

СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ



Центр параллельных вычислительных технологий

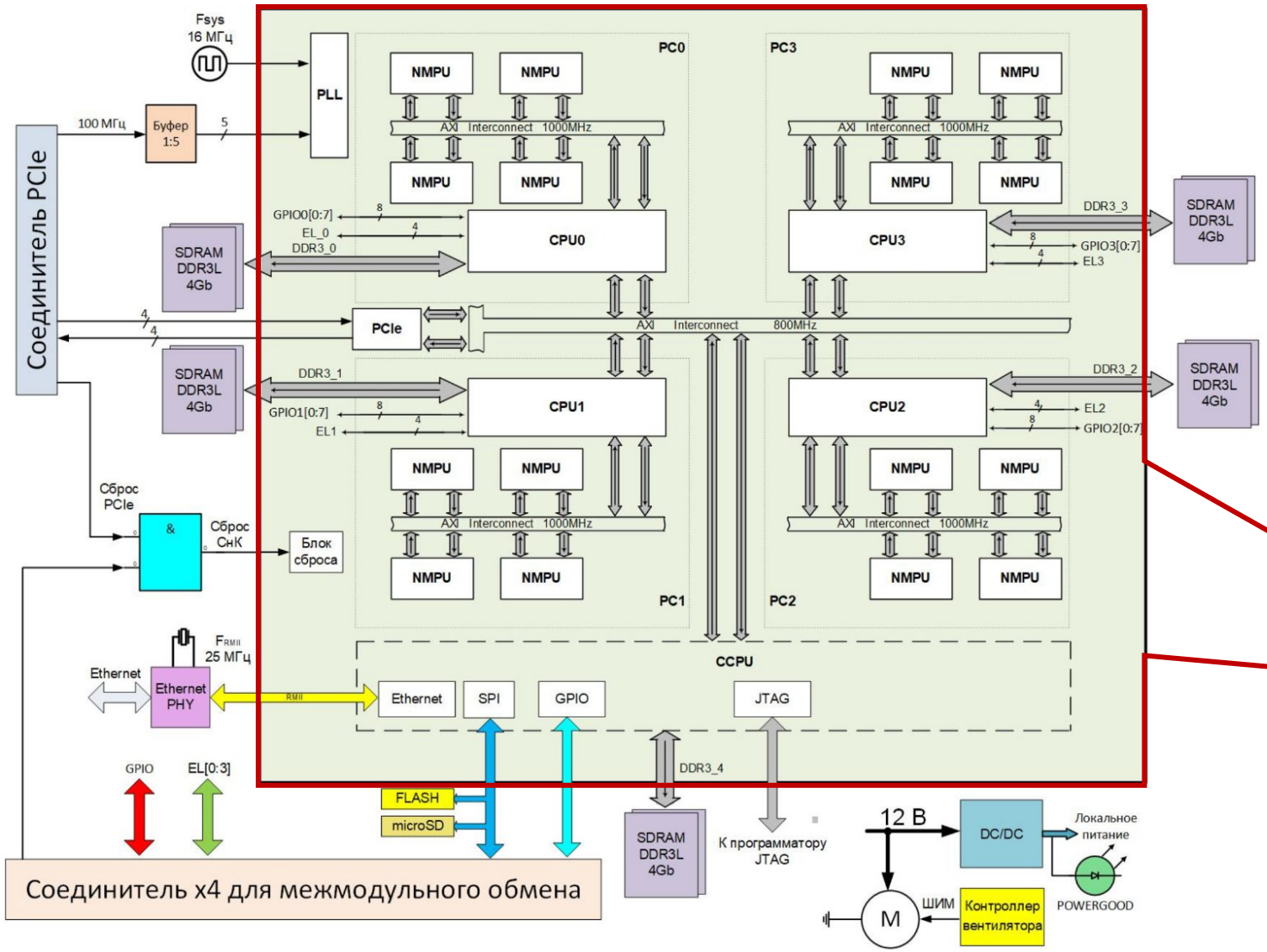
Семинар 3

Обмен данными с NM6408, параллельные вычисления на NMC-ядрах

Михаил Георгиевич Курносов

Цикл семинаров по архитектуре процессоров цифровой обработки сигналов НТЦ Модуль NeuroMatrix

Модуль NM Card Mini



- **СБИС К1879ВМ8Я (NM6408)**
  - 16 ядер NMC4
  - 5 ядер ARM Cortex-A5
- **SDRAM**: 5GB DDR3L
- **NIC**: Ethernet 100 Мб/с
- **PCIe** 2.0 x4
- **4 коммуникационных порта** E-Link для межмодульного обмена (до 16 ГБ/с)

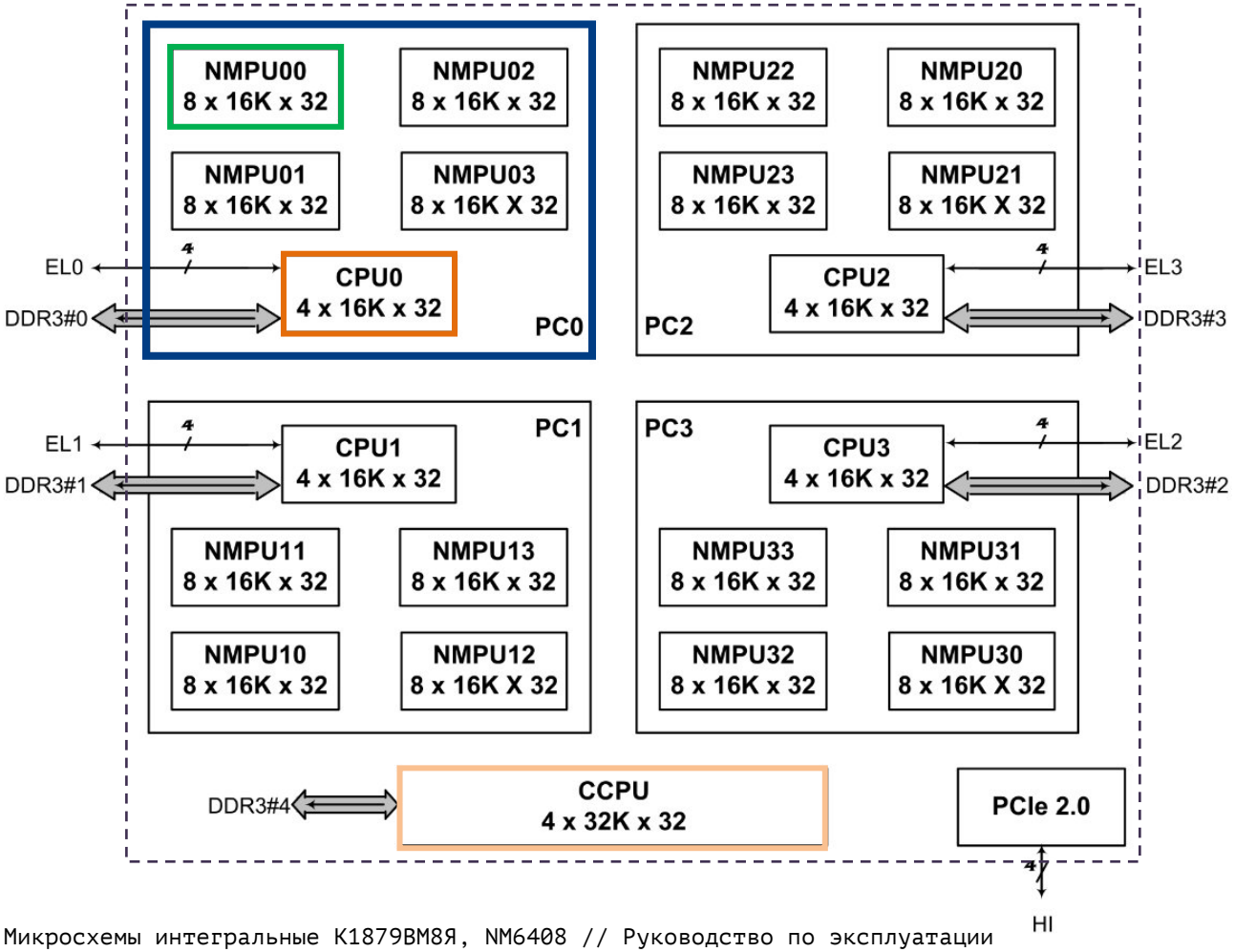


Функциональная схема Модуля ЮФКВ.469535.006 // Руководство по эксплуатации NM Card Mini

СБИС К1879ВМ8Я (SoC NM6408)



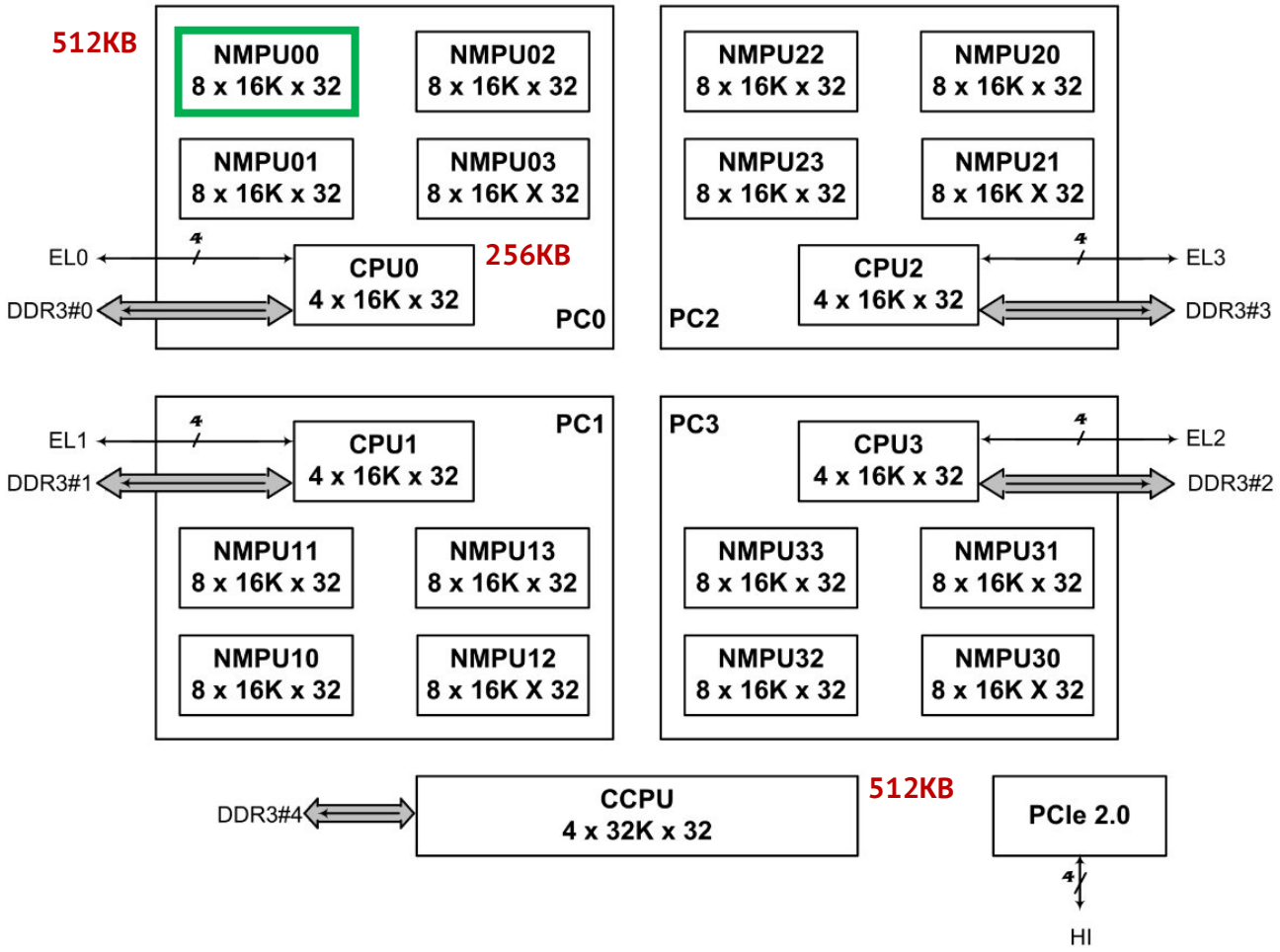
- **Гетерогенная многопроцессорная система на кристалле (СнК, SoC), 512 GFLOPS**
  - https://www.module.ru/products/1/26-18798
  - 16 процессорных ядер NeuroMatrix Core 4 (DSP)
  - 5 ядер ARM Cortex-A5 (управление системой)



- **Процессорный узел (ПУ)** – ядро ARM/NMC и 4 или 8 банков памяти (16Kx32 бита)
- **Процессорный кластер** (PC – Processing Cluster) – 4 ПУ на базе NMC, 1 ПУ ARM, контроллер DDR3, 128 GFLOPS
- **NMPU** (NeuroMatrix Processing Unit) – ПУ на базе ядра NMC4 (1 GHz), 8 банков памяти, 32 GFLOPS
- **CPU** (Cortex Processing Unit) – управляющий ПУ на базе ядра ARM (800 MHz), 4 банка памяти, доступ к DDR3, доступ к другим модулям через каналы EL
- **CCPU** (Central Cortex Processing Unit) - центральный управляющий ПУ на базе ARM (600 MHz), 4 банка внутренней памяти, управление системой и доступ к памяти DDR3 по 32-разрядной шине данных
- **Host** – системная плата с процессором x86-64 (Elbrus, ARM) к шине PCIe, которого подключен модуль
- **ARM-часть** – ПУ на ядре ARM SoC NM6408
- **NMC-часть** – ПУ на ядре NMC SoC NM6408
- **Board** – SoC NM6408 + его ресурсы

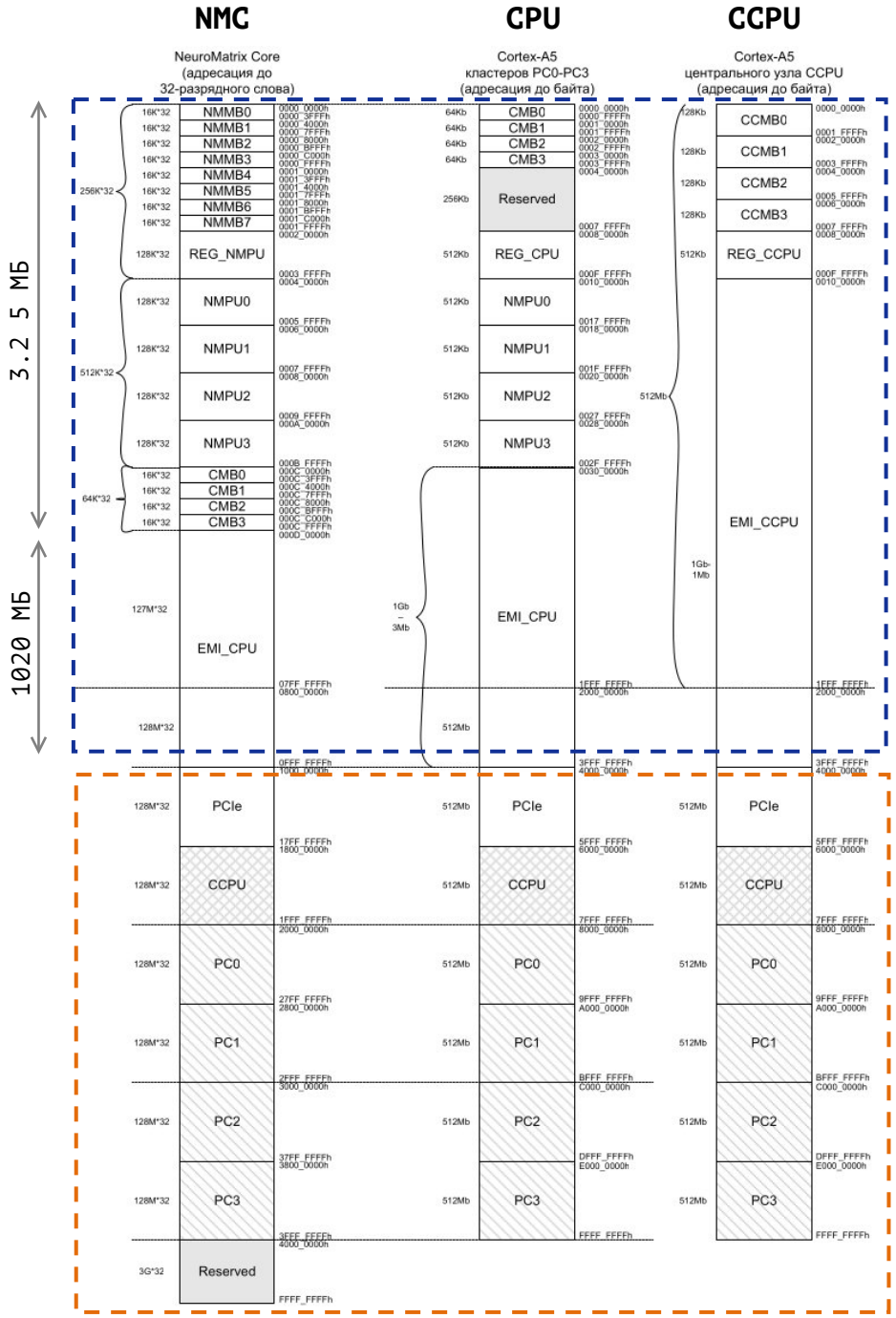
Иерархия памяти SoC NM6408

- **Каждый NMC** имеет 8 банков внутренней памяти SRAM (16K x 32 бита, 512КБ), адресация блоками по 32 бита
- **Каждый CPU** имеет 4 банка внутренней памяти (16K x 32 бита, 256КБ), адресация байтовая
- **CCPU** имеет 4 банка внутренней памяти (32K x 32 бита, 512КБ), адресация байтовая
- Внутренняя память: 76 Мбит (9.5 MB)
- Внешняя память DDR3: 40 Гбит (5 ГБ)
- Внутренняя память и DDR3 включена в единое адресное пространство, доступно всем ядрам ARM и NMC
- Единое адресное пространство в SoC охватывает 4ГБ, используются физические адреса
- Все NMPU имеют доступ к внутренней памяти друг друга и DDR3
- NMC и ARM формируют 32-разрядные адреса

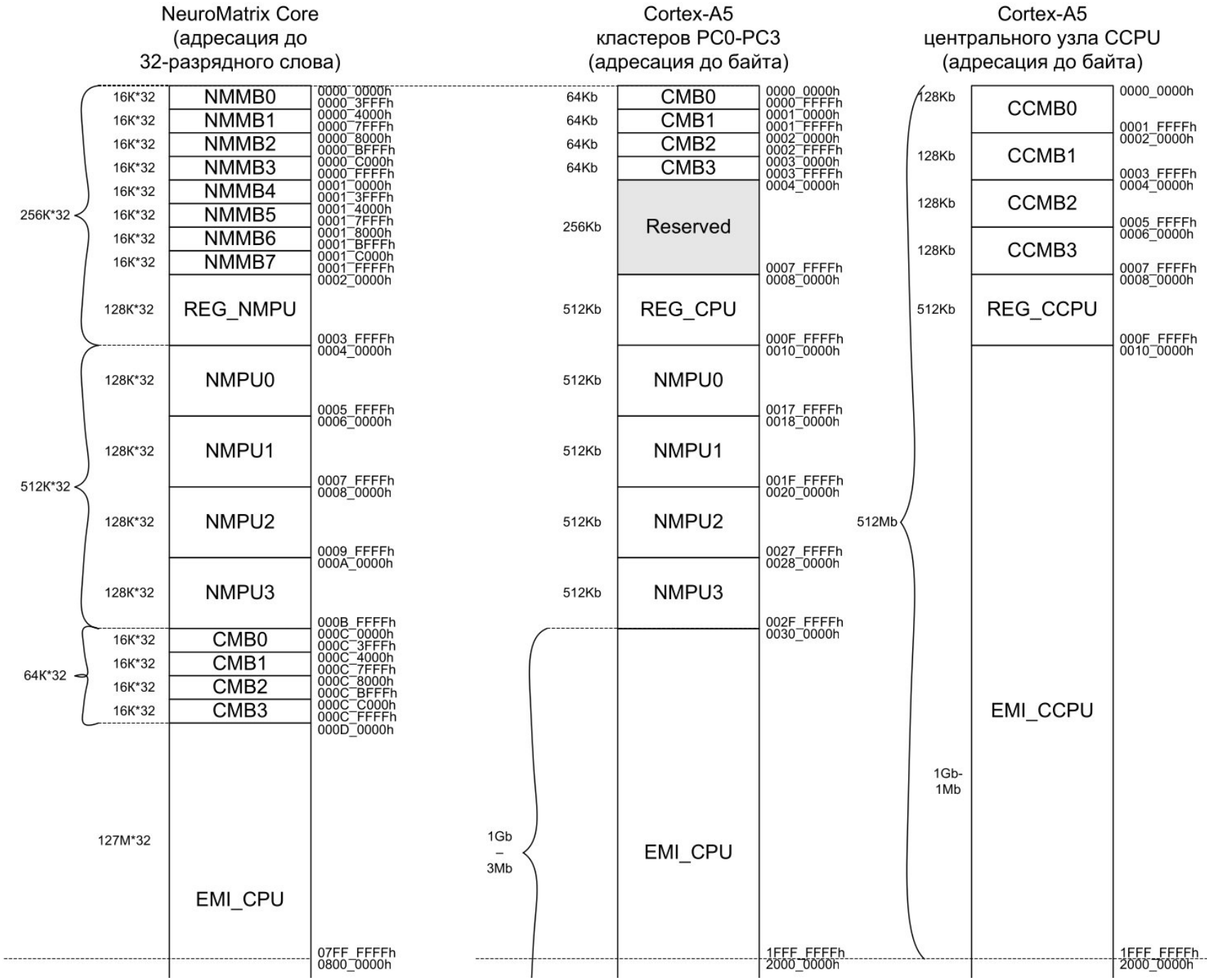


Карта памяти К1879ВМ8Я

- 7 областей памяти 0..4ГБ
- **Область младших адресов в 1 Гбайт (ПУ одного кластера):** адресация ячеек памяти разная для NMC, CPU, CCPU
  - **NMMB**i – банк i внутренней памяти NMPU (64KB), в банке NMMB0 хранятся программы обработки прерываний и начальный загрузчик (i=0,7)
  - **REG_NMPU** – регистры внешних устройств NMPU
  - **NMPU**j – внутренняя память NMPU j=0,3 кластера (512KB)
  - **CMB**j – банк j=0,3 внутренней памяти CPU (256KB)
  - **EMI_CPU** – банк внешней памяти DDR3 своего кластера (1ГБ), со стороны NMPU недоступны 1М младших ячеек
- **Адресное пространство в области 3 старших ГБ** одинаковое для любого типа процессора NMC, CPU, CCPU (6 областей):
  - **PCIe** – 512 Мбайт
  - **CCPU** – 512 Мбайт CCPU (младшая половина области 1 CCPU)
  - **PC0** – 512 Мбайт PC0 (младшая половина области 1 CPU 0)
  - **PC1** – 512 Мбайт PC1 (младшая половина области 1 CPU 1)
  - **PC2** – 512 Мбайт PC2 (младшая половина области 1 CPU 2)
  - **PC3** – 512 Мбайт PC3 (младшая половина области 1 CPU 3)
- Любому процессору при обращении к DDR-памяти другого кластера будет доступно не более 127M её ячеек (512MB)

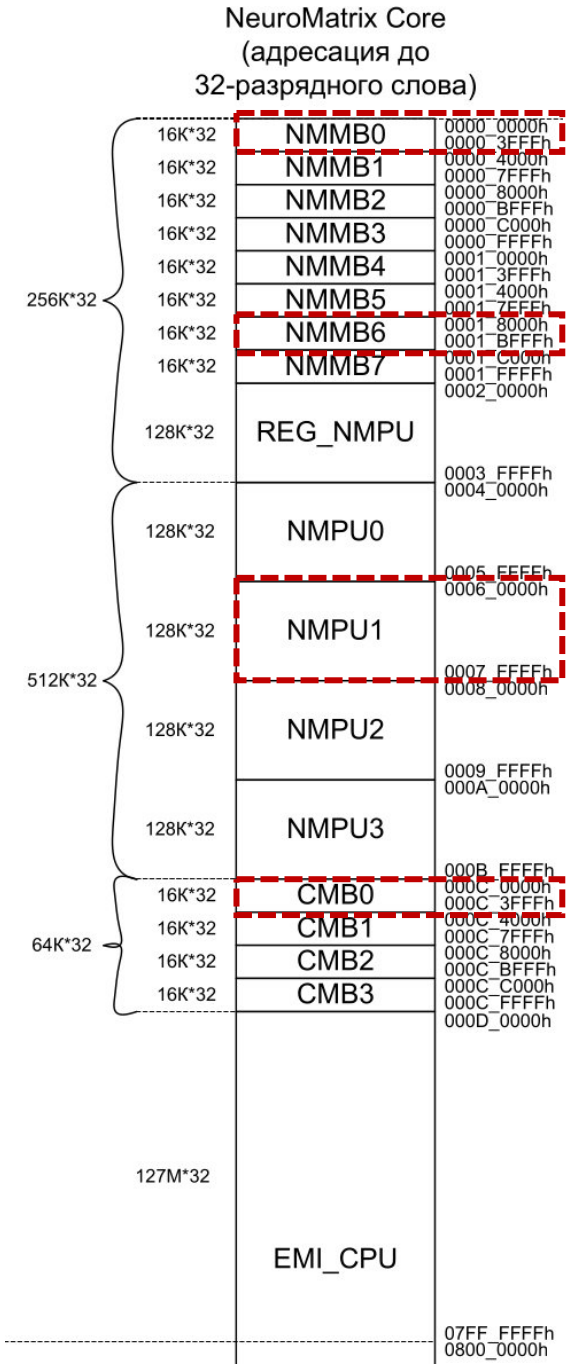


Карта памяти область младших адресов



Карта памяти NMC: область <= 1ГБ

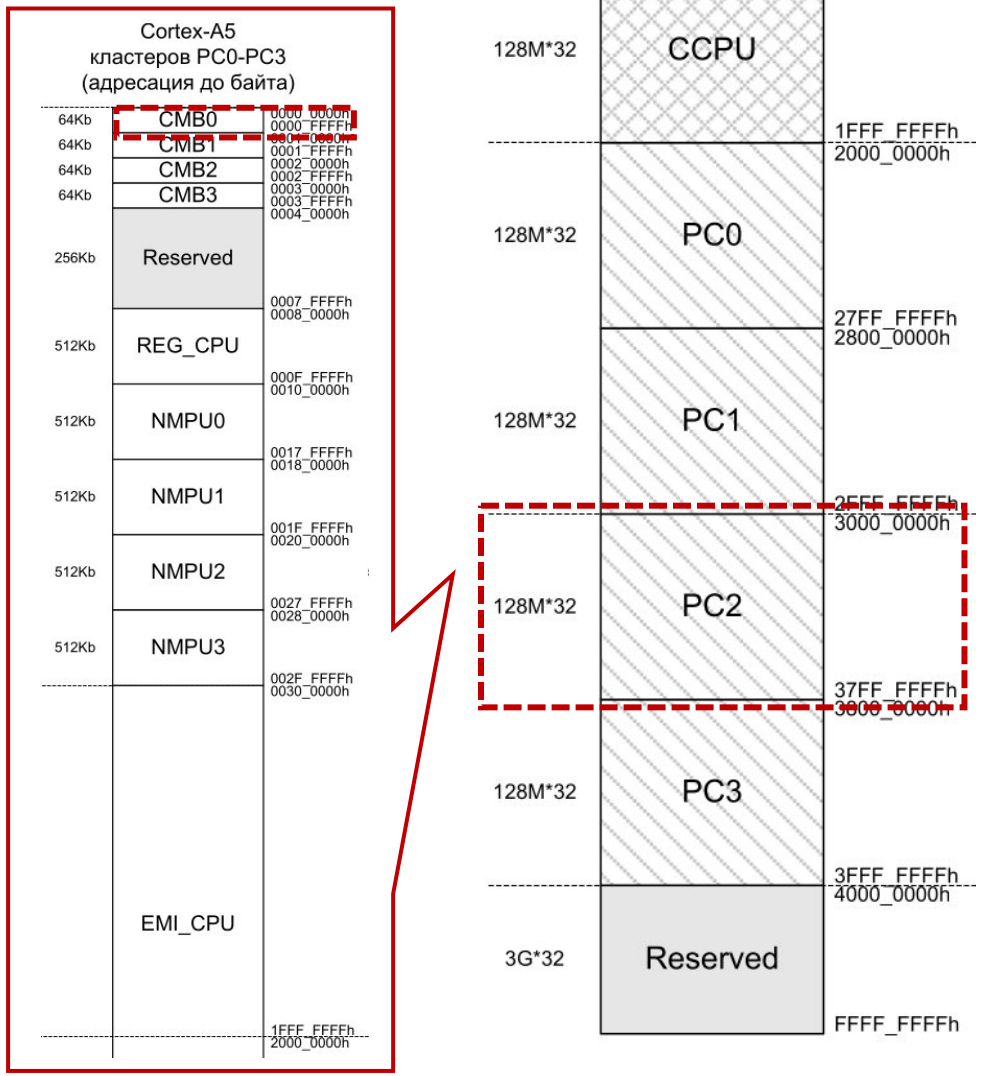
- **Область младших адресов в 1 Гбайт (ПУ одного кластера):**
  - **NMMB**i – банк i внутренней памяти NMPU (64KB), в банке NMMB0 хранятся программы обработки прерываний и начальный загрузчик (i=0,7)
  - **REG_NMPU** – регистры внешних устройств NMPU
  - **NMPU**j – внутренняя память NMPU j=0,3 кластера (512KB)
  - **CMB**j – банк j=0,3 внутренней памяти CPU (256KB)
  - **EMI_CPU** – банк внешней памяти DDR3 своего кластера (1ГБ), со стороны NMPU недоступны 1М младших ячеек
- Если NMC обращается по адресу 0x0000_0000 – маршрутизатор направляет запрос в банк NMMB0
- Если NMC обращается по адресу 0x0001_8000 – маршрутизатор направляет запрос в банк NMMB6
- Если NMC обращается по адресу 0x0006_0000 – это регион NMPU1, маршрутизатор направляет запрос к внутренней памяти NMPU1 текущего кластера, банк NMMB0
- Если NMC обращается по адресу 0x000C_0000 – это регион CMB0, маршрутизатор направляет запрос к внутренней памяти CPU текущего кластера, банк CMB0
- Если NMC обращается по адресу 0x000D_0000 – это регион EMI_CPU, маршрутизатор направляет запрос к DDR3 текущего кластера



Карта памяти NMC: область > 1ГБ

- **Адресное пространство в области 3 старших ГБ** одинаковое для любого типа процессора NMC, CPU, CCPU (6 областей):
  - **PCIe** – 512 Мбайт
  - **CCPU** – 512 Мбайт CCPU (младшая половина области 1 CCPU)
  - **PC0** – 512 Мбайт PC0 (младшая половина CPU 0)
  - **PC1** – 512 Мбайт PC1 (младшая половина CPU 1)
  - **PC2** – 512 Мбайт PC2 (младшая половина CPU 2)
  - **PC3** – 512 Мбайт PC3 (младшая половина CPU 3)

- Если NMC обращается по адресу 0x3000_0000 – это регион PC2, маршрутизатор направляет запрос в область памяти кластера 1, его CPU (младшая половина первого 1 ГБ), банк памяти CMB0
- Если обращение к PC2 выполнятся с NMC кластера 0, то происходит выход за пределы кластера на центральный коммутатор AXI 600MHz



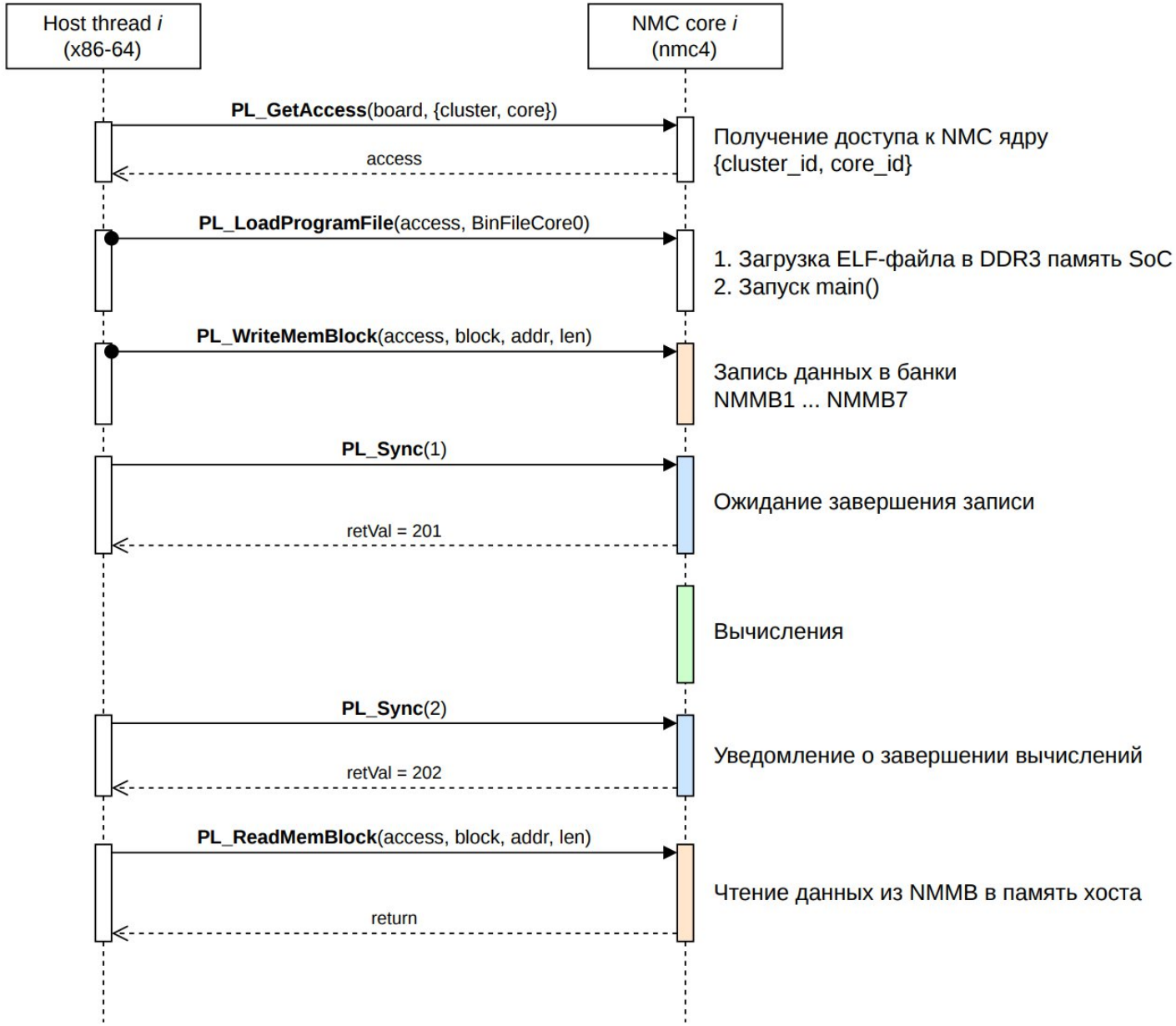
Запуск вычислений на 16 ядрах NMC

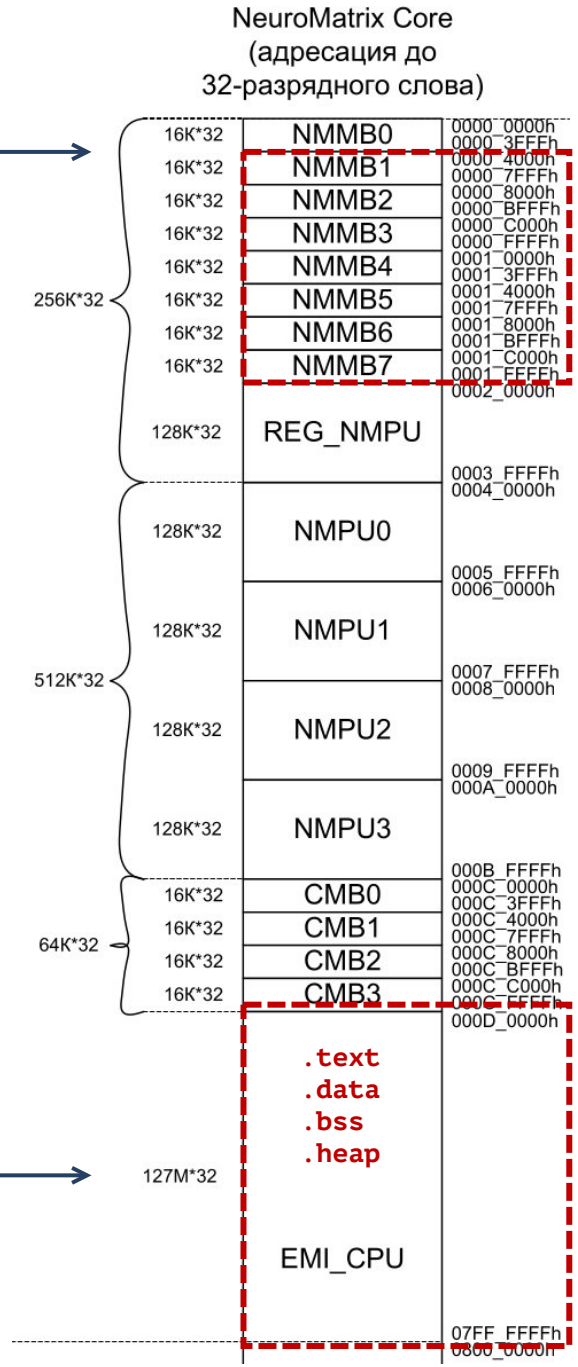
- **Host-сторона x86-64**
  - Главный поток запускает 16 потоков для управления каждым NMC-ядром в SoC NM6408
  - Поток для NMC ядра i:
    1. Запускает программу на NMC ядре (PL_LoadProgramFile)
    2. Передает данные во внутреннюю память NMC ядра (PL_WriteMemBlock)
    3. Уведомляет NMC ядро о завершении записи данных в его память (PL_Sync)
    4. Ожидает завершения вычислений NMC ядром (PL_Sync)
    5. Копирует данные из памяти NMC в память Host-машины (PL_ReadMemBlock)

- **NMC-сторона** – NMC ядро i
  1. Ожидает уведомления от Host-потока i о записи данных в память NMC (ncl_hostSync)
  2. Выполняет вычисления
  3. Уведомляет поток i о завершении вычислений (ncl_hostSync)

- **Где размещать данные на стороне NMC ядра?** NMMB (8x16Kx32b) или EMI (127Mx32b)
- **Как выделять буфер на стороне NMC?** фиксированный адрес, static (.data), dynamic (.heap)

[NMMB] Данные в локальной памяти NMMB

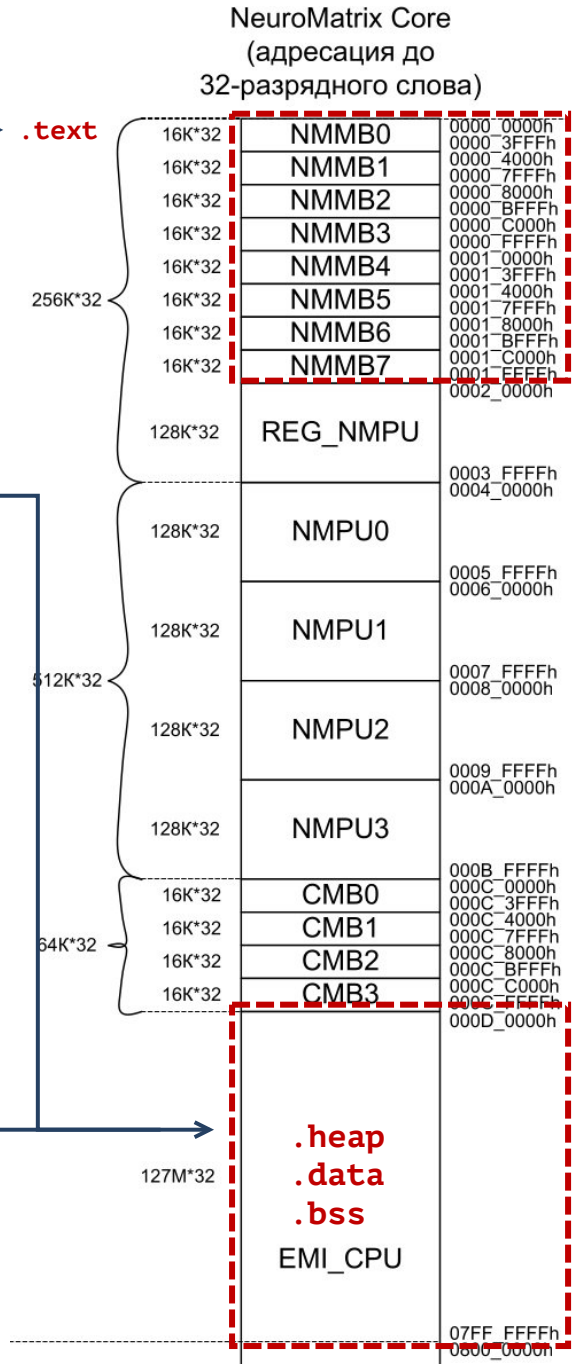




[EMI.heap] Данные в (DDR3, .heap)

- **Host-сторона x86-64**
  - Главный поток запускает 16 потоков для управления каждым NMC-ядром в SoC NM6408
  - Поток для NMC ядра i:
    1. Запускает программу на NMC ядре (PL_LoadProgramFile)
    2. **Передает в NMC размер блока (PL_SyncArray)**
    3. **Получат от NMC адрес буфера в .heap (DDR3, PL_SyncArray)**
    4. Передает данные в память NMC ядра (PL_WriteMemBlock)
    5. Уведомляет NMC ядро о завершении записи данных в его память (PL_Sync)
    6. Ожидает завершения вычислений NMC ядром (PL_Sync)
    7. Копирует данные из памяти NMC в память Host-машины (PL_ReadMemBlock)

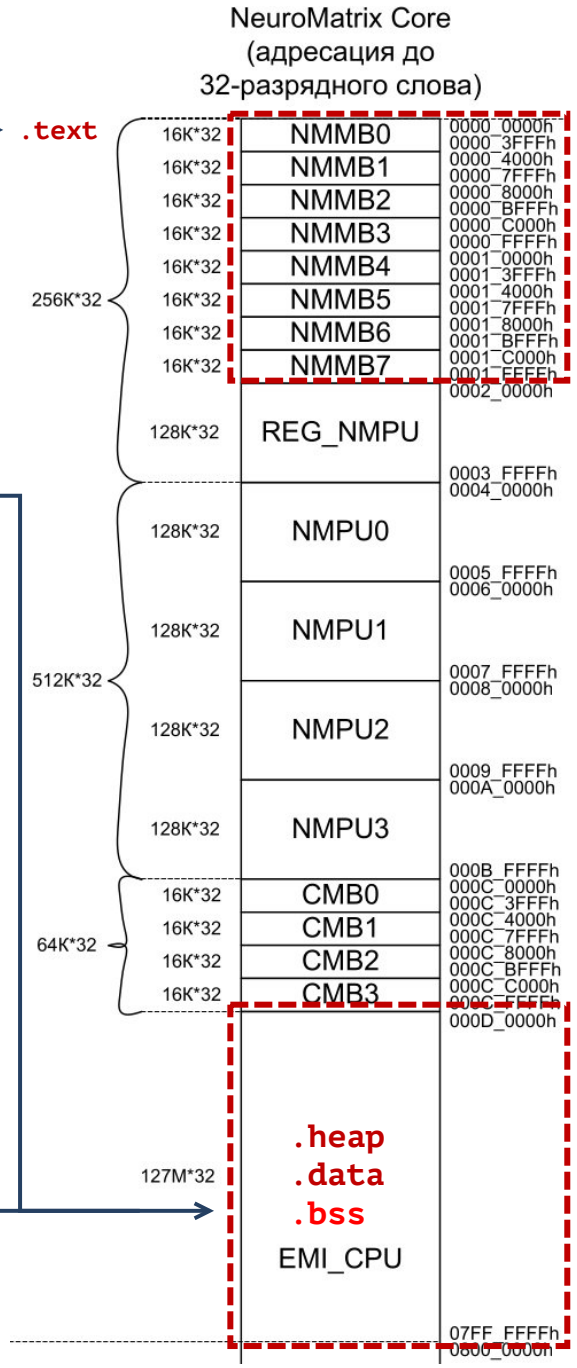
- **NMC-сторона** – NMC ядро i
  1. Получает от потока размер буфера (ncl_hostSyncArray)
  2. Динамически выделяет память под буфер: malloc()
  3. Передает хост-потоку адрес буфера (ncl_hostSyncArray)
  4. Ожидает уведомления от Host-потока i о записи данных в память NMC (ncl_hostSync)
  5. Выполняет вычисления
  6. Уведомляет поток i о завершении вычислений (ncl_hostSync)



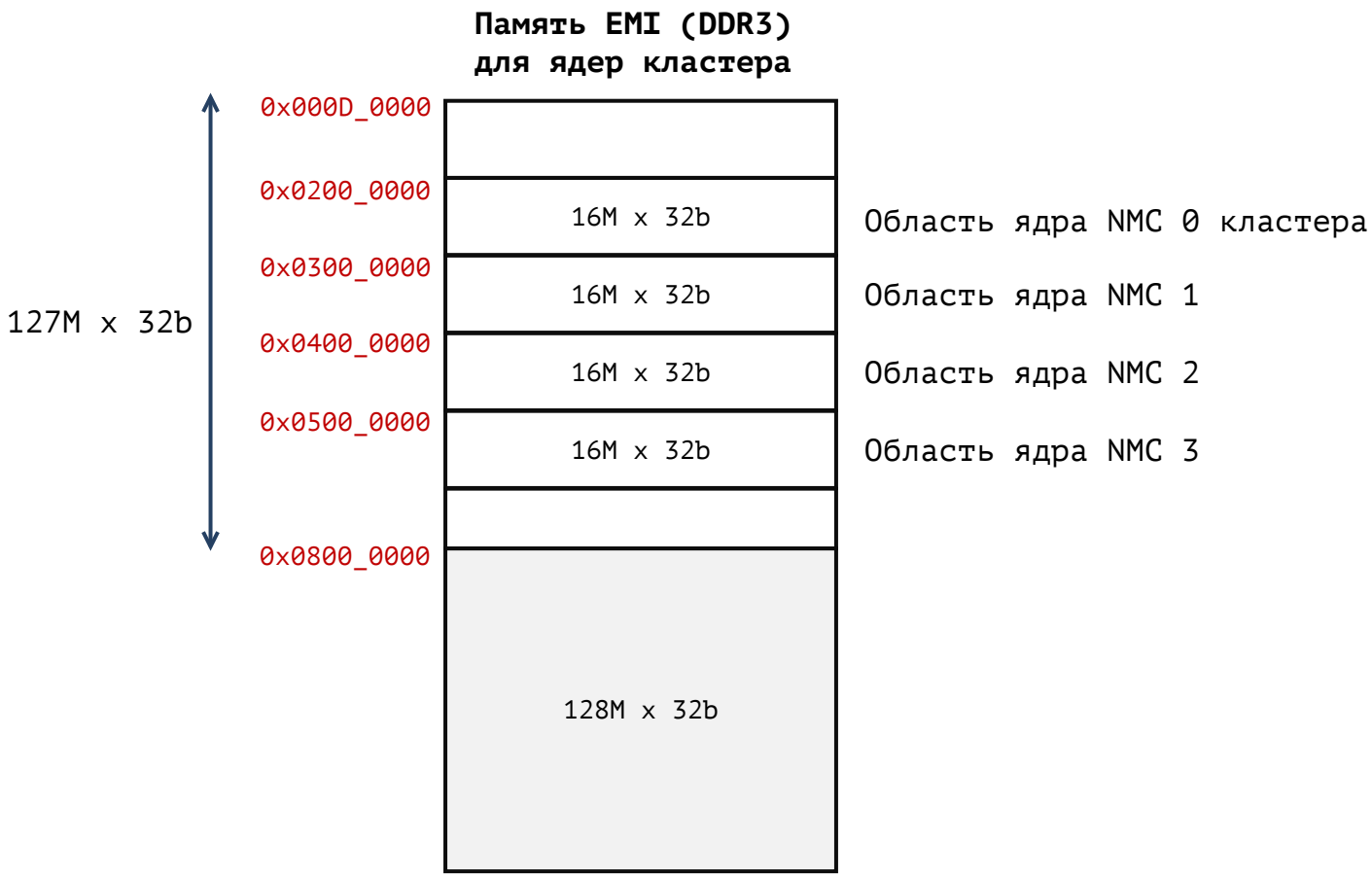
[EMI.bss] Данные в .bss (EMI)

- **Host-сторона x86-64**
  - Главный поток запускает 16 потоков для управления каждым NMC-ядром в SoC NM6408
  - Поток для NMC ядра i:
    1. Запускает программу на NMC ядре (PL_LoadProgramFile)
    2. **Передает в NMC размер блока (PL_SyncArray)**
    3. **Получат от NMC адрес буфера в .bss (DDR3, PL_SyncArray)**
    4. Передает данные в память NMC ядра (PL_WriteMemBlock)
    5. Уведомляет NMC ядро о завершении записи данных в его память (PL_Sync)
    6. Ожидает завершения вычислений NMC ядром (PL_Sync)
    7. Копирует данные из памяти NMC в память Host-машины (PL_ReadMemBlock)

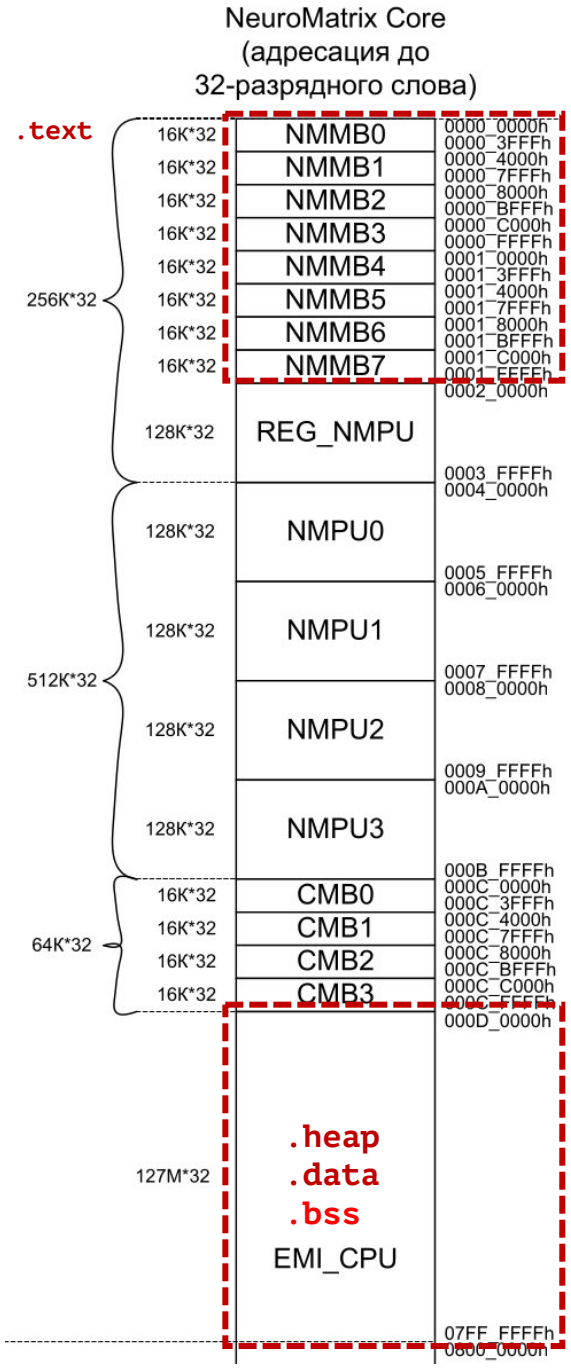
- **NMC-сторона** – NMC ядро i
  1. Получает от потока размер буфера (ncl_hostSyncArray)
  2. Динамически выделяет память под буфер: malloc()
  3. Передает хост-потоку адрес буфера (ncl_hostSyncArray)
  4. Ожидает уведомления от Host-потока i о записи данных в память NMC (ncl_hostSync)
  5. Выполняет вычисления
  6. Уведомляет поток i о завершении вычислений (ncl_hostSync)



[EMI.bss] Данные в .bss (EMI)



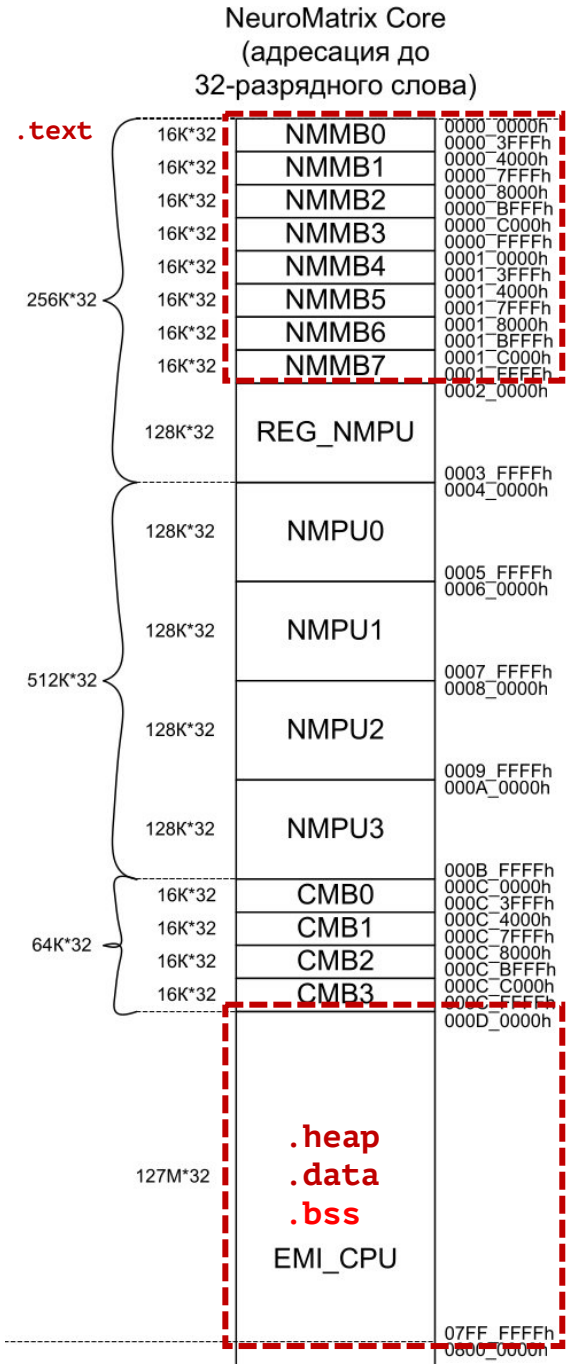
Конфигурация отображения секций в память
(lds-скрипт загрузчика)



[EMI.bss] Данные в .bss (EMI) **20M**



Конфигурация отображения секций в память
(lds-скрипт загрузчика)



Вызов функций NMPP из кода на NMC

```
enum { SIZE_MAX = 1 * 1024 * 1024};
double x[SIZE_MAX];
double y[SIZE_MAX];

int nmc_kernel()
{
    int cluster_id = ncl_getClusterID();
    int core_id = ncl_getCoreID();
    int core_index = cluster_id * 4 + core_id;

    /* Получаем размер блока от хоста */
    size_t buf_size;
    ncl_hostSyncArray(1, NULL, 0, NULL, &buf_size);

    /* Передаем адрес буфера хосту */
    ncl_hostSyncArray(2, x, 0, NULL, NULL);
    ncl_hostSyncArray(3, y, 0, NULL, NULL);

    /* Барьерная синхронизация с host-процессом, ждем готовности входных данных */
    ncl_hostSync(4);

    /* Вычисления */
    double res = nmblas_ddot(buf_size, x, 1, y, 1);

    printf("# nmc kernel: %2d.%2d, index %2d: ddot: %.4f\n", cluster_id, core_id,
           core_index, res);

    /* Барьерная синхронизация с host-процессом, уведомляем о завершении вычислений */
    ncl_hostSync(5);
    return 0;
}
```

