

Отчёт по лабораторной работе №4

**Создание и процесс обработки программ на языке ассемблера
NASM**

Скобеева Алиса Алексеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выполнение самостоятельной работы	9
5	Выводы	11

Список иллюстраций

3.1	Последовательно выполняем программы, проверяем наличие файла и открываем его	7
3.2	Внимательно переписываем текст в файл	7
3.3	Проверяем, чтобы объектный файл был создан	7
3.4	Проверяем наличие файлов	8
3.5	С помощью команды ls проверяем, чтобы исполняемый файл hello был создан	8
3.6	Проверяем наличие исполняемого файла	8
3.7	Вводим команду ./hello	8
4.1	Создаем и запускаем файл	9
4.2	Вписываем имя и фамилию	9
4.3	Последовательно выполняем команды	10
4.4	Вводим команды	10
4.5	Загружаем в удалённый репозиторий	10
4.6	Вводим команды	10
4.7	Загружаем файлы в репозиторий	10

Список таблиц

1 Цель работы

Освоить процедуры компиляции и сборки программ, познакомиться с языком ассемблера NASM.

2 Задание

Написать 2 программы: “Hello world”; “Имя Фамилия”

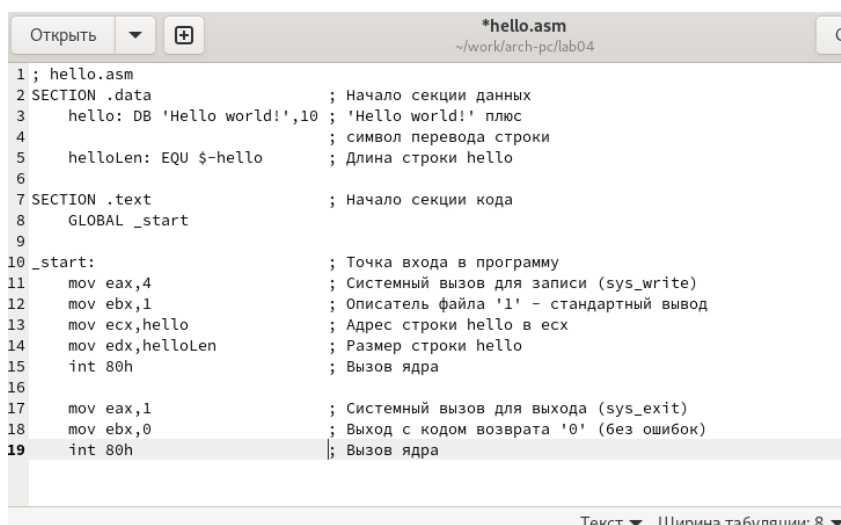
3 Выполнение лабораторной работы

Создаем каталог для работы с программами на языке ассемблера NASM, переходим в него и создаем текстовый файл:

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

Рис. 3.1: Последовательно выполняем программы, проверяем наличие файла и открываем его

После того, как открыли файл, вводим в него текст как в примере:



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
7 SECTION .text                ; Начало секции кода
8     GLOBAL _start
9
10 _start:
11     mov eax,4                ; Системный вызов для записи (sys_write)
12     mov ebx,1                ; Описатель файла '1' - стандартный вывод
13     mov ecx,hello            ; Адрес строки hello в ecx
14     mov edx,helloLen         ; Размер строки hello
15     int 80h                  ; Вызов ядра
16
17     mov eax,1                ; Системный вызов для выхода (sys_exit)
18     mov ebx,0                ; Выход с кодом возврата '0' (без ошибок)
19     int 80h                  ; Вызов ядра
```

Рис. 3.2: Внимательно переписываем текст в файл

Вводим команду для превращения текста в объектный код:

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

Рис. 3.3: Проверяем, чтобы объектный файл был создан

Вводим команду, которая компилирует исходный файл hello.asm в obj.o:

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

Рис. 3.4: Проверяем наличие файлов

Передаем объектный файл на обработку компоновщику:

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

Рис. 3.5: С помощью команды ls проверяем, чтобы исполняемый файл hello был создан

Выполняем следующую команду:

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

Рис. 3.6: Проверяем наличие исполняемого файла

Запускаем на выполнение созданный исполняемый файл:

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

Рис. 3.7: Вводим команду ./hello

4 Выполнение самостоятельной работы

В нужном каталоге создаем копию файла `hello.asm` с именем `lab4.asm`:

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

Рис. 4.1: Создаем и запускаем файл

Открытый файл редактируем в соответствии с заданием:

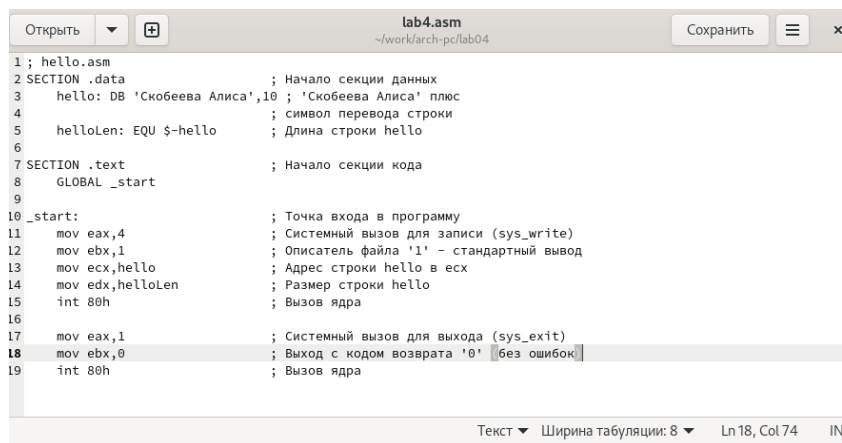


Рис. 4.2: Вписываем имя и фамилию

Вводим необходимые команды для превращения текста в объектный код и превращения файла в объектный файл; выполняем компоновку объектного файла и запускаем получившийся исполняемый файл:

```
askobeeva@fedora: ~/work/arch-pc/lab04$ nasm -f elf lab4.asm
askobeeva@fedora: ~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst lab4.asm
askobeeva@fedora: ~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4.asm list.lst main obj.o
askobeeva@fedora: ~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o hello
```

Рис. 4.3: Последовательно выполняем команды

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

Рис. 4.4: Вводим команды

С помощью команды `./hello` запускаем на выполнение исполняемый файл:

```
askobeeva@fedora: ~/work/arch-pc/lab04$ ./hello
Скобеева Алиса
```

Рис. 4.5: Загружаем в удалённый репозиторий

Копируем файлы `hello.asm` и `lab4.asm` в необходимый каталог:

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

Рис. 4.6: Вводим команды

Загружаем файлы на Github:

```
askobeeva@fedora: ~/work/arch-pc/lab04$ cd ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc
askobeeva@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git add .
askobeeva@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git commit -am 'feat(main): add fi
les lab-4'
[main 4a86ef0] feat(main): add files lab-4
5 files changed, 282 insertions(+), 74 deletions(-)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
delete mode 100644 labs/lab04/report/image/placeimg_800_600_tech.jpg
askobeeva@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git push
Перечисление объектов: 100% (11/11), готово.
Подсчет объектов: 100% (16/16), готово.
Сжатие объектов: 100% (11/11), готово.
Запись объектов: 100% (11/11), 8.91 КиБ | 1.78 МБ/с, готово.
Total 11 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 3 local objects.
To github.com:skalisa/study_2024-2025_arch-pc.git
864e295..4a86ef0 master -> master
```

Рис. 4.7: Загружаем файлы в репозиторий

5 Выводы

После выполнения данной самостоятельной работы я познакомилась с языком ассемблера NASM и научилась создавать работающие программы.