Отчёт по лабораторной работе №4

Дисциплина: Архитектура компьютера

Осина Виктория Александровна

Содержание

# 1 Цель работы

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

# 2 Задание

1. Создание программы Hello world!
2. Компиляция текста программы при помощи транслятора NASM
3. Работа с расширенным синтаксисом командной строки NASM
4. Передача объектного файла на обработку компановщику LD
5. Запуска исполняемого файла
6. Задания 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. Заметим, что 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 — Ассемблер.  
Программы, написанные на языке ассемблера, не уступают в качестве и скорости программам, написанным на машинном языке, так как транслятор просто переводит мнемонические обозначения 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:

• для DOS/Windows: Borland Turbo Assembler (TASM), Microsoft Macro Assembler (MASM) и Watcom assembler (WASM); • для GNU/Linux: gas (GNU Assembler), использующий AT&T-синтаксис, в отличие от большинства других популярных ассемблеров, которые используют Intel-синтаксис.

NASM — это открытый проект ассемблера, версии которого доступны под различные операционные системы и который позволяет получать объектные файлы для этих систем. В NASM используется Intel-синтаксис и поддерживаются инструкции x86-64.  
Типичный формат записи команд NASM имеет вид:

[метка:] мнемокод [операнд {, операнд}] [; комментарий] Здесь мнемокод — непосредственно мнемоника инструкции процессору, которая является обязательной частью команды. Операндами могут быть числа, данные, адреса регистров или адреса оперативной памяти. Метка — это идентификатор, с которым ассемблер ассоциирует некоторое число, чаще всего адрес в памяти. Т.о. 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 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 созданный каталог(рис. [[1](#fig:001)])

Figure 1: Рис. 1. Создание каталога

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

Figure 2: Рис. 2. Создание текстового файла

Открываю этот файл с помощью текстового редактора gedit (рис. [[3](#fig:003)])

Figure 3: Рис. 3. Открытие файла в текстовом редакторе

и ввожу в него необходимый текст (рис. [[4](#fig:004)])

Figure 4: Рис. 4. Ввод текста

## 4.2 Транслятор NASM

Для компиляции текста программы «Hello World» пишу следующую команду (рис. [[5](#fig:005)])

Figure 5: Рис. 5. Компиляция программы

С помощью команды ls проверяю, что объектный файл был создан. Файл создается в формате ELF (рис. [[6](#fig:006)])

Figure 6: Рис. 6. Проверка создания файла

## 4.3 Расширенный синтаксис командной строки NASM

Выполняю следующую команду: (рис. [[7](#fig:007)])

Figure 7: Рис. 7. Выполнение команды

С помощью команды ls проверяю, что файл был создан (рис. [[8](#fig:008)])

Figure 8: Рис. 8. Проверка создания файла

## 4.4 Компоновщик LD

Чтобы получить исполняемую программу, передаю объектный файл на обработку компоновщику LD (рис. [[9](#fig:009)])

Figure 9: Рис. 9. Передача файла компановщику

С помощью команды ls проверяю, что исполняемый файл hello был создан. (рис. [[10](#fig:010)])

Figure 10: Рис. 10. Проверка создания файла

Выполняю следующую команду: (рис. [[11](#fig:011)])

Figure 11: Рис. 11. Выполнение команды

Исполняемый файл имеет имя main, объектный файл имеет имя obj.o

## 4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл, находящийся в текущем каталоге, при помощи команды ./hello (рис. [[12](#fig:012)])

Figure 12: Рис. 12. Запуск созданного файла на выполнение

## 4.6 Задание для самостоятельной работы

В каталоге ~/work/arch-pc/lab04 с помощью команды cp создаю копию файла hello.asm с именем lab4.asm (рис. [[13](#fig:013)])

Figure 13: Рис. 13. Создание копий файлов

С помощью текстового редактора gedit вношу изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с моими фамилией и именем (рис. [[14](#fig:014)])

Figure 14: Рис. 14. Изменений текста программы

Транслирую полученный текст программы lab4.asm в объектный файл (рис. [[15](#fig:015)])

Figure 15: Рис. 15. Трансляция текста в объектный файл

Выполняю компоновку объектного файла (рис. [[16](#fig:016)])

Figure 16: Рис. 16. Выполнение комановки

Запускаю получившийся исполняемый файл (рис. [[17](#fig:017)])

Figure 17: Рис. 17. Запуск получившегося файла

Копирую файлы hello.asm и lab4.asm в мой локальный репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/. (рис. [[18](#fig:018)]), (рис. [[19](#fig:019)]) и (рис. [[20](#fig:020)])

Figure 18: Рис. 18. Создание копии файла hello.asm

Figure 19: Рис. 19. Создание копии файла lab04.asm

Figure 20: Рис. 20. Проверка создания копий

Загружаю файлы на Github (рис. [[21](#fig:021)])

Figure 21: Рис. 21. Загрузка файлов на Github

# 5 Выводы

Я освоила процедуру компиляции и сборки программ, написанных на ассемблере NASM

# 6 Список литературы

::: [Архитектура ЭВМ - Лабораторная работа №4](https://esystem.rudn.ru/pluginfile.php/2089084/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%964.%20%D0%A1%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%B8%20%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%20%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B0%20NASM.pdf) :::