

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: Ниemek 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
```

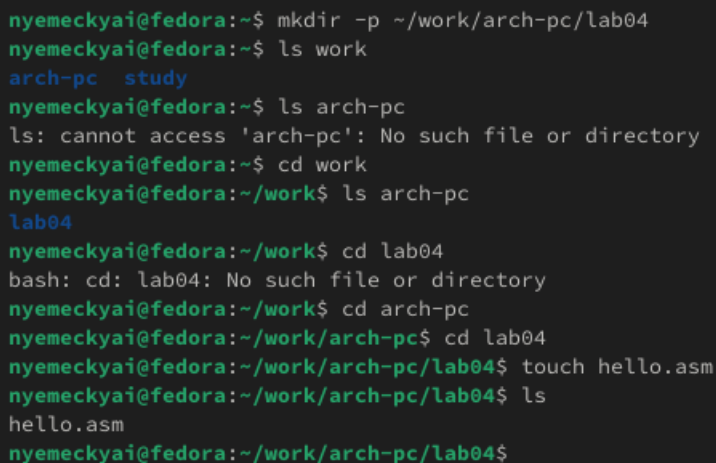
2. Создание каталога и файла

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
```



```
nyemeckyai@fedora:~$ mkdir -p ~/work/arch-pc/lab04  
nyemeckyai@fedora:~$ ls work  
arch-pc  study  
nyemeckyai@fedora:~$ ls arch-pc  
ls: cannot access 'arch-pc': No such file or directory  
nyemeckyai@fedora:~$ cd work  
nyemeckyai@fedora:~/work$ ls arch-pc  
lab04  
nyemeckyai@fedora:~/work$ cd lab04  
bash: cd: lab04: No such file or directory  
nyemeckyai@fedora:~/work$ cd arch-pc  
nyemeckyai@fedora:~/work/arch-pc$ cd lab04  
nyemeckyai@fedora:~/work/arch-pc/lab04$ touch hello.asm  
nyemeckyai@fedora:~/work/arch-pc/lab04$ ls  
hello.asm  
nyemeckyai@fedora:~/work/arch-pc/lab04$
```

3. Напиши код программы

Вставь в `hello.asm` следующий код:

```
SECTION .data  
hello: DB 'Hello world!',10  
helloLen: EQU $-hello
```

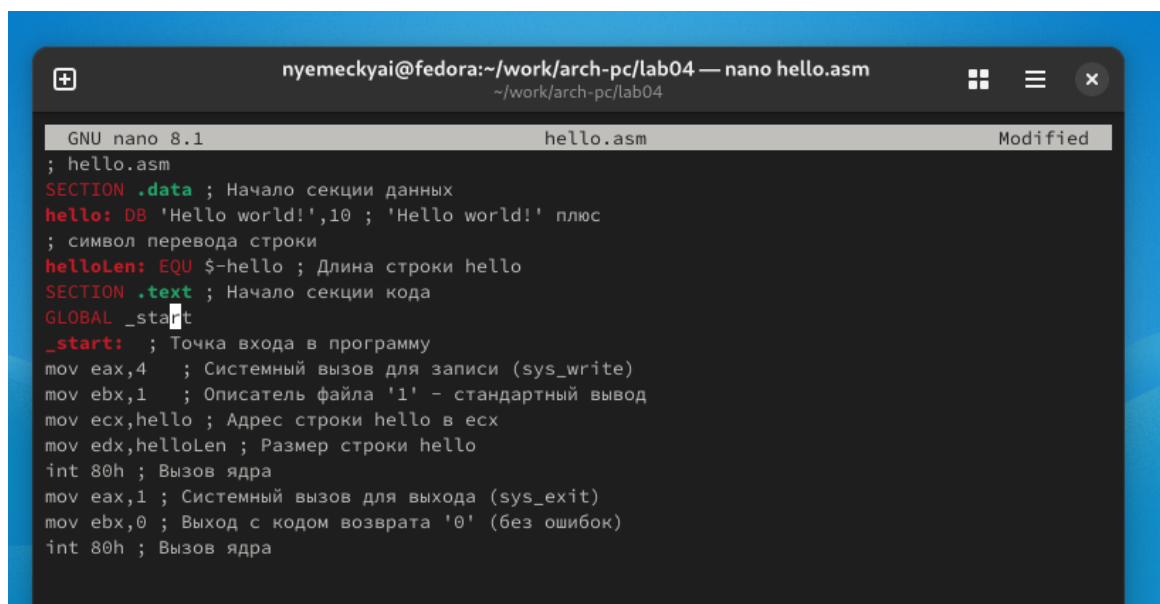
```

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

```

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
```

```
nyemeckyai@fedora:~/work/arch-pc/lab04$ nasm -f elf hello.asm
bash: nasm: command not found...
Install package 'nasm' to provide command 'nasm'? [N/y] y

* Waiting in queue...
The following packages have to be installed:
nasm-2.16.03-2.fc41.x86_64    A portable x86 assembler which uses Intel-like syntax
Proceed with changes? [N/y] y

* Waiting in queue...
* Waiting for authentication...
* Waiting in queue...
* Downloading packages...
* Requesting data...
* Testing changes...
* Installing packages...

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

6. Запуск программы

Запусти исполняемый файл:

```
./hello
```

Ты должен увидеть "Hello world!" в терминале.

```
nyemeckyai@fedora:~/work/arch-pc/lab04$ ls
hello.asm hello.o
nyemeckyai@fedora:~/work/arch-pc/lab04$

nyemeckyai@fedora:~/work/arch-pc/lab04$ nasm -f elf hello.asm
nyemeckyai@fedora:~/work/arch-pc/lab04$ ls
hello.asm hello.o
nyemeckyai@fedora:~/work/arch-pc/lab04$ nasm -o obj.o -f elf -g -l list.lst hello.asm
nyemeckyai@fedora:~/work/arch-pc/lab04$ ls
hello.asm hello.o list.lst obj.o
nyemeckyai@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 hello.o -o hello
nyemeckyai@fedora:~/work/arch-pc/lab04$ ls
hello hello.asm hello.o list.lst obj.o
nyemeckyai@fedora:~/work/arch-pc/lab04$
```

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:

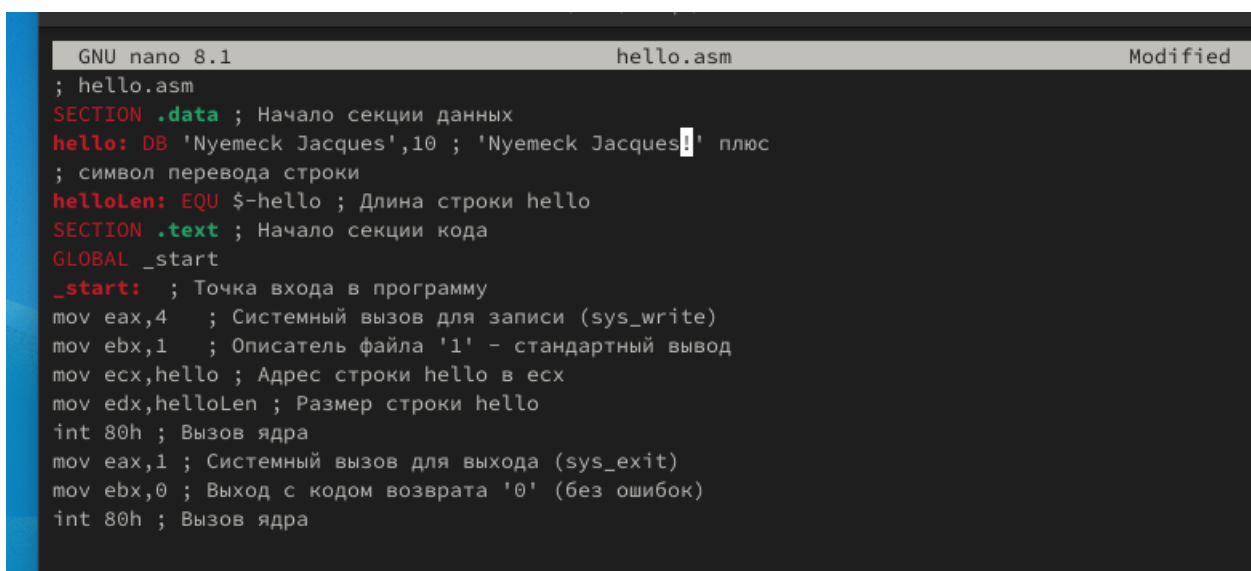
```
git init
git remote add origin <твой_репозиторий_на_GitHub>
```

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
```



```
GNU nano 8.1                                hello.asm                                Modified
; hello.asm
SECTION .data ; Начало секции данных
hello: DB 'Nyemetsk Jacques',10 ; 'Nyemetsk Jacques!' 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 esx
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
```

```
nyemeckyai@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 obj.o -o main
nyemeckyai@fedora:~/work/arch-pc/lab04$ ./hello
Hello world!
nyemeckyai@fedora:~/work/arch-pc/lab04$ ./main
Hello world!
nyemeckyai@fedora:~/work/arch-pc/lab04$
```

```
nyemeckyai@fedora:~/work/arch-pc/lab04$ nano lab4.asm
nyemeckyai@fedora:~/work/arch-pc/lab04$ nasm -f elf lab4.asm
nyemeckyai@fedora:~/work/arch-pc/lab04$ ld -m elf_i386 lab4.o -o lab4
nyemeckyai@fedora:~/work/arch-pc/lab04$ ./lab4
Nyemeck Jacques
nyemeckyai@fedora:~/work/arch-pc/lab04$
```

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)**.