Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Абрамова Ульяна МИхайловна

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 Задание

1. Реализация переходов в NASM
2. Изучение структуры файла листинга
3. Выполнение заданий для самостоятельной работы

# 3 Теоретическое введение

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 про- граммы без каких-либо условий

## 3.1 Команды безусловного перехода

Безусловный переход выполняется инструкцией jmp (от англ. jump – прыжок), которая включает в себя адрес перехода, куда следует передать управление: jmp Адрес перехода может быть либо меткой, либо адресом области памяти, в которую предва- рительно помещен указатель перехода. Кроме того, в качестве операнда можно использовать имя регистра, в таком случае переход будет осуществляться по адресу, хранящемуся в этом регистре

## 3.2 Команды условного перехода

Как отмечалось выше, для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов.

## 3.3 Файл листинга и его структура

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, созда- ваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию. Все ошибки и предупреждения, обнаруженные при ассемблировании, транслятор выводит на экран, и файл листинга не создаётся. Итак, структура листинга: - 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); - адрес — это смещение машинного кода от начала текущего сегмента; - машинный код представляет собой ассемблированную исходную строку в виде шестна- дцатеричной последовательности. (например, инструкция int 80h начинается по сме- щению 00000020 в сегменте кода; далее идёт машинный код, в который ассемблируется инструкция, то есть инструкция int 80h ассемблируется в CD80 (в шестнадцатеричном представлении); CD80 — это инструкция на машинном языке, вызывающая прерывание ядра); - исходный текст программы — это просто строка исходной программы вместе с ком- ментариями (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).

# 4 Выполнение лабораторной работы

## 4.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы №7 и,перейдя в него, создаю файл lab7-1.asm, в который ввожу программу с использованием инструкции jmp (рис. 1).

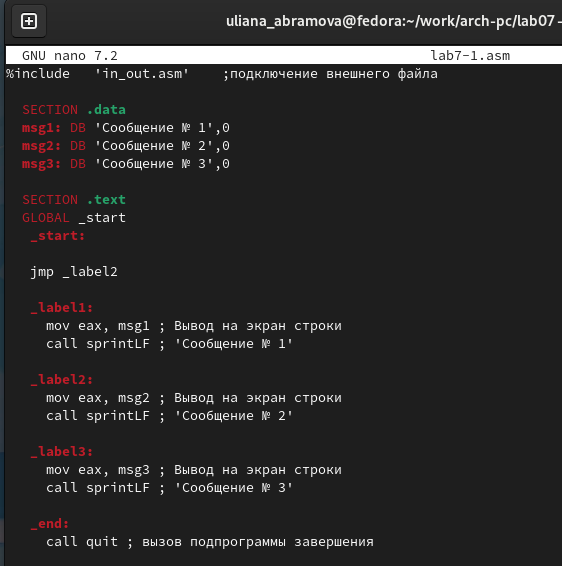


Рис. 1: Написание программы

Создаю исполняемый файл и запускаю его. Программа отработана верно (рис. 2).

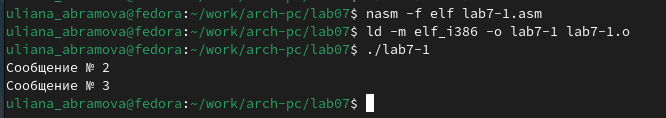


Рис. 2: Запуск программы

Инструкция jmp позволяет осуществлять переходы нетолько вперед,но и назад. Изменяю 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. 3)

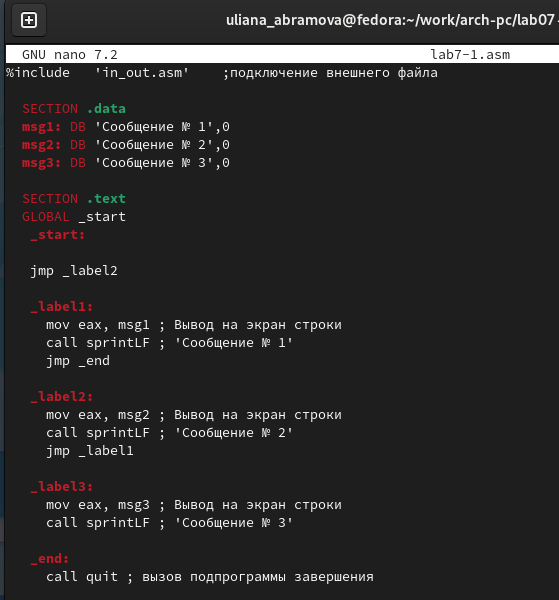


Рис. 3: Изменение текста программы

Создаю исполняемый файл и проверяю его работу (рис. 4)

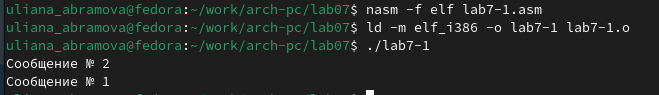


Рис. 4: Запуск программы

Изменяю текст программы, чтобы в выводе было сначала ‘Сообщение №3’, потом ‘Сообщение №2’ и последним ‘Сообщение №1’ (рис. 5).

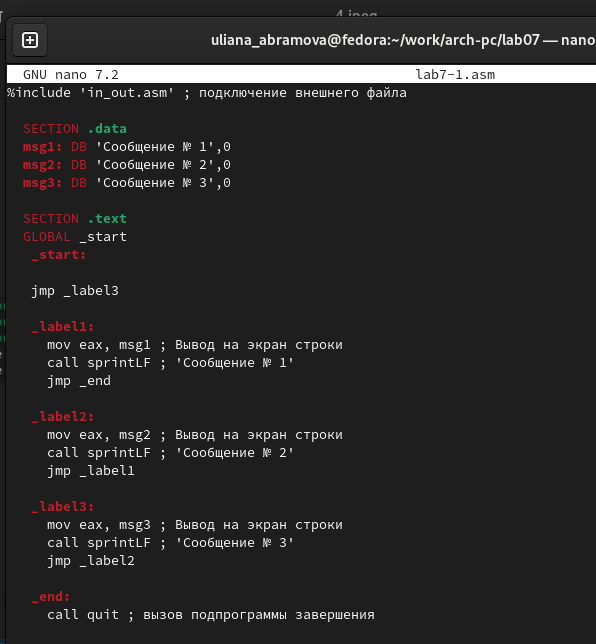


Рис. 5: Изменение программы

Создаю исполняемый файл и проверяю корректность написания программы (рис. 6).

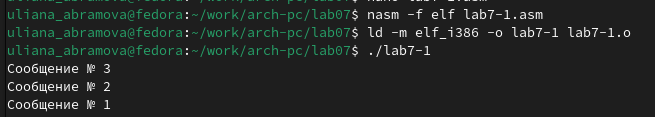


Рис. 6: Заупкс программы

Создаю в том же каталоге файл lab7-2.asm и ввожу в нем программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A,B и C (рис. 7)

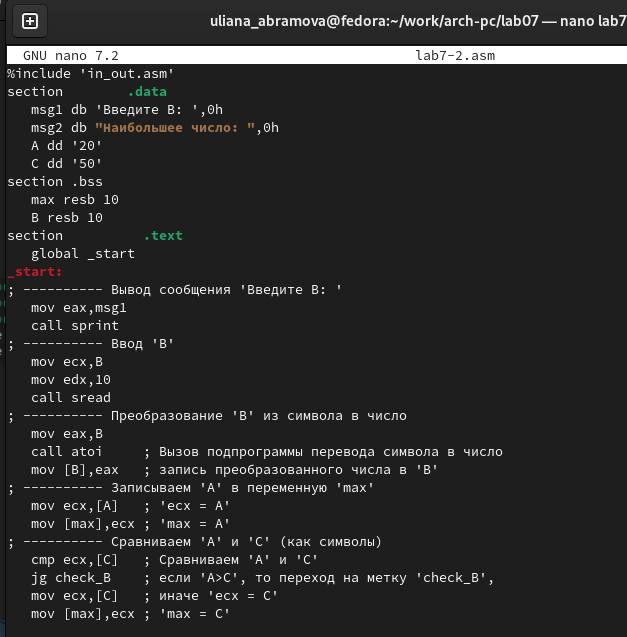


Рис. 7: Написание программы

Создаю исполняемый файл и проверяю его работу для различных знаений B (рис. 8,9)

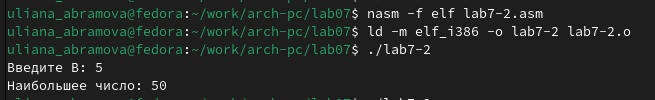


Рис. 8: Запуск программы

Рис. 9: Дополнительная проверка

Рис. 9: Дополнительная проверка

## 4.2 Изучение структуры файла листинга

Создаю файл листинга для программы из файла lab7-2.asm и открываю с помощью текстового редактора mcedit для ознакомления с его форматом и содержимым (рис. 10)

Рис. 10: Создание и открытие файла листинга

Рис. 10: Создание и открытие файла листинга

### 4.2.1 Объяснение содержимого некоторых строк

1. Эта строка находится на 21 месте, ее адрес “00000101”, Машинный код - В8 [0A000000],

mov eax,B – исходный текст программы, означающий что в регистр eax мы вносим значения переменной B. (рис. 11)

Рис. 11: mov eax,B

Рис. 11: mov eax,B

1. Эта строка находится на 35 месте, ее адрес “00000134”, Машинный код - E863FFFFFF, call atoi – исходный текст программы, означающий что символ лежащий в строке выше переводится в число. (рис. 12)

Рис. 12: call atoi

Рис. 12: call atoi

1. Эта строка находится на 47 месте, ее адрес “00000162”, Машинный код - A1[00000000], mov eax,[max] – исходный текст программы, означающий что число хранившееся в переменной max записывается в регистр eax. (рис. 13)

Рис. 13: mov eax,[max]

Рис. 13: mov eax,[max]

В инструкции ‘mov eax,max’ удаляю max и пытаюсь создать исполняемый файл, однако возникает ошибка, так как для программы требуется два операнда (рис. 14,15).

Рис. 14: Удаление одного из операндов в инструкции

Рис. 14: Удаление одного из операндов в инструкции

Рис. 15: Попытка создания файла

Рис. 15: Попытка создания файла

В файле листинга показывается где именно ошибка и с чем она связана (рис. 16)

Рис. 16: Отображение ошибки

Рис. 16: Отображение ошибки

## 4.3 Выполнение заданий для самостоятельной работы

### 4.3.1 Написание программы нахождения наименьшей из 3 целочисленных переменных a,b и с

Так как числа были изначально указаны в условии, я сделала возможным их ввести с клавиатуры. Числа: a=94; b=5; c=58 (рис. 17)

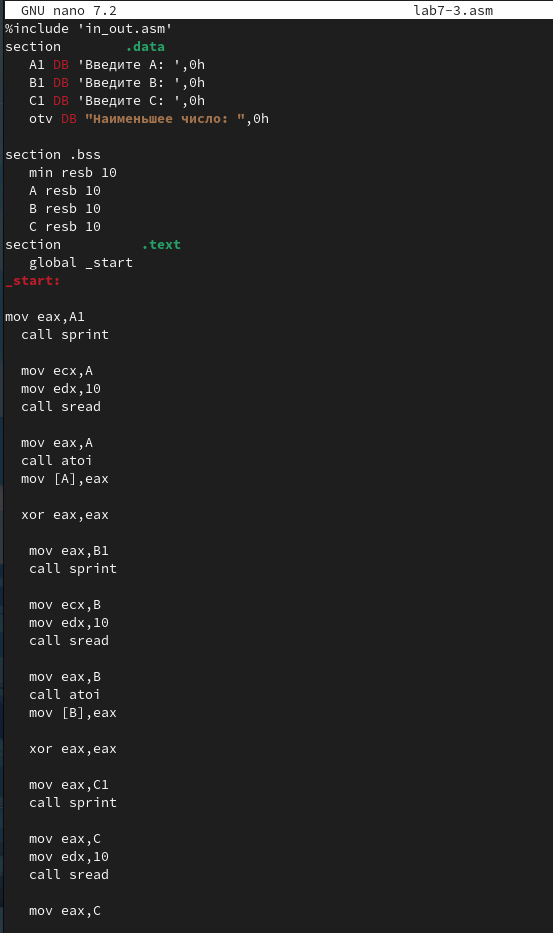


Рис. 17: Написание программы

Создаю исполняемый файл и проверяю верную отработку программы (рис. 18).

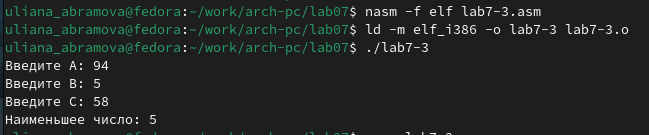


Рис. 18: Запуск программы

### 4.3.2 Написание программы вычисления выражения при введенных х и а

Программа написана для вида функции 3 варианта (рис. 19).

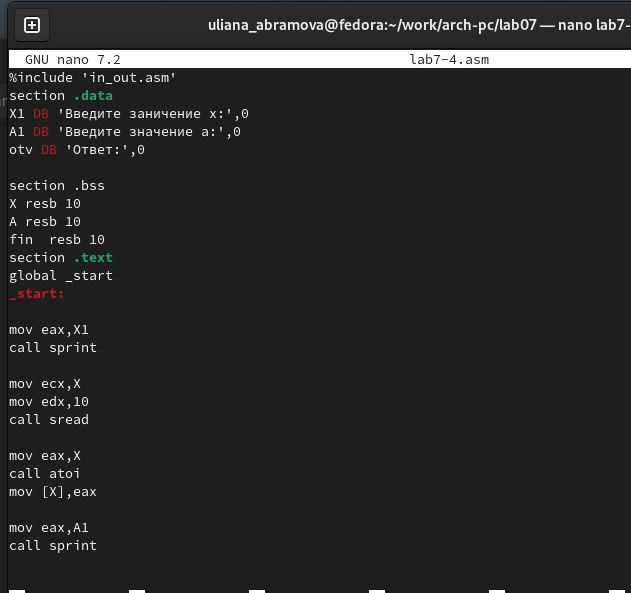


Рис. 19: Написание программы

Создаю исполняемый файл и проверяю работу программы (рис. 20).

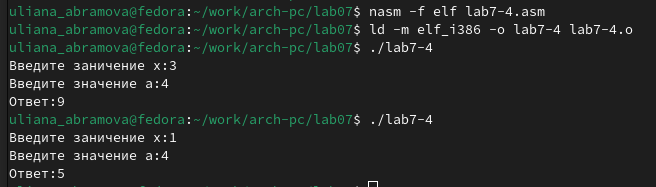


Рис. 20: Запуск программы

# 5 Выводы

Я изучила команды условного и безусловного перехода. Приобрела навыки написания программ с переходами.

# 6 Список литературы

1. [Архитектура ЭВМ](https://esystem.rudn.ru/pluginfile.php/2089545/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%967.%20%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B%20%D0%B1%D0%B5%D0%B7%D1%83%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%B8%20%D1%83%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%BF%D0%B5%D1%80%D0%B5%D1%85%D0%BE%D0%B4%D0%BE%D0%B2%20%D0%B2%20Nasm.%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%B2%D0%B5%D1%82%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9..pdf)