Отчёта по лабораторной работе #8

НПИ-03-23

Махмудов Суннатилло Баходир угли

Содержание

# 1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

# 2 Задание

Реализация циклов в NASM

Обработка аргументов командной строки

Задание для самостоятельной работы

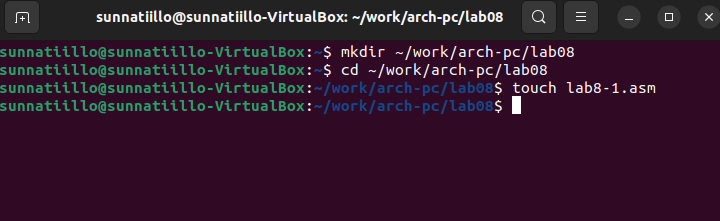
# 3 Теоретическое введение

Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды. Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров. На рис. 8.1 показана схема организации стека в процессоре. Стек имеет вершину, адрес последнего добавленного элемента, который хранится в ре- гистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указа- тель стека уменьшается, а при извлечении — увеличивается. Для стека существует две основные операции: • добавление элемента в вершину стека (push); • извлечение элемента из вершины стека (pop).

# 4 Выполнение лабораторной работы

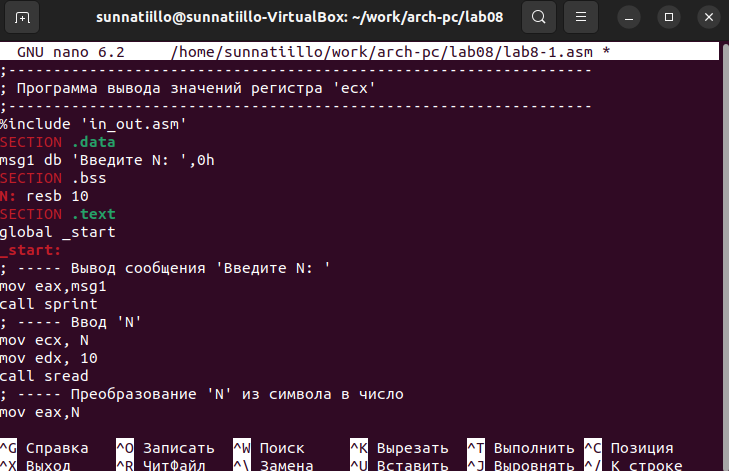
Реализация циклов в NASM

Создал каталог для программам лабораторной работы № 8, перешел в него и создал файл lab8-1.asm: (рис. ??).



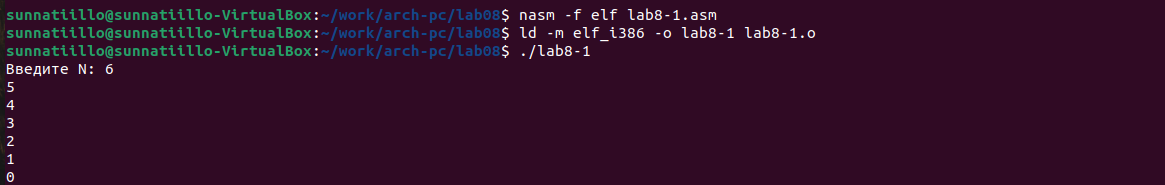
Созданный файл lab8-1.asm

Ввел в файл lab8-1.asm текст программы из листинга 8.1.(рис. ??).



Текст из листинга 8.1

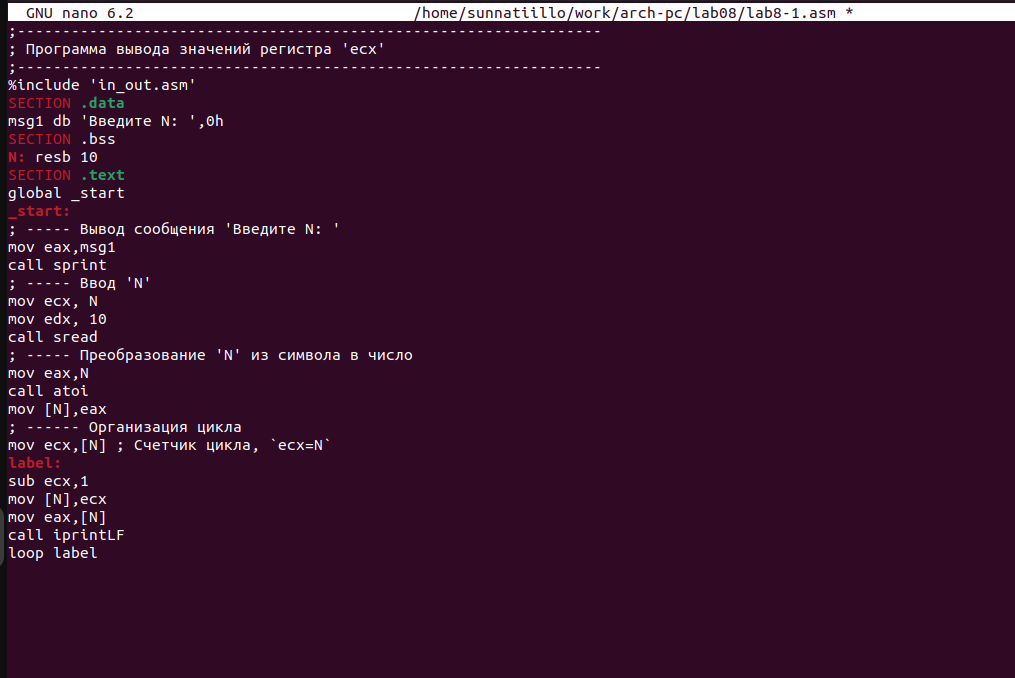
Исполняемый файли проверял его работу.(рис. ??).



Резултать работу

Данный пример показывает, что использование регистра ecx в теле цилка loop может привести к некорректной работе программы.

Изменил текст программы добавил изменение значение регистра ecx в цикле(рис. ??).



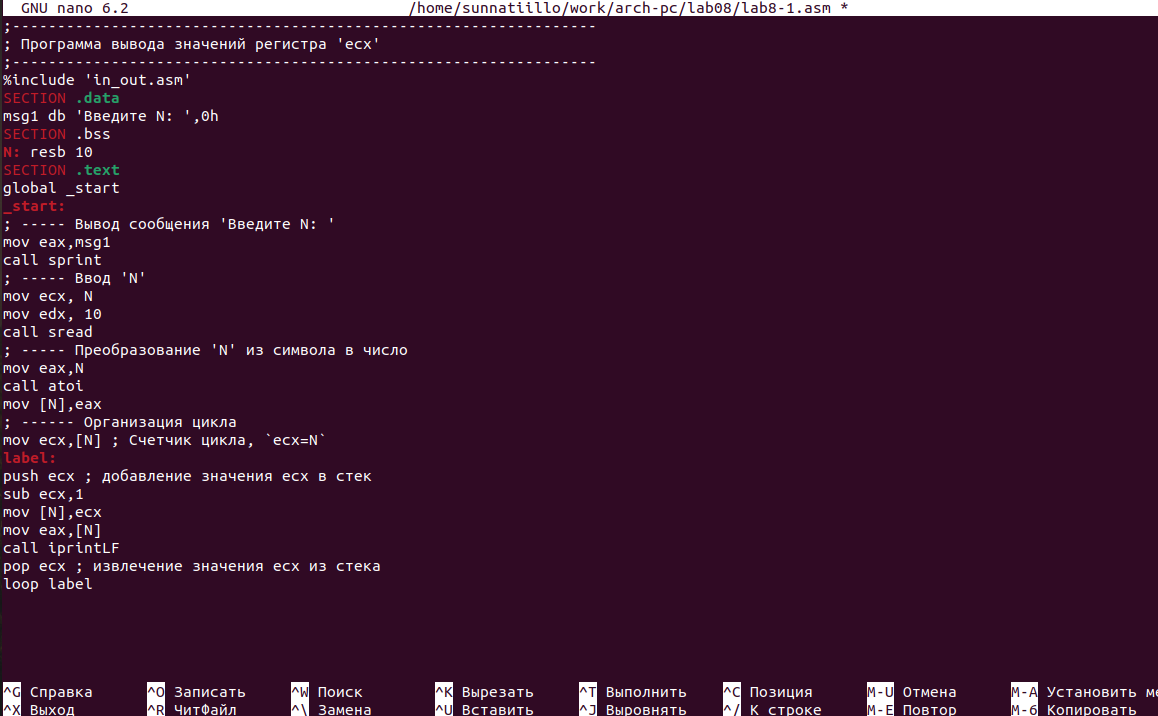
Измененный текст



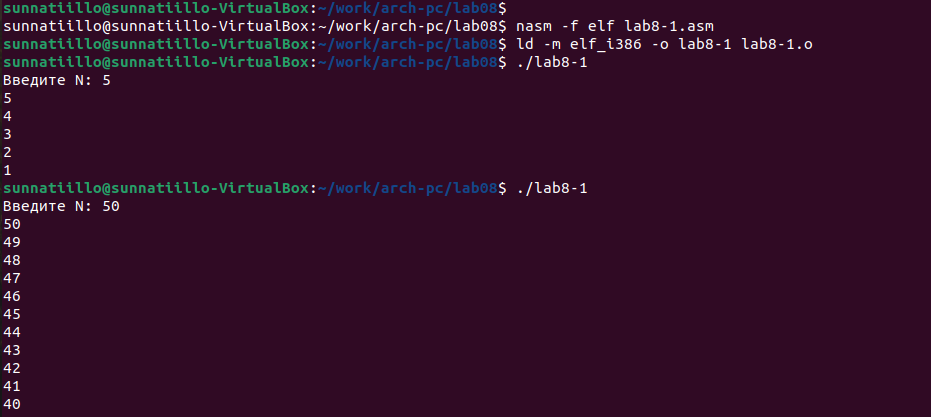
Проверял его работу

Для использования регистра ecx в цикле и сохранения корректности работы программы можно использовать стек. В изменения в текст программы добавив команды push и pop для сохранения значения счетчика цикла loop:

Опять изменил текст программы (рис. ??).



Измененный текст



Проверял его работу

Обработка аргументов командной строки

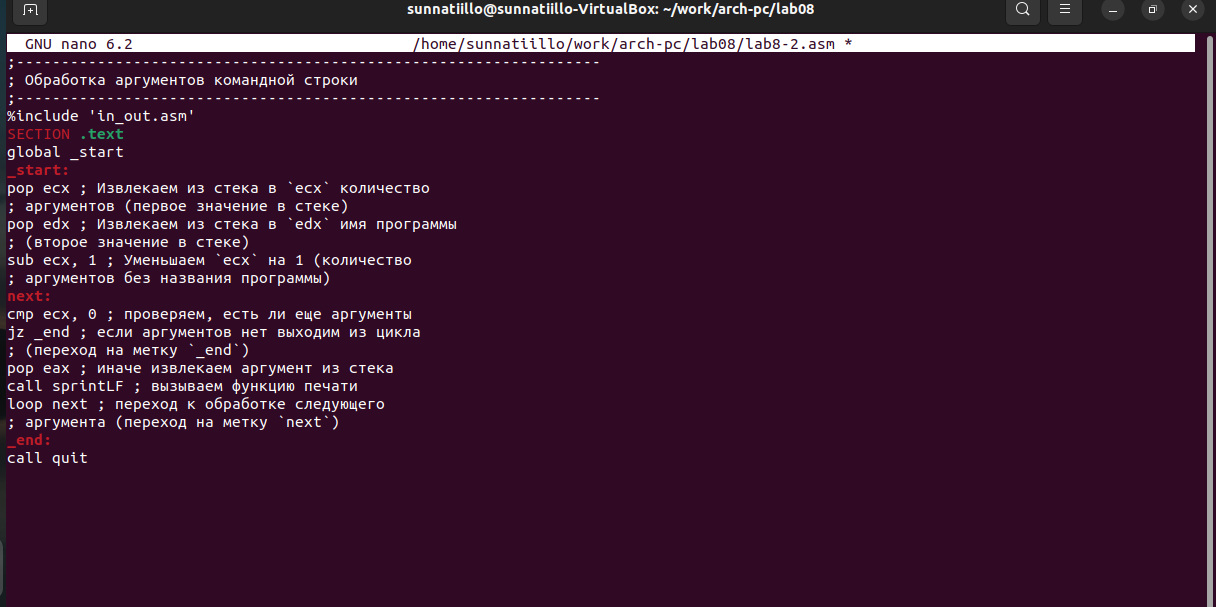
При запуске программы в NASM аргументы командной строки загружаются в стек в обрат- ном порядке, кроме того в стек записывается имя программы и общее количество аргументов. Последние два элемента стека для программы, скомпилированной NASM, – это всегда имя программы и количество переданных аргументов.

Создал файл lab8-2.asm (рис. ??).

Файл lab8-2.asm

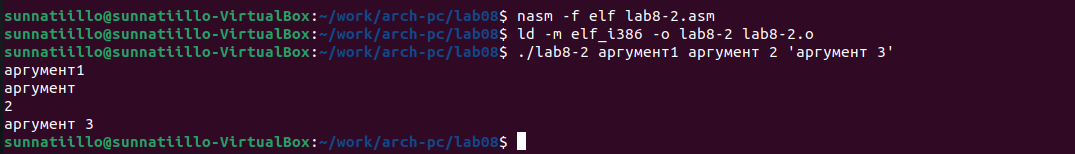
Файл lab8-2.asm

Ввел в него текст программы из листинга 8.2.(рис. ??).



Текст из листинга 8.2

На исполняемый файл запустил указав аргументы:(рис. ??).



Аргументы полученые

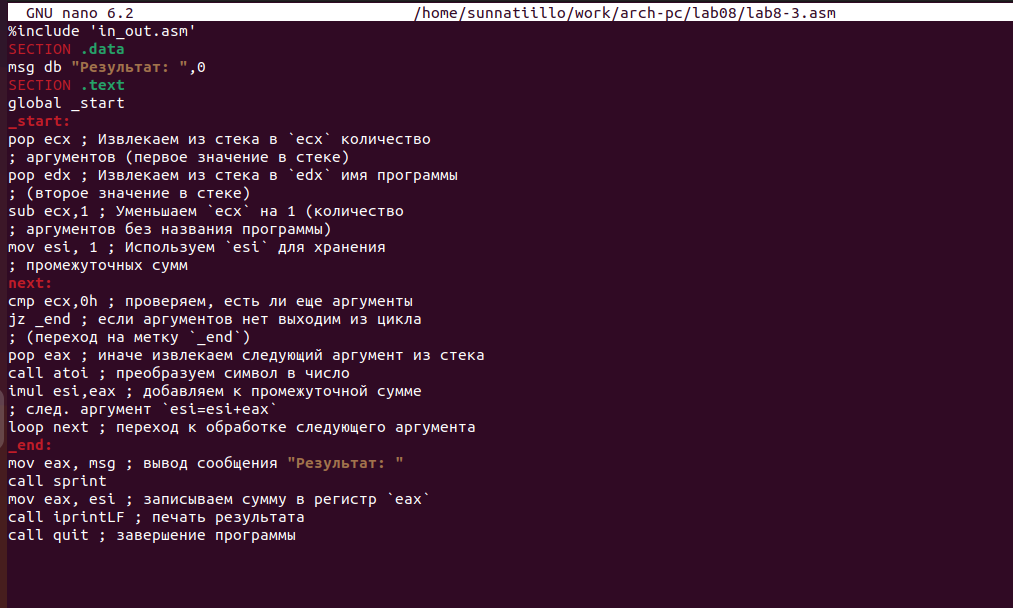
Рассмотрим еще один пример программы которая выводит сумму чисел, которые пере- даются в программу как аргументы.

Создал файл lab8-3.asm (рис. ??).

файл lab8-3.asm

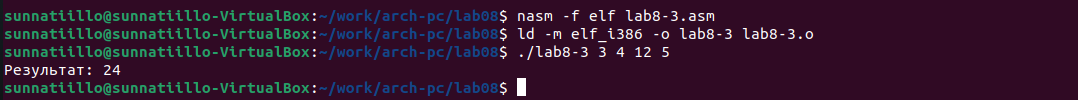
файл lab8-3.asm

Ввел на файл lab8-3.asm текст программы из листинга 8.3.(рис. ??).



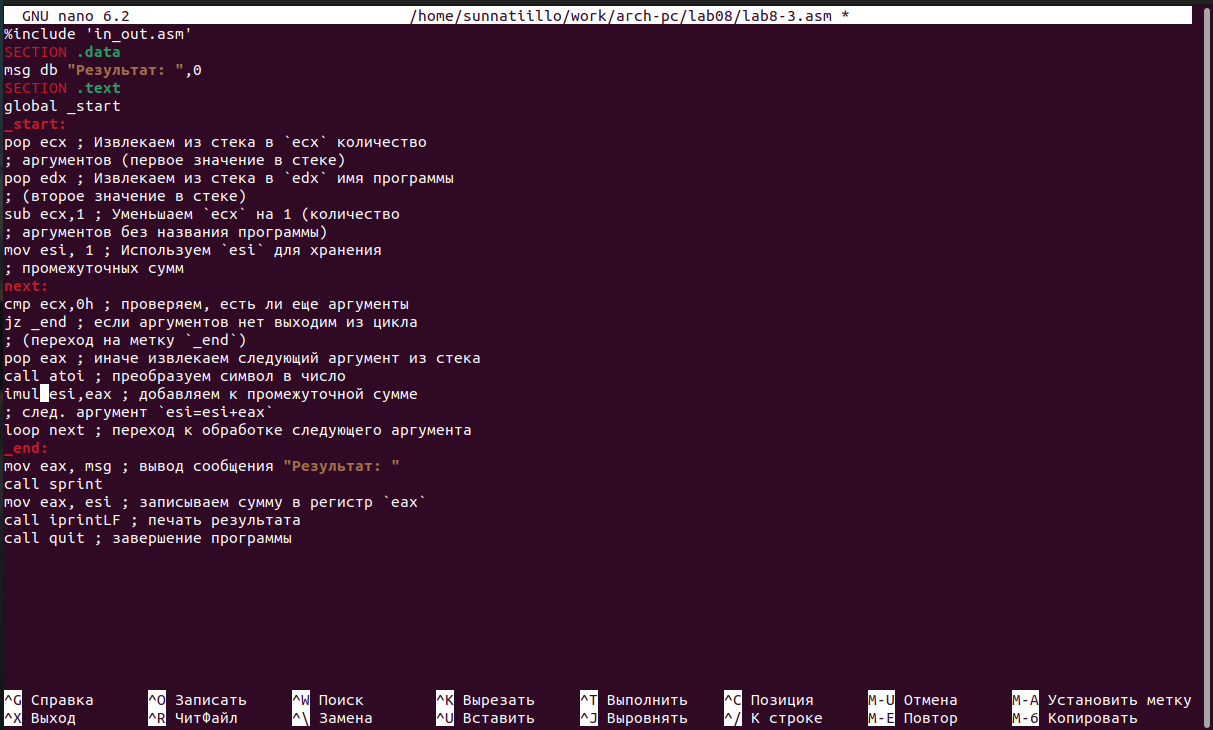
Текст из листинга 8.3

Создал исполняемый файл и запустите его, указав аргументы.(рис. ??).



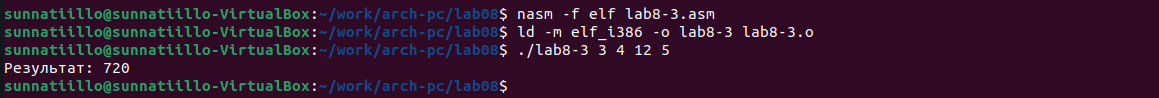
Резултат

Изменил текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.(рис. ??).



Изменный текст

Проверял его работу (рис. ??).

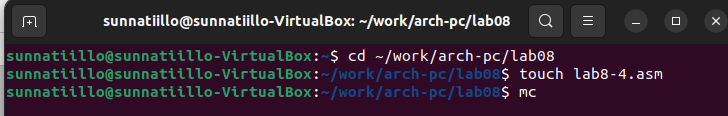


Резултать работы

# 5 Задание для самостоятельной работы

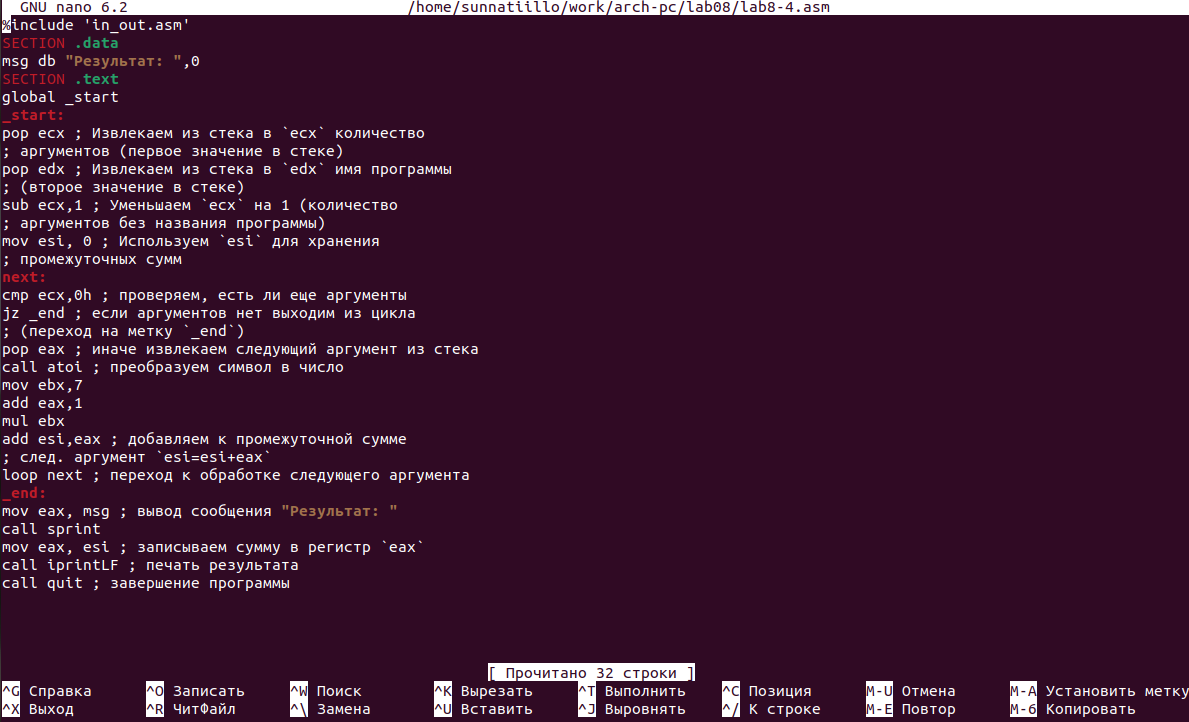
Вариант 14:

Я создал файл lab8-4.asm чтобы сделать свою вариант :(рис. ??).



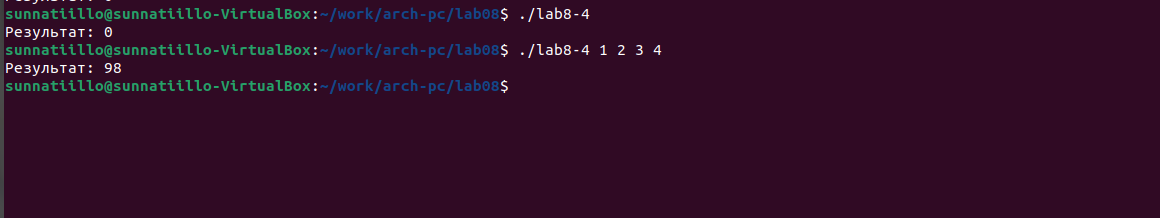
файл lab8-4.asm

Ввел на файл lab8-4.asm текст из листинга 8.3 и поменял (рис. ??).



Текст

Проверял его работу (рис. ??).



Резултат работы

# 6 Выводы

Я научился писать программы с использованием циклов и обрабатывать аргументы командной строки.

# 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. 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. — БХВ- Петербург, 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,
17. — 1120 с. — (Классика Computer Science).