

Отчет по лабораторной работе 4

**Создание и процесс обработки программ на языке ассемблера
NASM**

Симонова Полина Игоревна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Программа Hello world!	8
4.2	Транслятор NASM	10
4.3	Расширенный синтаксис командной строки NASM	10
4.4	Компоновщик LD	11
4.5	Запуск исполняемого файла	12
4.6	Задание для самостоятельной работы	12
5	Выводы	15
6	Список литературы	16

Список иллюстраций

3.1	Процесс создания ассемблерной программы	6
4.1	Установка nasm через терминал	8
4.2	Создание каталога для работы на языке NASM	9
4.3	Перехожу в созданный каталог	9
4.4	Создание файла и редактирование в gedit	9
4.5	Команда для компиляции	10
4.6	Проверка созданного файла	10
4.7	Выполнение команды	11
4.8	Проверка созданных файлов	11
4.9	Передача объектного файла компоновщику и проверка	11
4.10	Исполнение команды	11
4.11	Запуск созданного файла	12
4.12	Создаю копию файла с новым именем	12
4.13	Открываю gedit	12
4.14	Вношу свои имя и фамилию	13
4.15	Компиляция объектного файла	13
4.16	Передача объектного файла компоновщику	13
4.17	Запуск программы	13
4.18	Отправка на гитхаб	14

1 Цель работы

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

2 Задание

1. Программа Hello world!
2. Транслятор NASM
3. Расширенный синтаксис командной строки NASM
4. Компоновщик LD
5. Запуск исполняемого файла
6. Задание для самостоятельной работы

3 Теоретическое введение

Процесс создания ассемблерной программы можно изобразить в виде следующей схемы.



Рис. 3.1: Процесс создания ассемблерной программы

В процессе создания ассемблерной программы можно выделить четыре шага:

1. 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`.
2. 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.

3. 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.
4. 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.

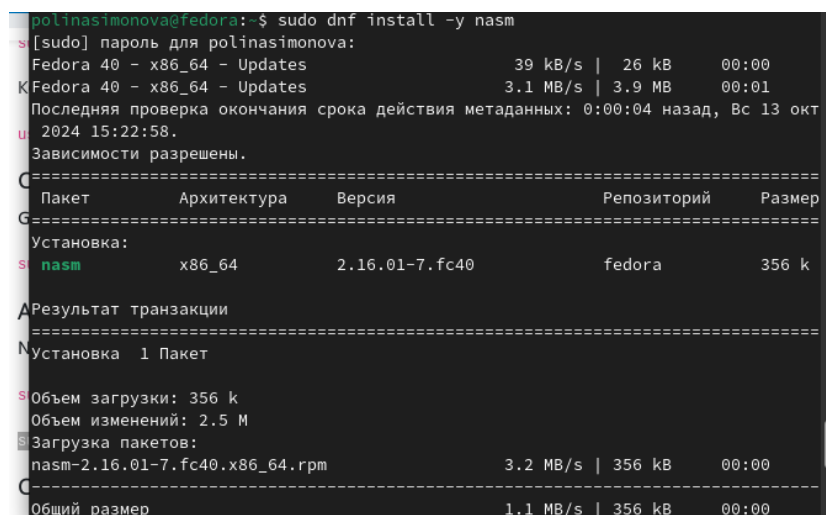
Из-за специфики программирования, а также по традиции для создания программ на языке ассемблера обычно пользуются утилитами командной строки (хотя поддержка ассемблера есть в некоторых универсальных интегрированных средах).

4 Выполнение лабораторной работы

4.1 Программа Hello world!

Для начала, использую команды для установки nasm через терминал:(рис. 4.1).

```
sudo dnf install -y nasm
```



```
polinasimonova@fedora: ~$ sudo dnf install -y nasm
[sudo] пароль для polinasimonova:
Fedora 40 - x86_64 - Updates 39 kB/s | 26 kB 00:00
Fedora 40 - x86_64 - Updates 3.1 MB/s | 3.9 MB 00:01
Последняя проверка окончания срока действия метаданных: 0:00:04 назад, Вс 13 окт
2024 15:22:58.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
=====
Установка:
s nasm      x86_64      2.16.01-7.fc40  fedora      356 k
=====
Результат транзакции
=====
Установка 1 Пакет
=====
s Объем загрузки: 356 k
s Объем изменений: 2.5 M
s Загрузка пакетов:
nasm-2.16.01-7.fc40.x86_64.rpm 3.2 MB/s | 356 kB 00:00
=====
Общий размер 1.1 MB/s | 356 kB 00:00
```

Рис. 4.1: Установка nasm через терминал

Рассмотрим пример простой программы на языке ассемблера NASM. Традиционно первая программа выводит приветственное сообщение Hello world! на экран.

Создаю каталог для работы с программами на языке ассемблера NASM:(рис. 4.2).

```
mkdir -p ~/work/arch-pc/lab04
```



```
polinasimonova@fedora:~$ mkdir -p ~/work/arch-pc/lab04
polinasimonova@fedora:~$
```

Рис. 4.2: Создание каталога для работы на языке NASM

Перехожу в созданный каталог (рис. 4.3).

```
cd ~/work/arch-pc/lab04
```

```
polinasimonova@fedora:~$ cd ~/work/arch-pc/lab04
polinasimonova@fedora:~/work/arch-pc/lab04$
```

Рис. 4.3: Перехожу в созданный каталог

Создаю текстовый файл с именем hello.asm (рис. 4.4).

```
touch hello.asm
```

и открываю этот файл с помощью текстового редактора gedit (рис. 4.4).

```
gedit hello.asm
```

```
polinasimonova@fedora:~/work/arch-pc/lab04$ touch hello.asm
polinasimonova@fedora:~/work/arch-pc/lab04$ gedit hello.asm
```

Рис. 4.4: Создание файла и редактирование в gedit

и ввожу в него следующий текст:

```
; hello.asm
```

```
SECTION .data ; Начало секции данных
```

```
hello: DB 'Hello world!',10 ; 'Hello world!' плюс
```

```
; символ перевода строки
```

```
helloLen: EQU $-hello ; Длина строки hello
```

```
SECTION .text ; Начало секции кода
```

```
GLOBAL _start
```

```
_start: ; Точка входа в программу
```

```
mov eax,4 ; Системный вызов для записи (sys_write)
```

```
mov ebx,1 ; Описатель файла '1' - стандартный вывод
```

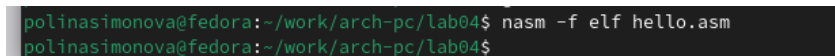
```
mov ecx,hello ; Адрес строки hello в ecx
```

```
mov edx,helloLen ; Размер строки hello
int 80h ; Вызов ядра
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
int 80h ; Вызов ядра
```

4.2 Транслятор NASM

NASM превращает текст программы в объектный код. Например, для компиляции приведённого выше текста программы «Hello World» прописываю: (рис. 4.5).

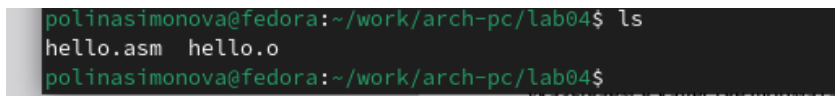
```
nasm -f elf hello.asm
```



```
polinasimonova@fedora:~/work/arch-pc/lab04$ nasm -f elf hello.asm
polinasimonova@fedora:~/work/arch-pc/lab04$
```

Рис. 4.5: Команда для компиляции

Транслятор преобразовывает текст программы из файла hello.asm в объектный код, который записывается в файл hello.o. С помощью команды ls проверяю, что объектный файл был создан. Созданный объектный файл имеет имя hello.o (рис. 4.6).



```
polinasimonova@fedora:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
polinasimonova@fedora:~/work/arch-pc/lab04$
```

Рис. 4.6: Проверка созданного файла

4.3 Расширенный синтаксис командной строки NASM

Выполняю следующую команду: (рис. 4.7).

```
nasm -o obj.o -f elf -g -l list.lst hello.asm
```

```
polinasimonova@fedora:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst
hello.asm
polinasimonova@fedora:~/work/arch-pc/lab04$
```

Рис. 4.7: Выполнение команды

С помощью команды `ls` проверяю, что файлы были созданы.

```
polinasimonova@fedora:~/work/arch-pc/lab04$ ls
hello.asm hello.o list.lst obj.o
polinasimonova@fedora:~/work/arch-pc/lab04$
```

Рис. 4.8: Проверка созданных файлов

4.4 Компоновщик LD

Чтобы получить исполняемую программу, объектный файл передаю на обработку компоновщику:

```
ld -m elf_i386 hello.o -o hello
```

и с помощью команды `ls` проверяю, что исполняемый файл `hello` был создан.

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

Рис. 4.9: Передача объектного файла компоновщику и проверка

Выполняю следующую команду:

```
ld -m elf_i386 obj.o -o main
```

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

Рис. 4.10: Исполнение команды

Исполняемый файл имеет имя `hello`. Объектный файл из которого собран этот исполняемый файл имеет имя `hello.o`

4.5 Запуск исполняемого файла

Запускаю созданный исполняемый файл, находящийся в текущем каталоге, набрав в командной строке:

`./hello`


A terminal window with a dark background. The prompt is 'polinasimonova@fedora:~/work/arch-pc/lab04\$'. The user enters './hello'. The output is 'Hello world!'. The prompt returns to 'polinasimonova@fedora:~/work/arch-pc/lab04\$'.

Рис. 4.11: Запуск созданного файла

4.6 Задание для самостоятельной работы

В каталоге `~/work/arch-pc/lab04` с помощью команды `cp` создаю копию файла (рис. 4.12).

`hello.asm` с именем `lab4.asm`

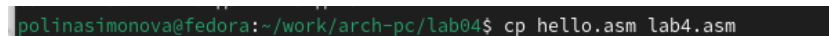
A terminal window with a dark background. The prompt is 'polinasimonova@fedora:~/work/arch-pc/lab04\$'. The user enters 'cp hello.asm lab4.asm'.

Рис. 4.12: Создаю копию файла с новым именем

Открываю текстовый редактор `gedit` (рис. 4.13).

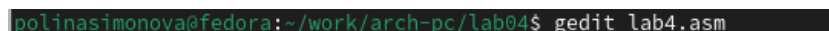
A terminal window with a dark background. The prompt is 'polinasimonova@fedora:~/work/arch-pc/lab04\$'. The user enters 'gedit lab4.asm'.

Рис. 4.13: Открываю gedit

Вношу изменения в текст программы в файле `lab4.asm` так, чтобы вместо `Hello world!` на экран выводилась строка с моим фамилией и именем. (рис. 4.14).

```

|; lab4.asm
SECTION .data ; Начало секции данных
lab4: DB 'Полина Симонова',10 ; 'Hello world!' плюс
; символ перевода строки
lab4len: EQU $-lab4 ; Длина строки hello
SECTION .text ; Начало секции кода
GLOBAL _start
_start: ; Точка входа в программу
mov eax,4 ; Системный вызов для записи (sys_write)
mov ebx,1 ; Описатель файла '1' - стандартный вывод
mov ecx,lab4 ; Адрес строки hello в ecx
mov edx,lab4len ; Размер строки hello
int 80h ; Вызов ядра
mov eax,1 ; Системный вызов для выхода (sys_exit)
mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
int 80h ; Вызов ядра

```

Рис. 4.14: Вношу свои имя и фамилию

Компилирую текст программы в объектный файл (рис. 4.15).

```

polinasimonova@fedora:~/work/arch-pc/lab04$ nasm -f elf lab4.asm

```

Рис. 4.15: Компиляция объектного файла

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4 (рис. 4.16).

```

polinasimonova@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4

```

Рис. 4.16: Передача объектного файла компоновщику

Запускаю исполняемый файл, на экран действительно выводятся мои имя и фамилия (рис. 4.17).

```

polinasimonova@fedora:~/work/arch-pc/lab04$ ./lab4
Полина Симонова
polinasimonova@fedora:~/work/arch-pc/lab04$

```

Рис. 4.17: Запуск программы

С помощью команд `git add .` и `git commit` добавляю файлы на гитхаб и отправляю файлы на сервер с помощью команды `git push` (рис. 4.18).

```
polinasimonova@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git
add .
polinasimonova@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git
commit -am 'feat(main): add files lab-4'
[master 4a9af13] feat(main): add files lab-4
2 files changed, 32 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
polinasimonova@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git
push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 1020 байтов | 1020.00 КиБ/с, готово.
Total 6 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:o5o6am/study_2023-2024_arh--pc-.git
4a808ce..4a9af13 master -> master
polinasimonova@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc$
```

Рис. 4.18: Отправка на гитхаб

5 Выводы

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

6 Список литературы

Архитектура 4