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

# 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. 1)

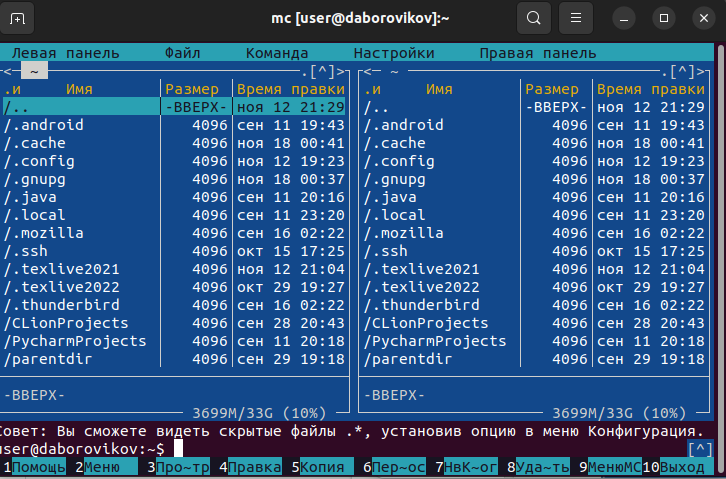


Рис. 1: Midnight Commander

Перейдем в каталог ~/work/arch-pc созданный при выполнении лабораторной работы №5(рис. 2)

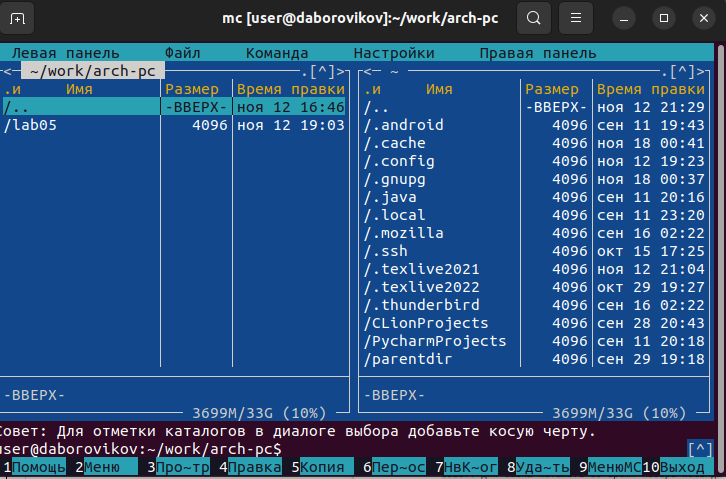


Рис. 2: Переход в каталог ~/work/arch-pc

С помощью функциональной клавиши F7 создадим папку lab06 и перейдем в созданный каталог.(рис. 3)

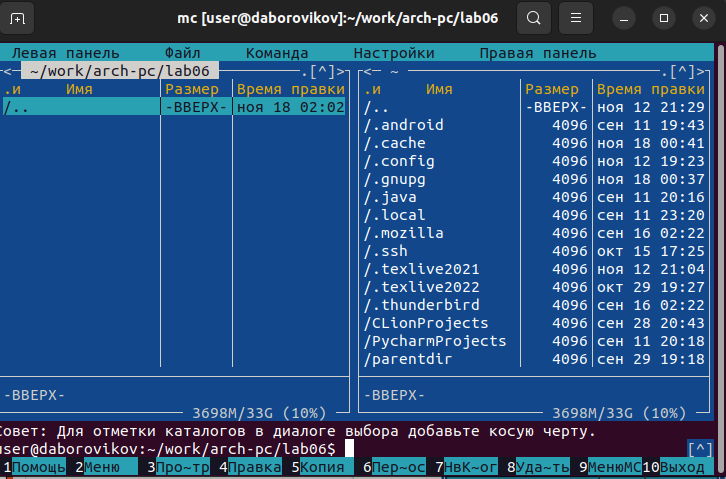


Рис. 3: Переход в каталог ~/work/arch-pc/lab06

Пользуясь строкой ввода и командой touch создадим файл lab6-1.asm(рис. 4)

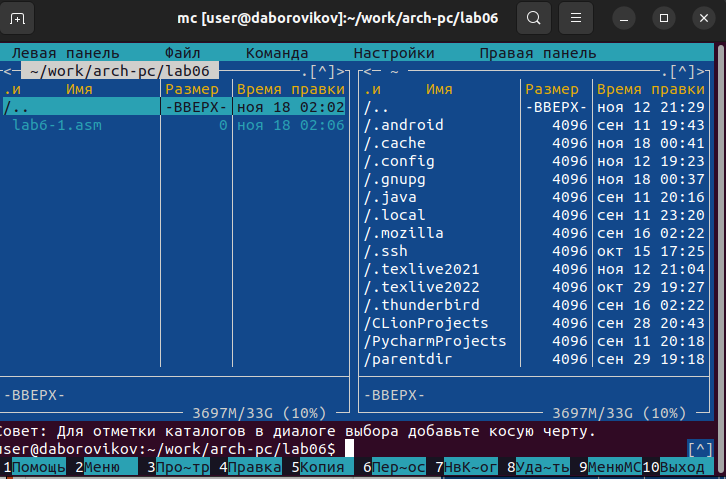


Рис. 4: Создание файла lab6-1.asm

С помощью функциональной клавиши F4 откроем файл lab6-1.asm для редактирования во встроенном редакторе. В качестве редактора выберем nano(рис. 5)



Рис. 5: lab6-1.asm в редакторе nano

Введем текст программы из листинга 6.1, созраним его и закроем файл(рис. 6)

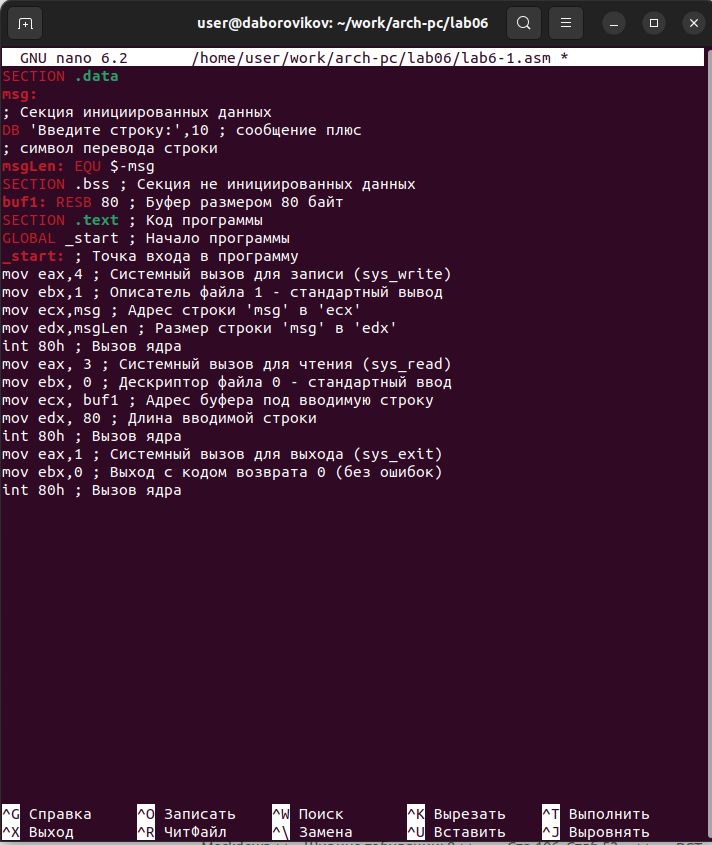


Рис. 6: Ввод текста программы с последующим сохранением

С помощью функциональной клавиши F3 откроем файл lab6-1.asm для просмотра и убедимся, что файл содержит текст программы.(рис. 7)

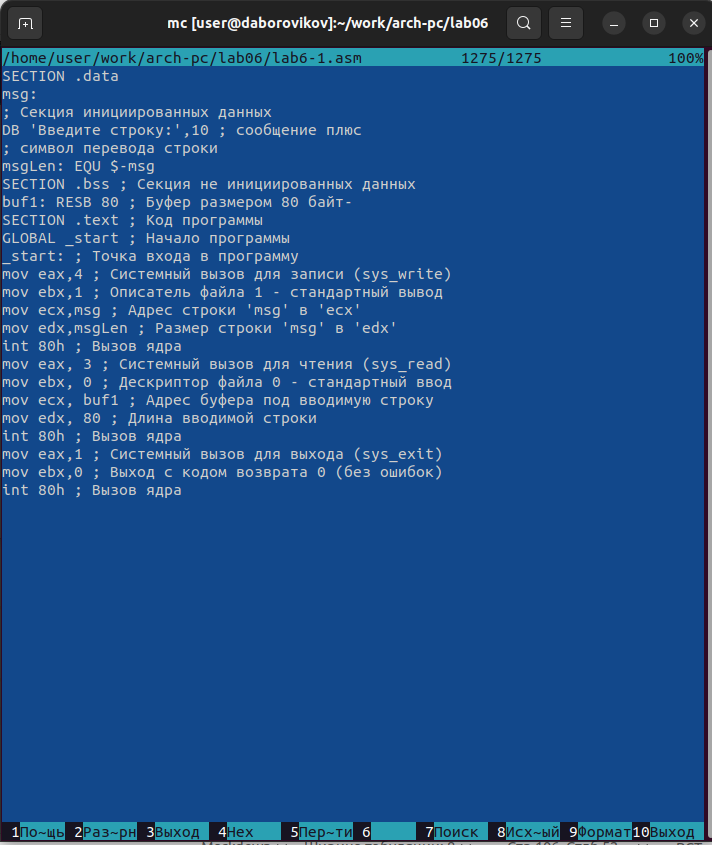


Рис. 7: Проверка сохранения файла

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. 8)

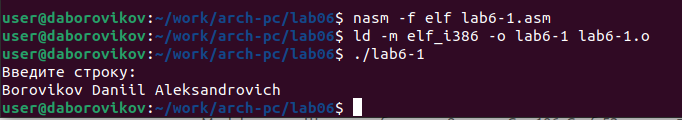


Рис. 8: Запуск программы

Скопируем файл in\_out.asm в каталог с файлом lab6-1.asm с помощью функциональной клавиши F5(рис. 9)

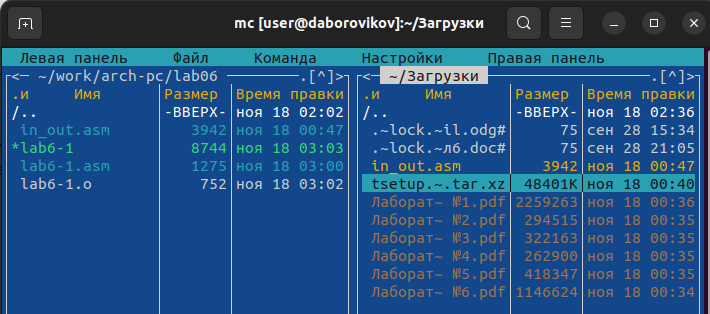


Рис. 9: копирование файла in\_out.asm

С помощью функциональной клавиши F6 создадим копию файла lab6-1.asm с именем lab6-2.asm.(рис. 10)

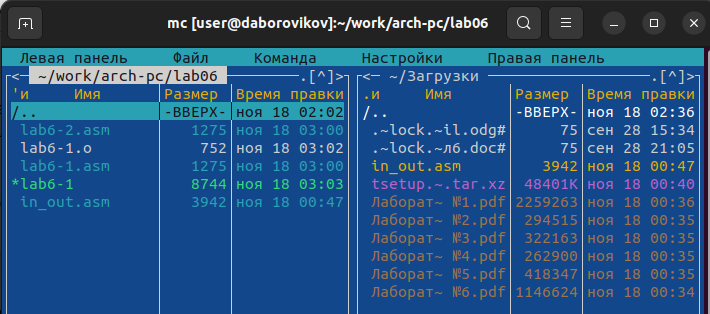


Рис. 10: Копирование файла lab6-1.asm

Исправьте текст программы в файле lab6-2.asm с использование подпрограмм из внешнего файла in\_out.asm (используйте подпрограммы sprintLF, sread и quit) в соответствии с листингом 6.2.(рис. 11)

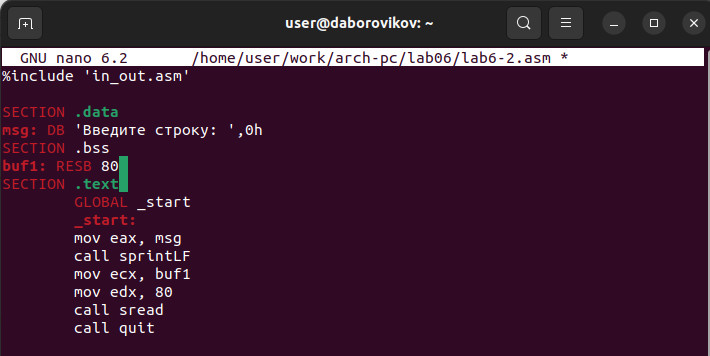


Рис. 11: Включаем функции в программу lab6-2.asm

Создадим исполняемый файл и проверим его работу(рис. 12)

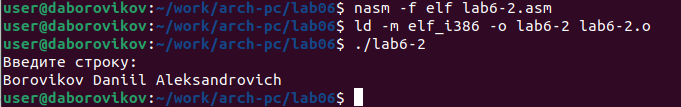


Рис. 12: Проверка работы исполняемого файла

В файле lab6-2.asm заменим подпрограмму sprintLF на sprint. Создадим исполняемый файл и проверьте его работу. Выясним, что от первоначальной версии полученная программа отличается вводом и выводом на одной строке(рис. 13)

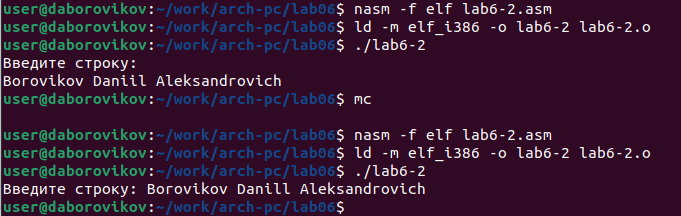


Рис. 13: Файл lab6-2.asm с вводом и выводом на одной строке

# 3 Самостоятельная работа

Программа работающая по алгоритму без использования подпрограмм:

вывести приглашение типа “Введите строку:”;

ввести строку с клавиатуры;

вывести введённую строку на экран.(рис. 14)

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

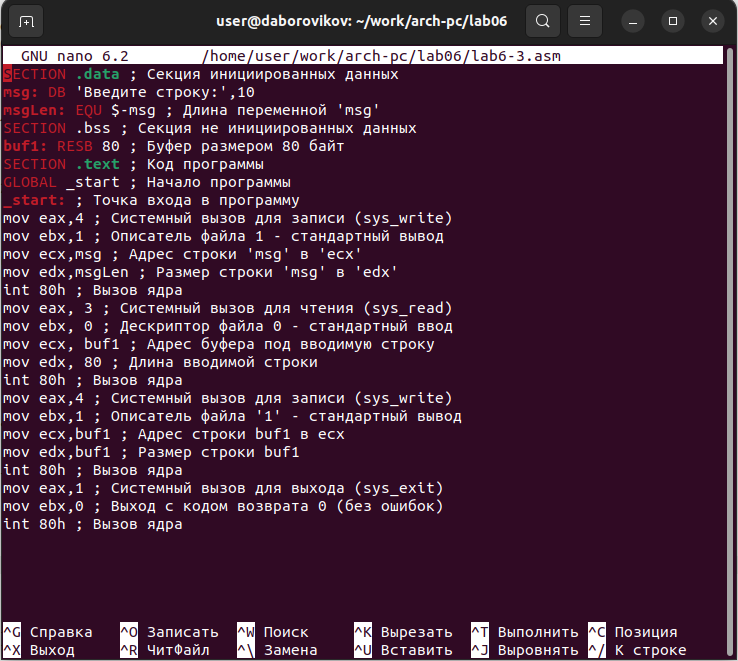


Рис. 14: Код программы без использования подпрограмм

Получим исполняемый файл требуемой программы и проверим его работу. На приглашение ввести строку введем свою фамилию(рис. 15)

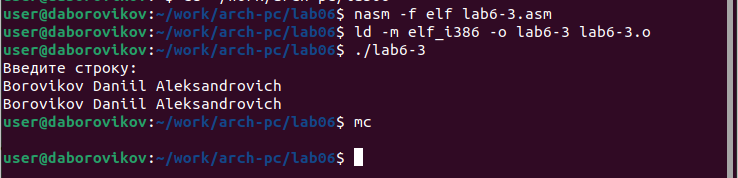


Рис. 15: Запуск программы без использрвания подпрограмм

Программа работающая по алгоритму с использованием подпрограмм:

вывести приглашение типа “Введите строку:”;

ввести строку с клавиатуры;

вывести введённую строку на экран.(рис. 16)

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

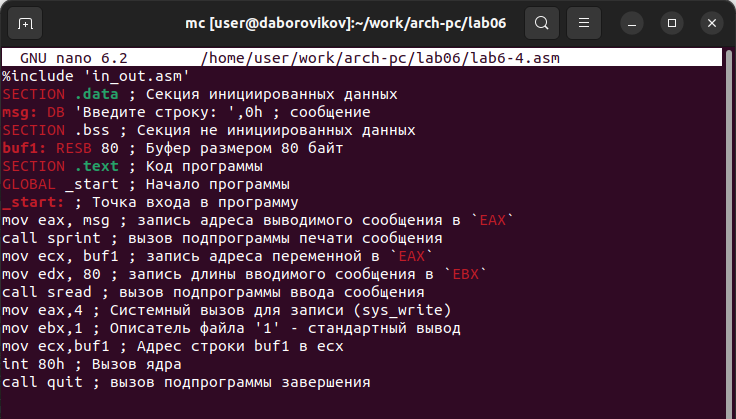


Рис. 16: Код программы с использованием подпрограмм

Получим исполняемый файл требуемой программы и проверим его работу. На приглашение ввести строку введем свою фамилию(рис. 17)

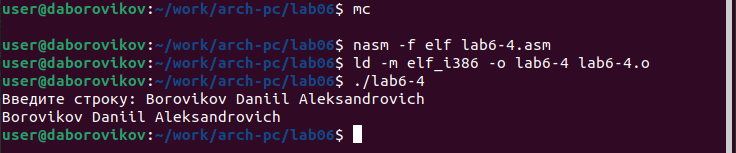


Рис. 17: Запуск программы с использованием подпрограмм

Ссылка на github: 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.