

# **Лабораторная работа №4**

**Архитектура вычислительных систем**

Ягмыров Сохбет

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Выводы</b>	<b>16</b>
	<b>Список литературы</b>	<b>17</b>

## Список иллюстраций

4.1	создание каталога . . . . .	8
4.2	gedit . . . . .	8
4.3	файл hello.asm . . . . .	9
4.4	успешная компиляция . . . . .	9
4.5	транслятор . . . . .	10
4.6	ged it report.md . . . . .	11
4.7	картинки . . . . .	12
4.8	файл . . . . .	12
4.9	самостоятельная работа.png . . . . .	13
4.10	самостоятельная работа.png . . . . .	13
4.11	самостоятельная работа.png . . . . .	14
4.12	самостоятельная работа.png . . . . .	14
4.13	самостоятельная работа.png . . . . .	14
4.14	самостоятельная работа.png . . . . .	15

# Список таблиц

3.1	Описание некоторых каталогов файловой системы GNU Linux . .	7
-----	---	---

# 1 Цель работы

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

Цель данного шаблона — максимально упростить подготовку отчётов по лабораторным работам. Модифицируя данный шаблон, студенты смогут без труда подготовить отчёт по лабораторным работам, а также познакомиться с основными возможностями разметки Markdown.

## 2 Задание

1. В соответствующем каталоге сделайте отчёт по лабораторной работе No4 в формате Markdown. В качестве отчёта необходимо предоставить отчёты в 3 форматах: pdf, docx и md.
2. Загрузите файлы на github.

### 3 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы. Например, в табл. 3.1 приведено краткое описание стандартных каталогов Unix.

Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux

Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую систему
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно об Unix см. в [1–6].

## 4 Выполнение лабораторной работы

- 1) Создаём каталог для работы с программами на языке ассемблера NASM

```
smyagmihrov@dk3n31 ~/work $ mkdir arch-pc
smyagmihrov@dk3n31 ~/work $ cd arch-pc
smyagmihrov@dk3n31 ~/work/arch-pc $ mkdir lab04
smyagmihrov@dk3n31 ~/work/arch-pc $ cd lab04
```

Рис. 4.1: создание каталога

- 2) Создаём текстовый файл с именем hello.asm и открываем этот файл с помощью любого текстового редактора gedit:

```
smyagmihrov@dk3n31 ~/work/arch-pc/lab04 $ touch hello.asm
smyagmihrov@dk3n31 ~/work/arch-pc/lab04 $ gedit hello.asm
```

Рис. 4.2: gedit

- 3) Вводим в него следующий текст:




```
Открыть ▾  lab04.asm  
~/work/arch-pc/lab04  
1 ; hello.asm  
2 SECTION .data ; Начало секции данных  
3 hello: DB 'Ягмыров Сохбет',10 ; 'Ягмыров Сохбет' плюс  
4 ; символ перевода строки  
5 helloLen: EQU $-hello ; Длина строки hello  
6 SECTION .text ; Начало секции кода  
7 GLOBAL _start  
8 _start: ; Точка входа в программу  
9 mov eax,4 ; Системный вызов для записи (sys_write)  
10 mov ebx,1 ; Описатель файла '1' - стандартный вывод  
11 mov ecx,hello ; Адрес строки hello в ecx  
12 mov edx,helloLen ; Размер строки hello  
13 int 80h ; Вызов ядра  
14 mov eax,1 ; Системный вызов для выхода (sys_exit)  
15 mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)  
16 int 80h ; Вызов ядра  
17
```

Рис. 4.3: файл hello.asm

4)NASM превращает текст программы в объектный код. Например, для компиляции приведённого выше текста программы «Hello World» необходимо написать следующее

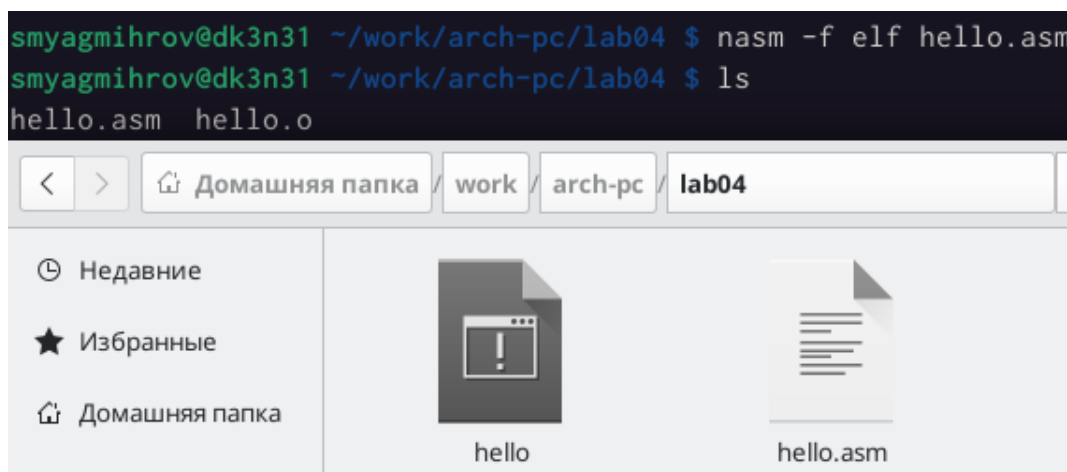


Рис. 4.4: успешная компиляция

Т. к. текст программы набран без ошибок, транслятор преобразует текст про-

граммы из файла `hello.asm` в объектный код, который записан в файл `hello.o`. 8 5) С помощью команды `ls` проверим, что объектный файл был создан. У нас есть два файла `hello.asm` и `hello.o`. Следующая команда скомпилирует исходный файл `hello.asm` в `obj.o`, при этом формат выходного файла будет `elf`, и в него будут включены символы для отладки (опция `-g`), кроме того, создается файл листинга `list.lst`. Выполним следующую команду:

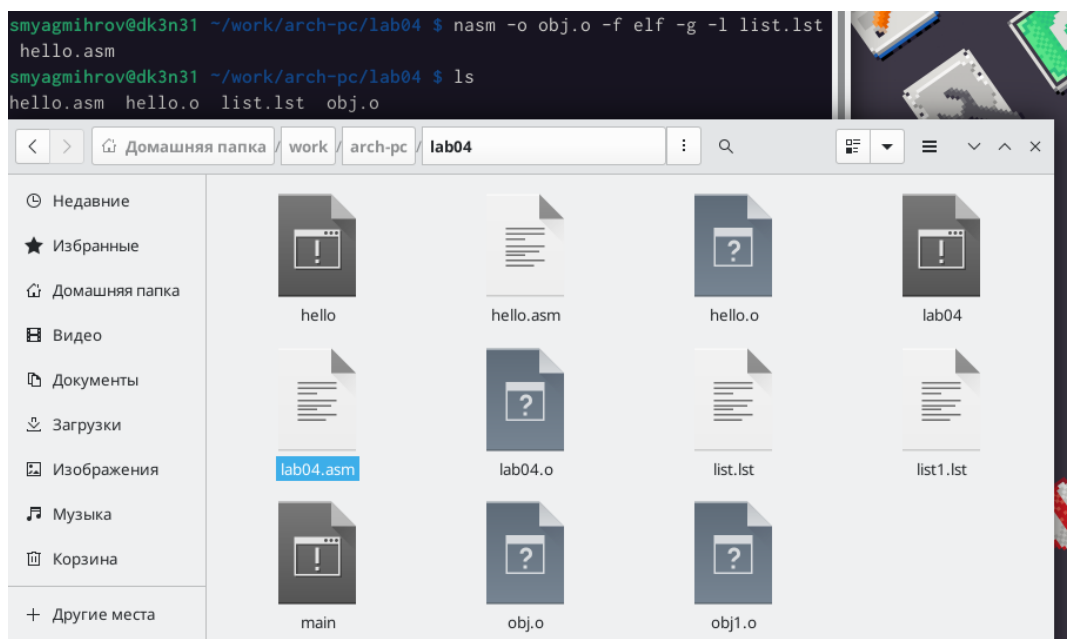


Рис. 4.5: транслятор

6) Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику, а потом с командой `ls` проверим содержимое:

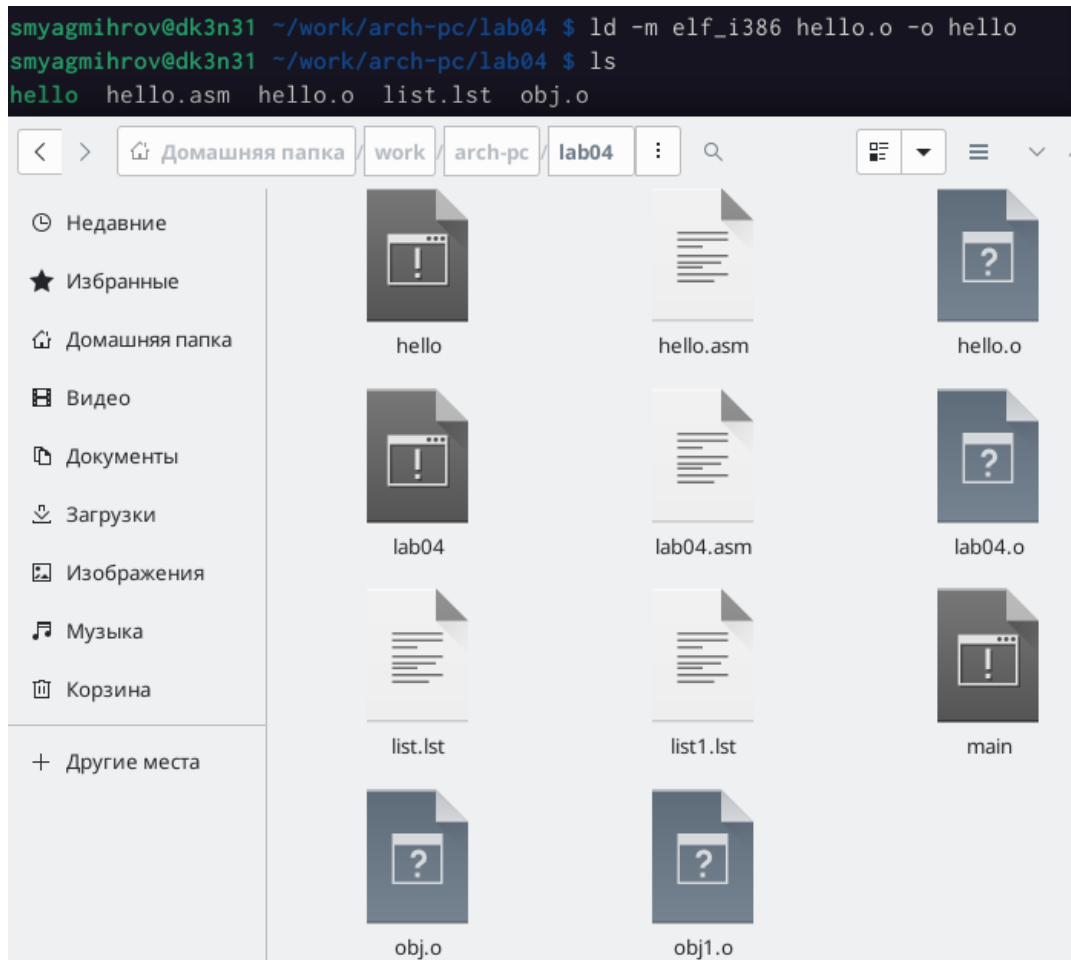


Рис. 4.6: ged it report.md

- 7) Ключ `-o` с последующим значением задаст в данном случае имя создаваемого исполняемого файла. Выполним следующую команду Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику, а потом с командой `ls` проверим содержимое:

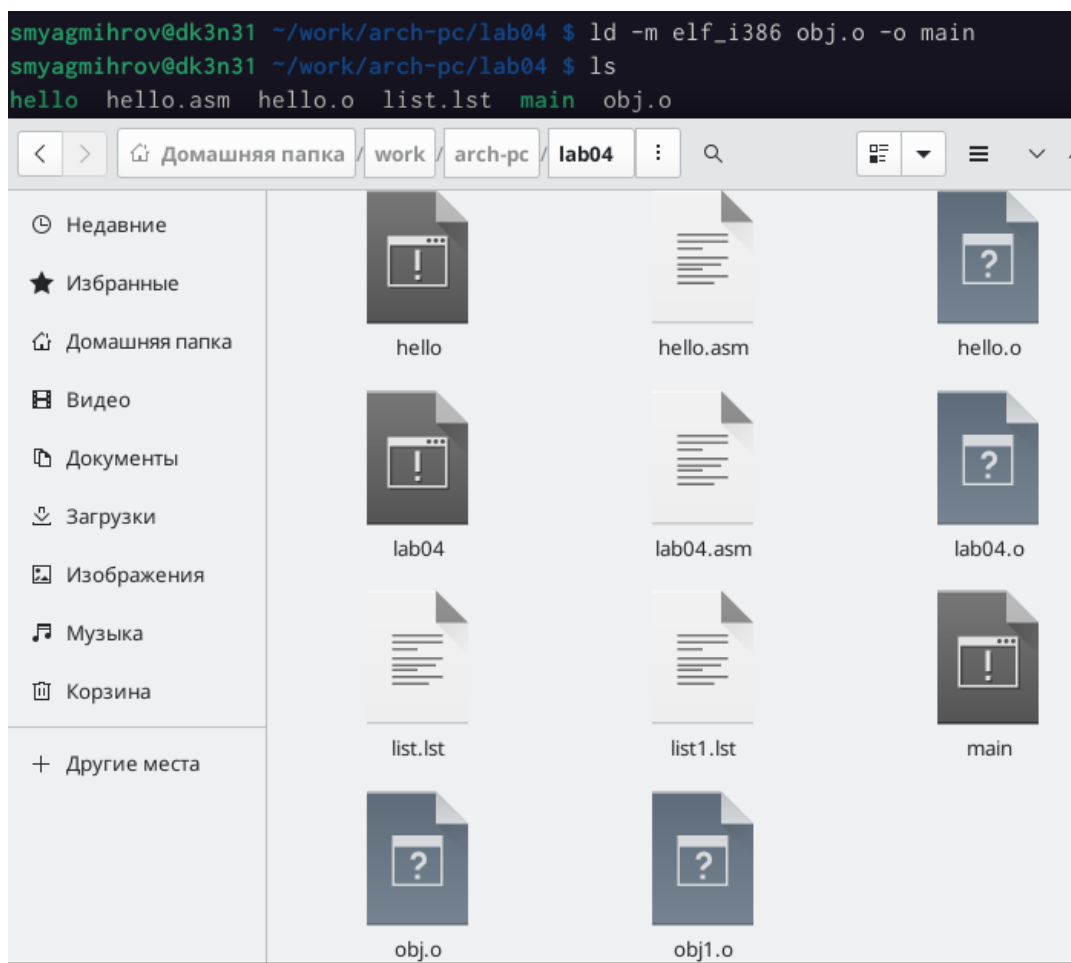


Рис. 4.7: картинки

11) Запустим на выполнение созданный исполняемый файл, находящийся в текущем каталоге, набрав в командной строке `./hello`:

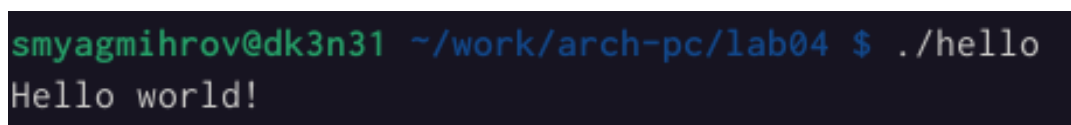
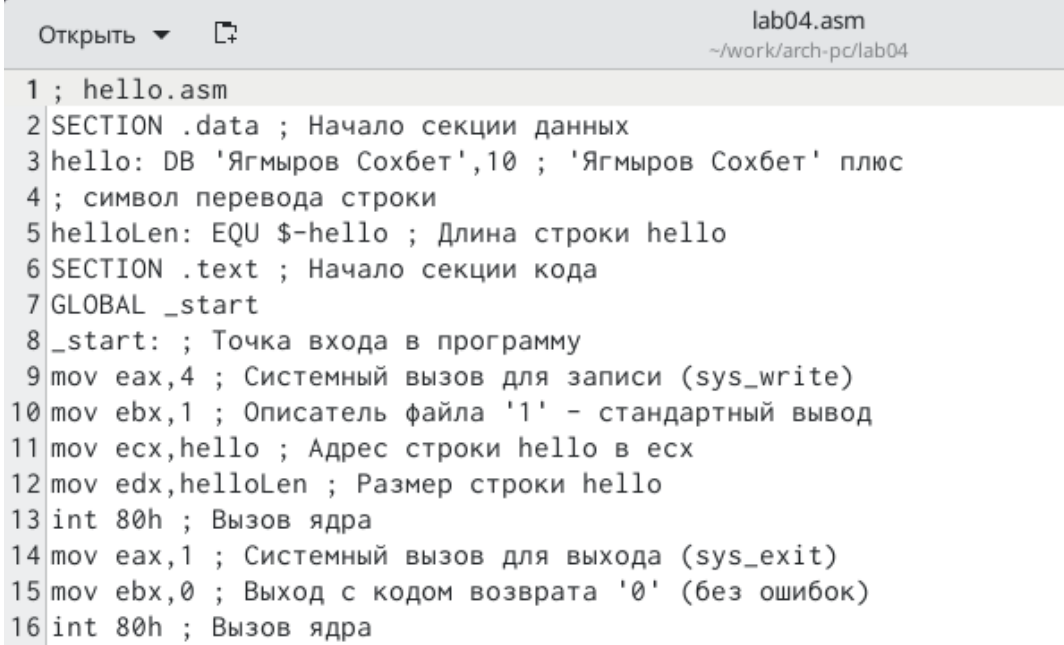


Рис. 4.8: файл

11.5 Выполнение самостоятельной работы 1) В каталоге `~/work/arch-pc/lab04` с помощью команды `cp` создали копию файла `hello.asm` с именем `lab04.asm`.

```
smyagmihrov@dk3n31 ~/work/arch-pc/lab04 $ cp hello.asm lab04.asm
smyagmihrov@dk3n31 ~/work/arch-pc/lab04 $ gedit lab04.asm
```



```
1 ; hello.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Ягмыров Сохбет',10 ; 'Ягмыров Сохбет' плюс
4 ; символ перевода строки
5 helloLen: EQU $-hello ; Длина строки hello
6 SECTION .text ; Начало секции кода
7 GLOBAL _start
8 _start: ; Точка входа в программу
9 mov eax,4 ; Системный вызов для записи (sys_write)
10 mov ebx,1 ; Описатель файла '1' - стандартный вывод
11 mov ecx,hello ; Адрес строки hello в ecx
12 mov edx,helloLen ; Размер строки hello
13 int 80h ; Вызов ядра
14 mov eax,1 ; Системный вызов для выхода (sys_exit)
15 mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
16 int 80h ; Вызов ядра
```

Рис. 4.9: самостоятельная работа.png

С помощью текстового редактора вносим изменения в текст программы в файле lab04.asm так, чтобы вместо Hello world! на экран выводилась строка с фамилией и именем. Для этого вместо “Hello world” пишем своё имя. 12

```
smyagmihrov@dk3n31 ~/work/arch-pc/lab04 $ ls
hello      hello.o  lab04.asm  list1.lst  main      obj.o
hello.asm  lab04   lab04.o    list.lst   obj1.o
```

Рис. 4.10: самостоятельная работа.png

Проводим схожие действия с лабораторной работой, но изменяем название файлов.

```

smyagmihrov@dk3n31 ~/work/arch-pc/lab04 $ nasm -f elf lab04.asm
smyagmihrov@dk3n31 ~/work/arch-pc/lab04 $ nasm -o obj1.o -f elf -g -l list1.l
st lab04.asm
smyagmihrov@dk3n31 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab04.o -o lab04
smyagmihrov@dk3n31 ~/work/arch-pc/lab04 $ gedit lab04.asm
smyagmihrov@dk3n31 ~/work/arch-pc/lab04 $ ls
hello      hello.o  lab04.asm  list1.lst  main      obj.o
hello.asm  lab04    lab04.o    list.lst   obj1.o

```

Рис. 4.11: самостоятельная работа.png

Проводим схожие действия с лабораторной работой, но изменяем название файлов.

```

smyagmihrov@dk3n31 ~/work/arch-pc/lab04 $ ./lab04
Ягмыров Сохбет

```

Рис. 4.12: самостоятельная работа.png

- 3) Оттранслируем полученный текст программы lab04.asm в объектный файл и запустим, получим вывод фамилии и имени.

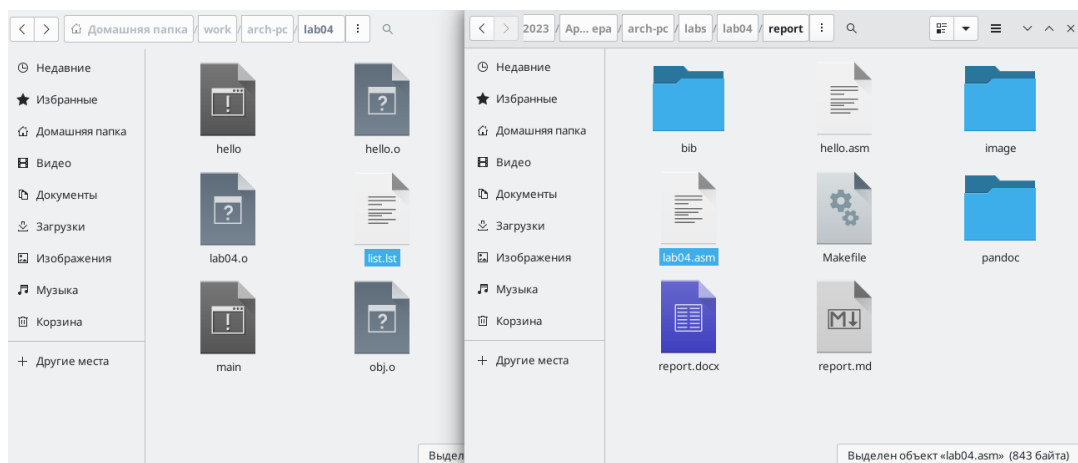


Рис. 4.13: самостоятельная работа.png

Переносим файлы в основную папку lab04: 13

```

smyagmihrov@dk3n31 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab04/report $ git add .
smyagmihrov@dk3n31 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab04/report $ git commit -am "4"
[master a7a5283] 4
18 files changed, 34 insertions(+)
create mode 100644 labs/lab04/report/hello.asm
create mode 100644 labs/lab04/report/image/1.png
create mode 100644 labs/lab04/report/image/10.png
create mode 100644 labs/lab04/report/image/11.png
create mode 100644 labs/lab04/report/image/12.png
create mode 100644 labs/lab04/report/image/2.png
create mode 100644 labs/lab04/report/image/3.png
create mode 100644 labs/lab04/report/image/4.png
create mode 100644 labs/lab04/report/image/5.png
create mode 100644 labs/lab04/report/image/6.png
create mode 100644 labs/lab04/report/image/7.png
create mode 100644 labs/lab04/report/image/8.png
create mode 100644 labs/lab04/report/image/9.png
create mode 100644 labs/lab04/report/image/Снимок экрана от 2022-12-09 17-42-51.png
create mode 100644 labs/lab04/report/image/Снимок экрана от 2022-12-09 19-22-30.png
create mode 100644 labs/lab04/report/image/Снимок экрана от 2022-12-09 19-34-31.png
create mode 100644 labs/lab04/report/lab04.asm
create mode 100644 labs/lab04/report/report.docx
smyagmihrov@dk3n31 ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab04/report $ git push
ssh: Could not resolve hostname github.com: Device or resource busy
fatal: Не удалось прочитать из внешнего репозитория.

Удостоверьтесь, что у вас есть необходимые права доступа
и репозиторий существует.

```

Рис. 4.14: самостоятельная работа.png

## 5 Выводы

Я освоил процедуру компиляции и сборки программ, написанных на ассемблере NASM.



## Список литературы

1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016.  
URL: <https://www.gnu.org/software/bash/manual/>.
2. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.
6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.