

Лабораторная работа №5

Архитектура вычислительных систем

Заболотная Кристина

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Самостоятельная часть лабораторной работы	17
6	Выводы	20
	Список литературы	21

Список иллюстраций

4.1	51.png	9
4.2	52.png	10
4.3	53.png	10
4.4	54.png	11
4.5	55.png	11
4.6	56.png	12
4.7	57.png	13
4.8	58.png	14
4.9	59.png	14
4.10	510.png	15
4.11	511.png	15
4.12	512.png	16
5.1	513.png	17
5.2	514.png	18
5.3	515.png	18
5.4	516.png	19
5.5	517.png	19

Список таблиц

1 Цель работы

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

2 Задание

1. Создать копию файла lab5-1.asm. Внести изменения в программу (без использования внешнего файла in_out.asm), так чтобы она работала по следующему алгоритму: • вывести приглашение типа “Введите строку:”; • ввести строку с клавиатуры; • вывести введенную строку на экран.
2. Получить исполняемый файл и проверить его работу. На приглашение ввести строку, свою фамилию.
3. Создать копию файла lab5-2.asm. Исправить текст программы с использованием подпрограмм из внешнего файла in_out.asm, так чтобы она работала по следующему алгоритму: • вывести приглашение типа “Введите строку:”; • ввести строку с клавиатуры; • вывести введенную строку на экран. Подключаемый файл in_out.asm должен лежать в том же каталоге, что и файл с программой, в которой он используется.
4. Создать исполняемый файл и проверить его работу.

3 Теоретическое введение

Midnight Commander (или просто `mc`) — это программа, которая позволяет просматривать структуру каталогов и выполнять основные операции по управлению файловой системой, т.е. `mc` является файловым менеджером. Midnight Commander позволяет сделать работу с файлами более удобной и наглядной. Для активации оболочки Midnight Commander достаточно ввести в командной строке `mc` и нажать клавишу `Enter`. Директивы используются для объявления простых переменных и для объявления массивов. Для определения строк принято использовать директиву `DB` в связи с особенностями хранения данных в оперативной памяти. Простейший диалог с пользователем требует наличия двух функций — вывода текста на экран и ввода текста с клавиатуры. Простейший способ вывести строку на экран — использовать системный вызов `write`. Этот системный вызов имеет номер 4, поэтому перед вызовом инструкции `int` необходимо поместить значение 4 в регистр `eax`. Первым аргументом `write`, помещаемым в регистр `ebx`, задаётся дескриптор файла. Для вывода на экран в качестве дескриптора файла нужно указать 1 (это означает «стандартный вывод», т. е. вывод на экран). Вторым аргументом задаётся адрес выводимой строки (помещаем его в регистр `ecx`, например, инструкцией `mov ecx, msg`). Строка может иметь любую длину. Последним аргументом (т.е. в регистре `edx`) должна задаваться максимальная длина выводимой строки. Для ввода строки с клавиатуры можно использовать аналогичный системный вызов `read`. Его аргументы – такие же, как у вызова `write`, только для «чтения» с клавиатуры используется файловый дескриптор 0 (стандартный ввод). Системный вызов `exit` является обязательным в конце лю-

бой программы на языке ассемблер. Для обозначения конца программы перед вызовом инструкции `int 80h` необходимо поместить в регистр `eax` значение 1, а в регистр `ebx` код завершения 0.

4 Выполнение лабораторной работы

1. Откроем Midnight Commander.

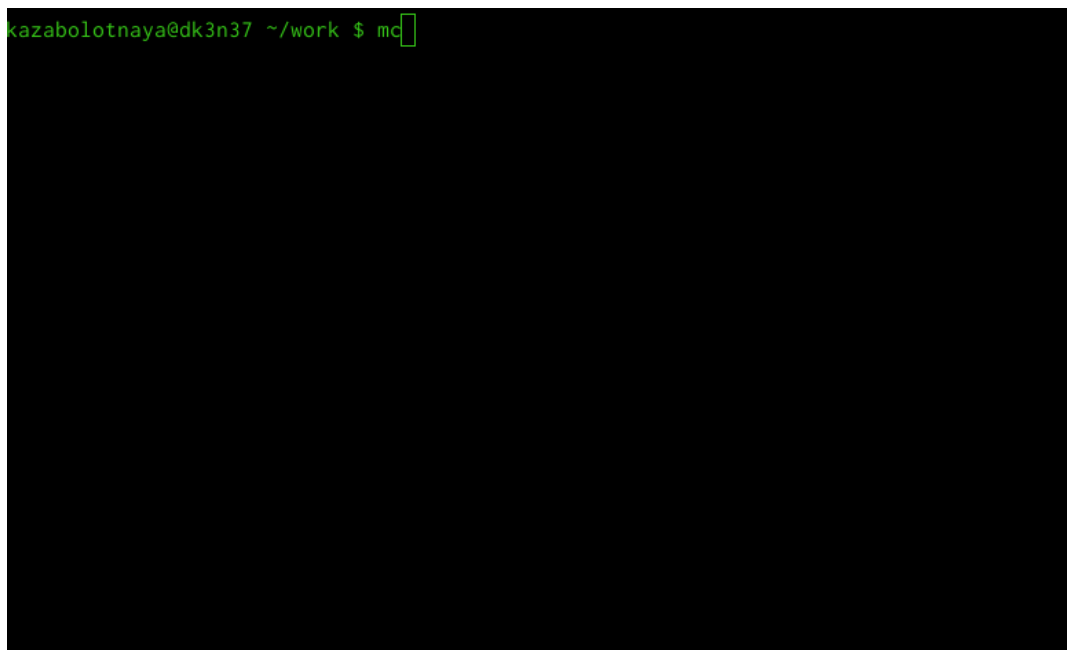


Рис. 4.1: 51.png

2. Создаем директорию.

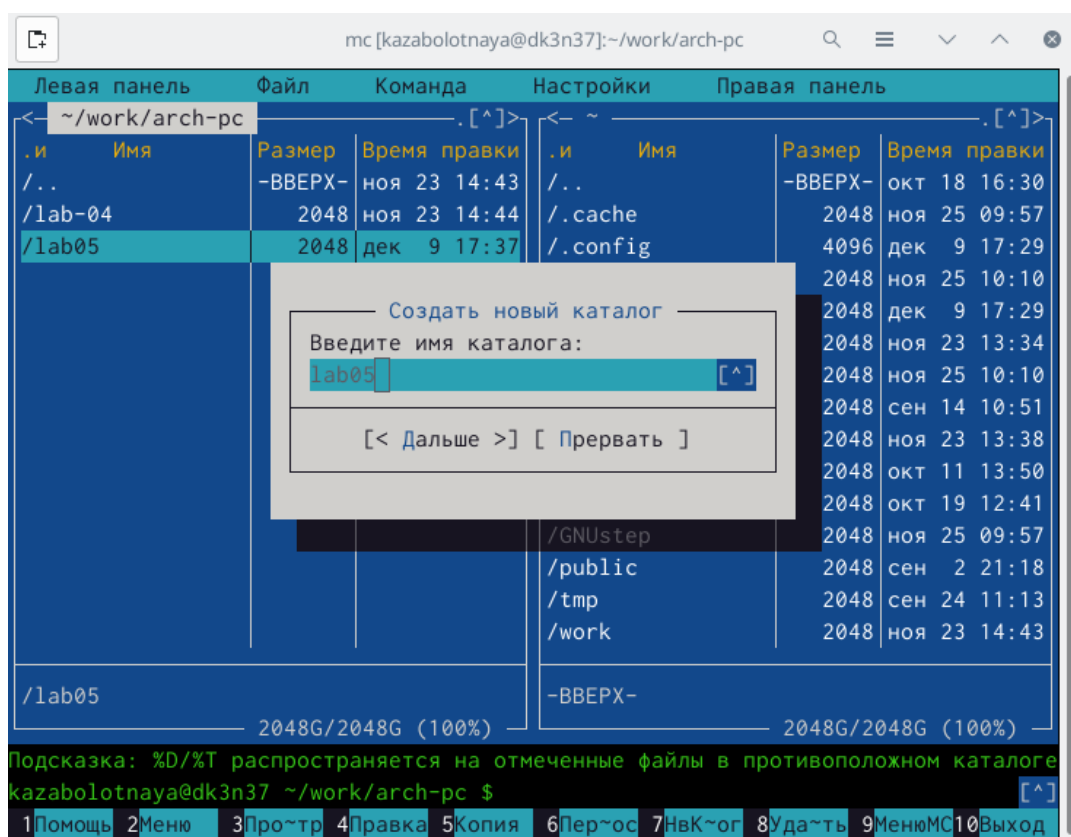


Рис. 4.2: 52.png

3. Создаем файл lab5-1.

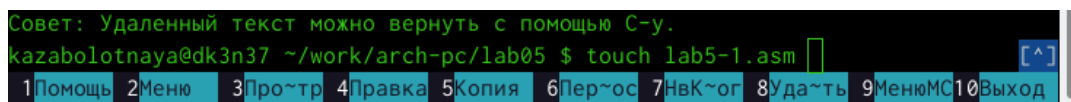


Рис. 4.3: 53.png

4. Вводим текст программы из листинга.

```
GNU nano 6.3 lab5-1.asm
SECTION .data ; Секция инициированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строкиmsgLen: 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 ; ОписательmsgLen: 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
[ Прочитано 48 строк ]
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить M-U Отмена
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^C Позиция M-E Повтор
```

Рис. 4.4: 54.png

5. Вводим текст программы из листинга.

```
kazabolotnaya@dk3n37 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1.asm
kazabolotnaya@dk3n37 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1 lab5-1.o
kazabolotnaya@dk3n37 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:
Кристина Заболотная Александровна
```

Рис. 4.5: 55.png

6. Создаем файл lab5-2.

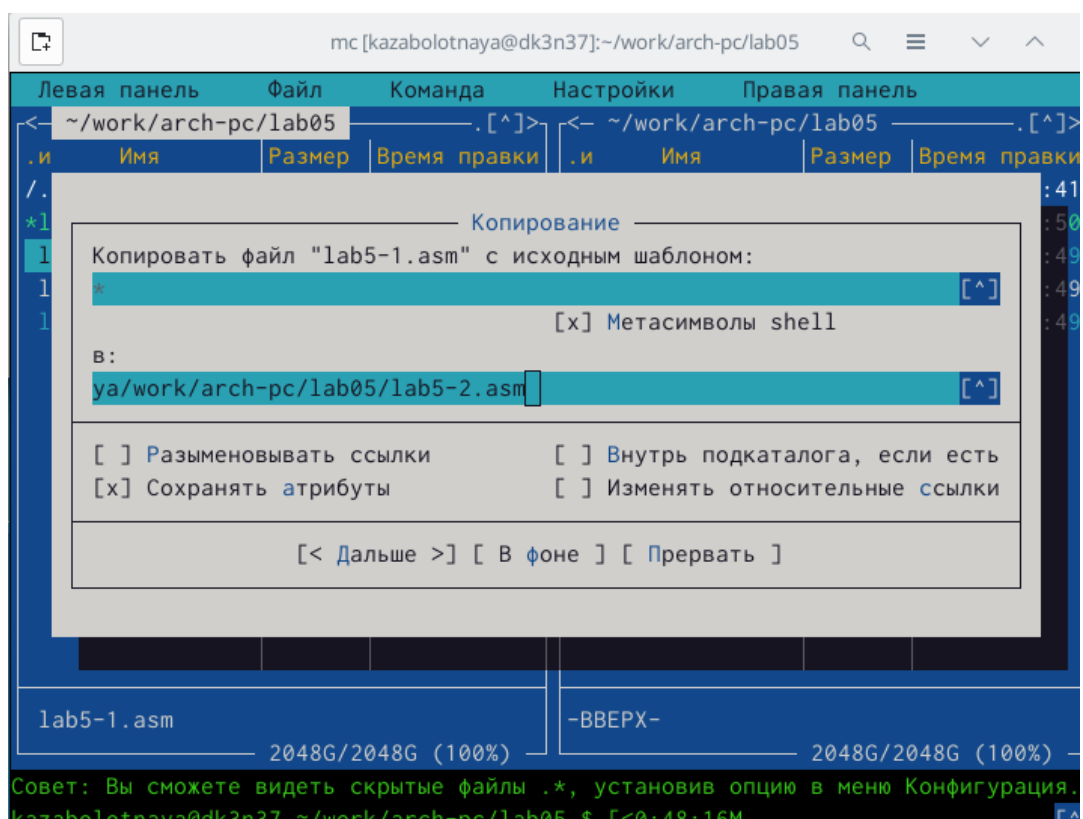
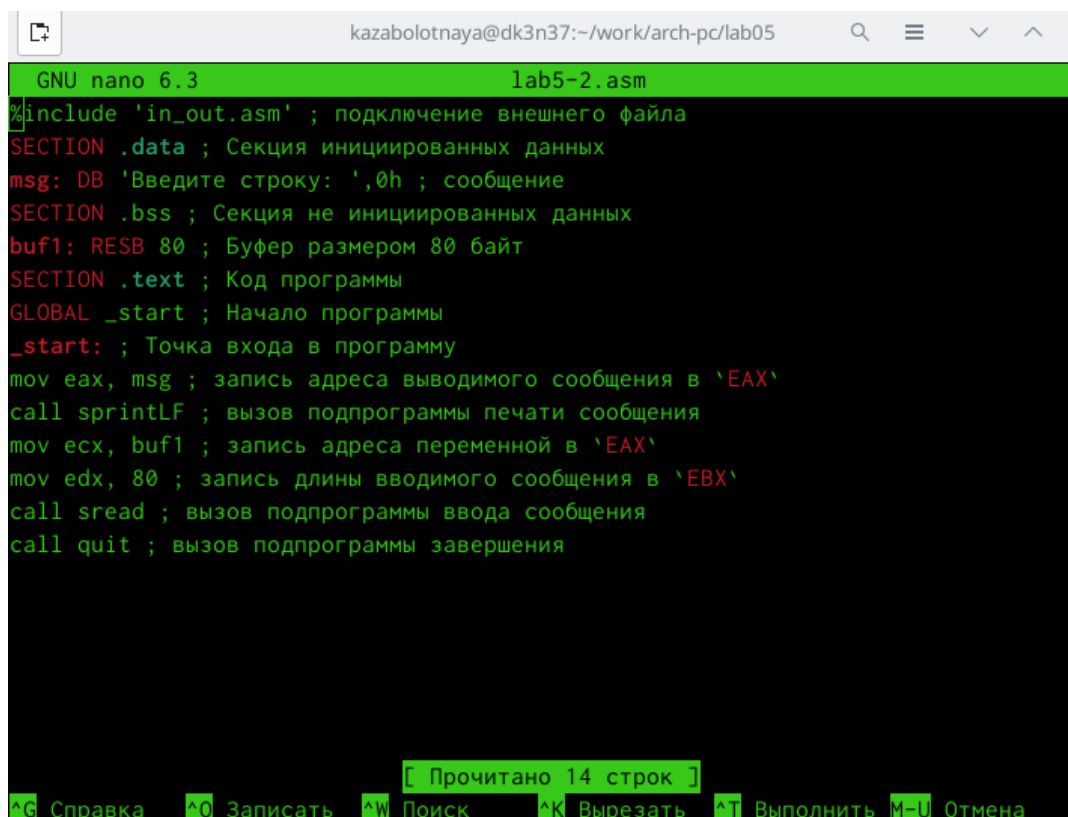


Рис. 4.6: 56.png

7. Изменяем код в lab5-2.



```
GNU nano 6.3 lab5-2.asm
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не иницированных 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 sprintf ; 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

[ Прочитано 14 строк ]
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить M-U Отмена
```

Рис. 4.7: 57.png

8. Переносим текст из заданий к лабораторной работе.

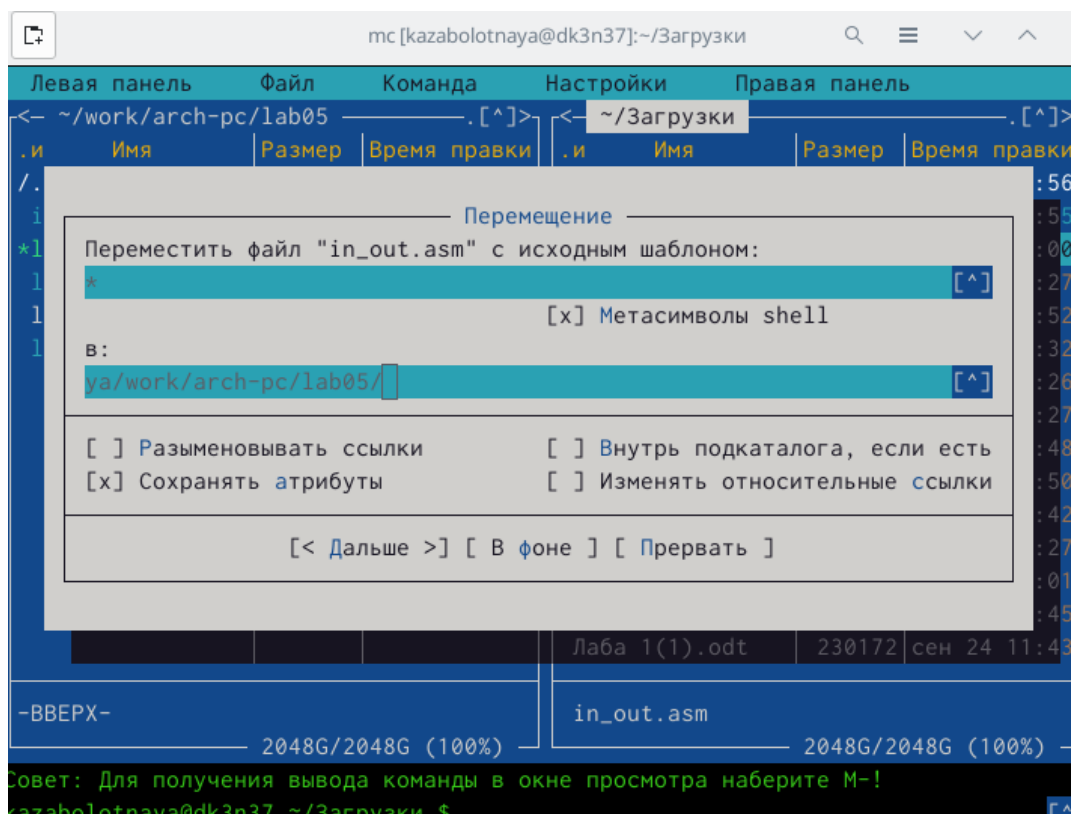


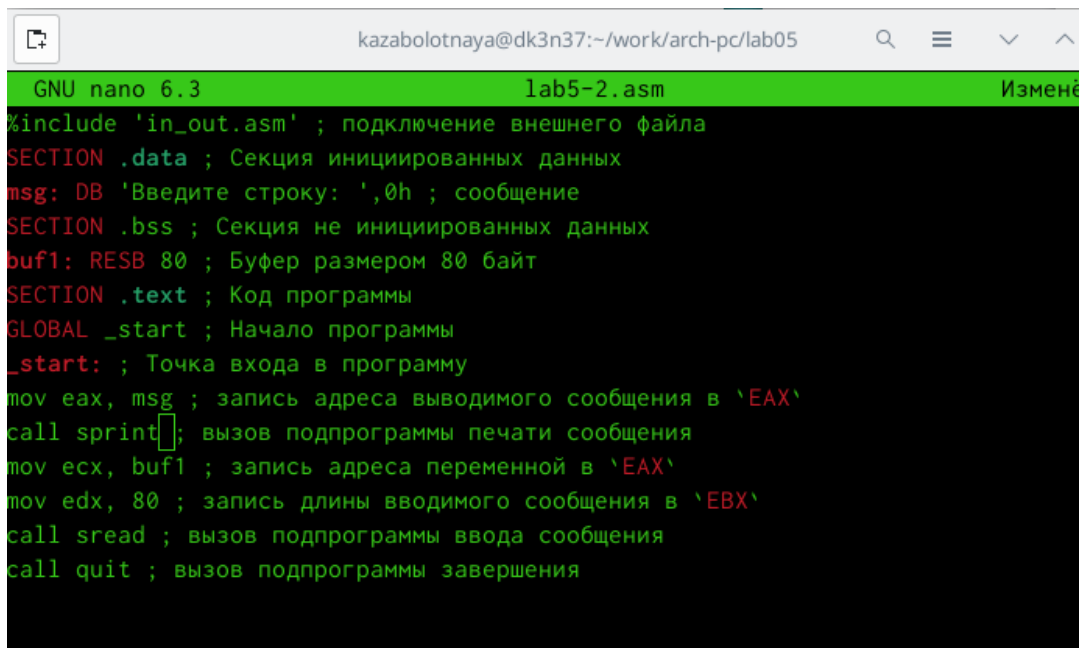
Рис. 4.8: 58.png

9. Вводим свои ФИО, работая в lab5-2.

```
kazaboltnaya@dk3n37 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
kazaboltnaya@dk3n37 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
kazaboltnaya@dk3n37 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку:
Кристина Заболотная Александровна
```

Рис. 4.9: 59.png

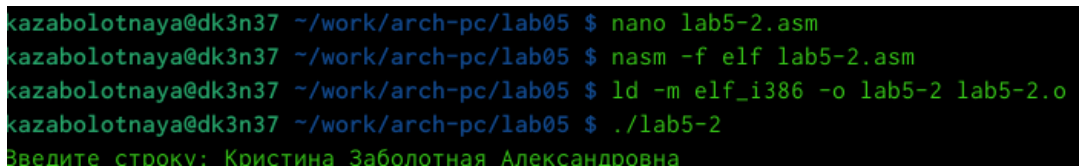
10. В файле lab5-2.asm заменим подпрограмму sprintLF на sprint.



```
GNU nano 6.3 lab5-2.asm
%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. 4.10: 510.png

11. В файле lab5-2.asm заменим подпрограмму sprintLF на sprint.



```
kazabolotnaya@dk3n37 ~/work/arch-pc/lab05 $ nano lab5-2.asm
kazabolotnaya@dk3n37 ~/work/arch-pc/lab05 $ nasm -f elf lab5-2.asm
kazabolotnaya@dk3n37 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-2 lab5-2.o
kazabolotnaya@dk3n37 ~/work/arch-pc/lab05 $ ./lab5-2
Введите строку: Кристина Заболотная Александровна
```

Рис. 4.11: 511.png

12. Создадим исполняемый файл и проверим его работу. Посмотрим, в чем разница между sprintLF и sprint.

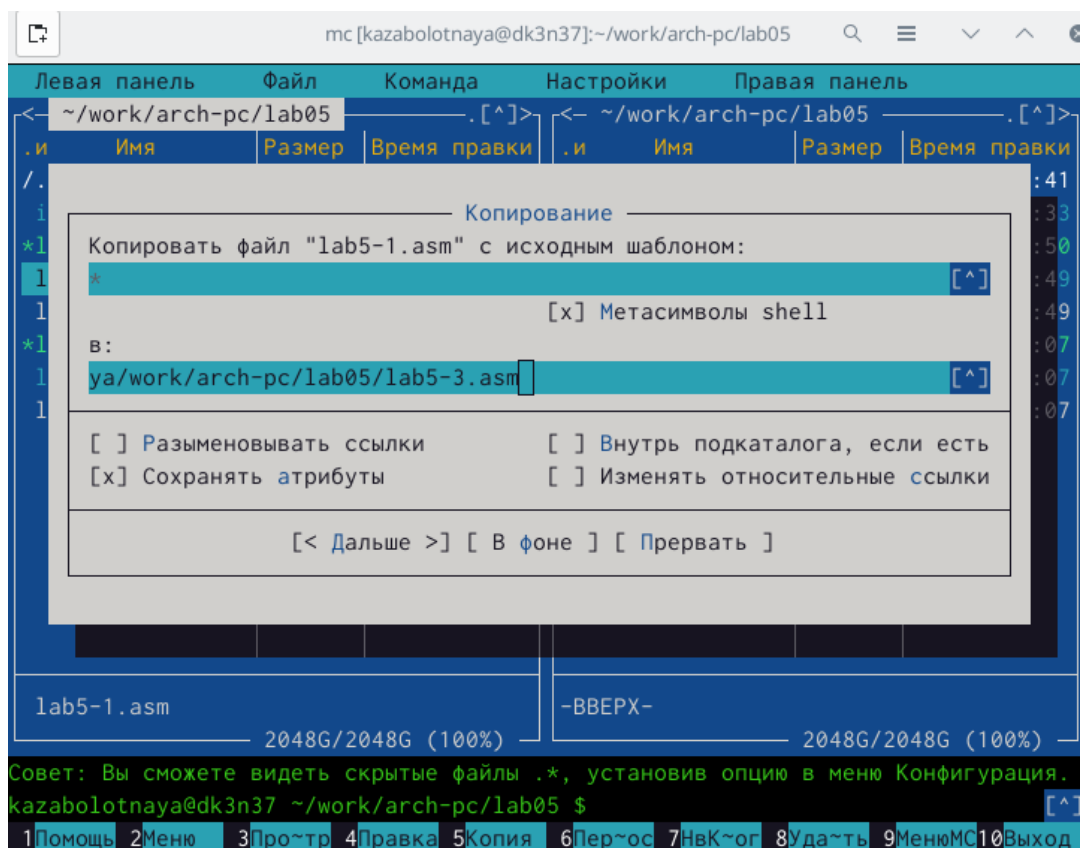
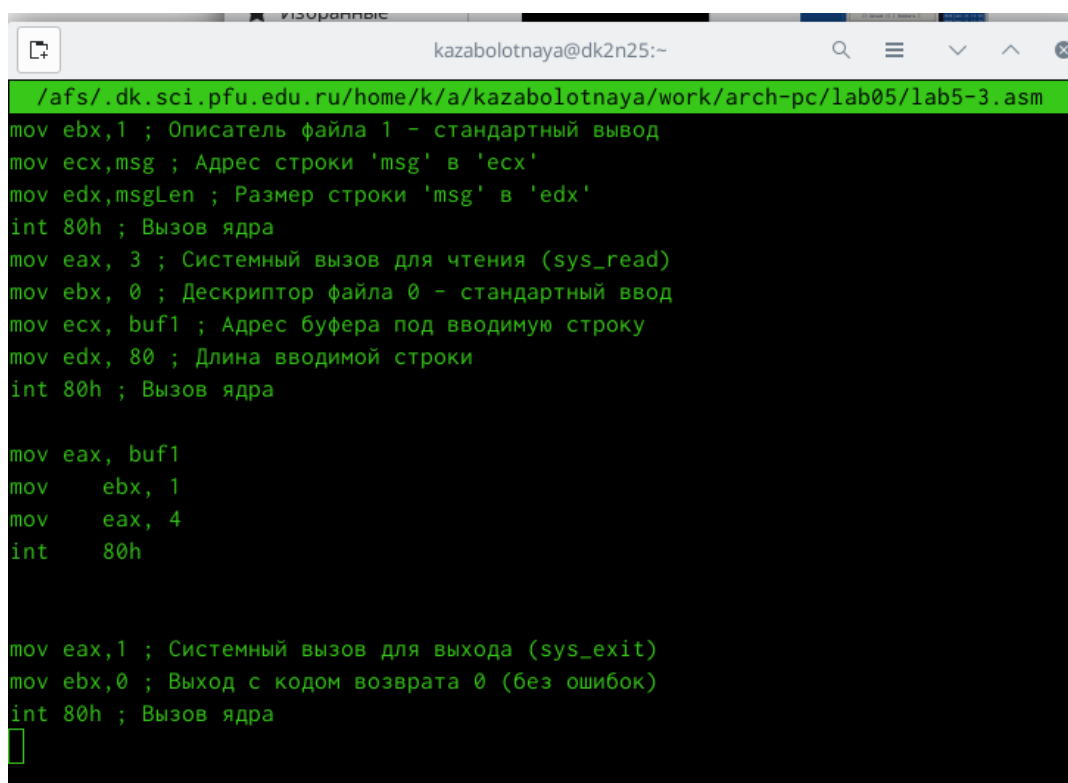


Рис. 4.12: 512.png

5 Самостоятельная часть лабораторной работы

13. Работаем в папке lab5-3.



```
/afs/.dk.sci.pfu.edu.ru/home/k/a/kazaboltnaya/work/arch-pc/lab05/lab5-3.asm
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 ;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, buf1
mov ebx, 1
mov eax, 4
int 80h

mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата 0 (без ошибок)
int 80h ; Вызов ядра
```

Рис. 5.1: 513.png

14. Получим исполняемый файл и проверим его работу. На приглашение введите строку вводим свою фамилию.

```
kazaboltnaya@dk2n25 ~ $ mc

kazaboltnaya@dk2n25 ~ $ cd work/arch-pc
kazaboltnaya@dk2n25 ~/work/arch-pc $ cd lab05
kazaboltnaya@dk2n25 ~/work/arch-pc/lab05 $ nasm -f elf lab5-3.asm
kazaboltnaya@dk2n25 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-3 lab5-3.o
kazaboltnaya@dk2n25 ~/work/arch-pc/lab05 $ ./lab5-3
Введите строку:
Заболотная Кристина Александровна
Заболотная Кристина Александровна
kazaboltnaya@dk2n25 ~/work/arch-pc/lab05 $
```

Рис. 5.2: 514.png

15. Копируем файл lab5-2.asm.

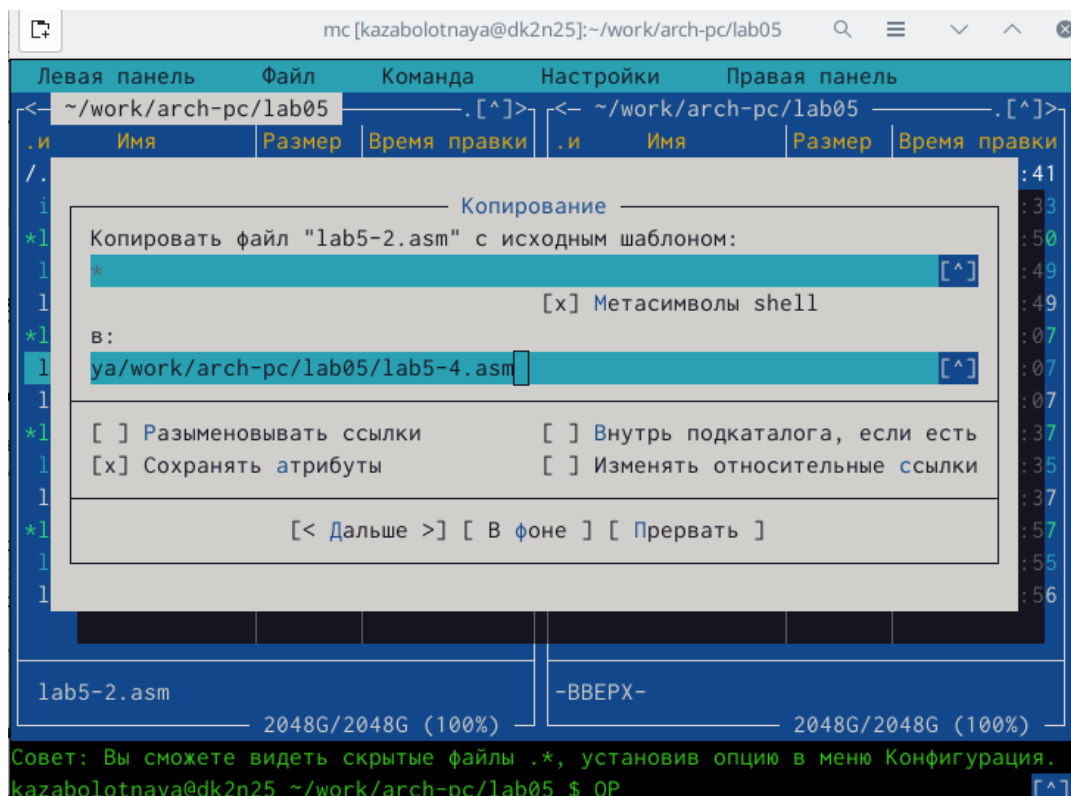


Рис. 5.3: 515.png

16. Копируем команды `mov eax, buf1; mov ecx, buf1; mov edx, 80;` для выполнения команды `call sprintLF`.

```

%include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку: ',0h ; сообщение
SECTION .bss ; Секция не иницированных 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, buf1
call sprintLF ; вызов подпрограммы печати сообщения
mov ecx, buf1 ; запись адреса переменной в 'EAX'
mov edx, 80 ; запись длины вводимого сообщения в 'EBX'

call quit ; вызов подпрограммы завершения

```

Рис. 5.4: 516.png

17. Оттранслируем текст программы lab5-2.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 введем ФИО

```

kazaboltnaya@dk2n25 ~/work/arch-pc/lab05 $ nasm -f elf lab5-4.asm
kazaboltnaya@dk2n25 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-4 lab5-4.o
kazaboltnaya@dk2n25 ~/work/arch-pc/lab05 $ ./lab5-4
Введите строку: Заболотная Кристина Александровна
Заболотная Кристина Александровна

```

Рис. 5.5: 517.png

6 Выводы

В ходе выполнения данной лабораторной работы были приобретены практические навыки работы в Midnight Commander. Были освоены инструкции языка ассемблера `mov` и `int`.

Список литературы