

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук
Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

Дисциплина: **Архитектура компьютеров**

Студент: Болдырева Дельгир

Группа: НКАбд-01-25

Москва

2025 г.

Оглавление

| | |
|---|----|
| 1 Цель работы..... | 3 |
| 2 Задание..... | 4 |
| 3 Теоретическое введение..... | 5 |
| 4 Выполнение лабораторной работы..... | 7 |
| 4.1 Программа Hello world! | 7 |
| 4.2 Транслятор NASM..... | 8 |
| 4.3 Расширенный синтаксис командной строки..... | 8 |
| 4.4 компоновщик LD..... | 9 |
| 4.5 Задания для самостоятельной работы..... | 9 |
| 5 Выводы..... | 11 |
| 6 Список литературы..... | 12 |

1 Цель работы

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

2 Задания

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!

Я создаю каталог в домашней директории, в котором буду хранить файлы для текущей лабораторной работы.

```
deboldihreva1@dk3n06 ~ $ mkdir -p ~/work/arch-pc/lab04
deboldihreva1@dk3n06 ~ $ cd ~/work/
deboldihreva1@dk3n06 ~/work $ cd ~/work/arch-pc/lab04/
deboldihreva1@dk3n06 ~/work/arch-pc/lab04 $
```

Рис. 4.1.1 Создание рабочей директории.

После создаю в нем файл hello.asm, в котором буду писать программу на языке ассемблера (рис. 4.1.2)

```
bash: touch: команда не найдена
deboldihreva1@dk3n06 ~/work/arch-pc/lab04 $ touch hello.asm
deboldihreva1@dk3n06 ~/work/arch-pc/lab04 $ gedit hello.asm
█
```

Рис. 4.1.2 Создания файла asm

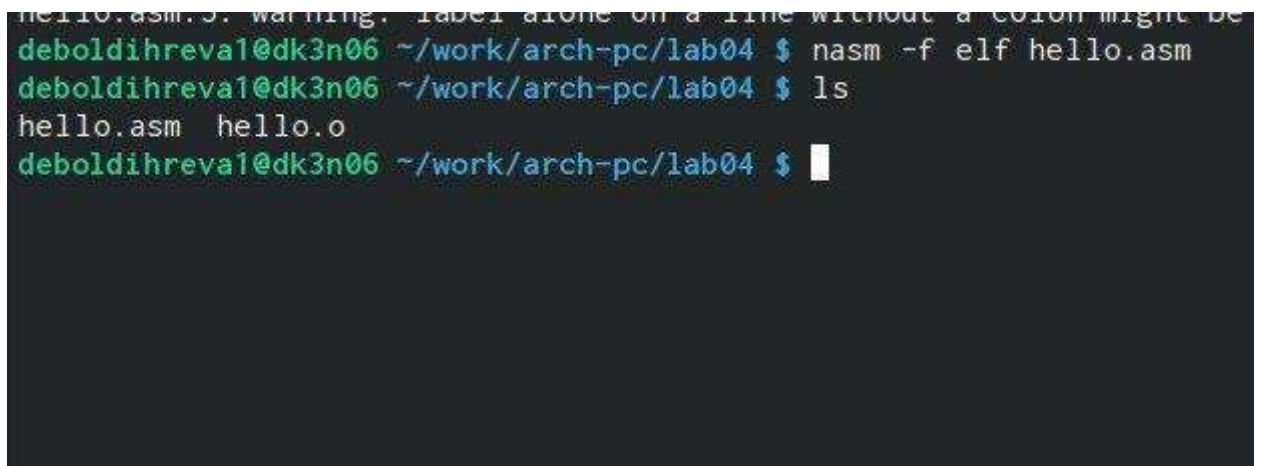
С помощью редактора пишу программу в созданном файле. (рис. 4.1.3)



```
1 SECTION .data
2     hello: db "Hello, world!",0xa
3           helloLen: equ $ - hello
4 SECTION .text
5     global _start
6
7 _start:
8     mov eax,4
9     mov ebx,1
10    mov ecx,hello
11    mov edx,helloLen
12    int 0x80
13
14    mov eax,1
15    mov ebx,0
16    int 0x80
```

4.2 Транслятор NASM.

Компилирую с помощью NASM свою программу.(рис. 4.2.1)



```
hello.asm.3: warning: label alone on a line without a colon might be
deboldihreva1@dk3n06 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
deboldihreva1@dk3n06 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o
deboldihreva1@dk3n06 ~/work/arch-pc/lab04 $
```

Рис. 4.2.1 Компиляция программы

4.3 Расширенный синтаксис командной строки

Выполняю команду, указанную на (рис. 4.3.1), 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.


```

deboldihreval@dk3n06 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
deboldihreval@dk3n06 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o
deboldihreval@dk3n06 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
deboldihreval@dk3n06 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o  list.lst  obj.o
deboldihreval@dk3n06 ~/work/arch-pc/lab04 $

```

Рис. 4.3.1 Возможности синтаксиса NASM

4.4 Компоновщик LD

Затем мне необходимо передать объектный файл компоновщику, делаю это с помощью команды ld. (рис. 4.4.1)

```

deboldihreval@dk3n06 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
deboldihreval@dk3n06 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o
deboldihreval@dk3n06 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
deboldihreval@dk3n06 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o  list.lst  obj.o
deboldihreval@dk3n06 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
deboldihreval@dk3n06 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  obj.o
deboldihreval@dk3n06 ~/work/arch-pc/lab04 $

```

Рис. 4.4.1 Отправка файла компоновщику

```

deboldihreval@dk3n06 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o
deboldihreval@dk3n06 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
deboldihreval@dk3n06 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o  list.lst  obj.o
deboldihreval@dk3n06 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
deboldihreval@dk3n06 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  obj.o
deboldihreval@dk3n06 ~/work/arch-pc/lab04 $ ./hello
Hello world!
deboldihreval@dk3n06 ~/work/arch-pc/lab04 $

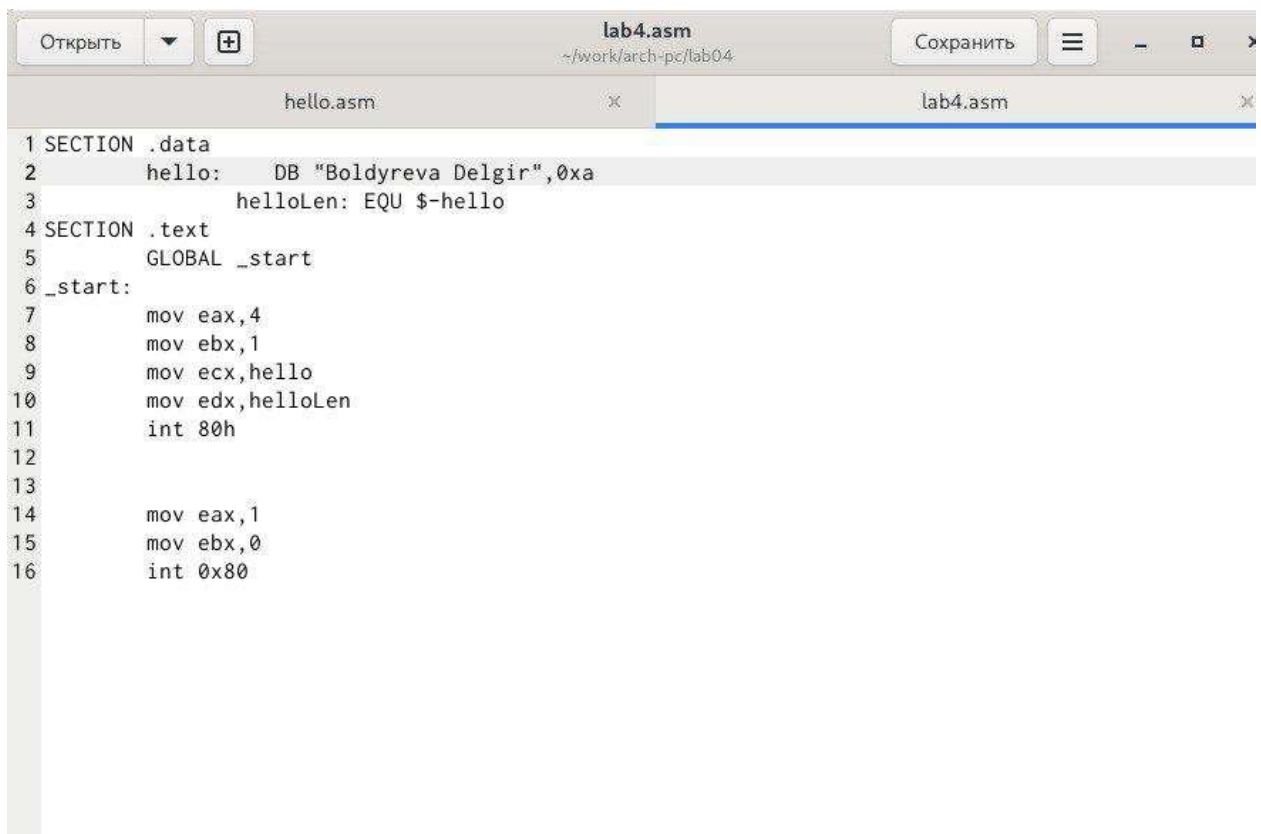
```

Рис. 4.4.2 Создание исполняемого файла и запуск программы

4.5 Задания для самостоятельной работы

```
hello hello.asm hello.o list.lst obj.o
deboldihreva1@dk3n06 ~/work/arch-pc/lab04 $ ./hello
Hello world!
deboldihreva1@dk3n06 ~/work/arch-pc/lab04 $ cp hello.asm lab4.asm
deboldihreva1@dk3n06 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o lab4.asm list.lst obj.o
deboldihreva1@dk3n06 ~/work/arch-pc/lab04 $
```

Рис. 4.5.1 Создание копии файла для последующей работы с ней



```
lab4.asm
~/work/arch-pc/lab04
Сохранить
hello.asm lab4.asm
1 SECTION .data
2     hello:    DB "Boldyreva Delgir",0xa
3     helloLen: EQU $-hello
4 SECTION .text
5     GLOBAL _start
6 _start:
7     mov eax,4
8     mov ebx,1
9     mov ecx,hello
10    mov edx,helloLen
11    int 80h
12
13
14    mov eax,1
15    mov ebx,0
16    int 0x80
```

Рис. 4.5.2 Редактирую копию файла, заменив текст на свое имя и фамилию

```
deboldihreva1@dk3n06 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm -o lab4.o
deboldihreva1@dk3n06 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o lab4
deboldihreva1@dk3n06 ~/work/arch-pc/lab04 $ ./lab4
Boldyreva Delgir
deboldihreva1@dk3n06 ~/work/arch-pc/lab04 $
```

Рис. 4.5.3 Проверка работает ли программа

5 Вывод

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

6 Список литературы

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