Шаблон отчёта по лабораторной работе №4

Архитектура компьютера

Гафоров Нурмухаммад Вомикович

Содержание

# 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:

формирование адреса в памяти очередной команды;  
считывание кода команды из памяти и её дешифрация;  
выполнение команды;  
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

# 5 Создание програма Hello word

Открыли терминал (рис.[??]).

Открить Терминал

Открить Терминал

Создайте каталог для работы с программами на языке ассемблера NASM(рис. [??]).

Создал каталог NASM

Создал каталог NASM

Прейдите в созданный каталог (рис. [??])

Приходил в созданный каталог

Приходил в созданный каталог

Создаю в текущем каталоге пустой текстовый файл Hello.asm с помощью touch (рис. [??])

cоздали файл

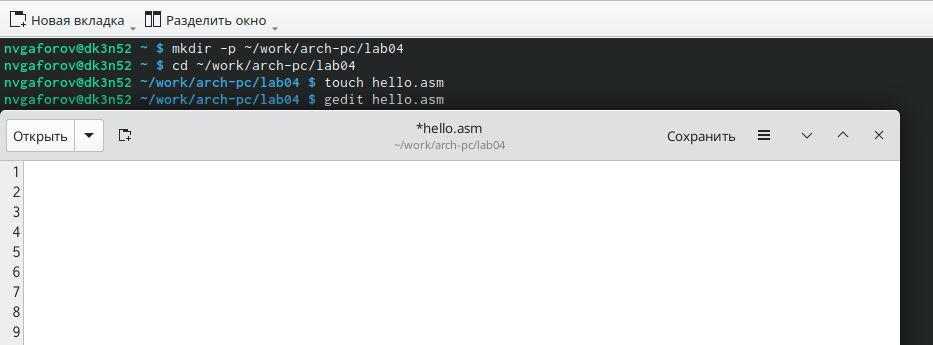
cоздали файл

Откройте файл с помощью gedit(рис. [??])

Открили файл

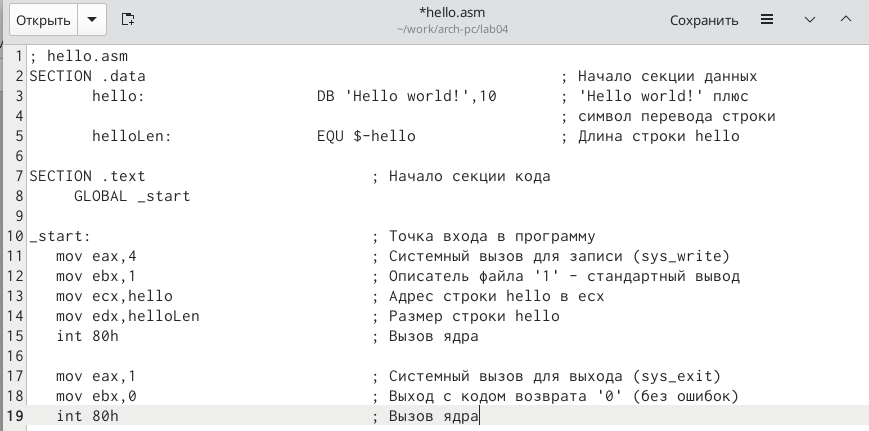
Открили файл

Введите в него следующий текст(рис. [??])



Созданный файл Hello asm

Напишите текст (рис. [??])



Написали текст

## 5.1 Работа с транслятором NASM

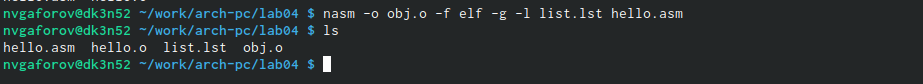
Превращаю текст программы для вывода “Hello world!” в объектный код с помощью транслятора NASM, используя команду nasm -f elf hello.asm, ключ -f указывает транслятору nasm, что требуется создать бинарный файл в формате ELF . Далее проверяю правильность выполнения команды с помощью утилиты ls: действительно,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

Компиляция текста программы транслятор NASM

Компиляция текста программы транслятор NASM

## 5.2 Работа с расширенным синтаксисом командной строки NASM

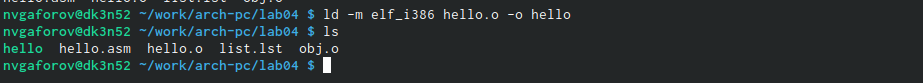
Ввожу команду, которая скомпилирует файл hello.asm в файл obj.o, при этом в файл будут включены символы для отладки (ключ -g), также с помощью ключа -l будет создан файл листинга list.lst . Далее проверяю с помощью утилиты ls правильность выполнения команды.(рис. [??])



Расширенный синтаксис командной строки NASM

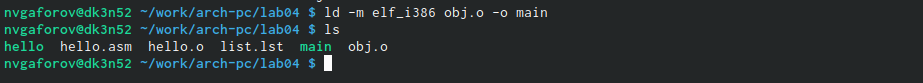
## 5.3 Работа с компоновщиком LD

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



Передача объектного файла на обработку компоновщику

Выполняю следующую команду. 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 (рис. [??])



Передача объектного файла на обработку компоновщику

## 5.4 Запуск исполняемого файла

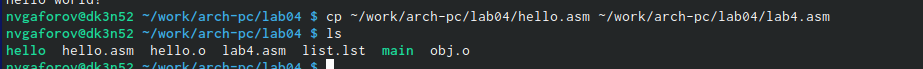
Запускаю на выполнение созданный исполняемый файл hello (рис. [??])

Запуск исполняемого файла

Запуск исполняемого файла

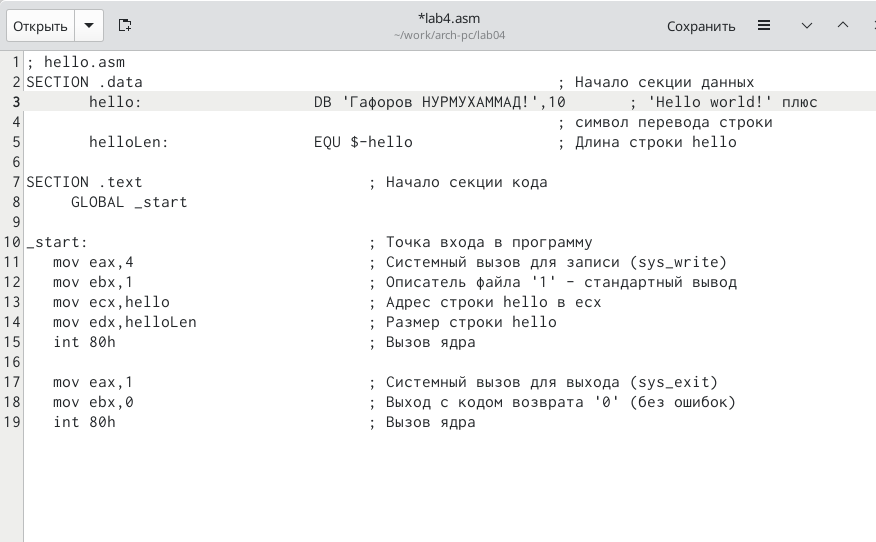
## 5.5 Выполнение заданий для самостоятельной работы.

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



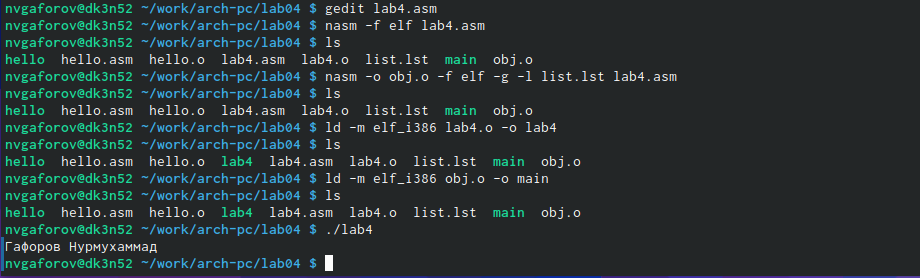
Создаль файл lab4

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



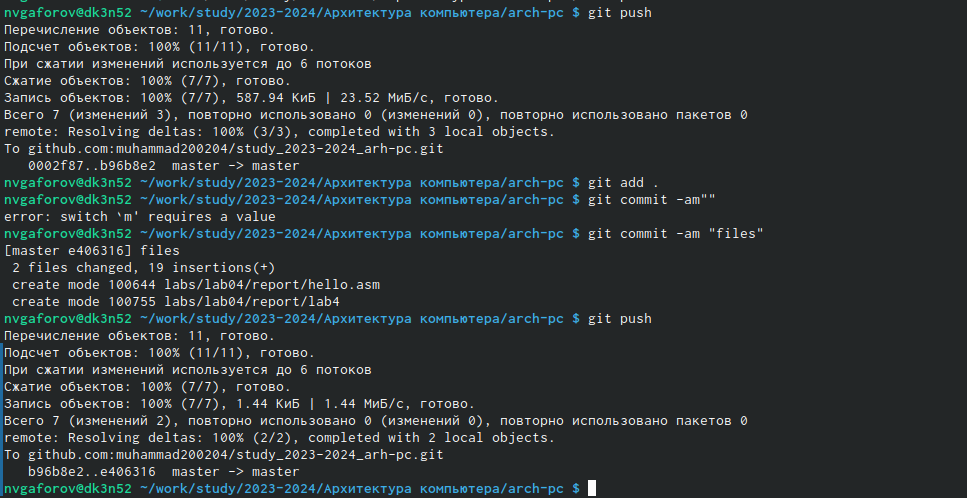
Изменение программы

Оттранслируйте полученный текст программы lab4.asm в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл на экран выводилась строка с вашими фамилией и именем(рис. [??])



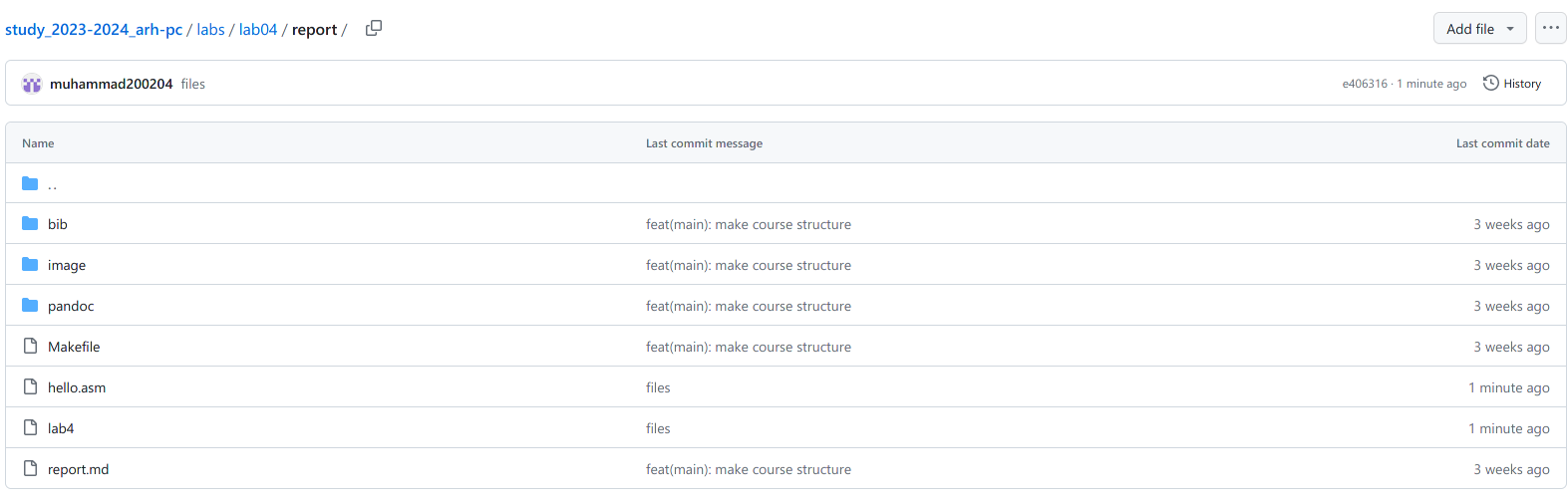
создал файл lab4 и откриль

Скопируйте файлы hello.asm и lab4.asm в Ваш локальный репозиторий в ката- лог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/. Загрузите файлы на Github.(рис. [??])



Загрузит файл на Github

Проверяем файли (рис. [??])



Проверка файли

# 6 Выводы

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

# Список литературы

https://esystem.rudn.ru/pluginfile.php/1584628/mod\_resource/content/1/%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%965.pdf