

Отчёт по лабораторной работе №4

дисциплина: Архитектура компьютера

Лаптев Тимофей Сергеевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
4.1	Создание программы Hello world!	10
4.2	Работа с транслятором NASM	12
4.3	Работа с расширенным синтаксисом командной строки NASM . . .	12
4.4	Работа с компоновщиком LD	13
4.5	Запуск исполняемого файла	14
5	Выполнение заданий для самостоятельной работы.	15
6	Выводы	17
7	Список литературы	18

Список иллюстраций

4.1	Создание дерикторий и перемещение между ними	10
4.2	Создание пустого файла и его открытие	11
4.3	ЗАполнение файла	11
4.4	Компиляция текста программы	12
4.5	Компиляция текста программы	13
4.6	Передача объектного файла на обработку компоновщику	13
4.7	Запуск исполняемого файла	14
5.1	Изменение программы	16
5.2	Компиляция и компоновка текста программы, также ее запуск . .	16

Список таблиц

1 Цель работы

Цель данной лабораторной работы - освоить процедуры компиляции и сборки программ, написанных на ассемблере NASM.

2 Задание

- Создание программы Hello world!
- Работа с транслятором NASM
- Работа с расширенным синтаксисом командной строки NASM
- Работа с компоновщиком LD
- Запуск исполняемого файла
- 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

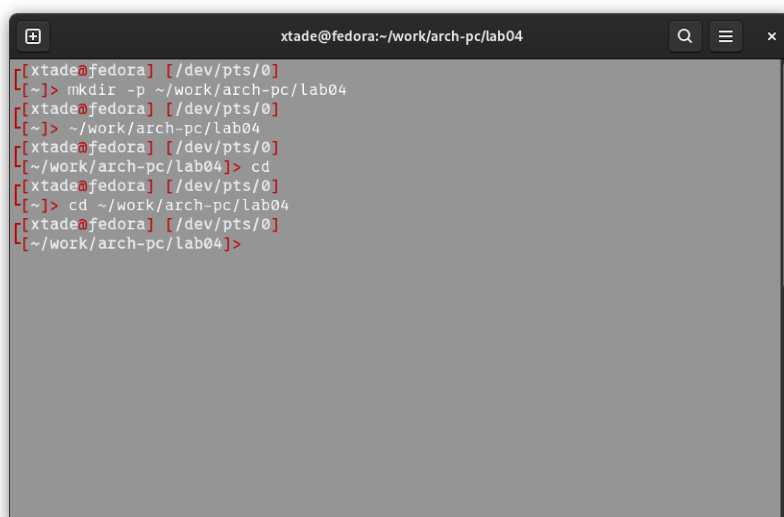
следующей команде.

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 Создание программы Hello world!

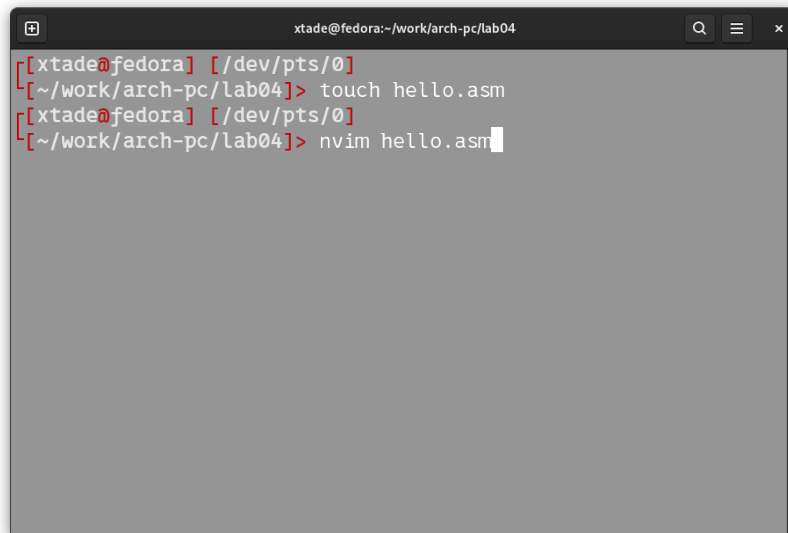
Создаю каталог и с помощью утилиты `cd` перемещаюсь в него (рис. 4.1).



```
xtade@fedora: ~/work/arch-pc/lab04
[xtade@fedora] [/dev/pts/0]
[~]> mkdir -p ~/work/arch-pc/lab04
[xtade@fedora] [/dev/pts/0]
[~]> ~/work/arch-pc/lab04
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab04]> cd
[xtade@fedora] [/dev/pts/0]
[~]> cd ~/work/arch-pc/lab04
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab04]>
```

Рис. 4.1: Создание дерикторий и перемещение между ними

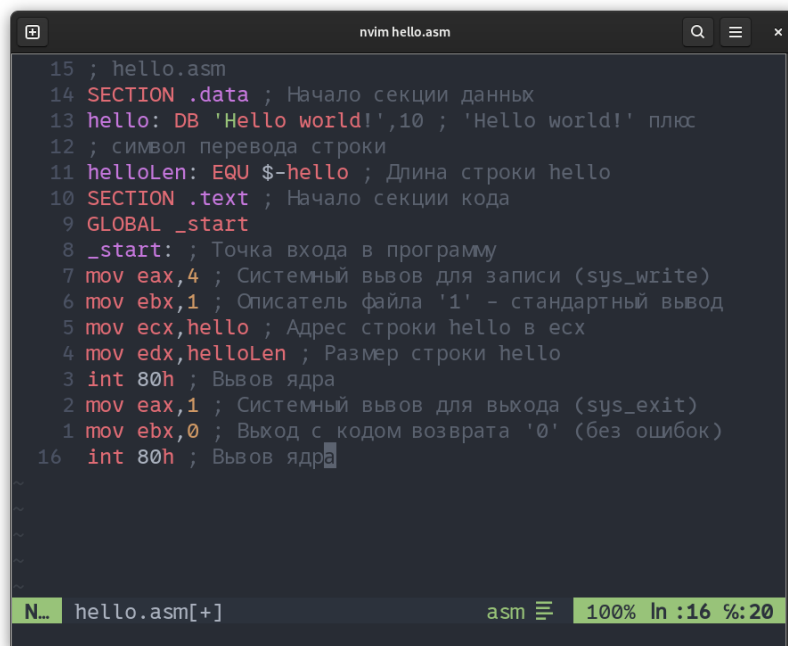
Создаю в текущем каталоге пустой текстовый файл `hello.asm` с помощью утилиты `touch` и открываю созданный файл в текстовом редакторе `neovim` (рис. 4.2).



```
xtade@fedora: ~/work/arch-pc/lab04
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab04]> touch hello.asm
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab04]> nvim hello.asm
```

Рис. 4.2: Создание пустого файла и его открытие

Заполняю файл, вставляя в него программу для вывода “Hello word!” (рис. 4.3).




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

Рис. 4.3: 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. Далее проверяю правильность выполнения команды с помощью утилиты `ls`: действительно, создан файл “hello.o” (рис. 4.4).

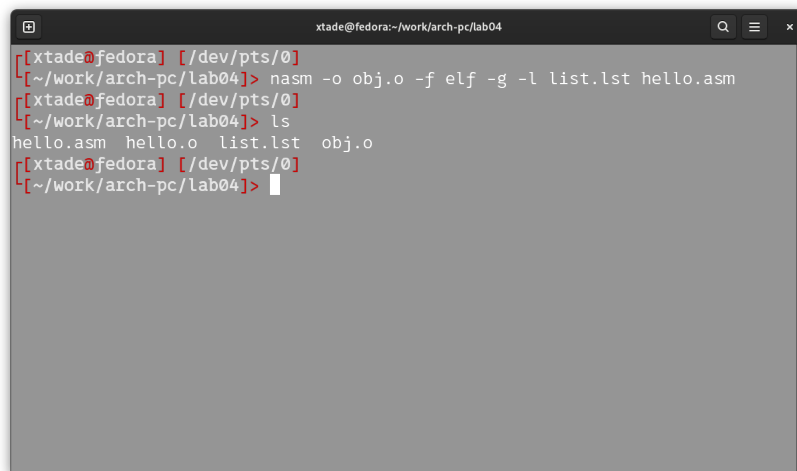


```
xtade@fedora:~/work/arch-pc/lab04
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab04]> nasm -f elf hello.asm
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab04]> ls
hello.asm  hello.o
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab04]>
```

Рис. 4.4: 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. 4.5). Далее проверяю с помощью утилиты `ls` правильность выполнения команды.

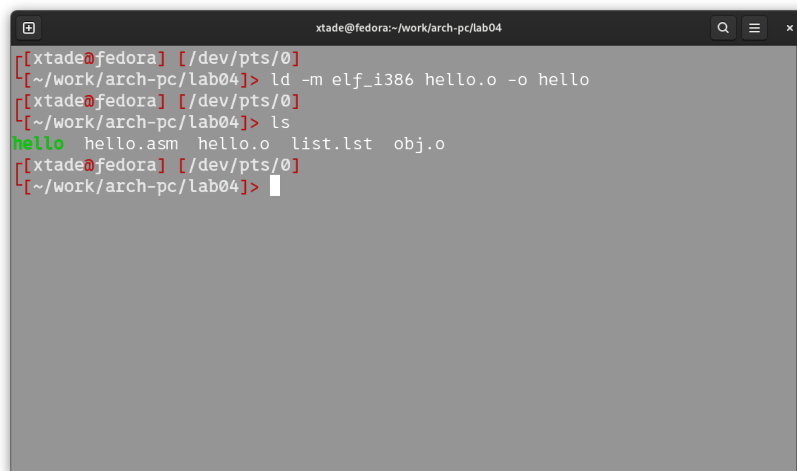


```
xtade@fedora:~/work/arch-pc/lab04
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab04]> nasm -o obj.o -f elf -g -l list.lst hello.asm
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab04]> ls
hello.asm hello.o list.lst obj.o
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab04]>
```

Рис. 4.5: Компиляция текста программы

4.4 Работа с компоновщиком LD

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello. Ключ -o задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты ls правильность выполнения команды (рис. 4.6).



```
xtade@fedora:~/work/arch-pc/lab04
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab04]> ld -m elf_i386 hello.o -o hello
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab04]> ls
hello hello.asm hello.o list.lst obj.o
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab04]>
```

Рис. 4.6: Передача объектного файла на обработку компоновщику

4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello (рис. 4.7).

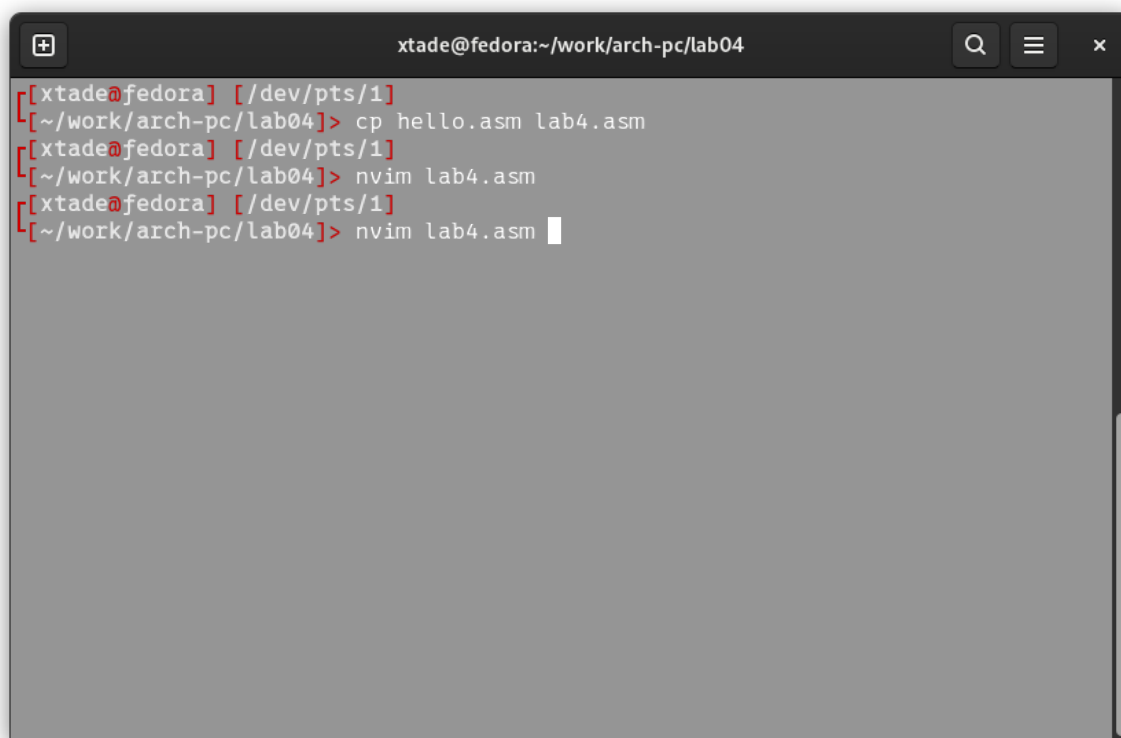
A terminal window with a dark background and light text. The window title is 'xtade@fedora: ~/work/arch-pc/lab04'. The prompt is '[xtade@fedora] [/dev/pts/0]'. The user enters the command './hello' and the terminal outputs 'Hello world!'. The prompt changes to '[xtade@fedora] [/dev/pts/0]' and then to '[~/work/arch-pc/lab04]>' after a newline.

```
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab04]> ./hello
Hello world!
[xtade@fedora] [/dev/pts/0]
[~/work/arch-pc/lab04]>
```

Рис. 4.7: Запуск исполняемого файла

5 Выполнение заданий для самостоятельной работы.

С помощью утилиты `ср` создаю в текущем каталоге копию файла `hello.asm` с именем `lab4.asm`. С помощью текстового редактора `mousepad` открываю файл `lab4.asm` (рис. ??).

A terminal window titled 'xtade@fedora:~/work/arch-pc/lab04'. The window shows a sequence of commands and their outputs. The first command is '[xtade@fedora] [/dev/pts/1]', followed by '[~/work/arch-pc/lab04]> cp hello.asm lab4.asm'. The second command is '[xtade@fedora] [/dev/pts/1]', followed by '[~/work/arch-pc/lab04]> nvim lab4.asm'. The third command is '[xtade@fedora] [/dev/pts/1]', followed by '[~/work/arch-pc/lab04]> nvim lab4.asm'. The cursor is at the end of the last command. The window has a search icon, a menu icon, and a close icon in the top right corner.

```
[xtade@fedora] [/dev/pts/1]
[~/work/arch-pc/lab04]> cp hello.asm lab4.asm
[xtade@fedora] [/dev/pts/1]
[~/work/arch-pc/lab04]> nvim lab4.asm
[xtade@fedora] [/dev/pts/1]
[~/work/arch-pc/lab04]> nvim lab4.asm
```

#fig:009width=70% }

Вношу изменения в программу так, чтобы она выводила мои имя и фамилию. (рис. 5.1).

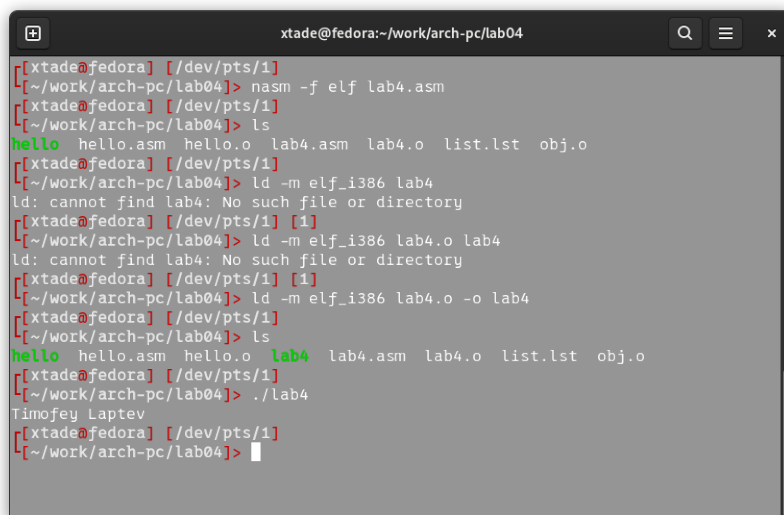


```
2 ; hello.asm
1 SECTION .data ; Начало секции данных
3 hello: DB 'Timofey Laptev',10 ; 'Hello world!' плюс
1 ; символ перевода строки
2 helloLen: EQU $-hello ; Длина строки hello
3 SECTION .text ; Начало секции кода
4 GLOBAL _start
5 _start: ; Точка входа в программу
6 mov eax,4 ; Системный вызов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

COMMAND lab4.asm[+] asm utf-8 18% ln :3/16=25
:wq
```

Рис. 5.1: Изменение программы

Компилирую текст программы в объектный файл. Проверяю с помощью утилиты ls, что файл lab4.o создан. Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4. Запускаю исполняемый файл lab5, на экран действительно выводятся мои имя и фамилия (рис. 5.2).



```
xtade@fedora:~/work/arch-pc/lab04
[xtade@fedora] [/dev/pts/1]
[~/work/arch-pc/lab04]> nasm -f elf lab4.asm
[xtade@fedora] [/dev/pts/1]
[~/work/arch-pc/lab04]> ls
hello hello.asm hello.o lab4.asm lab4.o list.lst obj.o
[xtade@fedora] [/dev/pts/1]
[~/work/arch-pc/lab04]> ld -m elf_i386 lab4
ld: cannot find lab4: No such file or directory
[xtade@fedora] [/dev/pts/1] [1]
[~/work/arch-pc/lab04]> ld -m elf_i386 lab4.o lab4
ld: cannot find lab4: No such file or directory
[xtade@fedora] [/dev/pts/1] [1]
[~/work/arch-pc/lab04]> ld -m elf_i386 lab4.o -o lab4
[xtade@fedora] [/dev/pts/1]
[~/work/arch-pc/lab04]> ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst obj.o
[xtade@fedora] [/dev/pts/1]
[~/work/arch-pc/lab04]> ./lab4
Timofey Laptev
[xtade@fedora] [/dev/pts/1]
[~/work/arch-pc/lab04]>
```

Рис. 5.2: Компиляция и компоновка текста программы, также ее запуск

6 Выводы

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

7 Список литературы

1. https://esystem.rudn.ru/pluginfile.php/1584628/mod_resource/content/1/%D0%9B%D0%B0%