**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: Армихос Гонзалез Карла

Группа: НКАбд-02-24

**МОСКВА**

2024 г.

Содержание

[1. Цель работы 4](#_65smd4tv8isq)

[2. Задание 5](#_qs0a932mqoyf)

[3. Технические введение 6](#_7wijli3jdso5)

[4. Порядок выполнения лабораторной работы 8](#_aegffh1k9ms2)

[4.1 Программа Hello world! 8](#_49gvipf7a5tc)

[4.2 Транслятор NASM 8](#_j3lu7y28r7e1)

[4.3 Расширенный синтаксис командной строки NASM 9](#_djp4r95lng03)

[4.4 Компоновщик LD 9](#_ny0s208cvuln)

[4.5 Запуск исполняемого файла 10](#_y95ng2qpisov)

[5. Задание для самостоятельной работы 11](#_ieosai5l3xcf)

[5.1 Создать копия 11](#_87kk5tieav6t)

[6. Выводы 13](#_k70p842lbzxt)

[7. Список используемой литературы 14](#_6n2mzfcpbmb1)

Список иллюстраций

### 

[рис. 4.1.1 новый каталог. 7](#_fv0w0cufgdxo)

[рис. 4.1.2 Создание .asm файла 7](#_45lca4q60zmk)

[рис. 4.1.3 8](#_l43d9aehd4rv)

[рис. 4.2.1 транслятор nasm 8](#_f8kar6tjmo4n)

[рис. 4.3.1 8](#_u5rylq2xjn0f)

[рис. 4.4.1 8](#_hdy5m2y3nao4)

[рис. 4.4. ключ 9](#_dciucez8qhhv)

[рис. 4.5.1 9](#_5e0lab69dgul)

[рис. 5.1 Копия 10](#_rh0h3xhpgzrc)

[рис. 5.2 Редактировать текст 11](#_9bksxftt9sl0)

[рис. 5.3 Запустить программу 11](#_ejlabqen2q1n)

### 

## 

## 

# 1. Цель работы

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

# **2. Задание**

1. Создание программы Hello world!
2. Основные концепты работы с файловым менеджером "mc".
3. Структура приложения на языке ассемблера NASM.
4. Процесс подключения внешнего файла.
5. Выполнение заданий для самостоятельной работы.

# **3. Технические введение**

Основные отличия ассемблерных программ от программ на языках высокого уровня заключаются в уровне абстракции и сложности. Ассемблерные программы пишутся для конкретного процессора и оперируют напрямую его инструкциями, что позволяет максимальный контроль над аппаратной частью. Программы на языках высокого уровня, таких как C или Python, скрывают аппаратные детали, предлагая более понятный и читаемый код, но уступают в скорости и контроле.

Инструкция на языке ассемблера — это команда, которую процессор выполняет непосредственно, например, операция сложения или перемещения данных. Директива — это команда, предназначенная для ассемблера, помогающая ему организовать код, но которая не превращается в машинный код, как, например, определение секций данных. (рис.1)

Основные правила оформления программ на ассемблере включают использование комментариев для пояснения кода, правильное отступление и группировку кода в секции данных, текстовую секцию и стек. Это облегчает чтение и поддержку кода.



рис. 1.

Этапы получения исполняемого файла включают написание исходного кода, трансляцию (ассемблирование), компоновку и, наконец, запуск программы. Эти шаги позволяют перевести код из понятного для человека вида в машинный код.

Назначение этапа трансляции — преобразовать ассемблерный код в машинный код. На этом этапе создается объектный файл, содержащий машинные инструкции, которые соответствуют ассемблерным командам.

Назначение этапа компоновки заключается в объединении всех объектных файлов и библиотек в один исполняемый файл, включая установку всех ссылок и адресов.

При трансляции программы могут создаваться различные файлы, такие как объектные файлы (.o) и исполняемые файлы. По умолчанию, в результате трансляции создается объектный файл, а при компоновке — исполняемый файл.

Форматы файлов для nasm и ld обычно включают объектные файлы в формате ELF (Executable and Linkable Format) на Linux или COFF (Common Object File Format) на Windows.

# **4. Порядок выполнения лабораторной работы**

## 4.1 Программа Hello world!

Создать новый каталог а затем войдите в созданную папку.(рис. 4.1.1)



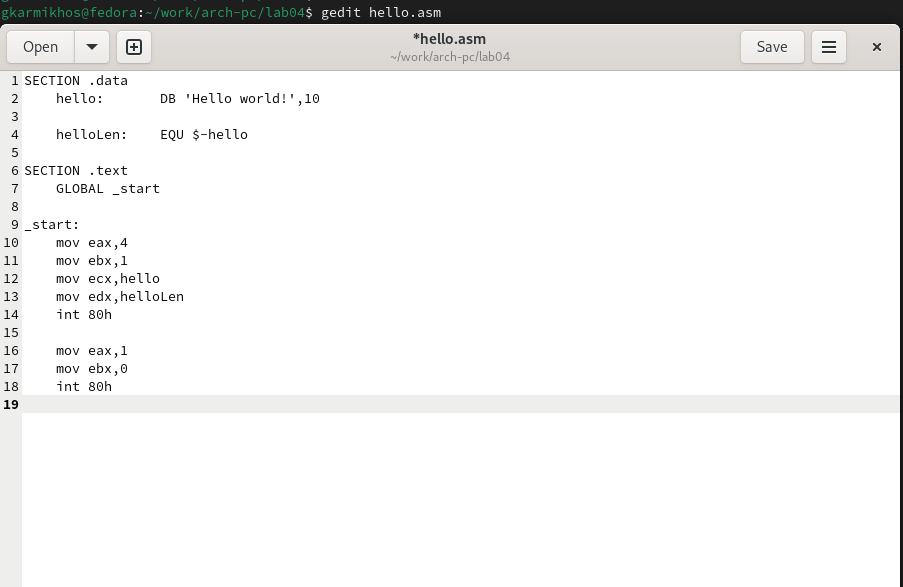
### рис. 4.1.1 новый каталог.

Создайте текстовый файл с именем hello.asm (рис. 4.1.2)



### рис. 4.1.2 Создание .asm файла

Открываем файл в текстовом редакторе gedit и пишем программу(рис.4.1.3)

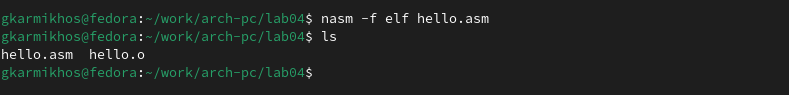


### рис. 4.1.3 файл

# 

## 4.2 Транслятор NASM

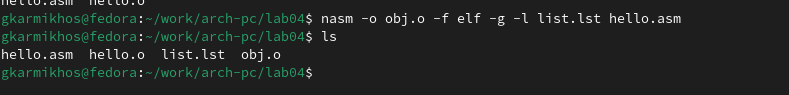
Преобразование текста программы в код, затем для подтверждения вводим команду ls (рис. 4.2.1)



### рис. 4.2.1 транслятор nasm

## 4.3 Расширенный синтаксис командной строки NASM

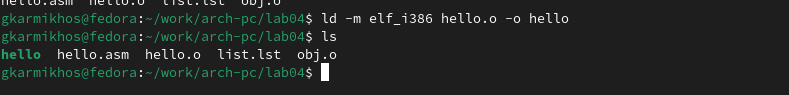
Скомпилируйте файл и убедитесь, что он создан правильно (рис. 4.3.1)



### рис. 4.3.1

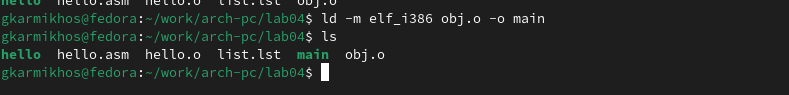
## 4.4 Компоновщик LD

Связать объектный файл для обработки(рис. 4.4.1)



### рис. 4.4.1

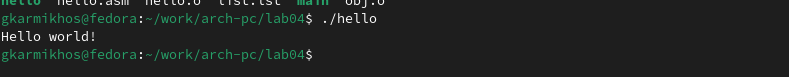
Выполнить ключ (рис. 4.4.2)



### рис. 4.4. ключ

## 4.5 Запуск исполняемого файла

Запустить созданный каталог(рис. 4.5.1)

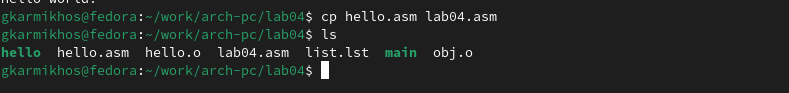
****

### рис. 4.5.1

# **5. Задание для самостоятельной работы**

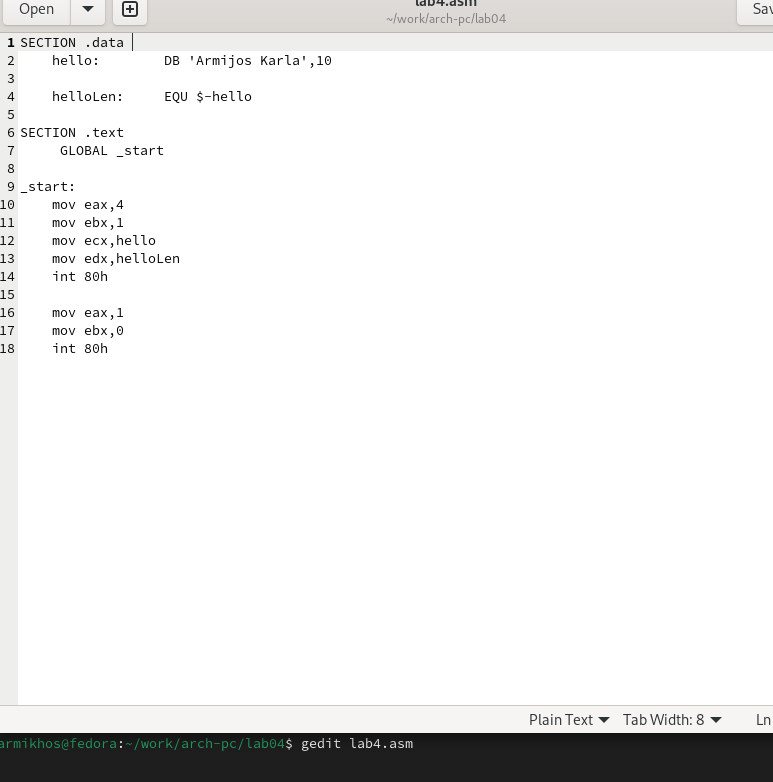
## 5.1 Создать копия

Создать копию файла с помощью команды cp(рис. 5.1)

****

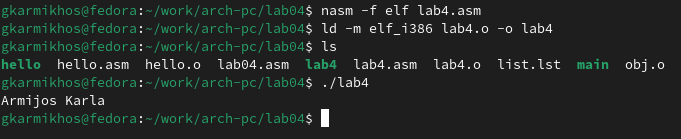
### рис. 5.1 Копия

Отредактируйте текст, чтобы при его запуске отображались мои имя и фамилия(рис. 5.2)



### рис. 5.2 Редактировать текст

Воспроизвести отредактированный текст.(рис. 5.3)



### рис. 5.3 Запустить программу

# 6. Выводы

В ходе выполнения задания были изучены основы работы с языком ассемблера, его специфика и практические применения.

Эта задача также включала сохранение файлов в локальном репозитории и их загрузку на GitHub.

# 7**. Список используемой литературы**

1. [Архитектура ЭВМ](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)