Отчёт по лабораторной работе №8

Простейший вариант

Волчкова Елизавета Дмитриевна

Содержание

# Цель работы

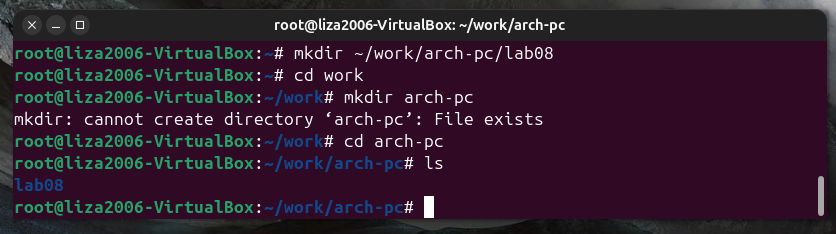
Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

# Задание

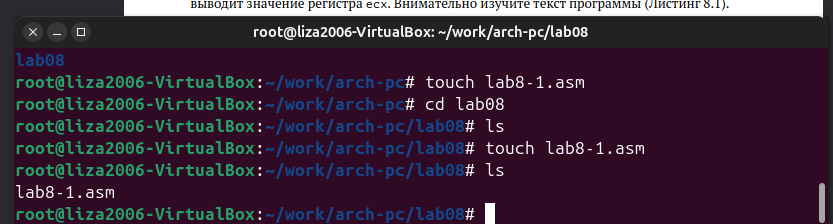
Напишите программу, которая находит сумму значений функции 𝑓(𝑥) для 𝑥 = 𝑥1, 𝑥2, …, 𝑥𝑛, т.е. программа должна выводить значение 𝑓(𝑥1) + 𝑓(𝑥2) + … + 𝑓(𝑥𝑛). Значения 𝑥𝑖 передаются как аргументы. Вид функции 𝑓(𝑥) выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах 𝑥 = 𝑥1, 𝑥2, …, 𝑥𝑛.

# Выполнение лабораторной работы

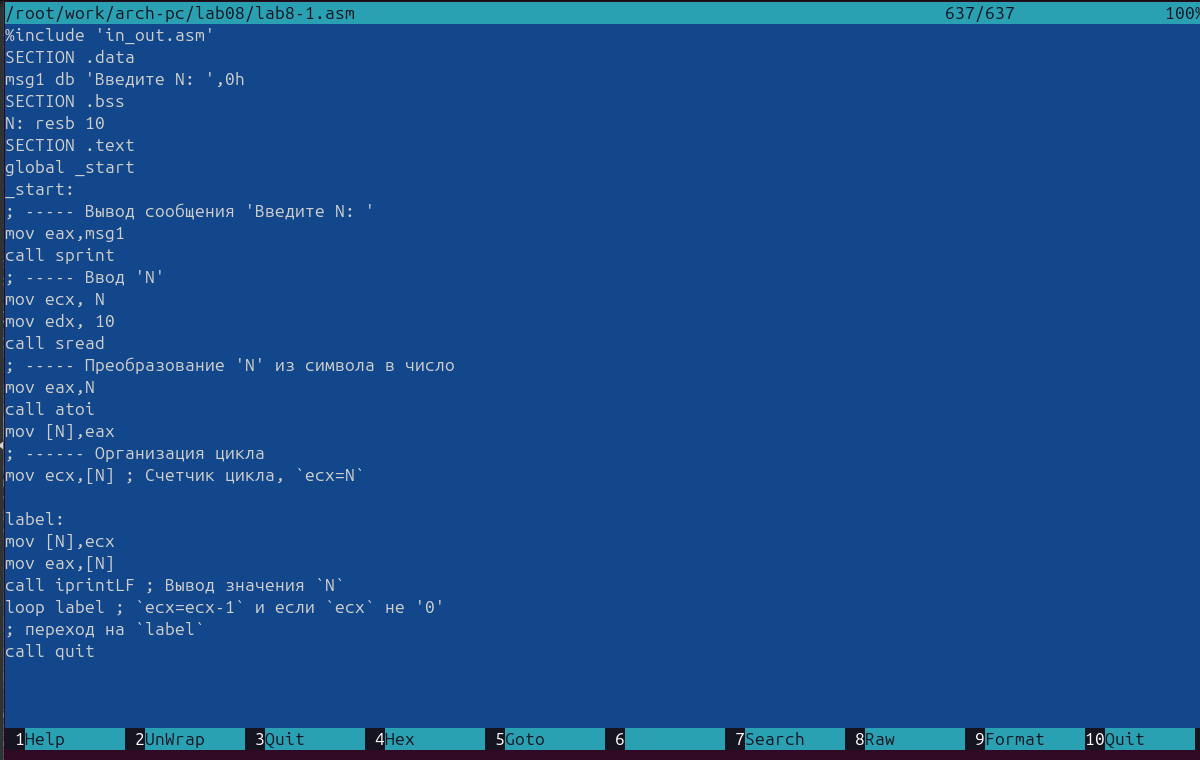
Создала каталог lab08



Ввела в файл lab8-1.asm текст программы из листинга 8.1.

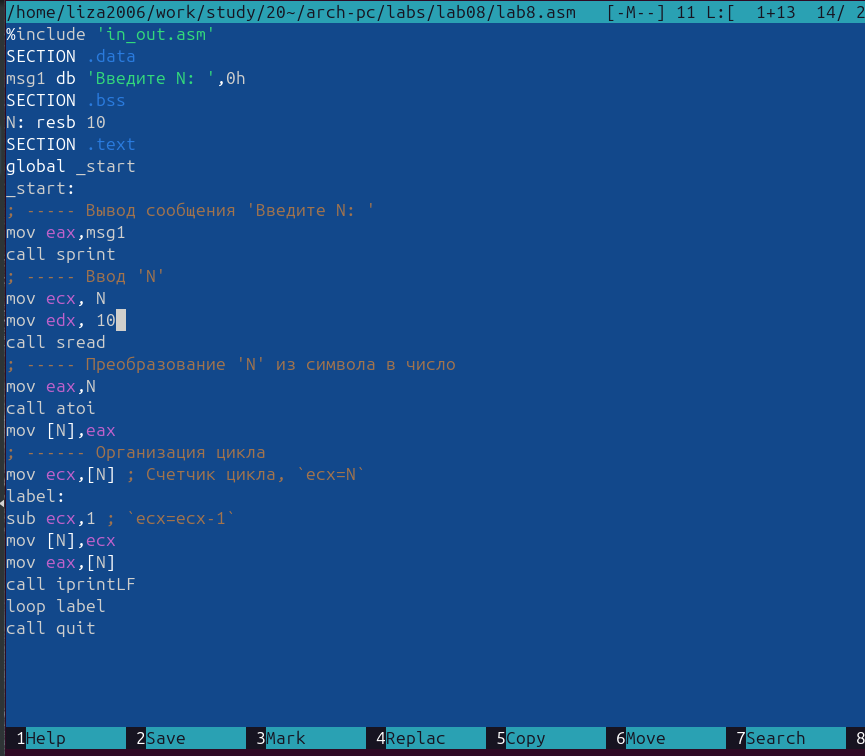


Затем создала исполняемый файл и проверила его работу.



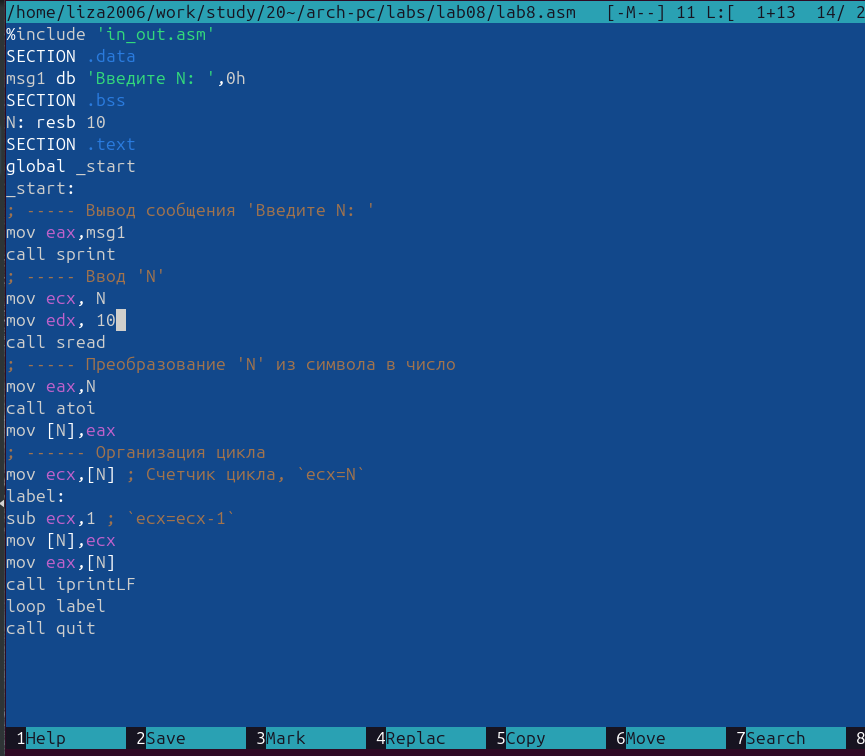
Данный пример показывает, что использование регистра ecx в теле цикла loop может привести к некорректной работе программы. Далее я изменила текст программы добавив изменение значение регистра ecx в цикле.

Потом создала исполняемый файл и проверила его работу. В качестве примера рассмотрела программу, которая выводит значение регистра ecx. Внимательно изучила текст программы. Создала исполняемый файл и проверила его работу. Регистр ecx принимает значение, равное значению 𝑁, введенному с клавиатуры.

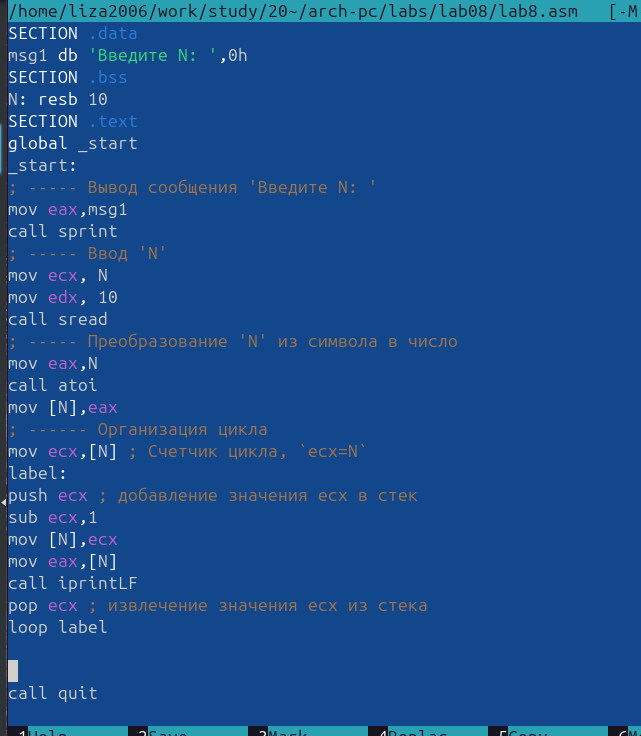


После внесла изменения в текст программы, добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop. Создала исполняемый файл и проверила его работу. Соответствует ли в данном случае число проходов цикла значению 𝑁 введенному с клавиатуры? - нет! Циклов было больше 100.

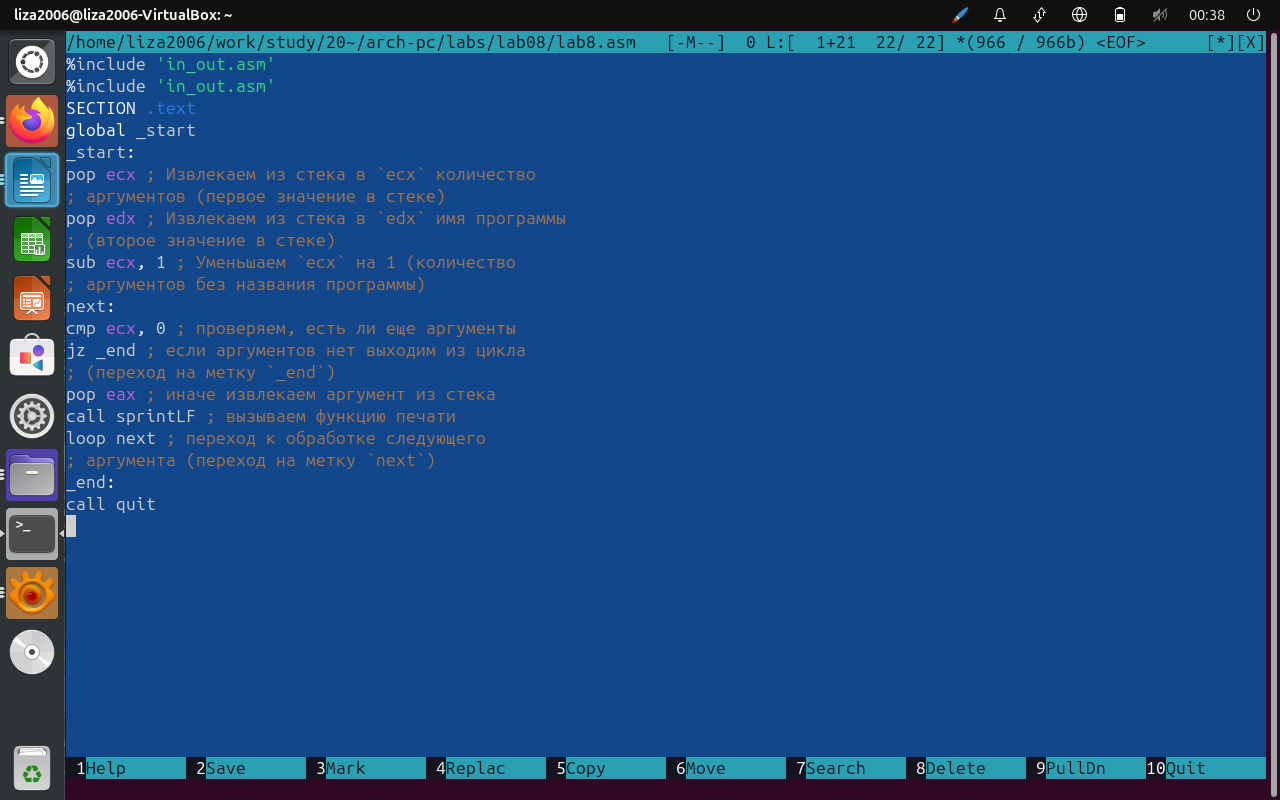




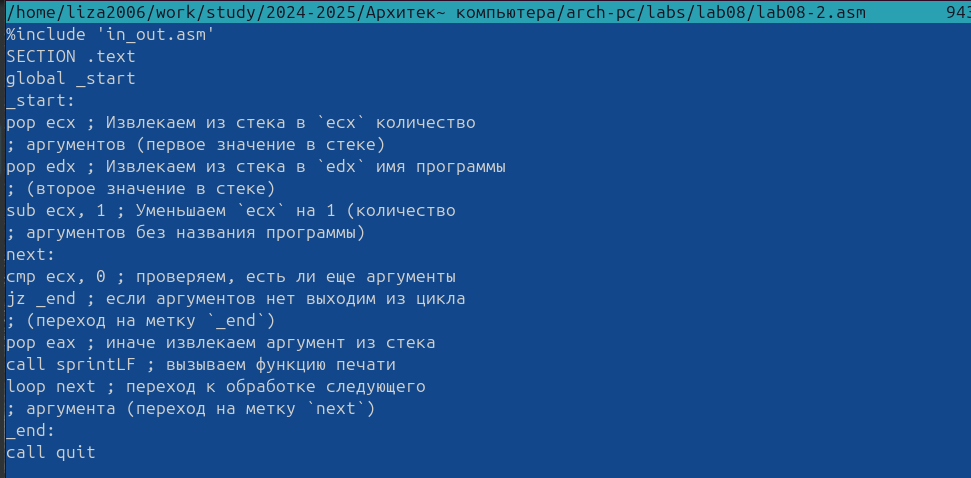
Для использования регистра ecx в цикле и сохранения корректности работы программы можно использовать стек. Внесите изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop:



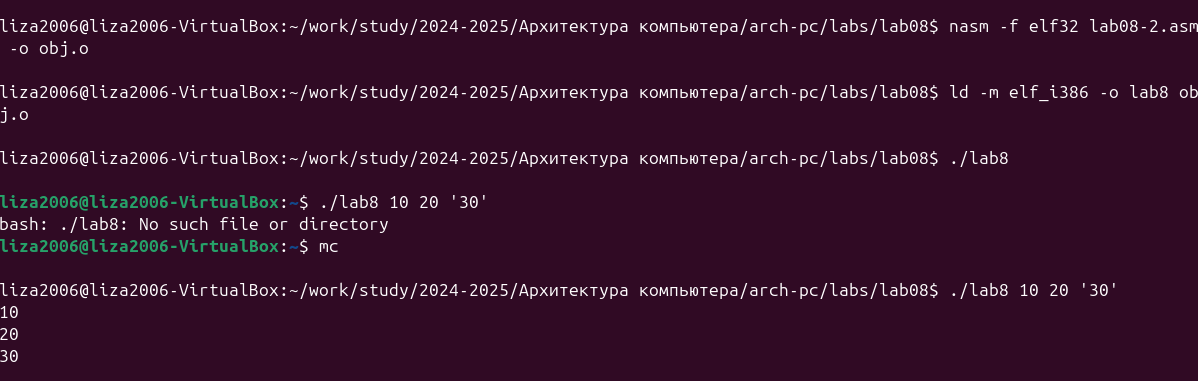
Для того чтобы использовать аргументы в программе, их просто нужно извлечь из стека. Обработку аргументов нужно проводить в цикле. В качестве примера рассмотрела программу, которая выводит на экран аргументы командной строки. Изучила текст программы:



Создала файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввела в него текст программы из листинга 8.2.

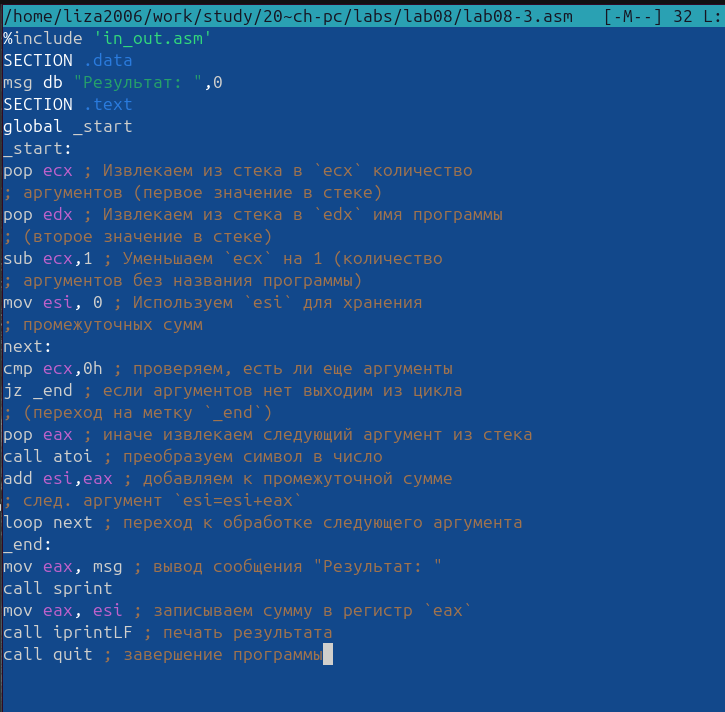


Создала исполняемый файл и запустила его, указав аргументы:

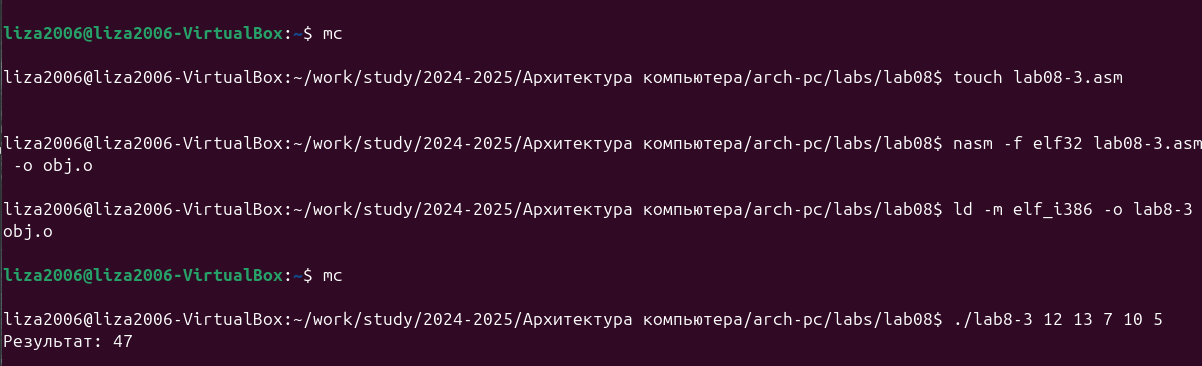


Сколько аргументов было обработано программой? -Было обработано 3!

Рассмотрела еще один пример программы, которая выводит сумму чисел, которые передаются в программу как аргументы. Создала файл lab8-3.asm в каталоге ~/work/arch- pc/lab08 и ввела в него текст программы из листинга 8.3



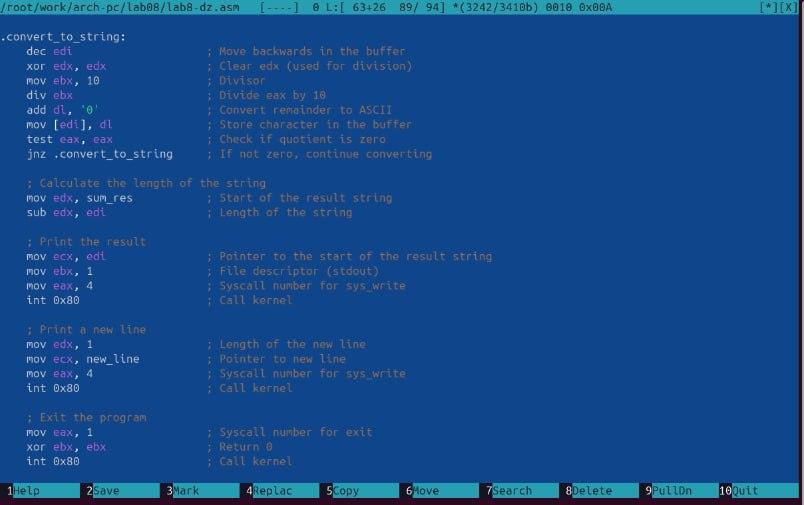
Создала исполняемый файл и запустила его, указав аргументы. Пример результата работы программы:



Затем изменила текст программы из листинга 8.3 для вычисления произведения аргументов командной строки

Написала программу, которая находит сумму значений функции 𝑓(𝑥) для

𝑥 = 𝑥1, 𝑥2, …, 𝑥𝑛, т.е. программа должна выводить значение 𝑓(𝑥1) + 𝑓(𝑥2) + … + 𝑓(𝑥𝑛). Значения 𝑥𝑖 передаются как аргументы. Вид функции 𝑓(𝑥) выбрала из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7 и создав исполняемый файл проверила его работу на нескольких наборах 𝑥 = 𝑥1, 𝑥2, …, 𝑥𝑛



# Выводы

Целью работы было приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки. Проделав данные задания я усвоила материал и теперь имею представления о том, как правильно использовать циклы и командные 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, 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. — 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, 2015. — 1120 с. — (Классика Computer Science). ::: {#refs} :::