# Отчет по лабораторной работе №4

## Архитектура компьютеров

Маныев Ресулбег НКАбд-03-23

# Содержание

### [Цель работы](#_bookmark0) 5

1. [**Выполнение лабораторной работы**](#_bookmark1) **6**

[2.0.1 1](#_bookmark2) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

[2.0.2 2](#_bookmark4) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

[2.0.3 3](#_bookmark6) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

[2.0.4 4](#_bookmark8) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

[2.0.5 5](#_bookmark11) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

[2.0.6 6](#_bookmark13) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

[2.0.7 7](#_bookmark15) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

[2.0.8 8](#_bookmark17) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

[2.0.9 9](#_bookmark19) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

### [Самостоятельная работа](#_bookmark21) 10

[3.0.1 1](#_bookmark22) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

[3.0.2 2](#_bookmark24) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

[3.0.3 3](#_bookmark26) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

[3.0.4 4](#_bookmark28) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

### [Ответы на вопросы](#_bookmark32) 13

1. [**Выводы**](#_bookmark33) **16**

# Список иллюстраций

* 1. [Создание каталога с помощью команд mkdir -p ~/work/arch-pc/lab04](#_bookmark3) 6
  2. [Переход в созданный каталог с помощью команд cd ~/work/arch-](#_bookmark5) [pc/lab04](#_bookmark5) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
  3. [Создание текстового файла с помощью команд touch hello.asm](#_bookmark7) . . 7
  4. [Открытие текстового редактора gedit с помощью команды gedit](#_bookmark9) [hello.asm](#_bookmark9) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
  5. [И ввожу в него следующий текст.](#_bookmark10) . . . . . . . . . . . . . . . . . . 7
  6. [Ввожу команду nasm -f elf hello.asm](#_bookmark12) . . . . . . . . . . . . . . . . . 8
  7. [Расширенный синтаксис командной строки NASM.](#_bookmark14) . . . . . . . . 8 [2.8 Компоновщик LD.](#_bookmark16) . . . . . . . . . . . . . . . . . . . . . . . . . . 8

[2.9 Ввожу команду ld -m elf\_i386 obj.o -o main](#_bookmark18) . . . . . . . . . . . . . 9 [2.10 Ввожу команду ./hello](#_bookmark20) . . . . . . . . . . . . . . . . . . . . . . . . 9

* 1. [Создаю копию файла hello.asm с именем lab04.asm](#_bookmark23) 10
  2. [Ввожу свое имя фамилию.](#_bookmark25) 10
  3. [Запускаю получившийся исполняемый файл.](#_bookmark27) 11
  4. [Копирую файлы hello.asm и lab4.asm с помошью команды](#_bookmark29) [cp hello.asm lab04.asm ~/work/study/2023-2024/“Архитектура](#_bookmark29)

[компьютера”/arch-pc/labs/lab04/](#_bookmark29) 11

* 1. [Проверяю.](#_bookmark30) 12
  2. [Загржаю файлы на Github.](#_bookmark31) 12

# Список таблиц

1. **Цель работы**

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

# Выполнение лабораторной работы

## 2.0.1 1

Создаю каталог для работы с программами на языке ассемблера NASM.



Рис. 2.1: Создание каталога с помощью команд mkdir -p ~/work/arch-pc/lab04

## 2.0.2 2

Перехожу в созданный каталог.



Рис. 2.2: Переход в созданный каталог с помощью команд cd ~/work/arch-pc/lab04

## 2.0.3 3

Создаю текстовый файл с именем hello.asm



Рис. 2.3: Создание текстового файла с помощью команд touch hello.asm

## 2.0.4 4

Открываю этот файл с помощью текстового редактора gedit.

Рис. 2.4: Открытие текстового редактора gedit с помощью команды gedit hello.asm

C:\Users\1\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2023-11-06_00-51-37.png

Рис. 2.4

## 2.0.5 5

NASM превращает текст программы в объектный код.



Рис. 2.5: Ввожу команду nasm -f elf hello.asm

## 2.0.6 6

Полный вариант командной строки nasm выглядит следующим образом:



Рис. 2.6: Расширенный синтаксис командной строки NASM.

## 2.0.7 7

Чтобы получить исполняемую программу, объектный файл необходимо пере- дать на обработку компоновщику:



Рис. 2.7: Компоновщик LD.

## 2.0.8 8

Ключ -o с последующим значением задаёт в данном случае имя создаваемого исполняемого файла.



Рис. 2.8: Ввожу команду ld -m elf\_i386 obj.o -o main

## 2.0.9 9

Запуск исполняемого файла.



Рис. 2.9: Ввожу команду ./hello

# Самостоятельная работа

## 3.0.1 1

В каталоге ~/work/arch-pc/lab04 с помощью команды cp



Рис. 3.0: Создаю копию файла hello.asm с именем lab4.asm

## 3.0.2 2

С помощью текстового редактора gedit ввожу изменения в тексте программы в файле lab04.asm вместо Hello world! ввожу Маныев Ресул.

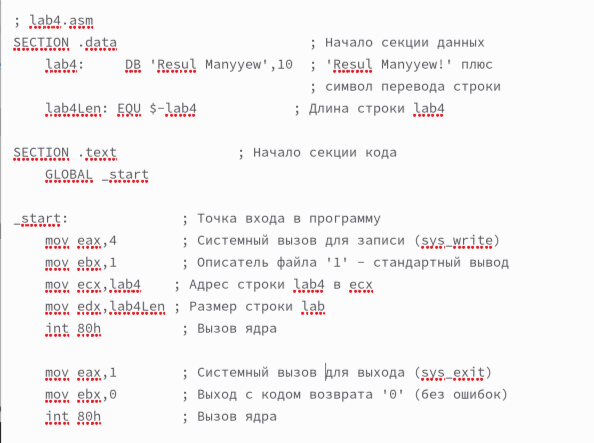


Рис. 3.1: Ввожу свое имя фамилию.

## 3.0.3 3

Оттранслирую полученный текст программы lab04.asm в объектный файл.

Выполняю компоновку объектного файла.

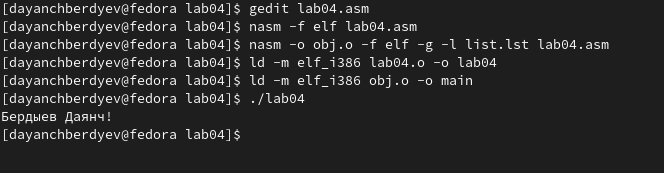


Рис. 3.2: Запускаю получившийся исполняемый файл.

## 3.0.4 4

Копирую файлы hello.asm и lab04.asm в локальный репозиторий в каталог

~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/.



Рис. 3.3: Копирую файлы hello.asm и lab4.asm с помошью команды cp hello.asm lab04.asm ~/work/study/2023-2024/“Архитектура компьюте- ра”/arch-pc/labs/lab04/

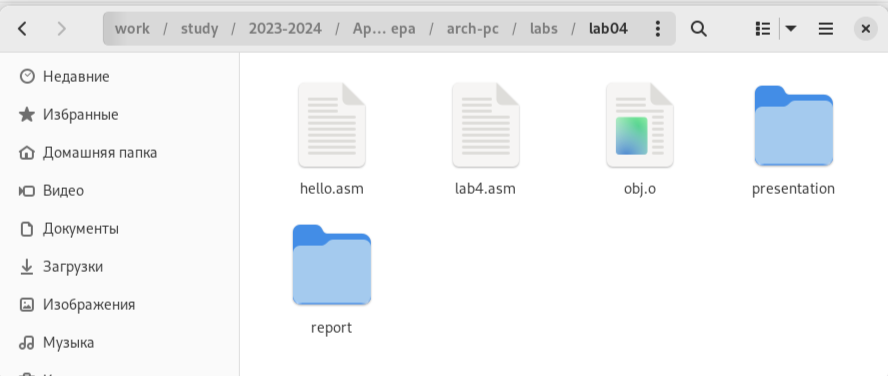


Рис. 3.4: Проверяю.

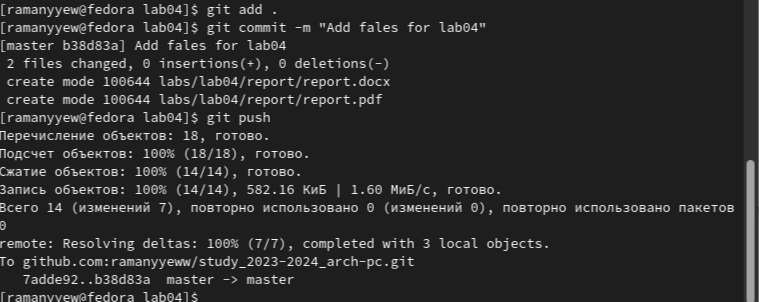


Рис. 3.5: Загржаю файлы на Github.

# Ответы на вопросы

* 1. Основное отличие ассемблера от языков высокого уровня — Байт-код или байтко́д (англ. byte-code), иногда также используется термин псевдоко́д — машинно-независимый код низкого уровня, генерируемый транслятором и исполняемый интерпретатором. Большинство инструкций байт-кода эк- вивалентны одной или нескольким командам ассемблера. Трансляция в байт-код занимает промежуточное положение между компиляцией в ма- шинный код и интерпретацией.
  2. Инструкция ассемблера генерирует машинный код, таким образом, спо- собствует размеру программы. Директива ассемблера не создает какого

-либо машинного кода, таким образом, не способствует размеру програм- мы. IT приказывает ассемблеру выполнять определенные действия на этапе сборки.

* 1. Правила написания программ на языке assembler Исходный текст програм- мы на языке ассемблера имеет определенный формат. Каждая команда и директива представляет собой строку: Метка, операция,операнд(ы), ком- ментарии.
  2. Создание исполняемого файла издавна производилось в три этапа: (1) об- работка исходного кода препроцессором, (2) компиляция в объектный код и (3) компоновка объектных модулей, включая модули из объектных биб- лиотек, в исполняемый файл. Это классическая схема для компилируемых языков.
  3. На этапе трансляции осуществляется перевод команд ассемблера в соот- ветствующие машинные команды. В результате трансляции формируются файл объектного модуля и файл листинга.
  4. Если в процессе ассемблирования не было выявлено ошибок в ассемблерном листинге, то программа-ассемблер создаcт объектный файл (с расширением OBJ).

Затем необходимо воспользоваться компоновщиком (линковщиком), кото- рый входит в комплект программы-ассемблера. Данная процедура выполняется гораздо быстрее ассемблирования.

Именно компоновщик создает готовый к запуску файл (программу) с расши- рением COM или EXE из объектного файла (OBJ). Оба типа имеют отличия в структуре ассемблерной программы. Первый тип (COM) не может превышать 64 Кбайт и используется только в MS-DOS (и для совместимости поддерживается в Windows), однако он очень компактный и удобный для написания небольших программ и резидентов. В большинстве случаев, если программа написана на чистом ассемблере под MS-DOS, нет необходимости создавать EXE-файлы. В этой книге в части I рассматриваются именно программы типа COM.

В отличие от создания программ типа COM, при создании стандартных EXE- программ под MS-DOS нет необходимости указывать какие-либо параметры линковщику при компоновке. Дело в том, что компоновщик не может автомати- чески определить, какой тип подвергается компоновке.

Линковщик также проверяет, нет ли каких-либо ошибок в объектном файле, но не грамматических, а логических. Например, отсутствие необходимой объектной библиотеки, указанной в самом файле либо в командной строке (программаас- семблер этого не делает).

Если ошибки не были обнаружены, компоновщик создает машинный код (про- грамму типа COM или EXE), которую можно запускать на выполнение.

* 1. Для того чтобы выполнить пробный прогон ассемблерной программы, ее

необходимо сначала оттранслировать и скомпоновать. Пусть текст исходной программы хранится в файле с именем SIMPLE.ASM. Трансляцию можно осуществить вызовом турбо ассемблера TASM.EXE с помощью, например, следующей команды DOS:

tasm /l/z/zi/n simple.asm

* 1. NASM поддерживает множество форматов выходных файлов, среди них: bin — файл произвольного формата, определяемого только исходным кодом.

Пригоден как для файлов данных, так и для модулей с исполняемыми кодами — например, системных загрузчиков, образов ПЗУ, модулей операционных систем, драйверов .SYS в MS-DOS или исполняемых файлов .COM. obj — объектный мо- дуль в формате OMF, совместимый с MASM и TASM. win32 и win64 — объектный модульдля 32- и 64-битного кода, совместимый с Win32- и Win64-компиляторами Microsoft. aout — объектный модуль в варианте формата a.out, использовавшего- ся в ранних Linux-системах. aoutb — версия формата a.out для BSD-совместимых операционных систем. coff — объектный модуль в формате COFF, совместимом с компоновщиком из DJGPP. elf32 и elf64 — объектный модуль в форматах ELF32 и ELF64, используемых в Linux и Unix System V, включая Solaris x86, UnixWare и SCO Unix. Формат выходного файла можно задать с помощью ключа командной стро- ки -f. Форматы могут расширять синтаксис некоторых инструкций и добавлять собственные инструкции.

# Выводы

В ходе выполнения этой лабораторной работы я освоил процедуру компиляции и сборки программ, написанных на ассемблере NASM.