Лабораторная работа №4

дисциплина: Архитектура компьютера

Савенкова Алиса Евгеньевна

Содержание

# 1 Цель работы

Целью данной лабораторной работы является освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

# 2 Задание

1. Программа Hello world!
2. Транслятор NASM
3. Расширенный синтаксис командной строки NASM
4. Компоновщик LD
5. Запуск исполняемого файла
6. Задания для самостоятельной работы

# 3 Теоретическое введение

Основными функциональными элементами любой электронно-вычислительной машины (ЭВМ) являются центральный процессор, память и периферийные устройства. Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора (ЦП) входят следующие устройства: • арифметико-логическое устройство (АЛУ); • устройство управления (УУ); • регистры. Для того, чтобы писать программы на ассемблере, необходимо знать, какие регистры процессора существуют и как их можно использовать. В качестве примера приведем названия основных регистров общего назначения (именно эти регистры чаще всего используются при написании программ): • RAX, RCX, RDX, RBX, RSI, RDI — 64-битные • EAX, ECX, EDX, EBX, ESI, EDI — 32-битные • AX, CX, DX, BX, SI, DI — 16-битные • AH, AL, CH, CL, DH, DL, BH, BL — 8-битные (половинки 16-битных регистров). Другим важным узлом ЭВМ является оперативное запоминающее устройство (ОЗУ). ОЗУ — это быстродействующее энергозависимое запоминающее устройство, которое напрямую взаимодействует с узлами процессора, предназначенное для хранения программ и данных, с которыми процессор непосредственно работает в текущий момент. В состав ЭВМ также входят периферийные устройства, которые можно разделить на: • устройства внешней памяти; • устройства ввода-вывода. Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. Можно считать, что он больше любых других языков приближен к архитектуре ЭВМ и её аппаратным возможностям, что позволяет получить к ним более полный доступ, нежели в языках высокого уровня,таких как C/C++, Perl, Python и пр. Следует отметить, что процессор понимает не команды ассемблера, а последовательности из нулей и единиц — машинные коды. Наиболее распространёнными ассемблерами для архитектуры x86 являются: • для DOS/Windows: Borland Turbo Assembler (TASM), Microsoft Macro Assembler (MASM) и Watcom assembler (WASM); • для GNU/Linux: gas (GNU Assembler), использующий AT&T-синтаксис, в отличие от большинства других популярных ассемблеров, которые используют Intel-синтаксис. NASM — это открытый проект ассемблера, версии которого доступны под различные операционные системы и который позволяет получать объектные файлы для этих систем. В NASM используется Intel-синтаксис и поддерживаются инструкции x86-64. В процессе создания ассемблерной программы можно выделить четыре шага: • Набор текста. • Трансляция. • Компоновка или линковка. • Запуск программы.

# 4 Выполнение лабораторной работы

## 4.1 Программа Hello world!

Первым действием создаю каталог для работы с программами на языке ассемблера NASM (рис. 1).

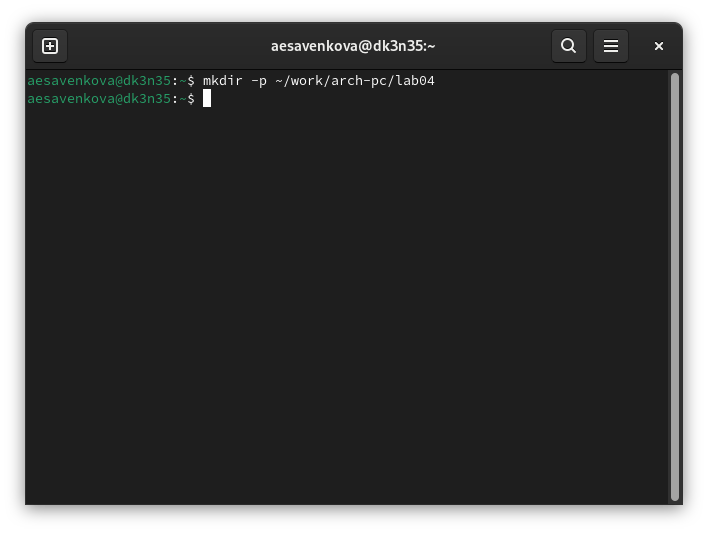


Рис. 1: Создание каталога

Далее перехожу в созданный каталог (рис. 2).

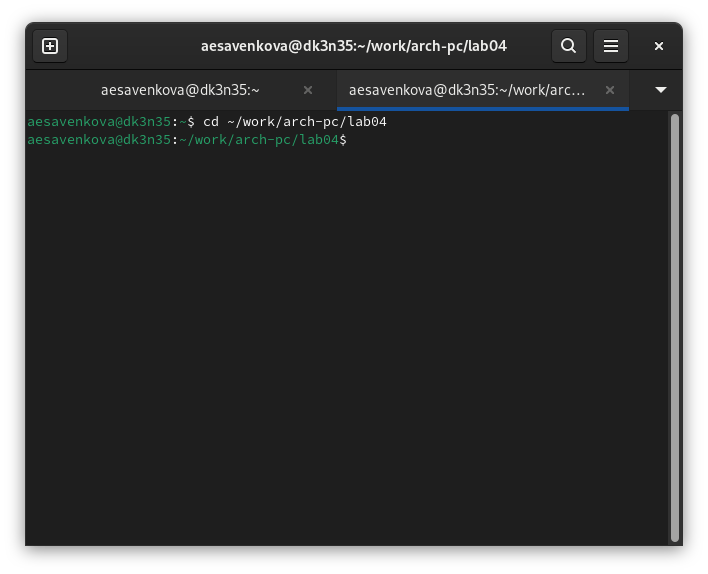


Рис. 2: Каталог для работы с программами на языке ассемблера NASM

В этом каталоге создаю текстовый файл с именем hello.asm (рис. 3).

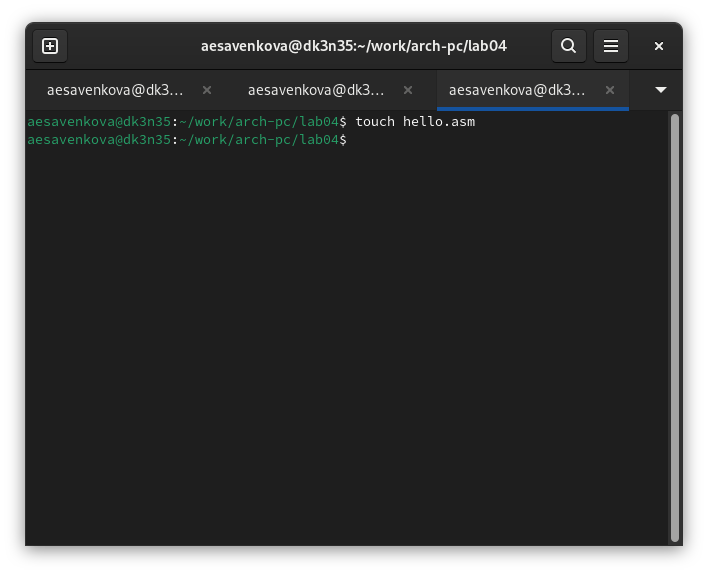


Рис. 3: Создание файла hello.asm

Открываю этот файл с помощью текстового редактора gedit (рис. 4 - 5).

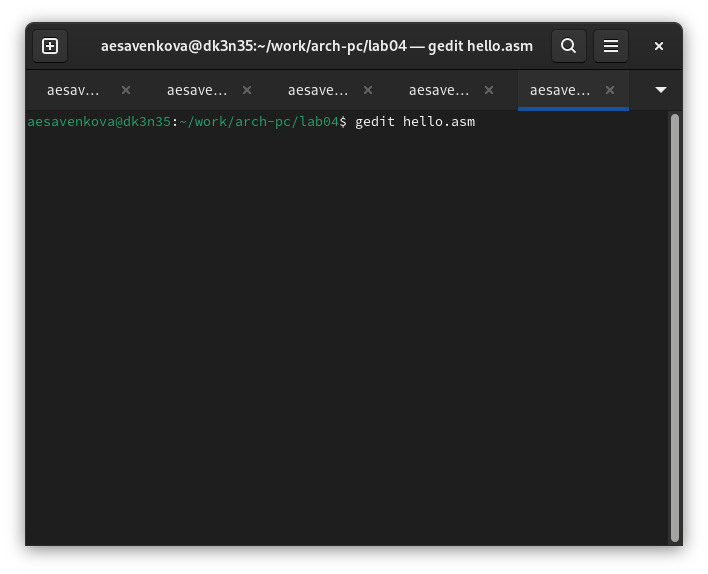


Рис. 4: Открытие файла hello.asm с помощью gedit

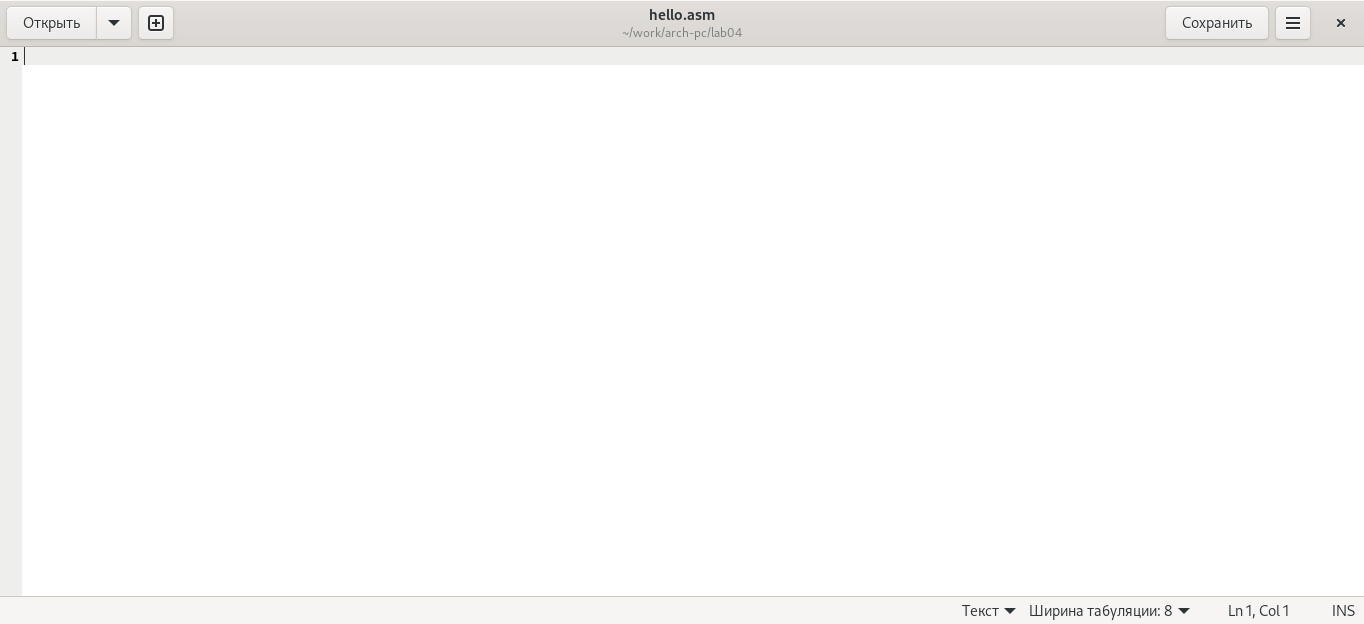


Рис. 5: Файл hello.asm

В открывшемся файле ввожу предложенный текст (рис. 6).

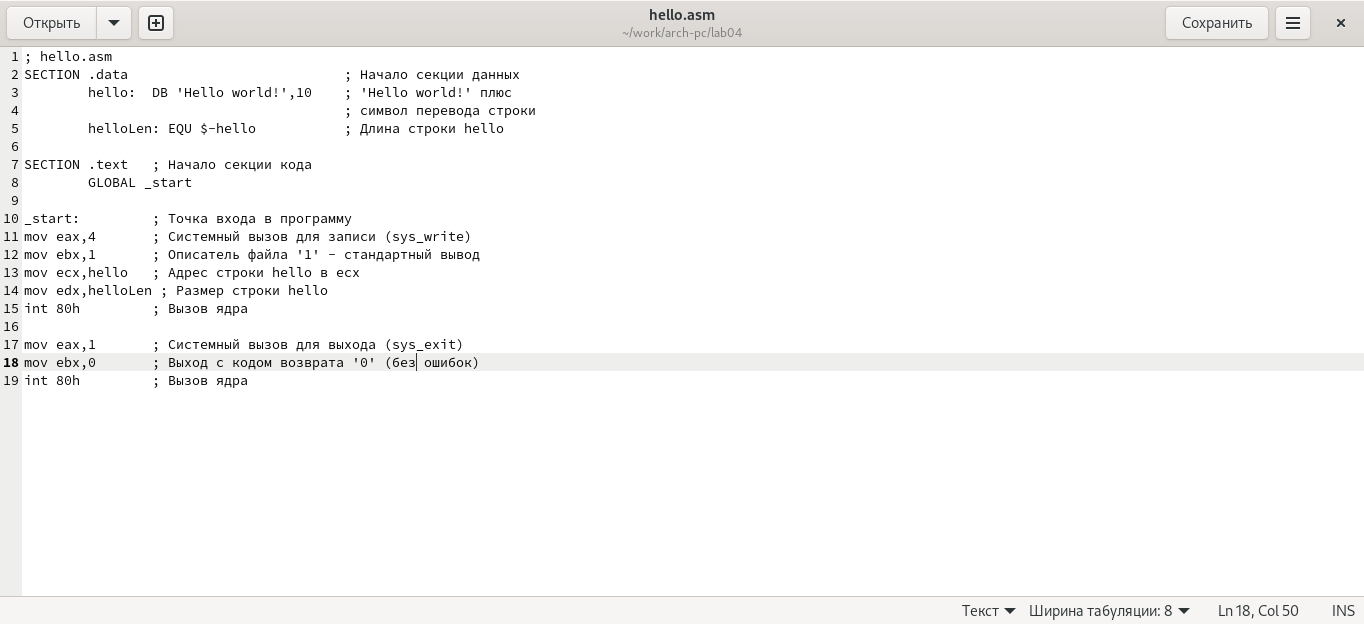


Рис. 6: Ввод текста

## 4.2 Транслятор NASM

Lля компиляции приведённого выше текста программы «Hello World» ввожу команду nasm -f elf hello.asm (ключ -f указывает транслятору, что требуется создать бинарные файлы в формате ELF) (рис. 7).

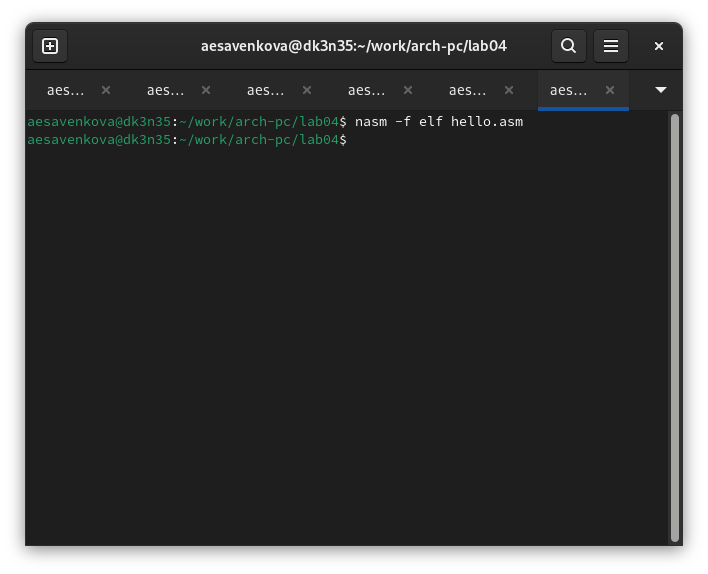


Рис. 7: Компиляция текста

С помощью команды ls проверяю, что объектный файл под именем hello.o был создан (рис. 8).

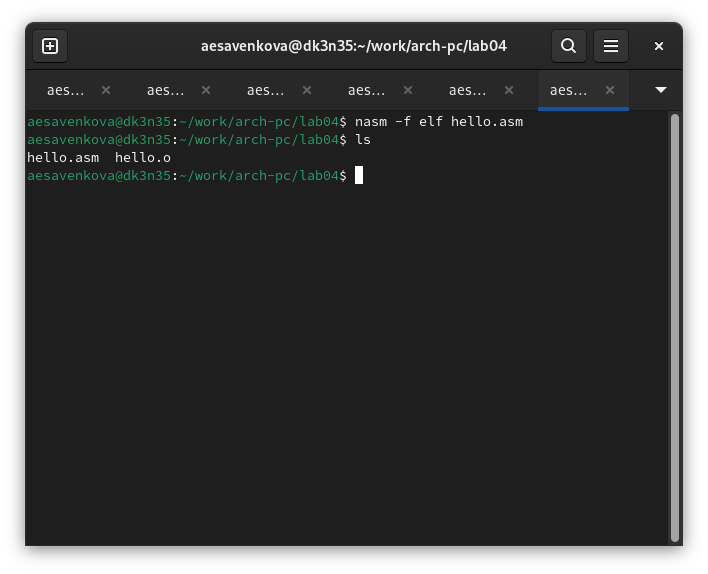


Рис. 8: Наличие файла hello.o в каталоге

## 4.3 Расширенный синтаксис командной строки NASM

Далее для того, чтобы скомпилировать исходный файл hello.asm в obj.o (опция -o позволяет задать имя объектного файла, в данном случае obj.o), ввожу команду nasm -o obj.o -f elf -g -l list.lst hello.asm, при этом формат выходного файла будет elf, и в него будут включены символы для отладки (опция -g), кроме того, будет создан файл листинга list.lst (опция -l) (рис. 9).

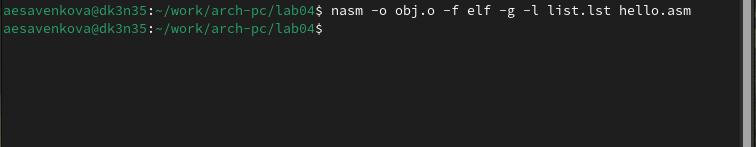


Рис. 9: Компиляция в obj.o

Затем с помощью команды ls проверяю, что файлы были созданы (рис. 10).

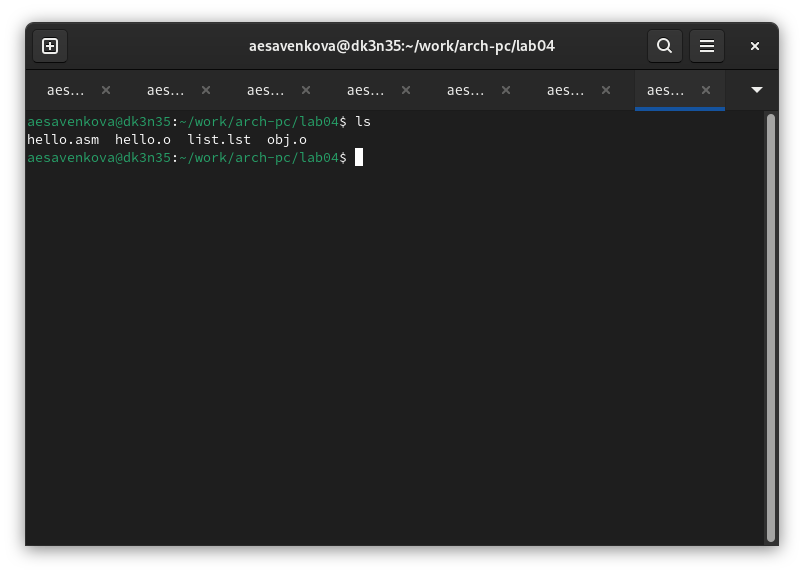


Рис. 10: Наличие файлов в каталоге

## 4.4 Компоновщик LD

Чтобы передать объектный файл на обработку компоновщику ввожу команду ld -m elf\_i386 hello.o -o hello (рис. 11).

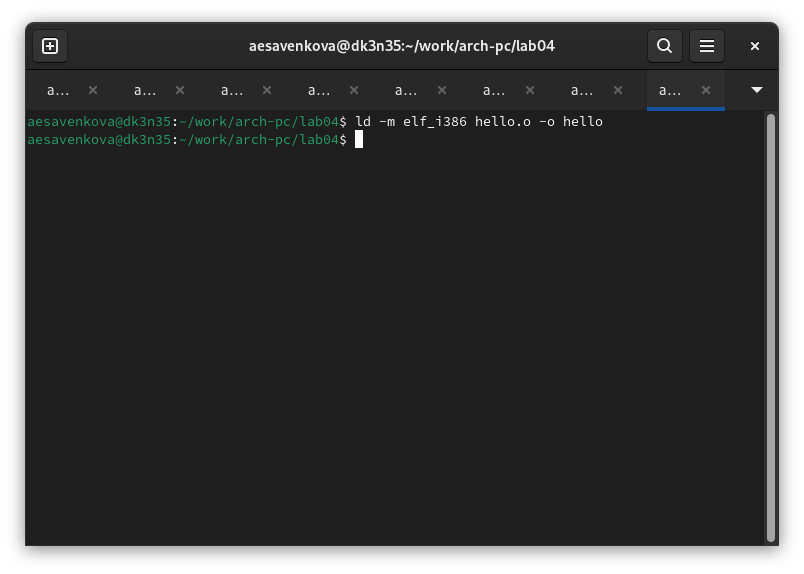


Рис. 11: Передача файлов компоновщику

С помощью команды ls проверяю, что исполняемый файл hello был создан (рис. 12).

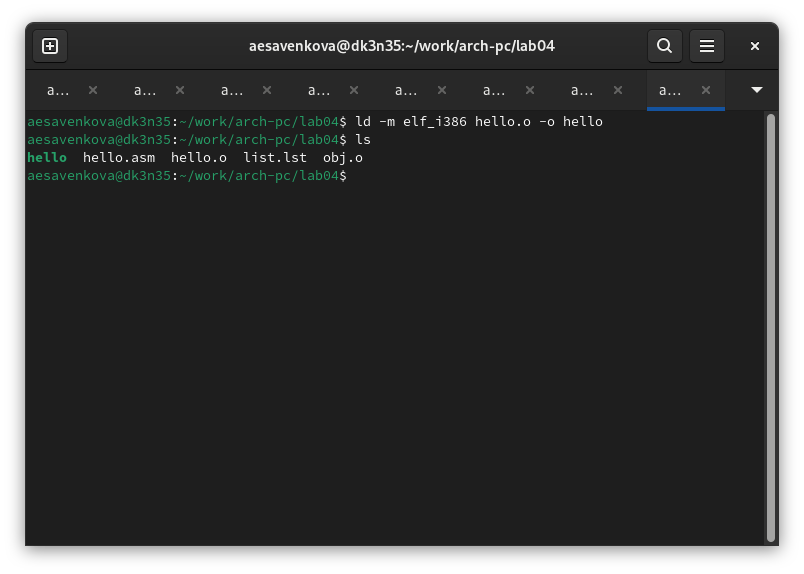


Рис. 12: Наличие исполняемого файла в каталоге

Ввожу следующую команду ld -m elf\_i386 obj.o -o main (рис. 13).

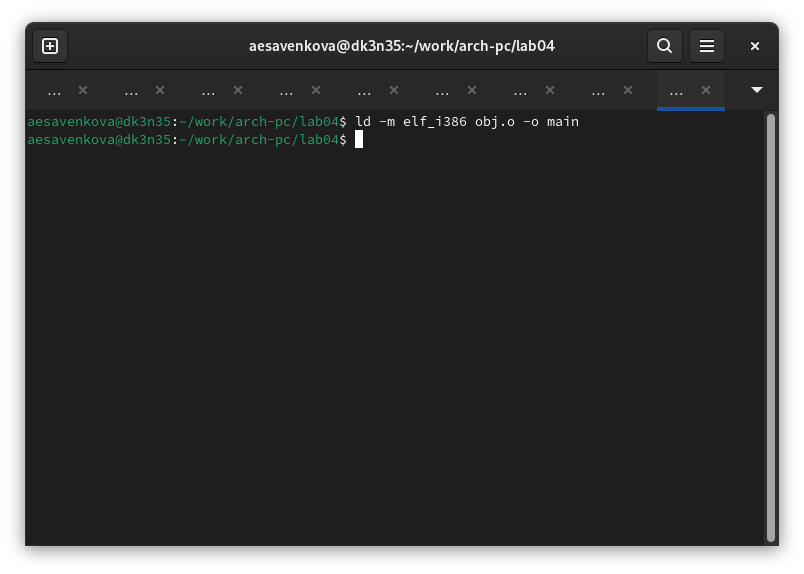


Рис. 13: Передача файлов на обработку компоновщику

После ключа -о было задано значение main, поэтому у файла будет имя main (рис. 14).

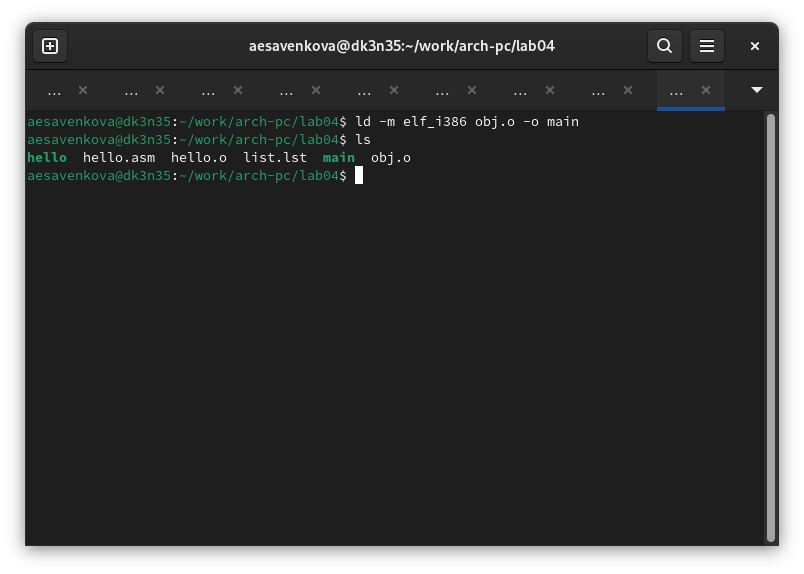


Рис. 14: Передача файлов на обработку компоновщику со значением main

Чтобы увидеть формат командной строки LD, набираю ld –help (рис. 15).

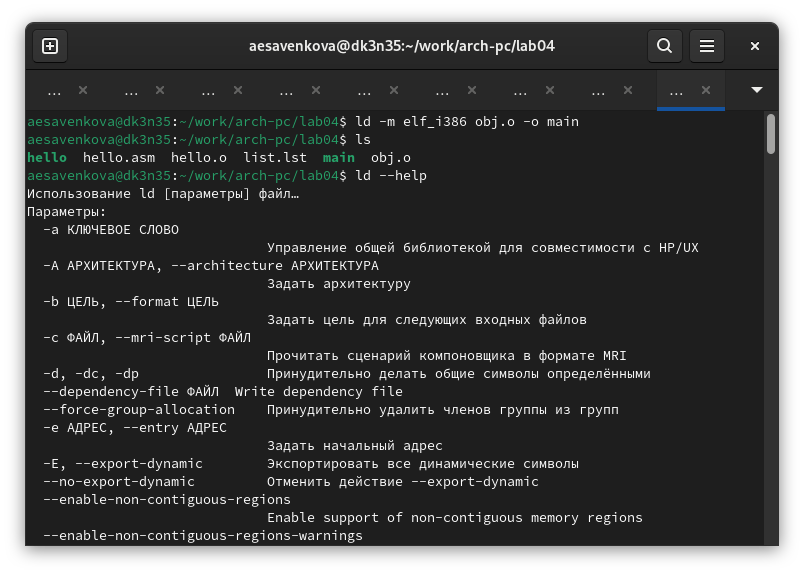


Рис. 15: Командная строка LD

Запускаю на выполнение созданный исполняемый файл, введя ./hello (рис. 16).

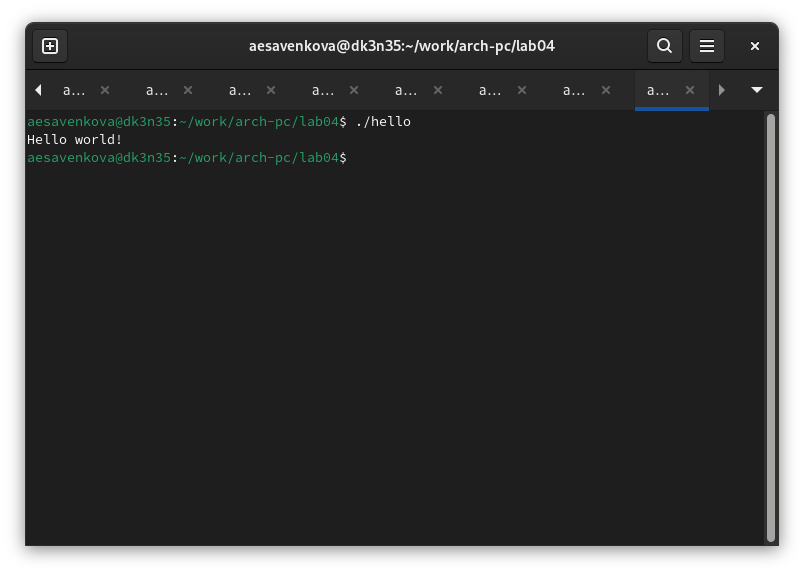


Рис. 16: Запуск исполняемого файла

## 4.5 Задание для самостоятельной работы

С помощью команды cp создаю копию файла hello.asm с именем lab4.asm в каталоге ~/work/arch-pc/lab04 (рис. 17).

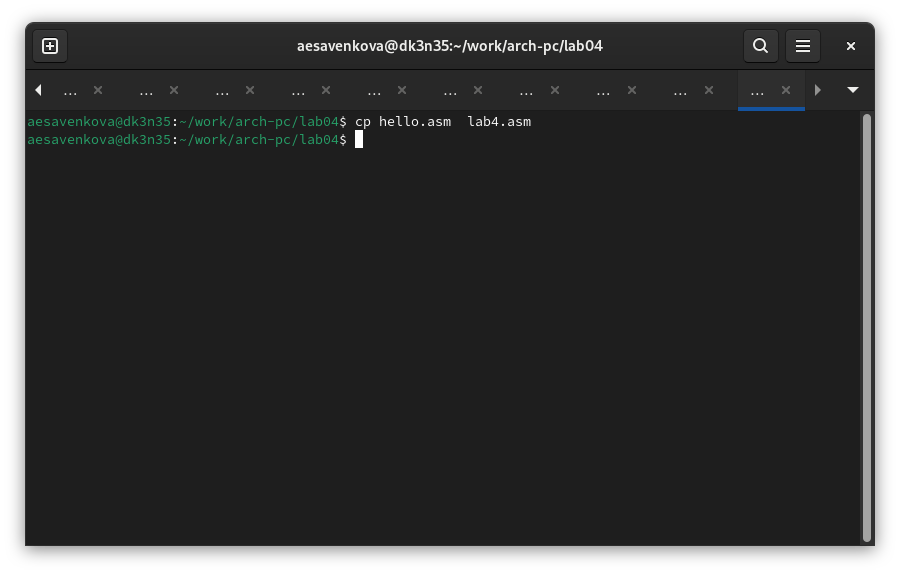


Рис. 17: Создание копии файла

Проверяю наличие файла в каталоге (рис. 18).

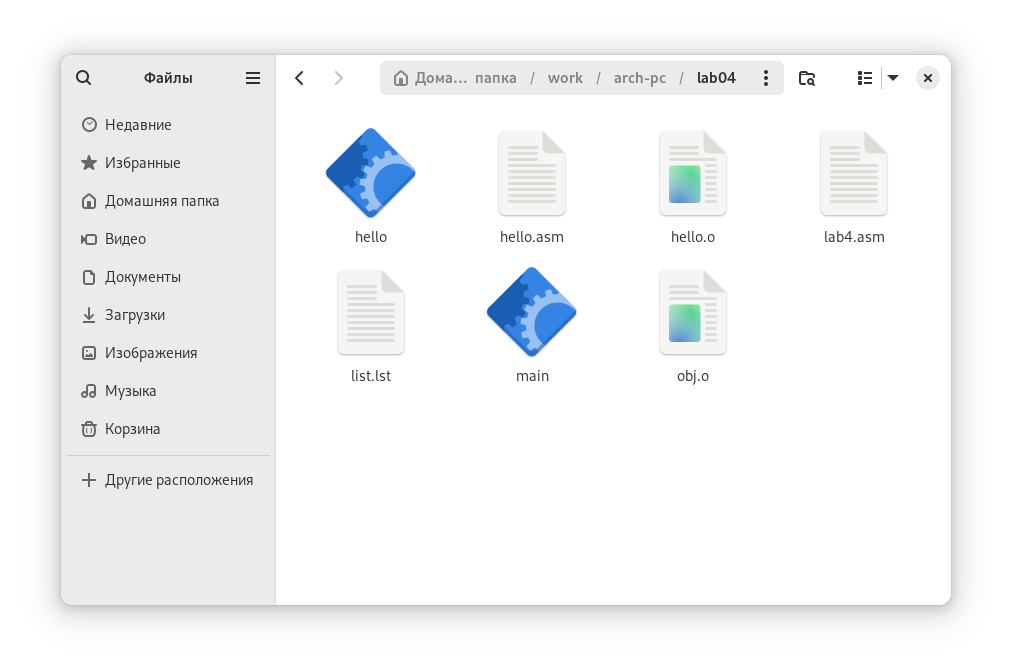


Рис. 18: Наличие файла lab4.asm в каталоге

Открываю файл lab4.asm с помощью текстового редактора gedit (рис. 19).

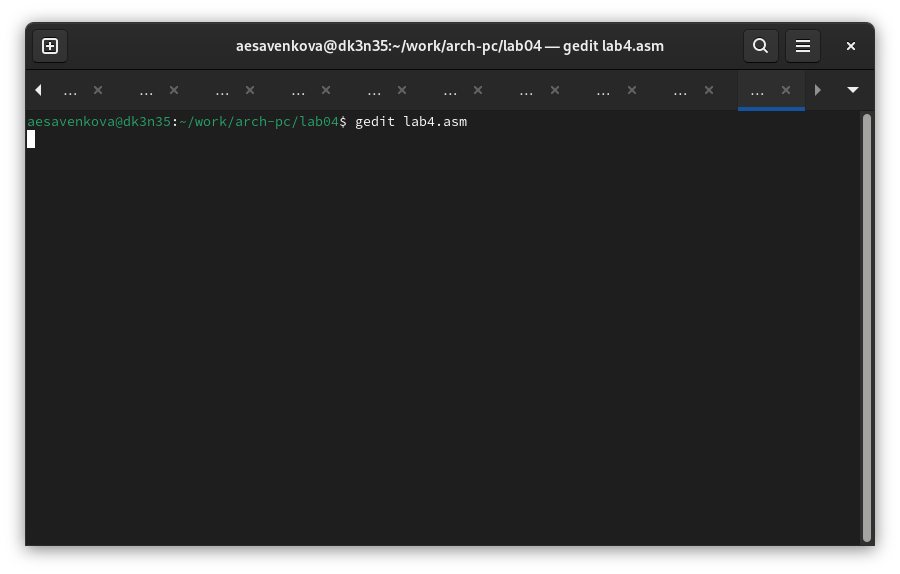


Рис. 19: Открытие файла lab4.asm с помощью gedit

Следующим шагом вношу изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с моими фамилией и именем (рис. 20).

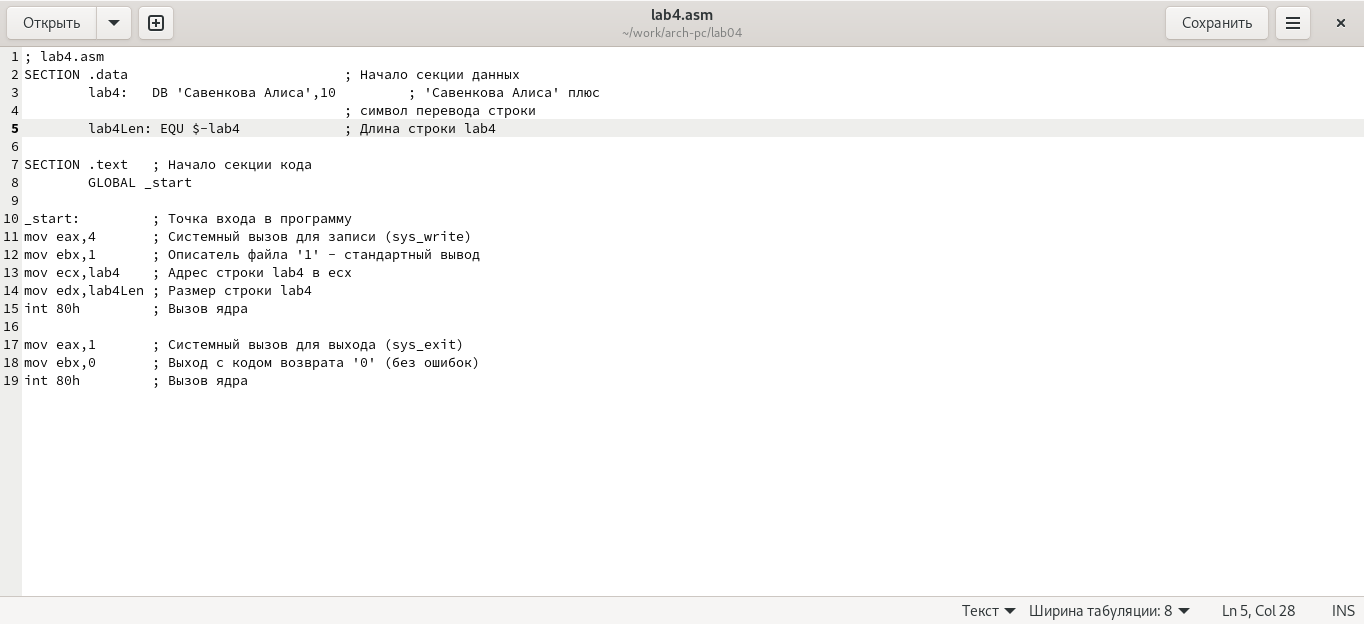


Рис. 20: Внесение изменений в lab4.asm

Затем с помощью команды nasm -f elf lab4.asm компилирую текст программы из файла lab4.asm в объектный код, который запишется в файл lab4.o, и проверяю наличие файла с помощью ls (рис. 21).

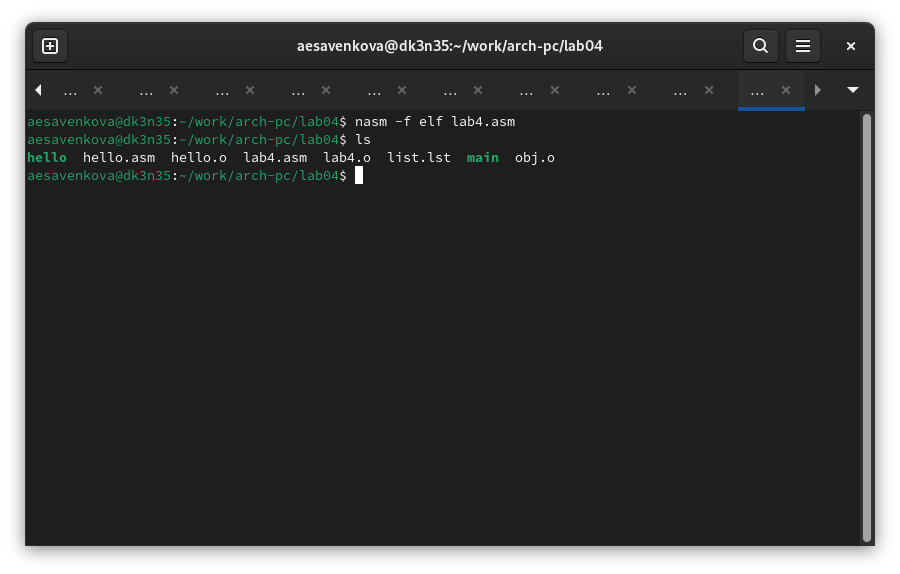


Рис. 21: Внесение изменений в lab4.asm

Далее ввожу команду nasm -o obj.o -f elf -g -l list.lst lab4.asm, чтобы скомпилировать исходный файл lab4.asm в obj.o (рис. 22).

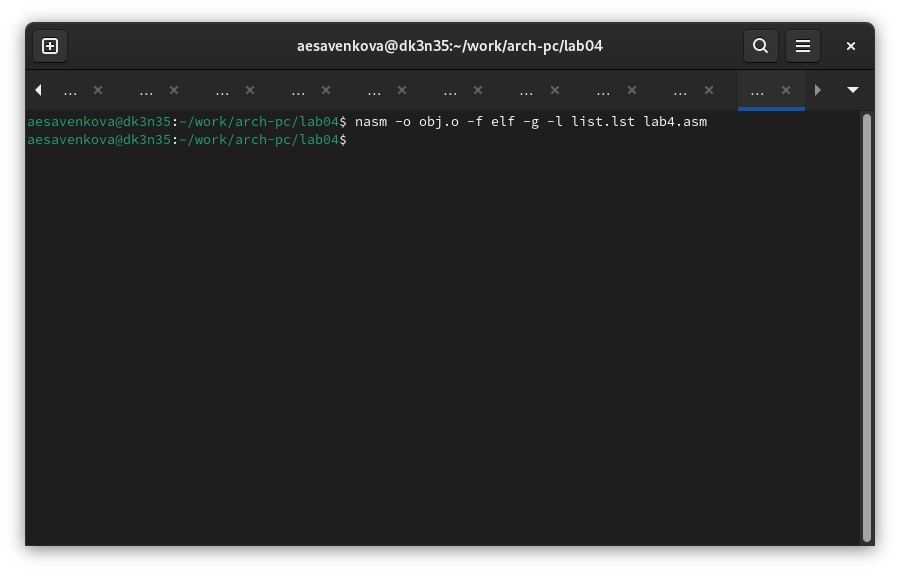


Рис. 22: Компиляция lab4.asm

Заключительным шагом передаю объектный файл на обработку компоновщику (рис. 23).

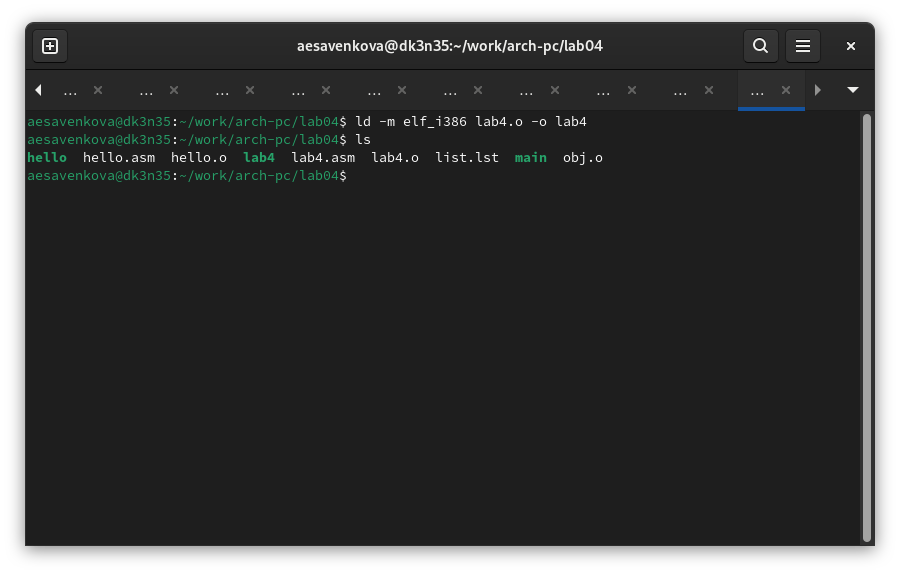


Рис. 23: Передача lab4.asm на обработку компоновщику

Запускаю исполняемый файл и удостоверяюсь, что на экран действительно выводятся мои фамилия и имя (рис. 24).

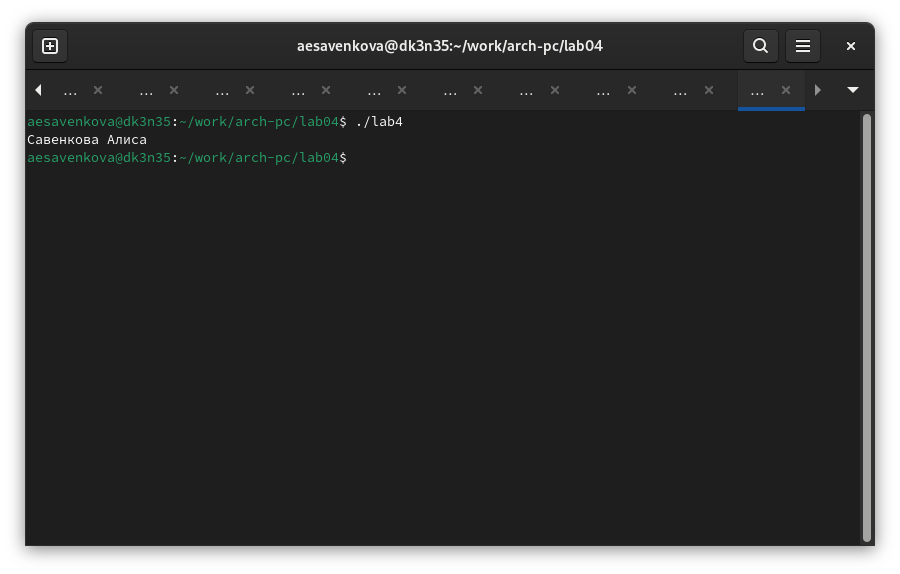


Рис. 24: Запуск исполняемого файла

Копирую файлы hello.asm и lab4.asm в свой локальный репозиторий в каталог ~/work/study/2024-2025/“Архитектура компьютера”/arch-pc/labs/lab04/.Загружаю файлы на Github (рис. 25).

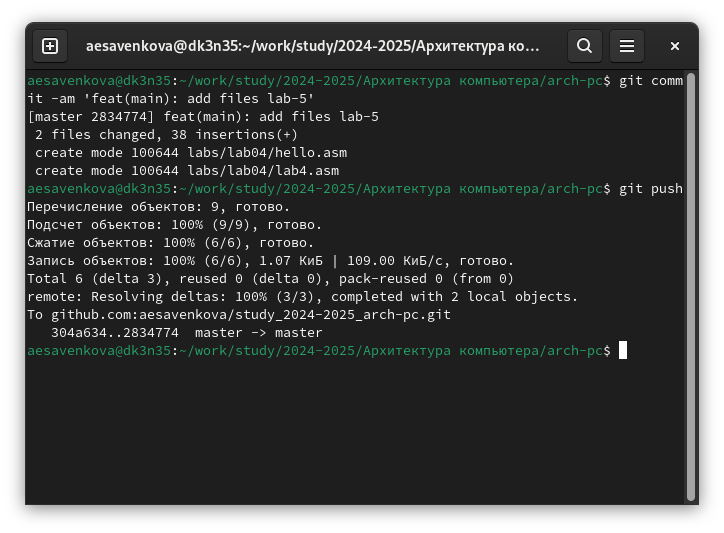


Рис. 25: Загрузка файлов на Github

# 5 Выводы

В ходе данной лабораторной работы я освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM.

# Список литературы

1. https://esystem.rudn.ru/mod/resource/view.php?id=1030832