

Отчет по выполнению лабораторной работы №4

Дисциплина: архитектура компьютеров

Намруев Максим Саналович

Содержание

| | | |
|----------|---|-----------|
| 1 | Цель работы | 5 |
| 2 | Задание | 6 |
| 3 | Теоретическое введение | 7 |
| 4 | Выполнение лабораторной работы | 9 |
| 4.1 | Программа Hello world! | 9 |
| 4.2 | Транслятор NASM | 10 |
| 4.3 | Расширенный синтаксис командной строки NASM | 11 |
| 4.4 | Коспоновщик LD | 11 |
| 4.5 | Запуск исполняемого файла | 12 |
| 5 | Задание для самостоятельной работы | 13 |
| 6 | Выводы | 16 |

Список иллюстраций

| | | |
|-----|--|----|
| 4.1 | Создание каталога | 9 |
| 4.2 | Переход в каталог | 9 |
| 4.3 | Создание файла | 9 |
| 4.4 | Открывание файла | 10 |
| 4.5 | Выполнение компиляции | 10 |
| 4.6 | Компиляция текста | 11 |
| 4.7 | Передача объектного файла компановщику | 11 |
| 4.8 | Передача объектного файла компановщику | 11 |
| 4.9 | Запуск исполняемого файла | 12 |
| 5.1 | Копирование файла | 13 |
| 5.2 | Редактирование файла | 14 |
| 5.3 | Компиляция файла | 14 |
| 5.4 | Отправка файла на компоновку | 14 |
| 5.5 | Запуск файла | 14 |
| 5.6 | Копирование файлов | 15 |
| 5.7 | Отправка файлов | 15 |

Список таблиц

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

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 языка ассемблера в исполняемый машинный код осуществляется специальной программой транслятором — Ассемблер. Программы, написанные на языке ассемблера, не уступают в качестве и скорости программам, написанным на машинном языке, так как транслятор просто переводит мнемонические обозначения команд в последовательности

бит (нулей и единиц). Используемые мнемоники обычно одинаковы для всех процессоров одной архитектуры или семейства архитектур (среди широко известных — мнемоники процессоров и контроллеров x86, ARM, SPARC, PowerPC, M68k). Таким образом для каждой архитектуры существует свой ассемблер и, соответственно, свой язык ассемблера

4 Выполнение лабораторной работы

4.1 Программа Hello world!

Открываю терминал. Создаю каталог для работы с программами на языке ассемблера NASM. (рис. [4.1]).

```
msnamruev@dk3n33 ~ $ mkdir -p ~/work/arch-pc/lab04  
msnamruev@dk3n33 ~ $
```

Рис. 4.1: Создание каталога

Перехожу в созданный каталог. (рис. [4.2])

```
msnamruev@dk3n33 ~ $ cd ~/work/arch-pc/lab04  
msnamruev@dk3n33 ~/work/arch-pc/lab04 $
```

Рис. 4.2: Переход в каталог

Создаю текстовый файл с именем hello.asm. (рис. [4.3])

```
msnamruev@dk3n33 ~/work/arch-pc/lab04 $ touch hello.asm  
msnamruev@dk3n33 ~/work/arch-pc/lab04 $ ls  
hello.asm  
msnamruev@dk3n33 ~/work/arch-pc/lab04 $
```

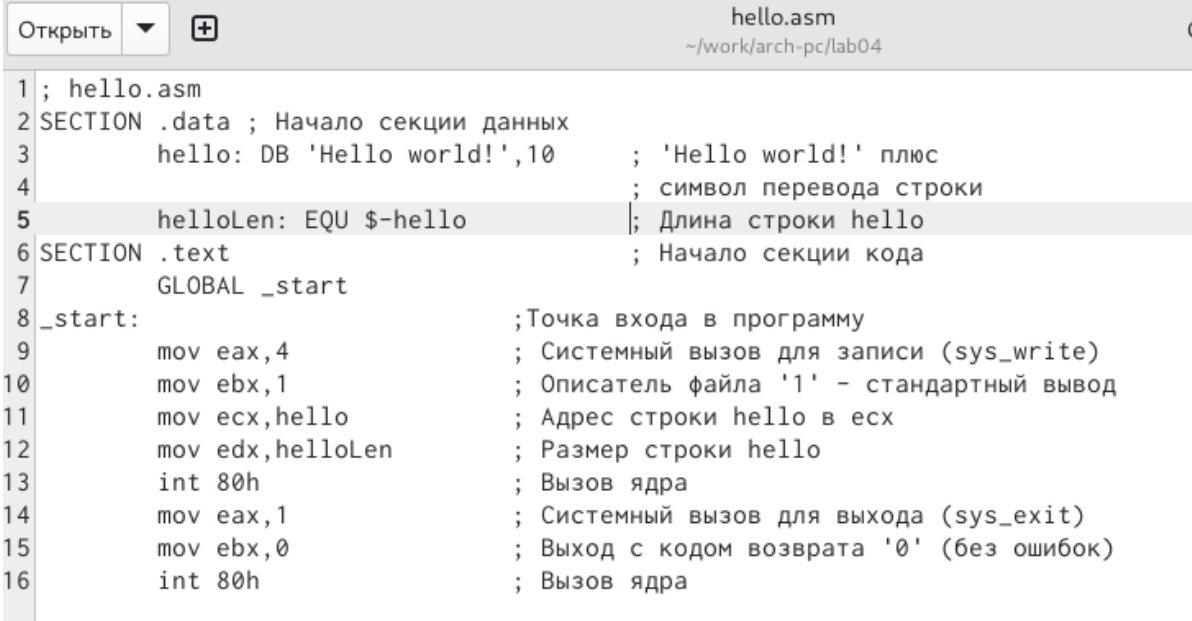
Рис. 4.3: Создание файла

Открываю этот файл с помощью текстового редактора gedit. (рис. [4.4])

```
msnamruev@dk3n33 ~/work/arch-pc/lab04 $ gedit hello.asm
```

Рис. 4.4: Открывание файла

Ввожу в файл следующий текст.(рис. [??])



The screenshot shows a text editor window titled 'hello.asm' with the following assembly code:

```
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
```

{#fig:fig5

width=70%

4.2 Транслятор NASM

Для выполнения компиляции текста программы “Hello world!” пишу: (рис. [4.5])

```
msnamruev@dk3n33 ~/work/arch-pc/lab04 $ nasm -f elf hello.asm
msnamruev@dk3n33 ~/work/arch-pc/lab04 $
```

Рис. 4.5: Выполнение компиляции

4.3 Расширенный синтаксис командной строки NASM

Ввожу команду, которая компилирует исходный файл hello.asm в obj.o при этом формат выходного файла будет elf, и в него будут включены символы для отладки(опция -g), кроме того будет создан файл листинга list.lst(опция -l). С помощью команды ls проверяю, что файлы были созданы.(рис. [4.6])

```
msnamruev@dk2n25 ~/work/arch-pc/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
msnamruev@dk2n25 ~/work/arch-pc/lab04 $ ls
hello.asm  hello.o  list.lst  obj.o
msnamruev@dk2n25 ~/work/arch-pc/lab04 $
```

Рис. 4.6: Компиляция текста

4.4 Коспоновщик LD

Передаю объектный файл на обработку компоновщику и проверяю, что исполняемый файл hello был создан. (рис. [4.7])

```
msnamruev@dk2n25 ~/work/arch-pc/lab04 $ ld -m elf_i386 hello.o -o hello
msnamruev@dk2n25 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  obj.o
msnamruev@dk2n25 ~/work/arch-pc/lab04 $
```

Рис. 4.7: Передача объектного файла компоновщику

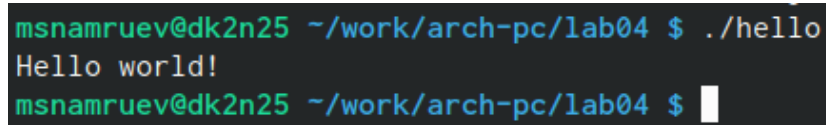
Выполняю следующую команду (рис. [4.8]), после которой файл будет иметь имя main. объектный файл из которого создан исполняемый файл, имеет имя obj.o.

```
msnamruev@dk2n25 ~/work/arch-pc/lab04 $ ld -m elf_i386 obj.o -o main
msnamruev@dk2n25 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  list.lst  main  obj.o
msnamruev@dk2n25 ~/work/arch-pc/lab04 $
```

Рис. 4.8: Передача объектного файла компоновщику

4.5 Запуск исполняемого файла

Запуском созданный исполняемый файл (рис. [4.9])

A terminal window with a dark background. The prompt is 'msnamruev@dk2n25 ~/work/arch-pc/lab04 \$'. The user enters './hello' and the output is 'Hello world!'. The prompt is then shown again with a cursor.

```
msnamruev@dk2n25 ~/work/arch-pc/lab04 $ ./hello
Hello world!
msnamruev@dk2n25 ~/work/arch-pc/lab04 $
```

Рис. 4.9: Запуск исполняемого файла

5 Задание для самостоятельной работы

1. В каталоге `~/work/arch-pc/lab04` с помощью команды `cp` создайте копию файла `hello.asm` с именем `lab4.asm`

Для этого использую команду `cp`, указывая сначала тот файл, который хочу скопировать, а затем копию файла с новым названием. Также не забываю проверить правильность выполненных действий с помощью команды `ls`. (рис. [5.1])

```
msnamruev@dk2n25 ~/work/arch-pc/lab04 $ cp ~/work/arch-pc/lab04/hello.asm ~/work/arch-pc/lab04/lab04.asm
msnamruev@dk2n25 ~/work/arch-pc/lab04 $ ls
hello  hello.asm  hello.o  lab04.asm  list.lst  main  obj.o
msnamruev@dk2n25 ~/work/arch-pc/lab04 $
```

Рис. 5.1: Копирование файла

2. С помощью любого текстового редактора внесите изменения в текст программы в файле `lab4.asm` так, чтобы вместо `Hello world!` на экран выводилась строка с вашими фамилией и именем.

Для этого использую текстовый редактор `gedit`. Далее вместо “Hello World!” пишу свое имя и фамилию. (рис. [5.2])

```

1 ; hello.asm
2 SECTION .data ; Начало секции данных
3     hello: DB 'Namruev Maxim',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. 5.2: Редактирование файла

3. Оттранслируйте полученный текст программы lab4.asm в объектный файл. Выполните компоновку объектного файла и запустите получившийся исполняемый файл

Далее компилирую текст программы в объектный файл и проверяю выполнились ли действия.(рис. [5.3])

```

msnamruev@dk2n25 ~/work/arch-pc/lab04 $ nasm -f elf lab04.asm
msnamruev@dk2n25 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o lab04.asm lab04.o list.lst main obj.o

```

Рис. 5.3: Компиляция файла

После этого передаю файл на обработку компоновщику ld.(рис. [5.4])

```

msnamruev@dk2n25 ~/work/arch-pc/lab04 $ ld -m elf_i386 lab04.o -o lab04
msnamruev@dk2n25 ~/work/arch-pc/lab04 $ ls
hello hello.asm hello.o lab04 lab04.asm lab04.o list.lst main obj.o

```

Рис. 5.4: Отправка файла на компоновку

Запускаю файл,чтобы убедиться, что всё выполнено правильно.(рис. [5.5])

```

msnamruev@dk2n25 ~/work/arch-pc/lab04 $ ./lab04
Namruev Maxim
msnamruev@dk2n25 ~/work/arch-pc/lab04 $

```

Рис. 5.5: Запуск файла

4. Скопируйте файлы `hello.asm` и `lab4.asm` в Ваш локальный репозиторий в каталог `~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/`. Загрузите файлы на Github.

Копирую файлы в свой локальный репозиторий с помощью команды `cp`, не забывая проверить это с помощью команды `ls`. (рис. [5.6])

```
msnamruev@dk2n25 ~/work/arch-pc/lab04 $ cp ~/work/arch-pc/lab04/hello ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/
msnamruev@dk2n25 ~/work/arch-pc/lab04 $ cp ~/work/arch-pc/lab04/lab4 ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/
msnamruev@dk2n25 ~/work/arch-pc/lab04 $ cd ~/work/study/2023-2024/“Архитектура компьютера”/arch-pc/labs/lab04/
msnamruev@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello lab04 presentation report
msnamruev@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

Рис. 5.6: Копирование файлов

Добавляю файлы на GitHub, используя команды `git add.`, `git commit -m` и `git push`. (рис. [5.7])

```
msnamruev@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ git add .
msnamruev@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ git commit -m
error: switch 'm' requires a value
msnamruev@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ git commit -m "Add existing files"
[master fe2e4dc] Add existing files
3 files changed, 64 insertions(+), 30 deletions(-)
create mode 100755 labs/lab04/hello
create mode 100755 labs/lab04/lab4
msnamruev@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ git push
Перечисление объектов: 100% (13/13), готово.
Подсчет объектов: 100% (13/13), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (8/8), готово.
Запись объектов: 100% (8/8), 3.96 КиБ | 3.96 МБ/с, готово.
Всего 8 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 3 local objects.
To github.com:msnamruev/study_2023-2024_arh-pc.git
415dc3c..fe2e4dc master -> master
```

Рис. 5.7: Отправка файлов

6 Выводы

После выполнения данной работы я освоил написание, сборку и компиляцию программ, написанных на ассемблере NASM.