

# **Отчёта по лабораторной работе №4**

**Дисциплина: архитектура компьютера**

Ицков Андрей Станиславович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	Создание программы Hello world! . . . . .	9
4.2	Работа с транслятором NASM . . . . .	9
4.3	Работа с расширенным синтаксисом командной строки NASM . .	10
4.4	Работа с компоновщиком LD . . . . .	10
4.5	Запуск исполняемого файла . . . . .	11
4.6	Выполнение заданий для самостоятельной работы. . . . .	11
<b>5</b>	<b>Выводы</b>	<b>14</b>
<b>6</b>	<b>Список литературы</b>	<b>15</b>

## Список иллюстраций

4.1	Перемещение и создание файла . . . . .	9
4.2	Открытие и заполнение файла . . . . .	9
4.3	Транслирование текста программы . . . . .	10
4.4	Компиляция текста программы . . . . .	10
4.5	Передача объектного файла на обработку компоновщику LD . . .	10
4.6	Передача объектного файла на обработку компоновщику . . . .	11
4.7	Запуск исполняемого файла . . . . .	11
4.8	Копирование файла . . . . .	11
4.9	Редактирование программы . . . . .	11
4.10	Компиляция текста программы . . . . .	12
4.11	Передача объектного файла на обработку компоновщику . . . .	12
4.12	Запуск исполняемого файла . . . . .	12
4.13	Копирование файлов . . . . .	12
4.14	Загрузка файлов на GitHub . . . . .	13

# **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 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.

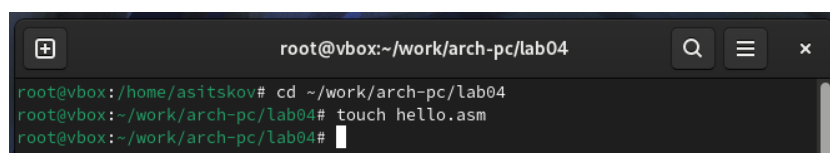
Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. NASM — это открытый проект ассемблера, версии которого доступны под различные операционные системы и который позволяет получать объектные файлы для этих систем. В NASM используется Intel-синтаксис и поддерживаются инструкции x86-64.



## 4 Выполнение лабораторной работы

### 4.1 Создание программы Hello world!

Перемещаюсь в нужный каталог и создаю в нем пустой файл hello.asm с помощью команд cd и touch (рис. 4.1).

A terminal window with a dark background. The title bar shows 'root@vbox:~/work/arch-pc/lab04'. The command history shows: 'root@vbox:~/home/asitskov# cd ~/work/arch-pc/lab04', 'root@vbox:~/work/arch-pc/lab04# touch hello.asm', and 'root@vbox:~/work/arch-pc/lab04#'.

```
root@vbox:~/home/asitskov# cd ~/work/arch-pc/lab04
root@vbox:~/work/arch-pc/lab04# touch hello.asm
root@vbox:~/work/arch-pc/lab04#
```

Рис. 4.1: Перемещение и создание файла

Открываю созданный файл и заполняю его данной программой (рис. 4.2).

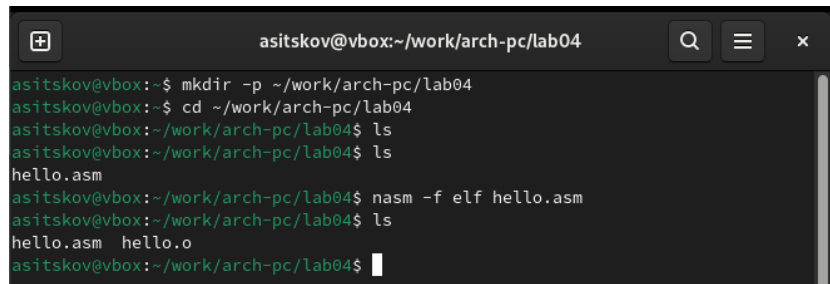
A text editor window titled 'hello.asm' with a light gray background. The file path is '~/.work/arch-pc/lab04'. The code is as follows:

```
1 ; hello.asm
2 SECTION .data
3     hello: DB 'Hello world!',10
4
5     helloLen: EQU $-hello
6
7 SECTION .text
8     GLOBAL _start
9 _start:
10     mov eax,4
11     mov ebx,1
12     mov ecx,hello
13     mov edx,helloLen
14     int 80h
15
16     mov eax,1
17     mov ebx,0
18     int 80h
```

Рис. 4.2: Открытие и заполнение файла

### 4.2 Работа с транслятором NASM

Транслирую текст программы в объектный код с помощью команды `nasm -f elf hello.asm` и проверяю правильность выполнения команды (рис. 4.3).



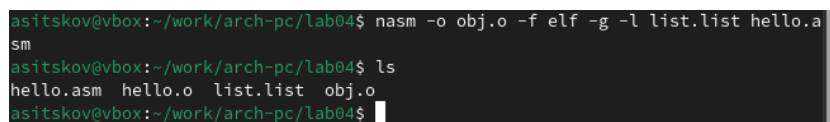
```
asitskov@vbox:~/work/arch-pc/lab04
asitskov@vbox:~$ mkdir -p ~/work/arch-pc/lab04
asitskov@vbox:~$ cd ~/work/arch-pc/lab04
asitskov@vbox:~/work/arch-pc/lab04$ ls
asitskov@vbox:~/work/arch-pc/lab04$ ls
hello.asm
asitskov@vbox:~/work/arch-pc/lab04$ nasm -f elf hello.asm
asitskov@vbox:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
asitskov@vbox:~/work/arch-pc/lab04$
```

Рис. 4.3: Транслирование текста программы

## 4.3 Работа с расширенным синтаксисом командной строки

### NASM

Компилирую файл `hello.asm` в файл `obj.o`, параллельно создавая файл листинга `list.list`, и проверяю правильность выполнения команды (рис. 4.4).

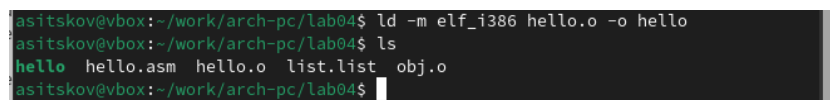


```
asitskov@vbox:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.list hello.a
sm
asitskov@vbox:~/work/arch-pc/lab04$ ls
hello.asm  hello.o  list.list  obj.o
asitskov@vbox:~/work/arch-pc/lab04$
```

Рис. 4.4: Компиляция текста программы

## 4.4 Работа с компоновщиком LD

Передаю объектный файл `hello.o` на обработку компоновщику LD, чтобы получить исполняемый файл `hello` и проверяю правильность выполнения команды (рис. 4.5).



```
asitskov@vbox:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
asitskov@vbox:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  list.list  obj.o
asitskov@vbox:~/work/arch-pc/lab04$
```

Рис. 4.5: Передача объектного файла на обработку компоновщику LD

Создаю исполняемый файл с названием `main` с помощью компоновщика LD и ключа `-o` и проверяю правильность выполнения команды (рис. 4.6).

```
asitskov@vbox:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
asitskov@vbox:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  list.list  main  obj.o
asitskov@vbox:~/work/arch-pc/lab04$
```

Рис. 4.6: Передача объектного файла на обработку компоновщику

## 4.5 Запуск исполняемого файла

Запускаю исполняемый файл hello (рис. 4.7).

```
asitskov@vbox:~/work/arch-pc/lab04$ ./hello
Hello world!
asitskov@vbox:~/work/arch-pc/lab04$
```

Рис. 4.7: Запуск исполняемого файла

## 4.6 Выполнение заданий для самостоятельной работы.

Копирую файл hello.asm и называю его lab4.asm (рис. 4.8).

```
asitskov@vbox:~/work/arch-pc/lab04$ cp hello.asm lab5.asm
asitskov@vbox:~/work/arch-pc/lab04$
```

Рис. 4.8: Копирование файла

Редактирую текст программы lab4.asm так, чтобы при запуске он выводил в терминал мое имя и фамилию (рис. 4.9).



```
1 ; lab5.asm
2 SECTION .data
3     lab5: DB 'Andrei Itskov' ,10
4
5     lab5Len: EQU $-lab5
6
7 SECTION .text
8     GLOBAL _start
9 _start:
10     mov eax,4
11     mov ebx,1
12     mov ecx,lab5
13     mov edx,lab5Len
14     int 80h
15
16     mov eax,1
17     mov ebx,0
18     int 80h
```

Рис. 4.9: Редактирование программы

Компилирую текст программы в объектный файл и проверяю правильность выполнения команды (рис. 4.10).

```
asitskov@vbox:~/work/arch-pc/lab04$ gedit lab5.asm
asitskov@vbox:~/work/arch-pc/lab04$ nasm -f elf lab5.asm
asitskov@vbox:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab5.asm lab5.o list.list main obj.o
```

Рис. 4.10: Компиляция текста программы

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4 и проверяю правильность выполнения команды (рис. 4.11).

```
asitskov@vbox:~/work/arch-pc/lab04$ ld -m elf_i386 lab5.o -o lab5
asitskov@vbox:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab5 lab5.asm lab5.o list.list main obj.o
asitskov@vbox:~/work/arch-pc/lab04$
```

Рис. 4.11: Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4 (рис. 4.12).

```
asitskov@vbox:~/work/arch-pc/lab04$ ./lab5
Andrei Itskov
asitskov@vbox:~/work/arch-pc/lab04$
```

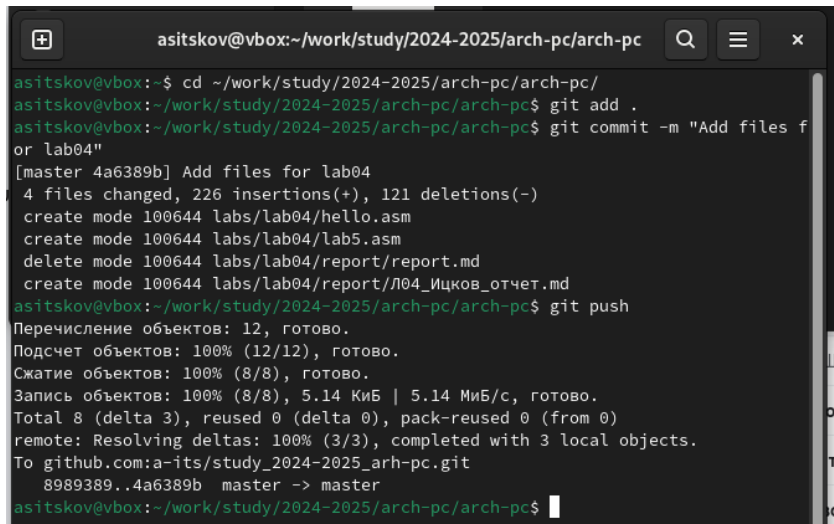
Рис. 4.12: Запуск исполняемого файла

Копирую файлы hello.asm и lab4.asm в нужный каталог с помощью команды cp и проверяю наличие этих файлов с помощью команды ls (рис. 4.13).

```
asitskov@vbox:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab5 lab5.asm lab5.o list.list main obj.o
asitskov@vbox:~/work/arch-pc/lab04$ cp hello.asm ~/work/study/2024-2025/arch-pc/
arch-pc/labs/lab04/
asitskov@vbox:~/work/arch-pc/lab04$ ^[[200~cp hello.asm ~/work/study/2024-2025/a
rch-pc/arch-pc/labs/lab04/
bash: cp: команда не найдена...
asitskov@vbox:~/work/arch-pc/lab04$ cp lab5.asm ~/work/study/2024-2025/arch-pc/a
rch-pc/labs/lab04/
asitskov@vbox:~/work/arch-pc/lab04$ ls ~/work/study/2024-2025/arch-pc/arch-pc/l
abs/lab04/
hello.asm lab5.asm presentation report
asitskov@vbox:~/work/arch-pc/lab04$
```

Рис. 4.13: Копирование файлов

Загружаю файлы на Github с помощью команд git add и git commit и отправляю их на сервер с помощью команды git push (рис. 4.14).



```
asitskov@vbox: ~/work/study/2024-2025/arch-pc/arch-pc
asitskov@vbox:~/work/study/2024-2025/arch-pc/arch-pc$ git add .
asitskov@vbox:~/work/study/2024-2025/arch-pc/arch-pc$ git commit -m "Add files for lab04"
[master 4a6389b] Add files for lab04
 4 files changed, 226 insertions(+), 121 deletions(-)
 create mode 100644 labs/lab04/hello.asm
 create mode 100644 labs/lab04/lab5.asm
 delete mode 100644 labs/lab04/report/report.md
 create mode 100644 labs/lab04/report/л04_Ицков_отчет.md
asitskov@vbox:~/work/study/2024-2025/arch-pc/arch-pc$ git push
Перечисление объектов: 12, готово.
Подсчет объектов: 100% (12/12), готово.
Сжатие объектов: 100% (8/8), готово.
Запись объектов: 100% (8/8), 5.14 КиБ | 5.14 МБ/с, готово.
Total 8 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To github.com:a-its/study_2024-2025_arh-pc.git
 8989389..4a6389b master -> master
asitskov@vbox:~/work/study/2024-2025/arch-pc/arch-pc$
```

Рис. 4.14: Загрузка файлов на GitHub

## **5 Выводы**

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

## 6 Список литературы

- [illegible]