Отчёт по лабораторной работе №6

Дисциплина: Архитектура компьютера

Ищенко Ирина Олеговна

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

Открываем Midnight Commander, переходим в каталог ~/work/arch-pc (рис. 1).

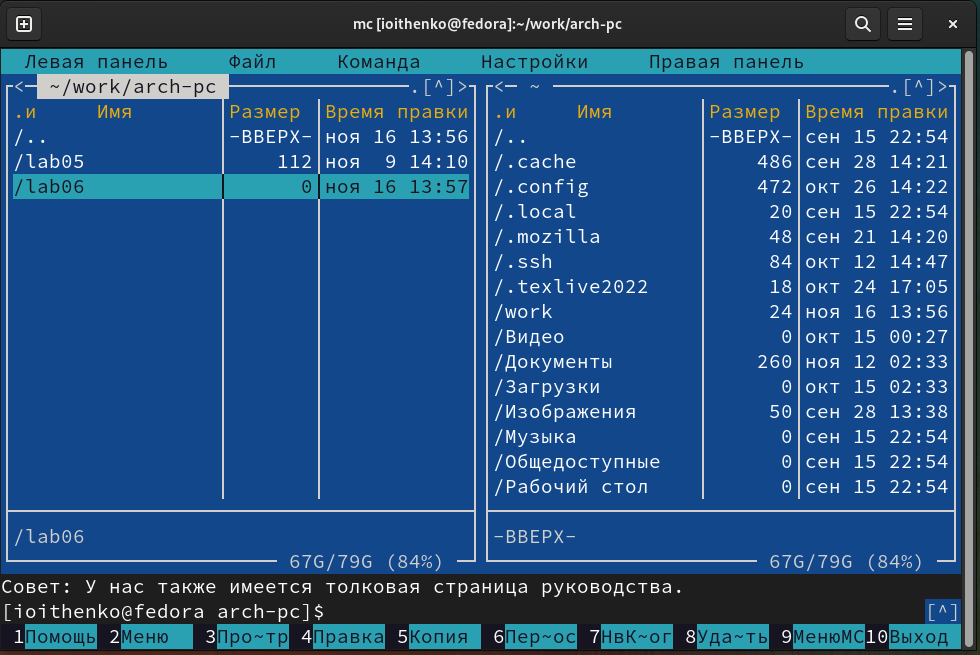


Рис. 1: Каталог arch-pc

С помощью функциональной клавиши F7 создаем папку lab06 и переходим в созданный каталог, создаем файл lab6-1.asm (рис. 2).

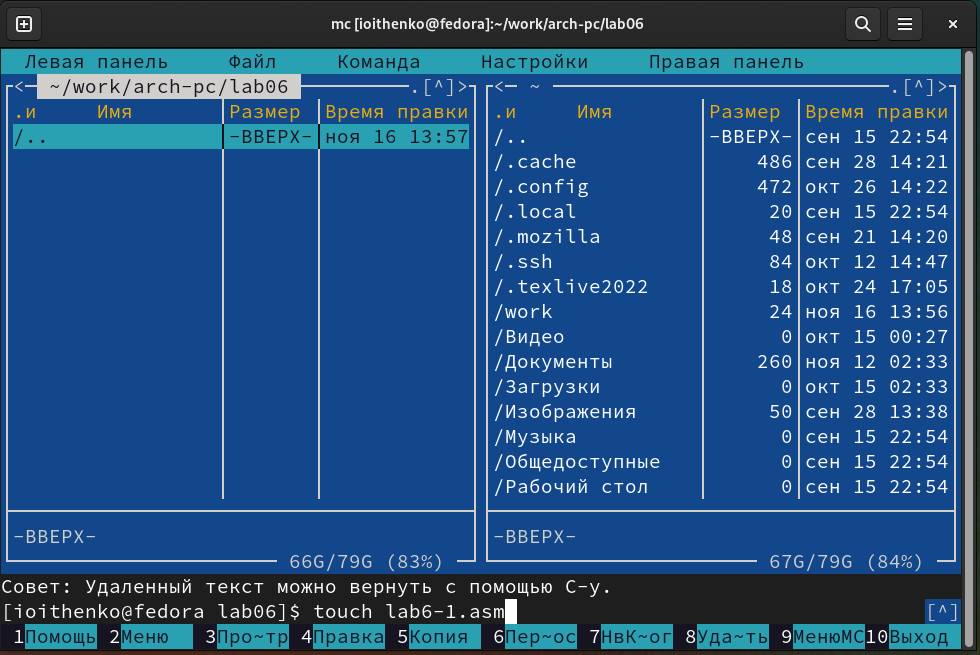


Рис. 2: Каталог lab06 и создание файла

Откроем файл lab6-1.asm для редактирования, введем текст программы (лист. 1) и сохраним изменения (рис. 3).

Листинг 1:

;------------------- Объявление переменных ----------------  
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  
;------------ Cистемный 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

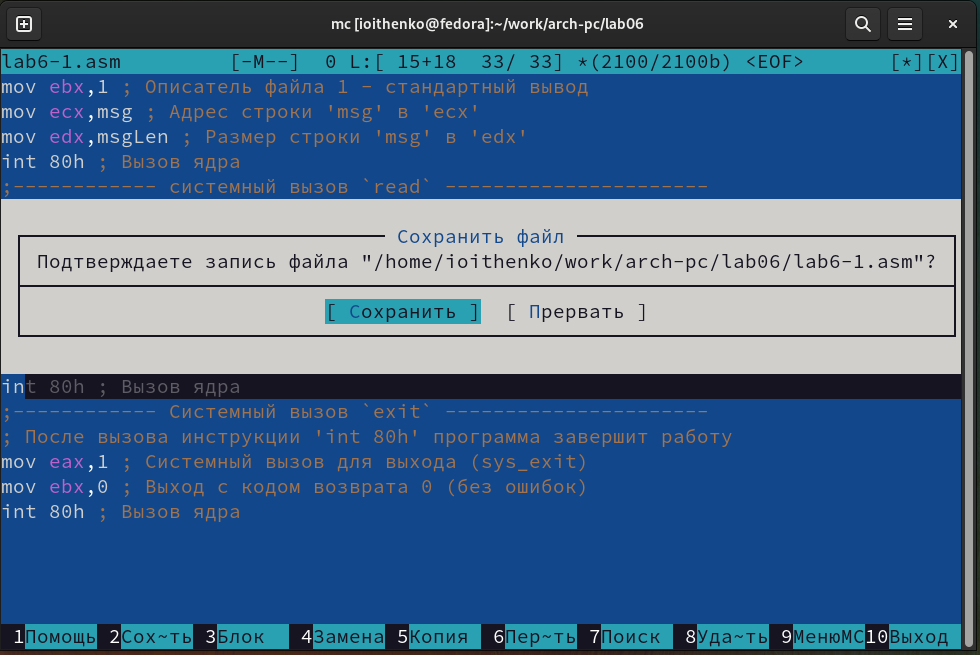


Рис. 3: Сохранение изменений

Проверяем, что изменения сохранены (рис. 4).

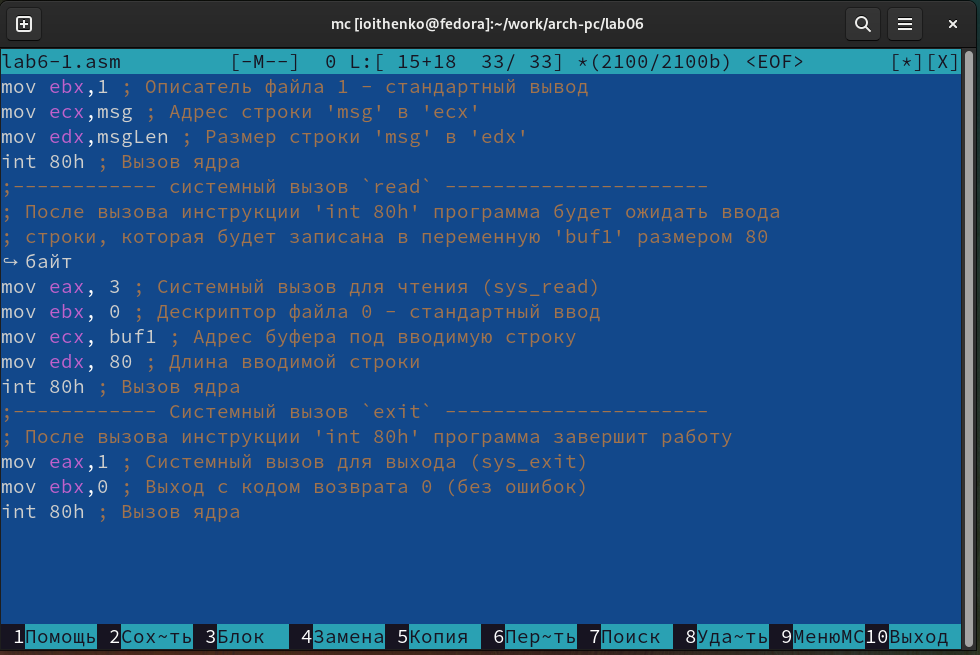


Рис. 4: Файл lab6-1.asm

Компилируем файлы и запускаем программу (рис. 5).

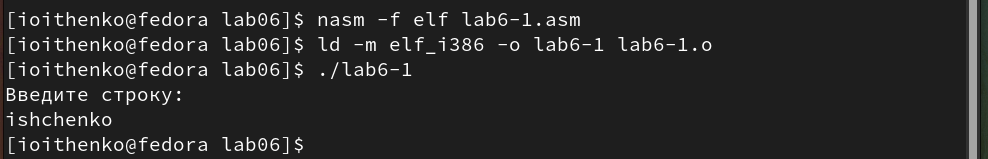


Рис. 5: Запуск программы

Скачиваем файл in\_out.asm, копируем файл в каталог lab06 (рис. 6).

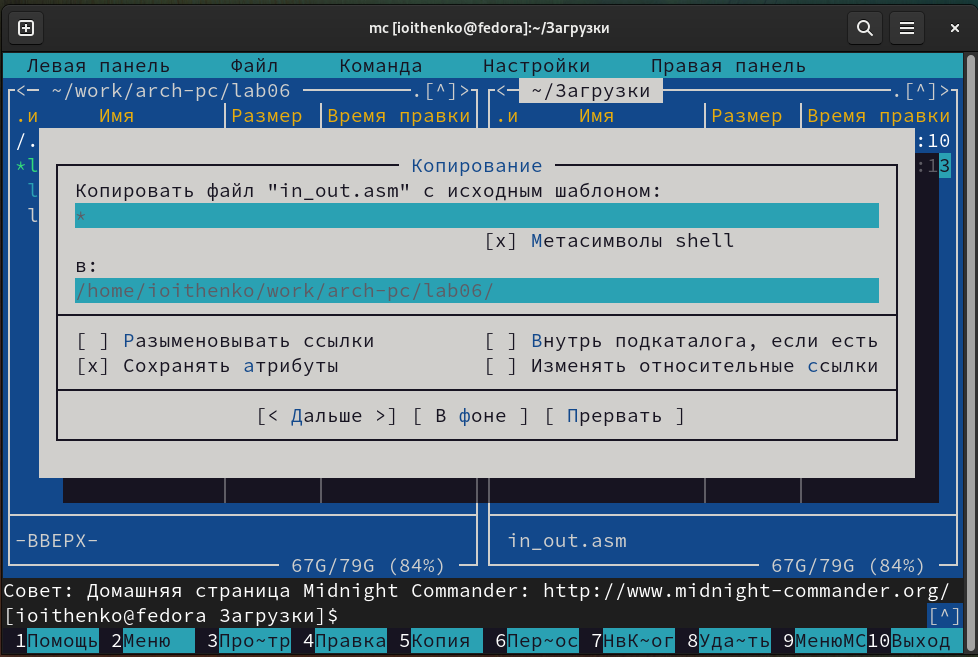


Рис. 6: Копирование файла in\_out.asm

Создаем копию файла lab6-1.asm с именем lab6-2.asm (рис. 7).

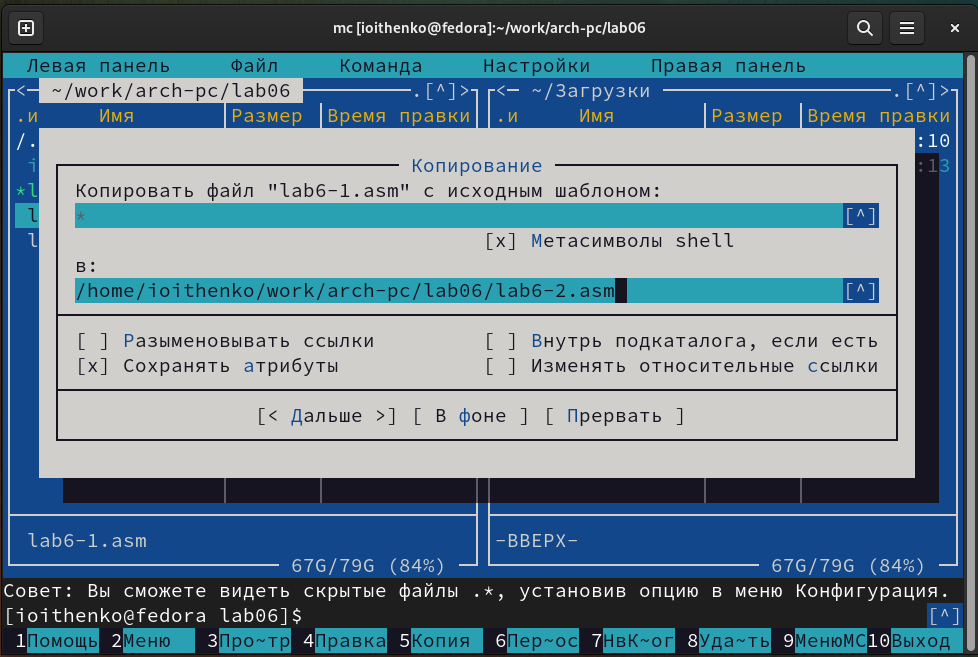


Рис. 7: Копирование файла

Изменяем текст программы с использование подпрограмм из внешнего файла in\_out.asm (лист. 2).

Листинг 2:

;--------------------------------------------------------------------  
; 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

Компилируем файлы и запускаем программу (рис. 8).

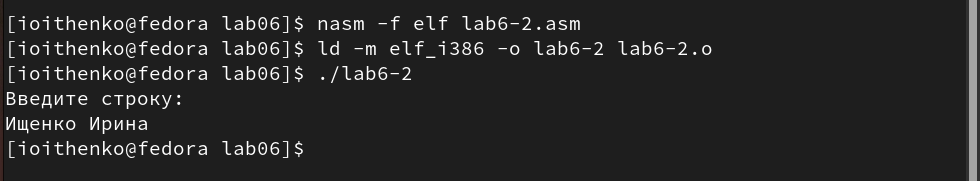


Рис. 8: Запуск второй программы

В файле lab6-2.asm заменяем подпрограмму sprintLF на sprint. Создаем исполняемый файл и проверяем его работу (рис. 9). Разница в работе подпрограмм sprintLF и sprint в том, что sprintLF выводит сообщение с новой строки, а sprint выводит сообщение в той же строке.

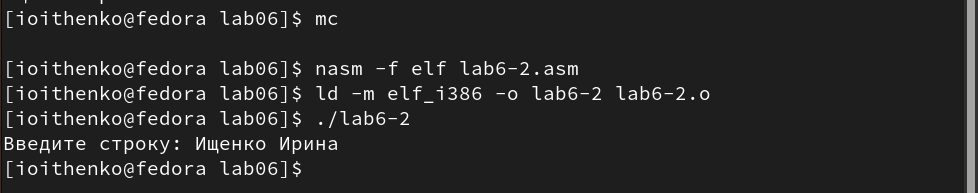


Рис. 9: Запуск измененной второй программы

# 3 Выполнение заданий для самостоятельной работы

Создаем копию файла lab6-1.asm. Вносим изменения в программу (лист. 3), так чтобы она работала по следующему алгоритму: • вывести приглашение типа “Введите строку:”; • ввести строку с клавиатуры; • вывести введённую строку на экран.

Листинг 3:

;------------------- Объявление переменных ----------------  
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  
;------------ Cистемный 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  
  
mov eax,4  
mov ebx,1  
mov ecx,buf1  
mov edx,80  
int 80h  
  
;------------ 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

Запускаем программу (рис. 10).

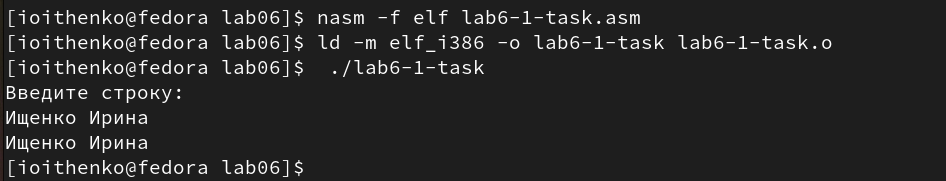


Рис. 10: Запуск программы

Создаем копию файла lab6-2.asm. Исправляем текст программы с использование подпрограмм из внешнего файла in\_out.asm, так чтобы она работала по аналогичному алгоритму (лист. 4)

Листинг 4:

;--------------------------------------------------------------------  
; 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  
call sprint  
call quit ; вызов подпрограммы завершения

Запускаем программу (рис. 11).

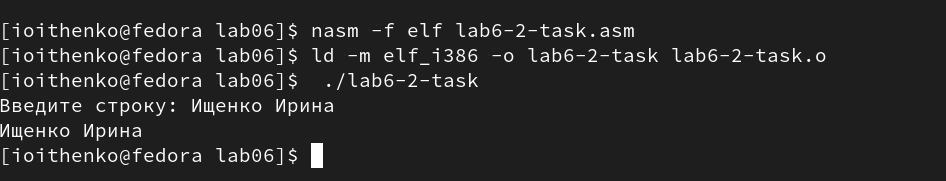


Рис. 11: Запуск программы

# 4 Выводы

В ходе лабораторной работы я приобрела практические навыки работы в Midnight Commander и освоила импользование инструкций языка ассемблера mov и int.