

Отчёт по лабораторной работе 4

Создание и процесс обработки программ на языке ассемблера NASM

Татьяна Соколова НММбд-03-24

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Программа Hello world!	9
4.2	Транслятор NASM	10
4.3	Компоновщик LD	11
4.4	Выполнение заданий для самостоятельной работы.	12
5	Выводы	13
	Список литературы	14

Список иллюстраций

4.1	Создание каталога и файла	9
4.2	Программа hello.asm	10
4.3	Трансляция hello.asm	10
4.4	Трансляция hello.asm с дополнительными опциями	11
4.5	Линковка программы	11
4.6	Линковка программы	11
4.7	Запуск программ	11
4.8	Код программы в файле lab4.asm	12
4.9	Запуск программы lab4.asm	12

Список таблиц

1 Цель работы

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

2 Задание

1. Изучите программу HelloWorld и скомпилируйте ее.
2. С помощью любого текстового редактора внесите изменения в текст программы так, чтобы вместо Hello world! на экран выводилась строка с вашими фамилией и именем.
3. Скомпилируйте новую программу и проверьте ее работу.
4. Загрузите файлы на GitHub.

3 Теоретическое введение

NASM (англ. Netwide Assembler) – это 80x86 ассемблер, разработанный исходя из принципов переносимости и модульности. Он поддерживает широкий диапазон форматов объектных файлов, включая форматы Linux.out и ELF, NetBSD/FreeBSD, COFF, Microsoft 16-bit OBJ и Win32. Он способен также создавать простые бинарные файлы. Синтакс NASM максимально упрощен для понимания и похож на синтакс Intel, но слегка посложнее. Он поддерживает инструкции Pentium, P6 и MMX, а также имеет макро-расширения.

NASM был создан Саймоном Тэтхемом совместно с Юлианом Холлом и в настоящее время развивается небольшой командой разработчиков на SourceForge.net. Первоначально он был выпущен согласно его собственной лицензии, но позже эта лицензия была заменена на GNU LGPL после множества проблем, вызванных выбором лицензии. Начиная с версии 2.07 лицензия заменена на «упрощённую BSD» (BSD из 2 пунктов).

NASM может работать на платформах, отличных от x86, таких как SPARC и PowerPC, однако код он генерирует только для x86 и x86-64.

NASM успешно конкурирует со стандартным в Linux- и многих других UNIX-системах ассемблером gas. Считается, что качество документации у NASM выше, чем у gas. Кроме того, ассемблер gas по умолчанию использует AT&T-синтаксис, ориентированный на процессоры не от Intel, в то время как NASM использует вариант традиционного для x86-ассемблеров Intel-синтаксиса; Intel-синтаксис используется всеми ассемблерами для под DOS/Windows, например, MASM, TASM, fasm.

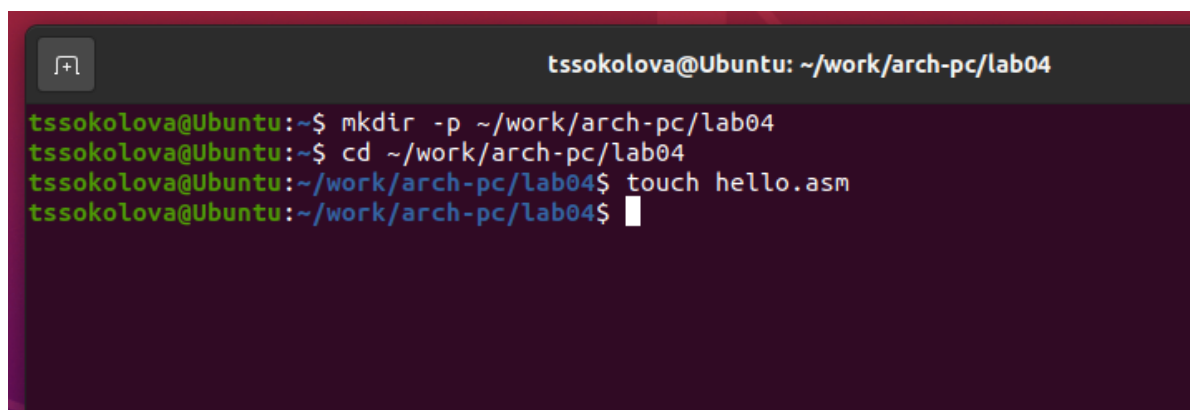
В NASM используется Intel-синтаксис записи инструкций. Предложение языка ассемблера NASM (строка программы) может состоять из следующих элементов: Метка, Инструкция, Операнды, Комментарий.

Операнды отделяются между собой запятой. Перед строкой и после инструкции можно использовать любое количество пробельных символов. Комментарий начинается с точки с запятой, а концом комментария считается конец строки. В качестве инструкции может использоваться команда или псевдокоманда (директива компилятора). Если строка очень длинная, то её можно перенести на следующую, используя обратный слеш (\), подобно тому, как это делается в языке Си.

4 Выполнение лабораторной работы

4.1 Программа Hello world!

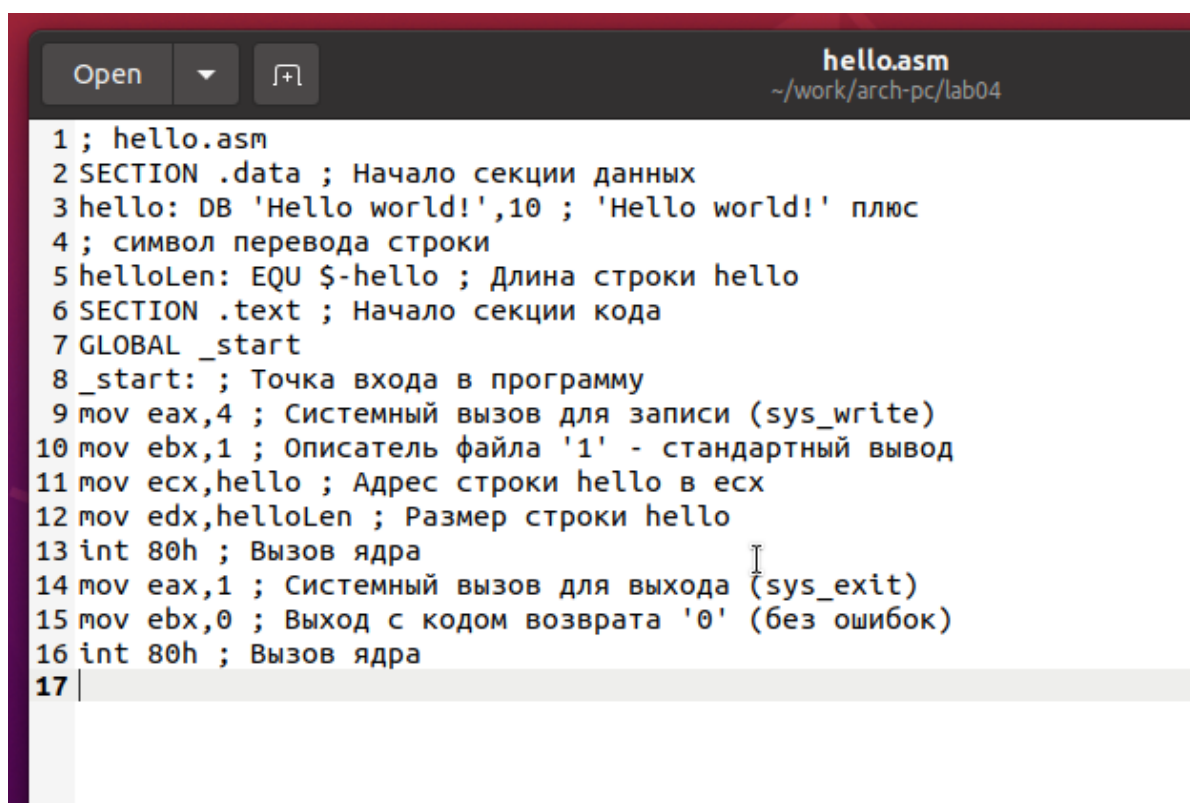
Создаю каталог lab04 командой mkdir, перехожу в него с помощью команды cd, создаю файл hello.asm. (рис. 4.1)

A screenshot of a terminal window with a dark background. The title bar at the top reads "tssokolova@Ubuntu: ~/work/arch-pc/lab04". The terminal shows four lines of commands and their outputs: 1. "tssokolova@Ubuntu:~\$ mkdir -p ~/work/arch-pc/lab04" followed by a new line. 2. "tssokolova@Ubuntu:~\$ cd ~/work/arch-pc/lab04" followed by a new line. 3. "tssokolova@Ubuntu:~/work/arch-pc/lab04\$ touch hello.asm" followed by a new line. 4. "tssokolova@Ubuntu:~/work/arch-pc/lab04\$ " followed by a cursor. The text is color-coded: green for the prompt, blue for the user, and white for the command and output.

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

Рис. 4.1: Создание каталога и файла

Открыла файл и написала код программы по заданию.(рис. 4.2)

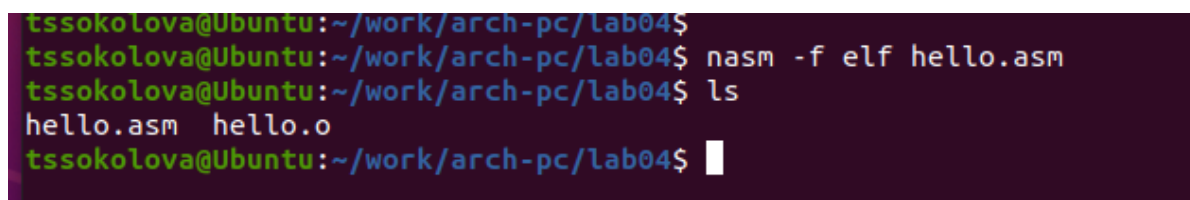


```
1 ; hello.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Hello world!',10 ; 'Hello world!' плюс
4 ; символ перевода строки
5 helloLen: EQU $-hello ; Длина строки hello
6 SECTION .text ; Начало секции кода
7 GLOBAL _start
8 _start: ; Точка входа в программу
9 mov eax,4 ; Системный вызов для записи (sys_write)
10 mov ebx,1 ; Описатель файла '1' - стандартный вывод
11 mov ecx,hello ; Адрес строки hello в ecx
12 mov edx,helloLen ; Размер строки hello
13 int 80h ; Вызов ядра
14 mov eax,1 ; Системный вызов для выхода (sys_exit)
15 mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
16 int 80h ; Вызов ядра
17
```

Рис. 4.2: Программа hello.asm

4.2 Транслятор NASM

Транслирую файл командой `nasm`. Получился объектный файл `hello.o` (рис. 4.3)



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

Рис. 4.3: Трансляция hello.asm

Транслирую файл командой `nasm` с дополнительными опциями. (рис. 4.4)
Получился файл листинга `list.lst`, объектный файл `obj.o`, в программу добавилась отладочная информация.

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

Рис. 4.4: Трансляция hello.asm с дополнительными опциями

4.3 Компоновщик LD

Выполняю линковку командой ld и получил исполняемый файл. (рис. 4.5)

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

Рис. 4.5: Линковка программы

Еще раз выполняю линковку для объектного файла obj.o и получаю исполняемый файл main.(рис. 4.6)

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

Рис. 4.6: Линковка программы

Запускаю исполняемые файлы.(рис. 4.7)

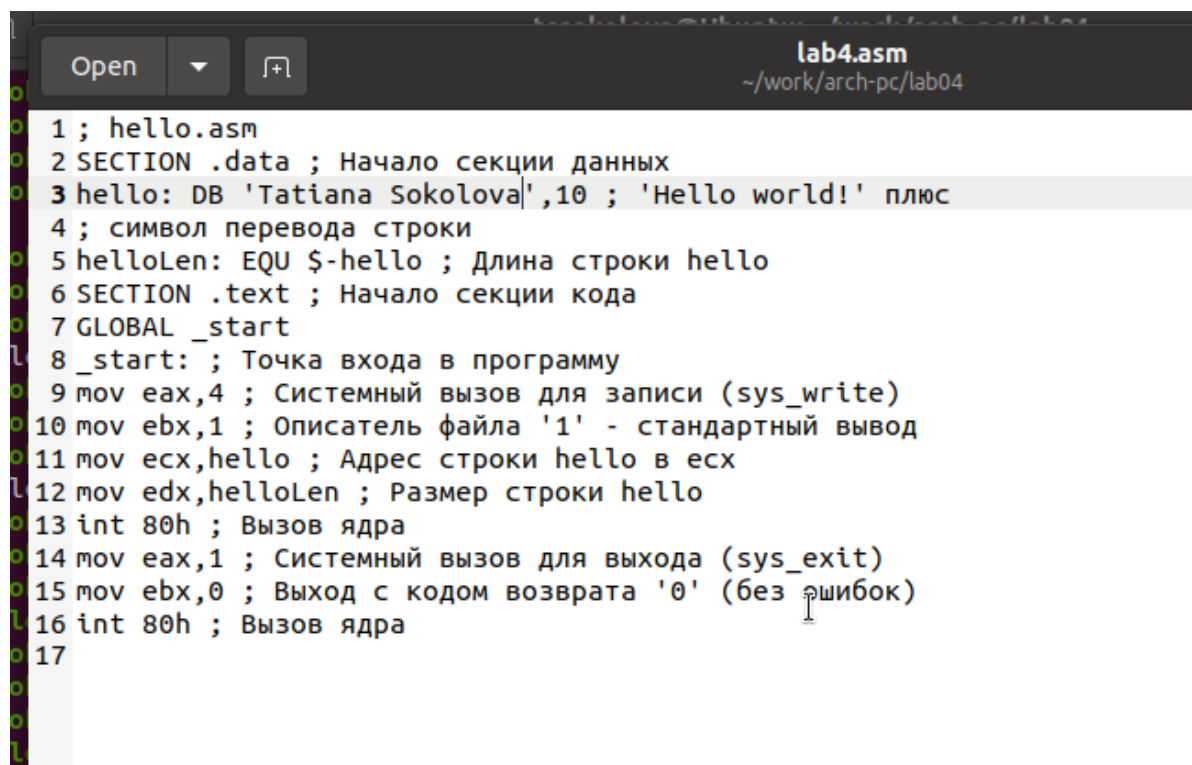
```
tssokolova@Ubuntu:~/work/arch-pc/lab04$
tssokolova@Ubuntu:~/work/arch-pc/lab04$ ./hello
Hello world!
tssokolova@Ubuntu:~/work/arch-pc/lab04$ ./main
Hello world!
tssokolova@Ubuntu:~/work/arch-pc/lab04$
```

Рис. 4.7: Запуск программ

4.4 Выполнение заданий для самостоятельной работы.


Копирую программу в новый файл.

Изменяю сообщение Hello world на свое имя (рис. 4.8) и запускаю новую программу. (рис. 4.9)



```
1 ; hello.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Tatiana Sokolova',10 ; 'Hello world!' плюс
4 ; символ перевода строки
5 helloLen: EQU $-hello ; Длина строки hello
6 SECTION .text ; Начало секции кода
7 GLOBAL _start
8 _start: ; Точка входа в программу
9 mov eax,4 ; Системный вызов для записи (sys_write)
10 mov ebx,1 ; Описатель файла '1' - стандартный вывод
11 mov ecx,hello ; Адрес строки hello в ecx
12 mov edx,helloLen ; Размер строки hello
13 int 80h ; Вызов ядра
14 mov eax,1 ; Системный вызов для выхода (sys_exit)
15 mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
16 int 80h ; Вызов ядра
17
```

Рис. 4.8: Код программы в файле lab4.asm



```
tssokolova@Ubuntu:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
tssokolova@Ubuntu:~/work/arch-pc/lab04$ gedit lab4.asm
tssokolova@Ubuntu:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
tssokolova@Ubuntu:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
tssokolova@Ubuntu:~/work/arch-pc/lab04$ ./lab4
Tatiana Sokolova
tssokolova@Ubuntu:~/work/arch-pc/lab04$
```

Рис. 4.9: Запуск программы lab4.asm

5 Выводы

При выполнении данной лабораторной работы я освоила процесс компиляции и сборки программ, написанных на ассемблере `nasm`.

Список литературы

1. Архитектура ЭВМ
2. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.