Сем1:

16 янв., 10:34 Без категории

Для чего мы изучаем ASM:
- Понимание что на самом деле исполняет процессор
- Отладка
- Оптимизация и анализ производительности
- Работа с тулчейном
- Понимание архитектуры
- Безопасность

Assembler - ов существует ли - ва
Для каждой архитектуры свой
Мы изучаем NASM -
- ассемблер для арх. x86

** пример с x86, пример с RISC-V **

Организация ассемблерного файла?

*** nasm -felf

* * *

- Данные отличаются от инструкций
- Секции нужны для того, чтобы процессор мог их различать.
- Информация о секциях хранится в таблице заголовков секций (section headers table)

***

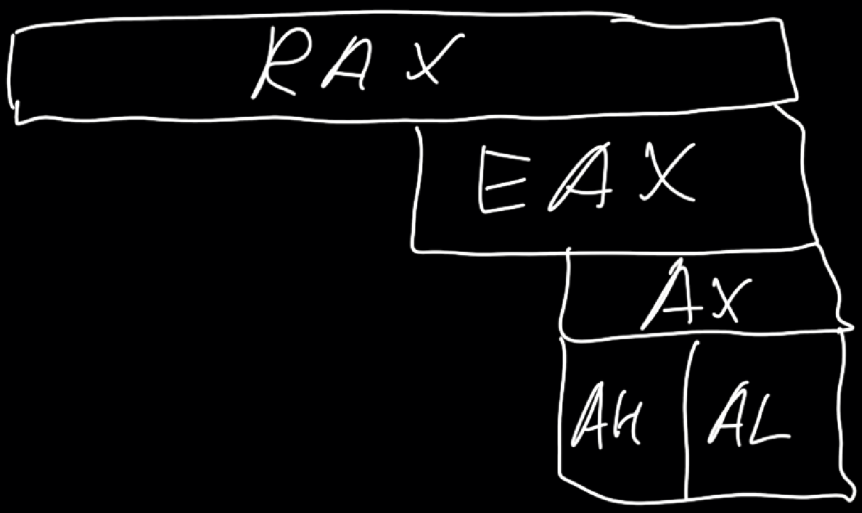
пример с readelf, objdump

***

- Секции:
  - .data - секция инициализированных данных (read + write)
  - .bss - неинициализированные данные
  - .text - инструкции программы (read + exec)
  - .rodata - данные только

Регистры

- 16 регистров общего назначения: сперва - 64 bit, второе - 32 bit



- rip - instruction pointer
- rax, eax - возвращаемое значение функции
- rsp, esp - указатель на вершину стека
- rbp, ebp - указатель на текущий фрейм
- rcx, ecx - счетчик цикла
- rsi, esi - source index, указатель на источник данных

— rdi, edi — destination index, указатель на место передачи данных

- Метки:

  main: " - метка: —

"

— задает адрес и связывает имя main с этим адресом, чтобы его мог использовать линкер

- „global main" —

— сделать символ глобальным, видимым для линкера

- Для него дальше метка

  main:

  ***

  попадет курсор с убирающей main и дальше с main

  ***

Реальная точка входа в программу
- _start, в которой ищется
метка _main

Линкер:
- производит компоновку
нескольких объектных файлов
и собирает из них программу