

## **Отчёта по лабораторной работе №4**

**Дисциплина: архитектура компьютера**

Кудинов Максим Сергеевич НКАбд-03-24

## Содержание

1	Цель работы .....	2
2	Задание .....	2
3	Теоретическое введение.....	3
4	Выполнение лабораторной работы.....	4
4.1	Создание программы Hello world!.....	4
4.2	Работа с транслятором NASM.....	5
4.3	Работа с расширенным синтаксисом командной строки NASM .....	6
4.4	Работа с компоновщиком LD .....	6
4.5	Запуск исполняемого файла .....	7
4.6	Выполнение заданий для самостоятельной работы.....	7
5	Выводы .....	10
6	Список литературы .....	10

## 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 объема, 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 объемов 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.

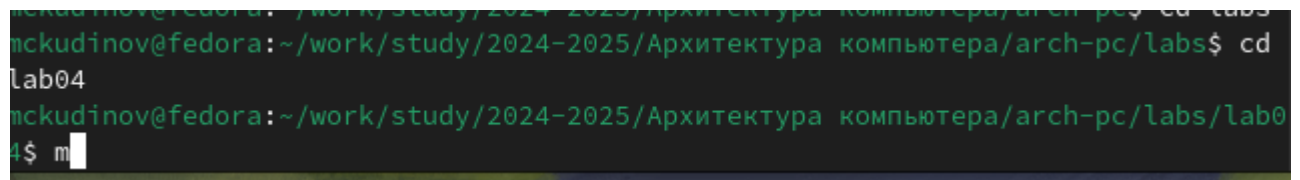
Коды команд представляют собой многоразрядные двоичные комбинации из 0 и 1. В коде машинной команды можно выделить две части: операционную и адресную. В операционной части хранится код команды, которую необходимо выполнить. В адресной части хранятся данные или адреса данных, которые участвуют в выполнении 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 Создание программы Hello world!

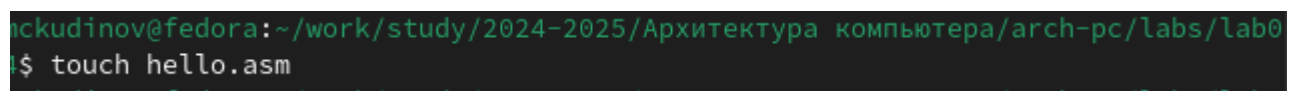
С помощью утилиты `cd` перемещаюсь в каталог, в котором буду работать (рис. 1).



```
nickudinov@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc$ cd lab04
nickudinov@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ m
```

Рис. 1: Перемещение между директориями

Создаю в текущем каталоге пустой текстовый файл `hello.asm` с помощью утилиты `touch` (рис. 2).



```
nickudinov@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ touch hello.asm
```

Рис. 2: Создание пустого файла

Открываю созданный файл в текстовом редакторе `mousepad` (рис. 3).

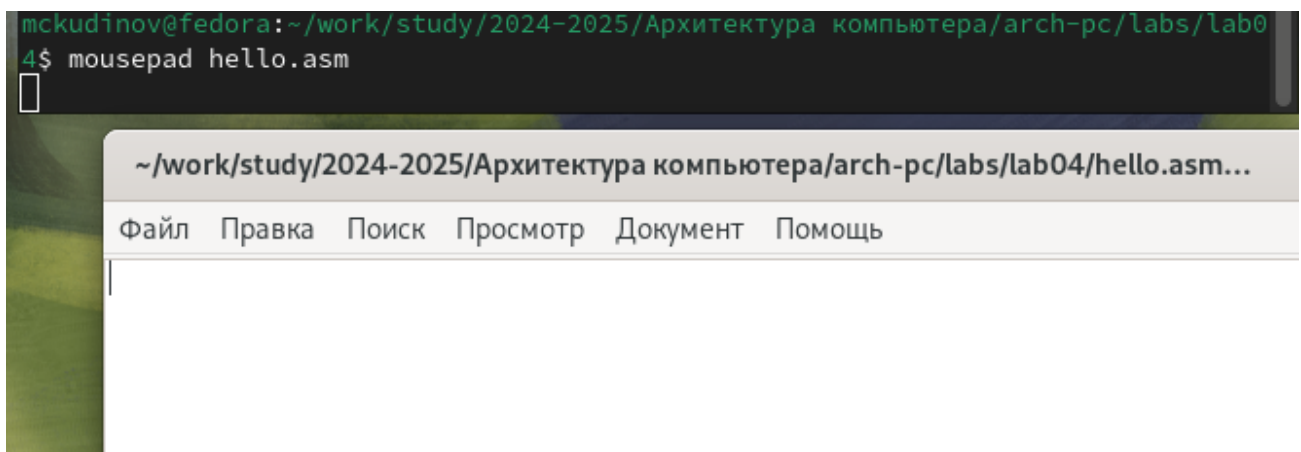


Рис. 3: Открытие файла в текстовом редакторе

Заполняю файл, вставляя в него программу для вывода “Hello word!” (рис. 4).

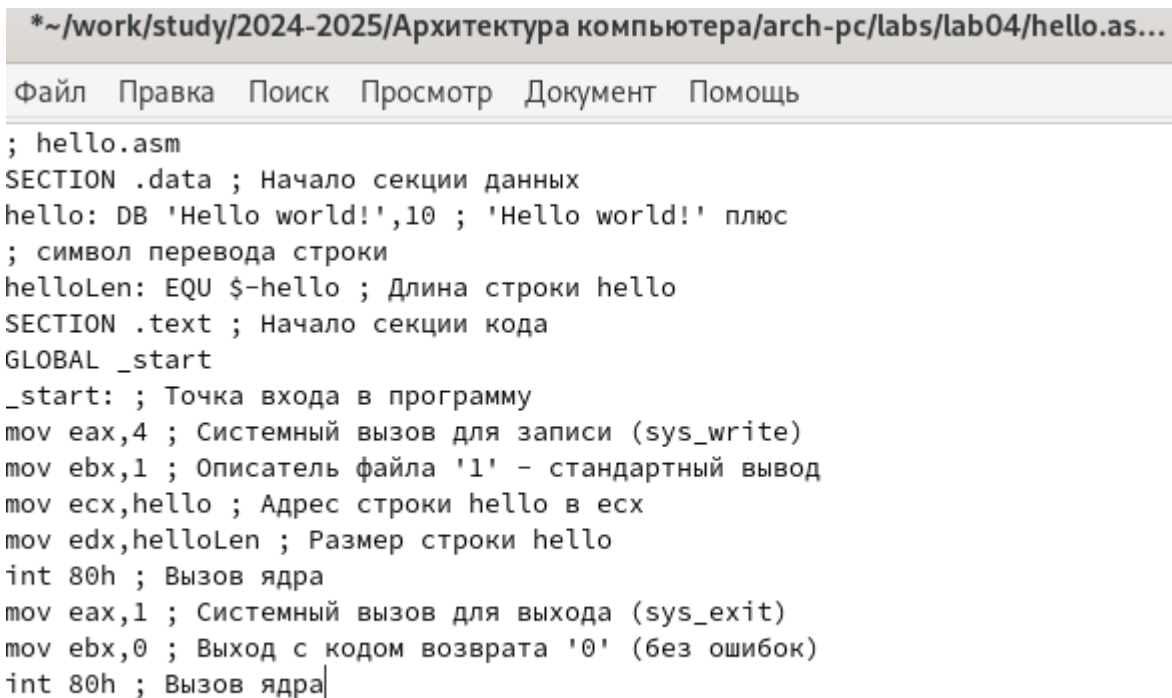


Рис. 4: Заполнение файла

## 4.2 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. 5). Далее проверяю правильность выполнения команды с помощью утилиты `ls`: действительно, создан файл “hello.o”.

```
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$ nasm -f elf hello.asm
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$ ls
hello.asm hello.o presentation report
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$
```

Рис. 5: Компиляция текста программы

### 4.3 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. 6). Далее проверяю с помощью утилиты ls правильность выполнения команды.

```
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$ nasm -o obj.o -f elf -g -l list.lst hello.asm
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$ ls
hello.asm hello.o list.lst obj.o presentation report
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$
```

Рис. 6: Компиляция текста программы

### 4.4 Работа с компоновщиком LD

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello (рис. 7). Ключ -o задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты ls правильность выполнения команды.

```
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$ ld -m elf_i386 hello.o -o hello
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$ ls
hello hello.asm hello.o list.lst obj.o presentation report
```

Рис. 7: Передача объектного файла на обработку компоновщику

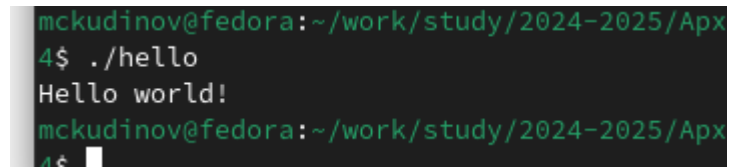
Выполняю следующую команду (рис. 8). Исполняемый файл будет иметь имя main, т.к. после ключа -o было задано значение main. Объектный файл, из которого собран этот исполняемый файл, имеет имя obj.o

```
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$ ls
hello hello.asm hello.o list.lst main obj.o presentation report
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$
```

Рис. 8: Передача объектного файла на обработку компоновщику

## 4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello (рис. 9).

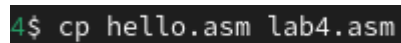
A terminal window with a dark background and green text. The prompt is 'mckudinov@fedora:~/work/study/2024-2025/Apx'. The user enters './hello' and the output is 'Hello world!'.

```
mckudinov@fedora:~/work/study/2024-2025/Apx
4$ ./hello
Hello world!
mckudinov@fedora:~/work/study/2024-2025/Apx
4$
```

Рис. 9: Запуск исполняемого файла

## 4.6 Выполнение заданий для самостоятельной работы.

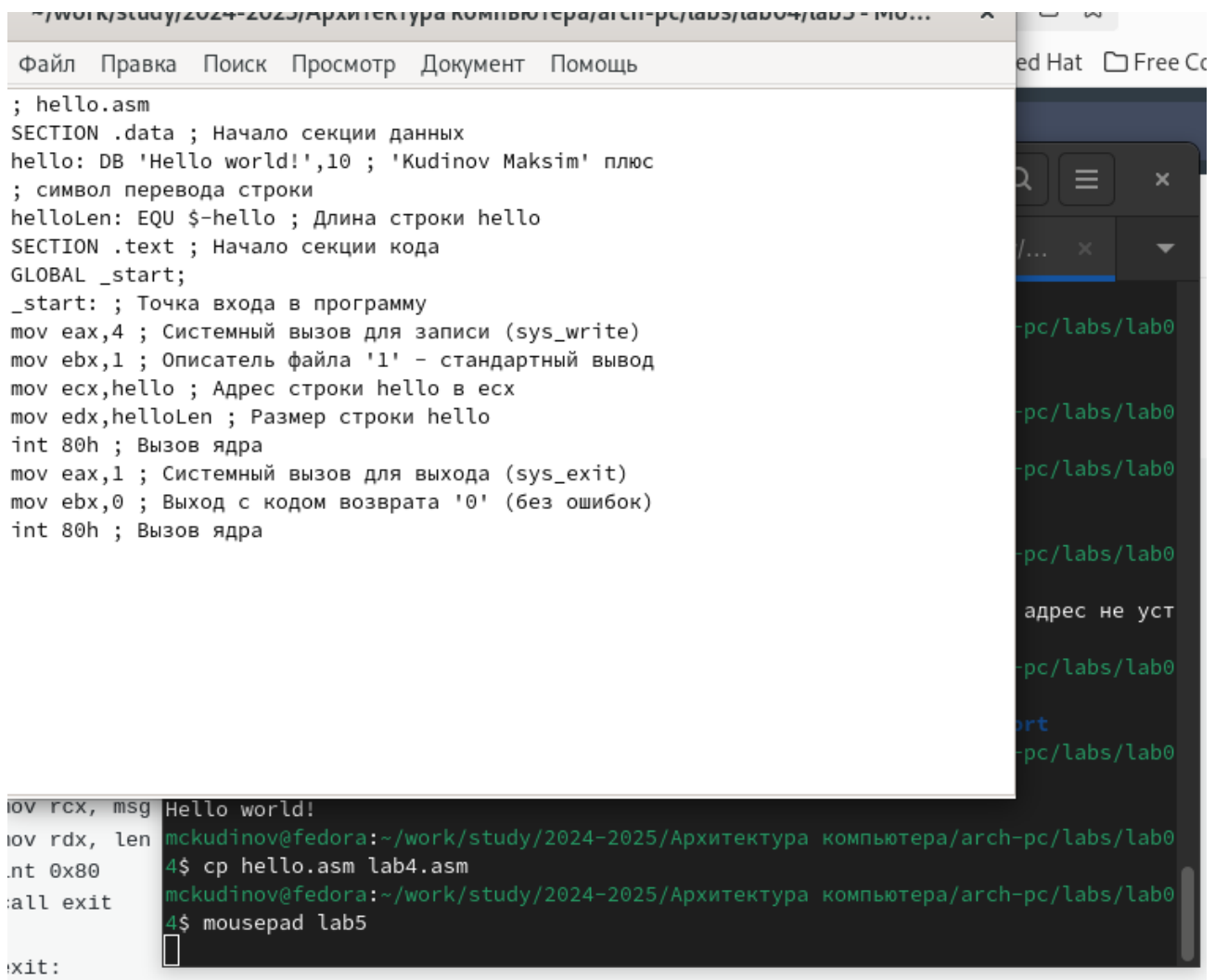
С помощью утилиты cp создаю в текущем каталоге копию файла hello.asm с именем lab4.asm (рис. 10).

A terminal window with a dark background and green text. The prompt is '4\$'. The user enters 'cp hello.asm lab4.asm'.

```
4$ cp hello.asm lab4.asm
```

Рис. 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. 11).



The image shows a text editor window with assembly code for a program named 'hello.asm'. The code defines a data section with a string 'Hello world!' and a text section with assembly instructions to print the string and exit. Below the editor, a terminal window shows the execution of the program. The terminal output displays 'Hello world!' and then returns to the prompt. The terminal also shows the execution of 'cp hello.asm lab4.asm' and 'mousepad lab5'.

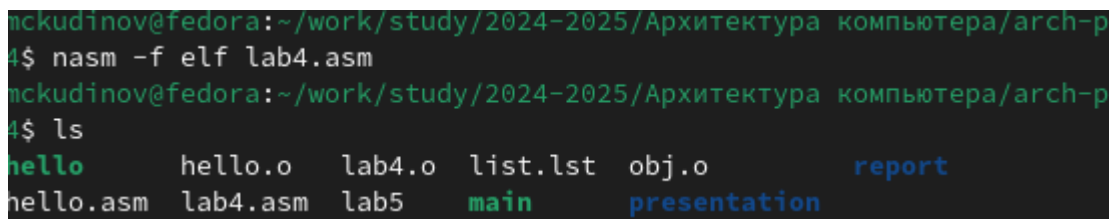
```
; hello.asm
SECTION .data ; Начало секции данных
hello: DB 'Hello world!',10 ; 'Kudinov Maksim' плюс
; символ перевода строки
helloLen: EQU $-hello ; Длина строки hello
SECTION .text ; Начало секции кода
GLOBAL _start;
_start: ; Точка входа в программу
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
mov ecx,hello ; Адрес строки hello в ecx
mov edx,helloLen ; Размер строки hello
int 80h ; Вызов ядра
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
int 80h ; Вызов ядра

mov rcx, msg
mov rdx, len
int 0x80
call exit
exit:
```

```
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$ cp hello.asm lab4.asm
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$ mousepad lab5
```

Рис. 11: Изменение программы

Компилирую текст программы в объектный файл (рис. 12). Проверяю с помощью утилиты ls, что файл lab4.o создан.



The image shows a terminal window where the assembly file 'lab4.asm' is compiled using 'nasm -f elf lab4.asm'. After compilation, the 'ls' command is used to list the files in the directory. The output shows the creation of 'lab4.o' and other files.

```
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$ nasm -f elf lab4.asm
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$ ls
hello.o  lab4.o  list.lst  obj.o  report
hello.asm  lab4.asm  lab5  main  presentation
```

Рис. 12: Компиляция текста программы

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4 (рис. 13).



```

mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$ ld -m elf_i386 lab4.o -o lab4
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$ ls
hello      hello.o  lab4.asm  list.lst  obj.o      report
hello.asm  lab4     lab4.o    main      presentation
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$

```

Рис. 13: Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4, на экран действительно выводятся мои имя и фамилия (рис. 14).

```

mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$ ./lab4
Kudinov Maxim!
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$

```

Рис. 14

Удаляю лишние файлы в текущем каталоге с помощью утилиты rm, ведь копии файлов остались в другой директории (рис. 15).

```

mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$ rm hello hello.o lab4 lab4.o list.lst main obj.o
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$ ls
hello.asm  lab4.asm  presentation  report
mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$

```

Рис. 15: Удаление лишних файлов в текущем каталоге

С помощью команд git add . и git commit добавляю файлы на GitHub, комментируя действие как добавление файлов для лабораторной работы №4 (рис. 16).

```

mckudinov@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab0
4$ git add .

```

Рис. 16: Добавление файлов на GitHub

Отправляю файлы на сервер с помощью команды git push (рис. 17).

```
4$ git pull
remote: Enumerating objects: 38, done.
remote: Counting objects: 100% (38/38), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 31 (delta 13), reused 0 (delta 0), pack-reused 0 (from 0)
Распаковка объектов: 100% (31/31), 1.97 МиБ | 3.38 МиБ/с, готово.
Из github.com:mckudinov/study_2024-2025_arhpc
   2a198cc..12ebf9b master    -> origin/master
Обновление 316462d..12ebf9b
error: Указанные неотслеживаемые файлы в рабочем каталоге будут потеряны при слиянии:
       labs/lab02/report/Л02_Кудинов_отчет.md
Переместите эти файлы или удалите их перед переключением веток.
Прерываю
```

Рис. 17: Отправка файлов

## 5 Выводы

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

## 6 Список литературы

1. [https://esystem.rudn.ru/pluginfile.php/1584628/mod\\_resource/content/1/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%965.pdf](https://esystem.rudn.ru/pluginfile.php/1584628/mod_resource/content/1/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%965.pdf)