

Отчёт по лабораторной работе №4

Дисциплина: архитектура компьютера

Горобцова Арина Романовна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Создание программы Hello world!	7
3.2	Работа с транслятором NASM	8
3.3	Работа с расширенным синтаксисом командной строки NASM . . .	8
3.4	Работа с компоновщиком LD	8
3.5	Запуск исполняемого файла	9
3.6	Выполнение заданий для самостоятельной работы	9
4	Выводы	12

Список иллюстраций

3.1	Создание каталога	7
3.2	Переход в созданный каталог	7
3.3	Создание текстового файла hello.asm и проверка через ls	7
3.4	Открытие файла с помощью текстового редактора gedit и ввод в него текста	7
3.5	Компиляция текста программы	8
3.6	Компиляция текста программы	8
3.7	Передача объектного файла на обработку компоновщику	9
3.8	Передача объектного файла на обработку компоновщику	9
3.9	Запуск исполняемого файла	9
3.10	Создание копии файла	9
3.11	Изменение программы	10
3.12	Компиляция текста программы	10
3.13	Передача объектного файла на обработку компоновщику	10
3.14	Запуск исполняемого файла	10
3.15	Компиляция файлов в репозиторий и загрузка их на github	11

Список таблиц

1 Цель работы

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

2 Задание

1.Работа с программой 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 Выполнение лабораторной работы

3.1 Создание программы Hello world!

Создаем каталог для работы с программами на языке ассемблера NASM (рис. 3.1).

```
argorobcova@dk3n55 ~ $ mkdir -p ~/work/arch-pc/lab04
```

Рис. 3.1: Создание каталога

Переходим в созданный каталог (рис. 3.2).

```
argorobcova@dk3n55 ~ $ cd ~/work/arch-pc/lab04
```

Рис. 3.2: Переход в созданный каталог

Создаем текстовый файл с именем hello.asm и проверяем через команду ls (рис. 3.3).

```
argorobcova@dk3n55 ~/work/arch-pc/lab04 $ touch hello.asm
argorobcova@dk3n55 ~/work/arch-pc/lab04 $ ls
hello.asm
```

Рис. 3.3: Создание текстового файла hello.asm и проверка через ls

Открываем этот файл с помощью любого текстового редактора, например, gedit и вводим в него текст (рис. 3.4).

```
argorobcova@dk3n55 ~/work/arch-pc/lab04 $ gedit hello.asm
```

Рис. 3.4: Открытие файла с помощью текстового редактора gedit и ввод в него текста

3.2 Работа с транслятором NASM

Превращаю текст программы для вывода “Hello world!” в объектный код с помощью транслятора NASM, используя команду `nasm -f elf hello.asm`, ключ `-f` указывает транслятору `nasm`, что требуется создать бинарный файл в формате ELF (рис. 3.5). Далее проверяю правильность выполнения команды с помощью утилиты `ls`: действительно, создан файл “hello.o”.

```
argorobcova@dk3n55 ~ $ cd ~/work/arch-pc/lab04
argorobcova@dk3n55 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
argorobcova@dk3n55 ~/work/arch-pc/lab04 $ ls
hello.asm hello.o
```

Рис. 3.5: Компиляция текста программы

3.3 Работа с расширенным синтаксисом командной строки NASM

Ввожу команду, которая скомпилирует файл `hello.asm` в файл `obj.o`, при этом в файл будут включены символы для отладки (ключ `-g`), также с помощью ключа `-l` будет создан файл листинга `list.lst` (рис. 3.6). Далее проверяю с помощью утилиты `ls` правильность выполнения команды.

```
argorobcova@dk3n55 ~ $ cd ~/work/arch-pc/lab04
argorobcova@dk3n55 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
argorobcova@dk3n55 ~/work/arch-pc/lab04 $ ls
hello.asm hello.o
```

Рис. 3.6: Компиляция текста программы

3.4 Работа с компоновщиком LD

Передаю объектный файл `hello.o` на обработку компоновщику `LD`, чтобы получить исполняемый файл `hello` (рис. 3.7). Ключ `-o` задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты `ls` правильность выполнения команды.


```
argorobcova@dk3n55 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
argorobcova@dk3n55 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  obj.o
```

Рис. 3.7: Передача объектного файла на обработку компоновщику

Выполняю следующую команду (рис. 3.8). Исполняемый файл будет иметь имя main, т.к. после ключа -o было задано значение main. Объектный файл, из которого собран этот исполняемый файл, имеет имя obj.o

```
argorobcova@dk3n55 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
argorobcova@dk3n55 ~/work/arch-pc/lab04 $ ls
```

Рис. 3.8: Передача объектного файла на обработку компоновщику

3.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello (рис. 3.9).

```
argorobcova@dk3n55 ~/work/arch-pc/lab04 $ ./hello
Hello world!
argorobcova@dk3n55 ~/work/arch-pc/lab04 $
```

Рис. 3.9: Запуск исполняемого файла

3.6 Выполнение заданий для самостоятельной работы

С помощью утилиты cp создаю в текущем каталоге копию файла hello.asm с именем lab4.asm (рис. 3.10).

```
argorobcova@dk3n55 ~/work/arch-pc/lab04 $ cp hello.asm lab4.asm
```

Рис. 3.10: Создание копии файла

С помощью текстового редактора mousepad открываю файл lab4.asm и вношу изменения в программу так, чтобы она выводила мои имя и фамилию. (рис. 3.11).

```
lab4.asm      [-M--] 27 L:[ 1+ 4 5/ 32] *(96 / 831b) 0039 0x027 [*][X]
; hello.asm

SECTION .data ; Начало секции данных

hello: DB 'Arina Gorobtsova',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

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход
```

Рис. 3.11: Изменение программы

Компилирую текст программы в объектный файл (рис. 3.12). Проверяю с помощью утилиты ls, что файл lab4.o создан.

```
argorobcova@dk3n55 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
argorobcova@dk3n55 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o lab4.asm lab4.o list.lst main obj.o
argorobcova@dk3n55 ~/work/arch-pc/lab04 $
```

Рис. 3.12: Компиляция текста программы

Передаю объектный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4 (рис. 3.13).

```
argorobcova@dk3n55 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o lab4
argorobcova@dk3n55 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o
argorobcova@dk3n55 ~/work/arch-pc/lab04 $
```

Рис. 3.13: Передача объектного файла на обработку компоновщику

Запускаю исполняемый файл lab4, на экран действительно выводятся мои имя и фамилия (рис. 3.14).

```
argorobcova@dk3n55 ~/work/arch-pc/lab04 $ ./lab4
Arina Gorobtsova
argorobcova@dk3n55 ~/work/arch-pc/lab04 $
```

Рис. 3.14: Запуск исполняемого файла

Копируем файлы hello.asm и lab4.asm в наш локальный репозиторий в каталог ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/ и загружаем их на Github (рис. 3.15).

```
argorobcova@dk8n76 ~/work/study/2024-2025/Архитектура компьютера/arch-pc $ git add .
argorobcova@dk8n76 ~/work/study/2024-2025/Архитектура компьютера/arch-pc $ git commit -am 'feat(main): add hello.asm and lab4.asm'
[master 89c929c] feat(main): add hello.asm and lab4.asm
2 files changed, 62 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
argorobcova@dk8n76 ~/work/study/2024-2025/Архитектура компьютера/arch-pc $ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 988 байтов | 988.00 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:Gorobtsova/study_2024-2025_arhpc-.git
 a8fe555..89c929c master -> master
argorobcova@dk8n76 ~/work/study/2024-2025/Архитектура компьютера/arch-pc $
argorobcova@dk8n76 ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04/report $
```

Рис. 3.15: Компиляция файлов в репозиторий и загрузка их на github

4 Выводы

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

...