arch-pc/lab06$
```

Рис. 3.2: Запуск программы без использования подпрограмм

Программа работающая по алгоритму с использованием подпрограмм:

вывести приглашение типа “Введите строку:”;

ввести строку с клавиатуры;

вывести введенную строку на экран.(рис. 3.3)

Листинг программы:

%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

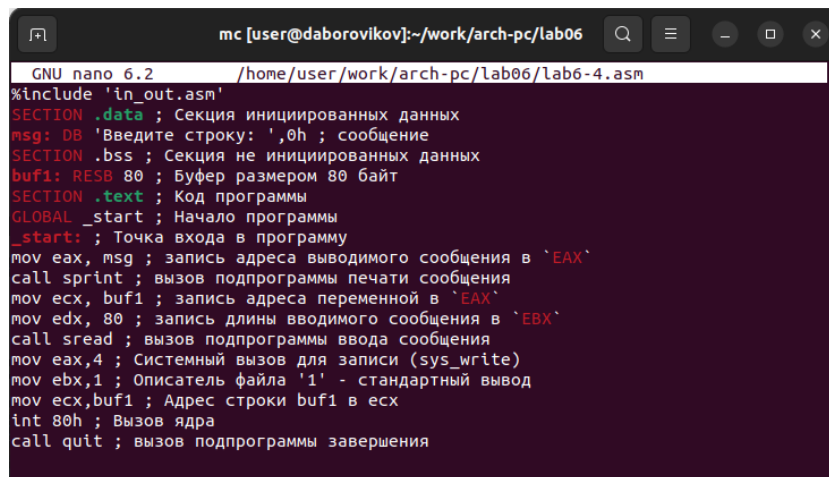
mov eax,4 ; Системный вызов для записи (sys\_write)

mov ebx,1 ; Описатель файла ‘1’ - стандартный вывод

mov ecx,buf1 ; Адрес строки buf1 в ecx

int 80h ; Вызов ядра

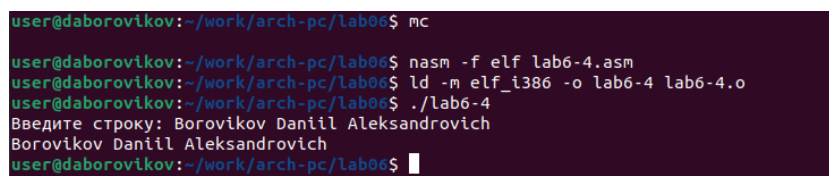
call quit ; вызов подпрограммы завершения



```
mc [user@daborovikov]:~/work/arch-pc/lab06
GNU nano 6.2 /home/user/work/arch-pc/lab06/lab6-4.asm
#include 'in_out.asm'
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 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. 3.3: Код программы с использованием подпрограмм

Получим исполняемый файл требуемой программы и проверим его работу. На приглашение ввести строку введем свою фамилию(рис. 3.4)



```
user@daborovikov:~/work/arch-pc/lab06$ mc
user@daborovikov:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
user@daborovikov:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
user@daborovikov:~/work/arch-pc/lab06$ ./lab6-4
Введите строку: Borovikov Daniil Aleksandrovich
Borovikov Daniil Aleksandrovich
user@daborovikov:~/work/arch-pc/lab06$
```

Рис. 3.4: Запуск программы с использованием подпрограмм

Ссылка на github: [https://github.com/daBorovikov/study\\_2022-2023\\_arh-pc-](https://github.com/daBorovikov/study_2022-2023_arh-pc-)

## 4 Выводы

В ходе лабораторной работы мы приобрели практические навыки работы в Midnight Commander, освоили инструкции языка ассемблера `mov` и `int`.