

# **Отчёта по лабораторной работе №5**

**Дисциплина: архитектура компьютера**

Гашимова Эсма Эльшан кызы

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>10</b>
4.1	Создание программы Hello world! . . . . .	10
4.2	Транслятор NASM . . . . .	10
4.3	Расширенный синтаксис командной строки NASM . . . . .	10
4.4	Компоновщик LD . . . . .	11
4.5	Запуск исполняемого файла . . . . .	11
4.6	Задания для самостоятельной работы . . . . .	11
<b>5</b>	<b>Выводы</b>	<b>13</b>
<b>6</b>	<b>Список литературы</b>	<b>14</b>

# Список иллюстраций

4.1	Запуск программы . . . . .	11
-----	----------------------------	----

## **Список таблиц**

# 1 Цель работы

Цель данной лабораторной работы - освоить процедуры компиляции и сборки программ, написанных на ассемблере NASM.

## 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

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.

## 4 Выполнение лабораторной работы

### 4.1 Создание программы Hello world!

В домашней директории создаю каталог, в котором буду хранить файлы для текущей лабораторной работы. (рис. -fig. ??). Создание рабочей директории

Создаю в текущем каталоге пустой текстовый файл hello.asm с помощью утилиты touch (рис. -fig. ??). Создание пустого файла

Открываю созданный файл в текстовом редакторе mousepad (рис. -fig. ??). Открытие файла в текстовом редакторе

Заполняю файл, вставляя в него программу для вывода “Hello word!”

### 4.2 Транслятор NASM

Превращаю 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. Далее проверяю правильность выполнения команды с помощью утилиты `ls`: действительно, создан файл “hello.o”. (рис. -fig. ??) Компиляция текста программы

### 4.3 Расширенный синтаксис командной строки NASM

Выполняю команду, указанную на (рис. -fig. ??), она скомпилировала исходный файл hello.asm в obj.o, расширение .o говорит о том, что файл - объектный,

помимо него флаги -g -l подготовят файл отладки и листинга соответственно. Возможности синтаксиса NASM

## 4.4 Компоновщик LD

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello (рис. -fig. ??). Ключ -o задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты ls правильность выполнения команды. Отправка файла компоновщику

Выполняю следующую команду (рис. -fig. ??). Исполняемый файл будет иметь имя main, т.к. после ключа -o было задано значение main. Объектный файл, из которого собран этот исполняемый файл, имеет имя obj.o Передача объектного файла на обработку компоновщику

## 4.5 Запуск исполняемого файла

Запускаю исполняемый файл из текущего каталога. (рис. -fig. 4.1)

Запуск программы

Рис. 4.1: Запуск программы

## 4.6 Задания для самостоятельной работы

Создаю копию файла для последующей работы с ней. (рис. -fig. ??) Создание копии

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

Компилирую текст программы в объектный файл (рис. -fig. ??). Проверяю с помощью утилиты ls, что файл lab4.o создан. Компиляция текста программы

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab5 (рис. -fig. ??). Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4, на экран действительно выводятся мои имя и фамилия (рис. -fig. ??) Запуск исполняемого файла

Копирую рабочие файлы в свой локальный репозиторий. (рис. -fig. ??) Отправка файлов в локальный репозиторий

С помощью команд `git add .` и `git commit` добавляю файлы на GitHub, комментируя действие как добавление файлов для лабораторной работы №5 (рис. -fig. ??) Добавление файлов на GitHub

Отправляю файлы на сервер с помощью команды `git` (рис. -fig. ??) Отправка файлов

## **5 Выводы**

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

## 6 Список литературы

1. [https://esystem.rudn.ru/pluginfile.php/1584628/mod\\_resource/content/1/%D0%9B%D0%B0%](https://esystem.rudn.ru/pluginfile.php/1584628/mod_resource/content/1/%D0%9B%D0%B0%)