**Ответ по лабораторной работе №8**

Чжан Сюйтун

**Содержание**

1. [Цель работы](#_bookmark0) 5
2. [Теоретическое введение](#_bookmark1) 6
   1. [Организация стека](#_bookmark2) . . . . . . . . . . . . . . . . . . . . . . . . . . 6
   2. [Инструкции организации циклов](#_bookmark3) . . . . . . . . . . . . . . . . . . 6
3. [Лабораторной работы](#_bookmark4) 8
   1. [Реализация циклов в NASM](#_bookmark5) . . . . . . . . . . . . . . . . . . . . . 8
   2. [Обработка аргументов командной строки](#_bookmark13) 13
4. [Задание для самостоятельной работы](#_bookmark21) 17
5. [Выводы](#_bookmark24) 19
6. [Список литературы](#_bookmark25) 20

**Список иллюстраций**

* 1. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
  2. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
  3. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
  4. 10
  5. 11
  6. 12
  7. 12
  8. 13
  9. 13
  10. 14
  11. 14
  12. 15
  13. 15
  14. 16
  15. 18
  16. 18

**Список таблиц**

# Цель работы

Приобретение навыков написания программ с использованием циклов и обра- боткой аргументов командной строки.

# Теоретическое введение

## Организация стека

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 хра- нится в регистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При по- мещении 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).

## Инструкции организации циклов

Для организации циклов существуют специальные инструкции. Для всех ин- струкций максимальное количество проходов задаётся в регистре ecx. Наиболее

простой является инструкция loop. Она позволяет организовать безусловный цикл, типичная структура.

Иструкция loop выполняется в два этапа. Сначала из регистра ecx вычитает- ся единица и его значение сравнивается с нулём. Если регистр не равен нулю, то выполняется переход к указанной метке. Иначе переход не выполняется и управление передаётся команде, которая следует сразу после команды loop.

# Лабораторной работы

## Реализация циклов в NASM

Создайте каталог для программам лабораторной работы № 8, перейдите в него и создайте файл lab8-1.asm:

mkdir ~/work/arch-pc/lab08 cd ~/work/arch-pc/lab08 touch lab8-1.asm

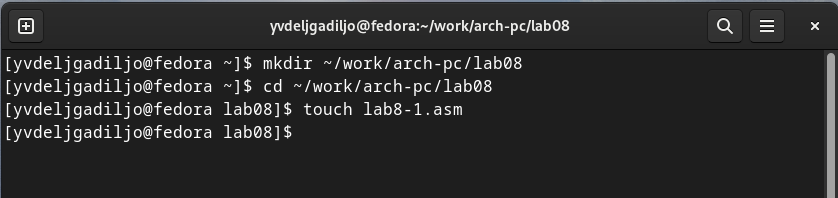


Рис. 3.1:

При реализации циклов в NASM с использованием инструкции loop необходи- мо помнить о том, что эта инструкция использует регистр ecx в качестве счетчика и на каждом шаге уменьшает его значение на единицу. В качестве примера рас- смотрим программу, которая выводит значение регистра ecx. Внимательно изу- чите текст программы (Листинг 8.1).Введите в файл lab8-1.asm текст программы из листинга 8.1. Создайте исполняемый файл и проверьте его работу.

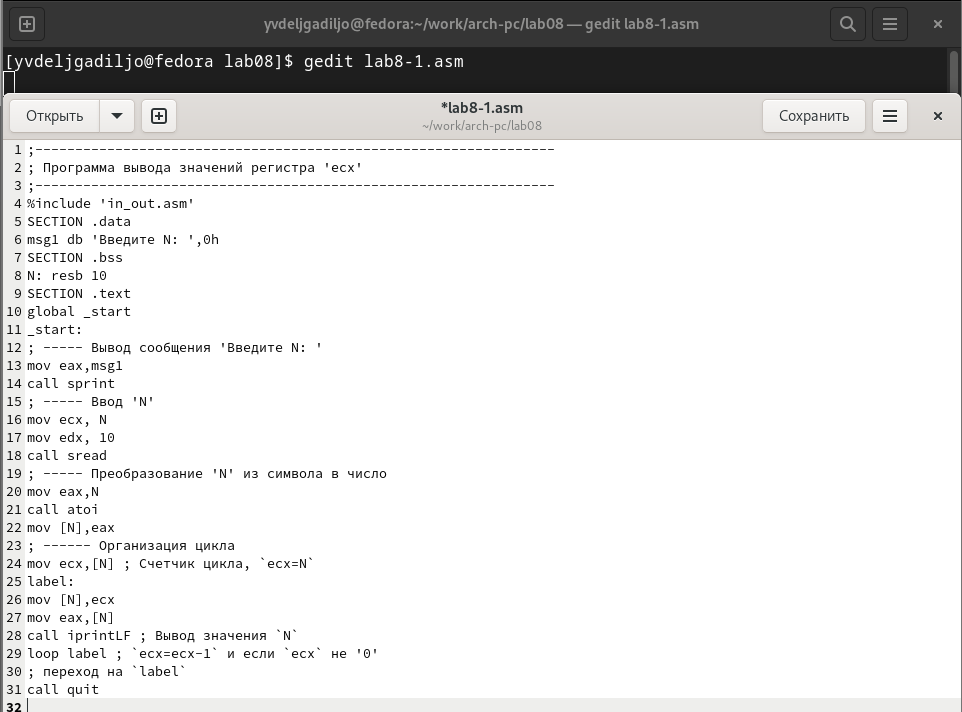


Рис. 3.2:

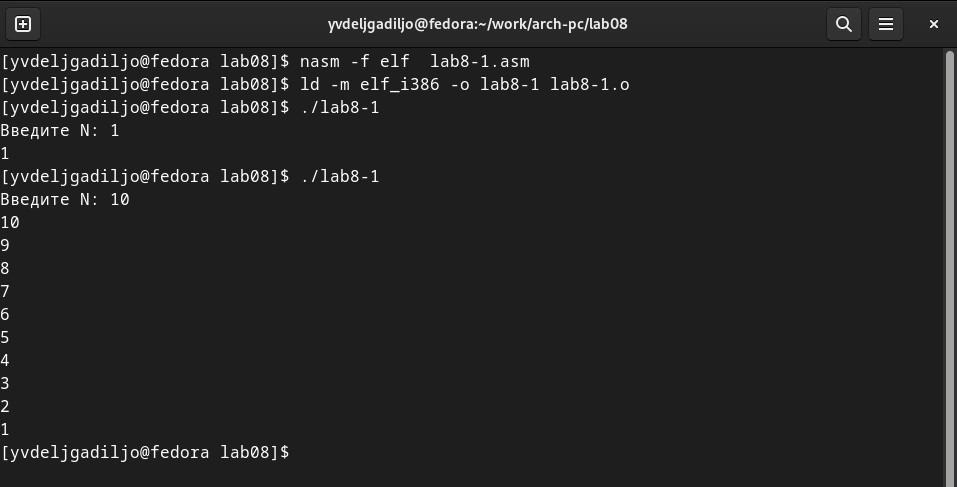


Рис. 3.3:

Данный пример показывает, что использование регистра ecx в теле цилка loop

может привести к некорректной работе программы. Измените текст программы добавив изменение значение регистра ecx в цикле:label:

sub ecx,1 ; ‘ecx=ecx-1‘ mov [N],ecx

mov eax,[N]

call iprintLFloop label

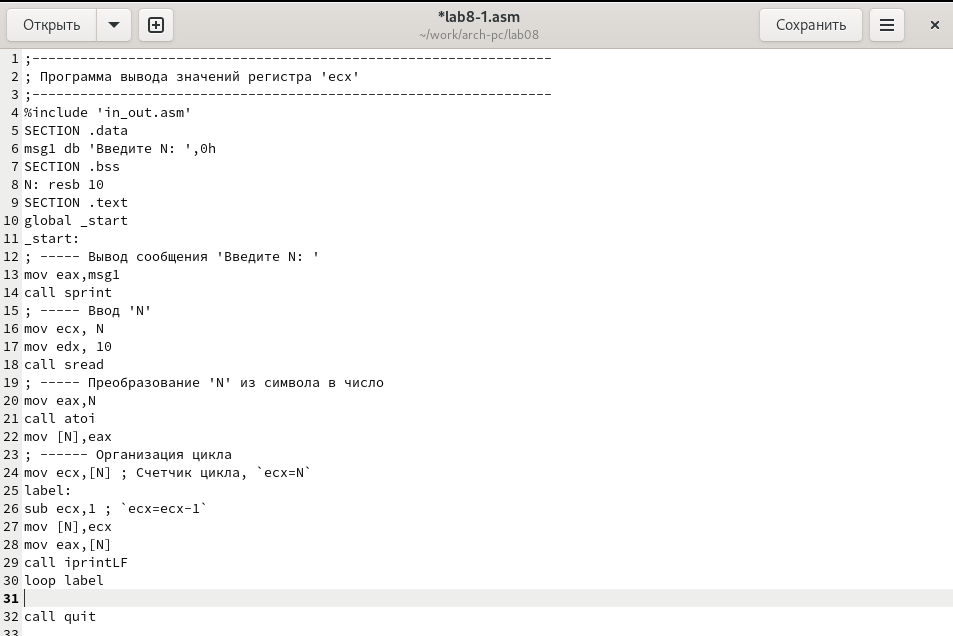


Рис. 3.4:

Создайте исполняемый файл и проверьте его работу.

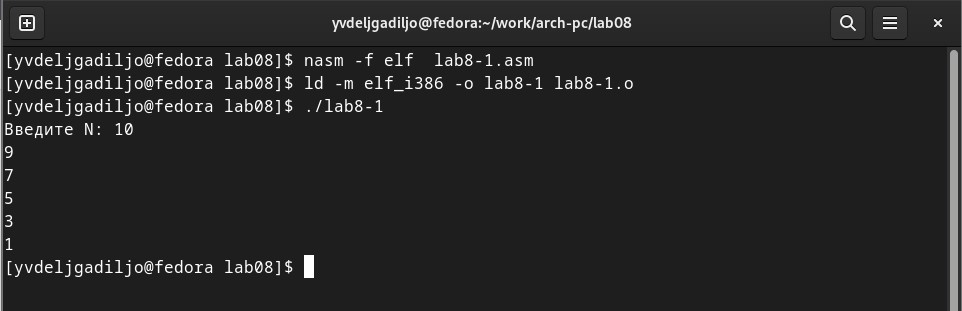


Рис. 3.5:

Какие значения принимает регистр ecx в цикле? Соответствует ли число про- ходов цикла значению N введенному с клавиатуры?

Нет, в данном случае N равно 10, а число проходов цикла равно 5.

Внесите изменения в текст программы добавив команды push и pop (добав- ления в стек и извлечения из стека) для сохранения значения счетчика цикла loop:

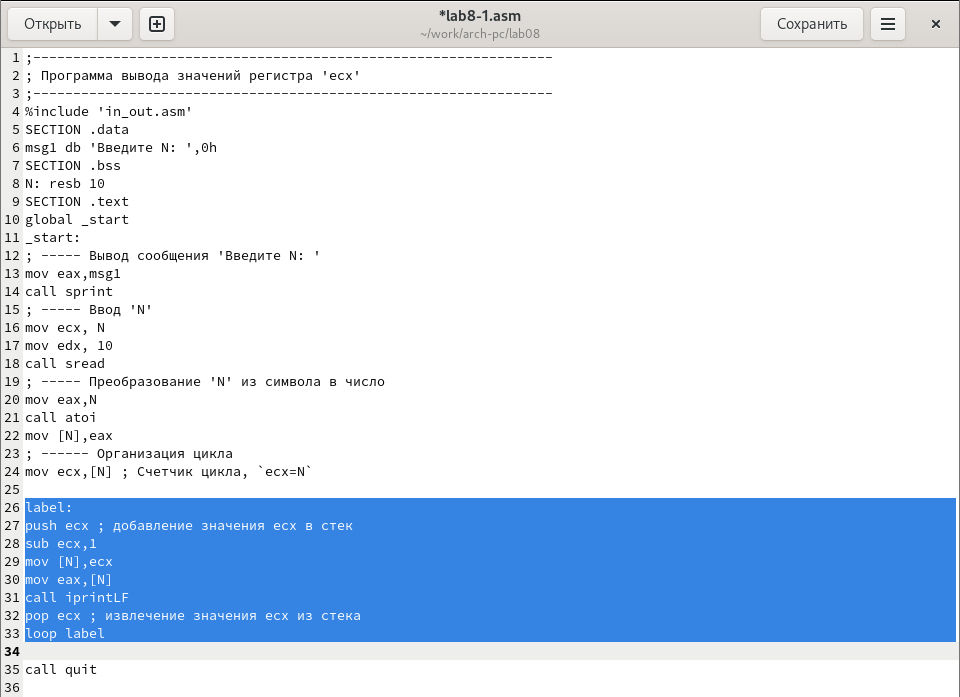


Рис. 3.6:

Создайте исполняемый файл и проверьте его работу.

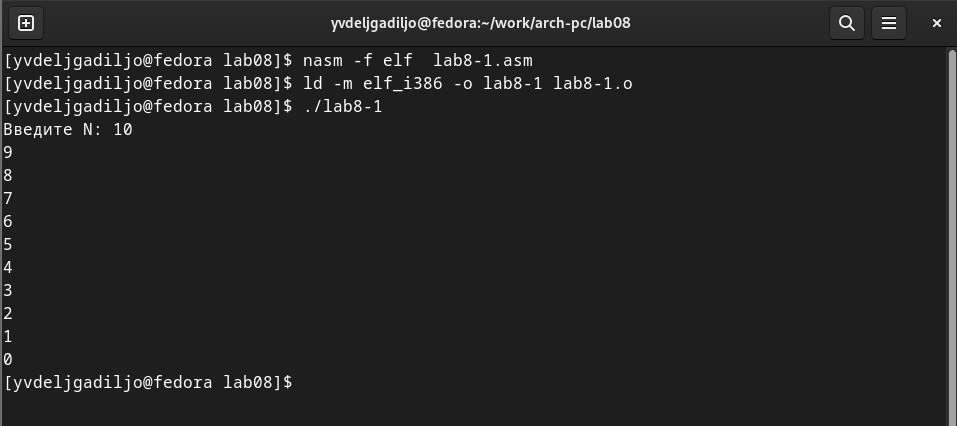


Рис. 3.7:

Соответствует ли в данном случае число проходов цикла значению N введен-

ному с клавиатуры? Да.

## Обработка аргументов командной строки

Создайте файл lab8-2.asm в каталоге ~/work/arch-pc/lab08

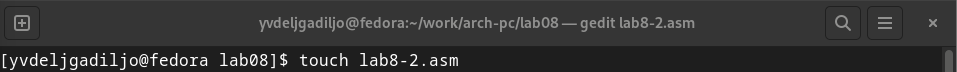


Рис. 3.8:

Bведите в него текст программы из листинга 8.2.

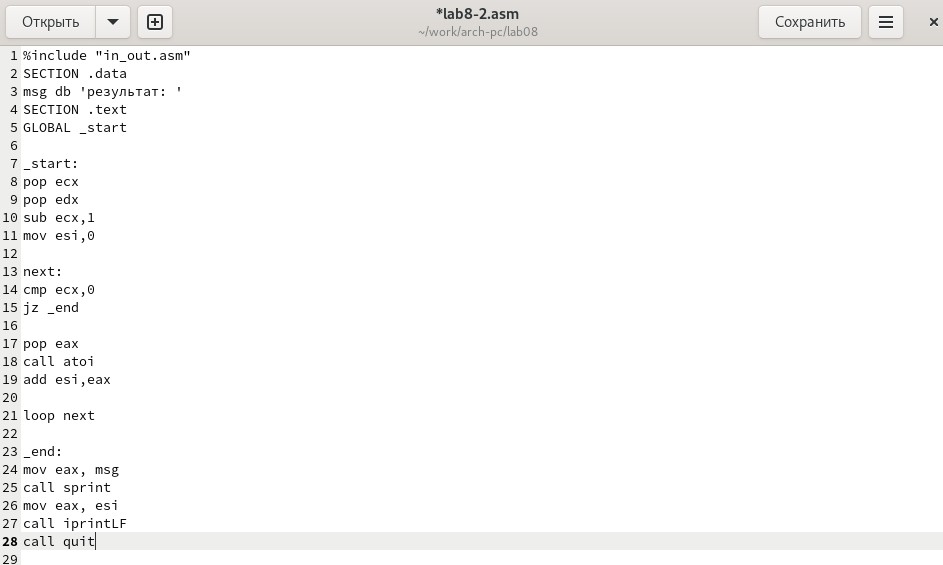


Рис. 3.9:

Создайте исполняемый файл и запустите его, указав аргументы: user@dk4n31:~$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'

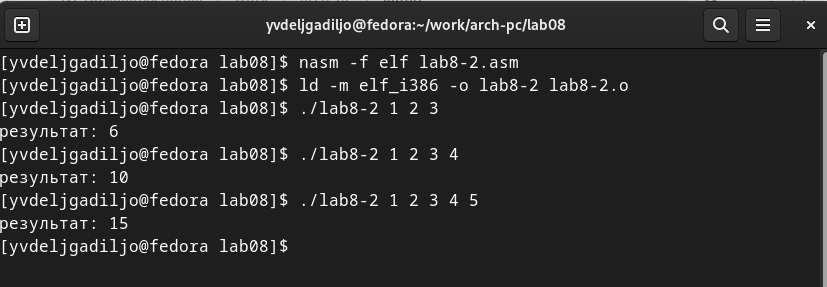


Рис. 3.10:

Создайте файл lab8-3.asm в каталоге ~/work/archpc/lab08 и введите в него текст программы из листинга 8.3.

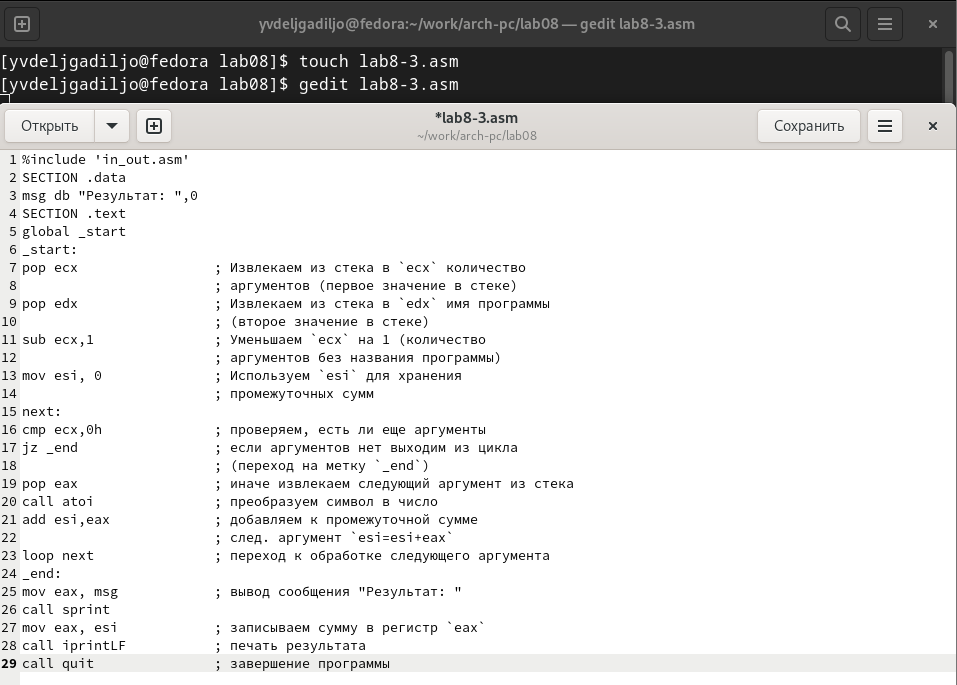


Рис. 3.11:

Создайте исполняемый файл и запустите его, указав аргументы. Пример ре- зультата работы программы:

user@dk4n31:~$ ./main 12 13 7 10 5

Результат: 47 user@dk4n31:~$

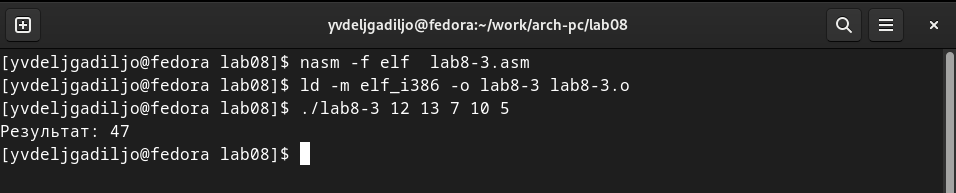


Рис. 3.12:

Измените текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.

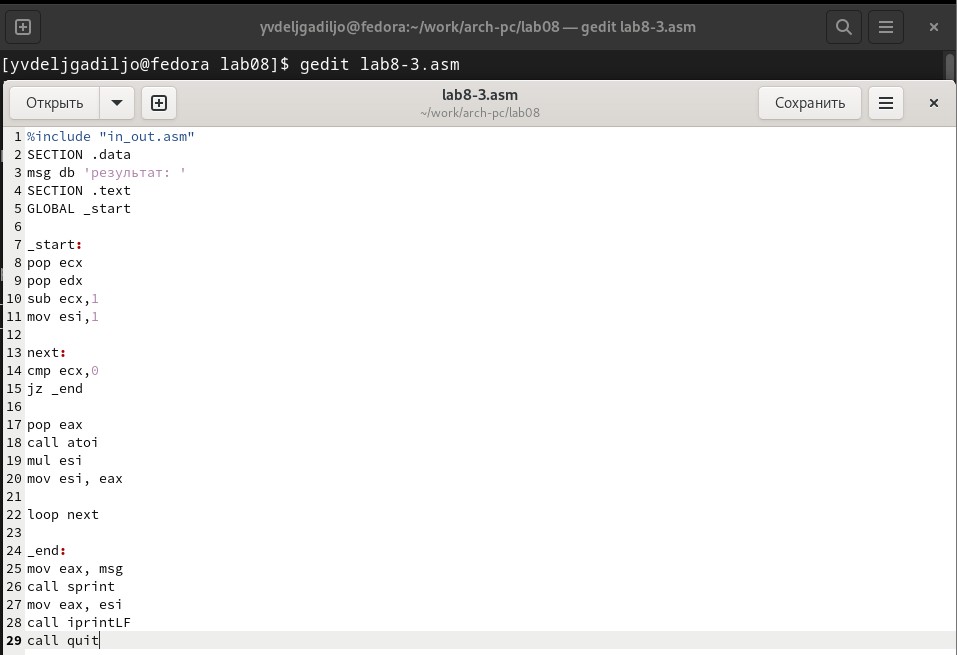


Рис. 3.13:

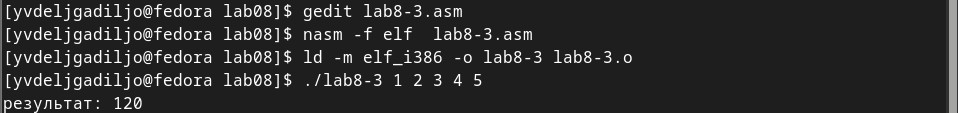


Рис. 3.14:

# Задание для самостоятельной работы

Номер варианта: 19 F(x) = 8x-3

Напишите программу, которая находит сумму значений функции f(X). Значе- ния xi передаются как аргументы. Создайте исполняемый файл и проверьте его работу на нескольких наборах.

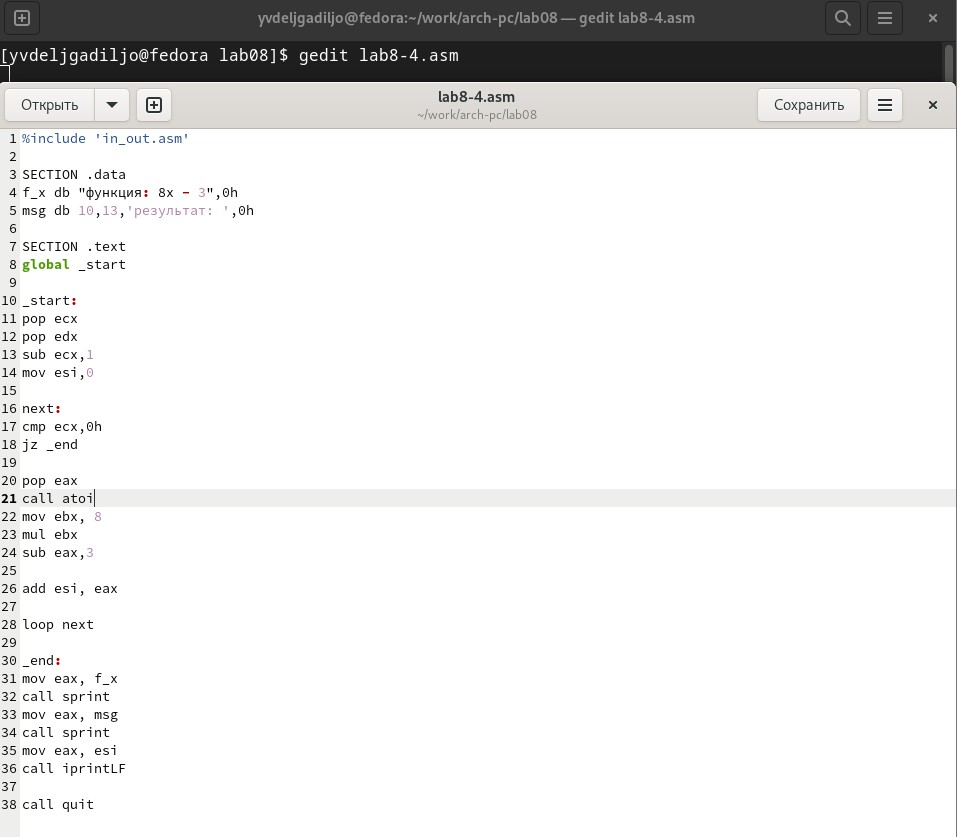


Рис. 4.1:

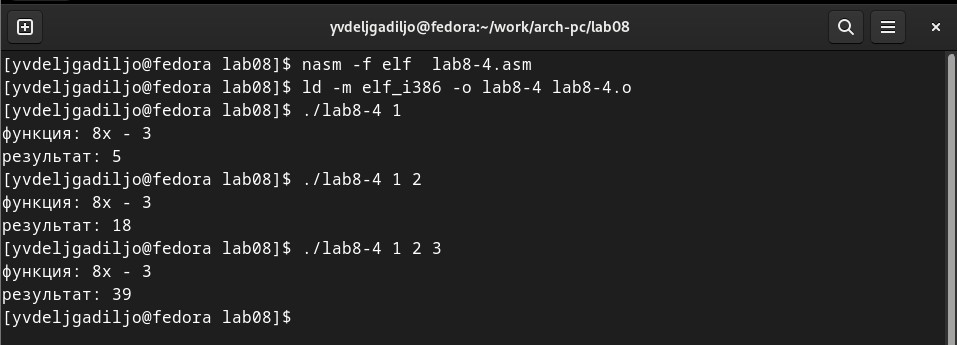


Рис. 4.2:

# Выводы

Были получены по организации циклов и работе со стеком на языке NASM.

# Список литературы

* GDB: The GNU Project Debugger. — URL: ht[tps://w](http://www.gnu.org/software/gdb/)ww[.gnu.org/software/gdb/.](http://www.gnu.org/software/gdb/)
* GNU Bash Manual. — 2016. — URL: ht[tps://w](http://www.gnu.org/software/bash/manual/)ww[.gnu.org/software/bash/manual/.](http://www.gnu.org/software/bash/manual/)
* Midnight Commander Development Center. — 2021. — URL: https://midnight- commander.org/.
* NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.
* 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.](http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658)

* Robbins A. Bash Pocket Reference. — O’Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
* The NASM documentation. — 2021. — URL: ht[tps://w](http://www.nasm.us/docs.php)ww[.nasm.us/docs.php.](http://www.nasm.us/docs.php)
* Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
* Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
* Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
* Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
* Расширенный ассемблер: NASM.— 2021.— URL: [https://w](http://www.opennet.ru/docs/RUS/nasm/)ww[.opennet.ru/docs/RUS/nasm/.](http://www.opennet.ru/docs/RUS/nasm/)
* Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
* 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.](http://www.stolyarov.info/books/asm_unix)

* Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
* Таненбаум Э., Бос Х. Современные операционн