Отчёт по лабораторной работе 6

Архитектура компьютеров и операционные системы

Горелашвили Лия Михайловна НКАбд-02-23

Содержание

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

# 2 Задание

1. Изучение арифметический операций в ассемблере
2. Изучение типов данных в ассемблере
3. Выполнение заданий, рассмотрение примеров
4. Выполнение заданий для самостоятельной работы

# 3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти.

Схема команды целочисленного сложения add (от англ. addition - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда.

Команда целочисленного вычитания sub (от англ. subtraction – вычитание) работает аналогично команде add.

Умножение и деление, в отличии от сложения и вычитания, для знаковых и беззнаковых чисел производиться по-разному, поэтому существуют различные команды. Для беззнакового умножения используется команда mul (от англ. multiply – умножение).

Для команд умножения один из сомножителей указывается в команде и должен находиться в регистре или в памяти, но не может быть непосредственным операндом. Второй сомножитель в команде явно не указывается и должен находиться в регистре EAX,AX или AL, а результат помещается в регистры EDX:EAX, DX:AX или AX, в зависимости от размера операнда.

Для деления, как и для умножения, существует 2 команды div (от англ. divide - деление) и idiv.

В командах указывается только один операнд – делитель, который может быть регистром или ячейкой памяти, но не может быть непосредственным операндом. Местоположение делимого и результата для команд деления зависит от размера делителя. Кроме того, так как в результате деления получается два числа – частное и остаток, то эти числа помещаются в определённые регистры 6.2.

# 4 Выполнение лабораторной работы

## 4.1 Символьные и численные данные в NASM

Создала каталог для программ лабораторной работы №6, перешла в него и создала файл с названием “lab6-1.asm”.

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

В данной программе, в регистр eax записан символ ‘6’, а в регистр ebx символ ‘4’. Затем мы прибавляем значение регистра ebx к значению в регистре eax (результат сложения будет записан в регистр eax). После этого мы выводим результат.

Так как для работы функции sprintLF в регистр eax должен быть записан адрес, мы используем дополнительную переменную. Мы записали значение регистра eax в переменную с именем “buf1”, а затем записали адрес переменной buf1 в регистр eax и вызвали функцию sprintLF.

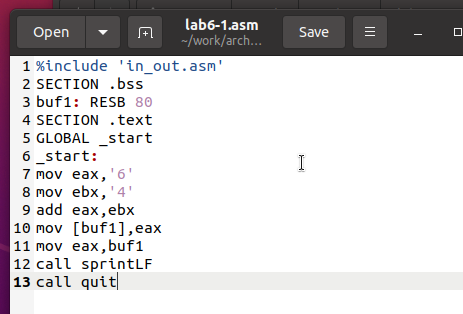


Figure 1: Редактирование файла lab6-1.asm

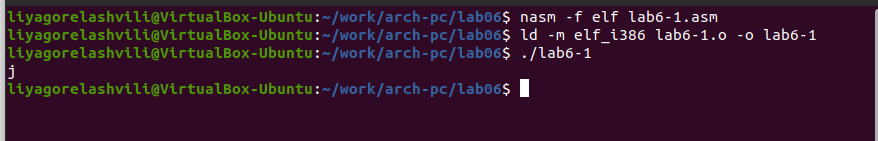


Figure 2: Проверка кода lab6-1.asm

В данном случае, при выводе значения регистра eax, ожидалось увидеть число 10. Однако, результатом был символ ‘j’. Это произошло 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 записала в регистр eax сумму кодов – 01101010 (106), что в свою очередь является кодом символа ‘j’.

Далее был изменен текст программы и вместо символов записаны числа.

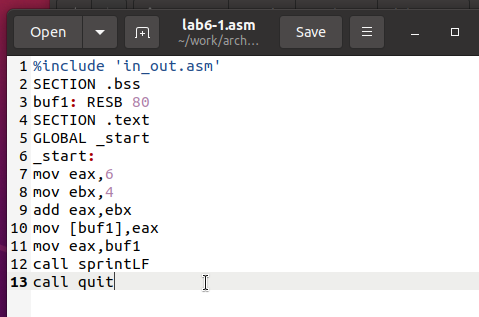


Figure 3: Редактирование файла lab6-1.asm

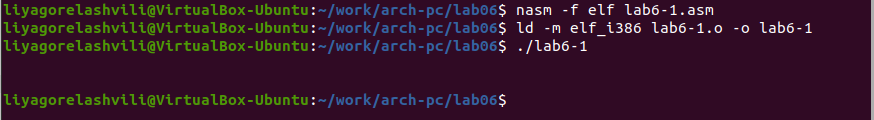


Figure 4: Проверка кода lab6-1.asm

В процессе выполнения программы не получили ожидаемое число 10. Вместо этого был выведен символ с кодом 10. Это символ конца строки (возврат каретки), который в консоли не отображается, но добавляет пустую строку.

В файле “in\_out.asm” реализованы подпрограммы для работы с числами и преобразования символов ASCII. Был модифицирован текст программы с использованием этих функций.

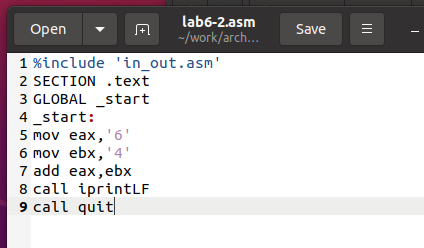


Figure 5: Редактирование файла lab6-2.asm

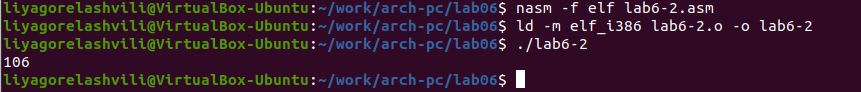


Figure 6: Проверка кода lab6-2.asm

В результате выполнения обновленной программы было выведено число 106. Здесь, как и в первом случае, команда add складывает коды символов ‘6’ и ‘4’ (54 + 52 = 106). Но в отличие от предыдущей версии, функция iprintLF позволяет напечатать само число, а не символ с соответствующим кодом.

По аналогии с предыдущим примером, были заменены символы на числа.

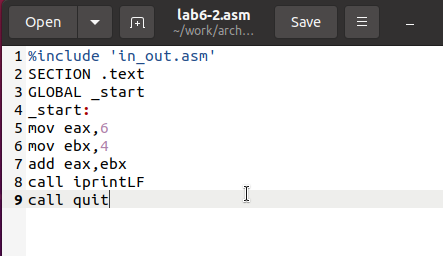


Figure 7: Редактирование файла lab6-2.asm

Функция iprintLF позволяет выводить числа, и на этот раз в качестве операндов использовались именно числа, а не коды символов. В результате мы получили число 10.

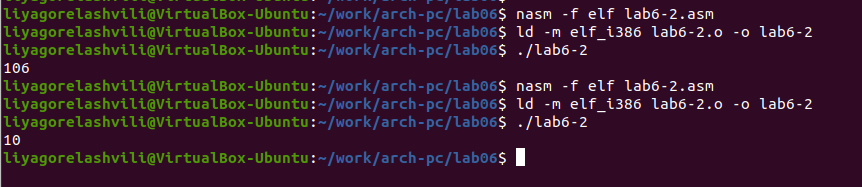


Figure 8: Проверка кода lab6-2.asm

Далее была заменена функция iprintLF на iprint, создан исполняемый файл и запущен. Вывод теперь отличается отсутствием перехода на новую строку.

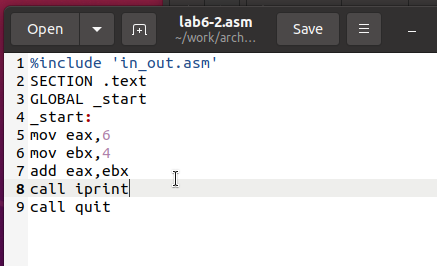


Figure 9: Редактирование файла lab6-2.asm

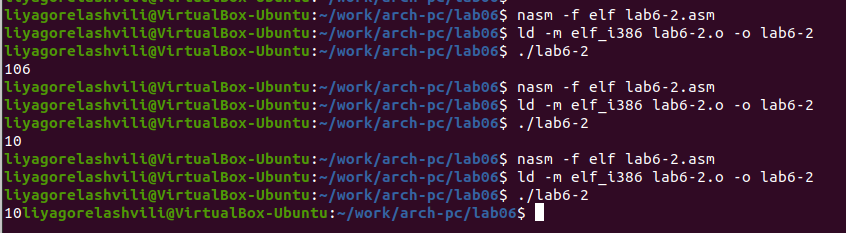


Figure 10: Проверка кода lab6-2.asm

## 4.2 Выполнение арифметических операций в NASM

В качестве примера выполнения арифметических операций в NASM рассмотрим программу для вычисления арифметического выражения

.

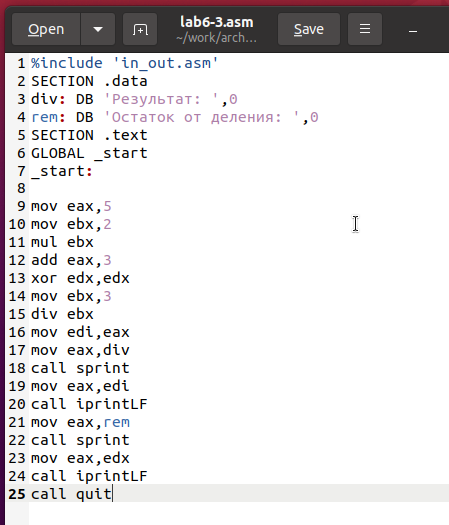


Figure 11: Редактирование файла lab6-3.asm

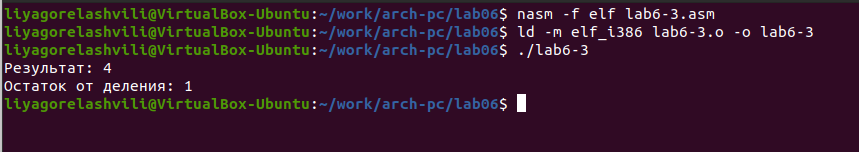


Figure 12: Проверка кода lab6-3.asm

Изменила текст программы для вычисления выражения

. Создала исполняемый файл и проверила его работу.

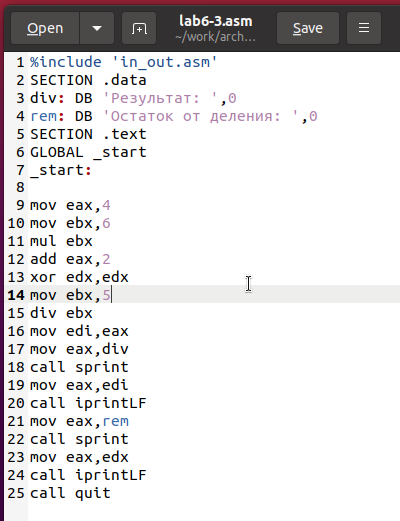


Figure 13: Редактирование файла lab6-3.asm

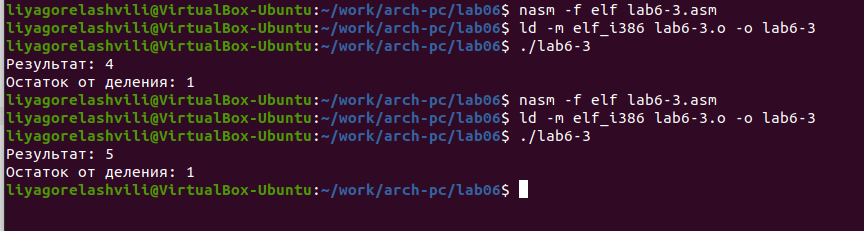


Figure 14: Проверка кода lab6-3.asm

В качестве еще одного примера рассмотрим программу для вычисления варианта задания на основе номера студенческого билета.

В этом случае число, над которым нужно выполнять арифметические операции, вводится с клавиатуры. Как уже отмечалось ранее, ввод с клавиатуры осуществляется в символьном виде. Для корректной работы арифметических операций в NASM эти символы необходимо преобразовать в числовой формат. С этой целью можно использовать функцию atoi из файла in\_out.asm. Она конвертирует строку символов в эквивалентное десятичное число.

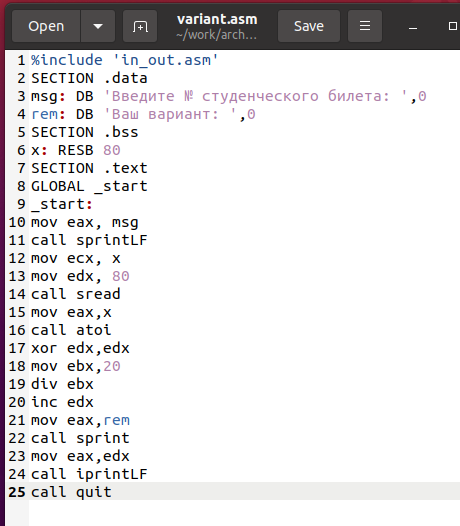


Figure 15: Редактирование файла variant.asm

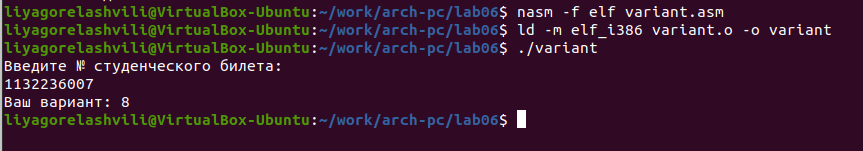


Figure 16: Проверка кода variant.asm

### 4.2.1 Ответы на вопросы по программе variant.asm

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

Ответ: Строки, отвечающие за вывод сообщения “Ваш вариант:”, - это строки, где происходит перемещение фразы в регистр eax с помощью инструкции mov eax, rem, а затем вызов подпрограммы вывода строки с помощью инструкции call sprint.

1. Для чего используется следующие инструкции?

Ответ:

* mov ecx, x: Инструкция mov ecx, x используется для сохранения значения регистра ecx в переменной x.
* mov edx, 80: Инструкция mov edx, 80 используется для присваивания значения 80 регистру edx.
* call sread: Инструкция call sread используется для вызова подпрограммы, которая считывает данные из консоли.

1. Для чего используется инструкция “call atoi”?

Ответ: Инструкция call atoi используется для преобразования введенных символов в числовой формат.

1. Какие строки листинга отвечают за вычисления варианта?

Ответ: Строки, отвечающие за вычисление варианта, включают следующие инструкции:

* xor edx, edx: Инструкция xor edx, edx используется для обнуления регистра edx.
* mov ebx, 20: Инструкция mov ebx, 20 используется для присваивания значения 20 регистру ebx.
* div ebx: Инструкция div ebx используется для деления номера студента на 20.
* inc edx: Инструкция inc edx используется для увеличения значения регистра edx на 1.

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

Ответ: При выполнении инструкции div ebx остаток от деления записывается в регистр edx.

1. Для чего используется инструкция “inc edx”?

Ответ: Инструкция inc edx используется для увеличения значения регистра edx на 1, что необходимо для вычисления варианта по формуле.

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

Ответ: Строки, отвечающие за вывод на экран результата вычислений, включают следующие инструкции:

* mov eax, edx: Инструкция mov eax, edx используется для помещения результата в регистр eax.
* call iprintLF: Инструкция call iprintLF используется для вызова подпрограммы вывода результата.

## 4.3 Выполнение заданий для самостоятельной работы

Написала программу для вычисления выражения y = f(x). Программа выводит выражение для вычисления, запрашивает ввод значения x, вычисляет заданное выражение в зависимости от введенного x и выводит результат вычислений. Для выбора вида функции f(x) использовала таблицу 6.3 вариантов заданий, в соответствии с номером, полученным при выполнении лабораторной работы.

Создала исполняемый файл и проверила его работу для значений x1 и x2 из таблицы 6.3.

Вариант 8 - для

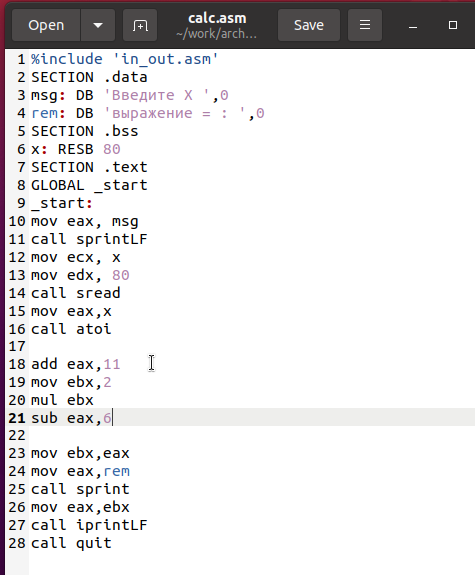


Figure 17: Редактирование файла calc.asm

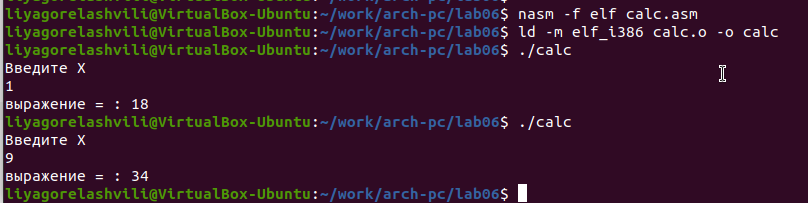


Figure 18: Проверка кода calc.asm

Программа считает верно.

# 5 Выводы

Изучили работу с арифметическими операциями.