

Лабораторная работа №4

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

Коровкин Никита Михайлович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение самостоятельной работы	9
4	Выводы	12

Список иллюстраций

2.1	Переход в каталог lab4	6
2.2	Создаем hello.asm	6
2.3	Файл,открытый с помощью gedit	6
2.4	Вставляем код для программы Hello World	7
2.5	Делаем файл объектным	7
2.6	Используем ls	7
2.7	Используем полный вариант команды Nasm	7
2.8	Создаем исполняемый файл	8
2.9	Собираем файл main	8
2.10	Запуск файла	8
3.1	Копируем файл	9
3.2	Открываем файл	9
3.3	Редактируем файл	10
3.4	Компилируем и собираем нужный файл	10
3.5	Запуск файла	10
3.6	Копируем файлы в нужный каталог	11
3.7	Загрузка файлов на сервер	11

Список таблиц

1 Цель работы

Научиться писать базовые программы на языке ассемблера NASM, компилировать их в объектные файлы и собирать из них исполняемые программы с помощью компоновщика.

2 Выполнение лабораторной работы

Сперва перейдем в нужную директорию.(рис.1)

```
[nikitak@fedora report]$ cd ~/study_2024-2025_arh-pc/labs/lab04  
[nikitak@fedora lab04]$
```

Рис. 2.1: Переход в каталог lab4

Теперь создадим файл нужного формата.(рис.2)

```
[nikitak@fedora lab04]$ touch hello.asm  
[nikitak@fedora lab04]$
```

Рис. 2.2: Создаем hello.asm

Откроем созданный нами файл.(рис.3)

```
[nikitak@fedora lab04]$ gedit hello.asm  
█
```

Рис. 2.3: Файл,открытый с помощью gedit

Отредактируем файл, вставив необходимый код.(рис.4)

```

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

```

Рис. 2.4: Вставляем код для программы Hello World

Отредактировав, скомпилируем код, Но для начала превратим файл в объектный при помощи Nasm.(рис.5)

```

[nikitak@fedora lab04]$ nasm -f elf hello.asm
[nikitak@fedora lab04]$

```

Рис. 2.5: Делаем файл объектным

Проверим, все ли мы сделали правильно с помощью ls.(рис.5)

```

[nikitak@fedora lab04]$ nasm -f elf hello.asm
[nikitak@fedora lab04]$

```

Рис. 2.6: Используем ls

Теперь воспользуемся полным вариантом команды Nasm, где укажем, что файл hello.asm должен быть скомпилирован в файл с названием obj.o (рис.6)

```

[nikitak@fedora lab04]$ ls
hello.asm hello.o presentation report
[nikitak@fedora lab04]$

```

Рис. 2.7: Используем полный вариант команды Nasm

Дальше воспользуемся компоновщиком, который соберет объектный файл.(рис.7)

!Используем команду и делаем проверку(image/7.png)

Пропишем команду для создания исполняемого файла.(рис.8)

```
liveuser@localhost-live:~$ ld -m elf_i386 hello.o -o hello
liveuser@localhost-live:~$
```

Рис. 2.8: Создаем исполняемый файл

Затем соберем файл obj.o в файл main.(рис.9)

```
liveuser@localhost-live:~$ ld -m elf_i386 obj.o -o main
liveuser@localhost-live:~$
```

Рис. 2.9: Собираем файл main

Теперь запустим файл hello, написав ./ и название файла. Посмотрим, работает ли команда.(рис.10)

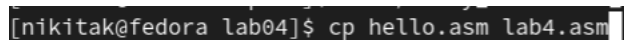
```
Hello world!
liveuser@localhost-live:~$
```

Рис. 2.10: Запуск файла

Мы получили Hello world, значит вся работа сделана правильно.

3 Выполнение самостоятельной работы

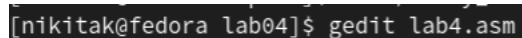
Теперь сделаем копию файла hello.asm и назовем lab4.asm.(рис.11)



```
[nikitak@fedora lab04]$ cp hello.asm lab4.asm
```

Рис. 3.1: Копируем файл

Так же откроем его с помощью gedit.(рис.12)



```
[nikitak@fedora lab04]$ gedit lab4.asm
```

Рис. 3.2: Открываем файл

Теперь в самом редакторе заменим Hello world на имя и фамилию.(рис.13)

```

1 ; hello.asm
2 SECTION .data ; Начало секции данных
3 hello: DB 'Korovkin Nikita',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. 3.3: Редактируем файл

Затем по тому же принципу скомпилируем и соберем файл.(рис.14)

```

liveuser@localhost-live:~$ nasm -f elf lab4.asm
liveuser@localhost-live:~$ ld -m elf_i386 lab4.o -o lab4

```

Рис. 3.4: Компилируем и собираем нужный файл

Пропишем команду, чтобы запустить файл и проверить его работу.(рис.15)

```

liveuser@localhost-live:~$ ./lab4
Korovkin Nikita

```

Рис. 3.5: Запуск файла

Все работает верно. Остается только скопировать оба файла и перенести в нужный каталог(рис.16)

```
[nikitak@fedora study_2024-2025_arh-pc]$ cp hello.asm labs/lab04
[nikitak@fedora study_2024-2025_arh-pc]$ cp lab4.asm labs/lab04
[nikitak@fedora study_2024-2025_arh-pc]$
```

Рис. 3.6: Копируем файлы в нужный каталог

Загрузим файлы на сервер. Самостоятельная работа сделана верно. (рис. 17)

```
delete mode 100644 labs/lab04/hero1 c/ image/rtase img_800_800_tesp.jpg
[nikitak@fedora lab04]$ git push
Перечисление объектов: 43, готово.
Подсчет объектов: 100% (43/43), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (33/33), готово.
Запись объектов: 100% (33/33), 393.43 КИБ | 4.57 МИБ/с, готово.
Всего 33 (изменений 9), повторно использовано 0 (изменений 0), повторно
вано пакетов 0
remote: Resolving deltas: 100% (9/9), completed with 7 local objects.
To github.com:Pancakeboy1987/study_2024-2025_arh-pc.git
   3bcf2c9..e33f782  master -> master
[nikitak@fedora lab04]$
```

Рис. 3.7: Загрузка файлов на сервер

4 Выводы

В ходе выполнения лабораторной работы мы получили знания о том, как работает алгоритм создания исполняемого файла из кода на ассемблере, а также приобрели навыки работы с языком `asm`.