РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

дисциплина: Архитектура компьютера

Студент: Загидуллина Э. Д.

Группа: НБИбд-01-25

МОСКВА

2025 г.

Оглавление

Список иллюстраций

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 взаимодействуетс 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—этоадрес хранящихся в нейданных.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) — машинноориентированный 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

С помощью утилиты cd перемещаюсь в каталог, в котором буду работать (рис. 4.1).

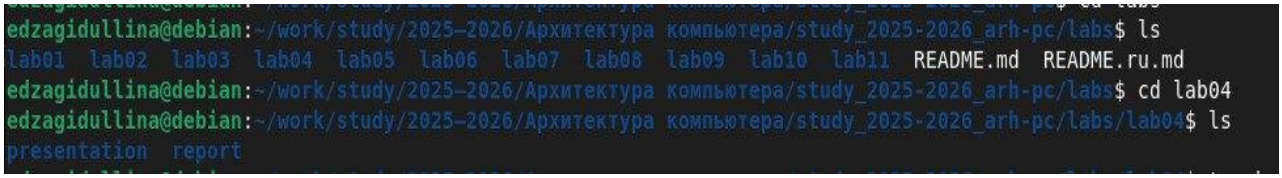

Рис. 4.1 Перемещение между директориями

Создаю в текущем каталоге пустой текстовый файл hello.asm с помощью утилиты touch (рис. 4.2).

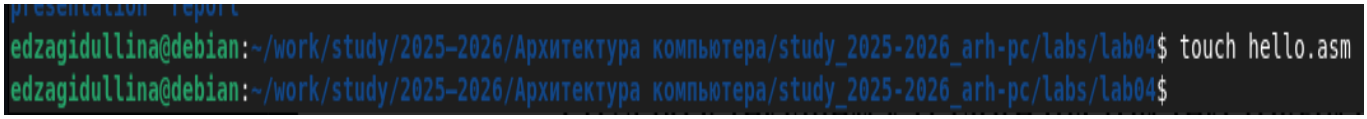

Рис. 4.2 Создание пустого файла

Открываю созданный файл в текстовом редакторе gedit (рис. 4.3)

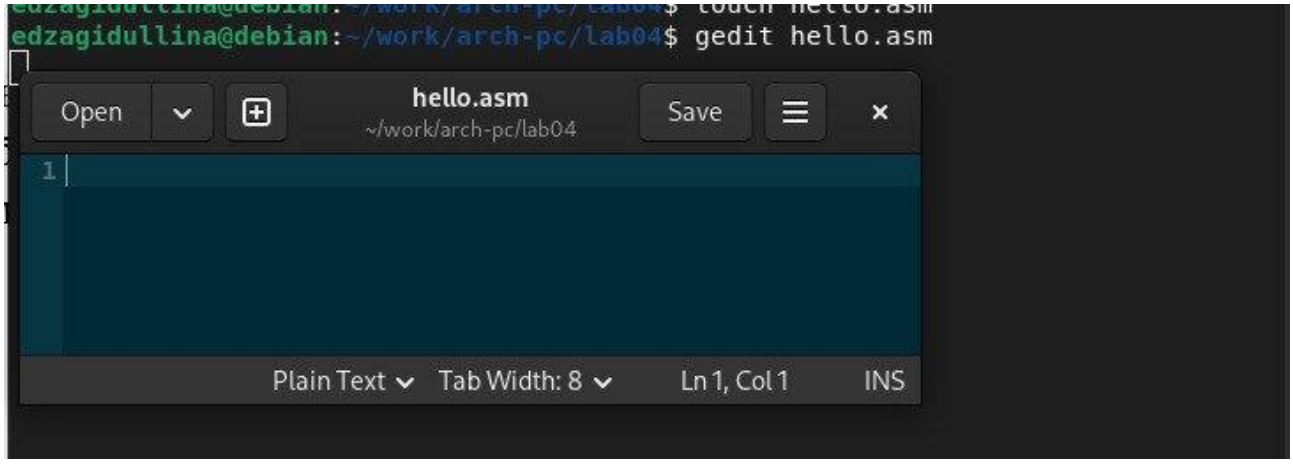

Рис. 4.3 Открытие файла в текстовом редакторе

Заполняю файл, вставляя в него программу для вывода “Hello word!” (рис. 4.4)

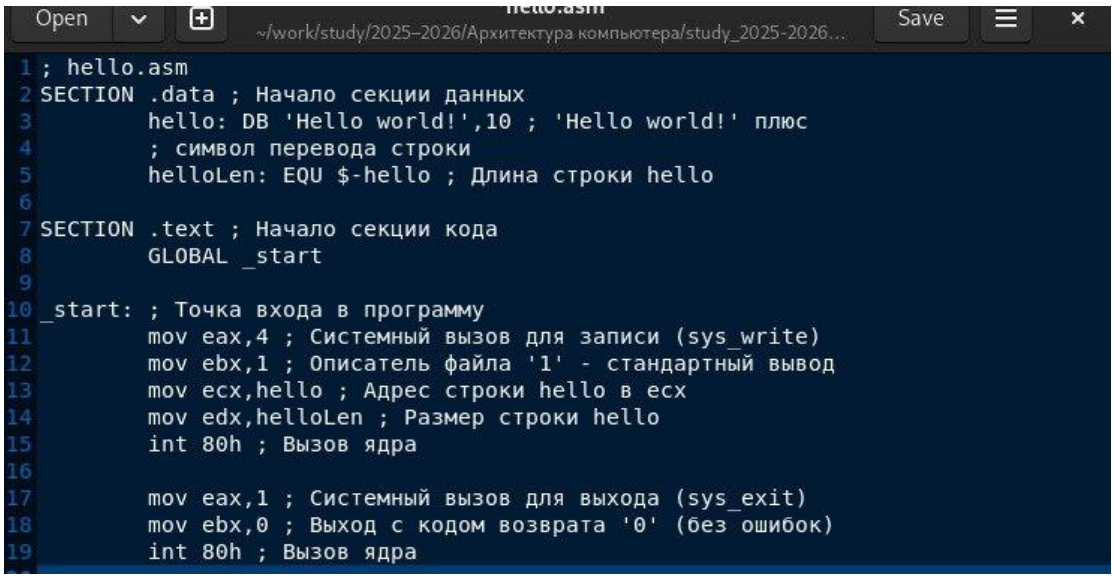

Рис. 4.4 Заполнение файла

4.2 Работа с транслятором NASM

Превращаю текст программы для вывода “Hello world!” в объектный код с помощью транслятора NASM, используя команду nasm -f elf hello.asm, ключ -f указываеттранслятору nasm, что требуется создать бинарный файл в формате ELF (рис. 4.5). Далее проверяю правильность выполнения команды с помощью утилиты ls: действительно, создан файл “hello.o”.

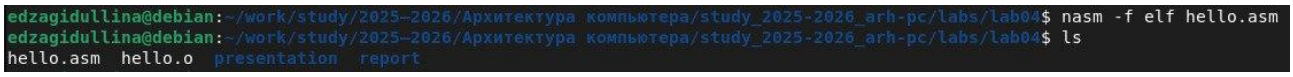

Рис. 4.5 Компиляция текста программы и проверка правильности выполнения

4.3 Работа с расширенным синтаксисом командной строки NASM

Скачиваю библиотеку NASM для работы с командой nasm (рис.4.6)

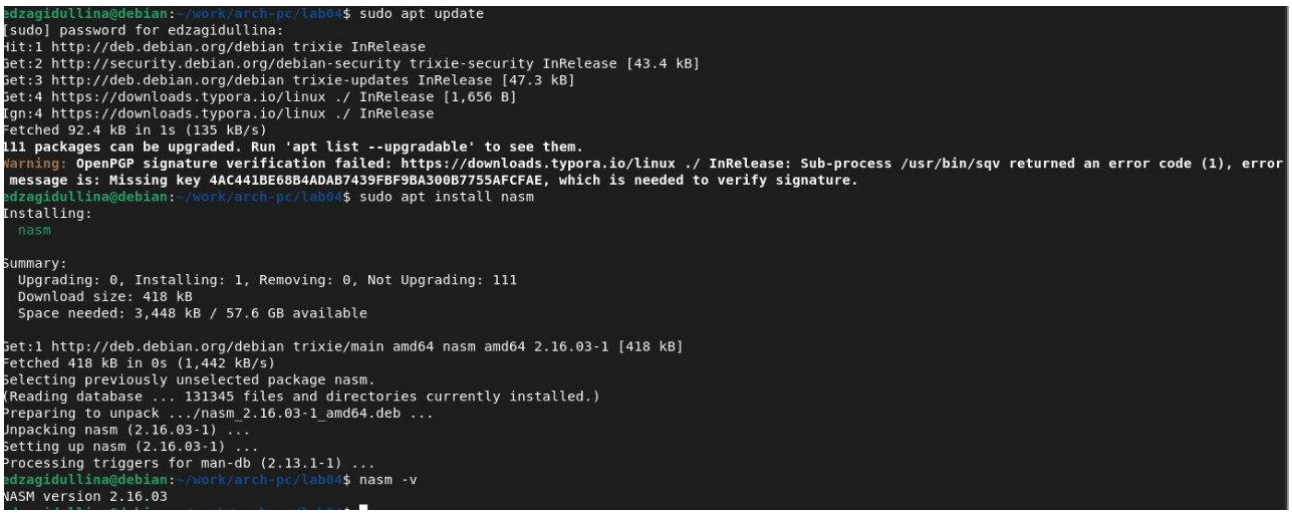

Рис. 4.6 Скачивание библиотеки NASM

Ввожу команду, которая скомпилирует файл hello.asm в файл obj.o, при этом в файл будут включены символы для отладки (ключ -g),также с помощью ключа -l будет создан файл list.lst (рис.4.7). Далее проверяю с помощью ls правильность выполнения команды

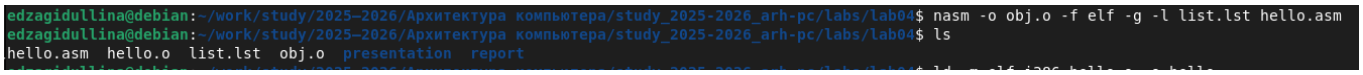

Рис. 4.7 Компиляция текста программы и проверка правильности выполнения

4.4 Работа с компоновщиком LD

Скачиваю библиотеку binutils для работы с LD (рис. 4.8)

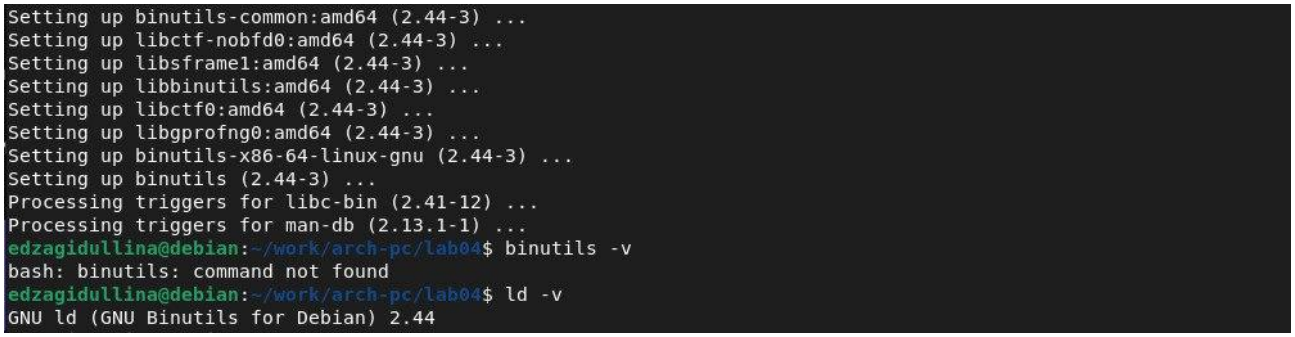

Рис. 4.8 Скачивание библиотеки binutils

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello (рис. 4.9). Ключ -o задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты ls правильность выполнения команды.

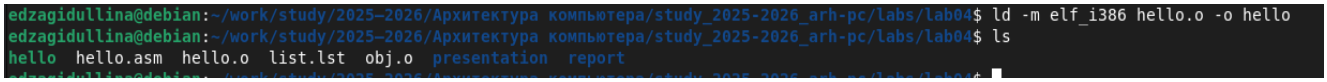

Рис. 4.9 Передача объектного файла на обработку компоновщику проверка правильности выполнения

4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello (рис. 4.10)

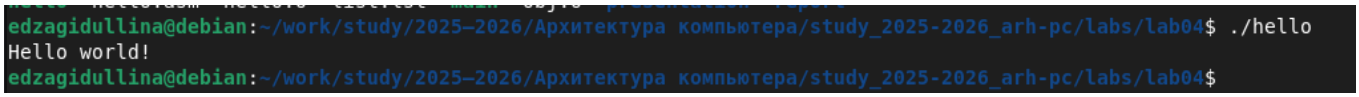

Рис. 4.10 Запуск исполняемого файла

4.6 Задание для самостоятельной работы

С помощью утилиты cp создаю в текущем каталоге копию файла hello.asm с именем lab4.asm (рис. 4.11).

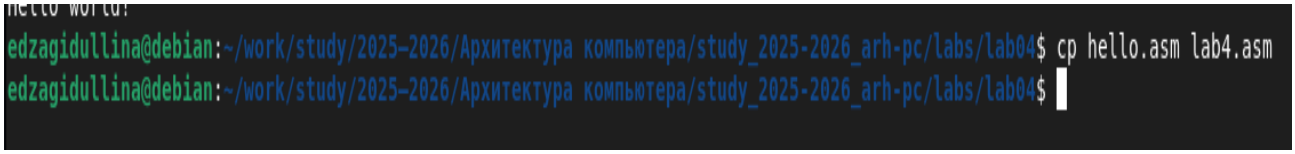

Рис. 4.11 Создание копии файла

С помощью текстового редактора gedit вношу изменения в текст программы в файле lab4.asm так, чтобы вместо “Hello World!” на экране выводилась строка с моим именем и фамилией (“Elina Zagidullina”) (рис. 4.12) (рис. 4.13)

```
edzagidullina@debian:~/work/study/2025–2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab04$ gedit lab4.asm
edzagidullina@debian:~/work/study/2025–2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab04$
```

Рис. 4.12 Открытие текстового редактора

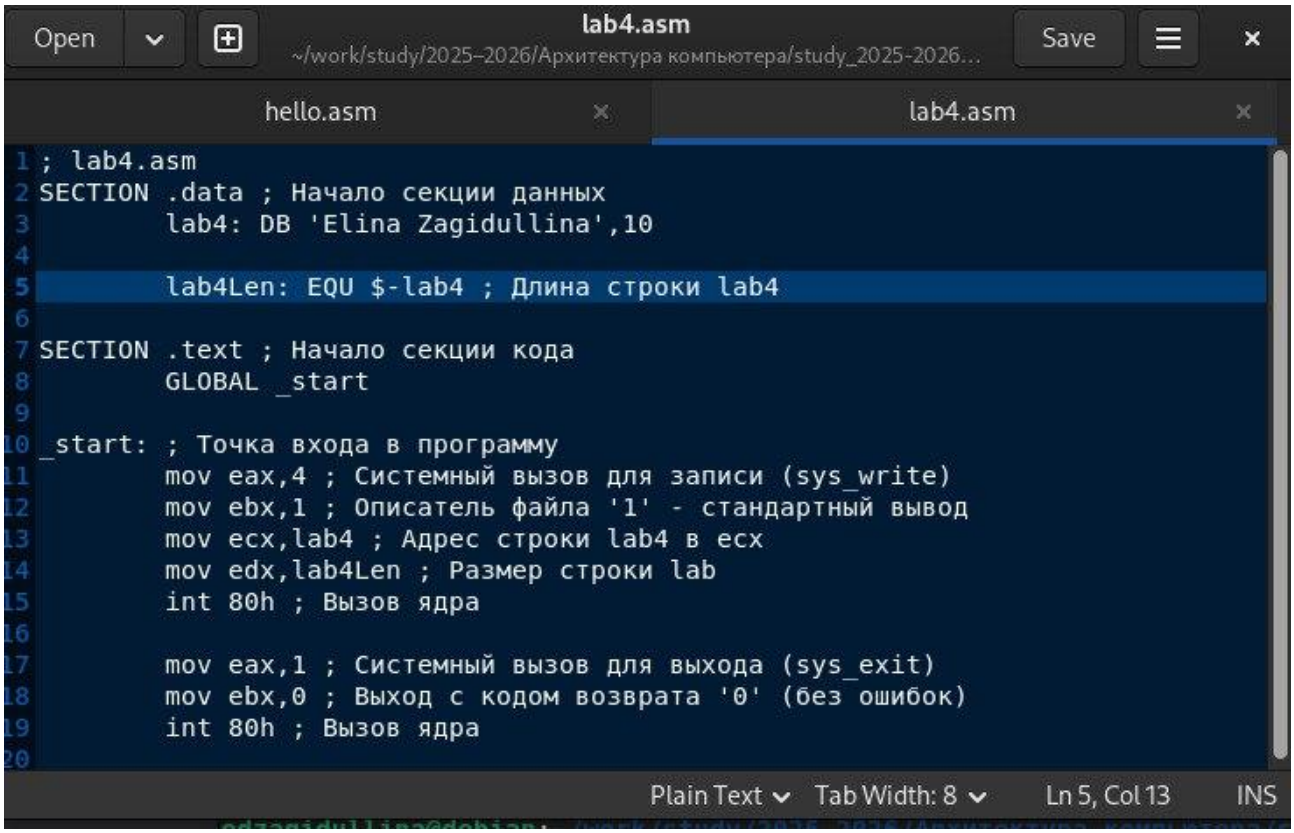

Рис. 4.13 Редактирование текста программы

Компилирую текст программы в объектный файл (рис. 4.14). Проверяю с помощью утилиты ls, что файл lab4.o создан.

```
edzagidullina@debian:~/work/study/2025–2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab04$ nasm -f elf lab4.asm
edzagidullina@debian:~/work/study/2025–2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  lab4.o  list.lst  main  obj.o  presentation  report
```

Рис. 4.14 Компиляция текста программы и проверка создания файла

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab5 (рис. 4.15)

```
edzagidullina@debian:~/work/study/2025–2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab04$ ld -m elf_i386 lab4.o -o lab4
edzagidullina@debian:~/work/study/2025–2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab04$ ды
bash: ды: command not found
edzagidullina@debian:~/work/study/2025–2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab04$ ls
hello  hello.asm  hello.o  lab4  lab4.asm  lab4.o  list.lst  main  obj.o  presentation  report
```

Рис. 4.15 Передача объектного файла на обработку компоновщику и проверка

Запускаю исполняемый файл lab4, на экран действительно выводятся мои имя и фамилия (рис. 4.16)

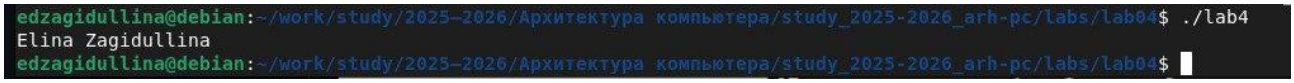

Рис. 4.16 Запуск исполняемого файла и вывод содержимого на экран

Копирую файлы hello.asm и lab.asm в личный репозиторий и загружаю файлы на GitHub (рис. 4.17)

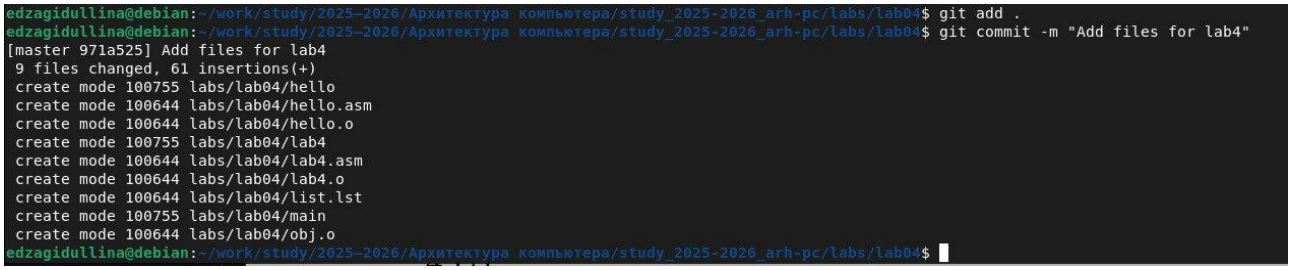

Рис. 4.17 Загрузка файлов на GitHub

5 Выводы

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