

# **Отчет по лабораторной работе №4**

**Архитектура компьютеров**

Виктор Вацаев Андреевич

# 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>8</b>

## **Список иллюстраций**

## **Список таблиц**

# 1 Цель работы

Цель данной лабораторной работы — изучение основ программирования на языке ассемблера, а также получение навыков работы с компиляторами и линковщиками для создания исполняемых файлов.

## **2 Задание**

В данной лабораторной работе было предложено написать программу на языке ассемблера, которая выводит строку с личными данными на экран.

### 3 Теоретическое введение

Ассемблер — это низкоуровневый язык программирования, который предоставляет программисту возможность напрямую управлять аппаратным обеспечением компьютера. В отличие от языков высокого уровня, ассемблерные программы требуют большего внимания к деталям, так как они ближе к машинному коду.

Основные этапы получения исполняемого файла: 1. **Трансляция**: преобразование исходного кода в объектный файл. 2. **Компоновка**: объединение объектных файлов в исполняемый файл.

## 4 Выполнение лабораторной работы

В 1 пункте были выполнены следующие действия: `-f elf32` — указывает формат выходного файла (32-битный ELF). `lab4.asm` — ваш исходный файл программы. `-o`

`lab4.o` — имя выходного объектного файла.

```
victor@fedora:~$ cd /home/victor/work  
victor@fedora:~/work/arch-pc/lab04$
```

Пункт 2: `-m elf_i386` — указывает линковщику использовать 32-битную архитектуру. `-s` — удаляет символическую таблицу из конечного исполняемого файла (опционально, для уменьшения размера файла). `-o lab4` — задаёт имя выходного ис-



```

Установить пакет «nasm», предоставляющий команду «nasm»? [N/y] y

* Ожидание в очереди...
* Загрузка списка пакетов....
Следующие пакеты должны быть установлены:
nasm-2.16.01-7.fc40.x86_64      A portable x86 assembler which uses Intel-like sy
Продолжить с этими изменениями? [N/y] y

* Ожидание в очереди...
* Ожидание аутентификации...
* Ожидание в очереди...
* Загрузка пакетов...
* Запрос данных...
* Проверка изменений...
* Установка пакетов...

victor@fedora:~/work/arch-pc/lab04$ nasm -f elf hello.asm
victor@fedora:~/work/arch-pc/lab04$ gedit hello.asm
victor@fedora:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
victor@fedora:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
victor@fedora:~/work/arch-pc/lab04$ ls
hello.asm  hello.o  list.lst  obj.o
victor@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
victor@fedora:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o
victor@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
victor@fedora:~/work/arch-pc/lab04$ ./hello
Hello world!

```

полняемого файла.

Пункт 3 Запуск исполняемого файла: ./lab4

```

victor@fedora:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
victor@fedora:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
victor@fedora:~/work/arch-pc/lab04$ nasm -f elf32 lab4.asm -o lab4.o
victor@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 -s -o lab4 lab4.o
victor@fedora:~/work/arch-pc/lab04$ ./lab4
Мое имя Виктор, моя фамилия Вацаев!
victor@fedora:~/work/arch-pc/lab04$

```

```

victor@fedora:~$ cp /home/victor/work/arch-pc/lab04/hello.asm ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-
victor@fedora:~$ cp /home/victor/work/arch-pc/lab04/ab4.asm ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-
cp: не удалось выполнить stat для '/home/victor/work/arch-pc/lab04/ab4.asm':
victor@fedora:~$ cp /home/victor/work/arch-pc/lab04/lab4.asm ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-
victor@fedora:~$ cd /work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-
bash: cd: /work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-: не такой путь
victor@fedora:~$ cd /home/victor/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-
victor@fedora:~$ cd /home/victor/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-
Текущая ветка: master
Эта ветка соответствует «origin/master».

Изменения, которые не в индексе для коммита:
(используйте «git add <файл>...», чтобы добавить файл в индекс)
(используйте «git restore <файл>...», чтобы отменить изменения в рабочем к
изменено:    ../lab02/report/report.md

Неотслеживаемые файлы:
(используйте «git add <файл>...», чтобы добавить в то, что будет включено
../lab02/report/report.docx
../lab02/report/reportlab2.pdf
hello.asm
lab4.asm

индекс пуст (используйте «git add» и/или «git commit -a»)
victor@fedora:~$ git add .
victor@fedora:~$ git commit -m "Добавлены файлы hello.asm и lab4.asm"
[master b4c132b] Добавлены файлы hello.asm и lab4.asm
2 files changed, 32 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
victor@fedora:~$ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 12 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 1.03 КиБ | 1.03 МБ/с, готово.
Total 6 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:victorzeffirovski/study_2023-2024_arh--pc.git
7eb3077..b4c132b master -> master
victor@fedora:~$

```

Также все было скопировано и загружено на GitHub

## Выводы

В ходе выполнения лабораторной работы были достигнуты поставленные цели. Успешно написаны, протестированы и скомпилированы программы на языке ассемблера. Полученные результаты подтвердили понимание основ работы с ассемблером и компиляторами. Вопросы для самопроверки

Какие основные отличия ассемблерных программ от программ на языках высокого уровня?

Ассемблерные программы ближе к машинному коду и требуют большей детализации в написании, в то время как языки высокого уровня предоставляют абстракцию и удобство написания кода.

В чём состоит отличие инструкции от директивы на языке ассемблера?

Инструкция выполняет определённые операции и обрабатывается процессором, тогда как директива управляет процессом ассемблирования и не выполняется процессором.

Перечислите основные правила оформления программ на языке ассемблера.

Правильное использование отступов и комментариев, чёткая структура секций данных и кода, а также соблюдение синтаксиса языка ассемблера.

Каковы этапы получения исполняемого файла?

Трансляция исходного кода в объектный файл, компоновка объектных файлов в исполняемый файл.

Каково назначение этапа трансляции?

Преобразование исходного кода в объектный код, который может быть использован для создания исполняемого файла.

Каково назначение этапа компоновки?

Объединение одного или нескольких объектных файлов в единый и исполняемый файл и разрешение всех ссылок между ними.

Какие файлы могут создаваться при трансляции программы, какие из них создаются по умолчанию?

При трансляции могут создаваться объектные файлы (.o, .obj) и файлы с отладочной информацией; по умолчанию создаются объектные файлы.

Каковы форматы файлов для `nasm` и `ld`?

Формат файлов для `nasm` может быть `elf32`, `elf64`, `bin`, и другие; для `ld` — это обычно `elf`, но может включать другие форматы в зависимости от целевой платформы.