

Отчёт по лабораторной работе 4

Создание и процесс обработки программ на языке ассемблера NASM

Окафор Чуквуемезуго Келвин

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	8
3.1	Программа Hello world!	8
3.2	Транслятор NASM	9
3.3	Расширенный синтаксис командной строки NASM	10
3.4	Компоновщик LD	10
3.5	Запуск исполняемого файла	11
3.6	Задание для самостоятельной работы	11
4	Выводы	13

Список иллюстраций

3.1	Создан каталог для работы и файл для программы	8
3.2	Программа в файле hello.asm	9
3.3	Трансляция программы	9
3.4	Трансляция программы с дополнительными опциями	10
3.5	Компоновка программы	10
3.6	Компоновка программы	11
3.7	Запуск программы	11
3.8	Скопировал файл	11
3.9	Программа в файле lab4.asm	12
3.10	Проверка программы lab4.asm	12

Список таблиц

1 Цель работы

Целью работы является освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

2 Теоретическое введение

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.

В процессе создания ассемблерной программы можно выделить четыре шага:

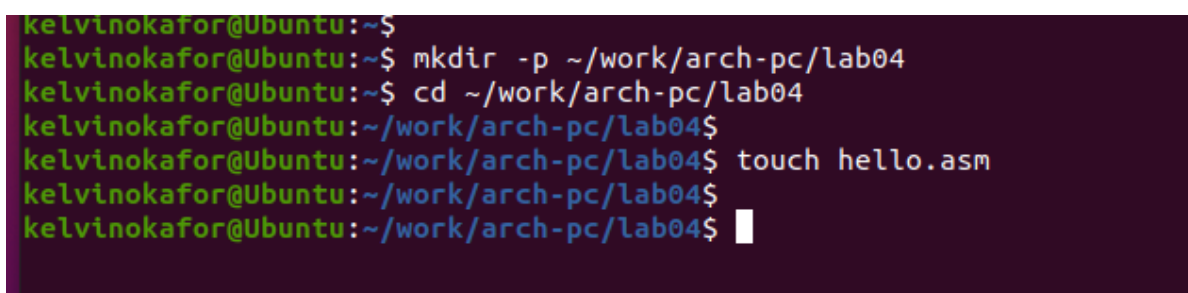
- Набор текста программы в текстовом редакторе и сохранение её в отдельном файле. Каждый файл имеет свой тип (или расширение), который определяет назначение файла. Файлы с исходным текстом программ на языке ассемблера имеют тип asm.
- Трансляция — преобразование с помощью транслятора, например nasm, текста программы в машинный код, называемый объектным. На данном этапе также может быть получен листинг программы, содержащий кроме текста программы различную дополнительную 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.

3 Выполнение лабораторной работы

3.1 Программа Hello world!

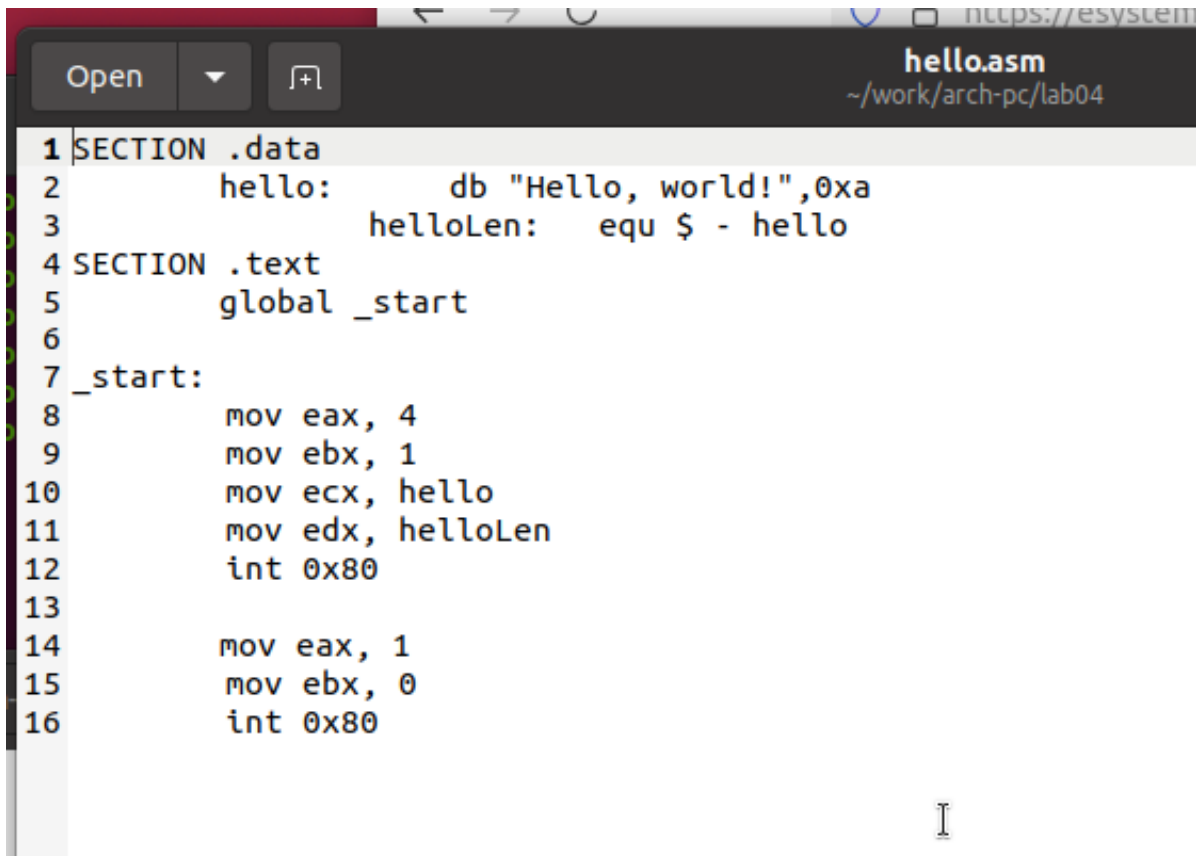
Создал каталог lab04 командой `mkdir`, перешел в него с помощью команды `cd` и создал файл `hello.asm`, в который напишу программу. Убеждаюсь с помощью команды `ls`, что создал файл.



```
kelvinokafor@Ubuntu:~$  
kelvinokafor@Ubuntu:~$ mkdir -p ~/work/arch-pc/lab04  
kelvinokafor@Ubuntu:~$ cd ~/work/arch-pc/lab04  
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$  
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$ touch hello.asm  
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$  
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$
```

Рис. 3.1: Создан каталог для работы и файл для программы

Написал программу по заданию на языке ассемблера.



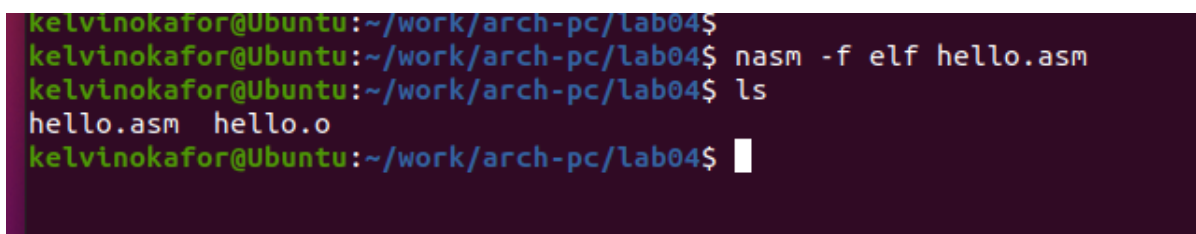
```
1 SECTION .data
2     hello:      db "Hello, world!",0xa
3     helloLen:   equ $ - hello
4 SECTION .text
5     global _start
6
7 _start:
8     mov eax, 4
9     mov ebx, 1
10    mov ecx, hello
11    mov edx, helloLen
12    int 0x80
13
14    mov eax, 1
15    mov ebx, 0
16    int 0x80
```

Рис. 3.2: Программа в файле hello.asm

3.2 Транслятор NASM

NASM превращает текст программы в объектный код. Если текст программы набран без ошибок, то транслятор преобразует текст программы из файла hello.asm в объектный код, который запишется в файл hello.o.

Транслировал файл командой `nasm`. Получился объектный файл hello.o.



```
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$ nasm -f elf hello.asm
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$
```

Рис. 3.3: Трансляция программы

3.3 Расширенный синтаксис командной строки NASM

Полный вариант командной строки `nasm` выглядит следующим образом:

```
nasm [-@ косвенный_файл_настроек] [-o объектный_файл] [-f формат_объектного_файла]
[-l листинг] [параметры...] [--] исходный_файл
```

Транслировал файл командой `nasm` с дополнительными опциями. С опцией `-l` Получил файл листинга `list.lst`, с опцией `-f` объектный файл `obj.o`, с опцией `-g` в программу добавилась отладочная информация.

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

Рис. 3.4: Трансляция программы с дополнительными опциями

3.4 Компоновщик LD

Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику.

Выполнил команду `ld` и получил исполняемый файл `hello` из объектного файла `hello.o`.

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

Рис. 3.5: Компоновка программы

Еще раз выполнил команду `ld` для объектного файла `obj.o` и получил исполняемый файл `main`.

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

Рис. 3.6: Компоновка программы

3.5 Запуск исполняемого файла

Запустил исполняемые файлы.

```
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$ ./hello
Hello, world!
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$ ./main
Hello, world!
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$
```

Рис. 3.7: Запуск программы

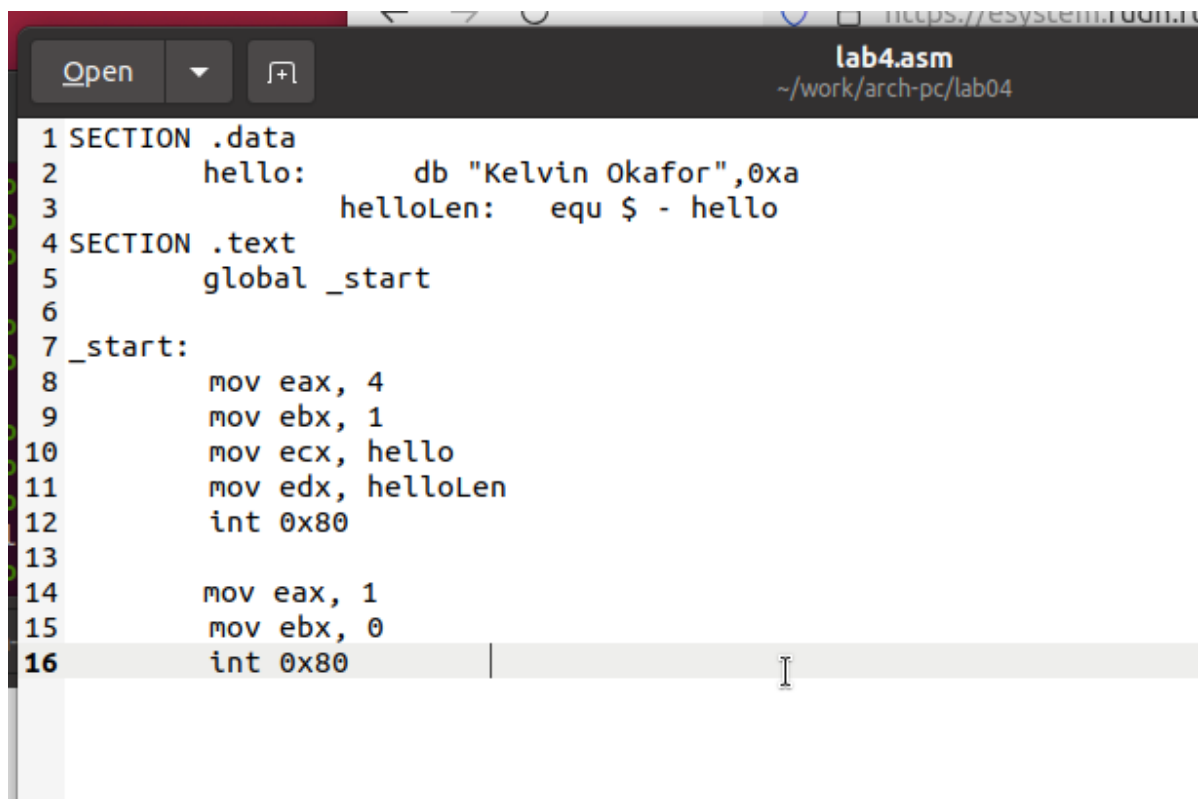
3.6 Задание для самостоятельной работы

Скопировал файл hello.asm в файл lab4.asm.

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

Рис. 3.8: Скопировал файл

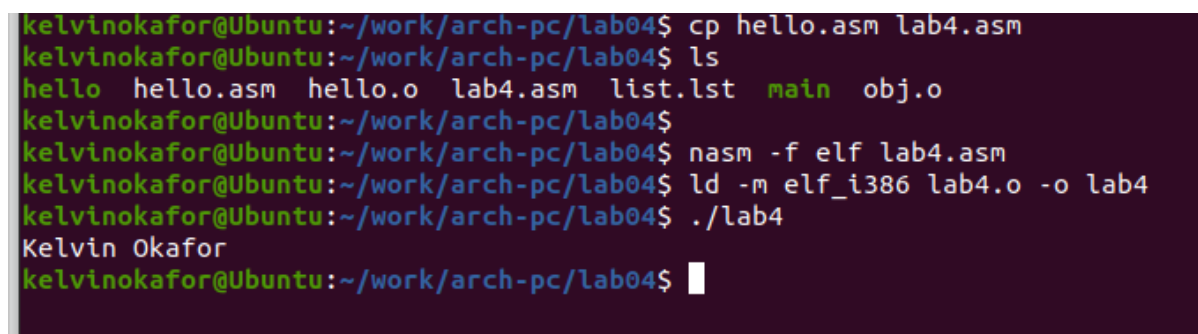
Изменил сообщение Hello world на свое имя.



```
1 SECTION .data
2     hello:      db "Kelvin Okafor",0xa
3     helloLen:   equ $ - hello
4 SECTION .text
5     global _start
6
7 _start:
8     mov eax, 4
9     mov ebx, 1
10    mov ecx, hello
11    mov edx, helloLen
12    int 0x80
13
14    mov eax, 1
15    mov ebx, 0
16    int 0x80
```

Рис. 3.9: Программа в файле lab4.asm

Запустил программу и проверил.



```
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4.asm list.lst main obj.o
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$ ./lab4
Kelvin Okafor
kelvinokafor@Ubuntu:~/work/arch-pc/lab04$
```

Рис. 3.10: Проверка программы lab4.asm

4 Выводы

Освоил процесс компиляции и сборки программ, написанных на ассемблере `nasm`.