

# Отчёт по лабораторной работе №4

## Дисциплина: архитектура компьютера

Ларина Наталья Денисовна

### Содержание

1	Цель работы .....	1
2	Задание.....	1
3	Теоретическое введение .....	2
4	Выполнение лабораторной работы .....	2
4.1	Создание программы "Hello world!" .....	2
4.2	Работа с транслятором NASM.....	3
4.3	Работа с расширенным синтаксисом командой строки NASM.....	3
4.4	Работа с компоновщиками LD .....	3
4.5	Запуск исполняемого файла .....	4
4.6	Выполнение заданий для самостоятельной работы .....	4
5	Выводы.....	5
	Список литературы.....	5

### 1 Цель работы

Целью данной лабораторной работы является освоение процедуры и сборки программ, написанных на ассемблере NASM.

### 2 Задание

1. Создание программы Hello, world!
2. Работа с транслятором NASM
3. Работа с расширенным синтаксисом командой строки NASM
4. Работа с компоновщиками LD
5. Запуск исполняемого файла
6. Выполнение заданий для самостоятельной работы

### 3 Теоретическое введение

Основные принципы работы компьютера Основными функциональными элементами любой электронно-вычислительной машины (ЭВМ) являются центральный процессор, память и периферийные устройства (рис. 4.1). Взаимодействие этих устройств осуществляется через общую шину, к которой они подклю- чены. Физически шина представляет собой большое количество проводников, соединяющих устройства друг с другом. В современных компьютерах проводники выполнены в виде элек- тропроводящих дорожек на материнской (системной) плате. Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора (ЦП) входят следующие устройства: • арифметико-логическое устройство (АЛУ) — выполняет логические и арифметиче- ские действия, необходимые для обработки информации, хранящейся в памяти; • устройство управления (УУ) — обеспечивает управление и контроль всех устройств компьютера; • регистры — сверхбыстрая оперативная память небольшого объёма, входящая в со- став процессора, для временного хранения промежуточных результатов выполнения инструкций; регистры процессора делятся на два типа: регистры общего назначения и специальные регистры. Для того, чтобы писать программы на ассемблере, необходимо знать, какие регистры процессора существуют и как их можно использовать. Большинство команд в программах написанных на ассемблере используют регистры в качестве операндов. Практически все команды представляют собой преобразование данных хранящихся в регистрах процессора, это например пересылка данных между регистрами или между регистрами и памятью, пре- образование (арифметические или логические операции) данных хранящихся в регистрах.

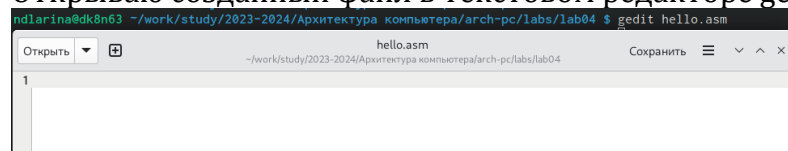
## 4 Выполнение лабораторной работы

### 4.1 Создание программы “Hello world!”

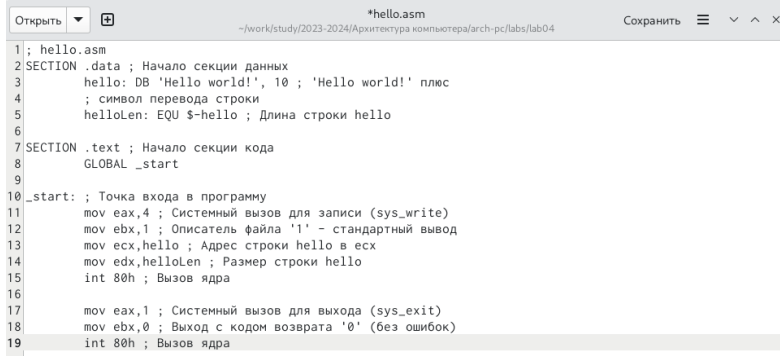
С помощью утилиты `cd` перемещаюсь в каталог, в котором буду выполнять работу, и создаю в текущем каталоге пустой текстовый файл `hello.asm` с помощью утилиты `touch` (рис. [??]).

```
ndlarina@dk8n63 ~ $ cd work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ touch hello.asm
```

Открываю созданный файл в текстовом редакторе `gedit` (рис. [??]).



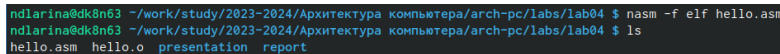
Заполняю файл, вставляя в него программу “Hello world!” (рис. [??]).



```
1; hello.asm
2SECTION .data ; Начало секции данных
3hello: DB 'Hello world!', 10 ; 'Hello world!' плюс
4        ; символ перевода строки
5helloLen: EQU $-hello ; Длина строки hello
6
7SECTION .text ; Начало секции кода
8GLOBAL _start
9
10_start: ; Точка входа в программу
11mov eax,4 ; Системный вызов для записи (sys_write)
12mov ebx,1 ; Описатель файла '1' - стандартный вывод
13mov ecx,hello ; Адрес строки hello в ecx
14mov edx,helloLen ; Размер строки hello
15int 80h ; Вызов ядра
16
17mov eax,1 ; Системный вызов для выхода (sys_exit)
18mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
19int 80h ; Вызов ядра
```

## 4.2 Работа с транслятором NASM

Превращаю текст программы для вывода “Hello world!” в объектный код с помощью транслятора NASM, используя команду `nasm -f elf hello.asm`, ключ `-f ELF` (рис. [??]). Далее проверяю правильность выполнения команды с помощью утилиты `ls`.

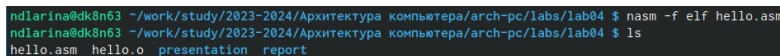


```
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ nasm -f elf hello.asm
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello.asm  hello.o  presentation  report
```

*Компиляция текста программы*

## 4.3 Работа с расширенным синтаксисом командой строки NASM

Сначала ввожу команду, которая скомпилирует файл `hello.asm` в файл `obj.o`, при этом в файл будут включены символы для отладки (ключ `-g`), также с помощью ключа `-l` будет создан файл листинга `list.lst` (рис. [??]). И проверяю правильность выполнения команды, используя утилиту `ls`.

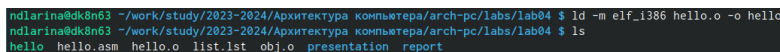


```
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ nasm -f elf hello.asm
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello.asm  hello.o  presentation  report
```

*Компиляция текста программы*

## 4.4 Работа с компоновщиками LD

Передаю объектный файл `hello.o` на обработку компоновщику LD, чтобы получить исполняемый файл `hello` (рис. [??]). Ключ `-o` задаёт имя создаваемого исполняемого файла. затем проверяю правильность выполнения команды с помощью `ls`.



```
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ld -m elf_i386 hello.o -o hello
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello  hello.asm  hello.o  list.lst  obj.o  presentation  report
```

*Передача объектного файла на обработку компоновщику*

Выполняю следующую команду (рис. [??]). Исполняемый файл будет иметь имя `main`, так как после ключа `-o` было задано значение `main`. Объектный файл, из которого собран этот исполняемый файл, имеет имя `obj.o`.

```
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ld -m elf_i386 obj.o -o main
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello hello.asm hello.o list.lst main obj.o presentation report
```

*Передача объектного файла на обработку компоновщику*

## 4.5 Запуск исполняемого файла

Запускаю на выполнение созданный исполняемый файл hello (рис. [??]).

```
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ./hello
Hello world!
```

## 4.6 Выполнение заданий для самостоятельной работы

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

```
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ cp hello.asm lab4.asm
```

*Создание копии файла*

Далее с помощью текстового редактора `gedit` открываю файл `lab4.asm` и вношу изменения в программу так, чтобы она выводила моё имя и фамилию (рис. [??]).

```
1 ; lab4.asm
2 SECTION .data ; Начало секции данных
3     lab4: DB 'Natalia Larina', 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 ecx
14     mov edx,lab4Len ; Размер строки lab4
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
```

*Изменение программы*

Компилирую текст программы в объектный файл (рис. [??]). И проверяю правильность выполнения команды с помощью утилиты `ls`.

```
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ nasm -f elf lab4.asm
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello hello.asm hello.o lab4.asm lab4.o list.lst main obj.o presentation report
```

*Компиляция текста программы*

Передаю объектный файл `lab4.o` на обработку компоновщику `LD`, чтобы получить исполняемый файл `lab4` (рис. [??]).

```
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ld -m elf_i386 lab4.o -o lab4
ndlarina@dk8n63 ~/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
```

*Передача объектного файла на обработку компоновщику*

Запускаю исполняемый файл lab4. Всё работает (рис. [??]).

```
ndlarina@dk8n63 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ./lab4
Natalia Larina
```

### *Запуск исполняемого файла*

Далее с помощью команд `git add`, `git commit` и `git push` добавляю файлы на github, комментируя действие как добавление файлов для лабораторной работы №4, и затем отправляю файлы на сервер (рис. [??]).

```
ndlarina@dk8n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ git add .
ndlarina@dk8n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ git commit -m "Add fales for lab04"
Текущая ветка: master
Ваша ветка опережает «origin/master» на 1 коммит.
(используйте «git push», чтобы опубликовать ваши локальные коммиты)

Неотслеживаемые файлы:
(используйте «git add <файл>...», чтобы добавить в то, что будет включено в коммит)
.. /lab02/report/report.docx
.. /lab02/report/r02_Ларина_отчет.odf
.. /lab02/report/r02_Ларина_отчет.pdf
.. /lab02/report/r02_Ларина_отчет.docx
.. /lab02/report/r02_Ларина_отчет.md
.. /lab03/report/r03_Ларина_отчет.docx
.. /lab03/report/r03_Ларина_отчет.md

индекс пуст, но есть неотслеживаемые файлы
(используйте «git add», чтобы проиндексировать их)
ndlarina@dk8n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ git push
Перечисление объектов: 16, готово.
Подсчет объектов: 100% (16/16), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (13/13), готово.
Запись объектов: 100% (13/13), 3.41 КБ | 3.41 МБ/с, готово.
Всего 13 (изменений 7), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (7/7), completed with 2 local objects.
To github.com:ndlarina/study_2023-2024_arhpc.git
3b94c83..bebf582 master -> master
```

### *Добавление и отправка файлов на GitHub*

## 5 Выводы

В результате выполнения лабораторной работы мне удалось освоить процедуры компиляции и сборки программ, написанных на ассемблере NASM.

## Список литературы

1. <https://esystem.rudn.ru/mod/resource/view.php?id=1030552>