

Отчет по лабораторной работе №4

Костеренко Полина

Содержание

1	1 Цель работы	5
2	2 Порядок выполнения лабораторной работы	6
3	2.1 Программа Hello world!	7
4	2.2 Транслятор NASM	9
5	2.3 Расширенный синтаксис командной строки NASM	10
6	2.4 компоновщик LD	11
7	2.5 Запуск исполняемого файла	12
8	3 Задание для самостоятельной работы	13
9	3.5 Выводы	18

Список иллюстраций

3.1	1	7
3.2	2	7
3.3	3	8
3.4	4	8
4.1	5	9
5.1	6	10
6.1	7	11
6.2	8	11
7.1	9	12
8.1	10	13
8.2	11	14
8.3	12	14
8.4	13	15
8.5	14	15
8.6	15	15
8.7	16	16
8.8	17	16
8.9	18	16
8.10	19	17

Список таблиц

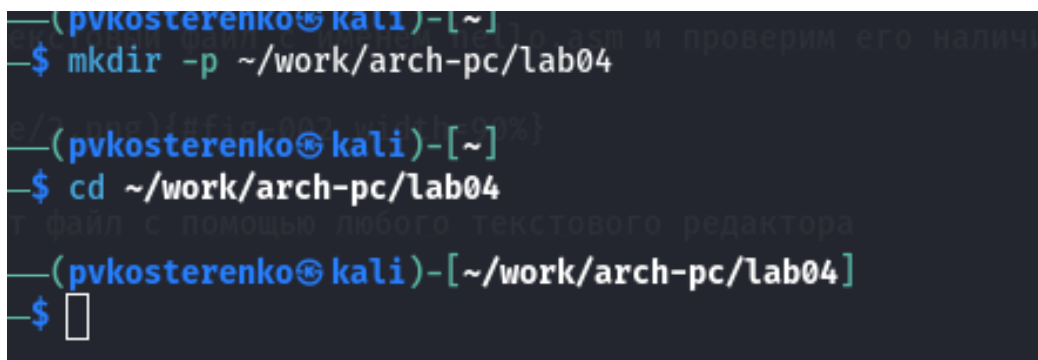
1 1 Цель работы

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

2 2 Порядок выполнения лабораторной работы

3 2.1 Программа Hello world!

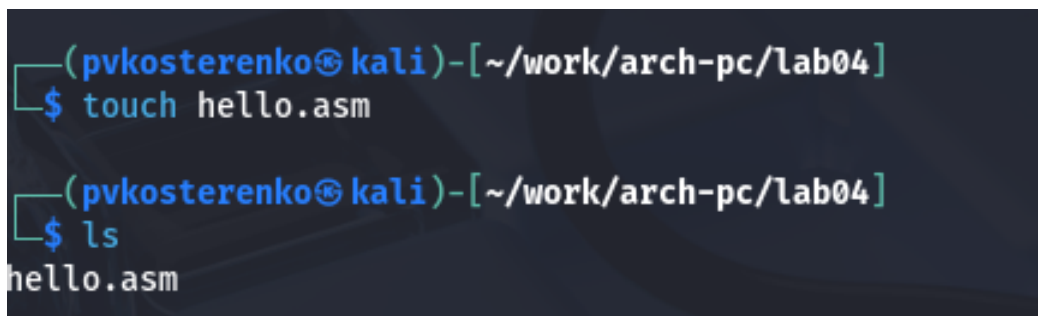
Создадим каталог для работы с программами на языке ассемблера NASM и перейдем в созданный каталог рис. 3.1



```
(pvkosterenko@kali)-[~]  
$ mkdir -p ~/work/arch-pc/lab04  
  
(pvkosterenko@kali)-[~]  
$ cd ~/work/arch-pc/lab04  
  
(pvkosterenko@kali)-[~/work/arch-pc/lab04]  
$
```

Рисунок 3.1: 1

Создадим текстовый файл с именем hello.asm и проверим его наличие рис. 3.2



```
(pvkosterenko@kali)-[~/work/arch-pc/lab04]  
$ touch hello.asm  
  
(pvkosterenko@kali)-[~/work/arch-pc/lab04]  
$ ls  
hello.asm
```

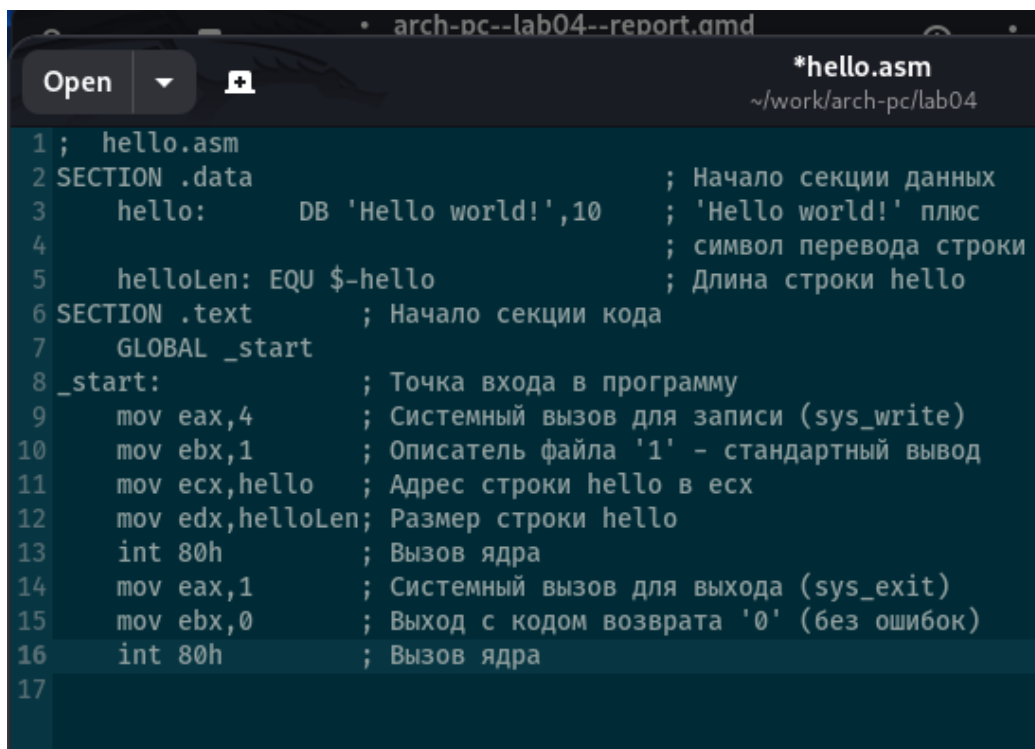
Рисунок 3.2: 2

Откроем этот файл с помощью любого текстового редактора рис. 3.3

```
(pvkosterenko@kali)-[~/work/arch-pc/lab04]
$ gedit hello.asm
```

Рисунок 3.3: 3

Введем в него следующий текст рис. 3.4

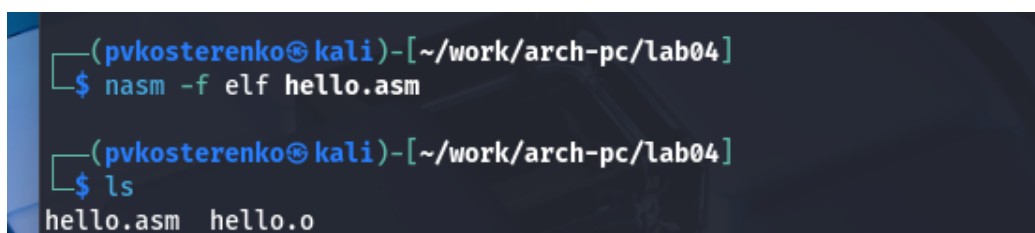


```
1 ; hello.asm
2 SECTION .data
3     hello:      DB 'Hello world!',10      ; Начало секции данных
4                                           ; 'Hello world!' плюс
5     helloLen:   EQU $-hello              ; символ перевода строки
6                                           ; Длина строки hello
7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

Рисунок 3.4: 4

4 2.2 Транслятор NASM

Для компиляции приведённого выше текста программы «Hello World» необходимо написать `nasm -f elf hello.asm` и проверим объектный файл был создан рис. 4.1

A terminal window with a dark background and light blue text. The prompt is `(pvkosterenko@kali) - [~/work/arch-pc/lab04]`. The first command entered is `$ nasm -f elf hello.asm`. The second prompt is the same, and the command entered is `$ ls`. The output of the `ls` command is `hello.asm hello.o`.

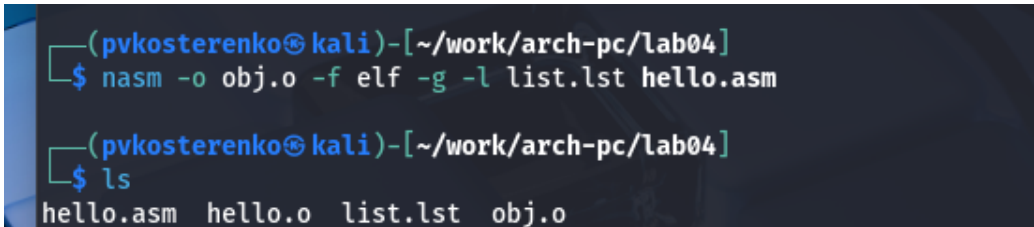
```
(pvkosterenko@kali) - [~/work/arch-pc/lab04]
$ nasm -f elf hello.asm

(pvkosterenko@kali) - [~/work/arch-pc/lab04]
$ ls
hello.asm hello.o
```

Рисунок 4.1: 5

5 2.3 Расширенный синтаксис командной строки NASM

Выполним следующую команду `nasm -o obj.o -f elf -g -l list.lst hello.asm` и проверим его наличие в каталоге рис. 5.1



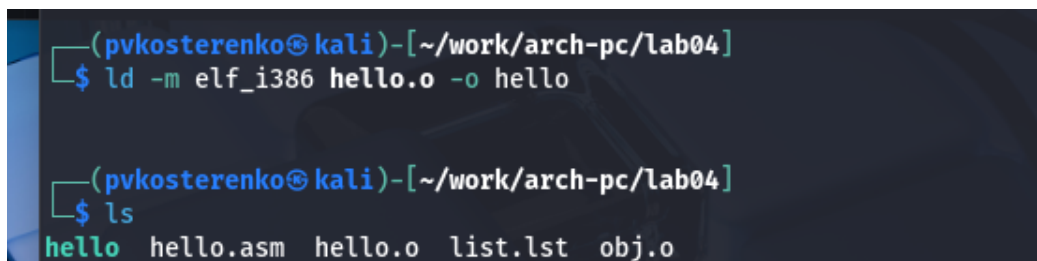
```
(pvkosterenko@kali)-[~/work/arch-pc/lab04]
$ nasm -o obj.o -f elf -g -l list.lst hello.asm

(pvkosterenko@kali)-[~/work/arch-pc/lab04]
$ ls
hello.asm  hello.o  list.lst  obj.o
```

Рисунок 5.1: 6

6 2.4 Компоновщик LD

Объектный файл передадим на обработку компоновщику и проверим рис. 6.1

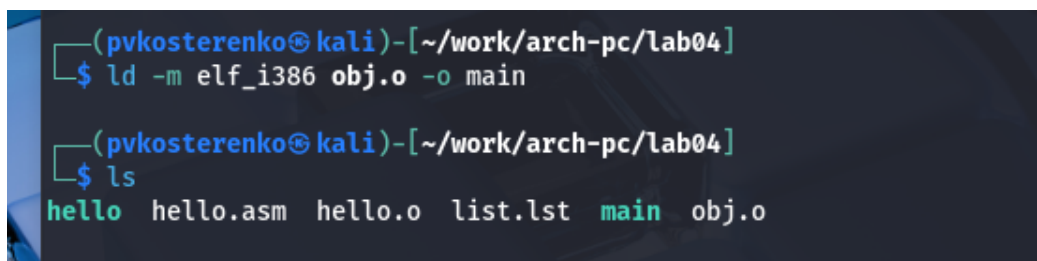


```
(pvkosterenko@kali)-[~/work/arch-pc/lab04]
$ ld -m elf_i386 hello.o -o hello

(pvkosterenko@kali)-[~/work/arch-pc/lab04]
$ ls
hello hello.asm hello.o list.lst obj.o
```

Рисунок 6.1: 7

Выполним следующую команду `ld -m elf_i386 obj.o -o main` и проверим рис. 6.2



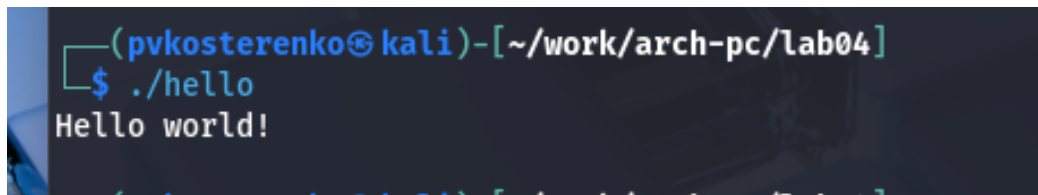
```
(pvkosterenko@kali)-[~/work/arch-pc/lab04]
$ ld -m elf_i386 obj.o -o main

(pvkosterenko@kali)-[~/work/arch-pc/lab04]
$ ls
hello hello.asm hello.o list.lst main obj.o
```

Рисунок 6.2: 8

7 2.5 Запуск исполняемого файла

Наберем в командной строке `./hello` и проверим что выводит рис. 7.1

A terminal window with a dark background. The prompt is `(pvkosterenko@kali) - [~/work/arch-pc/lab04]`. The user enters `$./hello` and the output is `Hello world!`.

```
(pvkosterenko@kali) - [~/work/arch-pc/lab04]  
$ ./hello  
Hello world!
```

Рисунок 7.1: 9

8 3 Задание для самостоятельной работы

3.1 1 Задание В каталоге `~/work/arch-pc/lab04` с помощью команды `cp` создайте копию файла `hello.asm` с именем `lab4.asm`. Скопируем в `~/work/arch-pc/lab04` файлы `hello.asm` с именем `lab4.asm` и проверяем каталог рис. 8.1

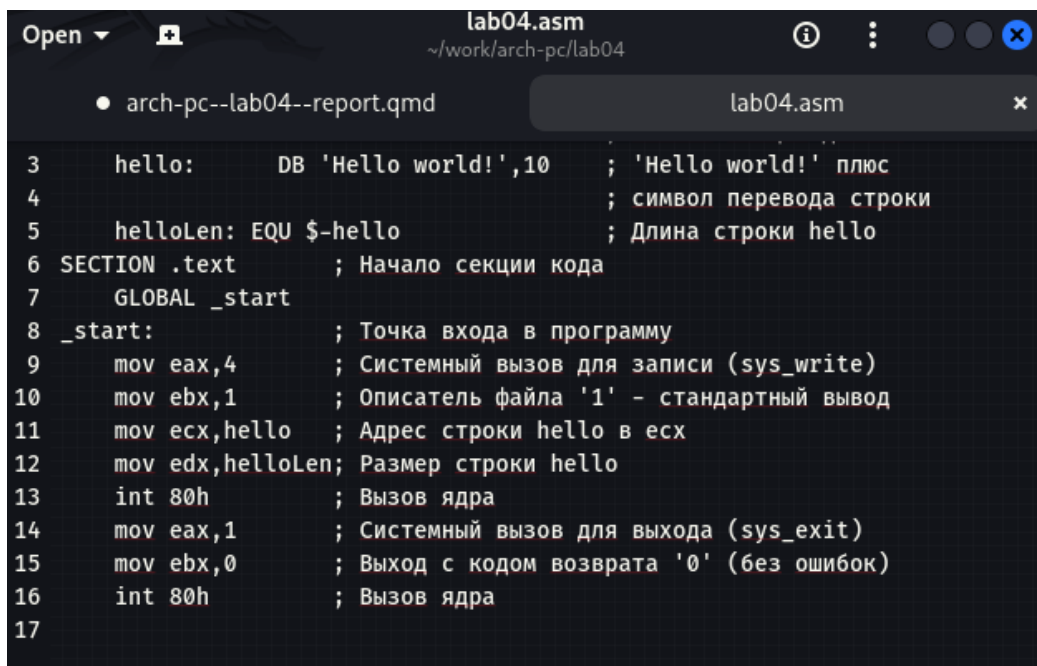


```
(pvkosterenko@kali)-[~/work/arch-pc/lab04]
$ cp hello.asm lab04.asm

(pvkosterenko@kali)-[~/work/arch-pc/lab04]
$ ls
hello  hello.asm  hello.o  lab04.asm  list.lst  main  obj.o
```

Рисунок 8.1: 10

3.2 2 Задание С помощью любого текстового редактора внесите изменения в текст программы в файле `lab4.asm` так, чтобы вместо `Hello world!` на экран выводилась строка с вашими фамилией и именем С помощью `gedit` открываем файл рис. 8.2



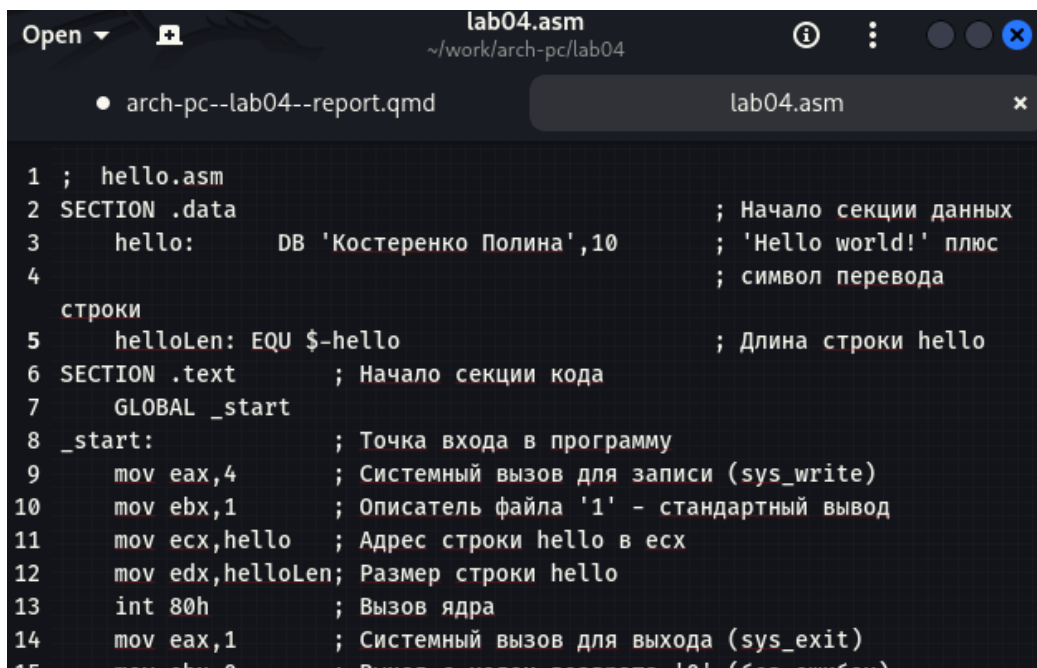
```
lab04.asm
~/work/arch-pc/lab04

● arch-pc--lab04--report.qmd lab04.asm x

3  hello:      DB 'Hello world!',10      ; 'Hello world!' 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
```

Рисунок 8.2: 11

Редактируем файл рис. 8.3



```
lab04.asm
~/work/arch-pc/lab04

● arch-pc--lab04--report.qmd lab04.asm x

1 ; hello.asm
2 SECTION .data                         ; Начало секции данных
3  hello:      DB 'Костеренко Полина',10 ; 'Hello world!' 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)
```

Рисунок 8.3: 12

3.3 3 Задание Оттранслируйте полученный текст программы lab4.asm в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл. Оттранслируем полученный текст программы lab4.asm в объектный файл и проверим файлрис. 8.4

```
(pvkosterenko@kali)-[~/work/arch-pc/lab04]
$ nasm -f elf lab04.asm

(pvkosterenko@kali)-[~/work/arch-pc/lab04]
$ ls
hello hello.asm hello.o lab04.asm lab04.o list.lst main obj.o
```

Рисунок 8.4: 13

Выполним компоновку объектного файла рис. 8.5

```
(pvkosterenko@kali)-[~/work/arch-pc/lab04]
$ ld -m elf_i386 lab04.o -o lab04

(pvkosterenko@kali)-[~/work/arch-pc/lab04]
$ ls
hello hello.asm hello.o lab04 lab04.asm lab04.o list.lst main obj.o
```

Рисунок 8.5: 14

Запустим получившийся исполняемый файл рис. 8.6

```
(pvkosterenko@kali)-[~/work/arch-pc/lab04]
$ ./lab04
Костеренко Полина
```

Рисунок 8.6: 15

3.4 4 Задание 4. Скопируйте файлы hello.asm и lab4.asm в Ваш локальный репозиторий в каталог ~/work/study/2023-2024/«Архитектура компьютера»/arch-pc/labs/lab04/. Загрузите файлы на Github.

Скопируем и проверим файлы hello.asm и lab4.asm в локальный репозиторий в каталог ~/work/study/2023-2024/«Arhitektura comp»/arch-pc/labs/lab04/ рис. 8.7

```

(pvkosterenko@kali)-[~/work/arch-pc/lab04]
$ cp hello.asm ~/work/study/2025-2026/"Arhitektura comp"/arch-pc/labs/lab04 88
cp lab04.asm ~/work/study/2025-2026/"Arhitektura comp"/arch-pc/labs/lab04

(pvkosterenko@kali)-[~/work/arch-pc/lab04]
$ ls
hello hello.asm hello.o lab04 lab04.asm lab04.o list.lst main obj.o

```

Рисунок 8.7: 16

Проверим рис. 8.8

```

(pvkosterenko@kali)-[~/work/arch-pc/lab04]
$ ls ~/work/study/2025-2026/"Arhitektura comp"/arch-pc/labs/lab04
hello.asm lab04.asm presentation report

```

Рисунок 8.8: 17

Загрузим файлы на Github рис. 8.9

```

(pvkosterenko@kali)-[~]
$ cd work/study/2025-2026/"Arhitektura comp"/arch-pc/labs/lab04

(pvkosterenko@kali)-[~/../Arhitektura comp/arch-pc/labs/lab04]
$ git add hello.asm lab04.asm

(pvkosterenko@kali)-[~/../Arhitektura comp/arch-pc/labs/lab04]
$ git commit -m "add lab04 assembly files"
[master 8da7917] add lab04 assembly files
2 files changed, 34 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab04.asm

(pvkosterenko@kali)-[~/../Arhitektura comp/arch-pc/labs/lab04]
$ git push

```

Рисунок 8.9: 18

Проверяем в Github загруженные файлы рис. 8.10

study_2025-2026_arh-pc.git-arch-pc / labs / lab04 /

prkosterenko

add lab04 assembly files

8da7917 · 1 minute ago

History

Name	Last commit message	Last commit date
..		
presentation	feat(main): make course structure	last month
report	feat(main): make course structure	last month
hello.asm	add lab04 assembly files	1 minute ago
lab04.asm	add lab04 assembly files	1 minute ago

Рисунок 8.10: 19

9 3.5 Выводы

В рамках лабораторной работы были успешно освоены ключевые этапы разработки программ на ассемблере NASM: создание исходного кода, трансляция с помощью компилятора NASM, компоновка объектных файлов через компоновщик ld и выполнение результирующих исполняемых файлов. В процессе изучения были рассмотрены основные ассемблерные директивы и механизмы системных вызовов ОС Linux для реализации операций ввода-вывода. Приобретённые компетенции позволяют разрабатывать низкоуровневые приложения с прямым управлением системными ресурсами.