---

## Front matter

title: "Отчёт по лабораторной работе"

subtitle: "Дисциплина: архитектура компьютера"

author: "Даровских Александра Сергеевна"

## Generic otions

lang: ru-RU

toc-title: "Содержание"

## Bibliography

bibliography: bib/cite.bib

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

## Pdf output format

toc: true # Table of contents

toc-depth: 2

lof: true # List of figures

lot: true # List of tables

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"

lotTitle: "Список таблиц"

lolTitle: "Листинги"

## Misc options

indent: true

header-includes:

- \usepackage{indentfirst}

- \usepackage{float} # keep figures where there are in the text

- \floatplacement{figure}{H} # keep figures where there are in the text

---

# Цель работы

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

#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.

#Теоретическое введение

Основными функциональными элементами любой ЭВМ являются центральный процессор, память и периферийные устройства. Взаимодействие этих устройств осуществляется через общую шину, к которой они подключены. Физически шина представляет собой большое количество проводников, соединяющих устройства друг с другом. В современных компьютерах проводники выполнены в виде электропроводящих дорожек на материнской плате. Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора входят следующие устройства:

` `- арифметико-логическое устройство (АЛУ) — выполняет логические и арифметические действия, необходимые для обработки информации, хранящейся в памяти; - устройство управления (УУ) — обеспечивает управление и контроль всех устройств компьютера;

\- регистры — сверхбыстрая оперативная память небольшого объёма, входящая в состав процессора, для временного хранения промежуточных результатов выполнения инструкций; регистры процессора делятся на два типа: регистры общего назначения и специальные регистры.

Для того, чтобы писать программы на ассемблере, необходимо знать, какие регистры процессора существуют и как их можно использовать. Большинство команд в программах, написанных на ассемблере, используют регистры в качестве операндов. Практически все команды представляют собой преобразование данных, хранящихся в регистрах процессора, это, например пересылка данных между регистрами или между регистрами и памятью, преобразование (арифметические или логические операции) данных хранящихся в регистрах. Доступ к регистрам осуществляется не по адресам, как к основной памяти, а по именам. Каждый регистр процессора архитектуры x86 имеет свое название, состоящее из 2 или 3 букв латинского алфавита. В качестве примера приведем названия основных регистров общего назначения (именно эти регистры чаще всего используются при написании программ): - RAX, RCX, RDX, RBX, RSI, RDI — 64-битные - EAX, ECX, EDX, EBX, ESI, EDI — 32-битные - AX, CX, DX, BX, SI, DI — 16-битные - AH, AL, CH, CL, DH, DL, BH, BL — 8-битные Другим важным узлом ЭВМ является оперативное запоминающее устройство (ОЗУ). ОЗУ — это быстродействующее энергозависимое запоминающее устройство, которое напрямую взаимодействует с узлами процессора, предназначенное для хранения программ и данных, с которыми процессор непосредственно работает в текущий момент. ОЗУ состоит из одинаковых пронумерованных ячеек памяти. Номер ячейки памяти — это адрес хранящихся в ней данных. Периферийные устройства в составе ЭВМ: - устройства внешней памяти, которые предназначены для долговременного хранения больших объёмов данных. - устройства ввода-вывода, которые обеспечивают взаимодействие ЦП с внешней средой. В основе вычислительного процесса ЭВМ лежит принцип программного управления. Это означает, что компьютер решает поставленную задачу как последовательность действий, записанных в виде программы. Коды команд представляют собой многоразрядные двоичные комбинации из 0 и 1. В коде машинной команды можно выделить две части: операционную и адресную. В операционной части хранится код команды, которую необходимо выполнить. В адресной части хранятся данные или адреса данных, которые участвуют в выполнении данной операции. При выполнении каждой команды процессор выполняет определённую последовательность стандартных действий, которая называется командным циклом процессора. Он заключается в следующем: 1. формирование адреса в памяти очередной команды; 2. считывание кода команды из памяти и её дешифрация; 3. выполнение команды; 4. переход к следующей команде. Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. 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.

#Выполнение лабораторной работы

##4.1 Создание программы Hello world!

С помощью команды cd перемещаемся в каталог, в котором будем работать(рис.[-@fig:001])

![Перемещение между директориями](/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/1.png){#@fig:001 width=70%}

Создаем в текущем каталоге пустой текстовый файл hello.asm с помощью команды touch (рис.[-@fig:002]).

![Создание пустого файла](/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/2.png){#@fig:002 width=70%}

Открываем созданный файл в текстовом редакторе gedit (рис. [-@fig:003],[-@fig:004]).

![Открытие файла в текстовом редакторе](/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/3.png,/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/4.png){#@fig:003 width=70%},{#@fig:004 width=70%}

Заполняем файл, вставляя в него программу для вывода “Hello word!” (рис.[-@fig:005]).

![Заполнение файла](/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/5.png){#@fig:005 width=70%}

##4.2 Работа с транслятором NASM

Превращаем текст программы для вывода “Hello world!” в объектный код с помощью транслятора NASM, используя команду nasm -f elf hello.asm, ключ -f указывает транслятору nasm, что требуется создать бинарный файл в формате ELF (рис.

[-@fig:006]). Далее проверяем правильность выполнения команды с помощью команды ls: действительно, создан файл “hello. o”.

![Компиляция текста программы](/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/6.png){#@fig:006 width=70%}

##4.3 Работа с расширенным синтаксисом командной строки NASM\*\*

Вводим 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 10 будет создан файл листинга list.lst (рис.[-@fig:007]). Далее проверяем с помощью команды ls правильность выполнения команды.

![Компиляция текста программы](/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/7.png){#@fig:007 width=70%}

##4.4 Работа с компоновщиком LD

Передаем объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello (риc.[-@fig:008]). Ключ -о задает имя создаваемого исполняемого файла. Далее проверяем с помощью команды ls правильность выполнения команды.

![Передача объектного файла на обработку компоновщику](/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/8.png){#@fig:008 width=70%}

Выполняем следующую команду (рис.[-@fig:009]). Исполняемый файл будет иметь имя main, т.к. после ключа -о было задано значение main. Объектный файл, из которого собран этот исполняемый файл, имеет имя obj.o

![Передача объектного файла на обработку компоновщику](/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/9.png){#@fig:009 width=70%}

##4.5 Запуск исполняемого файла

Запускаем на выполнение созданный исполняемый файл hello (рис.[-@fig:010]).

![Запуск исполняемого файла](/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/10.png){#@fig:010 width=70%}

##4.6 Выполнение заданий для самостоятельной работы

С помощью команды cp создаем в текущем каталоге копию файла hello.asm с именем lab04.asm (рис. [-@fig:011]).

![Создание копии файла](/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/11.png){#@fig:011 width=70%}

С помощью текстового редактора gedit открываем файл lab04.asm и вносим изменения в программу так, чтобы она выводила имя и фамилию. (рис. [-@fig:012]).

![Изменение программы](/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/12.png){#@fig:012 width=70%}

Компилируем текст программы в объектный файл (рис.[-@fig:013]). Проверяем с помощью команды ls, что файл lab04.o создан.

![Компиляция текста программы](/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/13.png){#@fig:013 width=70%}

Передаем объектный файл lab04.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4 (рис.[-@fig:014]).

![Передача объектного файла на обработку компоновщику](/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/14.png){#@fig:014 width=70%}

Запускаем исполняемый файл lab4, на экран действительно выводятся мои имя и фамилия (рис.[-@fig:015]).

![Запуск исполняемого файла](/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/15.png){#@fig:015 width=70%}

Так как по заданию нужно было создать каталог для работы с программами на языке ассемблера NASM, чего мною было не сделано, то я создаю другую директорию lab04 с помощью mkdir, прописывая полный путь к каталогу, в котором хочу создать эту директорию. Далее копирую из текущего каталога файлы, созданные в процессе выполнения лабораторной работы, с помощью утилиты cp, указывая вместо имени файла символ \\*, чтобы скопировать все файлы. Команда проигнорирует директории в этом каталоге, т. к. не указан ключ -r, это мне и нужно (рис. [-@fig:016]).

Проверяю с помощью утилиты ls правильность выполнения команды.

![Создании копии файлов в новом каталоге](/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/16.png){#@fig:016 width=70%}

Удаляем лишние файлы в текущем каталоге с помощью команды rm, ведь копии файлов остались в другой директории (рис. [-@fig:017]).

![Удаление лишних файлов в текущем каталоге](/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/17.png){#@fig:017 width=70%}

С помощью команд git add . и git commit добавляем файлы на GitHub, комментируя действие как добавление файлов для лабораторной работы №5 (рис.[-@fig:018]).

![Добавление файлов на GitHub](/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/18.png){#@fig:018 width=70%}

Отправляем файлы на сервер с помощью команды git push (рис.[-@fig:019]).

![Отправка файлов](/home/asdarovskikh/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04/report/image/19.png){#@fig:019 width=70%}

#Выводы

При выполнении данной лабораторной работы я освоила процедуры компиляции и сборки программ, написанных на ассемблере NASM.

#Список литературы

1. [ref1]: 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