Отчёт по лабораторной работе №6

Дисциплина: архитектура компьютера

Захаров Кирилл Юрьевич

Содержание

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

# 2 Выполнение лабораторной работы

## 2.1 Символьные и численные данные в NASM

Я создал папку для хранения файлов лабораторной работы номер шесть, перешел в нее и сформировал файл с названием lab6-1.asm. (рис. [[1](#fig:001)])

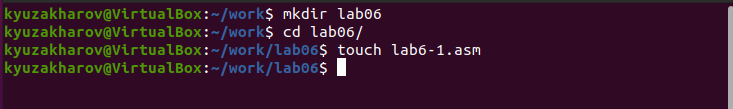


Figure 1: Создание каталога и файлов

Давайте изучим примеры программ, которые выполняют вывод символов и числовых данных. Эти программы будут отображать данные, которые хранятся в регистре eax.

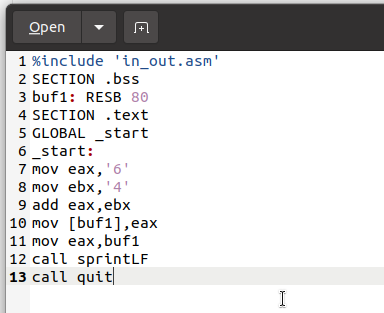


Figure 2: Изменение программы lab6-1.asm

В одной из таких программ в регистр eax помещается символ ‘6’ (используя команду mov eax, ‘6’), а в регистр ebx помещается символ ‘4’ (mov ebx, ‘4’). Затем мы складываем значения регистров eax и ebx (выполняем команду add eax, ebx, и результат сложения сохраняется в регистре eax). После этого мы производим вывод результата. (рис. [[2](#fig:002)]) (рис. [[3](#fig:003)])

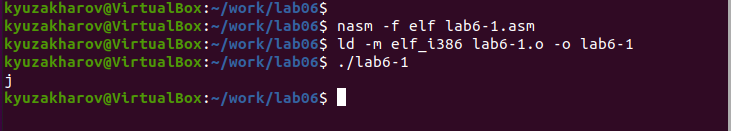


Figure 3: Компиляция и запуск исполняемого файла lab6-1.asm

Так как функция sprintLF требует, чтобы в регистре eax находился адрес, нам нужно ввести вспомогательную переменную. Мы передадим значение из регистра eax в переменную buf1 (с помощью команды mov [buf1], eax), после чего поместим адрес переменной buf1 обратно в регистр eax (команда mov eax, buf1) и вызовем функцию sprintLF.

При выводе содержимого регистра eax мы ожидаем получить значение 10. Тем не менее, на экране появится символ ‘j’. Это объясняется тем, что двоичный код для символа ‘6’ составляет 00110110, что в десятичной системе соответствует числу 54, а для символа ‘4’ двоичный код - 00110100, или 52 в десятичном исчислении. Когда выполняется команда add eax, ebx, в регистр eax записывается их сумма, равная 01101010 в двоичном формате, или 106 в десятичном, что соответствует коду символа ‘j’.

Затем я изменил текст программы, заменив символы на числа для записи в регистры. (рис. [[4](#fig:004)]) (рис. [[5](#fig:005)])

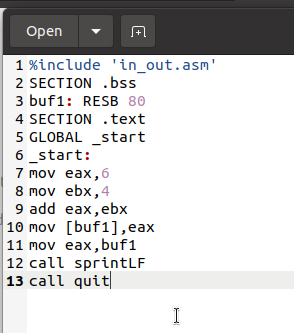


Figure 4: Изменение программы lab6-1.asm

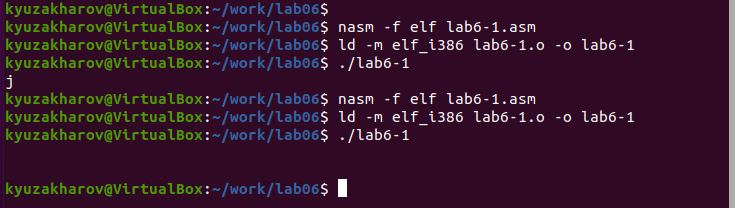


Figure 5: Компиляция и запуск исполняемого файла lab6-1.asm

Как и в прошлый раз, при выполнении программы мы не увидим цифру 10. В этом случае на экран выводится символ с кодом 10, который является символом новой строки (или возвратом каретки). В консоли он не виден, но добавляет пустую строку.

Как уже упоминалось, в файле in\_out.asm имеются вспомогательные функции для конвертации символов ASCII в числа и наоборот. Используя эти функции, я модифицировал текст программы. (рис. [[6](#fig:006)]) (рис. [[7](#fig:007)])

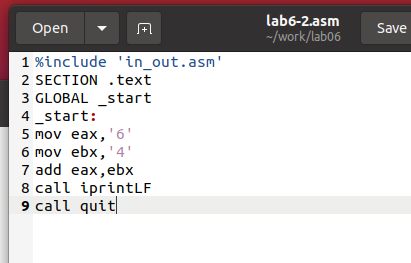


Figure 6: Изменение программы lab6-2.asm

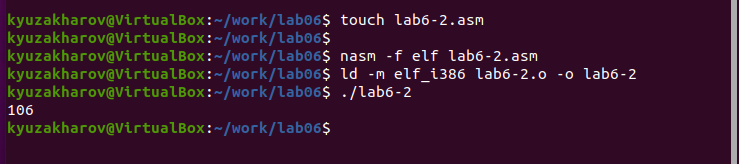


Figure 7: Компиляция и запуск исполняемого файла lab6-2.asm

Теперь программа выдаст число 106. В этом случае, так же как и в первом примере, команда add суммирует коды символов ‘6’ и ‘4’ (54+52=106). Но в отличие от предыдущей программы, функция iprintLF позволяет отобразить именно число, а не символ с соответствующим числовым кодом.

Таким же образом, как и в предыдущем примере, заменяем символы на числа. (рис. [[8](#fig:008)])

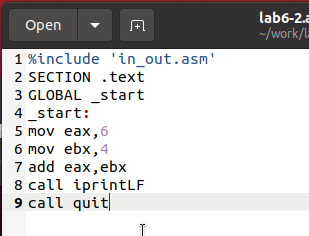


Figure 8: Изменение программы lab6-2.asm

Функция iprintLF дает возможность вывода числа, и поскольку в качестве операндов использовались числа (не коды символов), результатом является число 10. (рис. [[9](#fig:009)])

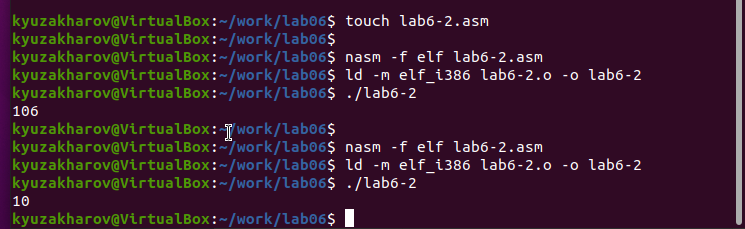


Figure 9: Компиляция и запуск исполняемого файла lab6-2.asm

Затем я заменил функцию iprintLF на iprint, создал исполняемый файл и запустил его. Результат отличается отсутствием переноса строки после вывода. (рис. [[10](#fig:010)])

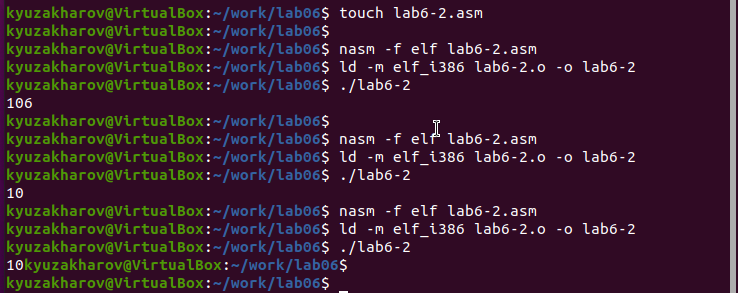


Figure 10: Компиляция и запуск исполняемого файла lab6-2.asm

## 2.2 Выполнение арифметических операций в NASM

Давайте рассмотрим пример выполнения арифметических действий в NASM на основе программы, которая рассчитывает значение функции

. (рис. [[11](#fig:011)]) (рис. [[12](#fig:012)])

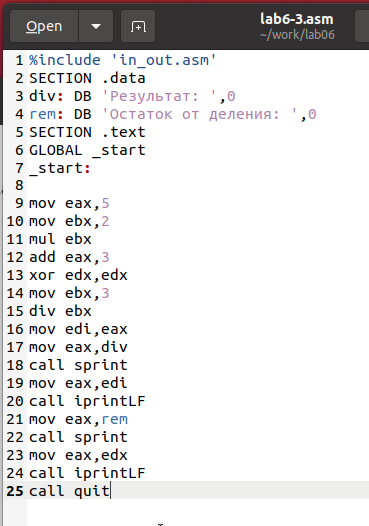


Figure 11: Изменение программы lab6-3.asm



Figure 12: Компиляция и запуск исполняемого файла lab6-3.asm

Я изменил код программы таким образом, чтобы она теперь вычисляла новое выражение

, после чего собрал исполняемый файл и проверил его функционирование. (рис. [[13](#fig:013)]) (рис. [[14](#fig:014)])

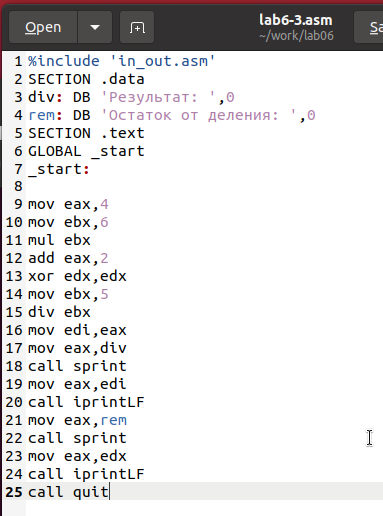


Figure 13: Изменение программы lab6-3.asm

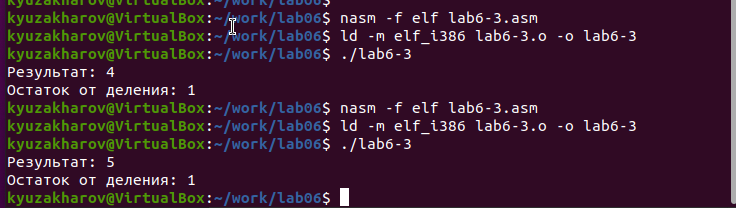


Figure 14: Компиляция и запуск исполняемого файла lab6-3.asm

Дополнительный пример демонстрирует программу для расчета индивидуального задания, основанного на номере студенческого билета. (рис. [[15](#fig:015)]) (рис. [[16](#fig:016)])

В этой ситуации число для операций вводится через клавиатуру. Напомним, что ввод осуществляется в текстовом формате, и для того чтобы арифметические операции в NASM работали правильно, текстовые символы необходимо конвертировать в числовые значения. В этом может помочь функция atoi из файла in\_out.asm.

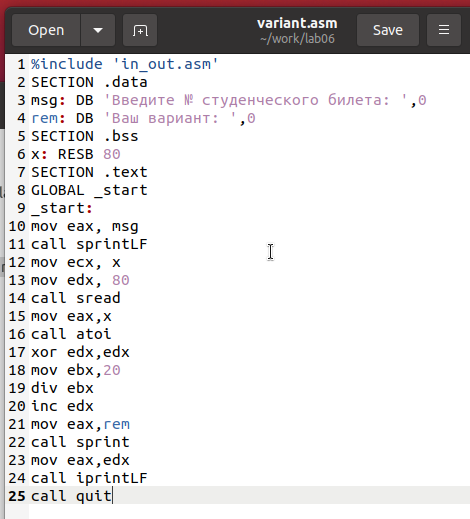


Figure 15: Изменение программы variant.asm

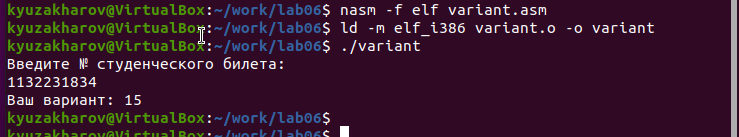


Figure 16: Компиляция и запуск исполняемого файла variant.asm

### 2.2.1 Ответы на вопросы

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

* Команда mov eax, rem загружает в регистр значение, содержащее фразу “Ваш вариант:”.
* Используя call sprint, происходит вызов функции, которая отображает строку на экране.

1. Для чего используются следующие инструкции?

* mov ecx, x помещает значение в регистр ecx.
* mov edx, 80 устанавливает в регистр edx значение 80.
* call sread активирует функцию чтения данных, которая запрашивает ввод студенческого билета и сохраняет его в переменную X.

1. Для чего используется инструкция “call atoi”?

* Эта функция конвертирует строку символов в их численное представление.

1. Какие строки листинга отвечают за вычисления варианта?

* xor edx, edx обнуляет регистр edx.
* mov ebx, 20 устанавливает в регистр ebx число 20.
* div ebx выполняет деление, используя значение в ebx.
* inc edx увеличивает значение в регистре edx на единицу, что соответствует вычислению варианта.

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

* Остаток от деления помещается в регистр edx.

1. Для чего используется инструкция “inc edx”?

* Команда увеличивает значение в регистре edx на один, что необходимо по алгоритму вычисления варианта.

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

* mov eax, edx переносит результат вычислений в регистр eax.
* call iprintLF вызывает функцию вывода числа на экран с переводом строки.

## 2.3 Задание для самостоятельной работы

Написать программу вычисления выражения y = f(x). Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x, вычислять заданное выражение в зависимости от введенного x, выводить результат вычислений. Вид функции f(x) выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. (рис. [[17](#fig:017)]) (рис. [[18](#fig:018)]) Создайте исполняемый файл и проверьте его работу для значений x1 и x2 из 6.3.

Получили вариант 15 -

для

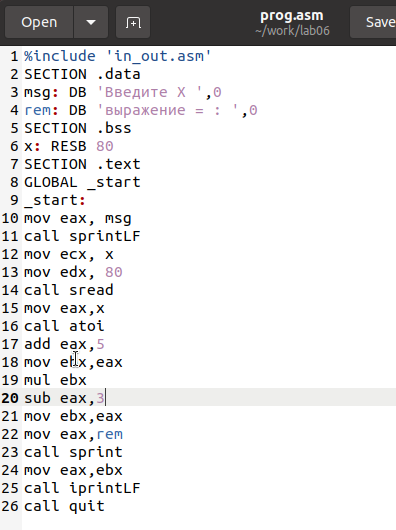


Figure 17: Изменение программы prog.asm

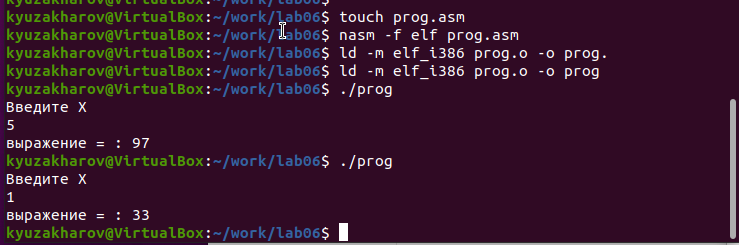


Figure 18: Компиляция и запуск исполняемого файла prog.asm

# 3 Выводы

Изучили работу с арифметическими операциями.