

Отчёт по лабораторной работе №4

Дисциплина: Архитектура компьютера

Филатов Илья Гурамович

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	10
4.1	Программа Hello world!	10
4.2	Транслятор NASM	11
4.3	Расширенный синтаксис командной строки NASM	12
4.4	Компоновщик LD	12
4.5	Запуск исполняемого файла	14
4.6	Задание для самостоятельной работы.	14
5	Выводы	17
6	Список литературы	18

Список иллюстраций

4.1	Создание каталогов	10
4.2	Переход в каталог	10
4.3	Создание файла hello.asm	10
4.4	Компиляция шаблона	11
4.5	Ввод команд	11
4.6	Создание объектного кода	11
4.7	Проверка создания файла	12
4.8	Ввод команды	12
4.9	Обработка компоновщиком	12
4.10	Обработка компоновщиком	13
4.11	Проверка создания исполняемого файла	13
4.12	Выполнение команды	13
4.13	Формат командной строки LD	14
4.14	Запуск исполняемого файла	14
4.15	Копирование файла	14
4.16	Открытие файла	15
4.17	Преобразование текста программы	15
4.18	Трансляция и компоновка	15
4.19	Запуск файла	16
4.20	Копирование файлов	16
4.21	Загрузка файлов на Github	16

1 Цель работы

Целью работы является обучение компиляции и сборке программ, написанных с помощью ассемблера NASM.

2 Задание

1. Программа Hello world!
2. Транслятор NASM
3. Расширенный синтаксис командной строки NASM
4. Компоновщик LD
5. Запуск исполняемого файла
6. Задание для самостоятельной работы

3 Теоретическое введение

Основными функциональными элементами любой электронно-вычислительной машины (ЭВМ) являются центральный процессор, память и периферийные устройства. Взаимодействие этих устройств осуществляется через общую шину, которая представляет собой большое количество проводников, соединяющих устройства друг с другом. 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 арифметические 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 состав процессора, для временного хранения промежуточных результатов выполнения инструкций; регистры процессора делятся на два типа: регистры общего назначения и специальные регистры.

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, преобразование данных хранящихся в регистрах.

Доступ к регистрам осуществляется по именам. Каждый регистр процессора

архитектуры x86 имеет свое название, состоящее из 2 или 3 букв латинского алфавита.

Названия основных регистров общего назначения:

- RAX, RCX, RDX, RBX, RSI, RDI — 64-битные
- EAX, ECX, EDX, EBX, ESI, EDI — 32-битные
- AX, CX, DX, BX, SI, DI — 16-битные
- AH, AL, CH, CL, DH, DL, BH, BL — 8-битные (половинки 16-битных регистров).

Например, AH (high AX) — старшие 8 бит регистра AX, AL (low AX) — младшие 8 бит регистра AX.

Оперативное запоминающее устройство (ОЗУ) — это быстродействующее энергозависимое запоминающее устройство, которое напрямую взаимодействует с узлами процессора, предназначенное для хранения программ и данных, с которыми процессор непосредственно работает в текущий момент. ОЗУ состоит из одинаковых пронумерованных ячеек памяти. Номер ячейки памяти — это адрес хранящихся в ней данных.

В состав ЭВМ также входят периферийные устройства, которые можно разделить на:

- устройства внешней памяти, которые предназначены для долговременного хранения больших объёмов данных (жёсткие диски, твердотельные накопители, магнитные ленты)
- устройства ввода-вывода, которые обеспечивают взаимодействие ЦП с внешней средой.

В основе вычислительного процесса ЭВМ лежит принцип программного управления. Программа состоит из машинных команд, которые указывают, какие операции и над какими данными, в какой последовательности необходимо выполнить. Набор машинных команд определяется устройством конкретного процессора. Коды команд представляют собой многоразрядные двоичные комбинации из 0 и 1. В коде машинной команды можно выделить две части: операционную и адресную. В операционной части хранится код команды, которую необходи-

мо выполнить. В адресной части хранятся данные или адреса данных, которые участвуют в выполнении данной операции.

При выполнении каждой команды процессор выполняет определённую последовательность стандартных действий, которая называется командным циклом процессора. В самом общем виде он заключается в следующем:

1. формирование адреса в памяти очередной команды.
2. считывание кода команды из памяти и её дешифрация.
3. выполнение команды.
4. переход к следующей команде.

Язык ассемблера — машинно-ориентированный язык низкого уровня. Он больше любых других языков приближен к архитектуре ЭВМ и её аппаратным возможностям, что позволяет получить к ним более полный доступ, нежели в языках высокого уровня. Получить полный доступ к ресурсам компьютера в современных архитектурах нельзя, самым низким уровнем работы прикладной программы является обращение напрямую к ядру операционной системы. Именно на этом уровне и работают программы, написанные на ассемблере. Таким образом язык ассемблера — это язык, с помощью которого понятным для человека образом пишутся команды для процессора. Преобразование или трансляция команд языка ассемблера в исполняемый машинный код осуществляется специальной программой транслятором — Ассемблер. Наиболее распространёнными ассемблерами для архитектуры x86 являются:

- для DOS/Windows: Borland Turbo Assembler (TASM), Microsoft Macro Assembler (MASM) и Watcom assembler (WASM);
- для GNU/Linux: gas (GNU Assembler), использующий AT&T-синтаксис, в отличие от большинства других популярных ассемблеров, которые используют Intel-синтаксис.

NASM — это открытый проект ассемблера, версии которого доступны под различные операционные системы и который позволяет получать объектные файлы для этих систем. В NASM используется Intel-синтаксис и поддерживаются инструкции x86-64.

В процессе создания ассемблерной программы можно выделить четыре шага:

- 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 Выполнение лабораторной работы

4.1 Программа Hello world!

Открываю терминал. Используя команду `mkdir` и ключ `-p` создаю иерархическую цепочку каталогов (рис. 4.1).



```
igfilatov@igfilatov:~  
[igfilatov@igfilatov ~]$ mkdir -p ~/work/arch-pc/lab04  
[igfilatov@igfilatov ~]$
```

Рис. 4.1: Создание каталогов

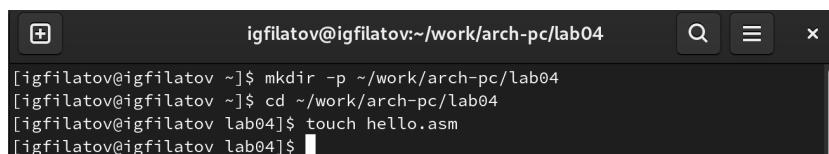
С помощью команды `cd` и относительного пути перехожу в созданный каталог (рис. 4.2).



```
igfilatov@igfilatov:~/work/arch-pc/lab04  
[igfilatov@igfilatov ~]$ mkdir -p ~/work/arch-pc/lab04  
[igfilatov@igfilatov ~]$ cd ~/work/arch-pc/lab04  
[igfilatov@igfilatov lab04]$
```

Рис. 4.2: Переход в каталог

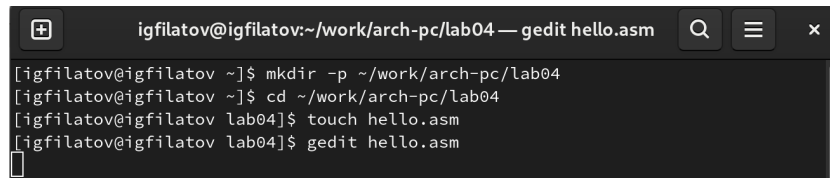
Командой `touch` создаю файл `hello.asm` (рис. 4.3).



```
igfilatov@igfilatov:~/work/arch-pc/lab04  
[igfilatov@igfilatov ~]$ mkdir -p ~/work/arch-pc/lab04  
[igfilatov@igfilatov ~]$ cd ~/work/arch-pc/lab04  
[igfilatov@igfilatov lab04]$ touch hello.asm  
[igfilatov@igfilatov lab04]$
```

Рис. 4.3: Создание файла `hello.asm`

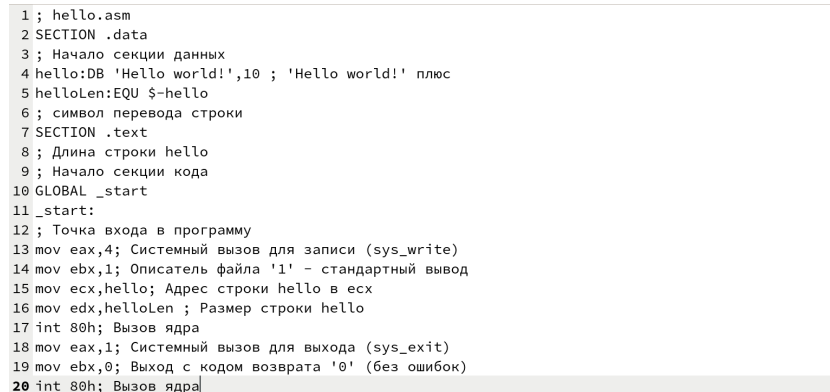
Открываю этот файл с помощью текстового редактора `gedit` (рис. 4.4).



```
igfilatov@igfilatov:~/work/arch-pc/lab04 — gedit hello.asm
[igfilatov@igfilatov ~]$ mkdir -p ~/work/arch-pc/lab04
[igfilatov@igfilatov ~]$ cd ~/work/arch-pc/lab04
[igfilatov@igfilatov lab04]$ touch hello.asm
[igfilatov@igfilatov lab04]$ gedit hello.asm
```

Рис. 4.4: Компиляция шаблона

Ввожу последовательность команд (рис. 4.5).

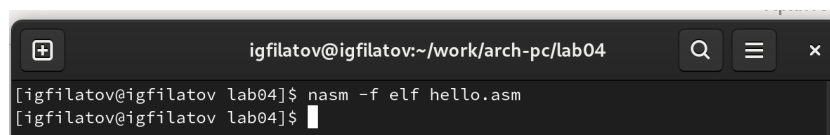


```
1 ; hello.asm
2 SECTION .data
3 ; Начало секции данных
4 hello:DB 'Hello world!',10 ; 'Hello world!' плюс
5 helloLen:EQU $-hello
6 ; символ перевода строки
7 SECTION .text
8 ; Длина строки hello
9 ; Начало секции кода
10 GLOBAL _start
11 _start:
12 ; Точка входа в программу
13 mov eax,4; Системный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

Рис. 4.5: Ввод команд

4.2 Транслятор NASM

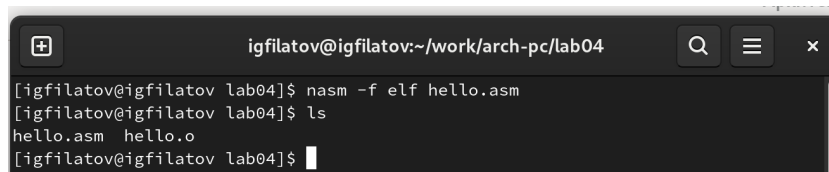
Преобразую текст программы в объектный код с помощью команды `nasm` (рис. 4.6).



```
igfilatov@igfilatov:~/work/arch-pc/lab04
[igfilatov@igfilatov lab04]$ nasm -f elf hello.asm
[igfilatov@igfilatov lab04]$
```

Рис. 4.6: Создание объектного кода

С помощью команды `ls` проверяю, что файл с именем `hello.o` был создан (рис. 4.7).



```
igfilatov@igfilatov:~/work/arch-pc/lab04
[igfilatov@igfilatov lab04]$ nasm -f elf hello.asm
[igfilatov@igfilatov lab04]$ ls
hello.asm  hello.o
[igfilatov@igfilatov lab04]$
```

Рис. 4.7: Проверка создания файла

4.3 Расширенный синтаксис командной строки NASM

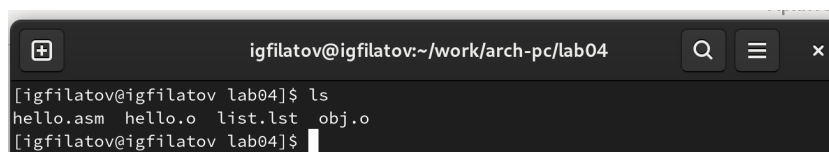
Выполняю команду, которая компилирует файл `hello.asm` в `obj.o` (опция `-o` задаёт имя), при этом формат выходного файла будет `elf`, и в него будут включены символы для отладки (опция `-g`). Также команда создаёт файл листинга `list.lst` (опция `-l`) (рис. 4.8).



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

Рис. 4.8: Ввод команды

Командой `ls` проверяю, что все файлы созданы (рис. 4.9).

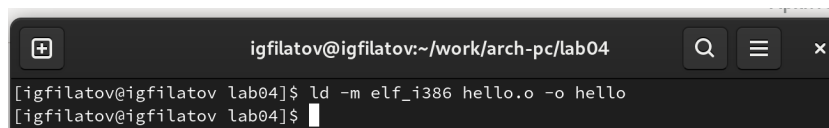


```
igfilatov@igfilatov:~/work/arch-pc/lab04
[igfilatov@igfilatov lab04]$ ls
hello.asm  hello.o  list.lst  obj.o
[igfilatov@igfilatov lab04]$
```

Рис. 4.9: Обработка компоновщиком

4.4 Компоновщик LD

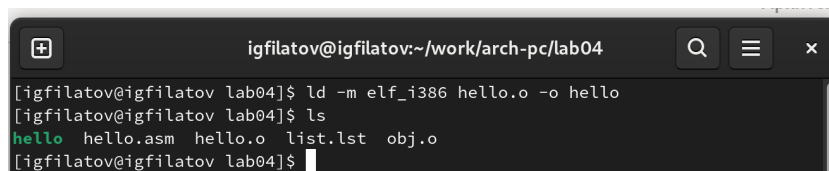
Передаю объектный файл на обработку компоновщику командой `ld` для получения программы (рис. 4.10).



```
igfilatov@igfilatov:~/work/arch-pc/lab04
[igfilatov@igfilatov lab04]$ ld -m elf_i386 hello.o -o hello
[igfilatov@igfilatov lab04]$
```

Рис. 4.10: Обработка компоновщиком

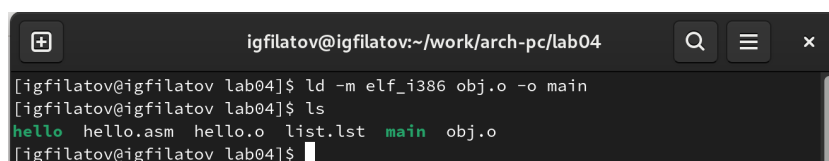
Командой `ls` проверяю, что исполняемый файл создан (рис. 4.11).



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

Рис. 4.11: Проверка создания исполняемого файла

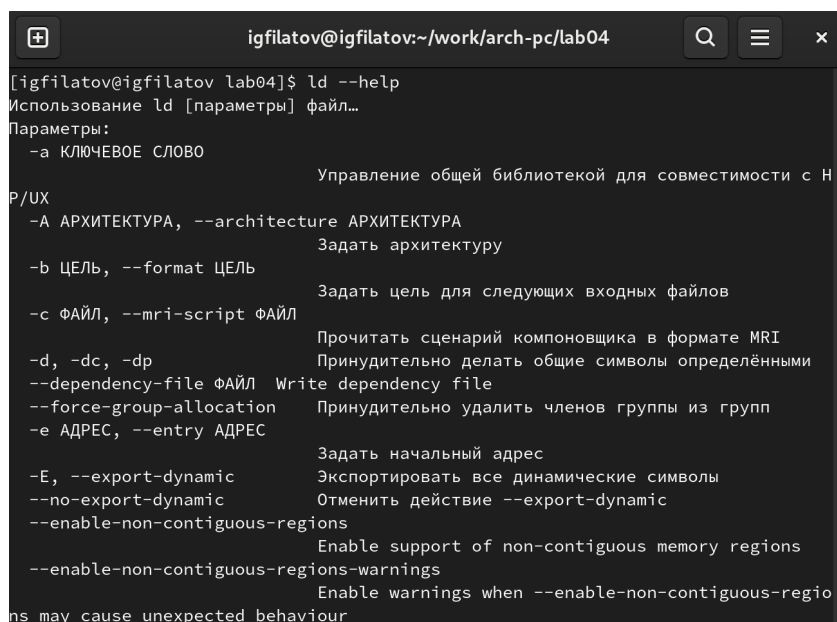
Выполняю команду, которая преобразует объектный файл с именем `obj.o` в исполняемый файл, которому с помощью опции `-o` задаётся имя `main`. Командой `ls` проверяю, что файлы созданы корректно (рис. 4.12).



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

Рис. 4.12: Выполнение команды

Набрав `ld --help`, можно увидеть формат командной строки LD.(рис. 4.13).

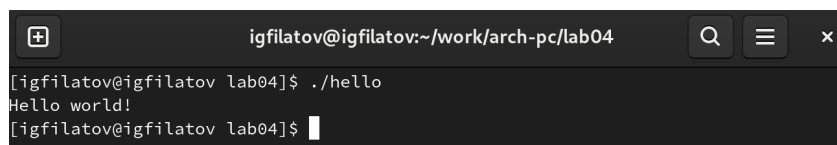


```
igfilatov@igfilatov:~/work/arch-pc/lab04
[igfilatov@igfilatov lab04]$ ld --help
Использование ld [параметры] файл...
Параметры:
  -a КЛЮЧЕВОЕ СЛОВО          Управление общей библиотекой для совместимости с H
P/UX
  -A АРХИТЕКТУРА, --architecture АРХИТЕКТУРА      Задать архитектуру
  -b ЦЕЛЬ, --format ЦЕЛЬ          Задать цель для следующих входных файлов
  -c ФАЙЛ, --mri-script ФАЙЛ      Прочитать сценарий компоновщика в формате MRI
  -d, -dc, -dp                  Принудительно делать общие символы определёнными
  --dependency-file ФАЙЛ        Write dependency file
  --force-group-allocation      Принудительно удалить членов группы из групп
  -e АДРЕС, --entry АДРЕС        Задать начальный адрес
  -E, --export-dynamic           Экспортировать все динамические символы
  --no-export-dynamic            Отменить действие --export-dynamic
  --enable-non-contiguous-regions Enable support of non-contiguous memory regions
  --enable-non-contiguous-regions-warnings Enable warnings when --enable-non-contiguous-regio
ns may cause unexpected behaviour
```

Рис. 4.13: Формат командной строки LD

4.5 Запуск исполняемого файла

Запускаю созданный исполняемый файл (рис. 4.14).



```
igfilatov@igfilatov:~/work/arch-pc/lab04
[igfilatov@igfilatov lab04]$ ./hello
Hello world!
[igfilatov@igfilatov lab04]$
```

Рис. 4.14: Запуск исполняемого файла

4.6 Задание для самостоятельной работы.

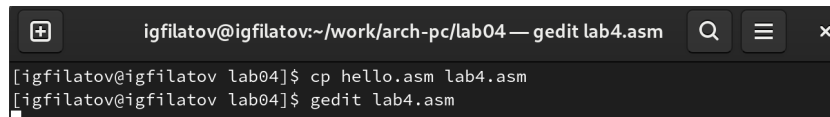
Копирую файл hello.asm командой cp, изменив его имя. (рис. 4.15).



```
igfilatov@igfilatov:~/work/arch-pc/lab04
[igfilatov@igfilatov lab04]$ cp hello.asm lab4.asm
[igfilatov@igfilatov lab04]$
```

Рис. 4.15: Копирование файла

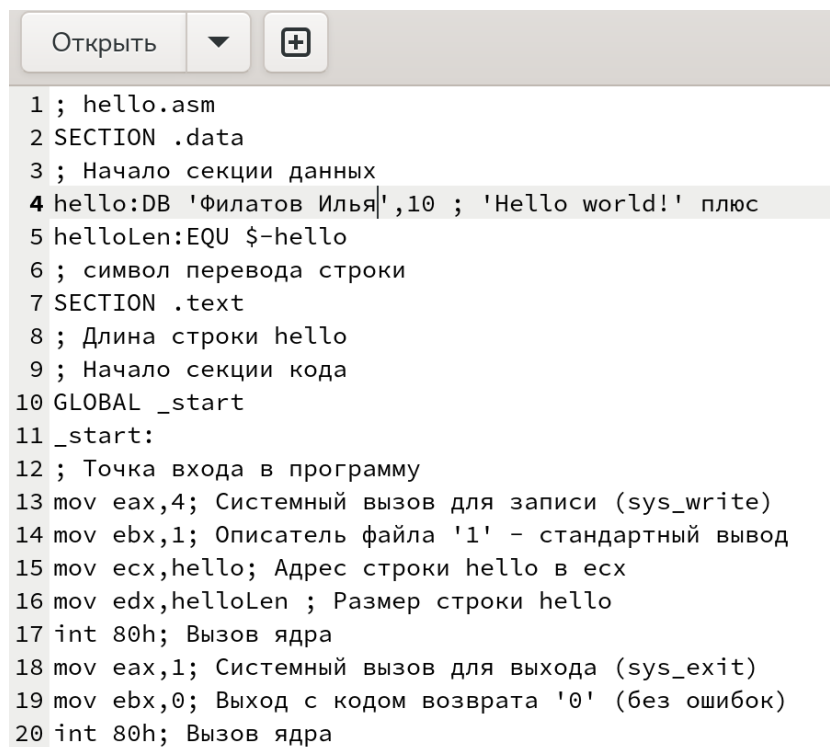
С помощью редактора gedit открываю новый файл (рис. 4.16).



```
igfilatov@igfilatov:~/work/arch-pc/lab04 — gedit lab4.asm
[igfilatov@igfilatov lab04]$ cp hello.asm lab4.asm
[igfilatov@igfilatov lab04]$ gedit lab4.asm
```

Рис. 4.16: Открытие файла

Преобразовываю текст программы его так, чтобы на экран выводились фамилия и имя (рис. 4.17).



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

Рис. 4.17: Преобразование текста программы

Провожу трансляцию и компоновку файлов командами nasm и ld (рис. 4.18).



```
igfilatov@igfilatov:~/work/arch-pc/lab04
[igfilatov@igfilatov lab04]$ nasm -f elf lab4.asm
[igfilatov@igfilatov lab04]$ ld -m elf_i386 lab4.o -o lab4
[igfilatov@igfilatov lab04]$
```

Рис. 4.18: Трансляция и компоновка

Запускаю полученный файл (рис. 4.19).

```
igfilatov@igfilatov:~/work/arch-pc/lab04
[igfilatov@igfilatov lab04]$ ./lab4
Филатов Илья
[igfilatov@igfilatov lab04]$
```

Рис. 4.19: Запуск файла

Командой `ls` проверяю, что на `.asm` заканчиваются только нужные файлы. С помощью команды `cp` и относительного пути копирую все файлы из этого каталога, заканчивающиеся на `.asm` (рис. 4.20).

```
[igfilatov@igfilatov lab04]$ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o
[igfilatov@igfilatov lab04]$ cp *.asm ~/work/study/2024-2025/"Архитектура компью
тера"/arch-pc/labs/lab04/
[igfilatov@igfilatov lab04]$
```

Рис. 4.20: Копирование файлов

Командой `cd` перехожу в каталог с репозиторием. Командами `git add .`, `git commit` и `git push` загружаю файлы на Github (рис. 4.21).

```
igfilatov@igfilatov:~/work/study/2024-2025/Архитектура ком...
[igfilatov@igfilatov lab04]$ cd ~/work/study/2024-2025/"Архитектура компьютера"/
arch-pc
[igfilatov@igfilatov arch-pc]$ git add .
[igfilatov@igfilatov arch-pc]$ git commit -m "files used in lab4 was added"
[master 34ea6ad] files used in lab4 was added
2 files changed, 40 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
[igfilatov@igfilatov arch-pc]$ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 988 байтов | 247.00 КиБ/с, готово.
Total 6 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:igfilatov/study_2024-2025_arh-pc.git
ca53d4a..34ea6ad master -> master
[igfilatov@igfilatov arch-pc]$
```

Рис. 4.21: Загрузка файлов на Github

5 Выводы

Я освоил процедуру компиляции и сборки программ, написанных на ассемблере NASM.

6 Список литературы

1. Архитектура ЭВМ