Отчёт по лабораторной работе №6

Дисциплина: архитектура компьютера

Зиани Сид Ахмед

Содержание

# 1 Цель работы

Цель данной лабораторной работы - освоение арифметческих инструкций языка ассемблера NASM.

# 2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

# 3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти.

Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: mov ax,bx. Непосредственная адресация – значение операнда задается непосредственно в команде, Например: mov ax,2. Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию. Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно

# 4 Выполнение лабораторной работы

### 4.0.1 Символьные и численные данные в NASM

С помощью утилиты mkdir создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6

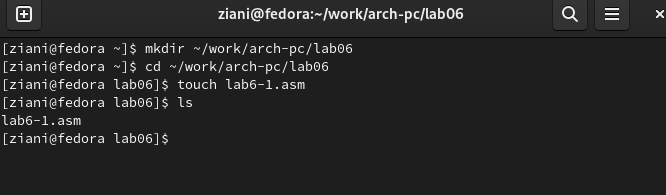


Figure 1: Перехожу в созданный каталог с помощью утилиты cd. С помощью утилиты touch создаю файл lab6-1.asm.

Копирую в текущий каталог файл in\_out.asm с помощью утилиты cp, т.к. он будет использоваться в других программах

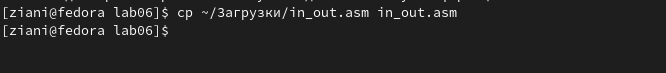


Figure 2: Копирование in\_out.asm.

Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения регистра eax

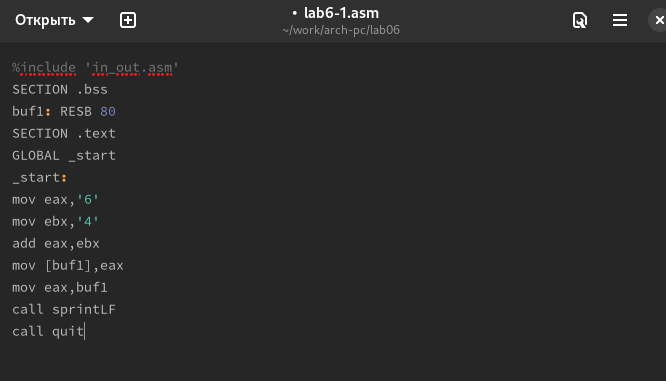


Figure 3: Открываю файл lab6-1.asm с помощью текстового редактора nano.

Создаю исполняемый файл программы и запускаю его. Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

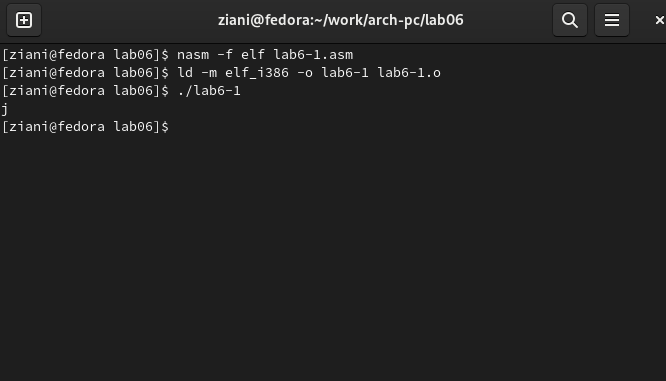


Figure 4: Запуск исполняемого файла.

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4.

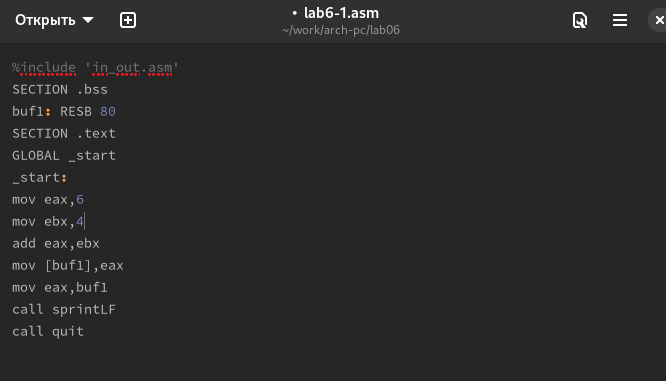


Figure 5: Редактирование файла.

Создаю новый исполняемый файл программы и запускаю его. Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

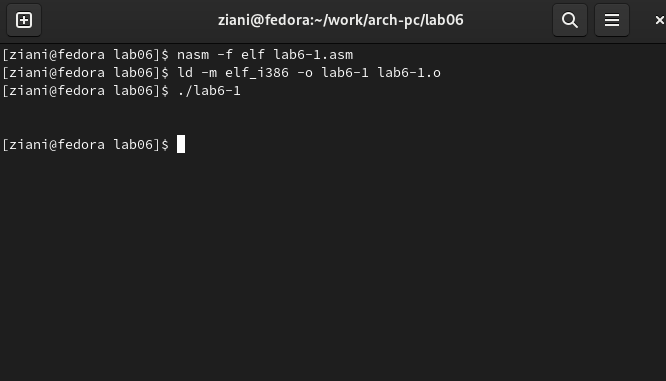


Figure 6: Запуск исполняемого файла.

Создаю новый файл lab6-2.asm с помощью утилиты touch.



Figure 7: Создание файла.

Ввожу в файл текст другойпрограммы для вывода значения регистра eax.

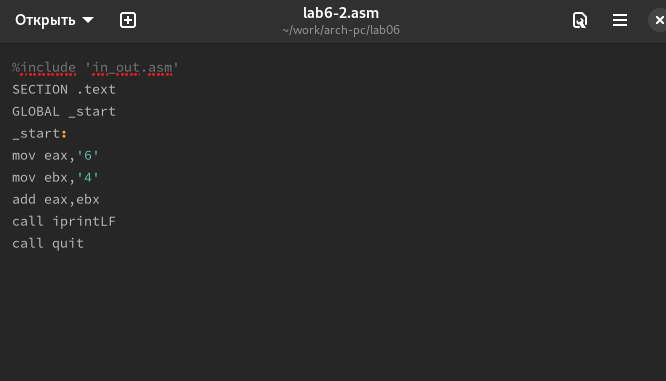


Figure 8: Редактирование файла.

Создаю и запускаю исполняемый файл lab6-2. Теперь вывод число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.

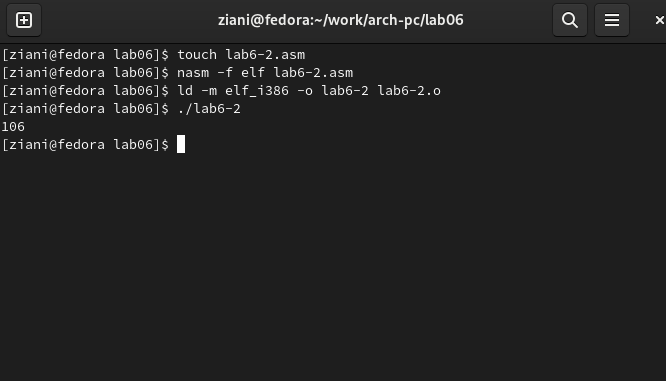


Figure 9: Запуск исполняемого файла.

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4.

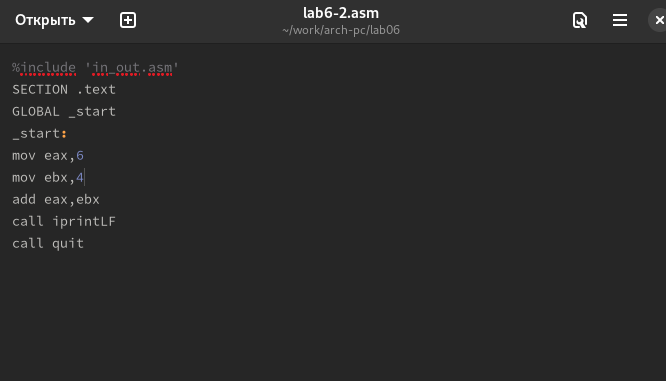


Figure 10: Редактирование файла.

Создаю и запускаю новый исполняемый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

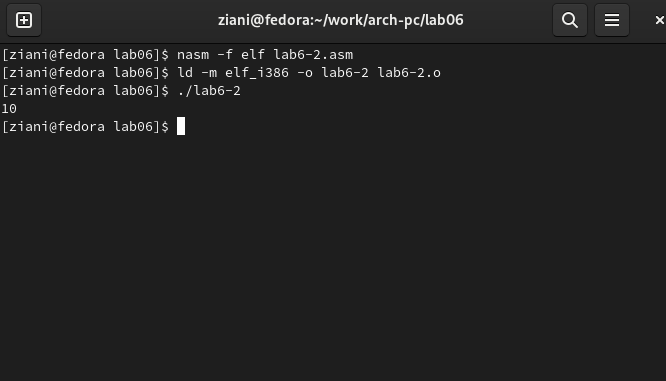


Figure 11: Запуск исполняемого файла.

Заменяю в тексте программы функцию iprintLF на iprint.

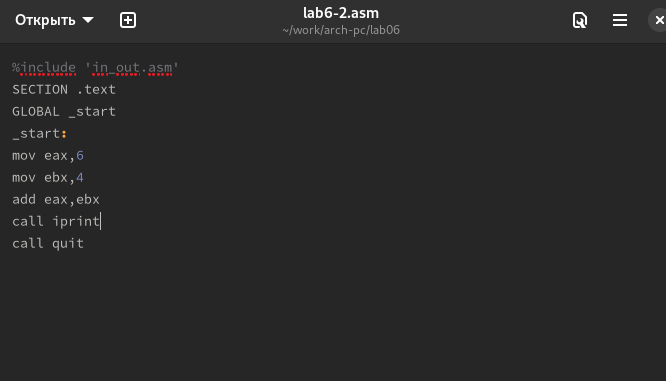


Figure 12: Редактирование файла.

Создаю и запускаю новый исполняемый файл. Вывод не изменился, потому что символ переноса строки не отображался, когда программа исполнялась с функцией iprintLF, а iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF.

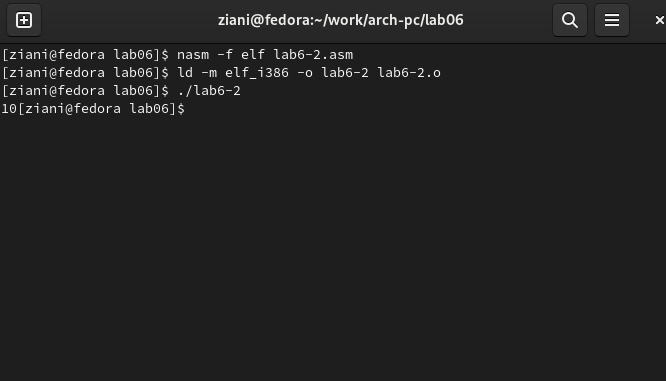


Figure 13: Запуск исполняемого файла.

### 4.0.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью утилиты touch. И проверяю с помощью ls.

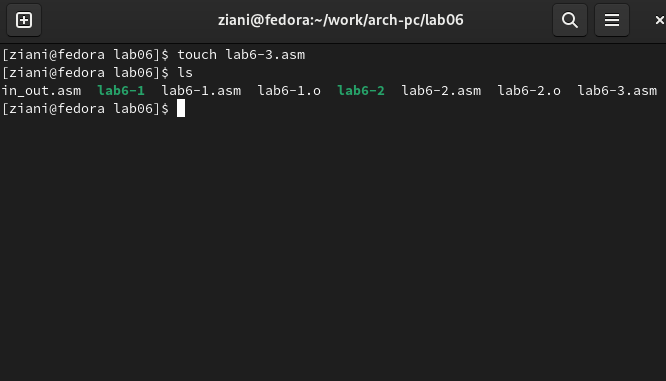


Figure 14: Создание и проверка файла.

Ввожу в созданный файл текст программы для вычисления значения выражения f(x) = (5 \* 2 + 3)/3

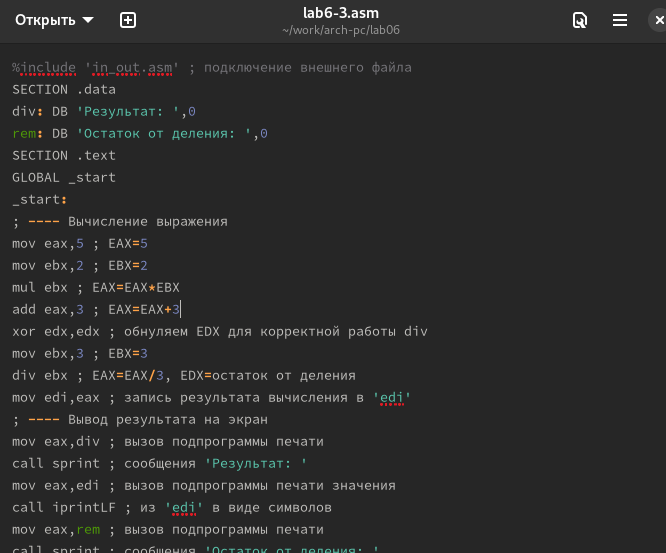


Figure 15: Редактирование файла.

Создаю исполняемый файл и запускаю его.

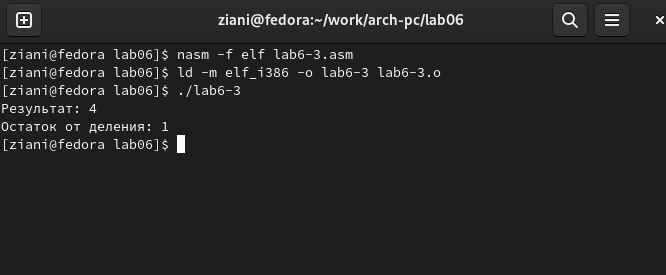


Figure 16: Запуск исполняемого файла.

Изменяю программу так, чтобы она вычисляла значение выражения f(x) = (4 \* 6 + 2)/5

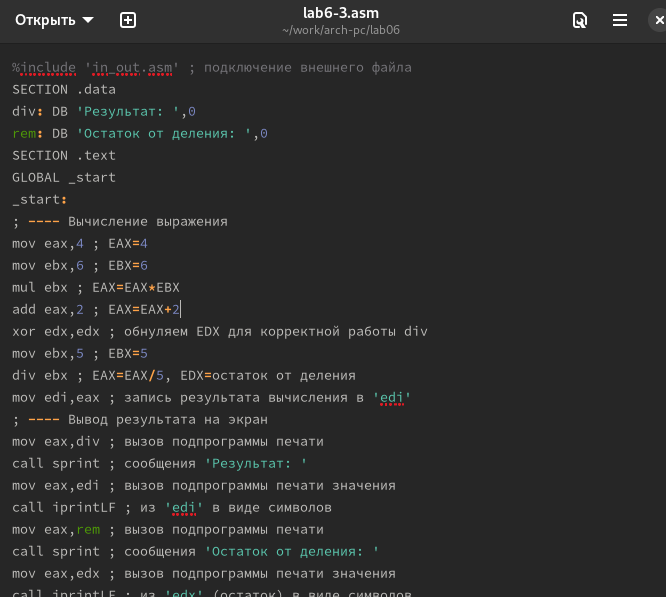


Figure 17: Изменение программы.

Создаю и запускаю новый исполняемый файл.

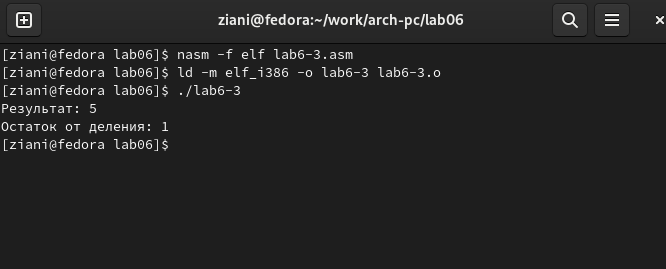


Figure 18: Запуск исполняемого файла.

Создаю файл variant.asm с помощью утилиты touch.



Figure 19: Создание файла.

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета.

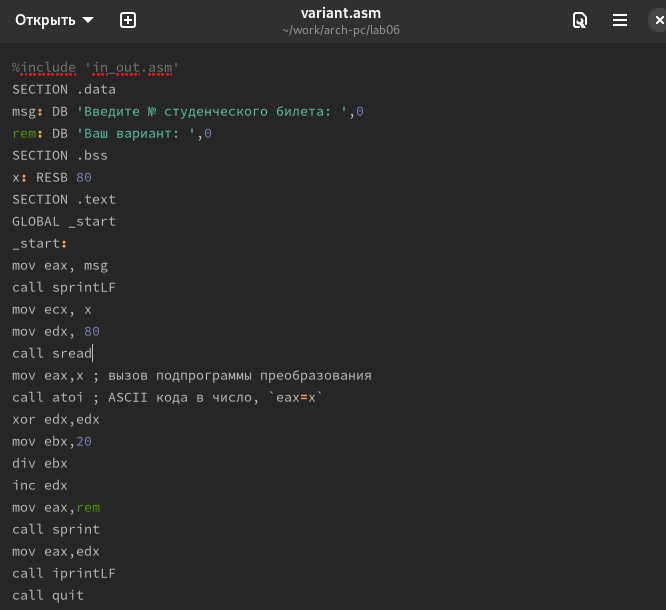


Figure 20: Редактирование файл.

Создаю и запускаю исполняемый файл. Ввожу номер своего студенческого билета с клавиатуры, программа вывела, что мой вариант 19.

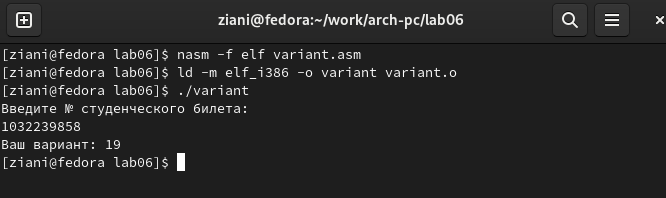


Figure 21: Запуск исполняемого файл.

### 4.0.3 Ответы на вопросы по программе

1. За вывод сообщения “Ваш вариант” отвечают строки кода: mov eax, rem call sprint
2. Инструкция mov ecx, x используется, чтобы положить адрес вводимой строки x в регистр ecx mov edx, 80 - запись в регистр edx длины вводимой строки call sread - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры.
3. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax
4. За вычисления варианта отвечают строки: xor edx,edx ; обнуление edx для корректной работы div mov ebx,20 ; ebx = 20 div ebx ; eax = eax/20, edx - остаток от деления inc edx ; edx = edx + 1
5. При выполнении инструкции div ebx остаток от деления записывается в регистр edx
6. Инструкция inc edx увеличивает значение регистра edx на 1
7. За вывод на экран результатов вычислений отвечают строки: mov eax,edx call iprintLF

# 5 Выполнение заданий для самостоятельной работы

Создаю файл lab6-4.asm с помощью утилиты touch.

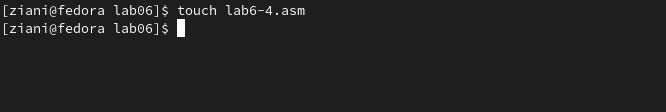


Figure 22: Создание файла.

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения X в степени (1/3\*x+21). Это выражение было под вариантом 19.

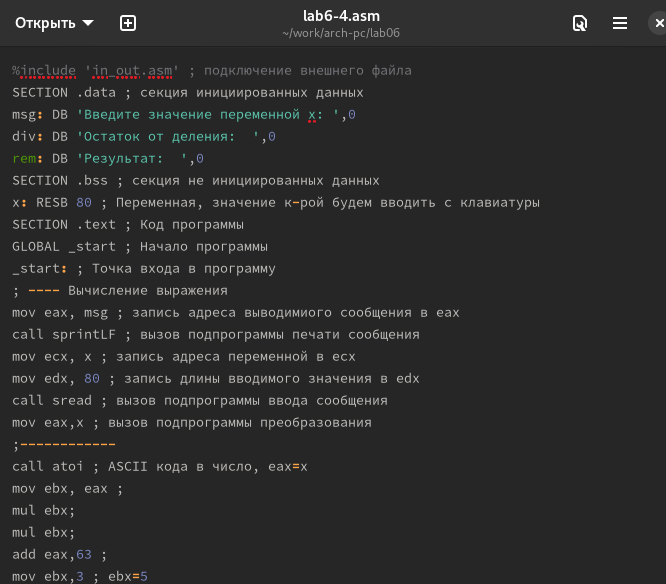


Figure 23: Написание программ.

Создаю и запускаю исполняемый файл. При вводе значения 3, вывод 30.

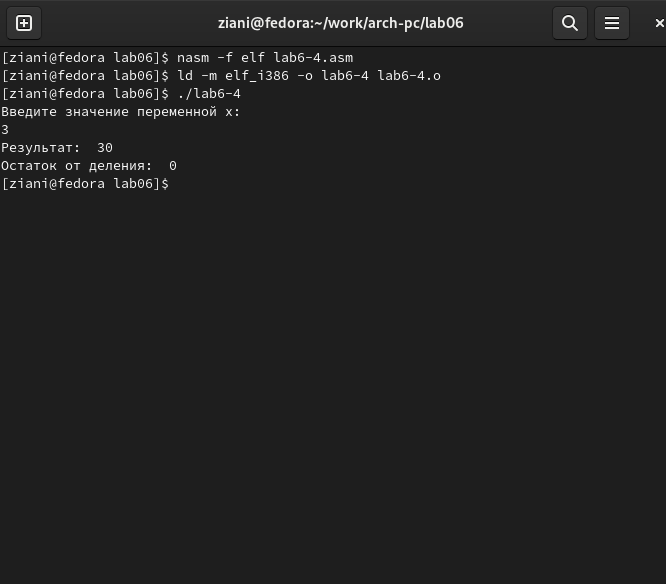


Figure 24: Запуск исполняемого файла.

Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе. Программа отработала верно.

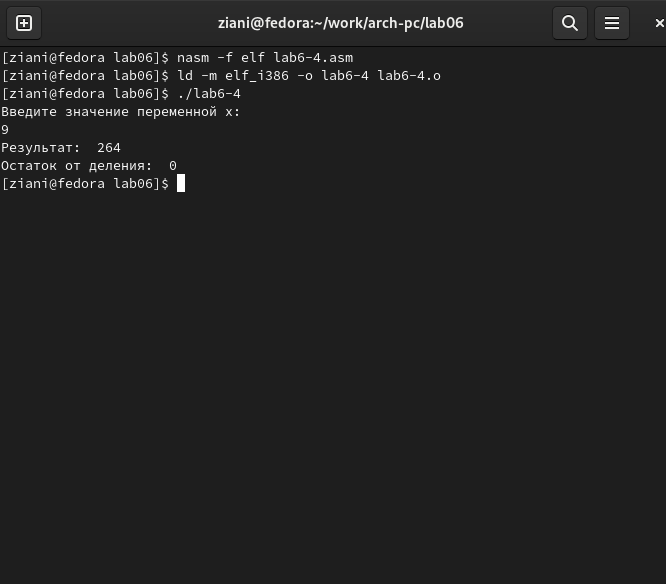


Figure 25: Запуск исполняемого файла.

# 6 Выводы

При выполнении данной лабораторной работы я освоил арифметические инструкции языка ассемблера 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).

z