

# **Отчёт по лабораторной работе №4**

**Дисциплина: Архитектура Компьютера**

София Андреевна Кудякова

# Содержание

|          |                                       |           |
|----------|---------------------------------------|-----------|
| <b>1</b> | <b>Цель работы</b>                    | <b>4</b>  |
| <b>2</b> | <b>Задание</b>                        | <b>5</b>  |
| <b>3</b> | <b>Теоретическое введение</b>         | <b>6</b>  |
| <b>4</b> | <b>Выполнение лабораторной работы</b> | <b>8</b>  |
| <b>5</b> | <b>Выводы</b>                         | <b>14</b> |
|          | <b>Список литературы</b>              | <b>15</b> |

## Список иллюстраций

|      |   |    |
|------|---|----|
| 4.1  | Перемещение в нужную директорию . . . . .                     | 8  |
| 4.2  | Создание файла и проверка . . . . .                           | 8  |
| 4.3  | Открытие файла . . . . .                                      | 9  |
| 4.4  | Ввод текста . . . . .   | 10 |
| 4.5  | Компиляция текста программы . . . . .                         | 10 |
| 4.6  | Компиляция текста программы . . . . .                         | 11 |
| 4.7  | Передача объектного файла на обработку компановщику . . . . . | 11 |
| 4.8  | Передача объектного файла на обработку компановщику . . . . . | 11 |
| 4.9  | Запуск исполняемого файла . . . . .                           | 12 |
| 4.10 | Создание копии файла . . . . .                                | 12 |
| 4.11 | Изменение текста . . . . .                                    | 12 |
| 4.12 | Компиляция текста программы . . . . .                         | 12 |
| 4.13 | Передача файла на обработку . . . . .                         | 13 |
| 4.14 | Запуск файла . . . . .  | 13 |
| 4.15 | Добавление файлов . . . . .                                   | 13 |
| 4.16 | Отправка файлов . . . . .                                     | 13 |

# 1 Цель работы

Цель данной работы - научиться работать с программами, написанными на ассемблере NASM, а именно - освоить процедуры компиляций и сборки программ.

## 2 Задание

1. Создание программы 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 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. Но 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. Программы, написанные на языке ассемблера, не уступают в качестве и скорости программам, написанным на машинном языке, ибо 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. Наиболее распространёнными ассемблерами для архитектуры x86 являются: 1) Для DOS/Windows: Borland Turbo Assembler (TASM), Microsoft Macro Assembler (MASM) и Watcom assembler (WASM). 2) Для GNU/Linux: gas (GNU Assembler), использующий AT&T-синтаксис, в отличие от большинства других популярных ассемблеров, которые используют Intel-синтаксис. Для записи команд в NASM используются: 1) Мнемокод — непосредственно мнемоника инструкции процессору, которая является обязательной частью команды. 2) Операнды — числа, данные, адреса регистров или адреса оперативной памяти. 3) Метка — идентификатор, с которым ассемблер ассоциирует некоторое число, чаще всего адрес в памяти. (Метка перед командой связана с адресом данной команды). Допустимыми символами в метках являются буквы, цифры, а также следующие символы: `,`, `$`, `#`, `@`, `~`, `.` и `?`. *Начинаться метка или идентификатор могут с буквы, `.`, и `?`.* Перед идентификаторами, которые пишутся как зарезервированные слова, нужно писать `$`, чтобы компилятор трактовал его верно (так называемое экранирование). Максимальная длина идентификатора составляет 4095 символов. Программа на языке ассемблера также может содержать директивы — инструкции, не переводящиеся непосредственно в машинные команды, а управляющие работой транслятора. Например, директивы используются для определения данных (констант и переменных) и обычно пишутся большими буквами.

## 4 Выполнение лабораторной работы

### 1. Программа Hello World!

Используя команду `cd` перемещаюсь в `lab05`. (рис. 4.1).

```
sakudyakova@dk2n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ cd labs/lab04
sakudyakova@dk2n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

Рис. 4.1: Перемещение в нужную директорию

Используя утилиту `touch`, создаю пустой текстовый файл `hello.asm` и проверяю корректность выполненных действий с помощью `ls`. (рис. 4.2).

```
sakudyakova@dk2n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ touch hello.asm
sakudyakova@dk2n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello.asm  presentation  report
sakudyakova@dk2n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ █
```

Рис. 4.2: Создание файла и проверка

Далее созданный файл открываю в текстовом редакторе `gedit`. (рис. 4.3).



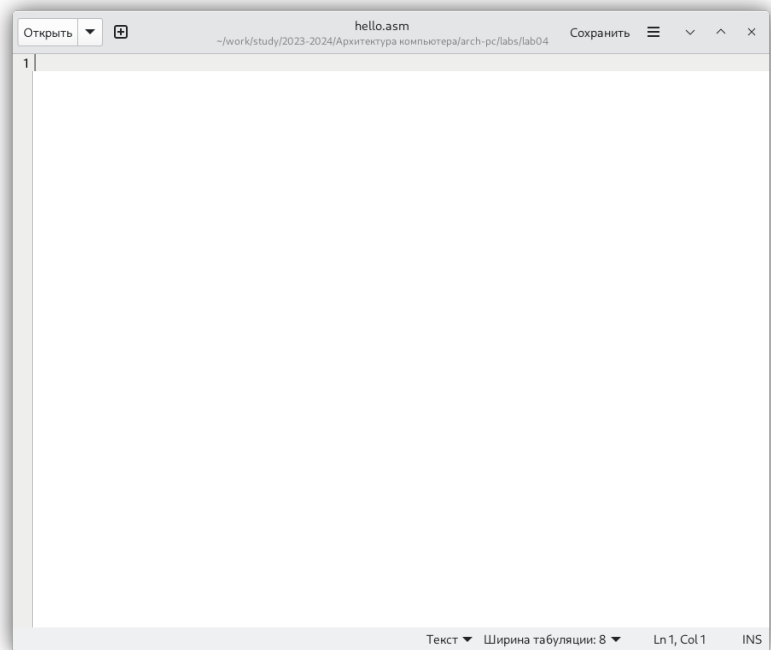


Рис. 4.3: Открытие файла

Заполняю файл текстом, нужным для вывода “Hello World!”. (рис. 4.4).

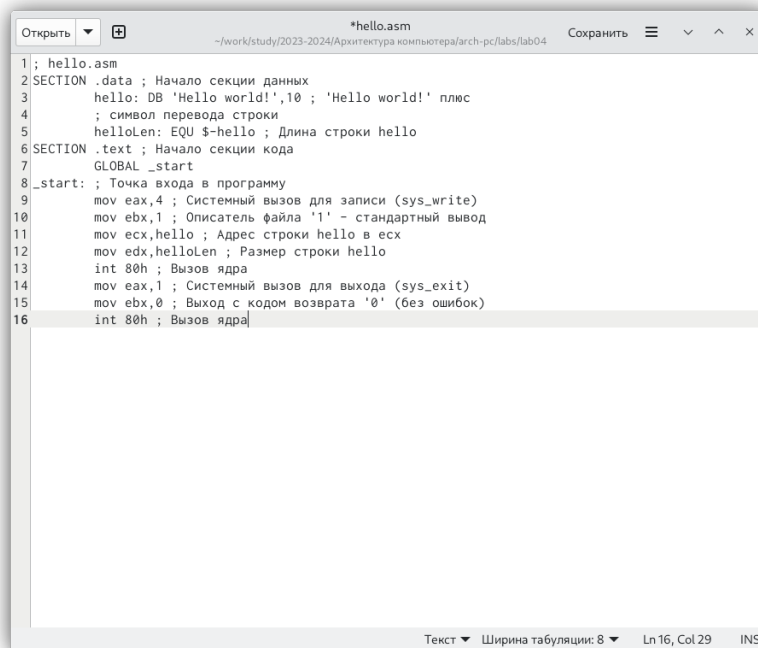


Рис. 4.4: Ввод текста

## 2. Работа с транслятором NASM

NASM превращает текст программы в объектный код. Для приведённого выше текста программы “Hello World” выполнил компиляцию, используя команду `nasm -f elf hello.asm`, ключ `-f elf` указывает транслятору `nasm`, что необходимо создать бинарный файл в формате ELF. Затем проверяю корректность выполненных действий с помощью `ls`. (рис. 4.5).

```

sakudyakova@dk2n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ nasm -f elf hello.asm
sakudyakova@dk2n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello.asm hello.o presentation report
sakudyakova@dk2n22 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ █

```

Рис. 4.5: Компиляция текста программы

После проверки убеждаюсь, что был создан файл “hello.o”

## 3. Работа с расширенным синтаксисом командной строки NASM

Ввожу команду, которая скомпилирует файл `hello.asm` в файл `obj.o`, опция `-o` позволяет задать имя объектного файла, в данном случае `obj.o`, при этом формат выходного файла будет `elf` и в файл будут включены символы для отладки (ключ `-g`), также с помощью ключа `-l` будет создан файл листинга `list.lst`. Проверяю правильность выполнения команды с помощью команды `ls`. (рис. 4.6).

```
sakudyakova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
sakudyakova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello.asm hello.o list.lst obj.o presentation report
sakudyakova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

Рис. 4.6: Компиляция текста программы

#### 4. Работа с компоновщиком LD

Передаю объектный файл `hello.o` на обработку компоновщику `LD`, чтобы получить исполняемый файл `hello`. Ключ `-o` задает имя создаваемого исполняемого файла. С помощью команды `ls` убеждаюсь, что исполняемый файл `hello` был создан. (рис. 4.7).

```
sakudyakova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ld -m elf_i386 hello.o -o hello
sakudyakova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello hello.asm hello.o list.lst obj.o presentation report
sakudyakova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

Рис. 4.7: Передача объектного файла на обработку компоновщику

Выполняю следующую команду: `ld -m elf_i386 obj.o -o main`. Так как я использовала ключ `-o`, которое задавало значение `main`, исполняемый файл будет иметь имя `main`. Объектный файл, из которого собран этот исполняемый файл, имеет имя `obj.o` (рис. 4.8)

```
sakudyakova@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ld -m elf_i386 obj.o -o main
sakudyakova@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello hello.asm hello.o list.lst main obj.o presentation report
```

Рис. 4.8: Передача объектного файла на обработку компоновщику

#### 5. Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл `hello`, находящийся в текущем каталоге. (рис. 4.9).

```
sakudyakova@dk3n35 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ./hello
Hello world!
```

Рис. 4.9: Запуск исполняемого файла

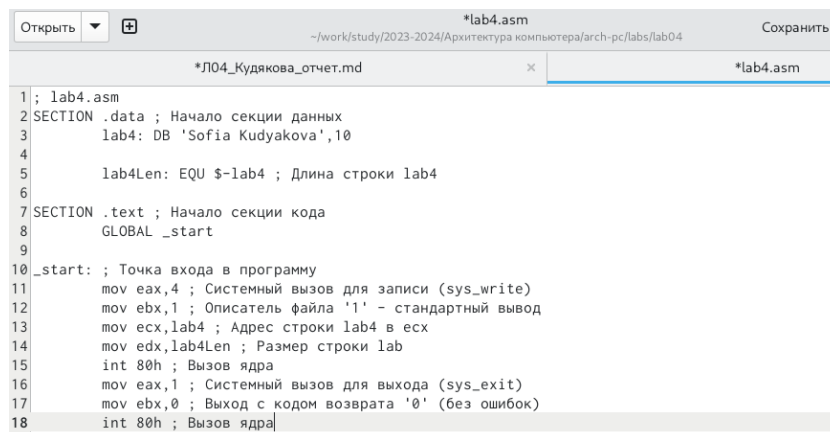
## 6. Выполнение заданий для самостоятельной работы

С помощью команды `cp` создаю в текущем каталоге копию файла `hello.asm` с именем `lab4.asm`. (рис. 4.10).

```
sakudyakova@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ cp hello.asm lab4.asm
sakudyakova@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

Рис. 4.10: Создание копии файла

С помощью текстового редактора `gedit` открываю файл `lab4.asm` и вношу изменения в программу так, чтобы она выводила мое имя и фамилию. (рис. 4.11)



```
1 ; lab4.asm
2 SECTION .data ; Начало секции данных
3     lab4: DB 'Sofia Kudyakova',10
4
5     lab4Len: EQU $-lab4 ; Длина строки lab4
6
7 SECTION .text ; Начало секции кода
8     GLOBAL _start
9
10 _start: ; Точка входа в программу
11     mov eax,4 ; Системный вызов для записи (sys_write)
12     mov ebx,1 ; Описатель файла '1' - стандартный вывод
13     mov ecx,lab4 ; Адрес строки lab4 в есх
14     mov edx,lab4Len ; Размер строки lab
15     int 80h ; Вызов ядра
16     mov eax,1 ; Системный вызов для выхода (sys_exit)
17     mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
18     int 80h ; Вызов ядра
```

Рис. 4.11: Изменение текста

Компилирую текст программы в объектный файл. (рис. 4.12)

```
sakudyakova@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ nasm -f elf lab4.asm
sakudyakova@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

Рис. 4.12: Компиляция текста программы

Передаю файл `lab4.asm`, на обработку компоновщику `LD`, чтобы получить исполняемый файл. (рис. 4.13)

```
sakudyakova@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ld -m elf_i386 lab4.o -o la
sakudyakova@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o presentation_report
```

Рис. 4.13: Передача файла на обработку

Запускаю исполняемый файл lab4.asm. Программа сработала корректно, на экран действительно выводятся мои имя и фамилия.(рис. 4.14)

```
sakudyakova@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ./lab4
Sofia Kudyakova
```

Рис. 4.14: Запуск файла

Добавляю файлы в GitHub, используя команду git add . и git commit, комментируя свое действие как New files for lab 4. (рис. 4.15)

```
sakudyakova@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ git add .
sakudyakova@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ git commit -m "New files for lab4"
[master 234c1aa] New files for lab4
26 files changed, 232 insertions(+)
```

Рис. 4.15: Добавление файлов

Отправляю файлы на сервер с помощью команды git push.(рис. 4.16)

```
sakudyakova@dk2n25 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ git push
Перечисление объектов: 41, готово.
Подсчет объектов: 100% (41/41), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (33/33), готово.
Запись объектов: 100% (33/33), 446.13 Киб | 3.54 Миб/с, готово.
Всего 33 (изменений 9), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (9/9), completed with 5 local objects.
To github.com:sakudyakova/study_2023-2024_arh-pc.git
 32a3e65..234c1aa master -> master
```

Рис. 4.16: Отправка файлов

## **5 Выводы**

В ходе данной лабораторной работы я научилась работать с программами, написанными на ассемблере NASM, а именно - смогла освоить процедуры компиляции и сборки программ.

# Список литературы

Архитектура ЭВМ