

Отчёт по лабораторной работе №5

Дисциплина: Архитектура компьютеров

Карпова Анастасия Александровна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
5	Выводы	15
	Список литературы	16

Список иллюстраций

4.1	Открытие mc	8
-----	-----------------------	---

1 Цель работы

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

2 Задание

1. Основы работы с тс
2. Структура программы на языке ассемблера NASM
3. Подключение внешнего файла
4. Выполнение заданий для самостоятельной работы

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. Программа на языке ассемблера NASM, как правило, состоит из трёх секций: секция кода программы (`SECTION .text`), секция иницированных (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 иницированных данных в секции `.data` используются директивы `DB`, `DW`, `DD`, `DQ` и `DT`, которые резервируют память и указывают, какие значения должны храниться в этой памяти:

`DB` (define byte) — определяет переменную размером в 1 байт; `DW` (define word) — определяет переменную размером в 2 байта (слово); `DD` (define double word) — определяет переменную размером в 4 байта (двойное слово); `DQ` (define quad word) — определяет переменную размером в 8 байт (учетверённое слово); `DT` (define ten bytes) — определяет переменную размером в 10 байт. 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. Инструкция языка ассемблера `mov` предназначена для дублирования данных источника в приёмнике. `mov dst,src` Здесь операнд `dst` — приёмник, а `src` — источник. В качестве операнда могут высту-

пать регистры (register), ячейки памяти (memory) и непосредственные значения (const). Инструкция языка ассемблера `int n` предназначена для вызова прерывания с указанным номером. `int n` Здесь `n` — номер прерывания, принадлежащий диапазону 0–255. При программировании в Linux с использованием вызовов ядра `sys_calls` `n=80h` (принято задавать в шестнадцатеричной системе счисления).

4 Выполнение лабораторной работы

Основы работы с mc

Открываю Midnight Commander, введя mc в терминал (рис. 4.1).

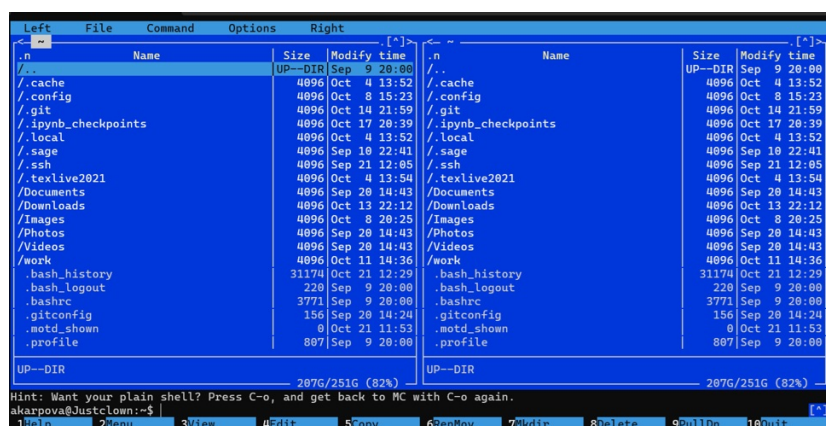
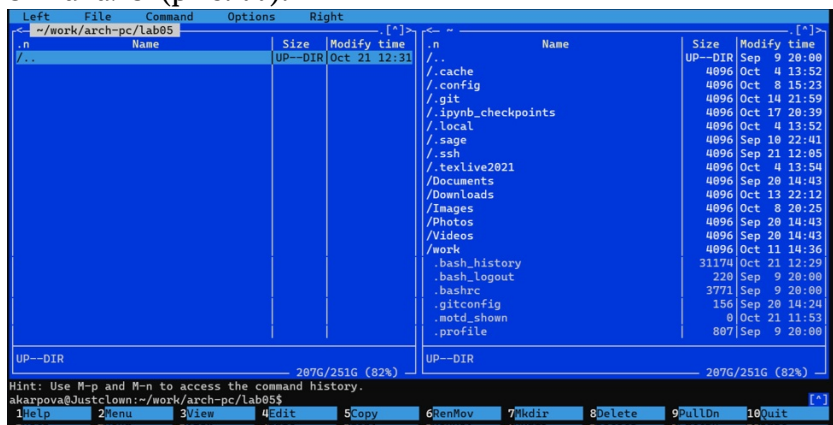
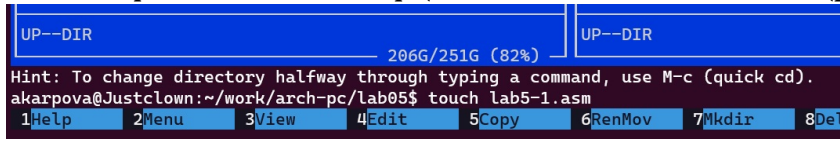


Рис. 4.1: Открытие mc

Перехожу в каталог ~/work/arch-pc и создаю в нем каталог lab05, перехожу в этот каталог(рис. ??).



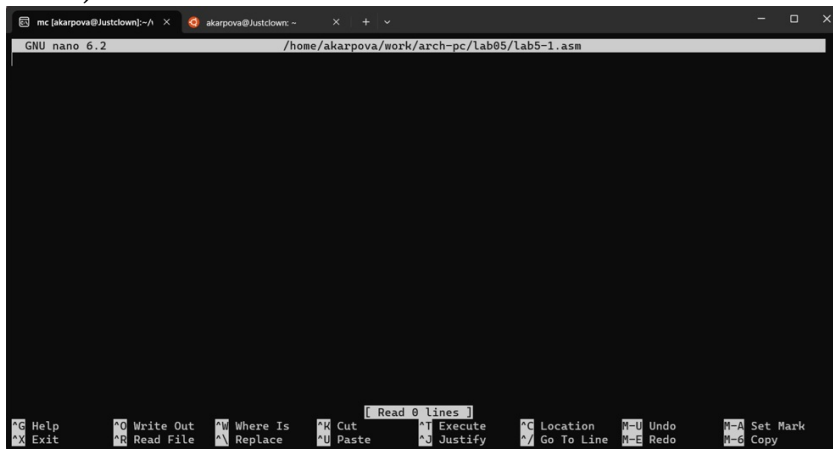
Создаю файл lab5-1.asm при помощи команды touch (рис. ??)



```
UP--DIR 206G/251G (82%) UP--DIR
Hint: To change directory halfway through typing a command, use M-c (quick cd).
akarpova@Justclown:~/work/arch-pc/lab05$ touch lab5-1.asm
1Help 2Menu 3View 4Edit 5Copy 6RenMov 7Mkdir 8Del
```

Структура программы на языке ассемблера NASM

При помощи клавиши f4 открываю файл lab5-1.asm для редактирования в nano (рис. ??)



```
mc [akarpova@Justclown] ~ / akarpova@Justclown ~
GNU nano 6.2 /home/akarpova/work/arch-pc/lab05/lab5-1.asm
[ Read 0 lines ]
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location ^U Undo ^A Set Mark
^X Exit ^R Read File ^N Replace ^U Paste ^J Justify ^_ Go To Line ^M Redo ^-G Copy
```

Ввожу в файл код программы для запроса строки у пользователя. Потом выхожу из файла при помощи Ctrl+X, сохранив изменения - Y, Enter. (рис. ??)



```
GNU nano 6.2 /home/akarpova/work/arch-pc/lab05/lab5-1.asm
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,4
mov ebx,1
mov ecx,msg
mov edx,msgLen
int 80h
mov eax,3
mov ebx,0
mov ecx,buf1
mov edx,80
int 80h
mov eax,1
mov ebx,0
int 80h
[ Read 21 lines ]
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location ^U Undo ^A Set Mark
^X Exit ^R Read File ^N Replace ^U Paste ^J Justify ^_ Go To Line ^M Redo ^-G Copy
```

Проверяю файл на содержание кода при помощи f3(для просмотра файла) (рис. ??)

```

/home/akarpova/work/arch-pc/lab05/lab5-1.asm 276/276 100%
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,4
mov ebx,1
mov ecx,msg
mov edx,msgLen
int 80h
mov eax,3
mov ebx,0
mov ecx,buf1
mov edx,80
int 80h
mov eax,1
mov ebx,0
int 80h
1 Help 2 InWrap 3 Quit 4 Hex 5 Goto 6 7 Search 8 Raw 9 Format 10 Quit

```

Транслирую текст программы в объектный файл командой `nasm -f elf lab5-1.asm` (создался объектный файл `lab5-1.o`). Затем выполняю компоновку объектного файла при помощи команды `ld -m elf_i386 -o lab5-1 lab5-1.o`. (рис. ??)

```

akarpova@Justclown:~/work/arch-pc/lab05$ nasm -f elf lab5-1.asm
akarpova@Justclown:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1 lab5-1.o

```

Запускаю исполняемый файл. (Программа выводит строку “Введите строку” - соответственно мы вводим своё ФИО и программа заканчивает свою работу, выведя наше ФИО (рис. ??)

```

akarpova@Justclown:~/work/arch-pc/lab05$ ./lab5-1
Введите строку:
Карпова Анастасия Александровна

```

Запуск исполняемого файла

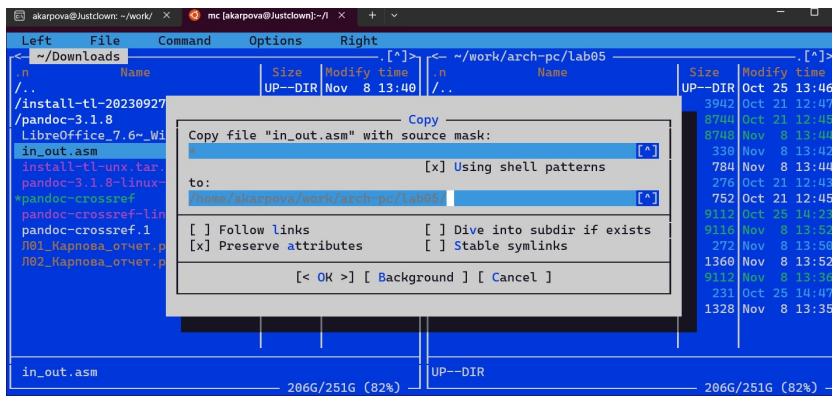
Скачиваю файл `in_out.asm` из ТУИСа. (рис. ??)

```

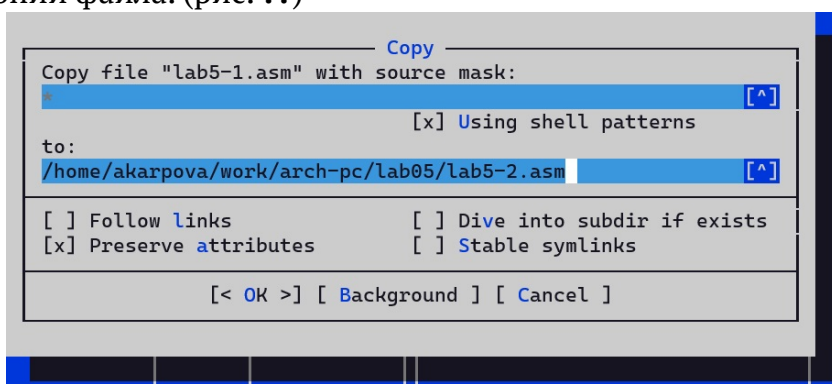
< ~/Downloads .[^>
.n      Name      Size  Modify time
/..     UP--DIR   Nov  8 13:40
/install-tl-20230927 4096 Sep 27 02:46
/pandoc-3.1.8       4096 Sep  9 20:46
LibreOffice_7.6~_Win_x86-64.msi 353368K Sep 22 18:07
in_out.asm ←       3942 Oct 21 12:47
install-tl-unx.tar.gz 5743135 Sep 28 12:17

```

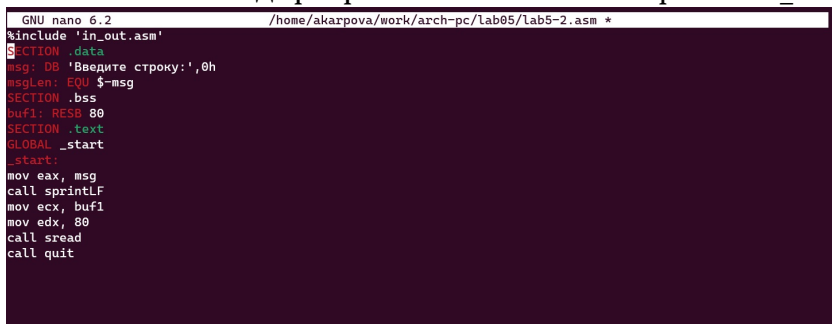
С помощью функциональной клавиши `f5` копирую файл `in_out.asm` из каталога `Downloads` в созданный каталог `lab05`. (рис. ??)



С помощью функциональной клавиши f5 копирую файл lab5-1 в тот же каталог, но с другим именем, для этого в появившемся окне mc прописываю имя для копии файла. (рис. ??)



Изменяю содержимое файла lab5-2.asm редакторе nano, чтобы в программе использовались подпрограммы из внешнего файла in_out.asm. (рис. ??)



Транслирую текст программы файла в объектный файл командой `nasm -f elf lab5-2.asm` (Создался объектный файл lab5-2.o). Выполняю компоновку объектного файла с помощью команды `ld -m elf_i386 -o lab5-2 lab5-2.o` (Создался исполняемый файл lab5-2). Запускаю исполняемый файл. (рис. ??)

```

akarpova@Justclown:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
akarpova@Justclown:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
akarpova@Justclown:~/work/arch-pc/lab05$ ./lab5-2
Введите строку:
Карпова Анастасия Александровна

```

Открываю файл lab5-2.asm в редакторе в nano функциональной клавишей f4. Изменяю в нем подпрограмму sprintLF на sprint. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий. (рис. ??)

```

GNU nano 6.2 /home/akarpova/work/arch-pc/lab05/lab5-2.asm *
#include "in_out.asm"
SECTION .data
msg: DB 'Введите строку:',0h
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, buf1
mov edx, 80
call read
call quit

```

Снова транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл. (рис. ??)

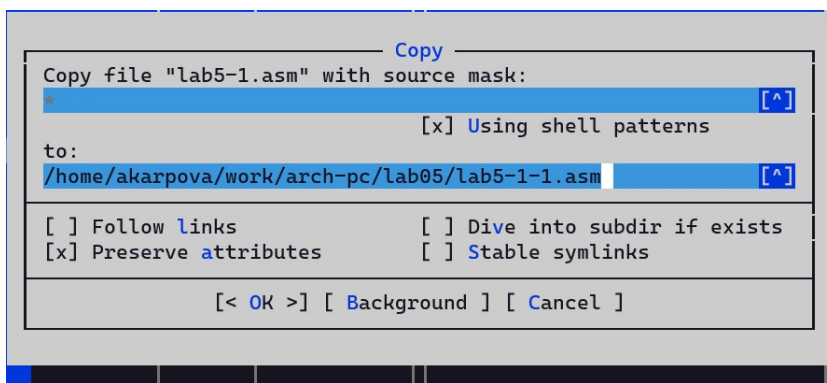
```

akarpova@Justclown:~/work/arch-pc/lab05$ nasm -f elf lab5-2.asm
akarpova@Justclown:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-2 lab5-2.o
akarpova@Justclown:~/work/arch-pc/lab05$ ./lab5-2-2
Введите строку:Карпова Анастасия Александровна

```

Выполнение заданий для самостоятельной работы

1. Создаю копию файла lab5-1.asm с именем lab5-1-1.asm с помощью клавиши f5. (рис. ??)



С помощью клавиши f4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку. (рис. ??)

```

GNU nano 6.2 /home/akarpova/work/arch-pc/lab05/lab5-1-1.asm
SECTION .data
msg: DB 'Введите строку:',10
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax,4
mov ebx,1
mov ecx,msg
mov edx,msgLen
int 80h
mov eax,3
mov ebx,0
mov ecx,buf1
mov edx,80
int 80h
mov eax,4
mov ebx,1
mov ecx,buf1
mov edx,buf1

```

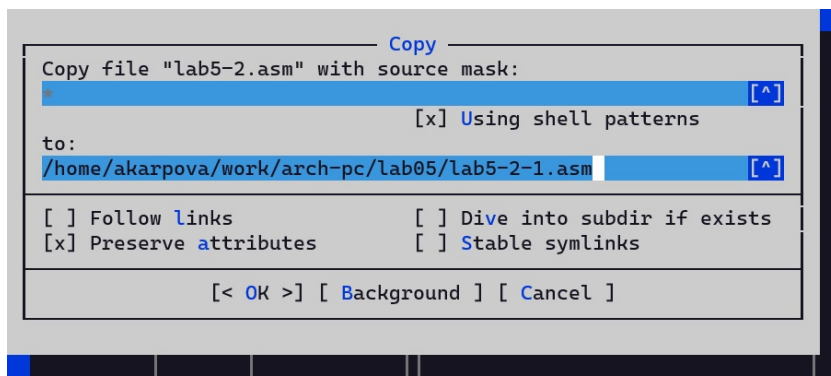
2. Создаю объектный файл lab5-1-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-1-1, запускаю полученный исполняемый файл. Программа запрашивает ввод, ввожу свои ФИО, и программа соответственно выводит введенные мною данные (рис. ??)

```

akarpova@Justclown:~/work/arch-pc/lab05$ nasm -f elf lab5-1-1.asm
akarpova@Justclown:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-1-1 lab5-1-1.o
akarpova@Justclown:~/work/arch-pc/lab05$ ./lab5-1-1
Введите строку:
Карпова Анастасия Александровна

```

3. Создаю копию файла lab5-2.asm с именем lab5-2-1.asm с помощью клавиши f5. (рис. ??)



С помощью клавиши f4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку. (рис. ??)

```

GNU nano 6.2 /home/akarpova/work/arch-pc/lab05/lab5-2-1.asm
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите строку:',0h
msgLen: EQU $-msg
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, buf1
mov edx, 80
call sread
mov eax, 4
mov ebx, 1
mov ecx, buf1
int 80h
call quit

```

4. Создаю объектный файл lab5-2-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab5-2-1, запускаю полученный исполняемый файл. Программа запрашивает ввод без переноса на новую строку, ввожу свои ФИО, далее программа выводит введенные мною данные. (рис. ??)

```

akarpova@Justclown:~/work/arch-pc/lab05$ nasm -f elf lab5-2-1.asm
akarpova@Justclown:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab5-2-1 lab5-2-1.o
akarpova@Justclown:~/work/arch-pc/lab05$ ./lab5-2-1
Введите строку:Карпова Анастасия Александровна
Карпова Анастасия Александровна

```

5 Выводы

В ходе выполнения лабораторной работы я приобрела практические навыки работы в Midnight Commander, а также освоила инструкции языка ассемблера `mov` и `int`.

Список литературы