Отчёт по лабораторной работе №6

Дисциплина: Архитектура компьютеров

Карпова Анастасия Александровна

Содержание

[1 Цель работы 1](#_Toc151231590)

[2 Задание 1](#_Toc151231591)

[3 Теоретическое введение 1](#_Toc151231592)

[4 Выполнение лабораторной работы 2](#_Toc151231593)

[5 Выводы 8](#_Toc151231594)

[Список литературы 8](#_Toc151231595)

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

# 2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических действий в NASM
3. Выполнение заданий для самостоятельной работы

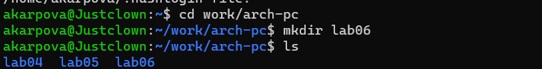
# 3 Теоретическое введение

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. 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. 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. 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. 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

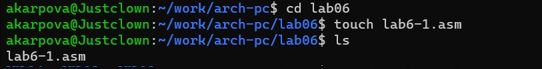
4.1 Символьные и численные данные в NASM

C помощью mkdir создаю каталог lab06 в ~/work/arch-pc. Перехожу в созданный каталог с помощью утилиты cd (рис. 1).



Создание каталога lab06

С помощью утилиты touch создаю файл lab7-1.asm (рис. 2).



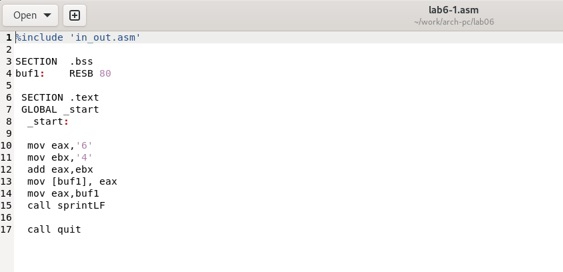
Создание файла lab7-1.asm

Копирую в текущий каталог файл in\_out.asm с помощью утилиты cp (рис. 3).

Копирование файла

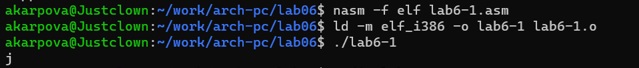
Копирование файла

Открываю созданный файл lab7-1.asm, вставляю в него программу вывода значения регистра eax (рис. 4).



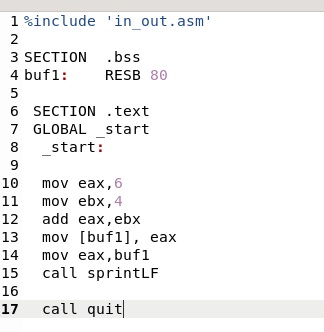
Ввод программы значения регистра eax

Создаю исполняемый файл программы и запускаю его. Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6. (рис. 5)



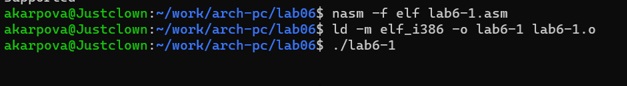
Создание исполняемого файла и его запуск

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 6)



Изменение строки

Создаю новый исполняемый файл и запускаю его. Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран (рис. 7)



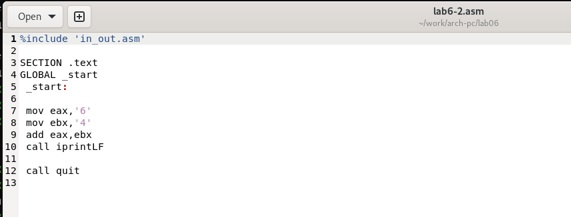
Создание нового исполняемого файла и его запуск

C помощью touch cоздаю новый файл lab6-2.asm (рис. 8)

Создание исполняемого файла

Создание исполняемого файла

Ввожу в файл текст другой программы для вывода значения регистра eax (рис. 9).



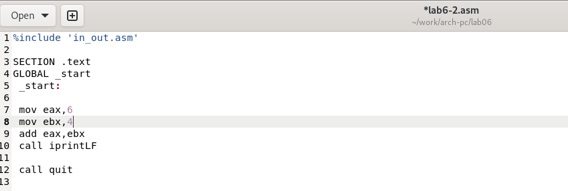
Ввод программы значения регистра eax

Создаю и запускаю исполняемый файл lab6-2. Теперь выводится число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4” (рис. 10)

Создание и запуск исполняемого файла

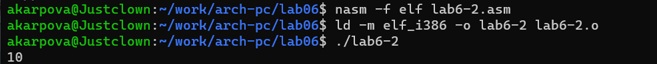
Создание и запуск исполняемого файла

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 11)



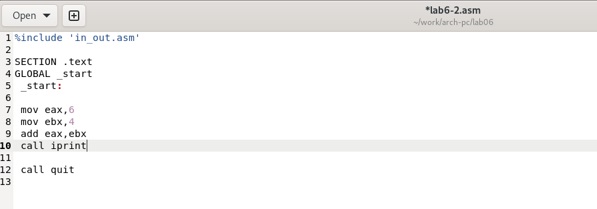
Замена

Создаю и запускаю новый исполняемый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10 (рис. 12)



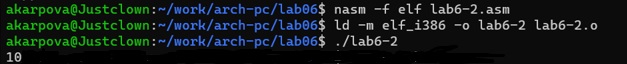
Создание и запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. 13)



Замена

Создаю и запускаю новый исполняемый файл. Вывод не изменяется, потому что символ переноса строки не отображался, когда программа исполнялась с функцией iprintLF, а iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF (рис. 14)



Создание и запуск исполняемого файла

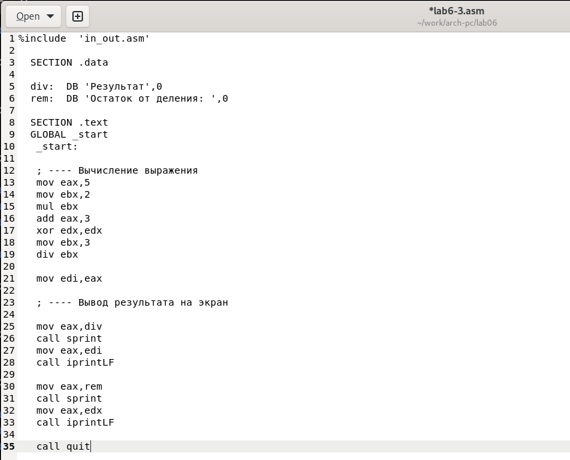
4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью touch (рис. 15)

Создание файла

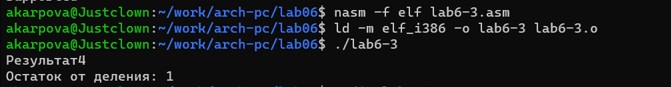
Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения f(x) = (5 \* 2 + 3)/3 (рис. 16)



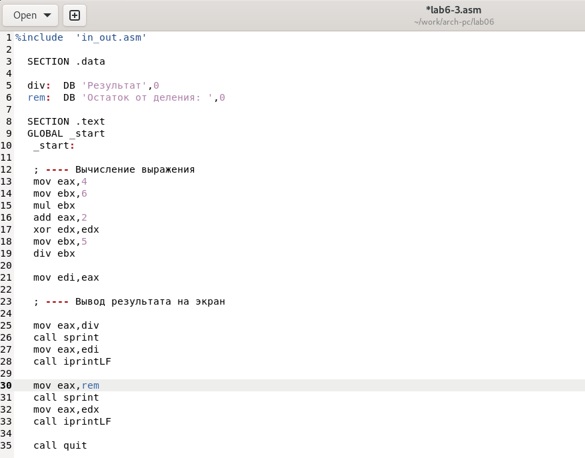
Ввод программы

Создаю исполняемый файл и запускаю его (рис. 17)



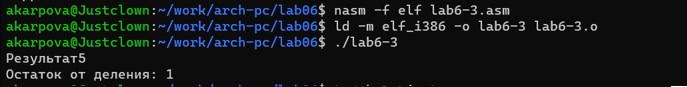
Создание файла

Изменяю программу так, чтобы она вычисляла значение выражения f(x) = (4 \* 6 + 2)/5 (рис. 18)



Изменение программы

Создаю и запускаю новый исполняемый файл (рис. 19)



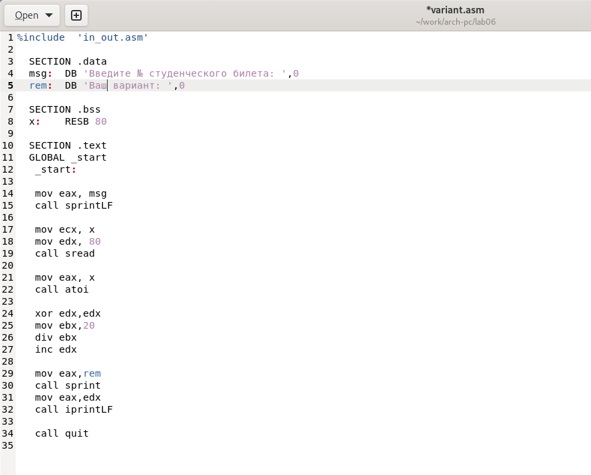
Создание и запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch (рис. 20)

Создание файла

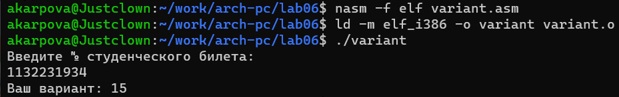
Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 21).



Ввод программы

Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета программа вывела, что мой вариант - 15 (рис. 22)



Создание и запуск исполняемого файла

4.2.1 Ответы на вопросы по программе

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

mov eax,rem call sprint

1. Инструкция mov ecx, x используется, чтобы положить адрес вводимой строки x в регистр ecx mov edx, 80 - запись в регистр edx длины вводимой строки call sread - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры
2. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax
3. За вычисления варианта отвечают строки:

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

1. При выполнении инструкции div ebx остаток от деления записывается в регистр edx
2. Инструкция inc edx увеличивает значение регистра edx на 1
3. За вывод на экран результатов вычислений отвечают след. строки:

mov eax,edx call iprintLF

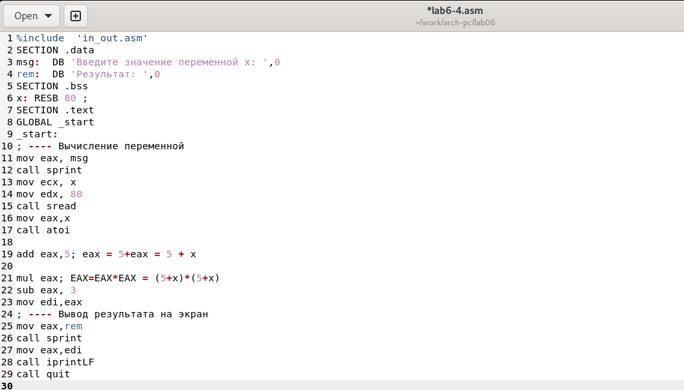
4.3 Выполнение заданий для самостоятельной работы

Создаю файл lab6-4.asm с помощью touch (рис. 23)

Создание файла

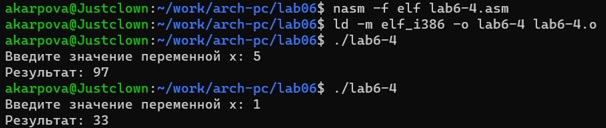
Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения (5+x)^2-3 (рис. 24)



Редактирование

Создаю и запускаю исполняемый файл (рис. 25). При вводе значения 5, вывод - 97; 1 - 33.



Создание и запуск исполняемого файла

# 5 Выводы

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

# Список литературы

1. Архитектура ЭВМ