Отчёт по лабораторной работе №4

дисциплина: Архитектура компьютера

Баранова Анна Андреевна

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 9
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 **16**

List of Figures

4.1 Создание каталога для работы с программами на языке ассемблера NASM 8
4.2 Переход в созданный каталог 8
4.3 Создание текстового файлв с именем hello.asm 8
4.4 Открытие файла с помощью текстового редактора gedit 8
4.5 Ввод текста 9
4.6 Компиляция текста 9
4.7 Проверка создания объектного файла 9
4.8 Компиляция файла hello.asm в obj.o 10
4.9 Проверка создания файлов 10
4.10 Получение более подробной информации 10
4.11 Получение более подробной информации 10
4.12 Получение списка форматов объектного файла 11
4.13 Передача объектного файла на обработку компоновщику 11
4.14 Проверка создания файла 11
4.15 Дача имени файлу 11
4.16 Получение более подробной информации 12
4.17 Получение более подробной информации 12
4.18 Запуск исполняемого файла 12
4.19 Создание копии файла 13
4.20 Проверка создания файла 13
4.21 Открытие файла с помощью текстового редактора gedit 13
4.22 Изменеие текста программы в файле lab4.asm 13
4.23 Компиляция текста 13
4.24 Компиляция файла lab4.asm в obj.o 14
4.25 Передача объектного файла на обработку компоновщику 14
4.26 Дача имени файлу 14
4.27 Проверка создания файлов 14
4.28 Запуск получившегося исполняемого файла 14
4.29 Копирование файла hello.asm 14
4.30 Копирование файла lab4.asm 14
4.31 Загрузка файлов на Github 15

1 Цель работы

Цель работы - освоить процедуры компиляции и сборки программ, которые написаны на ассемблере NASM.

2 Задание

В ходе выполнения данной лабораторной работы необходимо изучить:

- Что такое ассемблер и язык ассемблера;
- как создавать и обрабатывать программы на языке ассемблера;
- как оформлять изображения в Markdown.

Выполнив эту работу, мы приобретём навыки работы процедуру оформления отётов с помощью легковесного языка разметки Markdown.

3 Теоретическое введение

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.

В процессе создания ассемблерной программы можно выделить четыре шага:

- Набор текста программы в текстовом редакторе и сохранение её в отдель-

ном 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 про- граммы в машинный код, называемый объектным. На данном этапе также может быть получен листинг программы, содержащий кроме текста программы различную допол- нительную 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

4.1 Программа Hello world!

Создадим каталог для работы с программами на языке ассемблера NASM (рис. 4.1).

```
aabaranova@dk5n60 ~ $ mkdir -p ~/work/arch-pc/lab04
aabaranova@dk5n60 ~ $
```

Рис. 4.1: Создание каталога для работы с программами на языке ассемблера NASM

Перейдём в созданный каталог (рис. 4.2).

```
aabaranova@dk5n60 ~ $ cd ~/work/arch-pc/lab04
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.2: Переход в созданный каталог

Создадим текстовый файл с именем hello.asm (рис. 4.3).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ touch hello.asm
```

Рис. 4.3: Создание текстового файлв с именем hello.asm

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

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ gedit hello.asm
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.4: Открытие файла с помощью текстового редактора gedit

Введём в него следующий текст (рис. 4.5).

```
SECTION .data
        hello:      db "Hello, world!",0xa
                helloLen:   equ $ - hello
SECTION .text
        global _start

_start:
        mov eax, 4
        mov ebx, 1
        mov ecx, hello
        mov edx, helloLen
        int 0x80

        mov eax, 1
        mov ebx, 0
        int 0x80
```

Рис. 4.5: Ввод текста

4.2 Транслятор NASM

Скомпилируем приведённый выше текст программы «Hello World» (рис. 4.6).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.6: Компиляция текста

С помощью команды ls проверим, что объектный файл был создан (рис. 4.7).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.7: Проверка создания объектного файла

4.3 Расширенный синтаксис командной строки NASM

Скомпилируем исходный файл hello.asm в obj.o, при этом формат выходного файла будет elf, и в него будут включены символы для отладки, кроме того, будет создан файл листинга list.lst (рис. 4.8).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.8: Компиляция файла hello.asm в obj.o

С помощью команды ls проверим, что файлы были созданы (рис. 4.9).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o  list.lst  obj.o
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.9: Проверка создания файлов

Для более подробной информации используем man nasm (рис. 4.10), (рис. 4.11).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ man nasm
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.10: Получение более подробной информации

```
NASM(1)                    The Netwide Assembler Project                    NASM(1)

NAME
       nasm - the Netwide Assembler, a portable 80x86 assembler

SYNOPSIS
       nasm [-@ response file] [-f format] [-o outfile] [-l listfile] [options...] filename

DESCRIPTION
       The nasm command assembles the file filename and directs output to the file outfile
       if specified. If outfile is not specified, nasm will derive a default output file
       name from the name of its input file, usually by appending '.o' or '.obj', or by
       removing all extensions for a raw binary file. Failing that, the output file name
       will be 'nasm.out'.

OPTIONS
       -@ filename
           Causes nasm to process options from filename as if they were included on the
           command line.
```

Рис. 4.11: Получение более подробной информации

Для получения списка форматов объектного файла используем nasm -hf (рис. 4.12).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ nasm -hf
Usage: nasm [-@ response_file] [options...] [--] filename
       nasm -v (or --v)

Options (values in brackets indicate defaults):

    -h            show this text and exit (also --help)
    -v (or --v)   print the NASM version number and exit
    -@ file       response file; one command line option per line

    -o outfile    write output to outfile
    --keep-all    output files will not be removed even if an error happens

    -Xformat      specify error reporting format (gnu or vc)
    -s            redirect error messages to stdout
    -Zfile        redirect error messages to file
```

Рис. 4.12: Получение списка форматов объектного файла

4.4 Компоновщик LD

Передадим объектный файл на обработку компоновщику (рис. 4.13).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.13: Передача объектного файла на обработку компоновщику

С помощью команды ls проверим, что исполняемый файл hello был создан (рис. 4.14).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  obj.o
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.14: Проверка создания файла

Зададим имя создаваемого исполняемого файла (рис. 4.15).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.15: Дача имени файлу

Для получения более подробной информации используем man ld (рис. 4.16), (рис. 4.17).

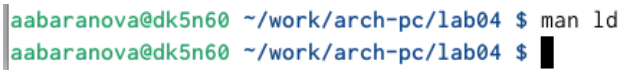

Рис. 4.16: Получение более подробной информации

Рис. 4.17: Получение более подробной информации

4.5 Запуск исполняемого файла

Запустим на выполнение созданный исполняемый файл, находящийся в текущем каталоге (рис. 4.18).

Рис. 4.18: Запуск исполняемого файла

4.6 Задание для самостоятельной работы

В каталоге ~/work/arch-pc/lab04 с помощью команды cp создадим копию файла hello.asm с именем lab4.asm (рис. 4.19).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ cp hello.asm lab4.asm
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.19: Создание копии файла

С помощью команды ls проверим, что файл lab4.asm был создан (рис. 4.20).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4.asm  list.lst  main  obj.o
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.20: Проверка создания файла

С помощью текстового редактора gedit внесём изменения в текст программы в файле lab4.asm так, чтобы вместо Hello world! на экран выводилась строка с моими фамилией и именем (рис. 4.21), (рис. 4.22).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ gedit lab4.asm
```

Рис. 4.21: Открытие файла с помощью текстового редактора gedit

Рис. 4.22: Изменеие текста программы в файле lab4.asm

Скомпилируем текст файла lab4.asm (рис. 4.23).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ nasm -f elf lab4.asm
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.23: Компиляция текста

Скомпилируем исходный файл lab4.asm в obj.o, при этом формат выходного файла будет elf, и в него будут включены символы для отладки, кроме того, будет создан файл листинга list1.lst (рис. 4.24).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list1.lst lab4.asm
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.24: Компиляция файла lab4.asm в obj.o

Передадим объектный файл на обработку компоновщику (рис. 4.25).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o lab4
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.25: Передача объектного файла на обработку компоновщику

Зададим имя создаваемого исполняемого файла (рис. 4.26).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab4.o -o main1
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.26: Дача имени файлу

С помощью команды ls проверим, что нужные файлы были создан (рис. 4.27).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab4  lab4.asm  lab4.o  list1.lst  list.lst  main  main1  obj.o
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.27: Проверка создания файлов

Запустим получившийся исполняемый файл (рис. 4.28).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ ./lab4
Anna Baranova
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.28: Запуск получившегося исполняемого файла

Скопируем файлы hello.asm и lab4.asm в мой локальный репозиторий в каталог ~/work/study/2024-2025/“Архитектура компьютера”/arch-pc/labs/lab04/ (рис. 4.29), (рис. 4.30).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ cp hello.asm ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab04/
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.29: Копирование файла hello.asm

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ cp lab4.asm ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab04/
aabaranova@dk5n60 ~/work/arch-pc/lab04 $
```

Рис. 4.30: Копирование файла lab4.asm

Загрузим файлы на Github (рис. 4.31).

```
aabaranova@dk5n60 ~/work/arch-pc/lab04 $ cd ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc
aabaranova@dk5n60 ~/work/study/2024-2025/Архитектура компьютера/arch-pc $ git add .
aabaranova@dk5n60 ~/work/study/2024-2025/Архитектура компьютера/arch-pc $ git commit -am 'feat(main): ac
[master 19abb5c] feat(main): add files lab-3
 17 files changed, 32 insertions(+), 1 deletion(-)
 delete mode 100644 labs/lab02/report/.__afs0FFB
 delete mode 100644 labs/lab02/report/.~lock.ЛО2_Баранова_отчет.docx#
 create mode 100644 labs/lab03/report/скрины.zip
 create mode 100644 labs/lab03/report/скрины/pic1.png
 create mode 100644 labs/lab03/report/скрины/pic10.png
 create mode 100644 labs/lab03/report/скрины/pic11.png
 create mode 100644 labs/lab03/report/скрины/pic2.png
 create mode 100644 labs/lab03/report/скрины/pic3.png
 create mode 100644 labs/lab03/report/скрины/pic4.png
 create mode 100644 labs/lab03/report/скрины/pic5.png
 create mode 100644 labs/lab03/report/скрины/pic6.png
 create mode 100644 labs/lab03/report/скрины/pic7.png
 create mode 100644 labs/lab03/report/скрины/pic8.png
 create mode 100644 labs/lab03/report/скрины/pic9.png
 create mode 100644 labs/lab03/report/скрины/placeimg_800_600_tech.jpg
 create mode 100644 labs/lab04/hello.asm
 create mode 100644 labs/lab04/lab4.asm
aabaranova@dk5n60 ~/work/study/2024-2025/Архитектура компьютера/arch-pc $ git commit -am 'feat(main): ac
Текущая ветка: master
Ваша ветка опережает «origin/master» на 1 коммит.
  (используйте «git push», чтобы опубликовать ваши локальные коммиты)

нечего коммитить, нет изменений в рабочем каталоге
aabaranova@dk5n60 ~/work/study/2024-2025/Архитектура компьютера/arch-pc $ git push
Перечисление объектов: 18, готово.
Подсчет объектов: 100% (18/18), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (11/11), готово.
Запись объектов: 100% (11/11), 686.15 КиБ | 5.01 МиБ/с, готово.
Total 11 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (5/5), completed with 4 local objects.
To github.com:aabaranova/study_2024-2025_arch-pc.git
   aaaa657..19abb5c  master -> master
aabaranova@dk5n60 ~/work/study/2024-2025/Архитектура компьютера/arch-pc $
```

Рис. 4.31: Загрузка файлов на Github

5 Выводы

В ходе выполнения данной лабораторной работы были освоены процедуры компиляции и сборки программ, которые написаны на ассемблере NASM