# Отчёт по лабораторной работе №4

**Дисциплина: архитектура компьютеров и операционные системы**

Хань Цзянтао

# Содержание

1. [Цель работы](#_bookmark0) 5
2. [Задание](#_bookmark1) 6
3. [Теоретическое введение](#_bookmark2) 7
4. [Выполнение лабораторной работы](#_bookmark3) 9

[4.1 Программа Hello world!](#_bookmark4) . . . . . . . . . . . . . . . . . . . . . . 9

* 1. [Транслятор NASM](#_bookmark7) 10
  2. [Расширенный синтаксис командной строки NASM](#_bookmark9) 10
  3. [Компоновщик LD](#_bookmark11) 10
  4. [Запуск исполняемого файла](#_bookmark14) 11
  5. [Задание для самостоятельной работы](#_bookmark16) 11

1. [Выводы](#_bookmark22) 14
2. [Список литературы](#_bookmark23) 15

**Список иллюстраций**

[4.1 Переход в каталог](#_bookmark5) . . . . . . . . . . . . . . . . . . . . . . . . . . 9

* 1. [Ввод данного текста в файл](#_bookmark6) . . . . . . . . . . . . . . . . . . . . . 9
  2. [Компиляция текста с помощью команды](#_bookmark8) 10
  3. [Компиляция файла с помощью команды](#_bookmark10) 10
  4. [Получение исполняемой программы](#_bookmark12) 10
  5. [Создание исполняемого файла](#_bookmark13) 11
  6. [Запуск исполняемого файла с помощью команды](#_bookmark15) 11
  7. [Копирование файла](#_bookmark17) 11
  8. [Изменение файла с заданными условиями](#_bookmark18) 12
  9. [Изменение файла с заданными условиями](#_bookmark19) 12
  10. [Копирование файлов в каталог](#_bookmark20) 13
  11. [Загрузка файлов на Github](#_bookmark21) 13

# Список таблиц

# Цель работы

Целью данной лабораторной работы является освоение процедуры компиля- ции и сборки программ, написанных на ассемблере NASM.

# Задание

* 1. Программа Hello world!
  2. Транслятор NASM
  3. Расширенный синтаксис командной строки NASM
  4. Компоновщик LD
  5. Запуск исполняемого файла
  6. Задание для самостоятельной работы

# Теоретическое введение

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 качестве операндов. Практически все команды представляют собой преобразование дан- ных 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 (половинки 16-битных регистров). Напри- мер, AH (high AX) — старшие 8 бит регистра AX, AL (low AX) — младшие 8 бит регистра AX. Язык ассемблера (assembly language, сокращённо asm) — машинно- ориентированный язык низкого уровня. Можно считать, что он больше любых других языков приближен к архитектуре ЭВМ и её аппаратным возможностям, что позволяет получить к ним более полный доступ, нежели в языках высокого уровня,таких как C/C++, Perl, Python и пр. Заметим, что получить полный до- ступ к ресурсам компьютера в современных архитектурах нельзя, самым низким уровнем работы прикладной программы является обращение напрямую к ядру

операционной системы. Именно на этом уровне и работают программы, напи- санные на ассемблере. Но в отличие от языков высокого уровня ассемблерная программа содержит только тот код, который ввёл программист. Таким обра- зом язык ассемблера — это язык, с помощью которого понятным для человека образом пишутся команды для процессора. Следует отметить, что процессор понимает не команды ассемблера, а последовательности из нулей и единиц — машинные коды. До появления языков ассемблера программистам приходи- лось писать программы, используя только лишь машинные коды, которые были крайне сложны для запоминания, так как представляли собой числа, записан- ные в двоичной или шестнадцатеричной системе счисления. Преобразование или трансляция команд с языка ассемблера в исполняемый машинный код осу- ществляется специальной программой транслятором — Ассемблер. NASM — это открытый проект ассемблера, версии которого доступны под различные операци- онные системы и который позволяет получать объектные файлы для этих систем. В NASM используется Intel-синтаксис и поддерживаются инструкции x86-64. Про- грамма на языке ассемблера также может содержать директивы — инструкции, не переводящиеся непосредственно в машинные команды, а управляющие рабо- той транслятора. Например, директивы используются для определения данных (констант и переменных) и обычно пишутся большими буквами.

# Выполнение лабораторной работы

## Программа Hello world!

Создаю каталог для работы с программами на языке ассемблера NASM и пере- хожу в созданный каталог. (рис. [4.12)](#_bookmark21)



Рис. 4.1: Переход в каталог

Создаю текстовый файл с именем hello.asm, открываю этот файл с помощью текстового редактора и ввожу следующий текст. (рис. [4.12)](#_bookmark21)

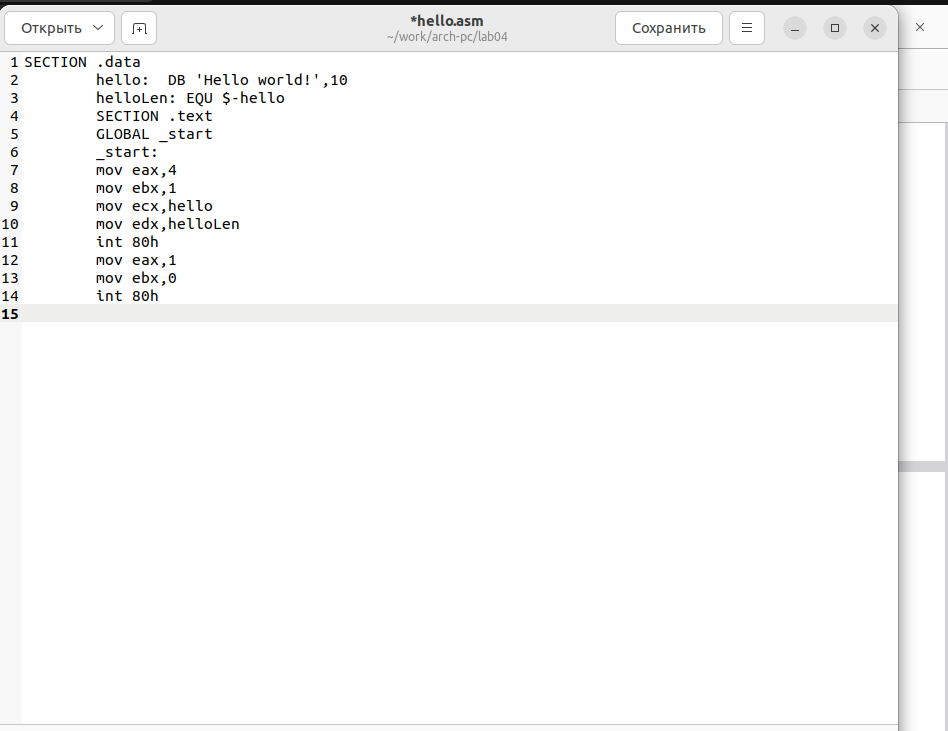


Рис. 4.2: Ввод данного текста в файл

## Транслятор NASM

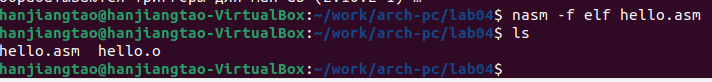
Компилируем приведённый выше текст программы при помощи команды nasm -f elf hello.asm «Hello World» и проверим, что файл создан. (рис. [4.12)](#_bookmark21)

Рис. 4.3: Компиляция текста с помощью команды

## Расширенный синтаксис командной строки NASM

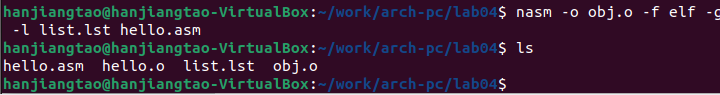
С помощью команды nasm -o obj.o -f elf -g -l list.lst hello.asm скомпилируем исходный файл hello.asm в obj.o и проверим, что файл создан. (рис. [4.12)](#_bookmark21)

Рис. 4.4: Компиляция файла с помощью команды

## Компоновщик LD

Передаем объектный файл на обработку компоновщику с помощью команды ld

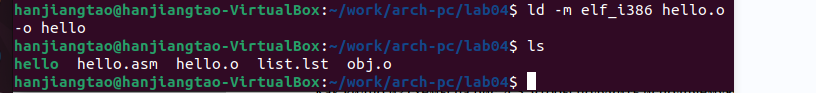
-m elf\_i386 hello.o -o hello и проверяем, что исполняемый файл hello был создан. (рис. [4.12)](#_bookmark21)

Рис. 4.5: Получение исполняемой программы

Создадим еще один файл с помощью команды ld -m elf\_i386 obj.o -o main. (рис.

[4.12)](#_bookmark21)

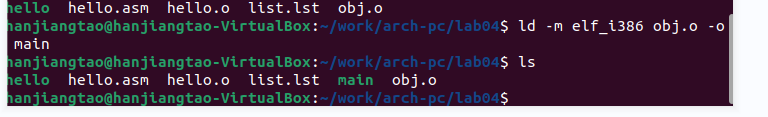


Рис. 4.6: Создание исполняемого файла

Имя исполняемого файа - main, имя объектного файла - obj.o

## Запуск исполняемого файла

Запустим созданный исполняемый файл с помощью команды ./hello. (рис. [4.12)](#_bookmark21)

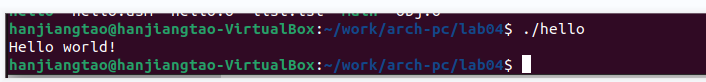


Рис. 4.7: Запуск исполняемого файла с помощью команды

## Задание для самостоятельной работы

* + 1. В каталоге ~/work/arch-pc/lab04 с помощью команды cp создаю копию файла hello.asm с именем lab4.asm. (рис. [4.12)](#_bookmark21)

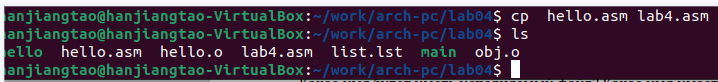


Рис. 4.8: Копирование файла

* + 1. С помощью 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. [4.12)](#_bookmark21)

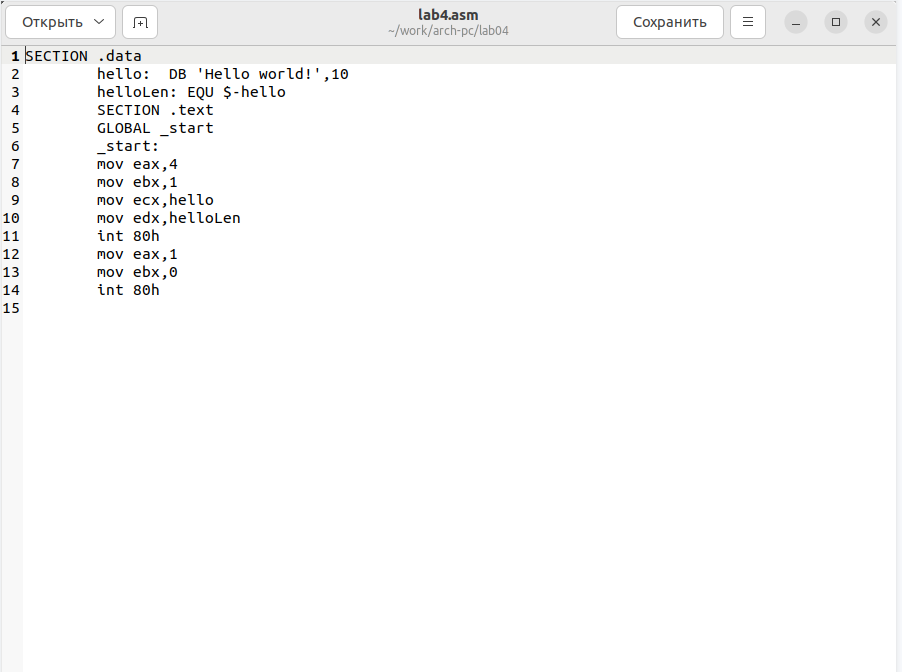


Рис. 4.9: Изменение файла с заданными условиями

* + 1. Компилирую полученный текст программы lab4.asm в объектный файл. Выполняю компоновку объектного файла и запускаю получившийся испол- няемый файл. (рис. [4.12)](#_bookmark21)

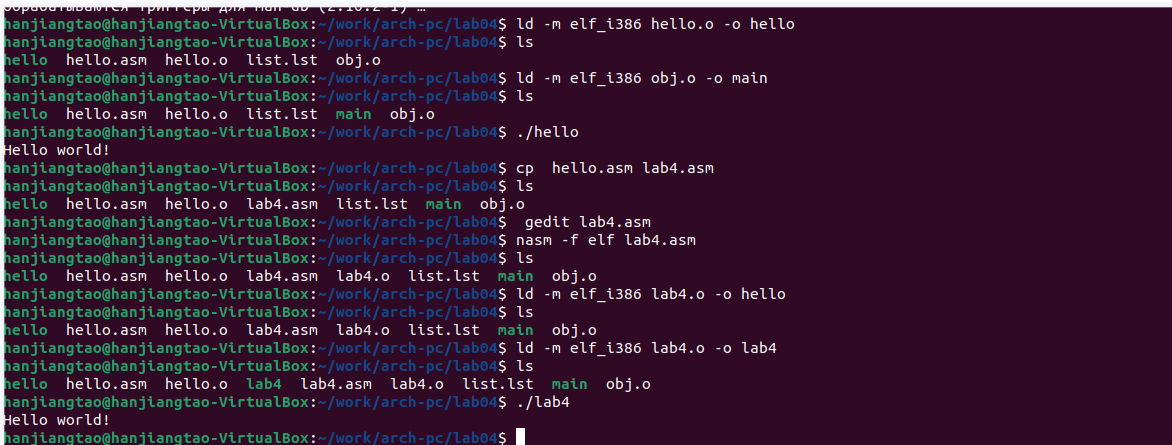


Рис. 4.10: Изменение файла с заданными условиями

* + 1. Копирую файлы hello.asm и lab4.asm в локальный репозиторий в каталог

~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/. (рис. [4.12)](#_bookmark21)

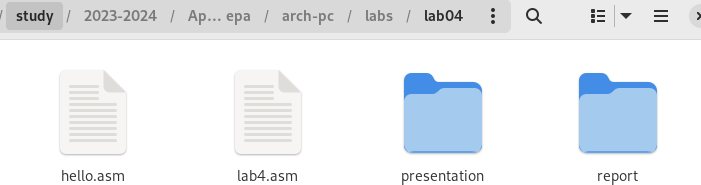


Рис. 4.11: Копирование файлов в каталог

Загружаю файлы на Github. (рис. [4.12)](#_bookmark21)

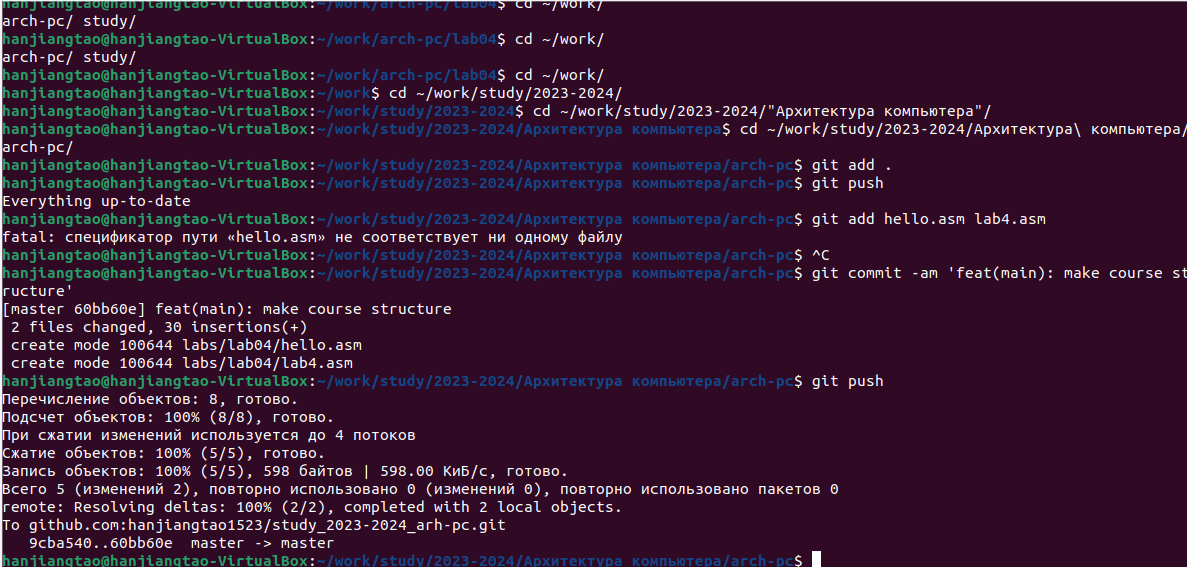


Рис. 4.12: Загрузка файлов на Github

# Выводы

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

# Список литературы

* 1. GDB: The GNU Project Debugger. — URL: ht[tps://w](http://www.gnu.org/software/gdb/)ww[.gnu.org/software/gdb/.](http://www.gnu.org/software/gdb/)
  2. GNU Bash Manual. — 2016. — URL: ht[tps://w](http://www.gnu.org/software/bash/manual/)ww[.gnu.org/software/bash/manual/.](http://www.gnu.org/software/bash/manual/)
  3. Midnight Commander Development Center. — 2021. — URL: https://midnight- commander. org/.
  4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.
  5. Newham C. Learning the bash Shell: Unix Shell Programming. — O’Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: [http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658.](http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658)
  6. Robbins A. Bash Pocket Reference. — O’Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
  7. The NASM documentation. — 2021. — URL: ht[tps://w](http://www.nasm.us/docs.php)ww[.nasm.us/docs.php.](http://www.nasm.us/docs.php)
  8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
  9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
  10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
  11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
  12. Расширенный ассемблер: NASM.— 2021.— URL: [https://w](http://www.opennet.ru/docs/RUS/nasm/)ww[.opennet.ru/docs/RUS/nasm/.](http://www.opennet.ru/docs/RUS/nasm/)
  13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
  14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix.— 2-

е изд.— М.: МАКС Пресс, 2011.— URL: [http://www.stolyarov.info/books/asm\_unix.](http://www.stolyarov.info/books/asm_unix)

* 1. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
  2. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер 2015. — 1120 с. — (Классика Computer Science).