Шаблон отчёта по лабораторной работе

7

Сильвен Макс Грегор Филс , НКАбд-03-22

Содержание

# 1 Цель работы

* В седьмой лабораторной работе можно будет освоить арифметические опе- рации языка ассемблера.

# 2 Выполнение лабораторной работы :

## 2.1 Символьные и численные данные в NASM :

* Здесь мы начали с создания, а затем переместились в седьмой каталог ла- боратории “~/work/arch-pc/lab07”, после чего мы создали файл “lab7-1.asm”.(рис. **¿fig:fig1?**)

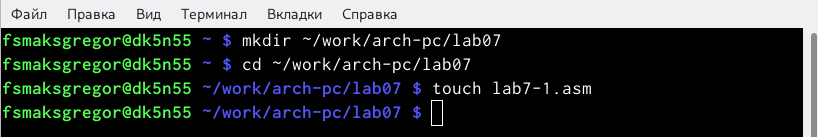


Рис. 1: Ресунок 1

* После этого мы заполнили файл .asm кодом программы, отображающей значение регистра eax. (рис. **¿fig:fig2?**)

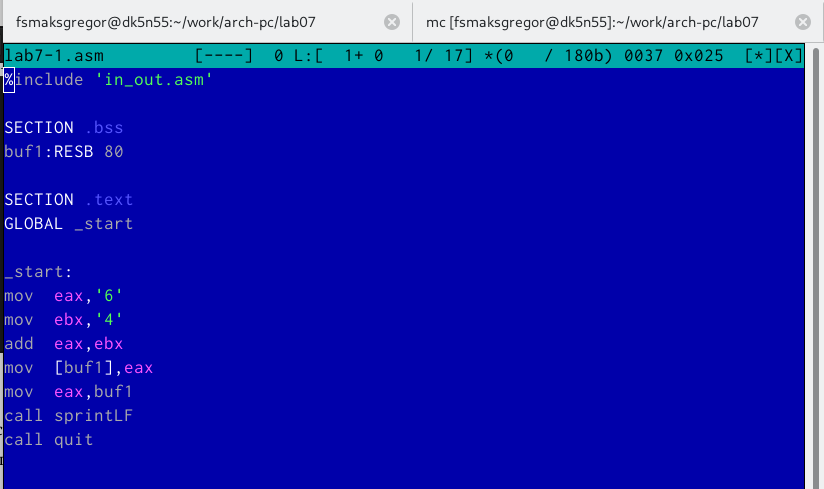


Рис. 2: Ресунок 2

* Затем мы скомпилировали файл, создали исполняемый файл и запустили программу, все это после перемещения файла in\_out.asm в тот же каталог, где находится lab7-1.asm.(рис. **¿fig:fig3?**)

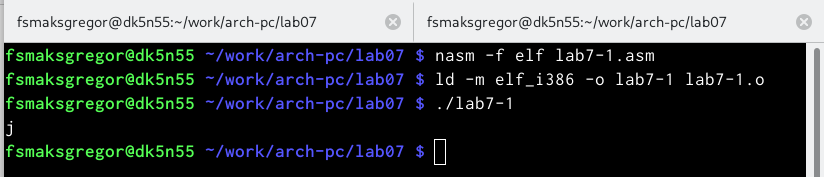


Рис. 3: Ресунок 3

* После этого мы изменили код в листинге следующим образом : mov eax,6 mov ebx,4 (рис. **¿fig:fig4?**)

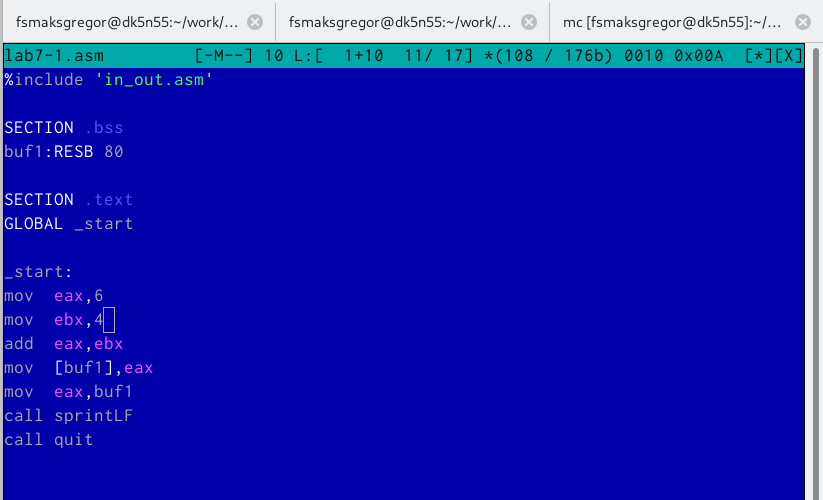


Рис. 4: Ресунок 4

* Затем мы снова скомпилировали файл и создали исполняемый файл.(рис. **¿fig:fig5?**)

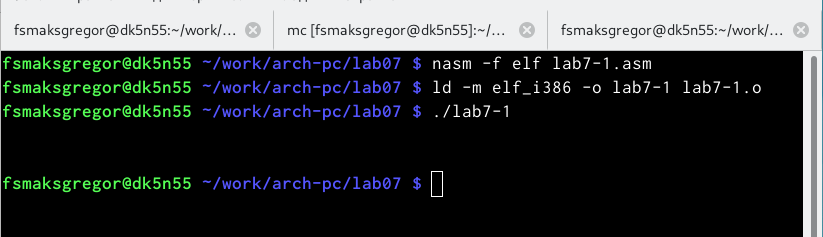


Рис. 5: Ресунок 5

* Проверив ASCII tbale символ, соответствующий коду 10 это новая строка, и мы можем сказать, что это было отображено, потому что при запуске программы она отобразила новую строку в качестве вывода.
* После этого мы создали файл lab-2.asm, в котором мы использовали под- программы, расположенные в файле in\_out.asm. (рис. **¿fig:fig6?**)

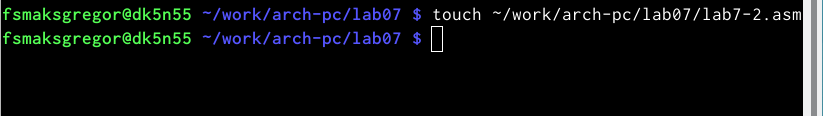


Рис. 6: Ресунок 6

* После этого мы заполнили файл необходимым кодом для вывода значения реестра с помощью подпрограммы. (рис. **¿fig:fig7?**)

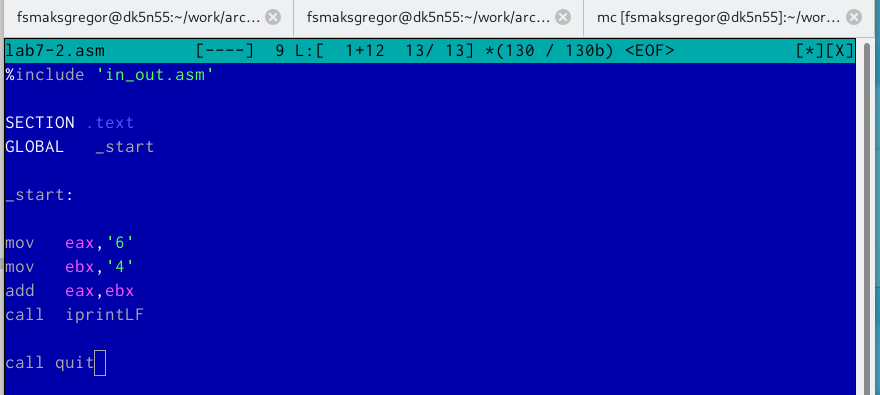


Рис. 7: Ресунок 7

* мы скомпилировали файл, создали исполняемый файл и запустили его.(рис. **¿fig:fig8?**)

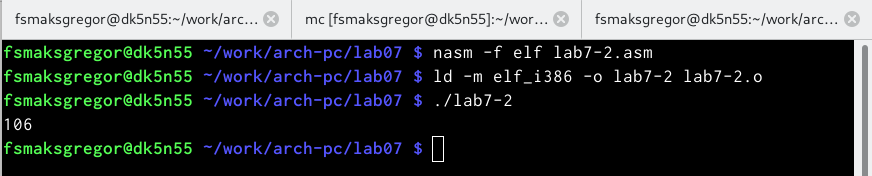


Рис. 8: Ресунок 8

* Аналогично предыдущему примеру, мы меняем символы на цифры, заменяя строки на : mov eax,6 mov ebx,4 (рис. **¿fig:fig9?**)

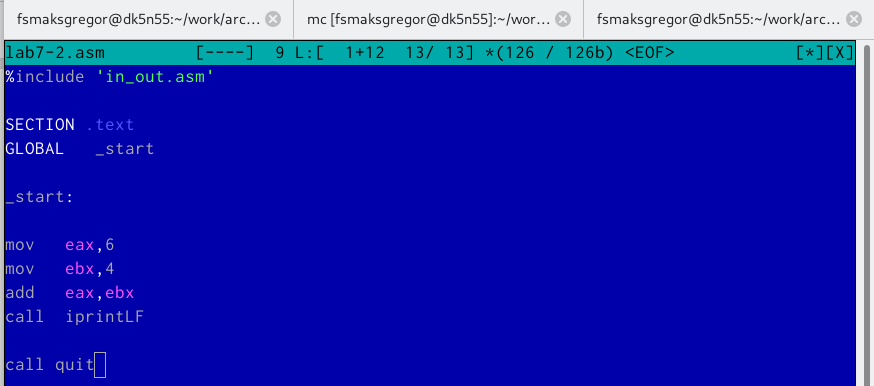


Рис. 9: Ресунок 9

* Затем мы снова скомпилировали файл и создали исполняемый файл.(рис. **¿fig:fig10?**)

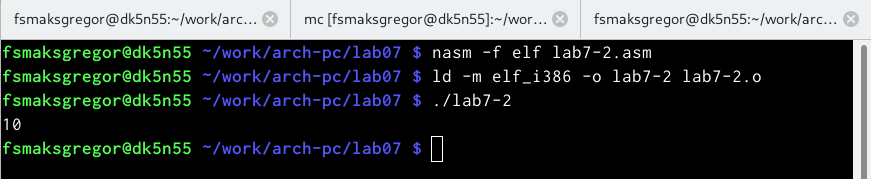
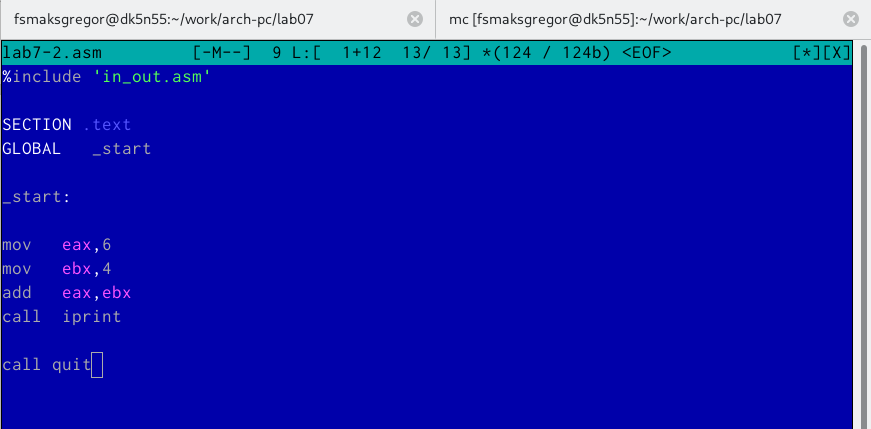
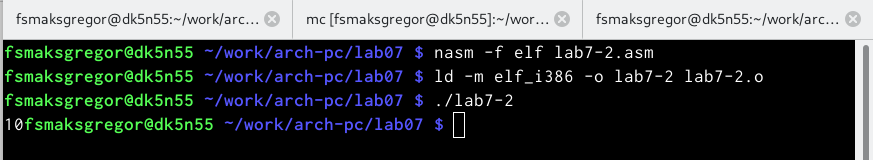


Рис. 10: Ресунок 10

* На этот раз результатом, который мы получили, действительно было добав- ление 6 и 4 который 10.

-Затем мы заменили функцию iprintLF на iprint. После этого был создан исполняемый файл, и мы запустили его. (рис. **¿fig:fig11?**) (рис. **¿fig:fig12?**)

* Когда мы использовали подпрограмму iprint, мы заметили, что вывод отли- чается от предыдущего,потому что при использовании iprint не создается новая строка после вывода.

## 2.2 Выполнение арифметических операций в NASM :

* В качестве примера выполнения арифметических операций в NASM при- ведем программу вычисления арифметического выражения 𝑓(𝑥) = (5 ∗ 2 + 3)/3
* Мы создали файл lab7-3.asm в каталоге ~/work/arch-pc/lab07.(рис. **¿fig:fig13?**)

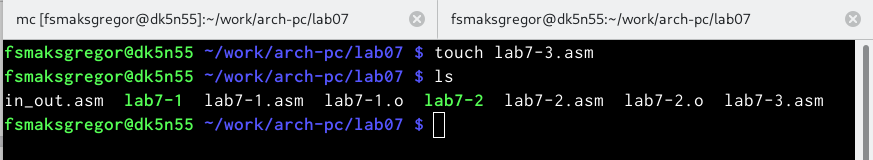


Рис. 11: Ресунок 13

* Затем мы заполнили файл необходимым кодом.(рис. **¿fig:fig14?**)

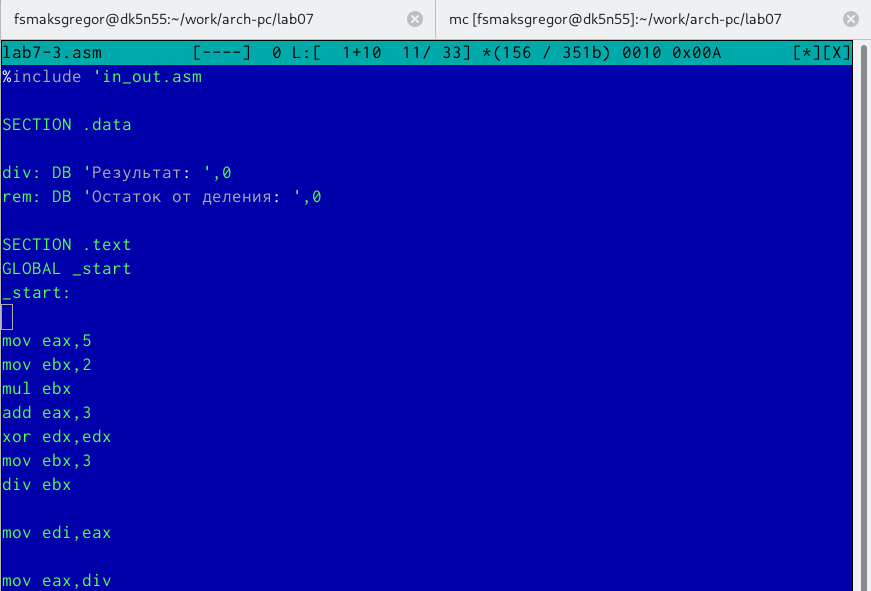


Рис. 12: Ресунок 14

* Создали исполняемый файл и запустили его.(рис. **¿fig:fig15?**)

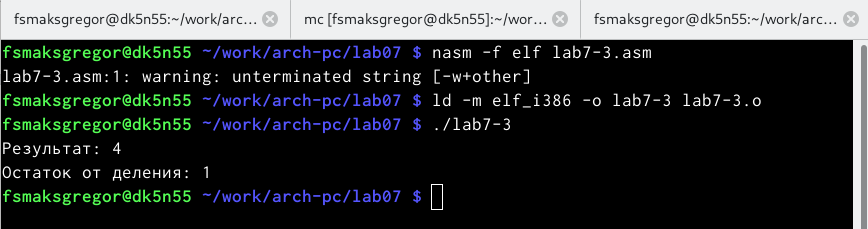


Рис. 13: Ресунок 15

* Затем мы изменили текст программы, чтобы вычислить выражение: 𝑓(𝑥) = (4 ∗ 6 + 2)/5 (рис. **¿fig:fig16?**)

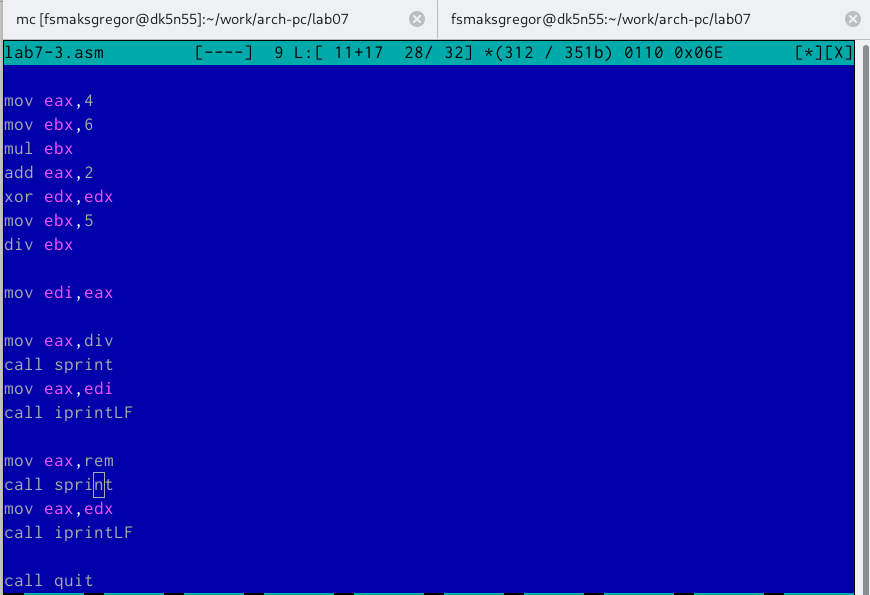


Рис. 14: Ресунок 16

* мы создали исполняемый файл и проверили его работу. (рис. **¿fig:fig17?**)

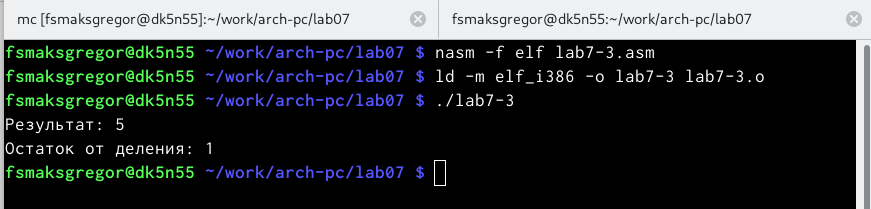


Рис. 15: Ресунок 17

* На этом шаге мы написали программу, которая может вычислить диспер- сию, которую мы получаем из номера студенческого билета.
* Мы начали с создания файла variant.asm. (рис. **¿fig:fig18?**)

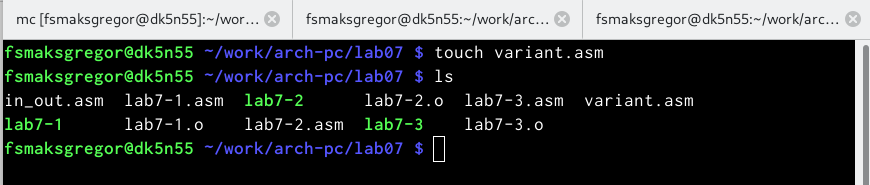


Рис. 16: Ресунок 18

* После этого мы написали код программы. (рис. **¿fig:fig19?**)

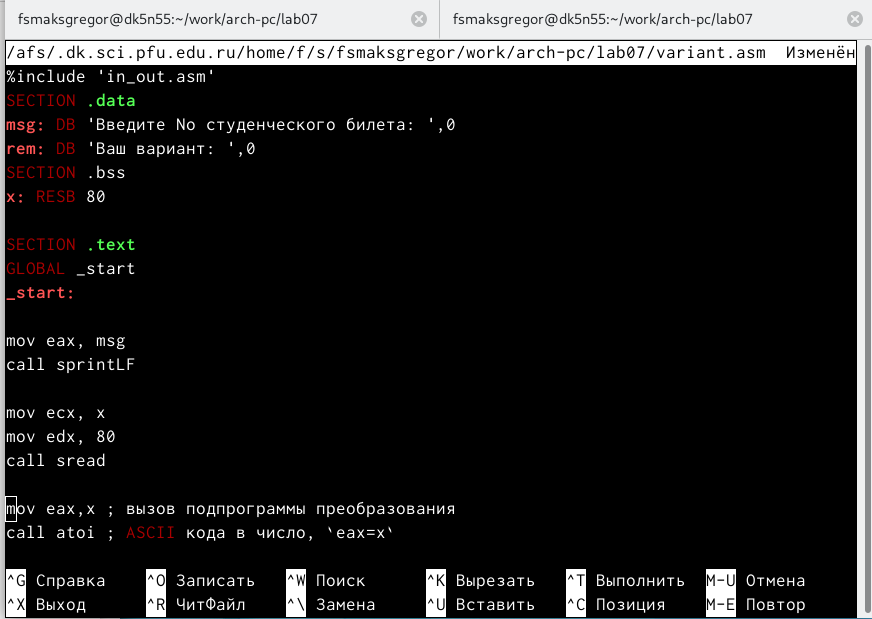


Рис. 17: Ресунок 19

* мы создали исполняемый файл и проверили его работу, и действительно, в зависимости от номера студента он генерирует номер варианта. (рис. **¿fig:fig20?**)

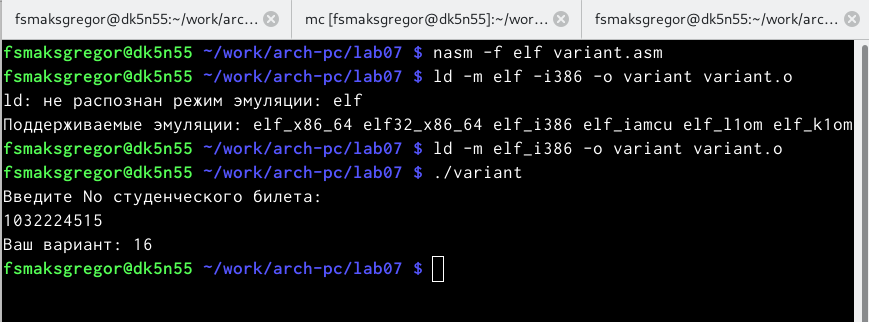


Рис. 18: Ресунок 20

## 2.3 Вопросы :

* Какие строки листинга 7.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’? **O :** rem: DB ‘Ваш вариант:’,0 mov eax,rem call sprint
* Для чего используется следующие инструкции? mov ecx, x / mov edx, 80 / call sread **O :** Эти инструкции были использованы для того, чтобы позволить пользовате- лю вводить данные.
* Для чего используется инструкция “call atoi”? **O :** Эта инструкция используется для преобразования значения x из ASCII-кода в целое число.
* Какие строки листинга 7.4 отвечают за вычисления варианта? **O :** xor edx,edx mov ebx,20 div ebx inc edx
* В какой регистр записывается остаток от деления при выполнении ин- струкции “div ebx”? **O :** Остаток был записан в регистре **edx**
* Для чего используется инструкция “inc edx”? **O :** Эта инструкция была использована для увеличения значения в регистре **edx**
* Какие строки листинга 7.4 отвечают за вывод на экран результата вычис- лений? **O :** mov eax,edx call iprintLF

## 2.4 Выводы по результатам выполнения заданий :

* В ходе лабораторной работы мы освоили выполнение арифметических опе- раций на языке ассемблера и углубились в использование подпрограммы.

# 3 Задание для самостоятельной работы :

* В этой работе нам пришлось написать программу, которая просит пользо- вателя ввести значение переменной и решить математическое выражение.
* Мой вариант : 16
* математическое выражение : (10𝑥 − 5)^2
* Итак, мы начали с создания asm-файла, в котором будет находиться наш код.(рис. **¿fig:fig21?**)

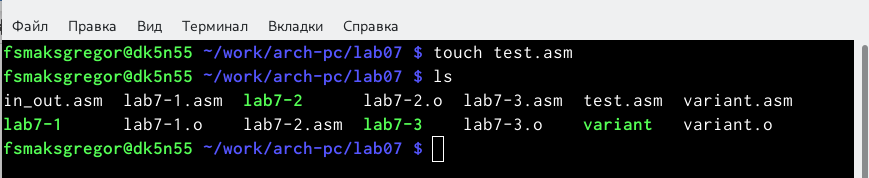


Рис. 19: Ресунок 21

* После этого мы написали код нашей программы. (рис. **¿fig:fig22?**)

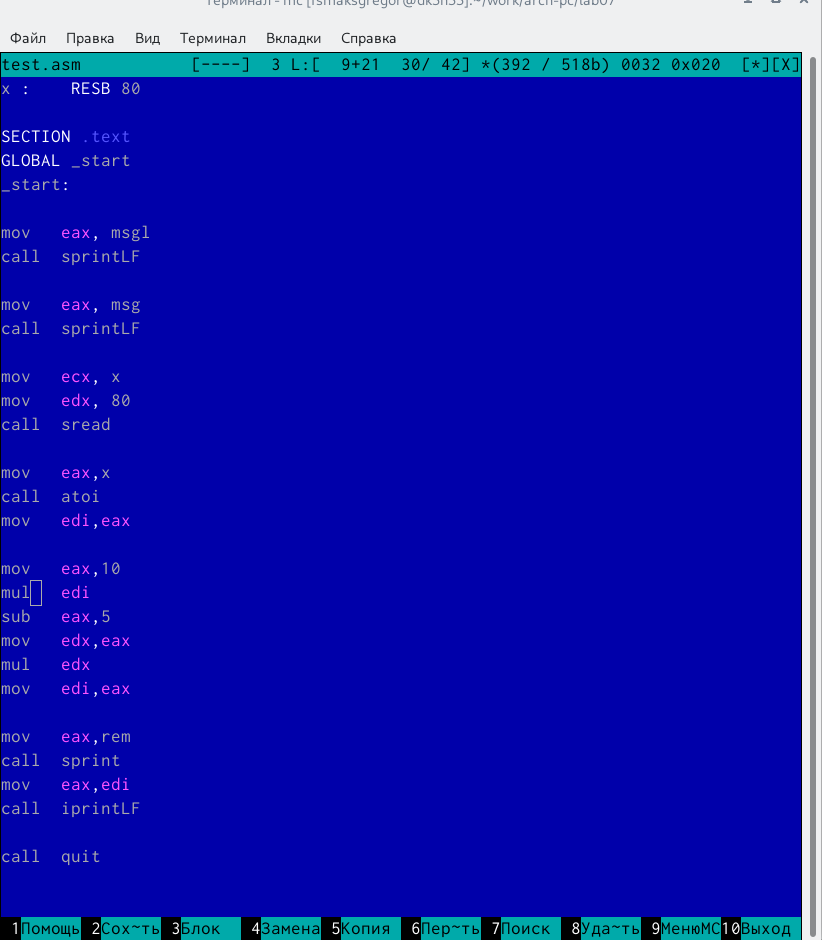
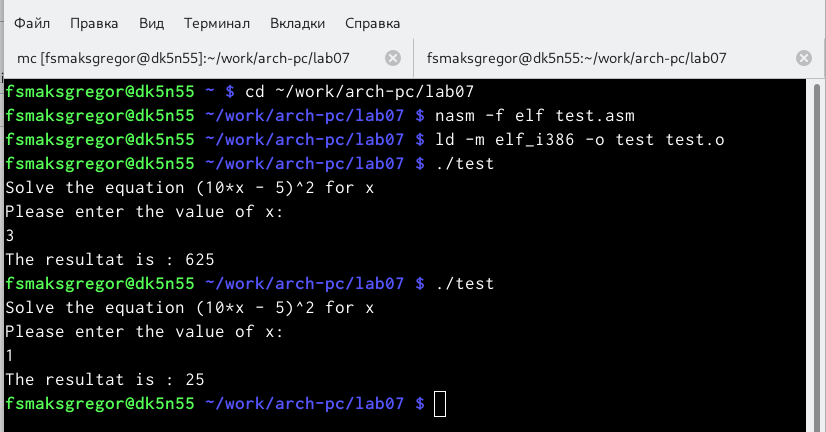


Рис. 20: Ресунок 22

* и, наконец, мы проверяем корректность кода, который мы написали, ис- пользуя два разных значения 𝑥1 = 3 𝑥2 = 1

Как указано на следующем рисунке (рис. **¿fig:fig23?**) 

## 3.1 Выводы по результатам выполнения заданий :

* В этой части мы смогли узнать, как преобразовать некоторые математические идеи в реальный код на ассемблере, что помогло нам получить более глубокое представление о том, как работать с регистрами.

# 4 Выводы

* В седьмой лаборатории мы в основном научились писать программы, вы- полняющие арифметические операции, и научились вычислять математи- ческие выражения средней сложности.