

# **Отчёт по лабораторной работе №4**

**Дисциплина: Архитектура компьютера**

Хатамов Эзиз

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>10</b>
4.1	Изучения программы Hello world! . . . . .	10
4.2	Транслятор NASM . . . . .	12
4.3	Расширенный синтаксис командной строки NASM . . . . .	12
4.4	Компоновщик LD . . . . .	13
4.5	Запуск исполняемого файла . . . . .	13
<b>5</b>	<b>Задание для самостоятельной работы</b>	<b>14</b>
<b>6</b>	<b>Выводы</b>	<b>17</b>
	<b>Список литературы</b>	<b>18</b>

# Список иллюстраций

4.1	Создание каталога . . . . .	10
4.2	переход на созданный каталог . . . . .	10
4.3	Создания файла . . . . .	10
4.4	Открывания файла . . . . .	11
4.5	переход на созданный каталог . . . . .	12
4.6	превращения текста в объектный код . . . . .	12
4.7	Скомпилирования файла . . . . .	12
4.8	передача файла на обработку компоновщику . . . . .	13
4.9	передача файла на обработку компоновщику №2 . . . . .	13
4.10	Запуск на выполнения файла . . . . .	13
5.1	Создания копии файла . . . . .	14
5.2	Внесение изменений на файл . . . . .	15
5.3	Транслирования из файла . . . . .	15
5.4	Компановка объектного файла . . . . .	15
5.5	копирования файлов . . . . .	16
5.6	запуск файла . . . . .	16
5.7	Отправка файлов на Github . . . . .	16

## **Список таблиц**

# 1 Цель работы

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

## 2 Задание

1. Изучение программы Hello world!
2. Транслятор NASM
3. Расширенный синтаксис командной строки NASM
4. Компоновщик LD
5. Запуск исполняемого файла
6. Выполнение заданий самостоятельной работы
- 7.

### 3 Теоретическое введение

Основными функциональными элементами любой электронно-вычислительной машины (ЭВМ) являются центральный процессор, память и периферийные устройства (рис.4.1). 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. 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 (половинки 16-битных регистров).

Например, AH (high AX) — старшие 8 бит регистра AX, AL (low AX) — младшие 8 бит регистра AX. Таким образом можно отметить, что вы можете написать в своей программе, например, такие команды (mov – команда пересылки данных на языке ассемблера):

```
mov ax, 1
mov eax, 1
```

Обе команды поместят в регистр AX число 1. Разница будет заключаться только в том, что вторая команда обнулит старшие разряды регистра EAX, то есть после выполнения второй команды в регистре EAX будет число 1. А первая команда оставит в старших разрядах регистра EAX старые данные. И если там были данные, отличные от нуля, то после выполнения первой команды в регистре EAX будет какое-то число, но не 1. А вот в регистре AX будет число 1. 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. В состав ЭВМ также входят периферийные устройства, которые можно разделить на:

- устройства внешней памяти, которые предназначены для долговременного хранения больших объёмов данных (жёсткие диски,



твердотельные накопители, магнитные ленты); • 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. Программа состоит из машинных команд, которые указывают, какие операции и над какими данными (или операндами), в какой последовательности необходимо выполнить. Набор машинных команд определяется устройством конкретного процессора. 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. В самом общем виде 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. Данный алгоритм позволяет выполнить хранящуюся в ОЗУ программу. Кроме того, в зависимости от команды при её выполнении могут проходить не все этапы.

## 4 Выполнение лабораторной работы

### 4.1 Изучения программы Hello world!

Для начало я создал каталог для работы с программами на языке Assambler NASM;

```
ekhatamov@vbox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04/report$ ge
dit report.md
ekhatamov@vbox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04/report$ mk
dir -p ~/work/arch-pc/lab04
ekhatamov@vbox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04/report$
```

Рис. 4.1: Создание каталога

Потом перехожу на созданный каталог с помощью cd

```
ekhatamov@vbox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04/report$ cd
~
ekhatamov@vbox: ~$ cd work
ekhatamov@vbox: ~/work$ cd arch-pc
ekhatamov@vbox: ~/work/arch-pc$ cd lab04
ekhatamov@vbox: ~/work/arch-pc/lab04$
```

Рис. 4.2: переход на созданный каталог

создал текстрвый файл с именем hello.asm с помощью touch

```
ekhatamov@vbox: ~$ cd work
ekhatamov@vbox: ~/work$ cd arch-pc
ekhatamov@vbox: ~/work/arch-pc$ cd lab04
ekhatamov@vbox: ~/work/arch-pc/lab04$ touch hello.asm
ekhatamov@vbox: ~/work/arch-pc/lab04$
```

Рис. 4.3: Создания файла

Открыл файл с помощью gedit

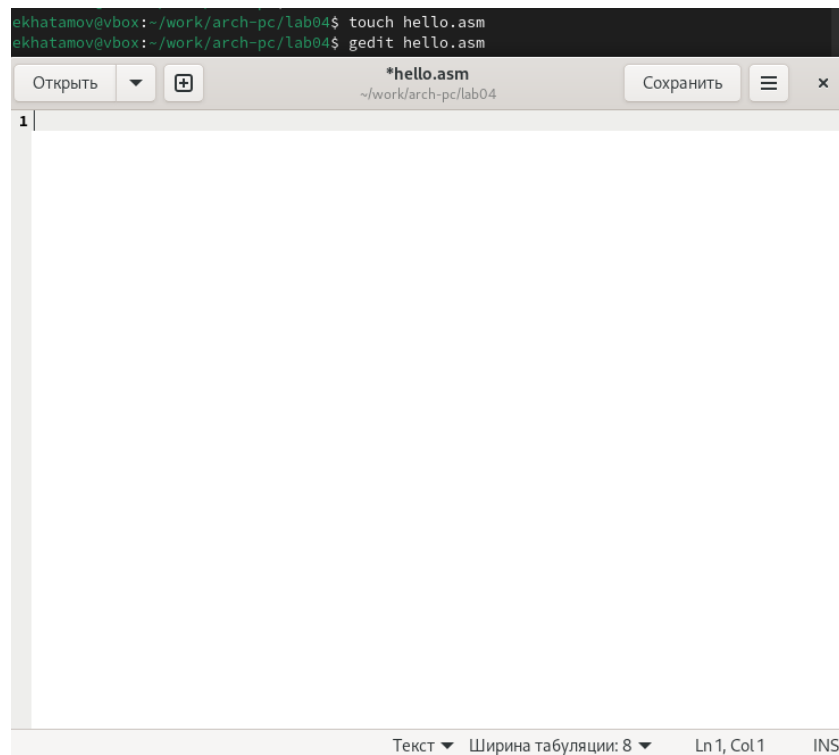


Рис. 4.4: Открывания файла

Потом ввел туда нужные команды

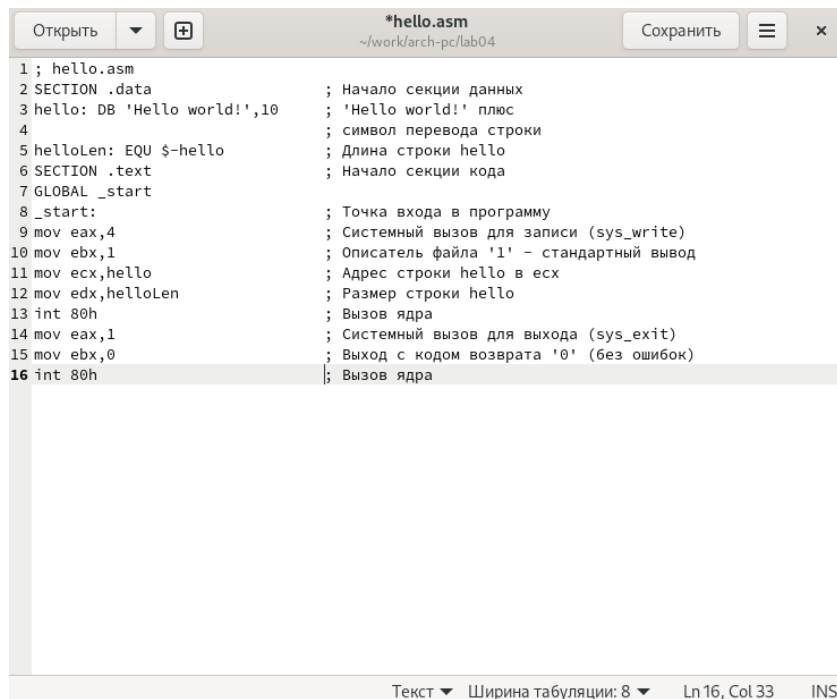


Рис. 4.5: переход на созданный каталог

## 4.2 Транслятор NASM

С помощью кода `nasm -f elf` превращаю свой текст в объектный код

```

ekhatamov@vbox:~/work/arch-pc/lab04$ nasm -f elf hello.asm
ekhatamov@vbox:~/work/arch-pc/lab04$ ls
hello.asm  hello.o

```

Рис. 4.6: превращения текста в объектный код

## 4.3 Расширенный синтаксис командной строки NASM

С помощью следующей командой я скомпилирую исходный файл `hello.asm` в `obj.o`

```

ekhatamov@vbox:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
ekhatamov@vbox:~/work/arch-pc/lab04$ ls
hello.asm  hello.o  list.lst  obj.o
ekhatamov@vbox:~/work/arch-pc/lab04$

```

Рис. 4.7: Скомпилирования файла

## 4.4 Компоновщик LD

С помощью команды `ld -m elf_i386 hello.o -o hello` я передаю файл на обработку компоновщику

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

Рис. 4.8: передача файла на обработку компоновщику

Потом я ввел следующую команду

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

Рис. 4.9: передача файла на обработку компоновщику №2

## 4.5 Запуск исполняемого файла

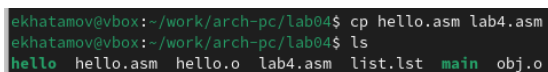
Восле этих действ я запускаю на выполнения созданный исполняемый файл

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

Рис. 4.10: Запуск на выполнения файла

## 5 Задание для самостоятельной работы

1. В новом созданном каталоге с помощью `cp` создал копию файла с именем `lab4.asm`



```
ekhatamov@vbox:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
ekhatamov@vbox:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
```

Рис. 5.1: Создания копии файла

2. В текстовом редакторе внес вместо “Hello world” своё имя и фамилию на файле `lab4`.

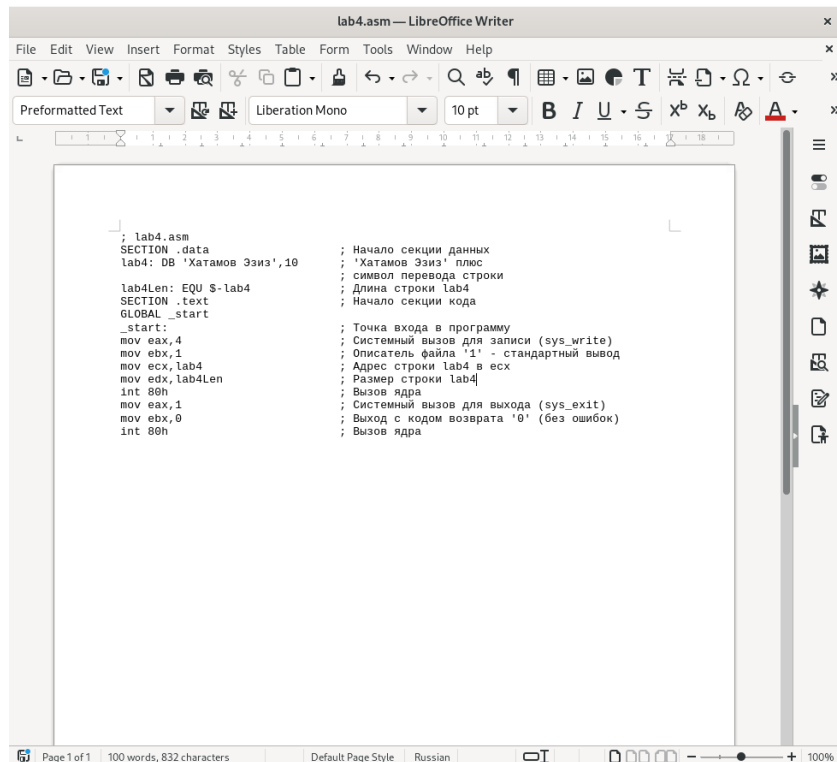


Рис. 5.2: Внесение изменений на файл

3. Транслировал текст файла lab4.asm в объектный файл.Выполнил компоновку объектного файла и запустил получившийся исполняемый файл

```

ekhatamov@vbox:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
ekhatamov@vbox:~/work/arch-pc/lab04$ ls
hello.asm hello.o lab4.asm lab4.o list.lst main obj.o
ekhatamov@vbox:~/work/arch-pc/lab04$

```

Рис. 5.3: Транслирования из файла

```

ekhatamov@vbox:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o hello
ekhatamov@vbox:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
ekhatamov@vbox:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4.asm lab4.o list.lst main obj.o
ekhatamov@vbox:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
ekhatamov@vbox:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o
ekhatamov@vbox:~/work/arch-pc/lab04$

```

Рис. 5.4: Компановка объектного файла

4. Потом я скопирую файлы hello.asm и lab4.asm в мой локальный репозиторий в главный каталог

```
ekhatamov@vbox: ~/work/arch-pc/lab04$ cp hello.asm lab4.asm ~/work/study/2024-2025/"Архитектура компьютера"
ekhatamov@vbox: ~/work/arch-pc/lab04$ cd ~
ekhatamov@vbox: ~$ cd ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab04
ekhatamov@vbox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello.asm lab4.asm presentation report
```

Рис. 5.5: копирования файлов

После копирования я запустил на выполнение созданный исполняемый файл чтобы убедиться все ли работает

```
ekhatamov@vbox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ ./lab4
Хатамов Эзиз
```

Рис. 5.6: запуск файла

Все готова теперь спокойно смогу все вложить в Github

```
ekhatamov@vbox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git add
ekhatamov@vbox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git commit -m "Added files for Lab4"
[master 4281212] Added files for Lab4
39 files changed, 202 insertions(+), 32 deletions(-)
create mode 100644 lab4/lab02/report/figs/Common_report_01_2024-10-20_03-02-10.png
create mode 100644 lab4/lab02/report/figs/Common_report_02_2024-10-20_03-05-10.png
create mode 100644 lab4/lab02/report/figs/Common_report_03_2024-10-20_03-13-22.png
create mode 100644 lab4/lab02/report/002_Karamea_greet.docx
create mode 100644 lab4/lab02/report/002_Karamea_greet.md
create mode 100644 lab4/lab02/report/002_Karamea_greet_des Markdown.pdf
create mode 100755 lab4/lab04/lab4
create mode 100644 lab4/lab04/lab4.asm
create mode 100755 lab4/lab04/lab4
create mode 100644 lab4/lab04/lab4.s
create mode 100644 lab4/lab04/lab4.s.txt
create mode 100755 lab4/lab04/lab4
create mode 100644 lab4/lab04/lab4.s
create mode 100644 lab4/lab04/report/figs/1.png
create mode 100644 lab4/lab04/report/figs/10.png
create mode 100644 lab4/lab04/report/figs/11.png
create mode 100644 lab4/lab04/report/figs/12.png
create mode 100644 lab4/lab04/report/figs/13.png
create mode 100644 lab4/lab04/report/figs/14.png
create mode 100644 lab4/lab04/report/figs/15.png
create mode 100644 lab4/lab04/report/figs/16.png
create mode 100644 lab4/lab04/report/figs/17.png
create mode 100644 lab4/lab04/report/figs/18.png
create mode 100644 lab4/lab04/report/figs/19.png
create mode 100644 lab4/lab04/report/figs/20.png
create mode 100644 lab4/lab04/report/figs/21.png
create mode 100644 lab4/lab04/report/figs/22.png
create mode 100644 lab4/lab04/report/figs/23.png
create mode 100644 lab4/lab04/report/figs/24.png
create mode 100644 lab4/lab04/report/figs/25.png
create mode 100644 lab4/lab04/report/figs/26.png
create mode 100644 lab4/lab04/report/figs/27.png
create mode 100644 lab4/lab04/report/figs/28.png
create mode 100644 lab4/lab04/report/figs/29.png
create mode 100644 lab4/lab04/report/figs/30.png
create mode 100644 lab4/lab04/report/figs/31.png
create mode 100644 lab4/lab04/report/figs/32.png
create mode 100644 lab4/lab04/report/report.docx
create mode 100644 lab4/lab04/report/report.pdf
delete mode 100644 prepare
Внесено объектов: 52, готово.
Всего объектов: 1006 (52/954) готово.
При сканировании использовано 20 8 объектов
Всего объектов: 1006 (52/954) готово.
Время объектов: 100% (42/42) - 3.49 МБ | 122.00 MB/s, готово.
data: 43 (40/17) - 100% (40/40) - 0 - 0.00 - 0.00 (0) (0.00)
remote: Resolving deltas: 100% (17/17), completed with 3 local objects.
```

Рис. 5.7: Отправка файлов на Github



## **6 Выводы**

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

# Список литературы

1.Архитектура ЭВМ - РУДН ::: :::