Лабораторная работа №4

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

Селиванов Вячеслав Алексеевич

Содержание

# 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 размеров 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.

Открываю Midnight Commander, используя команду mc(рис. [[1](#fig:001)]).

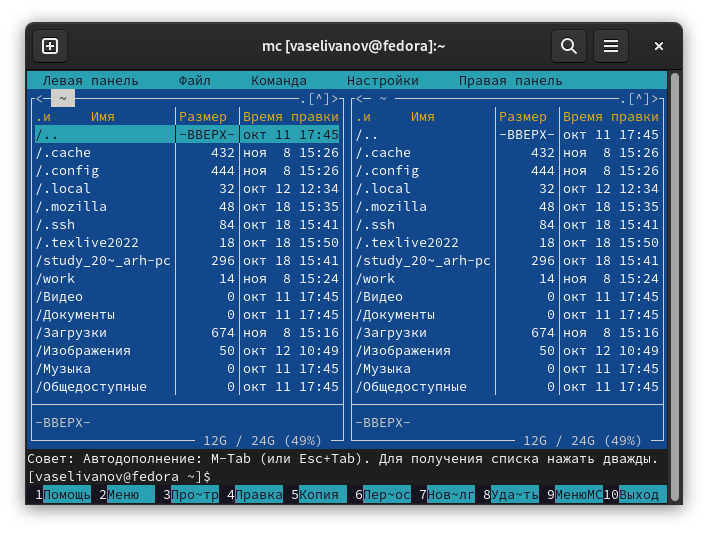


Figure 1: Открытый Midnight Commander

Перехожу в каталог work,созданный при выполнении предыдущей лабораторной работы(рис. [[2](#fig:002)]).

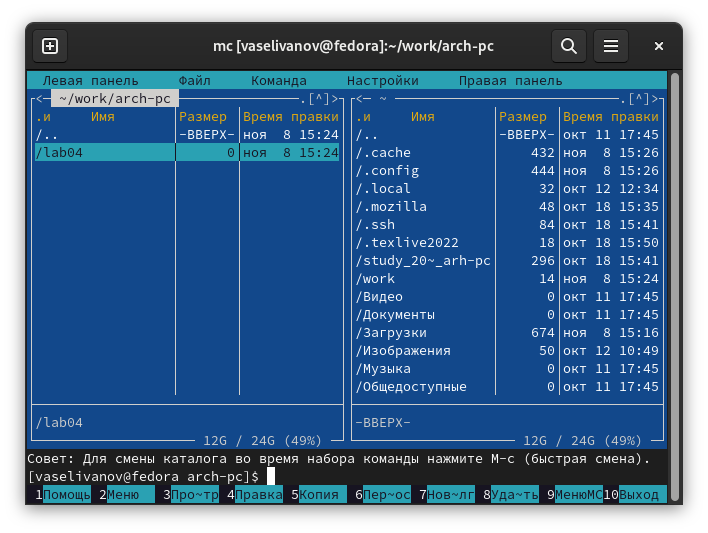


Figure 2: Перемещение в каталог

Создаю новый каталог lab05(рис. [[3](#fig:003)]).

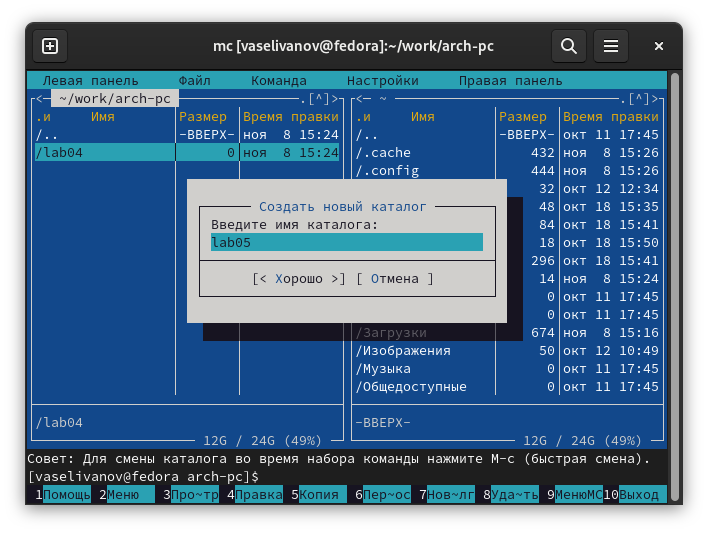


Figure 3: Создание каталога

В новом каталоге создаю файл lab5-1.asm,в котором я буду работать далее, используя команду touch(рис. [[4](#fig:004)]).

Figure 4: Создание файла

Figure 4: Создание файла

## 4.2 Структура программы на языке ассемблера NASM.

С помощью клавиши F4 открываю созданный файл в редакторе nano(рис. [[5](#fig:005)]).

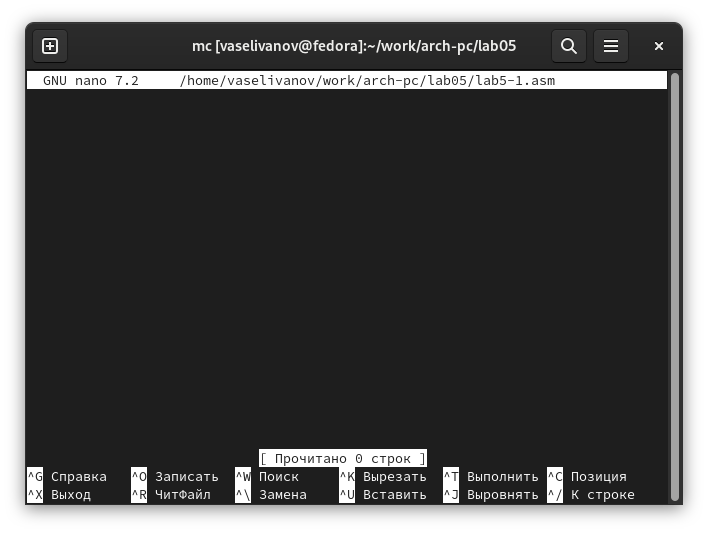


Figure 5: Файл в редакторе

Ввожу в файл код программы для запроса строки(рис. [[6](#fig:006)]). Далее выхожу из редактора,сохраняя изменения.

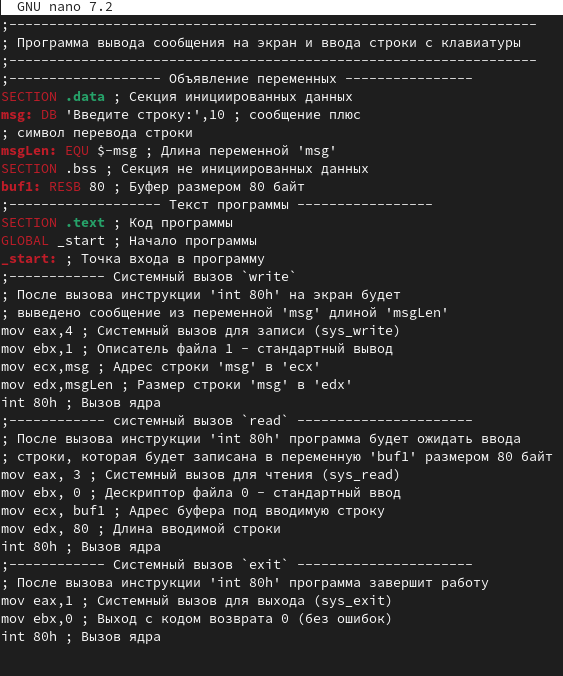


Figure 6: Редактирование файла

Используя клавишу F4, открываю файл для просмотра,чтобы проверить,сохранилась ли в нем написанная программа(рис. [[7](#fig:007)]).

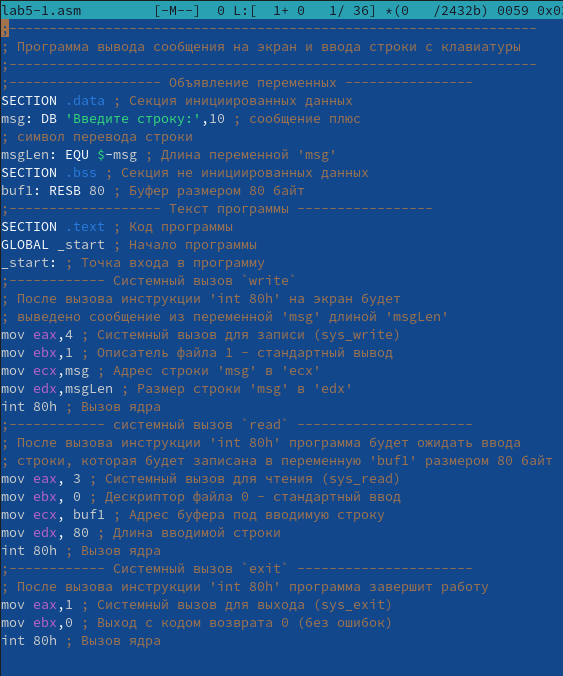


Figure 7: Открытие файла для проверки

Создаю для текста программы объектный файл. Выполняю его компоновку(рис. [[8](#fig:008)]) (рис. [[9](#fig:009)]). После чего создался исполняемый файл lab5-1.

Figure 8: Создание объектного файла

Figure 8: Создание объектного файла

Figure 9: Компоновка

Figure 9: Компоновка

Запускаю исполняемый файл. Программа выводит строку и ждет ввода с клавиатуры, после ввода своего ФИО программа завершает работу(рис. [[10](#fig:010)]).

Figure 10: Запуск программы

Figure 10: Запуск программы

## 4.3 Подключенние внешнего файла.

Скачиваю файл in\_out.asm со страницы ТУИС. Файл сохранился в ‘Загрузки’(рис. [[11](#fig:011)]).

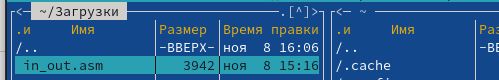


Figure 11: Файл in\_out.asm

Копирую данный файл в каталог lab05, используя клавишу F5(рис. [[12](#fig:012)]).

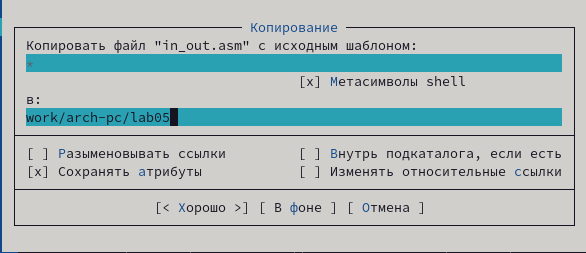


Figure 12: Копирование файла в нужную директорию

С помощью той же утилиты F5 копирую файл lab5-1.asm, но уже с другим названием(рис. [[13](#fig:013)]).

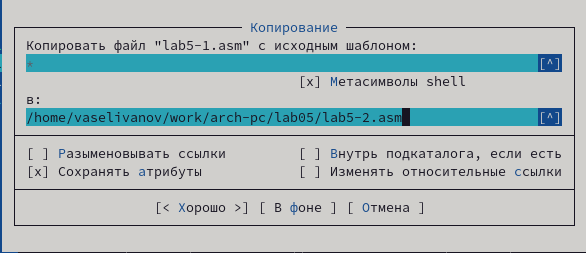


Figure 13: Копирование файла с изменением названия

Меняю содержимое файла lab5-2.asm в редакторе nano, чтобы в программе использовались подпрограммы из внешнего файла in\_out.asm(рис. [[14](#fig:014)]).

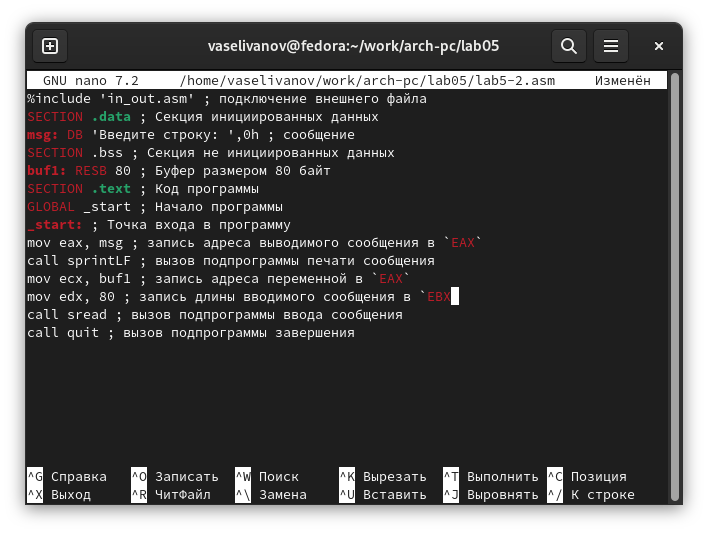


Figure 14: Редактирование файла,для использования in\_put.asm

Создаю объектный файл для lab5-2.asm(рис. [[15](#fig:015)]).

Figure 15: Создание объектного файла

Figure 15: Создание объектного файла

Компоную данный файл,после чего создается исполняемый файл. Запускаю его и проверяю,работает ли данная программа(рис. [[16](#fig:016)]).

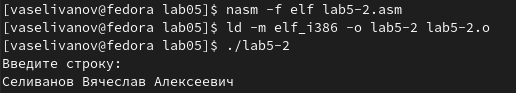


Figure 16: Компоновка файла и запуск программы

Открываю файл lab5-2.asm для редактирования в nano,используя F4. Изменяю в нем подпрограмму sprintLF на sprint, сохраняю изменения и открываю файл для проверки(рис. [[17](#fig:017)]).

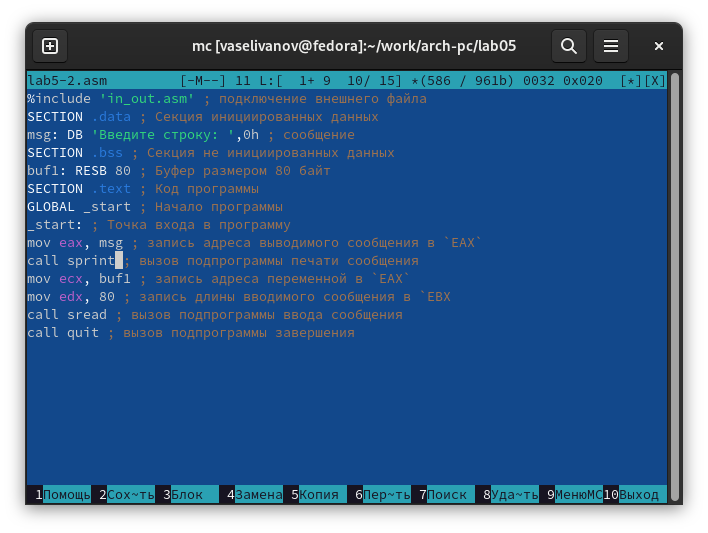


Figure 17: Редактирование файла

Выполняю компоновку объектного файла и запускаю новый исполняемый файл(рис. [[18](#fig:018)]).

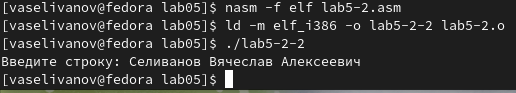


Figure 18: Исполнение файла

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

## 4.4 Выполнение заданий для самостоятельной работы.

Создаю копию файла lab5-1.asmс именем lab5-1-1.asm с помощью клавиши F5(рис. [[19](#fig:019)]).

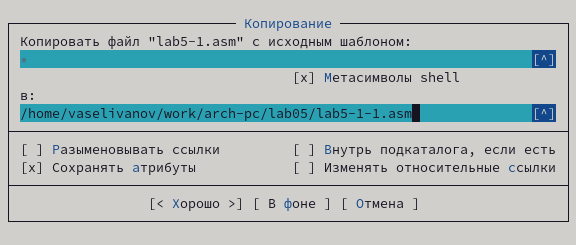


Figure 19: Копирование файла lab5-1.asm

Используя клавишу F4 открываю данный файл в nano и редактирую файл так,чтобы кроме вывода приглашения и запроса ввода, она выводила строку,которую пользователь ввел с клавиатуры(рис. [[20](#fig:020)]).

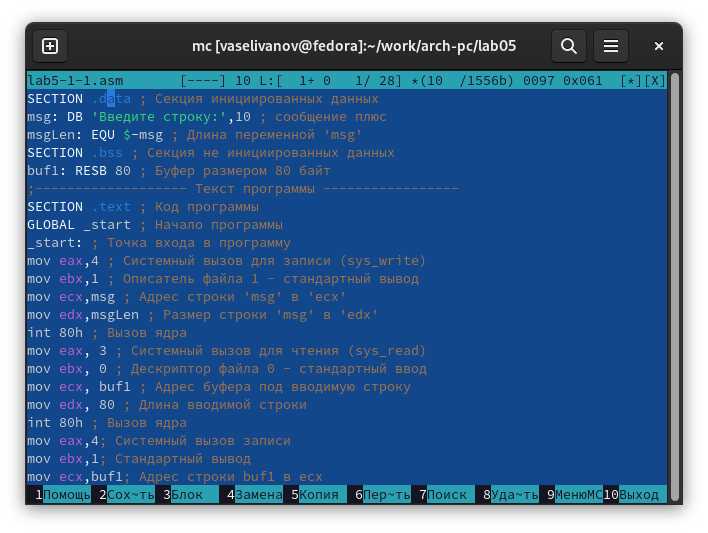


Figure 20: Редактирование программы

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 ; 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 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

Создаю объектный файл lab5-1-1.o и обрабатываю его,используя компоновщик,запускаю созданный исполняемый файл, ввожу своё имя, после этого программа выводит то,что я напечатал(рис. [[21](#fig:021)]).

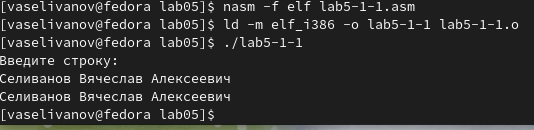


Figure 21: Исполнение файла

Копирую файл lab5-2.asm,используя F5, переименовываю его в lab5-2-3.asm(рис. [[22](#fig:022)]).

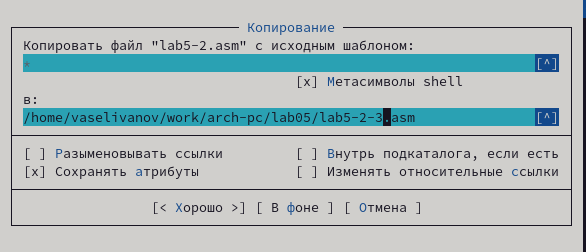


Figure 22: Копирование файла

Используя клавишу F4 открываю данный файл в nano и редактирую файл так,чтобы кроме вывода приглашения и запроса ввода, она выводила строку,которую пользователь ввел 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 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

Создаю объектный файл lab5-2-3.o и обрабатываю его,используя компоновщик,запускаю созданный исполняемый файл, ввожу своё имя, после этого программа выводит то,что я напечатал(рис. [[23](#fig:023)]).

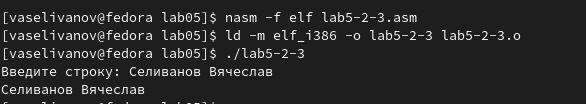


Figure 23: Исполнение файла

# 5 Выводы

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

# Список литературы

::: {#Лабораторная работа №5} :::