Отчет по лабораторной работе №4

Дисциплина: Архитектура ЭВМ

Дагделен Зейнап Реджеповна

Содержание

# 1 Цель работы

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

# 2 Задание

1. Создание программы Hello world!
2. Работа с транслятором NASM
3. Работа с расширенным синтаксисом командной строки NASM
4. Работа с компоновщиком LD
5. Запуск исполняемого файла
6. Выполнение заданий для самостоятельной работы.

# 3 Теоретическое введение

## 3.1 Основные принципы работы компьютера

Основными функциональными элементами любой ЭВМ являются *центральный процессор, память и периферийные устройства*.  
Взаимодействие этих устройств осуществляется через общую шину, к которой они подключены. 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)* — выполняет логические и арифметические действия, необходимые для обработки информации, хранящейся в памяти; - *устройство управления (УУ)* — обеспечивает управление и контроль всех устройств компьютера; - *регистры* — сверхбыстрая оперативная память небольшого объёма, входящая в состав процессора, для временного хранения промежуточных результатов выполнения инструкций; регистры процессора делятся на: - регистры общего назначения - специальные регистры. Большинство команд в программах написанных на ассемблере используют регистры в качестве операндов. Практически все команды представляют собой преобразование данных хранящихся в регистрах процессора, это например пересылка данных между регистрами или между регистрами и памятью, преобразование (арифметические или логические операции) данных хранящихся в регистрах.

Доступ к регистрам осуществляется не по адресам, а по именам. Каждый регистр процессора архитектуры x86 имеет свое название, состоящее из 2 или 3 букв латинского алфавита.

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. ОЗУ состоит из одинаковых пронумерованных ячеек памяти. Номер ячейки памяти — это адрес хранящихся в ней данных.   В состав ЭВМ также входят периферийные устройства, которые можно разделить на: - устройства внешней памяти, которые предназначены для долговременного хранения больших объёмов данных (жёсткие диски, твердотельные накопители, магнитныеленты); - устройства ввода-вывода, которые обеспечивают взаимодействие ЦП с внешней средой.

В основе вычислительного процесса ЭВМ лежит *принцип программного управления*.

В коде машинной команды можно выделить две части: операционную и адресную. В операционной части хранится код команды, которую необходимо выполнить. В адресной части хранятся данные или адреса данных, которые участвуют в выполнении данной операции. При выполнении каждой команды процессор выполняет определённую последовательность стандартных действий, которая называется командным циклом процессора: 1. формирование адреса в памяти очередной команды; 2. считывание кода команды из памяти и её дешифрация; 3. выполнение команды; 4. переход к следующей команде. В зависимости от команды при её выполнении могут проходить не все этапы.

## 3.2 Ассемблер и язык ассемблера

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. В отличие от языков высокого уровня ассемблерная программа содержит только тот код, который ввёл программист. Таким образом язык ассемблера — это язык, с помощью которого понятным для человека образом пишутся команды для процессора. Следует отметить, что процессор понимает не команды ассемблера, а последовательности из нулей и единиц — машинные коды. Преобразование или трансляция команд с языка ассемблера в исполняемый машинный код осуществляется специальной программой транслятором — *Ассемблер*.  
Программы, написанные на языке ассемблера, не уступают в качестве и скорости программам, написанным на машинном языке, так как транслятор просто переводит мнемонические обозначения команд в последовательности бит (нулей и единиц). Используемые мнемоники обычно одинаковы для всех процессоров одной архитектуры или семейства архитектур. Таким образом для каждой архитектуры существует свой ассемблер и, соответственно, свой язык ассемблера.

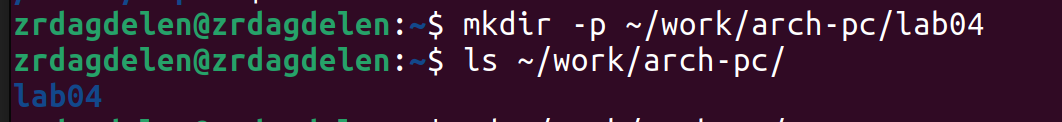
## 3.3 Процесс создания и обработки программы на языке ассемблера

В процессе создания ассемблерной программы можно выделить четыре шага: - Набор текста программы в текстовом редакторе и сохранение её в отдельном файле. - Трансляция — преобразование с помощью транслятора, например nasm, текста программы в машинный код, называемый объектным. - Компоновка или линковка — этап обработки объектного кода компоновщиком (ld), который принимает на вход объектные файлы и собирает по ним исполняемый файл. - Запуск программы. Из-за специфики программирования, а также по традиции для создания программ на языке ассемблера обычно пользуются утилитами командной строки.

# 4 Выполнение лабораторной работы

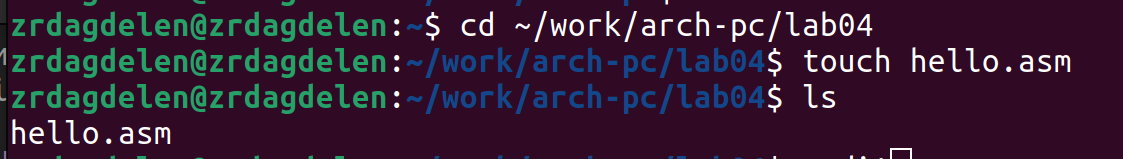
## 4.1 Программа Hello world!

Создаю каталог для работы с программами на языке ассемблера NASM с помощью команды mkdir и проверяю(рис.[??]).



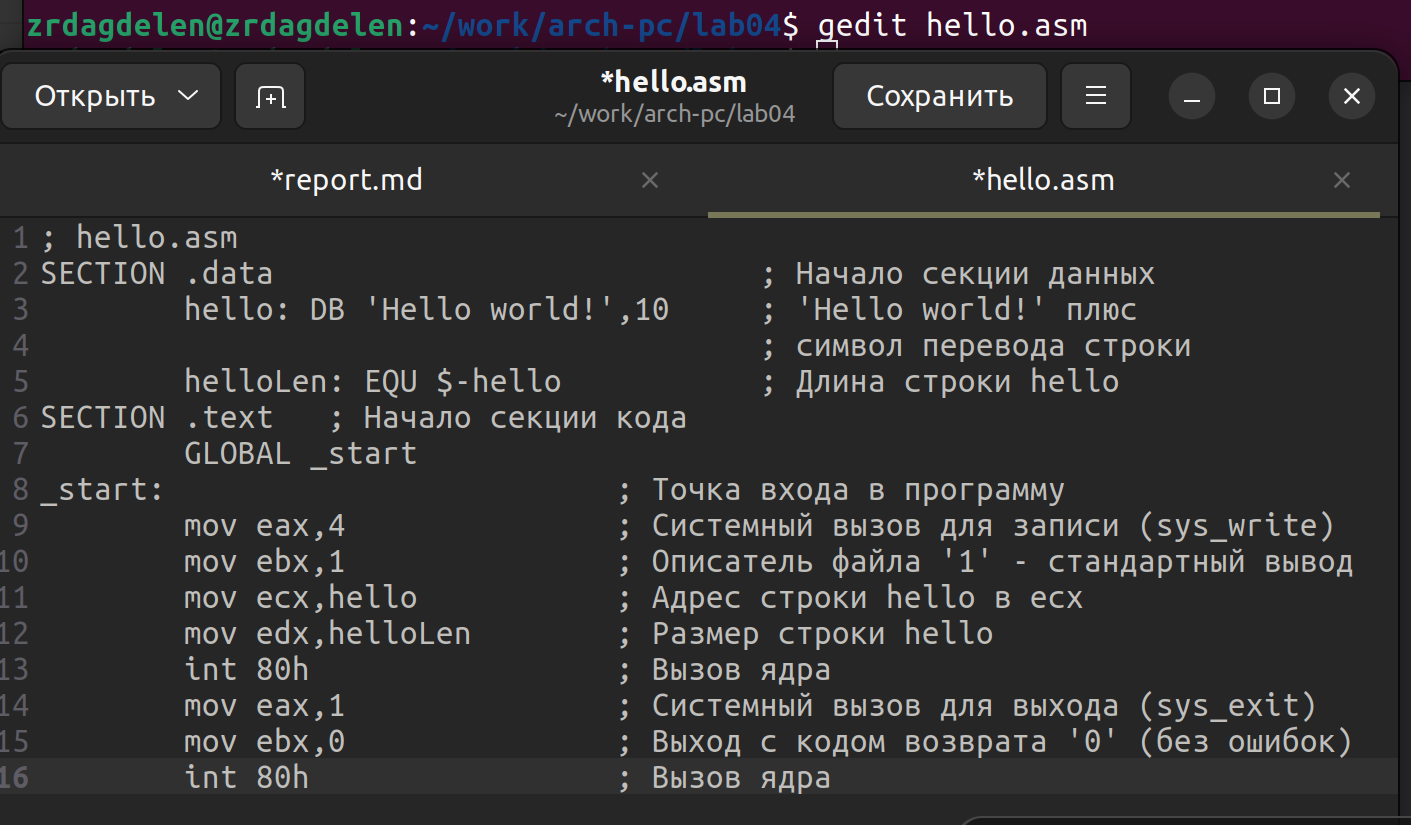
Создание каталога.

Перехожу в созданный каталог и с помощью touch создаю текстовый файл с именем hello.asm, проверяю (рис.[??]).



Создание hello.asm

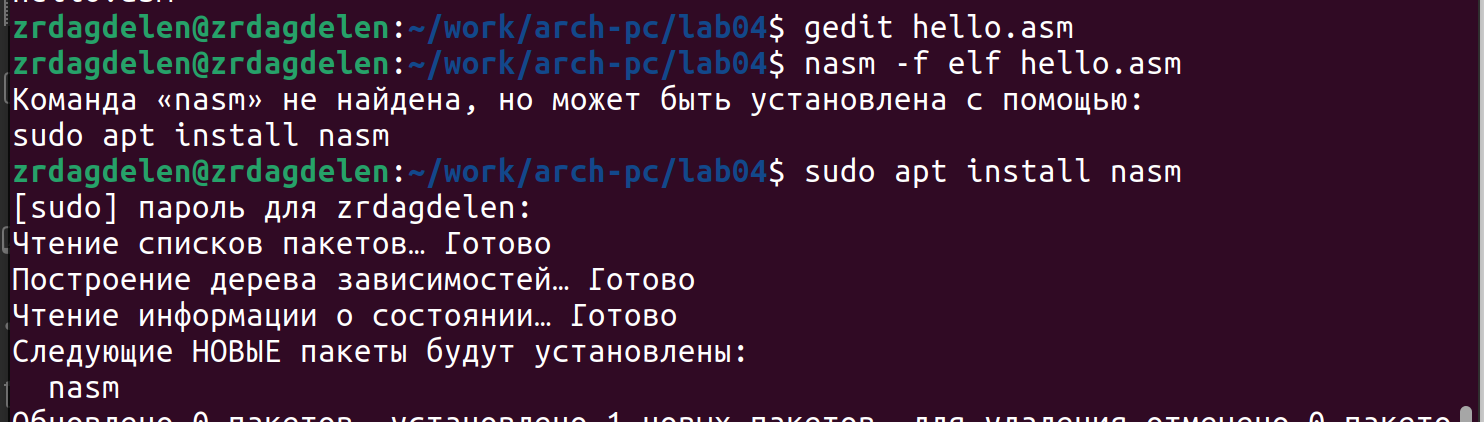
Открываю этот файл с помощью текстового редактора gedit и ввожу в него необходимый текст (рис.[??]).



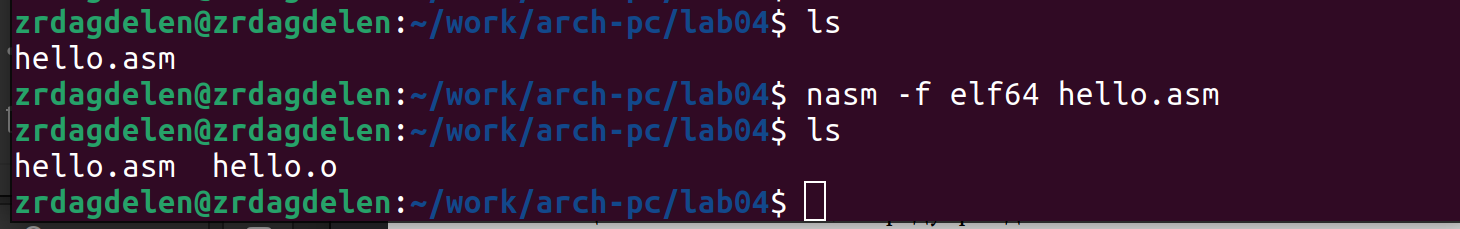
Открытие файла hello.asm

## 4.2 Транслятор NASM

NASM превращает текст программы в объектный код. Например, для компиляции приведённого выше текста программы «Hello World» необходимо написать так (рис. [??]), скачав команду nasm(рис.[??]).

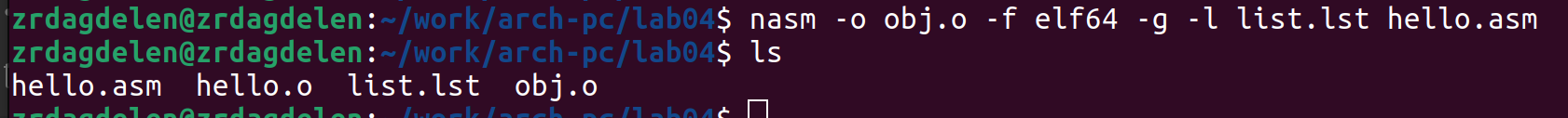


Скачивание программы nasm



Компилляция текста программы (у меня 64-битная версия Lunix)

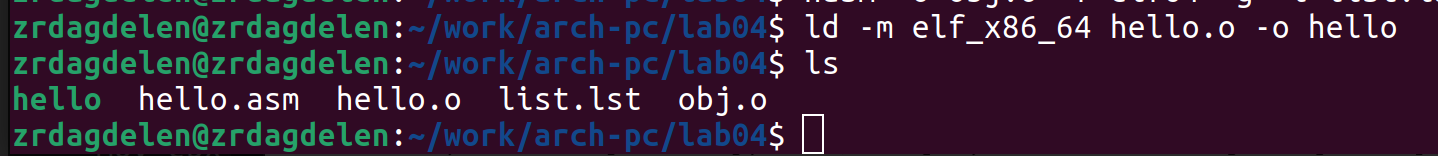
Так как текст программы был набран без ошибок, транслятор преобразовал текст программы из файла hello.asm в объектный код, который записался в файл hello.o. Таким образом, имена всех файлов получаются из имени входного файла и расширения по умолчанию. ## Расширенный синтаксис командной строки NASM Скомпилирую исходный файл hello.asm в obj.o (опция -o позволяет задать имя объектного файла, в данном случае obj.o), при этом формат выходного файла будет elf64, и в него будут включены символы для отладки (опция -g), кроме того, будет создан файл листинга list.lst (опция -l); с помощью команды ls проверяю, что файлы были созданы (рис.[??]).



Компиляция исходного файла в obj.o

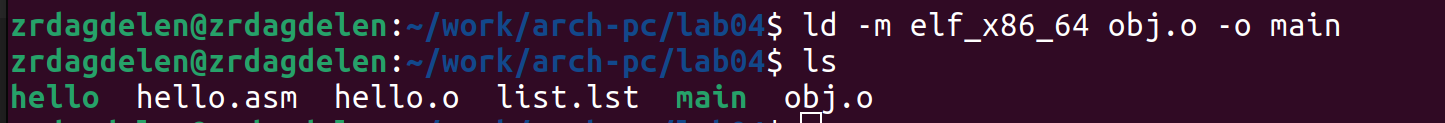
## 4.3 Компоновщик LD

Чтобы получить исполняемую программу, объектный файл передаю на обработку компоновщику (рис.[??]). Ключ -o с последующим значением задаёт в данном случае имя создаваемого исполняемого файла. Проверяю с помощью ls.



Передача файла на обработку компоновщику

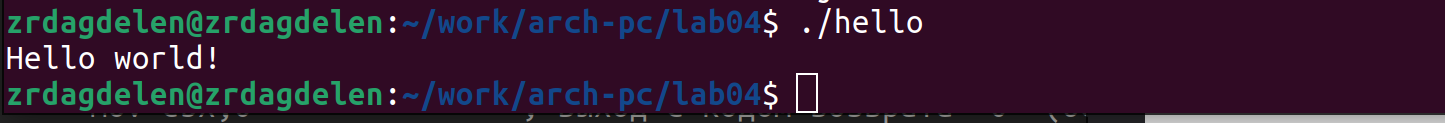
Выполняю следующую команду (рис. [??]). Исполняемый файл будет иметь имя main, т.к. после ключа -о было задано значение main. Объектный файл, из которого собран этот исполняемый файл, имеет имя obj.o.



Передача файла на обработку компоновщику

## 4.4 Запуск исполняемого файла

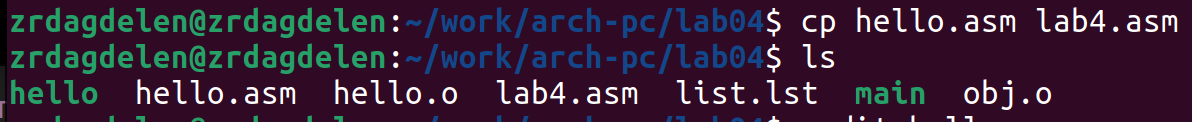
Запускаю на выполнение созданный исполняемый файл hello (рис. [??]).



Запуск исполняемого файла

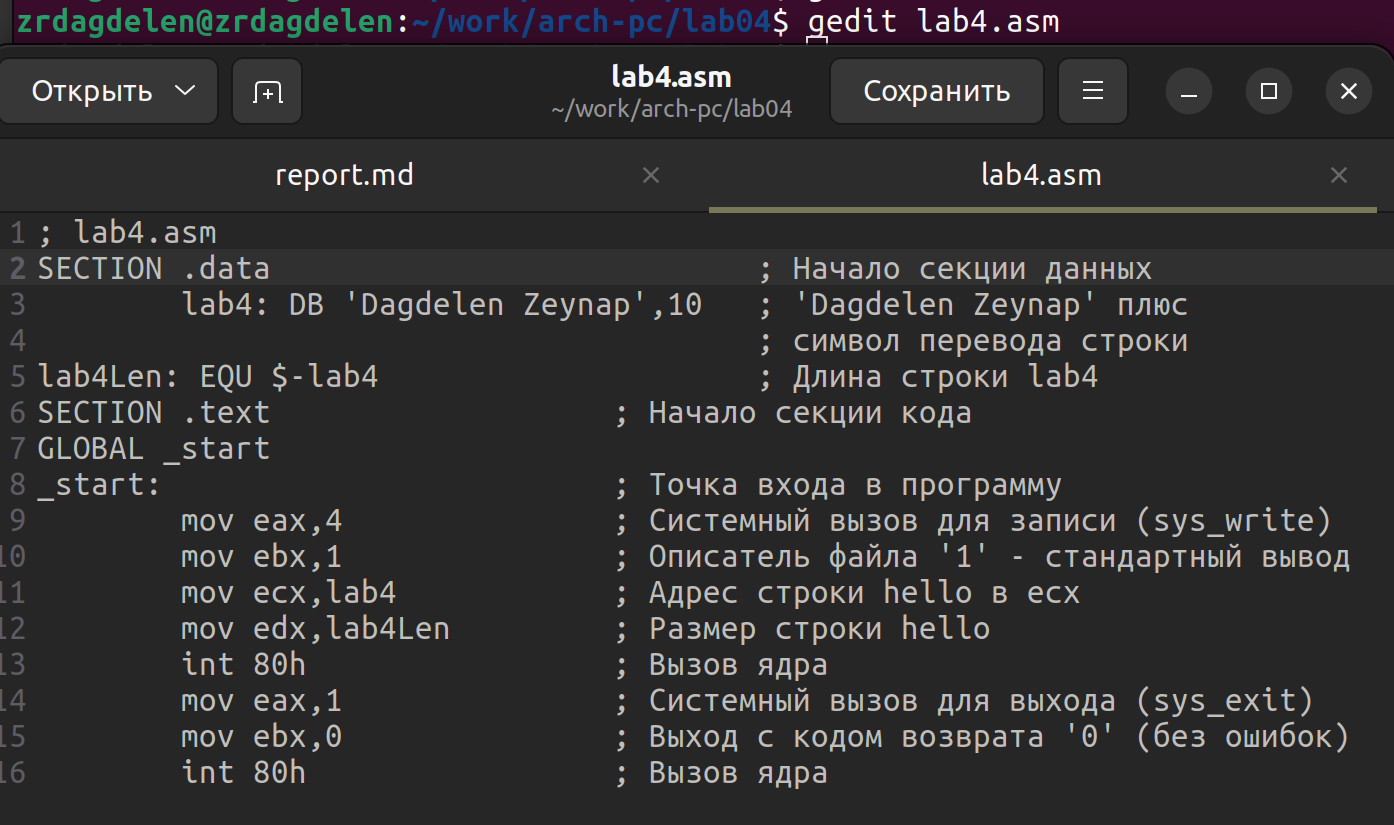
## 4.5 Задание для самостоятельной работы

В каталоге ~/work/arch-pc/lab04 с помощью команды cp создаю копию файла hello.asm с именем lab4.asm (рис. [??]).



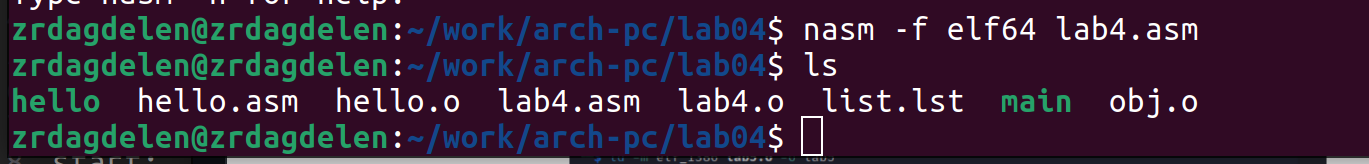
Копирование файлов в нужную папку

С помощью текстового редактора вношу изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с моей фамилией и именем (рис. [??]).



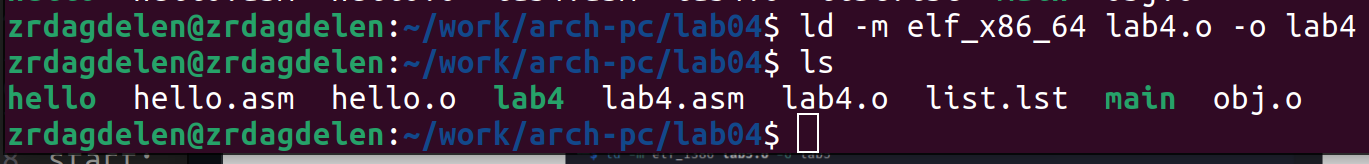
Внесение изменений в текст программы

Оттранслирую полученный текст программы lab4.asm в объектный файл (рис. [??]).

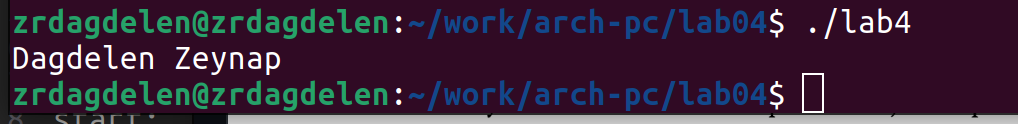


Транслирование в объектный файл (Компиляция текста программы)

Выполняю компоновку объектного файла (рис.[??]) и запускаю получившийся исполняемый файл(рис.[??]).

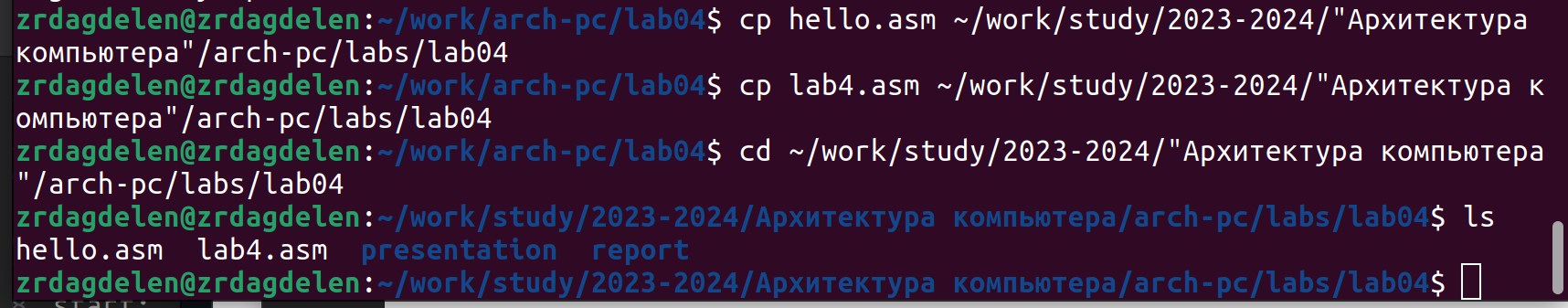


Передача объектного файла на обработку компоновщику



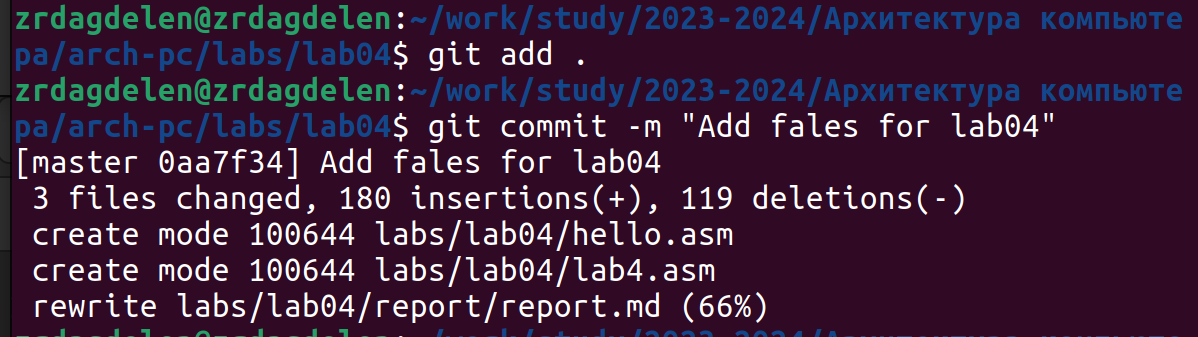
Запуск исполняемого файла

Копирую файлы hello.asm и lab4.asm в свой локальный репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/ (рис.[??]).



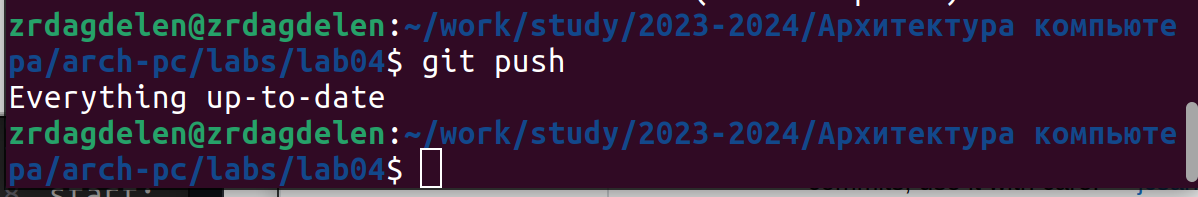
Копирование файлов из одной папки в другую

Загружаю файлы на Github: 1. С помощью команд git add . и git commit добавляю файлы на GitHub, комментируя действие как добавление файлов для лабораторной работы №4 (рис. [??]).



Добавление файлов на GitHub

1. Отправляю файлы на сервер с помощью команды git push (рис. [??]).



Отправка файлов

# 5 Выводы

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

# 6 Список литературы

[Архитектура АВМ](https://esystem.rudn.ru/pluginfile.php/2089084/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%964.%20%D0%A1%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%B8%20%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%20%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B0%20NASM.pdf) [Ссылка на GitHub](https://github.com/zrdagdelen/study_2023-2024_arh-pc/tree/master/labs/lab04/report)