РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ Факультет физико-математических и естественных наук Кафедра компьютерных и информационных наук

ОТЧЕТ

по лабораторной работе № 6

дисциплина: Архитектура компьютеров и операционные системы

Студент: Дьяконова Софья Номер студенческого билета: 1132220829 Группа: НКАбд-01-22

МОСКВА 2022 г

Цель работы:

Целью данной лабораторной работы является приобретение практических навыков работы в Midnight Commander, освоение инструкций языка ассемблера mov и int. Изучить идеологию и применение средств контроля версий. Приобрести практические навыки по работе с системой git.

Задание:

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

Теоретическое введение:

Midnight Commander — программа, позволяющая просматривать структуру каталогов и выполнять основные операции по управлению файловой системой, mc является файловым менеджером. Midnight Commander позволяет сделать работу с файлами удобной и наглядной. Программа на языке ассемблера NASM, состоит из 3 секций: секция кода программы (SECTION .text), секция инициированных данных (SECTION .data) и секция неинициализированных данных (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: - DB (define byte) — определяет переменную размером в 1 байт; - DW (define word) — определяет переменную размеров в 2 байта; DD (define double word) — определяет переменную размером в 4 байта (двойное слово); - DQ (define quad word) — определяет переменную размером в 8 байт (учетве- рённое слово); - DT (define ten bytes) — определяет переменную размером в 10 байт.

Директивы используются для объявления простых переменных и для

объявления массивов. Для определения строк используется директива 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 выступать регистры (register), ячейки памяти (memory) и непосредственные значения (const). Инструкция языка ассемблера intпредназначена для вызова прерывания с указанным номером.

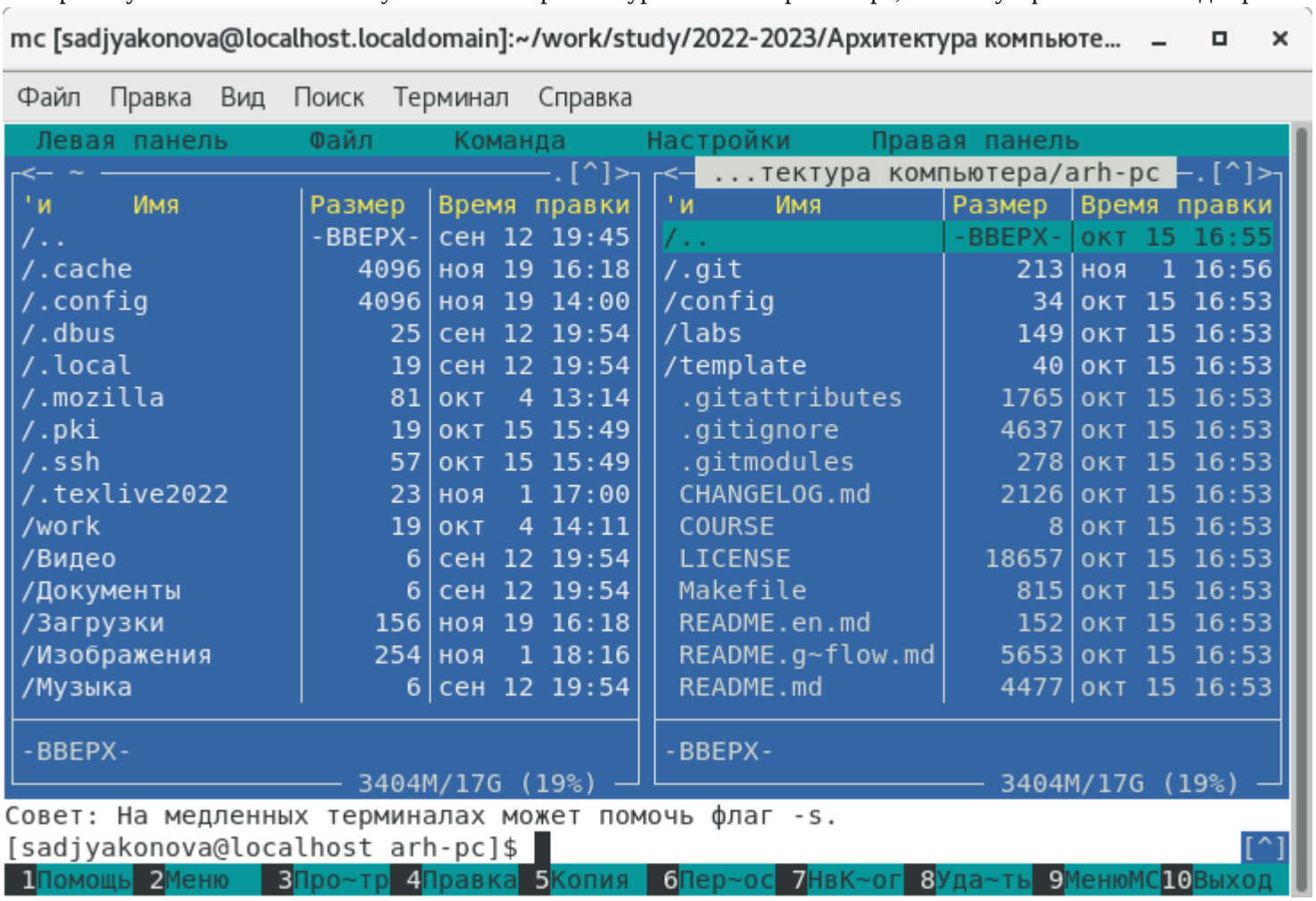
int n

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).

Ход работы:

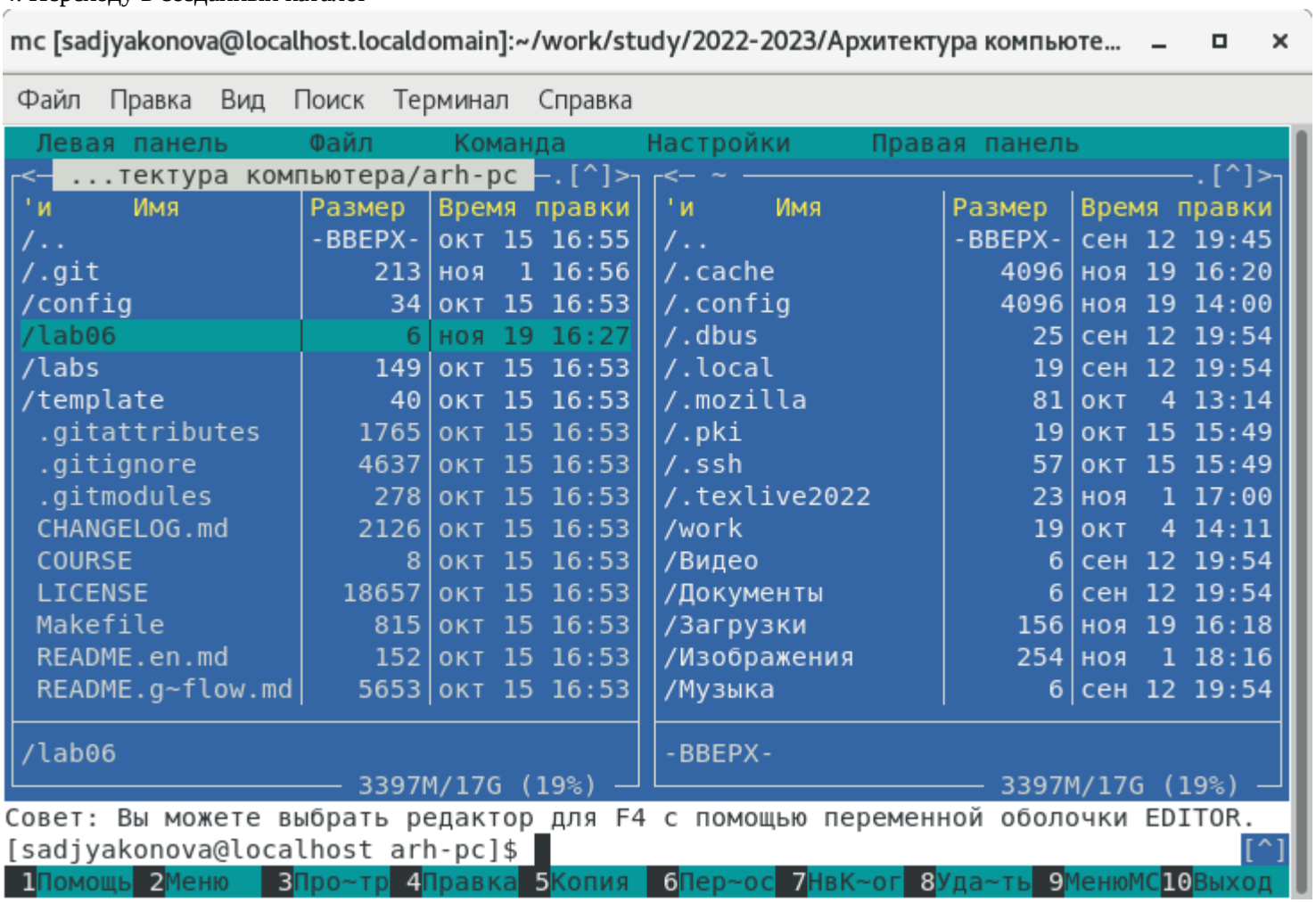
1. Открываю Midnight Commander, введя в терминал mc.

2. Перехожу в каталог ~/work/study/2022-2023/Архитектура Компьютера/arch-pc, используя файловый менеджер mc.

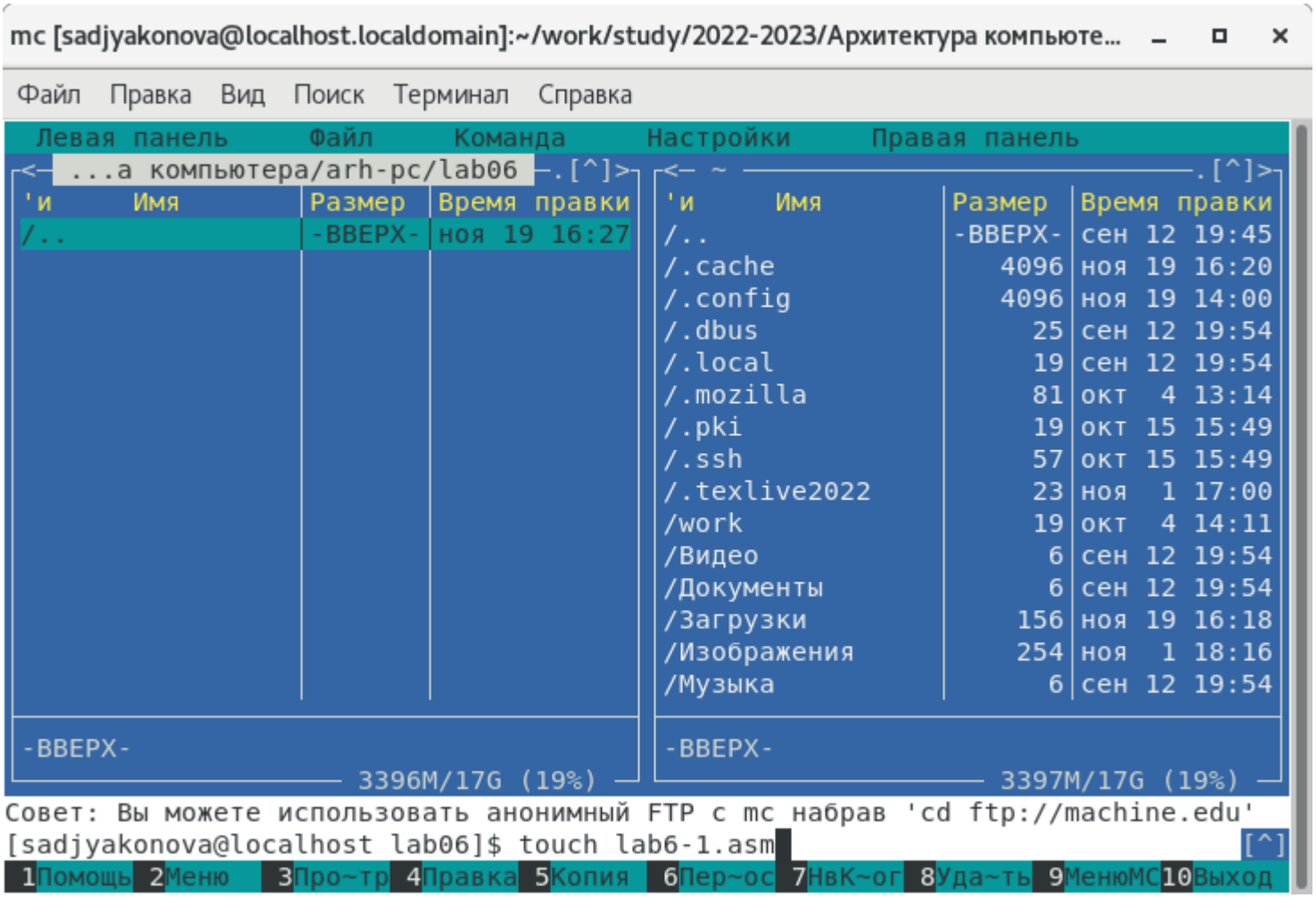


3. С помощью функциональной клавиши F7 создаю каталог lab06.

4. Переходу в созданный каталог



5. В строке ввода прописываю команду touch lab6-1.asm, чтобы создать файл, в котором буду работать



6. С помощью функциональной клавиши F4 открываю созданный файл для редактирования в редакторе nano 7. Ввожу в файл код программы для запроса строки у пользователя. Далее выхожу из файла (Ctrl+X), сохраняя изменения (Y, Enter).

mc [sadjyakonova@localhost.localdomain]:~/work/study/2022-2023/Архитектура компьюте...

Файл Правка Вид Поиск Терминал Справка

```
lab6-1.asm        [-M--]  0 L:[  1+ 0   1/ 37] *(0   /2435b) 0059 0x03B  [*][X]
;-----------------------------------------------------------------------
; 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)
 1Помощь 2Сох~ть 3Блок  4Замена 5Копия  6Пер~ть 7Поиск  8Уда~ть 9МенюМС10Выход
```

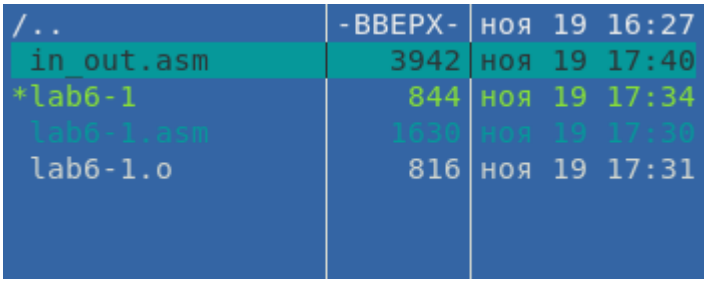
8. С помощью функциональной клавиши F3 открываю файл для просмотра, чтобы проверить, содержит ли файл текст программы
9. Транслирую текст программы файла в объектный файл командой nasm

-f elf lab6-1.asm. Создался объектный файл lab6-1.o. Выполняю компоновку объектного файла с помощью команды ld -m elf_i386 -o lab6-1 lab6-1.o. Создался исполняемый файл lab6-1.

10. Запускаю исполняемый файл. Ввожу свои ФИО

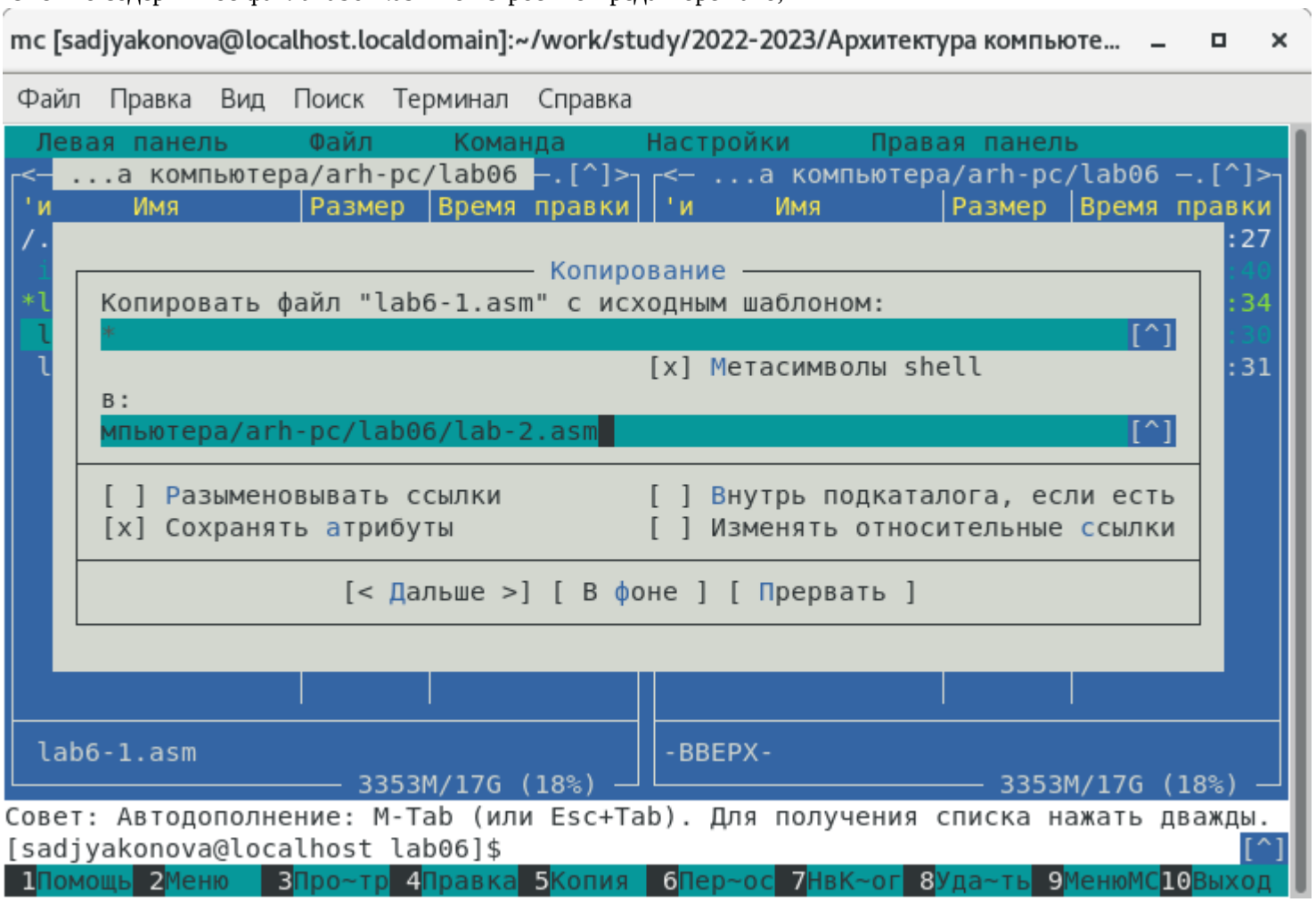
```
[sadjyakonova@localhost lab06]$ ./lab6-1
Enter string:
Dyakonova Sophya
[sadjyakonova@localhost lab06]$
```

11. Скачиваю файл in_out.asm со страницы курса в ТУИС. Он сохранился в каталог “Загрузки”.
12. С помощью функциональной клавиши F5 копирую файл in_out.asm из каталога Загрузки в созданный каталог lab06.



13. С помощью F5 копирую файл lab6-1 в тот же каталог, но с другим именем, для этого в появившемся окне mc прописываю имя для копии файла.

14.Изменяю содержимое файла lab6-2.asm во встроенном редакторе nano,



15.Транслирую текст программы файла в объектный файл командой nasm -f elf lab6-2.asm. Создался объектный файл lab6-2.o. Выполняю компоновку объектного файла с помощью команды ld -m elf_i386 -o lab6-2 lab6-2.o Создался исполняемый файл lab6-2. Запускаю исполняемый файл.

16. Открываю файл lab6-2.asm для редактирования в nano функциональной клавишей F4. Изменяю в нем подпрограмму sprintLF на sprint. Сохраняю изменения и открываю файл для просмотра, чтобы проверить сохранение действий.

```
[sadjyakonova@localhost lab06]$ ./lab6-1
Enter string:
Dyakonova Sophya
[sadjyakonova@localhost lab06]$
```

17. Транслирую файл, выполняю компоновку созданного объектного файла, запускаю новый исполняемый файл.Разница между первым исполняемым файлом и вторым в том, что запуск первого запрашивает ввод с новой строки, а программа, которая исполняется при запуске второго, запрашивает ввод без переноса на новую строку, потому что в этом заключается различие между подпрограммами sprintLF и sprint.

Задания для самостоятельной работы:

1. Создаю копию файла lab6-1.asm с именем lab6-1-1.asm с помощью F5. Открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку.

Код подпрограммы

SECTION .data *; Секция инициированных данных*

msg: DB 'Введите строку:',10

msgLen: EQU $-msg *; Длина переменной 'msg'* **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,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*

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*

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,buf1 *; Размер строки buf1*

int 80h *; Вызов ядра*

mov eax,1 *; Системный вызов для выхода (sys_exit)* **mov ebx**,0 *; Выход с кодом возврата 0 (без ошибок)*

int 80h *; Вызов ядра*

2. Создаю объектный файл lab6-1-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab6-1-1, запускаю полученный исполняемый файл. Ввожу свои ФИО, далее программа выводит введенные мною данные.

```
[sadjyakonova@localhost lab06]$ nasm -f elf lab6-1-1.asm
[sadjyakonova@localhost lab06]$ ld -m elf_i386 -o lab6-1-1 lab6-1-1.o
[sadjyakonova@localhost lab06]$ ./lab6-1-1
Введите строку:
Дьяконова Софья
Дьяконова Софья
```

3. Создаю копию файла lab6-2.asm с именем lab6-2-1.asm с помощью функциональной клавиши F5. С помощью F4 открываю созданный файл для редактирования. Изменяю программу так, чтобы кроме вывода приглашения и запроса ввода, она выводила вводимую пользователем строку.

Код программы

%include 'in_out.asm'

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*

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*

call quit *; вызов подпрограммы завершения*

```
[sadjyakonova@localhost lab06]$ nasm -f elf lab6-2-1.asm
[sadjyakonova@localhost lab06]$ ld -m elf_i386 -o lab6-2-1 lab6-2-1.o
[sadjyakonova@localhost lab06]$ ./lab6-2-1
Введите строку: Дьяконова Софья
Дьяконова Софья
[sadjyakonova@localhost lab06]$
```

4. Создаю объектный файл lab6-2-1.o, отдаю его на обработку компоновщику, получаю исполняемый файл lab6-2-1, запускаю полученный исполняемый файл. Программа запрашивает ввод без переноса на новую строку, ввожу свои ФИО, далее программа выводит введенные мною данные.

Вывод: В ходе лабораторной работы лабораторной работы я приобрела практические навыки работы в Midnight Commander, а также освоила

инструкции языка ассемблера mov и int.

Список литературы:

\1. Лабораторная работа No6. Основы работы cMidnight Commander (mc). Структура программы на языке ассемблера NASM. Системные вызовы в ОС GNU Linux. 10