РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4.

дисциплина: *Архитектура компьютеров*

Студент: Подхалюзина Виолетта Михайловна

Группа: НКАбд-04-24

МОСКВА

2024 г.

Оглавление

1 Цель работы

Целью данной лабораторной работы является приобретение практических навыков работы в Midnight Commander. Освоение инструкций языка ассемблера mov и int.

2 Введение

Midnight Commander — 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 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:

- DB (define byte) — определяет переменную размером в 1 байт;
- DW (define word) — определяет переменную размеров 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

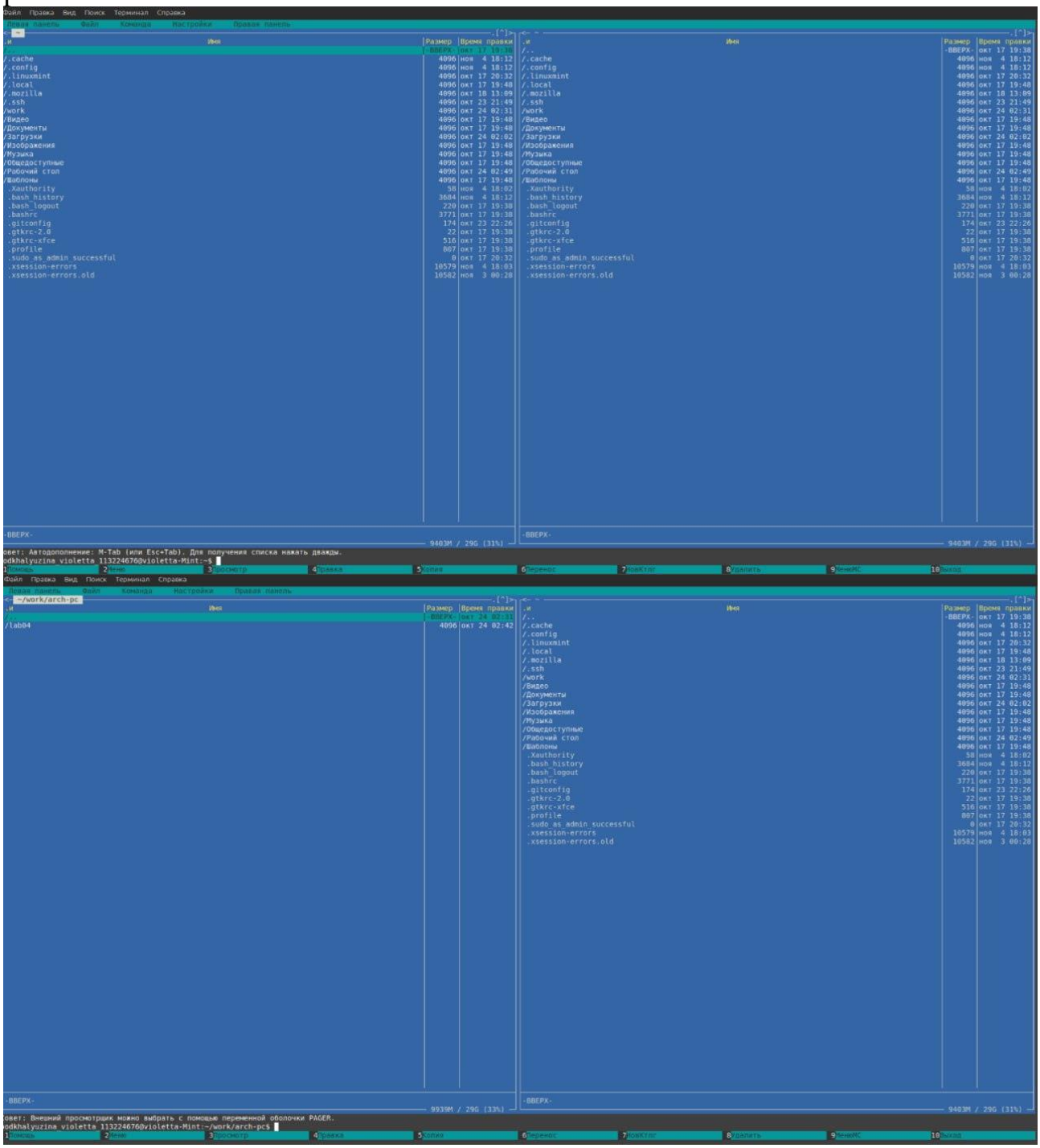
директиву DB в связи с особенностями хранения данных в оперативной памяти.

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.

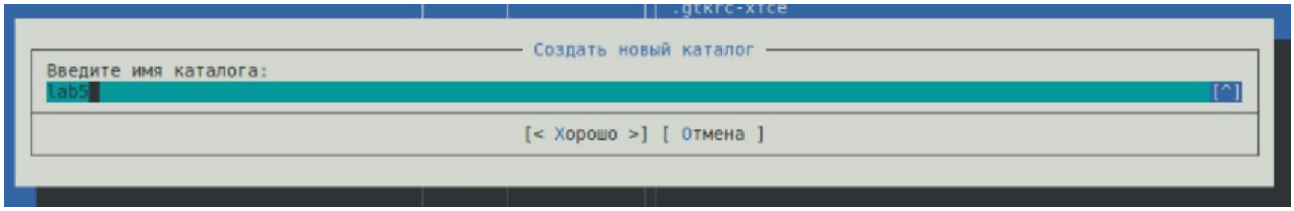
3 Выполнение лабораторной работы

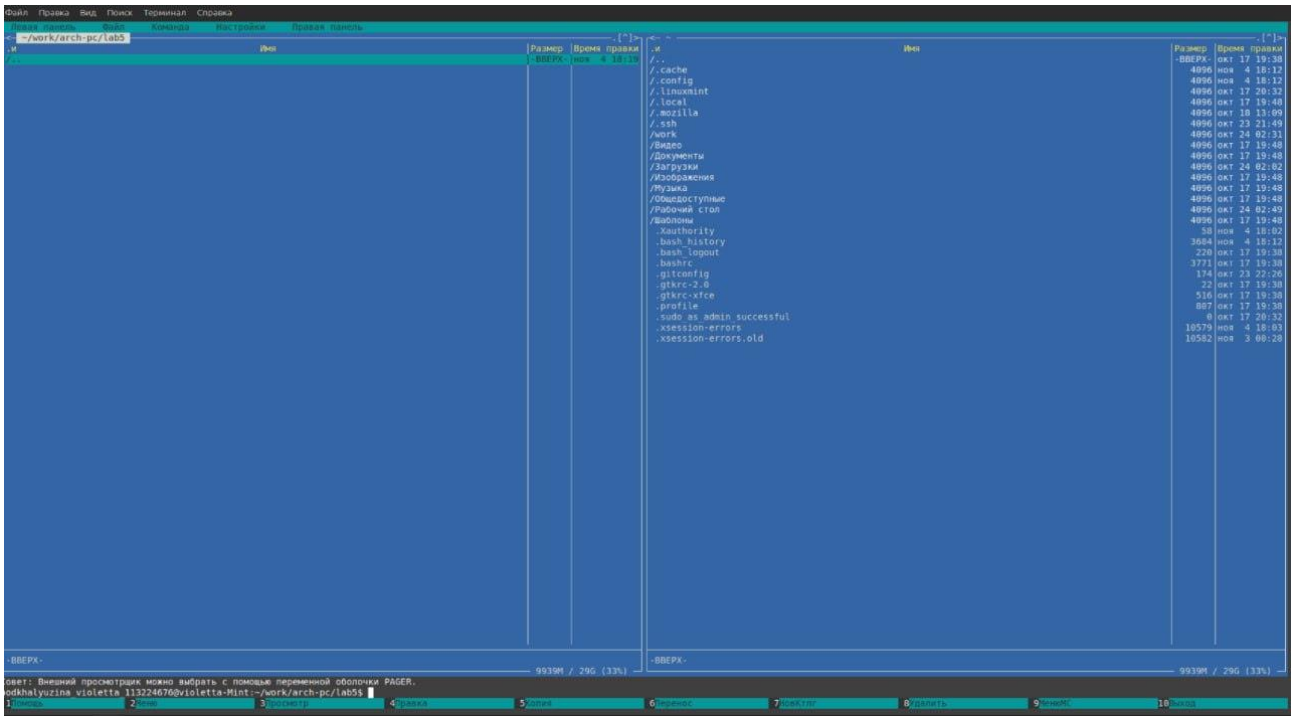
3.1 Начало работы

Сначала я открыла Midnight Commander, пользуясь клавишами ↑ , ↓ и Enter перешла в каталог ~/work/arch-pc созданный при выполнении лабораторной работы №4.

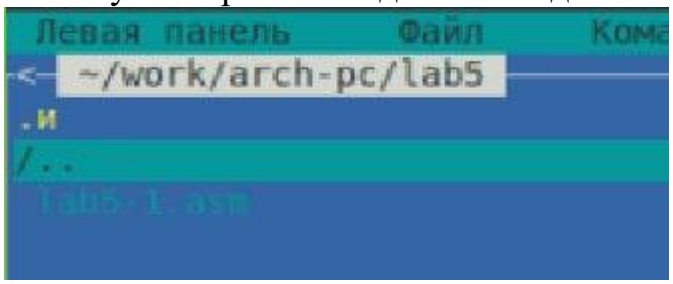


С помощью функциональной клавиши F7 создала папку lab05 и перешла в созданный каталог.



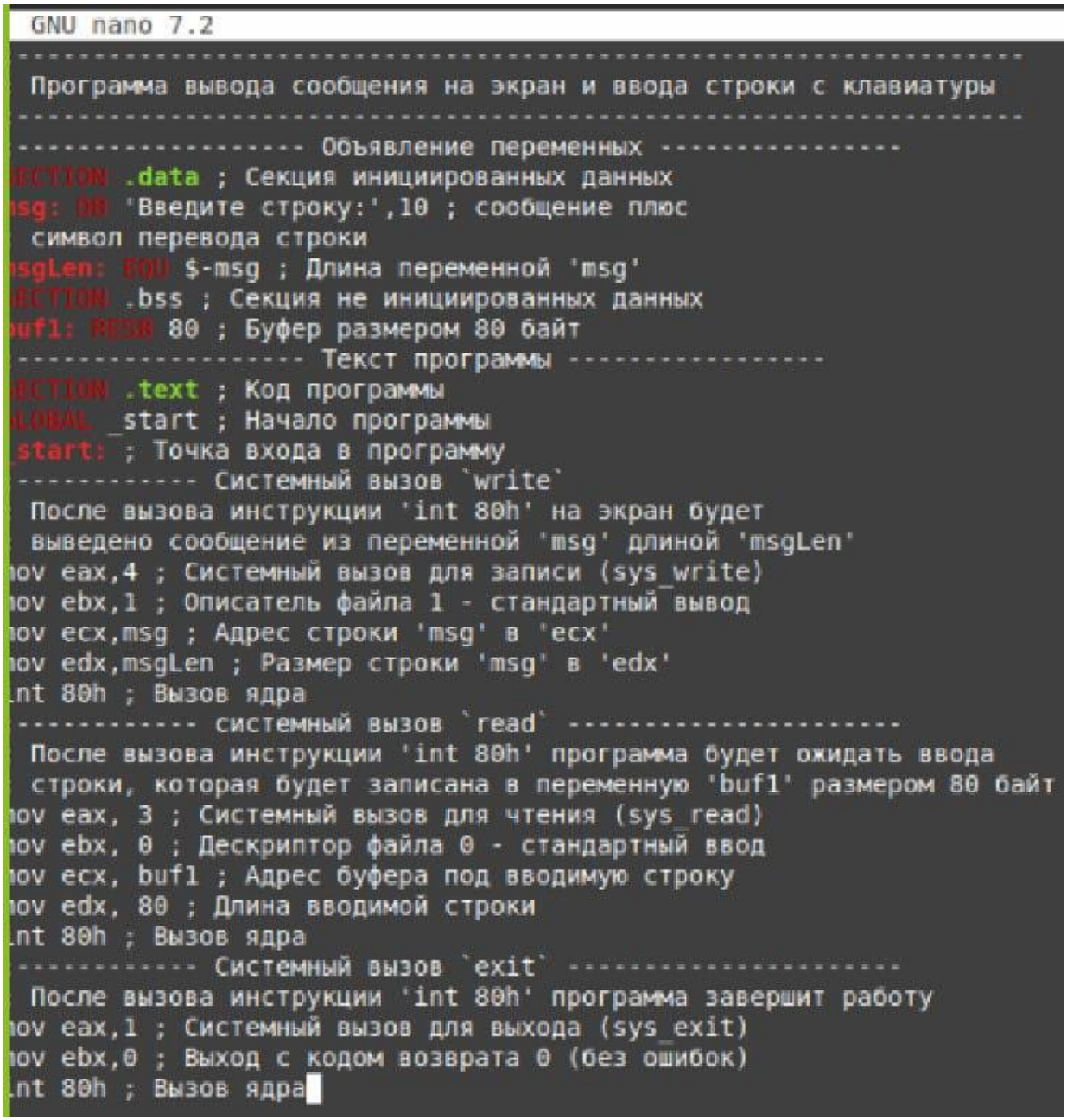


Пользуясь строкой ввода и командой touch создала файл lab5-1.asm.



3.2 Программа вывода сообщения на экран и ввода строки с клавиатуры

С помощью функциональной клавиши F4 я открыла файл lab5-1.asm для редактирования во встроенном редакторе, после ввела текст и сохранила изменения в файле. С помощью функциональной клавиши F3 я открыла файл lab5-1.asm для просмотра и убедилась, что файл содержит текст программы.


GNU nano 7.2
; 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

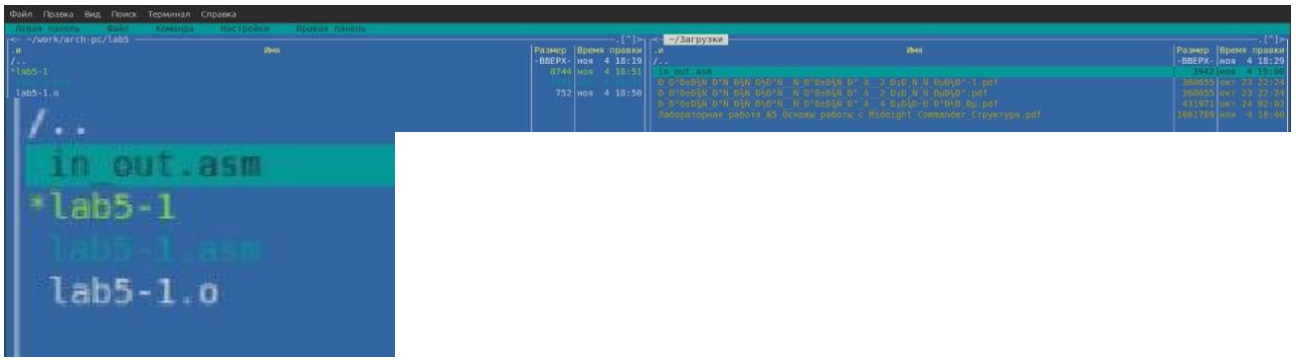
```
Файл  Правка  Вид  Поиск  Терминал  Справка
/home/podkhalyuzina_violetta_1132246676/work/arch-pc/lab5/lab5-1.asm
;----------------------------------------------------------------
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;----------------------------------------------------------------
;------------------- 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
```

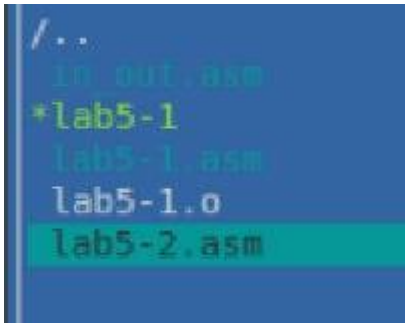
Я оттранслировала текст программы lab5-1.asm в объектный файл и выполнила компоновку объектного файла, запустив получившийся исполняемый файл. Программа выводит строку 'Введите строку:' и я могу ввести с клавиатуры своё имя.

```
podkhalyuzina_violetta_1132246676@violetta-Mint:~/work/arch-pc/lab5$ ./lab5-1
Введите строку:
Подхалюина Виолетта Михайловна
```

3.3 Подключение внешнего файла in_out.asm

Я скопировала файл in_out.asm в каталог с файлом lab5-1.asm и создала копию файла lab5-1.asm с именем lab5-2.asm. После выделила файл lab5-1.asm и ввела имя файла lab5-2.asm.






```
Файл  Правка  Вид  Поиск  Терминал  Справка
  GNU nano 7.2
;-----------------------------------------------------------------------
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----------------------------------------------------------------------
%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 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

podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$ nasm -f elf lab5-2.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$ ld -m elf_i386 -o lab5-2 lab5-2.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$ ls
in_out.asm  lab5-1  lab5-1.asm  lab5-1.o  lab5-2  lab5-2.asm  lab5-2.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$ ./lab5-2
Введите строку:
Подхалюзина Виолетта Михайловна
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$
```

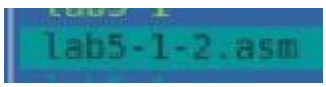
```
Файл  Правка  Вид  Поиск  Терминал  Справка
  GNU nano 7.2
;-----------------------------------------------------------------------
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----------------------------------------------------------------------
%include 'in_out.asm' ; подключение 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

podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$ nasm -f elf lab5-2.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$ ld -m elf_i386 -o lab5-2 lab5-2-2.o
ld: невозможно найти lab5-2-2.o: Нет такого файла или каталога
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$ ld -m elf_i386 -o lab5-2 lab5-2.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$ ls
in_out.asm  lab5-1  lab5-1.asm  lab5-1.o  lab5-2  lab5-2.asm  lab5-2.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$ ./lab5-2
Введите строку: Подхалюзина Виолетта Михайловна
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$
```

В файле lab5-2.asm я заменила подпрограмму, после создала исполняемый файл и проверила его работу. Разница в том, что в первом случае есть перенос на следующую строку, а во втором варианте такого нет.

3.4 Самостоятельная работа

Я создала копию файла lab5-1.asm и внесла изменения в программу (без использования внешнего файла in_out.asm), так чтобы она работала по следующему алгоритму:

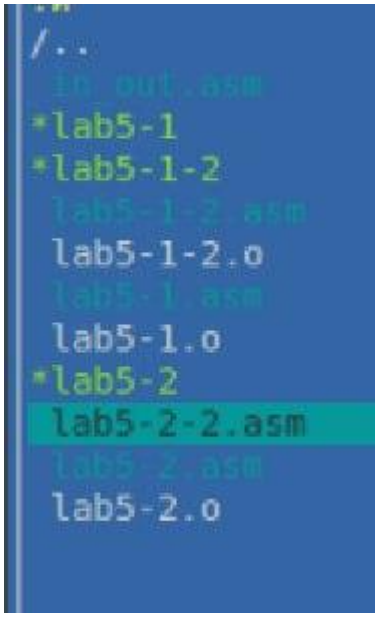
- вывести приглашение типа “Введите строку:”;
- ввести строку с клавиатуры;
- вывести введённую строку на экран.



```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$ nasm -f elf lab5-1-2.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$ ld -m elf_i386 -o lab5-1-2 lab5-1-2.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$ ./lab5-1-2
Введите строку:
Подхалюзина
Подхалюзина
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$
```

Создала копию файла lab5-2.asm и исправила 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 введённую строку на экран.




```
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$ nasm -f elf lab5-2-2.asm
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$ ld -m elf_i386 -o lab5-2-2 lab5-2-2.o
podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$ ./lab5-2-2
Введите строку: Подхалюзина
Подхалюзина

podkhalyuzina_violetta_113224676@violetta-Mint:~/work/arch-pc/lab5$
```

Создала исполняемый файл и проверила его работу.

```
;-----------------------------------------------------------------
; 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
 buf1:
 ; Секция не инициированных данных
 RESB 80 ; Буфер размером 80 байт
 ;-------------------
 SECTION .text
 GLOBAL _start
 _start:
 ; Код программы
 ; Начало программы---------------
; 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
 mov ebx,1
 mov ecx,msg
 mov edx,msgLen
 int 80h
 ; Системный вызов для записи (sys_write)
 ; Описатель файла 1- стандартный вывод
 ; Адрес строки 'msg' в 'ecx'
 ; Размер строки 'msg' в 'edx'
 ; 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
 mov ebx, 0
 mov ecx, buf1
 mov edx, 80
 int 80h
 ; Системный вызов для чтения (sys_read)
 ; Дескриптор файла 0- стандартный ввод
 ; Адрес буфера под вводимую строку
 ; Длина вводимой строки
 ; Вызов ядра
; Вывод введёной пользователем строки
mov     eax, 4
mov     edi, 1
mov     esi, buf1
mov     edx, 80
int 80h

 ;------------ Системный вызов exit---------------------
; После вызова инструкции 'int 80h' программа завершит работу
 mov eax,1
 mov ebx,0
 int 80h
 ; Выход с кодом возврата 0 (без ошибок)
 ; Вызов ядра
```

```
;-----------------------------------------------------------------
; Программа вывода сообщения на экран и ввода строки с клавиатуры
;-----------------------------------------------------------------
%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`
mov eax, buf1
mov edx, 80
call sread ; вызов подпрограммы ввода сообщени
mov eax, buf1
call sprintLF
call quit ; вызов подпрограммы завершения
```

4. Контрольные вопросы для самопроверки

1. Midnight Commander (mc) - это текстовый файловый менеджер для UNIX-подобных систем, работающий в консоли. Он предназначен для упрощения работы с файловой системой, предоставляя пользователю двухпанельный интерфейс для навигации, копирования, перемещения, удаления и редактирования файлов.

2. В Midnight Commander (mc) можно выполнять следующие операции с файлами:
- Копирование файлов: через команду cp в bash или через сочетание клавиш F5 в mc.
- Перемещение файлов: с помощью команды mv в bash или через F6 в mc.
- Удаление файлов: командой rm в bash или F8 в mc.
- Переименование файлов: командой mv в bash или через тот же F6 в mc.
- Просмотр файлов: через cat или less в bash и через F3 в mc.

3. Программа на языке ассемблера NASM обычно состоит из трех основных секций:
- Секция section .data для инициализированных данных.
- Секция section .bss для неинициализированных данных.
- Секция section .text для кода программы, где располагается основная логика, начиная с метки global _start для точки входа.

4. Секции .bss и .data в языке ассемблера NASM используются для следующих типов данных:
- .data - для инициализированных данных, то есть тех, которые имеют начальные значения.
- .bss - для неинициализированных данных, где просто резервируется место под переменные, но значения им не присваиваются до выполнения программы.

5. Компоненты db, dw, dd, dq, dt в NASM используются для определения переменных различной длины:

- db (define byte) - задает байт (8 бит).
- dw (define word) - задает слово (16 бит).
- dd (define double word) - задает двойное слово (32 бита).
- dq (define quad word) - задает четверное слово (64 бита).
- dt (define ten bytes) - задает 10 байт (80 бит), что иногда используется для хранения чисел с плавающей точкой.

6. Инструкция mov eax, esi копирует содержимое регистра ESI в регистр EAX. Значение в EAX после выполнения этой команды будет равно значению ESI, при этом сам ESI останется неизменным.

7. Инструкция int 80h в ассемблере используется для вызова системных прерываний в Linux. Она выполняет системный вызов, номер которого обычно указывается в регистре EAX, а аргументы передаются через другие регистры.

5. Список литературы

1. GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/.

2. GNU Bash Manual. — 2016. — URL: https://www.gnu.org/software/bash/manual/.

3. Midnight Commander Development Center. — 2021. — URL: https://midnight-commander. org/.

4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.

5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 c. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658.

6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 c. — ISBN 978-1491941591.

7. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.

8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 c. — ISBN 9781784396879.

9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.

10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.

11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.

12. Расширенный ассемблер: NASM. — 2021. — URL: https://www.opennet.ru/docs/RUS/nasm/.

13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.

14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL:

http://www.stolyarov.info/books/asm_unix.

15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).

16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).