Отчет по лабораторной работе номер 6

Архитектура компьютеров и операционные системы.

Ващаев Виктор Андреевич

Содержание

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

# 2 Задание

Выполнить лабораторную работу и получить максимально баллов

# 3 Теоретическое введение

Дрессация в NASM связана с указанием места хранения данных для их обработки в инструкциях. Операнды могут храниться либо в регистрах, либо в ячейках памяти. Способы задания адреса хранения операндов называются методами адресации.

Существует три основных метода адресации:

Регистровая адресация – операнды располагаются в регистрах, и инструкции содержат имена этих регистров. Пример:

mov ax, bx

Непосредственная адресация – значение операнда указывается прямо в инструкции. Пример:

mov ax, 2

Адресация памяти – операнд указывает адрес в памяти, где хранятся данные. В инструкции используется символическое имя ячейки памяти, с содержимым которой нужно работать.

Например, если определена переменная:

intg DD 3 ; Объявление области памяти размером 4 байта с меткой intg

то команда:

mov eax, [intg]

копирует данные из памяти по адресу intg в регистр eax.

Аналогично, команда:

mov [intg], eax

записывает данные из регистра eax в память по адресу intg.

Рассмотрим также команду:

mov eax, intg

В этом случае в регистр eax будет записан адрес метки intg. Если, например, для intg выделена память начиная с адреса 0x600144, то команда:

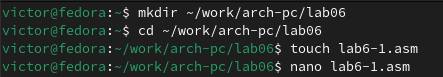
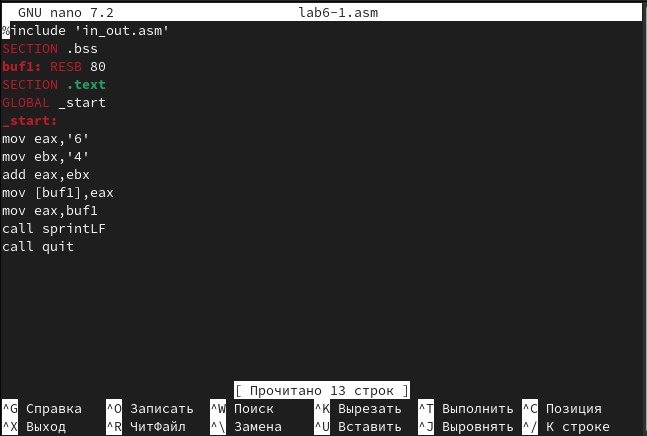
mov eax, intg

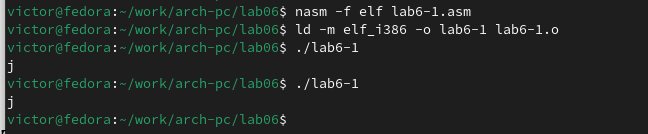
эквивалентна:

mov eax, 0x600144

Это означает, что в регистр eax записывается адрес 0x600144.

# 4 Выполнение лабораторной работы

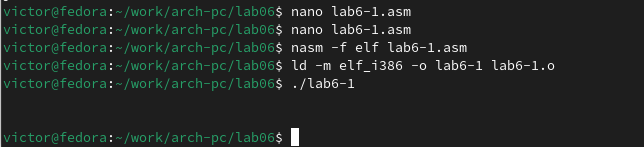
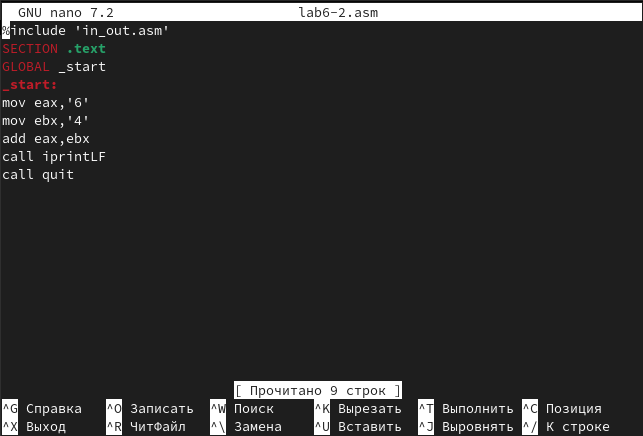
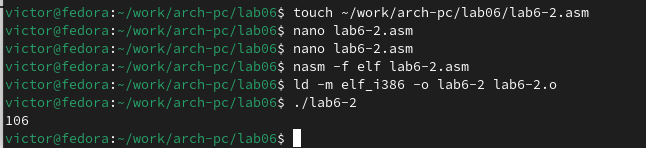
1. Создаём каталог для программам лабораторной работы № 6, переходим в него и создаём файл lab6-1.asm 
2. Я ввожу в файл lab6-1.asm текст программы из листинга 6.1. В этой программе я записываю символ ‘6’ в регистр eax с помощью команды mov eax, ‘6’, а символ ‘4’ — в регистр ebx с помощью команды mov ebx, ‘4’. Затем я прибавляю значение из регистра ebx к значению в eax (add eax, ebx), и результат сложения сохраняется в eax. Далее я вывожу результат. Поскольку функция sprintLF требует, чтобы в eax был записан адрес, мне нужно использовать дополнительную переменную. Для этого я записываю значение из регистра eax в переменную buf1 (mov [buf1], eax), затем записываю адрес переменной buf1 в eax (mov eax, buf1) и вызываю функцию sprintLF. 

 3. Теперь я изменяю текст программы, чтобы вместо символов в регистры записывались числа. Я исправляю текст программы (Листинг 6.1) следующим образом: заменяю строки

mov eax, ‘6’ mov ebx, ‘4’

на строки

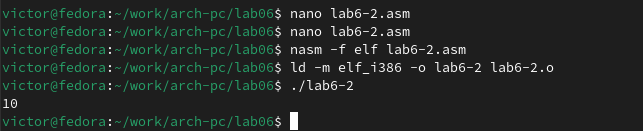
mov eax, 6 mov ebx, 4

  Затем я создаю исполняемый файл и запускаю его. Как и в предыдущем случае, при выполнении программы я не получаю число 10. Вместо этого выводится символ с кодом 10, что соответствует символу перевода строки в таблице ASCII. Определение символа для кода 10 по таблице ASCII: Код 10 в таблице ASCII соответствует символу перевода строки (LF — Line Feed). Этот символ не отображается как видимый знак на экране, а используется для перехода на новую строку. 4. Я создаю файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и ввожу в него текст программы из Листинга 6.2, с использованием подпроцедур из файла in\_out.asm для преобразования ASCII символов в числа и обратно. Эти функции помогут корректно работать с числами и выводить их в требуемом формате.   5. Теперь я заменяю строки:

mov eax, ‘6’ mov ebx, ‘4’

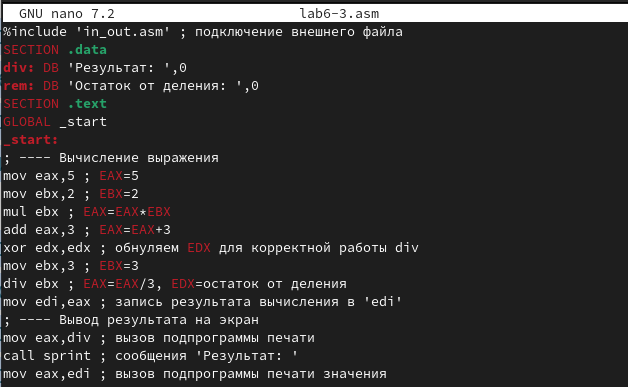
на строки:

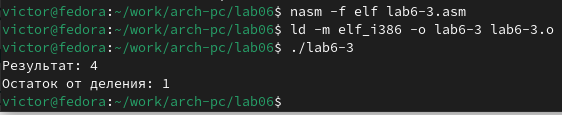
mov eax, 6 mov ebx, 4

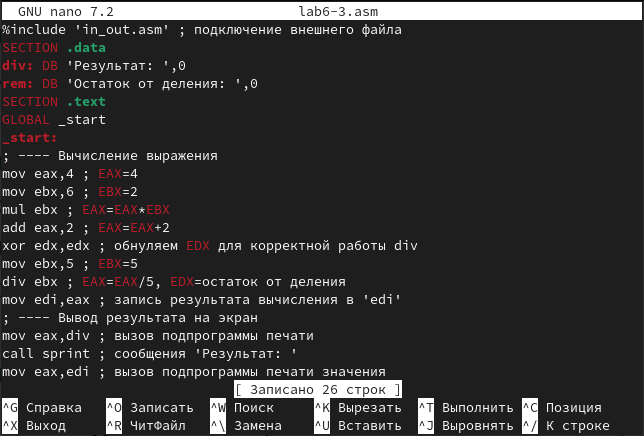
Затем я создаю исполняемый файл и запускаю его. При выполнении программы результатом будет отображение числа 10, поскольку выполняется сложение чисел.  Вывод исполняемого файла: На изображении видно, что программа выводит результат 10. Это, скорее всего, результат вычислений, выполненных в моей программе.

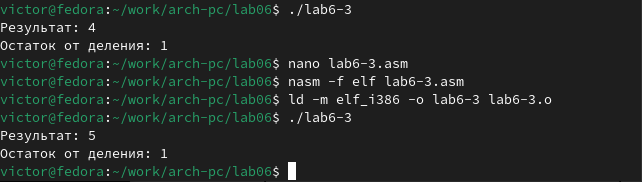
Различие между функциями iprintLF и iprint:

iprintLF — выводит значение и автоматически добавляет символ перевода строки (код 10), что приводит к переходу на новую строку после вывода числа.  
iprint — выводит только значение без добавления символа перевода строки, поэтому следующая команда будет выводиться на той же строке.

1. Я создаю файл lab6-3.asm в каталоге ~/work/arch-pc/lab06 и пишу в него программу для вычисления выражения 

После этого создаю исполняемый файл, компилирую и запускаю его. 

Далее изменяю текст программы для вычисления выражения f(x) = (4 \* 6 + 2) / 5: 

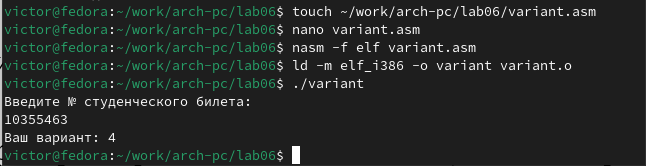
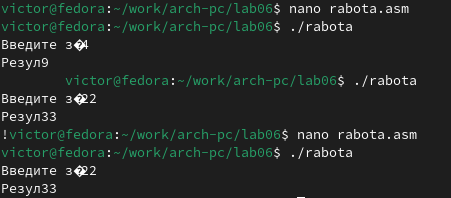
Создаю исполняемый файл и проверяю его работу. 

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

Вначале вывожу запрос на ввод номера студенческого билета. После этого с помощью функции sread программа считывает введённое значение с клавиатуры и сохраняет его в виде строки в переменную.

Затем, чтобы преобразовать введённые символы в число, использую функцию atoi из файла in\_out.asm. Эта функция преобразует строку с номером студенческого билета в целое число и сохраняет его в регистре EAX.

Далее вычисляю номер варианта по формуле (Sn mod 20) + 1, где Sn — это номер студенческого билета. Для получения остатка от деления использую команду div, где EAX делится на 20, а остаток сохраняется в EDX. Увеличиваю остаток на 1, чтобы получить номер варианта.

После этого вывожу на экран сообщение с результатом, используя функции sprint для текста и iprintLF для вывода самого номера варианта с переводом строки.  8. Самостоятельная работа! Я напишу программу для вычисления выражения y=f(x)y=f(x). Программа будет выводить выражение для вычисления, запрашивать ввод значения xx, вычислять заданное выражение в зависимости от введенного xx, а затем выводить результат вычислений. Я выберу вид функции f(x)f(x) из таблицы 6.3, в зависимости от номера, который я получу при выполнении лабораторной работы. Я создам исполняемый файл и проверю его работу для значений x1x1​ и x2x2​ из 6.3. 

# 5 Выводы

Заключение:

В ходе выполнения лабораторной работы были рассмотрены различные аспекты программирования на ассемблере, включая работу с регистрами, арифметические операции, вывод данных и использование библиотечных функций. Все задачи были успешно выполнены, что позволяет углубленно понять основы работы с ассемблером, а также взаимодействие с системными функциями для выполнения базовых операций.

# Список литературы

None