

# **Отчёт по лабораторной работе №4**

**Дисциплина: Архитектура компьютера**

Орлов Илья Сергеевич

# 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	Программа Hello world! . . . . .	10
4.2	Транслятор NASM . . . . .	11
4.3	Расширенный синтаксис командной строки NASM . . . . .	11
4.4	Компоновщик LD . . . . .	11
4.5	Запуск исполняемого файла . . . . .	12
4.6	Задания для самостоятельной работы . . . . .	12
<b>5</b>	<b>Выводы</b>	<b>14</b>
<b>6</b>	<b>Список литературы</b>	<b>15</b>

## Список иллюстраций

4.1	Создание рабочей директории . . . . .	10
4.2	Создание .asm файла . . . . .	10
4.3	Редактирование файла . . . . .	10
4.4	Компиляция программы . . . . .	11
4.5	Возможности синтаксиса NASM . . . . .	11
4.6	Отправка файла компоновщику . . . . .	11
4.7	Создание исполняемого файла . . . . .	11
4.8	Запуск программы . . . . .	12
4.9	Создание копии . . . . .	12
4.10	Редактирование копии . . . . .	12
4.11	Проверка работоспособности скомпонованной программы . . . . .	13
4.12	Отправка файлов в локальный репозиторий . . . . .	13
4.13	Загрузка изменений . . . . .	13

## **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 Программа Hello world!

В домашней директории создаю каталог, в котором буду хранить файлы для текущей лабораторной работы. (рис. -fig. 4.1)

```
rutnixya@rutnix-VirtualBox:~$ mkdir -p ~/work/arch-pc/lab04
rutnixya@rutnix-VirtualBox:~$ cd ~/work/arch-pc/lab04
```

Рис. 4.1: Создание рабочей директории

Создаю в нем файл hello.asm, в котором буду писать программу на языке ассемблера. (рис. -fig. 4.2)

```
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab04$ touch hello.asm
```

Рис. 4.2: Создание .asm файла

С помощью редактора пишу программу в созданном файле. (рис. -fig. 4.3)

```
1; hello.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Hello world!',10 ; 'Hello world!' плюс
4 ; символ перевода строки
5 helloLen: EQU $-hello ; Длина строки hello
6 SECTION .text ; Начало секции кода
7 GLOBAL _start
8 _start: ; Точка входа в программу
9 mov eax,4 ; Системный вызов для записи (sys_write)
10 mov ebx,1 ; Описатель файла '1' - стандартный вывод
11 mov ecx,hello ; Адрес строки hello в ecx
12 mov edx,helloLen ; Размер строки hello
13 int 80h ; Вызов ядра
14 mov eax,1 ; Системный вызов для выхода (sys_exit)
15 mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
16 int 80h ; Вызов ядра
```

Рис. 4.3: Редактирование файла

## 4.2 Транслятор NASM

Компилирую с помощью NASM свою программу. (рис. -fig. 4.4)

```
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab04$ nasm -f elf hello.asm
```

Рис. 4.4: Компиляция программы

## 4.3 Расширенный синтаксис командной строки NASM

Выполняю команду, указанную на (рис. -fig. 4.5), 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.

```
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab04$ ls
hello.asm  hello.o  list.lst  obj.o
```

Рис. 4.5: Возможности синтаксиса NASM

## 4.4 Компоновщик LD

Затем мне необходимо передать объектный файл компоновщику, делаю это с помощью команды ld. (рис. -fig. 4.6)

```
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  list.lst  obj.o
```

Рис. 4.6: Отправка файла компоновщику

Выполняю следующую команду ..., результатом исполнения команды будет созданный файл main, скомпонованный из объектного файла obj.o. (рис. -fig. 4.7)

```
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
```

Рис. 4.7: Создание исполняемого файла

## 4.5 Запуск исполняемого файла

Запускаю исполняемый файл из текущего каталога. (рис. -fig. 4.8)

```
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab04$ ./hello
Hello world!
```

Рис. 4.8: Запуск программы

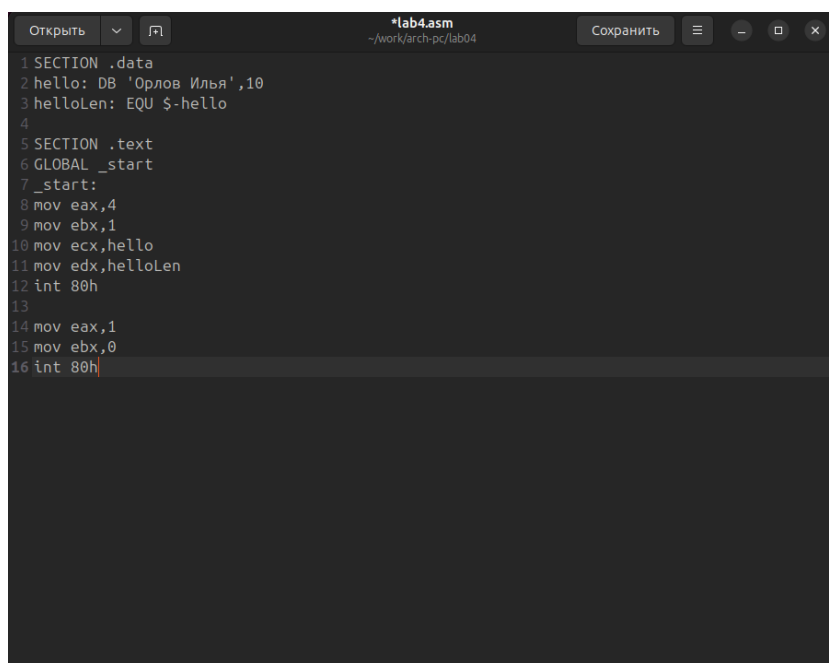
## 4.6 Задания для самостоятельной работы

Создаю копию файла для последующей работы с ней. (рис. -fig. 4.9)

```
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
```

Рис. 4.9: Создание копии

Редактирую копию файла, заменив текст на свое имя и фамилию. (рис. -fig. 4.10)



```
Открыть  [icon]  *lab4.asm  Сохранить  [icon]  [icon]  [icon]
~/work/arch-pc/lab04
1 SECTION .data
2 hello: DB 'Орлов Илья',10
3 helloLen: EQU $-hello
4
5 SECTION .text
6 GLOBAL _start
7 _start:
8 mov eax,4
9 mov ebx,1
10 mov ecx,hello
11 mov edx,helloLen
12 int 80h
13
14 mov eax,1
15 mov ebx,0
16 int 80h
```

Рис. 4.10: Редактирование копии

Транслирую копию файла в объектный файл, компоную и запускаю. (рис. -fig. 4.11)

```
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab04$ ./lab4
Орлов Илья
```

Рис. 4.11: Проверка работоспособности скомпонованной программы

Убедившись в корректности работы программы, копирую рабочие файлы в свой локальный репозиторий. (рис. -fig. 4.12)

```
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab04$ cp hello.asm ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04/
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab04$ cp lab4.asm ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04/
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab04$ cd ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04/
rutnixya@rutnix-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello.asm lab4.asm presentation report
```

Рис. 4.12: Отправка файлов в локальный репозиторий

Загрузка изменений на свой удаленный репозиторий на GitHub. (рис. -fig. 4.13)

```
rutnixya@rutnix-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ git add hello.asm lab4.asm
rutnixya@rutnix-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ git commit -m "Add lab04 files"
[master 324cd90] Add lab04 files
2 files changed, 32 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
rutnixya@rutnix-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 1.02 Киб | 1.02 Миб/с, готово.
Всего 6 (изменений 2), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:Cocico4kalnao/study_2024-2025_arch-pc.git
92608ac..324cd90 master -> master
rutnixya@rutnix-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ ls
```

Рис. 4.13: Загрузка изменений

## **5 Выводы**

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

## **6 Список литературы**

1. Пример выполнения лабораторной работы
2. Курс на ТУИС
3. Лабораторная работа №4
4. Программирование на языке ассемблера NASM Столяров А. В.