Арифметические операции в NASM

Лабораторная работа №6

Приходько Иван Иванович

Содержание

# 1 Цель работы

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

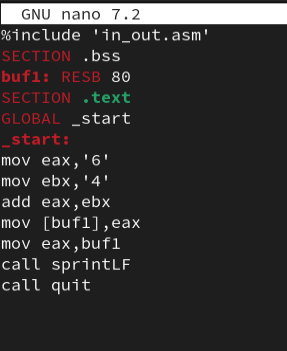
# 2 Выполнение лабораторной работы

Для начала выполнения лабораторной работы необходимо создать файл lab6-1.asm (рис. 2.1).

Создание файла lab6-1.asm

Создание файла lab6-1.asm

Вставим в наш созданный файл код из листинга 6.1 (рис. 2.2).



Запись кода в файл lab6-1.asm

Перед начало работы скопируем файл in\_out.asm для корректной работы (рис. 2.3).

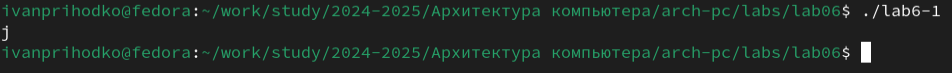


Рабочее пространство на данный момент

Соберем и запустим программу (рис. 2.4 и 2.5).

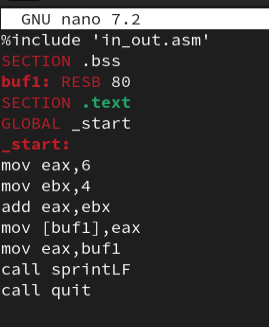
Сборка lab6-1.asm

Сборка lab6-1.asm



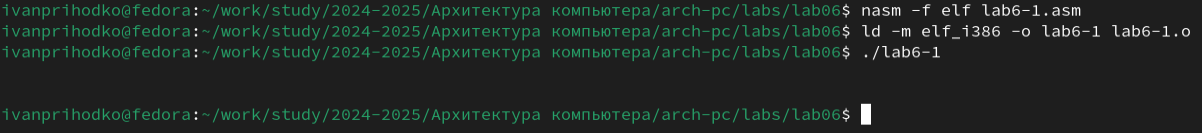
Запуск lab6-1.asm

Нам выводит символ j, однако это неправильный вывод. Наша цель - сложить 6 и 4, и получить в выводе число 10. Попробуем изменить наш файл (рис. 2.6).



Редактирование файла lab6-1.asm

Теперь соберем и запустим файл заново (рис. 2.7).



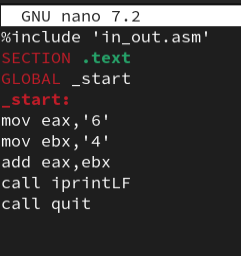
Повторный запуск lab6-1.asm

Когда мы вызываем команду sprintLF, она выводит не число 10, а символ с номером 10. Символ под номером 10 это символ перевода строки. Теперь создадим второй файл под названием lab6-2.asm (рис. 2.8).

Создание файла lab6-2.asm

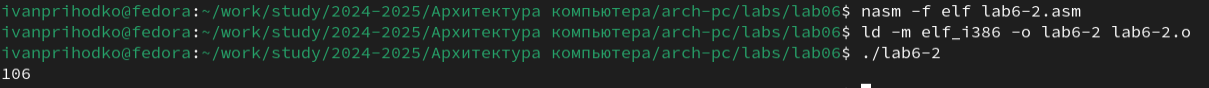
Создание файла lab6-2.asm

Теперь вставим в него код из листинга 6.2 (рис. 2.9).



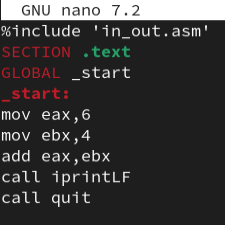
Редактирование файла lab6-2.asm

Соберем и запустим файл (рис. 2.10).



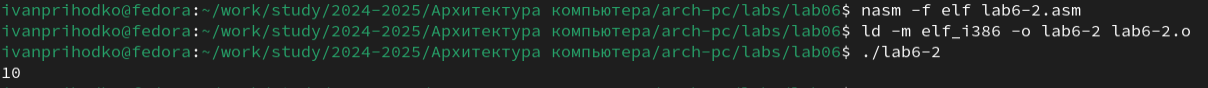
Сборка и запуск lab6-2.asm

Мы видим число 106. Так как цифры в коде указаны в кавычках, мы складываем их коды (54 и 52 в сумме дают 106). Изменим файл (рис. 2.11).



Повторное редактирование файла lab6-2.asm

Теперь запустим файл (рис. 2.12).

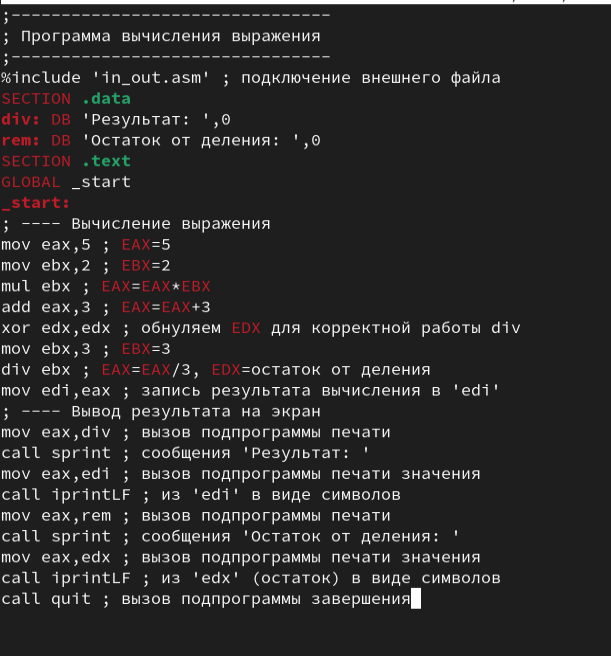


Повторный запуск lab6-2.asm

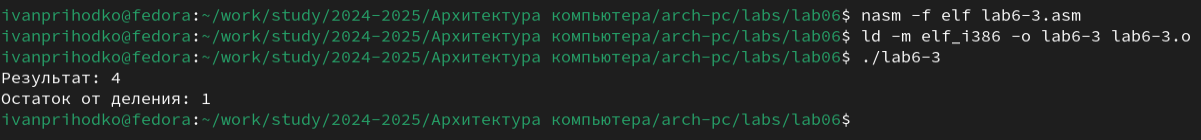
Создадим третий файл, вставим в него код из листинга 6.3, соберем и запустим его (рис. 2.13-2.15).

Создание lab6-3.asm

Создание lab6-3.asm

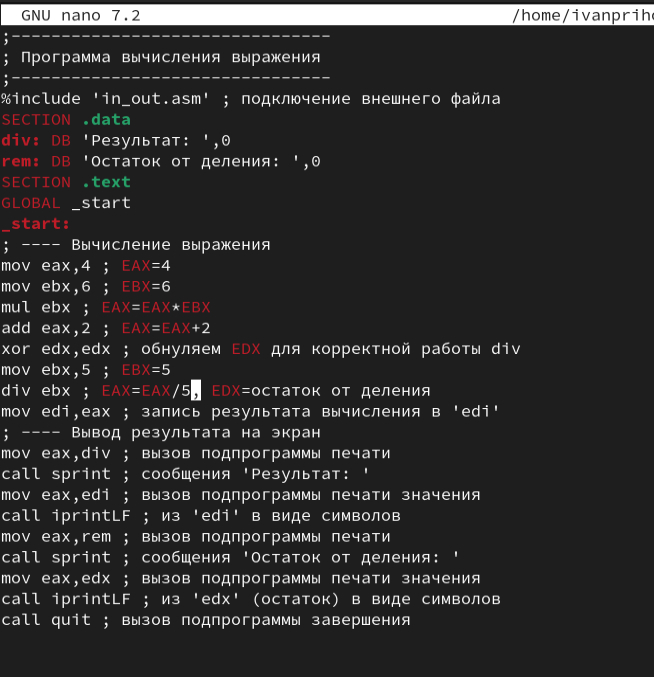


Редактирование lab6-3.asm



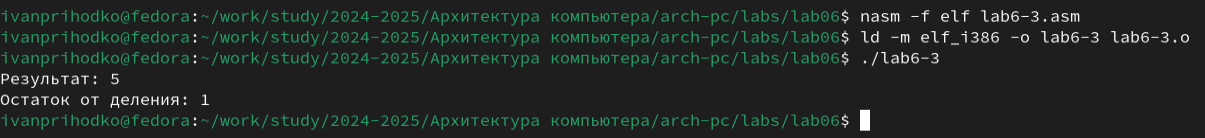
Сборка и запуск lab6-3.asm

Полученный результат совпадает с результатом, указанным в лабораторной работе. Теперь изменим файл так, чтобы он вычислял значение выражения (4\*6+2)/5. (рис. 2.16).



Повторное редактирование lab6-3.asm

Соберем и запустим его (рис. 2.17).

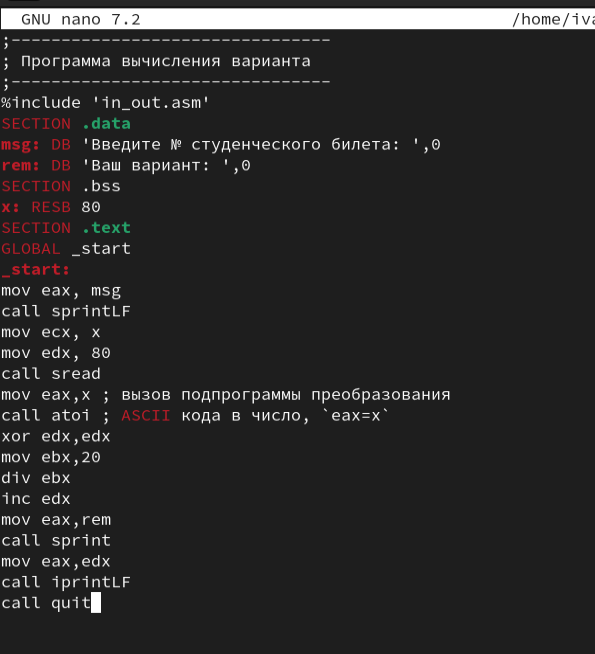


Повторная сборка и запуск lab6-3.asm

Теперь создадим файл variant.asm для вычисления варианта самостоятельной работы и вставим в него код из листинга 6.4 (рис. 2.18 и 2.19).

Создание variant.asm

Создание variant.asm



Редактирование variant.asm

Запишем номер своего студенческого и узнаем номер варианта (рис. 2.20).



Запуск variant.asm

Теперь отвечу на предложенные в лабораторной работе вопросы:

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

За это отвечает строчка call sprint в связке с mov eax,rem

1. Для чего используется следующие инструкции?

mov ecx, x

mov edx, 80

call sread

Эти строки используются для того, чтобы записать данные в переменную x

1. Для чего используется инструкция “call atoi”?

Для преобразования ASCII кода в число

1. Какие строки листинга 6.4 отвечают за вычисления варианта?

div ebx - для деления

inc edx - для прибавки единицы

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

В регистр edx

1. Для чего используется инструкция “inc edx”?

Для увеличения значения регистра edx на единицу

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

mov eax,edx - для переноса значение регистра edx в eax

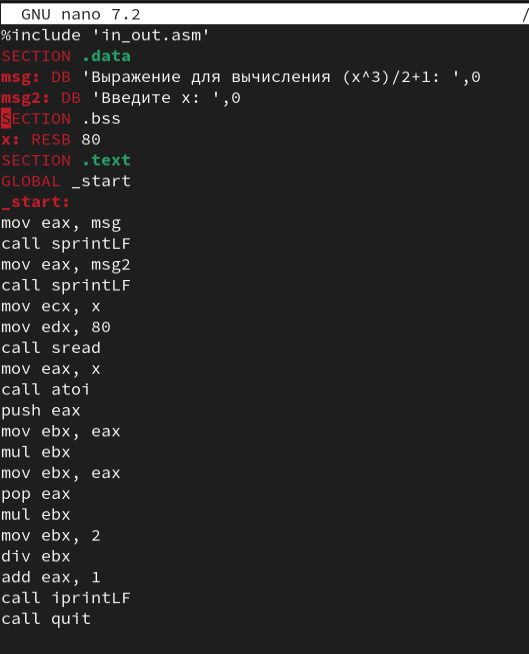
call iprintLF - выводит значение из регистра eax

# 3 Задание для самостоятельной работы

Теперь в качестве самостоятельной работы напишем код программы для вычисления выражения в варианте 6: (x^3)/2+1 (рис. 3.1).

Создание task6.asm

Создание task6.asm



Редактирование task6.asm

mov eax, msg

call sprintLF

mov eax, msg2

call sprintLF

Эти строчки отвечают за вывод сообщений пользователю

mov ecx, x

mov edx, 80

call sread

Эти строчки отвечают за считывание х у пользователя

mov eax, x

call atoi

Эти строчки конвертируют все строки в десятичные числа

push eax

mov ebx, eax

mul ebx

mov ebx, eax

pop eax

mul ebx

Эти строчки отвечают за счет x^3, сначала вы запоминаем оригинальное значение х, потом считает x^2, потом умножаем х на x^2

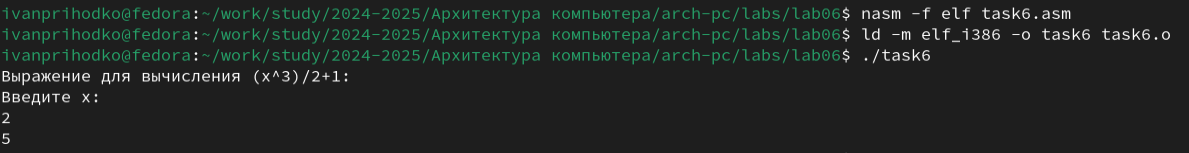
mov ebx, 2

div ebx

add eax, 1

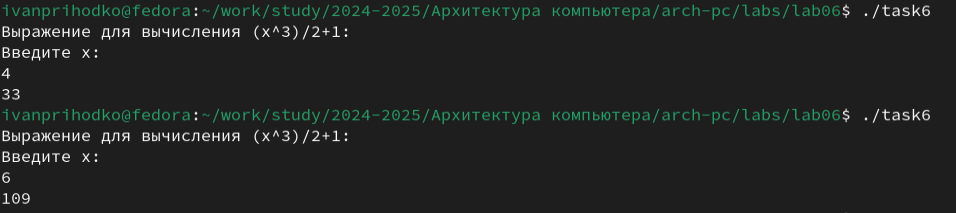
Эти строчки отвечают за деление на 2 и прибавку единицы

Теперь соберем и запустим код (рис. 3.3).



Сборка и запуск task6.asm

Все правильно, проведем еще несколько тестов (рис. 3.4).



Повторные запуски task6.asm

# 4 Выводы

В результате выполнения лабораторной работы было получено представление о том, какие арифметические операции есть в языке Ассемблера, и как они работают. Были написаны программы, использующие в себе операции сложения, вычитания, умножения и деления.