

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ

### ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

дисциплина: Архитектура компьютеров и операционные системы

Студент: Ян Роман Алексеевич

Группа: НПИбд-02-23

# Содержание

## 1 Цель работы

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

## 2 Выполнение лабораторной работы

### 2.1 Ознакомление с Midnight Commander

Откроем *Midnight Commander*

```
rayan@rayan-VirtualBox:~$ mc
```

Рис 2.1.1: Демонстрация ввода команды mc

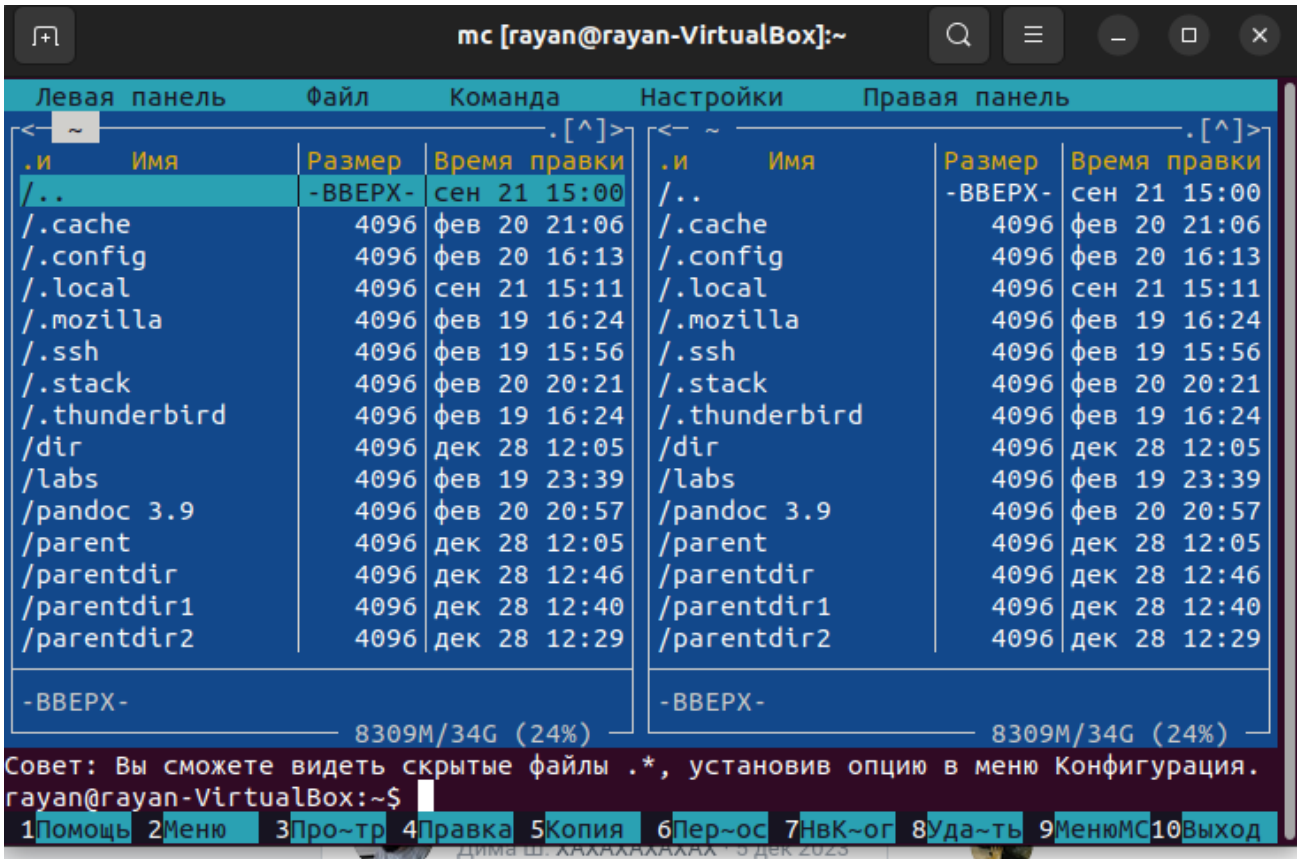


Рис 2.1.2: Демонстрация mc

Перейдем в каталог ~/work/arch-рс созданный при выполнении лабораторной работы №4

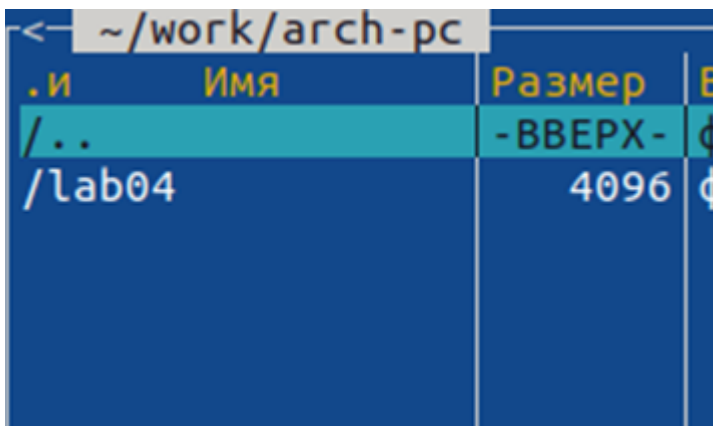


Рис 2.1.3: Переход в каталог

Создадим папку *lab05* с помощью функциональной клавиши **F7** и перейдем в этот каталог

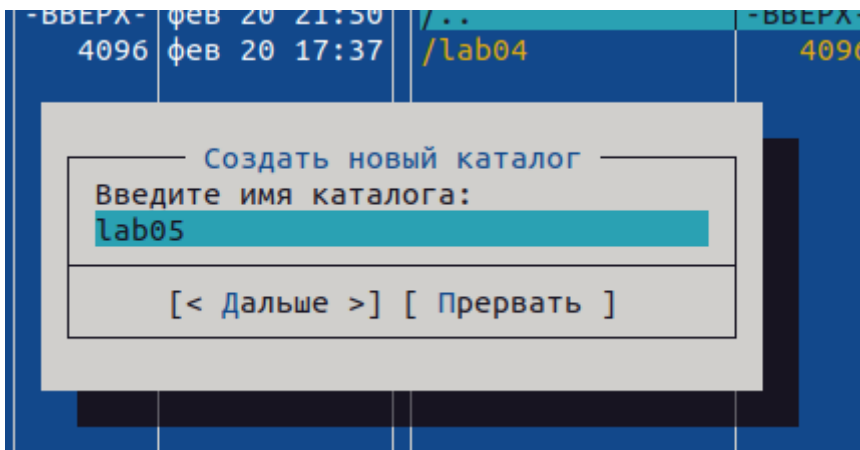


Рис 2.1.4: Создание папки *lab05*

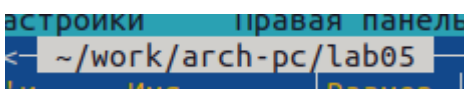


Рис 2.1.5: Демонстрация перехода в каталог *lab05*

Пользуясь строкой ввода и командой *touch* создадим файл *lab5-1.asm*

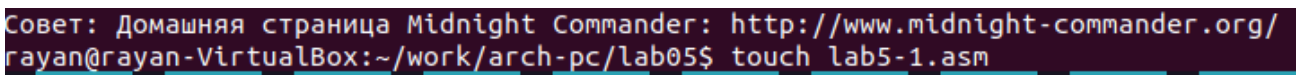


Рис 2.1.6: Демонстрация ввода команды *touch*



Рис 2.1.7: Создание файла *.asm*

С помощью функциональной клавиши **F4** откроем файл *lab5-1.asm* для редактирования во встроенном редакторе *mcedit*.

Введем текст программы из листинга 5.1 (взятый из лаб.№5), сохраним изменения и закроем файл

```
SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
                                ; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициированных данных
buf1: RESB 80 ; Буфер размером 80 байт

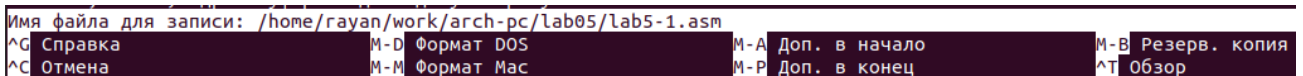
;----- Текст программы -----

SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу

;----- 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 ; Дескриптор 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 2.1.8: Демонстрация текста в файле



Имя файла для записи: /home/rayan/work/arch-pc/lab05/lab5-1.asm

⌘G Справка	⌘-D Формат DOS	⌘-A Доп. в начало	⌘-B Резерв. копия
⌘C Отмена	⌘-M Формат Mac	⌘-P Доп. в конец	⌘T Обзор

Рис 2.1.9: Сохранение

С помощью функциональной клавиши **F3** откроем файл *lab5-1.asm* для просмотра. Убедимся, что файл содержит текст программы.

```

/home/rajan/work/arch-pc/lab05/lab5-1.asm
;-----
; 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

```

Рис 2.1.10: Проверка содержимого текста в файле

Оттранслируем текст программы lab5-1.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 введем мое ФИО 'Ян Роман Алексеевич'

```

rajan@rajan-VirtualBox:~$ cd ~/work/arch-pc/lab05
rajan@rajan-VirtualBox:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
rajan@rajan-VirtualBox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o
rajan@rajan-VirtualBox:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Ян Роман Алексеевич
rajan@rajan-VirtualBox:~/work/arch-pc/lab05$

```

Рис 2.1.11: Демонстрация ввода команд для оттрансляции текста

## 2.2 Подключение внешнего файла in\_out.asm

Для упрощения написания программ часто встречающиеся одинаковые участки кода (такие как, например, вывод строки на экран или выход из программы) можно оформить в виде подпрограмм и сохранить в отдельные файлы, а во всех нужных местах поставить вызов нужной подпрограммы. Это позволяет сделать основную программу более удобной для написания и чтения. NASM позволяет подключать

внешние файлы с помощью директивы `%include`, которая предписывает ассемблеру заменить эту директиву содержимым файла. Подключаемые файлы также написаны на языке ассемблера. Важно отметить, что директива `%include` в тексте программы должна стоять раньше, чем встречаются вызовы подпрограмм из подключаемого файла. Для вызова подпрограммы из внешнего файла используется инструкция `call`, которая имеет следующий вид

`call`

где `function` имя подпрограммы.

Для выполнения лабораторных работ используется файл `in_out.asm1`, который содержит следующие подпрограммы [4]:

- `slen` – вычисление длины строки (используется в подпрограммах печати сообщения для определения количества выводимых байтов);
- `sprint` – вывод сообщения на экран, перед вызовом `sprint` в регистр `eax` необходимо записать выводимое сообщение (`mov eax,;`);
- `sprintLF` – работает аналогично `sprint`, но при выводе на экран добавляет к сообщению символ перевода строки;
- `sread` – ввод сообщения с клавиатуры, перед вызовом `sread` в регистр `eax` необходимо записать адрес переменной в которую введенное сообщение буд записано (`moveax,;`), в регистр `ebx` – длину вводимой строки (`mov ebx,;`);
- `iprint` – вывод на экран чисел в формате ASCII, перед вызовом `iprint` в регистр `eax` необходимо записать выводимое число (`mov eax,;`);
- `iprintLF` – работает аналогично `iprint`, но при выводе на экран после числа добавляет к символ перевода строки;
- `atoi` – функция преобразует `ascii`-код символа в целое число и записывает результат в регистр `eax`, перед вызовом `atoi` в регистр `eax` необходимо записать число (`moveax,;`);
- `quit` – завершение программы

Скачаем файл `in_out.asm` со страницы курса в ТУИС.

Подключаемый файл `in_out.asm` должен лежать в том же каталоге, что и файл с программой, в которой он используется. В одной из панелей `ms` откроем каталог с файлом `lab5-1.asm`. В другой панели каталог со скаченным файлом `in_out.asm`. Скопируем файл `in_out.asm` в каталог с файлом `lab5-1.asm` с помощью функциональной клавиши **F5**.

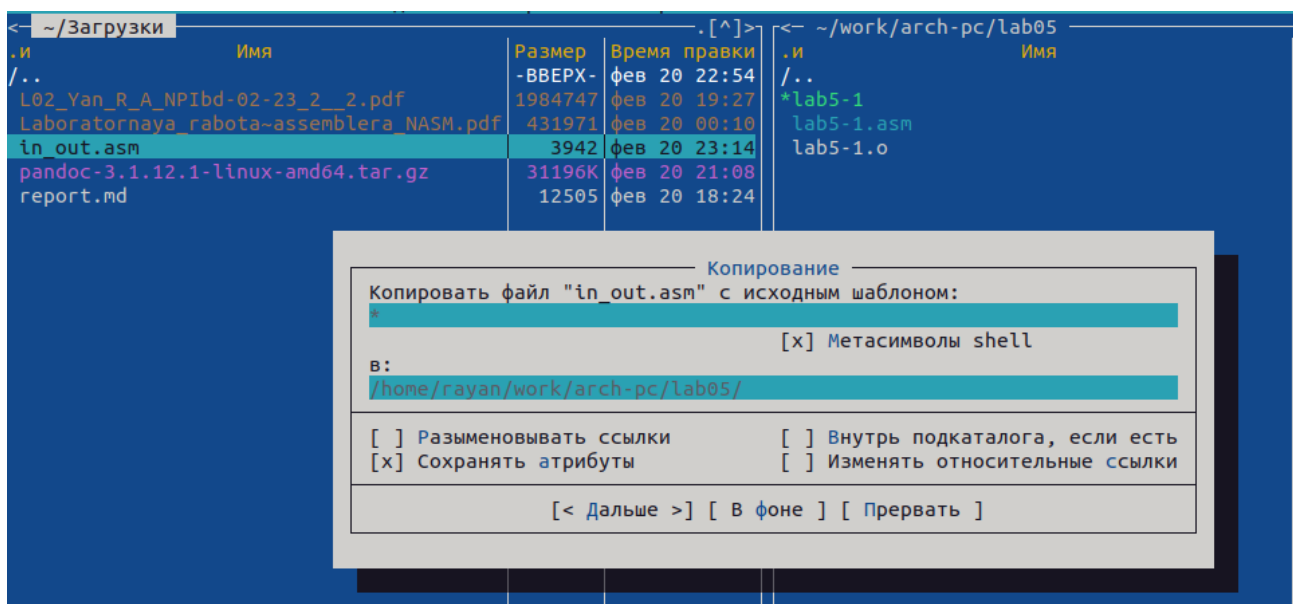


Рис 2.2.1: Копирование скаченного файла в каталог `lab05`

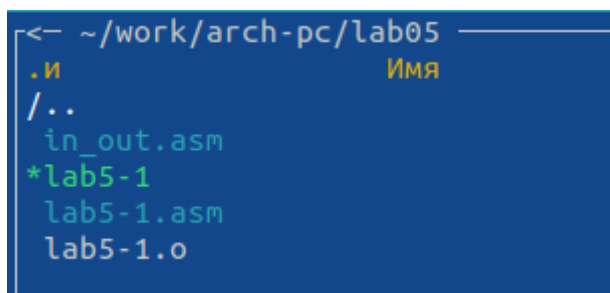


Рис 2.2.2: Демонстрация каталога после копирования

С помощью функциональной клавиши **F6** создадим копию файла `lab5-1.asm` с именем `lab5-2.asm`. Выделим файл `lab5-1.asm`, нажмем клавишу **F6**, введем имя файла `lab5-2.asm` и нажмем клавишу **Enter**.

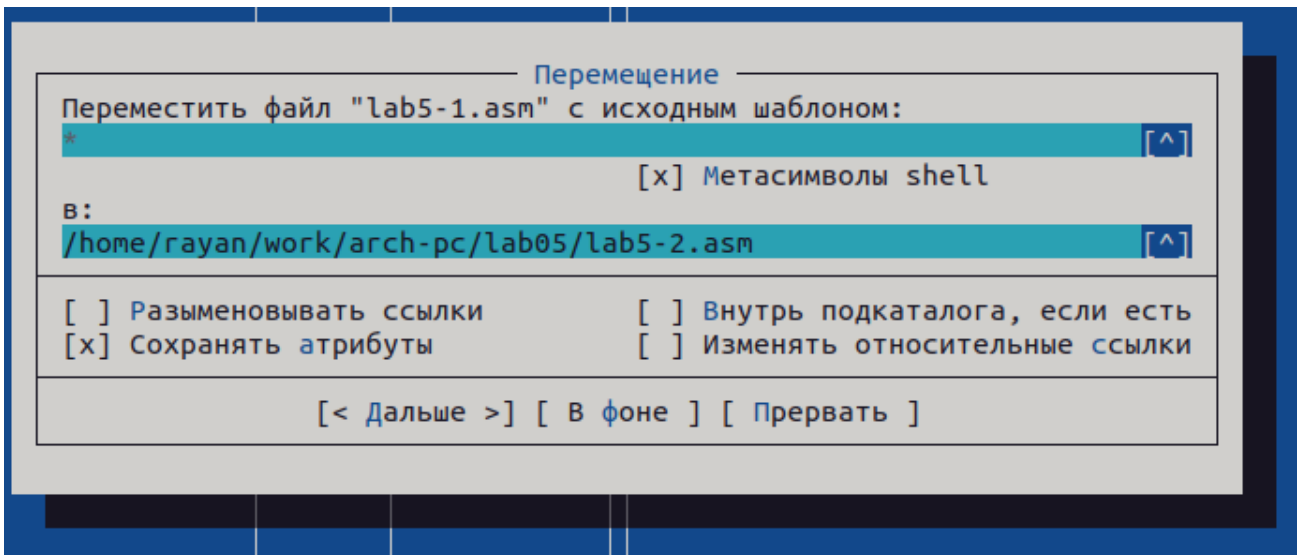


Рис 2.2.3: Создание копии файла с новым именем

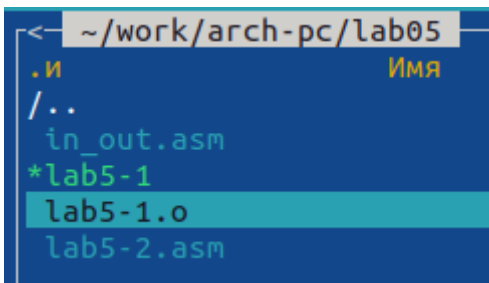


Рис 2.2.4: Демонстрация каталога после создания копии

Исправим текст программы в файле *lab5-2.asm* с использованием подпрограмм из внешнего файла *in\_out.asm* (используем подпрограммы *sprintf*, *sread* и *quit*) в соответствии с листингом 5.2. Создадим исполняемый файл и проверим его работу

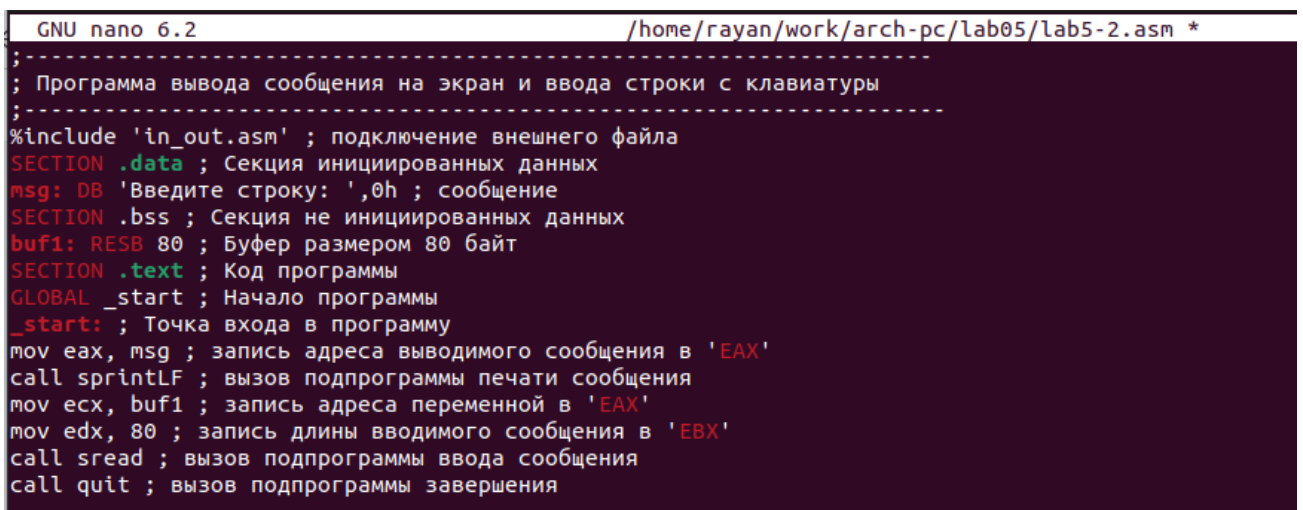


Рис 2.2.5: Демонстрация текста в файле



```

rayan@rayan-VirtualBox:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
rayan@rayan-VirtualBox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o

```

Рис 2.2.6: Демонстрация ввода команд для оттрансляции текста

```

rayan@rayan-VirtualBox:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Ян Роман Алексеевич
rayan@rayan-VirtualBox:~/work/arch-pc/lab05$

```

Рис 2.2.7: Проверка работы файлы

В файле *lab5-2.asm* заменим подпрограмму *sprintLF* на *sprint*. Создадим исполняемый файл и проверим его работу.

```

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 2.2.8: Демонстрация измененного текста в файле

```

rayan@rayan-VirtualBox:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
rayan@rayan-VirtualBox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
rayan@rayan-VirtualBox:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Ян Роман Алексеевич
rayan@rayan-VirtualBox:~/work/arch-pc/lab05$

```

Рис 2.2.9: Демонстрация повторного ввода команд для оттрансляции текста и проверка работы файла

В чем разница?

В случае *sprintLF* мы вводим сообщение в след строке, в случае *sprint* ввод сообщения происходит в той же строке, где нас просят ввести сообщение после :

### 3 Самостоятельная работа

**Задание №1** Создайте копию файла *lab5-1.asm*. Внесите изменения в программу (без использования внешнего файла *in\_out.asm*), так чтобы она работала по следующему алгоритму:

- вывести приглашение типа “Введите строку:”
- ввести строку с клавиатуры
- вывести введенную строку на экран

Для удобства создадим новую папку в каталоге *lab05* для самостоятельной работы

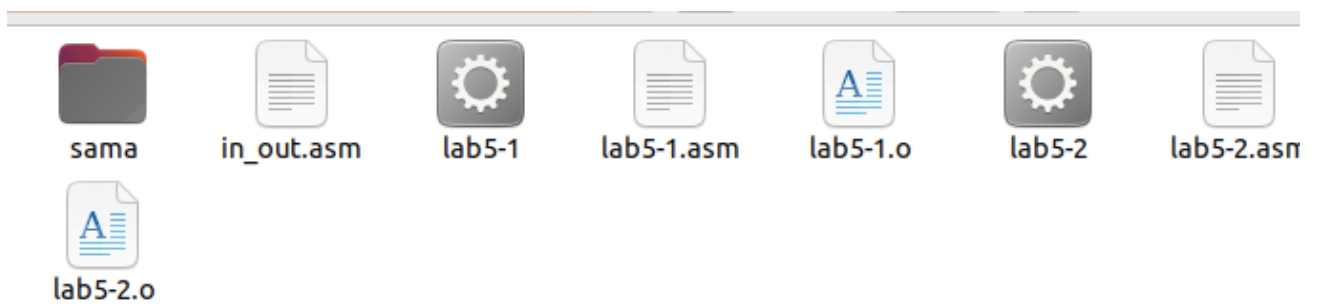


Рис 3.1.1: Демонстрация созданной папки

Создадим копию файла *lab5-1.asm* в *тс* с помощью **F5**

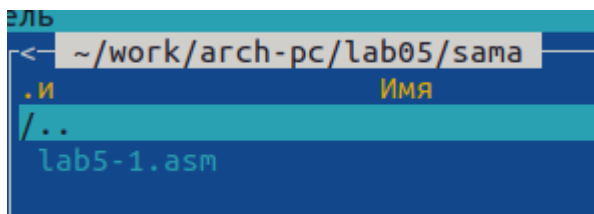


Рис 3.1.2: Создание копии файла

Изменим программу в *тс* под условие задания

```

SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
;----- Системный вызов write
; После вызова инструкции 'int 80h' на экр
; выведено сообщение из переменной 'msg'
mov eax,4 ; Системный вызов для записи (s
mov ebx,1 ; Описатель файла 1 - стандартн
mov ecx,msg ; Адрес строки 'msg' в 'ecx'
mov edx,msgLen ; Размер строки 'msg' в 'e
int 80h ; Вызов ядра
;----- системный вызов read -----
; После вызова инструкции 'int 80h' прогр
; строки, которая будет записана в перемен
mov eax, 3 ; Системный вызов для чтения (
mov ebx, 0 ; Дескриптор файла 0 - стандар
mov ecx, buf1 ; Адрес буфера под вводимую
mov edx, 80 ; Длина вводимой строки
int 80h ; Вызов ядра
;----- Системный вызов exit -----
; После вызова инструкции 'int 80h' прогр
mov eax,4 ; Системный вызов для выхода (s
mov ebx,1
mov ecx,buf1
mov edx,80 ; Выход с кодом возврата 0 (бе
int 80h ; Вызов ядра

```

Рис 3.1.3: Редактирование программы(кода)

**Задание №2 Получите исполняемый файл и проверьте его работу. На приглашение ввести строку введите свою фамилию.**

Скомпилируем и отправим файл на обработку компоновщику

```

ayan@ayan-VirtualBox:~/work/arch-pc/lab05$ cd ~/work/arch-pc/lab05/sama
ayan@ayan-VirtualBox:~/work/arch-pc/lab05/sama$ nasm -f elf lab5-1.asm
ayan@ayan-VirtualBox:~/work/arch-pc/lab05/sama$ ld -m elf_i386 -o lab5-1 lab5-1.o
ayan@ayan-VirtualBox:~/work/arch-pc/lab05/sama$

```

Рис 3.2.1: Компиляция и обработка файла

Проверим работоспособность файла(программы)

```

ayan@ayan-VirtualBox:~/work/arch-pc/lab05/sama$ ./lab5-1
Введите строку:
Ян
Ян

```

Рис 3.2.2: Проверка программы

**Задание№3** Создайте копию файла *lab5-2.asm*. Исправьте текст программы с использование подпрограмм из внешнего файла *in\_out.asm*, так чтобы она работала по следующему алгоритму:

- вывести приглашение типа “Введите строку:”
- ввести строку с клавиатуры
- вывести введенную строку на экран

Скопируем файлы *int\_out.asm* и *lab5-2.asm* в отдельную папку

./..	-ВВЕРХ-	фев 21 05:36	./..
in_out.asm	3942	фев 20 23:14	/sama
*lab5-1	8744	фев 21 05:49	in_out.asm
lab5-1.asm	2352	фев 21 05:46	*lab5-1
lab5-1.o	784	фев 21 05:49	lab5-1.asm
sama	3942	фев 20 23:14	lab5-1.o
			*lab5-2

Рис 3.3.1: Копирование *int\_out.asm*

lab5-1.o	784	фев 21 05:49	lab5-1.asm
lab5-2.asm	1224	фев 21 05:27	lab5-1.o
sama	3942	фев 20 23:14	*lab5-2
			lab5-2.asm
			lab5-2.o

Рис 3.3.2: Копирование *lab5-2.asm*

Изменим программу в тс под условие задания

```
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, buf1
call sprint
call quit ; вызов подпрограммы завершения
```

Рис 3.3.3: Редактирование программы(кода)

**Задание№4** Создайте исполняемый файл и проверьте его работу.

Скомпилируем и отправим файл на обработку

```
rayan@rayan-VirtualBox:~/work/arch-pc/lab05/sama$ nasm -f elf lab5-2.asm
rayan@rayan-VirtualBox:~/work/arch-pc/lab05/sama$ ld -m elf_i386 -o lab5-2 lab5-2.o
```

Рис 3.4.1: Компиляция и обработка файла

Проверим работоспособность программы

```
rayan@rayan-VirtualBox:~/work/arch-pc/lab05/sama$ ./lab5-2
Введите строку: Yan
Yan
rayan@rayan-VirtualBox:~/work/arch-pc/lab05/sama$
```

*Рис 3.4.2: Проверка программы*

Загрузим все файлы на github

## **4 Выводы**

Я приобрел практические навыки работы в Midnight Commander. Освоил инструкции языка ассемблера mov и int.