Лабораторная работа №5

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

Богату Ирина Владимировна

Содержание

Список иллюстраций

Список таблиц

Цель работы

Ознакомиться с программой Midnight commander и освоить написание программ на языке ассемблера с помощью инструкций mov и int

Выполнение лабораторной работы

Для начала выполнения лабораторной работы нам необходимо открыть Midnight commander с помощью команды mc (Рис. 2.1):

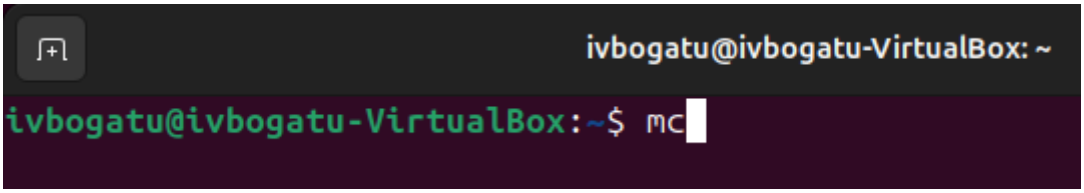

Рис. 1: Запуск Midnight commander

После ввода команды мы увидим такой интерфейс (Рис. 2.2):

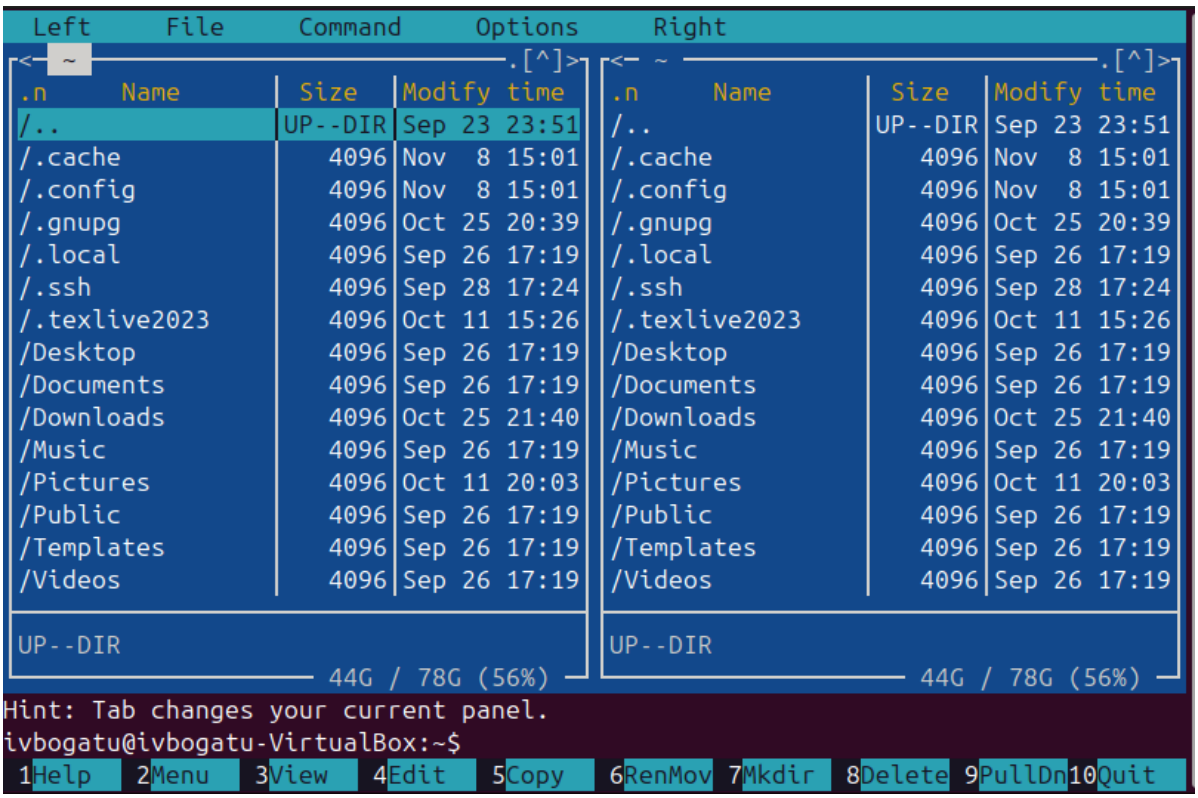

Рис. 2: Интерфейс midnight commander

С помощью стрелок и клавиши Enter перейдём в каталог ~/work/arch-pc (Рис. 2.3):

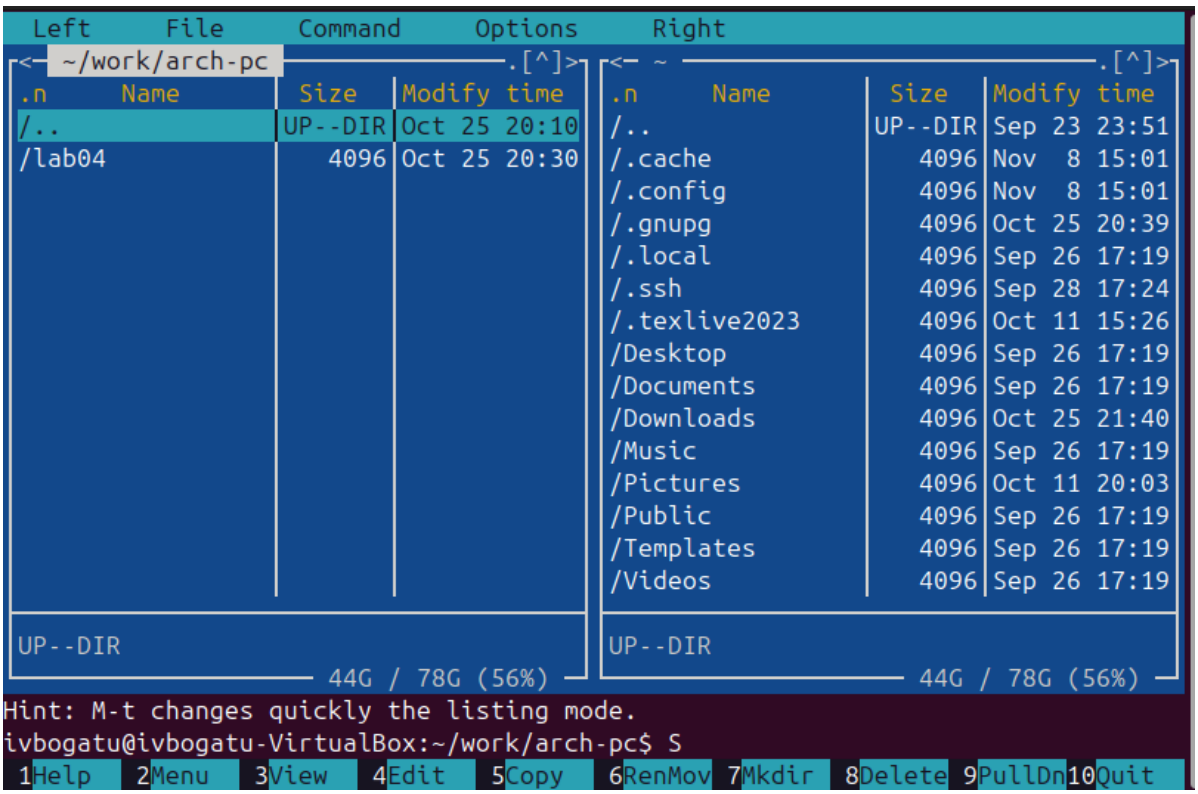

Рис. 3: Переход в нужный каталог (~/work/arch-pc)

Создадим папку lab05 с помощью клавиши F7 (Рис. 2.4):

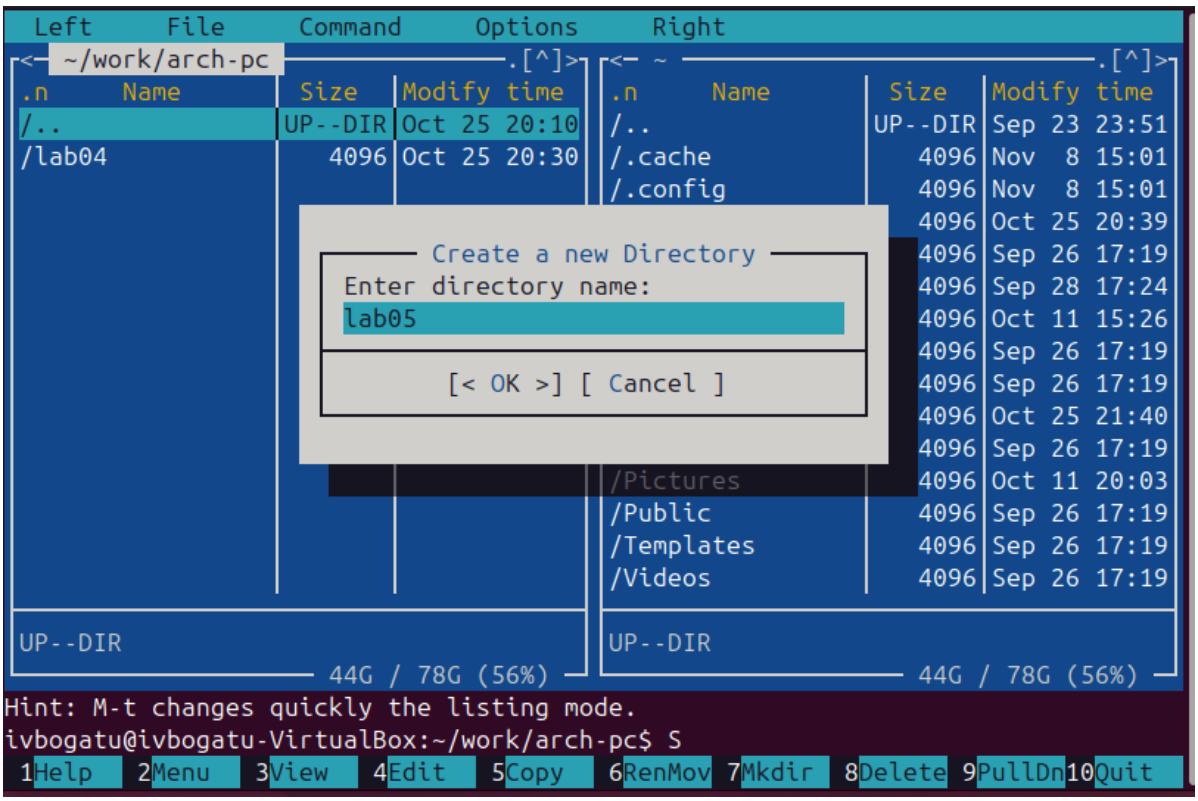

Рис. 4: Создание папки

Теперь с помощью команды touch создадим файл lab5-1.asm (Рис. 2.5):

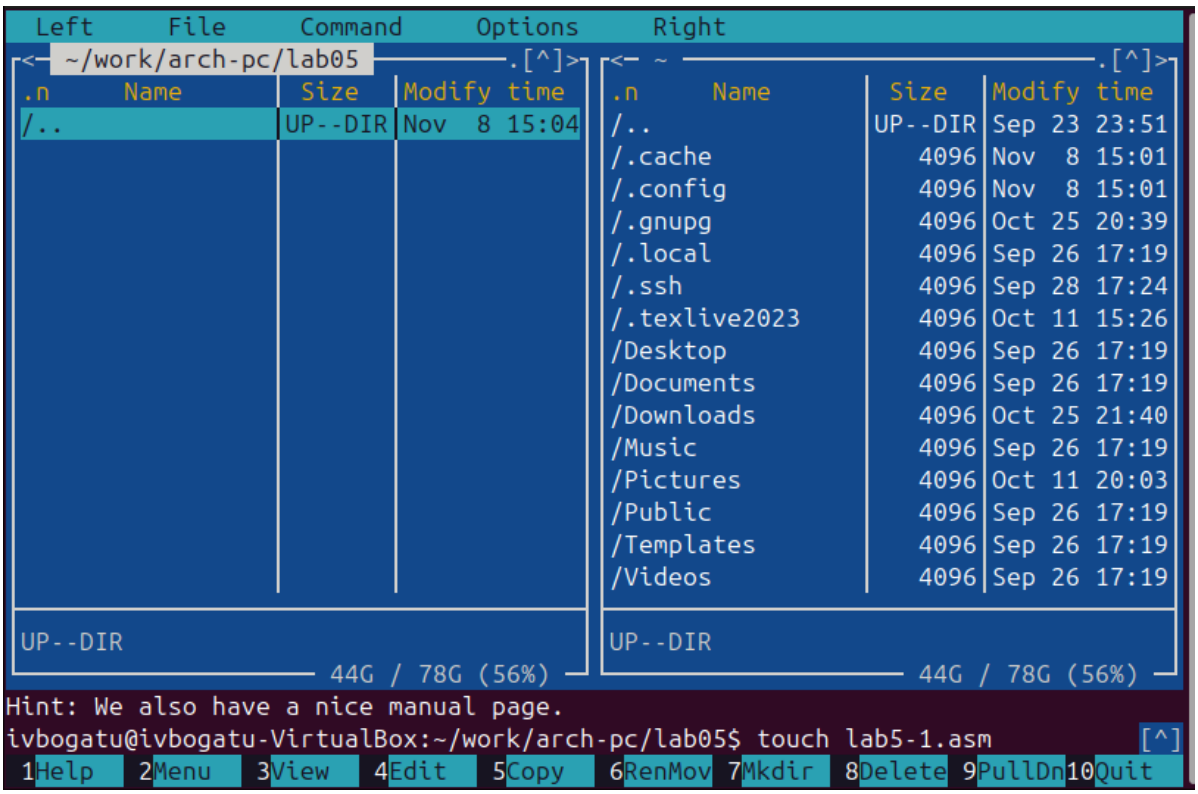

Рис. 5: Создание файла lab5-1.asm с помощью команды touch прямо в mc

Теперь с помощью клавиши F4 откроем только что созданный файл. Нам предложат выбор между несколькими редакторами. Мы выберем редактор по умолчанию, nano (Рис. 2.6):

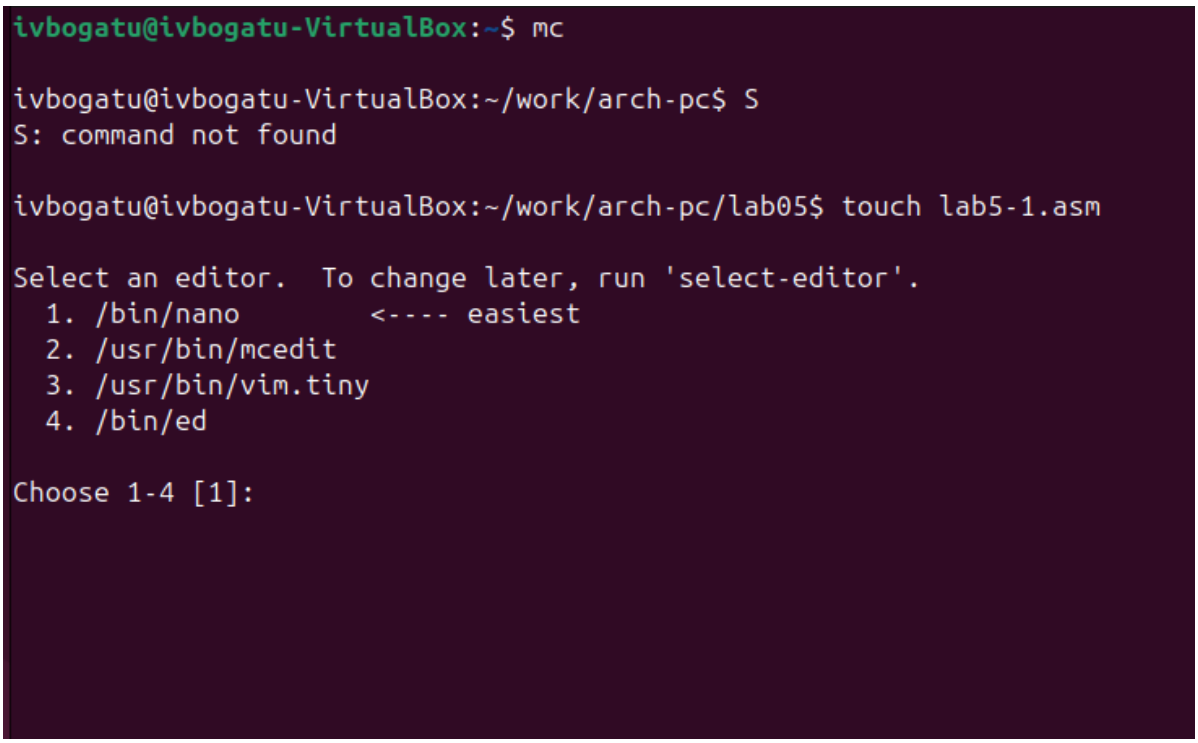

Рис. 6: Выбор текстового редактора

Теперь отредактируем файл и поместим в него следующий код (Рис. 2.7):

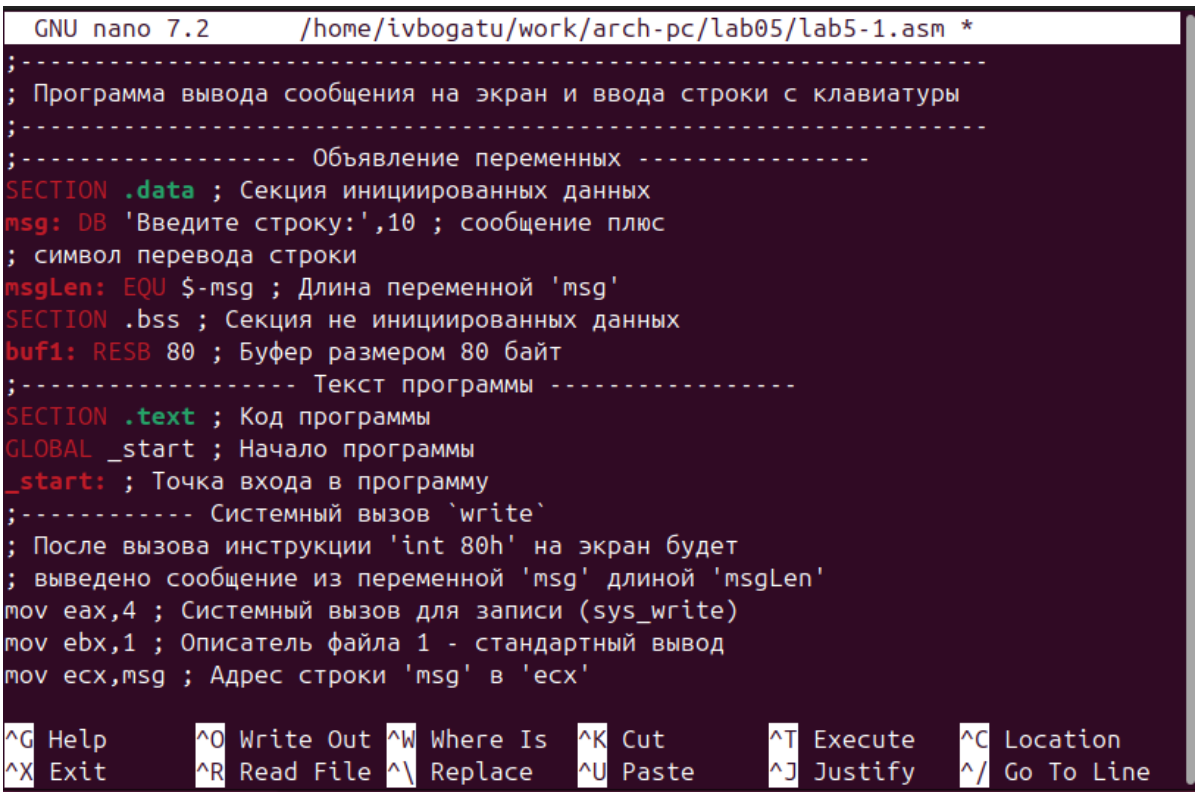

Рис. 7: Редактирование файла lab5-1.asm

Теперь сохраним его (сочетанием клавиш ctrl+x и согласившись с сохранением) и с помощью F3 откроем для просмотра, чтобы убедиться, что он сохранился корректно (Рис. 2.8):

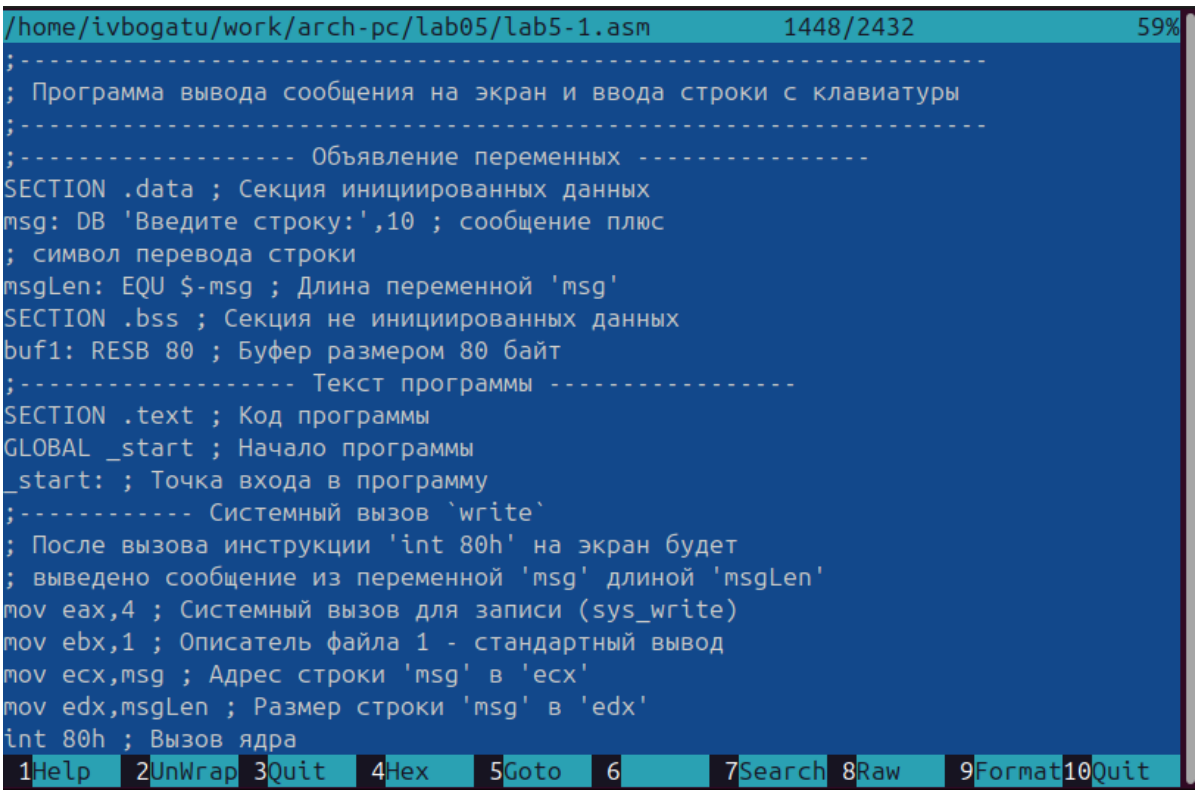

Рис. 8: Проверка успешного редактирования

Теперь скомпилируем его (Рис. 2.9):

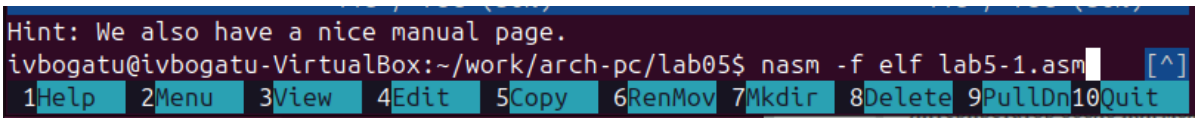

Рис. 9: Компиляция файла с помощью nasm

И соберём (Рис. 2.10):

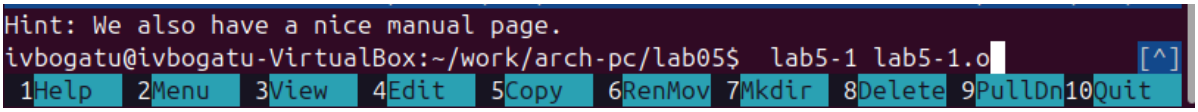

Рис. 10: Сборка исполняемого файла с помощью ld

После этого запустим получившийся исполняемый файл (Рис. 2.11):

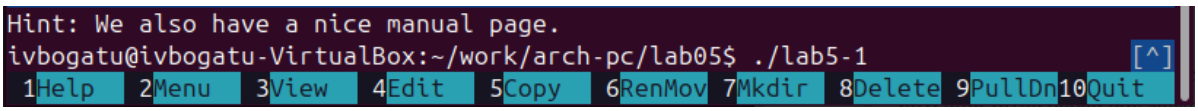

Рис. 11: Запуск исполняемого файла

Теперь введём ФИО (Рис. 2.12):

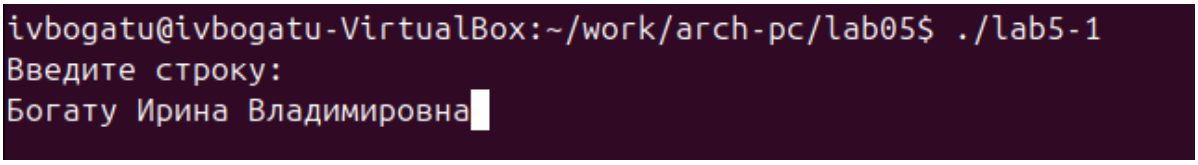

Рис. 12: Взаимодействие с программой

После нажатия Enter программа завершится и ничего не произойдёт. Теперь скачаем файл in_out.asm и откроем папку с ним в правой панели (Рис. 2.13):

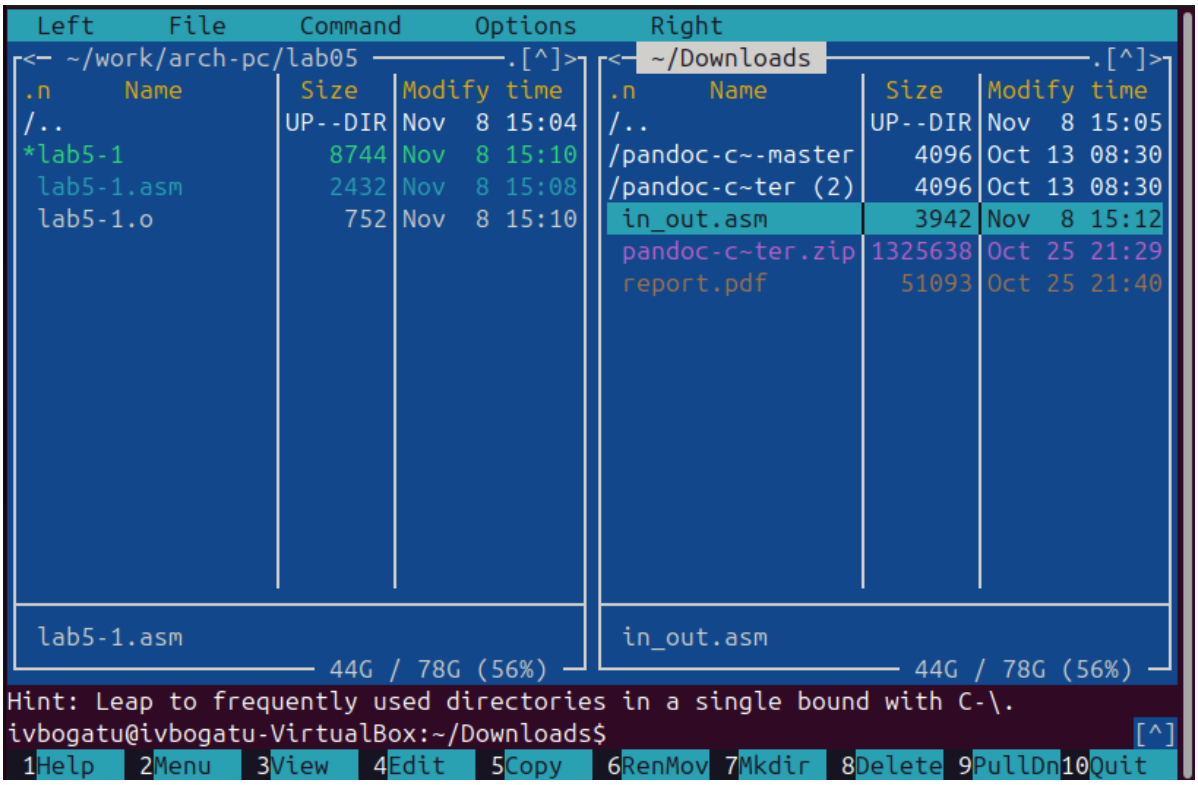

Рис. 13: Открытие папки с файлом in_out.asm в правой панели

Скопируем его в нашу рабочую папку с помощью F6 (Рис. 2.14):

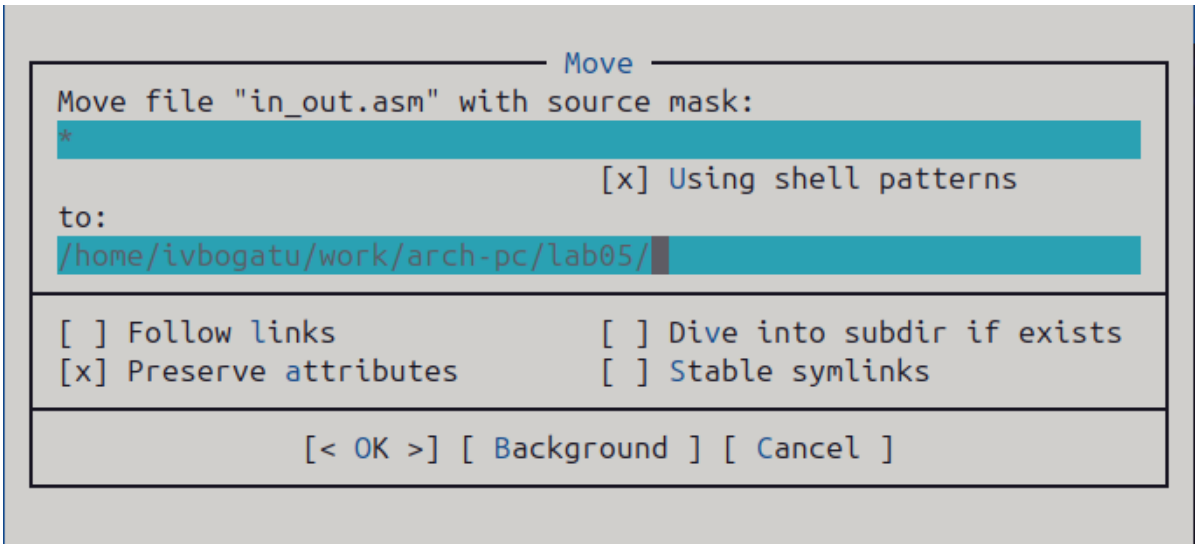

Рис. 14: Копирование файла с помощью F6

Теперь сделаем копию файла lab5-1.asm с помощью команды F5. Назовём копию lab5-2.asm (Рис. 2.15):

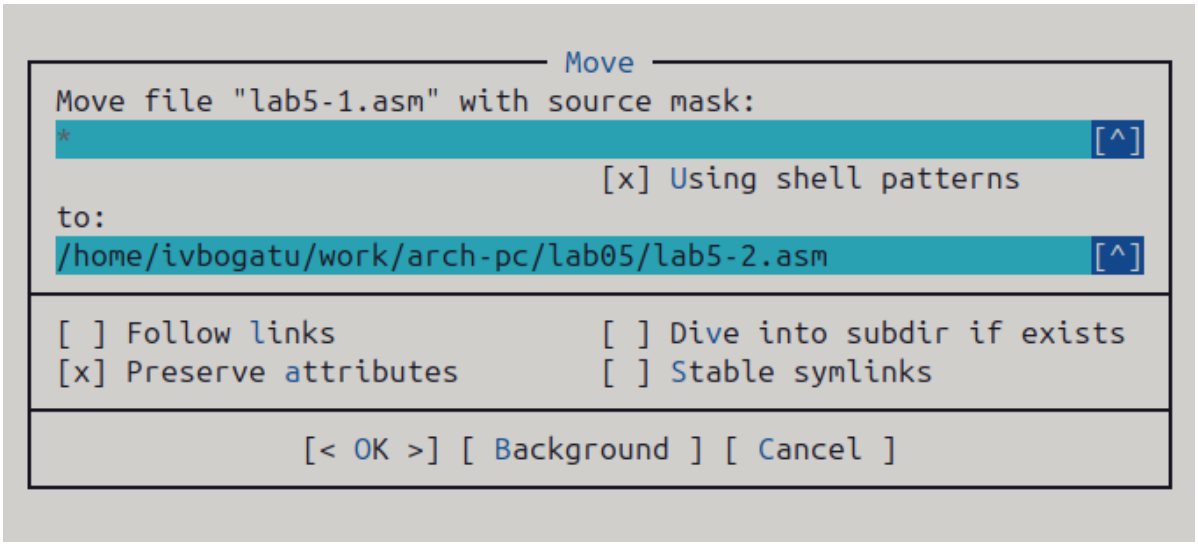

Рис. 15: Копирование файла с помощью F5

Теперь наша папка выглядит следующим образом (Рис. 2.16):

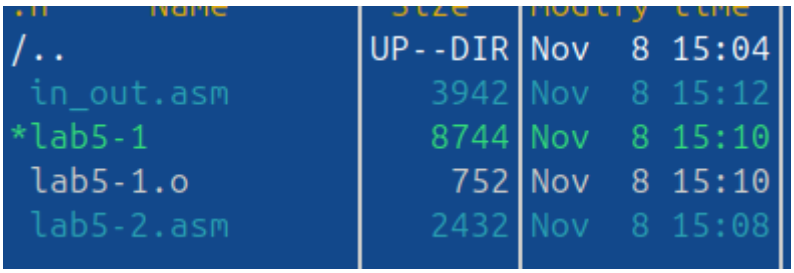

Рис. 16: Текущий вид рабочей папки

Откроем в текстовом редакторе файл lab5-2.asm и напишем туда следующий код (Рис. 2.17):

```
;-------------------------------------------------------------------
; 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 sprintLF ; 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. 17: Редактирование файла lab5-2.asm

После чего создадим исполняемый файл с помощью nasm и ld (Рис. 2.18):

```
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm

ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.0

ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab05$ ./lab5-2
```

Рис. 18: Создание исполняемого файла

Запустим созданный файл (Рис. 2.19):

```
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Богату Ирина Владимировна
```

Рис. 19: Запуск исполняемого файла

Он работает также, как и файл lab5-1, но использует для работы сторонний файл. Попробуем теперь вместо команды sprintLF использовать просто команду sprint (Рис. 2.20):

```
;------------------------------------------------------------------------
; 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 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. 20: Изменение файла lab5-2.asm

Точно также соберём исполняемый файл и запустим его (Рис. 2.21):

```
ivbogatu@ivbogatu-VirtualBox:~/work/arch-pc/lab05$ ./lab5-2
Введите строку: Богату Ирина Владимировна
```

Рис. 21: Запуск изменённого файла

Как мы видим, теперь нет переноса на следующую строку. Этим и отличаются команды sprintLF от sprint. Первая добавляет перенос после текста, а вторая нет

Выполнение задания для самостоятельной работы

Теперь создадим с помощью F6 копию файла lab5-1.asm (Рис. 3.1):

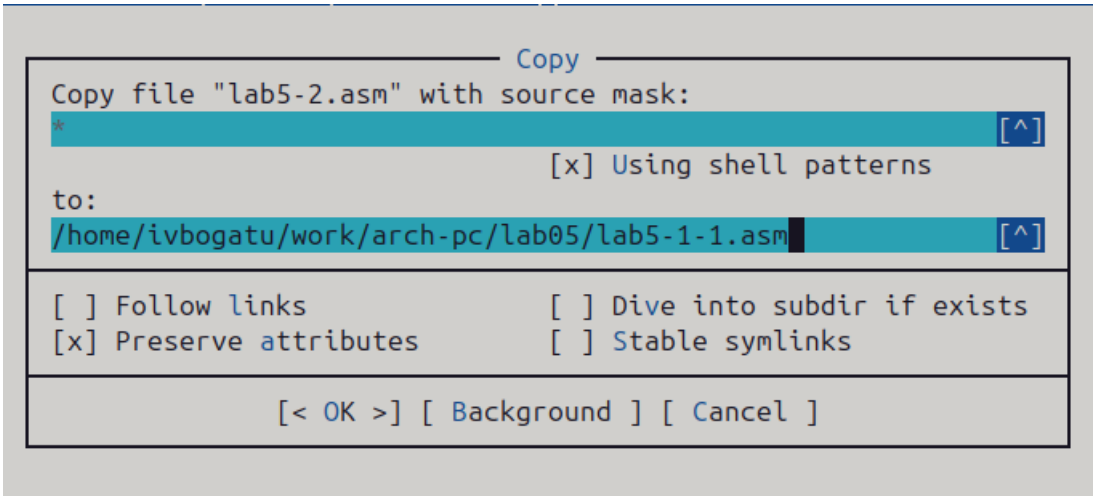

Рис. 1: Создание копии файла lab5-1.asm

Изменим копию так, чтобы она выводила тот текст, который получила на ввод. Для этого перед системным вызовом exit вставим текст с системным вызовом write. Он очень похож на системный вызов write, который уже был в коде, но есть несколько отличий. Так, мы перемещаем адрес строки buf1 в ecx и размер строки buf1 (80) в edx (Рис. 3.2):

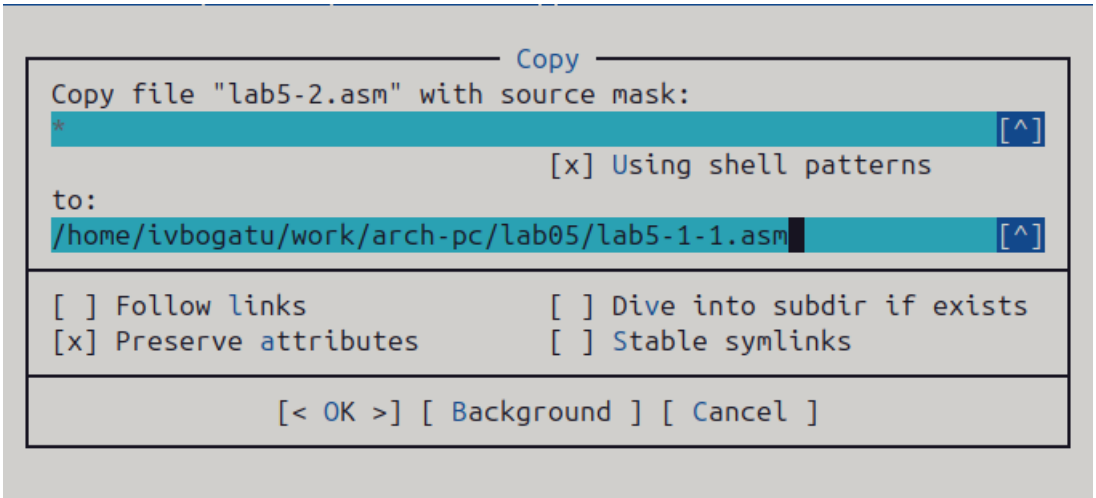

Рис. 2: Изменение файла lab5-1-1.asm

Сохраним изменения и создадим исполняемый файл (Рис. 3.3):

```
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 `write`
; После вызова инструкции 'int 80h' на экран будет
; выведено сообщение из переменной 'buf1' длиной 80
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,80 ; Размер строки 'buf1' в 'edx'
int 80h ; Вызов ядра
;------------ Системный вызов `exit` ---------------------
; После вызова инструкции 'int 80h' программа завершит работу
mov eax,1 ; Системный вызов для выхода (sys_exit)
```

Рис. 3: Создание исполняемого файла

Запустим его и проверим, что всё работает (Рис. 3.4):

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

Рис. 4: Проверка работы программы

Теперь создадим с помощью F5 копию файла lab5-2.asm (Рис. 3.5):

Рис. 5: Создание копии файла lab5-2.asm

теперь сделаем так, чтобы этот код также выводил тот текст, что получит на ввод. Для этого перед последней строкой добавим строчку, которая записывает в eax адрес buf1, а также строчку, которая вызывает подпрограмму sprintLF (Рис. 3.6):

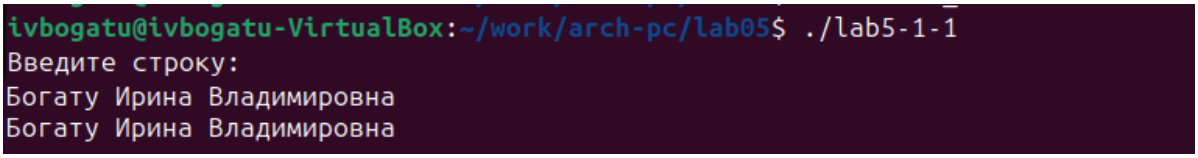

Рис. 6: Изменение файла lab5-2-1.asm

Теперь создадим исполняемый файл (Рис. 3.7):

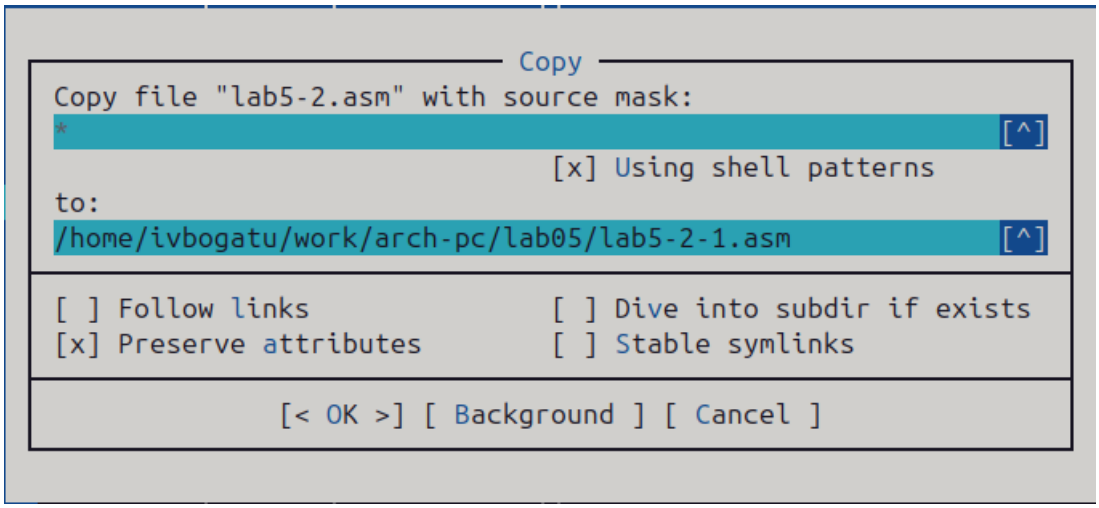

Рис. 7: Создание исполняемого файла

Теперь запустим программу и убедимся, что она работает (Рис. 3.8):

```
;-----------------------------------------------------------------------
; 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 sprintLF ; 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 sprintLF ; вызов подпрограммы печати сообщения
call quit ; вызов подпрограммы завершения
```

Рис. 8: Проверка работы программы

Выводы

В результате выполнения работы были получены навыки работы с Midnight commander, а также навыки написания простых программ ввода-вывода на языке ассемблера