**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 8.**

дисциплина: *Архитектура компьютеров*

Студент: Подхалюзина Виолетта Михайловна

Группа: НКАбд-04-24

**МОСКВА**

2024 г.

Оглавление

[1 Цель работы 2](#_Toc1551896633)

[2 Введение 3](#_Toc859028035)

[3 Выполнение лабораторной работы 3](#_Toc345695478)

[3.1 Начало работы 4](#_Toc1864924682)

[3.2 Самостоятельная работа 9](#_Toc1957175182)

[4 Контрольные вопросы для самопроверки 10](#_Toc1558872490)

[5 Список литературы 11](#_Toc63600490)

# Цель работы

Приобрести навыки написания программ с использованием циклов и обработкой аргументов командной строки.

# 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.

Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается.

Для стека существует две основные операции:

• добавление элемента в вершину стека (push);

• извлечение элемента из вершины стека (pop).

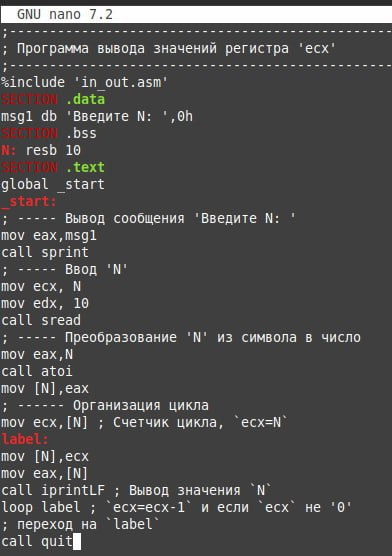
# Выполнение лабораторной работы

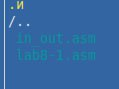
# Начало работы

Я создаю каталог для программ лабораторной работы №8, перехожу в него и создаю файл lab8-1.asm



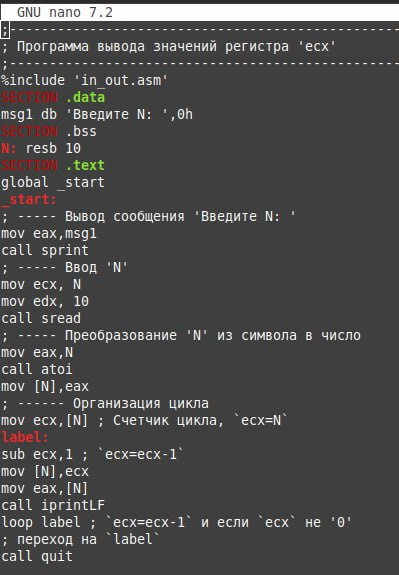
Далее я ввела в файл lab8-1.asm текст программы из листинга 8.1. Создала исполняемый файл и проверила его работу

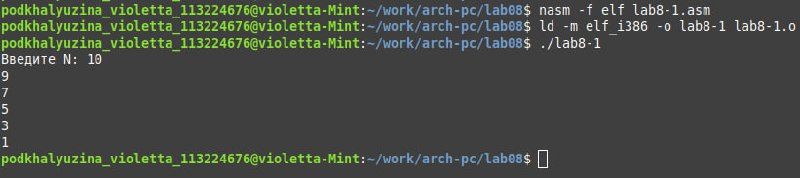




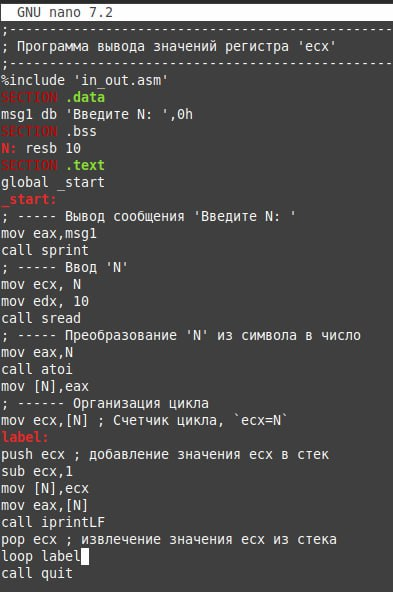


Далее я изменила текст программы, добавив изменение значение регистра ecx в цикле. Создала исполняемый файл и проверила его работу

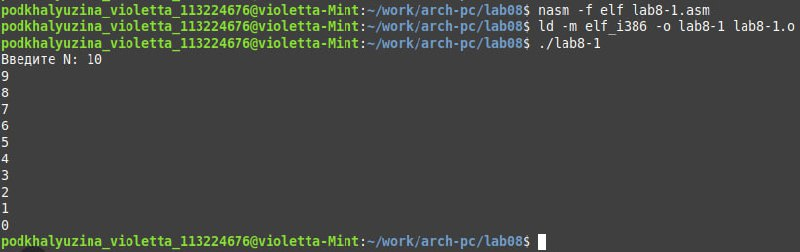


Можем заметить, что теперь регистр ecx на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое.

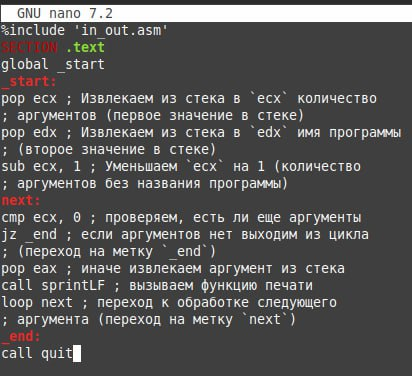
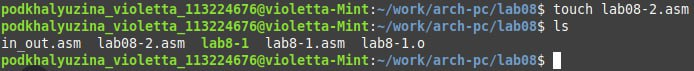
После я внесла изменения в текст программы, добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop

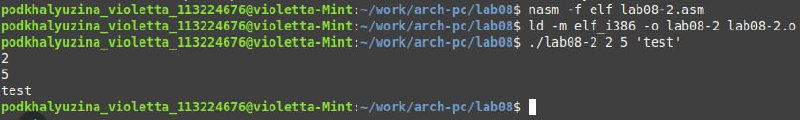


Создала исполняемый файл и проверила его работу

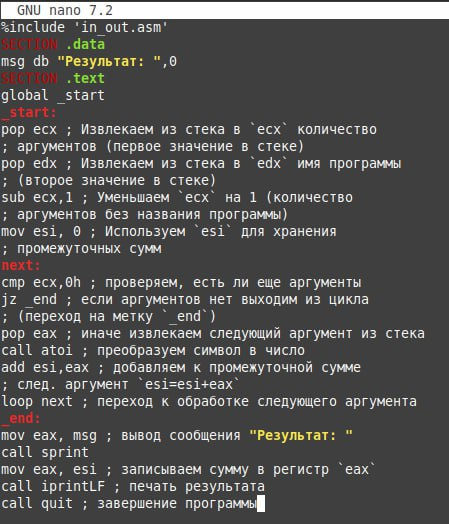
После этого мы можем сказать, что количество итераций совпадает введенному N, однако произошло смещение выводимых чисел на -1

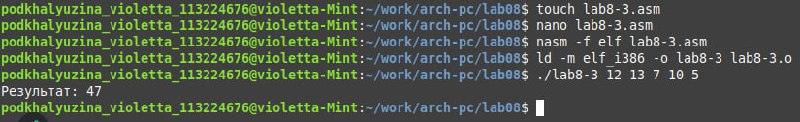
Я создала файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и ввела в него текст программы из листинга 8.2. Создала исполняемый файл и запустила его, указав аргументы

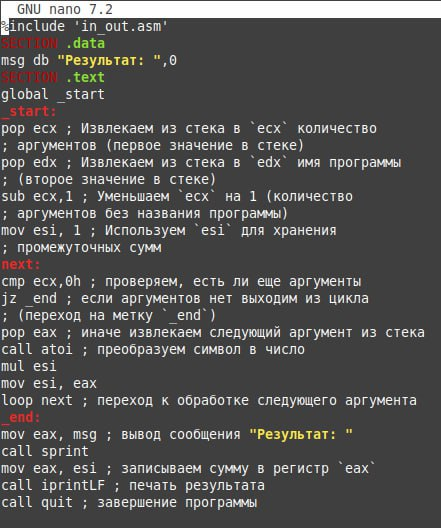


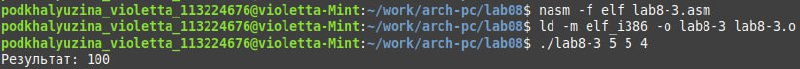


Создала файл lab8-3.asm в каталоге ~/work/archpc/lab08 и ввела в него текст программы из листинга 8.3. Создала исполняемый файл и запустила его, указав аргументы

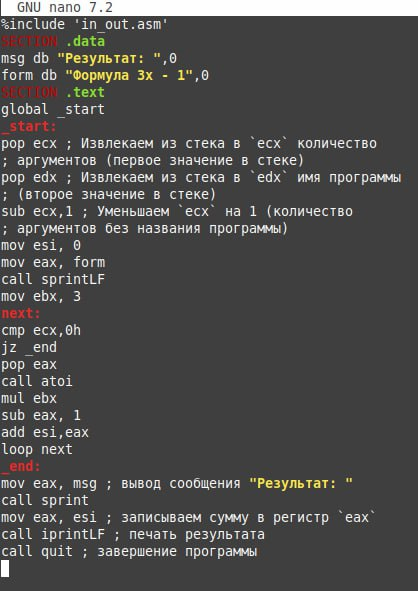


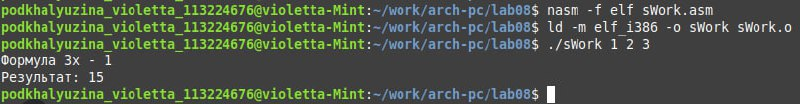
Изменила текст программы из листинга 8.3 для вычисления произведения аргументов командной строки





# 3.2 Самостоятельная работа





Вывод: в результате выполнения данной лабораторной работы я приобрела навыки написания программ с использованием циклов а также научился обрабатывать аргументы командной строки.

# Контрольные вопросы для самопроверки

1. Команда loop используется для организации циклов в ассемблере. Она уменьшает значение регистра ecx на единицу, после чего проверяет его значение. Если ecx не равен нулю, выполнение программы переходит на указанную метку, продолжая цикл. Если ecx равен нулю, выполнение передается следующей инструкции после команды loop.

2. Цикл можно организовать без использования команды loop, применяя команды условного перехода, такие как cmp и jnz.

3. Стек — это структура данных, работающая по принципу LIFO. Основные операции: добавление элемента в стек (push) и извлечение элемента из стека (pop). Стек используется для хранения адресов возврата, передачи аргументов, локальных переменных и временных данных.

4. Данные извлекаются из стека в обратном порядке их добавления (по принципу LIFO). Команда push уменьшает указатель стека esp, добавляя элемент, а команда pop увеличивает указатель esp, извлекая элемент.



# 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/Learningbash-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. — БХВПетербург, 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).