Отчёт по лабораторной работе №4

Дисциплина: Архитектура Компьютера

Дарина Андреевна Куокконен

Содержание

# 1 Цель работы

Целью данной лабораторной работы - приобретение практического опыта работы с программами, написанными на ассемблере NASM, а именно - освоение процедур компиляций и сборки.

# 2 Задание

1. Создание программы Hello World!  
2. Работа с транслятором NASM  
3. Работа с расширенным синтаксисом командной строки NASM  
4. Работа с компоновщиком LD  
5. Запуск исполняемого файла  
6. Выполнение заданий для самостоятельной работы

# 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. Но 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. Следует отметить, что процессор понимает не команды ассемблера, а последовательности из нулей и единиц — машинные коды. До появления языков ассемблера программистам приходилось писать программы, используя только лишь машинные коды, которые были крайне сложны для запоминания, так как представляли собой числа, записанные в двоичной или шестнадцатеричной системе счисления. Преобразование или трансляция команд с языка ассемблера в исполняемый машинный код осуществляется специальной программой транслятором Программы, написанные на языке ассемблера, не уступают в качестве и скорости программам, написанным на машинном языке, ибо транслятор просто переводит мнемоническиеиобозначения команд в последовательности бит (нулей и единиц). Используемые мнемоники обычно одинаковы для всех процессоров одной архитектуры или семейства архитектур (среди широко известных — мнемоники процессоров и контроллеров x86, ARM, SPARC, PowerPC,M68k). Таким образом для каждой архитектуры существует свой ассемблер и, соответственно, свой язык ассемблера. Наиболее распространёнными ассемблерами для архитектуры x86 являются: 1) Для DOS/Windows: Borland Turbo Assembler (TASM), Microsoft Macro Assembler (MASM) и Watcom assembler (WASM). 2) Для GNU/Linux: gas (GNU Assembler), использующий AT&T-синтаксис, в отличие отбольшинства других популярных ассемблеров, которые используют Intel-синтаксис. Для записи команд в NASM используются: 1) Мнемокод— непосредственно мнемоника инструкции процессору, которая является обязательной частью команды. 2) Операнды - числа, данные, адреса регистров или адреса оперативной памяти. 3) Метка — идентификатор, с которым ассемблер ассоциирует некоторое число, чаще всего адрес в памяти. (Метка перед командой связана с адресом данной команды). Допустимыми символами в метках являются буквы, цифры, а также следующие символы: *, $, #, @,~,. и ?. Начинаться метка или идентификатор могут с буквы, .,*  и ?. Перед идентификаторами, которые пишутся как зарезервированные слова, нужно писать $, чтобы компилятор трактовал его верно (так называемое экранирование). Максимальная длина идентификатора составляет 4095 символов. Программа на языке ассемблера также может содержать директивы — инструкции, не переводящиеся непосредственно в машинные команды, а управляющие работой транслятора. Например, директивы используются для определения данных (констант и переменных) и обычно пишутся большими буквами.

# 4 Выполнение лабораторной работы

4.1) Программа Hello World! Создаю каталог для работы с программами на языке ассемблера NASM. И перехожу в него для дальнейшей работы (рис. [1](#fig:001)).

Figure 1: Создание каталога

Figure 1: Создание каталога

Создаю текстовый файл hello.asm. (рис. [2](#fig:002)).

Figure 2: Создание файла

Figure 2: Создание файла

Открываю созданный файл. (рис. [3](#fig:003)).

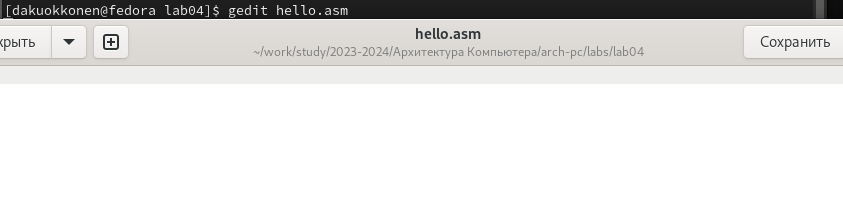


Figure 3: открытие файла

Ввожу нужный текст. (рис. [4](#fig:004)).



Figure 4: Ввод текста

4.2) Работа с транслятором NASM

NASM превращает текст программы в объектный код. Выполним компиляцию приведённого выше текста программы “Hello World”. Сделаем проверку. (рис. [5](#fig:005)).

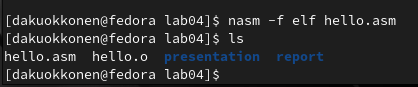


Figure 5: Компиляция текста программы

4.3) Работа с расширенным синтаксисом командной строки NASM

Далее, ввожу команду, которая скомпилирует файл hello.asm в файл obj.o, при этом в файл будут включены символы для отладки (ключ -g), также с помощью ключа -l будет создан файл листинга list.lst. Проверяю правильность выполнения команды. (рис. [6](#fig:006)).

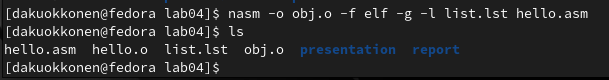


Figure 6: Компиляция текста программы

4.4) Работа с компоновщиком LD

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello. Ключ -о задает имя создаваемого исполняемого файла. С помощью команды “ls” проверяю правильность выполнения.(рис. [7](#fig:007)).

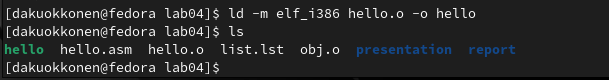


Figure 7: Передача объектного файла на обработку компановщику

Выполняю ту же самую команду со значением main. Объектный файл, из которого собран этот исполняемый файл, имеет имя obj.o (рис. [8](#fig:008))

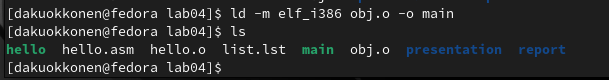


Figure 8: Передача объектного файла на обработку компановщику

4.5) Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello. (рис. [9](#fig:009)).

Figure 9: Запуск исполняемого файла

Figure 9: Запуск исполняемого файла

4.6) Выполнение заданий для самостоятельной работы

С помощью утилиты “cp” создаю в текущем каталоге копию файла hello.asm с именем lab4.asm.(рис. [10](#fig:010)).

Figure 10: Создание копии файла

Figure 10: Создание копии файла

С помощью текстового редактора открываю файл lab4.asm и вношу изменения в программу так, чтобы она выводила мои имя и фамилию. (рис. [11](#fig:011))

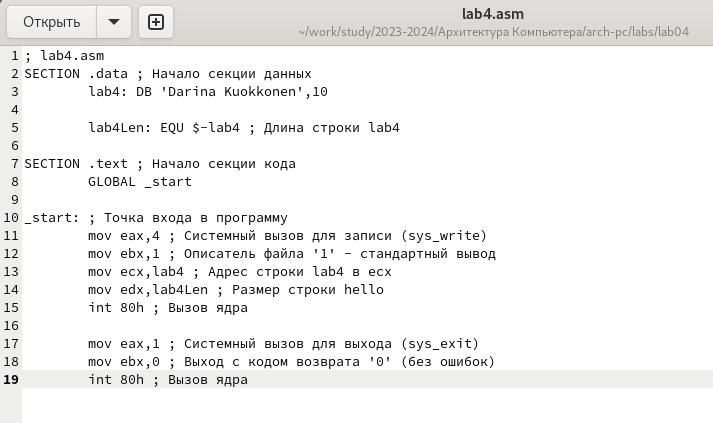


Figure 11: Создание копии файла

Компилирую текст программы в объектный файл, проверяю что файл создан.(рис. [12](#fig:012))

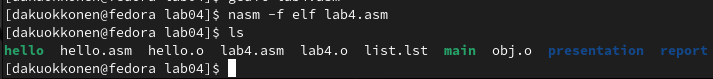


Figure 12: Компиляция текста программы

Передаю файл lab4.asm, на обработку компоновщику LD, чтобы получить исполняемый файл.(рис. [13](#fig:013))

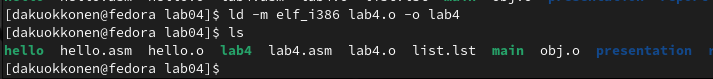


Figure 13: Передача файла на обработку

Затем, запускаю исполняемый файл lab4.asm. Убеждаюсь, что на экран действительно выводятся мои имя и фамилия.(рис. [14](#fig:014))

Figure 14: Запуск файла

Figure 14: Запуск файла

Добавляю файлы в GitHub, используя команду “git add .” и “git commit”, комментируя свое действие как добавление файлов для лабораторной работы №4(рис. [15](#fig:015))

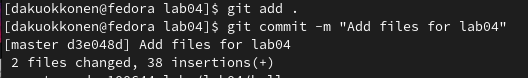


Figure 15: Добавление файлов

Отправляю файлы на сервер с помощью команды “git push”.(рис. [16](#fig:016))



Figure 16: Отправка файлов

# 5 Выводы

При выполнении данной лабораторной работы я приебрела практический опыт работы с программами, написанными на ассемблере NASM, конктретнее - освоила процедуры компиляций и сборки.

# Список литературы

[Архитектура ЭВМ](https://esystem.rudn.ru/pluginfile.php/2089084/mod_resource/content/0/Лабораторная%20работа%20№4.%20Создание%20и%20процесс%20обработки%20программ%20на%20языке%20ассемблера%20NASM.pdf)