Лабораторная работа №6

Архитектура вычислительных систем

Курникова Елизавета Денисовна

Содержание

# 1 Цель работы

Освоить арифметические инструкций языка ассемблера NASM.

# 2 Задание

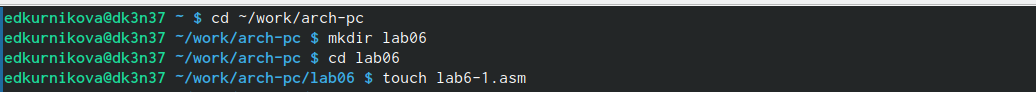
Написать программу вычисления выражения. Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x, вычислять заданное выражение в зависимости от введенного x, выводить результат вычислений. Вид функции выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создайть исполняемый файл и проверить его работу для значений из 6.3.

# 3 Теоретическое введение

1. Адресация в NASM 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 команде, Например: mov ax,2. • Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.
2. Арифметические операции в NASM 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.
3. Целочисленное вычитание sub Команда целочисленного вычитания sub (от англ. subtraction – вычитание) работает аналогично команде add.
4. Команды инкремента и декремента Довольно часто при написании программ встречается операция прибавления или вычитания единицы. 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.
5. Команда изменения знака операнда neg Команда рассматривает свой операнд как число со знаком и меняет знак операнда на противоположный. Операндом может быть регистр или ячейка памяти любого размера.
6. Команды умножения mul и imul 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.
7. Команды деления div и idiv Для деления, как и для умножения, существует 2 команды div (от англ. divide - деление) и idiv. Для беззнакового умножения используется команда div. Для знакового умножения используется команда idiv.

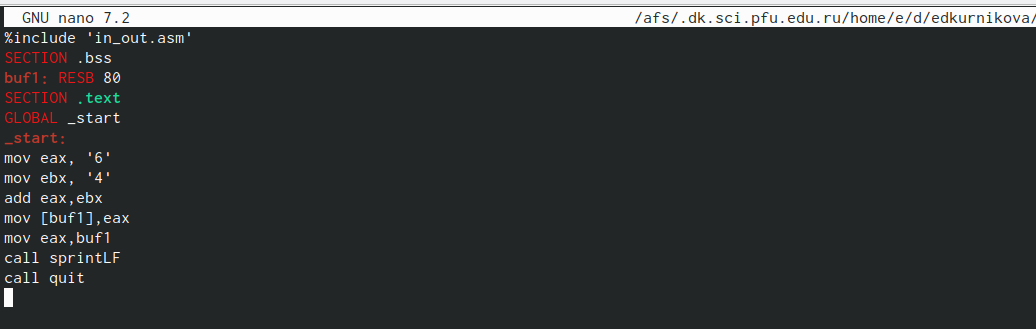
# 4 Выполнение лабораторной работы

1. Создаём каталог для программам лабораторной работы No 7, перейдём в него и создаём файл lab6-1.asm



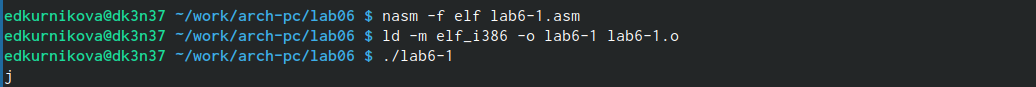
61.png

1. Введем в файл lab6-1.asm текст программы из листинга 6.1.



62.png

1. Создадим исполняемый файл и запустим его.



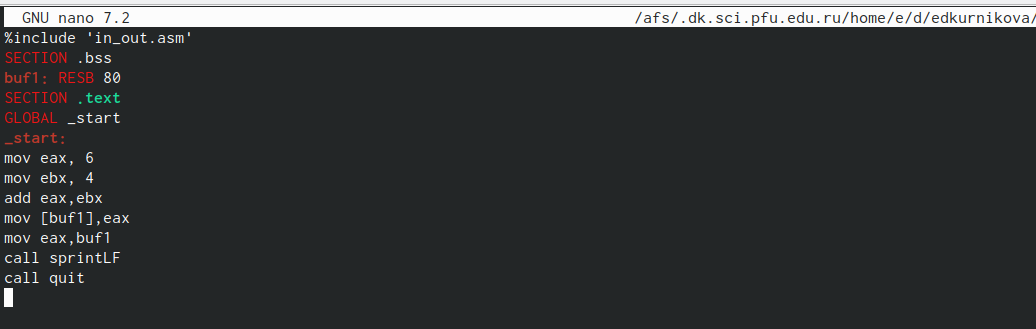
64.png

1. Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправим текст программы.

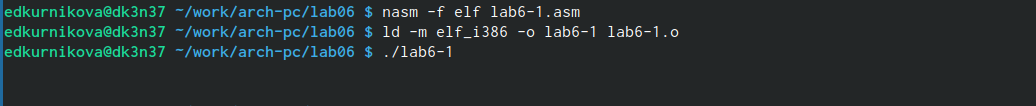


65.png

1. Создадим исполняемый файл и запустим его (6-1).

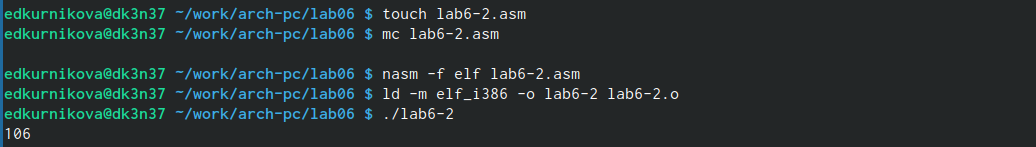


66.png



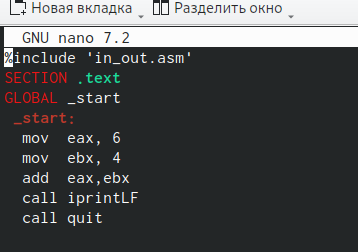
67.png

1. Создадим файл lab6-2.asm в каталоге. Введем в него текст программы из листинга 6.2 и запустим его.



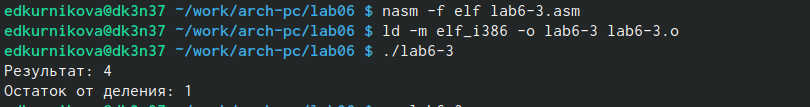
68.png

1. Изменим символы на числа в lab6-2. Создадим исполняемый файл и запустим его.



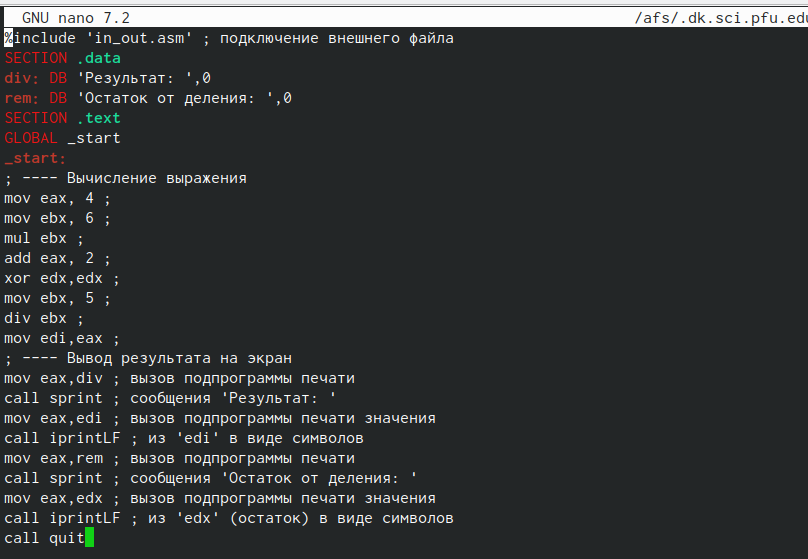
69.png

1. Создадим файл lab6-3.asm в каталоге. Введем в файл lab6-3.asm текст программы из листинга 6.3



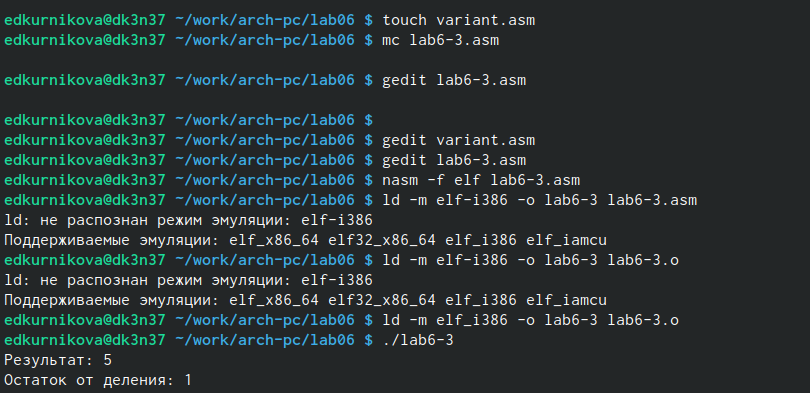
610.png

1. Создадим исполняемый файл и запустим его.



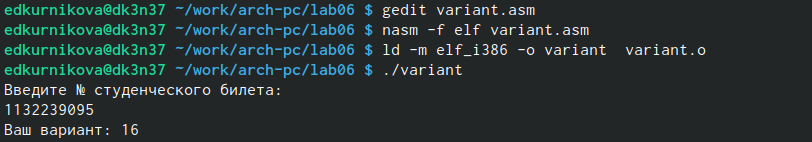
611.png

1. Введем в файл lab6-3 программу вычисления выражения .



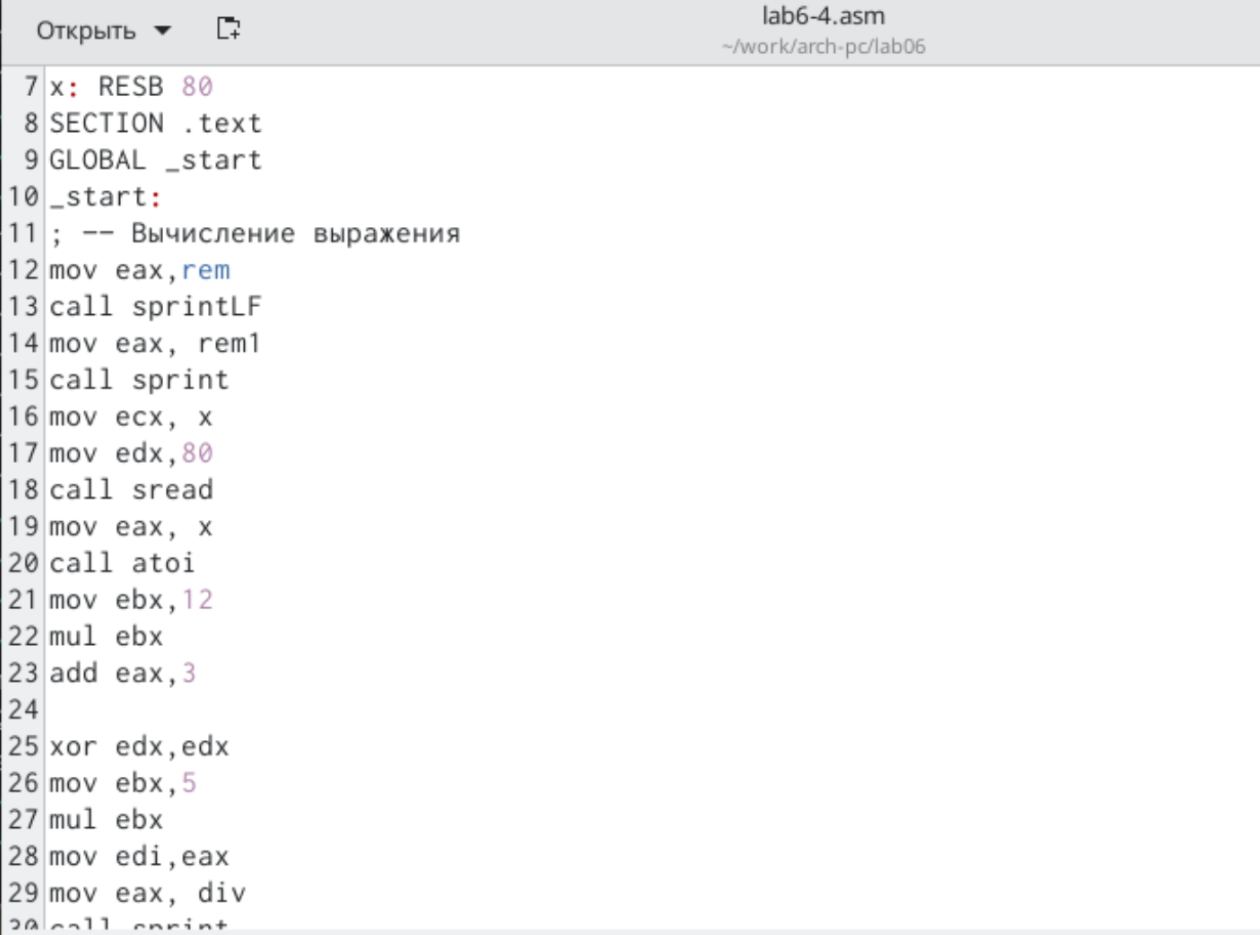
612.png

1. Создадим файл variant.asm в каталоге ~/work/arch-pc/lab06:



614png

1. Составляем программу для нашего варианта lab6-4 (Самостоятельная работа).



616png

1. Запускаем программу и вводим два числа из условия, убеждаемся что программа работает верно.



617png

# 5 Ответы на вопросы:

1. строки листинга 7.4 отвечают за вывод на экран сообщения ’Ваш вариант: mov eax и rem call sprint;
2. mov ecx,x - запись входной переменной в регистр ecx; mov edx, 80 - запись размера переменной в регистр edx; call sread - вызов процедуры чтения данных;
3. call atoi - функция преобразующая ASCII код символа в целое число и записывающая результат в регистр eax;
4. xor edx, edx mov ebx, 20 div ebx, inc edx;
5. div ebx - ebx;
6. inc - используется для увеличения операнда на единицу;
7. Следующие строки листинга отвечают за вывод на экран результата вычис- лений mov eax, rem call sprint mov eax, edx call iprintLF.

# 6 Выводы

В ходе выполнения данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

# Список литературы