## 0.1 Front matter

title: “Отчёт по лабораторной работе №4” subtitle: “Дисциплина: Архитектура Компьютера” author: “Курилко-Рюмин Евгений Михайлович”

## 0.2 Generic otions

lang: ru-RU toc-title: “Содержание”

## 0.3 Bibliography

bibliography: bib/cite.bib csl: pandoc/csl/gost-r-7-0-5-2008-numeric.csl

## 0.4 Pdf output format

toc: true # Table of contents toc-depth: 2 lof: true # List of figures

fontsize: 12pt linestretch: 1.5 papersize: a4 documentclass: scrreprt ## I18n polyglossia polyglossia-lang: name: russian options: - spelling=modern - babelshorthands=true polyglossia-otherlangs: name: english ## I18n babel babel-lang: russian babel-otherlangs: english ## Fonts mainfont: PT Serif romanfont: PT Serif sansfont: PT Sans monofont: PT Mono mainfontoptions: Ligatures=TeX romanfontoptions: Ligatures=TeX sansfontoptions: Ligatures=TeX,Scale=MatchLowercase monofontoptions: Scale=MatchLowercase,Scale=0.9 ## Biblatex biblatex: true biblio-style: “gost-numeric” biblatexoptions: - parentracker=true - backend=biber - hyperref=auto - language=auto - autolang=other\* - citestyle=gost-numeric ## Pandoc-crossref LaTeX customization figureTitle: “Рис.” tableTitle: “Таблица” listingTitle: “Листинг” lofTitle: “Список иллюстраций”

lolTitle: “Листинги” ## Misc options indent: true header-includes: -

# keep figures where there are in the text

## # keep figures where there are in the text

# 1 Цель работы

Целью данной работы является 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 также перехожу в каталог и создаю текстовый файл hello.asm. (рис.1).

|  |
| --- |
| Создание каталога |

Создание каталога

Открытие файла. (рис.2).

|  |
| --- |
| Перемещение |

Перемещение

Ввожу нужный текст. (рис.3).

|  |
| --- |
| Ввод текста |

Ввод текста

4.2) Транслятор NASM

NASM превращает текст программы в объектный код. Выполним компиляцию приведённого выше текста программы “Hello World”. Сделаем проверку. (рис.4).

|  |
| --- |
| Компиляция текста программы |

Компиляция текста программы

4.3) Расширенный синтаксис командной строки NASM

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.5).

|  |
| --- |
| Компиляция текста программы |

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. Выполняю ту же самую команду со значением main. Объектный файл, из которого собран этот исполняемый файл, имеет имя obj.o (рис.6).

|  |
| --- |
| Передача объектного файла на обработку компановщику |

Передача объектного файла на обработку компановщику

4.5) Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello. (рис.7).

|  |
| --- |
| Запуск исполняемого файла |

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. С помощью текстового редактора открываю файл lab4.asm и вношу изменения в программу так, чтобы она выводила мои имя и фамилию. (рис.8).

|  |
| --- |
| Создание копии файла и его открытие, редактирование |

Создание копии файла и его открытие, редактирование

Компилирую текст программы в объектный файл, передаю его компоновщику. Запускаю исполняемый файл lab4, на экран действительно выводятся мои имя и фамилия. (рис.9)

|  |
| --- |
| Компиляция, передача компоновщику |

Компиляция, передача компоновщику

Удаляю лишние файлы в текущем каталоге с помощью утилиты rm, ведь копии файлов остались в другой директории (рис. 4.10). Также с помощью команд git add . и git commit добавляю файлы на GitHub, комментируя действие как добавление файлов для лабораторной работы №4(рис.10)

|  |
| --- |
| Отправка файлов |

Отправка файлов

# 5 Выводы

При выполнении лабораторной работы я обрёл практический опыт работы с программами, написанными на ассемблере NASM, конкретнее - освоил процедуры компиляций и сборки.

# Список литературы

[Архитектура ЭВМ](https://esystem.rudn.ru/pluginfile.php/2089084/mod\_resource/content/0/Лабораторная%20работа%20№4.%2