**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**по лабораторной работе № 8**

*дисциплина: Архитектура компьютера*

Студент: Волгин А.А.

Группа: НПИбд-01-22

**МОСКВА**

2022 г.

**Цель работы:**

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

**Порядок выполнения лабораторной работы:**

**Реализация переходов в NASM.**

Создадим каталог для программ лабораторной работы №8, перейдем в него и создадим файл lab8-1.asm (рис. 1).

рис. 1. Создание каталога и файла lab8-1.asm

Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции jmp. Введем в файл lab8-1.asm следующий текст программы (рис. 2).

рис. 2. Текст программы lab8-1

Создадим исполняемый файл и запустим его (рис. 3).

рис. 3. Результат работы программы lab8-1

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) (рис. 4).

рис. 4. Измененный текст программы lab8-1

Создадим исполняемый файл и запустим его (рис. 5).

рис. 5. Результат работы измененной программы lab8-1

Далее изменим текст программы lab8-1 так, чтобы сообщения выводились в обратном порядке, затем запустим программу (рис. 6-7).

рис. 6. Измененный текст программы lab8-1

рис. 7. Результат работы измененной программы lab8-1

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 вводится с клавиатуры.

Создадим файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введем в него следующий текст программы (рис. 8-9).

рис. 8. Текст программы lab8-2 (1)

рис. 9. Текст программы lab8-2 (2)

Создадим файл и проверим его работу для разных значений B (рис. 10).

рис. 10. Работа программы lab8-2

Как видим, все работает корректно.

Обратим внимание, что 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 сравнить все 3 переменные как символы (т.е. не использовать функцию atoi). Однако если переменные преобразовать из символов в 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 lab8-2.asm (рис. 11).

рис. 11. Создание файла листинга для программы lab8-2

Затем откроем этот файл (рис. 12-13).

рис. 12. Файл листинга программы lab8-2 (1)

рис. 13. Файл листинга программы lab8-2 (2)

Как видим на рис. 12 показаны некоторые функции, прописанные в файле in\_out.asm, который мы подключаем, на рис. 13 отображена непосредственно часть текста программы lab8-2, разберем несколько строк из этого текста:

Строка 10: после обозначения строки видим 00000000 это адрес, т.е. смещение машинного кода от начала текущего сегмента, поскольку строка 10 является самым начало сегмента SECTION .bss, ее адрес будет 00000000, затем идет машинный код: <res Ah> показывает, что было зарезервировано A байт (то есть 10 байт) памяти для переменной max, которая уже отображена в самое правой строке: max resb 10 – это код программы, здесь мы выделяем память из 10 однобайтовых ячеек по адресу с меткой max.

Строка 33: ее адрес уже равняется 00000122, 7FOC – ассемблированная инструкция jg, которая используется в этой строке для условной передачи управления по результатам арифметического сравнения в 32 строке ecx и [C].

Откроем файл с программой lab8-2.asm и в любой инструкции с двумя операндами удалим один операнд. Выполним трансляцию с получением файла листинга (14-15).

рис. 14. Удаление операнда msg1 в строке move ax, msg1

рис. 15. Листинг программы с удаленным операндом

Как видим, ассемблер немного ругается: в листинге отображается, что указана неверная комбинация операндов как раз в той строке, в которой мы убрали один операнд.

**Порядок выполнения самостоятельной работы:**

Напишем программу (lab8-3) нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения для моего варианта (15) будут следующими: a = 32, b = 6, c = 54. Создадим исполняемый файл и проверим его работу (рис. 16-17).

рис. 16. Текст программы lab8-3

В данном случае я сначала сравниваю A и B, если A<B, идем сразу на метку check\_C, если нет, то присваиваем регистру ecx значение B, тот же процесс происходит, когда сравниваем ecx и С, только теперь программа переходит на метку fin.

рис. 17. Результат работы программы

Как видим, все работает корректно.

**Вывод:**

Во время выполнения лабораторной работы были изучены команды условного и безусловного переходов, приобретены навыки написания программ с использованием переходов, изучено назначение и структура файла листинга.