

Отчет по лабораторной работа №6

Группа НПИбд-02-22

Стариков Данила Андреевич

Содержание

1	Цель работы	3
2	Основная часть	4
2.1	Выполнение лабораторной работы	4
2.1.1	Подключение внешнего файла in_out.asm	6
2.2	Выполнение заданий для самостоятельной работы.	7
3	Выводы	13

1 Цель работы

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

2 Основная часть

2.1 Выполнение лабораторной работы

С помощью Midnight Commander открыли каталог `~/work/arch-pc` и создали папку `lab06` (клавиша F7) для текущей работы (Рис. 2.1).

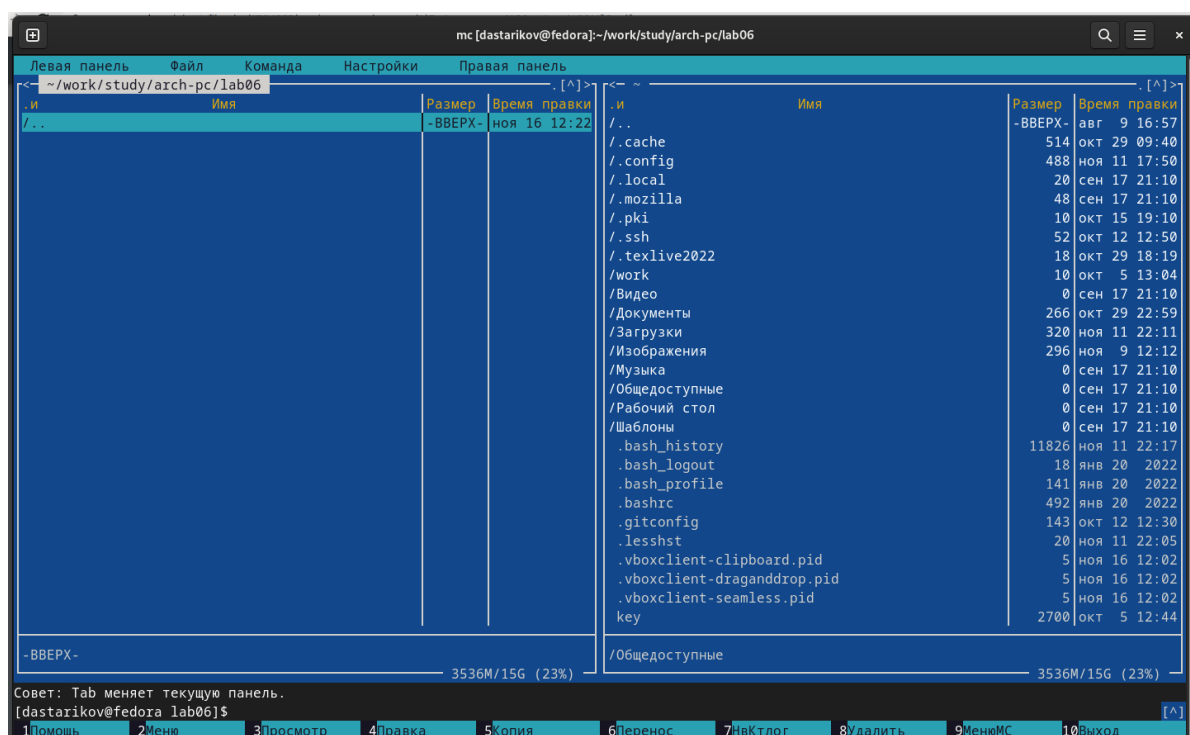
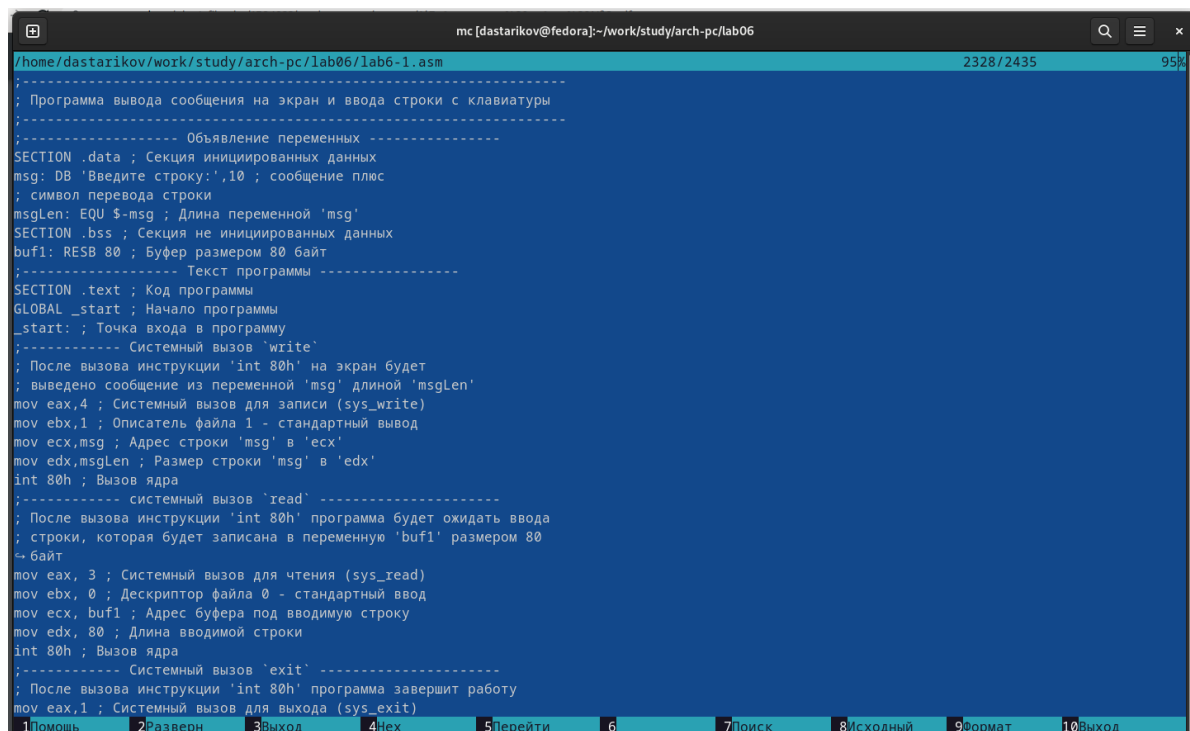


Рис. 2.1: Окно Midnight Commander. На левой стороне открыт каталог `lab06`.

Создали новый файл `lab6.asm` с помощью команды `touch`, открыли через редактор `mcedit` по клавише F4 и скопировали текст программы из Листинга 2.1.

Сохранили файл и проверили, что текст сохранился, открыв файл в режиме

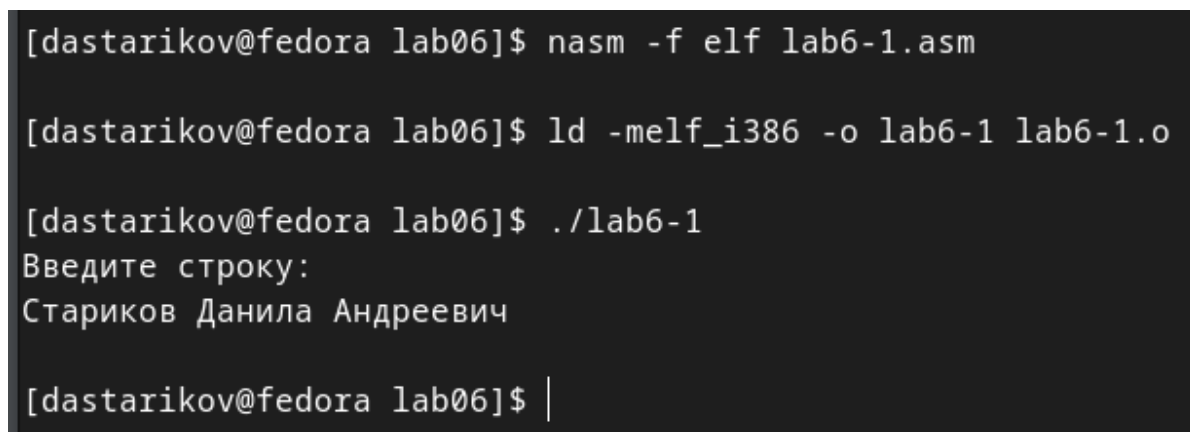
чтения (Рис. 2.2).



```
mc [dastarikov@fedora]~/.work/study/arch-pc/lab06
/home/dastarikov/work/study/arch-pc/lab06/lab6-1.asm 2328/2435 95%
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----
;----- Объявление переменных -----
SECTION .data ; Секция инициализированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не инициализированных 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)
```

Рис. 2.2: Просмотр текста программы lab6-1.asm в режиме чтения.

Выполнили трансляцию текста программы, компоновку полученного объектного файла и запустили программу (Рис. 2.3).



```
[dastarikov@fedora lab06]$ nasm -f elf lab6-1.asm
[dastarikov@fedora lab06]$ ld -melf_i386 -o lab6-1 lab6-1.o
[dastarikov@fedora lab06]$ ./lab6-1
Введите строку:
Стариков Данила Андреевич
[dastarikov@fedora lab06]$ |
```

Рис. 2.3: Результат работы программы lab6-1.

2.1.1 Подключение внешнего файла in_out.asm

Скачали файл in_out.asm со страницы курса в ТУИС и положили в рабочий каталог.

С помощью клавиши F5 создали копию файла lab6-1.asm с именем lab6-2.asm и в соответствии с Листингом 2.2 исправили текст программы (Рис. 2.4). Создали и запустили исполняемый файл (Рис. 2.5). Если в тексте программы заменить подпрограмму sprintLF на sprint, то после вывода строки не будет печататься пустая строка (Рис. 2.6).

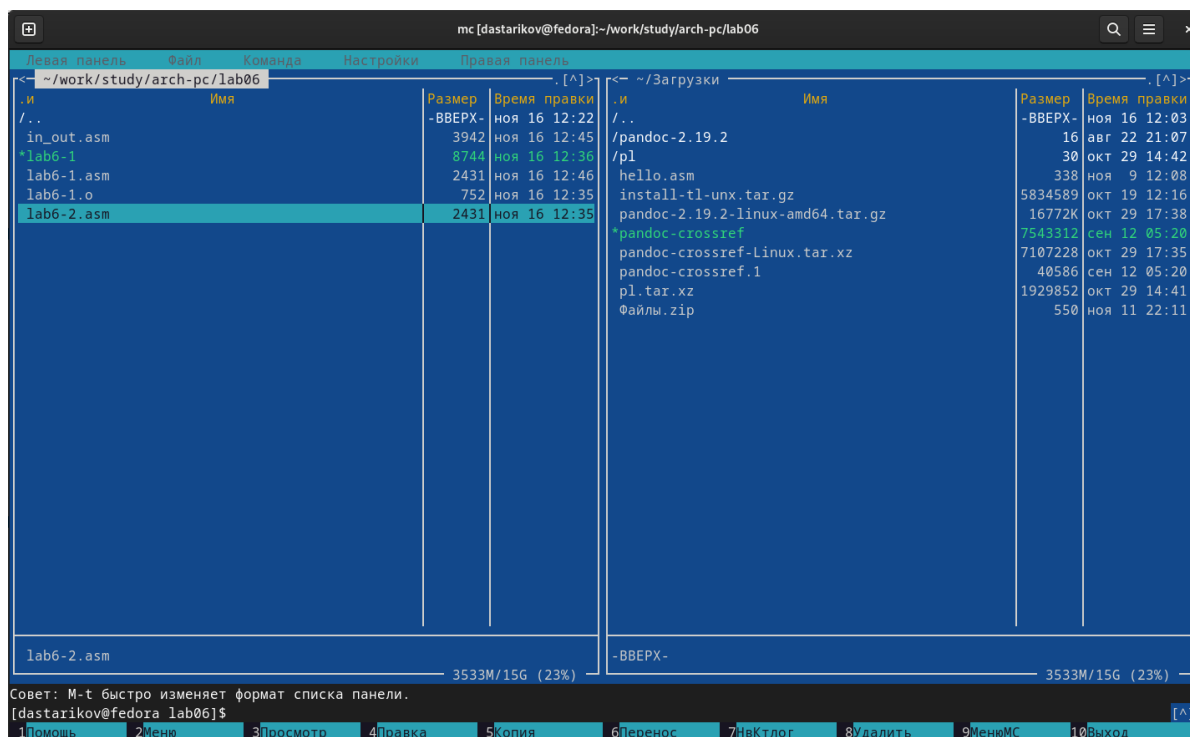


Рис. 2.4: Каталог lab06 после создания файла lab6-2.asm

```
[dastarikov@fedora lab06]$ nasm -f elf lab6-2.asm
[dastarikov@fedora lab06]$ ld -melf_i386 -o lab6-2 lab6-2.o
[dastarikov@fedora lab06]$ ./lab6-2
Введите строку:
Стариков Данила Андреевич
```

Рис. 2.5: Результат выполнения программы lab6-1 с подпрограммой sprintLF

```
[dastarikov@fedora lab06]$ nasm -f elf lab6-2.asm
[dastarikov@fedora lab06]$ ld -melf_i386 -o lab6-2 lab6-2.o
[dastarikov@fedora lab06]$ ./lab6-2
Введите строку:
Стариков Данила Андреевич
[dastarikov@fedora lab06]$ |
```

Рис. 2.6: Результат выполнения программы lab6-1 с подпрограммой sprint

2.2 Выполнение заданий для самостоятельной работы.

Создали копию lab6-1-new.asm файла lab6-1.asm, в котором написали блок вывода строки, полученной вводом с клавиатуры (Листинг 2.3). Результат работы программы на Рисунке 2.7

```
[dastarikov@fedora lab06]$ nasm -f elf lab6-1-new.asm
[dastarikov@fedora lab06]$ ld -m elf_i386 -o lab6-1-new lab6-1-new.o
[dastarikov@fedora lab06]$ ./lab6-1-new
Введите строку:
Стариков
Стариков
[dastarikov@fedora lab06]$ |
```

Рис. 2.7: Результат выполнения программы lab6-1-new.

Создали копию lab6-2-new.asm файла lab6-2.asm, в котором, воспользовавшись подпрограммами из файла in_out.asm, также написали блок вывода строки, полученной вводом строки с клавиатуры (Листинг 2.4). Результат работы программы на Рисунке 2.8.

```
[dastarikov@fedora lab06]$ nasm -f elf lab6-2-new.asm
[dastarikov@fedora lab06]$ ld -melf_i386 -o lab6-2-new lab6-2-new.o
[dastarikov@fedora lab06]$ ./lab6-2-new
Введите строку:
Стариков
Стариков

[dastarikov@fedora lab06]$ |
```

Рис. 2.8: Результат выполнения программы lab6-2-new.

Созданные файлы *.asm скопировали в каталог ~/work/study/2022-2023/“Архитектура компьютера”/archpc/labs/lab06/ и загрузили на Github.

Листинг 2.1 Программа вывода сообщения на экран и ввода строки с клавиатуры.

```
;-----  
; Программа вывода сообщения на экран и ввода строки с клавиатуры  
;-----  
;----- 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
```

Листинг 2.2 Текст программы lab6-2.asm

```
;-----  
; Программа вывода сообщения на экран и ввода строки с клавиатуры  
;-----  
%include 'in_out.asm'  
  
SECTION .data ; Секция инициированных данных  
msg: DB 'Введите строку:',10 ; сообщение плюс  
; символ перевода строки  
  
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 программы  
    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
```

Листинг 2.3 Текст программы lab6-1-new.asm

```
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 `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)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра
```

Листинг 2.4 Текст программы lab6-2-new.asm

```
%include 'in_out.asm'
```

```
SECTION .data ; Секция инициированных данных  
msg: DB 'Введите строку:',10 ; сообщение плюс  
; символ перевода строки
```

```
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 программы
```

```
    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  
ати сообщения
```

```
    mov eax, buf1 ; запись адреса выводимого сообщения в `EAX`
```

```
    call sprint ; вызов подпрограммы печ
```

```
    call quit ; вызов подпрограммы завершения
```

3 Выводы

В рамках лабораторной работы получили практические навыки работы в Midnight Commander, освоили инструкции языка ассемблера и использование внешних файлов.