

Отчет по лабораторной работе №5

дисциплина: Архитектура компьютера

Михайлова Регина Алексеевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение заданий для самостоятельной работы	13
4	Вывод	16
5	Список литературы	17

Список иллюстраций

2.1	Каталог	6
2.2	Создание файла	7
2.3	Окно Midnight Commander. Редактор nano	7
2.4	Окно Midnight Commander. Редактор mcedit	7
2.5	Исполняемая программа	9
2.6	Копирование файла	10
2.7	Проверка работы программ	11
3.1	Программа 1	13
3.2	Проверка работы программы 1	14
3.3	Программа 2	14
3.4	Проверка работы программы 2	15

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

1. Открываем Midnight Commander:

user@dk4n31:~\$ mc

2. Пользуясь клавишами **⌘**, **⌘** и Enter переходим в каталог ~/work/arch-pc созданный при выполнении лабораторной работы №4 (рис. 2.1).
3. С помощью функциональной клавиши F7 создайте папку lab05 и переходим в созданный каталог.
4. Пользуясь строкой ввода и командой touch создаем файл lab5-1.asm (рис. 2.2).

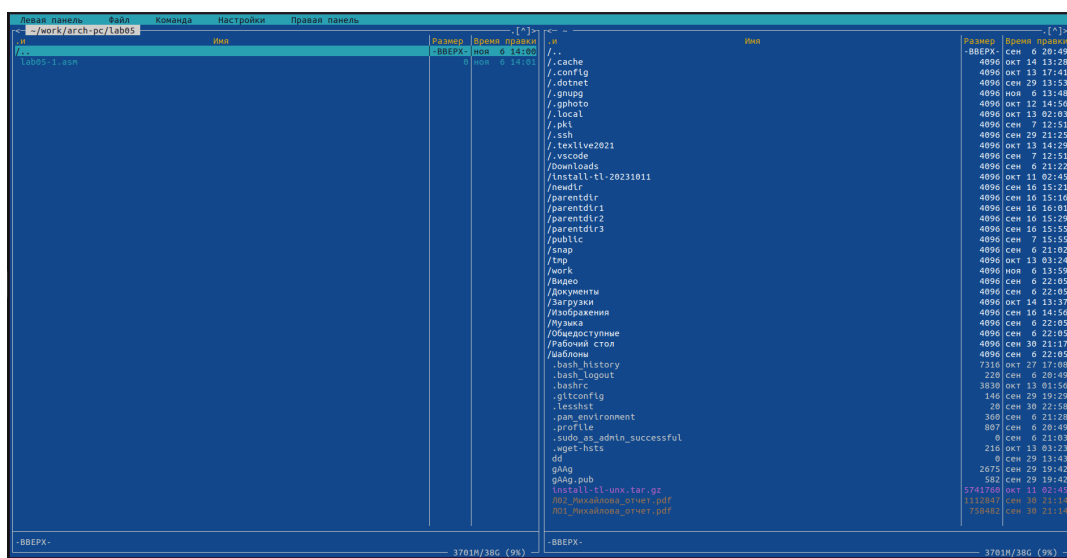


Рис. 2.1: Каталог

Рис. 2.2: Создание файла

```

GNU nano 2.2.1 /home/ranthkhalova/work/arch-pc/lab05/lab05-1.asm
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
buff: 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 ; Описание 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 'buff' размером 80 байт
mov eax,3 ; Системный вызов для чтения (sys_read)
mov ebx,0 ; Дескриптор файла 0 - стандартный ввод
mov ecx,buff ; 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

```

Рис. 2.3: Окно Midnight Commander. Редактор nano

```

/home/ranthkhalova/work/arch-pc/lab05/lab05-1.asm 2074/2074 100%
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
buff: 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 ; Описание 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 'buff' размером 80 байт
mov eax,3 ; Системный вызов для чтения (sys_read)
mov ebx,0 ; Дескриптор файла 0 - стандартный ввод
mov ecx,buff ; 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

```

Рис. 2.4: Окно Midnight Commander. Редактор mcedit

5. С помощью функциональной клавиши F4 открываем файл lab5-1.asm для

редактирования во встроенном редакторе. Как правило в качестве встроенного редактора Midnight Commander используется редакторы nano (рис. 2.3) или mcedit (рис. 2.4).

6. Вводим текст программы из листинга 5.1, сохраняем изменения и закрываем файл.

Листинг 5.1. Программа вывода сообщения на экран и ввода строки с клавиатуры

```
;-----  
; 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  
----- 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 ; Описание 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 ;Descriptor 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

```

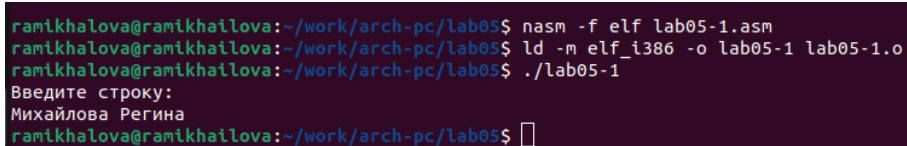
7. С помощью функциональной клавиши F3 открываем файл lab5-1.asm для просмотра. Убеждаемся, что файл содержит текст программы.

8. Оттранслируем текст программы lab5-1.asm в объектный файл. Выполняем компоновку объектного файла и запускаем получившийся исполняемый файл. Программа выводит строку 'Введите строку:' и ожидает ввода с клавиатуры. На запрос вводим наше ФИО. (рис. 2.5)

```

user@dk4n31:~$ nasm -f elf lab5-1.asm user@dk4n31:~$ ld -m elf_i386 -o
lab5-1 lab5-1.o user@dk4n31:~$ ./lab5-1 Введите строку: Имя пользователя
user@dk4n31:~$

```



```

ramikhalova@ramikhallova:~/work/arch-pc/lab05$ nasm -f elf lab05-1.asm
ramikhalova@ramikhallova:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab05-1 lab05-1.o
ramikhalova@ramikhallova:~/work/arch-pc/lab05$ ./lab05-1
Введите строку:
Михайлова Регина
ramikhalova@ramikhallova:~/work/arch-pc/lab05$

```

Рис. 2.5: Исполняемая программа

9. Скачиваем 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 про-граммой, в которой он используется. В одной из панелей `mc` открываем каталог с файлом `lab5-1.asm`. В другой панели каталог со скачанным файлом `in_out.asm`. Копируем файл `in_out.asm` в каталог с файлом `lab5-1.asm` с помощью функциональной клавиши `F5`.
11. С помощью функциональной клавиши `F6` создаем копию файла `lab5-1.asm` с именем `lab5-2.asm`. Выделяем файл `lab5-1.asm`, нажимаем клавишу `F6`, вводим имя файла `lab5-2.asm` и нажимаем клавишу `Enter` (рис. 2.6).

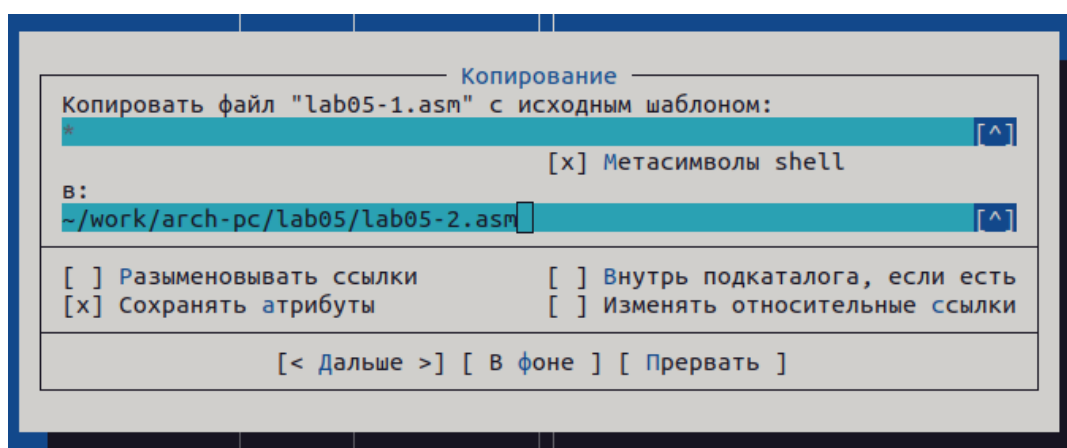


Рис. 2.6: Копирование файла

12. Исправляем текст программы в файле `lab5-2.asm` с использованием подпрограмм из внешнего файла `in_out.asm` (используйте подпрограммы `sprintf`, `sread` и `quit`) в соответствии с листингом 5.2. Создаем исполняемый файл и проверяем его работу. (рис. 2.7)

Листинг 5.2. Программа вывода сообщения на экран и ввода строки с клавиатуры с использованием файла `in_out.asm`

```

;-----
; 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 sprintfLF ; 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

```

13. В файле lab5-2.asm заменяем подпрограмму sprintfLF на sprintf. Создаем исполняемый файл и проверяем его работу. (рис. 2.7)

```

ranikhalova@ranikhalova: /work/arch-pc/lab05$ nasm -f elf lab05-1.asm
lab05-1.asm:7: error: label or instruction expected at start of line
ranikhalova@ranikhalova: /work/arch-pc/lab05$ nc

ranikhalova@ranikhalova: /work/arch-pc/lab05$ nc

ranikhalova@ranikhalova: /work/arch-pc/lab05$ nasm -f elf lab05-1.asm
ranikhalova@ranikhalova: /work/arch-pc/lab05$ ld -m elf_i386 -o lab05-1 lab05-1.o
ranikhalova@ranikhalova: /work/arch-pc/lab05$ ./lab05-1
Введите строку:
Михайлова Регина
ranikhalova@ranikhalova: /work/arch-pc/lab05$ nc

ranikhalova@ranikhalova: /work/arch-pc/lab05$ nasm -f elf lab05-2.asm
lab05-2.asm:13: error: label or instruction expected at start of line
lab05-2.asm:14: error: parser; instruction expected
ranikhalova@ranikhalova: /work/arch-pc/lab05$ nc

ranikhalova@ranikhalova: /work/arch-pc/lab05$ nasm -f elf lab05-2.asm
ranikhalova@ranikhalova: /work/arch-pc/lab05$ ls
in_out.asm lab05-1 lab05-1.asm lab05-1.o lab05-2.asm lab05-2.o
ranikhalova@ranikhalova: /work/arch-pc/lab05$ ld -m elf_i386 -o lab05-2 lab05-2.o
ranikhalova@ranikhalova: /work/arch-pc/lab05$ ls
in_out.asm lab05-1 lab05-1.asm lab05-1.o lab05-2 lab05-2.asm lab05-2.o
ranikhalova@ranikhalova: /work/arch-pc/lab05$ ./lab05-2
Введите строку:
Михайлова Регина
ranikhalova@ranikhalova: /work/arch-pc/lab05$ nc

ranikhalova@ranikhalova: /work/arch-pc/lab05$ nasm -f elf lab05-2.asm
ranikhalova@ranikhalova: /work/arch-pc/lab05$ ld -m elf_i386 -o lab05-2 lab05-2.o
ranikhalova@ranikhalova: /work/arch-pc/lab05$ ./lab05-2
Введите строку: Михайлова Регина
ranikhalova@ranikhalova: /work/arch-pc/lab05$ 

```

Рис. 2.7: Проверка работы программ

Сверяем две программы. Их разница в том, что `sprintf` делает перенос строки, а значит без этой команды текст будет выводиться сразу, без переноса.

3 Выполнение заданий для самостоятельной работы

1. Создаем копию файла lab5-1.asm. Вносим изменения в программу (без использования внешнего файла in_out.asm), так чтобы она работала по следующему алгоритму (рис. 3.1):

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

```
GNU nano 6.2 /home/rantkhalova/work/arch-pc/lab05/lab05-1f.asm
;SECTION .data; секция инициализированных данных
msg: db "Введите строку:", 10, " сообщениe плюс
; символ перевода строки
; Длина переменной 'msg'
SECTION .bss; секция не инициализированных данных
buf: resb 80; Буфер размером 80 байт
;SECTION .text; Код программы
global _start; Начало программы
_start:; Точка входа в программу
; 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; Файловый дескриптор 1 - стандартный вывод
mov ecx, msg; Адрес строки 'msg' в 'edx'
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 'buf' размером 80 байт
mov ebx, 3; Системный вызов для чтения (sys_read)
mov ecx, 0; Дескриптор файла 0 - стандартный ввод
mov edx, buf; Адрес буфера под вводную строку
mov edx, 80; Длина вводной строки
int 80h; Вызов ядра
mov eax, 4;
mov ebx, 1;
mov ecx, buf;
mov edx, 80;
int 80h;
;----- 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;
```

Рис. 3.1: Программа 1

2. Получаем исполняемый файл и проверяем его работу (рис. 3.2). На приглашение ввести строку вводим свою фамилию.

```

ramikhalova@ramikhalova:~/work/arch-pc/lab05$ nasm -f elf lab05-1f.asm
ramikhalova@ramikhalova:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab05-1f lab05-1f.o
ramikhalova@ramikhalova:~/work/arch-pc/lab05$ ./lab05-1f
Введите строку:
Михайлова Регина
Михайлова Регина
ramikhalova@ramikhalova:~/work/arch-pc/lab05$ 

```

Рис. 3.2: Проверка работы программы 1

3. Создаем копию файла lab5-2.asm. Исправляем текст программы с использованием под-программ из внешнего файла in_out.asm, так чтобы она работала по следующему алгоритму (рис. 3.3):

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

```

GNU nano 6.2 /home/ramikhalova/work/arch-pc/lab05/lab05-2f.asm
#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 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 ; запись адреса выводимого сообщения в 'EAX'
call sprintf ;
call quit ; вызов подпрограммы завершения

```

Рис. 3.3: Программа 2

4. Создаем исполняемый файл и проверяем его работу (рис. 3.4).

```
ramikhalova@ramikhalova:~/work/arch-pc/lab05$ nasm -f elf lab05-2f.asm
ramikhalova@ramikhalova:~/work/arch-pc/lab05$ ld -m elf_i386 -o lab05-2f lab05-2f.o
ramikhalova@ramikhalova:~/work/arch-pc/lab05$ ./lab05-2f
Введите строку: Михайлова Регина
Михайлова Регина

ramikhalova@ramikhalova:~/work/arch-pc/lab05$
```

Рис. 3.4: Проверка работы программы 2

4 Вывод

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

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 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — 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,
- 11.
12. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
13. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
14. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ- Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
15. 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.
16. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
 17. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер,
 18. — 1120 с. — (Классика Computer Science).