Отчёт по лабораторной работе №6

Дисциплина: Архитектура компьютера

Осина Виктория Александровна

Содержание

# 1 Цель работы

Целью данной лабораторной работы является освоение арифметических инструкций языка ассемблера NASM.

# 2 Задание

1. Освоение вывода символьных и численных значений.
2. Выполнение арифметических операций.
3. Выполнение задания для самостоятельной работы.

# 3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес опе- ранда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации. Существует три основных способа адресации: • Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: mov ax,bx. • Непосредственная адресация – значение операнда задается непосредственно в ко- манде, Например: mov ax,2. • Адресация памяти – операнд задает адрес в памяти. В команде указывается символи- ческое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

Схема команды целочисленного сложения add (от англ. addition - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда. Команда add работает как с числами со знаком, так и без знака.

Команда целочисленного вычитания sub (от англ. subtraction – вычитание) работает анало- гично команде add.

Прибавление единицы называется инкрементом, а вычитание — декрементом. Для этих операций существуют специальные команды: inc (от англ. increment) и dec (от англ. decrement), которые увеличивают и уменьшают на 1 свой операнд.

Команда neg рассматривает свой операнд как число со знаком и меняет знак операнда на противоположный.

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 знакового умножения используется команда imul.

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

для преобразования ASCII символов в числа и обратно реализованы следующие подпрограммы: • iprint – вывод на экран чисел в формате ASCII, перед вызовом iprint в регистр eax необходимо записать выводимое число (mov eax,). • iprintLF – работает аналогично iprint, но при выводе на экран после числа добавляет к символ перевода строки. • atoi – функция преобразует ascii-код символа в целое число и записает результат в регистр eax, перед вызовом atoi в регистр eax необходимо записать число (mov eax,).

# 4 Выполнение лабораторной работы

## 4.1 Освоение вывода символьных и численных значений

Создаю каталог для программ лабораторной работы №6, перехожу в него и создаю файл lab6-1.asm (рис. [1](#fig:001))

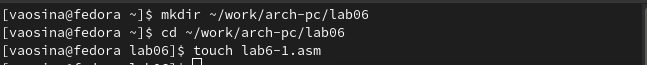


Figure 1: Создание каталога lab06 и файла lab6-1.asm в нем

Перед работой с программами копирую файл in\_out.asm в каталог(рис. [2](#fig:002)) (рис. [3](#fig:003))

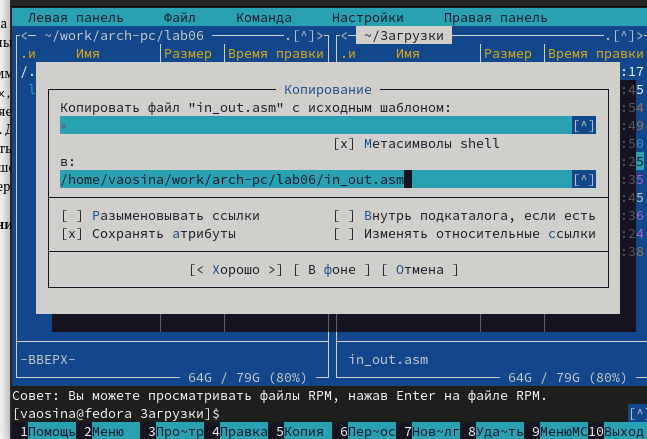


Figure 2: Копирование файла in\_out.asm

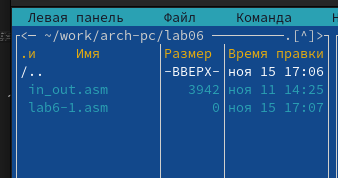


Figure 3: Проверка, что файл находится в нужном каталоге

Ввожу в файл lab6-1.asm текст программы (рис. [4](#fig:004)).

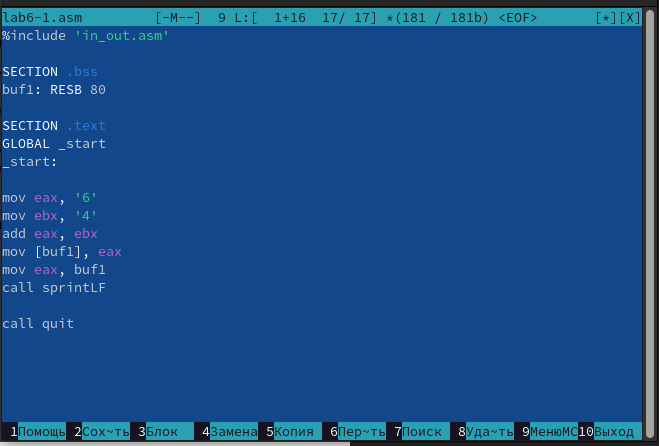


Figure 4: Ввод текста программы в файл

Создаю исполняемый файл и запускаю его (рис. [5](#fig:005)). В данном случае при выводе значения регистра eax мы ожидаем увидеть число 10. Однако результатом будет символ j. Это происходит 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 запишет в регистр eax сумму кодов – 01101010 (106), что в свою очередь является кодом символа j

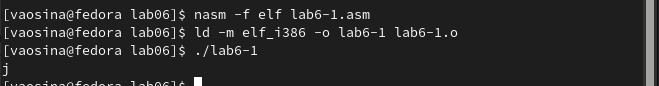


Figure 5: Создание исполняемого файла и его запуск

Изменяю текст программы и вместо символов, записываю в регистры числа (рис. [6](#fig:006)).

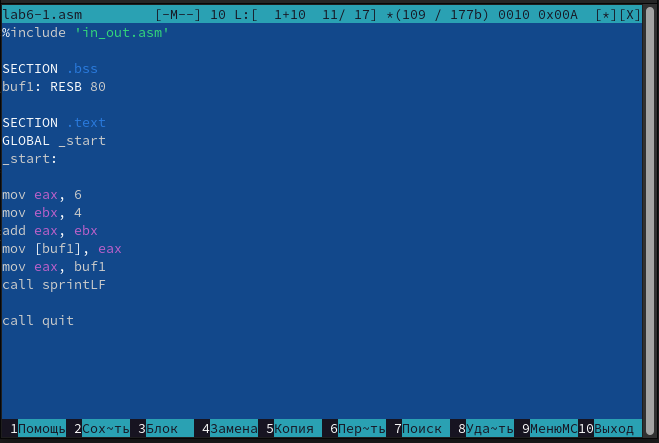


Figure 6: Изменение в тексте программы символов на числа

Создаю исполняемый файл и запускаю его. (рис. [7](#fig:007)) Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10, при этом этот символ не отображается при выводе на экран. Код 10 соответствует символу переноса строки

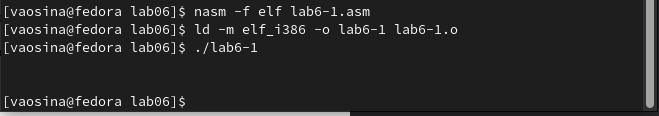


Figure 7: Создание исполняемого файла и его запуск

Создаю файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и ввожу в него текст про- граммы (рис. [8](#fig:081)) (рис. [9](#fig:008)).

Figure 8: Создание файла lab6-2.asm

Figure 8: Создание файла lab6-2.asm



Figure 9: Ввод текста программы в файл

Создаю исполняемый файл и запускаю его. (рис. [10](#fig:009)). В результате работы программы получаем число 106. В данном случае, как и в первом, команда add складывает коды символов ‘6’ и ‘4’ (54+52=106). Однако функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

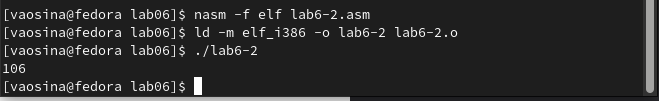


Figure 10: Создание исполняемого файла и его запуск

Изменяю текст программы и вместо символов, записываю в регистры числа (рис. ??) Создаю исполняемый файл и запускаю его. (рис. ??). Программа складывает не коды символов, а сами числа, поэтому в результате работы программы получаем 10.

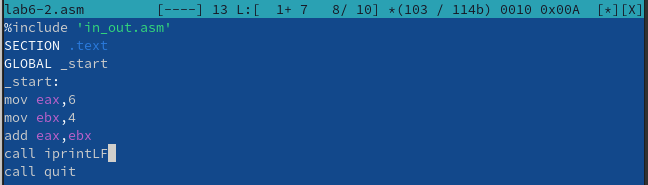


Figure 11: Изменение в тексте программы символов на числа

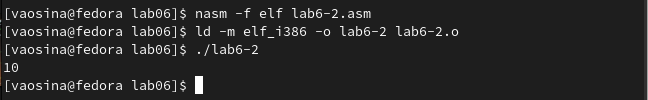


Figure 12: Создание исполняемого файла и его запуск

Заменяю функцию iprintLF на iprint. (рис. [13](#fig:010))

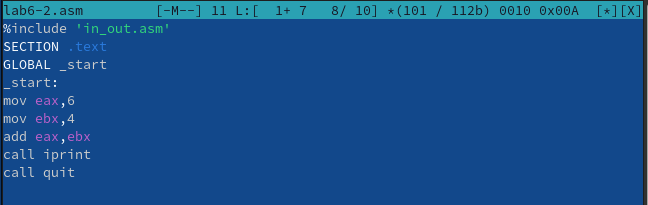


Figure 13: Изменение в тексте программы функции iprintLF на iprint

Создаю исполняемый файл и запускаю его. (рис. [14](#fig:011)) Вывод функций iprintLF и iprint отличается тем, что функция iprintLF после числа добавляет символ переноса строки, а iprint нет.

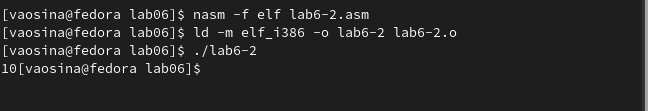


Figure 14: Создание исполняемого файла и его запуск

## 4.2 Выполнение арифметических операций

Создаю файл lab6-3.asm в каталоге ~/work/arch-pc/lab06 (рис. [15](#fig:012)).

Figure 15: Создание файла lab6-3.asm

Figure 15: Создание файла lab6-3.asm

Ввожу текст программы для вычисления арифметического выражения 𝑓(𝑥) = (5\*2+3)/3.(рис. [16](#fig:013)).

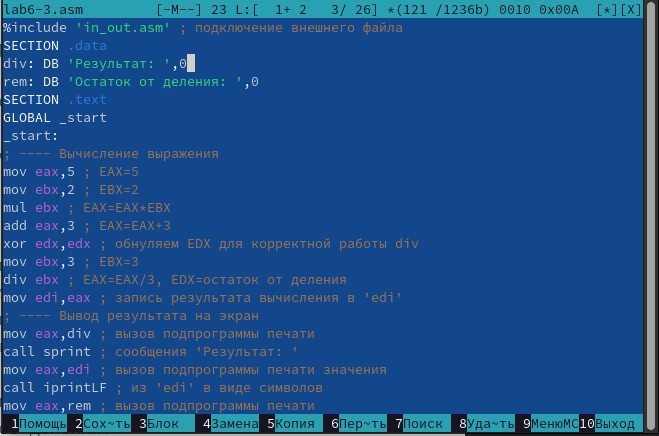


Figure 16: Ввод текста программы

Создаю исполняемый файл и запускаю его. (рис. [17](#fig:014)).

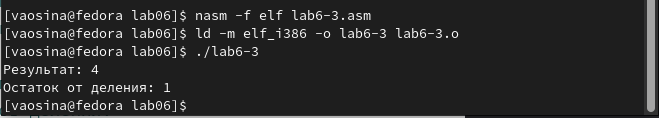


Figure 17: Создание исполняемого файла и его запуск

Изменяю текст программы для вычисления выражения 𝑓(𝑥) = (4\*6+2)/5 (рис. [18](#fig:015))

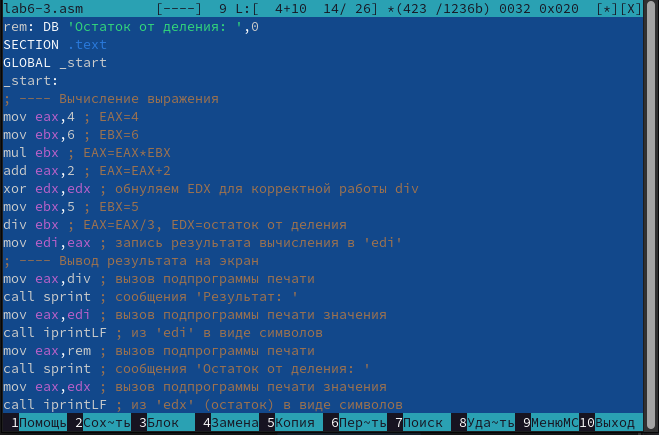


Figure 18: Изменение текста программы для вычисления другого выражения

Создаю исполняемый файл и запускаю его. (рис. [19](#fig:016)).

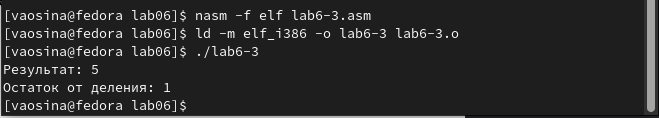


Figure 19: Создание исполняемого файла и его запуск

Создаю файл variant.asm в каталоге ~/work/arch-pc/lab06 (рис. [20](#fig:113)). Ввожу текст программы для вычисления варианта задания по номеру студенческого билета (рис. [21](#fig:017))

Figure 20: Создание файла variant.asm

Figure 20: Создание файла variant.asm

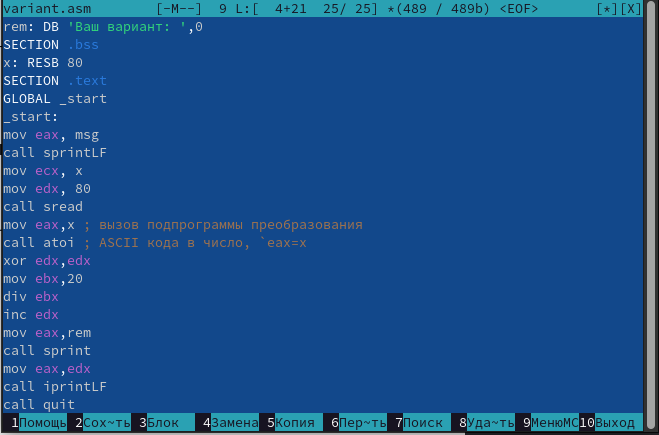


Figure 21: Ввод текста программы

Создаю исполняемый файл и запускаю его.(рис. [22](#fig:018))

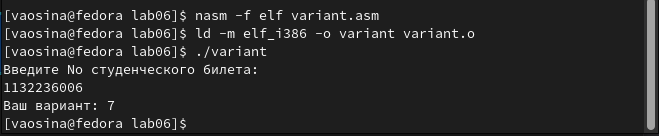


Figure 22: Создание исполняемого файла и его запуск

Проверяю результат работы программы вычислив номер варианта аналитически: 1132236006 делится на 20, в остатке остается 6, к этому значению прибавляется 1, получается 7

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

``` mov eax,rem call sprint ```

1. Для чего используется следующие инструкции?

mov ecx, x - для того, чтобы поместить адрес вводимой строки х в регистр ecx mov edx, 80 - запись в регитр edx длины вводимой строки call sread - вызов подпрограммы, обеспечивающей ввод сообщения с клавиатуры

1. Для чего используется инструкция call atoi?

Для преобразования ASCII кода в число

1. Какие строки отвечают за вычисления варианта?

За вычисление варианта отвечают:

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

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

Остаток от деления при выполнении функции div записывается в регистр edx

1. Для чего используется инструкция inc edx?

Инструкция inc edx используется для увеличения значения edx на 1

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

За вывод на экран результата вычислений отвечают:

``` mov eax, edx call iprintLF ```

## 4.3 Выполнение задания для самостоятельной работы

Необходимо написать программу вычисления выражения 𝑦 = 𝑓(𝑥) Согласно моему варианту, полученному при выполнении лабораторной работы, это выражение 5(𝑥−1)^2. Проверить работу программы вычисления нужно, используя значение переменных х1 (3) и х2 (5).

Создаю файл var7.asm в каталоге ~/work/arch-pc/lab06 (рис. [23](#fig:019))

Figure 23: Создание файла var7.asm

Figure 23: Создание файла var7.asm

Ввожу текст программы (рис. [24](#fig:020)).

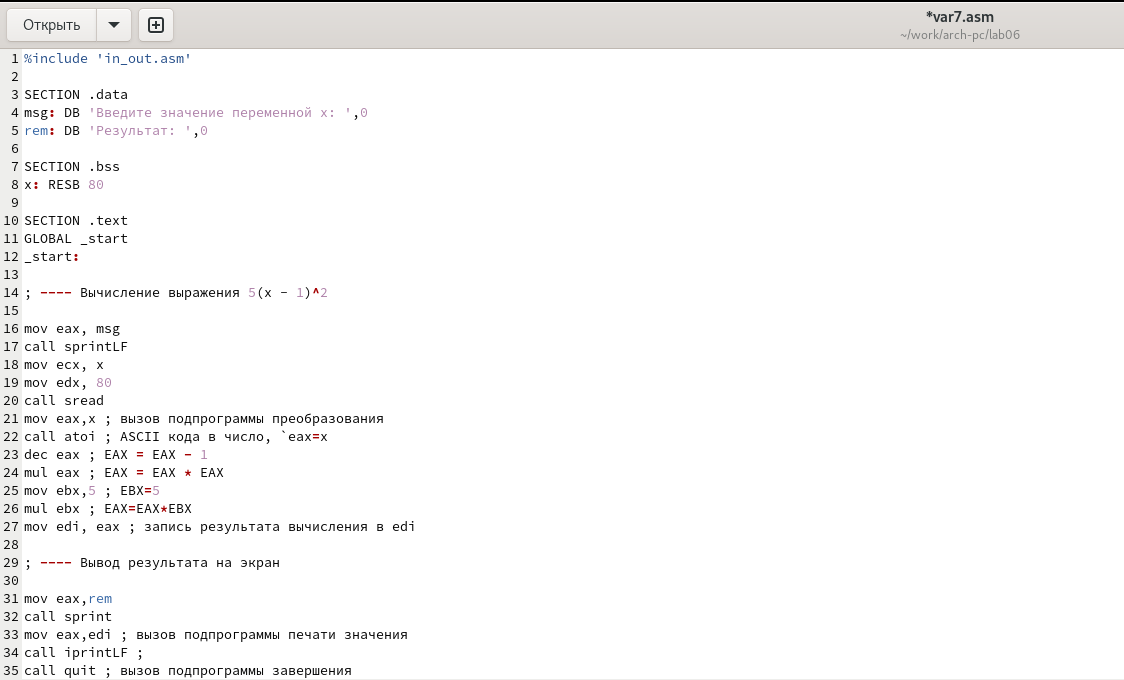


Figure 24: Ввод текста программы

Создаю исполняемый файл и запускаю его. Проверяю работу программы, испульзуя сначала х1 = 3, а затем х2 = 5 (рис. [25](#fig:021)).

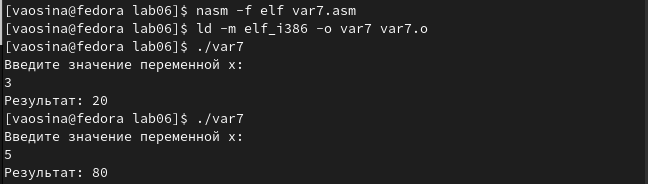


Figure 25: Создание исполняемого файла и его запуск

Действительно, если подставить 3, то 5*(3-1)^2 = 5*2^2 5*4 = 20. Если подставить 5: 5*(5-1)^2 = 5*4^2 = 5*16= 80.

# 5 Выводы

При выполнении лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

# 6 Список литературы

1. [Архитектура ЭВМ](https://esystem.rudn.ru/pluginfile.php/2089086/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%966.%20%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5%20%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8%20%D0%B2%20NASM..pdf)