Отчёт по лабораторной работе № 4

дисциплина: Архитектура компьютера. Создание и процесс обработки программ на языке ассемблера NASM

Студент: Святашова Ксения Евгеньевна

Содержание

# 1 Цель работы

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

# 2 Теоритическое введение

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 DOS/Windows, например, MASM, TASM, fasm.

NASM компилирует программы под различные операционные системы в пределах x86-совместимых процессоров. Находясь в одной операционной системе, можно беспрепятственно откомпилировать исполняемый файл для другой. В общем встроенные средства NASM позволяет компилировать не только программы, но и файлы с любым содержимым. Также мощный макро-препроцессор значительно расширяет возможности для программирования.

Компиляция программ в NASM состоит из двух этапов. Первый — ассемблирование, второй — компоновка. На этапе ассемблирования создаётся объектный код. В нём содержится машинный код программы и данные, в соответствии с исходным кодом, но идентификаторы (переменные, символы) пока не привязаны к адресам памяти. На этапе компоновки из одного или нескольких объектных модулей создаётся исполняемый файл (программа). Операция компоновки связывает идентификаторы, определённые в основной программе, с идентификаторами, определёнными в остальных модулях, после чего всем идентификаторам даются окончательные адреса памяти или обеспечивается их динамическое выделение.

Для компоновки объектных файлов в исполняемые в Windows можно использовать свободный бесплатно распространяемый компоновщик alink(для 64-битных программ компоновщик GoLink), а в Linux — компоновщик ld, который есть в любой версии этой операционной системы.

# 3 Выполнение лабораторной работы

## 3.1 Программа Hello world!

Рассмотрим пример простой программы на языке ассемблера NASM. Традиционно первая программа выводит приветственное сообщение “Hello world!” на экран. Создадим каталог для работы с программами на языке ассемблера NASM(рис. 1):

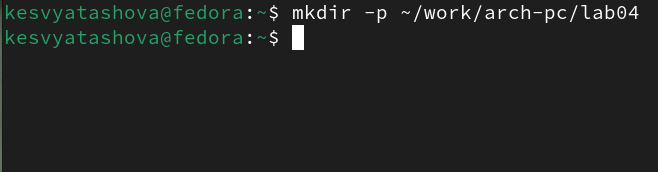


Рис. 1: Создание каталога

Перейдем в созданный каталог(рис. 2):

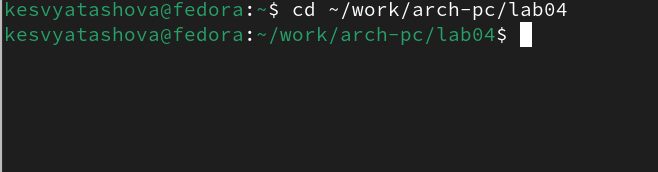


Рис. 2: Каталог

Создадим текстовый файл с именем hello.asm(рис. 3):

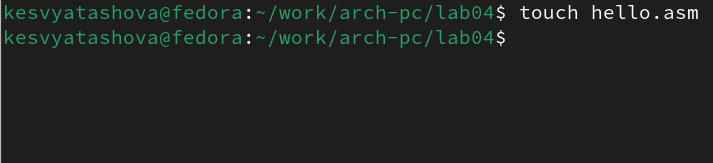


Рис. 3: Файл hello.asm

Откроем этот файл с помощью текстового редактора gedit(рис. 4):



Рис. 4: Открытие файла с помощью gedit

и введем в него следующий текст(рис. 5):

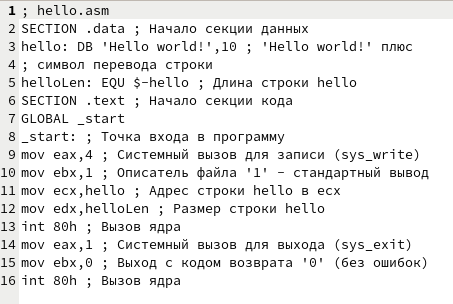


Рис. 5: Текст, выполняющий команду Hello world!

В отличие от многих современных и высокоуровневых языков программирования, в ассемблерной программе каждая команда распологается на *отдельной строке*. Размещение нескольких команд на одной строке **недопустимо**. Синтаксис ассемблера NASM является *чувствительным к регистру*, т.е. есть разница между большими и малыми буквами.

## 3.2 Транслятор NASM

NASM превращает текст программы в объектный код. Для компиляции приведенного выше текста программы “Hello World” необходимо написать(рис. 6):

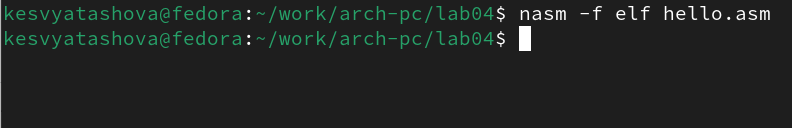


Рис. 6: Компиляция программы Hello world

Если текст программы набран без ошибок, то транслятор преобразует текст программы из файла hello.asm в объектный код, который запишется в файл hello.o. С помощью команды ls проверим, что объектный файл был создан(рис. 7):

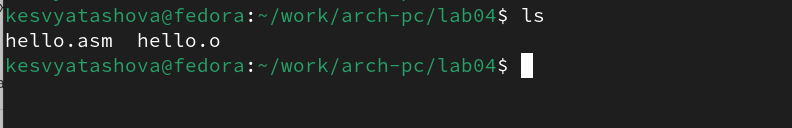


Рис. 7: Проверка объектного файла

## 3.3 Расширенный синтаксис командной строки NASM

Выполним следующую команду(рис. 8), которая скомпилирует исходный файл hello.asm в obj.o(опция -o),кроме того, будет создан файл листинга list.lst(опция -l):



Рис. 8: Компиляция файла obj.o

С помощью команды ls проверим, что файлы были созданы(рис. 9):



Рис. 9: Проверка созданных файлов

## 3.4 Компоновщик LD

Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику(рис. 10).Ключ -o с последующим значением задает имя создаваемого файла.

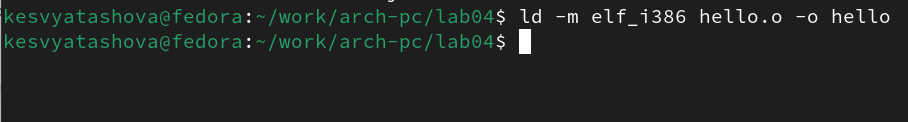


Рис. 10: Обработка программы компоновщиком

С помощью команды ls проверим, что исполняемый файл hello был создан(рис. 11):

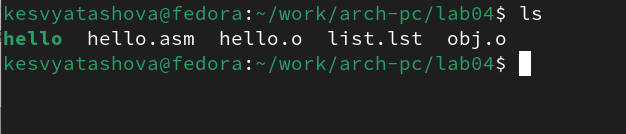


Рис. 11: Проверка исполняемого файла

Введем следующую команду(рис. 12). Исполняемый файл будет иметь имя main, потому что после ключа -o было задано значение main. Объектный файл, из которого собран исполняемый файл, будет называться obj.o.

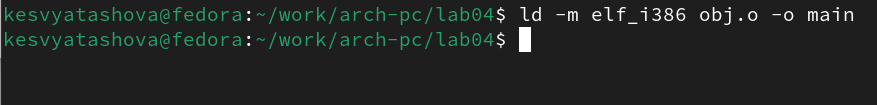


Рис. 12: Обработка компановщиком

Проверяем созданные файлы(рис. 13):

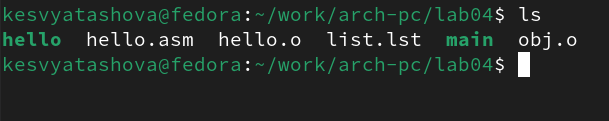


Рис. 13: Проверяем наличие файлов

## 3.5 Запуск исполняемого файла

Запустим на выполнение созданный исполняемый файл, находящийся в текущем каталоге, набрав в командной строке следующую команду(рис. 14):

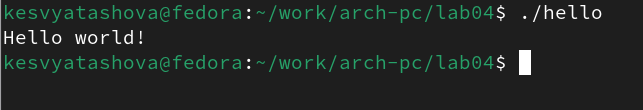


Рис. 14: Выполнение программы Hello world!

# 4 Задания для самостоятельной работы

1. В каталоге ~/work/arch-pc/lab04 с помощью команды cp создадим копию файла hello.asm с именем lab4.asm(рис. 15):

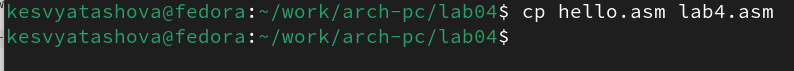


Рис. 15: Выполнение программы Hello world!

1. С помощью текстового редактора gedit внесем изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с нашим фамилией и именем(рис. 16):

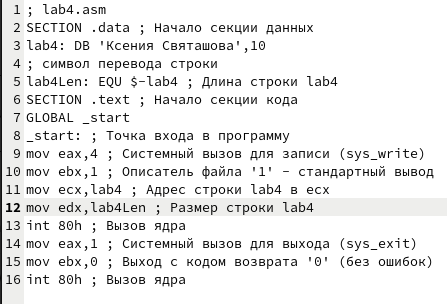


Рис. 16: Программа, выводящаю имя и фамилию

1. Оттранслируем полученный текст программы lab4.asm в объектный файл(рис. 17):

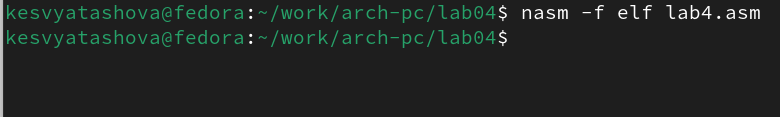


Рис. 17: Оттрансляция lab4.asm в объектный файл

Выполним компоновку объектного файла(рис. 18):

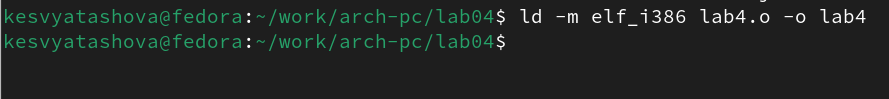


Рис. 18: Компоновка объектного файла

Запустим получившийся исполняемый файл(рис. 19):

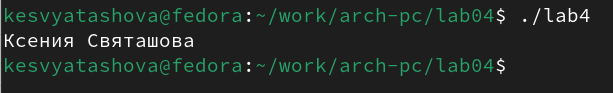


Рис. 19: Выполнение программы, выводящая имя и фамилию

1. Скопируем файлы hello.asm lab4.asm в наш локальный репозиторий в каталог ~/work/study/2024-2025/“Архитектура компьютера”/arch-pc/labs/lab04(рис. 20):

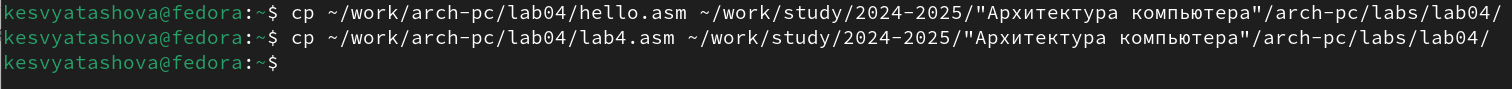


Рис. 20: Копируем файлы

Проверим, что файлы скопировались(рис. 21):

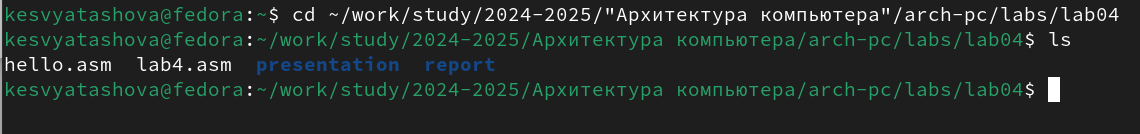


Рис. 21: Проверка файлов

Загрузим файлы на Github(рис. 22):

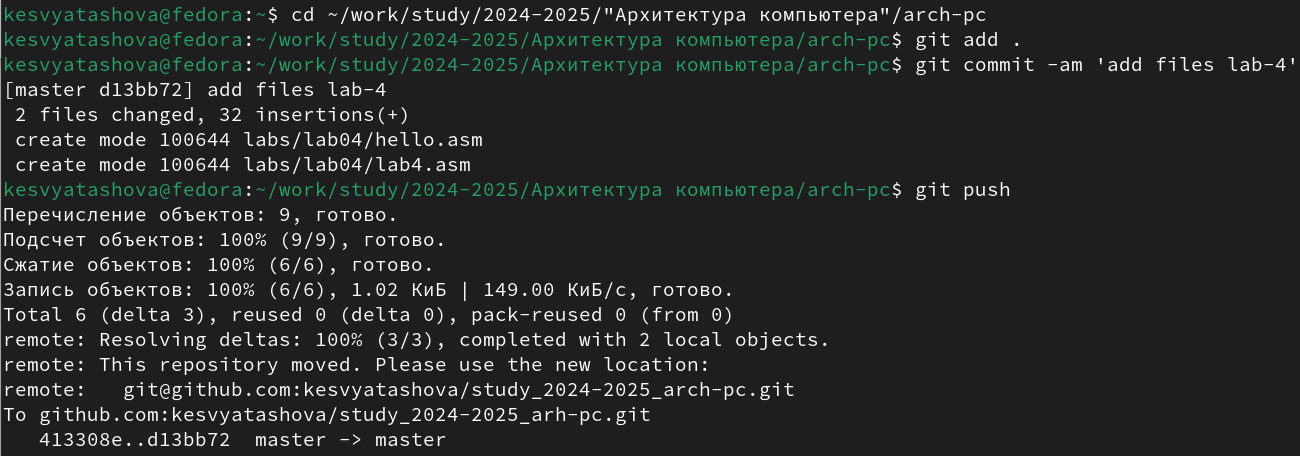


Рис. 22: Загрузка файлов на Github

# 5 Вывод

В результате выполнения работы я освоила процедуру компиляции и сборки программ, написанных на ассемблере NASM.