Лабораторная работа №7

Команды безусловного и условного переходов в Nasm. Программирование ветвлений.

Перфилов Александр Константинович | группа: НПИбд 02-23

Содержание

# 1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

# 2 Выполнение лабораторной работы

**Реализация переходов в NASM**

Создадим каталог для программам лабораторной работы № 7, перейдите в него и создадим файл lab7-1.asm

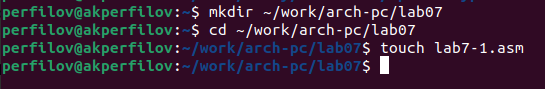


Рис 2.1.1: Создание каталога и файла .asm

Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции jmp. Введем в файл lab7-1.asm текст программы из листинга 7.1

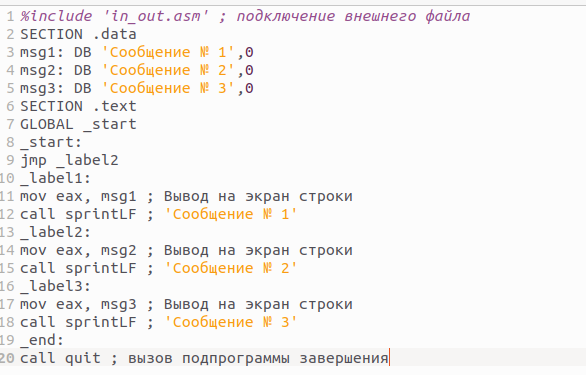


Рис 2.1.2: Программа

Создадим исполняемый файл и запустим его. Результат работы данной программы будет следующим:

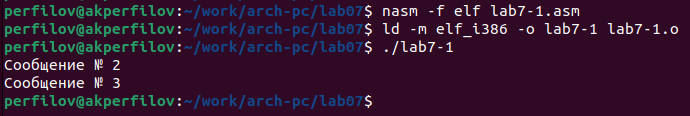


Рис 2.1.3: Создание исполняемого файла и проверка работы

Таким образом, использование инструкции jmp \_label2 меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки \_label2, пропустив вывод первого сообщения.

Инструкция jmp позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию jmp с меткой \_label1 (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию jmp с меткой \_end (т.е. переход к инструкции call quit). Измените текст программы в соответствии с листингом 7.2

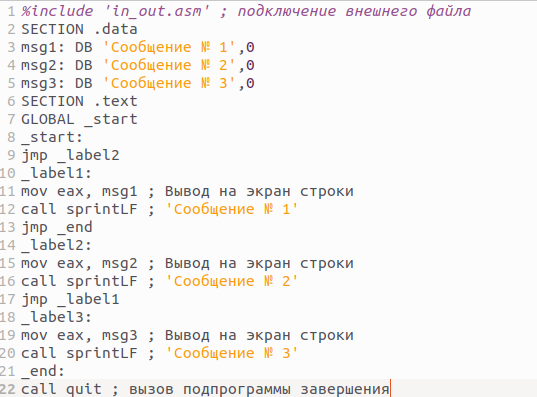


Рис 2.1.4: Измененная программа

Создадим исполняемый файл и проверим его работу.

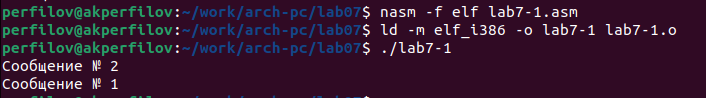


Рис 2.1.5: Создание исполняемого файла и проверка работы

Изменим текст программы добавив или изменив инструкции jmp.

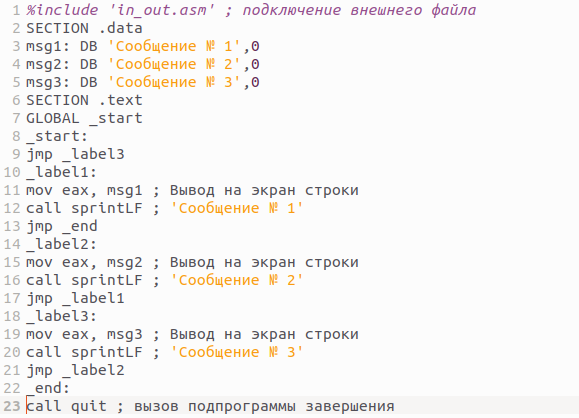


Рис 2.1.6: Демонстрация измененного текста программы

Проверим работу программы

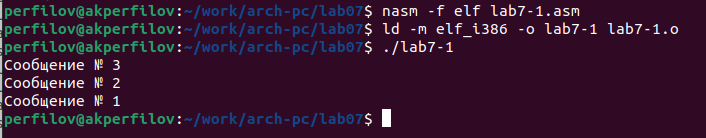


Рис 2.1.7: Проверка работы программы

Использование инструкции jmp приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A,B и C. Значения для A и C задаются в программе, значение B вводиться с клавиатуры

Создадим файл lab7-2.asm в каталоге ~/work/arch-pc/lab07.

Рис 2.1.8: Создание файла .asm

Рис 2.1.8: Создание файла .asm

Внимательно изучим текст программы из листинга 7.3 и введем в lab7-2.asm.

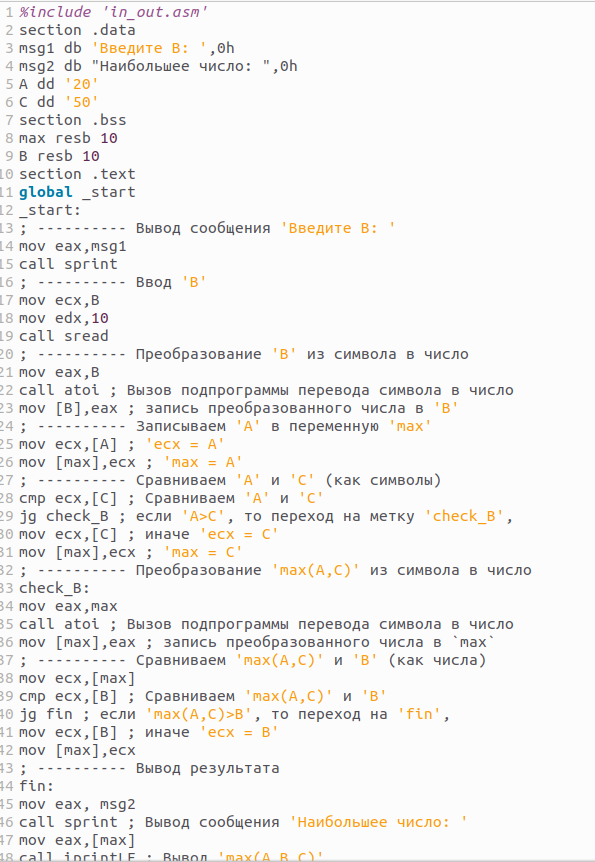


Рис 2.1.9: Программа

Создадим исполняемый файл и проверим его работу для разных значений B.

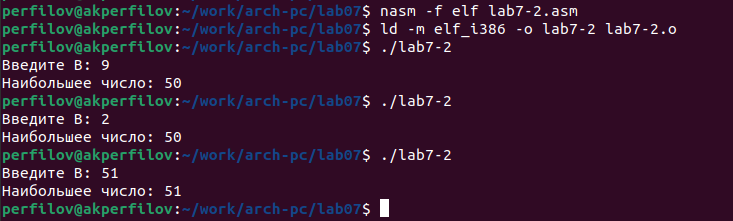


Рис 2.1.10: Создание исполняемого файла и проверка работы с разными значениями B

Обратим внимание, в данном примере переменные A и С сравниваются как символы, а переменная B и максимум из A и С как числа (для этого используется функция atoi преобразования символа в число). Это сделано для демонстрации того, как сравниваются данные. Данную программу можно упростить и сравнивать все 3 переменные как символы (т.е. не использовать функцию atoi). Однако если переменные преобразовать из символов числа, над ними можно корректно проводить арифметические операции.

**Изучение структуры файлы листинга**

Обычно nasm создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ -l и задав имя файла листинга в командной строке. Создадим файл листинга для программы из файла lab7-2.asm

Рис 2.2.1: Создание файла

Рис 2.2.1: Создание файла

Откроем файл листинга lab7-2.lst с помощью любого текстового редактора, например mcedit

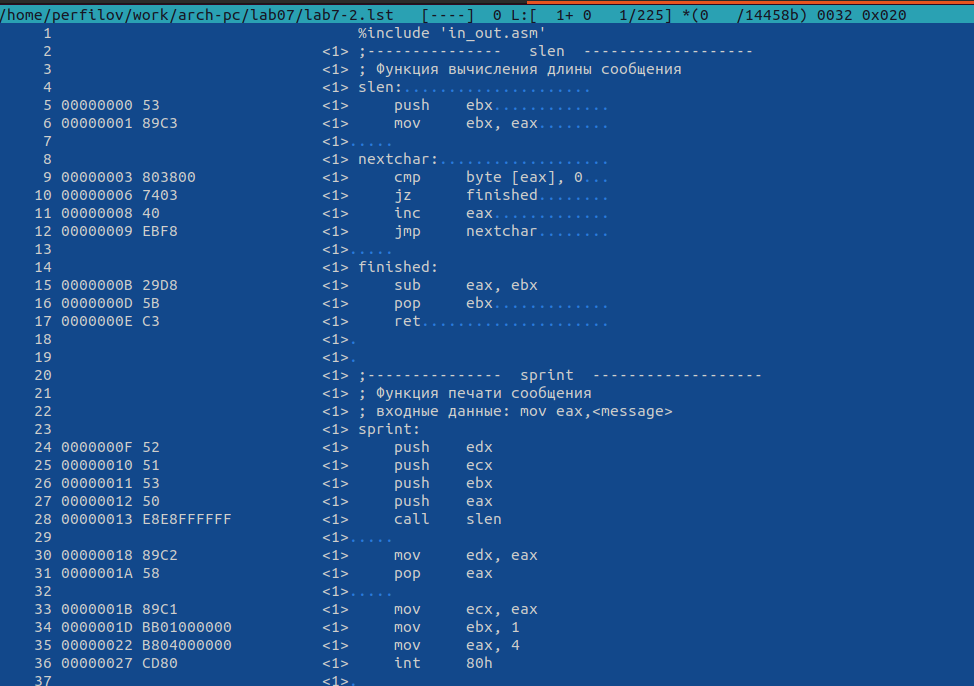


Рис 2.2.2: Вид файла .lst

Внимательно ознакомимся с его форматом и содержимым.

Возьмем 3, 4 и 5 строчки кода

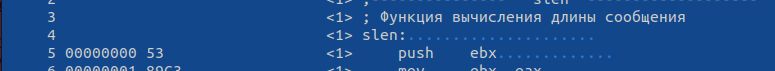


Рис 2.2.3: Взятые строки

3 - номер строки кода, “; Функция вычисления длинны сообщения” - оно не имеет отношения к работе кода. 4 - номер строки кода, “slen:…….” - название функции. 5 - номер строки кода, “00000000” - адрес строки, “53” - машинный код, “push ebx” - исходный текст программы, инструкция “push” помещает операнд “ebx” в стек.

Откроем файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалим один операнд. Выполним трансляцию с получением файла листинга:

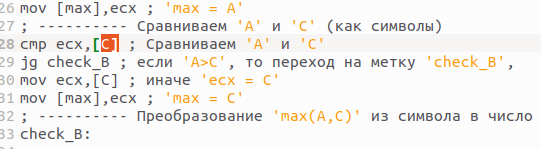


Рис 2.2.4: Удаление операнды

Выполним трансляцию с измененной программой

Рис 2.2.5: Трансляция файла

Рис 2.2.5: Трансляция файла

Рис 2.2.6: Ошибка в файле .lst

Рис 2.2.6: Ошибка в файле .lst

На выходе мы не получаем файла из-за ошибки.

# 3 Самостоятельная работа

Задание№1 Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу.

Создадим новый файл ex1.asm и напишем программу нахождения наименьшей из 3 целочисленных переменных a,b и с для варианта 19 (46, 32, 74).

Рис 3.1.1: Создание файла

Рис 3.1.1: Создание файла

Возьмем за основу код lab7-2.asm и переделаем его под 1-ое задание

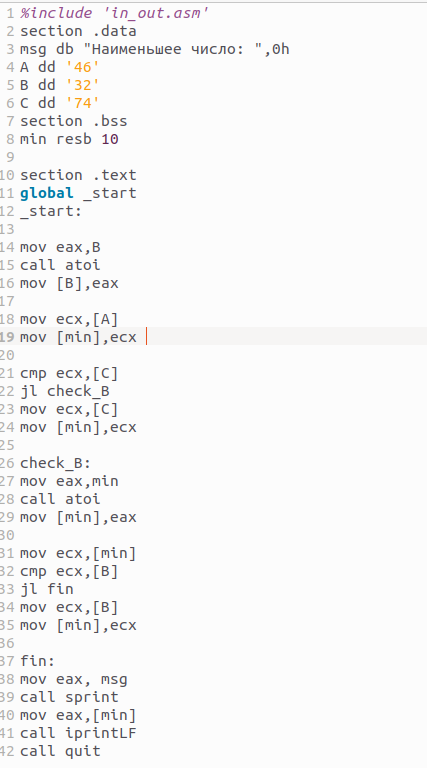


Рис 3.1.2: Программа

Проверим программу

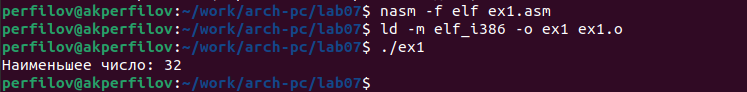


Рис 3.1.3: Проверка программы

Задание№2 Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции f(x) и выводит результат вычислений. Вид функции f(x) выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений x и a из 7.6

Создадим файл ex2.asm для 2-го задания

Рис 3.2.1: Демонстрация 19-го варианта

Рис 3.2.1: Демонстрация 19-го варианта

Напишем программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции f(x)

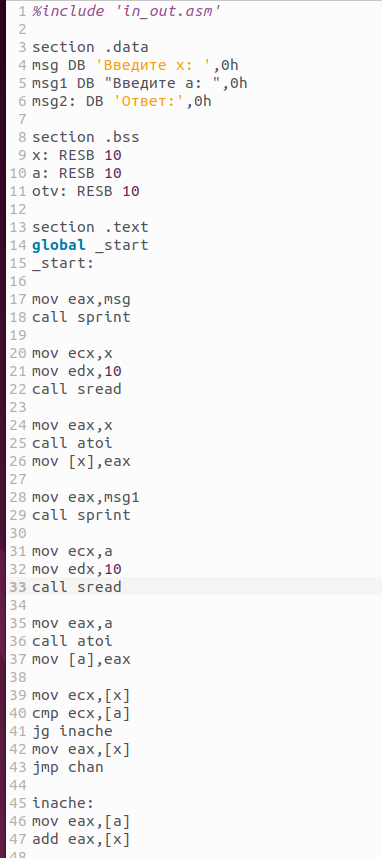


Рис 3.2.2: (1)Программа для задания

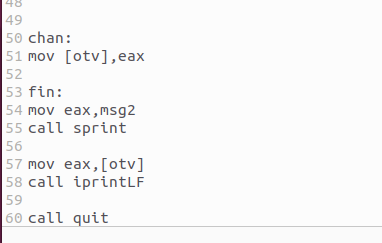


Рис 3.2.3: (2)Программа для задания

Проверим работу программы

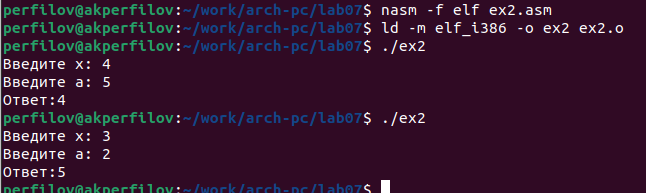


Рис 3.2.4: Проверка программы

Как видно по (Рис 3.2.4) программа работает верно

Загрузим все файлы на github

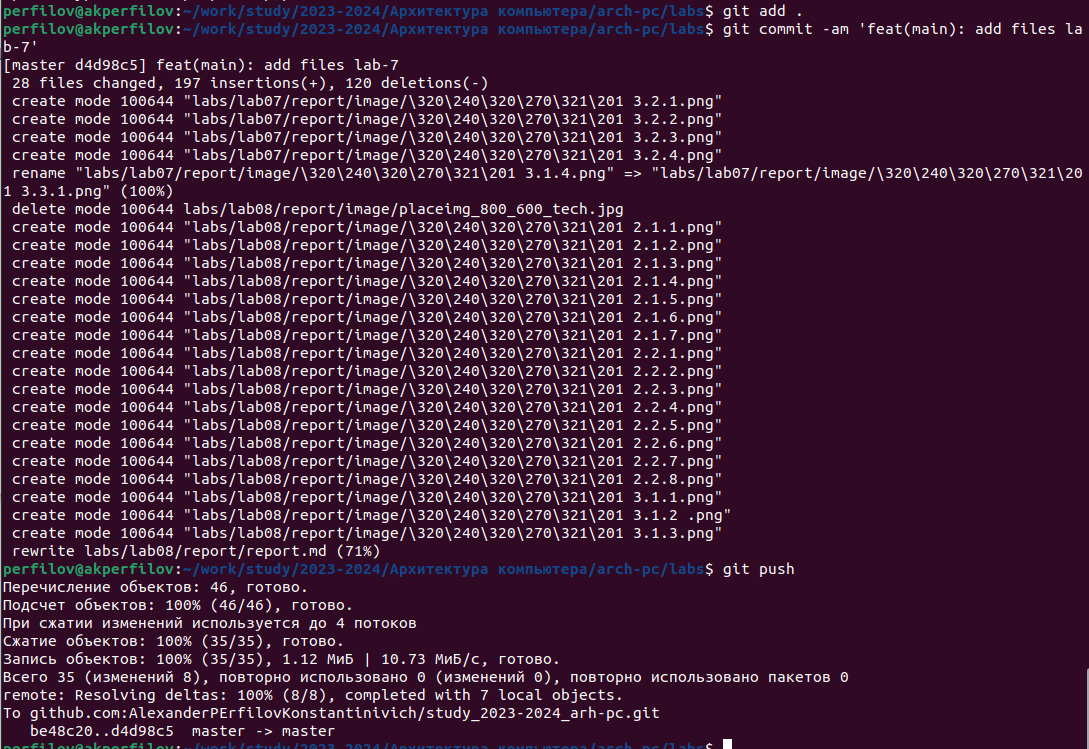


Рис 3.3.1: Загрузка файлов на github

# 4 Выводы

Я изучил команды условного и безусловного переходов. Приобрел навыки написания программ с использованием переходов. Познакомился с назначением и структурой файла листинга.