

Отчет по лабораторной работе №5

Архитектура компьютера

Рогожина Надежда Александровна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
3.1	Основные принципы работы компьютера	7
3.2	Ассемблер и язык ассемблера	9
3.3	Процесс создания и обработки программы на языке ассемблера .	9
4	Выполнение лабораторной работы	11
5	Вопросы для самопроверки	21
6	Выводы	23

Список иллюстраций

4.1	Создадим и перейдем в каталог ~/work/arch-pc/lab05	11
4.2	Создадим текстовый файл с именем hello.asm	12
4.3	Откроем этот файл с помощью текстового редактора	12
4.4	Введем в него следующий текст:	13
4.5	Скомпилируем программу	13
4.6	Проверим компиляцию программы и создание файла с объектным кодом	14
4.7	Скомпилируем файл с присвоением имени, а также создадим файл листинга	14
4.8	Передадим объектный файл на обработку компоновщику	15
4.9	Проверим создание исполняемого файла hello	15
4.10	Передадим файл компоновщику	16
4.11	Запустим исполняемый файл hello	16
4.12	Создадим копию файла hello.asm с именем lab5.asm	17
4.13	С помощью текстового редактора внесем изменения в текст программы	17
4.14	Трансируем lab5.asm в объектный файл	18
4.15	Трансируем lab5.asm в объектный файл	18
4.16	Запустим исполняемый файл lab5	19
4.17	Скопируем файлы hello.asm и lab5.asm в локальный репозиторий	19
4.18	Загрузим файлы на Github	20

Список таблиц

1 Цель работы

Целью данной работы является изучение языка Assembler, освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

2 Задание

1. В каталоге `~/work/arch-рс/lab05` с помощью команды `ср` создайте копию файла `hello.asm` с именем `lab5.asm`
2. С помощью любого текстового редактора внесите изменения в текст программы в файле `lab5.asm` так, чтобы вместо `Hello world!` на экран выводилась строка с вашими фамилией и именем.
3. Оттранслируйте полученный текст программы `lab5.asm` в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл.
4. Скопируйте файлы `hello.asm` и `lab5.asm` в Ваш локальный репозиторий в каталог `~/work/study/2022-2023/“Архитектура компьютера”/arch-рс/labs/lab05/`. Загрузите файлы на Github.

3 Теоретическое введение

3.1 Основные принципы работы компьютера

Основными функциональными элементами любой электронно-вычислительной машины (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 (системной) плате.

Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора (ЦП) 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*.

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.

В состав ЭВМ также входят периферийные устройства, которые можно разделить на:

- **устройства внешней памяти**, которые предназначены для долговременного хранения больших объёмов данных (жёсткие диски, твердотельные накопители, магнитные ленты);
- **устройства ввода-вывода**, которые обеспечивают взаимодействие ЦП с внешней средой.

При выполнении каждой команды процессор выполняет определённую последовательность стандартных действий, которая называется командным циклом процессора. В самом общем виде 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.

Данный алгоритм позволяет выполнить хранящуюся в ОЗУ программу. Кроме того, в зависимости от команды при её выполнении могут проходить не все этапы.

3.2 Ассемблер и язык ассемблера

Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. Можно считать, что он больше любых других языков приближен к архитектуре ЭВМ и её аппаратным возможностям, что позволяет получить к ним более полный доступ, нежели в языках высокого уровня, таких как C/C++, Perl, Python и пр. Заметим, что получить полный доступ к ресурсам компьютера в современных архитектурах нельзя, самым низким уровнем работы прикладной программы является обращение напрямую к ядру операционной системы. Именно на этом уровне и работают программы, написанные на ассемблере. Но в отличие от языков высокого уровня ассемблерная программа содержит только тот код, который ввёл программист. Таким образом язык ассемблера — это язык, с помощью которого понятным для человека образом пишутся команды для процессора.

Следует отметить, что процессор понимает не команды ассемблера, а последовательности из нулей и единиц — **машинные коды**. До появления языков ассемблера программистам приходилось писать программы, используя только лишь машинные коды, которые были крайне сложны для запоминания, так как представляли собой числа, записанные в двоичной или шестнадцатеричной системе счисления. Преобразование или трансляция команд с языка ассемблера в исполняемый машинный код осуществляется специальной программой транслятором — **Ассемблер**.

3.3 Процесс создания и обработки программы на языке ассемблера

В процессе создания ассемблерной программы можно выделить четыре шага:

- **Набор текста** программы в текстовом редакторе и сохранение её в отдельном файле. Каждый файл имеет свой тип (или расширение), который определяет

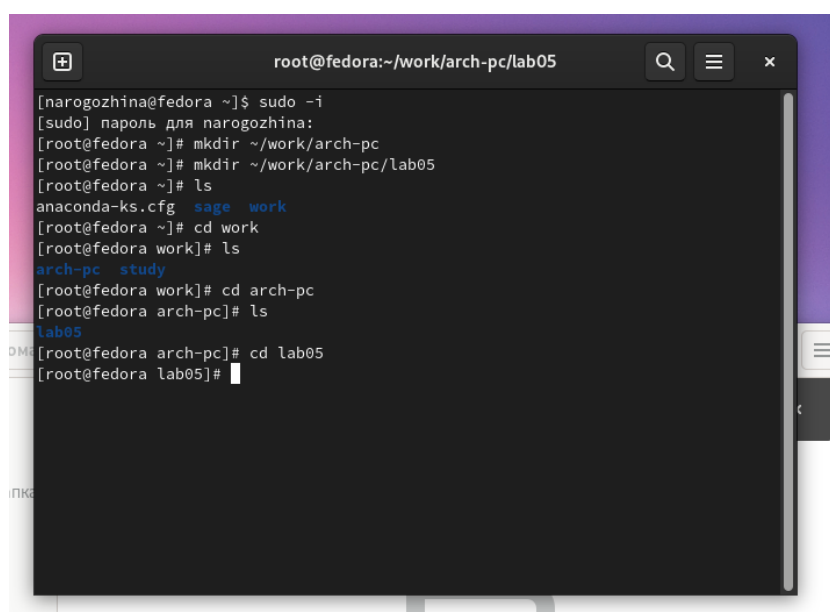
назначение файла. Файлы с исходным текстом программ на языке ассемблера имеют тип `asm`.

- **Трансляция** — преобразование с помощью транслятора, например `nasm`, текста программы в машинный код, называемый объектным. На данном этапе также может быть получен листинг программы, содержащий кроме текста программы различную дополнительную информацию, созданную транслятором. Тип объектного файла — `o`, файла листинга — `lst`.

- **Компоновка или линковка** — этап обработки объектного кода компоновщиком (`ld`), который принимает на вход объектные файлы и собирает по ним исполняемый файл. Исполняемый файл обычно не имеет расширения. Кроме того, можно получить файл карты загрузки программы в ОЗУ, имеющий расширение `map`.

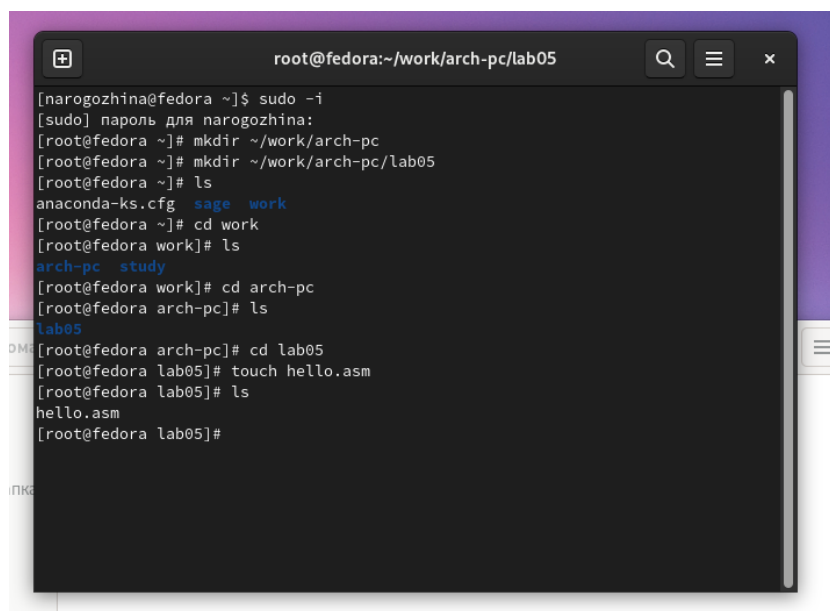
- **Запуск программы.** Конечной целью является работоспособный исполняемый файл. Ошибки на предыдущих этапах могут привести к некорректной работе программы, поэтому может присутствовать этап отладки программы при помощи специальной программы — отладчика. При нахождении ошибки необходимо провести коррекцию программы, начиная с первого шага.

4 Выполнение лабораторной работы



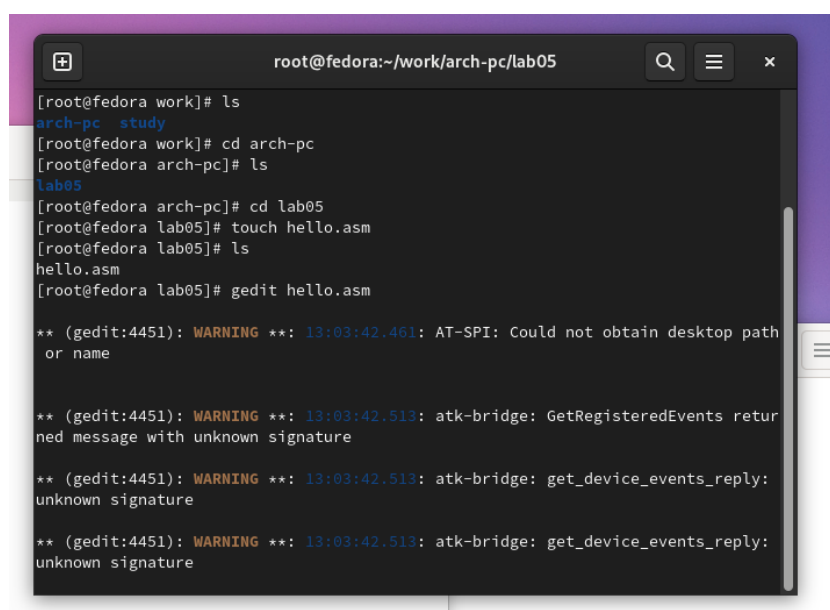
```
root@fedora:~/work/arch-pc/lab05
[narogozhina@fedora ~]$ sudo -i
[sudo] пароль для narogozhina:
[root@fedora ~]# mkdir ~/work/arch-pc
[root@fedora ~]# mkdir ~/work/arch-pc/lab05
[root@fedora ~]# ls
anaconda-ks.cfg  sage  work
[root@fedora ~]# cd work
[root@fedora work]# ls
arch-pc  study
[root@fedora work]# cd arch-pc
[root@fedora arch-pc]# ls
lab05
[root@fedora arch-pc]# cd lab05
[root@fedora lab05]#
```

Рис. 4.1: Создадим и перейдем в каталог ~/work/arch-pc/lab05



```
root@fedora:~/work/arch-pc/lab05
[narogozhina@fedora ~]$ sudo -i
[sudo] пароль для narogozhina:
[root@fedora ~]# mkdir ~/work/arch-pc
[root@fedora ~]# mkdir ~/work/arch-pc/lab05
[root@fedora ~]# ls
anaconda-ks.cfg  sage  work
[root@fedora ~]# cd work
[root@fedora work]# ls
arch-pc  study
[root@fedora work]# cd arch-pc
[root@fedora arch-pc]# ls
lab05
[root@fedora arch-pc]# cd lab05
[root@fedora lab05]# touch hello.asm
[root@fedora lab05]# ls
hello.asm
[root@fedora lab05]#
```

Рис. 4.2: Создадим текстовый файл с именем hello.asm



```
root@fedora:~/work/arch-pc/lab05
[root@fedora work]# ls
arch-pc  study
[root@fedora work]# cd arch-pc
[root@fedora arch-pc]# ls
lab05
[root@fedora arch-pc]# cd lab05
[root@fedora lab05]# touch hello.asm
[root@fedora lab05]# ls
hello.asm
[root@fedora lab05]# gedit hello.asm

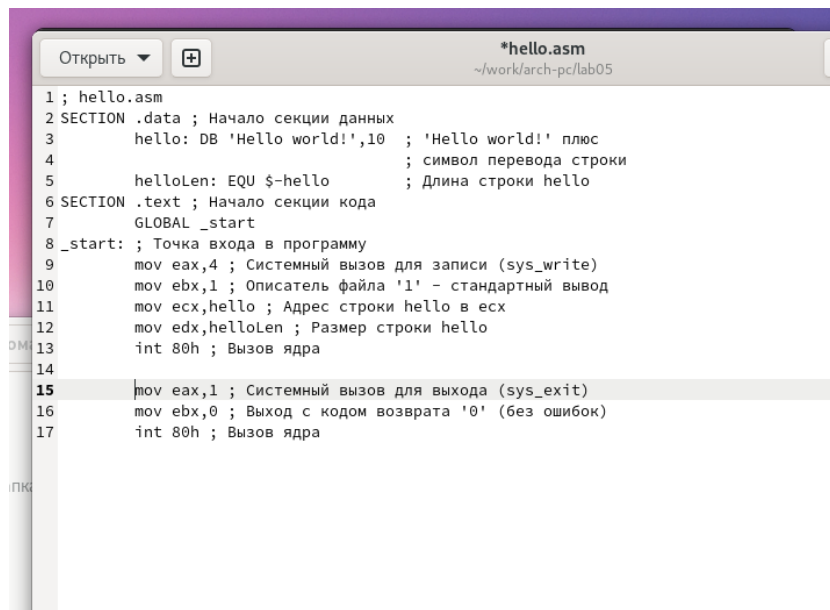
** (gedit:4451): WARNING **: 13:03:42.461: AT-SPI: Could not obtain desktop path
or name

** (gedit:4451): WARNING **: 13:03:42.513: atk-bridge: GetRegisteredEvents retur
ned message with unknown signature

** (gedit:4451): WARNING **: 13:03:42.513: atk-bridge: get_device_events_reply:
unknown signature

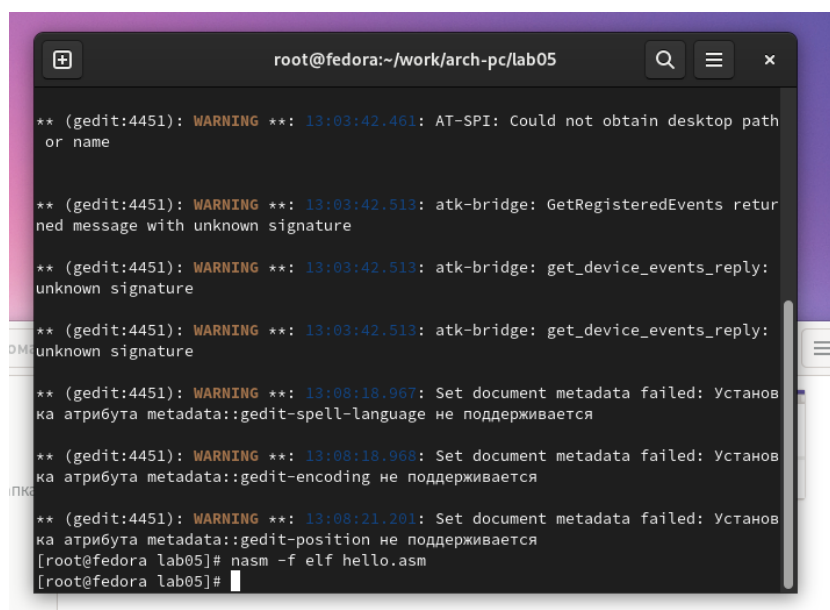
** (gedit:4451): WARNING **: 13:03:42.513: atk-bridge: get_device_events_reply:
unknown signature
```

Рис. 4.3: Откроем этот файл с помощью текстового редактора



```
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
15    mov eax,1 ; Системный вызов для выхода (sys_exit)
16    mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
17    int 80h ; Вызов ядра
```

Рис. 4.4: Введем в него следующий текст:



```
root@fedora:~/work/arch-pc/lab05

** (gedit:4451): WARNING **: 13:03:42.461: AT-SPI: Could not obtain desktop path
or name

** (gedit:4451): WARNING **: 13:03:42.513: atk-bridge: GetRegisteredEvents retur
ned message with unknown signature

** (gedit:4451): WARNING **: 13:03:42.513: atk-bridge: get_device_events_reply:
unknown signature

** (gedit:4451): WARNING **: 13:03:42.513: atk-bridge: get_device_events_reply:
unknown signature

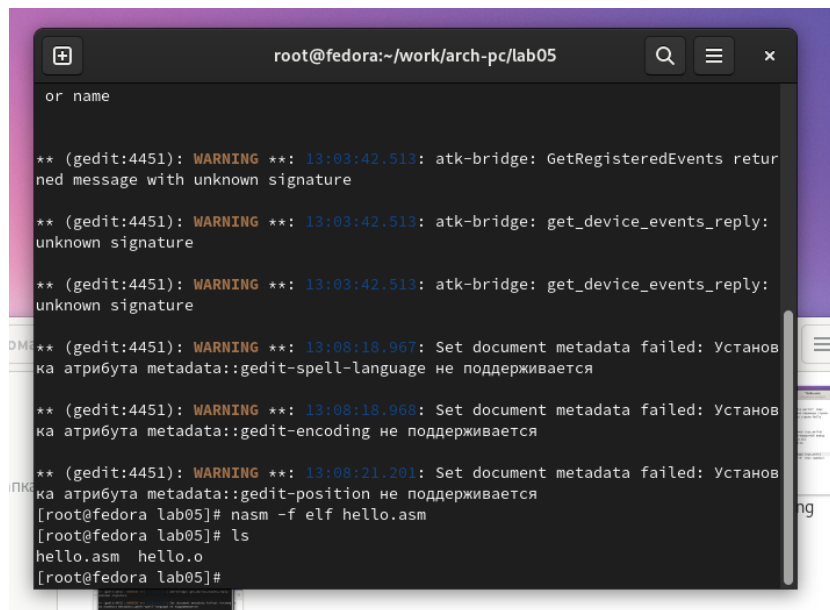
** (gedit:4451): WARNING **: 13:08:18.967: Set document metadata failed: Установ
ка атрибута metadata::gedit-spell-language не поддерживается

** (gedit:4451): WARNING **: 13:08:18.968: Set document metadata failed: Установ
ка атрибута metadata::gedit-encoding не поддерживается

** (gedit:4451): WARNING **: 13:08:21.201: Set document metadata failed: Установ
ка атрибута metadata::gedit-position не поддерживается

[root@fedora lab05]# nasm -f elf hello.asm
[root@fedora lab05]#
```

Рис. 4.5: Скомпилируем программу



```
root@fedora:~/work/arch-pc/lab05

or name

** (gedit:4451): WARNING **: 13:03:42.513: atk-bridge: GetRegisteredEvents returned message with unknown signature

** (gedit:4451): WARNING **: 13:03:42.513: atk-bridge: get_device_events_reply: unknown signature

** (gedit:4451): WARNING **: 13:03:42.513: atk-bridge: get_device_events_reply: unknown signature

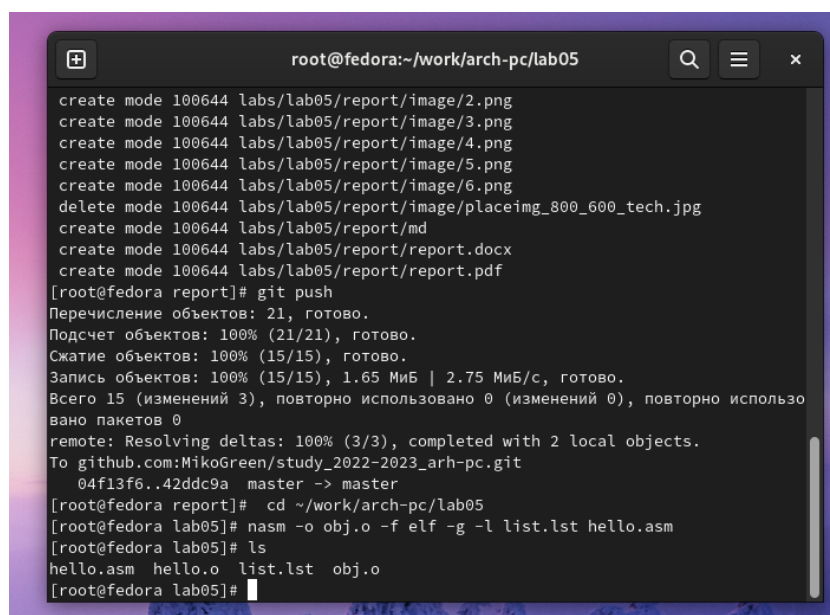
** (gedit:4451): WARNING **: 13:08:18.967: Set document metadata failed: Установка атрибута metadata::gedit-spell-language не поддерживается

** (gedit:4451): WARNING **: 13:08:18.968: Set document metadata failed: Установка атрибута metadata::gedit-encoding не поддерживается

** (gedit:4451): WARNING **: 13:08:21.201: Set document metadata failed: Установка атрибута metadata::gedit-position не поддерживается

[root@fedora lab05]# nasm -f elf hello.asm
[root@fedora lab05]# ls
hello.asm  hello.o
[root@fedora lab05]#
```

Рис. 4.6: Проверим компиляцию программы и создание файла с объектным кодом



```
root@fedora:~/work/arch-pc/lab05

create mode 100644 labs/lab05/report/image/2.png
create mode 100644 labs/lab05/report/image/3.png
create mode 100644 labs/lab05/report/image/4.png
create mode 100644 labs/lab05/report/image/5.png
create mode 100644 labs/lab05/report/image/6.png
delete mode 100644 labs/lab05/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab05/report/md
create mode 100644 labs/lab05/report/report.docx
create mode 100644 labs/lab05/report/report.pdf
[root@fedora report]# git push
Перечисление объектов: 21, готово.
Подсчет объектов: 100% (21/21), готово.
Сжатие объектов: 100% (15/15), готово.
Запись объектов: 100% (15/15), 1.65 Миб | 2.75 Миб/с, готово.
Всего 15 (изменений 3), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:MikoGreen/study_2022-2023_arh-pc.git
04f13f6..42ddc9a master -> master
[root@fedora report]# cd ~/work/arch-pc/lab05
[root@fedora lab05]# nasm -o obj.o -f elf -g -l list.lst hello.asm
[root@fedora lab05]# ls
hello.asm  hello.o  list.lst  obj.o
[root@fedora lab05]#
```

Рис. 4.7: Скомпилируем файл с присвоением имени, а также создадим файл листинга

```
root@fedora:~/work/arch-pc/lab05
create mode 100644 labs/lab05/report/image/3.png
create mode 100644 labs/lab05/report/image/4.png
create mode 100644 labs/lab05/report/image/5.png
create mode 100644 labs/lab05/report/image/6.png
delete mode 100644 labs/lab05/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab05/report/md
create mode 100644 labs/lab05/report/report.docx
create mode 100644 labs/lab05/report/report.pdf
[root@fedora report]# git push
Перечисление объектов: 21, готово.
Подсчет объектов: 100% (21/21), готово.
Сжатие объектов: 100% (15/15), готово.
Запись объектов: 100% (15/15), 1.65 МиБ | 2.75 МиБ/с, готово.
Всего 15 (изменений 3), повторно использовано 0 (изменений 0), повторно использо
вано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:MikoGreen/study_2022-2023_arh-pc.git
04f13f6..42ddc9a master -> master
[root@fedora report]# cd ~/work/arch-pc/lab05
[root@fedora lab05]# nasm -o obj.o -f elf -g -l list.lst hello.asm
[root@fedora lab05]# ls
hello.asm hello.o list.lst obj.o
[root@fedora lab05]# ld -m elf_i386 hello.o -o hello
[root@fedora lab05]#
```

Рис. 4.8: Передадим объектный файл на обработку компоновщику

```
root@fedora:~/work/arch-pc/lab05
create mode 100644 labs/lab05/report/image/5.png
create mode 100644 labs/lab05/report/image/6.png
delete mode 100644 labs/lab05/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab05/report/md
create mode 100644 labs/lab05/report/report.docx
create mode 100644 labs/lab05/report/report.pdf
[root@fedora report]# git push
Перечисление объектов: 21, готово.
Подсчет объектов: 100% (21/21), готово.
Сжатие объектов: 100% (15/15), готово.
Запись объектов: 100% (15/15), 1.65 МиБ | 2.75 МиБ/с, готово.
Всего 15 (изменений 3), повторно использовано 0 (изменений 0), повторно использо
вано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:MikoGreen/study_2022-2023_arh-pc.git
04f13f6..42ddc9a master -> master
[root@fedora report]# cd ~/work/arch-pc/lab05
[root@fedora lab05]# nasm -o obj.o -f elf -g -l list.lst hello.asm
[root@fedora lab05]# ls
hello.asm hello.o list.lst obj.o
[root@fedora lab05]# ld -m elf_i386 hello.o -o hello
[root@fedora lab05]# ls
hello hello.asm hello.o list.lst obj.o
[root@fedora lab05]#
```

Рис. 4.9: Проверим создание исполняемого файла hello

```
root@fedora:~/work/arch-pc/lab05
create mode 100644 labs/lab05/report/md
create mode 100644 labs/lab05/report/report.docx
create mode 100644 labs/lab05/report/report.pdf
[root@fedora report]# git push
Перечисление объектов: 21, готово.
Подсчет объектов: 100% (21/21), готово.
Сжатие объектов: 100% (15/15), готово.
Запись объектов: 100% (15/15), 1.65 МиБ | 2.75 МиБ/с, готово.
Всего 15 (изменений 3), повторно использовано 0 (изменений 0), повторно использо
вано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:MikoGreen/study_2022-2023_arh-pc.git
   04f13f6..42ddc9a  master -> master
[root@fedora report]# cd ~/work/arch-pc/lab05
[root@fedora lab05]# nasm -o obj.o -f elf -g -l list.lst hello.asm
[root@fedora lab05]# ls
hello.asm  hello.o  list.lst  obj.o
[root@fedora lab05]# ld -m elf_i386 hello.o -o hello
[root@fedora lab05]# ls
hello  hello.asm  hello.o  list.lst  obj.o
[root@fedora lab05]# ld -m elf_i386 obj.o -o main
[root@fedora lab05]# ls
hello  hello.asm  hello.o  list.lst  main  obj.o
[root@fedora lab05]#
```

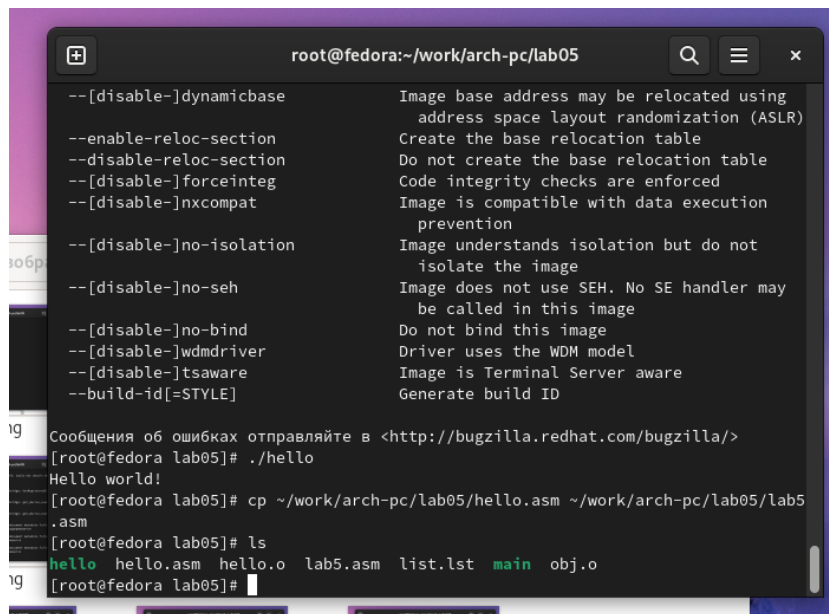
Рис. 4.10: Передадим файл компоновщику

В результате исполняемый файл имеет имя *main*, а объектный файл, из которого собран этот исполняемый файл имеет имя *obj.o*.

```
root@fedora:~/work/arch-pc/lab05
--enable-long-section-names    использовать длинные имена разделов COFF
                               даже в файлах исполняемых образов
--disable-long-section-names   не использовать длинные имена разделов
                               COFF даже в объектных файлах
--[disable-]dynamicbase        Image base address may be relocated using
                               address space layout randomization (ASLR)
--enable-reloc-section         Create the base relocation table
--disable-reloc-section        Do not create the base relocation table
--[disable-]forceinteg         Code integrity checks are enforced
--[disable-]nxcompat           Image is compatible with data execution
                               prevention
--[disable-]no-isolation       Image understands isolation but do not
                               isolate the image
--[disable-]no-seh             Image does not use SEH. No SE handler may
                               be called in this image
--[disable-]no-bind            Do not bind this image
--[disable-]wdmdriver          Driver uses the WDM model
--[disable-]tsaware            Image is Terminal Server aware
--build-id[=STYLE]            Generate build ID

Сообщения об ошибках отправляйте в <http://bugzilla.redhat.com/bugzilla/>
[root@fedora lab05]# ./hello
Hello world!
[root@fedora lab05]#
```

Рис. 4.11: Запустим исполняемый файл hello

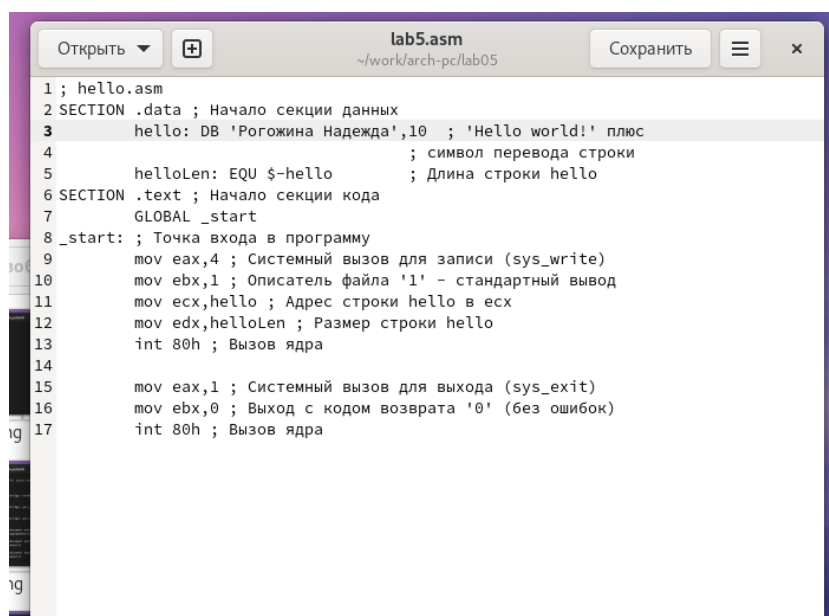


A terminal window titled 'root@fedora:~/work/arch-pc/lab05'. It displays various linker options and their descriptions, such as '--[disable-]dynamicbase' (Image base address may be relocated using address space layout randomization (ASLR)) and '--enable-reloc-section' (Create the base relocation table). Below the options, a message states: 'Сообщения об ошибках отправляйте в <http://bugzilla.redhat.com/bugzilla/>'. The user then runs the command '[root@fedora lab05]# ./hello', which outputs 'Hello world!'. Next, the user runs '[root@fedora lab05]# cp ~/work/arch-pc/lab05/hello.asm ~/work/arch-pc/lab05/lab5.asm'. Finally, the user runs '[root@fedora lab05]# ls', which lists the files: 'hello hello.asm hello.o lab5.asm list.lst main obj.o'.

```
root@fedora:~/work/arch-pc/lab05
--[disable-]dynamicbase      Image base address may be relocated using
                             address space layout randomization (ASLR)
--enable-reloc-section       Create the base relocation table
--disable-reloc-section      Do not create the base relocation table
--[disable-]forceinteg       Code integrity checks are enforced
--[disable-]nxcompat         Image is compatible with data execution
                             prevention
--[disable-]no-isolation     Image understands isolation but do not
                             isolate the image
--[disable-]no-seh           Image does not use SEH. No SE handler may
                             be called in this image
--[disable-]no-bind          Do not bind this image
--[disable-]wdmdriver         Driver uses the WDM model
--[disable-]tsaware          Image is Terminal Server aware
--build-id[=STYLE]          Generate build ID

Сообщения об ошибках отправляйте в <http://bugzilla.redhat.com/bugzilla/>
[root@fedora lab05]# ./hello
Hello world!
[root@fedora lab05]# cp ~/work/arch-pc/lab05/hello.asm ~/work/arch-pc/lab05/lab5
.asm
[root@fedora lab05]# ls
hello hello.asm hello.o lab5.asm list.lst main obj.o
[root@fedora lab05]#
```

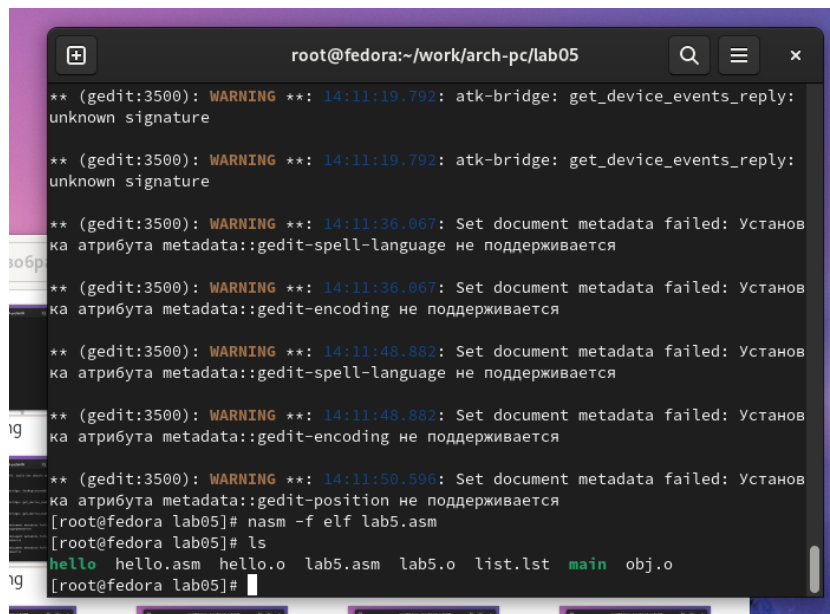
Рис. 4.12: Создадим компью файла hello.asm с именем lab5.asm



A text editor window titled 'lab5.asm' with the path '~/work/arch-pc/lab05'. The code is written in assembly and includes comments in Russian. It defines a data section with a string 'Hello world!' and a text section with instructions to write to stdout and exit. The code is as follows:

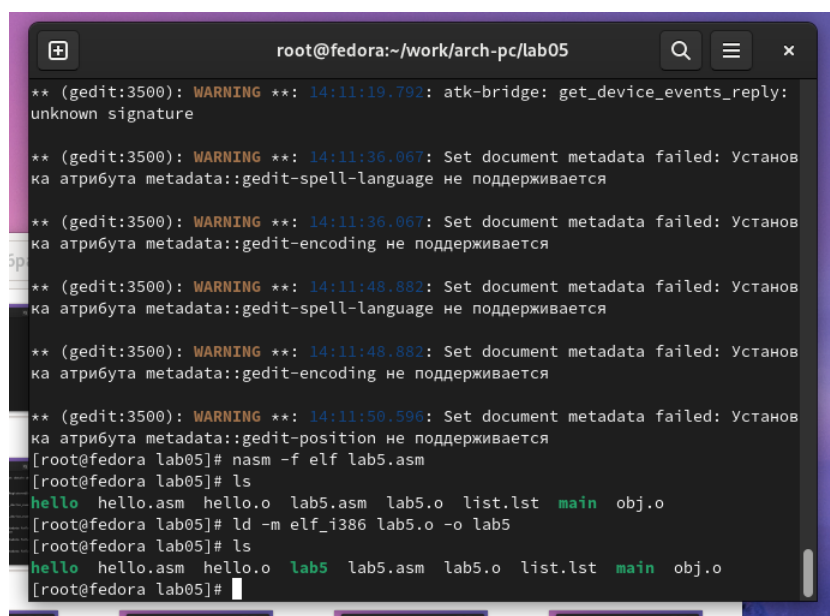
```
1 ; hello.asm
2 SECTION .data ; Начало секции данных
3     hello: DB 'Рогожина Надежда',10 ; 'Hello world!' плюс
4           ; символ перевода строки
5     helloLen: EQU $-hello ; Длина строки hello
6 SECTION .text ; Начало секции кода
7     GLOBAL _start
8 _start: ; Точка входа в программу
9   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

Рис. 4.13: С помощью текстового редактора внесем изменения в текст программы



```
root@fedora:~/work/arch-pc/lab05
** (gedit:3500): WARNING **: 14:11:19.792: atk-bridge: get_device_events_reply:
unknown signature
** (gedit:3500): WARNING **: 14:11:19.792: atk-bridge: get_device_events_reply:
unknown signature
** (gedit:3500): WARNING **: 14:11:36.067: Set document metadata failed: Установ
ка атрибута metadata::gedit-spell-language не поддерживается
** (gedit:3500): WARNING **: 14:11:36.067: Set document metadata failed: Установ
ка атрибута metadata::gedit-encoding не поддерживается
** (gedit:3500): WARNING **: 14:11:48.882: Set document metadata failed: Установ
ка атрибута metadata::gedit-spell-language не поддерживается
** (gedit:3500): WARNING **: 14:11:48.882: Set document metadata failed: Установ
ка атрибута metadata::gedit-encoding не поддерживается
** (gedit:3500): WARNING **: 14:11:50.596: Set document metadata failed: Установ
ка атрибута metadata::gedit-position не поддерживается
[root@fedora lab05]# nasm -f elf lab5.asm
[root@fedora lab05]# ls
hello hello.asm hello.o lab5.asm lab5.o list.lst main obj.o
[root@fedora lab05]#
```

Рис. 4.14: Транспируем lab5.asm в объектный файл



```
root@fedora:~/work/arch-pc/lab05
** (gedit:3500): WARNING **: 14:11:19.792: atk-bridge: get_device_events_reply:
unknown signature
** (gedit:3500): WARNING **: 14:11:36.067: Set document metadata failed: Установ
ка атрибута metadata::gedit-spell-language не поддерживается
** (gedit:3500): WARNING **: 14:11:36.067: Set document metadata failed: Установ
ка атрибута metadata::gedit-encoding не поддерживается
** (gedit:3500): WARNING **: 14:11:48.882: Set document metadata failed: Установ
ка атрибута metadata::gedit-spell-language не поддерживается
** (gedit:3500): WARNING **: 14:11:48.882: Set document metadata failed: Установ
ка атрибута metadata::gedit-encoding не поддерживается
** (gedit:3500): WARNING **: 14:11:50.596: Set document metadata failed: Установ
ка атрибута metadata::gedit-position не поддерживается
[root@fedora lab05]# nasm -f elf lab5.asm
[root@fedora lab05]# ls
hello hello.asm hello.o lab5.asm lab5.o list.lst main obj.o
[root@fedora lab05]# ld -m elf_i386 lab5.o -o lab5
[root@fedora lab05]# ls
hello hello.asm hello.o lab5 lab5.asm lab5.o list.lst main obj.o
[root@fedora lab05]#
```

Рис. 4.15: Транспируем lab5.asm в объектный файл

```
root@fedora:~/work/arch-pc/lab05

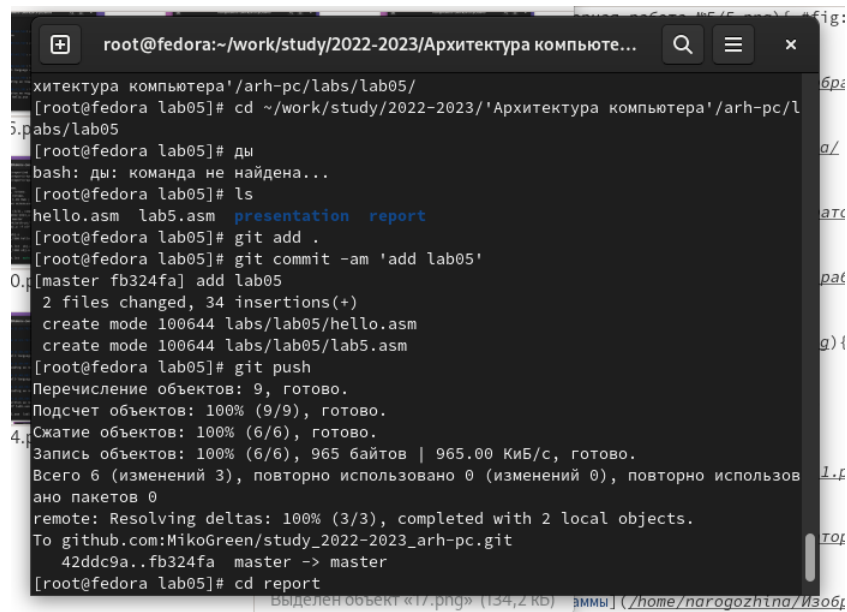
** (gedit:3500): WARNING **: 14:11:36.067: Set document metadata failed: Установка атрибута metadata::gedit-spell-language не поддерживается
** (gedit:3500): WARNING **: 14:11:36.067: Set document metadata failed: Установка атрибута metadata::gedit-encoding не поддерживается
** (gedit:3500): WARNING **: 14:11:48.882: Set document metadata failed: Установка атрибута metadata::gedit-spell-language не поддерживается
** (gedit:3500): WARNING **: 14:11:48.882: Set document metadata failed: Установка атрибута metadata::gedit-encoding не поддерживается
** (gedit:3500): WARNING **: 14:11:50.596: Set document metadata failed: Установка атрибута metadata::gedit-position не поддерживается
[root@fedora lab05]# nasm -f elf lab5.asm
[root@fedora lab05]# ls
hello  hello.asm  hello.o  lab5.asm  lab5.o  list.lst  main  obj.o
[root@fedora lab05]# ld -m elf_i386 lab5.o -o lab5
[root@fedora lab05]# ls
hello  hello.asm  hello.o  lab5  lab5.asm  lab5.o  list.lst  main  obj.o
[root@fedora lab05]# ./lab5
Рогожина Надежда
[root@fedora lab05]#
```

Рис. 4.16: Запустим исполняемый файл lab5

```
root@fedora:~/work/study/2022-2023/Архитектура компьюте...

** (gedit:3500): WARNING **: 14:11:48.882: Set document metadata failed: Установка атрибута metadata::gedit-encoding не поддерживается
** (gedit:3500): WARNING **: 14:11:50.596: Set document metadata failed: Установка атрибута metadata::gedit-position не поддерживается
[root@fedora lab05]# nasm -f elf lab5.asm
[root@fedora lab05]# ls
hello  hello.asm  hello.o  lab5.asm  lab5.o  list.lst  main  obj.o
[root@fedora lab05]# ld -m elf_i386 lab5.o -o lab5
[root@fedora lab05]# ls
hello  hello.asm  hello.o  lab5  lab5.asm  lab5.o  list.lst  main  obj.o
[root@fedora lab05]# ./lab5
Рогожина Надежда
[root@fedora lab05]# cp ~/work/arch-pc/lab05/hello.asm ~/work/study/2022-2023/'Архитектура компьютера'/arh-pc/labs/lab05/
[root@fedora lab05]# cp ~/work/arch-pc/lab05/lab5.asm ~/work/study/2022-2023/'Архитектура компьютера'/arh-pc/labs/lab05/
[root@fedora lab05]# cd ~/work/study/2022-2023/'Архитектура компьютера'/arh-pc/labs/lab05
[root@fedora lab05]# dy
bash: dy: команда не найдена...
[root@fedora lab05]# ls
hello.asm  lab5.asm  presentation  report
[root@fedora lab05]#
```

Рис. 4.17: Скопируем файлы hello.asm и lab5.asm в локальный репозиторий



```
root@fedora:~/work/study/2022-2023/Архитектура компьюте...
[root@fedora lab05]# cd ~/work/study/2022-2023/'Архитектура компьютера'/arh-pc/l
5.pabs/lab05
[root@fedora lab05]# ls
hello.asm  lab5.asm  presentation  report
[root@fedora lab05]# git add .
[root@fedora lab05]# git commit -am 'add lab05'
[master fb324fa] add lab05
2 files changed, 34 insertions(+)
create mode 100644 labs/lab05/hello.asm
create mode 100644 labs/lab05/lab5.asm
[root@fedora lab05]# git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 965 байтов | 965.00 КиБ/с, готово.
Всего 6 (изменений 3), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:MikoGreen/study_2022-2023_arh-pc.git
42ddc9a..fb324fa master -> master
[root@fedora lab05]# cd report
```

Рис. 4.18: Загрузим файлы на Github

5 Вопросы для самопроверки

1. Какие основные отличия ассемблерных программ от программ на языках высокого уровня?

Для того, чтобы писать программы на ассемблере, необходимо знать, какие регистры процессора существуют и как их можно использовать, так как большинство команд в программах написанных на ассемблере используют регистры в качестве операндов.

Грубо говоря, на ассемблере мы “программируем непосредственно железо”, а на языках высокого уровня - “программируем программы”. Программы, написанные на ассемблере, работают на самом низком уровне работы прикладной программы.

2. В чем состоит отличие инструкции от директивы на языке ассемблера?

Директива - это инструкция, не переводящаяся непосредственно в машинные команды, а управляющие работой транслятора.

3. Перечислите основные правила оформления программ на языке ассемблера.

Каждая программа располагается на отдельной строке. Размещение нескольких команд на одной строке недопустимо.

Синтаксис ассемблера является *чувствительным к регистру*.

Машинная команда представляет собой одну строку, имеющую синтаксический вид:

метка команда/директива операнд(ы) ;комментарии

4. Каковы этапы получения исполняемого файла?

Трансляция, компоновка.

5. Каково назначение этапа трансляции?

При наличии ошибок в тексте программы объектный файл, который создаётся на этапе трансляции, создан не будет, а после запуска транслятора появятся сообщения об ошибках или предупреждения.

6. Каково назначение этапа компоновки?

Компоновщик преобразует объектный файл в исполняемый.

7. Какие файлы могут создаваться при трансляции программы, какие из них создаются по умолчанию?

В результате трансляции образуется объектный файл с расширением `.o`. Также в работе мы создавали файл листинга `list.lst`.

8. Каковы форматы файлов для `nasm` и `ld`?

Для `nasm` формат файлов - **`.asm`**, который после превращается в объектный файл с расширением - **`.o`**

Для `ld` формат обрабатываемого файла - **`.o`**, после компоновки файл преобразуется в исполняемый

6 Выводы

Таким образом, мы изучение языка Assembler, освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.