Лабораторная работа №6

Арифметические операции в NASM.

Дмитрий Сергеевич Хохлов

Содержание

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

# 2 Выполнение лабораторной работы

Сформировал директорию для выполнения заданий лабораторной работы №6, перешел в созданную папку и создал файла lab6-1.asm. (рис. [[1](#fig:001)])

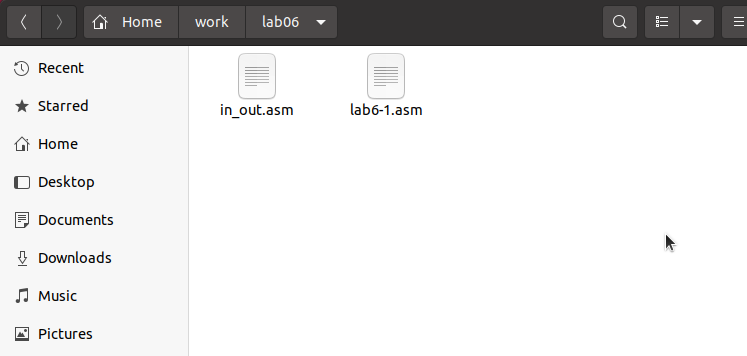


Figure 1: Подготовил каталог

Давайте рассмотрим примеры программ, которые демонстрируют вывод символов и числовых данных. Эти программы будут отображать данные, помещенные в регистр eax.

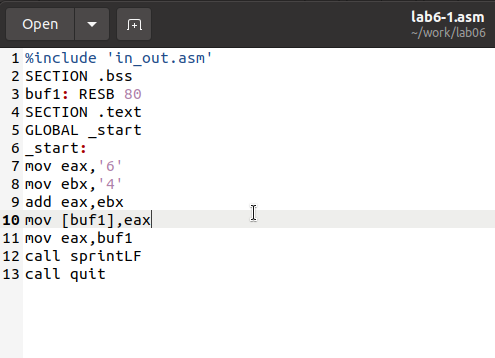


Figure 2: Программа в файле lab6-1.asm

В этом примере программы (см. рисунок [[2](#fig:002)]) мы помещаем символ ‘6’ в регистр eax с помощью команды (mov eax, ‘6’), а символ ‘4’ в регистр ebx (mov ebx, ‘4’). Далее выполняем сложение значений, содержащихся в регистрах eax и ebx, с помощью команды (add eax, ebx), где результат сложения сохраняется в регистре eax. Затем производим вывод полученного результата. Однако для того чтобы функция sprintLF сработала корректно, нужно, чтобы в регистре eax находился адрес. По этой причине мы используем вспомогательную переменную. Значение из регистра eax переносим в переменную buf1 (mov [buf1], eax), затем загружаем адрес переменной buf1 обратно в регистр eax (mov eax, buf1) и выполняем вызов функции sprintLF.

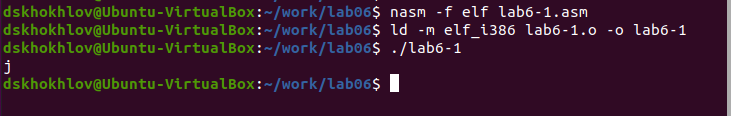


Figure 3: Запуск программы lab6-1.asm

В результате, когда мы ожидаем увидеть число 10 на выводе из регистра eax, на деле получим символ ‘j’. Это объясняется тем, что символу ‘6’ соответствует двоичный код 00110110 (или 54 в десятичной системе), а символу ‘4’ - двоичный код 00110100 (или 52 в десятичной системе). После выполнения операции сложения add eax, ebx, мы получаем сумму кодов - 01101010 (или 106 в десятичной системе), что соответствует символу ‘j’. (см. рисунок [[3](#fig:003)])

Затем произвожу корректировку текста программы, заменив символьные значения на числовые в регистрах. (рис. [[4](#fig:004)])

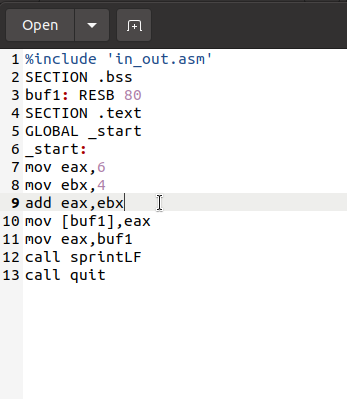


Figure 4: Программа в файле lab6-1.asm

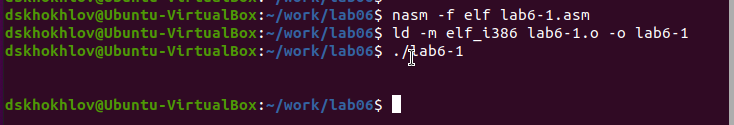


Figure 5: Запуск программы lab6-1.asm

Так же, как и в прошлый раз, в результате выполнения программы число 10 не будет выведено. Взамен этого на экране появится символ с кодом 10, который является символом новой строки (перевод строки). Смотрите (рис. [[5](#fig:005)]). Этот символ не виден в окне консоли, но он создаёт пустую строку после вывода.

Как было указано ранее, в файле in\_out.asm содержатся специальные подпрограммы для того, чтобы преобразовывать символы ASCII в числовые значения и наоборот. Я внес изменения в текст программы, чтобы использовать эти подпрограммы.

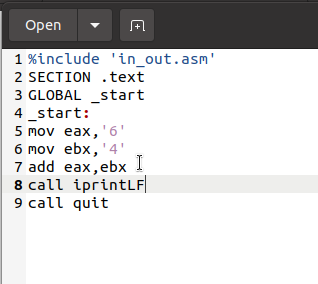


Figure 6: Программа в файле lab6-2.asm

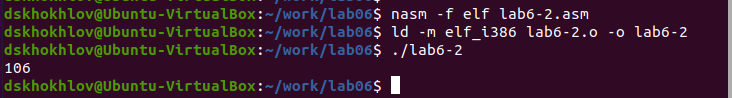


Figure 7: Запуск программы lab6-2.asm

По завершении работы программы на экран будет выведено число 106. (рис. [[7](#fig:007)]) В этом примере, так же как и в первом, инструкция add суммирует коды символов ‘6’ и ‘4’ (54+52=106). Но в этот раз, в отличие от предыдущего случая, благодаря функции iprintLF, выводится само число, а не символ с соответствующим числовым кодом.

В соответствии с предыдущим примером, мы заменим символы на числа. (рис. [[8](#fig:008)])

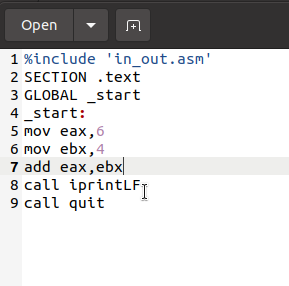


Figure 8: Программа в файле lab6-2.asm

Функция iprintLF теперь выводит число, так как в качестве операндов использовались числа, а не их символьные коды. В итоге на экране появляется число 10. (рис. [[9](#fig:009)])

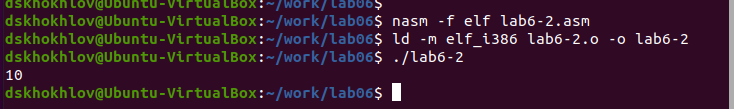


Figure 9: Запуск программы lab6-2.asm

Произвел замену функции iprintLF на iprint. Скомпилировал и запустил скомпилированный файл. Основное отличие в результате – отсутствие новой строки в конце вывода. (см. рисунок [[10](#fig:010)])

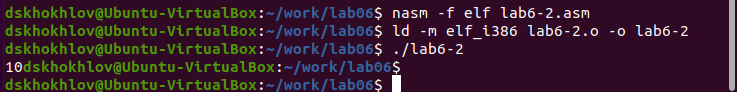


Figure 10: Запуск программы lab6-2.asm

Для демонстрации выполнения арифметических операций в NASM, рассмотрим программу, которая вычисляет следующее арифметическое выражение: (см. рисунки [[11](#fig:011)] и [[12](#fig:012)])

.

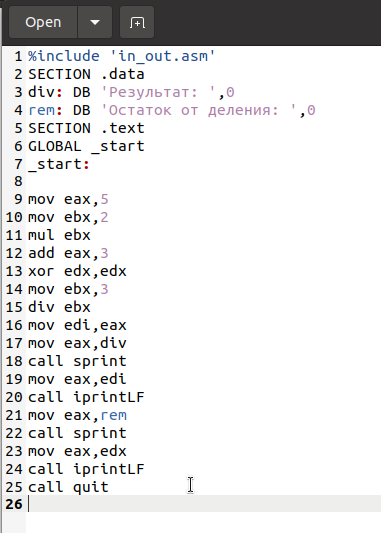


Figure 11: Программа в файле lab6-3.asm

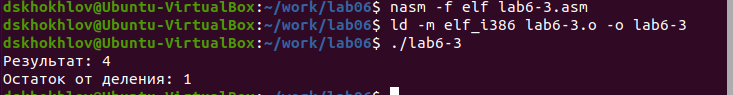


Figure 12: Запуск программы lab6-3.asm

Модифицировал код программы для расчета нового выражения:

. Собрал исполняемый файл и провел его тестирование. (см. рисунки [[13](#fig:013)] и [[14](#fig:014)])

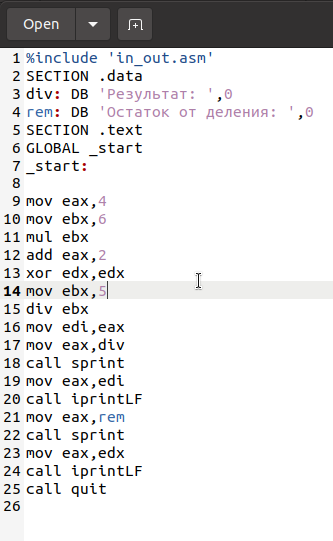


Figure 13: Программа в файле lab6-3.asm

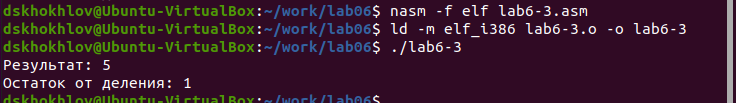


Figure 14: Запуск программы lab6-3.asm

В качестве еще одного примера возьмем программу, которая вычисляет вариант задания на основе номера студенческого билета. (см. рисунки [[15](#fig:015)] и [[16](#fig:016)])

В этом случае число для арифметических операций вводится пользователем через клавиатуру. Как было указано ранее, ввод происходит в символьной форме, и для правильного выполнения арифметических операций в NASM эти символы необходимо преобразовать в числовой формат. Для этого можно использовать функцию atoi, которая содержится в файле in\_out.asm.

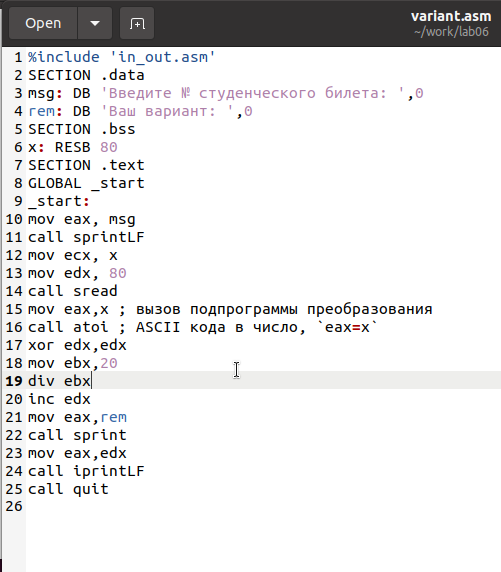


Figure 15: Программа в файле variant.asm

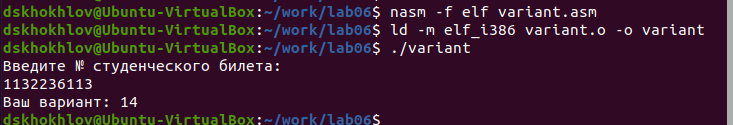


Figure 16: Запуск программы variant.asm

## 2.1 Ответы на вопросы по программе variant.asm

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

Команда “mov eax, rem” загружает в регистр значение, соответствующее строке “Ваш вариант:”

Команда “call sprint” инициирует процедуру, которая выводит строку на экран

1. Для чего используются следующие инструкции?

Команда “mov ecx, x” копирует в регистр ecx значение из переменной x

Команда “mov edx, 80” заносит число 80 в регистр edx

Команда “call sread” активирует процедуру, которая читает вводимые данные, например номер студенческого билета, с консоли

1. Для чего используется инструкция “call atoi”?

Команда “call atoi” служит для конвертации введенного текста в целочисленное значение

1. Какие строки листинга отвечают за вычисления варианта?

Команда “xor edx, edx” очищает содержимое регистра edx

Команда “mov ebx, 20” помещает число 20 в регистр ebx

Команда “div ebx” осуществляет деление числа, находящегося в аккумуляторе, на содержимое регистра ebx

Команда “inc edx” увеличивает на единицу значение в регистре edx

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

Регистр edx получает остаток от операции деления

1. Для чего используется инструкция “inc edx”?

Команда “inc edx” прибавляет единицу к содержимому регистра edx, что необходимо для корректного расчета варианта

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

Команда “mov eax, edx” переносит результат вычислений в регистр eax

Команда “call iprintLF” запускает процедуру, которая выводит результат на экран с новой строки

## 2.2 Самостоятельное задание

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. Создайте исполняемый файл и проверьте его работу для значений x1 и x2 из 6.3.

Получили вариант 14 -

для

(рис. [[17](#fig:017)]) (рис. [[18](#fig:018)])

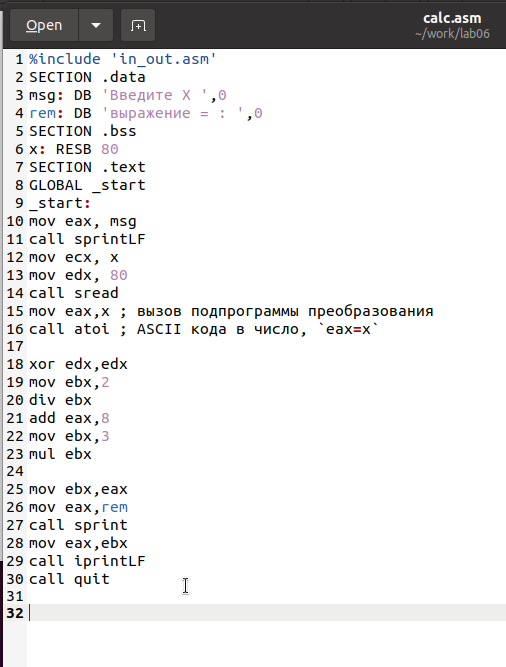


Figure 17: Программа в файле calc.asm

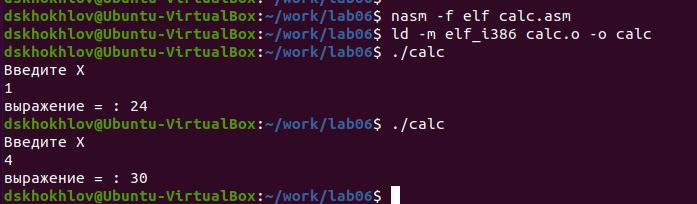


Figure 18: Запуск программы calc.asm

# 3 Выводы

Изучили работу с арифметическими операциями.

# Список литературы

1. GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/.
2. GNU Bash Manual. — 2016. — URL: https://www.gnu.org/software/bash/manual/.
3. Midnight Commander Development Center. — 2021. — URL: https://midnight-commander.org/.
4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O’Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658.
6. Robbins A. Bash Pocket Reference. — O’Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: https://www.opennet.ru/docs/RUS/nasm/.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm\_unix.
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).