Московский физико-технический институт (государственный университет)
Физтех-школа радиотехники и компьютерных технологий
Кафедра информатики и вычислительной техники

Реализация в бинарном компиляторе драйвера NVMe-контроллера, работающего на уровне x86bios

Студент: Худякова Е. В., Б01-813а

Научный руководитель: к. ф.-м. н. Рожин А. Ф.

Москва 2022

Интерфейс NVM Express

NVMe (**N**on-**V**olatile **M**emory **E**xpress) — протокол доступа к устройствам хранения данных по линиям PCIe.

Преимущества:

- Высокая производительность (в сравнении с SATA)
- Подключение напрямую по шине PCI Express
- Параллелизм (64К очередей)

Бинарный компилятор

Динамическая двоичная трансляция - технология преобразования машинных кодов x86 в коды e2k в реальном времени.

Компилятор уровня системы Lintel позволяет **запускать гостевую ОС** в машинных кодах x86 на процессорах архитектуры Эльбрус, которая может использовать **свои** драйверы устройств.

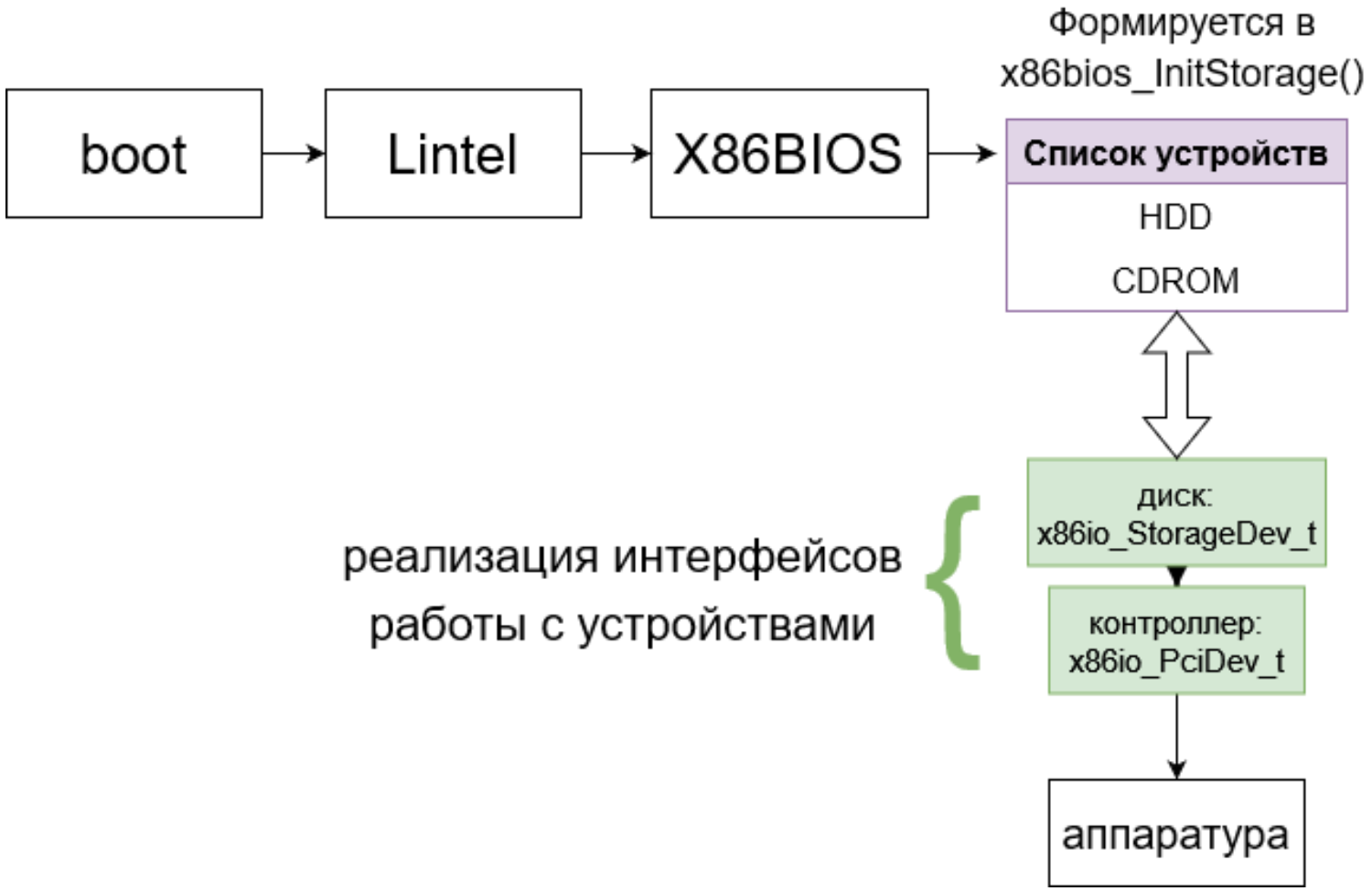
Проблема: до загрузки собственных драйверов гостевой ОС операции с диском и загрузка с него невозможны.

Решение: создание низкоуровневого драйвера устройства.

Функциональность BIOS, поддержанная в Lintel

Функция	Описание
x86bios_BootFromDev	Функция загрузки с устройства
x86bios_Int0x13Handler	Обработчик прерываний дискового сервиса BIOS
x86bios_InitStorage	Инициализация устройств хранения данных для работы с ними BIOS

Работа с устройствами хранения данных



Цель работы

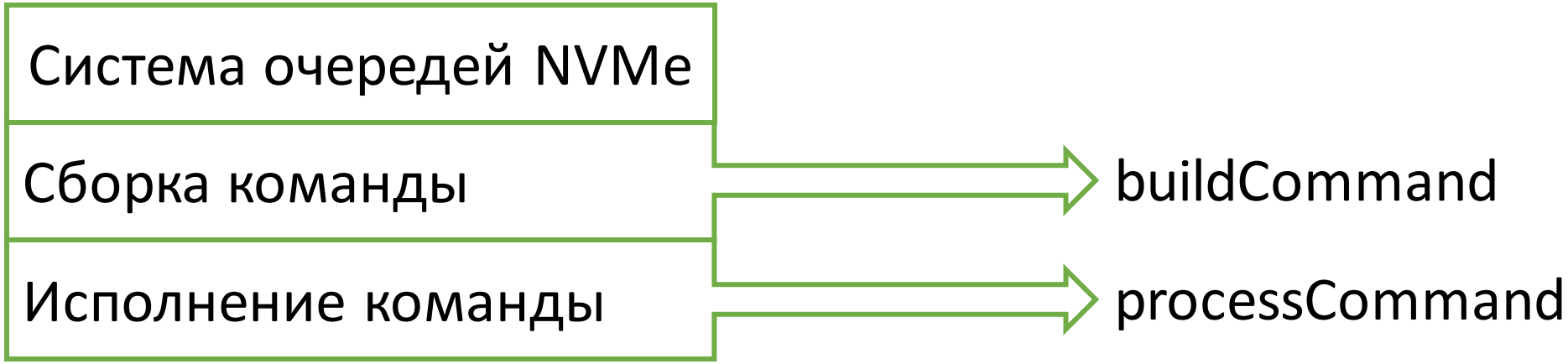
Поддержка опции загрузки ОС в кодах x86 на процессоре архитектуры Эльбрус с носителя с интерфейсом NVMe

Задачи:

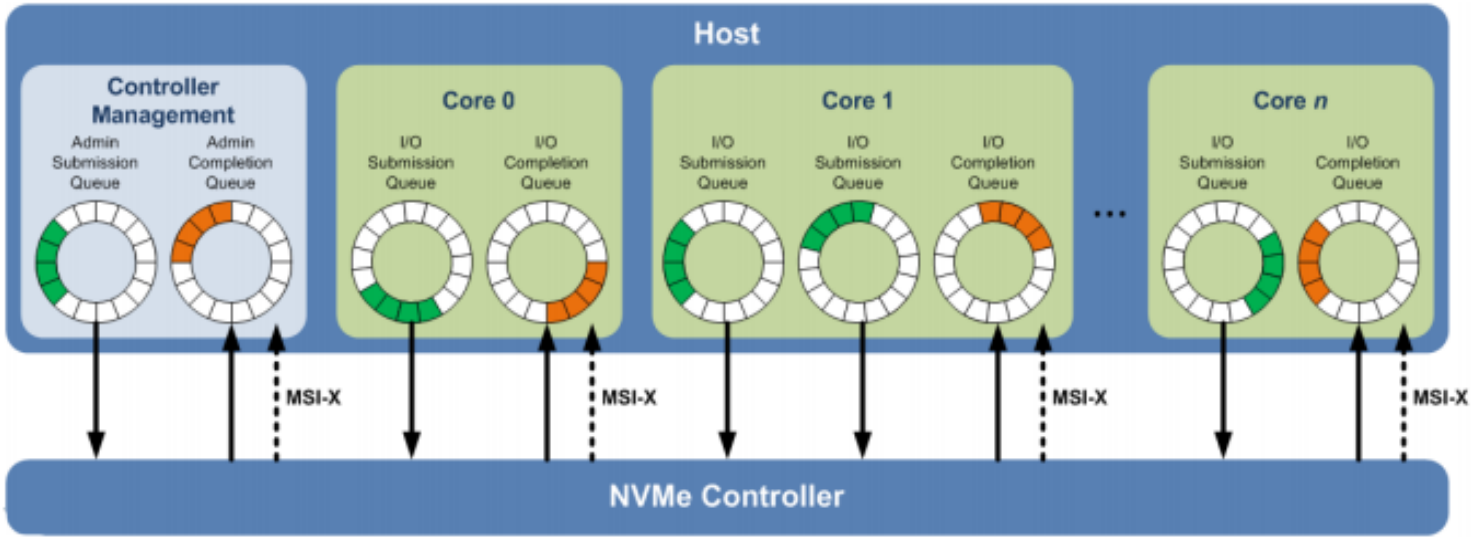
- Реализовать на C++ работу с NVMe-контроллером
- Реализовать на C++ работу с NVMe-устройством хранения данных
- Включить работу с NVMe в модель эмулируемого бинарным компилятором x86-компьютера

Реализация класса для работы с NVMe-контроллером

class x86io_PciDevNvm_t : x86io_PciDev_t



Система очередей



Вычисление адреса обращения к нужному слоту:

SQ slot = SQueues[y] + sq_tdbl[y] * SQ_ENTRY_SIZE

CQ slot = CQueues[y] + cq_hdbl[y] * CQ_ENTRY_SIZE

Переменные класса
x86io_PciDevNvm_t

Система очередей

Запись в очередь отправки

	31	23	15		7	0
DW0	Command Identifier		Reserved	F	Opcode	
DW1	Namespace Identifier (NSID)					
DW2	Reserved					
DW3						
DW4	Metadata Pointer (MPTR)					
DW5						
DW6	PRP Entry 1 (PRP1)					
DW7						
DW8	PRP Entry 2 (PRP2)					
DW9						
DW10	Command Specific					
...	...					
DW15	Command Specific					

Запись в очередь завершения

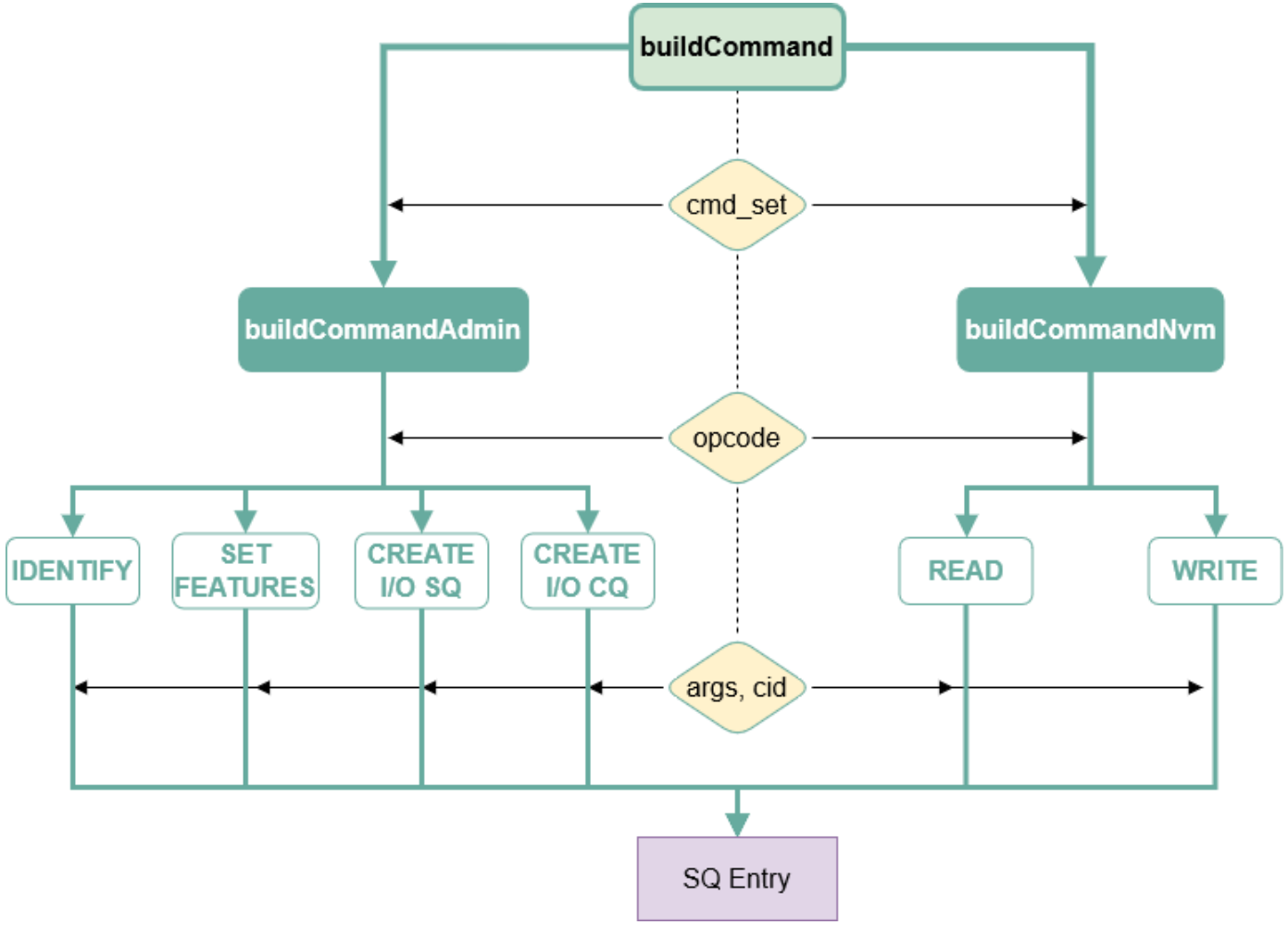
	31	23	15		7	0
DW0	Command Specific					
DW1	Reserved					
DW2	SQ Identifier			SQ Head Pointer		
DW3	Status Field			P	Command Identifier	

x86io_NvmCQEntry_t

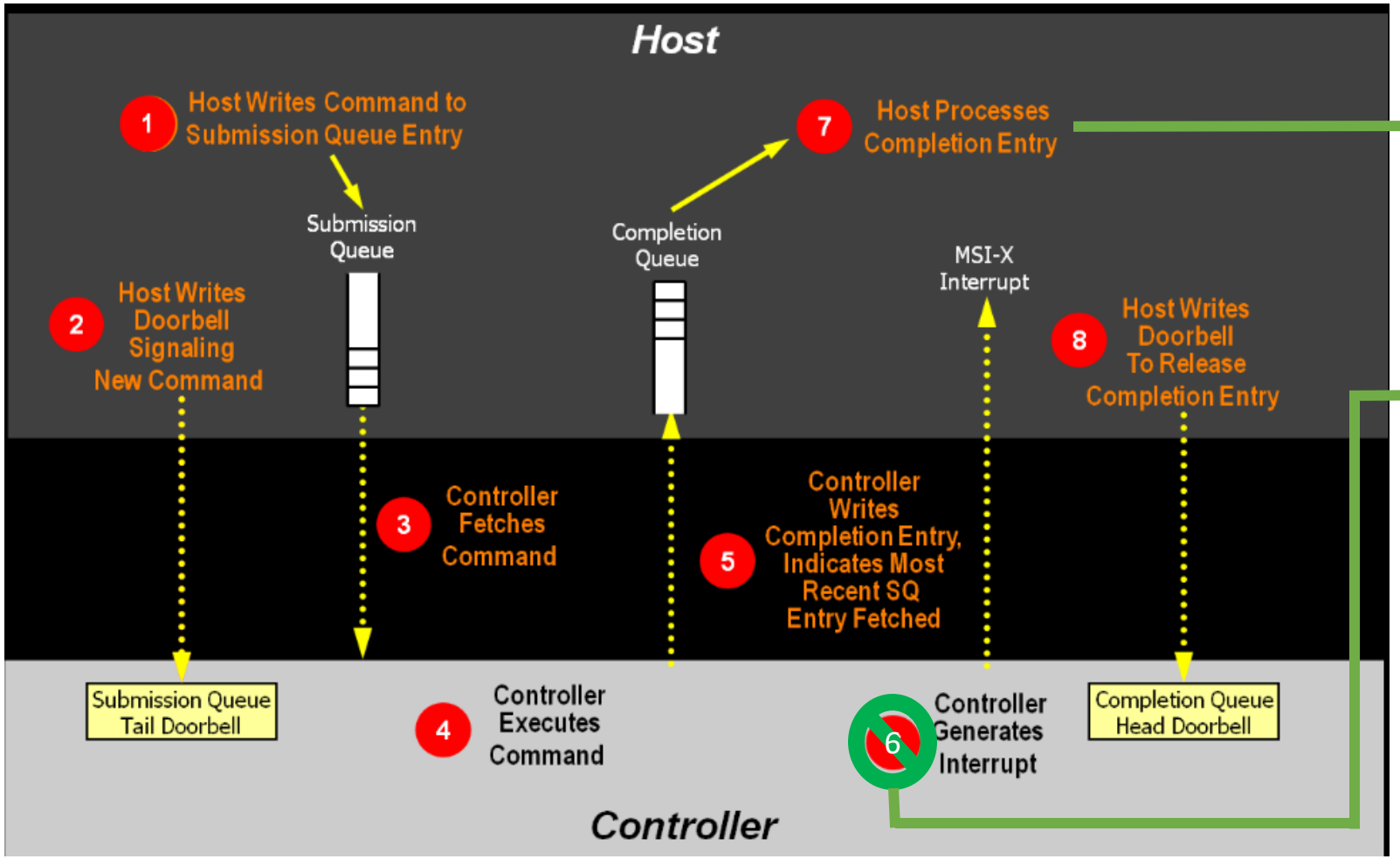
x86io_NvmSQEntry_t

Реализация класса для работы с NVMe-контроллером

Сборка команды



Исполнение команды



Происходит
за пределами метода
processCommand

Проверка бита
Interrupt Status
в регистре контроллера

Реализация класса для работы с NVMe-устройством хранения данных

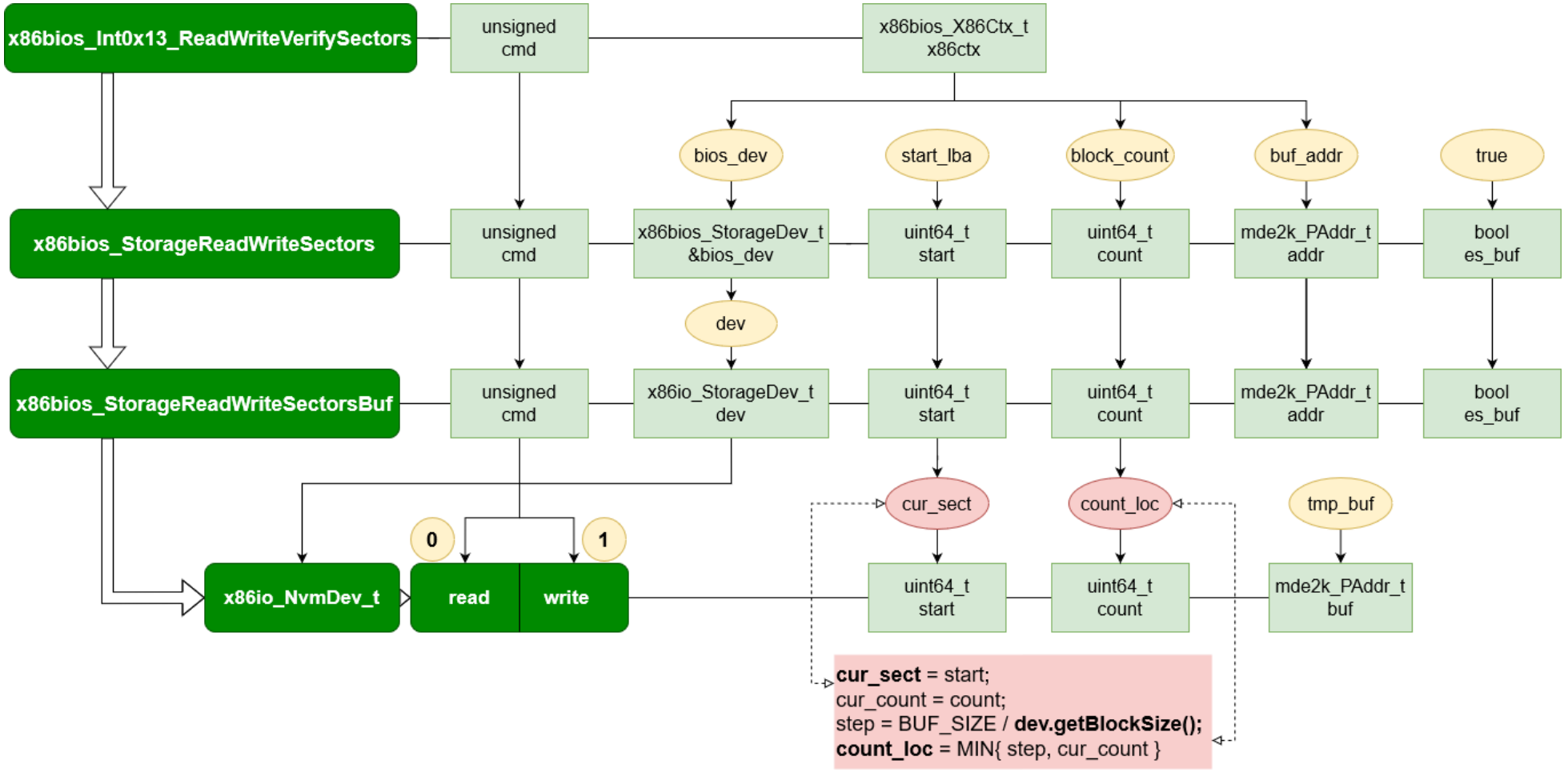
class x86io_NvmDev_t : x86io_StorageDev_t

NVMe Namespaces

- Индивидуальный формат и размер секторов
- Индивидуальная внутренняя нумерация секторов
- Обращение к пространству по идентификатору NSID

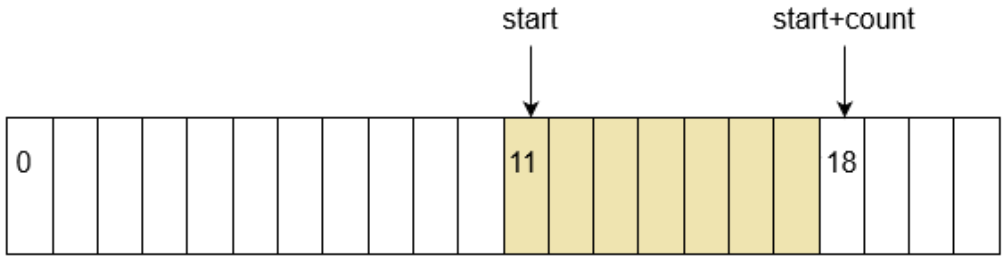
Реализация класса для работы с NVMe-накопителем

Обработка дисковых функций BIOS: чтение и запись

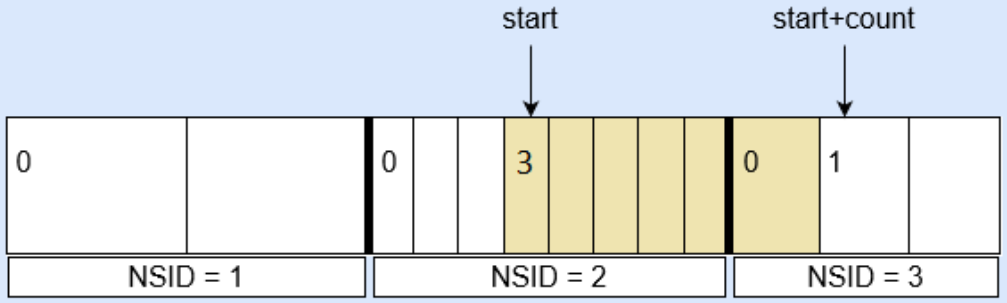


Проблема адресации пространств

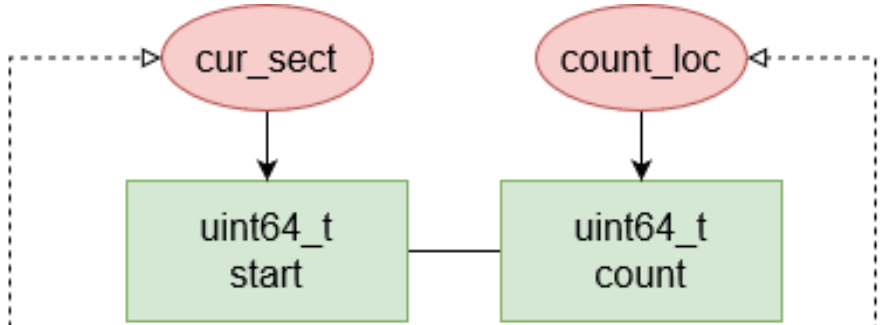
Разделение на пространства ⇒ **Некорректность аргументов**



start LBA = 11
count = 7



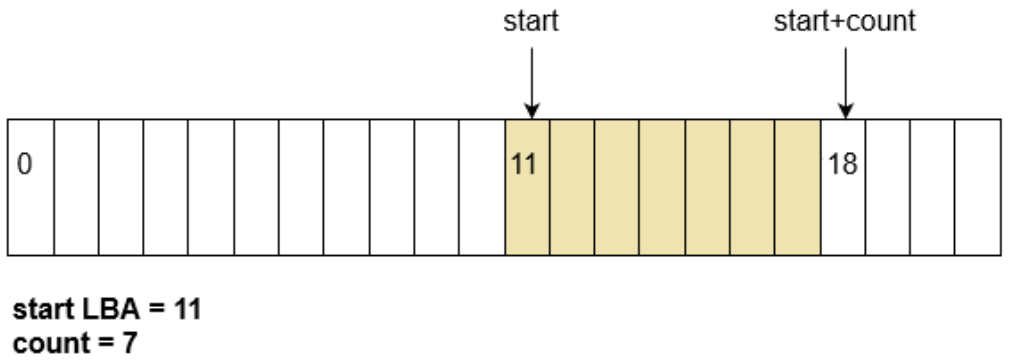
start NSID = 2
start LBA = 3
count = 6



cur_sect = start;
cur_count = count;
step = BUF_SIZE / **dev.getBlockSize();**
count_loc = MIN{ step, cur_count }

Решение в x86io_NvmDev_t

Разделение на пространства



- **getBlockSize:** возвращение 512
- **ns_data_[k].lbas:** получение размера сектора из массива данных о пространствах
- **prepareFormat:** преобразование аргументов и вычисление номера пространства

Чтение и запись секторов

- Предварительное форматирование аргументов с prepareFormat
- Проверка числа секторов на выход за пределы пространства имен
- Чтение/запись через выполнение команды READ/WRITE контроллера

Включение в X86BIOS

Новый тип устройств - STORAGE_DEV_NVMDRIVE

Функция	Правки
x86bios_BootFromDev	Добавлена ветка загрузки с устройства типа NVMDRIVE
x86bios_Int0x13Handler	Добавлен поиск устройств типа NVMDRIVE в функции, реализующие дисковый сервис BIOS
x86bios_InitStorage	Добавлены инициализация NVMe-устройств в функцию, инициализация геометрии для устройств типа NVMDRIVE

Результаты

- Реализован класс для работы с NVMe-контроллером (x86io_nvme.cpp)
- Реализована класс для работы с NVMe-устройством хранения данных (x86io_storage_nvme.cpp)
- Устройство типа NVMDRIVE добавлено в подсистему устройств хранения данных Lintel
- Проверена корректность работы модуля
  - Микропроцессор: "Эльбрус-8СВ"
  - NVMe SSD: Plextor M9Pe(Y)
  - ОС: Windows 10