**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: Ниемек Яи Жак

Группа: НММБд-04-24

**МОСКВА**

2025 г.

**Цель работы**

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

**1. Установка NASM и LD**

Перед началом работы убедись, что у тебя установлены необходимые инструменты:

sudo dnf install nasm binutils -y

**2. Создание каталога и файла**

Создай каталог для работы и перейди в него:

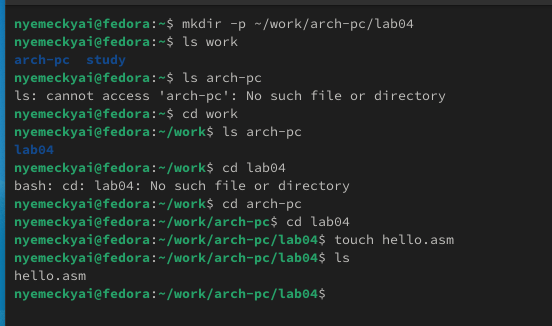
mkdir -p ~/work/arch-pc/lab04

cd ~/work/arch-pc/lab04

Создай файл hello.asm:

touch hello.asm

nano hello.asm # можно использовать любой редактор, например vim или gedit



**3. Напиши код программы**

Вставь в hello.asm следующий код:

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 0x80

mov eax, 1

mov ebx, 0

int 0x80

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.Сохрани файл и выйди из редактора.

**4. Компиляция с NASM**

Скомпилируй код в объектный файл:

nasm -f elf hello.asm -o hello.o

Проверь, что объектный файл создан:

ls -l hello.o

**5. Компоновка с LD**

Создай исполняемый файл:

ld -m elf\_i386 hello.o -o hello

Проверь, что он существует:

ls -l hello

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**6. Запуск программы**

Запусти исполняемый файл:

./hello

Ты должен увидеть "Hello world!" в терминале.

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

**Дополнительные задания**

**1. Создай копию файла и измени текст**

cp hello.asm lab4.asm

nano lab4.asm

Измени строку 'Hello world!' на своё имя, например:

hello: DB 'Иванов Иван!',10

Скомпилируй и запусти по аналогии с hello.asm:

nasm -f elf lab4.asm -o lab4.o

ld -m elf\_i386 lab4.o -o lab4

./lab4

**2. Загрузка на GitHub**

Перейди в каталог с репозиторием:

mkdir -p ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04/

cp hello.asm lab4.asm ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04/

cd ~/work/study/2023-2024/"Архитектура компьютера"/arch-pc/labs/lab04/

Если у тебя ещё нет Git-репозитория, инициализируй его:

git init

git remote add origin <твой\_репозиторий\_на\_GitHub>

Добавь файлы и отправь их в репозиторий:

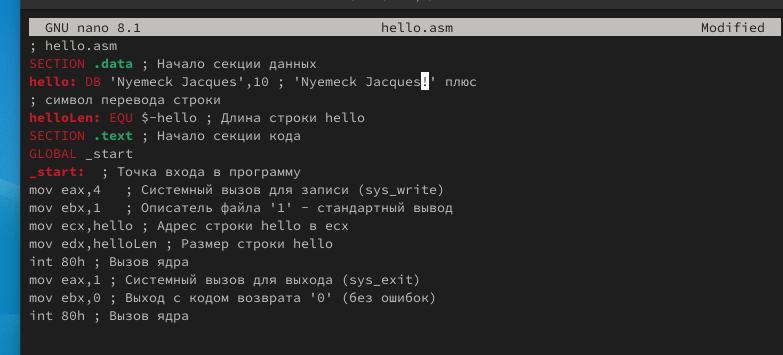
git add hello.asm lab4.asm

git commit -m "Добавлены файлы hello.asm и lab4.asm"

git push origin main

Если ветка называется иначе (например, master), используй:

git push origin master



Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

ответы на вопросы для самопроверки:

**1. Основные отличия ассемблерных программ от программ на языках высокого уровня**

* **Близость к "железу"** – программы на ассемблере работают с регистрами, памятью и инструкциями процессора напрямую.
* **Отсутствие абстракций** – нет готовых библиотек для работы с файлами, строками, сетью и т. д.
* **Высокая производительность** – код выполняется быстрее, так как нет накладных расходов на интерпретацию или компиляцию в машинный код.
* **Сложность написания и отладки** – требуется больше строк кода для простых операций, сложнее поддерживать.

**2. Отличие инструкции от директивы**

* **Инструкция** – выполняемая командой процессора операция (например, mov eax, 1).
* **Директива** – команда для ассемблера, не выполняемая процессором, а помогающая организовать код (например, SECTION .data).

**3. Основные правила оформления программ на ассемблере**

* Программа должна содержать **разделы** (SECTION .data, .bss, .text).
* Код начинается с **глобальной точки входа** (GLOBAL \_start).
* Команды должны быть **выравнены** и записаны в понятном порядке.
* В коде используются **комментарии** (; это комментарий).
* Данные определяются с помощью **меток** и **директив** (DB, DW, EQU).

**4. Этапы получения исполняемого файла**

1. **Написание кода** (файл .asm).
2. **Трансляция** (ассемблирование) с помощью NASM (nasm -f elf hello.asm -o hello.o).
3. **Компоновка** (линковка) с помощью LD (ld -m elf\_i386 hello.o -o hello).
4. **Запуск** (./hello).

**5. Назначение этапа трансляции**

* Преобразует **исходный код** (.asm) в **объектный файл** (.o).
* Проверяет **синтаксис** команд и директив.
* Размещает данные и инструкции в **памяти** программы.

**6. Назначение этапа компоновки**

* Соединяет объектные файлы в **исполняемый файл**.
* Добавляет **системные библиотеки** (если используются).
* Формирует таблицы **адресов и ссылок**.

**7. Какие файлы создаются при трансляции?**

* **Объектный файл** (.o) – результат работы ассемблера.
* **Исполняемый файл** (без расширения) – создаётся линковщиком.
* Возможны вспомогательные файлы (.lst – листинг, .map – карта памяти, .sym – символы).

**8. Форматы файлов для NASM и LD**

* **NASM** создаёт **объектные файлы** в формате **ELF** (-f elf для 32-битных, -f elf64 для 64-битных).
* **LD** создаёт **исполняемые файлы** в формате **ELF (Executable and Linkable Format)**.