Лабораторная работа №7

Команды безусловного и условного переходов в Nasm. Программирование ветвлений.

Сырцева Анастасия Романовна

Содержание

# 1 Цель работы

Целью данной работы является изучение команд перезодов, а также приобритение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

# 2 Задание

* Работа с командами переходов;
* Знакомство с назначением и структурой файла листинга;

# 3 Теоретическое введение

## 3.1 Команды условного и безусловного перехода

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

* условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
* безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

Безусловный переход выполняется инструкцией jmp (от англ. jump – прыжок), которая включает в себя адрес перехода, куда следует передать управление.

Для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов.

Флаг – это бит, принимающий значение 1 («флаг установлен»), если выполнено некоторое условие, и значение 0 («флаг сброшен») в противном случае. Флаги работают независимо друг от друга, и лишь для удобства они помещены в единый регистр — регистр флагов, отражающий текущее состояние процессора. В следующей таблице указано положение битовых флагов в регистре флагов (рис. 1).

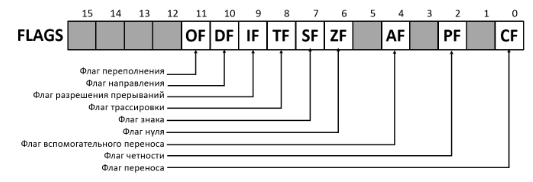


Рис. 1: Регистр флагов

Инструкция cmp является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция cmp является командой сравнения двух операндов и имеет такой же формат, как и команда вычитании.

Команда cmp, так же как и команда вычитания, выполняет вычитание, но результат вычитания никуда не записывается и единственным результатом команды сравнения является формирование флагов.

Мнемоника условного перехода связана со значением анализируемых флагов или со способом формирования этих флагов.

В табл. 1 представлены команды условного перехода, которые обычно ставятся после команды сравнения cmp. В их мнемокодах указывается тот результат сравнения, при котором надо делать переход.

Таблица 1: Инструкции условной передачи управления по результатам арифметического сравнения cmp a,b

| Типы операндов | Мнемокод | Критерий условного перехода | Комментарий |
| --- | --- | --- | --- |
| Любые | JE | a = b | Переход, если равно |
| Любые | JNE | a ≠ b | Переход, если не равно |
| Со знаком | JL/JNGE | a < b | Переход, если меньше |
| Со знаком | JLE/JNG | a ≤ b | Переход, если меньше или равно |
| Со знаком | JG/JNLE | a > b | Переход, если больше |
| Со знаком | JGE/JNL | a ≥ b | Переход, если больше или равно |
| Без знака | JB/JNAE | a < b | Переход, если ниже |
| Без знака | JBE/JNA | a ≤ b | Переход, если ниже или равно |
| Без знака | JA/JNBE | a > b | Переход, если выше |
| Без знака | JAE/JNB | a ≥ b | Переход, если выше или равно |

## 3.2 Файл листинга и его структура

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.

Строки листинга имеют следующую структуру:

* Номер строки — это номер строки файла листинга (нужно помнить, что номер строки в файле листинга может не соответствовать номеру строки в файле с исходным текстом программы);
* Арес — это смещение машинного кода от начала текущего сегмента;
* Машинный код - ассемблированная исходная строка в виде шестнадцатеричной последовательности.
* Исходный текст программы - это просто строка исходной программы вместе с комментариями (некоторые строки на языке ассемблера, например, строки, содержащие только комментарии, не генерируют никакого машинного кода, и поля «смещение» и «исходный текст программы» в таких строках отсутствуют, однако номер строки им присваивается).

Все ошибки и предупреждения, обнаруженные при ассемблировании, транслятор выводит на экран, и файл листинга не создаётся.

# 4 Выполнение лабораторной работы

Создаю папку для данной лабораторной работы, переходу в неё и создаю файл lab7-1.asm (рис. 2).

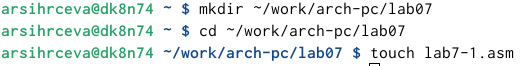


Рис. 2: Создание необходимого каталога и файла

Открываю файл и ввожу в файл текст программы с использованием новой инструкции, jmp(рис. 3).

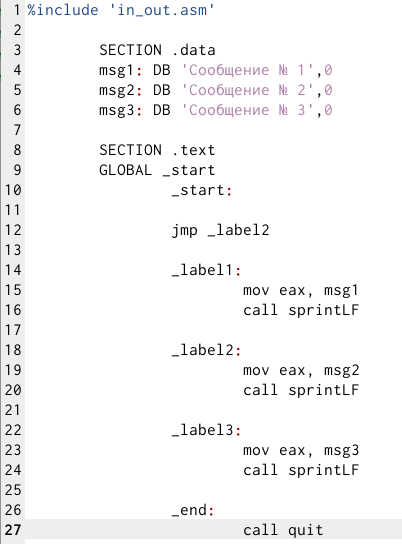


Рис. 3: Текст программы с использованием jmp

Копирую файл in\_out.asm для корректной работы программы (рис. 4).

Рис. 4: Копирование in_out.asm в рабочий каталог

Рис. 4: Копирование in\_out.asm в рабочий каталог

Создаю и запускаю исполняемый файл(рис. 5).

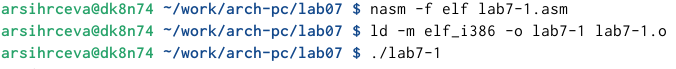


Рис. 5: Запуск исполняемого файла

Результат работы программы совпадает с тем, что дано в условии лабораторной работы(рис. 6).



Рис. 6: Результат работы программы

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

Изменяю программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу.(рис. 7).

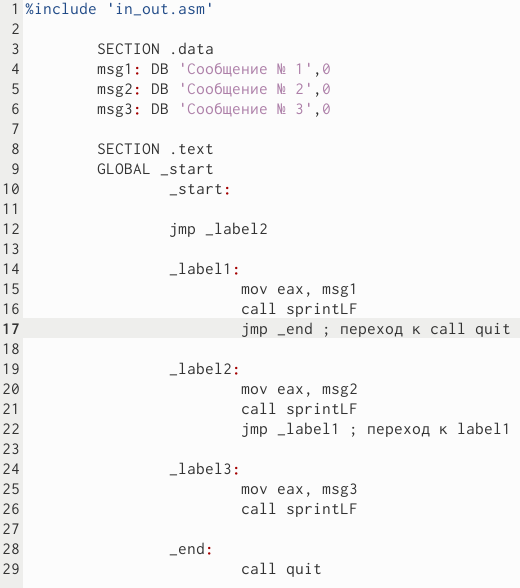


Рис. 7: Изменённый текст программы

Создаю и запускаю исполняемый файл(рис. 8).

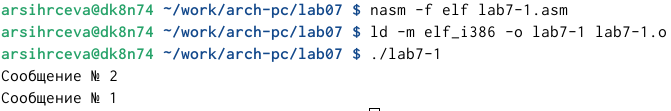


Рис. 8: Результат работы измененной программы

Результат удовлетворяет условию. Сначала выводится ‘Сообщение № 2’, потом ‘Сообщение № 1’, затем работа завершается.

Изменяю текст программы, используя jmp, таким образом, чтобы сообщения выводились в следующем порядке: ‘Сообщение №3’, ‘Сообщение № 2’, ‘Сообщение № 1’, (рис. 9).

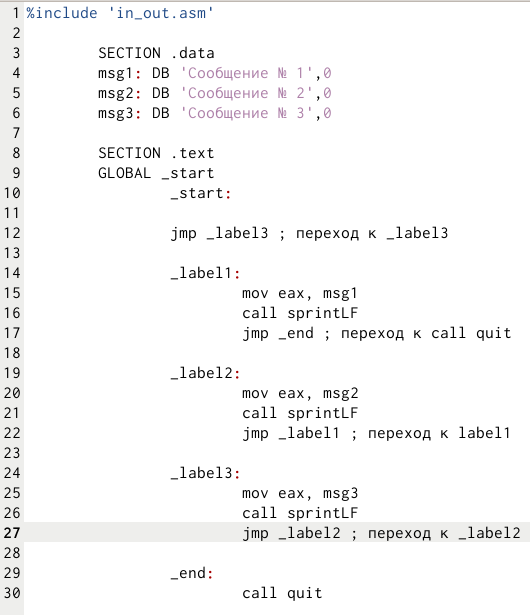


Рис. 9: Изменений текст программы

Создаю исполняемый файл и запускаю его(рис. 10).

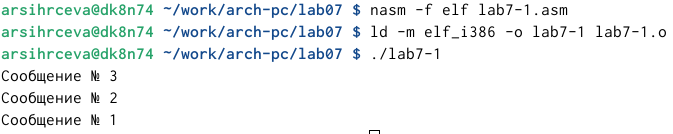


Рис. 10: Запуск файла и результат работы программы

Создаю файл lab7-2.asm и открываю его для редактирования(рис. 11).

Рис. 11: Создание файла lab7-2.asm

Рис. 11: Создание файла lab7-2.asm

Ввожу текст программы, которая выводит наибольшее из трёх чисел(рис. 12).

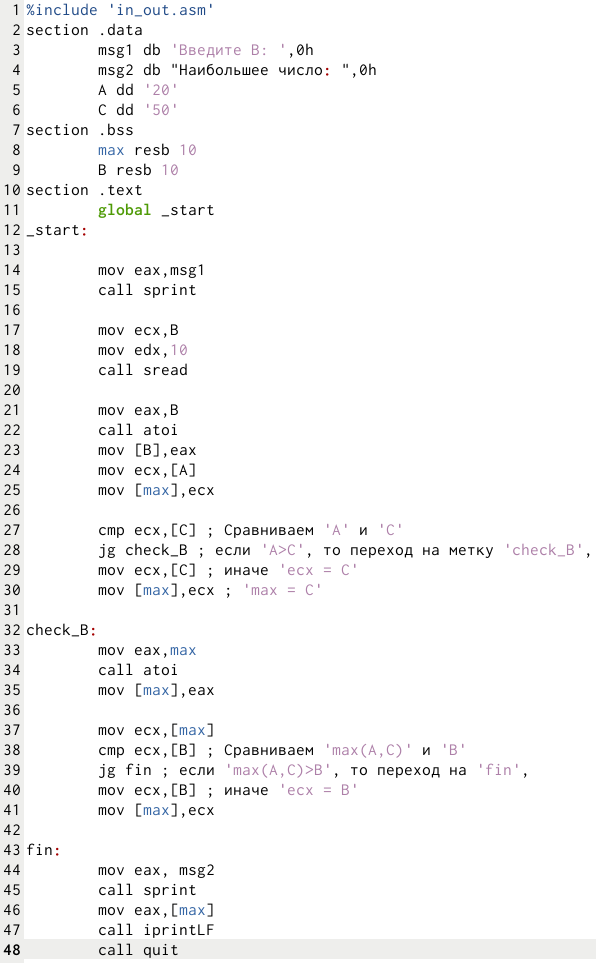


Рис. 12: Текст программы

Создаю и несколько раз запускаю исполняемый файл. На запрос программы ввожу различные числа(рис. 13).

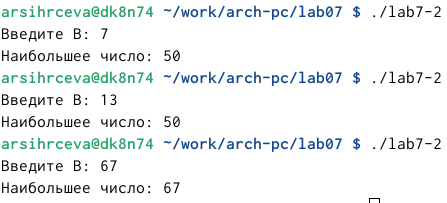


Рис. 13: Результат работы программы при различных B

Создаю файл листинга для программы из файла lab7-2.asm(рис. 14).

Рис. 14: Создание файла листинга

Рис. 14: Создание файла листинга

Открываю файл листинг в редакторе gedit(рис. 15).

Рис. 15: Открытие файла листинга в редакторе

Рис. 15: Открытие файла листинга в редакторе

Внимательно изучаю текст программы.

Подробно разбираю строку под номером 197 (рис. 16).

* ‘22’ - номер данной строки в исходном файле ;
* ‘00000106’ - адрес строки;
* ‘E891FFFFFF’ - машинный код;
* ‘call atoi’ - вызов подпрограммы перевода символа в число из исходного файла

Рис. 16: Строка 197 из файла листинга

Рис. 16: Строка 197 из файла листинга

Также разберу строку под номером 213(рис. 17).

* ‘38’ - номер данной строки в исходном файле ;
* ‘00000145’ - адрес строки;
* ‘3B0D[0A000000]’ - машинный код;
* ‘cmp ecx,[B] ; Сравниваем ’max(A,C)’ и ‘B’’ - строка из исходного файла, где ‘cmp ecx,[B]’ - команда для сравнения операнд ecx и [B], а ‘; Сравниваем ’max(A,C)’ и ‘B’’ - комментарий

Рис. 17: Строка 213 листинга

Рис. 17: Строка 213 листинга

Делаю то же самое для строки 220(рис. 18).

* ‘45’ - номер данной строки в исходном файле ;
* ‘0000015E’ - адрес строки;
* ‘E8ACFEFFFF’ - машинный код;
* ‘call sprint’’ - строка из исходного файла, вывод сообщения ‘Наибольшее число:’

Рис. 18: Строка 220 листинга

Рис. 18: Строка 220 листинга

Открываю файл lab7-2.asm и в строке 35 удаляю операнд eax(рис. 19).

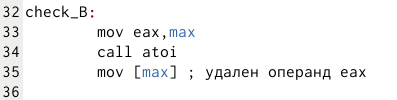


Рис. 19: Изменённая строка программы

Создаю файл листинга(рис. 20).

Рис. 20: Создание листинга

Рис. 20: Создание листинга

Выдаётся ошибка: invalid combination of opcode and operands.

Открываю файл lab7-2.lst(рис. 21).

Рис. 21: Файл изменённого листинга

Рис. 21: Файл изменённого листинга

В файле показано конкретное место, где есть ошибка(35 строка) и указано название ошибки(invalid combination of opcode and operands)

# 5 Самостоятельная работа

## 5.1 Первое задание

Создаю и открываю для редактирования файл lab7-3.asm(рис. 22).

Рис. 22: Создание файла для самостоятельной работы

Рис. 22: Создание файла для самостоятельной работы

Пишу текст программы для нахождения наименьшей из 3 целочисленных переменных a, b и c.(рис. 23).

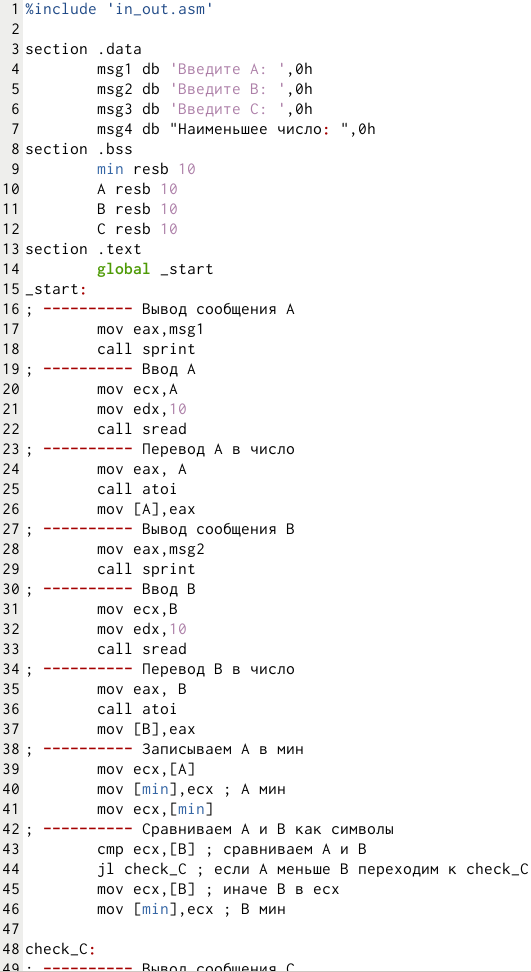


Рис. 23: Текст программы для нахождения наименьшего числа

Нахожу номер варианта, соответствующий варианту из лабораторной №6. А именно первый вариант (рис. 24).

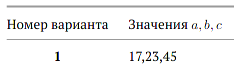


Рис. 24: Мой вариант

Создаю и запускаю исполняемый файл. На запросы программы ввожу числа из 1 варианта (рис. 25).

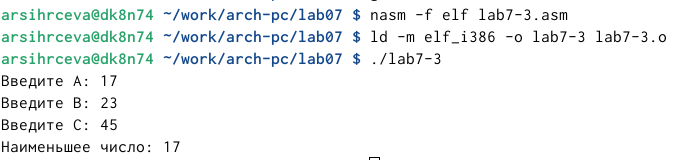


Рис. 25: Результат программы нахождения наименьшего числа

В ответе выводится число 17, наименьшее из трех введённых.

## 5.2 Второе задание

Создаю и открываю в текстовом редакторе файл lab7-4 (рис. 26).

Рис. 26: Создание файла для второго задания

Рис. 26: Создание файла для второго задания

Нахожу 1 вариант для выполнения задания (рис. 27).

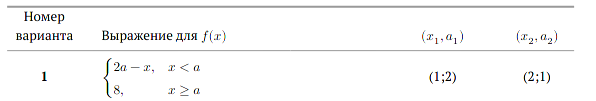


Рис. 27: Уравнение для второго задания

Ввожу текст программы, которая будет вычислять значение функции f для введённых переменных a и x(рис. 28).

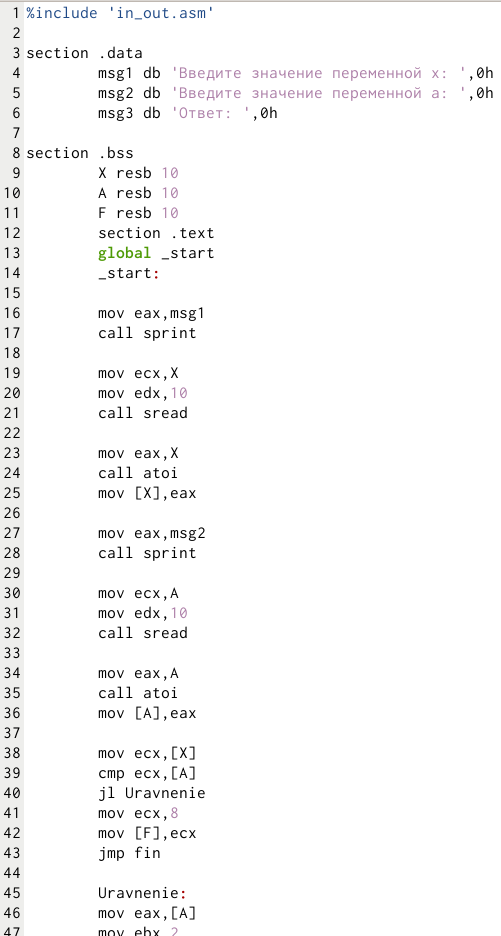


Рис. 28: Текст программы для вычисления значения функции

Создаю и запускаю исполняемый файл. На запрос программы ввожу значения x=1, a=2(рис. 29).

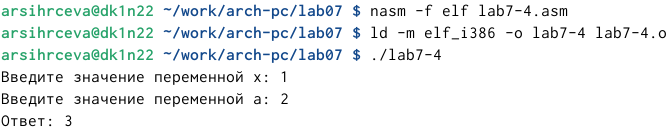


Рис. 29: Результат работы программы в первом случае

Запускаю файл ещё раз и ввожу новые значения переменных(рис. 30).



Рис. 30: Результат работы программы во втором случае

# 6 Выводы

Изучены команды условного и безусловного переходов. Получены навыки написания программ с использованием переходов. Также изучена структура и назначение файла листига.