Лабораторная работа номер 7

Архитектура компьютера

Титков Ярослав Максимович

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 файлы листинга
3. Задание для самостоятельной работы

# 3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Существует два типа переходов: условный и безусловный. Безусловный переход выполняется инструкцией jmp (от англ. jump – прыжок), которая включает в себя адрес перехода, куда следует передать управление: jmp <адрес\_перехода>. Адрес перехода может быть либо меткой, либо адресом области памяти, в которую предварительно помещен указатель перехода. В качестве операнда можно также использовать имя регистра, в таком случае переход будет осуществляться по адресу, хранящемуся в этом регистре.

Условный переход, в отличие от безусловного, требует проверки определенного условия. В ассемблере команды условного перехода анализируют флаги из регистра флагов для определения необходимости перехода. 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 помещены в единый регистр — регистр флагов, отражающий текущее состояние процессора.

Инструкция cmp является одной из команд, которая позволяет сравнить операнды и выставить флаги в зависимости от результата сравнения. Формат команды cmp аналогичен команде вычитания: cmp <операнд\_1>, <операнд\_2>. Команда cmp выполняет вычитание <операнд\_2> - <операнд\_1>, но результат вычитания никуда не записывается, и единственным результатом команды сравнения является формирование флагов.

Команда условного перехода имеет вид j<мнемоника перехода> label, где мнемоника перехода связана со значением анализируемых флагов или со способом формирования этих флагов. В таблице представлены команды условного перехода, которые обычно ставятся после команды сравнения cmp. В их мнемокодах указывается тот результат сравнения, при котором надо делать переход. Мнемоники, идентичные по своему действию, написаны в таблице через дробь (например, ja и jnbe). Программист выбирает, какую из них применить, чтобы получить более простой для понимания текст программы.

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. Ниже приведён фрагмент файла листинга.

# 4 Выполнение лабораторной работы

## 4.1 Реализация переходов в NASM:

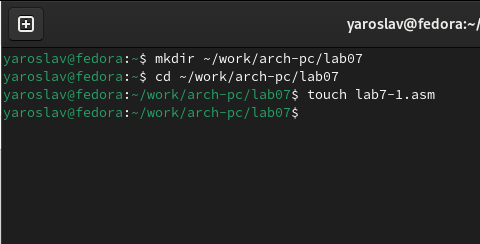


Рис. 1: Создал каталог lab07, затем перешёл в него и создал файл lab7-1.asm

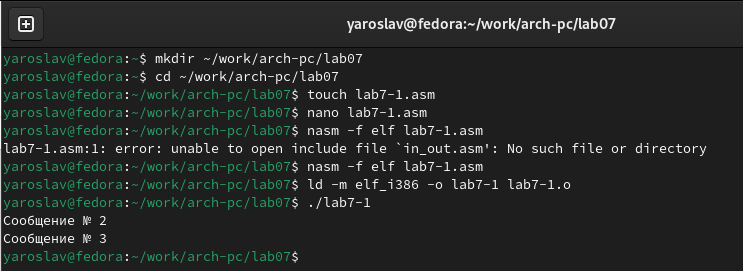


Рис. 2: Вставил нужную функцию и проверил из Листинга 7.1 и запустил

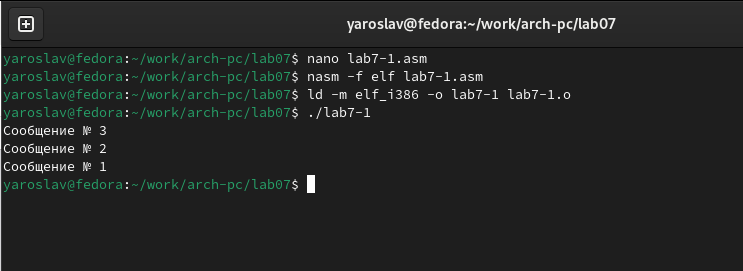


Рис. 3: Изменил программу lab7-1.asm из Листинга 7.2 и запустил её

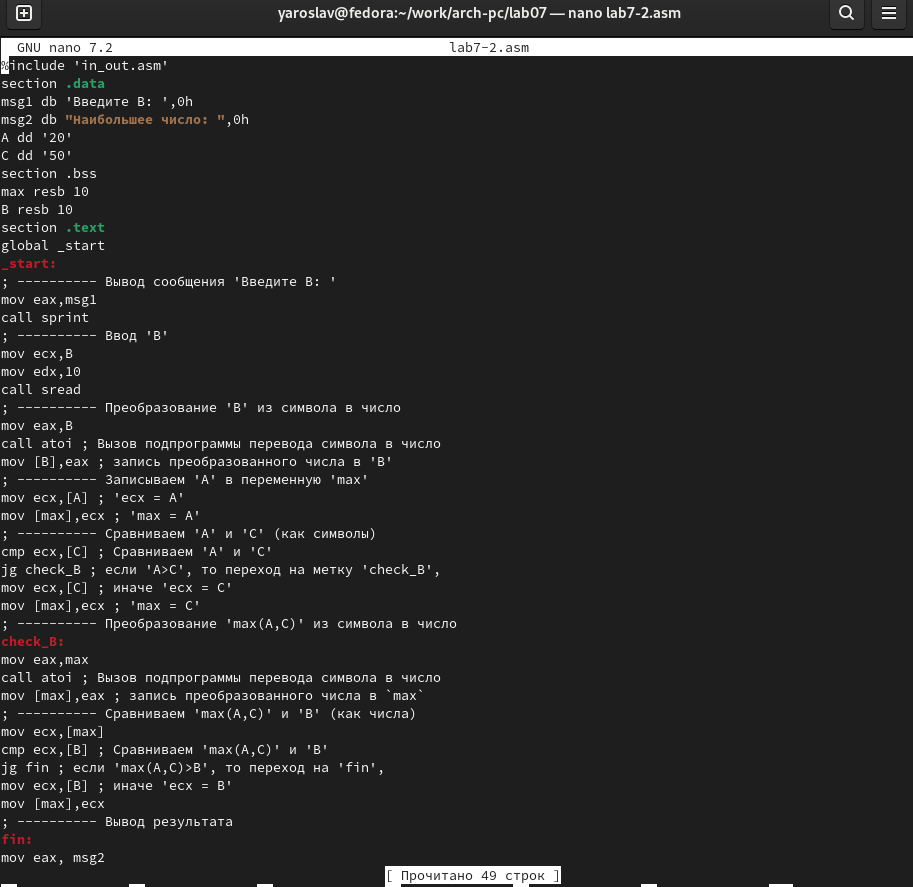


Рис. 4: Создал файл lab7-2.asm и внёс туда данные из Листинга 7.3

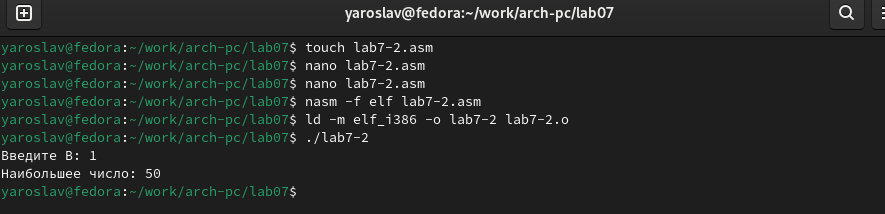


Рис. 5: Запустил программу lab7-2

## 4.2 Изучение структуры файла листинга:

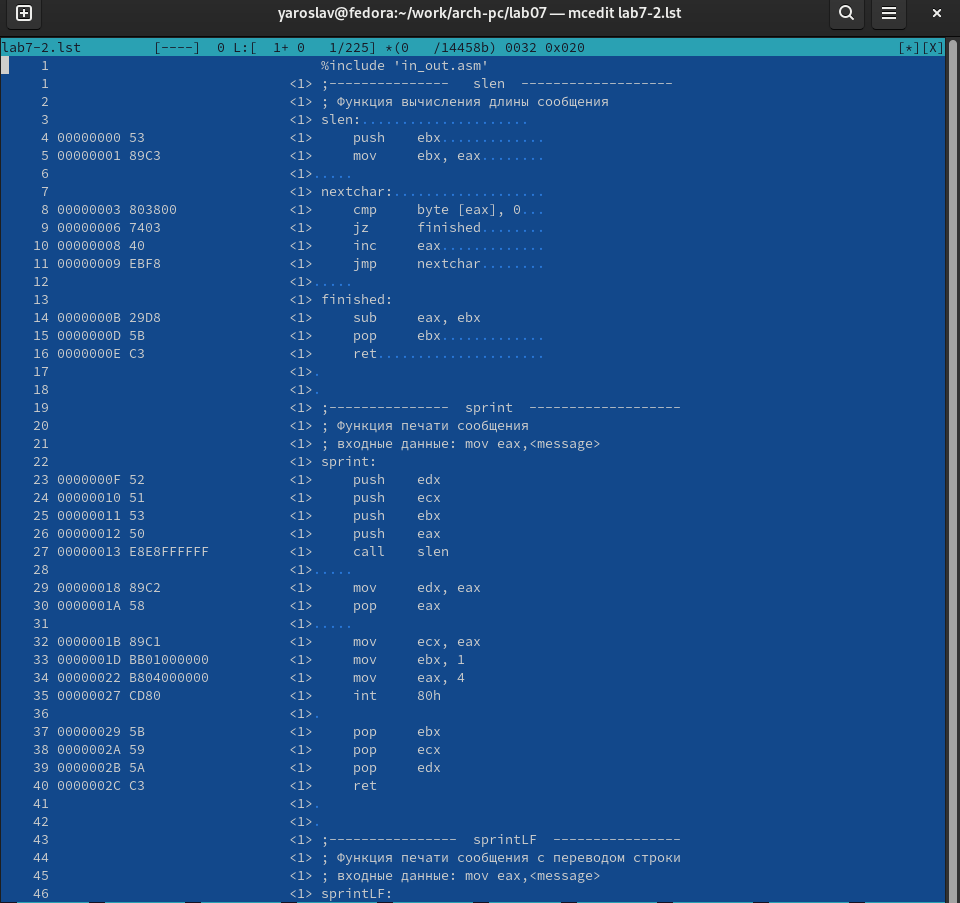


Рис. 6: Получил файл Листинга и открыл в mcedit

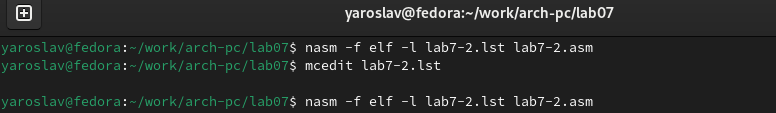


Рис. 7: Выполнил трансляцию с получением файла листинга

## 4.3 Ответы на вопросы:

1 -f elf: Указывает формат объектного файла (ELF). -l lab7-2.lst: Указывает имя файла листинга, который будет создан. lab7-2.asm: Исходный файл ассемблера, который будет ассемблирован.

2 Формат и содержимое файла листинга:

файл листинга (lab7-2.lst) содержит следующие основные элементы:  
 Номер строки: Указывает номер строки в исходном файле.  
 Адрес: Указывает адрес команды в памяти.  
 Машинный код: Отображает машинный код, соответствующий команде.  
 Исходный код: Отображает исходную строку ассемблера.

3

1. 10 00000000 B800000000 mov eax, 0:

* 10 Номер строки в исходном файле. 00000000: Адрес команды в памяти. B800000000: Машинный код команды mov eax, 0. mov eax, 0: Исходная строка ассемблера, которая загружает значение 0 в регистр eax.

1. 15 00000005 89C3 mov ebx, eax: 0: Номер строки в исходном файле. 00000000: Адрес команды в памяти. B800000000: Машинный код команды mov eax, 0. mov eax, 0: Исходная строка ассемблера, которая загружает значение 0 в регистр eax.
2. 20 00000007 CD80 int 0x80: 20: Номер строки в исходном файле. 00000007: Адрес команды в памяти. CD80: Машинный код команды int 0x80. int 0x80: Исходная строка ассемблера, которая вызывает прерывание 0x80 для выполнения системного вызова.

4

1. Что создаётся:

* Файл листинга (lab7-2.lst): Этот файл будет создан, но он будет содержать информацию об ошибке, так как трансляция не может быть успешно завершена из-за неполной инструкции. Объектный файл (lab7-2.o): Этот файл не будет создан, так как трансляция завершится с ошибкой.

1. Что добавляется: В файле листинга (lab7-2.lst) будет добавлена информация об ошибке, указывающая на неполную инструкцию

## 4.4 Задания для самостоятельной работы:

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и . Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу.

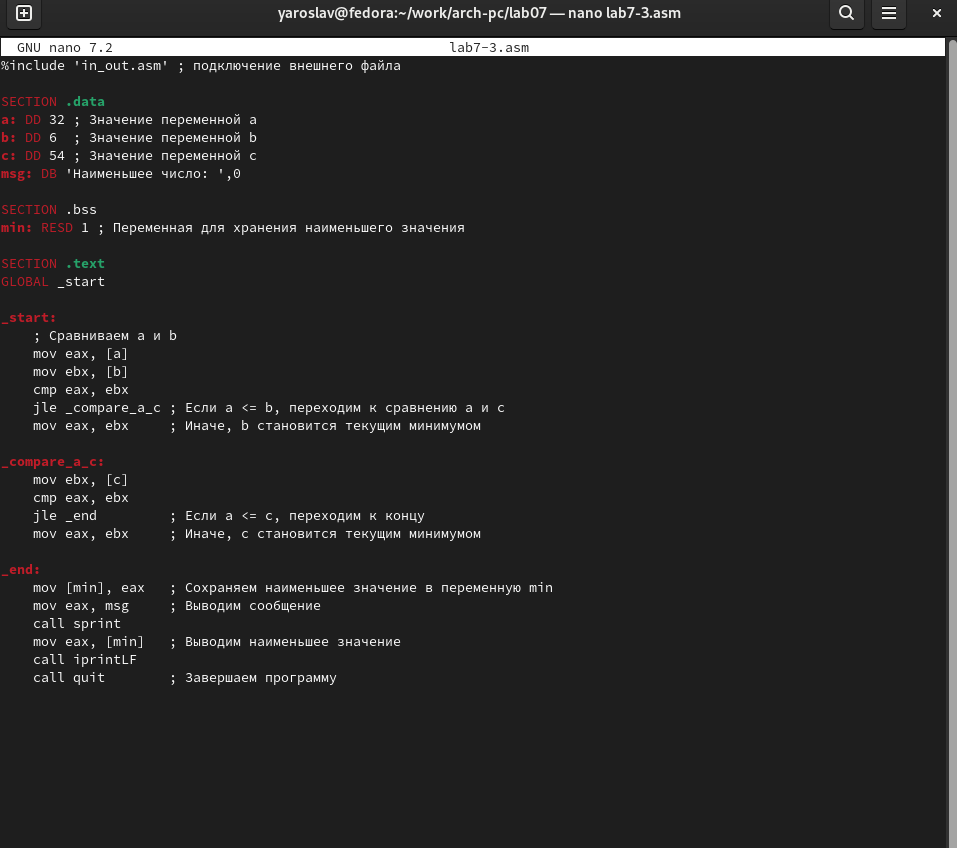


Рис. 8: Создал файл lab7-3.asm и написал программу с данными из 15 варианта

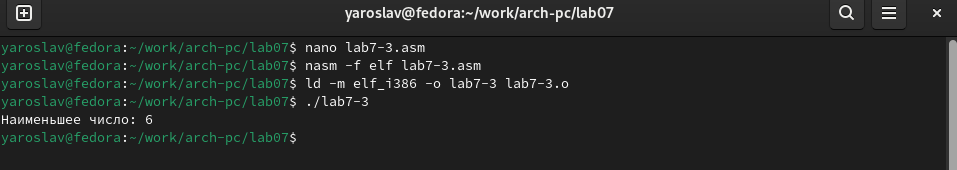


Рис. 9: Запустил программу

1. Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции f(x) и выводит результат вычислений. Вид функции f(x) выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы номер 7. Создайте исполняемый файл и проверьте его работу для значений x и a из 7.6.

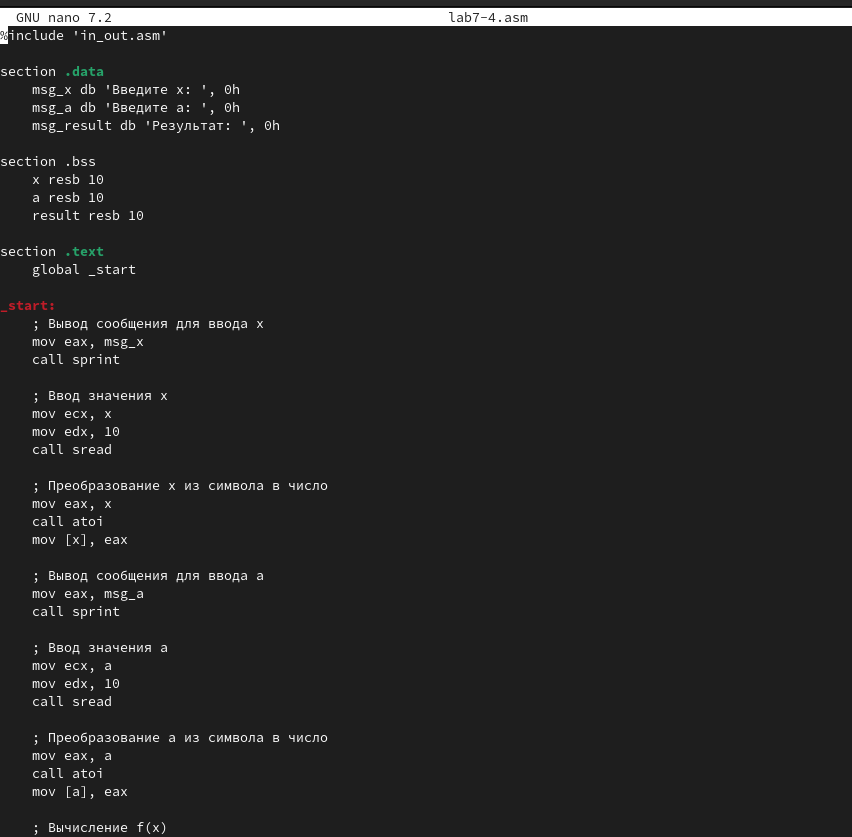


Рис. 10: Аналогично заданию 1 создал программу lab7-4.asm, которая выводит результат вычислений f(x) с переменными x и а с данными из 15 варианта

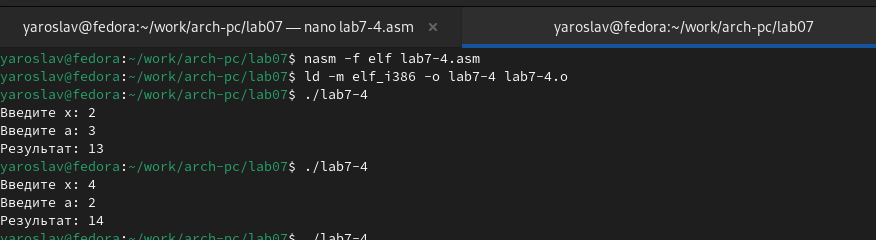


Рис. 11: Запустил, введя данные 2 и 3, 4 и 2

# 5 Выводы

В ходе выполнения лабораторной работы были изучены команды условного и безусловного переходов в ассемблере, что позволило приобрести навыки написания программ с использованием переходов. Команды переходов являются ключевыми для управления потоком выполнения программы, что позволяет реализовывать сложные алгоритмы и логические конструкции.