

# **Отчёт по лабораторной работе No5**

**Основы работы с Midnight Commander (mc)**

Четвергова Мария

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
3.1	Задание для самостоятельной работы . . . . .	12
<b>4</b>	<b>Выводы</b>	<b>15</b>

## Список иллюстраций

3.1	Создание папки lab05 . . . . .	8
3.2	Создание файла lab5-1.asm . . . . .	8
3.3	Текст программы из листинга 5.1 в файле . . . . .	9
3.4	Оттранслируем текст программы lab5-1.asm в объектный файл . .	10
3.5	Скопируйте файл в каталог с помощью функциональной клавиши	11
3.6	Создание копии файла lab5-1.asm с именем lab5-2.asm. . . . .	11
3.7	Создание исполняемого файла . . . . .	12
3.8	Создание копии файла lab5-1.asm . . . . .	13
3.9	Получение исполняемого файла и проверка его работы . . . . .	13
3.10	Исправьте текст программы с использованием подпрограмм из внешнего файла in_out.asm . . . . .	14
3.11	Создайте исполняемый файл и проверьте его работу . . . . .	14

## **Список таблиц**

# 1 Цель работы

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

## 2 Теоретическое введение

**##Структура программы на языке ассемблера NASM##**

Программа на языке ассемблера NASM, как правило, состоит из трёх секций: секция кода программы (SECTION .text), секция инициированных (известных во время компиляции) данных (SECTION .data) и секция неинициализированных данных (тех, под которые во время компиляции только отводится память, а значение присваивается в ходе выполнения программы) (SECTION .bss) Для объявления инициированных данных в секции .data используются директивы DB, DW, DD, DQ и DT, которые резервируют память и указывают, какие значения должны храниться в этой памяти

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.

**##Системные вызовы для обеспечения диалога с пользователем##**

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 любой 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.

После вызова инструкции `int 80h` выполняется системный вызов какой-либо функции ядра Linux. При этом происходит передача управления ядру операционной системы. Чтобы узнать, какую именно системную функцию нужно выполнить, ядро извлекает номер системного вызова из регистра `eax`. Поэтому перед вызовом прерывания необходимо поместить в этот регистр нужный номер. Кроме того, многим системным функциям требуется передавать какие-либо параметры. По принятым в ОС Linux правилам эти параметры помещаются в порядке следования в остальные регистры процессора: `ebx`, `ecx`, `edx`. Если системная функция должна вернуть значение, то она помещает его в регистр `eax`.

### 3 Выполнение лабораторной работы

1. Откройте Midnight Commander
2. Пользуясь клавишами **←** , **→** и **Enter** перейдите в каталог `~/work/arch-pc` созданный при выполнении лабораторной работы No4
3. С помощью функциональной клавиши **F7** создайте папку `lab05` и перейдите в созданный каталог.

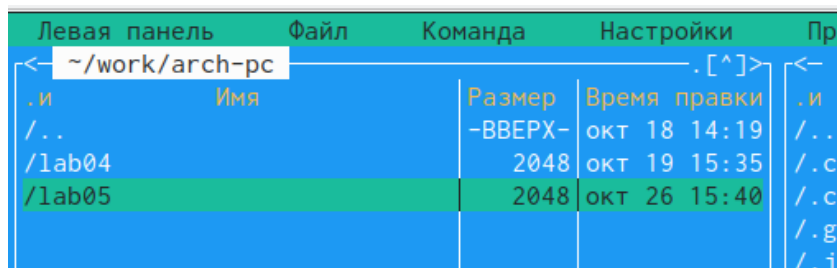


Рис. 3.1: Создание папку lab05

4. Пользуясь строкой ввода и командой `touch` создайте файл `lab5-1.asm`

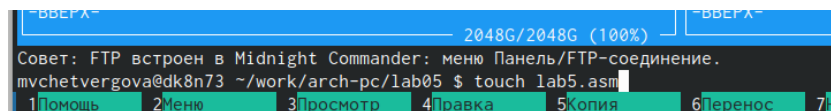


Рис. 3.2: Создание файла lab5-1.asm

5. С помощью функциональной клавиши **F4** откройте файл `lab5-1.asm` для редактирования во встроенном редакторе. Как правило в качестве встро-



енного редактора Midnight Commander используется редакторы nano или mcedit

6. Введите текст программы из листинга 5.1 (можно без комментариев), сохраните изменения и закройте файл.

```
/afs/.dk.sci.pfu.edu.ru/home/m/v/mv~gova/work/arch-pc/lab05/lab5-1.asm
;-----
; 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
52 Демидова А. В.
Архитектура ЭВМ
;----- 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. 3.3: Текст программы из листинга 5.1 в файле

7. С помощью функциональной клавиши F3 откройте файл lab5-1.asm для просмотра. Убедитесь, что файл содержит текст программы.
8. Оттранслируйте текст программы lab5-1.asm в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый

файл.

```
lab5-1.asm [----] 20 L: [ 1+36 37/ 37] *(2488/2488b) <EOF>
; 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
52 Демидова А. В.
Архитектура 386
;----- 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. 3.4: Оттранслируем текст программы lab5-1.asm в объектный файл

##5.3.1. Подключение внешнего файла in\_out.asm## Для упрощения написания программ часто встречающиеся одинаковые участки кода (такие как, например, вывод строки на экран или выход из программы) можно оформить в виде подпрограмм и сохранить в отдельные файлы, а во всех нужных местах поставить вызов нужной подпрограммы. Это позволяет сделать основную программу более удобной для написания и чтения.

9. Скачайте файл in\_out.asm со страницы курса в ТУИС.

10. Подключаемый файл `in_out.asm` должен лежать в том же каталоге, что и файл с программой, в которой он используется. В одной из панелей `mc` откройте каталог с файлом `lab5-1.asm`. В другой панели каталог со скаченным файлом `in_out.asm` (для перемещения между панелями используйте `Tab`). Скопируйте файл `in_out.asm` в каталог с файлом `lab5-1.asm` с помощью функциональной клавиши

```

mvchetvergova@dk6n62 ~/work/arch-pc/lab05 $ nasm -f elf lab5-1.asm
mvchetvergova@dk6n62 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-1 lab5-1.o
mvchetvergova@dk6n62 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:
mvchetvergova
mvchetvergova@dk6n62 ~/work/arch-pc/lab05 $ ./lab5-1
Введите строку:
Четвергова Мария

```

Рис. 3.5: Скопируйте файл в каталог с помощью функциональной клавиши

11. С помощью функциональной клавиши `F6` создайте копию файла `lab5-1.asm` с именем `lab5-2.asm`. Выделите файл `lab5-1.asm`, нажмите клавишу `F6`, введите имя файла `lab5-2.asm` и нажмите клавишу `Enter`.

Имя	Размер	Время правки	Имя	Размер	Время правки
..	-ВВЕРХ-	окт 26 15:28	..	-ВВЕРХ-	окт 26 15:28
in_out.asm	2042	окт 26 15:28	in_out.asm	2042	окт 26 15:28
lab5-1	8744	окт 26 16:04	lab5-1	8744	окт 26 16:04
lab5-1.asm	2042	окт 26 16:04	lab5-1.asm	2042	окт 26 16:04
lab5-1.asm.save	1	окт 26 15:32	lab5-1.asm.save	1	окт 26 15:32
lab5-1.o	752	окт 26 16:04	lab5-1.o	752	окт 26 16:04

Рис. 3.6: Создание копии файла `lab5-1.asm` с именем `lab5-2.asm`.

2. Исправьте текст программы в файле `lab5-2.asm` с использованием подпрограмм из внешнего файла `in_out.asm` (используйте подпрограммы `sprintLF`, `sread` и `quit`) в соответствии с листингом 5.2. Создайте исполняемый файл и проверьте его работу.

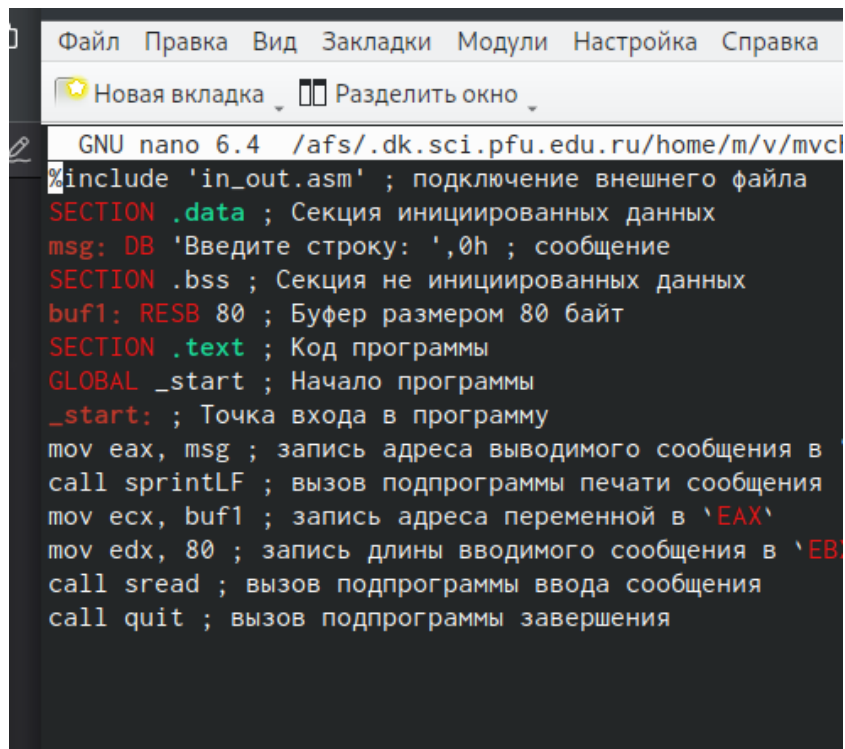
*lab5-1	8744	окт 26 16:04
lab5-1.asm.save	1	окт 26 15:32
lab5-1.o	752	окт 26 16:04
*lab5-2	9092	ноя 2 18:26
lab5-2.asm	1222	ноя 2 18:17
lab5-2.o	1312	ноя 2 18:24
lab5-3.asm	963	ноя 2 18:43

Рис. 3.7: Создание исполняемого файла

В файле lab5-2.asm замените подпрограмму sprintLF на sprint. Создайте исполняемый файл и проверьте его работу.

### 3.1 Задание для самостоятельной работы

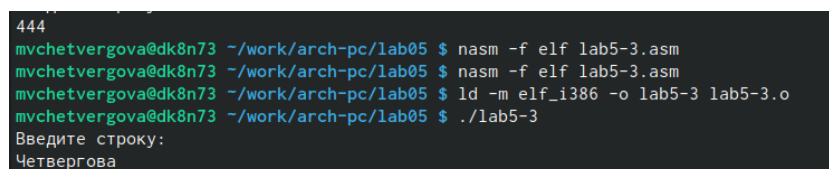
1. Создайте копию файла lab5-1.asm. Внесите изменения в программу (без использования внешнего файла in\_out.asm), так чтобы она работала по следующему алгоритму: • вывести приглашение типа “Введите строку:”; • ввести строку с клавиатуры;



```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/m/v/mvcl
%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
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
```

Рис. 3.8: Создание копии файла lab5-1.asm

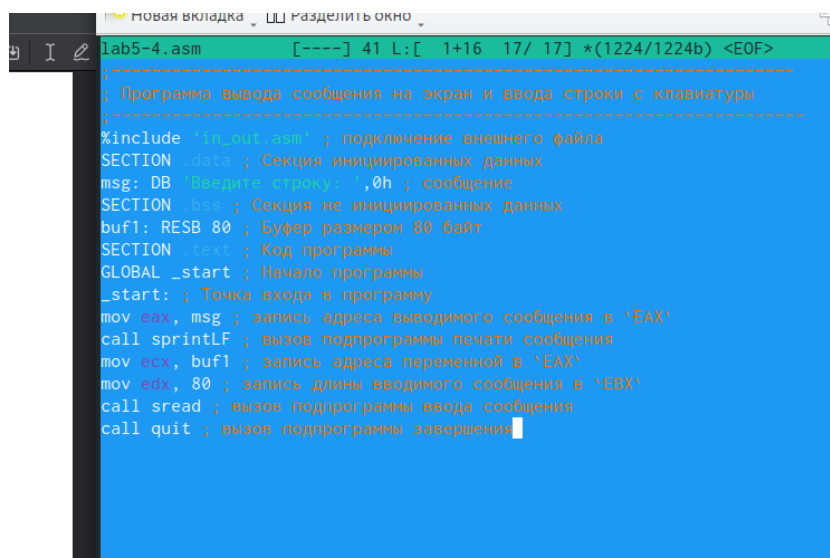
2. Получите исполняемый файл и проверьте его работу. На приглашение ввести строку введите свою фамилию.



```
444
mvchetvergova@dk8n73 ~/work/arch-pc/lab05 $ nasm -f elf lab5-3.asm
mvchetvergova@dk8n73 ~/work/arch-pc/lab05 $ nasm -f elf lab5-3.asm
mvchetvergova@dk8n73 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-3 lab5-3.o
mvchetvergova@dk8n73 ~/work/arch-pc/lab05 $ ./lab5-3
Введите строку:
Четвергова
```

Рис. 3.9: Получение исполняемого файла и проверка его работы

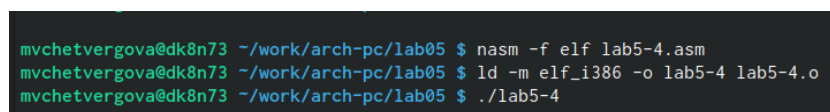
3. Создайте копию файла lab5-2.asm. Исправьте текст программы с использование подпрограмм из внешнего файла in\_out.asm, так чтобы она работала по следующему алгоритму:



```
lab5-4.asm [---] 41 L: [ 1+16 17/ 17] *(1224/1224b) <EOF>
; Программа вывода сообщения на экран и ввода строки с клавиатуры
#include "in_out.asm" ; подключение внешнего файла
SECTION .data ; Секция инициализированных данных
msg: DB "Введите строку: ",0h ; сообщение
SECTION .bss ; Секция не инициализированных 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 'ECX'
mov edx, 80 ; запись длины вводимого сообщения в 'EDX'
call sread ; вызов подпрограммы ввода сообщения
call quit ; вызов подпрограммы завершения
```

Рис. 3.10: Исправьте текст программы с использованием подпрограмм из внешнего файла in\_out.asm

#### 4. Создайте исполняемый файл и проверьте его работу



```
mvchetvergova@dk8n73 ~/work/arch-pc/lab05 $ nasm -f elf lab5-4.asm
mvchetvergova@dk8n73 ~/work/arch-pc/lab05 $ ld -m elf_i386 -o lab5-4 lab5-4.o
mvchetvergova@dk8n73 ~/work/arch-pc/lab05 $ ./lab5-4
```

Рис. 3.11: Создайте исполняемый файл и проверьте его работу

## 4 Выводы

Мы приобрели практические навыки работы в Midnight Commander, а также освоили инструкции языка ассемблера `mov` и `int`.