

# **Отчёт по лабораторной работе №6**

**Дисциплина: Архитектура компьютера**

**Ищенко Ирина Олеговна**

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение заданий для самостоятельной работы	13
4	Выводы	17

# Список иллюстраций

2.1	Каталог arch-pc . . . . .	6
2.2	Каталог lab06 и создание файла . . . . .	7
2.3	Сохранение изменений . . . . .	8
2.4	Файл lab6-1.asm . . . . .	9
2.5	Запуск программы . . . . .	9
2.6	Копирование файла in_out.asm . . . . .	10
2.7	Копирование файла . . . . .	10
2.8	Запуск второй программы . . . . .	11
2.9	Запуск измененной второй программы . . . . .	12
3.1	Запуск программы . . . . .	15
3.2	Запуск программы . . . . .	16

## **Список таблиц**

# 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. 2.1).

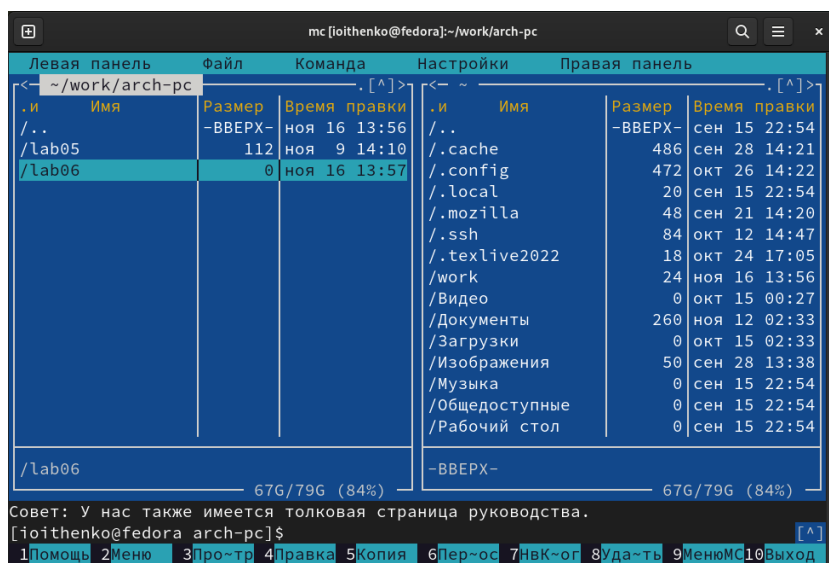


Рис. 2.1: Каталог arch-pc

С помощью функциональной клавиши F7 создаем папку lab06 и переходим в созданный каталог, создаем файл lab6-1.asm (рис. 2.2).

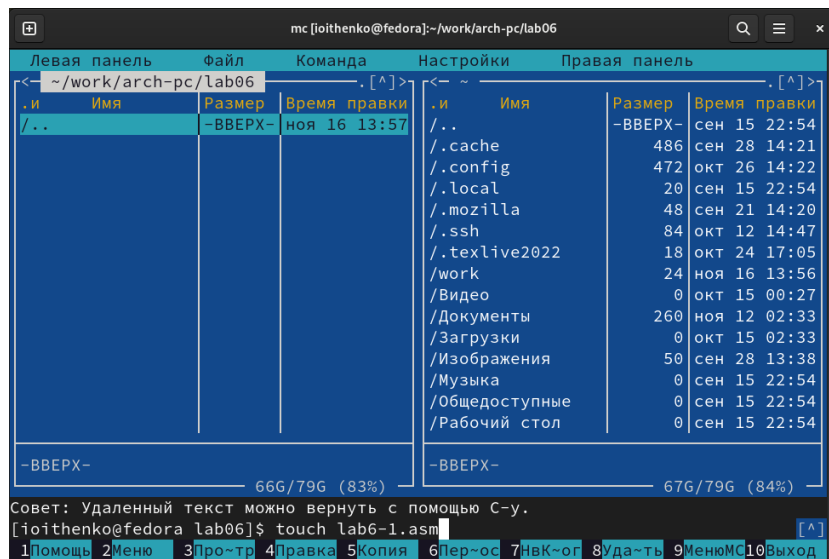


Рис. 2.2: Каталог lab06 и создание файла

Откроем файл lab6-1.asm для редактирования, введем текст программы (лист. 1) и сохраним изменения (рис. 2.3).

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
;----- 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

```

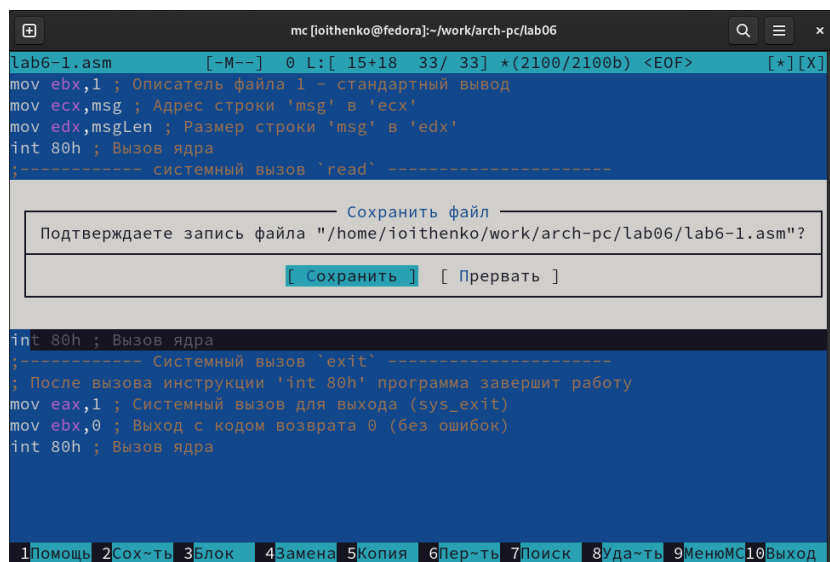
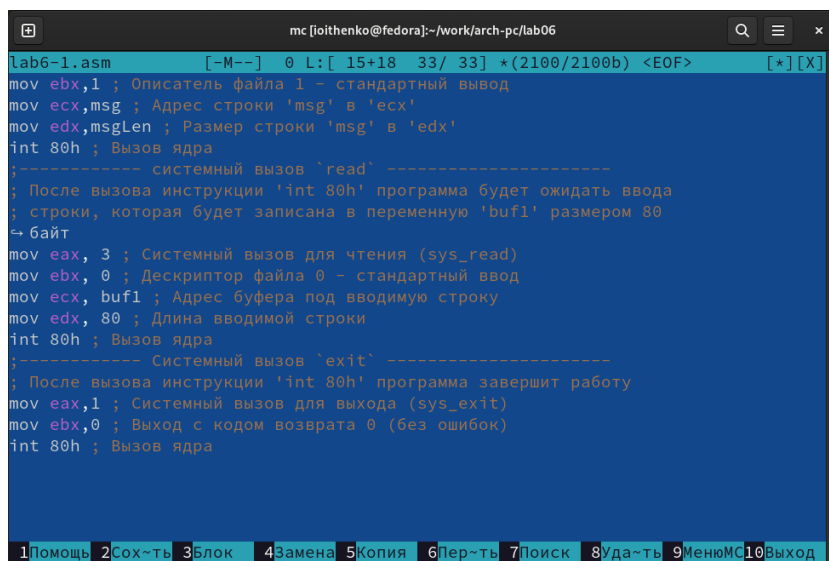


Рис. 2.3: Сохранение изменений



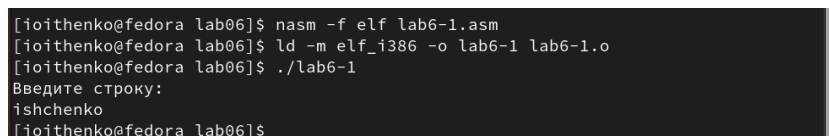
Проверяем, что изменения сохранены (рис. 2.4).



```
lab6-1.asm      [-M--]  0 L: [ 15+18  33/ 33] *(2100/2100b) <EOF>  [*] [X]
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 ;Descriptor 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. 2.4: Файл lab6-1.asm

Компилируем файлы и запускаем программу (рис. 2.5).



```
[ioithenko@fedora lab06]$ nasm -f elf lab6-1.asm
[ioithenko@fedora lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[ioithenko@fedora lab06]$ ./lab6-1
Введите строку:
ishchenko
[ioithenko@fedora lab06]$
```

Рис. 2.5: Запуск программы

Скачиваем файл in\_out.asm, копируем файл в каталог lab06 (рис. 2.6).

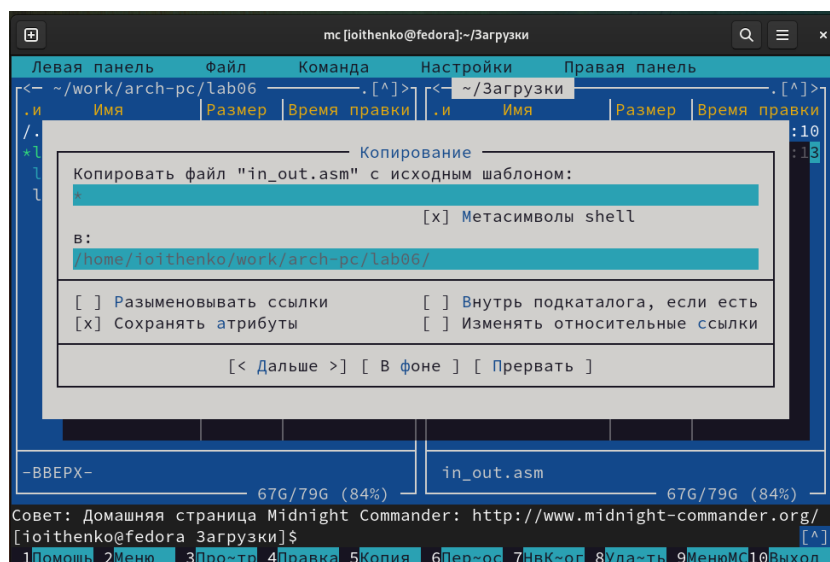


Рис. 2.6: Копирование файла in\_out.asm

Создаем копию файла lab6-1.asm с именем lab6-2.asm (рис. 2.7).

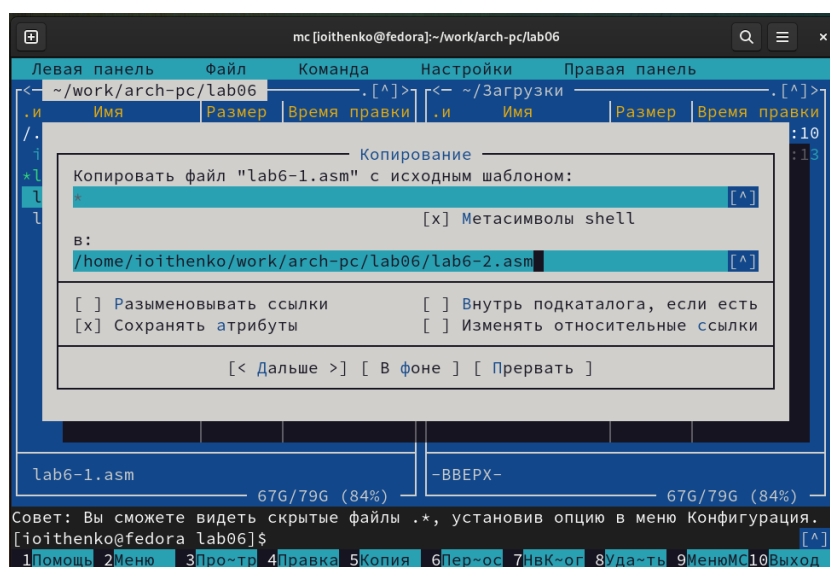


Рис. 2.7: Копирование файла

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

Листинг 2:

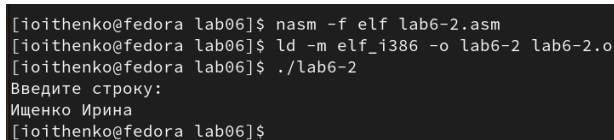
;

```

; 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. 2.8).



```

[ioithenko@fedora lab06]$ nasm -f elf lab6-2.asm
[ioithenko@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[ioithenko@fedora lab06]$ ./lab6-2
Введите строку:
Ищенко Ирина
[ioithenko@fedora lab06]$

```

Рис. 2.8: Запуск второй программы

В файле lab6-2.asm заменяем подпрограмму sprintLF на sprint. Создаем исполняемый файл и проверяем его работу (рис. 2.9). Разница в работе подпрограмм sprintLF и sprint в том, что sprintLF выводит сообщение с новой строки, а sprint выводит сообщение в той же строке.

```
[ioithenko@fedora lab06]$ mc  
[ioithenko@fedora lab06]$ nasm -f elf lab6-2.asm  
[ioithenko@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o  
[ioithenko@fedora lab06]$ ./lab6-2  
Введите строку: Ищенко Ирина  
[ioithenko@fedora lab06]$
```

Рис. 2.9: 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 введенную 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  
;----- 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 ;Descriptor 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. 3.1).

```
[ioithenko@fedora lab06]$ nasm -f elf lab6-1-task.asm
[ioithenko@fedora lab06]$ ld -m elf_i386 -o lab6-1-task lab6-1-task.o
[ioithenko@fedora lab06]$ ./lab6-1-task
Введите строку:
Ищенко Ирина
Ищенко Ирина
[ioithenko@fedora lab06]$
```

Рис. 3.1: 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. 3.2).

```
[ioithenko@fedora lab06]$ nasm -f elf lab6-2-task.asm
[ioithenko@fedora lab06]$ ld -m elf_i386 -o lab6-2-task lab6-2-task.o
[ioithenko@fedora lab06]$ ./lab6-2-task
Введите строку: Ищенко Ирина
Ищенко Ирина
[ioithenko@fedora lab06]$
```

Рис. 3.2: Запуск программы



## 4 Выводы

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