Лабораторная работа №4

НКАбд-06-23

Улитина Мария Максимовна

Содержание

1 Цель работы

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

2 Задание

1. Написание программы “Hello world!”;
2. Работа с транслятором NASM;
3. Работа с расширенным синтаксисом командной строки NASM;

4. Работа с компоновщиком LD;
5. Создание и запуск lab4.asm

3 Теоретическое введение

Основными функциональными элелементами любой электронно-вычислительной машины (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 (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 арифметиче- ские 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, пре- образование (арифметические или логические операции) данных хранящихся в регистрах.

3.1 Ассемблер и язык ассемблера

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. Можно считать, что он больше любых других языков приближен к архитектуре ЭВМ и её аппаратным возможностям, что позволяет получить к ним более полный доступ, нежели в языках высокого уровня, таких как C/C++, Perl, Python и пр. Заметим, что получить полный доступ к ресурсам компьютера в современных архитектурах нельзя, самым низким уровнем работы прикладной программы является обращение напрямую к ядру операционной системы. Именно на этом уровне и работают программы, написанные на ассемблере. Но в отличие от языков высокого уровня ассемблерная программа содержит только тот код, который ввёл программист. Таким образом язык ассемблера — это язык, с

помощью которого понятным для человека образом пишутся команды для процессора.

3.2 Процесс создания и обработки программы на языке ассемблера

В процессе создания ассемблерной программы можно выделить четыре шага:

• Набор текста программы в текстовом редакторе и сохранение её в отдельном файле. Каждый файл имеет свой тип (или расширение), который определяет назначение файла. Файлы с исходным текстом программ на языке ассемблера имеют тип asm.

• Трансляция — преобразование с помощью транслятора, например nasm, текста про- граммы в машинный код, называемый объектным. На данном этапе также может быть получен листинг программы, содержащий кроме текста программы различную допол- нительную информацию, созданную транслятором. Тип объектного файла — o, файла листинга — lst.

• Компоновка или линковка — этап обработки объектного кода компоновщиком (ld), который принимает на вход объектные файлы и собирает по ним исполняемый файл. Исполняемый файл обычно не имеет расширения. Кроме того, можно получить файл карты загрузки программы в ОЗУ, имеющий расширение map.

• Запуск программы. Конечной целью является работоспособный исполняемый файл. Ошибки на предыдущих этапах могут привести к некорректной работе программы, поэтому может присутствовать этап отладки программы при помощи специальной программы — отладчика. При нахождении ошибки необходимо провести коррекцию программы, начиная с первого шага.

4 Выполнение лабораторной работы

4.1 Программа Hello world!

Создадим каталог для работы с программами на языке ассемблера NASM и перейдем в созданный каталог(рис.1).

```
mmulitina@ubuntu:~$ mkdir -p ~/work/arch-pc/lab04
mmulitina@ubuntu:~$ cd ~/work/arch-pc/lab04
mmulitina@ubuntu:~/work/arch-pc/lab04$
```

Создадим текстовый файл с именем hello.asm, откроем его с помощью gedit, введём в него предоставленный в лабораторной работе текст(рис.2).

```
mmulitina@ubuntu:~/work/arch-pc/lab04$ touch hello.asm
mmulitina@ubuntu:~/work/arch-pc/lab04$ gedit hello.asm
```

4.2 Транслятор NASM

Для компиляции текста из файла программы создадим объектный файл(рис.3).

```
mmulitina@ubuntu:~/work/arch-pc/lab04$ nasm -f elf hello.asm
```

и проверим его наличие с помощью ls (рис.4).

```
mmulitina@ubuntu:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
```

4.3 Расширенный синтаксис командной строки NASM

Скомпилируем исходный файл hello.asm в obj.o (рис.5.)

```
mmulitina@ubuntu:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello
.asm
```

С помощью ls проверим, что файлы были созданы (рис.6).

```
mmulitina@ubuntu:~/work/arch-pc/lab04$ ls
hello.asm  hello.o  list.lst  obj.o
```

4.4 Компоновщик LD

Передадим объектный файл на обработку компоновщику (рис.7).

```
mmulitina@ubuntu:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
```

С помощью ls проверим, что исполняемый файл hello был создан (рис.8).

```
mmulitina@ubuntu:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o
```

Зададим с помощью ключа -o имя создаваемого исполняемого файла (рис.9).

```
mmulitina@ubuntu:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
mmulitina@ubuntu:~/work/arch-pc/lab04$ ./hello
Hello world!
```

4.5 Запуск исполняемого файла

Запустим на выполнение созданный исполняемый файл (рис.9).

```
mmulitina@ubuntu:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
mmulitina@ubuntu:~/work/arch-pc/lab04$ ./hello
Hello world!
```

4.6 Выполнение заданий для самостоятельной работы

Создадим с помощью команды cp создадим копию файла hello.asm с именем lab4.asm (рис. 10).

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

Проверим с помощью ls успешность создания копии. С помощью текстового редактора внесём изменения к текст программы, чтобы на экран выводилась строка с фамилией и именем. Оттранслируем полученный текст программы lab4.asm в объектный файл. Выполним компоновку объектного файла и запустим получившийся исполняемый файл(рис.12).

```
mmulitina@ubuntu:~/work/arch-pc/lab04$ gedit lab4.asm
mmulitina@ubuntu:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
mmulitina@ubuntu:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst lab4.
asm
mmulitina@ubuntu:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
mmulitina@ubuntu:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
mmulitina@ubuntu:~/work/arch-pc/lab04$ ./lab4
Maria Ulitina
```

5 Выводы

В процессе выполнения лабораторной работы я освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM.

6 Список литературы.

Архитектура ЭВМ. Лабораторная работа №4.