Знакомство с NASM. Написание первой программы

Установка* NASM

** Все, что касается первоначальной инсталляции, несет рекомендательный характер, вы можете использовать любой ваш любимый дистрибутив (Ubuntu / Linux Mint / Fedora, CentOS, Debian etc), любой удобный эмулятор терминала, любой текстовый редактор и при необходимости любую виртуальную машину (Virtual box, QEMU, KVM, UTM etc).*

Для начала выполнения лабораторных работ вам потребуется ОС на ядре Linux.
Все следующие примеры будут действительны для Ubuntu 22.04

ВАЖНО! NASM - ассемблер для архитектуры **Intel x86**. Поэтому обратите на это внимание при установке дистрибутива.
Если у вас ARM и есть проблемы с эмуляцией x86, то можно обратиться за помощью к Анне Тимофеевне 🙂

Итак, у нас установлена замечательная ОС и удобный терминал (мы же настоящие программисты 💪)
Теперь перейдем к NASM -->
Для установки NASM с помощью `apt` вам необходимо ввести две команды:

```
sudo apt update
sudo apt -y install nasm
```

Готово! 🎉

Работа с Git

Мы будем выкладывать лабораторные работы через Git.
Поэтому, если у вас нет аккаунта на GitHub, то нужно обязательно его завести.
Если вы еще не знакомы с Git, рекомендую почитать этот гайд.

Для вас уже создан репозиторий на GitHub: /nasm-train/
Перед началом выполнения лабораторных работ вам необходимо:

- Сделать `fork` этого репозитория себе
- Создать от ветки `develop` свою новую с названием формата `lab<number>/ secondname-firstname`, _например lab1/gordina-anna
- Написать замечательный код на не менее замечательном языке ассемблера
- Запушить все изменения к себе в репозиторий
- Создать `Pull Request` в оригинальный репозиторий в ветку `develop`
- Радоваться, что все получилось

Создание первой программы

В качестве знакомства с языком ассемблера, вашей первой программой будет написание "Hello, <Ваше имя>".
Пример файла `hello.asm`, с текстом программы, которая выводит на экран строку "Hello"

```
SECTION .data                           ; Начало секции данных
    hello:      DB 'Hello',10
    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
    mov ebx,1           ; Описатель файла $1$ - стандартный вывод
    mov ecx,hello       ; Адрес строки hello в ecx
    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
```

Процесс создания программы

1. Текст программы можно набирать в текстовом редакторе, например, Vim(если знаете, как из него выйти), VS Code и др. Файлы с исходным текстом программы на языке ассемблера имеют расширение ``.asm
2. Трансляция - преобразование с помощью транслятора, в нашем случае 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` Тип файла листинга — `.lst`
3. Линковка - обработка объектного кода компоновщиком (`ld`), он принимает на вход объектные файлы и собирает по ним исполняемый файл
4. Запуск программы. Все мы стремимся написать работоспособный исполняемый файл. Если на одном из предыдущих шагах произошли ошибки, может присутствовать еще один этап - отладка программы при помощи специальной программы — отладчика.

Запуск NASM

Синтаксис командной строки nasm:
`nasm [-f <format>] [-o <output>] [-l <listing>] [params] <filename>`
Для вывода подробной информации: `man nasm`
Для получения списка форматов объектного файла: `nasm -hf`

Подробнее в главе 2 документации

Трансляция NASM

Для компиляции текста программы "Hello" необходимо написать:
`nasm -f elf64 hello.asm На данном этапе транслятор преобразует исходный текст программы из файла` hello.asm `в объектный код в файл` hello.o `Ключ` -f`` указывает транслятору, что требуется создать бинарные файлы в формате ELF. Формат elf64 позволяет создавать исполняемый код, работающий под 64-битными версиями Linux. Для 32-битных версий ОС указываем в качестве формата просто elf

Компоновка (линковка)

Формат команд компоновщика `ld` :
`ld [params] obj_files`

Для вывода подробной информации: `man ld`
Подробнее о формате: `ld --help`

Компоновщик преобразует объектный файл в исполняемый, для этого введите команду:
`ld -o hello hello.o`

Запуск исполняемого файла

Запустить на выполнение созданный исполняемый файл можно, набрав в командной строке:
`./hello`

Порядок выполнения работы

1. Установить nasm
2. Настроить работу с Git
3. Написать текст программы, который будет выводить строку "Hello, <Ваше имя>"
4. Оттранслировать полученный текст программы в объектный файл
5. Выполнить компоновку объектного файла и запустите исполняемый файл
6. Запушить свои изменения в свой fork репозиторий на GitHub
7. Создать Pull Request в оригинальный репозиторий на GitHub

Полезные ссылки

- Документация по NASM: https://www.nasm.us/xdoc/2.16.01/html/nasmdoc0.html
- Официальный сайт NASM: https://www.nasm.us
- GitHub NASM: https://github.com/netwide-assembler/nasm
- Андрей Викторович Столяров “Программирование на языке ассемблера NASM для ОС UNIX”: http://www.stolyarov.info/books/asm_unix
- Андрей Викторович Столяров Программирование: введение в профессию. II: низкоуровневое программирование: http://www.stolyarov.info/books/programming_intro/vol2