

Отчёт по лабораторной работе №5

дисциплина: Архитектура компьютера

Веретенников Дмитрий Олегович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Основы работы с Midnight Commander	8
4.2	Подключение внешнего файла in_out.asm	11
5	Выводы	14
	Список литературы	15

Список иллюстраций

4.1	Midnight Commander	8
4.2	Текст программы из листинга 5.1	9
4.3	Текст программы	10
4.4	Проверка работы программы	11
4.5	Файл lab5-2.asm	12
4.6	Текст программы lab5-2.asm	13
4.7	Проверка работы программы	13

Список таблиц

1 Цель работы

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

2 Задание

1. Основы работы с mc
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 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 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 используют

4 Выполнение лабораторной работы

4.1 Основы работы с Midnight Commander

Открываю Midnight Commander и перехожу в каталог созданный при выполнении лабораторной работы №4 (рис. 4.1).

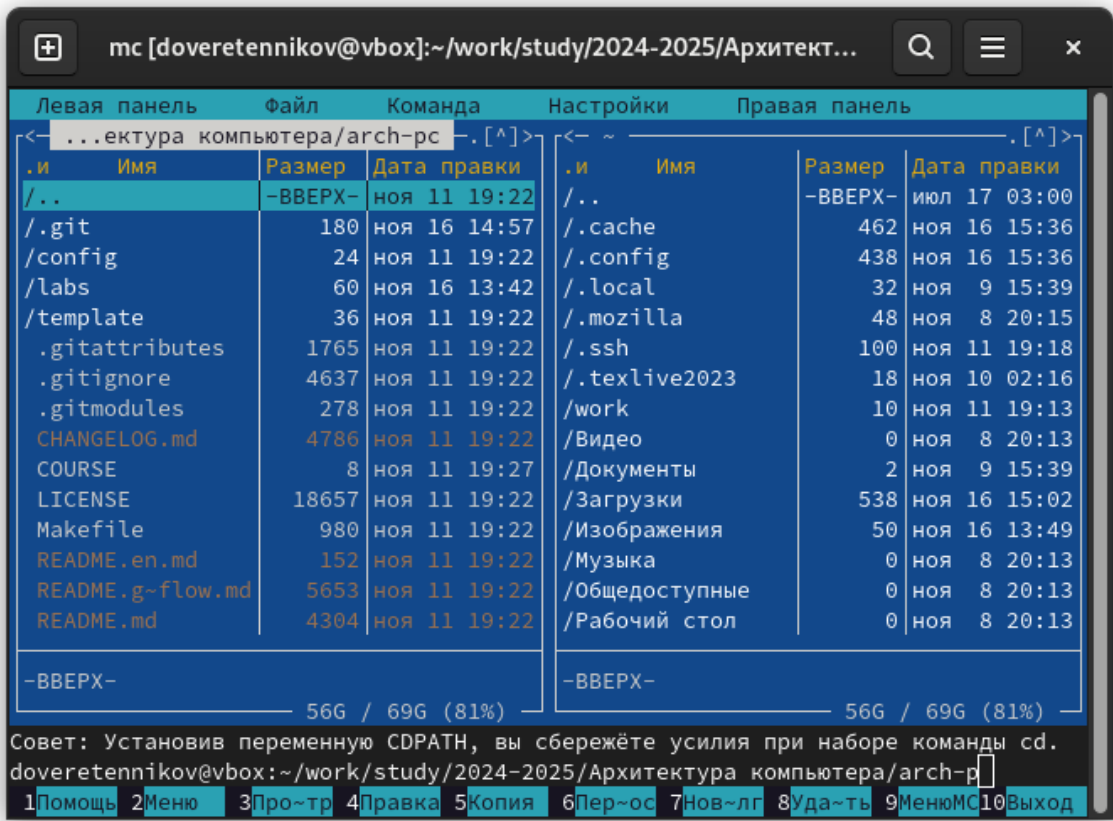


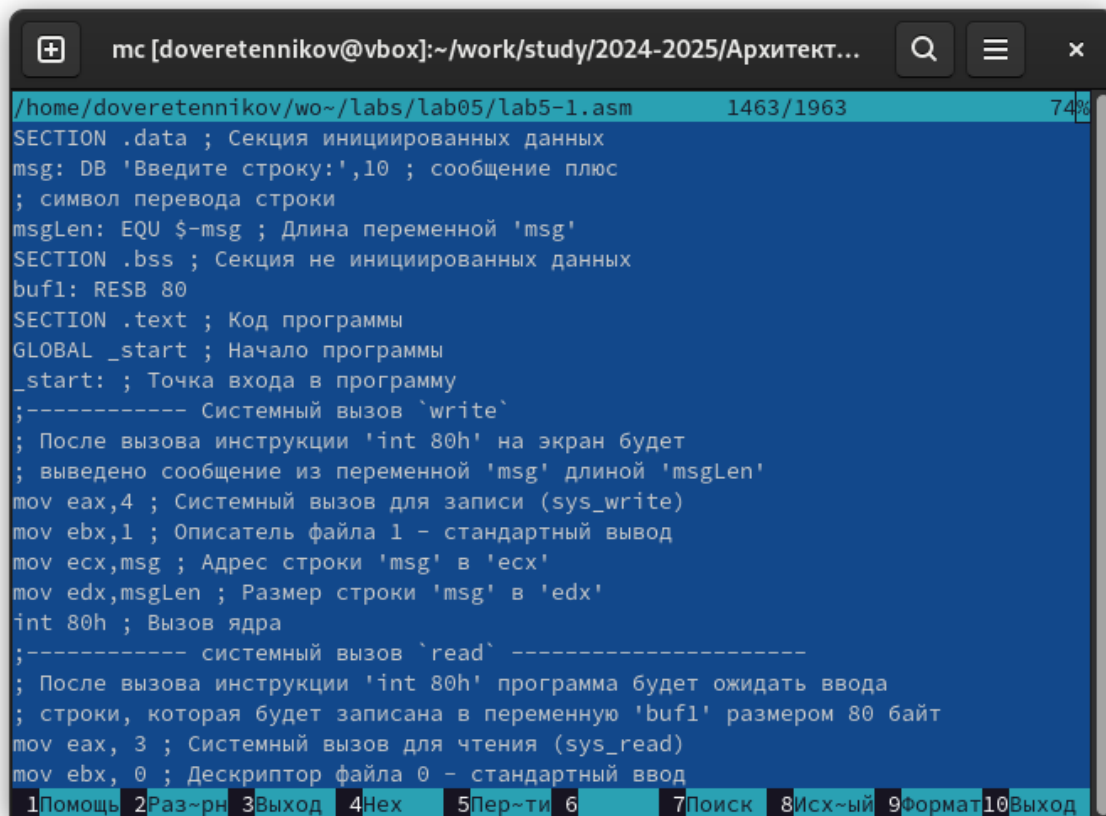
Рис. 4.1: Midnight Commander

Создаю файл lab5-1.asm, с помощью функциональной клавиши F4 открываю файл для редактирования и ввожу текст программы из листинга 5.1 (рис. 4.2).

```
lab5-1.asm [----] 20 L:[ 1+ 8 9/ 30] *(455 /1963b) 1072 0x430 [*][X]
SECTION .data ; Секция иницированных данных
msg: DB 'Введите строку:',10 ; сообщение плюс
; символ перевода строки
msgLen: EQU $-msg ; Длина переменной 'msg'
SECTION .bss ; Секция не иницированных данных
buf1: RESB 80.
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
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюМС10Выход
```

Рис. 4.2: Текст программы из листинга 5.1

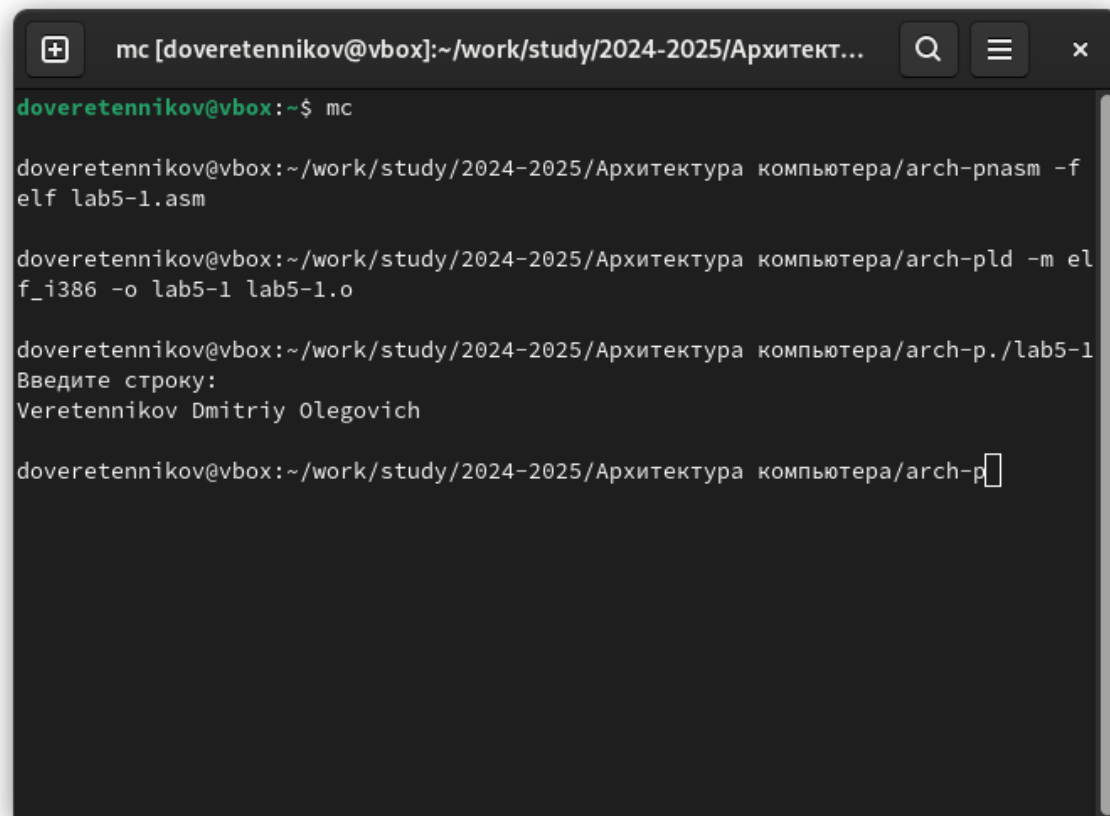
С помощью функциональной клавиши F3 открываю файл lab5-1.asm для просмотра и убеждаюсь, что файл содержит текст программы (рис. 4.3).



```
mc [doveretennikov@vbox]:~/work/study/2024-2025/Архитект...
/home/doveretennikov/wo~/labs/lab05/lab5-1.asm 1463/1963 74%
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
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
1Помощь 2Раз~рн 3Выход 4Нех 5Пер~ти 6 7Поиск 8Исх~ый 9Формат10Выход
```

Рис. 4.3: Текст программы

Транслирую текст программы lab5-1.asm в объектный файл, выполняю компоновку объектного файла и запускаю исполняемый файл (рис. 4.4).

A screenshot of a terminal window with a dark background. The window title is "mc [doveretennikov@vbox]:~/work/study/2024-2025/Архитект...". The terminal shows the following commands and output:

```
doveretennikov@vbox:~$ mc
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pnasm -f
elf lab5-1.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pld -m el
f_i386 -o lab5-1 lab5-1.o
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-p./lab5-1
Введите строку:
Veretennikov Dmitry Olegovich
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-p
```

Рис. 4.4: Проверка работы программы

4.2 Подключение внешнего файла in_out.asm

Создаю копию файла lab5-1.asm с именем lab5-2.asm (рис. 4.5).

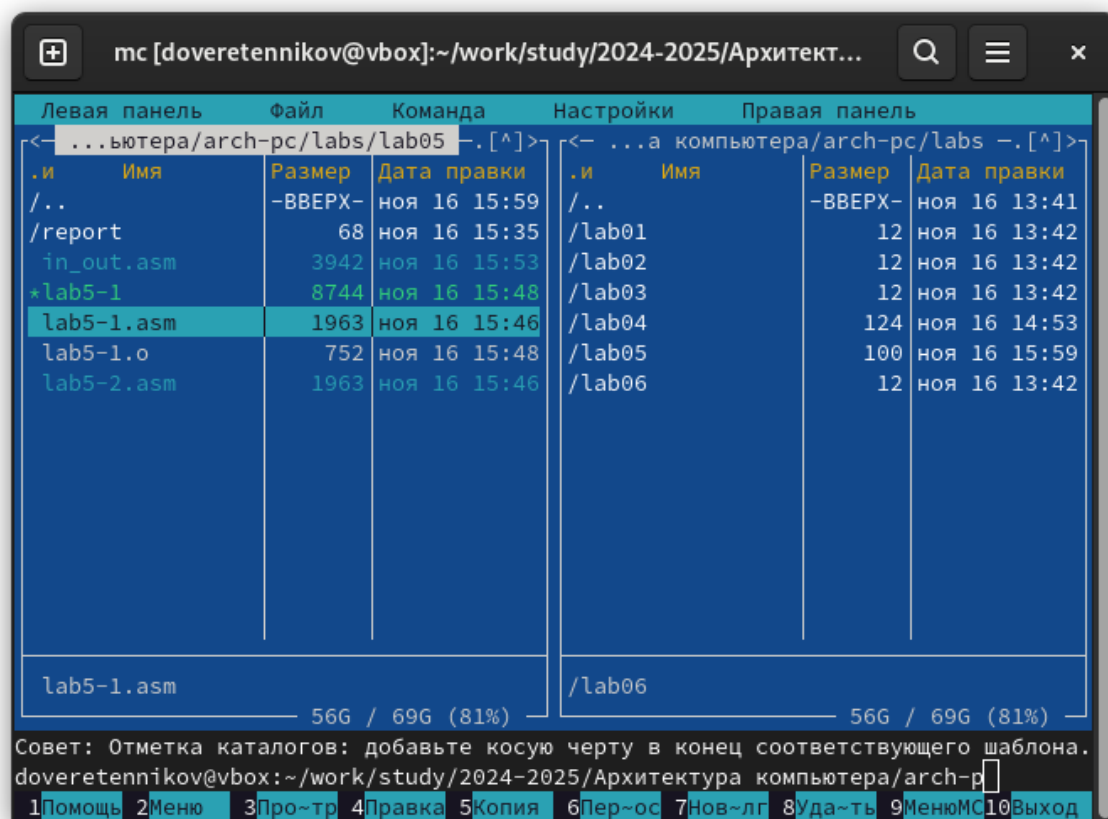
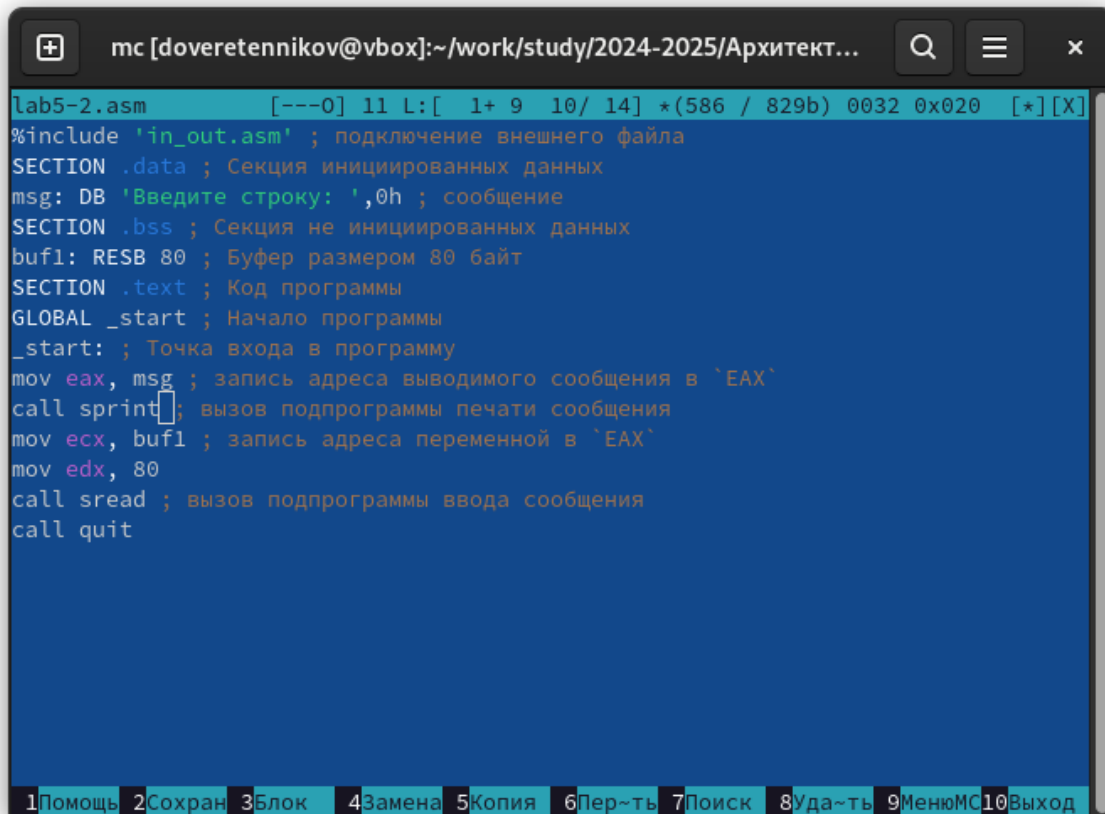


Рис. 4.5: Файл lab5-2.asm

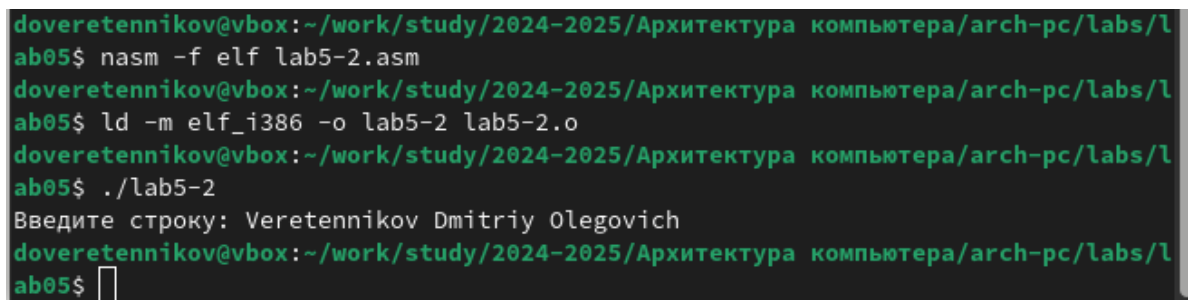
В файле lab5-2.asm заменяю подпрограмму `sprintLF` на `sprint` и проверяю работу программы (рис. 4.6).



```
lab5-2.asm      [---0] 11 L:[ 1+ 9 10/ 14] *(586 / 829b) 0032 0x020 [*][X]
#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
call sread ; вызов подпрограммы ввода сообщения
call quit
```

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер~ть 7Поиск 8Уда~ть 9МенюMC10Выход

Рис. 4.6: Текст программы lab5-2.asm



```
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab05$ nasm -f elf lab5-2.asm
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab05$ ld -m elf_i386 -o lab5-2 lab5-2.o
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab05$ ./lab5-2
Введите строку: Veretennikov Dmitry Olegovich
doveretennikov@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/l
ab05$
```

Рис. 4.7: Проверка работы программы

Разница подпрограмм в том, что вторая вызывает ввод на той же строке.
Далее копирую файлы в локальный репозиторий и загружаю файлы на Github.

5 Выводы

После выполнения данной лабораторной работы я приобрёл практические навыки работы в Midnight Commander, а также освоил инструкции языка ассемблера `mov` и `int`.

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).