**Отчёта по лабораторной работе №4**

#### Дисциплина: архитектура компьютера

### Зоригоо Номун

**Содержание**

1. [Цель работы](#_bookmark0) 4
2. [Задание](#_bookmark1) 5
3. [Теоретическое введение](#_bookmark2) 6
4. [Выполнение лабораторной работы](#_bookmark3) 9
   1. [Создание программы Hello world!](#_bookmark4) . . . . . . . . . . . . . . . . . . 9
   2. [Работа с транслятором NASM](#_bookmark9) 10
   3. [Работа с расширенным синтаксисом командной строки NASM](#_bookmark11) 10
   4. [Работа с компоновщиком LD](#_bookmark13) 11
   5. [Запуск исполняемого файла](#_bookmark16) 11
   6. [Выполнение заданий для самостоятельной работы.](#_bookmark18) 12
5. [Выводы](#_bookmark28) 15

**Список иллюстраций**

* 1. [Перемещение между директориями](#_bookmark5) . . . . . . . . . . . . . . . . 9
  2. [Создание пустого файла](#_bookmark6) . . . . . . . . . . . . . . . . . . . . . . . 9
  3. [Открытие файла в текстовом редакторе](#_bookmark7) . . . . . . . . . . . . . . 9
  4. [Заполнение файла](#_bookmark8) 10
  5. [Компиляция текста программы](#_bookmark10) 10
  6. [Компиляция текста программы](#_bookmark12) 11
  7. [Передача объектного файла на обработку компоновщику](#_bookmark14) 11
  8. [Передача объектного файла на обработку компоновщику](#_bookmark15) 11
  9. [Запуск исполняемого файла](#_bookmark17) 12
  10. [Создание копии файла](#_bookmark19) 12
  11. [Изменение программы](#_bookmark20) 12
  12. [Компиляция текста программы](#_bookmark21) 13
  13. [Передача объектного файла на обработку компоновщику](#_bookmark22) 13
  14. [Запуск исполняемого файла](#_bookmark23) 13
  15. [Создании копии файлов в новом каталоге](#_bookmark24) 14
  16. [Удаление лишних файлов в текущем каталоге](#_bookmark25) 14
  17. [Добавление файлов на GitHub](#_bookmark26) 14
  18. [Отправка файлов](#_bookmark27) 14

# Цель работы

Цель данной лабораторной работы - освоить процедуры компиляции и сборки программ, написанных на ассемблере NASM.

# 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.

# Теоретическое введение

Основными функциональными элементами любой ЭВМ являются централь- ный 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 про- водники 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 устрой- ство, которое напрямую взаимодействует с узлами процессора, предназначенное для хранения программ и данных, с которыми процессор непосредственно ра- ботает в текущий момент. ОЗУ состоит из одинаковых пронумерованных ячеек памяти. Номер ячейки памяти — это адрес хранящихся в ней данных. Периферий- ные устройства в составе ЭВМ: - устройства внешней памяти, которые предна- значены для долговременного хранения больших объёмов данных. - устройства ввода-вывода, которые обеспечивают взаимодействие ЦП с внешней средой.

В основе вычислительного процесса ЭВМ лежит принцип программного управ- ления. Это означает, что компьютер решает поставленную задачу как последова- тельность 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 необходи- мо 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 следу- ющем: 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.

# Выполнение лабораторной работы

## Создание программы Hello world!

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

[4.1).](#_bookmark5)

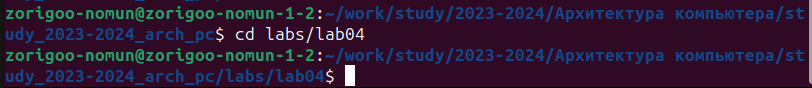


Рис. 4.1: Перемещение между директориями

Создаю в текущем каталоге пустой текстовый файл hello.asm с помощью ути- литы touch (рис. [4.2).](#_bookmark6)



Рис. 4.2: Создание пустого файла

Открываю созданный файл в текстовом редакторе mousepad (рис. [4.3).](#_bookmark7)

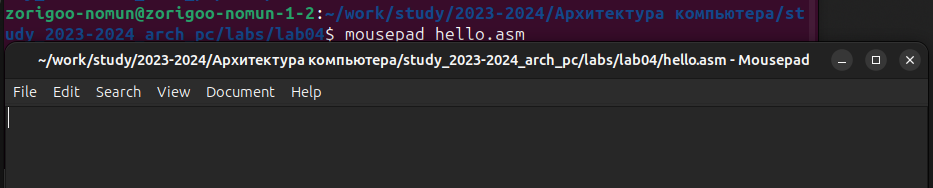


Рис. 4.3: Открытие файла в текстовом редакторе

Заполняю файл, вставляя в него программу для вывода “Hello word!” (рис. [4.4).](#_bookmark8)

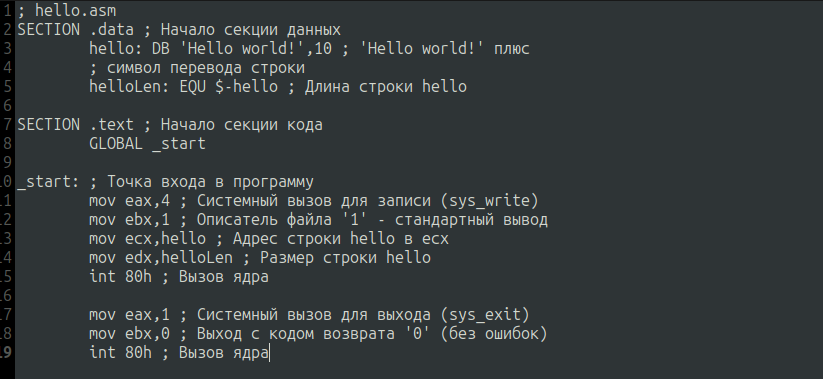


Рис. 4.4: Заполнение файла

## 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 (рис. [4.5).](#_bookmark10) Далее проверяю правильность выполнения команды с помощью утилиты ls: действительно, создан файл “hello.o”.

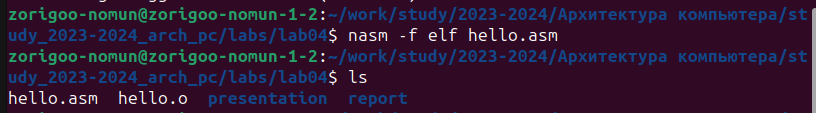


Рис. 4.5: Компиляция текста программы

## 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 (рис. [4.6).](#_bookmark12) Далее проверяю с помощью утилиты ls правильность выполнения команды.

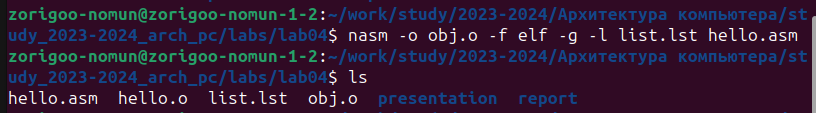


Рис. 4.6: Компиляция текста программы

## Работа с компоновщиком LD

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы по- лучить исполняемый файл hello (рис. [4.7).](#_bookmark14) Ключ -о задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты ls правильность вы- полнения команды.

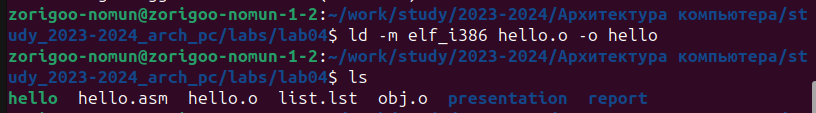


Рис. 4.7: Передача объектного файла на обработку компоновщику Выполняю следующую команду (рис. [4.8).](#_bookmark15) Исполняемый файл будет иметь

имя main, т.к. после ключа -о было задано значение main. Объектный файл, из которого собран этот исполняемый файл, имеет имя obj.o

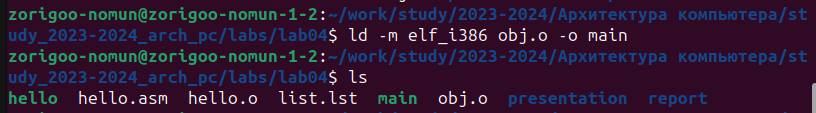


Рис. 4.8: Передача объектного файла на обработку компоновщику

## Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello (рис. [4.9).](#_bookmark17)

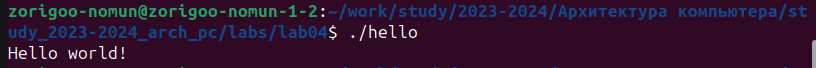


Рис. 4.9: Запуск исполняемого файла

## Выполнение заданий для самостоятельной работы.

С помощью утилиты cp создаю в текущем каталоге копию файла hello.asm с именем lab4.asm (рис. [4.10).](#_bookmark19)

Рис. 4.10: Создание копии файла

С помощью текстового редактора mousepad открываю файл lab4.asm и вношу изменения в программу так, чтобы она выводила мои имя и фамилию. (рис. [4.11).](#_bookmark20)

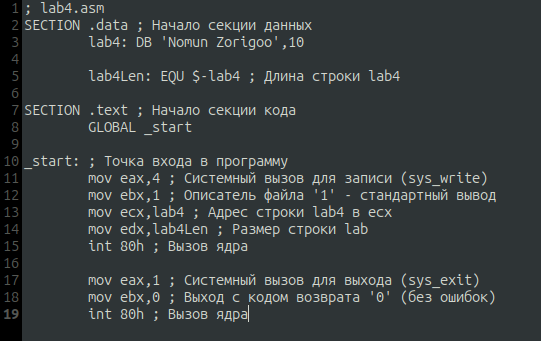


Рис. 4.11: Изменение программы

Компилирую текст программы в объектный файл (рис. [4.12).](#_bookmark21) Проверяю с помо- щью утилиты ls, что файл lab4.o создан.

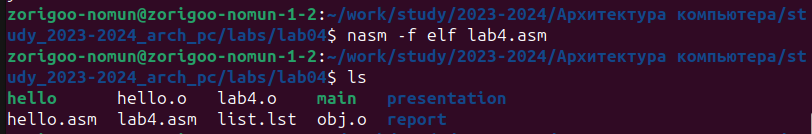


Рис. 4.12: Компиляция текста программы

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы полу- чить исполняемый файл lab4 (рис. [4.13).](#_bookmark22)

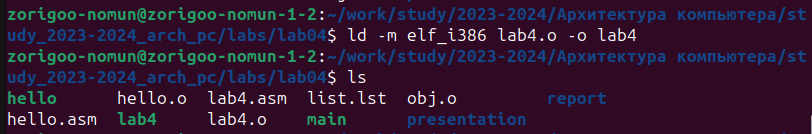


Рис. 4.13: Передача объектного файла на обработку компоновщику Запускаю исполняемый файл lab4, на экран действительно выводятся мои имя

и фамилия (рис. [4.14).](#_bookmark23)

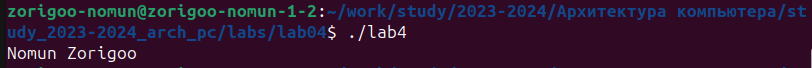


Рис. 4.14: Запуск исполняемого файла

К сожалению, я начала работу не в том каталоге, поэтому создаю другую дирек- торию lab04 с помощью mkdir, прописывая полный путь к каталогу, в котором хочу создать эту директорию. Далее копирую из текущего каталога файлы, со- зданные в процессе выполнения лабораторной работы, с помощью утилиты cp, указывая вместо имени файла символ \*, чтобы скопировать все файлы. Команда проигнорирует директории в этом каталоге, т. к. не указан ключ -r, это мне и нужно (рис. [4.15).](#_bookmark24) Проверяю с помощью утилиты ls правильность выполнения команды.

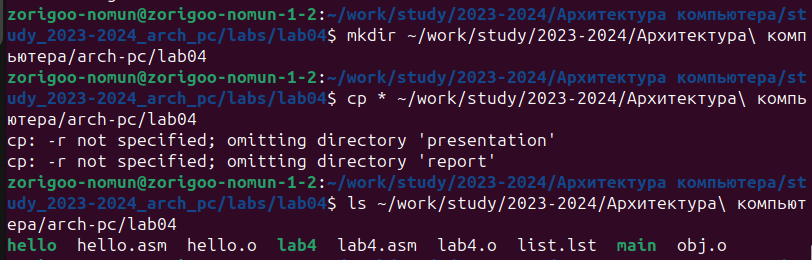


Рис. 4.15: Создании копии файлов в новом каталоге

Удаляю лишние файлы в текущем каталоге с помощью утилиты rm, ведь копии файлов остались в другой директории (рис. [4.16).](#_bookmark25)

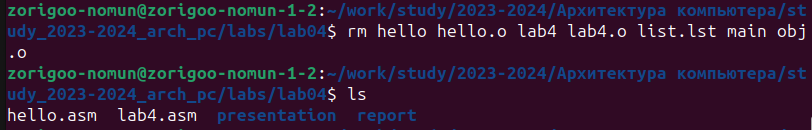


Рис. 4.16: Удаление лишних файлов в текущем каталоге

С помощью команд git add . и git commit добавляю файлы на GitHub, ком- ментируя действие как добавление файлов для лабораторной работы №4 (рис. [4.17).](#_bookmark26)

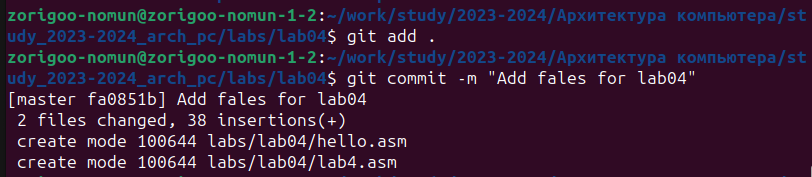


Рис. 4.17: Добавление файлов на GitHub

Отправляю файлы на сервер с помощью команды git push (рис. [4.18).](#_bookmark27)

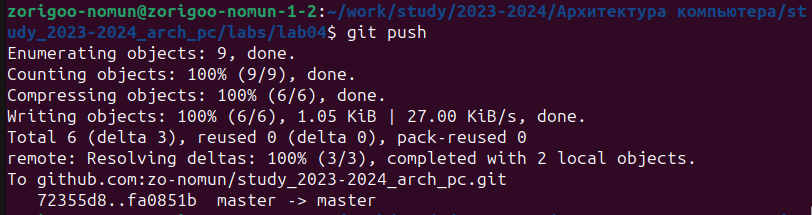


Рис. 4.18: Отправка файлов

# Выводы

При выполнении данной лабораторной работы я освоила процедуры компиля- ции и сборки программ, написанных на ассемблере N