Отчет по лабораторной работе №4

Архитектура компьютеров

Виктор Ващаев Андреевич

Содержание:

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.

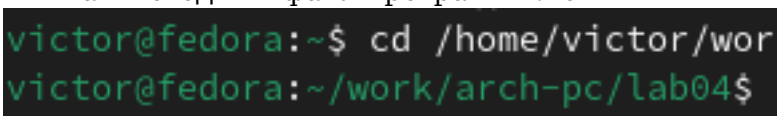
3 Теоретическое введение

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.



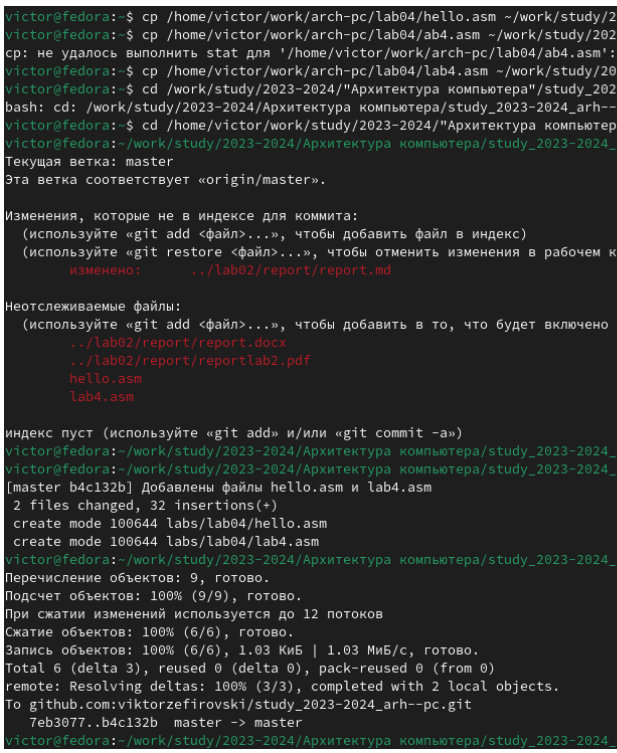
Пункт 2: -m elf_i386 — указывает линковщику использовать 32-битную архитектуру. -s — удаляет символическую таблицу из конечного исполняемого файла (опционально, для уменьшения размера файла). -o lab4 — задаёт имя выходного ис-

```
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 исполняемго 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 Ващаев!
victor@fedora:~/work/arch-pc/lab04$
```



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 сполняемый 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.