

Лабораторная работа №5

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

Демин Даниил

Содержание

1	Цель работы	3
2	Выполнение лабораторной работы	4
3	Выполнение самостоятельной работы	13
4	Выводы	18

1 Цель работы

Целью работы является приобретение практических навыков работы в Midnight Commander. Освоение инструкций языка ассемблера `mov` и `int`.

2 Выполнение лабораторной работы

Открыл mc (рис. 2.1).

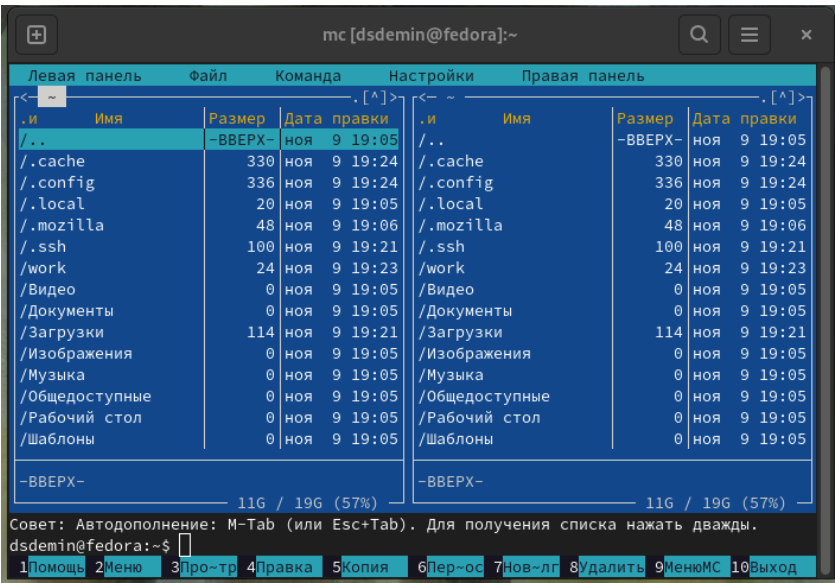


Рис. 2.1: Интерфейс mc

Перешел в папку ~/work/arch-рс при помощи клавиш, и создал папку lab5, клавишей f7 (рис. 2.2).

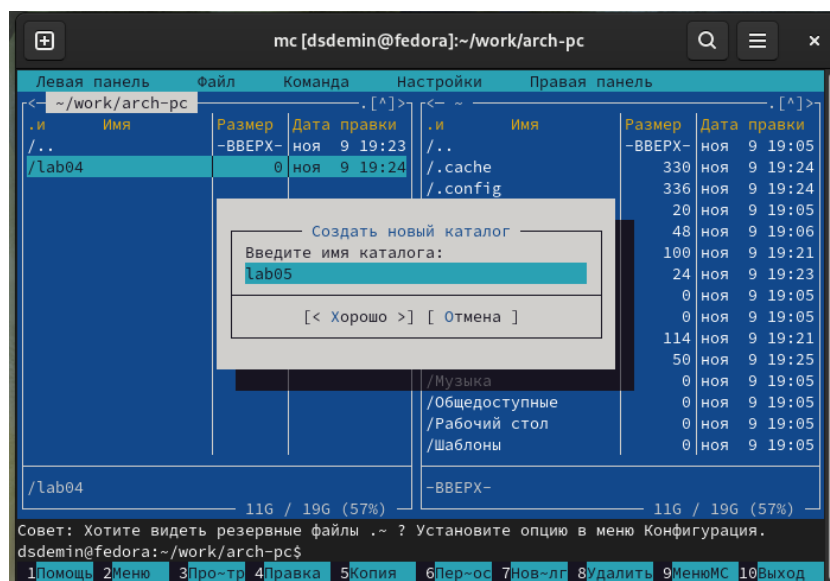


Рис. 2.2: Создание папки lab05

Перешел в папку и написал команду для создания файла (рис. 2.3).

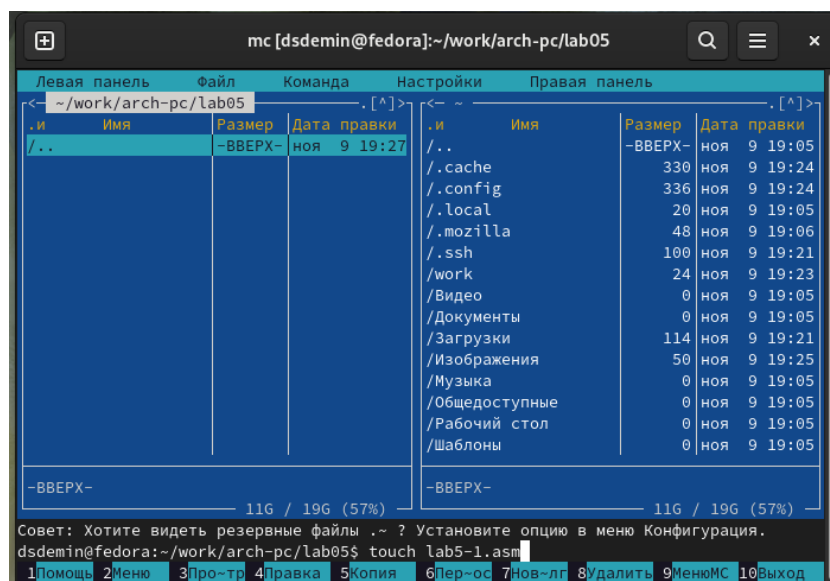


Рис. 2.3: Новая папка и команда

Открыл редактор mcedit клавишей f4 (рис. 2.4).

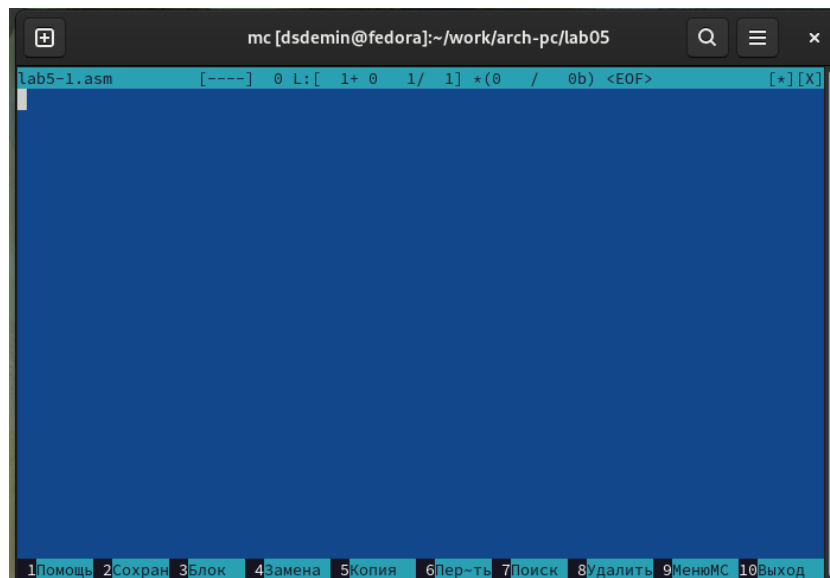


Рис. 2.4: Редактор mcedit

Переписал код, листинг приведен ниже, сохранил и проверил содержимое клавишей f3 (рис. 2.5).

Листинг кода:

```
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
SECTION .data                ; Секция инициированных данных
msg: DB 'Введите строку: ',10 ; сообщение

msgLen: EQU $-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
```

;----- 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
mov ebx, 0
mov ecx, buf1    ; запись адреса переменной в `EAX`
mov edx, 80
int 80h    ; запись длины вводимого сообщения в `EBX`
```

;----- 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
```

```

/home/dsdemin/work/arch-pc/lab05/lab5-1.asm 1305/2266 57%
;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
SECTION .data                ; Секция инициализированных данных
msg: DB 'Введите строку: ',10 ; сообщение

msgLen: EQU $-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

;----- 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'

```

Рис. 2.5: Содержимое файла

Создал объектный файл lab5-1.o при помощи ассемблера nasm. При помощи объектно компоновщика ld сделал исполняемый файл lab5-1 (рис. 2.6).

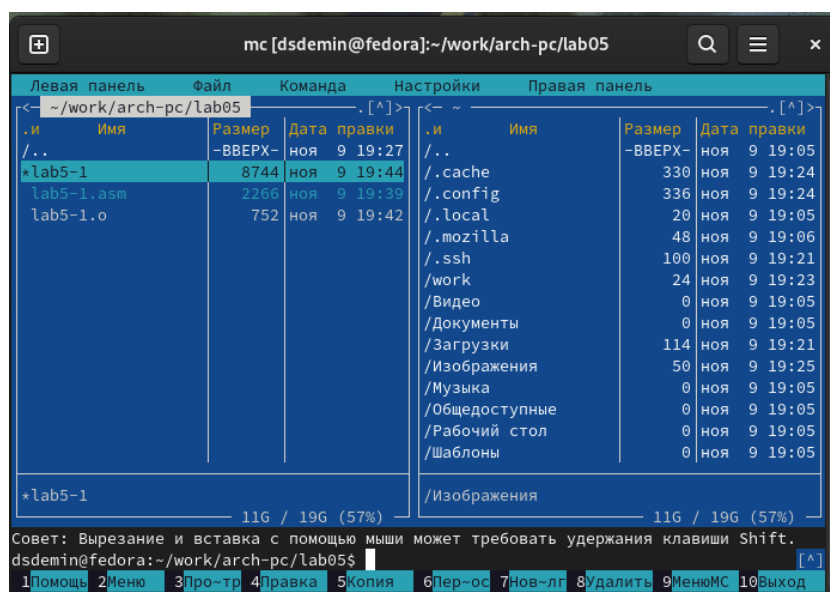


Рис. 2.6: Транслированный и исполняемый файлы

Запустил исполняемый файл, программа работает корректно(рис. 2.7).


```

dsdemin@fedora:~/work/arch-pc/lab05$ mc
dsdemin@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
dsdemin@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 la5-1.0 -o lab5-1
ld: невозможно найти la5-1.0: Нет такого файла или каталога
dsdemin@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 la5-1.o -o lab5-1
ld: невозможно найти la5-1.o: Нет такого файла или каталога
dsdemin@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab5-1.o -o lab5-1
dsdemin@fedora:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Демин Даниил

```

Рис. 2.7: Файл работает

Скачал и перенес в директорию файл in_out.asm (рис. 2.8).

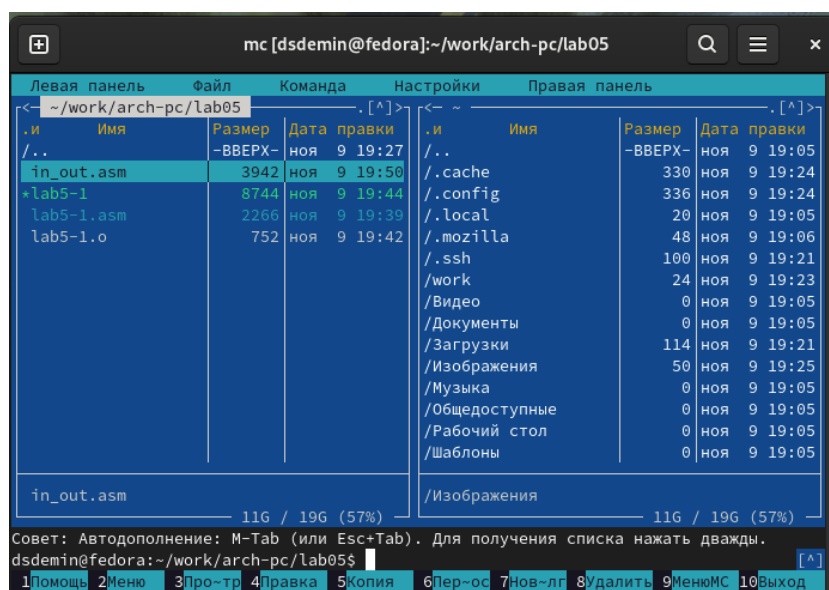


Рис. 2.8: Наличие файла

Сделал копию файла lab5-1.asm при помощи клавиши f5 (рис. 2.9).

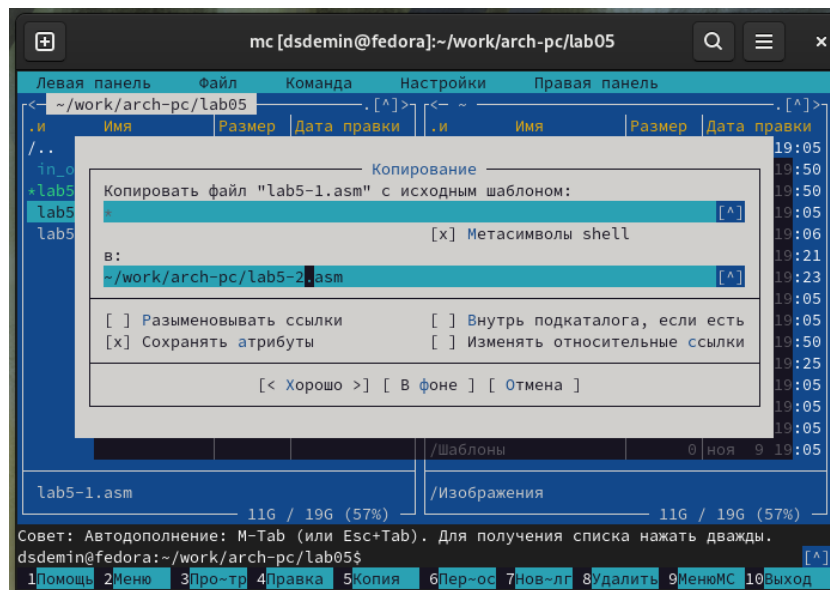


Рис. 2.9: Копирование файла

Переписал код в файле lab5-2.asm и сохранил содержимое, листинг код приведен ниже (рис. 2.10).

Листинг кода:

```

;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----

%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 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

```

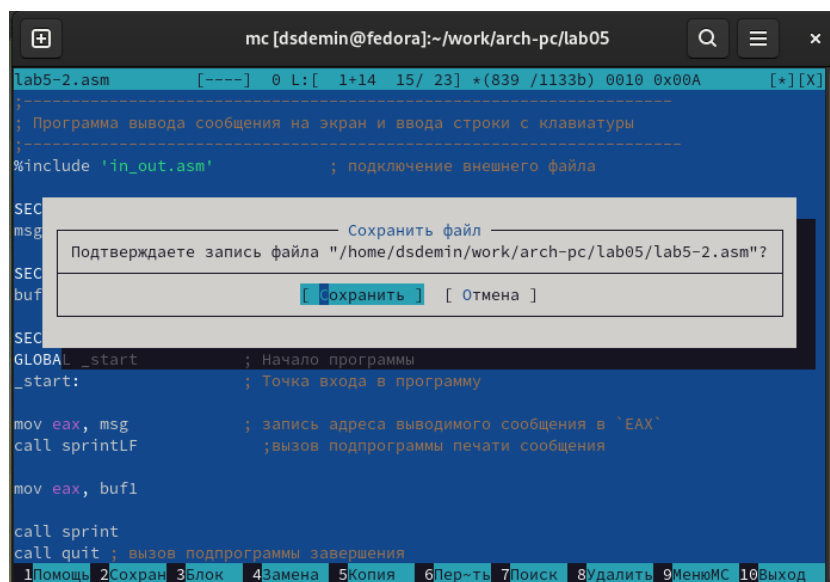


Рис. 2.10: Сохранение изменений

Создал объектный файл lab5-2.o при помощи ассемблера nasm. При помощи объектно компоновщика ld сделал исполняемый файл lab5-2. (рис. 2.11).

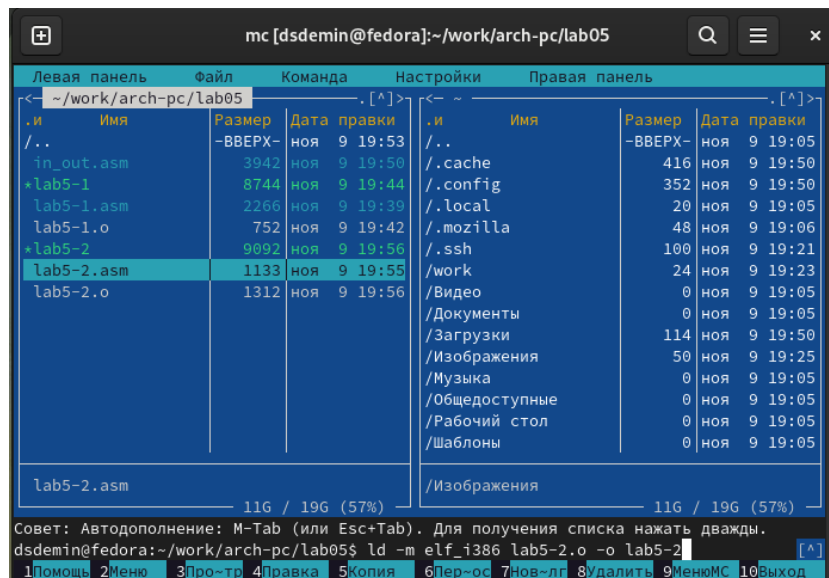


Рис. 2.11: Исполняемый файл

Проверил работу исполняемого файла lab5-2 (рис. 2.12).

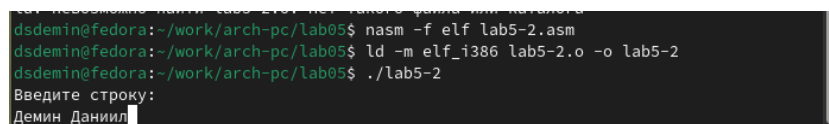


Рис. 2.12: Работает файла

Заменил в коде строку call stringLF на call string (рис. 2.13).

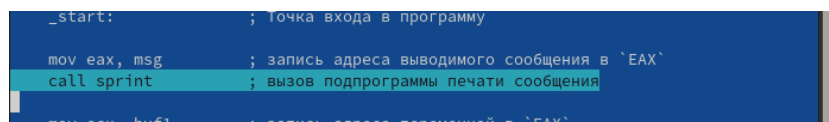


Рис. 2.13: Листинг кода

Проанализировал на результат работы, команда stringLF после строки переносит строку, команда string оставляет предыдущий уровень, поэтому фамилия и имя остались на той же строке (рис. 2.14).

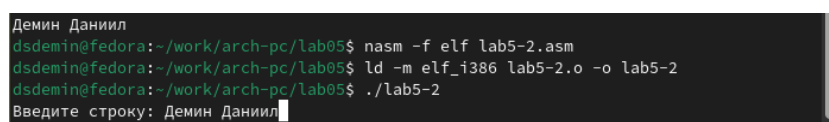


Рис. 2.14: Работа измененной программы

3 Выполнение самостоятельной работы

Создал копию файла lab5-1.asm с названием lab5-1_copy.asm (рис. 3.1).

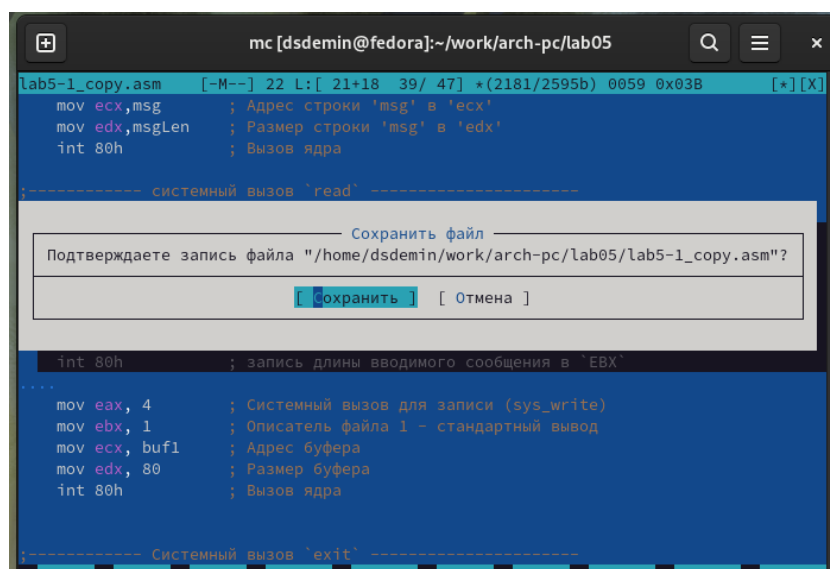


Рис. 3.1: Результат выполнения

Внес изменения в программу так, что бы после ввода данных с клавиатуры, программа выводила их на экран. Листинг обновленного кода ниже.

Листинг кода:

```
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
SECTION .data                                ; Секция инициированных данных
msg: DB 'Введите строку: ',10               ; сообщение
```

```
msgLen: EQU $-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
```

```
;----- 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
```

```
mov ebx, 0
```

```
mov ecx, buf1 ; запись адреса переменной в `EAX`
```

```
mov edx, 80
```

```
int 80h ; запись длины вводимого сообщения в `EBX`
```

```

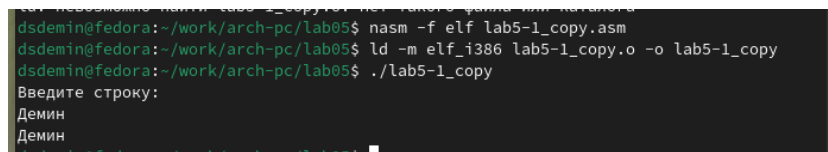
mov eax, 4      ; Системный вызов для записи (sys_write)
mov ebx, 1      ; Описатель файла 1 - стандартный вывод
mov ecx, buf1   ; Адрес буфера
mov edx, 80     ; Размер буфера
int 80h         ; Вызов ядра

;----- 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

```

Сделал трансляцию, компоновку и запустил и проверил работоспособность программы. Программа работает корректно (рис. 3.2).



```

dsdemin@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-1_copy.asm
dsdemin@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab5-1_copy.o -o lab5-1_copy
dsdemin@fedora:~/work/arch-pc/lab05$ ./lab5-1_copy
Введите строку:
Демин

```

Рис. 3.2: Фамилия выводится в консоли

Создал копию файла lab5-1.asm с названием lab5-1_copy.asm (рис. 3.3).

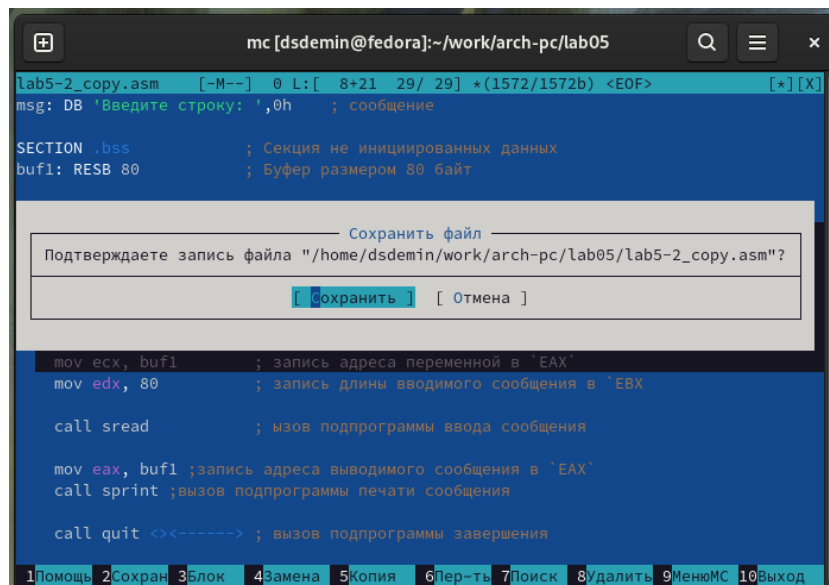


Рис. 3.3: Результат выполнения

Внес изменения в программу так, что бы после ввода данных с клавиатуры, программа выводила их на экран. Листинг обновленного кода ниже.

Листинг кода:

```

;-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----

%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 sread ; вызов подпрограммы ввода сообщения

mov eax, buf1          ; запись адреса выводимого сообщения в `EAX`
call sprintf          ; вызов подпрограммы печати сообщения

call quit ; вызов подпрограммы завершения

```

Сделал трансляцию, компоновку и запустил и проверил работоспособность программы. Программа работает корректно (рис. 3.4).



```

dsdemin@fedora:~/work/arch-pc/lab05$ nasm -f elf lab5-2_copy.asm
dsdemin@fedora:~/work/arch-pc/lab05$ ld -m elf_i386 lab5-2_copy.o -o lab5-2_copy
dsdemin@fedora:~/work/arch-pc/lab05$ ./lab5-2_copy
Введите строку:
Демин
dsdemin@fedora:~/work/arch-pc/lab05$

```

Рис. 3.4: Фамилия выводится в консоли

4 Выводы

Выполнив данную лабораторную работу я обрел теоретические и практические знания в использовании Midnight Commander. Освоил инструкции языка ассемблера `mov` и `int`.