Отчёт по лабораторной работе №6

дисциплина: Архитектура вычислительных систем

Мосолов Александр Денисович

Содержание

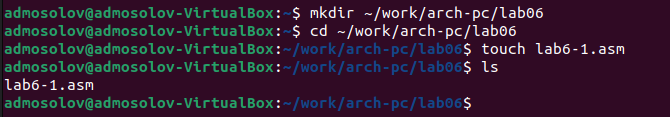
# 1 Цель работы

Освоение арифметических инструкций языка ассемблера *NASM*.

# 2 Выполнение лабораторной работы

## 2.1 Символьные и численные данные в NASM

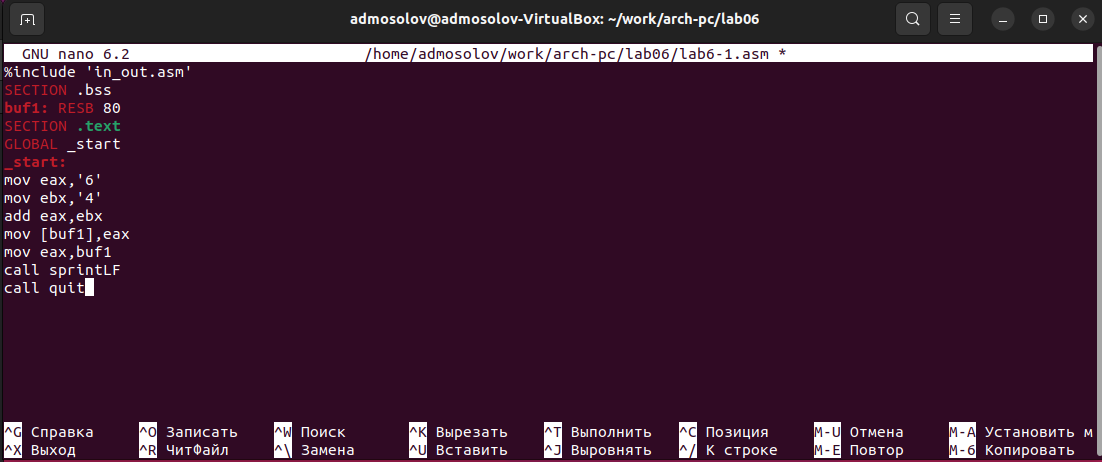
Создаём каталог для программ лабораторной работы № 6, переходим в него и создаём файл *lab6-1.asm*.



Создаем каталог для работы

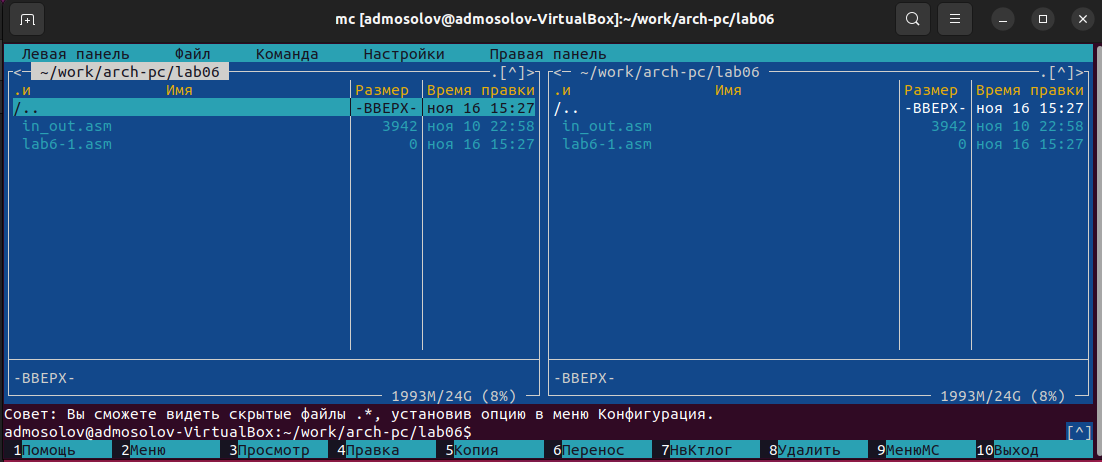
Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр *eax*.

Введём в файл *lab6-1.asm* текст программы из листинга *6.1*.



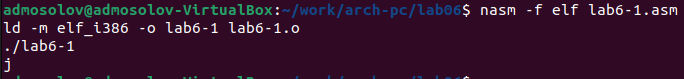
Текст из листинга 6.1

Для того, чтобы программа транслировалась без ошибок перенесем файл *in\_out.asm* в *~/work/arch-pc/lab06*, проверим содержимое каталога.



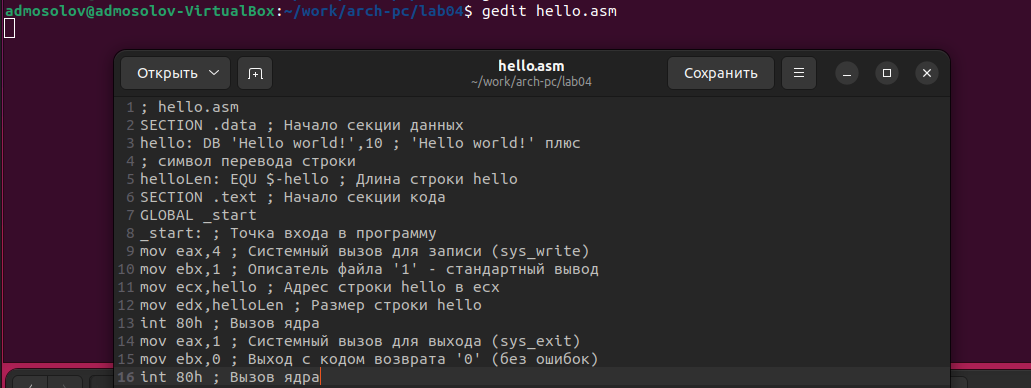
Каталог lab06

Транслируем полученный текст программы *lab6-1.asm* в объектный файл. Выполняем компоновку объектного файла и запускаем получившийся исполняемый файл.



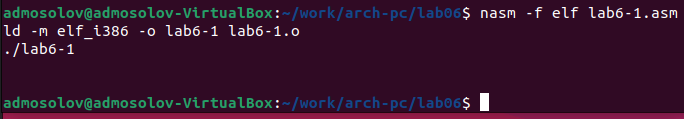
Запускаем lab6-1.asm

В данном случае при выводе значения регистра eax мы ожидаем увидеть число *10*. Однако результатом будет символ *j*. Это происходит потому, что код символа *6* равен *00110110* в двоичном представлении (или *54* в десятичном представлении), а код символа *4* – *00110100 (52)*. Команда *add eax,ebx* запишет в регистр *eax* сумму кодов – *01101010 (106)*, что в свою очередь является кодом символа *j*.



Вводим текст

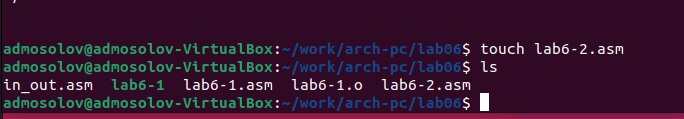
Далее изменим текст программы и вместо символов, запишем в регистры числа.



Вывод программы после изменения символов на числа

В данном случае выводится символ с кодом *10*. Но на экране он не отображается.

Создадим файл *lab6-2.asm* в каталоге *~/work/arch-pc/lab06*.



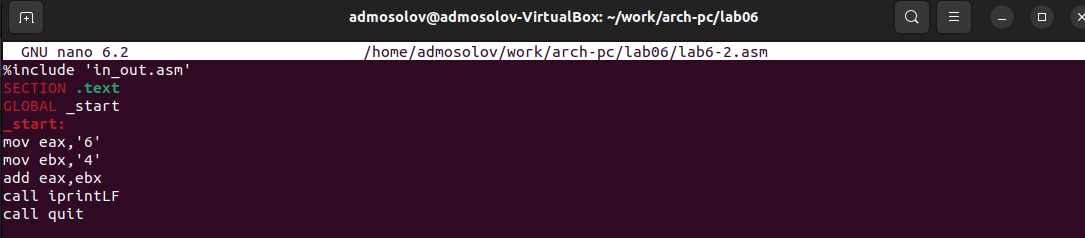
Создаём файл lab6-2.asm

Введём в него текст программы из листинга *6.2*, запустим программу с помощью команд:

*nasm -f elf lab6-2.asm*

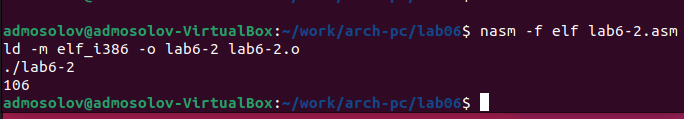
*ld -m elf\_i386 -o lab6-2 lab6-2.o*

*./lab6-2*



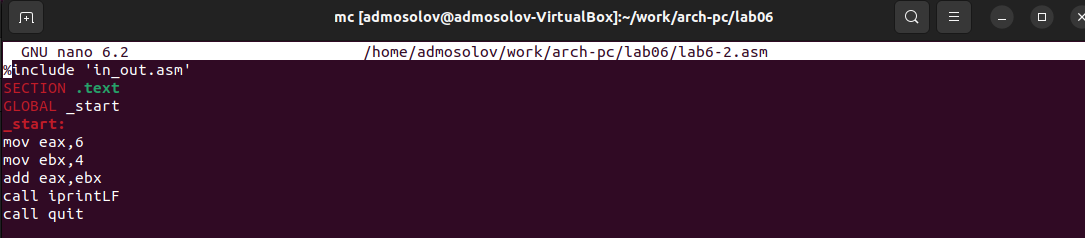
Текст из листинга 6.2

В результате работы программы мы получим число *106*. В данном случае, как и в первом, команда *add* складывает коды символов *‘6’* и *‘4’ (54+52=106)*. Однако, в отличии от программы из листинга *6.1*, функция *iprintLF* позволяет вывести число, а не символ, кодом которого является это число.



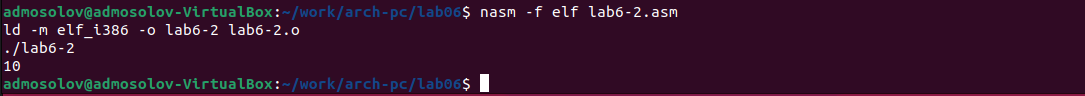
Вывод программы lab6-2

Аналогично предыдущему примеру изменим символы на числа.



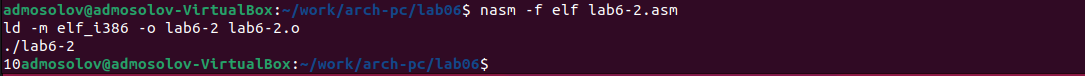
Текст программы после изменений

Посмотрим на вывод программы с использованием *iprintLF*.



Используем iprintLF

А теперь на вывод программы с использованием *iprint*.

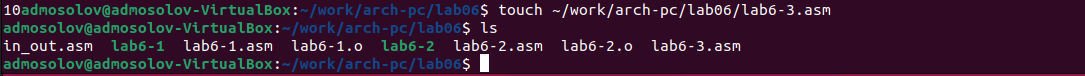


Используем iprint

Сравним полученные результаты. В случае с использованием *iprintLF* курсор переводится на следующую строчку, а в случае с использованием *iprint* курсор не переносится на новую строку.

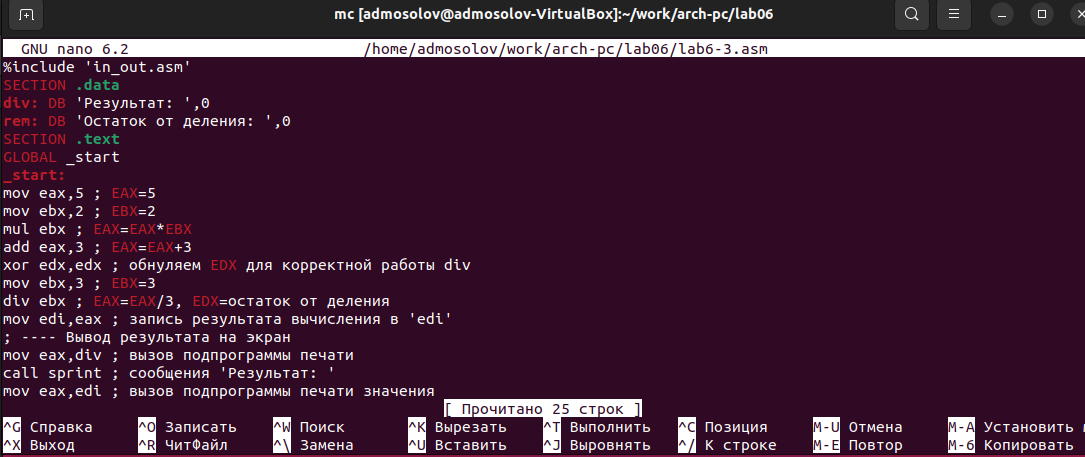
## 2.2 Выполнение арифметических операций в NASM

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения *f(x) = (5 \* 2 + 3)/3.*



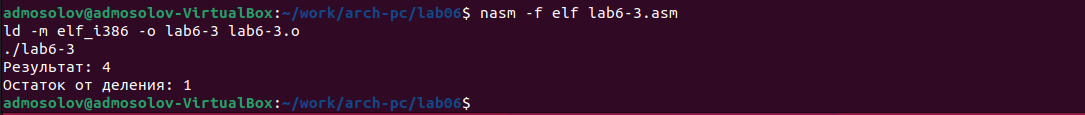
Создание файла lab6-3.asm

Изучим и введём текст из листинга *6.3* в *lab6-3.asm*.



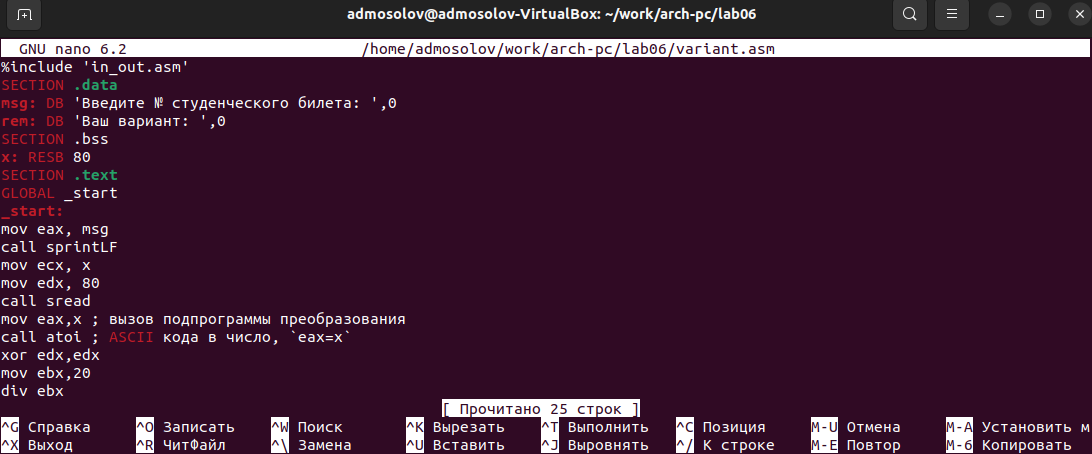
Текст программы lab6-3.asm

Запустим исполняемый файл, убедимся в правильности результата.



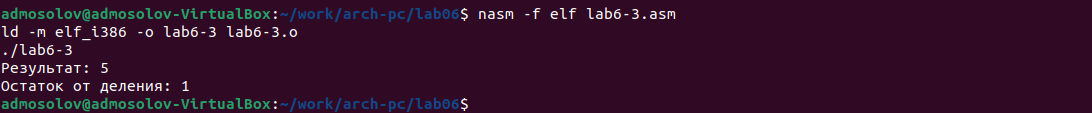
Результат компиляции файла lab6-3

Изменим текст программы для вычисления выражения *f(x) = (4 \* 6 + 2)/5*.



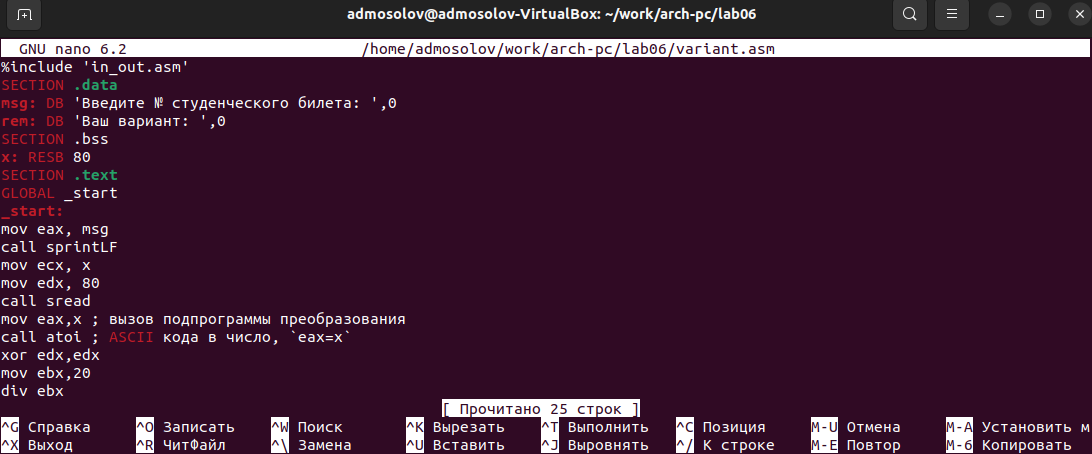
Текст файла lab6-3 после изменения

Транслируем полученный текст программы *lab6-3.asm* в объектный файл. Выполняем компоновку объектного файла и запускаем получившийся исполняемый файл.



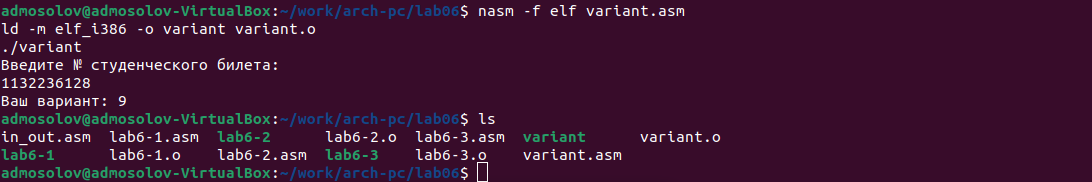
Результат запуска изменённого файла lab6-3

Создаём файл *variant.asm* в каталоге *~/work/arch-pc/lab06* с помощью команды: *touch ~/work/arch-pc/lab06/variant.asm*. Читаем текст из листинга *6.4* и вводим его в файл *variant.asm*.



Текст программы variant.asm

Транслируем текст программы *variant.asm* в объектный файл. Выполняем компоновку объектного файла и запускаем получившийся исполняемый файл.



Номер варианта

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

* *mov eax,rem* *call sprint*

1. Для чего используется следующие инструкции?

* *mov ecx, x* - запись адреса переменной в *EAX*
* *mov edx, 80* - запись длины вводимого сообщения в *EBX*
* *call sread* - вызов подпрограммы ввода сообщения

1. Для чего используется инструкция *“call atoi”*?

* Вызывается функция преобразования *ascii-код символа* в целое число.

1. Какие строки листинга *6.4* отвечают за вычисления варианта?

* mov ebx,20
* div ebx

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

* *eax*

1. Для чего используется инструкция *“inc edx”*?

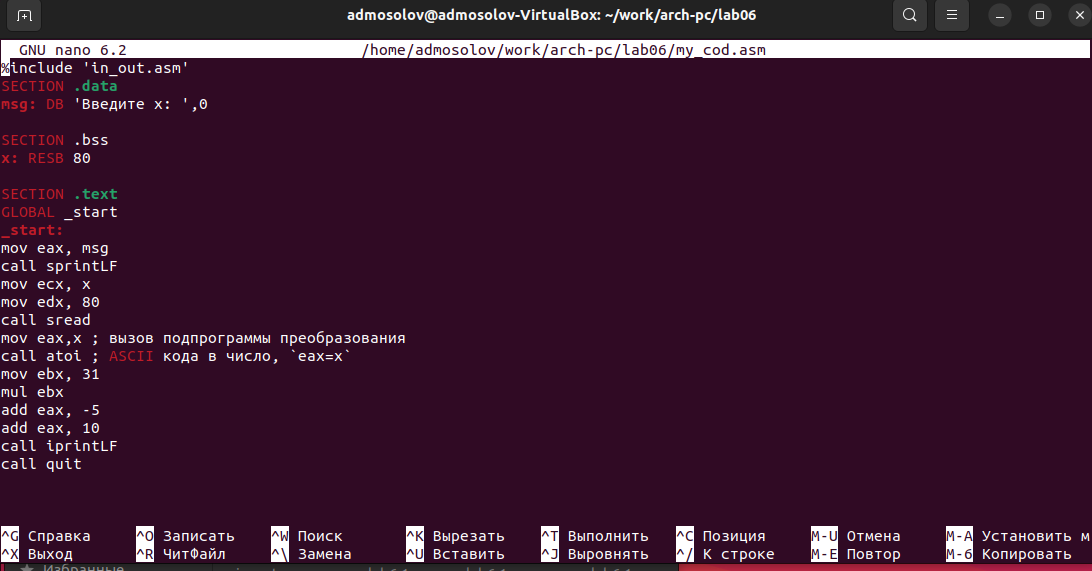
* Увеличиваем *edx* на единицу

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

* *mov eax,edx*
* *call iprintLF*

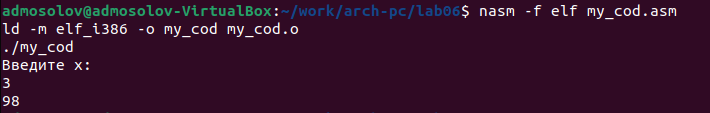
## 2.3 Задание для самостоятельной работы

Напишем программу для вычисления выражения *f(x) = 10 + (31x − 5)* (9 вариант).

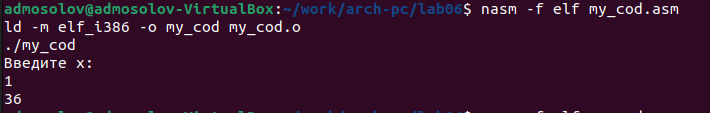


Текст программы - вычисление значения функции

Выведем значение функции при *x = 3* и *x = 1*.



Находим значение функции при *x = 3*



Находим значение функции при *x = 1*

Перенесём файлы с программой в *~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab06*, загрузим изменнения на *github*.

# 3 Выводы

В ходе лабораторной работы были освоены арифметических инструкции языка ассемблера *NASM*.