Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки.

Дмитрий Сергеевич Хохлов

Содержание

# 1 Цель работы

Целью работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки..

# 2 Выполнение лабораторной работы

Сформировал директорию для программ, связанных с лабораторной № 8, и создал исходный файл с именем lab8-1.asm.

При использовании циклов в ассемблере NASM и применении команды loop следует учитывать, что она автоматически декрементирует счетчик, находящийся в регистре ecx. Возьмем для примера программу, демонстрирующую вывод текущего значения регистра ecx.

В файл lab8-1.asm внес код из примера под номером 8.1, представленного на иллюстрации [[1](#fig:001)]. Скомпилировал программу, получил исполняемый файл и осуществил его тестирование, результаты которого отражены на рисунке [[2](#fig:002)].

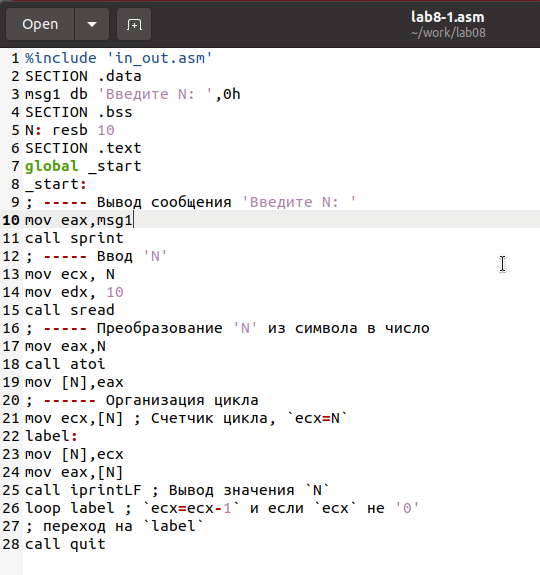


Figure 1: Программа в файле lab8-1.asm

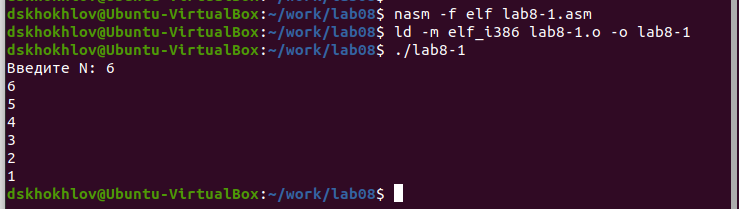


Figure 2: Запуск программы lab8-1.asm

Приведенный пример иллюстрирует, что манипуляции с регистром ecx внутри цикла loop могут привести к ошибкам в работе программы. Внес корректировки в код программы, добавив действия по изменению значения регистра ecx в процессе итераций, как показано на рисунке [[3](#fig:003)].

Программа инициирует бесконечный цикл, если N имеет нечетное значение, и выводит исключительно нечетные числа, если N четное, как видно на рисунке [[4](#fig:004)].

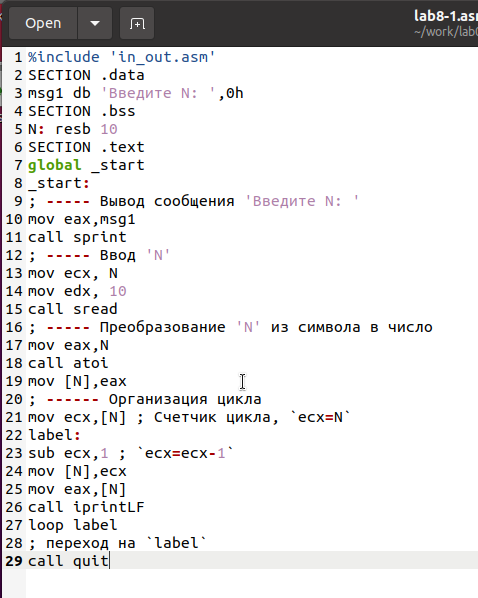


Figure 3: Программа в файле lab8-1.asm

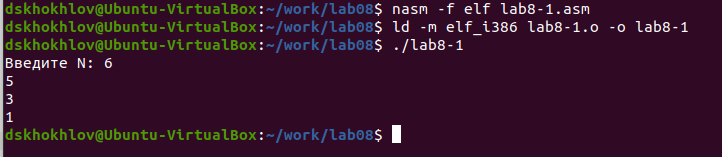


Figure 4: Запуск программы lab8-1.asm

Для корректного использования регистра ecx в цикле и избегания ошибок можно применять стек. Произвел модификацию программного кода, добавив инструкции push и pop для сохранения и восстановления счетчика цикла, что отображено на рисунке [[5](#fig:005)].

Скомпилировал исполняемый файл и провел его тестирование. Программа последовательно выводит числа от N-1 до 0, с количеством итераций, равным N, что подтверждается изображением [[6](#fig:006)].

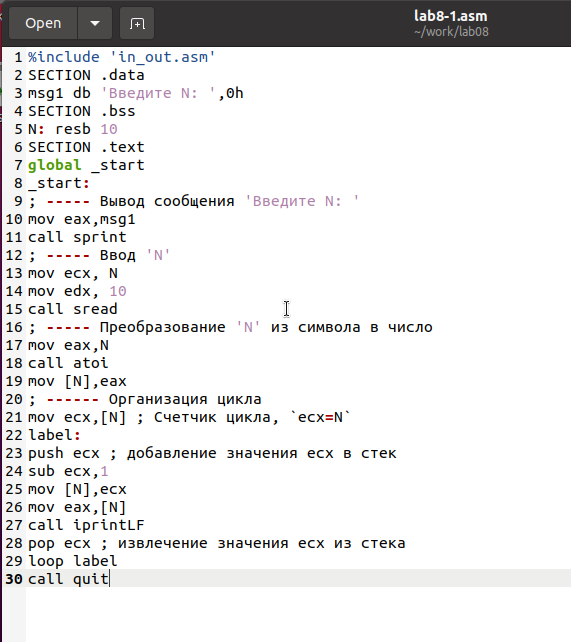


Figure 5: Программа в файле lab8-1.asm

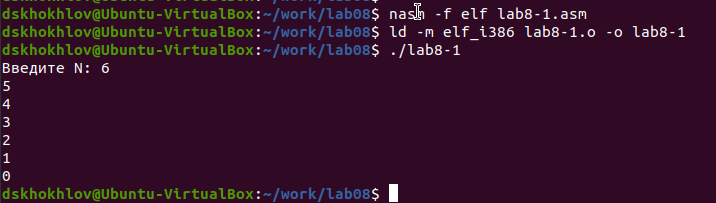


Figure 6: Запуск программы lab8-1.asm

Сформировал файл lab8-2.asm в директории ~/work/arch-pc/lab08 и занес в него код из примера 8.2, как показано на иллюстрации [[7](#fig:007)].

Скомпилировал и выполнил скомпилированный файл, предоставив ему в качестве параметров несколько аргументов. В результате, программа успешно обработала пять аргументов, которые определяются как отдельные слова или числа, разделенные пробелами, (рис. [[8](#fig:008)]).

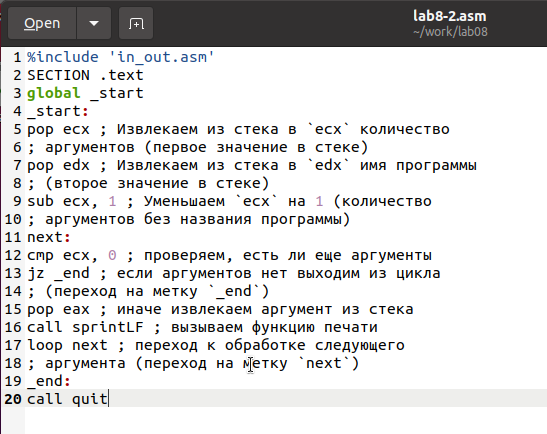


Figure 7: Программа в файле lab8-2.asm

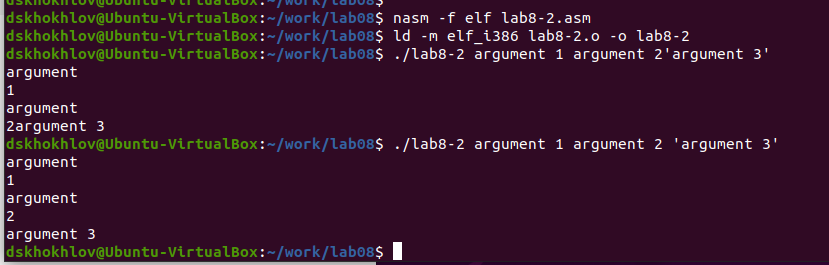


Figure 8: Запуск программы lab8-2.asm

Теперь давайте обратим внимание на другой пример программы, которая демонстрирует расчет суммы чисел, передаваемых в нее в виде аргументов командной строки, что визуализировано на рисунках [[9](#fig:009)] и [[10](#fig:010)].

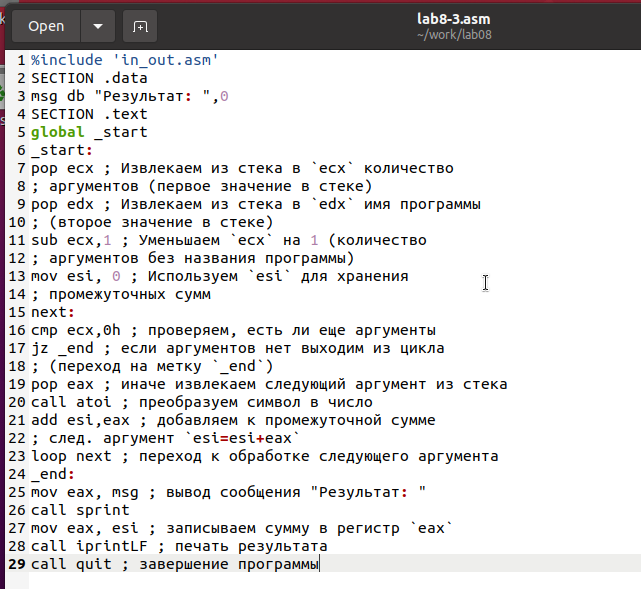


Figure 9: Программа в файле lab8-3.asm

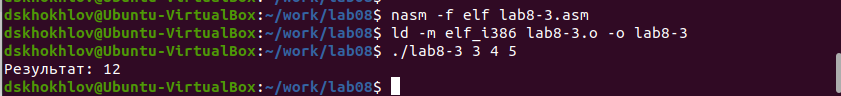


Figure 10: Запуск программы lab8-3.asm

Модифицировал код из примера 8.3 так, чтобы программа теперь выполняла вычисление произведения значений, переданных в командной строке, что отражено на иллюстрациях [[11](#fig:011)] и [[12](#fig:012)].

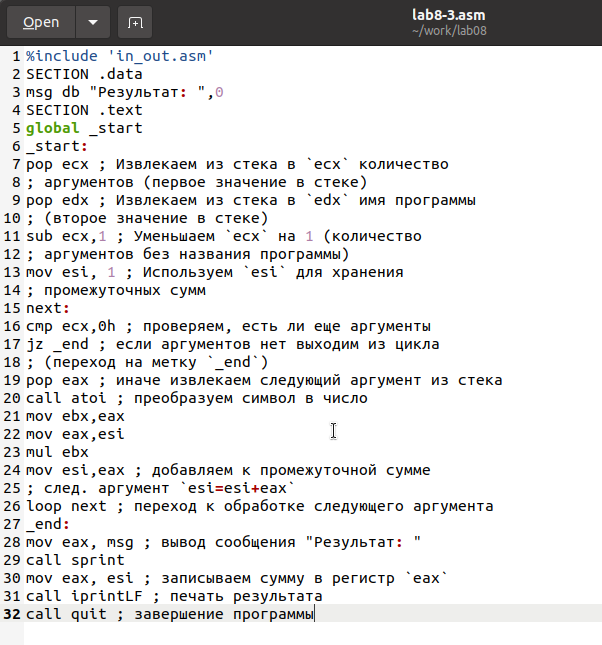


Figure 11: Программа в файле lab8-3.asm

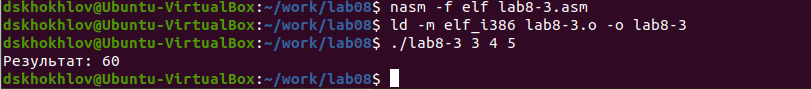


Figure 12: Запуск программы lab8-3.asm

## 2.1 Самостоятельное задание

Напишите программу, которая находит сумму значений функции для , т.е. программа должна выводить значение . Значения передаются как аргументы. Вид функции 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 .(рис. [[13](#fig:013)]) (рис. [[14](#fig:014)])

для варианта 14

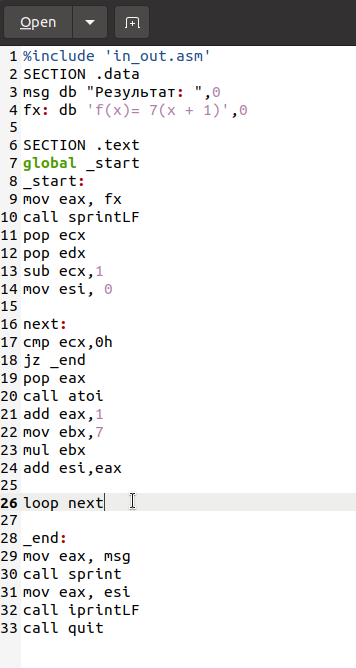


Figure 13: Программа в файле lab8-4.asm

Для проверки я запустил сначала с одним аргументом. Так, при подстановке

Затем подал несколько аргументов и получил сумму значений функции.

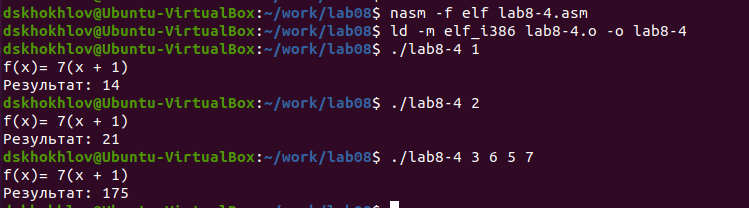


Figure 14: Запуск программы lab8-4.asm

# 3 Выводы

Освоили работы со стеком, циклом и аргументами на ассемблере nasm.

# 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).