

ОТЧЕТ по лабораторной работе № 5

дисциплина: Архитектура компьютера

Студент: Идрисов Д.А

Содержание

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-байтное 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.

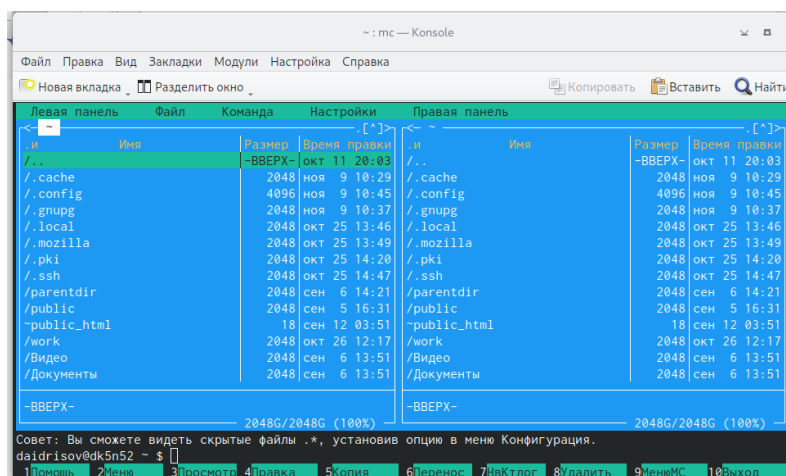
```
int n
```

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

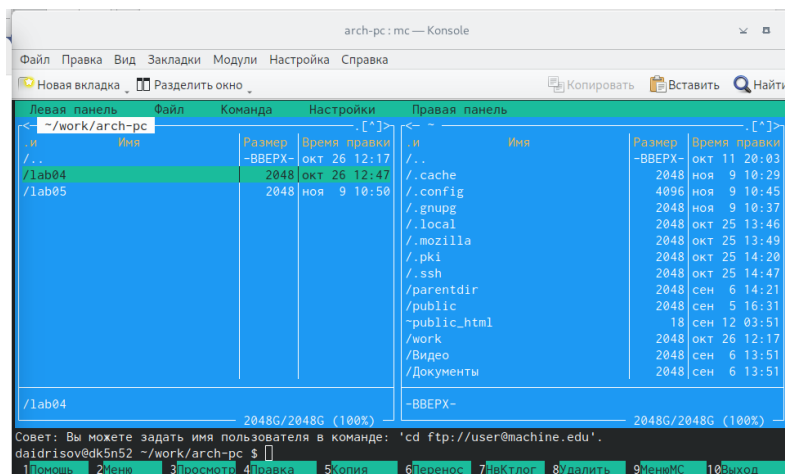
4.1 1 Основы работы с mc.

Я запускаю Midnight Commander, используя команду “mc”. (рис. ??).



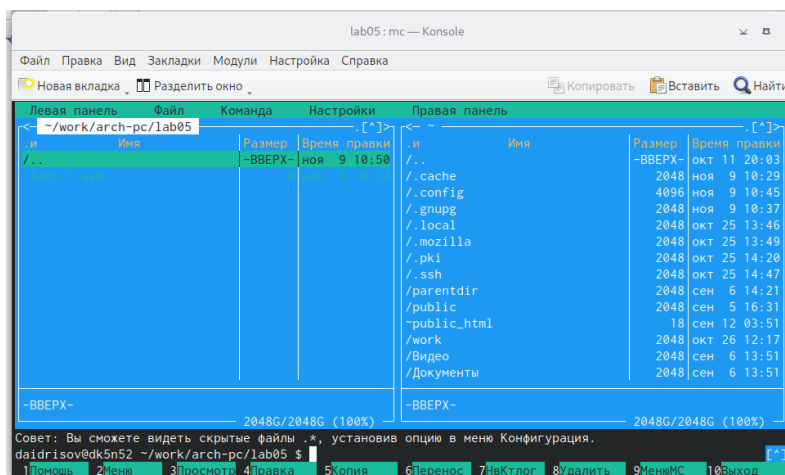
Открытый Midnight Commander

Перехожу в каталог ~/work/arch-pc (рис. ??).



Выполню переход

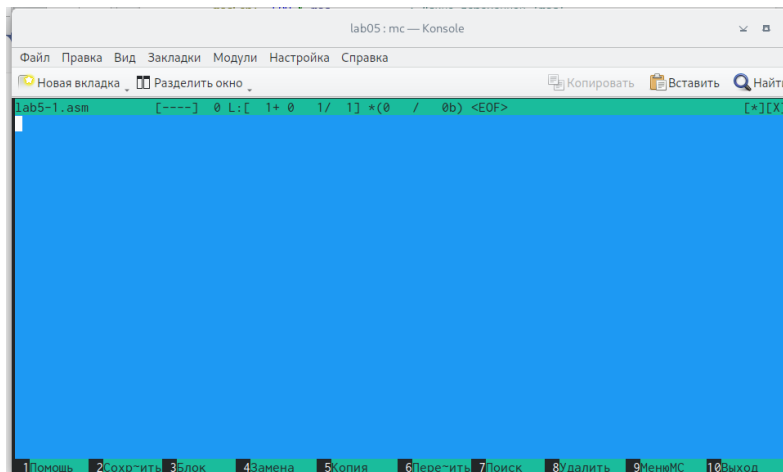
Создаю папку с именем "lab05" с помощью функциональной клавиши F7, и после этого выполняю команду "touch lab5-1.asm" (рис. ??).



Создание каталога

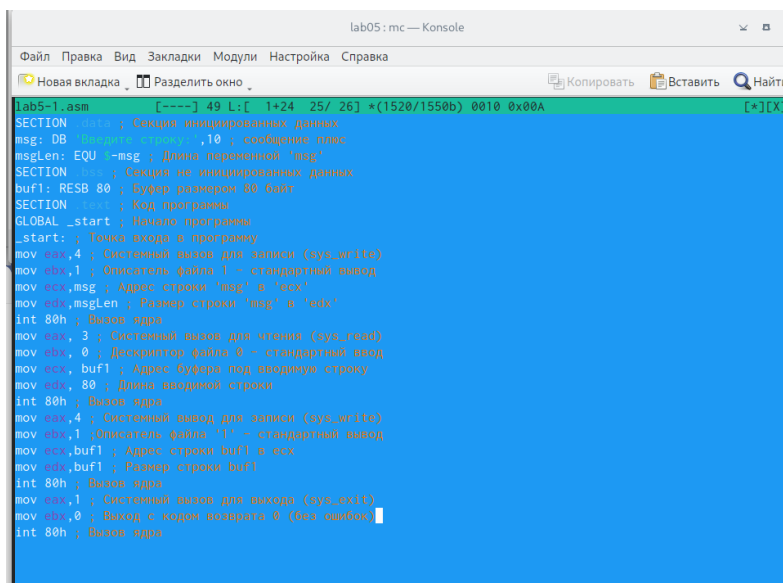
4.2 2 Структура программы на языке ассемблера NASM.

Используя функциональную клавишу F4, открою файл в текстовом редакторе nano для редактирования. (рис. ??).



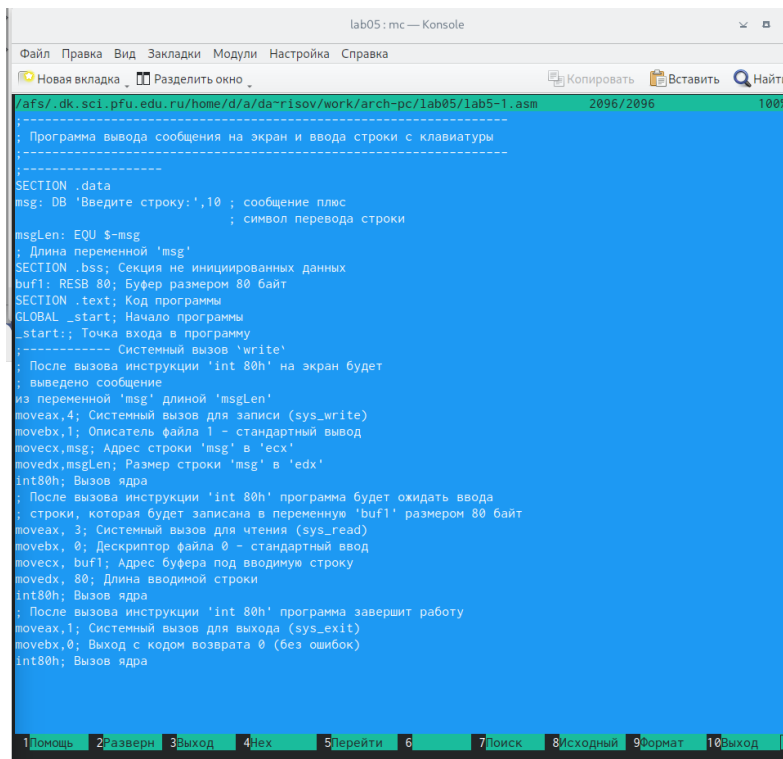
Открытие файла для редактирования

Добавлю в файл программный код, который будет запрашивать строку у пользователя. (рис. ??).



Редактирование файла

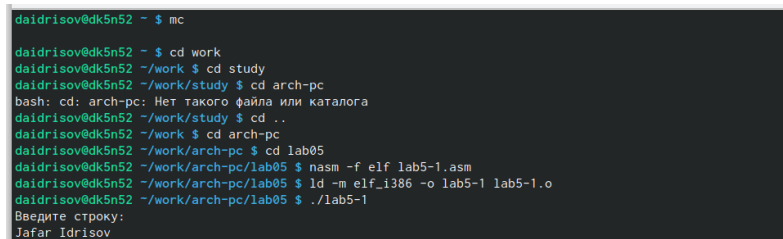
Используя функциональную клавишу F3, мы можем открыть файл для просмотра и проверить его содержимое, чтобы убедиться в наличии текста программы. (рис. ??).



```
lab05: mc — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
[Новая вкладка] [Разделить окно] [Копировать] [Вставить] [Найти]
~/afs/.dk.sci.pfu.edu.ru/home/daidrisov/work/arch-pc/lab05/lab5-1.asm 2096/2096 100%
-----
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;
;-----
SECTION .data
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg
; Длина переменной '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
;----- 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'
int80h; Вызов ядра
; 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; Дескриптор файла 0 - стандартный ввод
mov ecx,buf1; Адрес буфера под вводимую строку
mov edx,80; Длина вводимой строки
int80h; Вызов ядра
; 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)
int80h; Вызов ядра
1Помощь 2Разверн 3Выход 4Тех 5Перейти 6 7Поиск 8Сходный 9Формат 10Выход
```

Просмотр текста программы

Я проведу процесс компиляции моей программы, начиная с трансляции текста программы в объектный файл с использованием команды “nasm -f elf lab5-1.asm”. Затем выполню компоновку файла с помощью команды “ld -m elf_i386 -o lab5-1 lab5-1.o”. Наконец, запущу программу с помощью команды “./lab5-1”. (рис. ??).

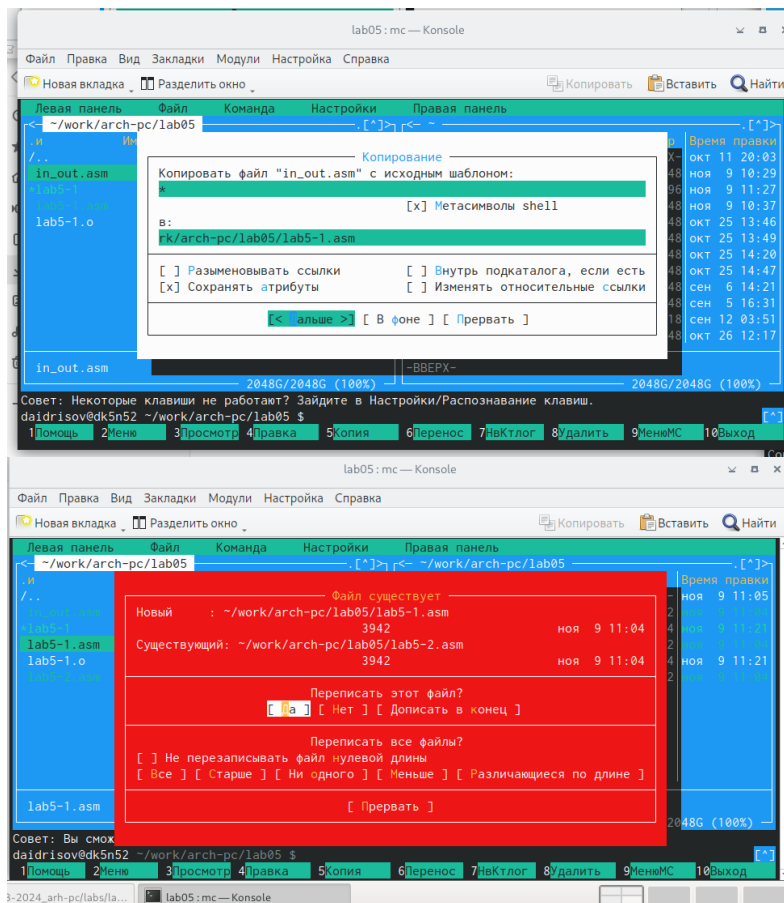


```
daidrisov@dk5n52 ~ - $ mc
daidrisov@dk5n52 ~ - $ cd work
daidrisov@dk5n52 ~/work $ cd study
daidrisov@dk5n52 ~/work/study $ cd arch-pc
bash: cd: arch-pc: Нет такого файла или каталога
daidrisov@dk5n52 ~/work/study $ cd ..
daidrisov@dk5n52 ~/work $ cd arch-pc
daidrisov@dk5n52 ~/work/arch-pc $ cd lab05
daidrisov@dk5n52 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1.asm
daidrisov@dk5n52 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1 lab5-1.o
daidrisov@dk5n52 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:
Jafar Idrisov
```

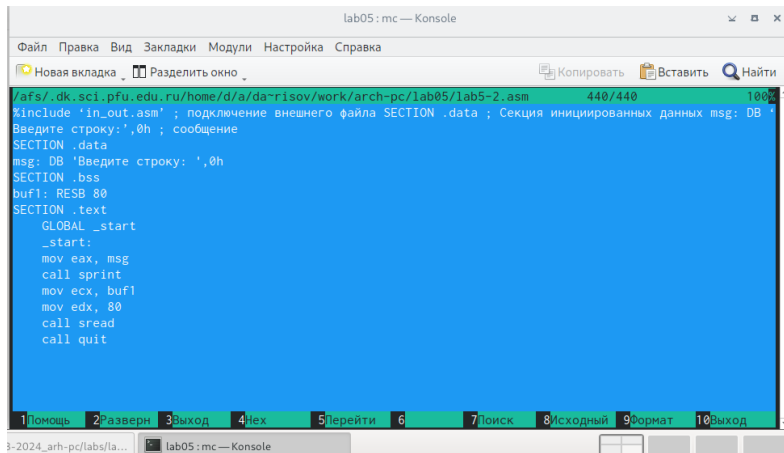
Трансляция и запуск

4.3 3.Подключение внешнего файла

Я загружаю файл на ТУИС и проверяю, что он был успешно загружен в папке, где лежит наша программа. Сразу же выполняю копирование этого файла в программу lab5-1.asm (рис. ??) (рис. ??).



Изменяю текст программы и просматриваю код (рис. ??).



Изменение кода

Запускаю программу (рис. ??).

```

lab05: bash — Konsole
Файл Правка Вид Закладки Модули Настройка Справка
Новая вкладка Разделить окно Копировать Вставить Найти
daidrisov@dk5n52 ~/work $ cd arch-pc
daidrisov@dk5n52 ~/work/arch-pc $ cd lab05
daidrisov@dk5n52 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1.asm
daidrisov@dk5n52 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1 lab5-1.o
daidrisov@dk5n52 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:
Jafar Idrisov
Jafar Idrisov
daidrisov@dk5n52 ~/work/arch-pc/lab05 $ mc

daidrisov@dk5n52 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
lab5-2.asm:1: error: '%include' expects a file name
daidrisov@dk5n52 ~/work/arch-pc/lab05 $ mc

daidrisov@dk5n52 ~/work/arch-pc/lab05 $ ls
in_out.asm lab5-1 lab5-1.o lab5-2.asm
daidrisov@dk5n52 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
daidrisov@dk5n52 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
daidrisov@dk5n52 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку: Jafar Idrisov
daidrisov@dk5n52 ~/work/arch-pc/lab05 $

```

Запуск программы

5 Выполнение заданий

№1 Я скопирую файл “lab5-1.asm”, используя клавишу F5, чтобы создать его дубликат. (рис. ??).

.и	Имя	Размер	Время правки	.и
/..		-ВВЕРХ-	ноя 9 11:05	/..
	in_out.asm	3942	ноя 9 11:04	in_out
*	lab05-1-1.asm	8744	ноя 9 11:21	*lab05-
*	lab5-1	8744	ноя 9 11:21	*lab5-1
	lab5-1.o	784	ноя 9 11:21	lab5-1
*	lab5-2	9092	ноя 9 11:54	*lab5-2
	lab5-2.asm	437	ноя 9 11:53	lab5-2
	lab5-2.o	1312	ноя 9 11:53	lab5-2

Создаю копию файла

Я внесу изменения в код, используя клавишу F4 для редактирования. (рис. ??).

```

lab05-1-1.asm [----] 50 L: [ 8+ 7 15/ 26] *(925 /1550b) 0010 0x00
_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

```

Изменения в коде

№2 Выполняем компоновку и запускаем программу. (рис. ??).

```

daidrisov@dk5n52 ~/work/arch-pc/lab05 $ nasm -f elf lab05-1-1.asm
daidrisov@dk5n52 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab05-1-1 lab05-1-1.o
daidrisov@dk5n52 ~/work/arch-pc/lab05 $ ./lab05-1-1
Введите строку:
Jafer Idrisov
Jafer Idrisov
daidrisov@dk5n52 ~/work/arch-pc/lab05 $

```

Компоновка и запуск

№3 Создам копию файла lab5-2.asm (использую клавишу F5). (рис. ??).

.и	Имя	Размер	Время правки
	in_out.asm	3942	ноя 9 11:04
*	lab05-1-1	8748	ноя 9 12:02
	lab05-1-1.asm	1550	ноя 9 12:00
	lab05-1-1.o	784	ноя 9 12:02
	lab05-2-2.asm	437	ноя 9 11:53
*	lab5-1	8744	ноя 9 11:21
	lab5-1.o	784	ноя 9 11:21
*	lab5-2	9092	ноя 9 11:54
	lab5-2.asm	437	ноя 9 11:53
	lab5-2.o	1312	ноя 9 11:53

lab05-2-2.asm 2048G/2048G (100%)

Совет: Вы сможете видеть скрытые файлы .*, установив

```

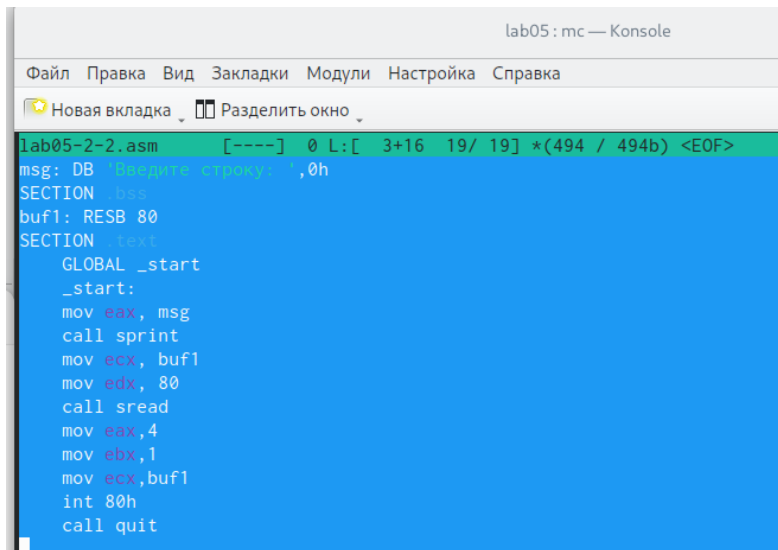
daidrisov@dk5n52 ~/work/arch-pc/lab05 $

```

1 Помощь 2 Меню 3 Просмотр 4 Правка 5 Копия

Создадим копию

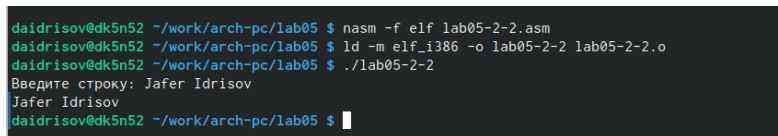
Я исправлю код программы и интегрирую в него внешний файл “in_out.asm” для его работы.(рис. ??).



```
lab05-2-2.asm  [----]  0 L: [ 3+16 19/ 19] *(494 / 494b) <EOF>
msg: DB "Введите строку: ",0h
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, buf1
mov edx, 80
call sread
mov eax, 4
mov ebx, 1
mov ecx, buf1
int 80h
call quit
```

Изменение кода

№4 Выполню компоновку и запускаю программу (рис. ??).



```
daidrisov@dk5n52 ~/work/arch-pc/lab05 $ nasm -f elf lab05-2-2.asm
daidrisov@dk5n52 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab05-2-2 lab05-2-2.o
daidrisov@dk5n52 ~/work/arch-pc/lab05 $ ./lab05-2-2
Введите строку: Jafer Idrisov
Jafer Idrisov
daidrisov@dk5n52 ~/work/arch-pc/lab05 $
```

Компоновка и запуск программы

6 Выводы

В результате выполнения этой лабораторной работы, я получил практические навыки, которые наверняка будут полезными при работе с различными языками программирования.

Список литературы

Лабораторная работа №5. Основы работы с Midnight Commander (mc). Структура программы на языке ассемблера NASM. Системные вызовы в ОС GNU Linux