Отчет по лабораторной работе №6

Архитектура компьютеров

Арутюнян Эрик Левонович

Содержание

# 1 Цель работы

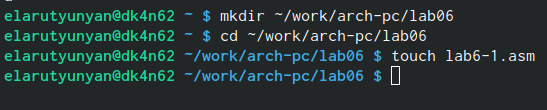
Освоение арифметических инструкций языка ассемблера NASM

# 2 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес опе- ранда 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 ко- манде, Например: mov ax,2. • Адресация памяти – операнд задает адрес в памяти. В команде указывается символи- ческое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию. Например, определим переменную intg DD 3 – это означает, что задается область памяти размером 4 байта, адрес которой обозначен меткой intg. В таком случае, команда mov eax,[intg] копирует из памяти по адресу intg данные в регистр eax. В свою очередь команда mov [intg],eax запишет в память по адресу intg данные из регистра eax. Также рассмотрим команду mov eax,intg В этом случае в регистр eax запишется адрес intg. Допустим, для intg выделена память начиная с ячейки с адресом 0x600144, тогда команда mov eax,intg аналогична команде mov eax,0x600144 – т.е. эта команда запишет в регистр eax число 0x600144.

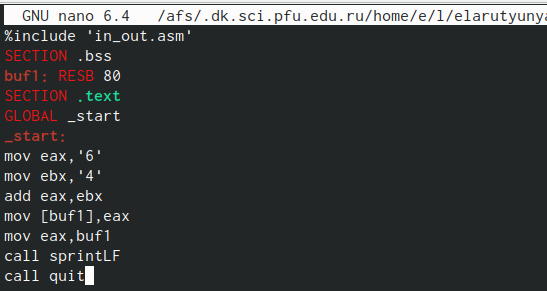
# 3 Выполнение лабораторной работы

Создал каталог для программ лабораторной работы № 6, перешел в него и создал файл lab-1.asm (рис. ??).



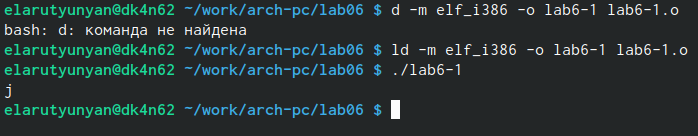
Создание файлов

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



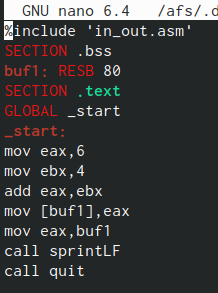
Ввод текста

Создал исполняемый файл и запустил его (рис. ??).



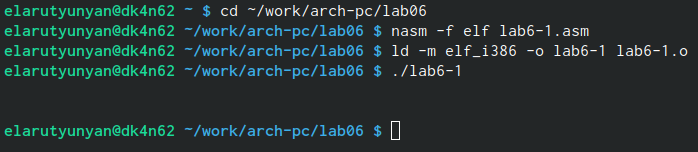
Запуск файла

Изменил исполняемый файл заменив строки (рис. ??).



Замена строк в файле

Запустил файл в котором заменил строки (рис. ??).



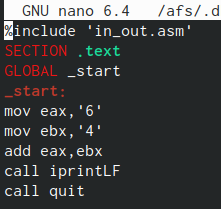
Запуск файла

Создайте файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 (рис. ??).

Создание файла

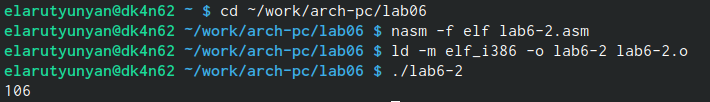
Создание файла

Ввел в созданный файл текст (рис. ??).



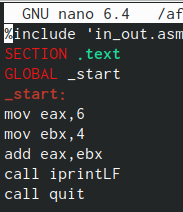
Ввод текста

Далее запустил программу и получил результат 106 (рис. ??).



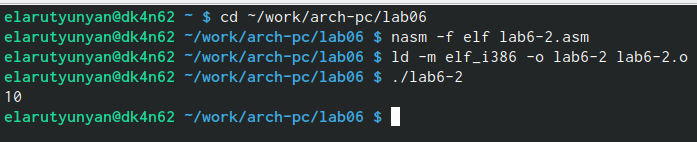
Запуск и результат

Аналогично предыдущему примеру изменил символы на числа. Потом заменил строки на другие (рис. ??).



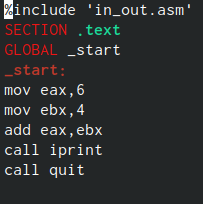
Замена строк

Далее запустил программу и получил результат 10 (рис. ??).



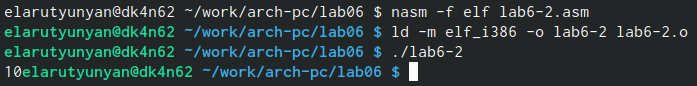
Запуск и результат

Далее заменил в том же файле функцию iprintLF на iprint(рис. ??).



Замена функции

Запустил измененную программу и получил результат 10 в той же строке и поняд чем отличается предыдущая программа от этой (рис. ??).



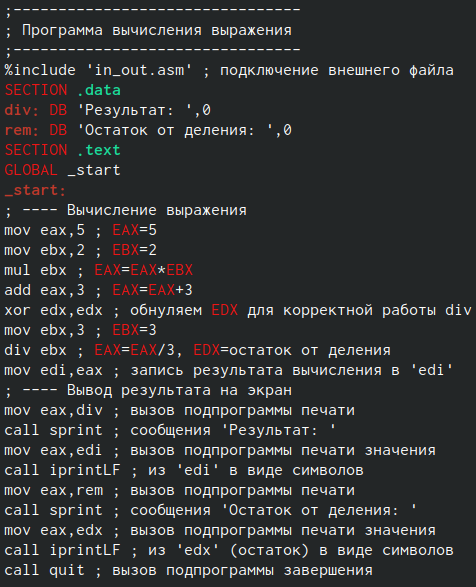
Запуск новой программы

Создайте файл lab6-3.asm в каталоге ~/work/arch-pc/lab06 (рис. ??).

Создание файла

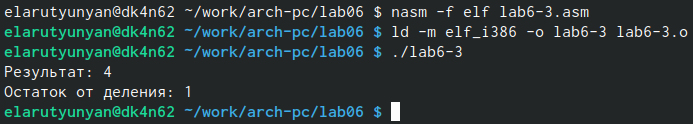
Создание файла

Далее ввел в него код для вычисления выражения (рис. ??).



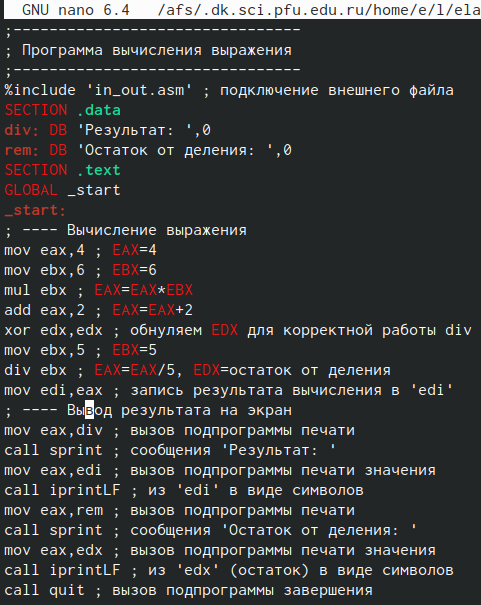
Ввод кода

После этого я запустил этот файл и получил результат (рис. ??).



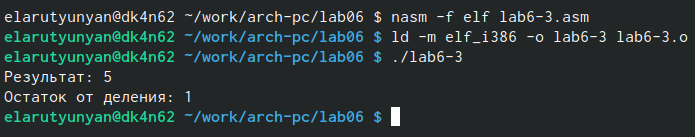
Запуск файла и получение результата

Измените текст программы для вычисления выражения 𝑓(𝑥) = (4 ∗ 6 + 2)/5. (рис. ??).



Изменение текста программы

Далее запустил измененный текст и получил желаемый результат (рис. ??).



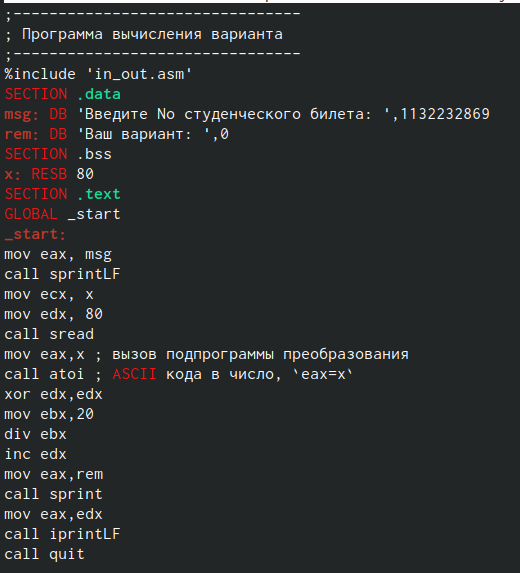
Запуск измененного файла

Создал файл variant.asm в каталоге ~/work/arch-pc/lab06 (рис. ??).

Создание файла variant.asm

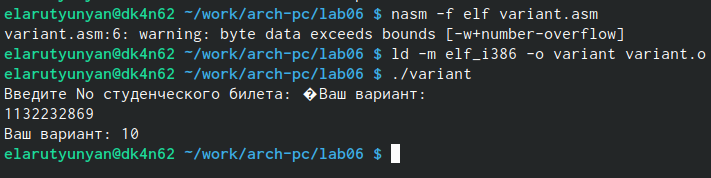
Создание файла variant.asm

Внимательно изучил текст программы из листинга 6.4 и ввел в файл variant.asm. (рис. ??).



Ввод текста в файл variant.asm

Запустил файл и узнал номер своего варианта (рис. ??).



Вариант 10

#Ответы на вопросы по программе

1. За вывод сообщения “Ваш вариант” отвечают строки кода: mov eax,rem call sprint
2. Инструкция mov ecx, x используется, чтобы положить адрес вводимой стро- ки 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
3. call atoi используется для вызова подпрограммы из внешнего файла, кото- рая преобразует ascii-код символа в целое число и записывает результат в регистр eax
4. 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
5. При выполнении инструкции div ebx остаток от деления записывается в регистр edx
6. Инструкция inc edx увеличивает значение регистра edx на 1
7. За вывод на экран результатов вычислений отвечают строки: mov eax,edx call iprintLF #Самостоятельная работа

# 4 Вывод

Освоил арифметические инструкции языка NASM

# Список литературы