Отчёт по лабораторной работе 6

Архитектура компьютера

Тумуреева Галина Аркадьевна

Содержание

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

# 2 Выполнение лабораторной работы

1. Я создала папку для программ лабораторной работы номер шесть, затем перешла в неё и сформировала файл с именем lab6-1.asm.
2. Давайте рассмотрим примеры программ, которые отображают символы и числовые данные. Эти программы будут выводить информацию, которая была помещена в регистр eax.

В одной из программ в регистр eax мы помещаем символ ‘6’ (mov eax, ‘6’), а в регистр ebx символ ‘4’ (mov ebx, ‘4’). После этого мы складываем значения, хранящиеся в регистрах eax и ebx (add eax, ebx, и результат сложения сохранится в eax). Затем мы выводим полученный результат на экран. Однако, поскольку функция sprintLF требует, чтобы в регистре eax находился адрес, нам нужно воспользоваться дополнительной переменной. Сначала мы переносим значение из регистра eax в переменную buf1 (mov [buf1],eax), а потом записываем адрес переменной buf1 обратно в регистр eax (mov eax, buf1) и вызываем функцию sprintLF.

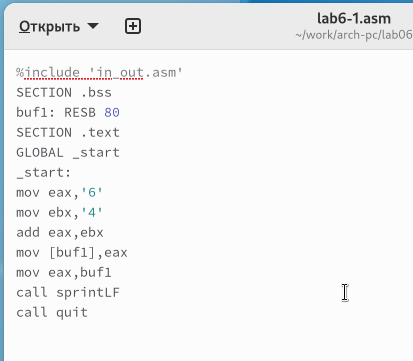


Figure 1: Программа в файле lab6-1.asm

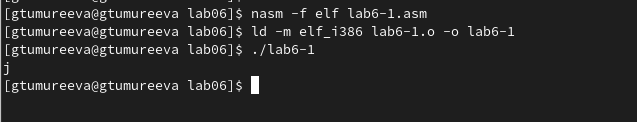


Figure 2: Запуск программы lab6-1.asm

Когда я смотрю на значение в регистре eax, я ожидаю увидеть цифру 10. Но вместо этого там отображается символ ‘j’. Это происходит из-за того, что двоичный код символа ‘6’ равен 00110110, что соответствует числу 54, а двоичный код символа ‘4’ – 00110100, или 52. Когда я использую команду add eax, ebx, то в регистр eax записывается их сумма – 01101010, что в десятичной системе равно 106, и это код для символа ‘j’.

1. Затем я внесла изменения в программу, чтобы в регистры записывались числа, а не символы.

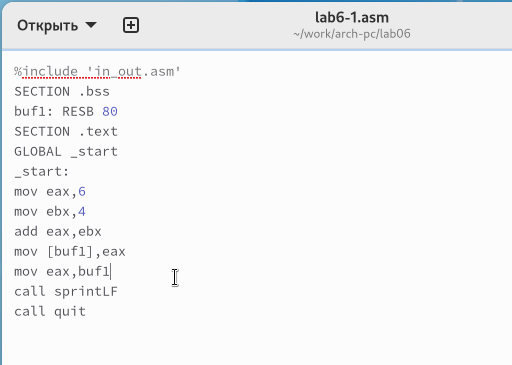


Figure 3: Программа в файле lab6-1.asm

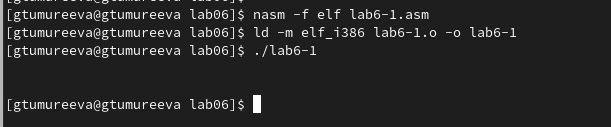


Figure 4: Запуск программы lab6-1.asm

Но даже после этих изменений, когда программа выполняется, она не показывает число 10. В этот раз выводится символ с кодом 10, который является символом конца строки. В консоли он не виден, но создает пустую строку.

1. Как я уже упоминала, для работы с числами в файле in\_out.asm были реализованы специальные подпрограммы, которые позволяют преобразовывать ASCII символы в числа и наоборот. Я использовала эти функции, чтобы изменить текст программы.

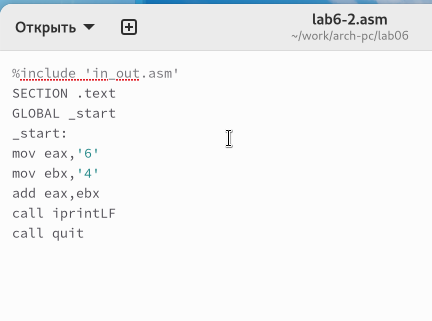


Figure 5: Программа в файле lab6-2.asm

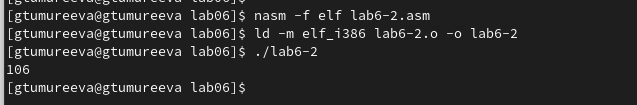


Figure 6: Запуск программы lab6-2.asm

Когда я запустила программу, она выдала мне число 106. Так же, как в первом случае, здесь функция add суммирует коды символов ‘6’ и ‘4’, что в сумме даёт 54+52=106. Но в отличие от предыдущей программы, здесь используется функция iprintLF, которая позволяет выводить число, а не символ, соответствующий этому числовому коду.

1. Подобно предыдущему примеру, я заменила символы на числа.

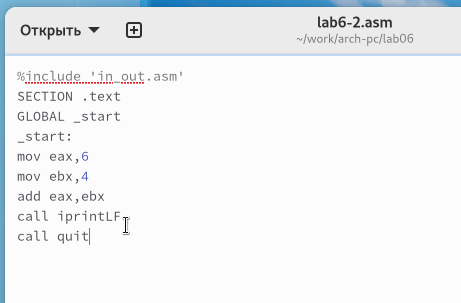


Figure 7: Программа в файле lab6-2.asm

Благодаря функции iprintLF, которая выводит числа, и тому, что в качестве операндов были использованы именно числа, а не коды символов, в результате получилось число 10.

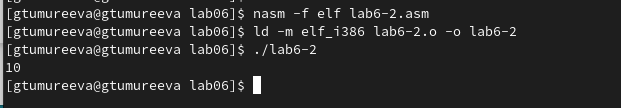


Figure 8: Запуск программы lab6-2.asm

Я изменила функцию iprintLF на iprint, собрала исполняемый файл и запустила его. Отличие заключалось в том, что теперь вывод не сопровождался переносом строки.

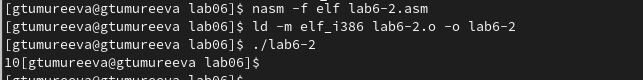


Figure 9: Запуск программы lab6-2.asm

1. В качестве примера, демонстрирующего выполнение арифметических операций в NASM, я написала программу для вычисления арифметического выражения

* .

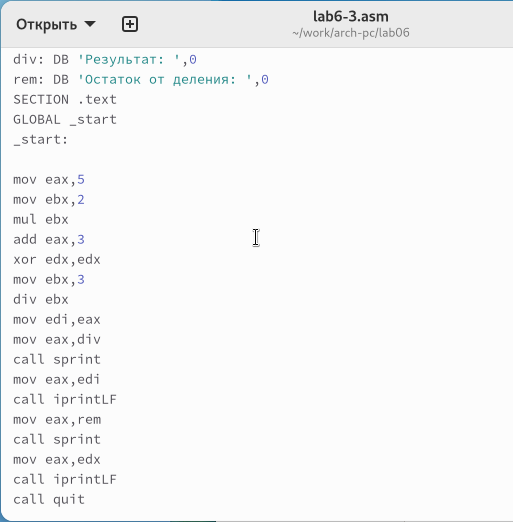


Figure 10: Программа в файле lab6-3.asm

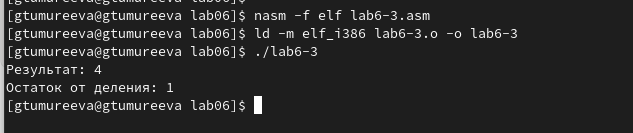


Figure 11: Запуск программы lab6-3.asm

Затем я изменила код программы, чтобы она вычисляла выражение

. После создания исполняемого файла я проверила, как он работает.

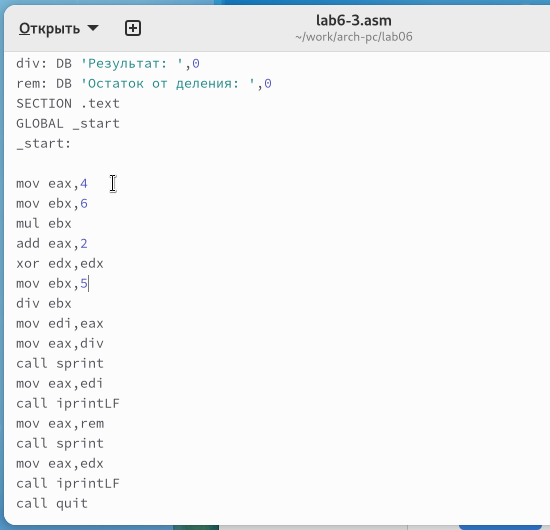


Figure 12: Программа в файле lab6-3.asm

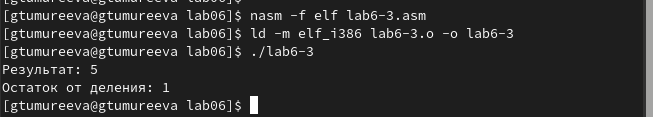


Figure 13: Запуск программы lab6-3.asm

1. Давайте возьмем для примера задачу, где нужно вычислить вариант упражнения на основе номера студенческого билета. Здесь нам придется работать с числом, которое мы введем через клавиатуру. Как я уже упоминала ранее, вводимые данные поступают в виде символов, и чтобы выполнять с ними математические операции в NASM, их нужно преобразовать в числовой формат. В этом может помочь функция atoi, которую можно найти в файле in\_out.asm.

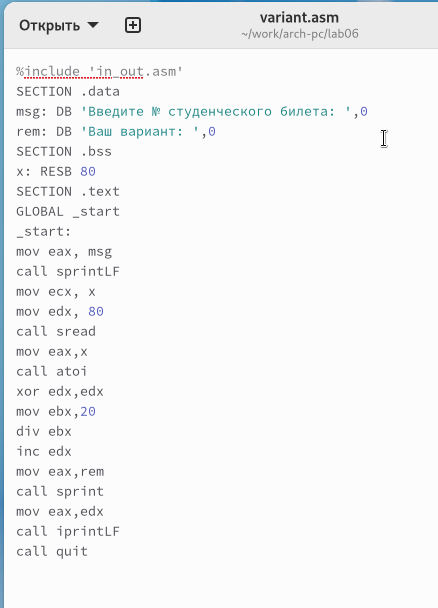


Figure 14: Программа в файле variant.asm

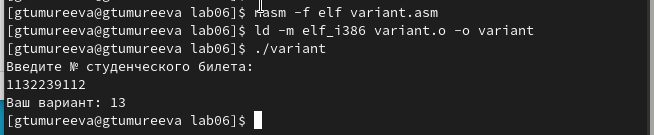


Figure 15: Запуск программы variant.asm

ответы на вопросы

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

* Команда mov eax, rem загружает в регистр значение, соответствующее строке “Ваш вариант:”.
* Использование call sprint приводит к выполнению функции, отображающей строку.

1. Для чего используется следующие инструкции?

mov ecx, x   
mov edx, 80   
call sread

Они используются для ввода номера студенческого билета и его сохранения в переменной X через терминал.

1. Для чего используется инструкция “call atoi”?

Данная функция преобразует введённые пользователем символы в числовое значение.

1. Какие строки листинга отвечают за вычисления варианта?

xor edx,edx  
mov ebx,20  
div ebx  
inc edx

Эти команды выполняют операцию деления номера студенческого билета на 20 и увеличивают остаток от деления на единицу.

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”?

Эта команда увеличивает значение в регистре edx на единицу, что необходимо для расчёта номера варианта по заданной формуле.

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

* mov eax, edx – перемещает результат вычислений в регистр eax.
* call iprintLF – инициирует функцию, которая выводит результат на экран с переводом строки.

1. 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 13 -

для

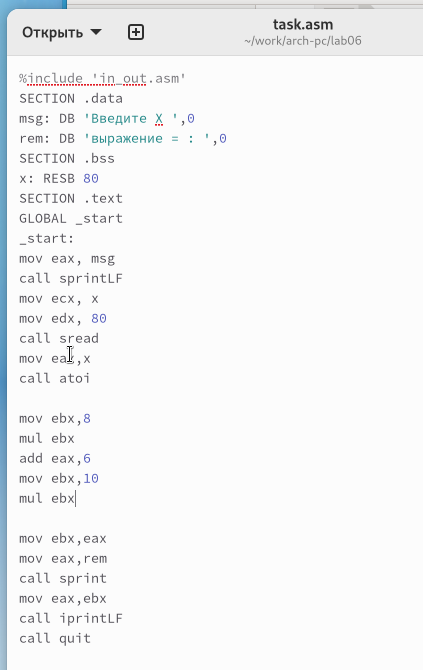


Figure 16: Программа в файле task.asm

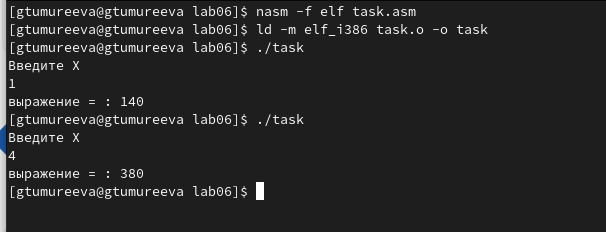


Figure 17: Запуск программы task.asm

# 3 Выводы

Изучили работу с арифметическими операциями.