Отчет по лабораторной работе №6

Архитектура компьютера и операционные системы

Александр Дмитриевич Собко

Содержание

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM

# 2 Задание

Изменить несколько программ с вводом, выводом и вычислением, затем написать свою

# 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. 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 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.

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

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.

Еще одна команда, которую можно отнести к арифметическим командам это команда изменения знака neg

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

Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. Расширенная таблица ASCII состоит из двух частей. Первая (символы с кодами 0-127) является универсальной, а вторая (коды 128-255) предназначена для специальных символов и букв национальных алфавитов и на компьютерах разных типов может меняться. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно. Для выполнения лабораторных работ в файле in\_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Это: • 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 Символьные и численные данные в NASM

1. Создайте каталог для программам лабораторной работы № 6, перейдите в него и создайте файл lab6-1.asm:
2. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр eax

Введите в файл lab6-1.asm текст программы из листинга 6.1. В данной программе в регистр eax записывается символ 6 (mov eax,‘6’), в регистр ebx символ 4 (mov ebx,‘4’). Далее к значению в регистре eax прибавляем значение регистра ebx (add eax,ebx, результат сложения запишется в регистр eax). Далее выводим результат. Так как для работы функции sprintLF в регистр eax должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра eax в переменную buf1 (mov [buf1],eax), а затем запишем адрес переменной buf1 в регистр eax (mov eax,buf1) и вызовем функцию sprintLF.



Рис. 1: Рисунок1

Создайте исполняемый файл и запустите его.

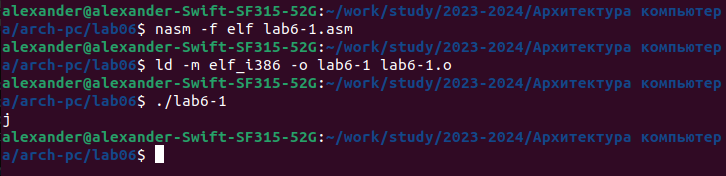


Рис. 2: Рисунок2

1. Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправьте текст программы (Листинг 6.1) следующим образом: замените строки

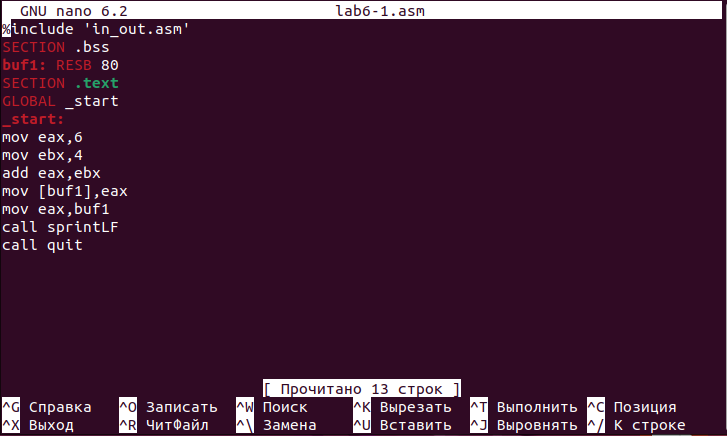


Рис. 3: Рисунок3

Создайте исполняемый файл и запустите его

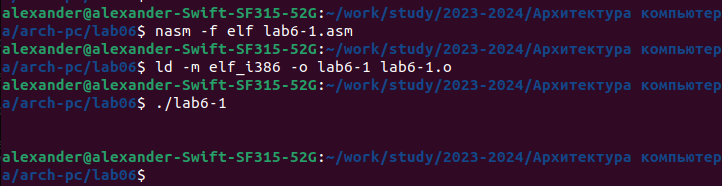


Рис. 4: Рисунок4

Да, отображается. Это символ перевода строки (line feed)

1. Как отмечалось выше, для работы с числами в файле in\_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразуем текст программы из Листинга 6.1 с использованием этих функций.

Создайте файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введите в него текст программы из листинга 6.2.



Рис. 5: Рисунок5

Создайте и запустите исполняемый файл.

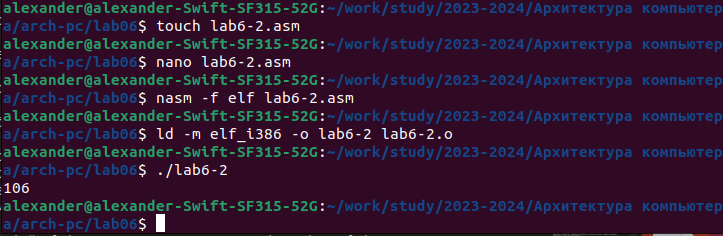


Рис. 6: Рисунок6

1. Аналогично предыдущему примеру изменим символы на числа. Замените строки



Рис. 7: Рисунок7

Создайте исполняемый файл и запустите его

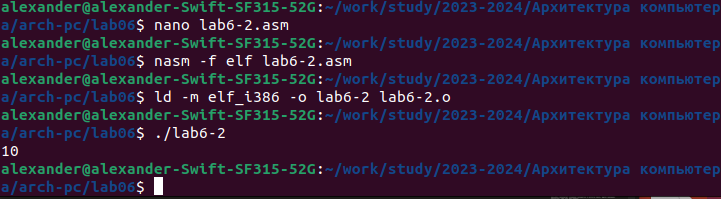


Рис. 8: Рисунок8

Замените функцию iprintLF на iprint. Создайте исполняемый файл и запустите его. Чем отличается вывод функций iprintLF и iprint?

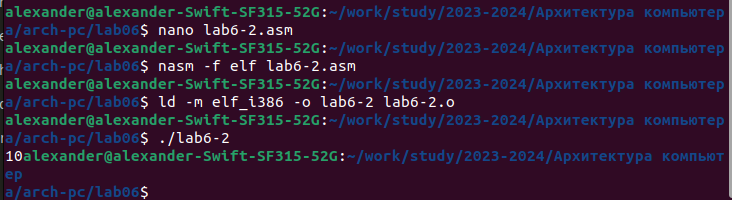


Рис. 9: Рисунок9

iprint не добавляет символ перевода строки.

## 4.2 Выполнение арифметических операций в NASM

1. В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения f(x) = (5 \* 2 + 3)/3.

Создайте файл lab6-3.asm в каталоге ~/work/arch-pc/lab06:

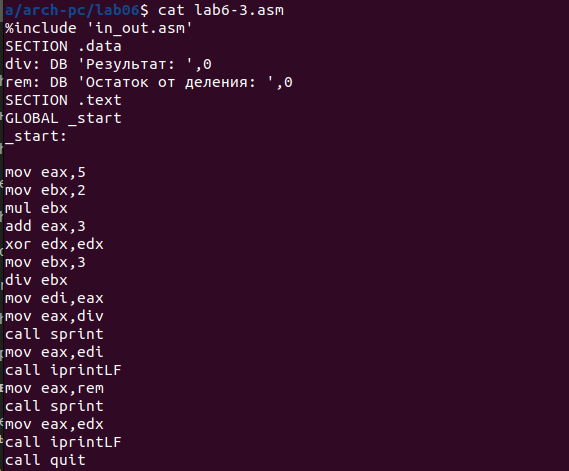


Рис. 10: Рисунок10

Создайте исполняемый файл и запустите его

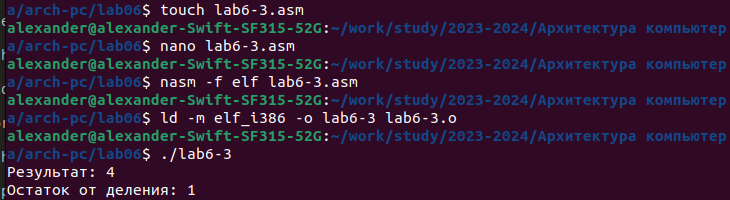


Рис. 11: Рисунок11

Измените текст программы для вычисления выражения f(x) = (4 \* 6 + 2)/5. Создайте исполняемый файл и проверьте его работу.

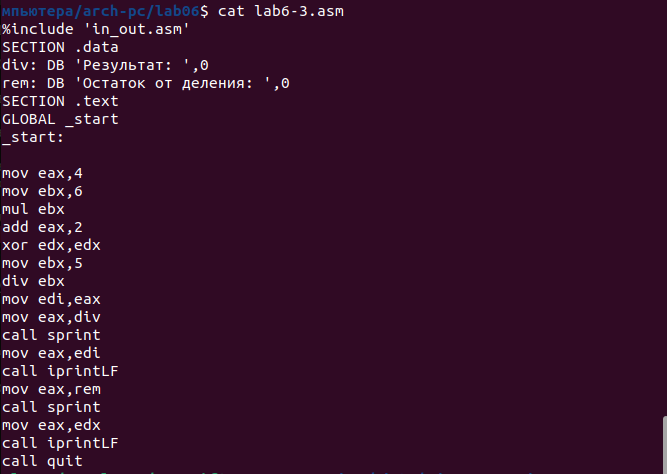


Рис. 12: Рисунок12

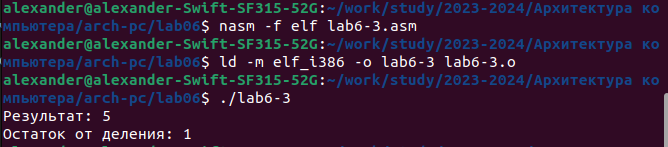


Рис. 13: Рисунок13

1. В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму: • вывести запрос на введение № студенческого билета • вычислить номер варианта по формуле: (S\_n mod 20) + 1, где S\_n – номер студенческого билета (В данном случае a mod b – это остаток от деления a на b). • вывести на экран номер варианта.

В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры. Как отмечалось выше ввод с клавиатуры осуществляется в символьном виде и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа. Для этого может быть использована функция atoi из файла in\_out.asm. Создайте файл variant.asm в каталоге ~/work/arch-pc/lab06:

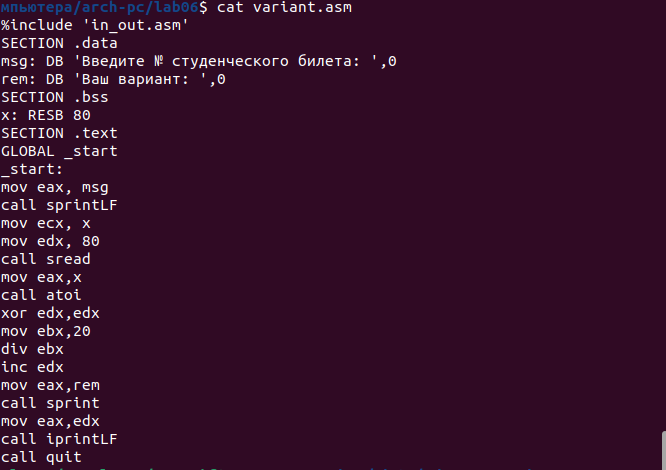


Рис. 14: Рисунок14

Создайте и запустите исполняемый файл

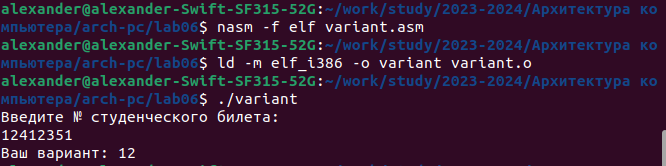


Рис. 15: Рисунок15

## 4.3 Ответы на вопросы

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?

mov eax,rem  
call sprint

1. Для чего используется следующие инструкции?

mov ecx, x  
mov edx, 80  
call sread

Для ввода строки с клавиатуры

1. Для чего используется инструкция “call atoi”?

Для вызова подпрограммы преобразования ASCII кода в число

1. Какие строки листинга 6.4 отвечают за вычисления варианта?

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

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

edx

1. Для чего используется инструкция “inc edx”?

Для увеличения edx на 1

1. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?

mov eax, edx  
call iprintLF

# 5 Задание для самостоятельной работы

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

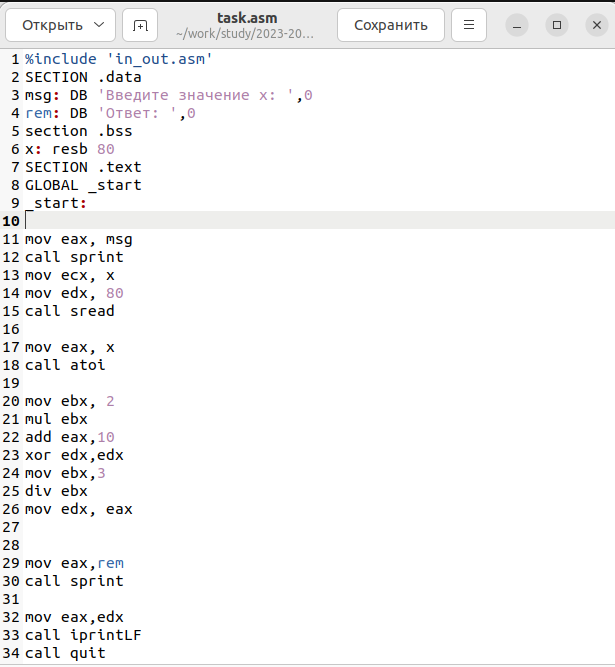


Рис. 16: Рисунок16

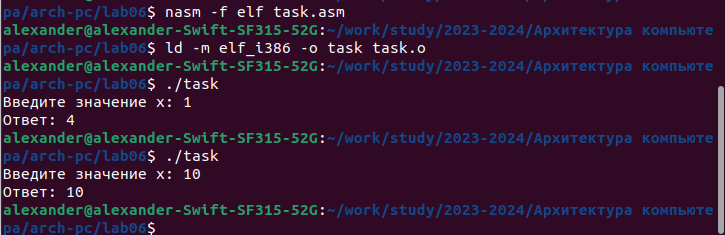


Рис. 17: Рисунок17

Я взял пример под номером 1

# 6 Выводы

Мы научились выделять память под переменные, также проводить арифметические операции с ними и в конце написали свою программу, считающую функцию от значения переменной х.

# Список литературы