Отчет по лабораторной работе №5

Создание и процесс обработки программ на языке ассемблера NASM

Виктор Максимович Кадров

Содержание

# 1 Цель работы

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

# 2 Задание

1. Написание программ Hello World!, ее трансляция, компоновка и выполнение.
2. Выполнение заданий для самостоятельной работы.
3. Ответы на вопросы для самопроверки.

# 3 Теоретическое введение

Язык ассемблера (assembly language, сокращённо asm) — машинноориентированный 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. Программы, написанные на языке ассемблера, не уступают в качестве и скорости программам, написанным на машинном языке, так как транслятор просто переводит мнемонические обозначения команд в последовательности бит (нулей и единиц). Используемые мнемоники обычно одинаковы для всех процессоров одной архитектуры или семейства архитектур (среди широко известных — мнемоники процессоров и контроллеров x86, ARM, SPARC, PowerPC,M68k). Таким образом для каждой архитектуры существует свой ассемблер и, соответственно, свой язык ассемблера.

# 4 Выполнение лабораторной работы

Создаем папку lab05 и файл hello.asm в ней (рис. 1). После этого проверяем, что файл действительно был создан (рис. 2). Пишем код программы в созданном файле (рис. 3). Транслируем его при помощи NASM в объектный файл (hello.o) и проверяем наличие (рис. 4). Выплоняем команду с дополнительными ключами (рис. 5). Компонуем и получаем на выходе исполняемый файл (рис. 6), выполняем команду с заданием другого имени (obj.o main) (рис. 7). Исполняем его (рис. 8).

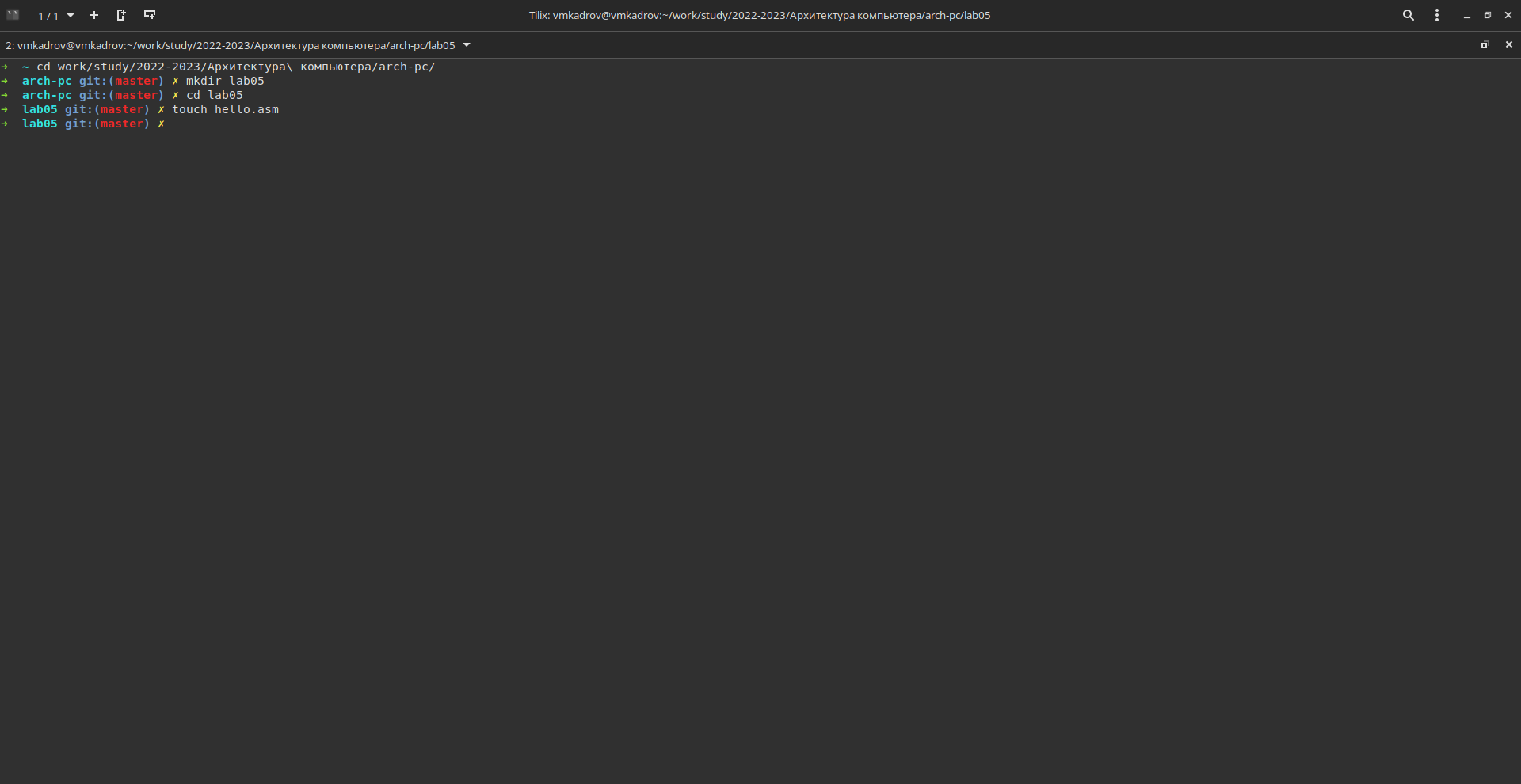


Рис. 1: Создание папки и файла

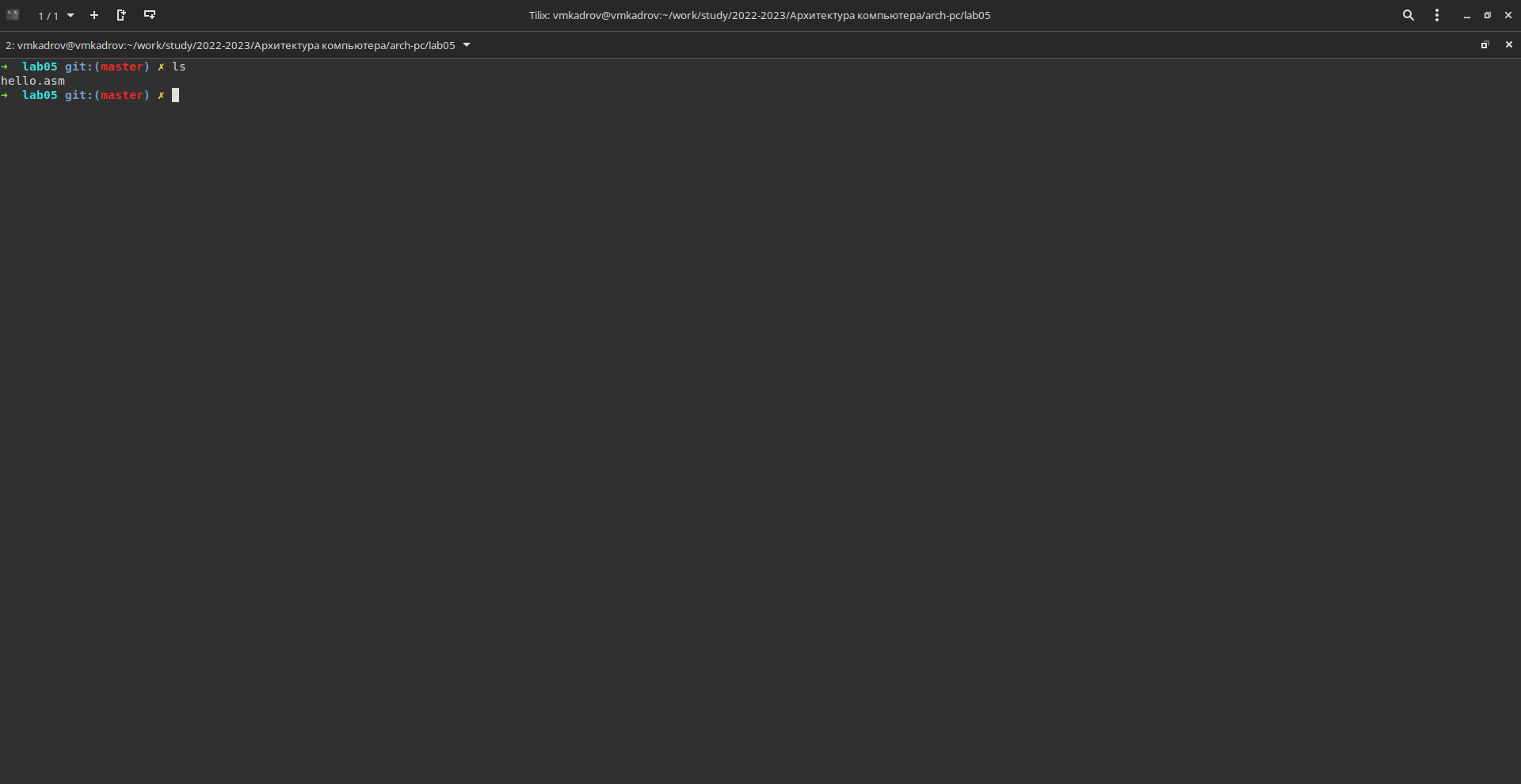


Рис. 2: Проверка с помощью ls

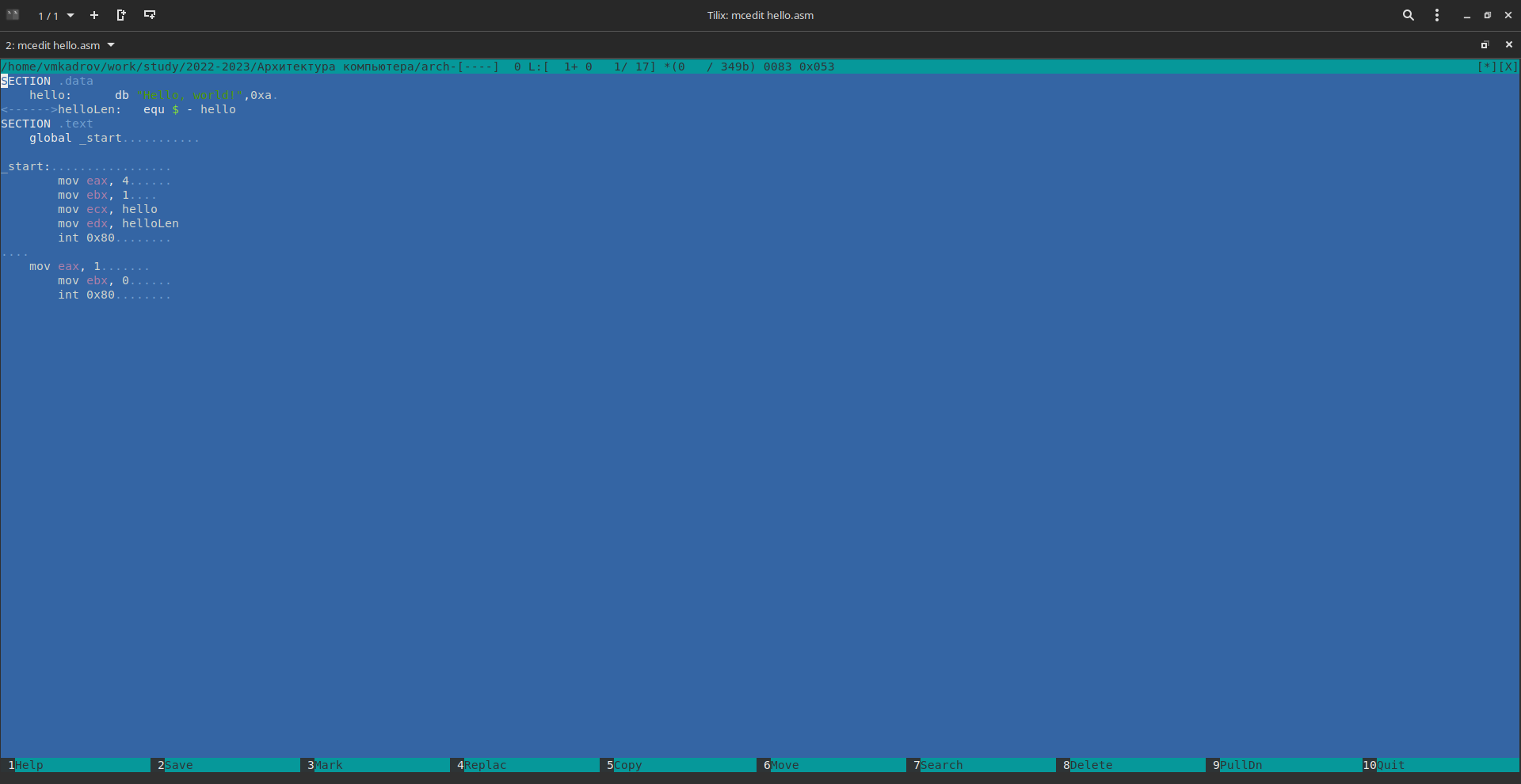


Рис. 3: Код программы

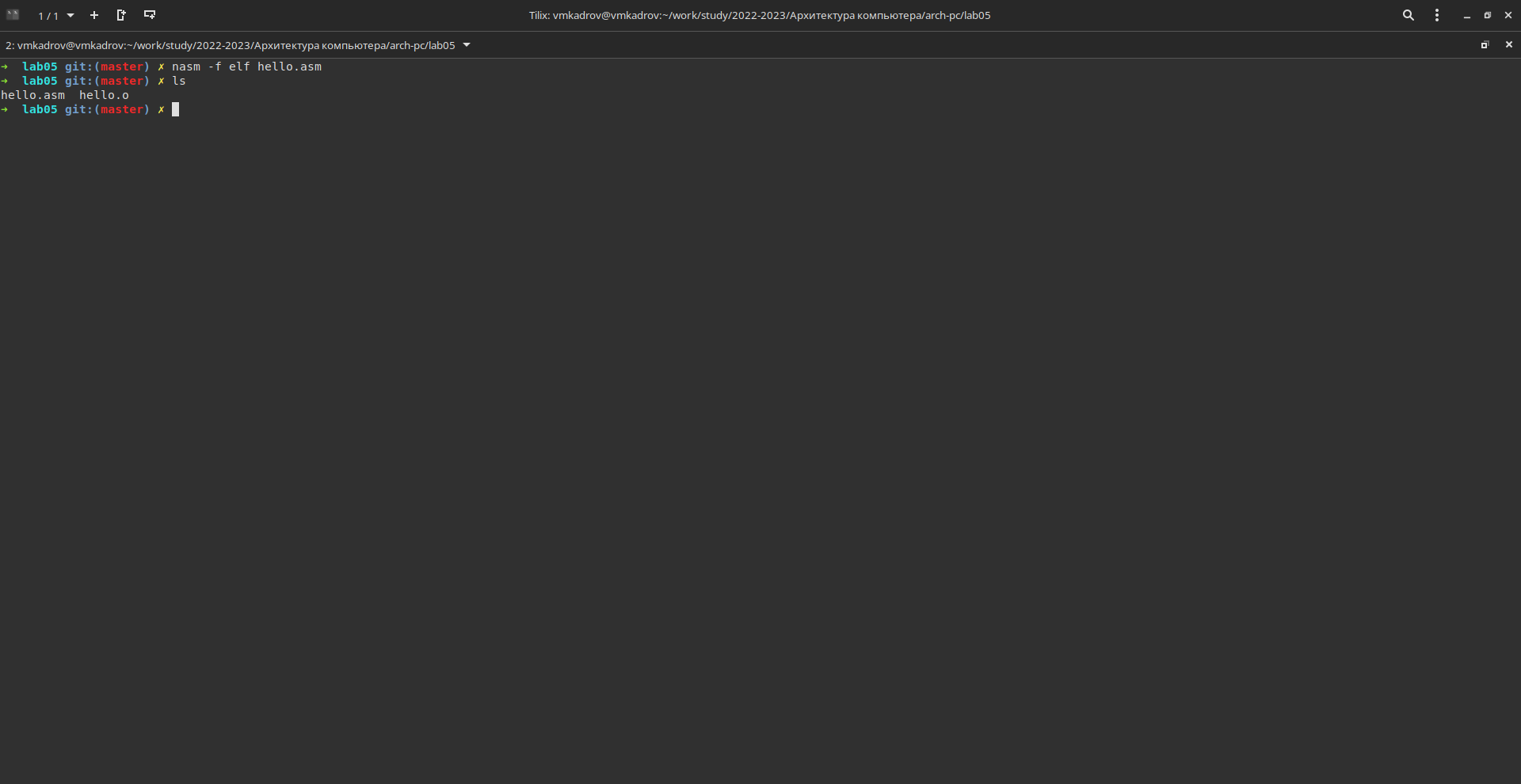


Рис. 4: Вызов транслятора nasm

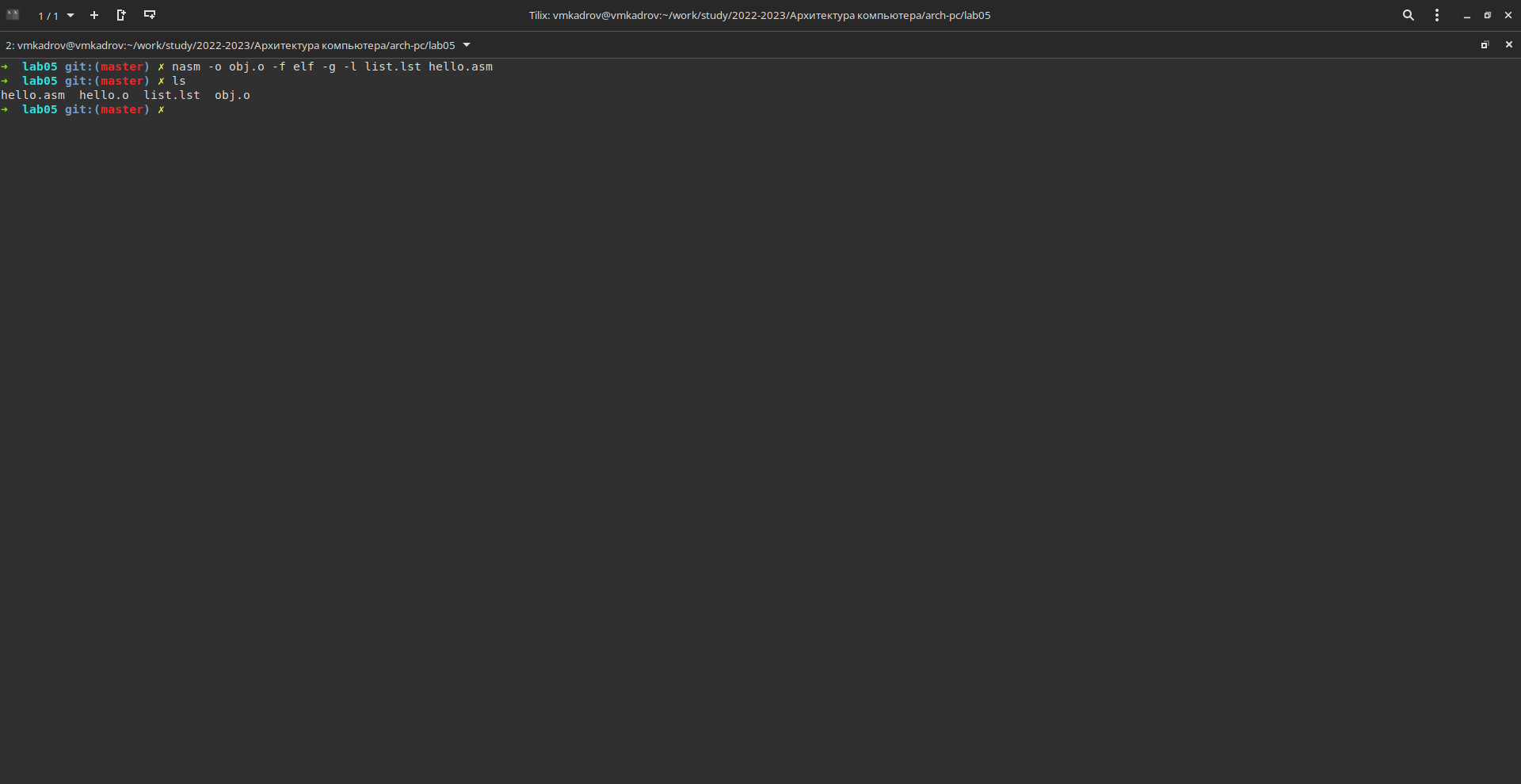


Рис. 5: Nasm с различными ключами

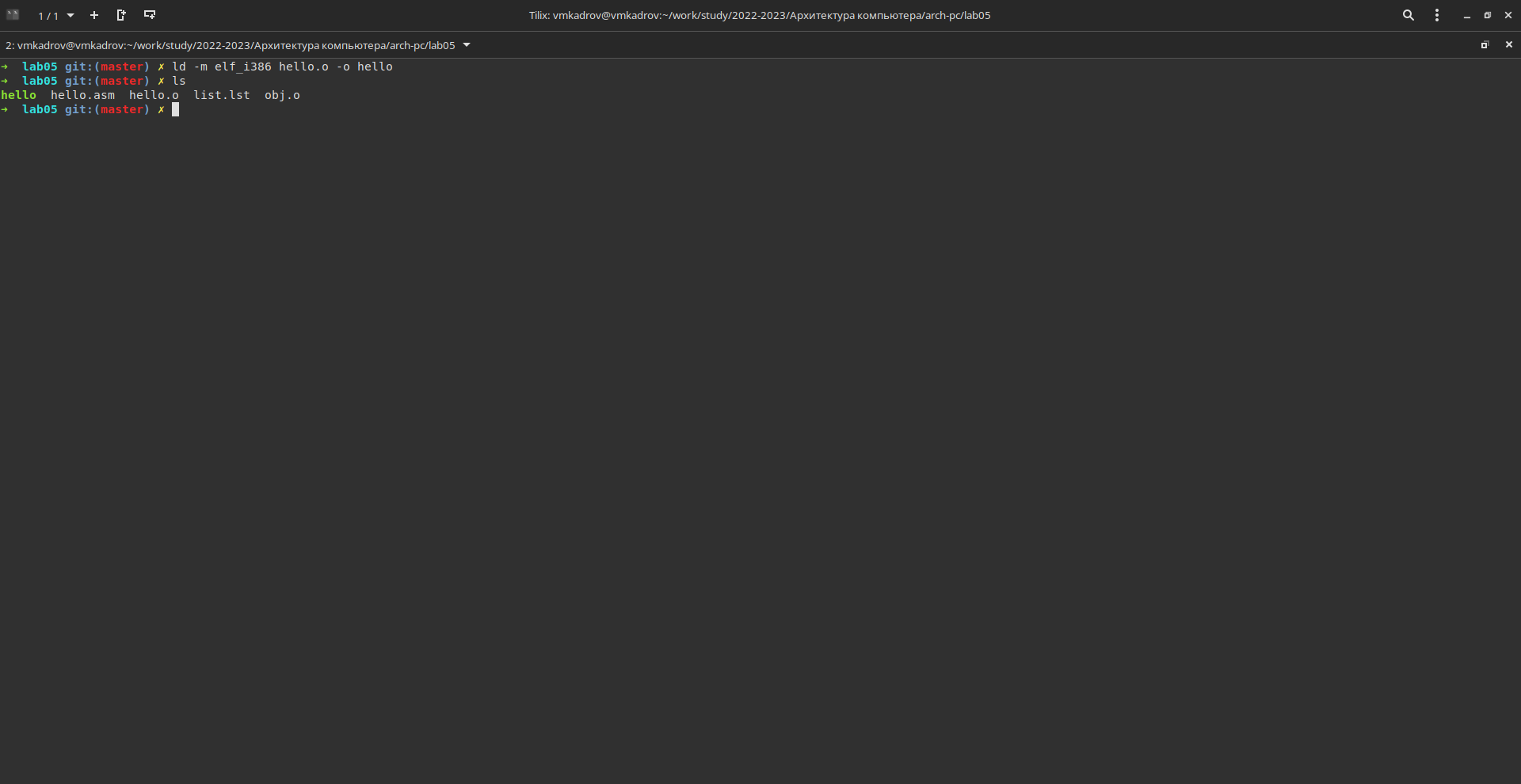


Рис. 6: Компоновка

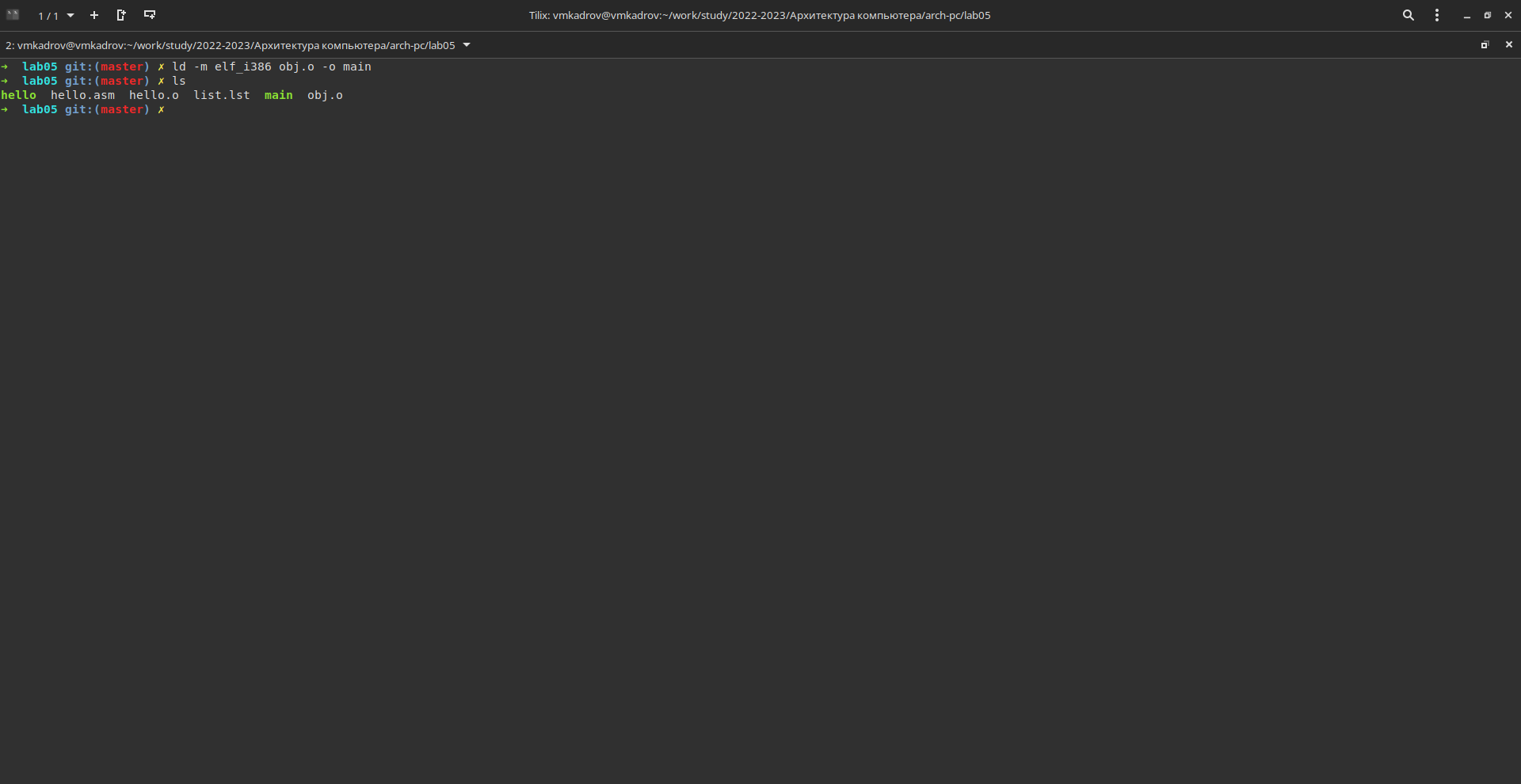


Рис. 7: Задание другого имени

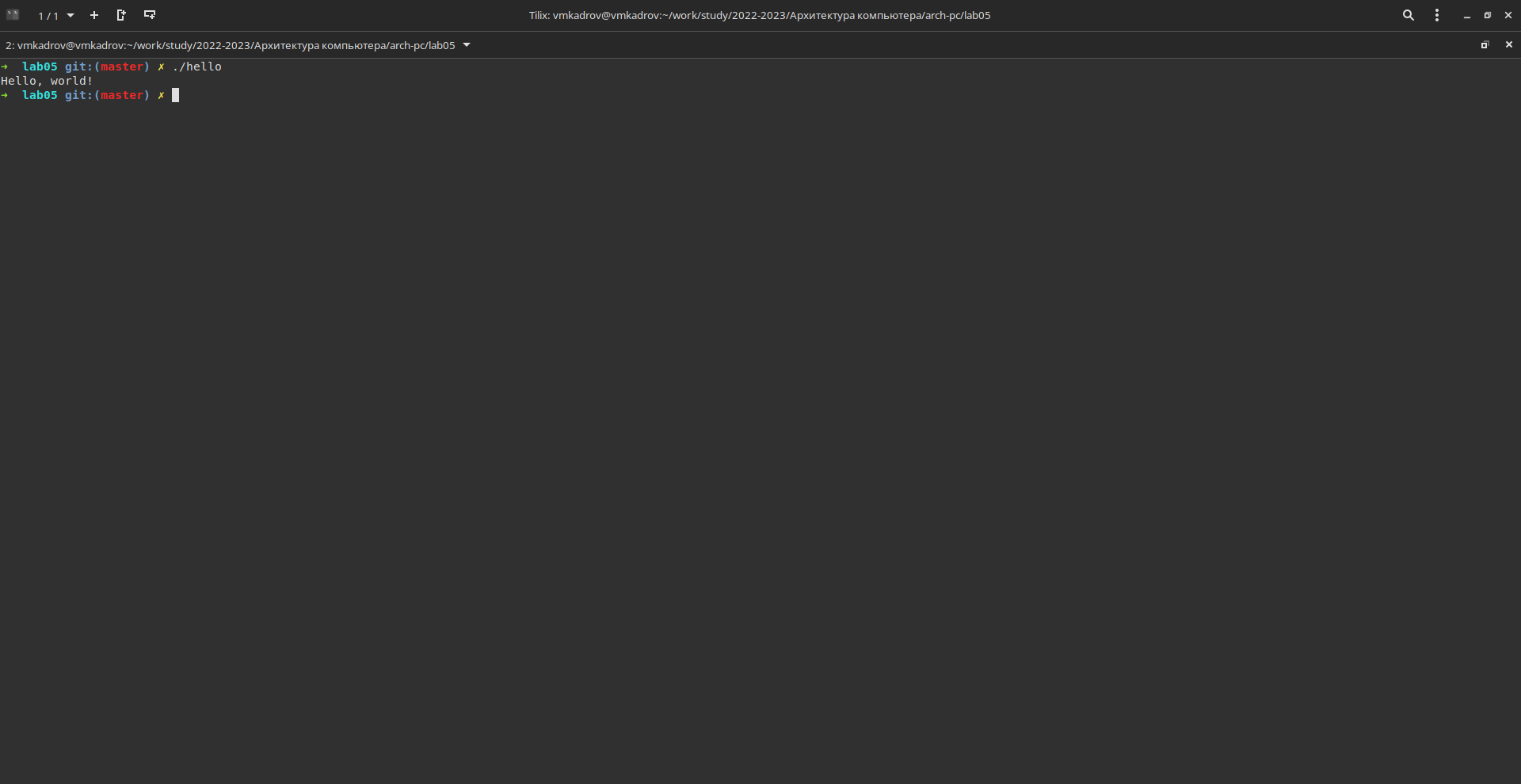


Рис. 8: Выполнение программы

# 5 Задания для самостоятельной работы

Изменяем код програмы в копии файла (рис. 9). Транслируем, компонуем и проверяем, что программа выполняется корректно (рис. 10). Копируем файлы в папку лаб. работы (рис. 11). После заполнения отчета загружаем файлы на github.

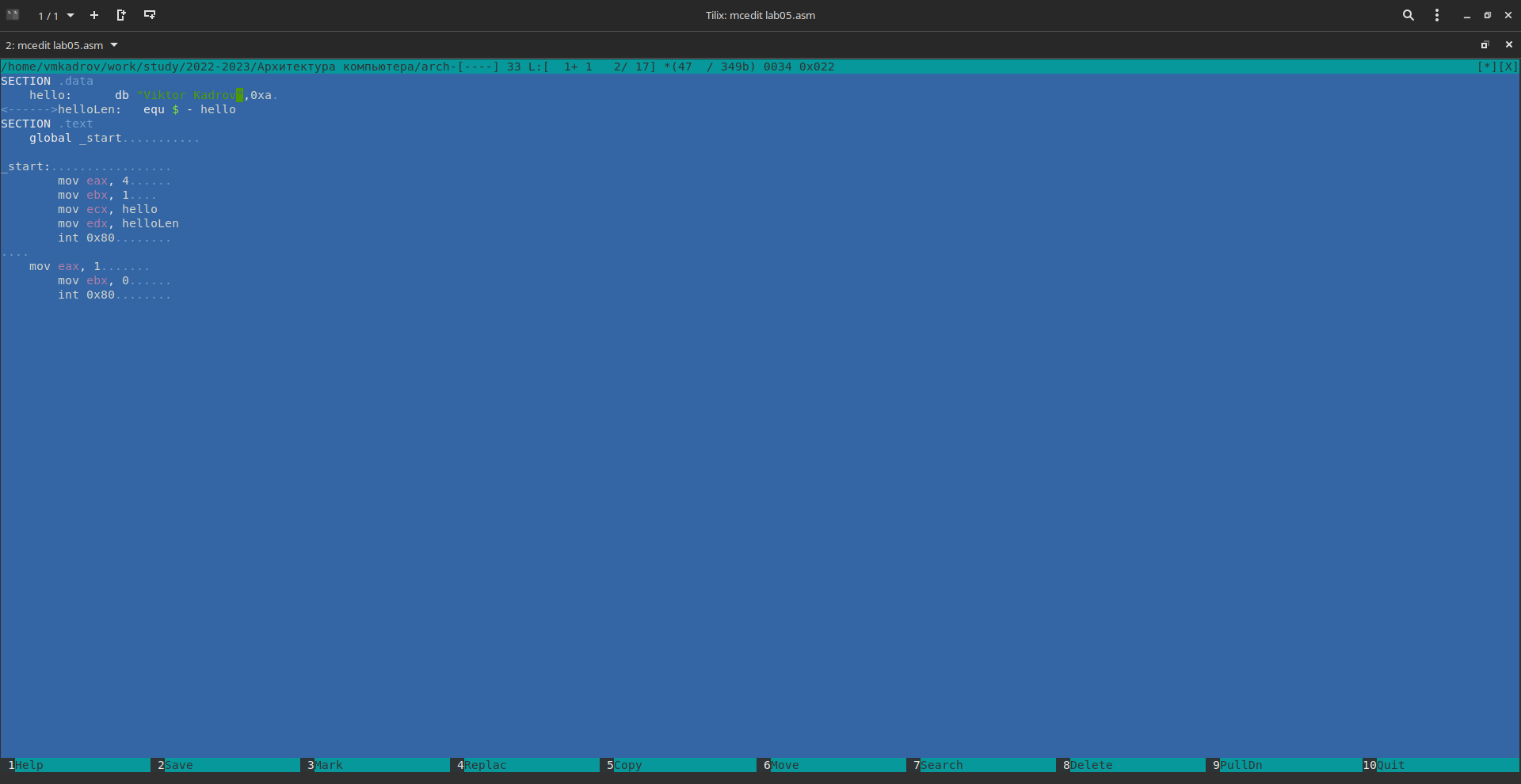


Рис. 9: Измененный код

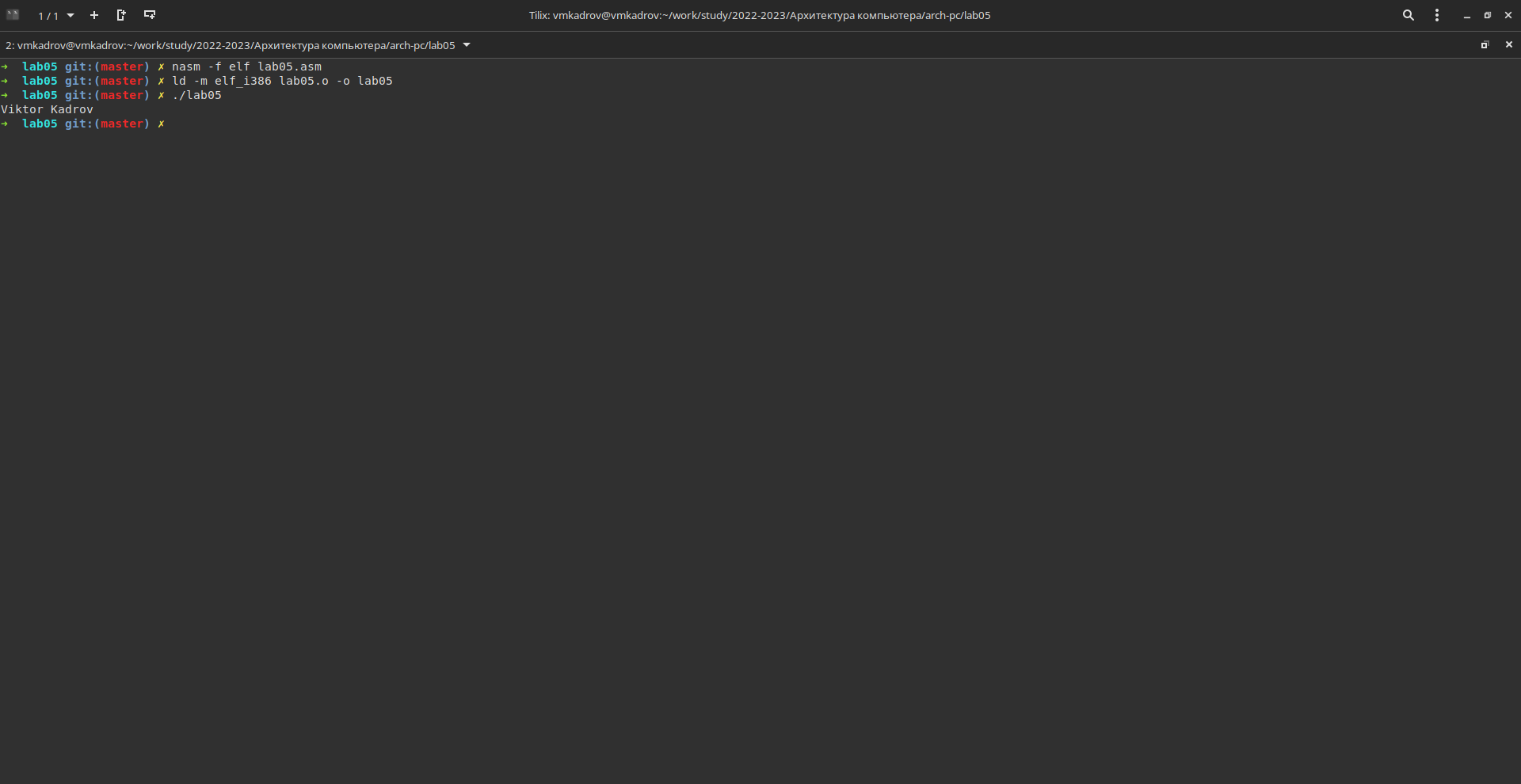


Рис. 10: Выполнение программы

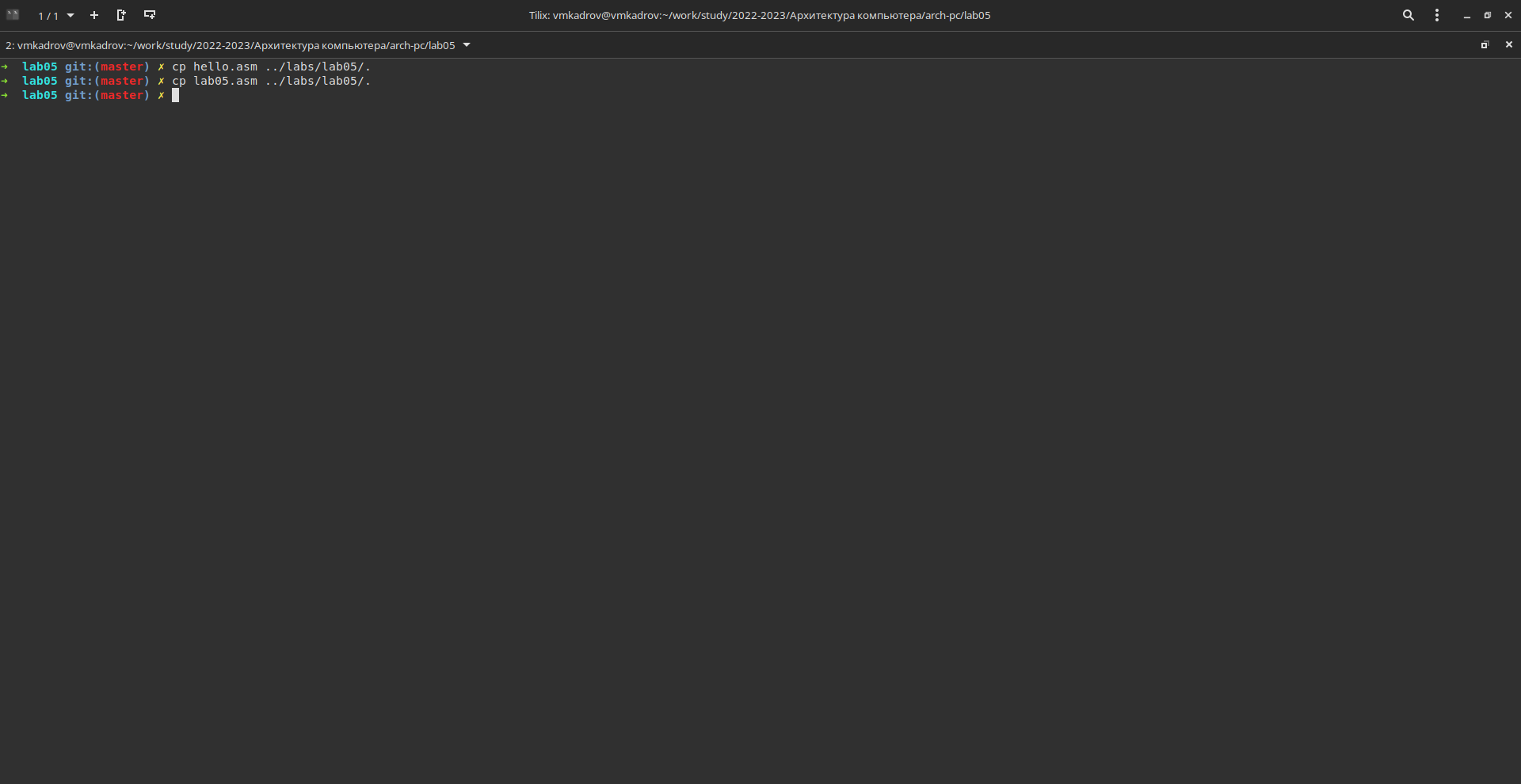


Рис. 11: Копирование файлов

# 6 Ответы на вопросы для самопроверки

1. Язык уровня ассемблера: язык низкого уровня, который позволяет пользователям писать программы, используя буквенно-цифровые мнемонические коды вместо числового кода для набора инструкций.  
   Язык высокого уровня: машинно-независимый язык. Он позволяет пользователю писать программы на языке, который напоминает английские слова и знакомые математические символы. Примерами языков высокого уровня являются python, c# и др.  
   Язык ассемблера (assembly language, сокращенно 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 доступ 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, написанные на ассемблере. Но в отличие от языков высокого уровня ассемблерная программа содержит только тот код, который ввел программист. Таким образом язык ассемблера — это язык, с помощью которого понятным для человека образом пишутся команды для процессора.
2. Программа на языке ассемблера также может содержать директивы — инструкции, не переводящиеся непосредственно в машинные команды, а управляющие работой транслятора. Например, директивы используются для определения данных (констант и переменных) и обычно пишутся большими буквами.
3. Правила оформления ассемблерных программ При наборе программ на языке ассемблера придерживайтесь следующих правил:

* директивы набираются большими буквами, инструкции – малыми;
* текст пишется широко;
* не стоит выходить за край экрана – его неудобно будет редактировать и печатать;
* для отступов используется табуляция (клавиша TAB);
* блоки комментариев задаются с одинаковым отступом. Оптимальной считается такая строка: moveax,ebx<(1-3)TAB>;текст комментария Количество табуляций перед комментарием определяется длиной аргументов команды и может быть от 1 до 3.

1. Трансляция и компоновка.
2. Трансляция исходного текста программы состоит в преобразовании предложений исходного языка в коды машинных команд и выполняется с помощью транслятора с языка ассемблера (т. е. с помощью программы ассемблера).
3. Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику.
4. После того, как текст программы набран и записан на жесткий диск, необходимо произвести трансляцию программы. В процессе трансляции создается результирующий (объектный) файл, который представляет собой ту же программу, но в машинных кодах, предназначенную для записи в программную память микроконтроллера. Результирующий файл имеет расширение hex. Кроме hex-файла транслятор создает еще несколько вспомогательных файлов. И главное, файл с расширением еер. Этот файл имеет точно такую же внутреннюю структуру, как файл hex. А содержит он информацию, предназначенную для записи в EEPROM. Такая информация появляется в том случае, когда в тексте программы переменным, размещенным в сегменте eeprom, присвоены начальные значения. В обоих случаях (hex и eep форматы) применяется так называемый НЕХ-формат, который практически является стандартом для записи результатов транслирования различных программ. Он поддерживается практически всеми трансляторами с любого языка программирования.
5. NASM поддерживает множество форматов выходных файлов, среди них:

* bin
* obj
* win32 и win64
* aout
* aoutb
* coff
* elf32 и elf64 Компоновщик ld не предполагает по умолчанию расширений для файлов. Но принято использовать следующие расширения:
* o для объектных файлов;
* без расширения для исполняемых файлов;
* map для файлов схемы программы;
* lib для библиотек.

# 7 Выводы

В ходе выполнения лабораторной работы были освоены процедуры компиляции и сборки программ, написанных на ассемблере NASM.