Лабораторная работа №6

Дисциплина: Архитектура компьютера

Карпачев Ярослав Олегович

Содержание

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

# 2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций 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.

* 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. 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.

# 4 Выполнение лабораторной работы

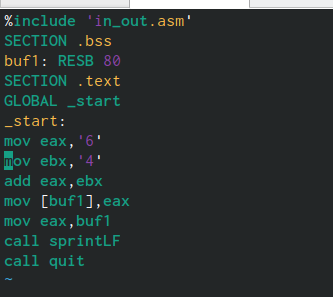
## 4.1 Символьные и численные данные в NASM

С помощью утилиты mkdir создал каталог для программ лабораторной работы № 6, перешлел в него и создал файл lab6-1.asm Скопировал в текущий каталог файл in\_out.asm с помощью утилиты cp, т.к. он будет использоваться. (рис. ??).

Создание файла lab6-1.asm

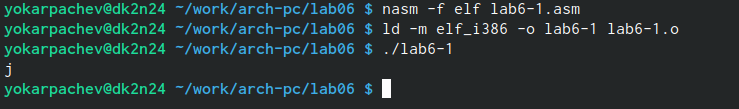
Создание файла lab6-1.asm

Ввел в файл lab6-1.asm текст программы из листинга 6.1 (рис. ??).



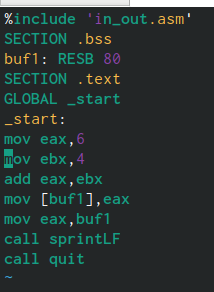
Редактированный файл

Создал исполняемый файл, скомпилировал и запустил его (рис. ??). Вывод - символ j, потому что программа вывела символ по системе ASCII сумме двоичных кодов символов 4 и 6.

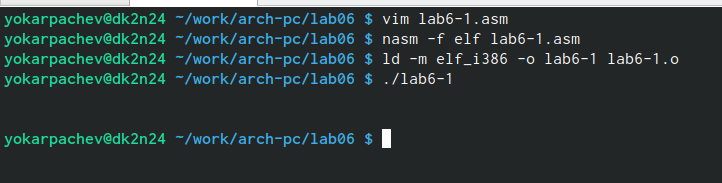


Создание исполняемого кода и его запуск

Исправил текст программы: поменяла символы “6” и “4” на цифры 6 и 4 (рис. ??). Скомпилировал файл и запустил его (рис. ??). Теперь вывелся символ с кодом 10, это символ перевода строки, поэтому он не отображается при выводе на экран.



Редактированнный файла

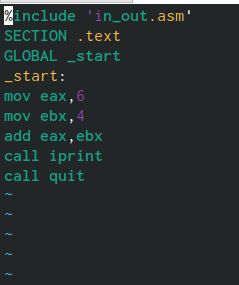


Создание исполняемого файла и его запуск

Создал lab6-2.asm в каталоге ~/work/arch-pc/lab06 и ввел текст программы из листинга 6.2 (рис. ??) и (рис. ??).

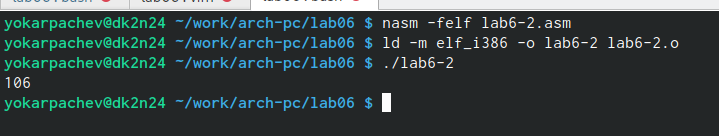
Создание файла lab6-2.asm

Создание файла lab6-2.asm



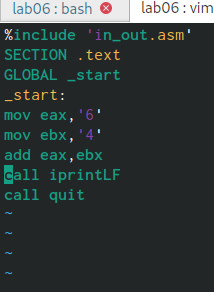
Редактированнный файла

Скомпилировал файл и запустил его (рис. ??). Теперь вывело число 106, потому что программа позволяет вывести число, а не символ, хотя происходит сложение кодов символов “6” и “4”.



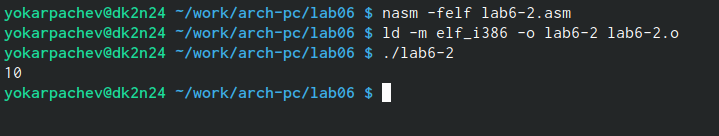
Компиляция файла и его запуск

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



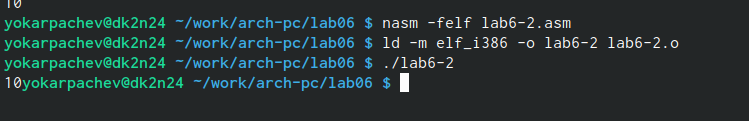
Редактирование программы

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



запуск файла

Заменяю в тексте программы iprintLF на iprint (рис. ??). Создал и скомпилировал файл (рис. ??). Вывод не изменился - 10, но теперь iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF.



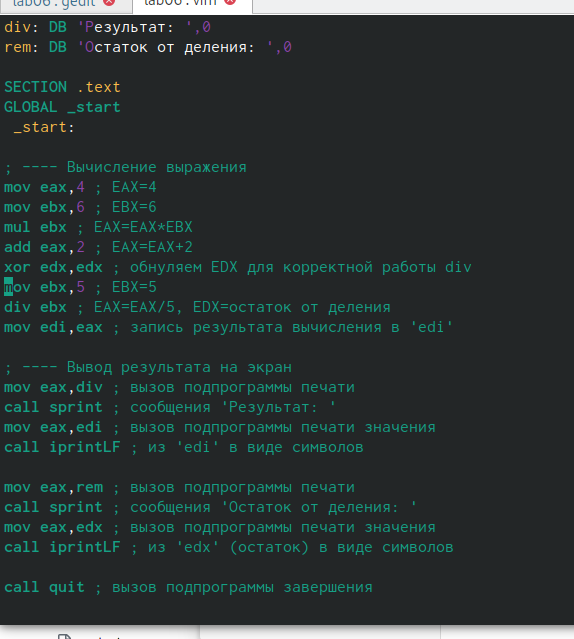
запуск нового варианта

## 4.2 Выполнение арифметических операций в NASM

Создал файл lab6-3.asm с помощью утилиты touch (рис. ??). Ввел в созданный файл текст программы для вычисления значения выражения f(x) = (5 \* 2 + 3)/3 (рис. ??) и (рис. ??).

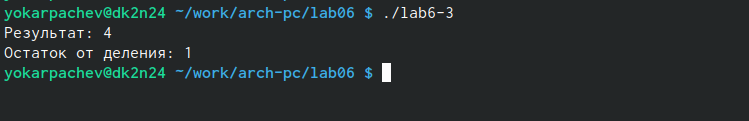
Создание файла

Создание файла



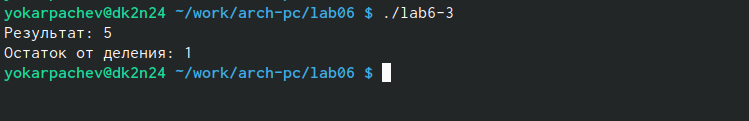
Редаактирование файла

Скомпилировал файл и запустил его (рис. [??]).



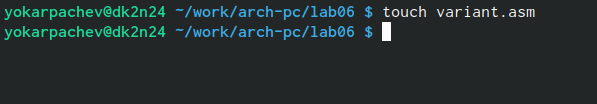
Запуск исполняемого файла

Изменил программу так, чтобы она вычисляла значение выражения f(x) = (4 \* 6 + 2)/5. Создал и запустил новый исполняемый файл (рис. ??). Программа отработала верно.

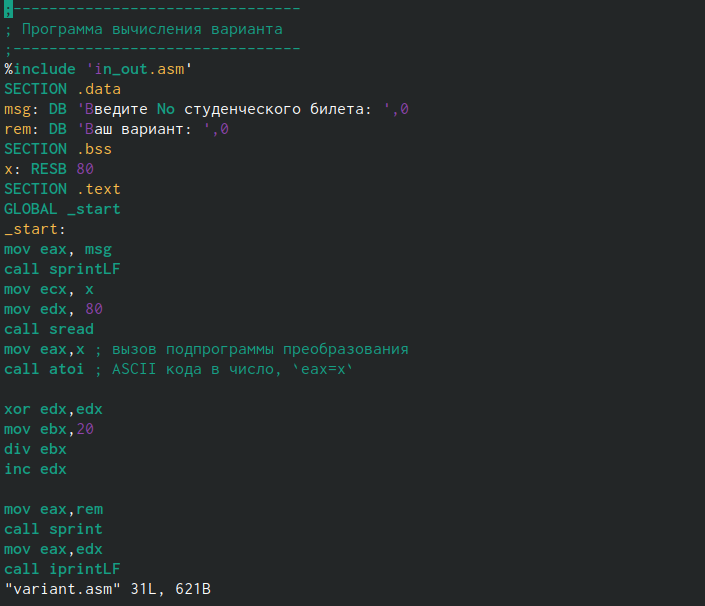


Запуск нового исполняемого файла

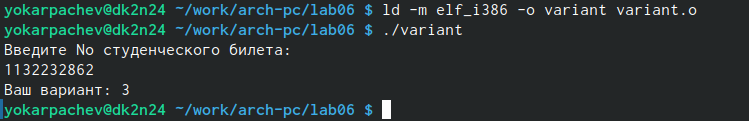
Создал файл variant.asm с помощью touch (рис. ??). Ввел код из листинга 6.4 (рис. ??). Создала исполняемый файл и запустила его. Ввела номер своего студенческого билета и получила номер своего варианта – 11 (рис. ??). Проверила правильность выполнения, вычислив номер варианта аналитически.



Создание файла



Редактирование файла



Запуск 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:

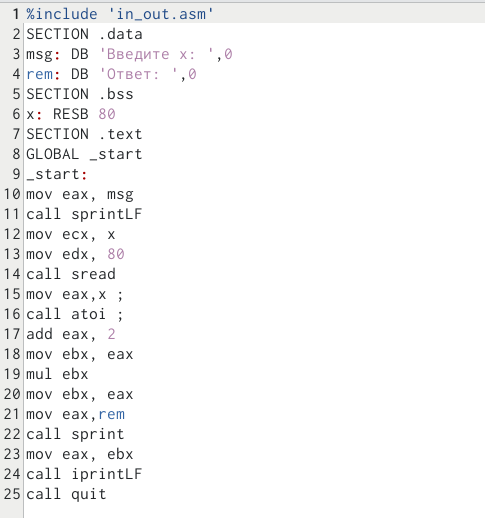
xor edx,edx ; обнуление edx для корректной работы div  
mov ebx,20 ; ebx = 20  
div ebx ; eax = eax/20, edx - остаток от деления  
inc edx ; edx = edx + 1

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 строки:

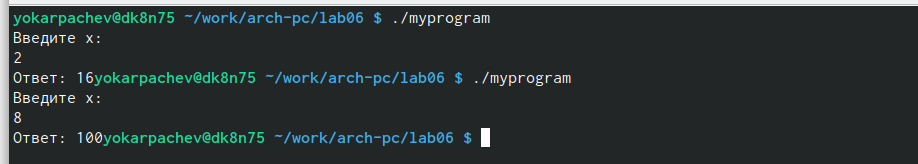
mov eax,edx  
call iprintLF

## 4.3 Выполнение заданий для самостоятельной работы

Создал myprogram.asm. Открыл его, ввела код программы для вычисления значения выражения (x + 2)^2 (вариант 3) (рис. ??). Создал и запустил исполняемый файл. При вводе значения 1 на входе вывод программы = 10. При вводе значения 7 на входе вывод программы = 70. Значит, программа работает верно. (рис. ??).



Код программы



Запуск программы для выполнения задания для самостоятельной работы

# 5 Выводы

При выполнении данной лабораторной работы я освоил арифметические инструкции языка ассемблера NASM.