РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук Кафедра прикладной информатики и теории вероятностей

Архипов Олег Константинович

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

## 2.1 Безусловный переход

Создаю каталог ~/work/arch-pc/lab07 , перехожу в него и создаю в нем файл lab7-1.asm (рис. [1](#fig:001)).

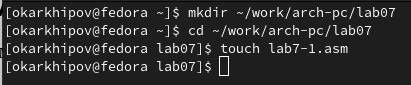


Figure 1: Директория программных файлов новой ЛР

Как и в прошлые разы копирую в созданный каталог внешний файл in\_out.asm , а затем при помощи команды gedit lab7-1.asm открываю файл в текстовом редакторе (рис. [2](#fig:002)).

Figure 2: Открываю файл в редакторе

Figure 2: Открываю файл в редакторе

Вбиваю текст программы с применением инструкции jmp из первого листинга ЛР (рис. [3](#fig:003)).

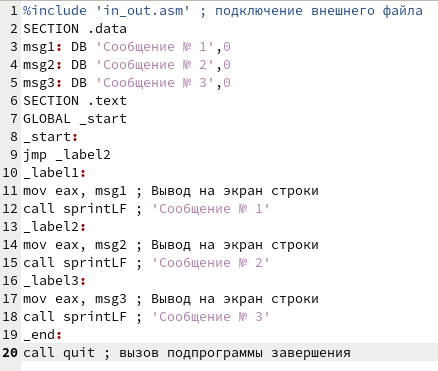


Figure 3: Текст программы

Создаю исполняемый файл для программы и запускаю его, получаю последовательность сообщений “2”->“3”, т.к. после начала программы реализуется переход к label2 минуя label1 посредством команды NASM jmp\_label2 (рис. [3](#fig:003)-[4](#fig:004)).

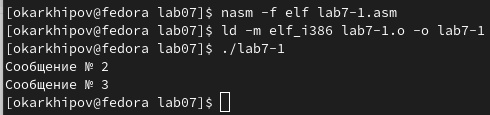


Figure 4: Исполнение программы

Изменяю программу так, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавлю инструкцию jmp с меткой \_label1 , а после вывода сообщения № 1 добавлю инструкцию jmp с меткой \_end (рис. [5](#fig:005)).

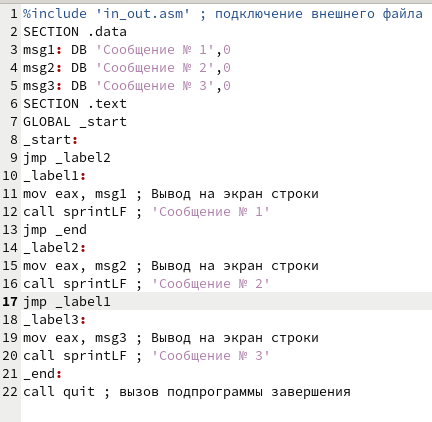


Figure 5: Текст измененной программы

Создаю и запускаю исполняемый файл, выводится верная последовательность сообщений (рис. [6](#fig:006)).

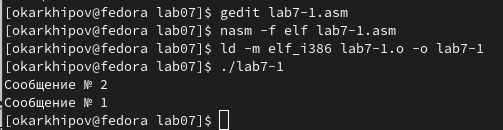


Figure 6: Результат работы измененной программы

В начале программы изменяю инструкцию jmp \_label2 на jmp \_label3 , затем после вывода сообщения № 3 добавляю jmp \_label2 , чтобы вывод програм- мы был следующим

[okarkhipov@fedora lab07]$ ./lab7-1  
Сообщение № 3  
Сообщение № 2  
Сообщение № 1  
[okarkhipov@fedora lab07]$

Получаю результат с изображений [7](#fig:007) и [8](#fig:008).

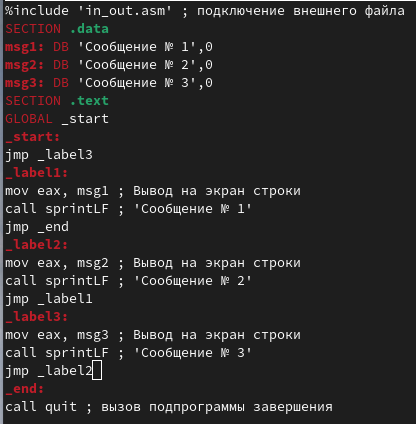


Figure 7: Второе изменение программы lab7-1.asm

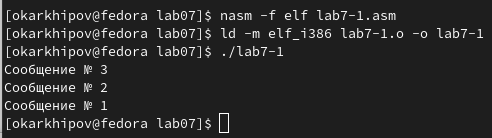


Figure 8: Результат работы второй измененной программы

## 2.2 Условный переход

Создаю файл lab7-2.asm (рис. [9](#fig:009)).

Figure 9: Создание файла lab7-2.asm

Figure 9: Создание файла lab7-2.asm

Ввожу из листинга ЛР программу, которая найдет наибольшее из трех целых чисел A, B, C (рис. [10](#fig:010)-[11](#fig:011)).

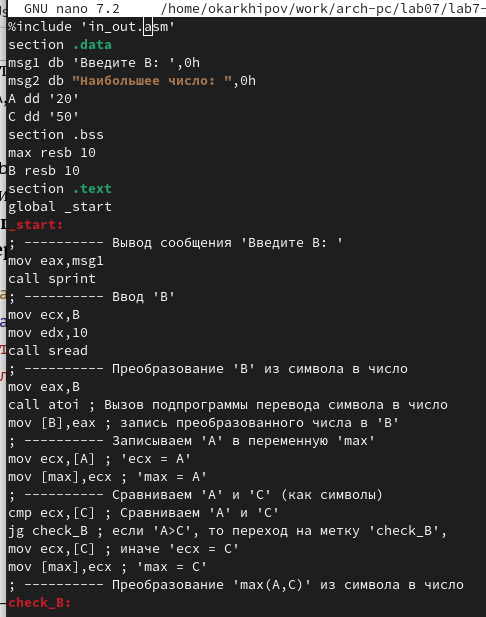


Figure 10: Ввод текста программы в lab7-2.asm

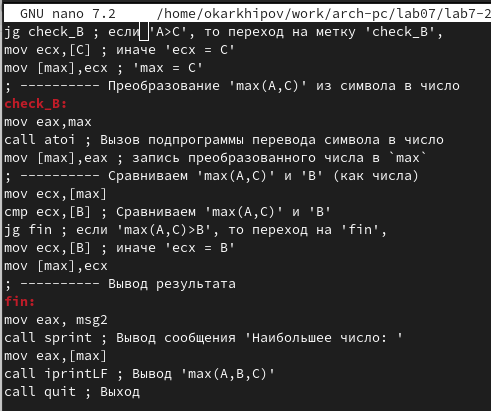


Figure 11: Ввод текста программы в lab7-2.asm прод.

Создаю исполняемый файл (рис. [12](#fig:012)).

Figure 12: Создание исполняемого файла lab7-2

Figure 12: Создание исполняемого файла lab7-2

Ввожу разные значения B: сначала 7 - наименьшее из трех чисел (рис. [13](#fig:013)), затем 51 - наибольшее (рис. [14](#fig:014)), 30 (рис. [15](#fig:015)) и 50, равное наибольшему (рис. [16](#fig:016)). Во всех случаях получаю верный результат.

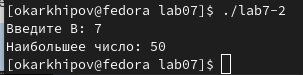


Figure 13: Работа программы lab7-2 с В=7

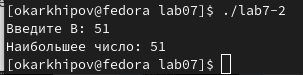


Figure 14: Работа программы lab7-2 с В=51

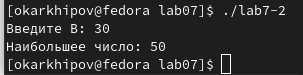


Figure 15: Работа программы lab7-2 с В=30

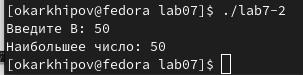


Figure 16: Работа программы lab7-2 с В=50

## 2.3 Изучение структуры файлы листинга

Ввожу <nasm -f elf -l lab7-2.lst lab7-2.asm> для создания файла листинга lab7-2.lst (рис. [17](#fig:017)).

Figure 17: Создание файла листинга

Figure 17: Создание файла листинга

Открываю его в текстовом редакторе (рис. [18](#fig:018)).

Figure 18: Команда для открытия файла lab7-2.lst в текстовом редакторе

Figure 18: Команда для открытия файла lab7-2.lst в текстовом редакторе

Рассмотрю, к примеру строки 28-30 файла листинга (отсчет строк без учета внешнего файла) (рис. [18](#fig:018)). Первая колонка, очевидно, обозначает номер строки, вторая адрес (смещение машинного кода от начала текущего сегмента), например 0000011C означает - адрес команды cmp ecx,[C] (в шестнадцатиричной системе счисления), аналогично для строк 29 и 30, 00000122 и 00000124 - адреса jg check\_B и mov ecx,[C] соответственно. Далее во всех трех строках идут машинные коды (3B0D[39000000], 7F0C, 8B0D[39000000]), в которые ассемблируются соответствующие инструкции. Наконец, сами cmp ecx,[C] , jg check\_B , mov ecx,[C] являются исходным текстом программы, первая инструкция сравнивает А и С, вторая в зависимости от результата сравнения переводит или не переводит программу на метку ‘check\_B’, третья предполагает действия, в случае, если условие jg не выполнено. Каждая из этих инструкций прокомментирована в тексте программы (рис. [19](#fig:019)).

Figure 19: Строки 28-30 файла lab7-2.lst

Figure 19: Строки 28-30 файла lab7-2.lst

Снова открываю lab7-2.asm . В строке “mov ecx, [C] ; иначе ‘ecx=C’” удаляю операнд [C] (рис. [20](#fig:020)).

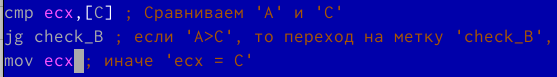


Figure 20: Удаляю операнд [C]

Выполняю трансляцию командой nasm -f elf -l lab7-2.lst lab7-2.asm , предварительно удалив старый файл листинга. Файл листинга создается, но без объектного файла (рис. [21](#fig:021)-[22](#fig:022)), вместо этого терминал выдаёт ошибку, т.к. теперь одна из инструкций записана некорректно.

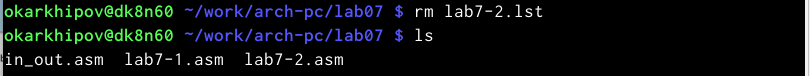


Figure 21: Удаляю старый файл листинга

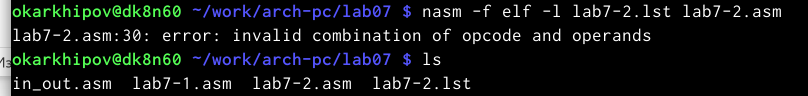


Figure 22: Выполняю трансляцию

Открываю файл листинга, на месте измененной строки стоят звездочки и описание ошибки (рис. [23](#fig:023)).

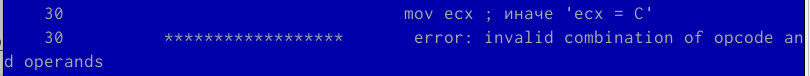


Figure 23: Файл листинга с сообщением об ошибке

После возвращаю операнд [C] на место, чтобы устранить ошибку и вновь выполняю трансляцию с получением файла листинга (рис. [24](#fig:024)-[25](#fig:025)).

Figure 24: Исходное состояние файла lab7-2.asm

Figure 24: Исходное состояние файла lab7-2.asm

Figure 25: Трансляция файла lab7-2.asm

Figure 25: Трансляция файла lab7-2.asm

# 3 Cамостоятельная работа

Напомню, в прошлой ЛР мой номер варианта был 4.

## 3.1 Задание 1

Создаю файл для программы sol1.asm (рис. [26](#fig:026)).

Figure 26: Создание файла sol1.asm

Figure 26: Создание файла sol1.asm

Чтобы решить задачу достаточно предыдущую программу изменить таким образом, чтобы она искала наименьшее значение. Для этого заменяю инструкции jg на jl и операнды max на min (рис. [27](#fig:027)-[28](#fig:028)).

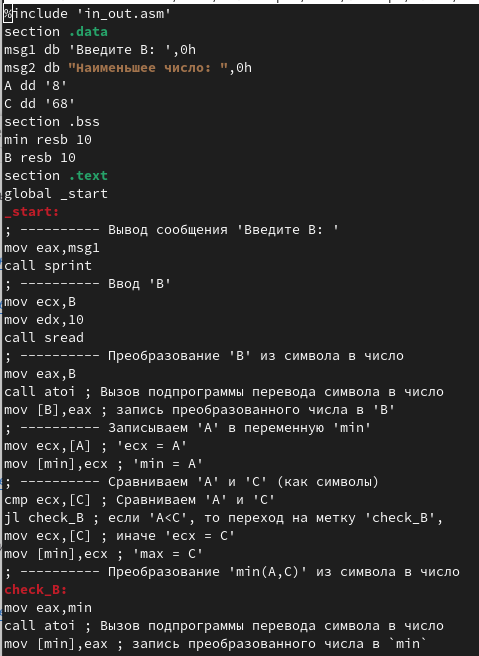


Figure 27: Программа для нахождения min 1

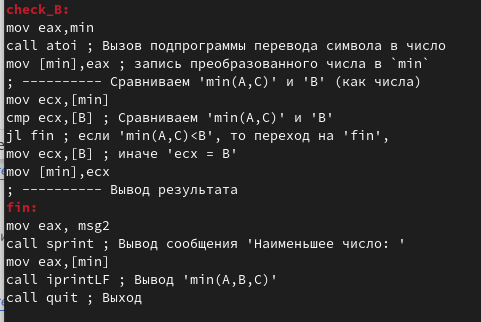


Figure 28: Программа для нахождения min 2

Создаю исполняемый файл и ввожу неинициализированную переменную B (другие 2 переменные уже есть в тексте программы), получаю ответ 8 (рис. [29](#fig:029)).

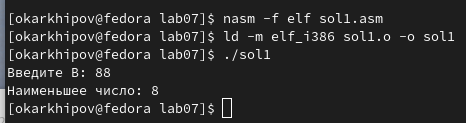


Figure 29: Результат работы программы для нахождения min

## 3.2 Задание 2



Figure 30: Мой вариант

Создаю файл для нового задания (рис. [31](#fig:031)).

Figure 31: Результат работы программы для нахождения min

Figure 31: Результат работы программы для нахождения min

Для начала задам переменные (инициализированные и неинициализированные), затем составлю часть кода вывода сообщений о вводе x и a , ввода x и a и преобразования этих переменных из символов в числа (рис. [32](#fig:032)).

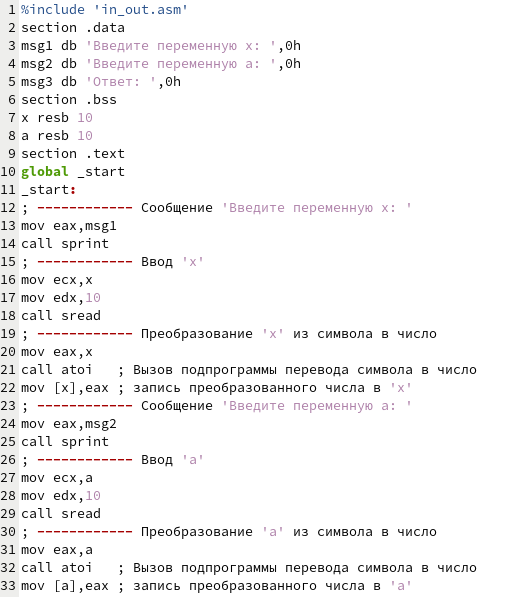


Figure 32: Первый этап программы

Затем напишу часть кода для решения задачи при a , не равном 0. Завершу программу для этого случая (рис. [33](#fig:033)).

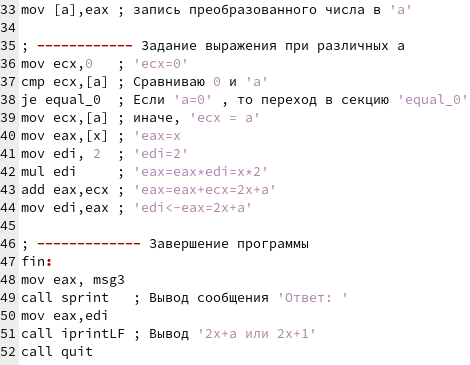


Figure 33: Второй этап программы (a не равно 0 и завершение)

В последней части рассмотрю случай ‘a=0’ и поставлю команду безусловного перехода к секции завершения программы (рис. [34](#fig:034)).

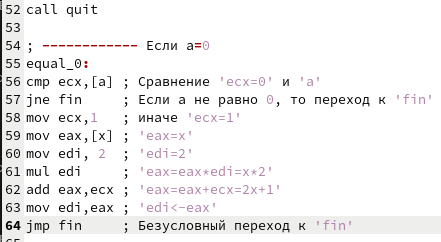


Figure 34: Третий этап программы

Запускаю то, что получилось и подставляю переменные. Получаю верные ответы в обоих случаях (рис. [35](#fig:035)).

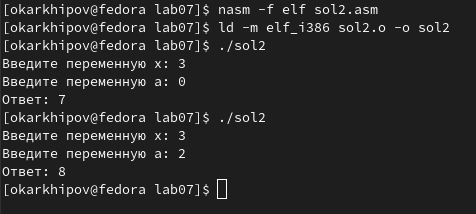


Figure 35: Запуск программы

# 4 Выводы

Усвоены команды условного и безусловного перехода, а также способы их применения. Помимо этого, была изучена структура файла листинга.