

Шаблон отчёта по лабораторной работе

Простейший вариант

Ариоке Габриэль Одафе; НКАБД-05-22

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	17
	Список литературы	18

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задание

1. Программа Hello world!
2. Транслятор NASM, Расширенный синтаксис командой строки NASM и Компоновщик LD.
3. Запуск исполняемого файла
4. Задание для самостоятельной работы
5. В каталоге `~/work/arch-pc/lab05` с помощью команды `cp` создайте копию файла `hello.asm` с именем `lab5.asm`
6. С помощью любого текстового редактора внесите изменения в текст программы в файле `lab5.asm` так, чтобы вместо `Hello world!` на экран выводилась строка с вашими фамилией и именем.
7. Оттранслируйте полученный текст программы `lab5.asm` в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл.
8. Скопируйте файлы `hello.asm` и `lab5.asm` в Ваш локальный репозиторий в каталог `~/work/study/2022-2023/"Архитектура компьютера"/arch-pc/labs/lab05/`. Загрузите файлы на Github.

3 Теоретическое введение

Основными функциональными элементами любой электронно-вычислительной машины (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 (системной) плате.

Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора (ЦП) 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.

4 Выполнение лабораторной работы

1. Программа Hello world! Я создал каталог для работы с программами на ассемблере NASM и открыл файл в текстовом редакторе. После этого я ввел текст, показанный в описании изображения ниже. (рис.??, ??)


```
[goarioke@fedora ~]$ mkdir -p work/arch-pc/lab05  
[goarioke@fedora ~]$ cd ~/work/arch-pc/lab05  
[goarioke@fedora lab05]$ touch hello.asm  
[goarioke@fedora lab05]$ gedit hello.asm
```

#fig:001

width=70%}

```

1 ; hello.asm
2 SECTION .data          ; The beginning of the data section
3     hello: DB 'Hello world!',10 ; 'Hello world!' plus
4             ; Line feed symbol
5     helloLen: EQU $-hello ; The lenght of the hello string
6
7 SECTION .text          ; Start of code section
8 GLOBAL _start
9
10 _start:                ; Program entry point
11     mov eax,4           ; System call to write (sys_write)
12     mov ebx,1           ; File description '1' -standard output
13     mov ecx,hello       ; Address of hello string in ecx
14     mov edx,helloLen    ; The size of the hello string
15     int 80h            ; Kernel call
16
17     mov eax,1           ; System call for output (sys_exit)
18     mov ebx,0           ; Output with return code '0' (no errors)
19     int 80h            ; Kernel call
20

```

#fig:002

width=70%}

2. Транслятор NASM, Расширенный ситаксис командой строки NASM и Ком-
поновщик LD. Я скомпилировал программу «Hello World», используя NASM,
расширенный синтаксис командной строки NASM и компоновщик LD, как
показано на рисунке ниже. (рис.??, ??)

```

[goarioke@fedora lab05]$ nasm -f elf hello.asm
[goarioke@fedora lab05]$ ls
hello.asm  hello.o
[goarioke@fedora lab05]$ nasm -o obj.o -f elf -g -l list.lst hello.asm
[goarioke@fedora lab05]$ ls
hello.asm  hello.o  list.lst  obj.o

```

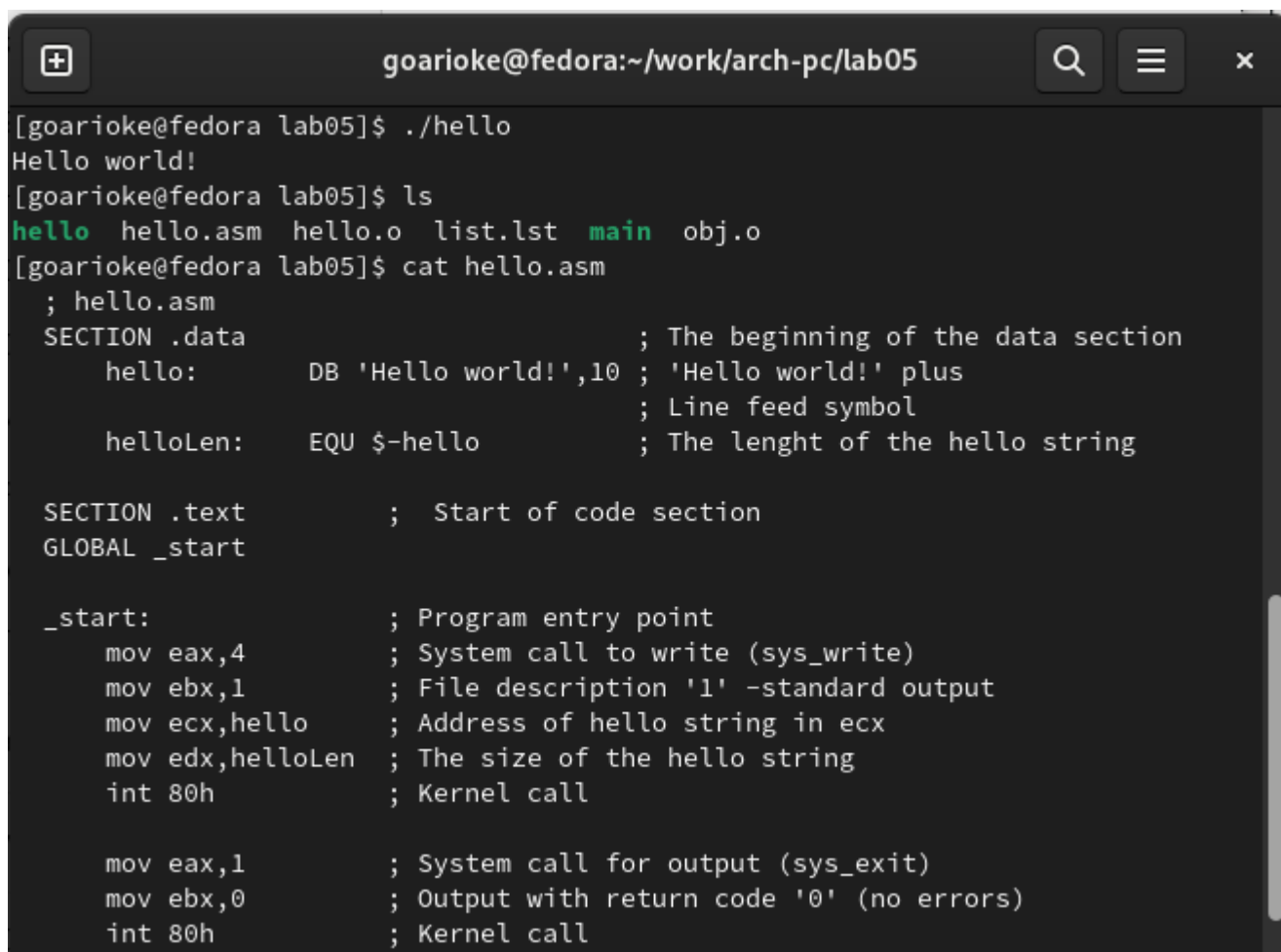
#fig:003

```
[goarioke@fedora lab05]$ ld -m elf_i386 obj.o -o main
[goarioke@fedora lab05]$ ./hello
bash: ./hello: No such file or directory
[goarioke@fedora lab05]$ ld -m elf_i386 obj.o -o hello
```

#fig:004

width=70%}

3. Запуск исполняемого файла Я выполнил сгенерированный исполняемый файл, расположенный в каталоге. (рис.??)



A terminal window titled "goarioke@fedora:~/work/arch-pc/lab05" with search, menu, and close icons. The terminal shows the execution of a program and the contents of its source file.

```
[goarioke@fedora lab05]$ ./hello
Hello world!
[goarioke@fedora lab05]$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
[goarioke@fedora lab05]$ cat hello.asm
; hello.asm
SECTION .data                                ; The beginning of the data section
    hello:      DB 'Hello world!',10         ; 'Hello world!' plus
                                                ; Line feed symbol
    helloLen:   EQU $-hello                  ; The lenght of the hello string

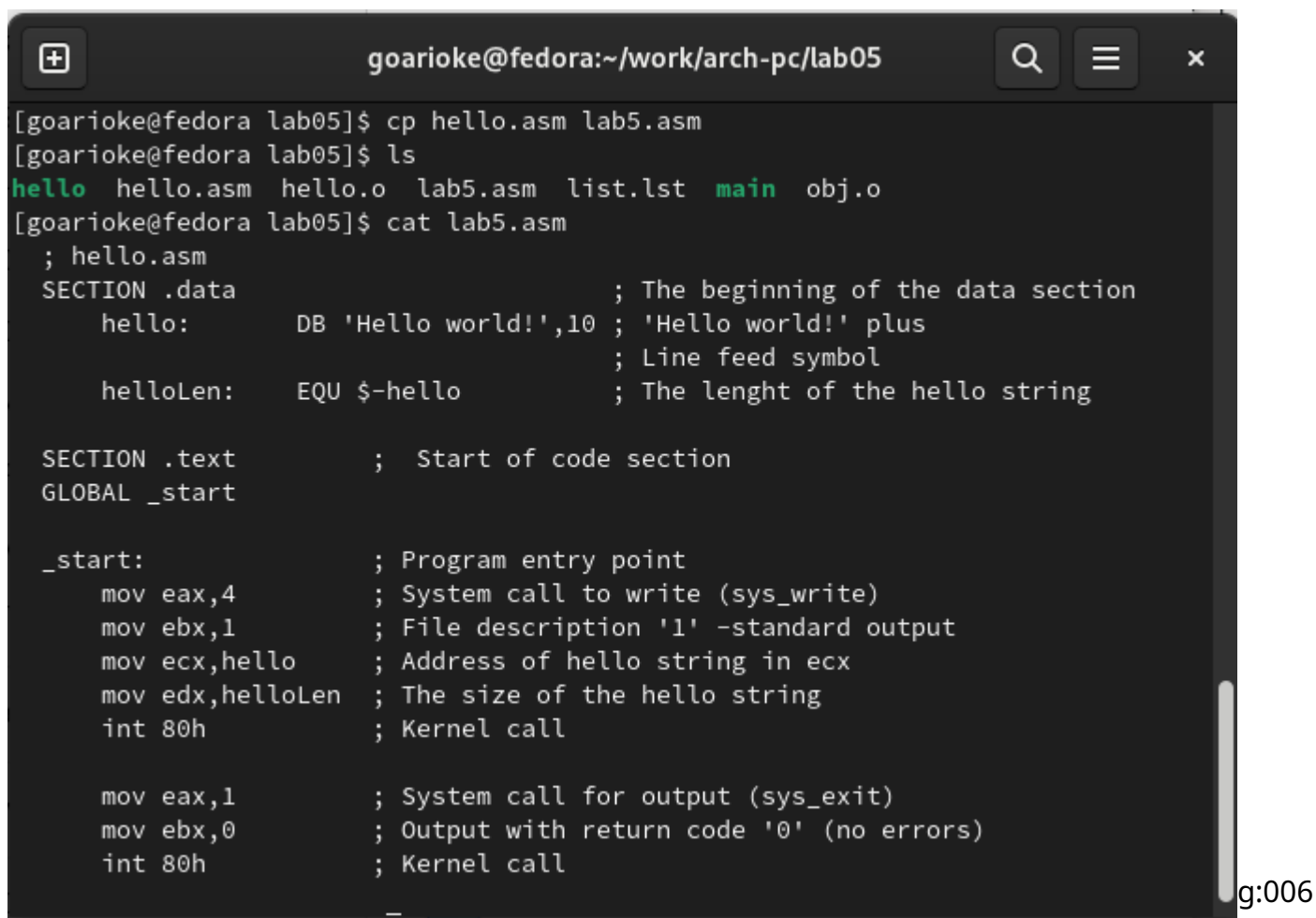
SECTION .text                                ; Start of code section
GLOBAL _start

_start:                                       ; Program entry point
    mov eax,4                                ; System call to write (sys_write)
    mov ebx,1                                ; File description '1' -standard output
    mov ecx,hello                            ; Address of hello string in ecx
    mov edx,helloLen                        ; The size of the hello string
    int 80h                                  ; Kernel call

    mov eax,1                                ; System call for output (sys_exit)
    mov ebx,0                                ; Output with return code '0' (no errors)
    int 80h                                  ; Kernel call
```

g:005

4. Задание для самостоятельной работы
5. В каталоге ~/work/arch-pc/lab05 я использовал команду `sr` для создания файла `hello.asm` с именем `lab5.asm`. (рис.??)

A terminal window titled 'goarioke@fedora:~/work/arch-pc/lab05' with search, menu, and close icons. It shows the execution of 'cp hello.asm lab5.asm', 'ls', and 'cat lab5.asm'. The output of 'cat' shows assembly code for a 'Hello world!' program. The code includes a data section with a string and its length, and a text section with the program's entry point and system calls for writing to stdout and exiting. A vertical scrollbar is on the right, and the text 'g:006' is at the bottom right.

```
[goarioke@fedora lab05]$ cp hello.asm lab5.asm
[goarioke@fedora lab05]$ ls
hello  hello.asm  hello.o  lab5.asm  list.lst  main  obj.o
[goarioke@fedora lab05]$ cat lab5.asm
; hello.asm
SECTION .data                ; The beginning of the data section
    hello:                   DB 'Hello world!',10 ; 'Hello world!' plus
                                ; Line feed symbol
    helloLen:                EQU $-hello          ; The lenght of the hello string

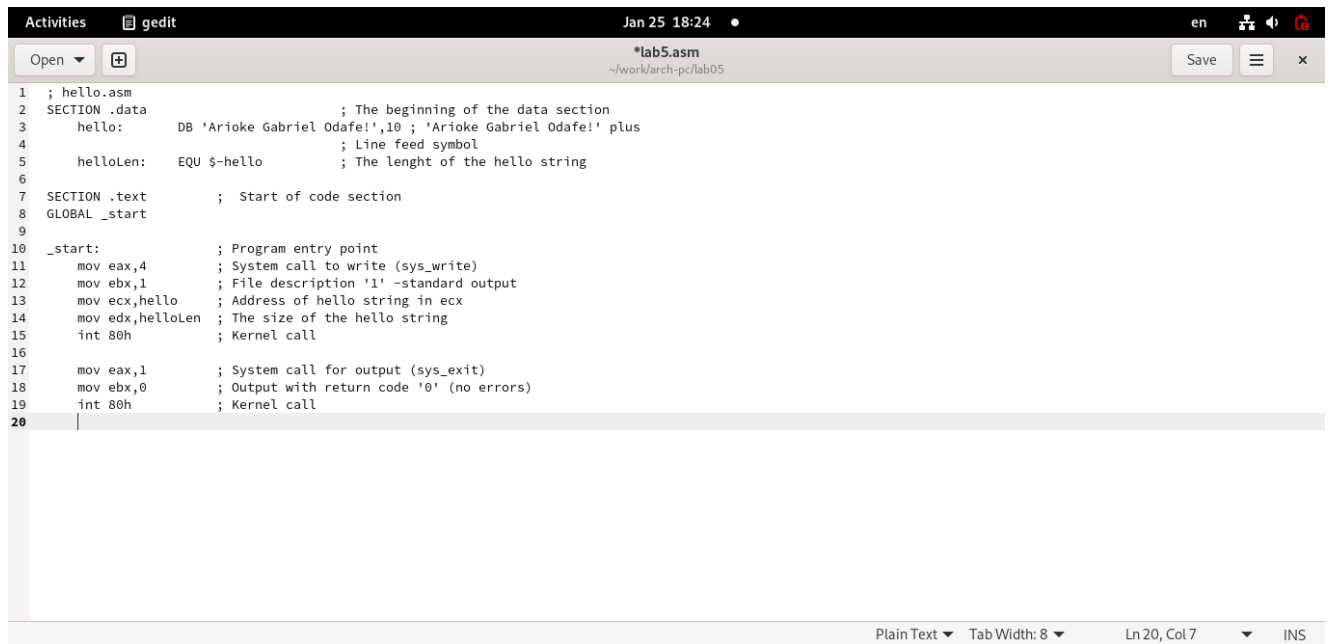
SECTION .text                ; Start of code section
GLOBAL _start

_start:                      ; Program entry point
    mov eax,4                ; System call to write (sys_write)
    mov ebx,1                ; File description '1' -standard output
    mov ecx,hello            ; Address of hello string in ecx
    mov edx,helloLen         ; The size of the hello string
    int 80h                  ; Kernel call

    mov eax,1                ; System call for output (sys_exit)
    mov ebx,0                ; Output with return code '0' (no errors)
    int 80h                  ; Kernel call
```

width=70%}

2. С помощью текстового редактора я отредактировал текст программы в файле lab5.asm, чтобы вместо Hello world! на экране отображалась строка моего имени и фамилии.(рис.??)

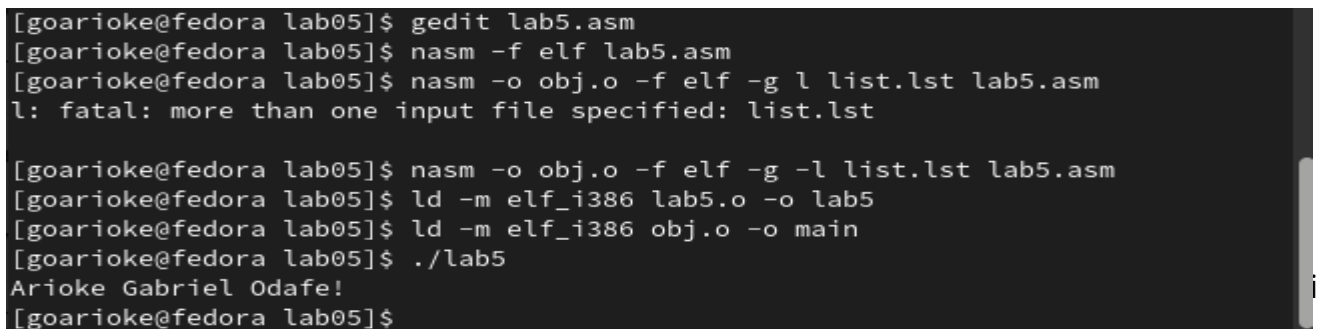


```
1 ; hello.asm
2 SECTION .data          ; The beginning of the data section
3     hello:              DB 'Arioke Gabriel Odafe!',10 ; 'Arioke Gabriel Odafe!' plus
4                          ; Line feed symbol
5     helloLen:           EQU $-hello          ; The lenght of the hello string
6
7 SECTION .text           ; Start of code section
8 GLOBAL _start
9
10 _start:                 ; Program entry point
11     mov eax,4            ; System call to write (sys_write)
12     mov ebx,1            ; File description '1' -standard output
13     mov ecx,hello        ; Address of hello string in ecx
14     mov edx,helloLen     ; The size of the hello string
15     int 80h             ; Kernel call
16
17     mov eax,1            ; System call for output (sys_exit)
18     mov ebx,0            ; Output with return code '0' (no errors)
19     int 80h             ; Kernel call
20
```

#fig:007

width=70%}

3. Я перевел получившийся текст программы lab5.asm в объектный файл. Связал объект и запустил полученный исполняемый файл.(рис.??, ??)



```
[goarioke@fedora lab05]$ gedit lab5.asm
[goarioke@fedora lab05]$ nasm -f elf lab5.asm
[goarioke@fedora lab05]$ nasm -o obj.o -f elf -g -l list.lst lab5.asm
l: fatal: more than one input file specified: list.lst

[goarioke@fedora lab05]$ nasm -o obj.o -f elf -g -l list.lst lab5.asm
[goarioke@fedora lab05]$ ld -m elf_i386 lab5.o -o lab5
[goarioke@fedora lab05]$ ld -m elf_i386 obj.o -o main
[goarioke@fedora lab05]$ ./lab5
Arioke Gabriel Odafe!
[goarioke@fedora lab05]$
```

ig:008

width=70%}

```
goarioke@fedora:~/work/arch-pc/lab05
[goarioke@fedora lab05]$ ls
hello  hello.asm  hello.o  lab5  lab5.asm  lab5.o  list.lst  main  obj.o
[goarioke@fedora lab05]$ cat lab5.asm
; hello.asm
SECTION .data                                ; The beginning of the data section
    hello:      DB 'Arioke Gabriel Odafe!',10 ; 'Arioke Gabriel Odafe!' plus
                                                ; Line feed symbol
    helloLen:   EQU $-hello                  ; The lenght of the hello string

SECTION .text                                ; Start of code section
GLOBAL _start

_start:                                       ; Program entry point
    mov eax,4                                ; System call to write (sys_write)
    mov ebx,1                                ; File description '1' -standard output
    mov ecx,hello                            ; Address of hello string in ecx
    mov edx,helloLen                        ; The size of the hello string
    int 80h                                  ; Kernel call

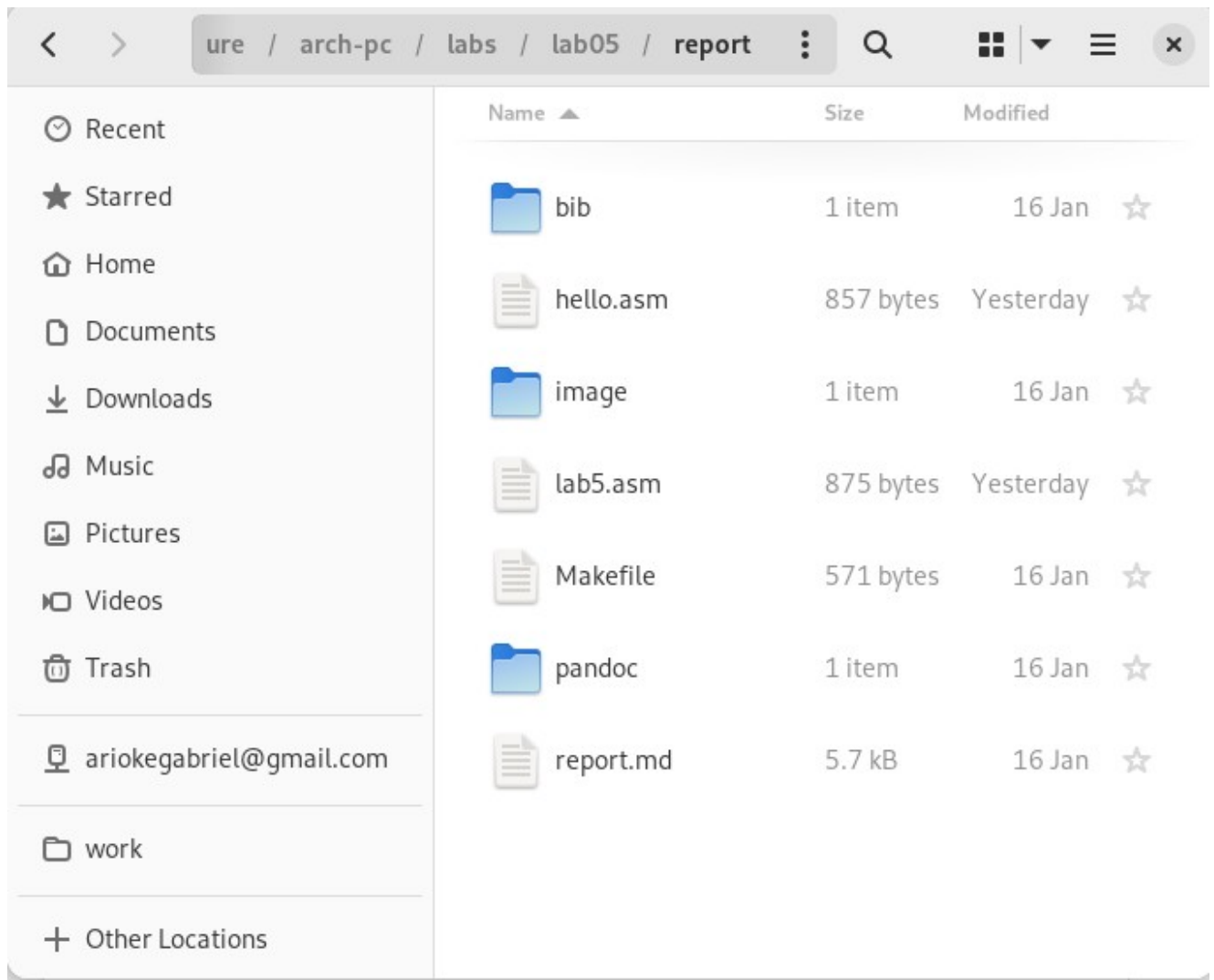
    mov eax,1                                ; System call for output (sys_exit)
    mov ebx,0                                ; Output with return code '0' (no errors)
    int 80h                                  ; Kernel call

[goarioke@fedora lab05]$
```

#fig:009

width=70%}

4. Я скопировал файлы hello.asm и lab5.asm в свой локальный репозиторий, а также загрузил файлы на github.(рис.??



```
goarioke@fedora:~/work/study/2022-2023/Computer Architect...  
[goarioke@fedora ~]$ cd ~/work/study/2022-2023/'Computer Architecture'/arch-pc  
[goarioke@fedora arch-pc]$ git add .  
[goarioke@fedora arch-pc]$ git commit -am 'add report lab 5'  
[master 16cafd6] add report lab 5  
4 files changed, 40 insertions(+)  
create mode 100644 labs/lab05/report/hello.asm  
create mode 100644 labs/lab05/report/lab5.asm  
[goarioke@fedora arch-pc]$ git push  
Enumerating objects: 19, done.  
Counting objects: 100% (19/19), done.  
Compressing objects: 100% (11/11), done.  
Writing objects: 100% (11/11), 1.85 KiB | 3.00 KiB/s, done.  
Total 11 (delta 6), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (6/6), completed with 5 local objects.  
To github.com:goarioke/study_2022-2023_arch-pc.git  
47dbfa6..16cafd6 master -> master  
[goarioke@fedora arch-pc]$
```


5 Выводы

В ходе этой лабораторной работы я приобрел практические навыки освоения процедур компиляции и ассемблера программ, написанных на ассемблере NASM.

Список литературы