Отчёт по лабораторной работе №4

Дисциплина: архитектура компьютера

Шангина Виктория Андреевна, группа: НКАбд-05-24

Содержание

[1 Цель работы 1](#_Toc180867196)

[2 Задание 1](#_Toc180867197)

[3 Теоретическое введение 1](#_Toc180867198)

[4 Выполнение лабораторной работы 3](#_Toc180867199)

[4.1 Создание программы Hello world! 3](#_Toc180867200)

[4.2 Работа с транслятором NASM 4](#_Toc180867201)

[4.3 Работа с расширенным синтаксисом командной строки NASM 4](#_Toc180867202)

[4.4 Работа с компоновщиком LD 5](#_Toc180867203)

[4.5 Запуск исполняемого файла 5](#_Toc180867204)

[4.6 Выполнение заданий для самостоятельной работы. 5](#_Toc180867205)

[5 Выводы 8](#_Toc180867206)

[6 Список литературы 8](#_Toc180867207)

# 1 Цель работы

Цель данной лабораторной работы - освоить 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

Основными функциональными элементами любой ЭВМ являются центральный процессор, память и периферийные устройства. Взаимодействие этих устройств осуществляется через общую шину, к которой они подключены. Физически шина представляет собой большое количество проводников, соединяющих устройства друг с другом. В современных компьютерах проводники выполнены в виде электропроводящих дорожек на материнской плате. Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора 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 два типа: регистры общего назначения и специальные регистры. Для того, чтобы писать программы на ассемблере, необходимо знать, какие регистры процессора существуют и как их можно использовать. Большинство команд в программах написанных на ассемблере используют регистры в каче- стве 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, как к основной памяти, а по именам. Каждый регистр процессора архитектуры x86 имеет свое название, состоящее из 2 или 3 букв латинского алфавита. 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

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. Периферийные устройства в составе ЭВМ: - устройства внешней памяти, которые предназначены для долговременного хранения больших объёмов данных. - устройства ввода-вывода, которые обеспечивают взаимодействие ЦП с внешней средой.

В основе вычислительного процесса ЭВМ лежит принцип программного управления. Это означает, что компьютер решает поставленную задачу как последовательность действий, записанных в виде программы.

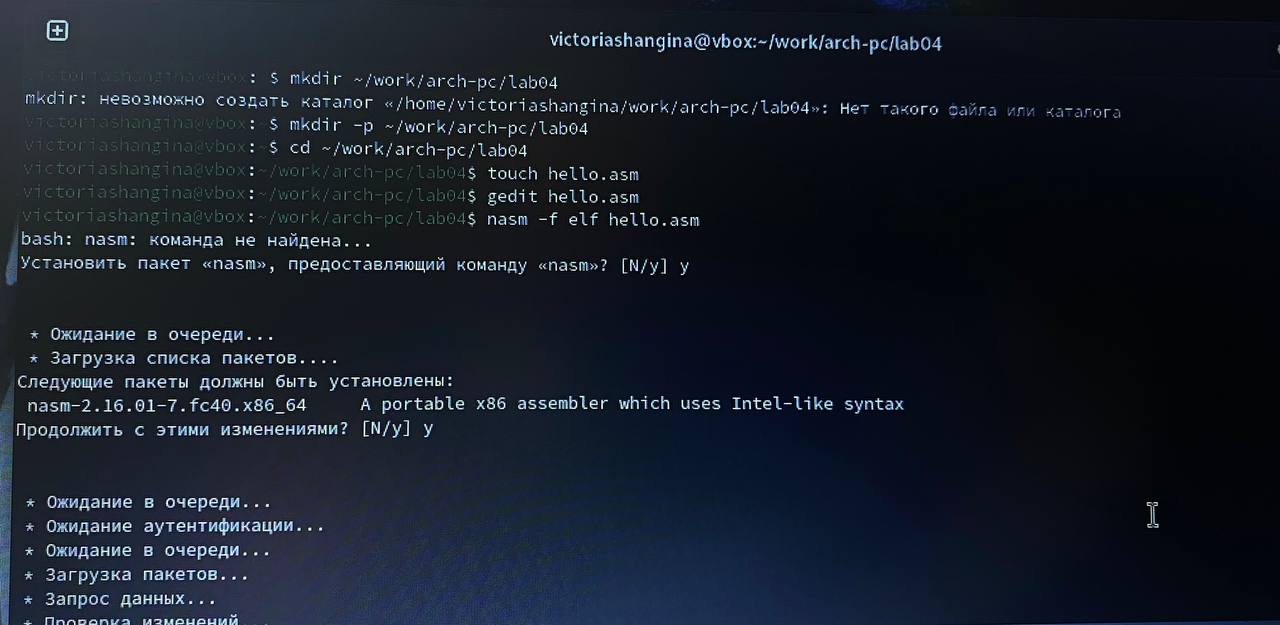
Коды команд представляют собой многоразрядные двоичные комбинации из 0 и 1. В коде машинной команды можно выделить две части: операционную и адресную. В операционной части хранится код команды, которую необходимо выполнить. В адресной части хранятся данные или адреса данных, которые участвуют в выполнении данной операции. При выполнении каждой команды процессор выполняет определённую последовательность стандартных действий, которая называется командным циклом процессора. Он заключается в следующем: 1. формирование адреса в памяти очередной команды; 2. считывание кода команды из памяти и её дешифрация; 3. выполнение команды; 4. переход к следующей команде.

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. 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.

# 4 Выполнение лабораторной работы

## 4.1 Создание программы Hello world!

При помощи mkdir создаю каталог для работы с программами на языке ассемблера NASM. Затем перехожу в созданный каталог и создаю текстовый файл с именем hello.asm. Далее открываем файл при помози текстового редактора gedit, так как этот пакет раннее не был установлен, следую подсказкам. (рис. 1).

[](https://github.com/victoriashangina/study_2024-2025_arh-pc/blob/master/labs/lab04/report/image/5246698694533834306.jpg)Рис.1 *Перемещение между директориями.*

В текстовом редакторе ввожу нужный текст. (рис.2).

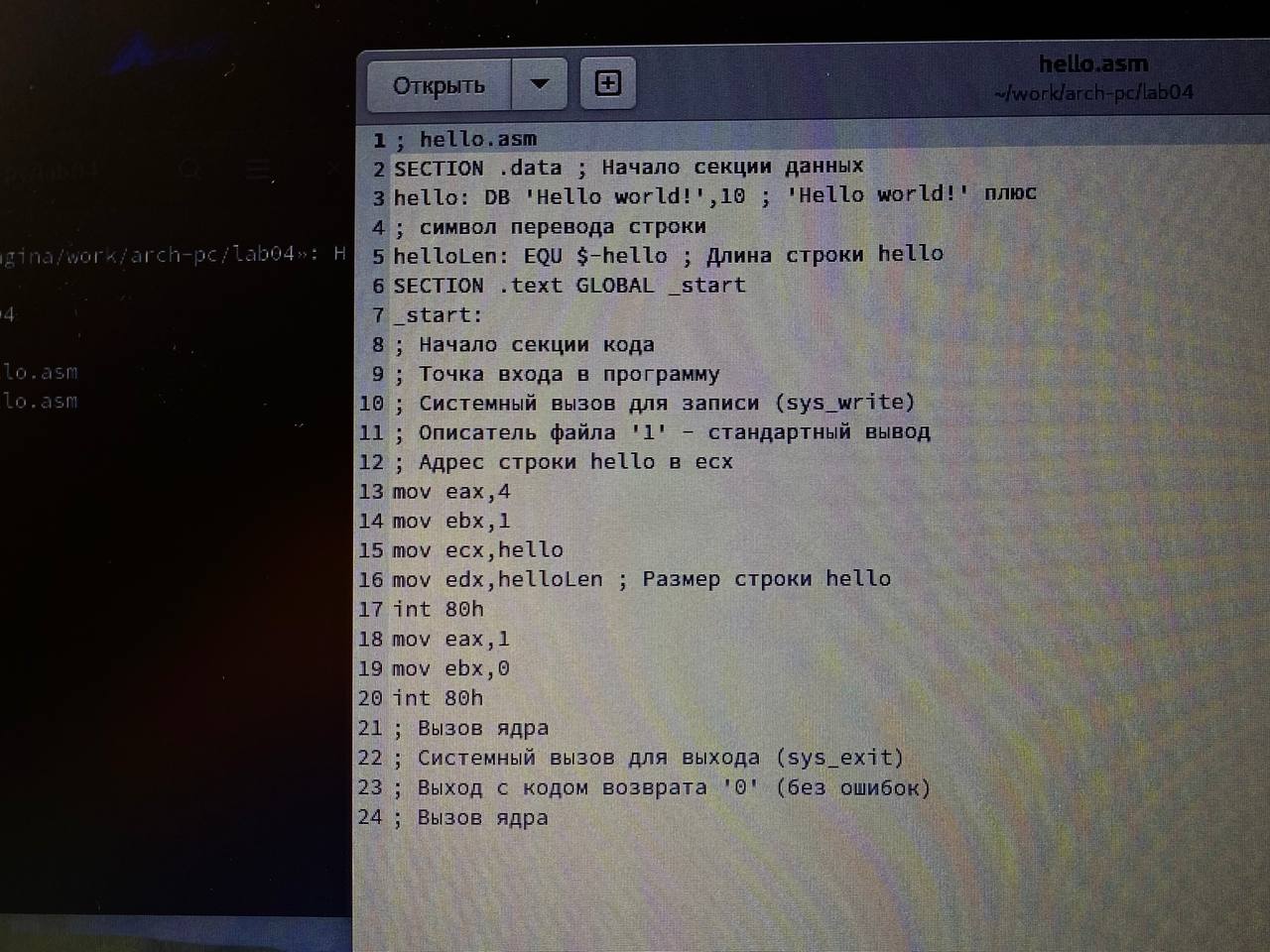
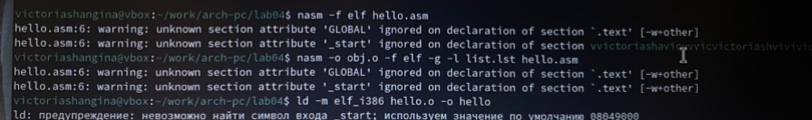
[](https://github.com/victoriashangina/study_2024-2025_arh-pc/blob/master/labs/lab04/report/image/5246698694533834309.jpg)

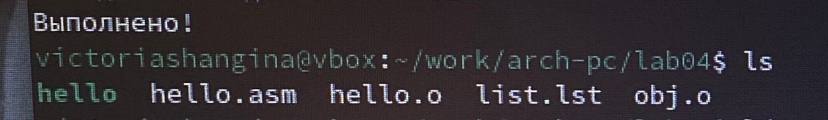
Рис. 2. *Заполнение файла в текстовом редакторе*

## 4.2 Работа с транслятором NASM и с раширенным синтаксисом командной строки NASM

Для компиляции приведенного выше текста программы "Hello World" использую команду nasm -f elf hello.asm, предварительно устанавливая пакет "nasm". Далее выполняем следующую команду. Далее скомпилирую исходный файл hello.asm в obj.o при помощи 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. 3).

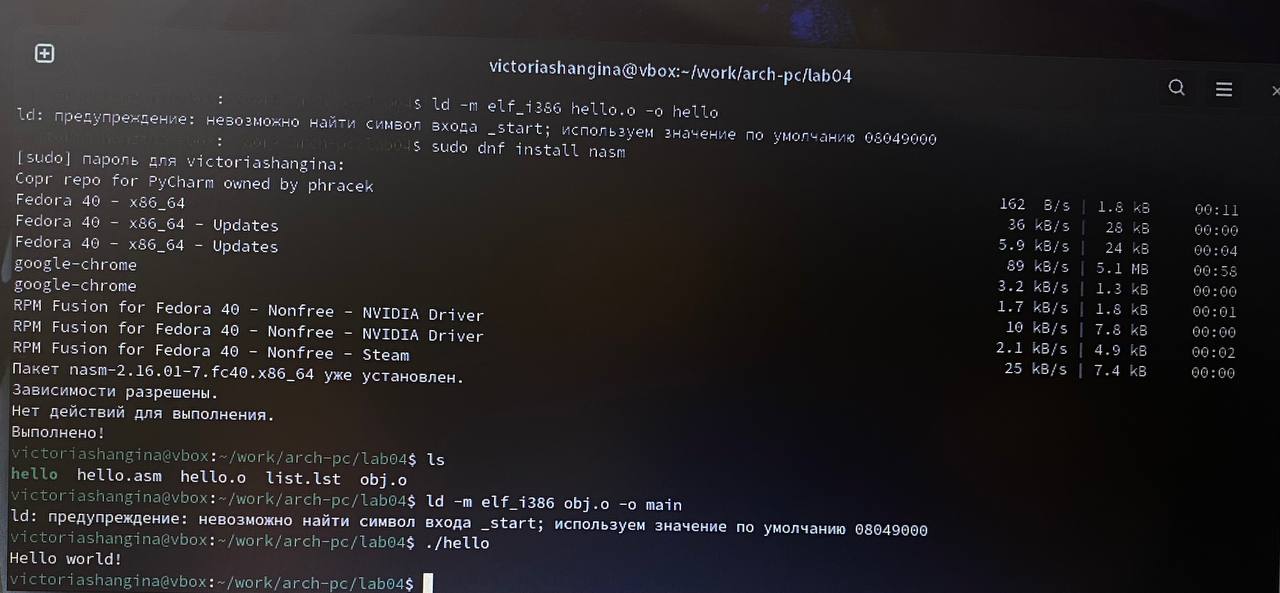
[](https://github.com/victoriashangina/study_2024-2025_arh-pc/blob/master/labs/lab04/report/image/5248950494347520364.jpg)Рис. 3. *Компиляция текста программы*

При помощи ls проверяем содержимое (рис. 4).

[](https://github.com/victoriashangina/study_2024-2025_arh-pc/blob/master/labs/lab04/report/image/5248950494347520365.jpg)Рис. 4. *Проверка содержимого*

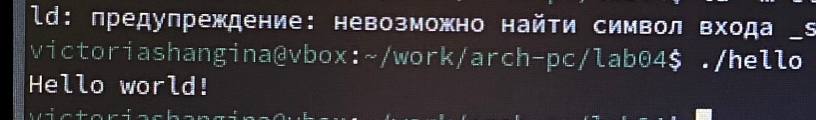
## 4.4 Работа с компоновщиком LD

Передаю объектный файл на обработку компоновщику при помощи команды ld -m elf\_i386 hello.o -o hello (рис.5).

[](https://github.com/victoriashangina/study_2024-2025_arh-pc/blob/master/labs/lab04/report/image/5246698694533834308.jpg)Рис. 5. *Передача объектного файла на обработку компоновщику*

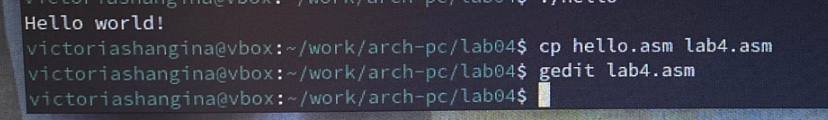
## 4.5 Запуск исполняемого файла

Запускаю на выполнение созданный файл hello (рис. 6).

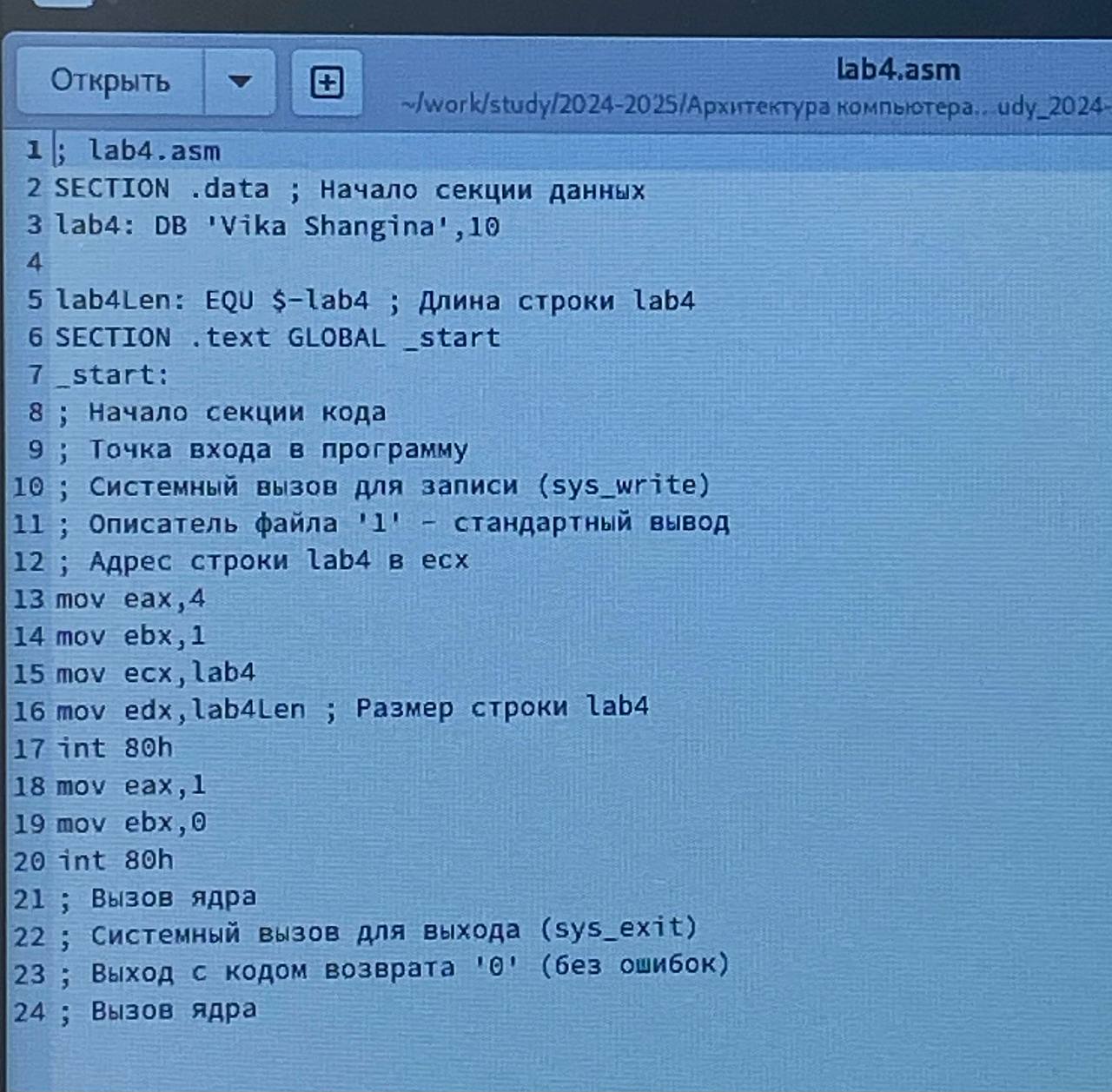
[](https://github.com/victoriashangina/study_2024-2025_arh-pc/blob/master/labs/lab04/report/image/5248950494347520451.jpg)Рис. 6. *Запуск исполняемого файла*

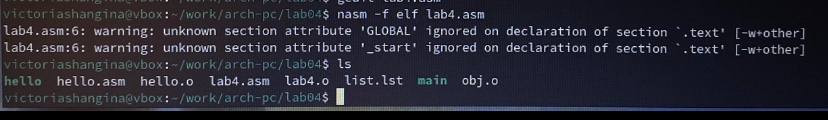
## 4.6 Выполнение заданий для самостоятельной работы.

1. В каталоге ~/work/arch-pc/lab04 с помощью команды cp создаю копию файла hello.asm с именем lab4.asm (рис. 7).

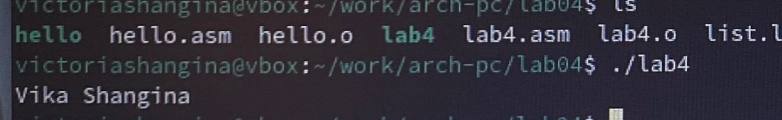
[](https://github.com/victoriashangina/study_2024-2025_arh-pc/blob/master/labs/lab04/report/image/5248950494347520508.jpg)Рис. 7. *Создание копии файла*

1. При помощи текстового редактора вношу изменения в текст программы в файле lab4.asm так,чтобы вместо Hello world! на экран выводилась строка с моими фамилией и именем. (рис. 8).

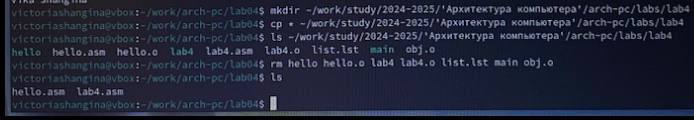
[](https://github.com/victoriashangina/study_2024-2025_arh-pc/blob/master/labs/lab04/report/image/5248950494347520576.jpg)Рис. 8. *Изменение программы*

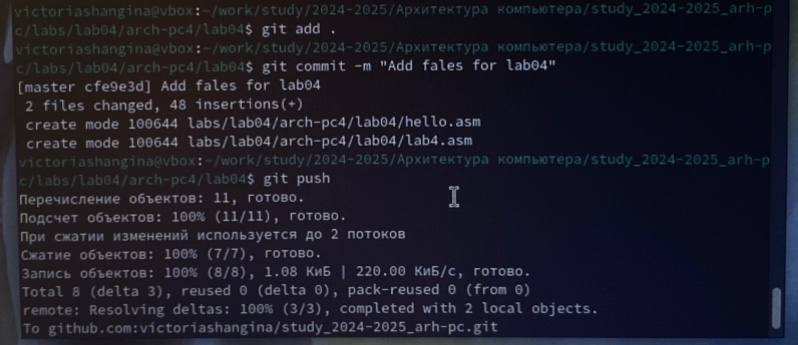
1. Оттранслирую полученный текст программы lab4.asm в объектный файл (рис. 9). [](https://github.com/victoriashangina/study_2024-2025_arh-pc/blob/master/labs/lab04/report/image/5248950494347520509.jpg)Рис. 9. *Компиляция и передача объектного файла на обработку компоновщику*

Выполняю компоновку объектного файла и запускаю получившийся исполняемый файл (рис. 10).

[](https://github.com/victoriashangina/study_2024-2025_arh-pc/blob/master/labs/lab04/report/image/5248950494347520510.jpg)Рис. 10. *Запуск исполняемого файла*

1. Копирую файлы hello.asm и lab4.asm в локальный репозиторий в каталог ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab04/ (рис. 11).

[](https://github.com/victoriashangina/study_2024-2025_arh-pc/blob/master/labs/lab04/report/image/5248950494347520511.jpg)Рис. 11. *Копирование файлов и удаление лишних*

1. Загружаю файлы на Github при помощи git add . , git commit и git push (рис. 12). [](https://github.com/victoriashangina/study_2024-2025_arh-pc/blob/master/labs/lab04/report/image/5248950494347520512.jpg)Рис.12*. Добавление файлов на Github*

# 5 Выводы

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

# 6 Список литературы

1. PDF файл: Лабораторная работа №4. Создание и процесс обработки программ на языке ассемблера NASM