

Отчёта по лабораторной работе №4

Создание и процесс обработки программ на языке ассемблера NASM

Гурбанов Сарча Оразмаммедович

Содержание

1	<u>Цель работы</u>	4
2	<u>Задание</u>	5
3	<u>Выполнение лабораторной работы</u>	6
1.	<u>Программа Hello world!</u>	6
2.	<u>Транслятор NASM</u>	7
3.	<u>Расширенный синтаксис командной строки NASM</u>	7
4.	<u>Компоновщик LD</u>	8
5.	<u>Запуск исполняемого файла</u>	9
6.	<u>Задание для самостоятельной работы</u>	9
4	<u>Выводы</u>	11

Список иллюстраций

1.	<u>Создаем каталоги с помощью команды mkdir</u>	6
2.	<u>Переходим в каталог с помощью команды cd</u>	6
3.	<u>Создаем текстовый файл hello.asm</u>	6
4.	<u>Открываем файл и заполняем его по примеру</u>	7
5.	<u>Используем команду nasm</u>	7
6.	<u>Проверяем работу команды</u>	7
7.	<u>Преобразуем файл hello.asm в obj.o</u>	7
8.	<u>Проверяем создание файла командой ls</u>	8
9.	<u>Используем команду ld</u>	8
10.	<u>Используем команду ls</u>	8
11.	<u>Используем команду ld, создавая файл main</u>	8
12.	<u>Используем команду ls</u>	8
13.	<u>Используем команду ./hello</u>	9
14.	<u>Используем команду cp</u>	9
15.	<u>Открываем файл в текстовом редакторе</u>	9
16.	<u>Редактируем файл для своего имени и фамилии</u>	9
17.	<u>Прописываем команды для работы файла и запускаем программу</u>	10
18.	<u>Копируем файлы в каталог с ЛР4</u>	10
19.	<u>Загружаем файлы</u>	10

1 Цель работы

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

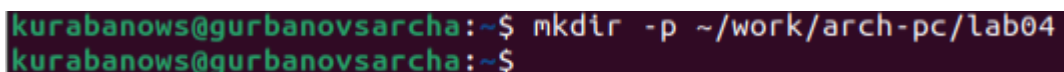
2 Задание

Написать 2 программы(Hello world, lab4(Имя Фамилия))

3 Выполнение лабораторной работы

3.1 Программа Hello world!

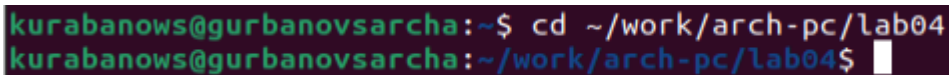
Создаем каталог для работы с программами на языке ассемблера NASM (рис. [3.1](#)).



```
kurabanows@gurbanovsarcha:~$ mkdir -p ~/work/arch-pc/lab04
kurabanows@gurbanovsarcha:~$
```

Рис. 3.1: Создаем каталоги с помощью команды mkdir

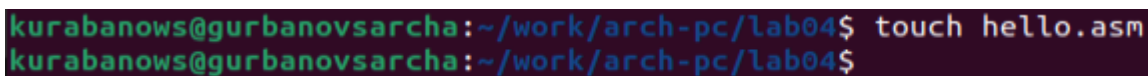
Переходим в созданный каталог (рис. [3.2](#)).



```
kurabanows@gurbanovsarcha:~$ cd ~/work/arch-pc/lab04
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$
```

Рис. 3.2: Переходим в каталог с помощью команды cd

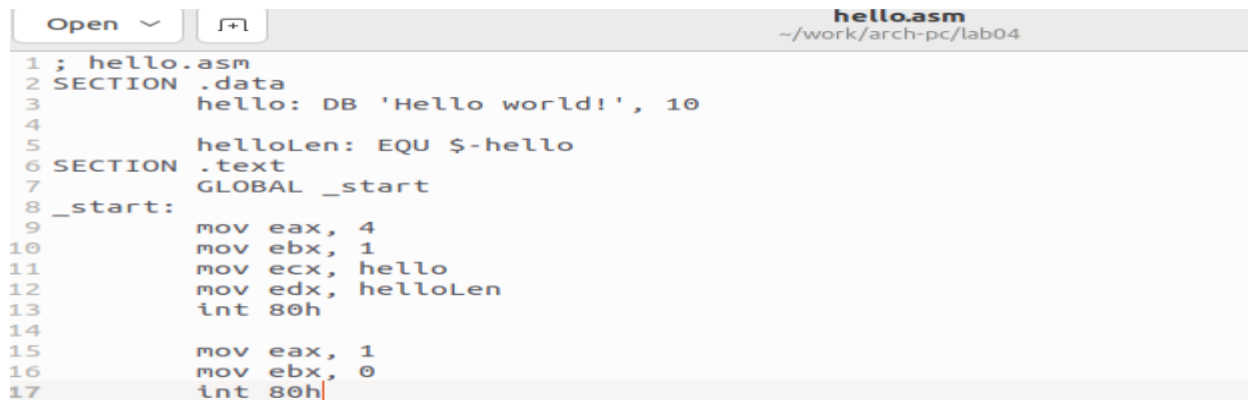
Создаем текстовый файл (рис. [3.3](#)).



```
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$ touch hello.asm
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$
```

Рис. 3.3: Создаем текстовый файл hello.asm

Открываем данный файл в текстовом редакторе (рис. [3.4](#)).



```
1 ; hello.asm
2 SECTION .data
3     hello: DB 'Hello world!', 10
4
5     helloLen: EQU $-hello
6 SECTION .text
7     GLOBAL _start
8 _start:
9     mov eax, 4
10    mov ebx, 1
11    mov ecx, hello
12    mov edx, helloLen
13    int 80h
14
15    mov eax, 1
16    mov ebx, 0
17    int 80h
```

Рис. 3.4: Открываем файл и заполняем его по примеру

3.2 Транслятор NASM

Преобразуем текст программы в объектный код (рис. 3.5).

```
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$ nasm -f elf hello.asm
```

Рис. 3.5: Используем команду nasm

Проверяем созданся ли объектный файл с помощью команды ls (рис. 3.6).

```
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$
```

Рис. 3.6: Проверяем работу команды

3.3 Расширенный синтаксис командной строки NASM

Компилируем исходный файл (рис. 3.7).

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

Рис. 3.7: Преобразуем файл hello.asm в obj.o

Проверяем, как сработала команда (рис. 3.8).

```
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$ ls  
hello.asm hello.o
```

Рис. 3.8: Проверяем создание файла командой ls

3.4 Компоновщик LD

Передаем объектный файл на обработку компоновщику (рис. 3.9).

```
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o hello
```

Рис. 3.9: Используем команду ld

Проверяем созданся ли исполняемый файл hello (рис. 3.10).

```
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$ ls  
hello.asm hello.o list.lst obj.o  
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$
```

Рис. 3.10: Используем команду ls

Передаем объектный файл на обработку компоновщику (рис. 3.11).

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

Рис. 3.11: Используем команду ld, создавая файл main

Проверяем созданся ли исполняемый файл hello (рис. 3.12).

```
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$ ls  
hello.asm hello.o list.lst main obj.o  
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$
```

Рис. 3.12: Используем команду ls

3.5 Запуск исполняемого файла

Запускаем на выполнение созданный исполняемый файл (рис. 3.13).

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

Рис. 3.13: Используем команду ./hello

3.6 Задание для самостоятельной работы

Создаем копию файла hello.asm (рис. 3.14).


```
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$
```

Рис. 3.14: Используем команду cp

```
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$ gedit lab4.asm
```

Открываем файл и редактируем его (рис. 3.15).

Рис. 3.15: Открываем файл в текстовом редакторе



```
1 ; hello.asm
2 SECTION .data
3     hello: DB 'Гурбанов Сарча', 10
4
5     helloLen: EQU $-hello
6 SECTION .text
7     GLOBAL _start
8 _start:
9         mov eax, 4
10        mov ebx, 1
11        mov ecx, hello
12        mov edx, helloLen
13        int 80h
14
15        mov eax, 1
16        mov ebx, 0
17        int 80h
```

Рис. 3.16: Редактируем файл для своего имени и фамилии

Прописываем те же команды, что и с первой программой (рис. 3.17).

```

kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst lab4.asm
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o hello
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o mail
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$ ./hello
Гурбанов Сарча
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$

```

Рис. 3.17: Прописываем команды для работы файла и запускаем программу

Копируем файлы в локальный репозиторий (рис. 3.18).

```

kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$ cp hello.asm ~/work/study/2023-2024/Архитектура\ комп
ьютера/arch-pc/labs/lab04
kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$

```

Рис. 3.18: Копируем файлы в каталог с ЛР4

Переходим в каталог лабораторных работ и загружаем файлы на Github (рис. 3.19).

```

kurabanows@gurbanovsarcha:~/work/arch-pc/lab04$ cd ~/work/study/2023-2024/Архитектура\ компьютера/arch-
h-pc/
kurabanows@gurbanovsarcha:~/work/study/2023-2024/Архитектура\ компьютера/arch-pc$ git add .
kurabanows@gurbanovsarcha:~/work/study/2023-2024/Архитектура\ компьютера/arch-pc$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   labs/lab04/hello.asm

kurabanows@gurbanovsarcha:~/work/study/2023-2024/Архитектура\ компьютера/arch-pc$ git commit -am "feat
(main): add files lab-4"
[master 1e4edee] feat(main): add files lab-4
1 file changed, 17 insertions(+)
 create mode 100644 labs/lab04/hello.asm
kurabanows@gurbanovsarcha:~/work/study/2023-2024/Архитектура\ компьютера/arch-pc$ git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 10 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 590 bytes | 295.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:GurbanowS/study_2023-2024_arh-pc.git
 491ab6e..1e4edee master -> master
kurabanows@gurbanovsarcha:~/work/study/2023-2024/Архитектура\ компьютера/arch-pc$ s

```

Рис. 3.19: Загружаем файлы

4 Выводы

Мы познакомились с языком ассемблера NASM и создали две работающих программы.