Отчет по лабораторной работе 8

Команды безусловного и условного переходов в Nasm. Программирование ветвлений.

Гисматуллин Артём Вадимович НПИбд-01-22

Содержание

# 1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навы- ков написания программ с использованием переходов. Знакомство с назначе- нием и структурой файла листинга.

# 2 Задание

1. Реализовать безусловный переход с помощью листинга 8.1. Выполнить соответствующее задание с выводом сообщений в определенном порядке.
2. Реализовать программу листинга 8.3, выводящую на экран наибольшее из трех целочисленных значений.
3. Изучить структуру файлов листинга. Ответить на вопросы.
4. Реализовать прорамму по нахождению наименьшего из трех целочисленных значений.
5. Написать программу, которая будет вычислять значение функции с определенным условием и данными значениями переменных (по вариантам).

# 3 Теоретическое введение

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

• условный переход – выполнение или не выполнение перехода в определен- ную точку программы в зависимости от проверки условия.

• безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

## 3.1 Команды безусловного перехода

Безусловный переход выполняется инструкцией jmp, которая включает в себя адрес перехода, куда следует передать управление:

jmp

Адрес перехода может быть либо меткой, либо адресом области памяти, в которую предварительно помещен указатель перехода. Кроме того, в качестве операнда можно использовать имя регистра, в таком случае переход будет осу- ществляться по адресу, хранящемуся в этом регистре.

## 3.2 Команды условного перехода

В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов.

### 3.2.1 Флаги

Флаг – это бит, принимающий значение 1 («флаг установлен»), если выпол- нено 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)

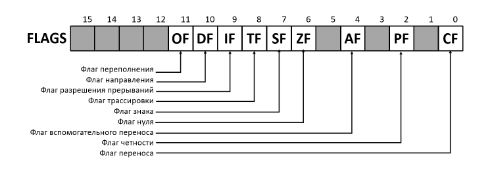


Рис. 1: Регистр флагов.

Флаги состояния (биты 0, 2, 4, 6, 7 и 11) отражают результат выполнения арифметических инструкций, таких как ADD, SUB, MUL, DIV.

### 3.2.2 Описание инструкции cmp

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 вычитания:

cmp ,

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

### 3.2.3 Описание команд условного перехода.

Команда условного перехода имеет вид

j label

Мнемоника перехода связана со значением анализируемых флагов или со способом формирования этих флагов. В таблице (рис. 2) представлены некоторые команды условного перехода, которые обычно ста- вятся после команды сравнения cmp. В их мнемокодах указывается тот резуль- тат сравнения, при котором надо делать переход. Мнемоники, идентичные по своему действию, написаны в таблице через дробь (например, ja и jnbe). Про- граммист выбирает, какую из них применить, чтобы получить более простой для понимания текст программы.

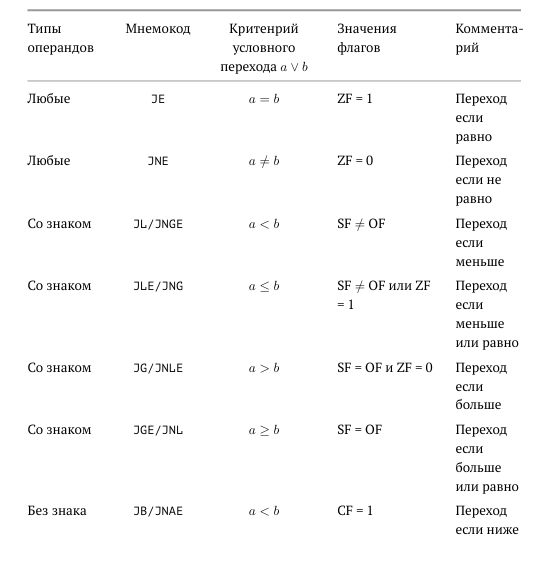


Рис. 2: Аримфетическое сравнение cmp a и b

## 3.3 Файл листинга и его структура

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

Все ошибки и предупреждения, обнаруженные при ассемблировании, транс- лятор 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 шестнадцатеричной последовательности. (например, инструкция int 80h начинается по смещению 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 вме- сте с комментариями (некоторые строки на языке ассемблера, например, строки, содержащие только комментарии, не генерируют никакого ма- шинного кода, и поля «смещение» и «исходный текст программы» в таких строках отсутствуют, однако номер строки им присваивается).

# 4 Выполнение лабораторной работы

1. Создаем каталог для программам лабораторной работы No 8, перейдим в него и создаем файл lab8-1.asm. (рис. 3)

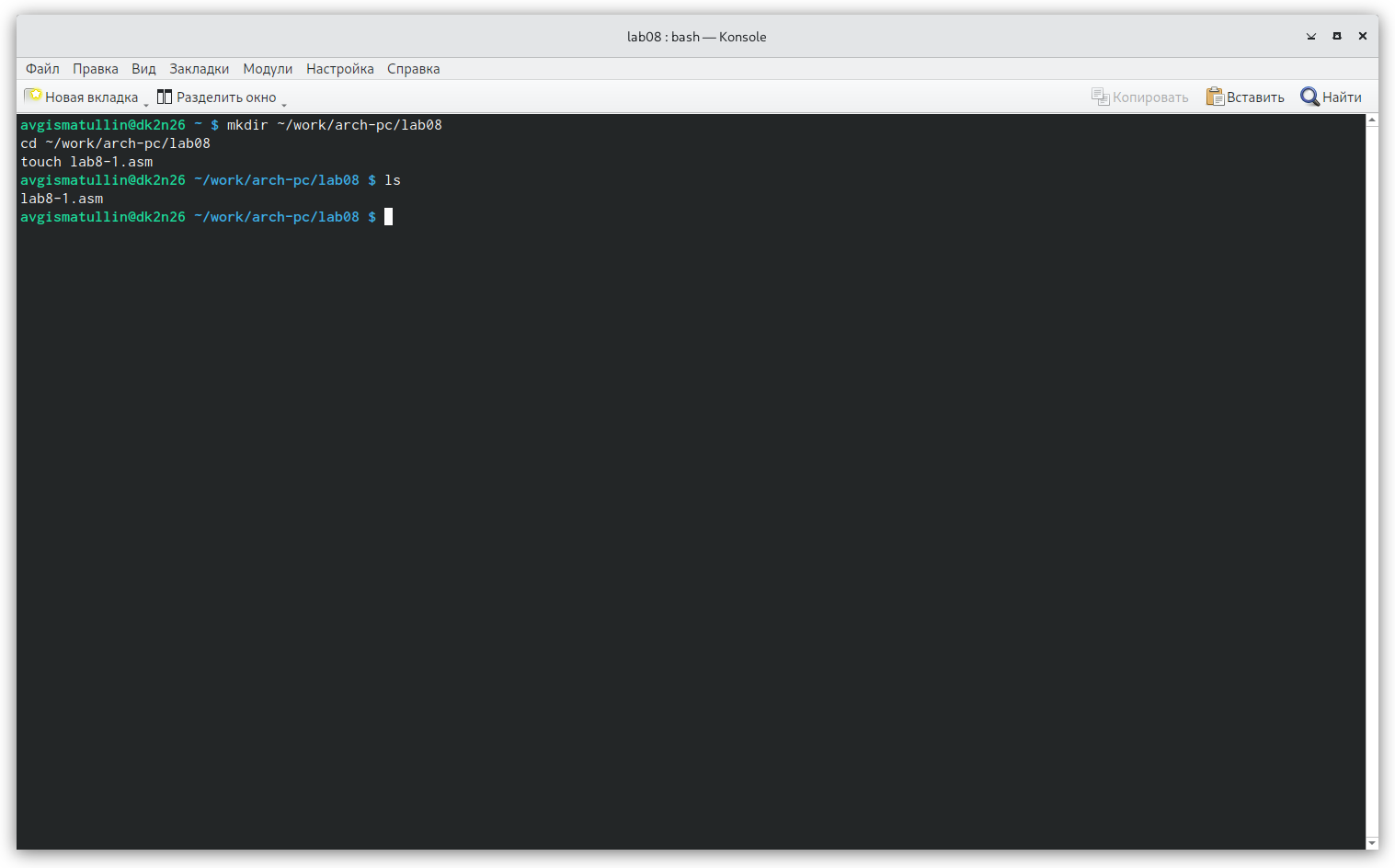


Рис. 3: Командная строка. каталог ~/work/arch-pc/lab08

1. Ввели в файл lab8-1.asm текст программы листинга, создали исполняемый файл и посмотрели результат программы. (рис. 4)

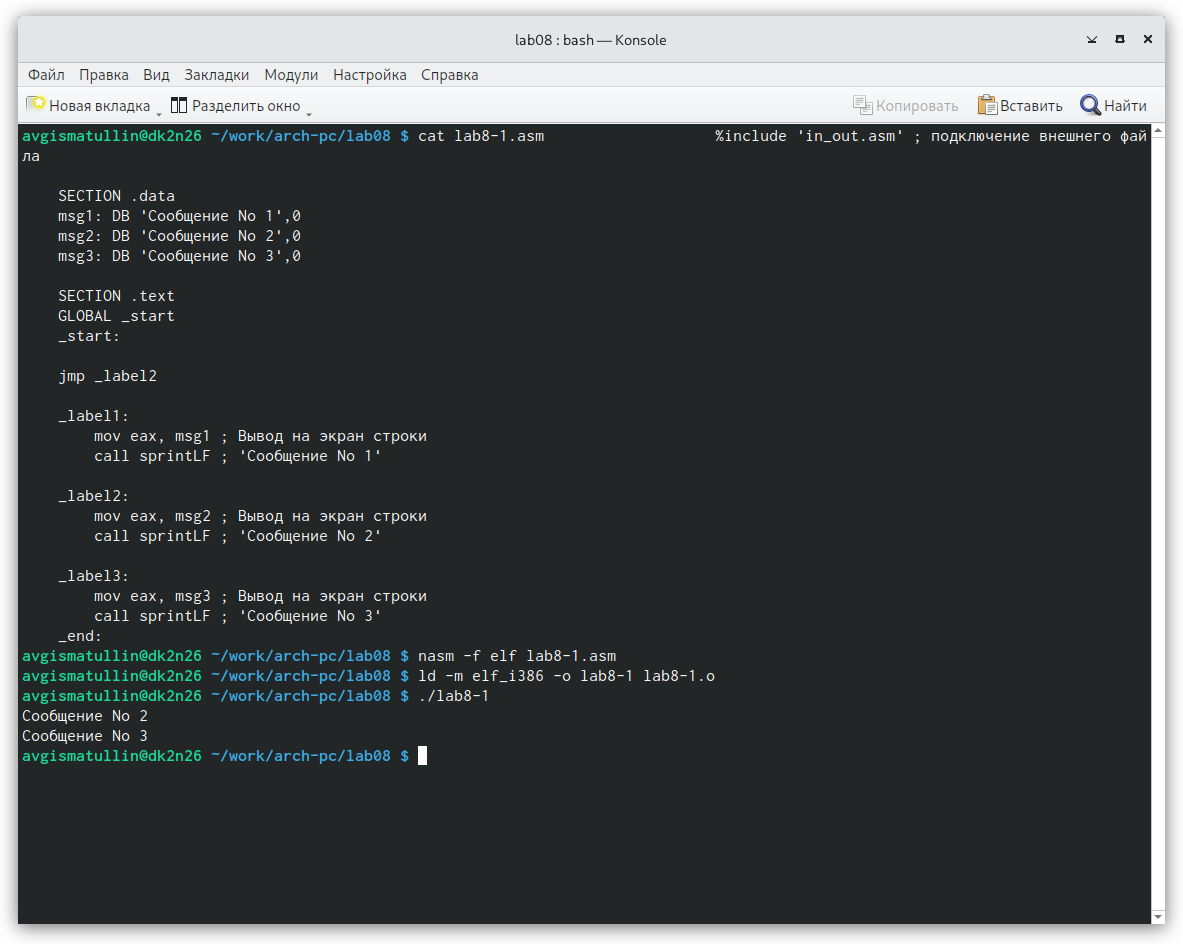


Рис. 4: Командная строка. Выполнение программы lab8-1.asm

1. Внесли в файл lab8-1.asm изменение такие, чтобы программа делала другую очередность вывода сообщения, создали исполняемый файл и посмотрели результат программы. (рис. 5)

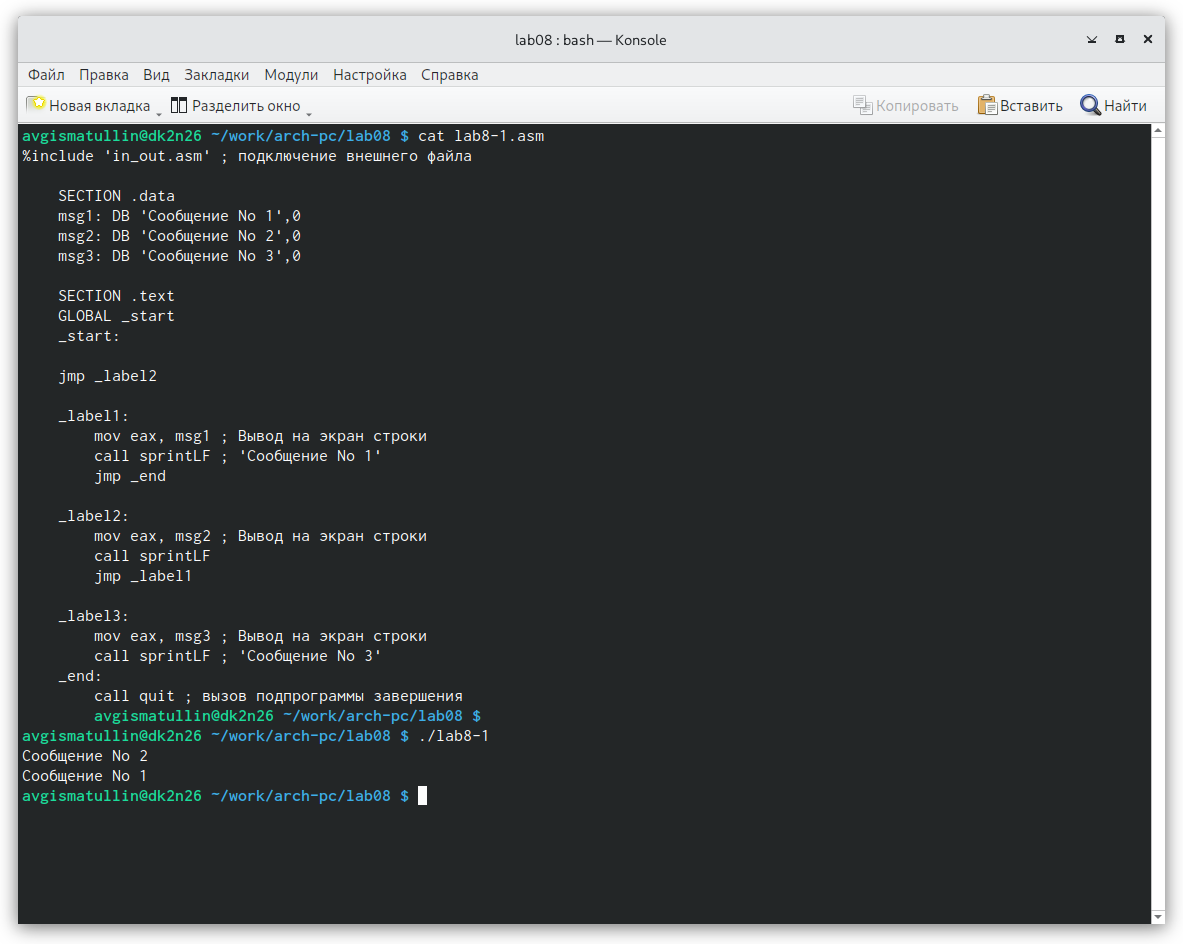


Рис. 5: Командная строка. Выполнение измененной программы lab8-1.asm

1. Внесли в файл lab8-1.asm другие изменения - теперь программа выводит сообщения в обратном порядке, создали исполняемый файл и посмотрели результат программы. (рис. 6)

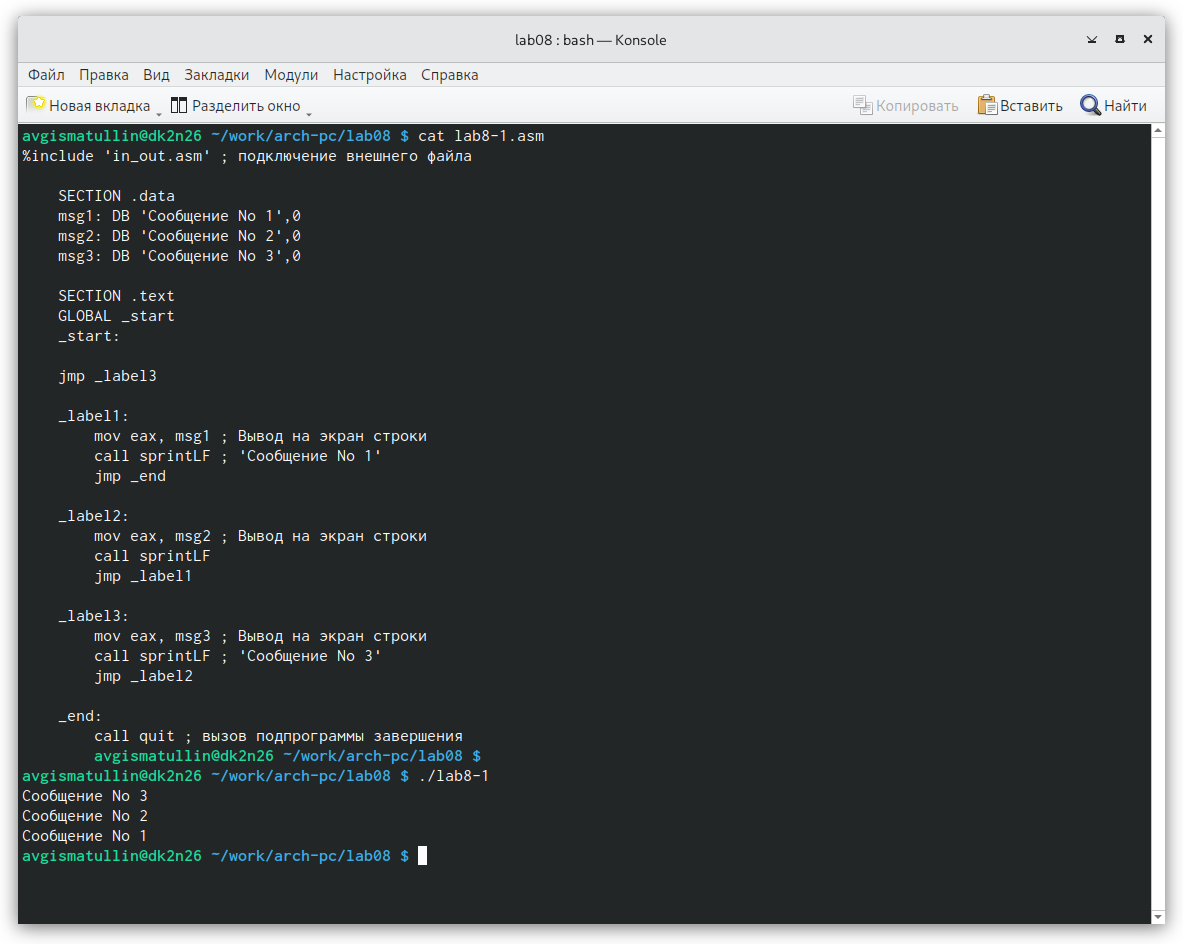


Рис. 6: Командная строка. Выполнение измененной программы lab8-1.asm 2

1. Создали файл lab8-2.asm в каталоге ~/work/arch-pc/lab08. Внимательно изучили текст программы из листинга 8.3 и ввели в lab8-2.asm. Проверили работоспособность, введя различные значения В. (рис. 7)

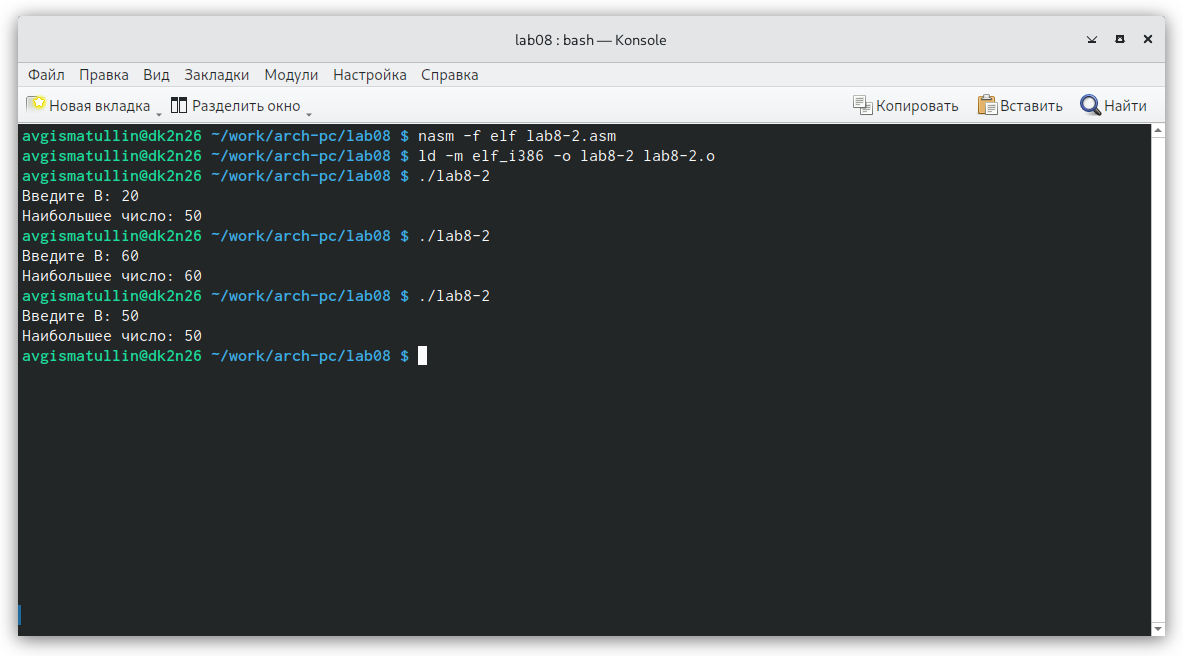


Рис. 7: Командная строка. Выполнение программы lab8-2.asm 2

1. Получили файл листинга программы lab8-2.asm. Изучили его содержимое. (рис. 8)

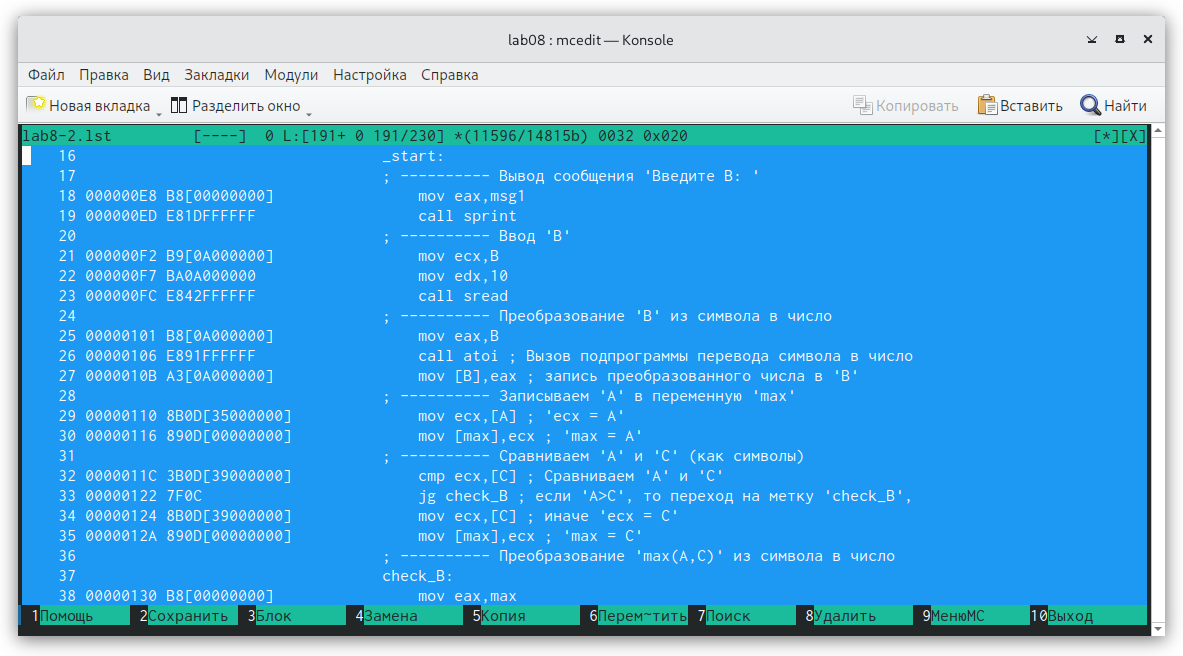


Рис. 8: Командная строка. Просмотр содержимого файла листинга

Если обратить внимание на строки 18-20, то можно сделать вывод о следующем:

* Исходный текст программы - это просто строка исходной программы с коммантариями. Комментарии не генерируют никакого машинного кода, но отдельной строке-комментарию присваивается ее номер (строка 20).
* Строки 18 и 19 содержат так же адрес (смещение машинного кода от начала текущего сегмента, например, 000000Е8) и машинный код (В8), который представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности. В8, к примеру, - это инструкция на машинном языке: в регистр eax помещается содержимое msg.

1. Далее мы удаляем один операнд инструкции mov и проверяем файл листинга. (рис. 9)

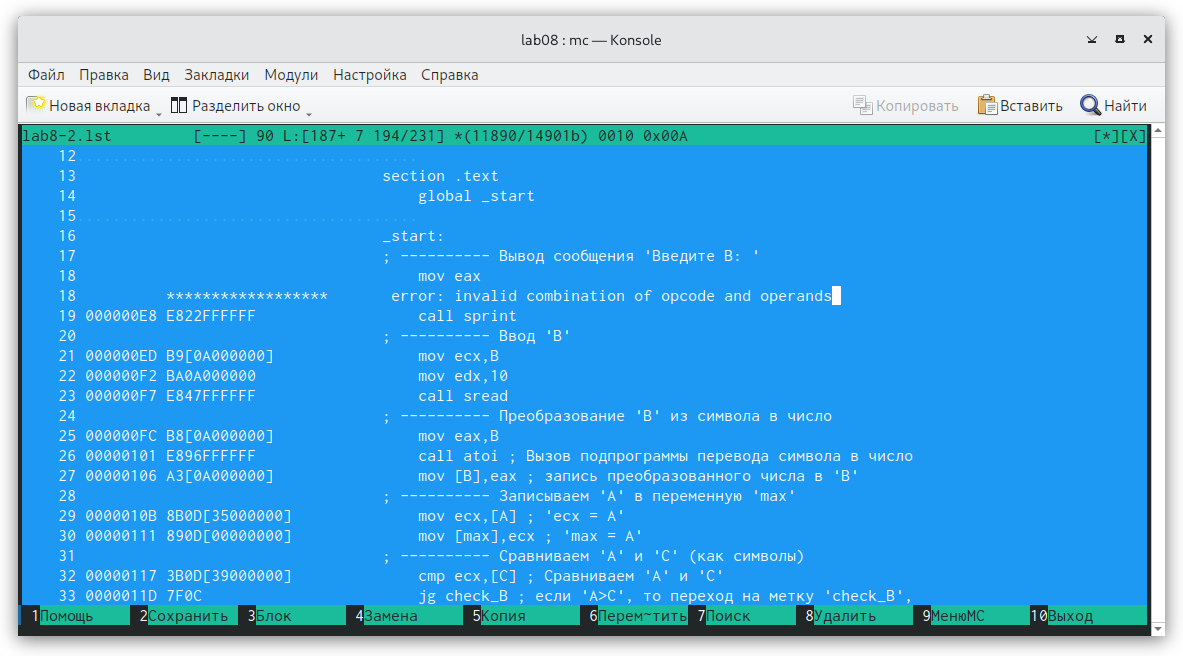


Рис. 9: Командная строка. Просмотр содержимого измененного файла листинга

Можем заметить, что на месте ошибки слева отображены звездочки вместо адреса и машинного кода, что уже свидетельствует об ошибке. Кроме того, справа отображено описание самой ошибки.

# 5 Выолнение задание для самостоятельной работы

Суть задания для самостоятельной работы заключается в том, чтобы написать программу для решения некоторой функции, а также программу, которая будет выводить по трем полученным целочисленным значениям переменных наименьшее из них. Так как на прошлой лабораторной работе я нашел, что мой вариант - 11, то значения я буду брать соответствующие.

1. Выполнение заданий начну с того, что напишу программу (lab8-3.asm), которая будет выводить наименьшее значение из трех. Основываясь на полученных знаниях, я написал ее следующим образом: (рис. 10)

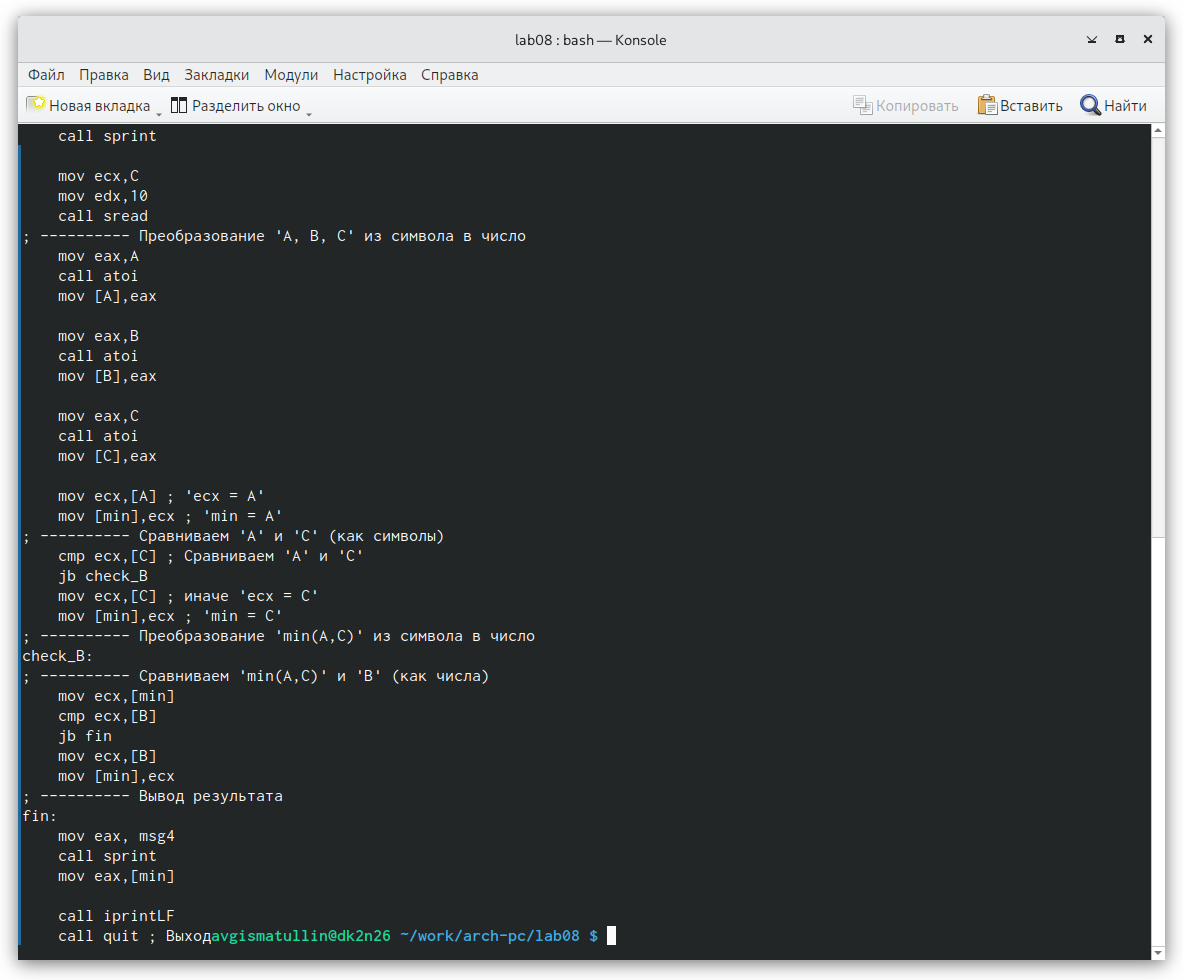


Рис. 10: Командная строка. Просмотр содержимого файла lab8-3.asm

Проверка работоспособности файла: (рис. 11)

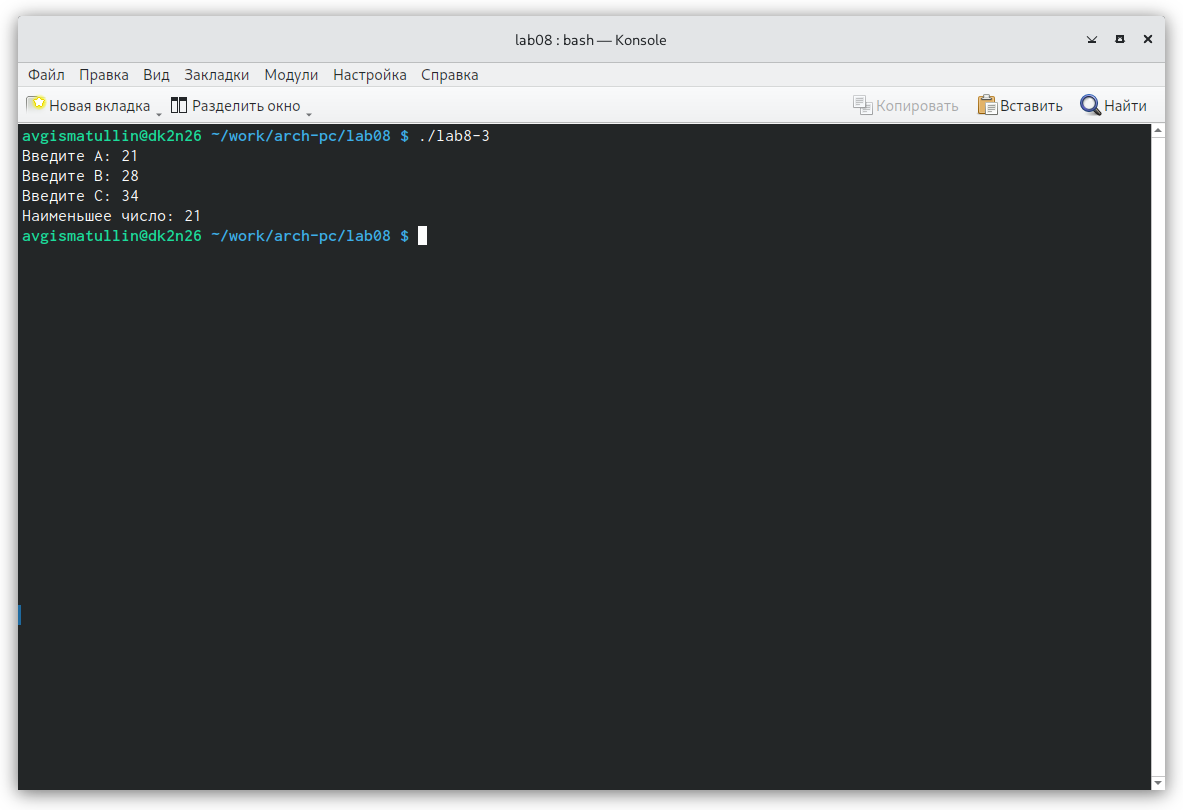


Рис. 11: Командная строка. Выполнение программы lab8-3.asm

1. Далее следует написать код (lab8-4.asm), который по введенным значениям проверяет их на условия и выводит получившийся результат. Условия прописаны в самой функции под вариантом 11. Код выглядит так: (рис. 12)

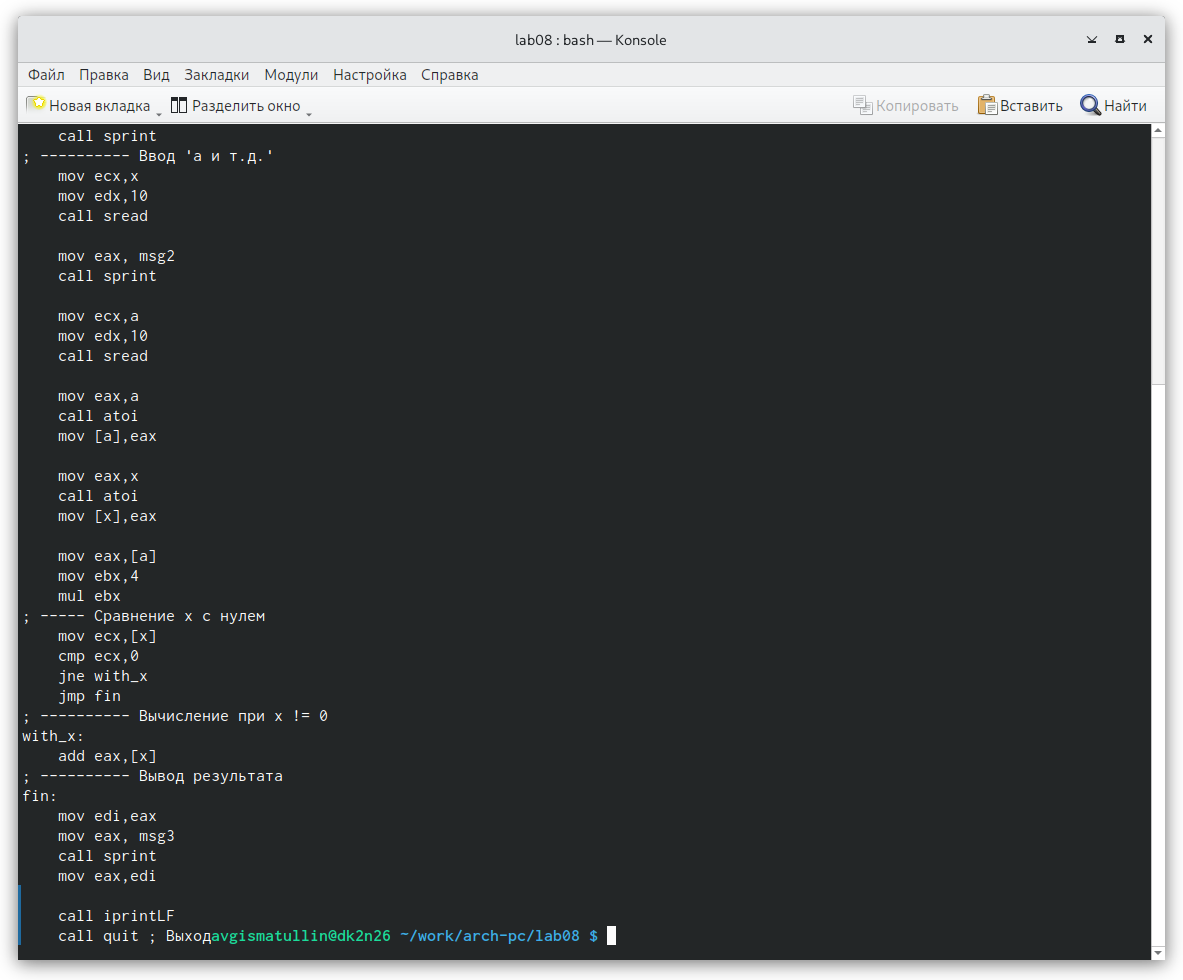


Рис. 12: Командная строка. Просмотр содержимого файла lab8-4.asm

Проверка работоспособности: (рис. 13)

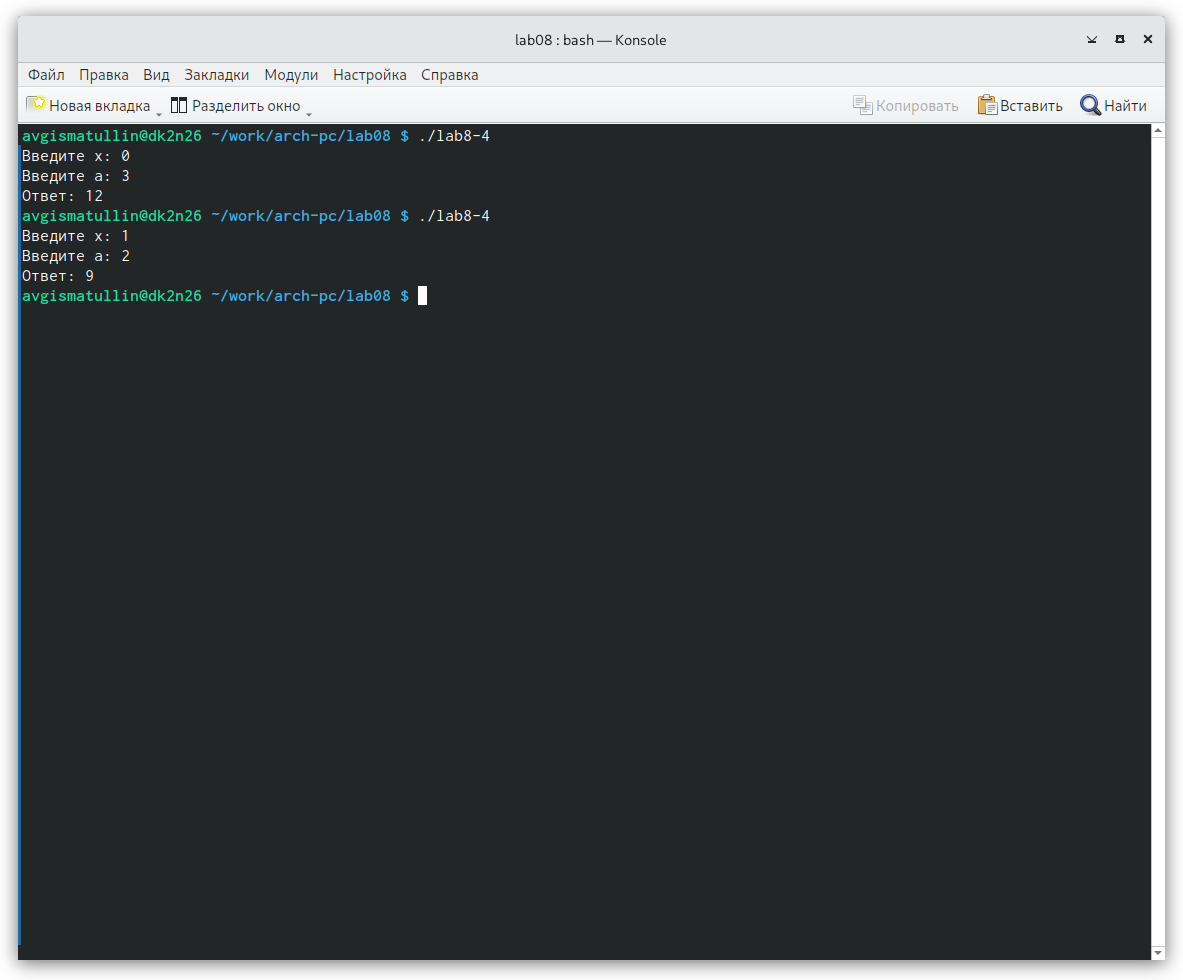


Рис. 13: Командная строка. Выполнение программы lab8-4.asm

# 6 Выводы

В ходе работы я изучил команды условного и безусловного переходов, приобрел навы- ки написания программ с использованием переходов, а также познакомился с назначе- нием и структурой файла листинга.

# Список литературы