

Отчёт по лабораторной работе 4

Архитектура компьютеров

Амир Расули

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	Компоновщик LD	10
3.4	Выполнение заданий для самостоятельной работы.	11
4	Выводы	13
	Список литературы	14

Список иллюстраций

3.1	Создание каталога и файла	8
3.2	Программа hello.asm	9
3.3	Трансляция hello.asm	9
3.4	Трансляция hello.asm с дополнительными опциями	10
3.5	Линковка программы	10
3.6	Линковка программы	10
3.7	Запуск программ	11
3.8	Код программы в файле lab4.asm	12
3.9	Запуск 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

NASM (англ. Netwide Assembler) – это 80x86 ассемблер, который был разработан с упором на переносимость и модульность. Он поддерживает множество форматов объектных файлов, таких как форматы Linux a.out и ELF, NetBSD/FreeBSD, COFF, Microsoft 16-bit OBJ и Win32. Помимо этого, NASM также может генерировать простые бинарные файлы. Синтаксис NASM схож с Intel-синтаксисом, хотя немного сложнее. Он поддерживает инструкции для процессоров Pentium, P6 и MMX, а также включает макро-расширения.

NASM был создан Саймоном Тэтхемом совместно с Юлианом Холлом и в настоящее время развивается командой разработчиков на SourceForge.net. Изначально он был выпущен под собственной лицензией, но после множества проблем с выбором лицензии она была заменена на GNU LGPL. Начиная с версии 2.07, лицензия заменена на “упрощённую BSD” (BSD из 2 пунктов).

NASM может работать на различных платформах, таких как SPARC и PowerPC, однако код генерируется исключительно для x86 и x86-64 архитектур.

NASM является конкурентом стандартному для Linux и многих UNIX-систем ассемблеру gas. Примечательно, что документация NASM считается более качественной, чем у gas. Кроме того, gas использует синтаксис AT&T, предназначенный для процессоров не от Intel, в то время как NASM придерживается Intel-синтаксиса, который традиционно используется в x86-ассемблерах, таких как MASM, TASM и fasm.

NASM использует Intel-синтаксис записи инструкций. Строка программы NASM может включать следующие элементы: метку, инструкцию, операнды и

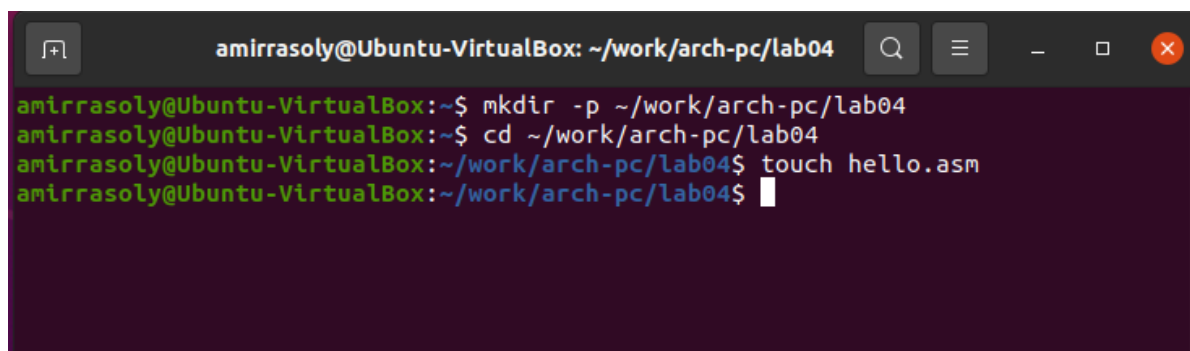
комментарий.

Операнды отделяются запятыми. Пробелы допустимы перед строкой и после инструкции. Комментарий начинается с точки с запятой, и его конец совпадает с концом строки. Если строка программы слишком длинная, её можно перенести на следующую с помощью обратного слэша (`\`), аналогично языку C.

3 Выполнение лабораторной работы

3.1 Программа Hello world!

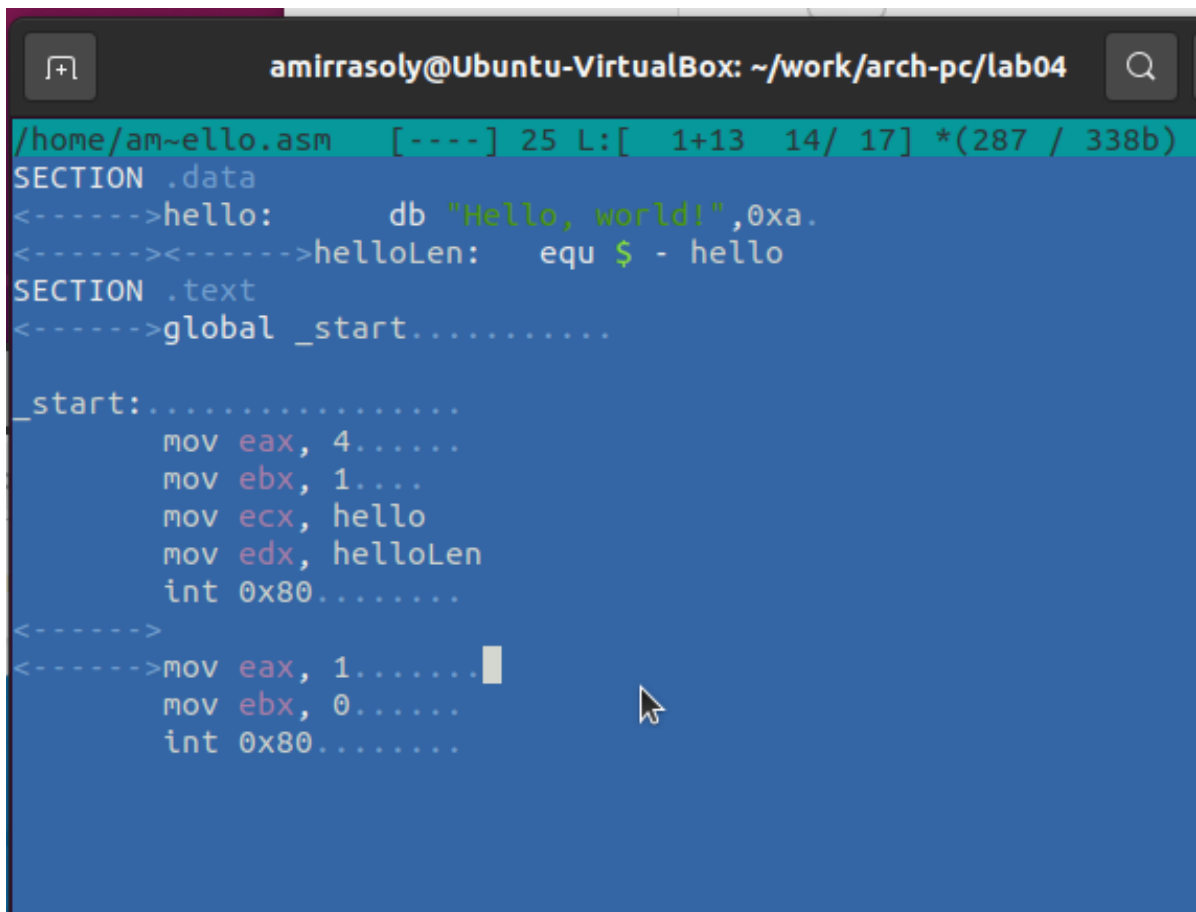
Создаю каталог `lab04` с помощью команды `mkdir`, перехожу в него с помощью `cd`, и создаю файл `hello.asm`. (рис. 3.1)



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

Рис. 3.1: Создание каталога и файла

Открываю файл и пишу код программы по заданию. (рис. 3.2)



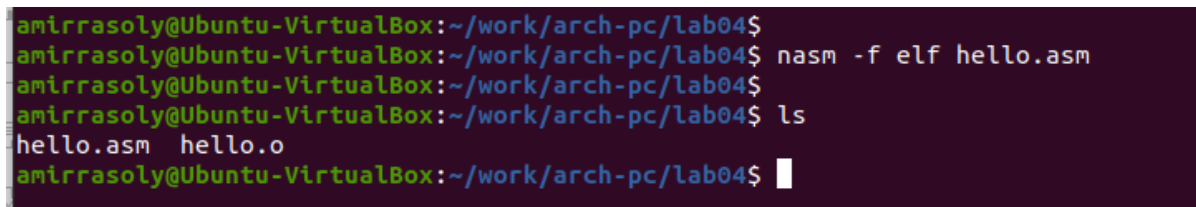
```
amirrasoly@Ubuntu-VirtualBox: ~/work/arch-pc/lab04
/home/amirrasoly/hello.asm [----] 25 L: [ 1+13 14/ 17] *(287 / 338b)
SECTION .data
<----->hello:      db "Hello, world!",0xa.
<-----><----->helloLen: equ $ - hello
SECTION .text
<----->global _start.....

_start:.....
    mov eax, 4.....
    mov ebx, 1....
    mov ecx, hello
    mov edx, helloLen
    int 0x80.....
<----->
<----->mov eax, 1.....
    mov ebx, 0.....
    int 0x80.....
```

Рис. 3.2: Программа hello.asm

3.2 Транслятор NASM

Транслирую файл командой `nasm`, что позволяет получить объектный файл `hello.o`. (рис. 3.3)



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

Рис. 3.3: Трансляция hello.asm

Использую команду `nasm` с дополнительными опциями для создания

файла листинга `list.lst`, объектного файла `obj.o`, и добавляю отладочную информацию в программу. (рис. 3.4)

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

Рис. 3.4: Трансляция `hello.asm` с дополнительными опциями

3.3 Компоновщик LD

Выполняю линковку с помощью команды `ld` и получаю исполняемый файл. (рис. 3.5)

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

Рис. 3.5: Линковка программы

Повторяю линковку для объектного файла `obj.o` и получаю исполняемый файл `main`. (рис. 3.6)

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

Рис. 3.6: Линковка программы

Запускаю полученные исполняемые файлы. (рис. 3.7)

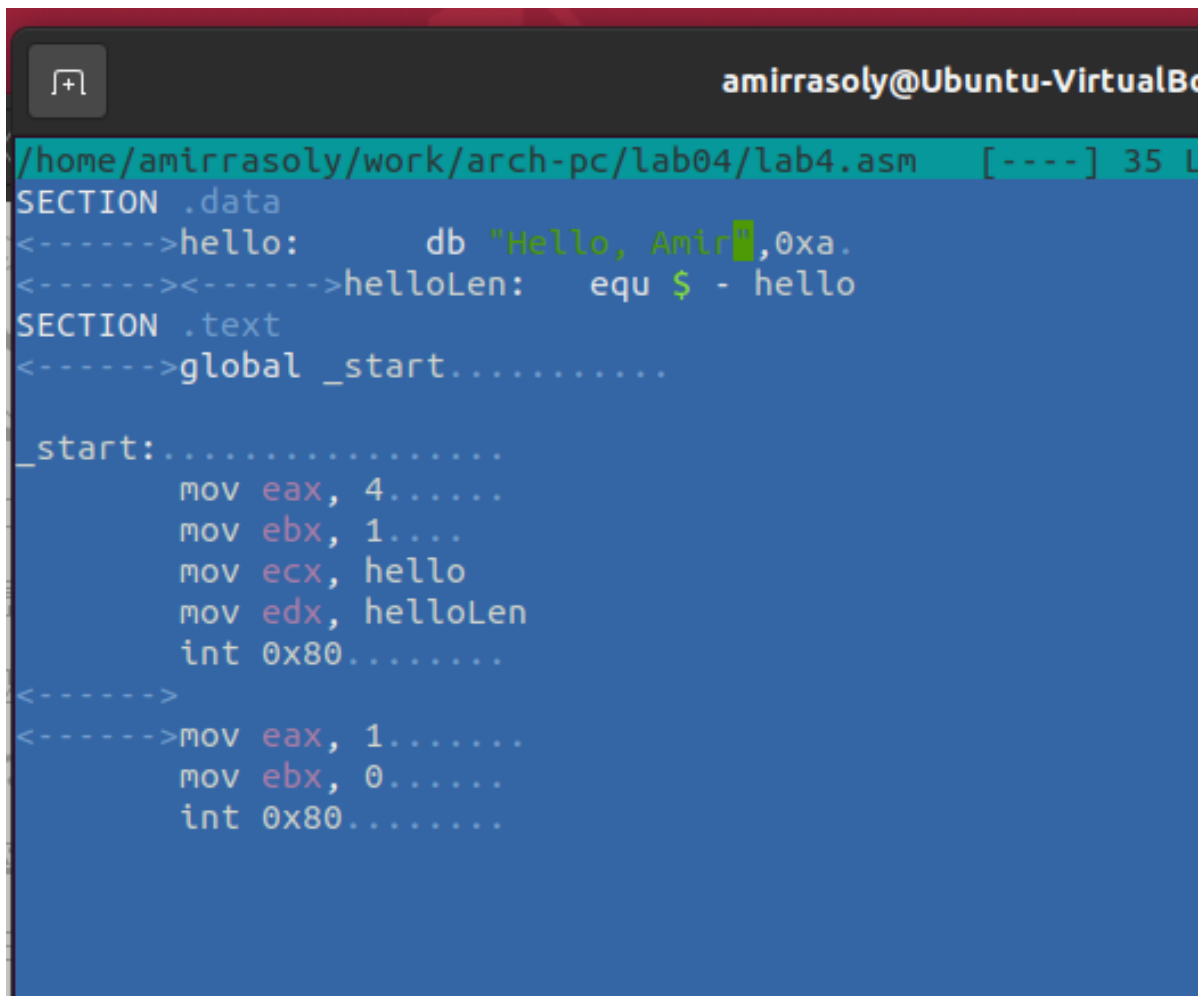
```
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab04$  
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab04$ ./hello  
Hello, world!  
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab04$ ./main  
Hello, world!  
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab04$
```

Рис. 3.7: Запуск программ

3.4 Выполнение заданий для самостоятельной работы.

Копирую программу в новый файл.

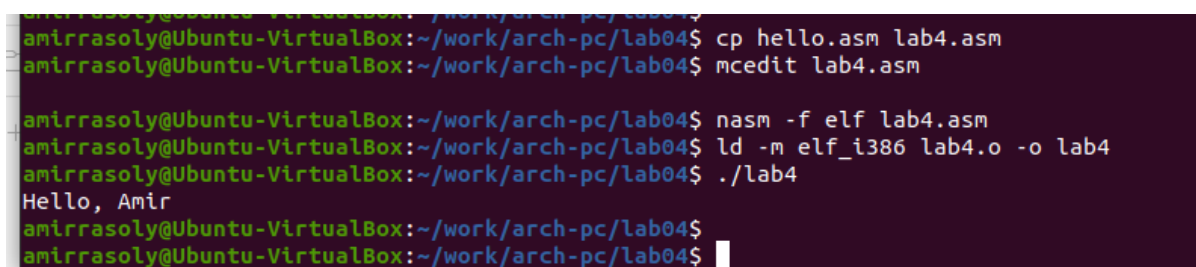
Изменяю сообщение “Hello world” на своё имя (рис. 3.8) и запускаю новую программу. (рис. 3.9)



```
amirrasoly@Ubuntu-VirtualBox
/home/amirrasoly/work/arch-pc/lab04/lab4.asm [----] 35 L
SECTION .data
<----->hello:      db "Hello, Amir",0xa.
<-----><----->helloLen:  equ $ - hello
SECTION .text
<----->global _start.....

_start:.....
    mov eax, 4.....
    mov ebx, 1....
    mov ecx, hello
    mov edx, helloLen
    int 0x80.....
<----->
<----->mov eax, 1.....
    mov ebx, 0.....
    int 0x80.....
```

Рис. 3.8: Код программы в файле lab4.asm



```
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab04$ mcedit lab4.asm

amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab04$ ./lab4
Hello, Amir
amirrasoly@Ubuntu-VirtualBox:~/work/arch-pc/lab04$
```

Рис. 3.9: Запуск программы lab4.asm

4 Выводы

В ходе выполнения лабораторной работы я освоил процесс компиляции и сборки программ на ассемблере NASM. Полученные навыки включают создание объектных файлов, использование транслятора и компоновщика, а также работу с отладочной информацией и выполнение программ.

Список литературы

1. Архитектура ЭВМ - Материалы курса
2. NASM Документация