

# **Лабораторная работа №5**

**Дисциплина: Архитектура компьютера**

**Кондратьев Арсений Вячеславович**

**03.10.2022**

# Содержание

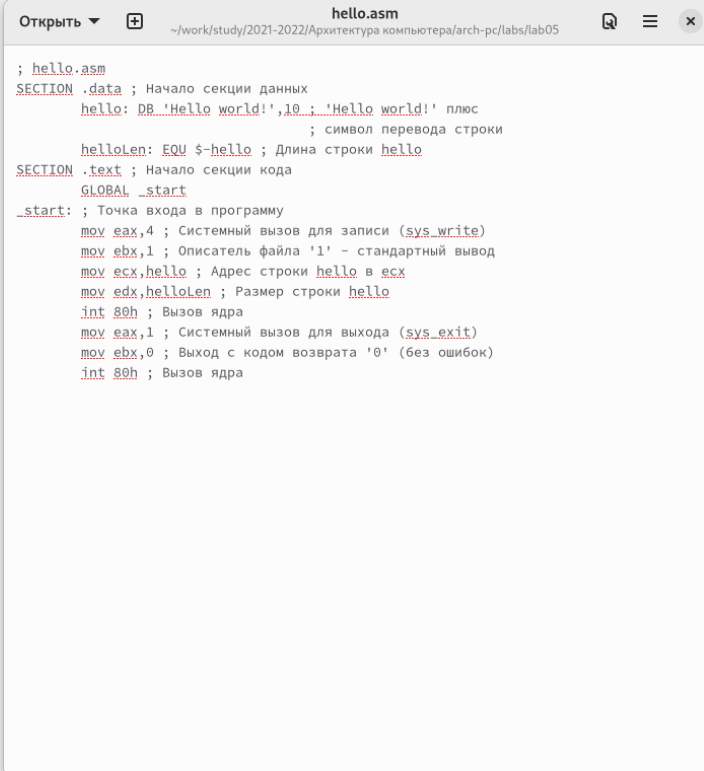
1	Цель работы	3
2	Выполнение лабораторной работы	4
3	Выводы	8
4	Контрольные вопросы	9

# 1 Цель работы

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM

## 2 Выполнение лабораторной работы

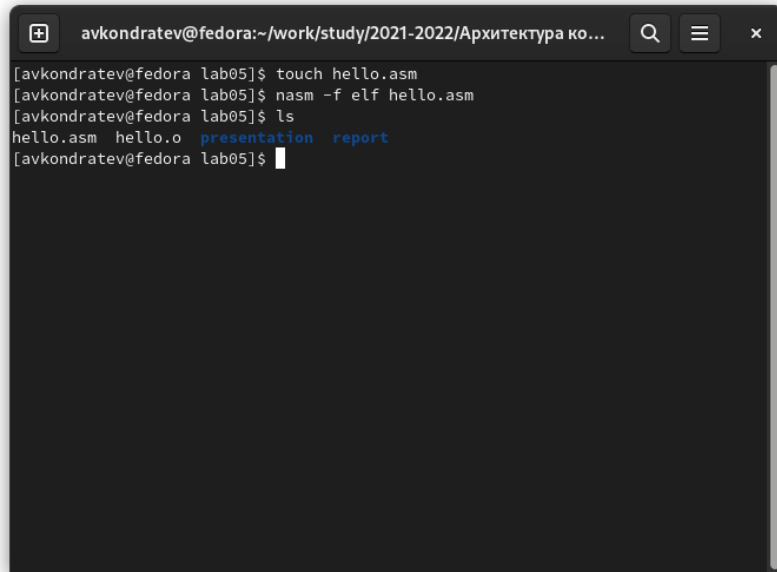
1. Создал текстовый файл с именем hello.asm и ввел в него данный текст(рис.2.1)



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

Figure 2.1: Рис. 1

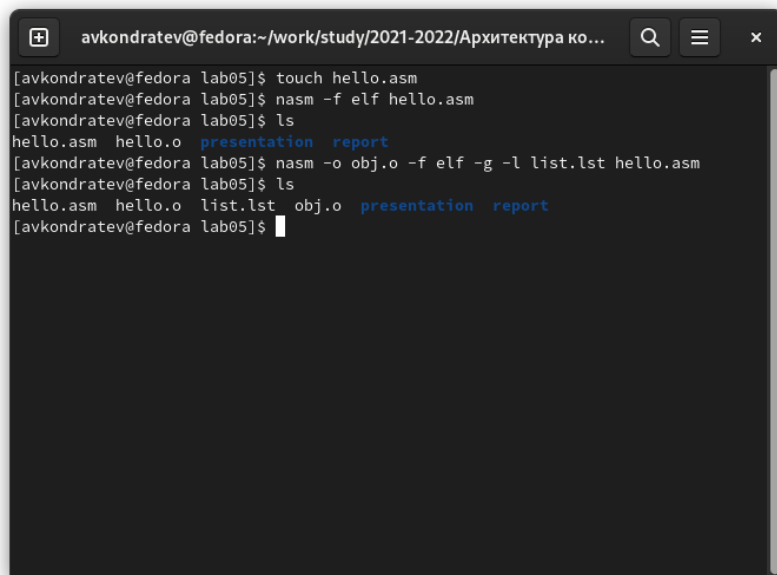
2. Скомпилировал приведённый выше текст программы «Hello World»(рис.2.2)



```
avkondratev@fedora:~/work/study/2021-2022/Архитектура ко...
[avkondratev@fedora lab05]$ touch hello.asm
[avkondratev@fedora lab05]$ nasm -f elf hello.asm
[avkondratev@fedora lab05]$ ls
hello.asm  hello.o  presentation  report
[avkondratev@fedora lab05]$
```

Figure 2.2: Рис. 2

3. Скомпилировал исходный файл hello.asm в obj.o и создал файл листинга list.lst(рис.2.3)



```
avkondratev@fedora:~/work/study/2021-2022/Архитектура ко...
[avkondratev@fedora lab05]$ touch hello.asm
[avkondratev@fedora lab05]$ nasm -f elf hello.asm
[avkondratev@fedora lab05]$ ls
hello.asm  hello.o  presentation  report
[avkondratev@fedora lab05]$ nasm -o obj.o -f elf -g -l list.lst hello.asm
[avkondratev@fedora lab05]$ ls
hello.asm  hello.o  list.lst  obj.o  presentation  report
[avkondratev@fedora lab05]$
```

Figure 2.3: Рис. 3

4. Получил исполняемую программу(рис.2.4)

```
avkondratev@fedora:~/work/study/2021-2022/Архитектура ко...
[avkondratev@fedora lab05]$ touch hello.asm
[avkondratev@fedora lab05]$ nasm -f elf hello.asm
[avkondratev@fedora lab05]$ ls
hello.asm  hello.o  presentation  report
[avkondratev@fedora lab05]$ nasm -o obj.o -f elf -g -l list.lst hello.asm
[avkondratev@fedora lab05]$ ls
hello.asm  hello.o  list.lst  obj.o  presentation  report
[avkondratev@fedora lab05]$ ld -m elf_i386 hello.o -o hello
[avkondratev@fedora lab05]$ ls
hello  hello.asm  hello.o  list.lst  obj.o  presentation  report
[avkondratev@fedora lab05]$
```

Figure 2.4: Рис. 4

5. Получил исполняемую программу, указав имя main.(рис.2.5)

```
avkondratev@fedora:~/work/study/2021-2022/Архитектура ко...
[avkondratev@fedora lab05]$ touch hello.asm
[avkondratev@fedora lab05]$ nasm -f elf hello.asm
[avkondratev@fedora lab05]$ ls
hello.asm  hello.o  presentation  report
[avkondratev@fedora lab05]$ nasm -o obj.o -f elf -g -l list.lst hello.asm
[avkondratev@fedora lab05]$ ls
hello.asm  hello.o  list.lst  obj.o  presentation  report
[avkondratev@fedora lab05]$ ld -m elf_i386 hello.o -o hello
[avkondratev@fedora lab05]$ ls
hello  hello.asm  hello.o  list.lst  obj.o  presentation  report
[avkondratev@fedora lab05]$ ld -m elf_i386 obj.o -o main
[avkondratev@fedora lab05]$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o  presentation  report
[avkondratev@fedora lab05]$
```

Figure 2.5: Рис. 5

6. Запустил на выполнение созданный исполняемый файл(рис.2.6)

```
avkondratev@fedora:~/work/study/2021-2022/Архитектура ко...
[avkondratev@fedora lab05]$ touch hello.asm
[avkondratev@fedora lab05]$ nasm -f elf hello.asm
[avkondratev@fedora lab05]$ ls
hello.asm  hello.o  presentation  report
[avkondratev@fedora lab05]$ nasm -o obj.o -f elf -g -l list.lst hello.asm
[avkondratev@fedora lab05]$ ls
hello.asm  hello.o  list.lst  obj.o  presentation  report
[avkondratev@fedora lab05]$ ld -m elf_i386 hello.o -o hello
[avkondratev@fedora lab05]$ ls
hello  hello.asm  hello.o  list.lst  obj.o  presentation  report
[avkondratev@fedora lab05]$ ld -m elf_i386 obj.o -o main
[avkondratev@fedora lab05]$ ls
hello  hello.asm  hello.o  list.lst  main  obj.o  presentation  report
[avkondratev@fedora lab05]$ ./hello
Hello world!
[avkondratev@fedora lab05]$
```

Figure 2.6: Рис. 6

7. Создал копию с именем lab5 и изменил код, чтобы выводились мое имя и фамилия(рис.2.7)

```
avkondratev@fedora:~/work/lab05
[avkondratev@fedora lab05]$ cp hello.asm lab5.asm
[avkondratev@fedora lab05]$ nasm -f elf lab5.asm
nasm: fatal: unable to open input file 'lab5.asm' No such file or directory
[avkondratev@fedora lab05]$ nasm -f elf lab5.asm
[avkondratev@fedora lab05]$ nasm -o obj.o -f elf -g -l list.lst lab5.asm
[avkondratev@fedora lab05]$ ld -m elf_i386 lab5.o -o lab5
[avkondratev@fedora lab05]$ ./lab5
Кондратьев Арсений
[avkondratev@fedora lab05]$
```

Figure 2.7: Рис. 7

## 3 Выводы

Я освоил процедуры компиляции и сборки программ, написанных на ассемблере NASM



## 4 Контрольные вопросы

1. 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
2. директивы не переводящиеся непосредственно в машинные команды, а управляющие работой транслятора
3. Набор текста программы в текстовом редакторе и сохранение её в отдельном файле. Каждый файл имеет свой тип (или расширение), который определяет назначение файла. Файлы с исходным текстом программ на языке ассемблера имеют тип `asm`.

Трансляция — преобразование с помощью транслятора, например `nasm`, текста программы в машинный код, называемый объектным. На данном этапе также может быть получен листинг программы, содержащий кроме текста программы различную дополнительную информацию, созданную транслятором. Тип объектного файла — `o`, файла листинга — `lst`.

Компоновка или линковка — этап обработки объектного кода компоновщиком (`ld`), который принимает на вход объектные файлы и собирает по ним исполняемый файл. Исполняемый файл обычно не имеет расширения. Кроме того, можно получить файл карты загрузки программы в ОЗУ, имеющий расширение `map`.

Запуск программы. Конечной целью является работоспособный исполняемый файл. Ошибки на предыдущих этапах могут привести к некорректной

работе программы, поэтому может присутствовать этап отладки программы при помощи специальной программы — отладчика. При нахождении ошибки необходимо провести коррекцию программы, начиная с первого шага

4. набор текста программы, трансляция, компоновка
5. преобразование с помощью транслятора, например `nasm`, текста программы в машинный код, называемый объектным. На данном этапе также может быть получен листинг программы, содержащий кроме текста программы различную дополнительную информацию, созданную транслятором
6. этап обработки объектного кода компоновщиком (`ld`), который принимает на вход объектные файлы и собирает по ним исполняемый файл
7. NASM не запускают без параметров. Ключ `-f` указывает транслятору, что требуется создать бинарные файлы в формате ELF. Следует отметить, что формат `elf64` позволяет создавать исполняемый код, работающий под 64-битными версиями Linux. Для 32-битных версий ОС указываем в качестве формата просто `elf`
8. `nasm -o;` `ld` - исполняемая программа