

Отчёт по лабораторной работе №4

дисциплина: Архитектура компьютера

Мазурский Александр Дмитриевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
4.1	Программа Hello world!	10
4.2	Транслятор NASM	11
4.3	Расширенный синтаксис командной строки NASM	12
4.4	Компоновщик LD	13
4.5	Запуск исполняемого файла	14
4.6	Задания для самостоятельной работы	14
5	Выводы	17
6	Список литературы	18

Список иллюстраций

4.1	Создание рабочей директории	10
4.2	Создание .asm файла	11
4.3	Редактирование файла	11
4.4	Компиляция программы	12
4.5	Возможности синтаксиса NASM	12
4.6	Отправка файла компоновщику	13
4.7	Создание исполняемого файла	13
4.8	Запуск программы	14
4.9	Создание копии	14
4.10	Редактирование копии	15
4.11	Проверка работоспособности скомпонованной программы	15
4.12	Отправка файлов в локальный репозиторий	16
4.13	Загрузка изменений	16

Список таблиц

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 качестве 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: 1. формирование адреса в памяти очередной команды; 2. считывание кода команды из памяти и её дешифрация; 3. выполнение команды; 4. переход к

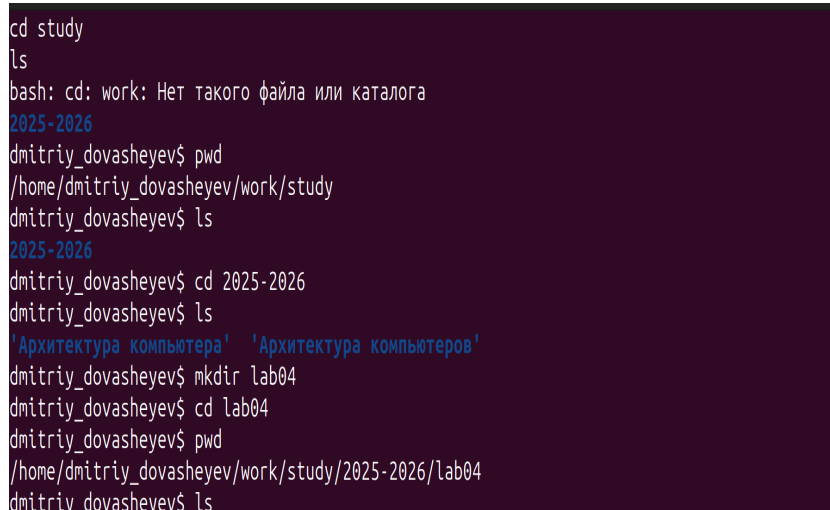
следующей команде.

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.

4 Выполнение лабораторной работы

4.1 Программа Hello world!

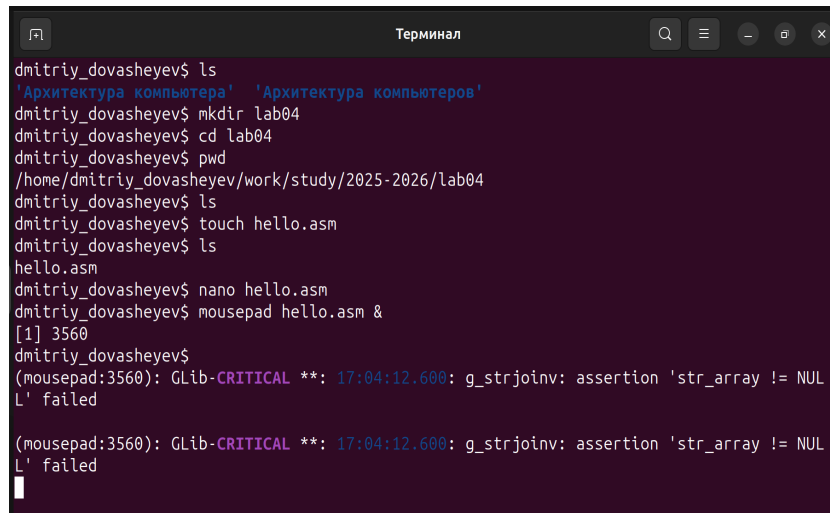
В домашней директории создаю каталог, в котором буду хранить файлы для текущей лабораторной работы. (рис. 4.1)



```
cd study
ls
bash: cd: work: Нет такого файла или каталога
2025-2026
dmitriy_dovasheyev$ pwd
/home/dmitriy_dovasheyev/work/study
dmitriy_dovasheyev$ ls
2025-2026
dmitriy_dovasheyev$ cd 2025-2026
dmitriy_dovasheyev$ ls
'Архитектура компьютера' 'Архитектура компьютеров'
dmitriy_dovasheyev$ mkdir lab04
dmitriy_dovasheyev$ cd lab04
dmitriy_dovasheyev$ pwd
/home/dmitriy_dovasheyev/work/study/2025-2026/lab04
dmitriy_dovasheyev$ ls
```

Рис. 4.1: Создание рабочей директории

Создаю в нем файл `hello.asm`, в котором буду писать программу на языке ассемблера. (рис. 4.2)

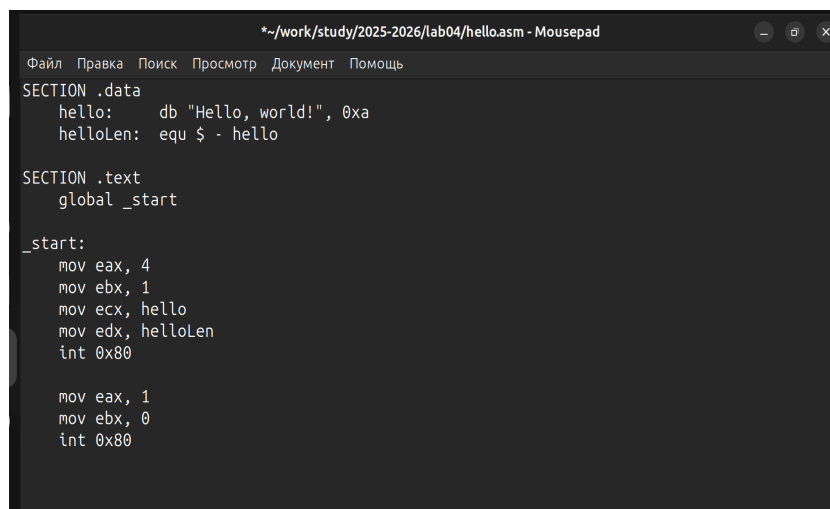


```
Терминал
dmitriy_dovasheyev$ ls
'Архитектура компьютера' 'Архитектура компьютеров'
dmitriy_dovasheyev$ mkdir lab04
dmitriy_dovasheyev$ cd lab04
dmitriy_dovasheyev$ pwd
/home/dmitriy_dovasheyev/work/study/2025-2026/lab04
dmitriy_dovasheyev$ ls
dmitriy_dovasheyev$ touch hello.asm
dmitriy_dovasheyev$ ls
hello.asm
dmitriy_dovasheyev$ nano hello.asm
dmitriy_dovasheyev$ mousepad hello.asm &
[1] 3560
dmitriy_dovasheyev$
(mousepad:3560): Glib-CRITICAL **: 17:04:12.600: g_strjoinv: assertion 'str_array != NUL
L' failed

(mousepad:3560): Glib-CRITICAL **: 17:04:12.600: g_strjoinv: assertion 'str_array != NUL
L' failed
```

Рис. 4.2: Создание .asm файла

С помощью редактора пишу программу в созданном файле. (рис. 4.3)



```
*/work/study/2025-2026/lab04/hello.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
SECTION .data
    hello:    db "Hello, world!", 0xa
    helloLen: equ $ - hello

SECTION .text
    global _start

_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, hello
    mov edx, helloLen
    int 0x80

    mov eax, 1
    mov ebx, 0
    int 0x80
```

Рис. 4.3: Редактирование файла

4.2 Транслятор NASM

Компилирую с помощью NASM свою программу. (рис. 4.4)

```
Терминал
dmitriy_dovasheyev$ touch hello.asm
dmitriy_dovasheyev$ mousepad hello.asm &
[1] 4295
dmitriy_dovasheyev$
(mousepad:4295): GLib-CRITICAL **: 17:24:20.407: g_strjoinv: assertion 'str_array != NUL
L' failed

(mousepad:4295): GLib-CRITICAL **: 17:24:20.407: g_strjoinv: assertion 'str_array != NUL
L' failed
nasm -f elf hello.asm
[1]+  Завершён      mousepad hello.asm
dmitriy_dovasheyev$ ls
hello.asm  parentdir2  temp  Документы  Общедоступные
hello.o    parentdir3  tmp   Загрузки   'Рабочий стол'
parentdir  quarto-1.5.57-linux-amd64.deb  work  Изображения  Шаблоны
parentdir1 snap        Видео    Музыка
dmitriy_dovasheyev$ cd ~/work/study/2025-2026/lab04
dmitriy_dovasheyev$ ls
hello hello.asm hello.o
dmitriy_dovasheyev$
```

Рис. 4.4: Компиляция программы

4.3 Расширенный синтаксис командной строки NASM

Выполняя команду, указанную на (рис. 4.5), она скомпилировала исходный файл hello.asm в obj.o, расширение .o говорит о том, что файл - объектный, помимо него флаги -g -l подготовят файл отладки и листинга соответственно.

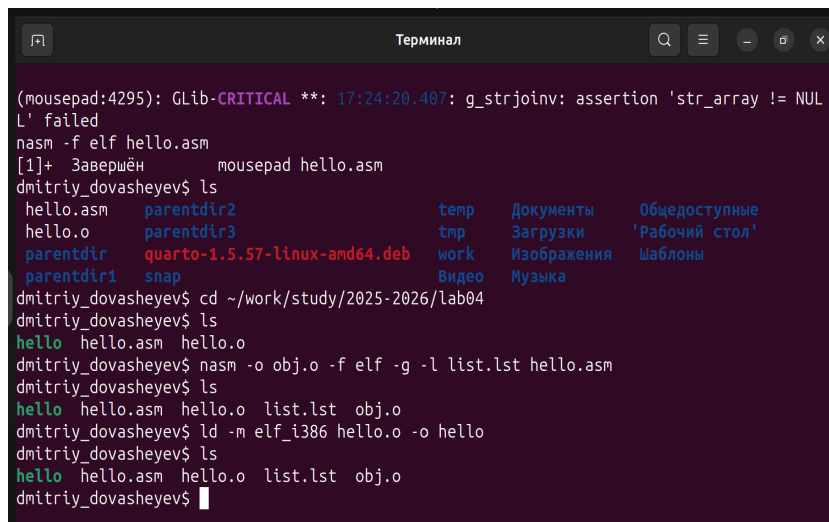
```
Терминал
dmitriy_dovasheyev$
(mousepad:4295): GLib-CRITICAL **: 17:24:20.407: g_strjoinv: assertion 'str_array != NUL
L' failed

(mousepad:4295): GLib-CRITICAL **: 17:24:20.407: g_strjoinv: assertion 'str_array != NUL
L' failed
nasm -f elf hello.asm
[1]+  Завершён      mousepad hello.asm
dmitriy_dovasheyev$ ls
hello.asm  parentdir2  temp  Документы  Общедоступные
hello.o    parentdir3  tmp   Загрузки   'Рабочий стол'
parentdir  quarto-1.5.57-linux-amd64.deb  work  Изображения  Шаблоны
parentdir1 snap        Видео    Музыка
dmitriy_dovasheyev$ cd ~/work/study/2025-2026/lab04
dmitriy_dovasheyev$ ls
hello hello.asm hello.o
dmitriy_dovasheyev$ nasm -o obj.o -f elf -g -l list.lst hello.asm
dmitriy_dovasheyev$ ls
hello hello.asm hello.o list.lst obj.o
dmitriy_dovasheyev$
```

Рис. 4.5: Возможности синтаксиса NASM

4.4 Компоновщик LD

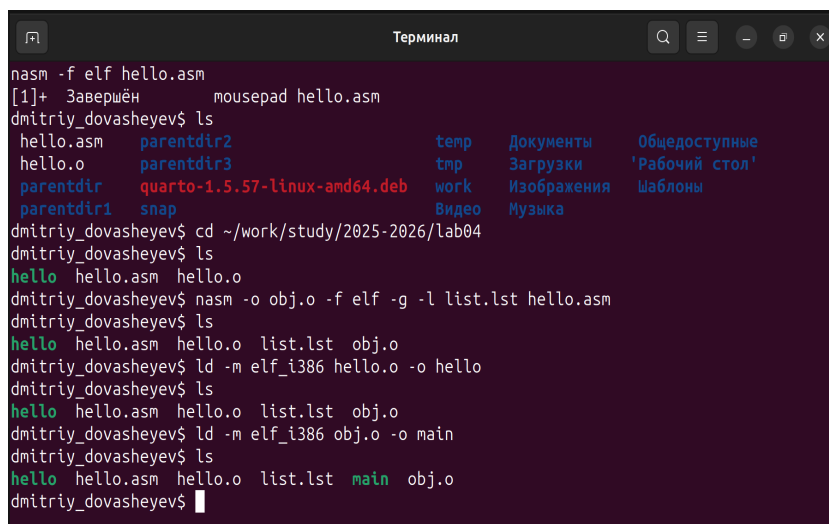
Затем мне необходимо передать объектный файл компоновщику, делаю это с помощью команды `ld`. (рис. 4.6)



```
(mousepad:4295): Glib-CRITICAL **: 17:24:20.407: g_strjoinv: assertion 'str_array != NUL
L' failed
nasm -f elf hello.asm
[1]+  Завершён      mousepad hello.asm
dmitriy_dovasheyev$ ls
hello.asm  parentdir2      tmp      Документы  Общедоступные
hello.o    parentdir3      tmp      Загрузки  'Рабочий стол'
parentdir  quarto-1.5.57-linux-amd64.deb  work  Изображения  Шаблоны
parentdir1 snap            Видео    Музыка
dmitriy_dovasheyev$ cd ~/work/study/2025-2026/lab04
dmitriy_dovasheyev$ ls
hello  hello.asm  hello.o
dmitriy_dovasheyev$ nasm -o obj.o -f elf -g -l list.lst hello.asm
dmitriy_dovasheyev$ ls
hello  hello.asm  hello.o  list.lst  obj.o
dmitriy_dovasheyev$ ld -m elf_i386 hello.o -o hello
dmitriy_dovasheyev$ ls
hello  hello.asm  hello.o  list.lst  obj.o
dmitriy_dovasheyev$
```

Рис. 4.6: Отправка файла компоновщику

Выполняю следующую команду ..., результатом исполнения команды будет созданный файл `main`, скомпонованный из объектного файла `obj.o`. (рис. 4.7)

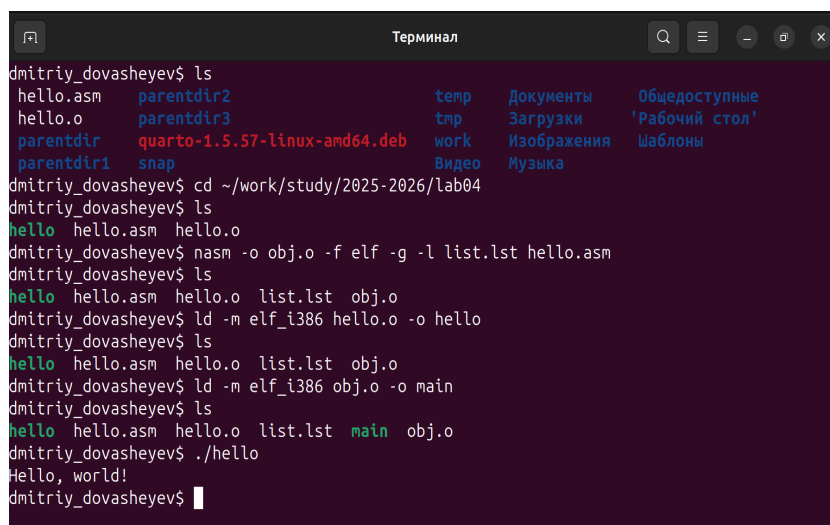


```
nasm -f elf hello.asm
[1]+  Завершён      mousepad hello.asm
dmitriy_dovasheyev$ ls
hello.asm  parentdir2      tmp      Документы  Общедоступные
hello.o    parentdir3      tmp      Загрузки  'Рабочий стол'
parentdir  quarto-1.5.57-linux-amd64.deb  work  Изображения  Шаблоны
parentdir1 snap            Видео    Музыка
dmitriy_dovasheyev$ cd ~/work/study/2025-2026/lab04
dmitriy_dovasheyev$ ls
hello  hello.asm  hello.o
dmitriy_dovasheyev$ nasm -o obj.o -f elf -g -l list.lst hello.asm
dmitriy_dovasheyev$ ls
hello  hello.asm  hello.o  list.lst  obj.o
dmitriy_dovasheyev$ ld -m elf_i386 hello.o -o hello
dmitriy_dovasheyev$ ls
hello  hello.asm  hello.o  list.lst  obj.o
dmitriy_dovasheyev$ ld -m elf_i386 obj.o -o main
dmitriy_dovasheyev$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
dmitriy_dovasheyev$
```

Рис. 4.7: Создание исполняемого файла

4.5 Запуск исполняемого файла

Запускаю исполняемый файл из текущего каталога. (рис. 4.8)



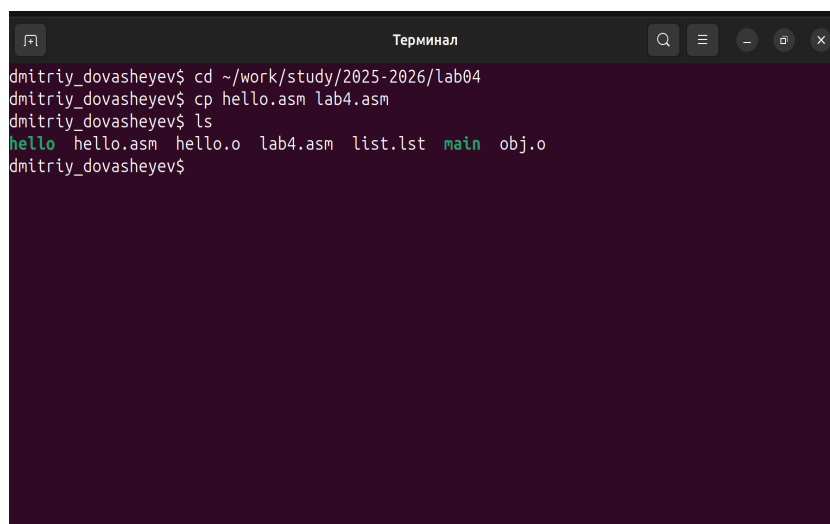
```
Терминал
dmitriy_dovasheyev$ ls
hello.asm  parentdir2  temp  Документы  Общедоступные
hello.o    parentdir3  tmp   Загрузки  'Рабочий стол'
parentdir  quarto-1.5.57-linux-amd64.deb  work  Изображения  Шаблоны
parentdir1 snap        Видео  Музыка

dmitriy_dovasheyev$ cd ~/work/study/2025-2026/lab04
dmitriy_dovasheyev$ ls
hello  hello.asm  hello.o
dmitriy_dovasheyev$ nasm -o obj.o -f elf -g -l list.lst hello.asm
dmitriy_dovasheyev$ ls
hello  hello.asm  hello.o  list.lst  obj.o
dmitriy_dovasheyev$ ld -m elf_i386 hello.o -o hello
dmitriy_dovasheyev$ ls
hello  hello.asm  hello.o  list.lst  obj.o
dmitriy_dovasheyev$ ld -m elf_i386 obj.o -o main
dmitriy_dovasheyev$ ls
hello  hello.o  list.lst  main  obj.o
dmitriy_dovasheyev$ ./hello
Hello, world!
dmitriy_dovasheyev$
```

Рис. 4.8: Запуск программы

4.6 Задания для самостоятельной работы

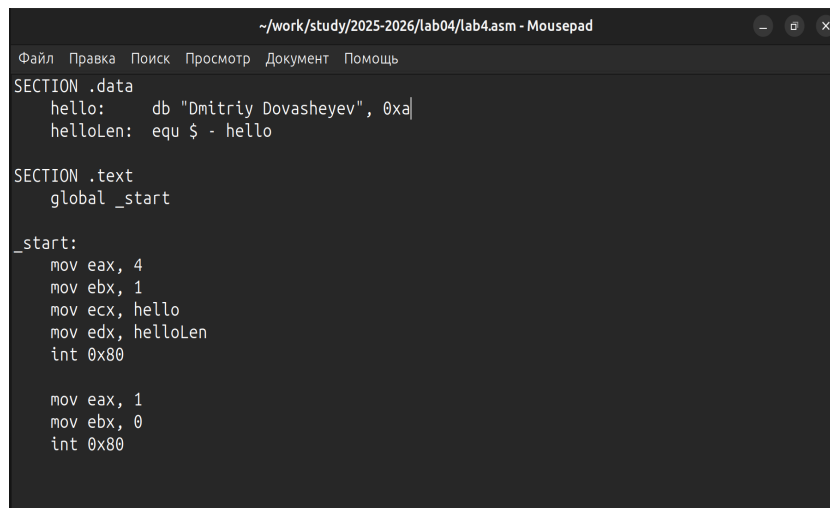
Создаю копию файла для последующей работы с ней. (рис. 4.9)



```
Терминал
dmitriy_dovasheyev$ cd ~/work/study/2025-2026/lab04
dmitriy_dovasheyev$ cp hello.asm lab4.asm
dmitriy_dovasheyev$ ls
hello  hello.o  lab4.asm  list.lst  main  obj.o
dmitriy_dovasheyev$
```

Рис. 4.9: Создание копии

Редактирую копию файла, заменив текст на свое имя и фамилию. (рис. 4.10)



```
~/work/study/2025-2026/lab04/lab4.asm - Mousepad
Файл  Правка  Поиск  Просмотр  Документ  Помощь
SECTION .data
    hello:    db "Dmitriy Dovasheyev", 0xa
    helloLen: equ $ - hello

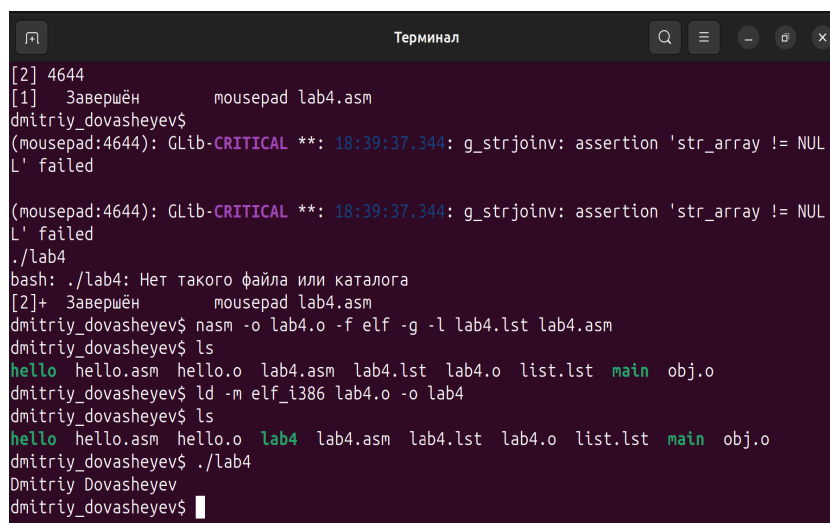
SECTION .text
    global _start

_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, hello
    mov edx, helloLen
    int 0x80

    mov eax, 1
    mov ebx, 0
    int 0x80
```

Рис. 4.10: Редактирование копии

Транслирую копию файла в объектный файл, компоную и запускаю. (рис. 4.11)



```
Терминал
[2] 4644
[1] Завершён      mousepad lab4.asm
dmitriy_dovasheyev$
(mousepad:4644): GLib-CRITICAL **: 18:39:37.344: g_strjoinv: assertion 'str_array != NUL
L' failed

(mousepad:4644): GLib-CRITICAL **: 18:39:37.344: g_strjoinv: assertion 'str_array != NUL
L' failed
./lab4
bash: ./lab4: Нет такого файла или каталога
[2]+  Завершён      mousepad lab4.asm
dmitriy_dovasheyev$ nasm -o lab4.o -f elf -g -l lab4.lst lab4.asm
dmitriy_dovasheyev$ ls
hello  hello.asm  hello.o  lab4.asm  lab4.lst  lab4.o  list.lst  main  obj.o
dmitriy_dovasheyev$ ld -m elf_i386 lab4.o -o lab4
dmitriy_dovasheyev$ ls
hello  hello.asm  hello.o  lab4  lab4.asm  lab4.lst  lab4.o  list.lst  main  obj.o
dmitriy_dovasheyev$ ./lab4
Dmitriy Dovasheyev
dmitriy_dovasheyev$
```

Рис. 4.11: Проверка работоспособности скомпонованной программы

Убедившись в корректности работы программы, копирую рабочие файлы в свой локальный репозиторий. (рис. 4.12)

5 Выводы

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

6 Список литературы

1. Пример выполнения лабораторной работы
2. Курс на ТУИС
3. Лабораторная работа №4
4. Программирование на языке ассемблера NASM Столяров А. В.