

Лабораторная работа №4

**Создание и процесс обработки программ на языке ассемблера
NASM**

Сырцева Анастасия Романовна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
5	Самостоятельная работа	12
6	Выводы	14

Список иллюстраций

3.1	Схема создания ассемблерной программы	8
4.1	Рабочий каталог	10
4.2	Файл hello.asm	10
4.3	Редактирование файла	10
4.4	Преобразование текста программы	11
4.5	Компилирование исходного файла	11
4.6	Проверка	11
4.7	Обработка объектного файла компоновщиком	11
4.8	Обработка объектного файла компоновщиком	11
4.9	Запуск программы	11
5.1	Копирование файла	12
5.2	Открытие редактора	12
5.3	Изменение файла	12
5.4	Компиляция	13
5.5	Компоновка	13
5.6	Вывод программы	13

Список таблиц

1 Цель работы

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

2 Задание

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

3 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. Следует отметить, что процессор понимает не команды ассемблера, а последовательности из нулей и единиц — машинные коды. Преобразование или трансляция команд с языка ассемблера в исполняемый машинный код осуществляется специальной программой транслятором — Ассемблер. Процесс создания ассемблерной программы можно изобразить в виде следующей схемы (рис. 3.1).



Рис. 3.1: Схема создания ассемблерной программы

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*. 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. Из-за специфики программирования, а также по традиции для создания программ на языке ассемблера обычно пользуются утилитами командной строки (хотя поддержка ассемблера есть в некоторых универсальных интегрированных средах).

4 Выполнение лабораторной работы

Создаю рабочий каталог и перехожу в него (рис. 4.1).

```
arsihrceva@dk3n53 ~ $ mkdir -p ~/work/arch-pc/lab04
arsihrceva@dk3n53 ~ $ cd ~/work/arch-pc/lab04
```

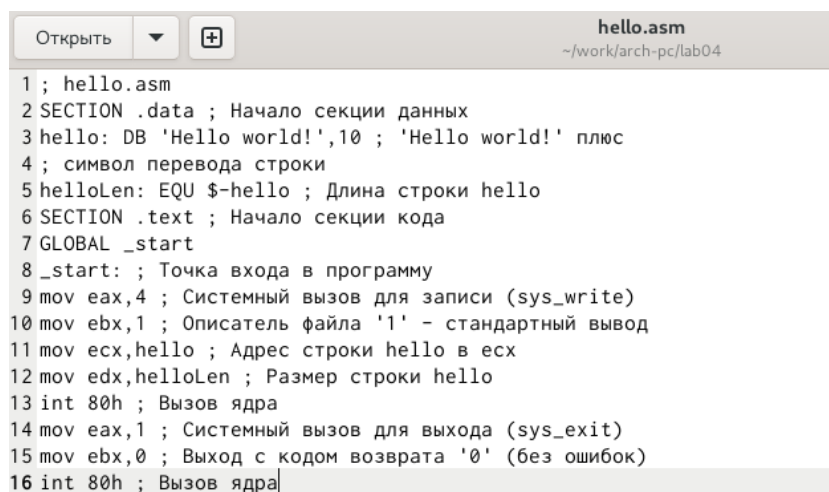
Рис. 4.1: Рабочий каталог

Создаю текстовый файл с именем hello.asm и открываю его в текстовом редакторе gedit (рис. 4.2).

```
arsihrceva@dk3n53 ~/work/arch-pc/lab04 $ touch hello.asm
arsihrceva@dk3n53 ~/work/arch-pc/lab04 $ gedit hello.asm
```

Рис. 4.2: Файл hello.asm

Ввожу в файл текст указанный в условии лабораторной работы (рис. 4.3).



```
Открыть ▼ + hello.asm
~/work/arch-pc/lab04
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
```

Рис. 4.3: Редактирование файла

Компилирую приведённый выше текст и проверяем наличие файла hello.o, в который запишется объектный код (рис. 4.4).

```
arsihrceva@dk3n53 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
arsihrceva@dk3n53 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o
```

Рис. 4.4: Преобразование текста программы

Компилирую исходный файл в obj.o и создаю файл листинга (рис. 4.5).

```
arsihrceva@dk3n53 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
```

Рис. 4.5: Компилирование исходного файла

Проверяю наличие нужных файлов (рис. 4.6).

```
arsihrceva@dk3n53 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o  list.lst  obj.o
```

Рис. 4.6: Проверка

Передаю объектный файл на обработку компоновщику для получения исполняемой программы (рис. 4.7).

```
arsihrceva@dk3n53 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
arsihrceva@dk3n53 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  obj.o
```

Рис. 4.7: Обработка объектного файла компоновщиком

Выполняю команду, указанную в задании (рис. 4.8).

```
arsihrceva@dk3n53 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
```

Рис. 4.8: Обработка объектного файла компоновщиком

Исполняемый файл будет называться main, а объектный файл - obj.o Запускаю исполняемый файл hello (рис. 4.9).

```
arsihrceva@dk3n53 ~/work/arch-pc/lab04 $ ./hello
Hello world!
```

Рис. 4.9: Запуск программы

5 Самостоятельная работа

Копирую файл `hello.asm` в рабочий каталог, меняя название на `lab4.asm` (рис. 5.1).

```
arsihrcева@dk3n53 ~/work/arch-pc/lab04 $ cp hello.asm lab4.asm
```

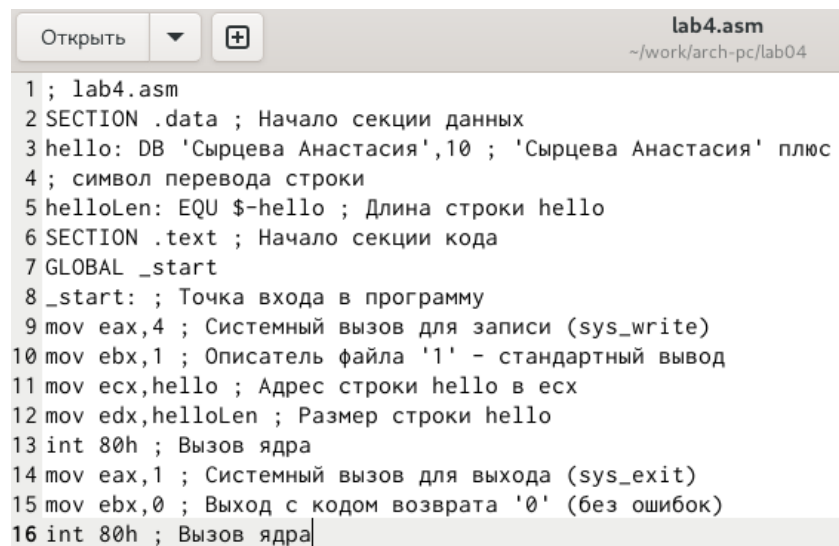
Рис. 5.1: Копирование файла

Открываю текстовый редактор `gedit` (рис. 5.2).

```
arsihrcева@dk3n53 ~/work/arch-pc/lab04 $ gedit lab4.asm
```

Рис. 5.2: Открытие редактора

В открывшемся редакторе вношу изменения в скопированный файл таким образом, чтобы выводилась строка с моим именем и фамилией (рис. 5.3).



```
1 ; lab4.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Сырцева Анастасия',10 ; 'Сырцева Анастасия' 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
```

Рис. 5.3: Изменение файла

Транслирую текст программы в объектный файл и проверяю правильность выполнения (рис. 5.4).

```
arsihrcева@dk3n53 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
arsihrcева@dk3n53 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst lab4.asm
arsihrcева@dk3n53 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4.asm  lab4.o  list.lst  main  obj.o  obj.o
```

Рис. 5.4: Компиляция

Компанирую объектный файл (рис. 5.5).

```
arsihrcева@dk3n53 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o lab4
```

Рис. 5.5: Компановка

Запускаю исполняемого файла (рис. 5.6).

```
arsihrcева@dk3n53 ~/work/arch-pc/lab04 $ ./lab4
Сырцева Анастасия
```

Рис. 5.6: Вывод программы

Копирую файлы hello.asm и lab4.asm в свой локальный репозиторий(https://github.com/Anastasia2025_arh-pc.git) и загружаю написанный отчёт.

6 Выводы

Освоены процедуры компиляции и сборки программ, написанных на ассемблере NASM. Написана программа “Hello world!” и программа, выводящая имя и фамилию.