

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

дисциплина: Архитектура компьютера

Студент: Юсупова Амина Руслановна

Группа: НКАбд-06-25

МОСКВА

2025 г.

Содержание

1	Цель работы.....	2
2	Задание.....	2
3	Теоретическое введение.....	2
4	Выполнение лабораторной работы.....	4
4.1	Программа Hello world!.....	4
4.2	Транслятор NASM.....	55
4.3	Расширенный синтаксис командной строки NASM.....	55
4.4	Компоновщик LD.....	55
4.5	Запуск исполняемого файла.....	66
4.6	Задания для самостоятельной работы.....	66
5	Выводы.....	78
6	Список литературы.....	80

Ошибка! Закладка не определена.

1 Цель работы

Цель данной лабораторной работы - освоить процедуры компиляции и сборки программ, написанных на ассемблере NASM.

2 Задание

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

3 Теоретическое введение

Основными функциональными элементами любой 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 плате. Основной задачей процессора является обработка информации, а также организация координации всех узлов компьютера. В состав центрального процессора 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

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. Периферийные устройства в составе ЭВМ: - устройства внешней памяти, которые предназначены для долговременного хранения больших объёмов данных. - 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. В операционной части хранится код команды, которую необходимо выполнить. В адресной части хранятся данные или адреса данных, которые участвуют в

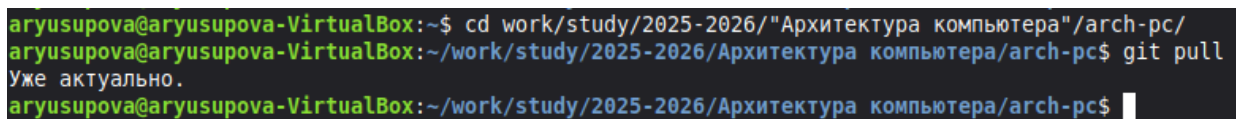
выполнении данной операции. При выполнении каждой команды процессор выполняет определённую последовательность стандартных действий, которая называется командным циклом процессора. Он заключается в следующем: 1. формирование адреса в памяти очередной команды; 2. считывание кода команды из памяти и её 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.

4 Выполнение лабораторной работы

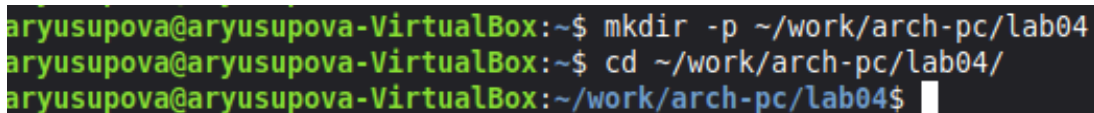
4.1 Программа Hello world!

В домашней директории создаю каталог, в котором буду хранить файлы для текущей лабораторной работы. (рис. 1). И перехожу в рабочую директорию (рис. 2).



```
aryusupova@aryusupova-VirtualBox:~$ cd work/study/2025-2026/"Архитектура компьютера"/arch-pc/
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git pull
Уже актуально.
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc$
```

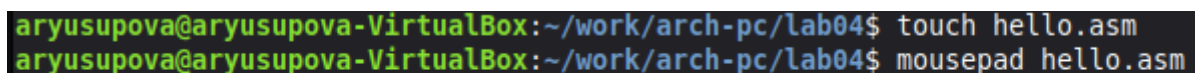
Рис. 1: Создание рабочей директории



```
aryusupova@aryusupova-VirtualBox:~$ mkdir -p ~/work/arch-pc/lab04
aryusupova@aryusupova-VirtualBox:~$ cd ~/work/arch-pc/lab04/
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$
```

Рис.2: Переход в рабочую директорию


Создаю в нем файл hello.asm, в котором буду писать программу на языке ассемблера. (рис. 3)



```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$ touch hello.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$ mousepad hello.asm
```

Рис. 3: Создание asm файла

С помощью редактора пишу программу в созданном файле. (рис. 4)



```
SECTION .data
    hello:      db "Hello, world!",10
    helloLen:   equ $ - hello
SECTION .text
    global _start

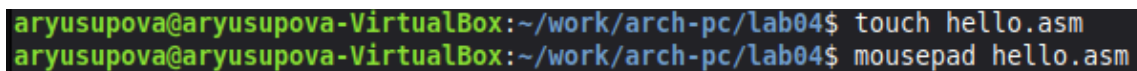
_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, hello
    mov edx, helloLen
    int 80h

    mov eax, 1
    mov ebx, 0
    int 80h
```

Рис. 4: Редактирование созданного файла

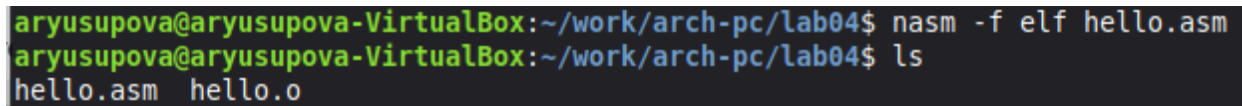
4.2 Транслятор NASM

Компилирую с помощью NASM свою программу. (рис. 5,6).



```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$ touch hello.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$ mousepad hello.asm
```

Рис. 5: Компиляция программы

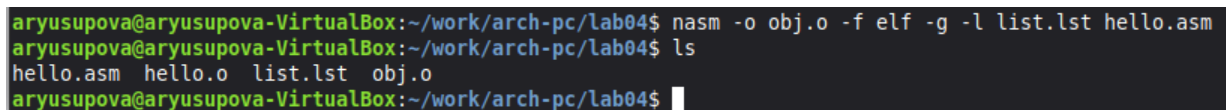


```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$ nasm -f elf hello.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$ ls
hello.asm  hello.o
```

Рис. 6: Компиляция программы

4.3 Расширенный синтаксис командной строки NASM

Выполняю команду, указанную на (рис. 7), она скомпилировала исходный файл hello.asm в obj.o, расширение .o говорит о том, что файл - объектный, помимо него флаги -g -l подготовят файл отладки и листинга соответственно.



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

Рис. 7: Возможности синтаксиса NASM

4.4 Компоновщик LD

Затем мне необходимо передать объектный файл компоновщику, делаю это с помощью команды ld. (рис. 8)

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

Рис. 8: Отправка файла компоновщику

Выполняю следующую команду ..., результатом исполнения команды будет созданный файл main, скомпонованный из объектного файла obj.o. (рис. 9)

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

Рис. 9: Создание исполняемого файла

4.5 Запуск исполняемого файла

Запускаю исполняемый файл из текущего каталога. (рис. 10)

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$ ./hello
Hello, world!
```

Рис. 10: Запуск программы

4.6 Задания для самостоятельной работы

Создаю копию файла для последующей работы с ней. (рис. 11)

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$ cp hello.asm lab4.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4.asm list.lst main obj.o
```

Рис. 11: Создание копии

Редактирую копию файла, заменив текст на свое имя и фамилию. (рис. 12)

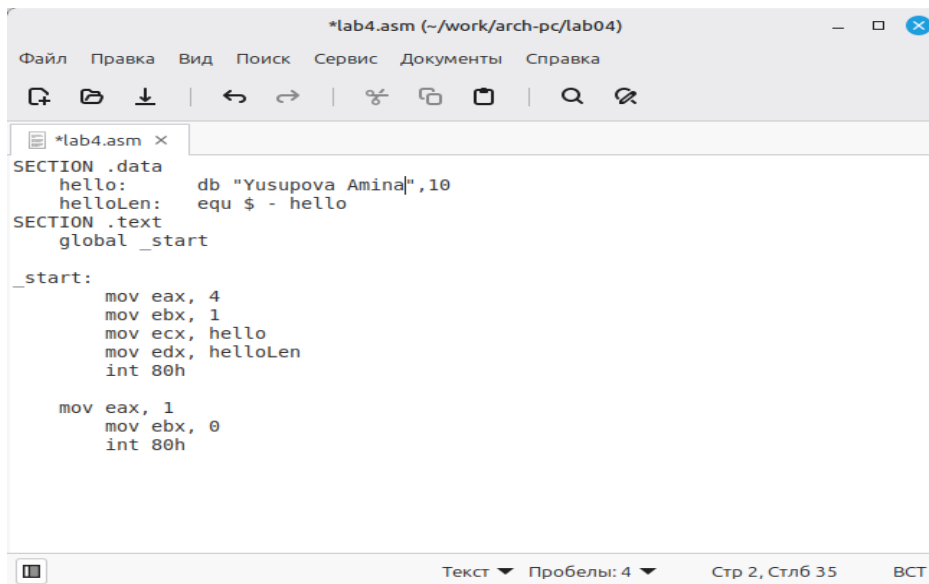


Рис. 12: Редактирование копии

Транслирую копию файла в объектный файл, компоную и запускаю. (рис. 13)

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$ mousepad lab4.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
nasm: fatal: unrecognised output format 'elf' - use -hf for a list
Type nasm -h for help.
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$ ./lab4
Yusupova Amina
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$
```

Рис. 13: Проверка работоспособности скомпонованной программы

Убедилась в корректности выполнения программы. (рис. 14)

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o
```

Рис. 14: Проверка содержимого файла

С помощью команд `git add .` и `git commit` добавляю файлы на GitHub, комментируя действие как добавление файлов для лабораторной работы №4. Отправляю файлы на сервер с помощью команды `git push` (рис. 15).

Загрузка изменений на свой репозиторий на GitHub. (рис. 15)

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$ git add .
fatal: не найден git репозиторий (или один из родительских каталогов): .git
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab04$ cd ~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab04
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab04$ git add .
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab04$ git commit -m 'Add fales for lab04'
[master 2cbcf38] Add fales for lab04
 2 files changed, 32 insertions(+)
 create mode 100644 labs/lab04/hello.asm
 create mode 100644 labs/lab04/lab4.asm
aryusupova@aryusupova-VirtualBox:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab04$ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 659 байтов | 36.00 КиБ/с, готово.
Всего 6 (изменений 3), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:Amina-colab/study_2025-2026_arh_pc.git
 57f9475..2cbcf38 master -> master
```

Рис. 15: Загрузка изменений на репозиторий

5 Выводы

В ходе выполнения данной лабораторной работы я освоила процесс компиляции и сборки программ, написанных на ассемблере NASM.

Список литературы

1. <https://esystem.rudn.ru/mod/page/view.php?id=1030505>
2. https://github.com/Amina-colab/study_2025-2026_arh_pc/tree/master
- 3.