

Отчёт по лабораторной работе № 4

**дисциплина: Архитектура компьютера. Создание и процесс
обработки программ на языке ассемблера NASM**

Студент: Святашова Ксения Евгеньевна

Содержание

1	Цель работы	4
2	Теоритическое введение	5
3	Выполнение лабораторной работы	7
3.1	Программа Hello world!	7
3.2	Транслятор NASM	9
3.3	Расширенный синтаксис командной строки NASM	10
3.4	Компоновщик LD	11
3.5	Запуск исполняемого файла	13
4	Задания для самостоятельной работы	14
5	Вывод	18

Список иллюстраций

3.1	Создание каталога	7
3.2	Каталог	8
3.3	Файл hello.asm	8
3.4	Открытие файла с помощью gedit	8
3.5	Текст, выполняющий команду Hello world!	9
3.6	Компиляция программы Hello world	10
3.7	Проверка объектного файла	10
3.8	Компиляция файла obj.o	11
3.9	Проверка созданных файлов	11
3.10	Обработка программы компоновщиком	11
3.11	Проверка исполняемого файла	12
3.12	Обработка компоновщиком	12
3.13	Проверяем наличие файлов	12
3.14	Выполнение программы Hello world!	13
4.1	Выполнение программы Hello world!	14
4.2	Программа, выводящая имя и фамилию	15
4.3	Оттрансляция lab4.asm в объектный файл	15
4.4	Компоновка объектного файла	16
4.5	Выполнение программы, выводящая имя и фамилию . . .	16
4.6	Копируем файлы	16
4.7	Проверка файлов	17
4.8	Загрузка файлов на Github	17

1 Цель работы

Целью работы является освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

2 Теоритическое введение

NASM успешно конкурирует со стандартным в Linux- и многих других UNIX-системах ассемблером `gas`. Считается, что качество документации у NASM выше, чем у `gas`. Кроме того, ассемблер `gas` по умолчанию использует AT&T-синтаксис, ориентированный на процессоры не от Intel, в то время как NASM использует вариант традиционного для x86-ассемблеров Intel-синтаксиса; Intel-синтаксис используется всеми ассемблерами для DOS/Windows, например, MASM, TASM, `fasm`.

NASM компилирует программы под различные операционные системы в пределах x86-совместимых процессоров. Находясь в одной операционной системе, можно беспрепятственно откомпилировать исполняемый файл для другой. В общем встроенные средства NASM позволяет компилировать не только программы, но и файлы с любым содержимым. Также мощный макро-препроцессор значительно расширяет возможности для программирования.

Компиляция программ в NASM состоит из двух этапов. Первый — ассемблирование, второй — компоновка. На этапе ассемблирования создаётся объектный код. В нём содержится машинный код программы и данные, в соответствии с исходным кодом, но идентификаторы (переменные, символы) пока не привязаны к адресам памяти. На этапе компоновки из одного или нескольких объектных модулей создаётся исполняемый


файл (программа). Операция компоновки связывает идентификаторы, определённые в основной программе, с идентификаторами, определёнными в остальных модулях, после чего всем идентификаторам даются окончательные адреса памяти или обеспечивается их динамическое выделение.

Для компоновки объектных файлов в исполняемые в Windows можно использовать свободный бесплатно распространяемый компоновщик alink(для 64-битных программ компоновщик GoLink), а в Linux — компоновщик ld, который есть в любой версии этой операционной системы.

3 Выполнение лабораторной работы

3.1 Программа Hello world!

Рассмотрим пример простой программы на языке ассемблера NASM. Традиционно первая программа выводит приветственное сообщение “Hello world!” на экран. Создадим каталог для работы с программами на языке ассемблера NASM(рис. 3.1):



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

Рис. 3.1: Создание каталога

Перейдем в созданный каталог(рис. 3.2):

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

Рис. 3.2: Каталог

Создадим текстовый файл с именем hello.asm(рис. 3.3):

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

Рис. 3.3: Файл hello.asm

Откроем этот файл с помощью текстового редактора gedit(рис. 3.4):

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

Рис. 3.4: Открытие файла с помощью gedit

и введем в него следующий текст(рис. 3.5):


```

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

```

Рис. 3.5: Текст, выполняющий команду Hello world!

В отличие от многих современных и высокоуровневых языков программирования, в ассемблерной программе каждая команда располагается на *отдельной строке*. Размещение нескольких команд на одной строке **недопустимо**. Синтаксис ассемблера NASM является *чувствительным к регистру*, т.е. есть разница между большими и малыми буквами.

3.2 Транслятор NASM

NASM превращает текст программы в объектный код. Для компиляции приведенного выше текста программы “Hello World” необходимо написать(рис. 3.6):

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

Рис. 3.6: Компиляция программы Hello world

Если текст программы набран без ошибок, то транслятор преобразует текст программы из файла `hello.asm` в объектный код, который запишется в файл `hello.o`. С помощью команды `ls` проверим, что объектный файл был создан(рис. 3.7):

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

Рис. 3.7: Проверка объектного файла

3.3 Расширенный синтаксис командной строки

NASM

Выполним следующую команду(рис. 3.8), которая скомпилирует исходный файл `hello.asm` в `obj.o`(опция `-o`), кроме того, будет создан файл листинга `list.lst`(опция `-l`):

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

Рис. 3.8: Компиляция файла obj.o

С помощью команды `ls` проверим, что файлы были созданы(рис. 3.9):

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

Рис. 3.9: Проверка созданных файлов

3.4 Компоновщик LD

Чтобы получить исполняемую программу, объектный файл необходимо передать на обработку компоновщику(рис. 3.10). Ключ `-o` с последующим значением задает имя создаваемого файла.

```
kesvyatashova@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
kesvyatashova@fedora:~/work/arch-pc/lab04$
```

Рис. 3.10: Обработка программы компоновщиком

С помощью команды `ls` проверим, что исполняемый файл `hello` был создан(рис. 3.11):

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

Рис. 3.11: Проверка исполняемого файла

Введем следующую команду(рис. 3.12). Исполняемый файл будет иметь имя `main`, потому что после ключа `-o` было задано значение `main`. Объектный файл, из которого собран исполняемый файл, будет называться `obj.o`.

```
kesvyatashova@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
kesvyatashova@fedora:~/work/arch-pc/lab04$
```

Рис. 3.12: Обработка компоновщиком

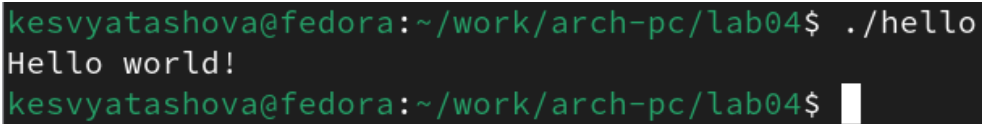
Проверяем созданные файлы(рис. 3.13):

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

Рис. 3.13: Проверяем наличие файлов

3.5 Запуск исполняемого файла

Запустим на выполнение созданный исполняемый файл, находящийся в текущем каталоге, набрав в командной строке следующую команду(рис. 3.14):


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

```
kesvyatashova@fedora:~/work/arch-pc/lab04$ ./hello
Hello world!
kesvyatashova@fedora:~/work/arch-pc/lab04$
```

Рис. 3.14: Выполнение программы Hello world!

4 Задания для самостоятельной работы

1. В каталоге `~/work/arch-pc/lab04` с помощью команды `cp` создадим копию файла `hello.asm` с именем `lab4.asm`(рис. 4.1):



```
kesvyatashova@fedora:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
kesvyatashova@fedora:~/work/arch-pc/lab04$
```

Рис. 4.1: Выполнение программы Hello world!

2. С помощью текстового редактора `gedit` внесем изменения в текст программы в файле `lab4.asm` так, чтобы вместо `Hello world!` на экран выводилась строка с нашей фамилией и именем(рис. 4.2):

```

1 ; lab4.asm
2 SECTION .data ; Начало секции данных
3 lab4: DB 'Ксения Святасова',10
4 ; символ перевода строки
5 lab4Len: EQU $-lab4 ; Длина строки lab4
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,lab4 ; Адрес строки lab4 в ecx
12 mov edx,lab4Len ; Размер строки lab4
13 int 80h ; Вызов ядра
14 mov eax,1 ; Системный вызов для выхода (sys_exit)
15 mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
16 int 80h ; Вызов ядра

```

Рис. 4.2: Программа, выводящая имя и фамилию

3. Оттранслируем полученный текст программы lab4.asm в объектный файл(рис. 4.3):

```

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

```

Рис. 4.3: Оттрансляция lab4.asm в объектный файл

Выполним компоновку объектного файла(рис. 4.4):

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

Рис. 4.4: Компоновка объектного файла

Запустим получившийся исполняемый файл(рис. 4.5):

```
kesvyatashova@fedora:~/work/arch-pc/lab04$ ./lab4
Ксения Святашова
kesvyatashova@fedora:~/work/arch-pc/lab04$
```

Рис. 4.5: Выполнение программы, выводящая имя и фамилию

4. Скопируем файлы hello.asm lab4.asm в наш локальный репозиторий в каталог ~/work/study/2024-2025/“Архитектура компьютера”/arch-pc/labs/lab04(рис. 4.6):

```
kesvyatashova@fedora:~$ cp ~/work/arch-pc/lab04/hello.asm ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab04/
kesvyatashova@fedora:~$ cp ~/work/arch-pc/lab04/lab4.asm ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab04/
kesvyatashova@fedora:~$
```

Рис. 4.6: Копируем файлы

Проверим, что файлы скопировались(рис. 4.7):


```
kesvyatashova@fedora:~$ cd ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab04
kesvyatashova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello.asm lab4.asm presentation report
kesvyatashova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$
```

Рис. 4.7: Проверка файлов

Загрузим файлы на Github(рис. 4.8):

```
kesvyatashova@fedora:~$ cd ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc
kesvyatashova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git add .
kesvyatashova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git commit -am 'add files lab-4'
[master d13bb72] add files lab-4
2 files changed, 32 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
kesvyatashova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 1.02 КиБ | 149.00 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.
remote: This repository moved. Please use the new location:
remote:   git@github.com:kesvyatashova/study_2024-2025_arch-pc.git
To github.com:kesvyatashova/study_2024-2025_arh-pc.git
413308e..d13bb72 master -> master
```

Рис. 4.8: Загрузка файлов на Github

5 Вывод

В результате выполнения работы я освоила процедуру компиляции и сборки программ, написанных на ассемблере NASM.