Отчёт по лабораторной работе 6

дисциплина: Архитектура компьютеров

Байрамов Керим Сапарович

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

Создаю папку для программ лабораторной работы № 6, перехожу в неё и создаю файл lab6-1.asm.

Рассмотрим примеры программ, которые выводят символы и числа. Программы будут выводить значения из регистра eax.

В этой программе в регистр eax записывается символ ‘6’ (инструкция mov eax,‘6’), а в регистр ebx — символ ‘4’ (инструкция mov ebx,‘4’). Затем к значению в eax добавляется значение из ebx (инструкция add eax,ebx), и результат записывается обратно в eax. После этого выводим результат.

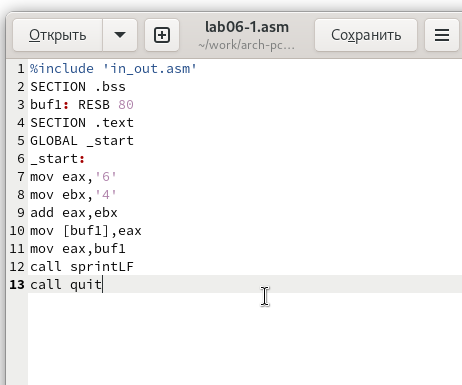


Рис. 1: Код программы lab6-1.asm

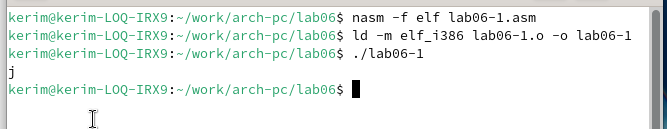


Рис. 2: Запуск программы lab6-1.asm

При выводе значения из eax ожидаем увидеть число 10. Однако вместо этого выводится символ ‘j’. Это связано с тем, что код символа ‘6’ в двоичном формате — 00110110 (54 в десятичной системе), а код символа ‘4’ — 00110100 (52). После сложения в eax получаем 01101010 (106), что соответствует символу ‘j’.

Теперь изменим программу и вместо символов запишем в регистры числа.

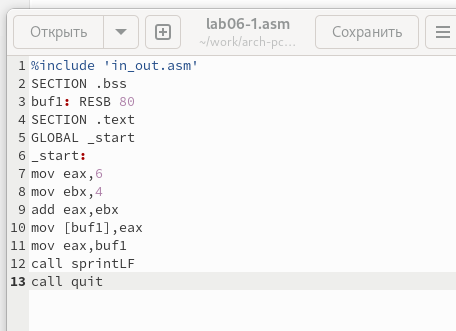


Рис. 3: Код программы lab6-1.asm с числами

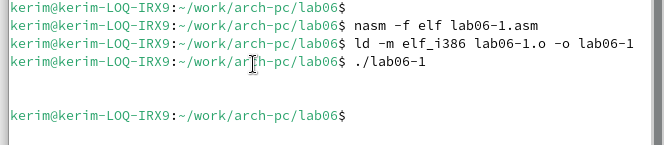


Рис. 4: Запуск программы lab6-1.asm с числами

Как и в предыдущем случае, при выполнении программы не получаем число 10. В этот раз выводится символ с кодом 10, что означает конец строки (возврат каретки). В консоли он не отображается, но добавляет пустую строку.

Для работы с числами в файле in\_out.asm есть подпрограммы, которые преобразуют символы ASCII в числа и обратно. Изменяем программу, используя эти функции.

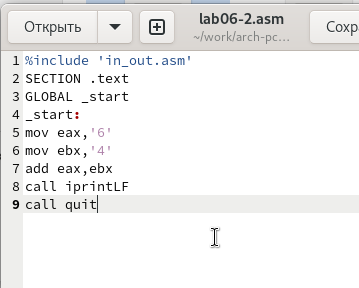


Рис. 5: Код программы lab6-2.asm

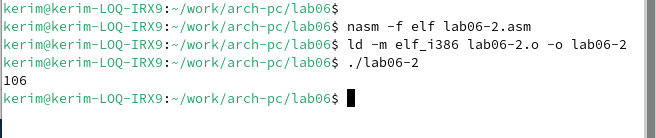


Рис. 6: Запуск программы lab6-2.asm

В результате выполнения программы мы получим число 106. Здесь команда add складывает коды символов ‘6’ и ‘4’ (54 + 52 = 106). Но, в отличие от предыдущей программы, функция iprintLF выводит число, а не соответствующий ему символ.

Теперь снова изменим символы на числа.

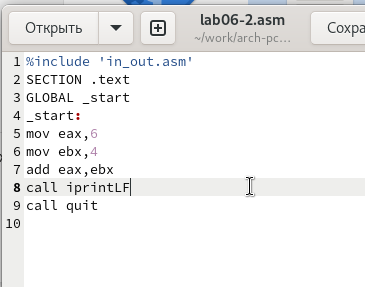


Рис. 7: Код программы lab6-2.asm с числами

Функция iprintLF позволяет вывести число, так как операндами являются числа. Поэтому получаем число 10.

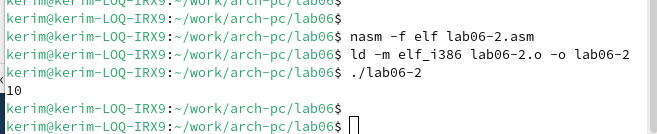


Рис. 8: Запуск программы lab6-2.asm с числами

Заменил функцию iprintLF на iprint. Создал исполняемый файл и запустил его. Вывод отличается тем, что нет переноса строки.

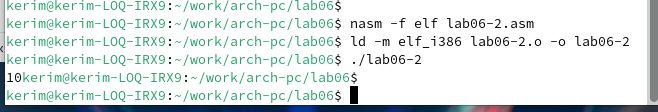


Рис. 9: Запуск программы lab6-2.asm без переноса строки

## 2.2 Выполнение арифметических операций в NASM

Для примера выполнения арифметических операций в NASM рассмотрим программу, вычисляющую выражение .

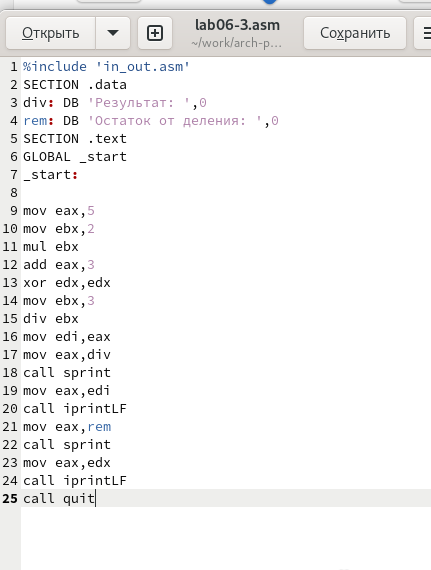


Рис. 10: Код программы lab6-3.asm

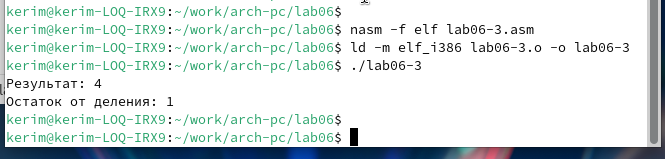


Рис. 11: Запуск программы lab6-3.asm

Изменяю программу для вычисления выражения . Создаю исполняемый файл и проверяю его работу.

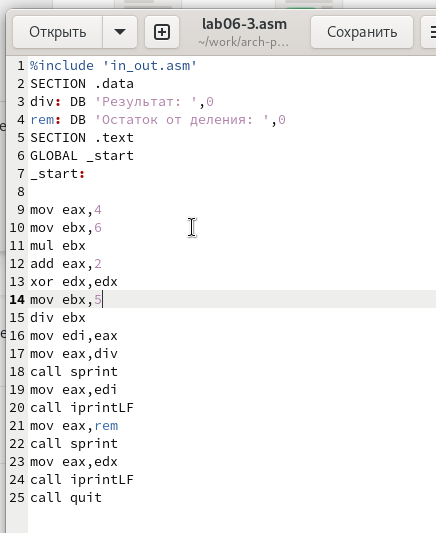


Рис. 12: Код программы lab6-3.asm с новым выражением

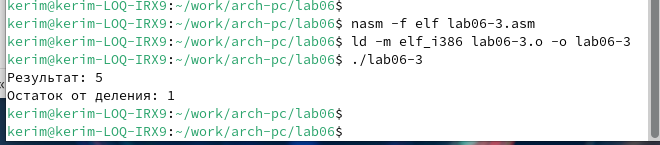


Рис. 13: Запуск программы lab6-3.asm с новым выражением

Рассмотрим ещё одну программу, вычисляющую вариант задания по номеру студенческого билета.

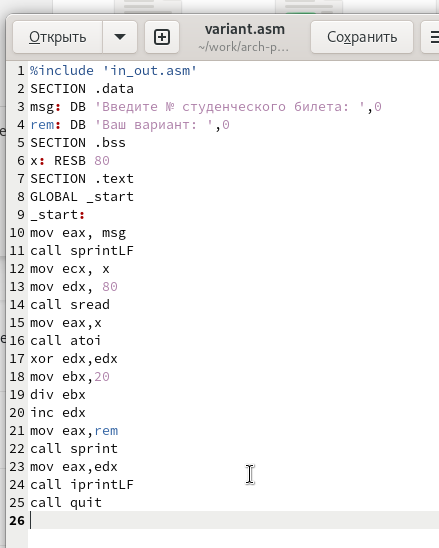


Рис. 14: Код программы variant.asm

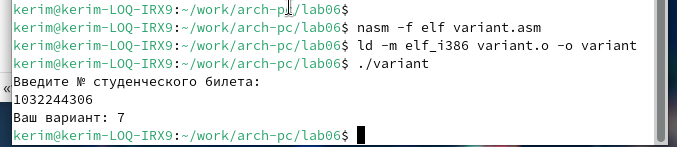


Рис. 15: Запуск программы variant.asm

Здесь число, над которым нужно выполнять арифметические операции, вводится с клавиатуры. Поскольку ввод осуществляется в символьном виде, символы нужно преобразовать в числа. Для этого можно использовать функцию atoi из файла in\_out.asm.

### 2.2.1 Ответы на вопросы

1. Какие строки отвечают за вывод сообщения ‘Ваш вариант:’?
   * Инструкция mov eax, rem загружает значение переменной с фразой ‘Ваш вариант:’ в регистр eax.
   * Инструкция call sprint вызывает подпрограмму для вывода строки.
2. Для чего нужны следующие инструкции?
   * Инструкция mov ecx, x перемещает значение переменной x в регистр ecx.
   * Инструкция mov edx, 80 перемещает значение 80 в регистр edx.
   * Инструкция call sread вызывает подпрограмму для считывания номера студенческого билета из консоли.
3. Для чего нужна инструкция call atoi?
   * Инструкция call atoi используется для преобразования введенных символов в числовой формат.
4. Какие строки отвечают за вычисления варианта?
   * Инструкция xor edx, edx обнуляет регистр edx.
   * Инструкция mov ebx, 20 загружает значение 20 в регистр ebx.
   * Инструкция div ebx делит номер студенческого билета на 20.
   * Инструкция inc edx увеличивает значение регистра edx на 1. Здесь происходит деление номера студенческого билета на 20, а в регистре edx хранится остаток, к которому прибавляется 1.
5. В какой регистр записывается остаток от деления при выполнении div ebx?
   * Остаток от деления записывается в регистр edx.
6. Для чего нужна инструкция inc edx?
   * Инструкция inc edx увеличивает значение в регистре edx на 1, как это предусмотрено формулой для вычисления варианта.
7. Какие строки отвечают за вывод результата вычислений на экран?
   * Инструкция mov eax, edx помещает результат вычислений в регистр eax.
   * Инструкция call iprintLF вызывает подпрограмму для вывода значения на экран.

## 2.3 Задание для самостоятельной работы

Напишите программу для вычисления выражения . Программа должна выводить формулу, запрашивать ввод значения x, вычислять выражение в зависимости от введенного x и выводить результат. Форму функции f(x) выберите из таблицы 6.3 вариантов заданий в соответствии с номером, полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его для значений x1 и x2 из 6.3. Получили вариант 7 - для .

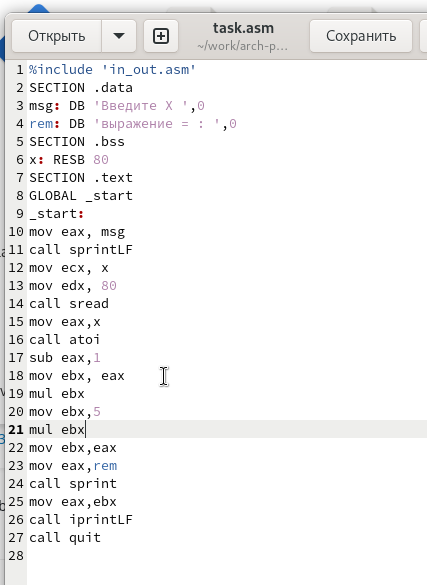


Рис. 16: Код программы task.asm

При результат — 20.

При результат — 80.

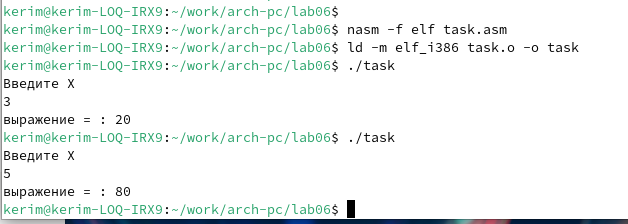


Рис. 17: Запуск программы task.asm

Программа работает корректно.

# 3 Выводы

Изучили работу с арифметическими операциями.