

# **Отчёт по лабораторной работе №4**

**Дистиплина: архитектура компьютера**

Худдыева Дженнет

# 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	Работа с транслятором NASM . . . . .	11
4.3	Работа с расширенным синтаксисом командной строки NASM . .	11
4.4	Работа с компоновщиком LD . . . . .	12
4.5	Запуск исполняемого файла . . . . .	12
4.6	Выполнение заданий для самостоятельной работы . . . . .	13
<b>5</b>	<b>Выводы</b>	<b>15</b>

## Список иллюстраций

4.1	Перемещение между директориями . . . . .	10
4.2	Создание пустого файла . . . . .	10
4.3	Заполнение файла . . . . .	11
4.4	Компиляция текста программы . . . . .	11
4.5	Компиляция текста программы . . . . .	12
4.6	Передача объектного файла на обработку компоновщику . . . . .	12
4.7	Передача объектного файла на обработку компоновщику . . . . .	12
4.8	Запуск исполняемого файла . . . . .	12
4.9	Создание копии файла . . . . .	13
4.10	Изменение программы . . . . .	13
4.11	Компиляция текста программы . . . . .	13
4.12	Передача объектного файла на обработку компоновщику . . . . .	14
4.13	Запуск исполняемого файла . . . . .	14

## **Список таблиц**

# 1 Цель работы

Цель данной лабораторной работы-освоить процедуры компиляции и сборки программ,написанных на ассемблера 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 компьютеров. В состав центрального процессора входят следующие устройства: - арифметико-логическое устройства (АЛУ) - выполняет логические и арифметические действия, необходимые для обработки информации, хранящейся в памяти. - устройства 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 ассемблера, необходимо знать, какие регистры процессора существуют и как их можно использовать. Большинство команд в программах написанных на ассемблера используют регистры в качестве операндов. Практически все команды представляют с собой преобразование данных хранящихся в регистрах процессоров, это например пересылка данных между регистрами или 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 3 или 3 букв латинского алфавита. В качестве примера приведём названия основных регистров общего назначения (именно эти регистры чаще всего используются при написании программ): - RAX, RCX, RDX, RBX, RSI, RDI - 64 bit - EAX, ECX, EDX, EBX, ESI, EDI - 32 bit - AX, CX, DX, BX, SI, DI - 16 bit - AH, AL, CH, DH, DL, BH, BL - 8 bit

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 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 команды представляют собой многозарядные 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 данные, 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.

## 4 Выполнение лабораторной работы

### 4.1 Создание программы Hello world!

С помощью утилиты `cd` перемещаюсь в каталог, в котором буду работать (рис. [4.1]).

```
dkhuddihevagdkhuddiheva-VirtualBox: $ cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc
dkhuddihevagdkhuddiheva-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc$ cd labs
dkhuddihevagdkhuddiheva-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs$ cd lab04
dkhuddihevagdkhuddiheva-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$
```

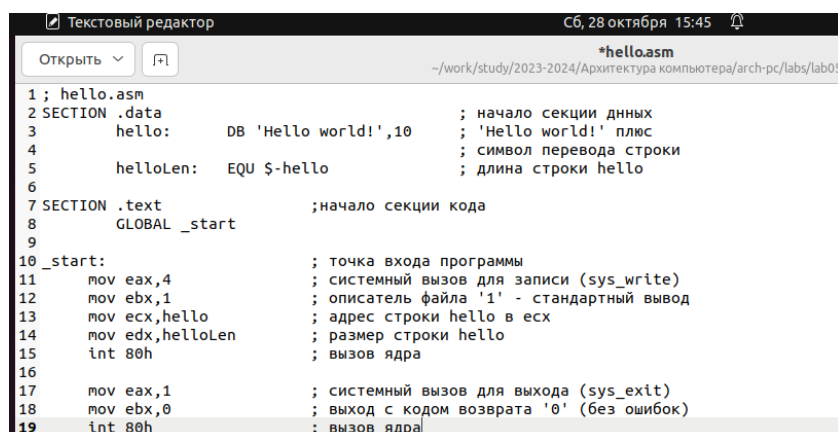
Рис. 4.1: Перемещение между директориями

Создаю в текущем каталоге пустой текстовый файл `hello.asm` с помощью утилиты `touch` (рис. [4.2]).

```
dkhuddihevagdkhuddiheva-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ touch hello.asm
dkhuddihevagdkhuddiheva-VirtualBox: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$
```

Рис. 4.2: Создание пустого файла

Открываю созданный файл в текстовом редакторе `gedit`. Заполняю файл, вставляя в него программу для вывода "Hello world!" (рис. [4.3]).

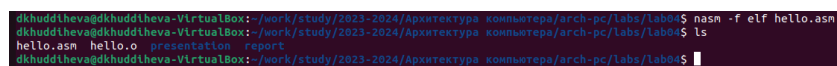


```
1 ; hello.asm
2 SECTION .data
3     hello:    DB 'Hello world!',10    ; начало секции данных
4                                     ; 'Hello world!' плюс
5     helloLen: EQU $-hello            ; символ перевода строки
6                                     ; длина строки hello
7 SECTION .text
8     GLOBAL _start                    ; начало секции кода
9
10 _start:
11     mov eax,4                        ; точка входа программы
12     mov ebx,1                        ; системный вызов для записи (sys_write)
13     mov ecx,hello                    ; описатель файла '1' - стандартный вывод
14     mov edx,helloLen                 ; адрес строки hello в ecx
15     int 80h                          ; размер строки hello
16                                     ; вызов ядра
17
18     mov eax,1                        ; системный вызов для выхода (sys_exit)
19     mov ebx,0                        ; выход с кодом возврата '0' (без ошибок)
20     int 80h                          ; вызов ядра
```

Рис. 4.3: Заполнение файла

## 4.2 Работа с транслятором NASM

Превращаю текст программы для вывода “Hello world!” в объектный код с помощью транслятора NASM, используя команду `-f elf hello.asm` ключ `-f` указывает транслятору nasm, что требуется создать бинарный файл в формате ELF. Далее проверяю выполнение команды с помощью `ls` (рис. [4.4]).



```
dkhuddhevagdkhuddheva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ nasm -f elf hello.asm
dkhuddhevagdkhuddheva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello.asm  hello.o  presentation  report
```

Рис. 4.4: Компиляция текста программы

## 4.3 Работа с расширенным синтаксисом 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` (рис. [4.5]).

```

dkhuddheva@dkhuddheva-VirtualBox: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ nasm -o obj.o -f elf -g -l list
.lst hello.asm
dkhuddheva@dkhuddheva-VirtualBox: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello.asm hello.o list.lst obj.o presentation report
dkhuddheva@dkhuddheva-VirtualBox: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$

```

Рис. 4.5: Компиляция текста программы

## 4.4 Работа с компоновщиком LD

Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello (рис. [4.6]).

```

dkhuddheva@dkhuddheva-VirtualBox: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ld -m elf_i386 hello.o -o hello
dkhuddheva@dkhuddheva-VirtualBox: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello.asm hello.o list.lst main obj.o presentation report
dkhuddheva@dkhuddheva-VirtualBox: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$

```

Рис. 4.6: Передача объектного файла на обработку компоновщику

Выполняю следующую команду (рис. [4.7]). Исполняемый файл будет иметь имя main, после ключа -o было задано значение main. Объектный файл, из которого собран этот исполняемый файл имеет имя obj.o

```

dkhuddheva@dkhuddheva-VirtualBox: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ld -m elf_i386 obj.o -o main
dkhuddheva@dkhuddheva-VirtualBox: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello.asm hello.o list.lst main obj.o presentation report
dkhuddheva@dkhuddheva-VirtualBox: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$

```

Рис. 4.7: Передача объектного файла на обработку компоновщику

## 4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello (рис. [4.8]).

```

dkhuddheva@dkhuddheva-VirtualBox: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ./hello
Hello world!
dkhuddheva@dkhuddheva-VirtualBox: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$

```

Рис. 4.8: Запуск исполняемого файла

## 4.6 Выполнение заданий для самостоятельной работы

С помощью утилиты `ср` создаю в текущем каталоге копию файла `hello.asm` с именем `lab4.asm` (рис. [4.9]).

```
dkhuddheva@dkhuddheva-VirtualBox: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ cp hello.asm lab4.asm
dkhuddheva@dkhuddheva-VirtualBox: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ gedit lab4.asm
```

Рис. 4.9: Создание копии файла

С помощью текстового редактора `gedit` открываю файл `lab4.asm` и вношу изменения в программу так, чтобы она выводила моё имя и фамилию (рис. [4.10]).

```
1 ; lab4.asm
2 SECTION .data                                ; начало секции данных
3     lab4:  DB 'Jennet Huddyyewa',10
4                                           ; символ перевода строки
5     lab4Len: EQU $-lab4                      ; длина строки lab4
6
7 SECTION .text                                ; начало секции кода
8     GLOBAL _start
9
10 _start:                                     ; точка входа в программу
11     mov eax,4                               ; системный вызов для записи (sys_write)
12     mov ebx,1                               ; описать файла '1' - стандартный вывод
13     mov ecx,lab4                            ; адрес строки lab4 в ecx
14     mov edx,lab4Len                         ; размер строки lab
15     int 80h                                ; вызов ядра
16
17     mov eax,1                               ; системный вызов для выхода (sys_exit)
18     mov ebx,0                               ; выход с кодом возврата '0' (без ошибок)
19     int 80h                                ; вызов ядра
20
```

Рис. 4.10: Изменение программы

Компилирую текст программы в объектный файл (рис. [4.11]). Проверяю с помощью утилиты `ls`, что файл `lab5.asm` создан.

```
dkhuddheva@dkhuddheva-VirtualBox: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ gedit lab4.asm
dkhuddheva@dkhuddheva-VirtualBox: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ nasm -f elf lab4.asm
dkhuddheva@dkhuddheva-VirtualBox: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello hello.o lab4.asm lab4.o list.lst main obj.o presentation report
dkhuddheva@dkhuddheva-VirtualBox: /work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$
```

Рис. 4.11: Компиляция текста программы

Передаю объектный файл `lab4.o` на обработку компоновщику `LD`, чтобы получить исполняемый файл `lab4` (рис. [4.12]).

```
hello hello.asm hello.o lab4.asm lab4.o list.lst main obj.o presentation report
dkhuddihevagdkhuddiheva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ld -n elf_i386 lab4.o -o lab4
dkhuddihevagdkhuddiheva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o presentation report
dkhuddihevagdkhuddiheva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$
```

Рис. 4.12: Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4, на экран действительно выводятся моё имя и фамилия (рис. [4.13]).

```
dkhuddihevagdkhuddiheva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ ./lab4
Jennet Huddyyewa
dkhuddihevagdkhuddiheva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$
```

Рис. 4.13: Запуск исполняемого файла

С помощью команд git add .и git commit добавляю файлы на GitHub (рис. [??]).

```
dkhuddihevagdkhuddiheva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ git add .
dkhuddihevagdkhuddiheva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$ git commit -m "Add fales for la
b04"
[master 80ba832] Add fales for lab04
2 files changed, 38 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
dkhuddihevagdkhuddiheva-VirtualBox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04$
```

Отправляю файлы на сервер с помощью команды git push.

## **5 Выводы**

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