Лабораторная работа №6

Арифметические операции в NASM

Коровкин Никита Михайлович

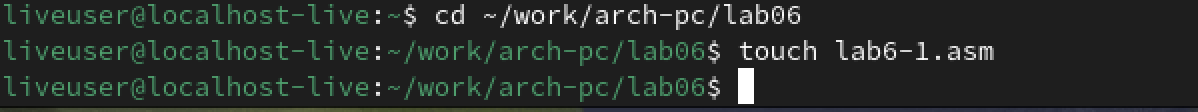
Содержание

# 1 Цель работы

Познакомиться с базовыми инструкциями языка Ассемблер, отвечающими за основные арифметические операции.

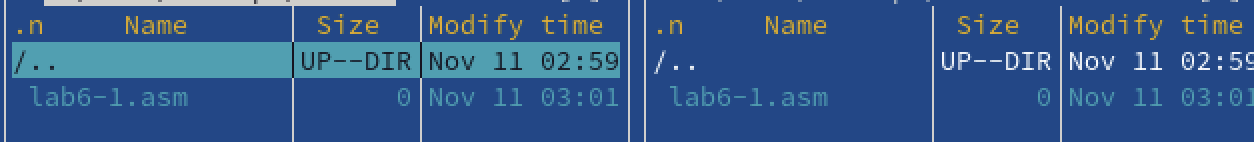
# 2 Выполнение лабораторной работы

В первую очередь переместимся в папку для 6-й лабораторной работы и создадим первый файл.(рис.1)



Создание файла lab6-1.nasm

Запустим Midnight Commander и откроем наш файл.(рис.2)



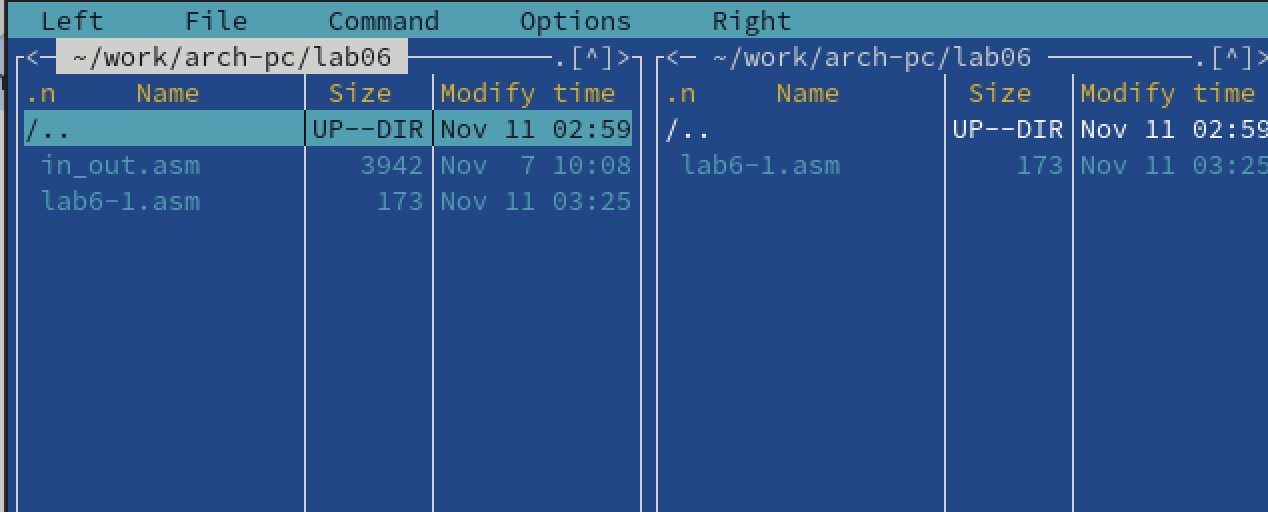
файл в Midnight Commander

Открыв файл, вставляем первый код из листинга 6.1.(рис.3)



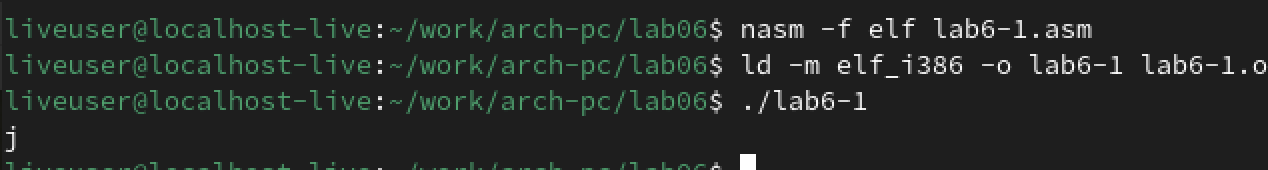
Записываем небходимый код в файл

После этого добавим в нашу папку файл in\_out.asm, так как без него наш код работать не будет.(рис.4)



Добавляем файл для корректной работы кода

Теперь запустим код из нашего перового файла. На выходе мы получим j. Это неправильный вывод, так как изначально мы складывали цифры.



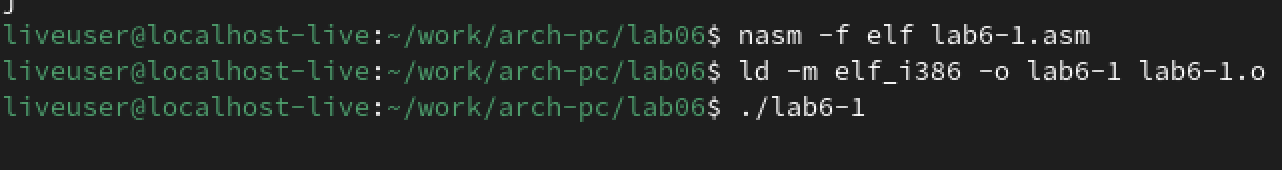
Запуск файла lab6-1.nasm

Чтобы получить на выходе верный результат, отредактируем код, убрав кавычки.(рис.6)



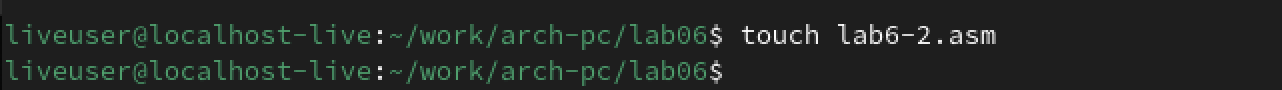
Редактируем код файла lab6-1.nasm

Теперь запустим код еще раз. НА выходе мы ничего не видим, однако это не так. По таблице ASCII под 10 номером обозначен перевод строки. Наш код выводит не число 10, а символ с данным номером.(рис.7)



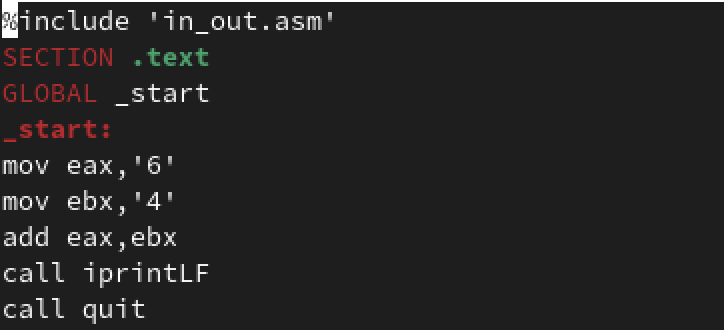
Повторный запуск кода

Создадим второй файл - lab6-2.nasm.(рис.8)



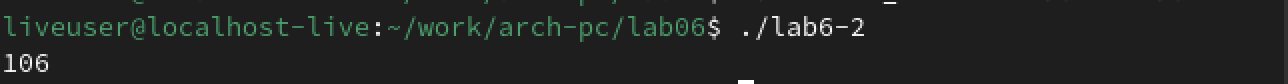
Создание файла lab6-2.nasm

Вставим в него код из листинга 6.2.(рис.9)



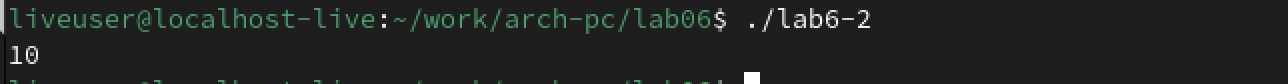
Записываем необходимый код в файл

Запускаем код и на выходе получаем 106.(рис.10)



Запуск кода

Мы получили 106, так как цифры были в кавычках и мы получили сложение их номеров. 54 и 52 в сумме дают 106. Тогда уберем кавычки и проверим программу.(рис.11)



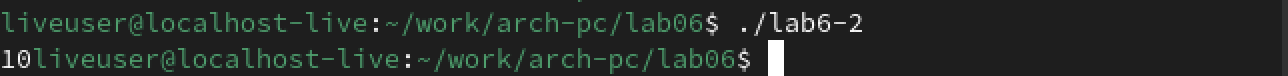
Запуск отредактированного кода

Все выводится верно. Теперь отредактируем код еще раз. Заменим в коде iprintLF на iprint, чтобы узнать отличие.(рис.12)



Заменяем в коде iprintLF на iprint

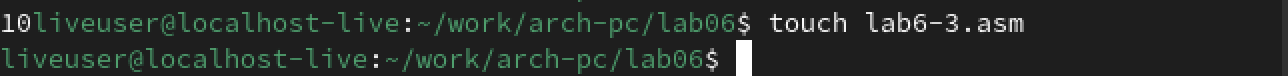
Теперь запустим файл.(рис.13)



Запуск кода

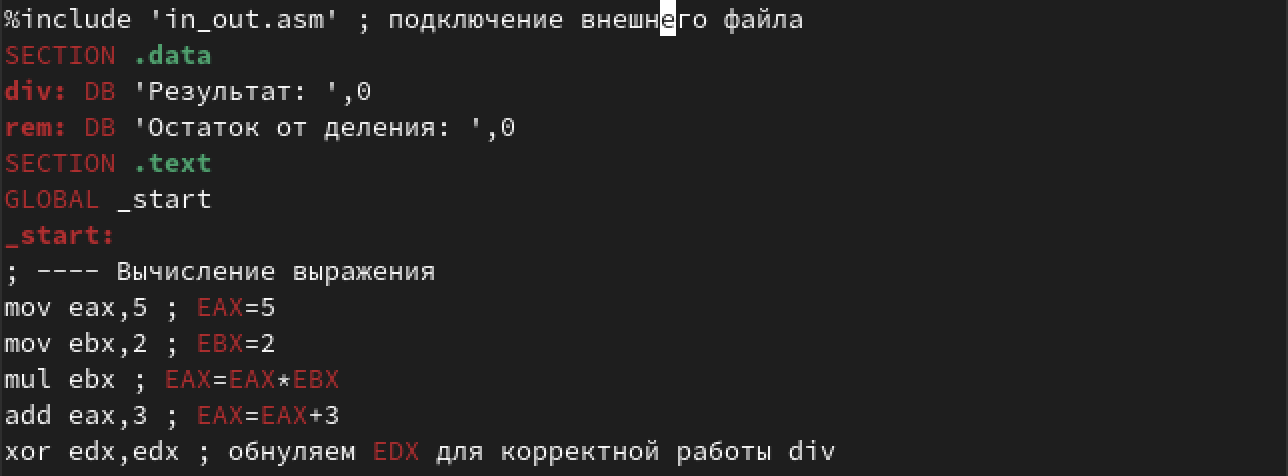
Как мы видим, после замены число выводится без переноса строки.

Создадим теперь третий файл.(рис.14)



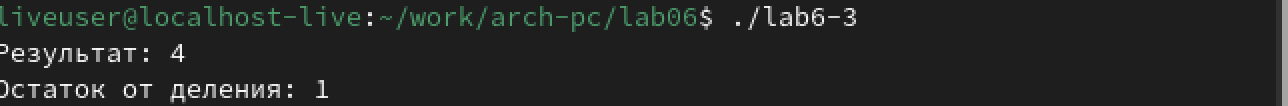
Создание файла lab6-3.nasm

Вставим в него код из листинга 6.3.(рис.15)



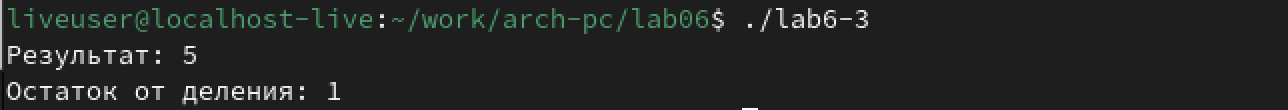
Вставляем необходимый код

Код должен выводить значение функции (5\*2+3)/3. Запустим код. На выходе у нас ответ 4 и остаток от деления 1.(рис.16)



Проверяем работу кода

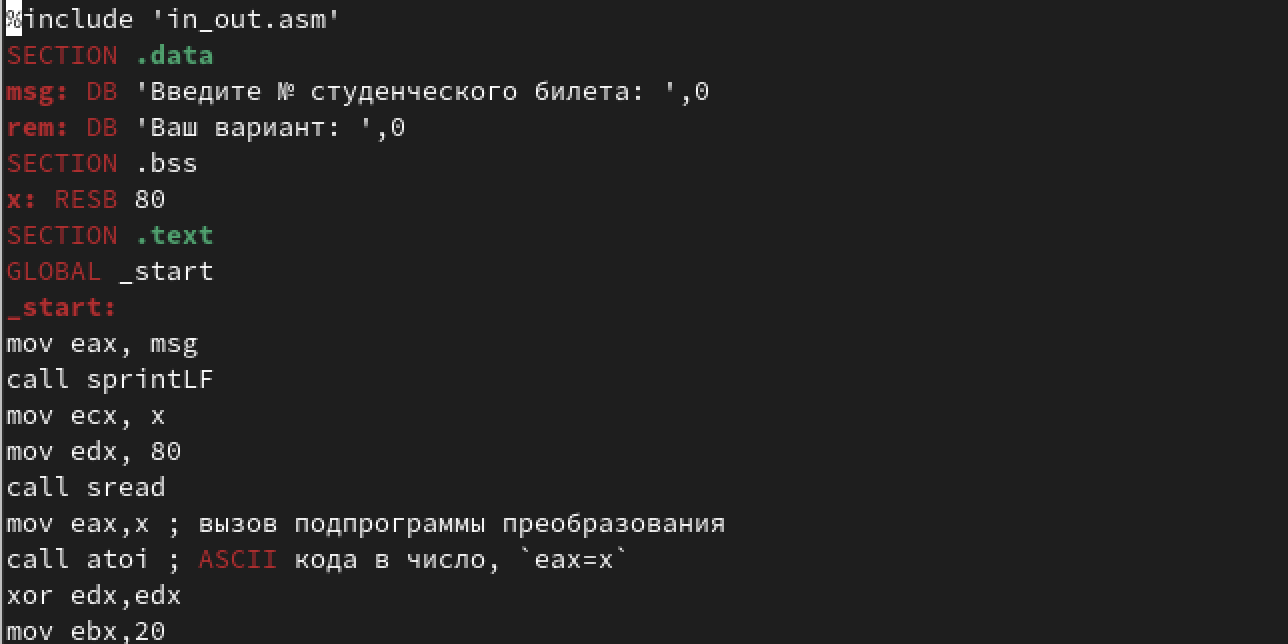
Наш результат совпадает с результатом, указанным в лабораторной работе. Теперь изменим файл так, чтобы он вычислял значение выражения (4\*6+2)/5. Запустим код и проверим, совпадет ли результат.(рис.17)



Запуск отредактированного кода

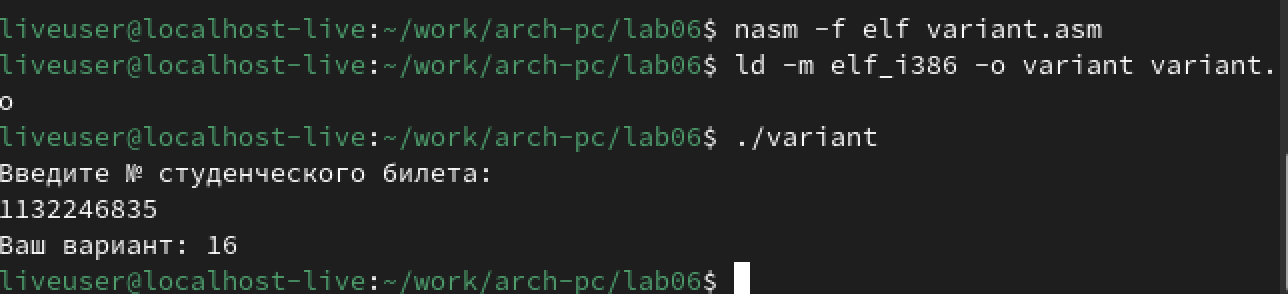
Результат действительно совпадает.

Теперь нам необходимо создать файл, который будет высчитывать вариант задания для самостоятельной работы. Вставляем необходимый код в файл.(рис.18)



Файл variant.asm

Теперь запустим код. Введем туда номер студенческого билета.(рис.19)



Запуск кода для высчитывания варианта

Мы получили на выходе 16. Таким будет номер нашего варианта.

Разберем то, как работает код, ответив на следующие вопросы.

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?

За вывод сообщения отвечает call sprint и строка mov eax,rem, перемещающая строку с фразой в регистр eax, из которого мы считываем данные для вывода

1. Для чего используется следующие инструкции?

mov ecx, x

mov edx, 80

call sread

Данные строки используются для записи данных в переменную х.

1. Для чего используется инструкция “call atoi”?

Инструкция используется для преобразования ASCII кода в число.

1. Какие строки листинга 6.4 отвечают за вычисления варианта?

div ebx и inc edx

Первая отвечает за деление, вторая - прибавляет единицу

1. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?

В регистр edx

1. Для чего используется инструкция “inc edx”?

Оно увеличивает значение на единицу

1. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?

mov eax,edx и call iprintLF

Первая строка переносит значение регистра edx в eax, а вторая выводит значение регистра eax

# 3 Выполнение самостоятельной работы

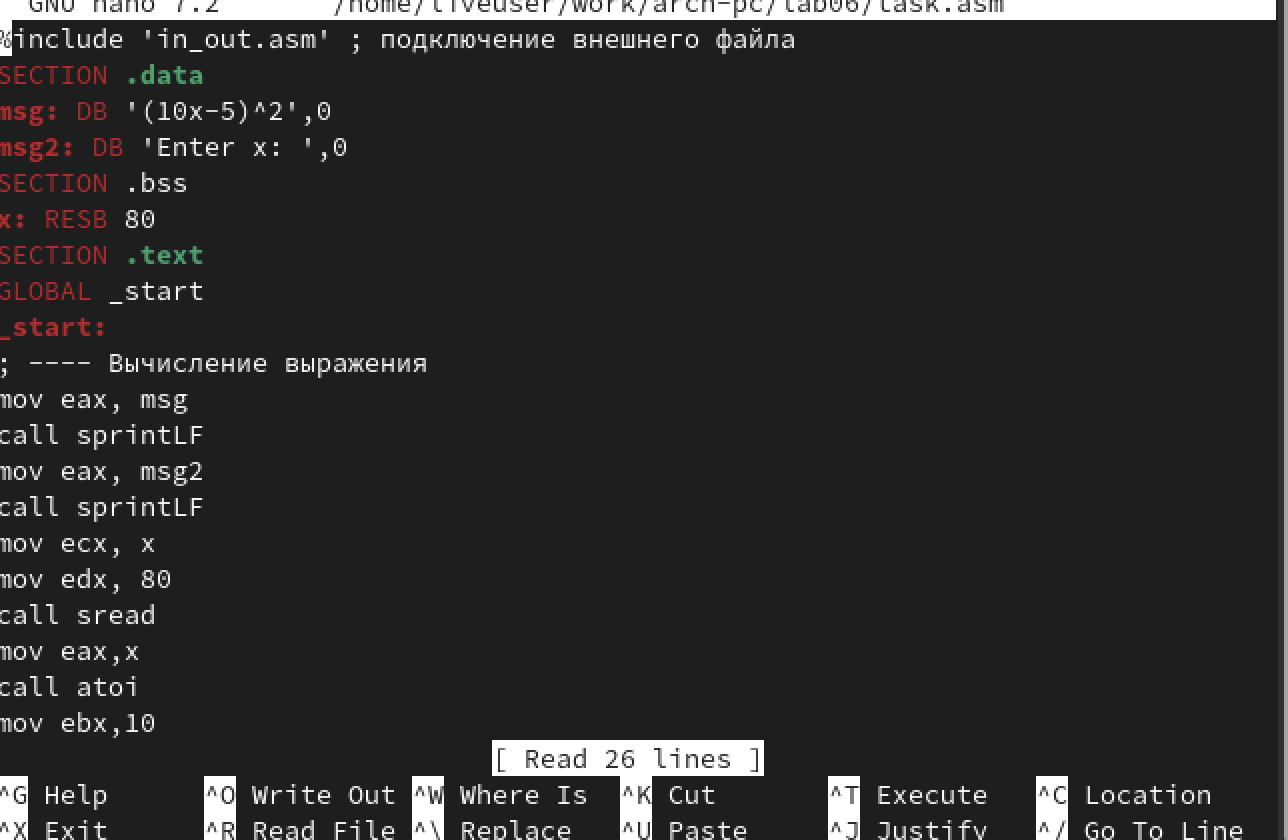
В качестве самостоятельной работы решим задание из 16-го варианта. Это уравнение (10𝑥 − 5)^2, где х1 = 1, х2 = 3.

Для начала создадим файл для вычисления.(рис.20)

Создание файл для самостоятельной работы

Создание файл для самостоятельной работы

Откроем файл и запишем туда код для вычисления.(рис.21)



Код для вычисления

Первые 4 строки отвечают за вывод сообщения на экран.

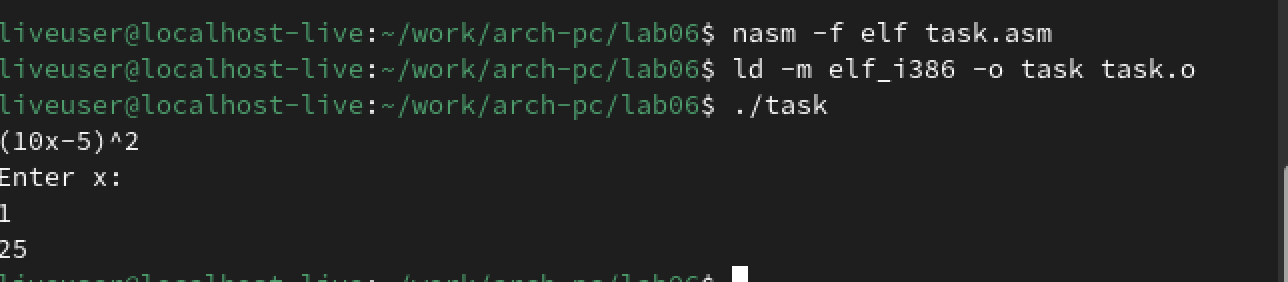
Следующие 3 строки мы используем для ввода значения переменной х.

mov eax, x и call atoi отвеча.т за преобразование ASCII кода в число.

Затем умножаем х на 10 и вычитаем 5. Полученное число умножаем на само себя, чтобы возвести в квадрат.

В конце выводим результат и завершаем программу.

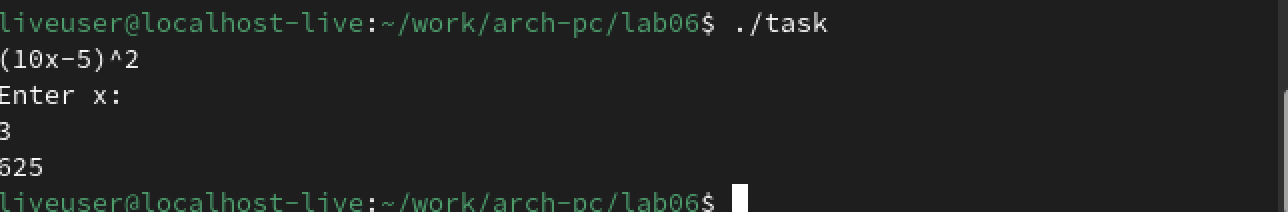
Запустим код и проверим результат.(рис.22)



Первый запуск кода

Введя 1, получаем 25. Посчитав вручную, мы понимаем, что ответ верный.

Запусти код второй раз, введя 3.(рис.23)



Второй запуск кода

Ответ 625, значит код написан верно.

# 4 Выводы

В ходе выполнения лабораторной работы мы получили знания о видах и работе арифметических операций на языке Ассемблера. Также мы самостоятельно написали программу, закрепив знания.