

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ЛАБОРАТОРНАЯ РАБОТА №4

**на тему «СОЗДАНИЕ И ПРОЦЕСС ОБРАБОТКИ ПРОГРАММ
НА ЯЗЫКЕ АССЕМБЛЕРА NASM»**

дисциплина: Архитектура компьютера

Студент: Трусова А.А.

Группа: НКАбд-05-24

№ ст. билета: 1132246715

МОСКВА

2024г.

Содержание

Цель работы

Теоретическое введение

Выполнение лабораторной работы

4.1. Программа Hello world!

4.2. Транслятор NASM

4.3. Расширенный синтаксис командной строки NASM

4.4. Компоновщик LD

4.5. Запуск исполняемого файла

4.6. Задание для самостоятельной работы

Вывод

Цель работы

Освоение процедуры компиляции и сборки программ, написанных на ассемблере NASM.

Теоретическое введение

Основными функциональными элементами любой электронно-вычислительной машины (ЭВМ) являются центральный процессор, память и периферийные устройства.

Взаимодействие этих устройств осуществляется через общую шину, к которой они подключены. Физически шина представляет собой большое количество проводников, соединяющих устройства друг с другом. В современных компьютерах проводники выполнены в виде электропроводящих 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 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.

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, преобразование (арифметические или логические операции) данных хранящихся в регистрах.

Доступ к регистрам осуществляется не по адресам, как к основной памяти, а по именам. Каждый регистр процессора архитектуры x86 имеет свое название, состоящее из 2 или 3 букв латинского алфавита.

В качестве примера приведем названия основных регистров общего назначения (именно эти регистры чаще всего используются при написании программ):

- 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).

Другим важным узлом ЭВМ является оперативное запоминающее устройство (ОЗУ). 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. Это означает, что компьютер решает поставленную задачу как последовательность действий, записанных в виде программы.

Коды команд представляют собой многоразрядные двоичные комбинации из 0 и 1. В коде машинной команды можно выделить две части: операционную и адресную.

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 (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. 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.

Выполнение лабораторной работы

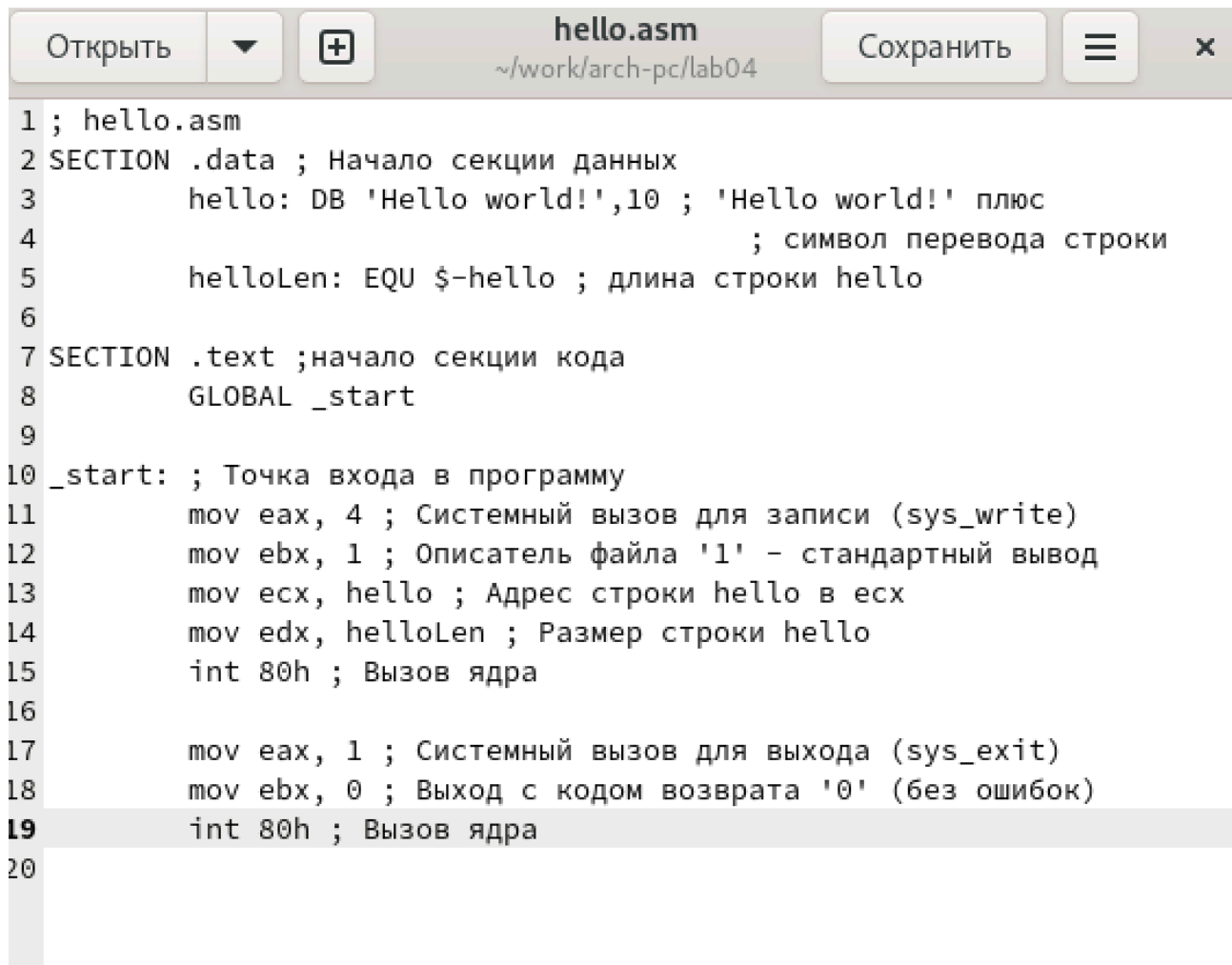
4.1. Программа Hello world!

Создала каталог для работы с программами на языке ассемблера NASM, перешла в него, создала файл hello.asm и открыла его с помощью gedit (рис.1).

```
aatrusova1132246715@fedora:~$ mkdir -p ~/work/arch-pc/lab04
aatrusova1132246715@fedora:~$ cd ~/work/arch-pc/lab04
aatrusova1132246715@fedora:~/work/arch-pc/lab04$ touch hello.asm
aatrusova1132246715@fedora:~/work/arch-pc/lab04$ gedit hello.asm
```

Рис.1

Записала в него текст программы (рис.2).



The screenshot shows the gedit text editor with the file 'hello.asm' open at the path '~/work/arch-pc/lab04'. The editor interface includes buttons for 'Открыть' (Open), 'Сохранить' (Save), and a menu icon. The code is as follows:

```
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
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, hello ; Адрес строки hello в ecx
14     mov edx, helloLen ; Размер строки hello
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
20
```

Рис.2

4.2. Транслятор NASM

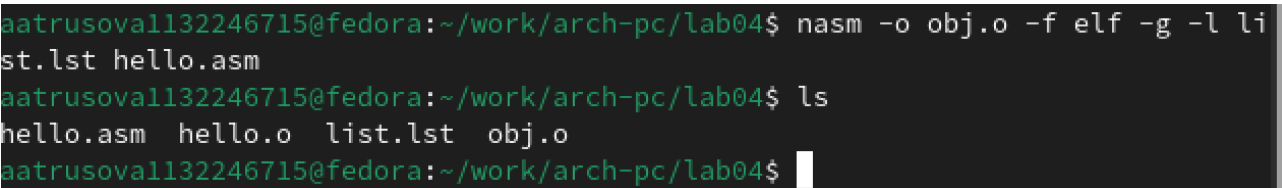
С помощью NASM преобразовала текст программы из файла hello.asm в объектный код в файл hello.o и проверила правильность выполнения(рис.3).

```
aatrusova1132246715@fedora:~/work/arch-pc/lab04$ nasm -f elf hello.asm
aatrusova1132246715@fedora:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
aatrusova1132246715@fedora:~/work/arch-pc/lab04$
```

Рис.3

4.3. Расширенный синтаксис командной строки NASM

Скомпилировала исходный файл hello.asm в obj.o с форматом выходного файла elf, создала файл листинга list.lst и проверила правильность выполнения (рис.4).

A terminal window with a dark background and green text. The prompt is 'aatrusoval132246715@fedora:~/work/arch-pc/lab04\$'. The first command is 'nasm -o obj.o -f elf -g -l list.lst hello.asm'. The second command is 'ls', which outputs 'hello.asm hello.o list.lst obj.o'. The prompt is shown again at the end of the output.

```
aatrusoval132246715@fedora:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
aatrusoval132246715@fedora:~/work/arch-pc/lab04$ ls
hello.asm  hello.o  list.lst  obj.o
aatrusoval132246715@fedora:~/work/arch-pc/lab04$
```

Рис.4

4.4. Компоновщик LD

Передаю объектный файл hello.o компоновщику ld на обработку и получаю исполняемый файл hello (рис.5).

```
aatrusoval132246715@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
aatrusoval132246715@fedora:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o list.lst obj.o
aatrusoval132246715@fedora:~/work/arch-pc/lab04$
```

Рис.5

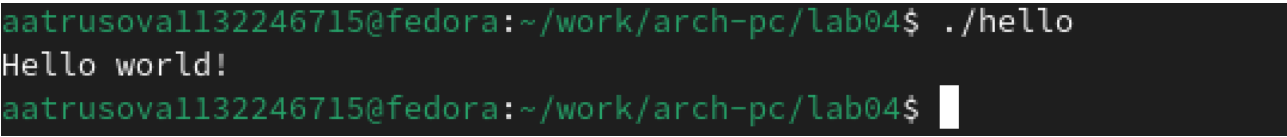
Передаю объектный файл obj.o компоновщику ld на обработку и получаю исполняемый файл main (рис.6).

```
aatrusoval132246715@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
aatrusoval132246715@fedora:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o list.lst main obj.o
aatrusoval132246715@fedora:~/work/arch-pc/lab04$
```

Рис.6

4.5. Запуск исполняемого файла

Запустила исполняемый файл. Работает корректно (рис. 7).

A terminal window with a black background and green text. The prompt is 'aatrusova1132246715@fedora:~/work/arch-pc/lab04\$'. The user has entered './hello' and the output is 'Hello world!'. The prompt is now on a new line.

```
aatrusova1132246715@fedora:~/work/arch-pc/lab04$ ./hello
Hello world!
aatrusova1132246715@fedora:~/work/arch-pc/lab04$
```

Рис.7

4.6. Задание для самостоятельной работы

В каталоге `~/work/arch-pc/lab04` с помощью команды `cp` создала копию файла `hello.asm` с именем `lab4.asm` и открыла её (рис.8).

```
aatrusova1132246715@fedora:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
aatrusova1132246715@fedora:~/work/arch-pc/lab04$ gedit lab4.asm
```

Рис.8

Переписала копию так, чтобы вместо «Hello world!» программа выводила «Alina Trusova» (рис.9).

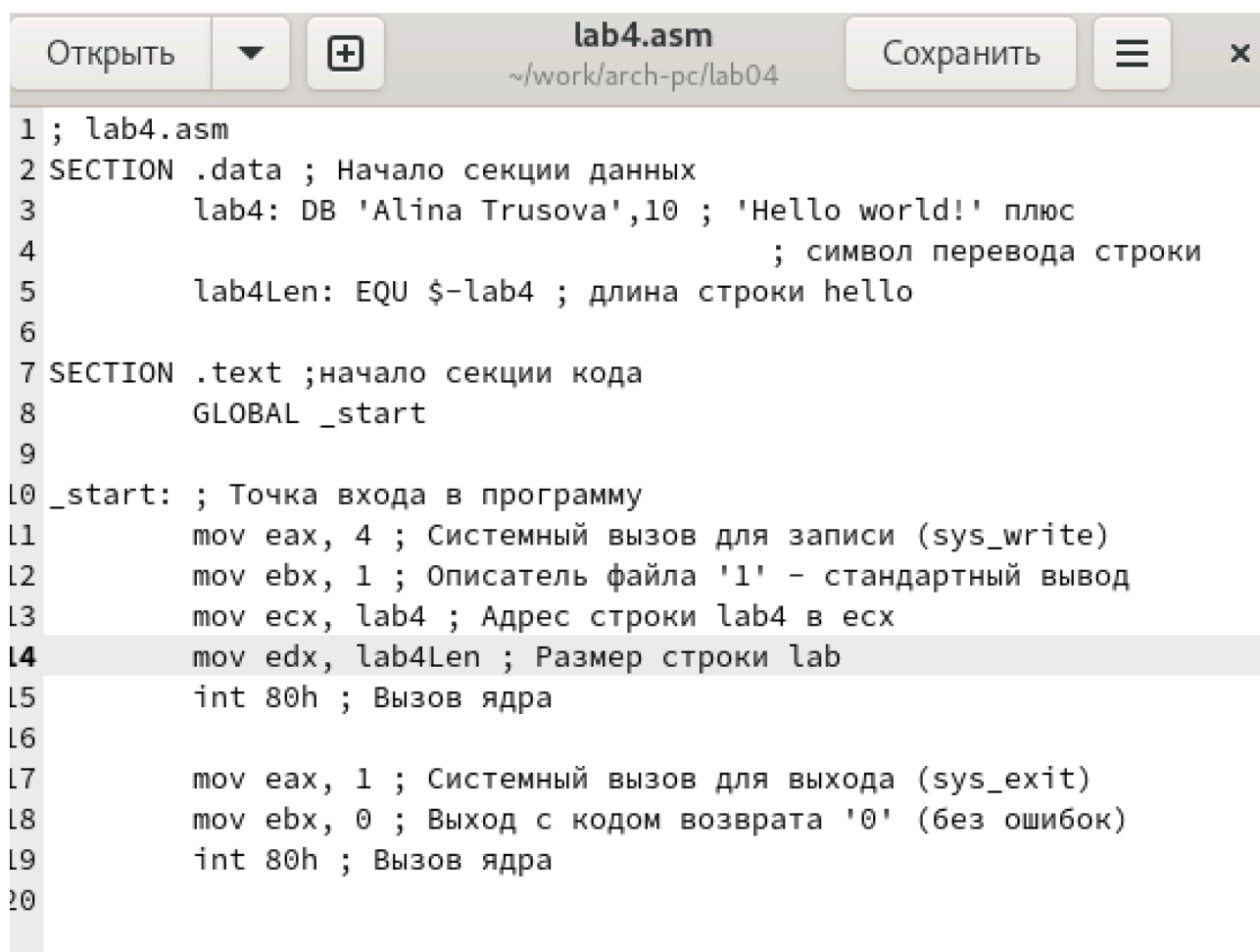


Рис.9

Оттранслировала файл `lab4.asm` в объектный файл `lab4.o` (рис. 10).

```
aatrusova1132246715@fedora:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
aatrusova1132246715@fedora:~/work/arch-pc/lab04$ ls
hello  hello.asm  hello.o  lab4.asm  lab4.o  list.lst  main  obj.o
aatrusova1132246715@fedora:~/work/arch-pc/lab04$
```

Рис.10

Скомпоновала объектный файл и получила исполняемый файл lab4 (рис.11).

```
aatrusova1132246715@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
aatrusova1132246715@fedora:~/work/arch-pc/lab04$ ./lab4
Alina Trusova
aatrusova1132246715@fedora:~/work/arch-pc/lab04$
```

Рис.11

Скопировала файлы hello.asm и lab4.asm в локальный репозиторий в каталог ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04/ (рис.12).

```
aatrusova1132246715@fedora:~/work/arch-pc/lab04$ cp hello.asm lab4.asm ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab04/
aatrusova1132246715@fedora:~/work/arch-pc/lab04$ cd ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab04
aatrusova1132246715@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$ ls
hello.asm lab4.asm presentation report
aatrusova1132246715@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab04$
```

Рис.12

Выгрузила изменения на GitHub (рис.13).

```
pc/labs/lab04$ cd ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc
aatrusova1132246715@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git add .
aatrusova1132246715@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git commit -m 'Add files for lab04'
[master b778eb6] Add files for lab04
 2 files changed, 40 insertions(+)
 create mode 100644 labs/lab04/hello.asm
 create mode 100644 labs/lab04/lab4.asm
aatrusova1132246715@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 1.03 КиБ | 529.00 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:alas-aline/study_2024-2025_arh-pc.git
   dda0051..b778eb6  master -> master
aatrusova1132246715@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$
```

Рис.13

Вывод

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

Список литературы

- 1. [https://esystem.rudn.ru/pluginfile.php/2089084/mod_resource/content/0/Лабораторная работа №4. Создание и процесс обработки программ на языке ассемблера NASM.pdf](https://esystem.rudn.ru/pluginfile.php/2089084/mod_resource/content/0/Лабораторная_работа_№4._Создание_и_процесс_обработки_программ_на_языке_ассемблера_NASM.pdf)**