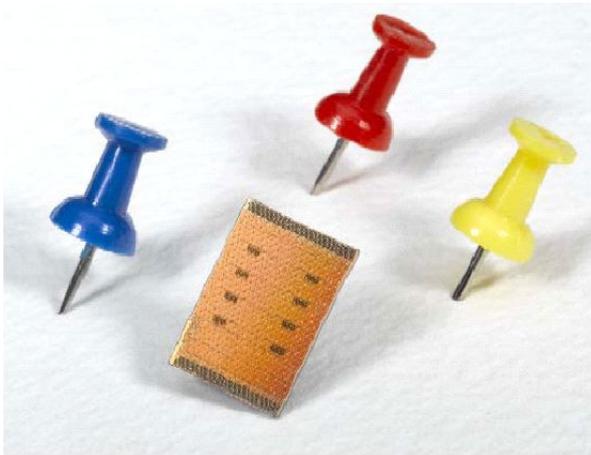




# Программирование процессора Cell

Киреев С.Е.





# Сравнение современных процессоров

Процессор	Год	Число ядер	Частота	Пиковая производительность	
				одинарная точность	двойная точность
Intel Itanium 9150M (Montvale)	2007	2	1.66 GHz	$2 \times 13.28 =$ <b>26.56 GFLOPS</b>	$2 \times 6.64 =$ <b>13.28 GFLOPS</b>
Intel Xeon X7460 (Dunnington)	2008	6	2.66 GHz	$6 \times 21.28 =$ <b>127.68 GFLOPS</b>	$6 \times 10.64 =$ <b>63.84 GFLOPS</b>
Intel Xeon i7 965 EE (Bloomfield)	2008	4	3.2 GHz	$4 \times 25.6 =$ <b>102.4 GFLOPS</b>	$4 \times 12.8 =$ <b>51.2 GFLOPS</b>
AMD Opteron 8384 (Shanghai)	2008	4	2.7 GHz	$4 \times 10.8 =$ <b>86.4 GFLOPS</b>	$4 \times 10.8 =$ <b>43.2 GFLOPS</b>
AMD Phenom X4 9950 (Agena)	2008	4	2.6 GHz	$4 \times 20.8 =$ <b>83.2 GFLOPS</b>	$4 \times 10.4 =$ <b>41.6 GFLOPS</b>
IBM PowerXCell 8i	2008	1 PPE 8 SPE	3.2 GHz	$8 \times 25.6 + 25.6 =$ <b>230.4 GFLOPS</b>	$8 \times 12.8 + 6.4 =$ <b>108.8 GFLOPS</b>

# Реализации систем на базе Cell

- Roadrunner – самый мощный суперкомпьютер в мире на базе процессоров Opteron и Cell

## TOP500 List - June 2009 (1-100)

Rank	Site	Computer/Year Vendor	Cores	R <sub>max</sub>	R <sub>peak</sub>	Power
1	DOE/NNSA/LANL United States	Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband / 2008 IBM	129600	1105.00	1456.70	2483.47
2	Oak Ridge National Laboratory United States	Jaguar - Cray XT5 QC 2.3 GHz / 2008 Cray Inc.	150152	1059.00	1381.40	6950.60
3	Forschungszentrum Juelich (FZJ) Germany	JUGENE - Blue Gene/P Solution / 2009 IBM	294912	825.50	1002.70	2268.00
4	NASA/Ames Research Center/NAS United States	Pleiades - SGI Altix ICE 8200EX, Xeon QC 3.0/2.66 GHz / 2008 SGI	51200	487.01	608.83	2090.00
5	DOE/NNSA/LLNL United States	BlueGene/L - eServer Blue Gene Solution / 2007 IBM	212992	478.20	596.38	2329.60

# Реализации систем на базе Cell

## □ Системы на базе Cell – самые «зеленые»

June 2009

Green500 Rank	MFLOPS/W	Site*	Computer*	Total Power (kW)	TOP500 Rank*
1	536.24	Interdisciplinary Centre for Mathematical and Computational Modeling, University of Warsaw	BladeCenter QS22 Cluster, PowerXCell 8i 4.0 Ghz, Infiniband	34.63	422
2	458.33	DOE/NNSA/LANL	BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHZ, Infiniband	138.00	61
2	458.33	IBM Poughkeepsie Benchmarking Center	BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHZ, Infiniband	138.00	62
4	444.94	DOE/NNSA/LANL	BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHZ, Voltaire Infiniband	2483.47	1
5	428.91	National Astronomical Observatory of Japan	GRAPE-DR accelerator Cluster, Infiniband	51.20	277
6	371.67	ASTRON/University Groningen	Blue Gene/P Solution	94.50	124
7	371.67	IBM - Rochester	Blue Gene/P Solution	126.00	84
7	371.67	IBM Thomas J. Watson Research Center	Blue Gene/P Solution	126.00	85
7	371.67	Max-Planck-Gesellschaft MPI/IPP	Blue Gene/P Solution	126.00	86

# Реализации систем на базе Cell

## 1. Продукты IBM

- IBM BladeCenter QS 21
  - 2 × Cell B.E. 3.2 GHz
- IBM BladeCenter QS 22
  - 2 × PowerXCell 8i 3.2 GHz
  - 2 × PowerXCell 8i 4.0 GHz

## 2. Продукты Mercury Computer Systems

- Mercury Dual Cell Based System 2
  - 2 × Cell B.E. 3.2 GHz
- Mercury Dual Cell Based Blade 2
  - 2 × Cell B.E. 3.2 GHz
- Mercury PCI Express Cell Accelerator Board 2
  - 1. 1 × Cell B.E. 2.8 GHz

## ■ Продукты Fixstars

- Fixstars GagaAccell 180 Accelerator Board
  - 1 × PowerXCell 8i 2.8 GHz

## ■ Продукты T-Платформы

- PeakCell S
  - 2 × PowerXCell 8i 3.2 GHz
- 1. PeakCell W
  - 2 × PowerXCell 8i 3.2 GHz
- PeakCell YPS
  - 4 × PowerXCell 8i 3.2 GHz





- ▶ Apple
- ▶ Роботы
- ▶ Компьютеры
- ▶ Мониторы
- ▶ Комплекующие
- ▶ Периферийные устройства
- ▶ Оборудование 220В
- ▶ Ноутбуки
- ▶ Карманные компьютеры, коммуникаторы, GPS-навигаторы
- ▶ Игровые приставки, игры и аксессуары
- ▶ Фото и видео
- ▶ Устройства хранения и переноса данных
- ▶ Элементы питания, зарядные устройства
- ▶ Плееры и диктофоны
- ▶ Наушники, гарнитуры и микрофоны
- ▶ Ламповые усилители
- ▶ Мидиклавиатуры и др. аудиооборудование
- ▶ Оргтехника
- ▶ Расходные материалы
- ▶ Сетевое оборудование
- ▶ Программное обеспечение
- ▶ Разное
- ▶ Уцененные товары

Скачать прайс-лист  

#### НАШИ МАГАЗИНЫ

**С 10 ДО 20**  
БЕЗ ВЫХОДНЫХ

#### Интернет-магазин ▶

Инструкция покупателя  
2-922-944  
Написать письмо

#### Центр ▶

г. Новосибирск  
Ленина, 12

▶ ТехноСити / Каталог товаров / Игровые приставки, игры и аксессуары / Sony PlayStation / Приставки

### ▶ Игровая приставка Sony PlayStation 3 (40 Gb)

Sony PlayStation 3 (PS3) - продвинутая игровая приставка нового поколения, третья в семействе игровых систем "PlayStation". С помощью PS3 можно играть, смотреть фильмы, слушать музыку, отправлять почту и выходить в Интернет. Корпус PlayStation 3 имеет немного выпуклую форму. Благодаря такой форме PS3 можно устанавливать вертикально без дополнительных подставок. Блок питания встроены в корпус консоли. Привод Blu-ray располагается на правой стороне консоли, диски загружаются не с помощью выезжающего лотка (как у PS2), а через отверстие дисковода. Воспроизведение форматов Blue-Ray, DVD, CD. Частичная совместимость с играми, выпущенными для PSOne и PS2.

Для управления в играх используется новый беспроводной игровой контроллер (джойстик) Sixaxis, который соединен с приставкой посредством связи Bluetooth. (время автономной работы – 24 часа, зарядка от консоли с помощью usb-кабеля). Главной отличительной новинкой Sixaxis является возможность контроллера определять своё положение в пространстве, а также улавливать движение и вращение в трёх плоскостях, что позволит использовать в играх новые приёмы, не требующие нажатия кнопок (при этом функция вибрации в новом джойстике отсутствует). Одновременно к PlayStation 3 можно подключить до семи контроллеров.

На приставке установлена полноценная операционная система, к которой через онлайн-сервис можно скачивать патчи и обновления. В систему включён полноценный веб-браузер, который позволяет просматривать Интернет-страницы и проверять почту.

Графический интерфейс PS3 (Cross Media Bar) разработан на основе интерфейса портативной приставки PSP и является центром управления возможностями консоли. Программа позволяет просматривать фотографии, проигрывать музыку и фильмы с жёсткого диска. Кроме того, операционная система обеспечивает совместимость с USB-устройствами - клавиатурой и мышью.

#### Технические характеристики

- **Центральный процессор Cell Broadband Engine:**  
Ядро на основе архитектуры PowerPC, работающее на частоте 3.2 ГГц  
1 векторный блок на ядро 512Кб кэша второго уровня  
7 вспомогательных ядер, работающих на частоте 3.2 ГГц (одно из восьми ядер отключено)  
7 x 128бит 128 SIMD GPRs 7 x 256Кб SRAM для вспомогательных ядер
- **Производительность:**  
218 ГФлоп (количество операций с плавающей точкой в секунду)
- **Графический процессор nVidia RSX:**  
рабочая частота 550 МГц  
Производительность: 1.8 ТФлоп  
Полная поддержка High-Definition, вплоть до 1080p (1920 x 1080, прогрессивная развертка) на два канала  
Многопоточные программируемые шейдерные конвейеры
- **Звук:**  
Dolby 5.1, DTS, LPCM (Обработка ЦП)
- **Память:**  
256 MB XDR оперативной памяти, частота 3.2ГГц (6.4 как DDR)  
256 MB GDDR3 видеопамяти, частота 700МГц
- **Пропускная способность:**  
Оперативная память - 25.6 Гб/с  
Видеопамять - 22.4 Гб/с RSX - 20 Гб/с (запись) + 15 Гб/с (чтение)

 Напечатать



**17 599 руб.**

Купить сейчас 

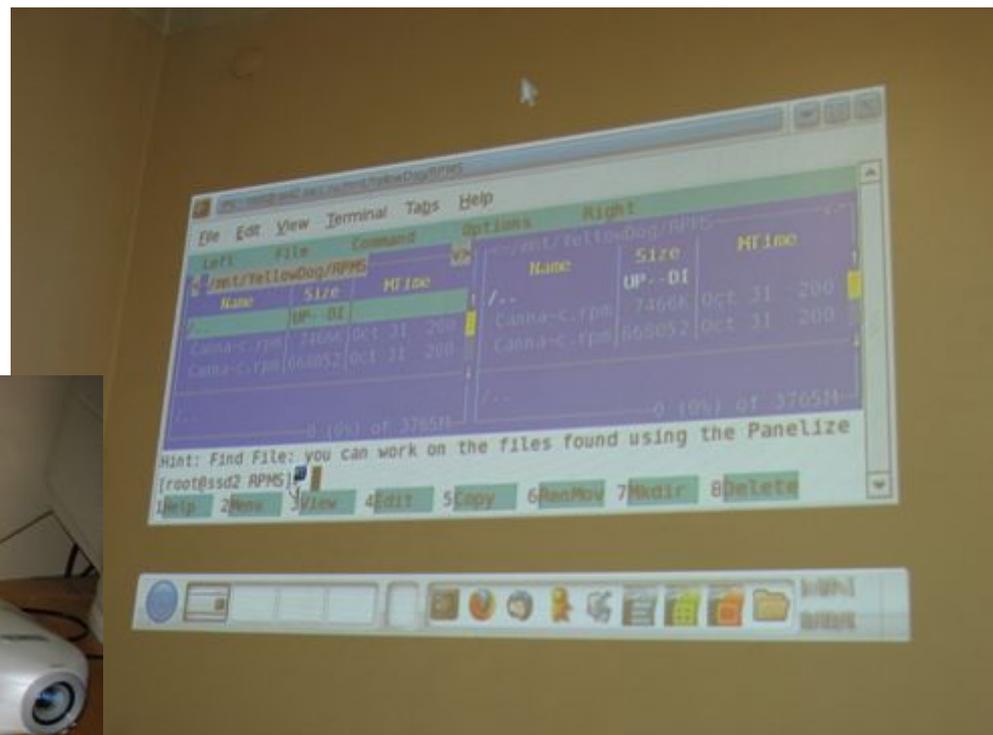
Рассчитать кредит

Добавить к сравнению

Есть на Лаврентьева, 4

 Напечатать

# Sony PlayStation3 в ИВМиМГ в отделе МО ВВС



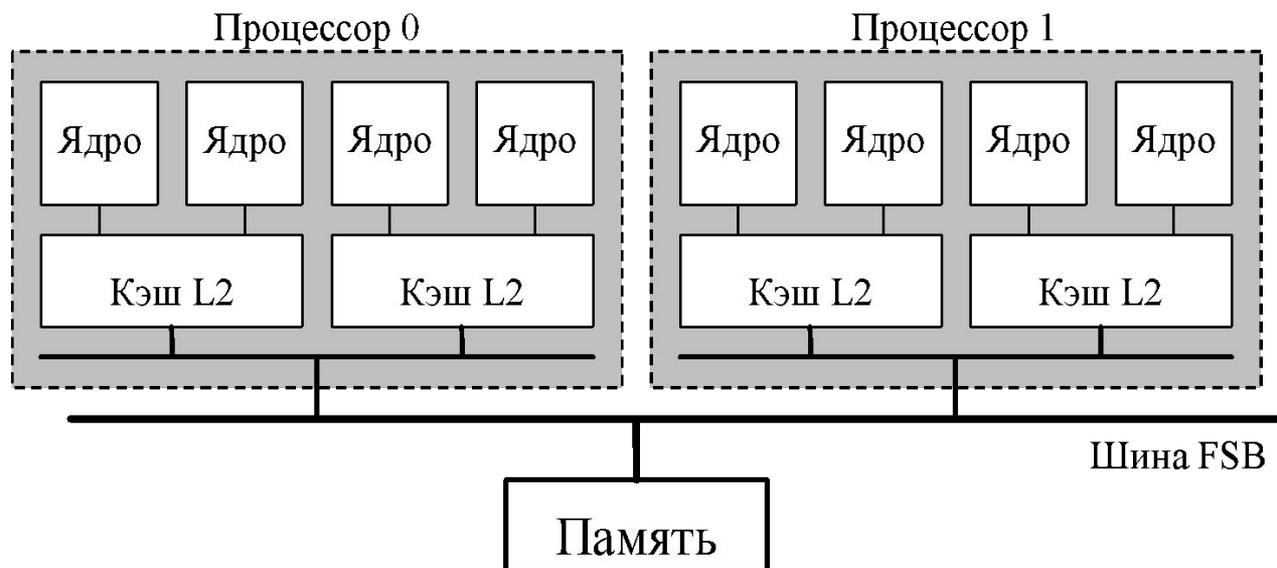
- 1 процессор CELL B.E. 3.2 GHz**
  - 1 ядро PPE
  - 6 ядер SPE
- 256 МВ оперативной памяти**



# План

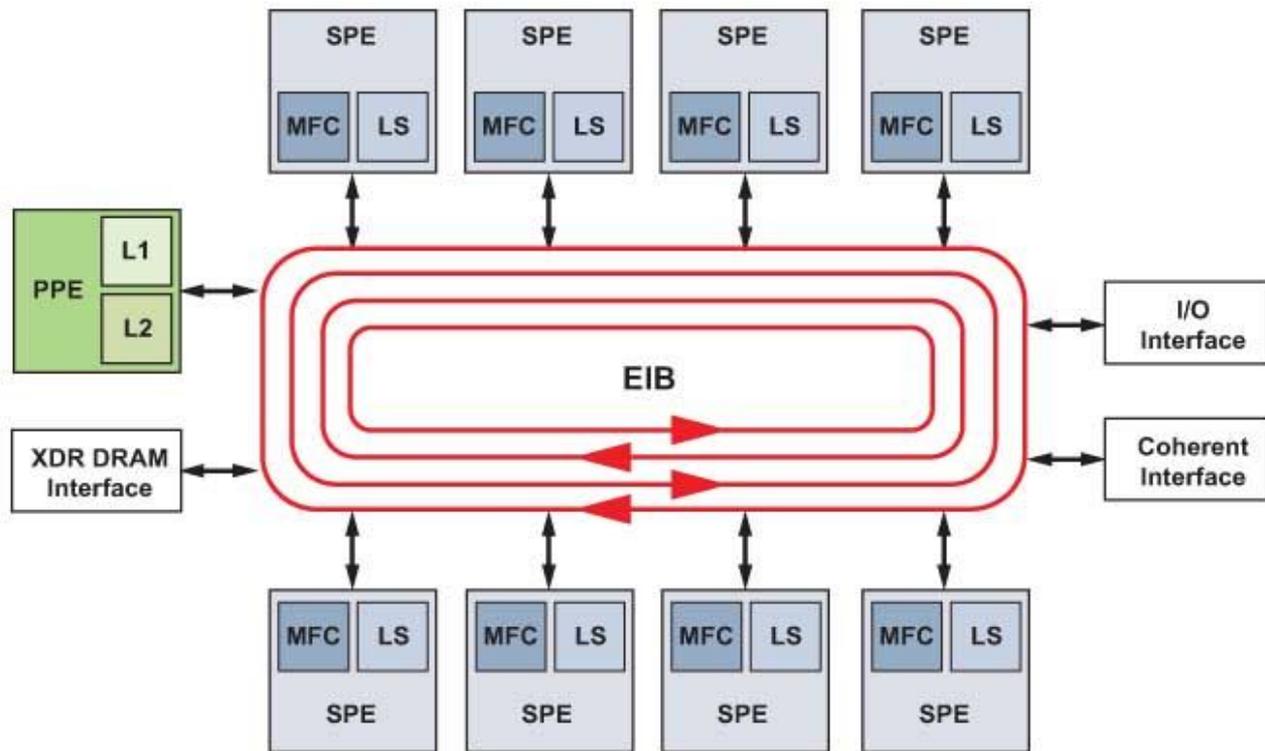
- ▣ **Архитектура процессора Cell**
- ▣ Принципы программирования
- ▣ Базовые средства программирования
  - Создание и компиляция простых программ
  - Механизмы передачи данных и сообщений
- ▣ Программирование вычислений на SPE
- ▣ Библиотеки

# Структура обычного многоядерного узла



- Все ядра в узле равноправные

# Структура процессора Cell



## Многоядерный процессор Cell

- Power Processor Element – ядро общего назначения (главное)
- Synergistic Processor Element – векторные ядра (дополнительные)
- Element Interconnect Bus – быстрая шина

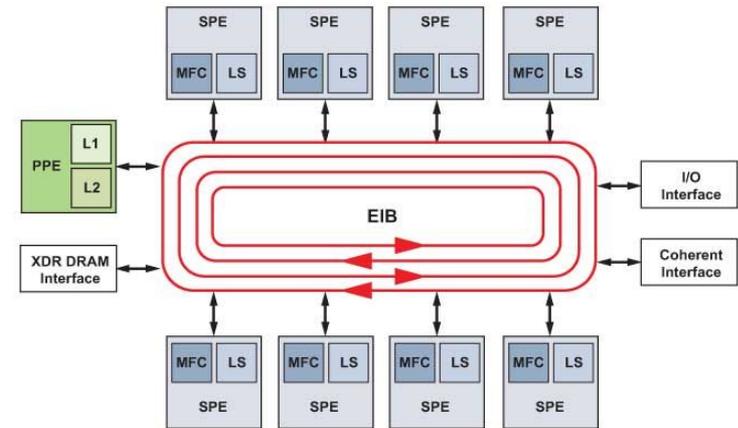
# Структура процессора Cell

## □ 1 ядро PPE

- ядро общего назначения (PowerPC, 2 потока)
- выполняет код операционной системы
- управляет ходом вычислений

## □ 8 ядер SPE

- специализированные векторные ядра (особый набор команд)
- обеспечивают вычислительную мощность процессора
- работают с ограниченной локальной памятью: 256 KB
- имеют много регистров: 128

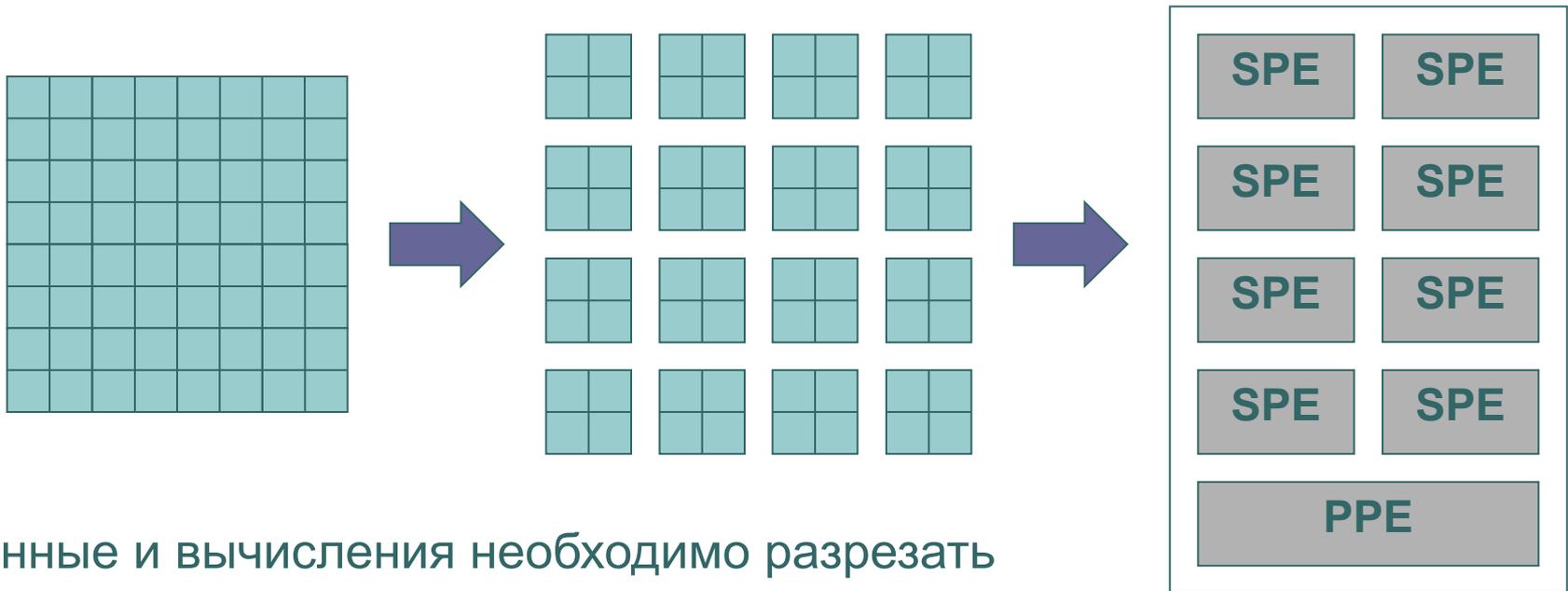


## □ Шина EIB

- асинхронная
- имеет высокую пропускную способность
- обеспечивает большое число одновременных запросов

# Виды параллелизма в процессоре Cell

1. **Многоядерность.** Независимые задачи, которые могут быть выполнены одновременно на разных ядрах:
  - 2 аппаратных потока PPE
  - 8 программ SPE

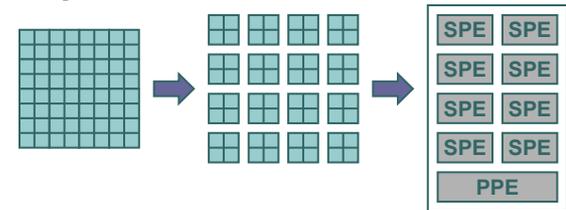


Данные и вычисления необходимо разрезать на фрагменты и распределять между ядрами

# Виды параллелизма в процессоре Cell

1. **Многоядерность.** Независимые задачи, которые могут быть выполнены одновременно на разных ядрах:

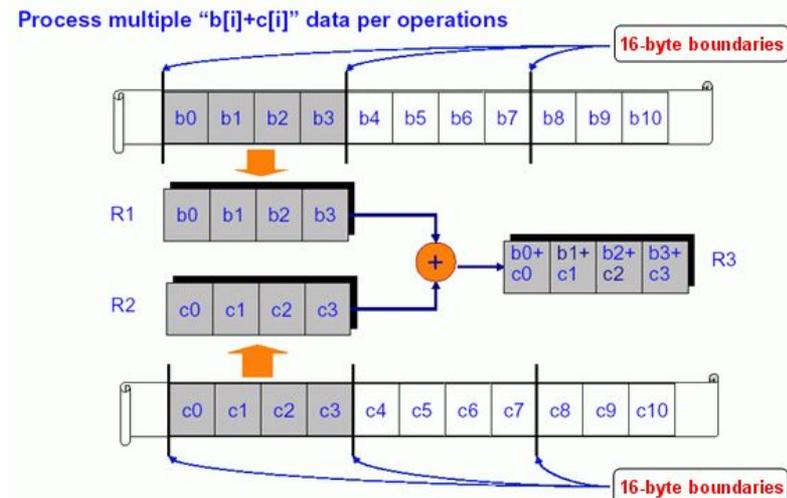
- 2 аппаратных потока PPE
- 8 программ SPE



2. **Векторизация.** Регулярные векторные данные, обработка которых может быть векторизована:

- SPE SIMD
- PPE VMX

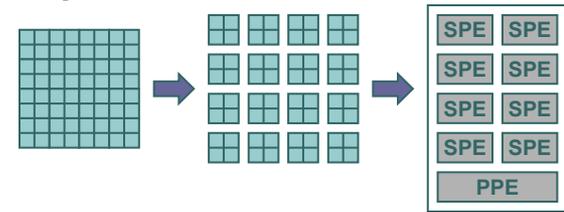
Данные необходимо группировать в вектора и следить за выравниванием



# Виды параллелизма в процессоре Cell

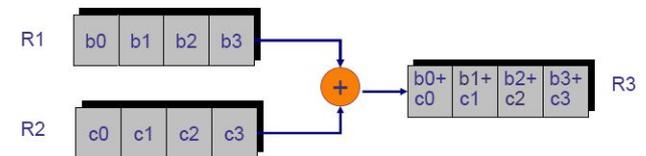
1. **Многоядерность.** Независимые задачи, которые могут быть выполнены одновременно на разных ядрах:

- 2 аппаратных потока PPE
- 8 программ SPE



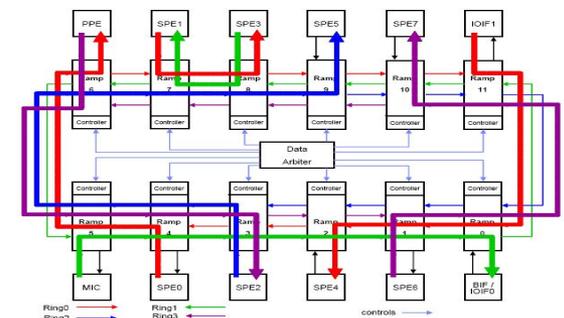
2. **Векторизация.** Регулярные векторные данные, обработка которых может быть векторизована:

- SPE SIMD
- PPE VMX

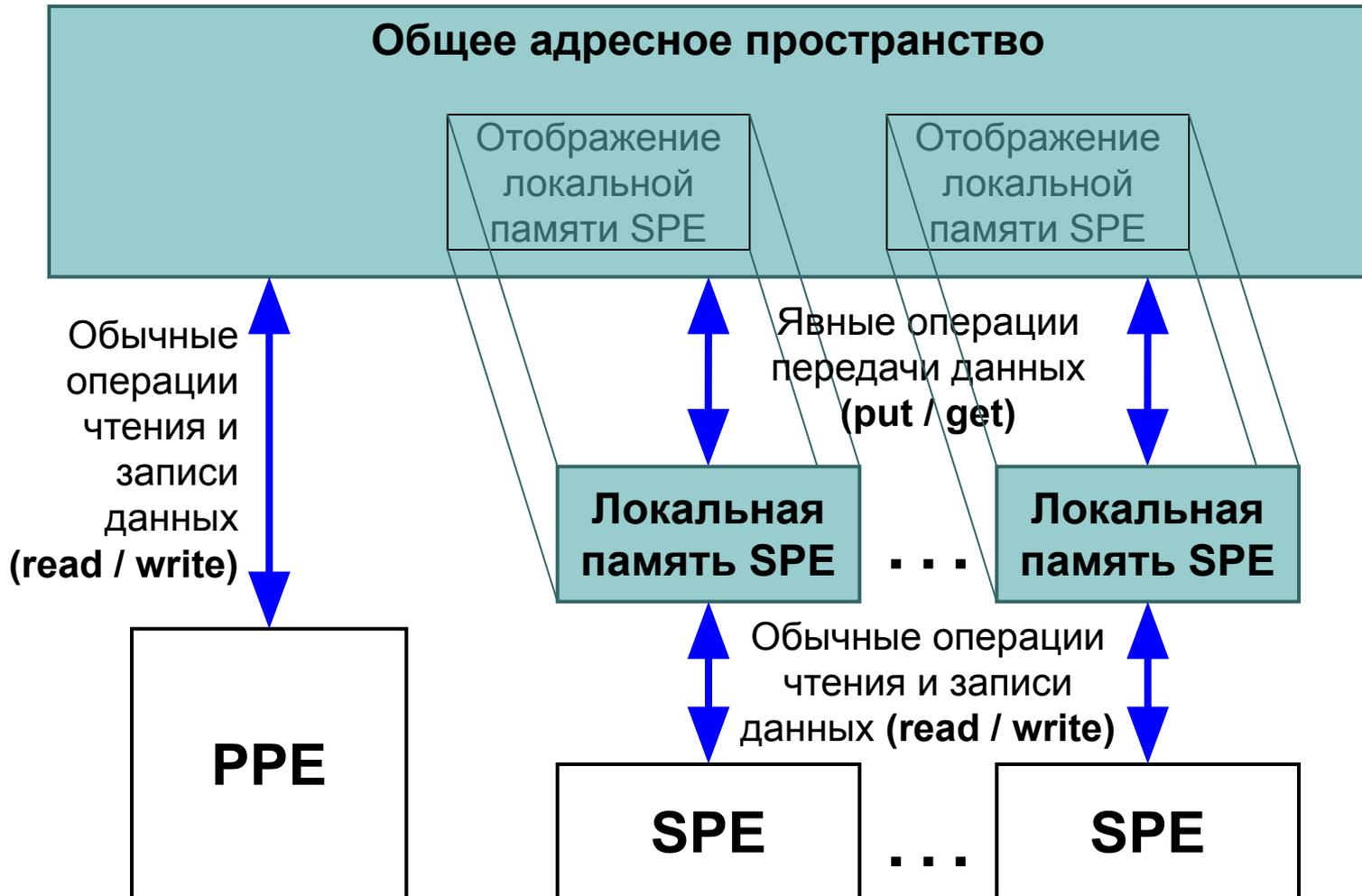


3. **Асинхронная передача данных**

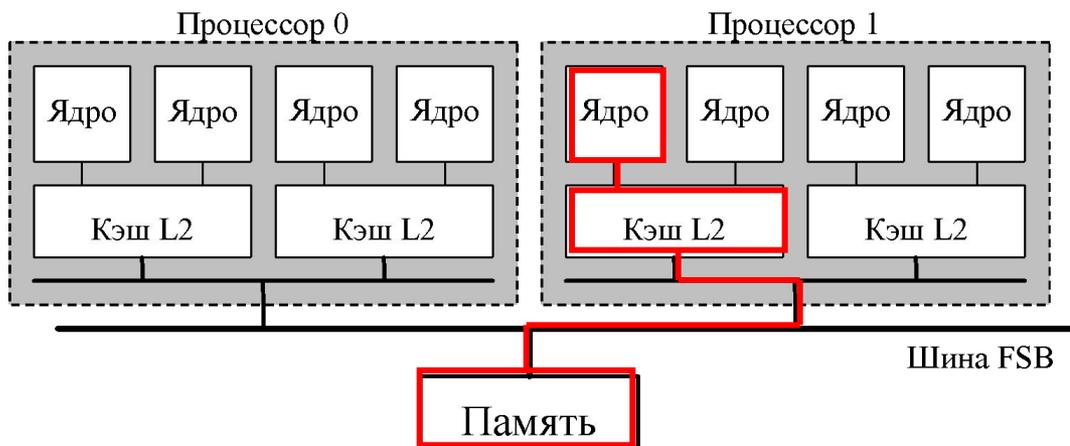
Необходимо использовать специальные алгоритмы для совмещения передач данных и вычислений



# Модель памяти

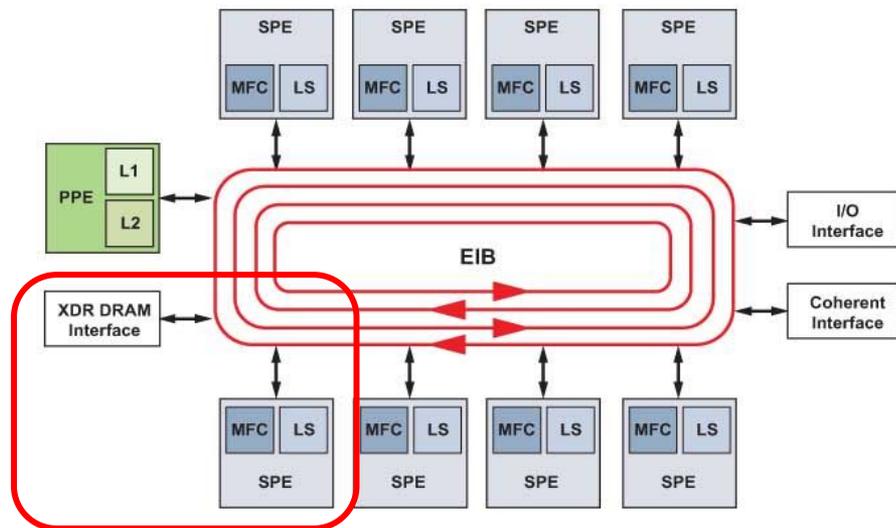


# Сравнение подсистем памяти



Обычная иерархия памяти

Иерархия памяти ядра SPE

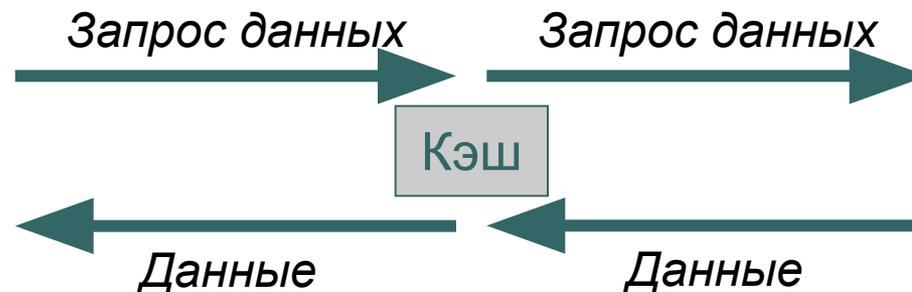
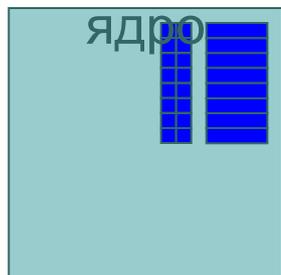


# Сравнение подсистем памяти

## ▣ Обычная иерархия памяти

- Ядро может обращаться к данным в оперативной памяти
- После запроса данные из памяти придут неизвестно когда
- Гарантировать приход данных вовремя можно с помощью:
  - Правильного порядка обращений к данным
  - Предвыборки

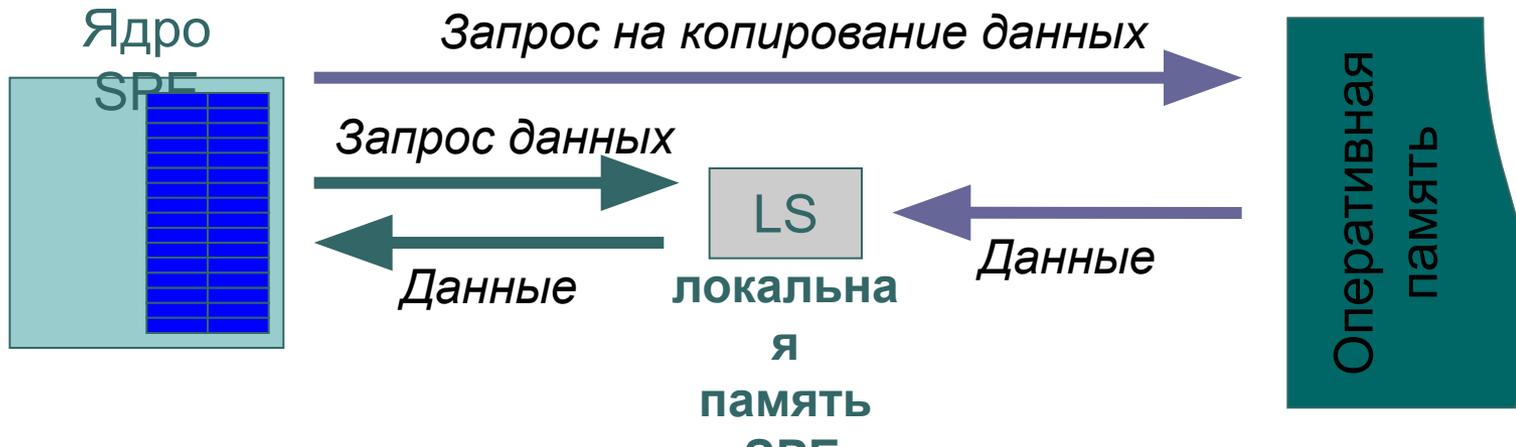
Обычное  
процессорное



# Сравнение подсистем памяти

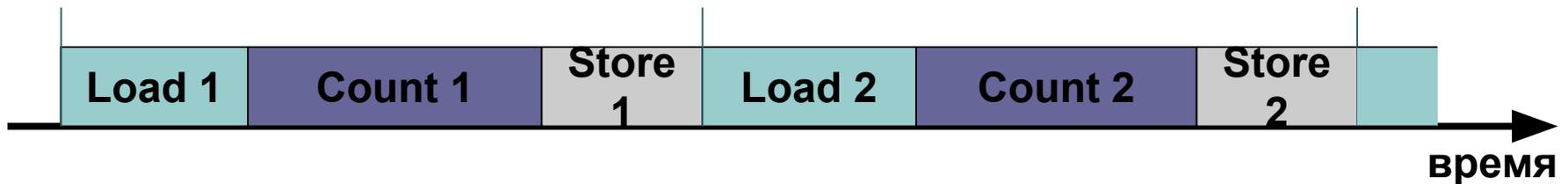
## Иерархия памяти ядра SPE

- Ядро SPE может обращаться к данным в локальной памяти
- Ядро может отправлять запрос на перемещение данных из оперативной в локальную память (или обратно) и проверять его завершение
- Данные из локальной памяти доступны за константное время

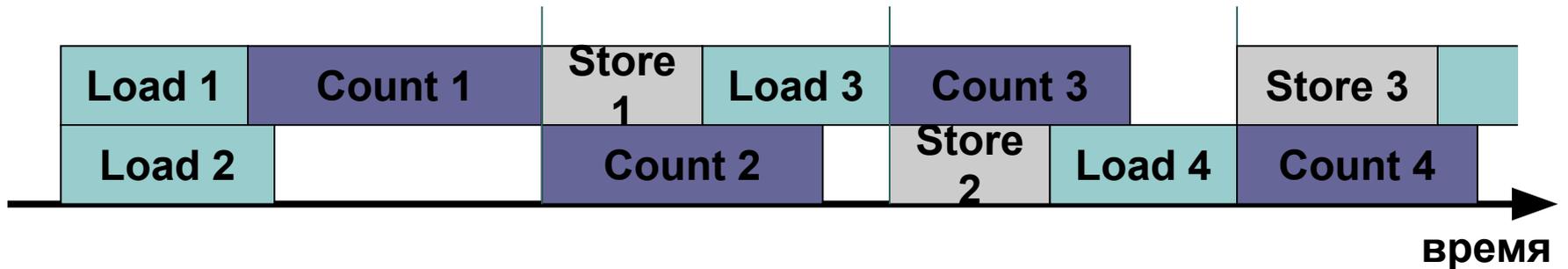


# Выполнение обменов на фоне вычислений

Вычисления с одним буфером на SPE



Вычисления с двумя буферами на SPE



**Load** – копирование данных в память SPE

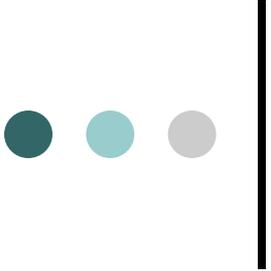
**Count** – выполнение вычислений

**Store** – копирование результата из памяти SPE



# План

- Архитектура процессора Cell
- **Принципы программирования**
- Базовые средства программирования
  - Создание и компиляция простых программ
  - Механизмы передачи данных и сообщений
- Программирование вычислений на SPE
- Библиотеки



# Программирование Cell

**Для организации вычислений на Cell необходимо:**

- Разделить задачу на независимые фрагменты для обработки на ядрах SPE
- Организовать передачу данных между ядрами на одновременно с вычислениями
- Выполнить векторизацию вычислений на ядрах SPE



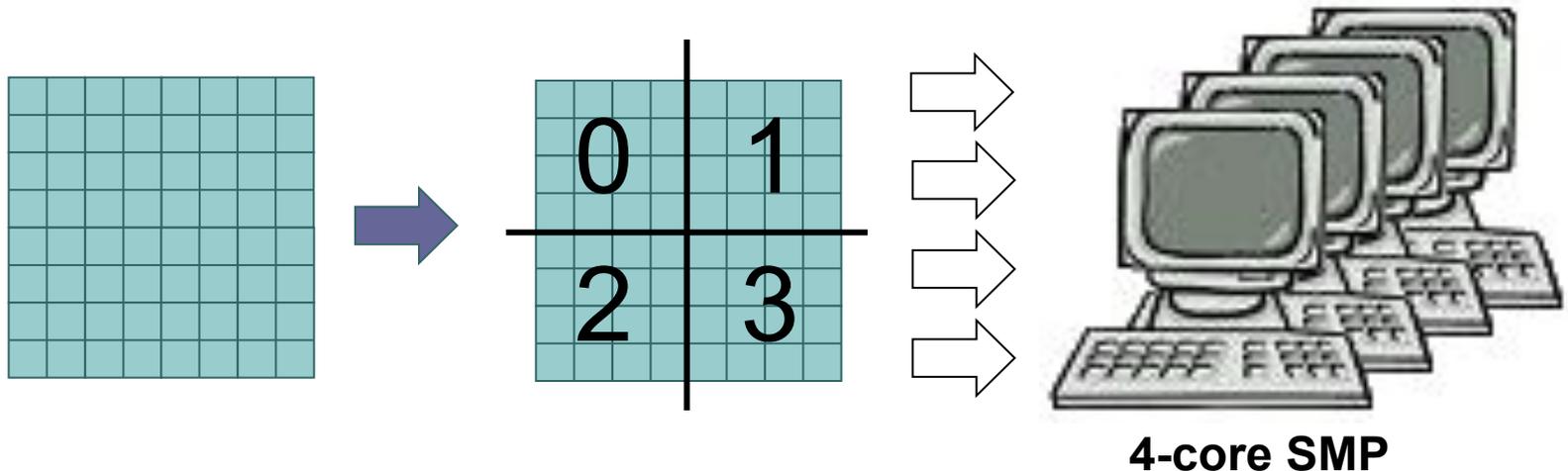
# Сравнение параллельных вычислений на Cell с «традиционными» подходами

- ▣ **SMP-узел + потоки (OpenMP, PThreads, ...)**
- ▣ **Кластер + библиотека MPI**
- ▣ **Узел на базе процессоров Cell**

# Сравнение параллельных вычислений на Cell с «традиционными» подходами

## □ SMP-узел + потоки (OpenMP, PThreads, ...)

- На одном узле работают несколько потоков исполнения
- Данные одинаково доступны всем потокам
- Каждый поток выполняет свою часть работы над общими данными

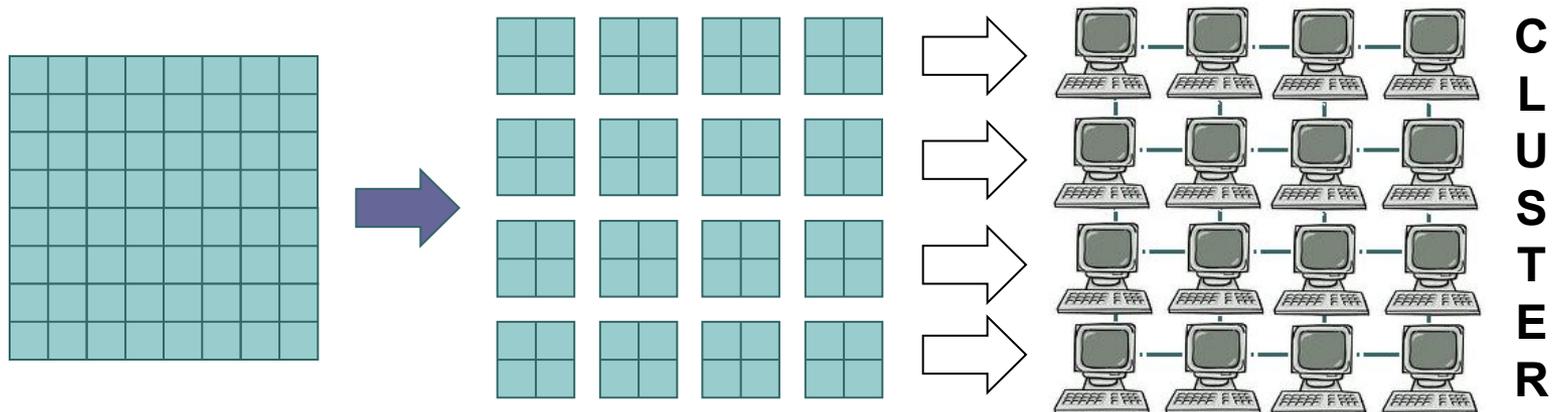


Данные обрабатываются в том месте, где они хранятся

# Сравнение параллельных вычислений на Cell с «традиционными» подходами

## □ Кластер + библиотека MPI

- На каждом узле работает один или несколько процессов
- Данные распределены по процессам
- Каждый процесс выполняет свою часть работы над своими данными
- Процессы обмениваются сообщениями

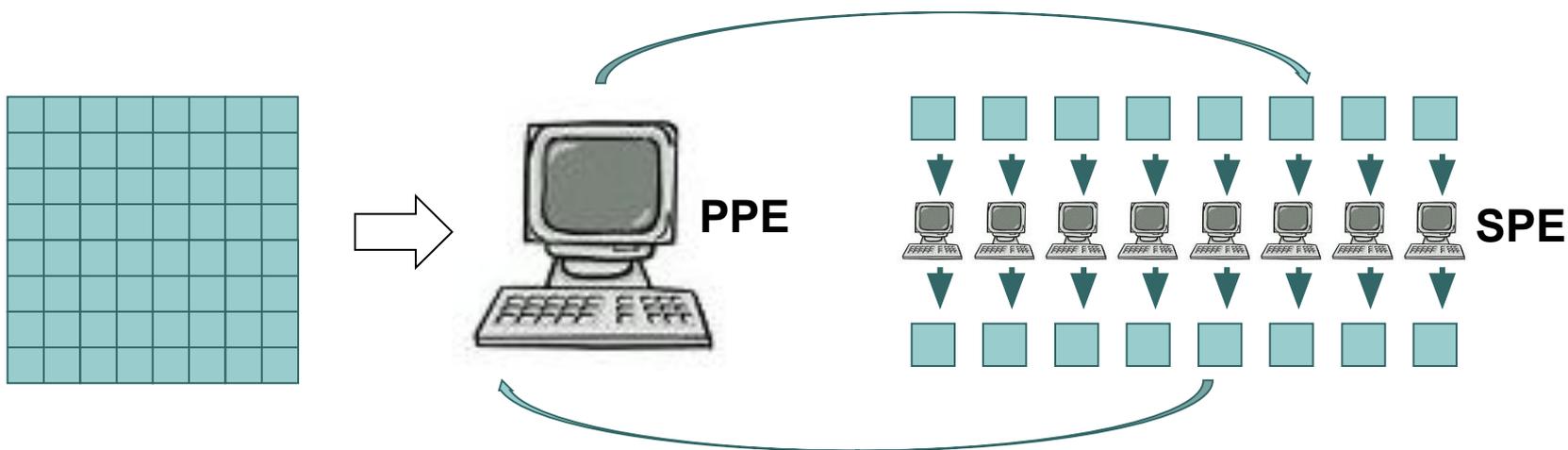


Данные обрабатываются в том месте, где они хранятся

# Сравнение параллельных вычислений на Cell с «традиционными» подходами

## □ Узел на базе процессоров Cell

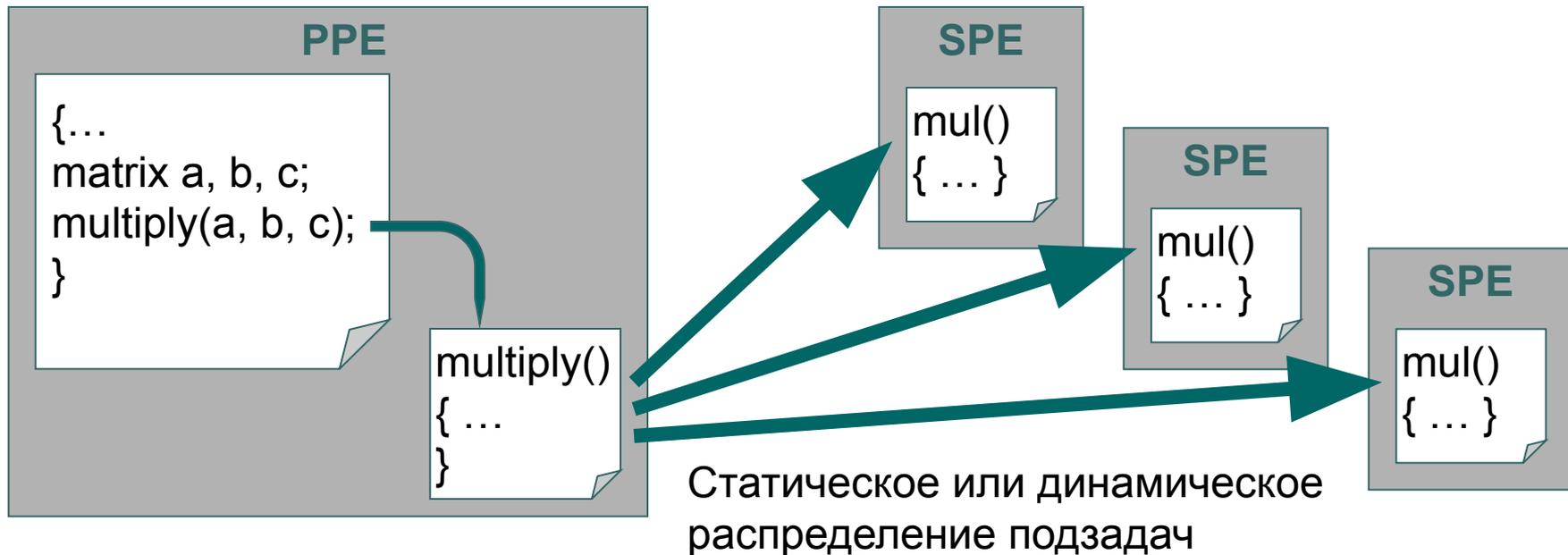
- На ядре PPE работает главный поток
- На ядрах SPE работают вычислительные потоки
  - копируют блоки данных из общей памяти в локальную память ядра
  - производят вычисления над данными в локальной памяти
  - копируют результаты в общую память



Для обработки данные необходимо скопировать в локальную память ядер SPE

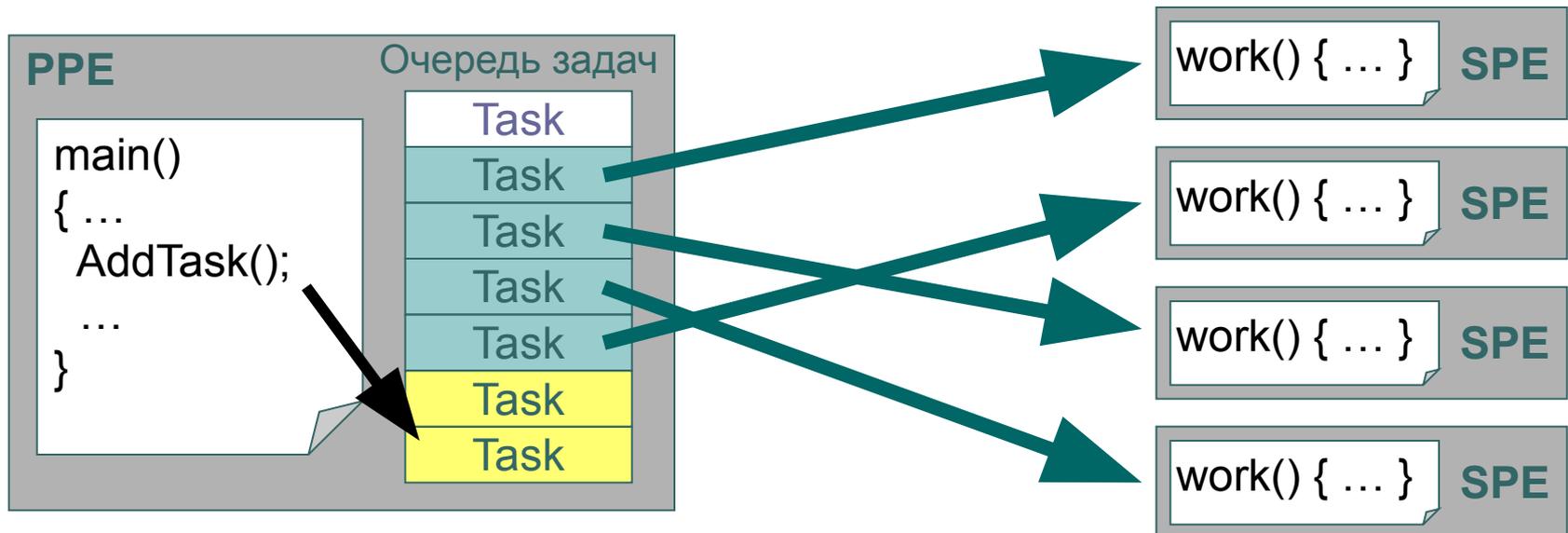
# Модели программирования процессора Cell

## □ Function offload model



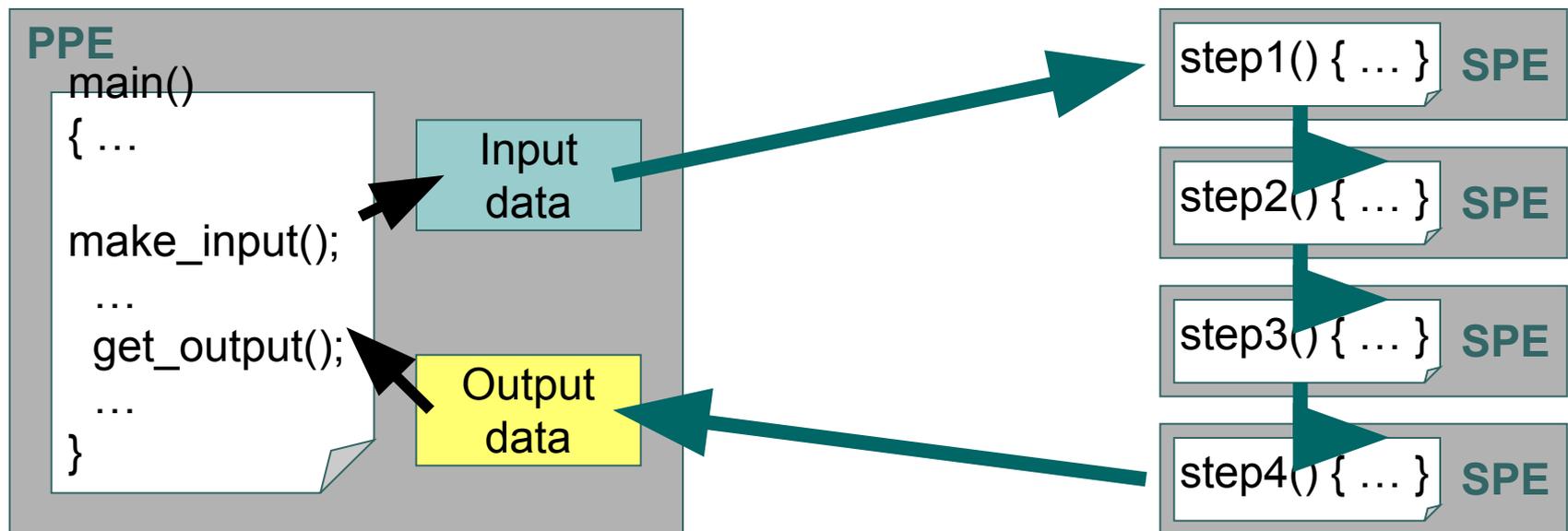
# Модели программирования процессора Cell

## □ Портфель задач



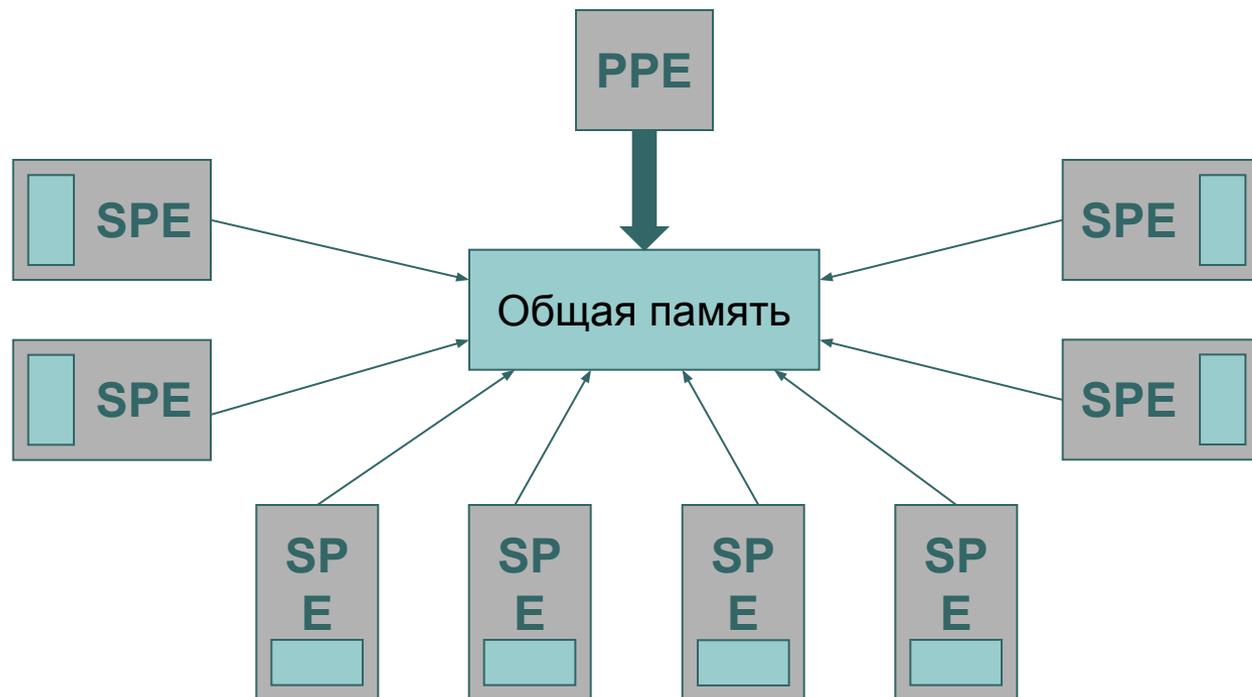
# Модели программирования процессора Cell

## □ Поточковый конвейер



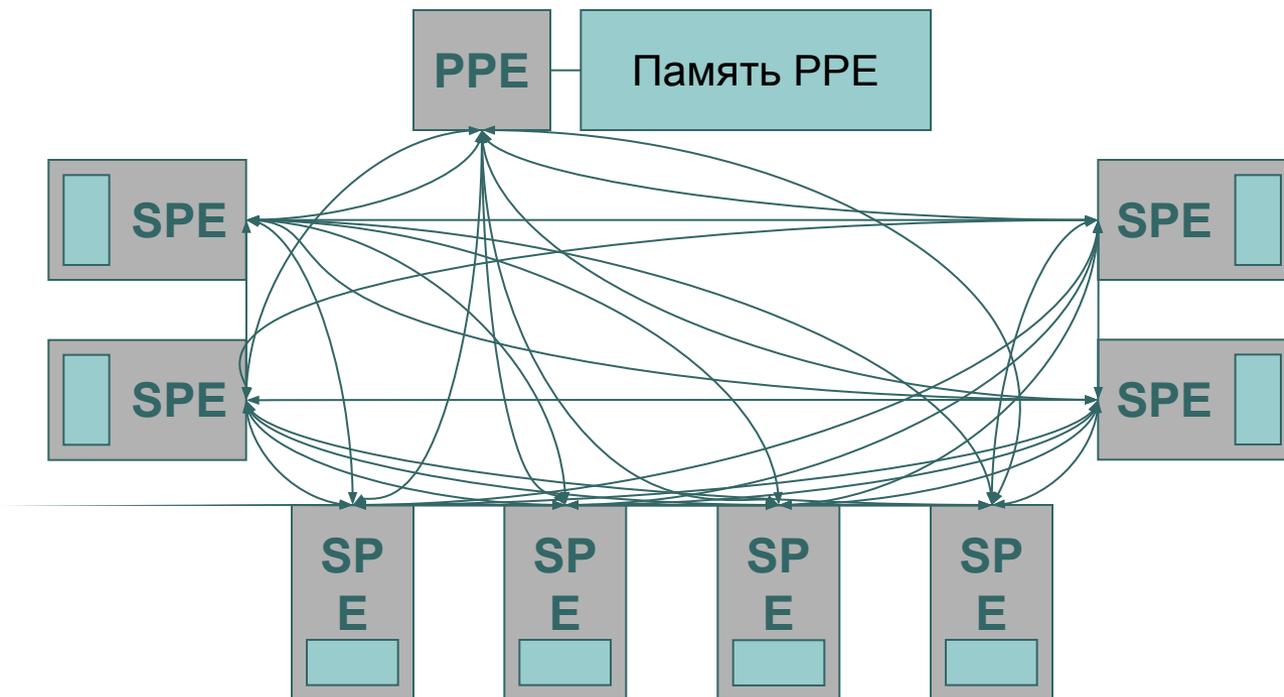
# Модели программирования процессора Cell

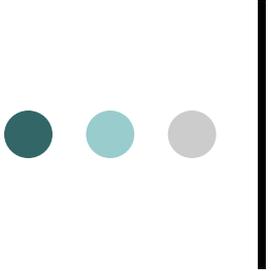
- Взаимодействующие процессы с общей памятью



# Модели программирования процессора Cell

- Взаимодействующие процессы с распределенной памятью



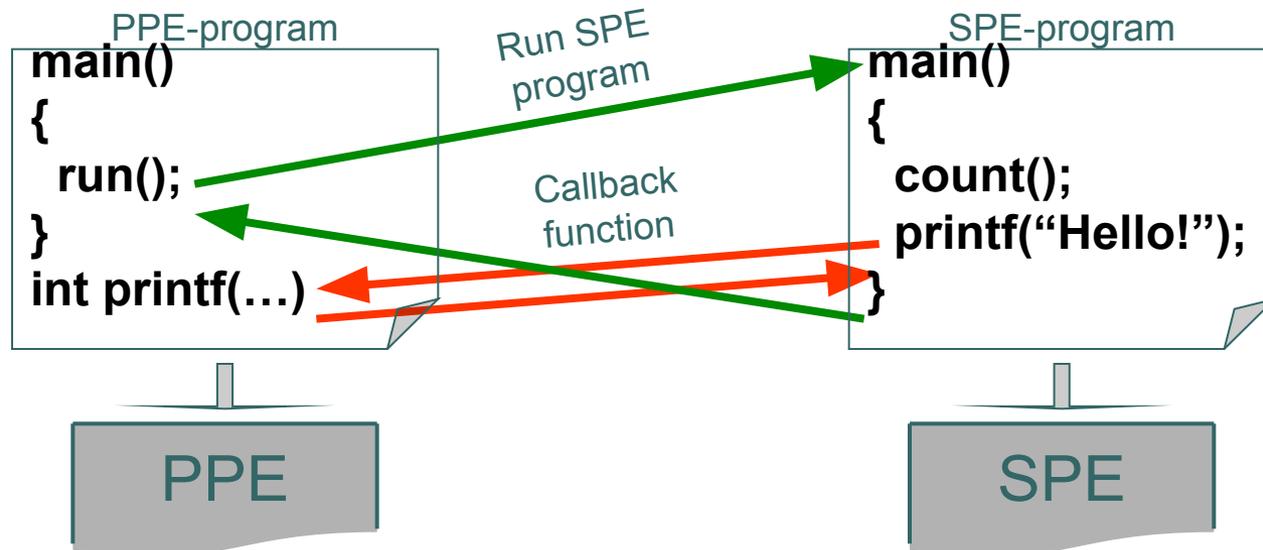


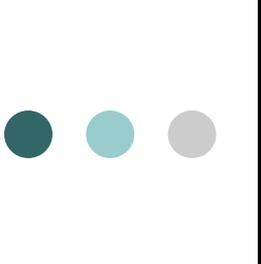
# План

- Архитектура процессора Cell
- Принципы программирования
- **Базовые средства программирования**
  - **Создание и компиляция простых программ**
  - Механизмы передачи данных и сообщений
- Программирование вычислений на SPE
- Библиотеки

# Библиотека libspe2

- LibSPE предоставляет интерфейс к **базовым** средствам программирования процессора Cell, реализованным аппаратно.
- Разработка программы:
  - Создается отдельная программа для PPE,
  - Создается отдельная программа для SPE,
  - PPE-программа запускает SPE-программу,
  - SPE-программа может вызвать callback-функцию вызвавшей ее PPE-программы.





# Библиотека libspe2: Программа «Hello, World!»

## □ Программа для PPE (ppu\_prog.c)

```
#include <libspe2.h>
extern spe_program_handle_t spu_hello;
int main ()
{
    unsigned int entry = SPE_DEFAULT_ENTRY;
    spe_context_ptr_t spe;

    spe = spe_context_create (0, NULL);
    spe_program_load (spe, &spu_hello);
    spe_context_run (spe, &entry, 0, (void *) 10, (void *) 20, NULL);
    spe_context_destroy (spe);

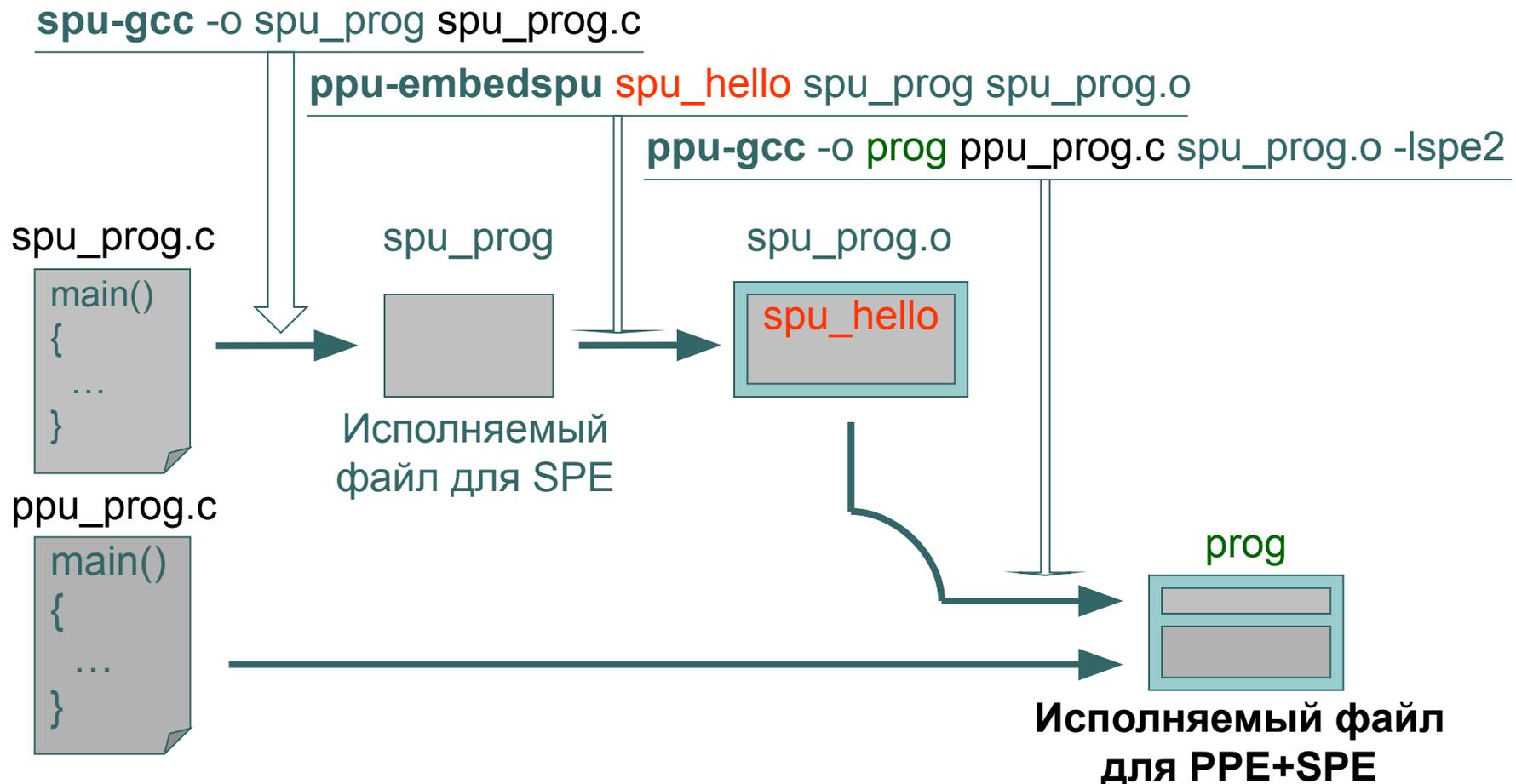
    return 0;
}
```

## □ Программа для SPE (spu\_prog.c)

```
#include <stdio.h>
int main (unsigned long long spe, unsigned long long argp, unsigned long long envp)
{
    printf("Hello, World! (%llu,%llu)\n", argp, envp);
    return 0;
}
```

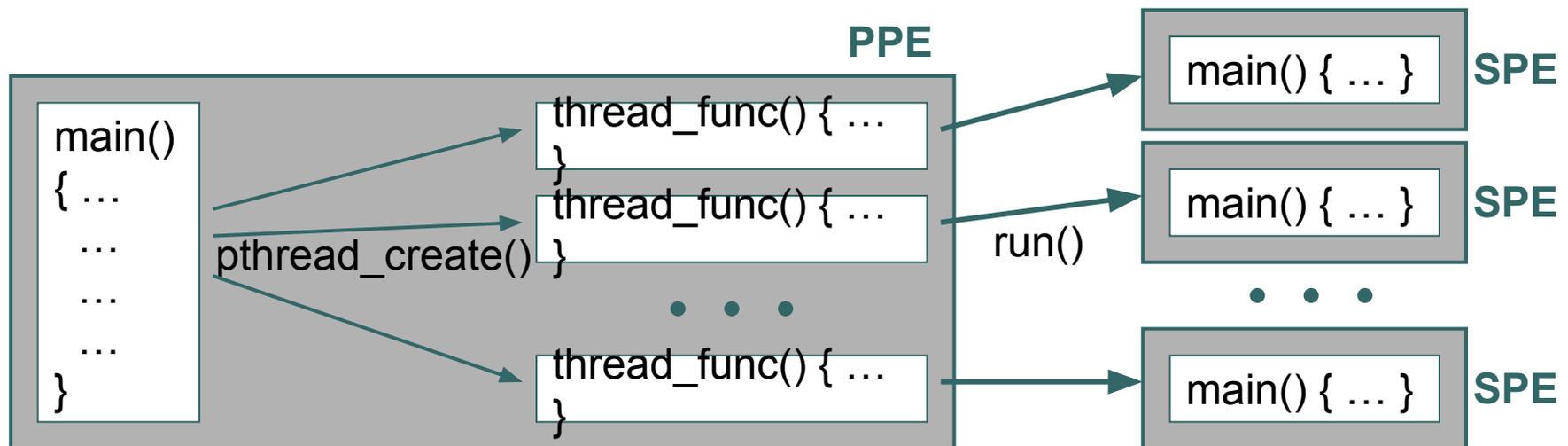
# Библиотека libspe2: Программа «Hello, World!»

## □ Компиляция программы для Cell:



# Библиотека libspe2: Многопоточная программа «Hello, World!»

- Создание параллельной программы для Cell:
  - В программе на PPE создать несколько параллельных потоков,
  - В потоках запустить программы на SPE.





# Библиотека libspe2:

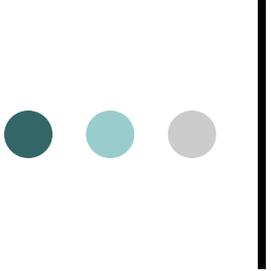
## Многопоточная программа «Hello, World!»

▣ Программа для PPE многопоточная

```
#include <libspe2.h>
#include <pthread.h>
#define NTHREADS 40
extern spe_program_handle_t spu_hello;

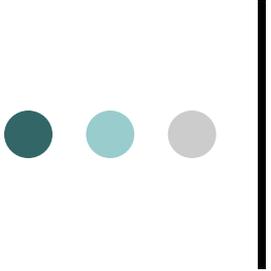
void *thread_func (void *data)
{
    unsigned int entry = SPE_DEFAULT_ENTRY;
    spe_context_ptr_t spe;
    spe = spe_context_create (0,NULL);
    spe_program_load (spe, &spu_hello);
    spe_context_run (spe, &entry, 0, (void *)data, (void *)NTHREADS, NULL);
    spe_context_destroy (spe);
    return 0;
}

int main ()
{ pthread_t tid [NTHREADS];
  unsigned long i;
  for (i=0;i<NTHREADS;i++) pthread_create (&tid[i], NULL, &thread_func, (void *) i);
  // Параллельная часть
  for (i=0;i<NTHREADS;i++) pthread_join (tid[i], NULL);
  return 0;
}
```



# План

- Архитектура процессора Cell
- Принципы программирования
- Базовые средства программирования
  - Создание и компиляция простых программ
  - **Механизмы передачи данных и сообщений**
- Программирование вычислений на SPE
- Библиотеки



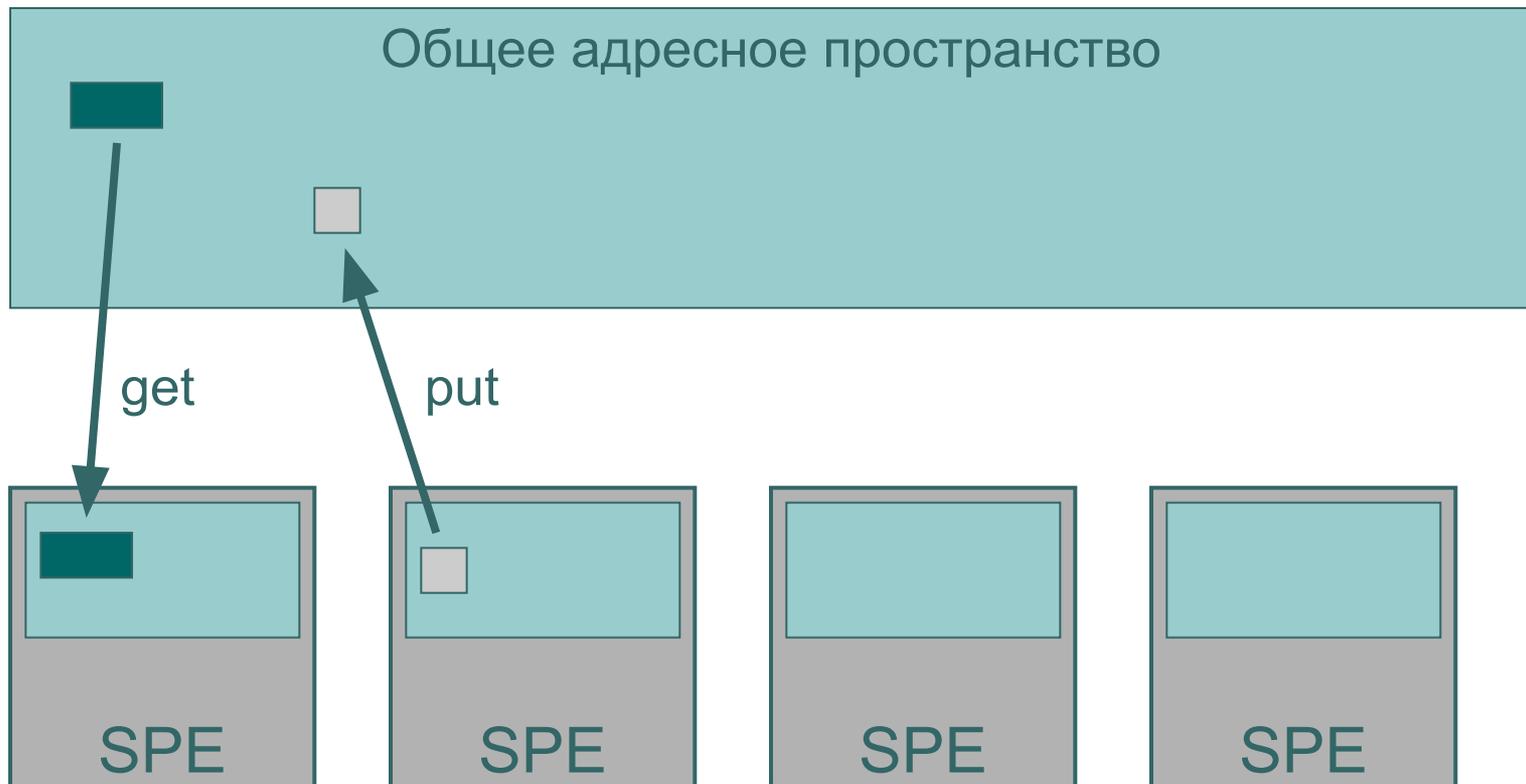
# Механизмы передачи данных и сообщений

- DMA-передача – блок данных до 16 KB:
  - Операция Get: общая память □ память SPE
  - Операция Put: память SPE □ общая память
- Mailbox-ы – очереди 32-битных сообщений:
  - SPE in (4)
  - SPE out (1)
  - SPE out interrupt (1)
- Сигналы – 32-битные сообщения:
  - Только к SPE

# Механизмы передачи данных и сообщений

## □ DMA-передача данных

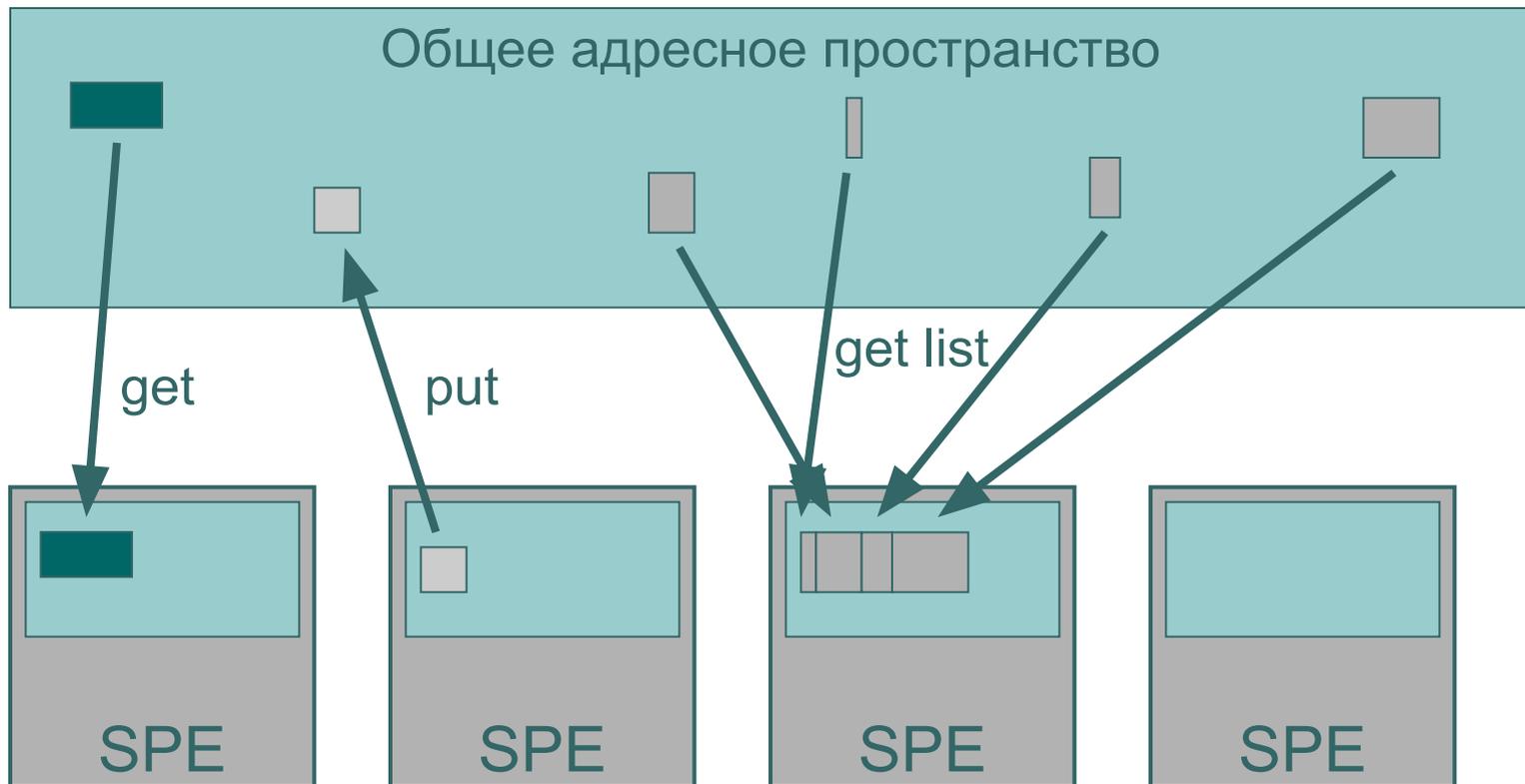
- Может запускаться и на PPE, и на SPE



# Механизмы передачи данных и сообщений

## □ DMA-передача данных

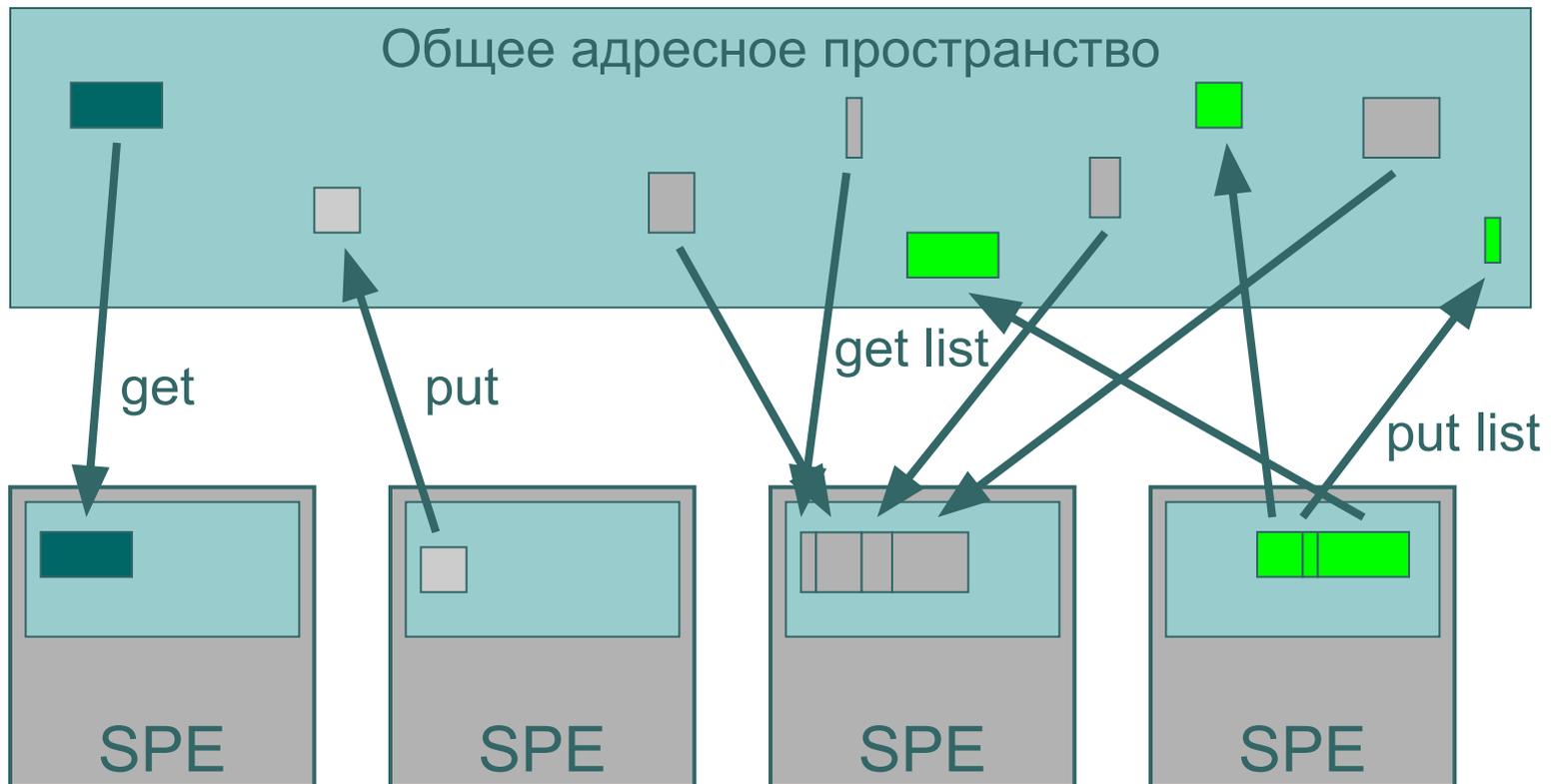
- Может запускаться и на PPE, и на SPE

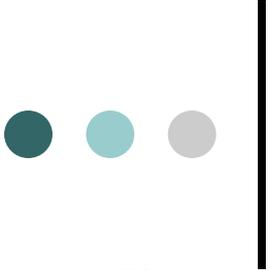


# Механизмы передачи данных и сообщений

## □ DMA-передача данных

- Может запускаться и на PPE, и на SPE





# Механизмы передачи данных и сообщений

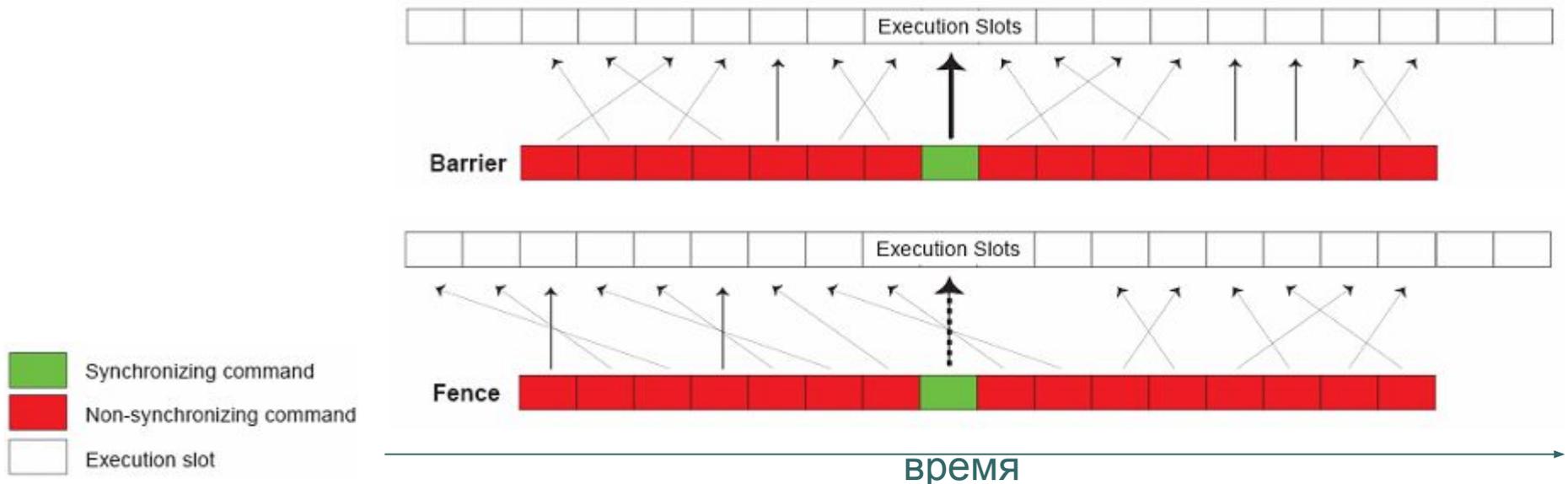
## □ DMA-передача данных

- Блоки данных должны быть выровнены в памяти по границе 16 байт.
- Размер передаваемых данных может быть 1,2,4,8,16 или  $16 \cdot N$  байтов.
- Для идентификации группы DMA-передач используются 5-битные DMA-тэги.
- Тэг позволяет проверить статус или дождаться завершения соответствующей группы DMA-передач.

# Механизмы передачи данных и сообщений

## □ DMA-передача данных

- Чтобы упорядочить выполнение передач, используются специальные варианты команд get и put:
  - Barrier: getb, putb
  - Fence: getf, putf





# Библиотека libspe 2.0:

## Пример передачи данных DMA

### ▣ Программа для SPE

**// GET: Передача данных из PPE в SPE**

```
void get (void *dest_lsa, unsigned long long sour_ea, unsigned long size)
```

```
{
```

```
    int tag=mfc_tag_reserve(), mask=1<<tag;
```

```
    mfc_get (dest_lsa, sour_ea, size, tag, 0, 0);    // запуск передачи данных
```

```
    mfc_write_tag_mask (mask);
```

```
    mfc_read_tag_status_any();                    // ожидание завершения передачи
```

```
    mfc_tag_release(tag);
```

```
}
```

**// PUT: Передача данных из SPE в PPE**

```
void put (void *sour_lsa, unsigned long long dest_ea, unsigned long size)
```

```
{
```

```
    int tag=mfc_tag_reserve(), mask=1<<tag;
```

```
    mfc_put (sour_lsa, dest_ea, size, tag, 0, 0);    // запуск передачи данных
```

```
    mfc_write_tag_mask (mask);
```

```
    mfc_read_tag_status_any();                    // ожидание завершения передачи
```

```
    mfc_tag_release();
```

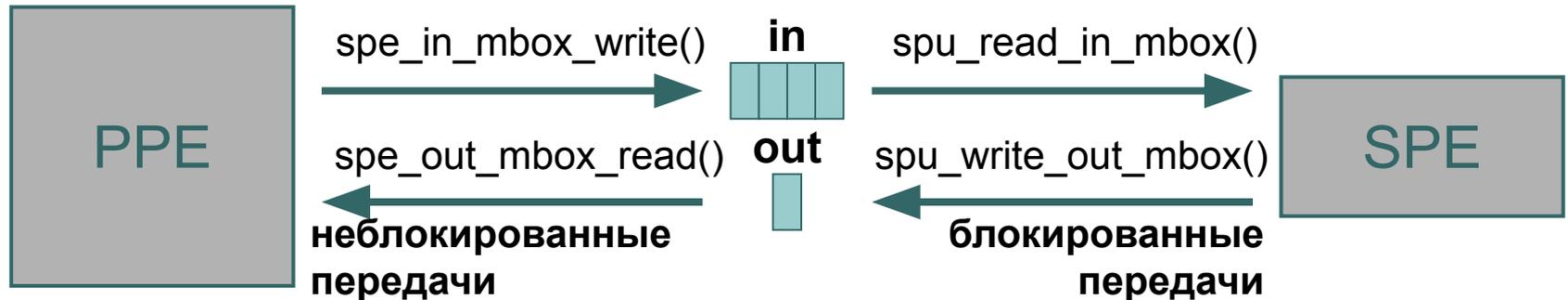
```
}
```

# Механизмы передачи данных и сообщений

- Mailbox: передача 4-байтовых сообщений

spe\_out\_mbox\_status()  
spe\_in\_mbox\_status()

spu\_stat\_in\_mbox()  
spu\_stat\_out\_mbox()

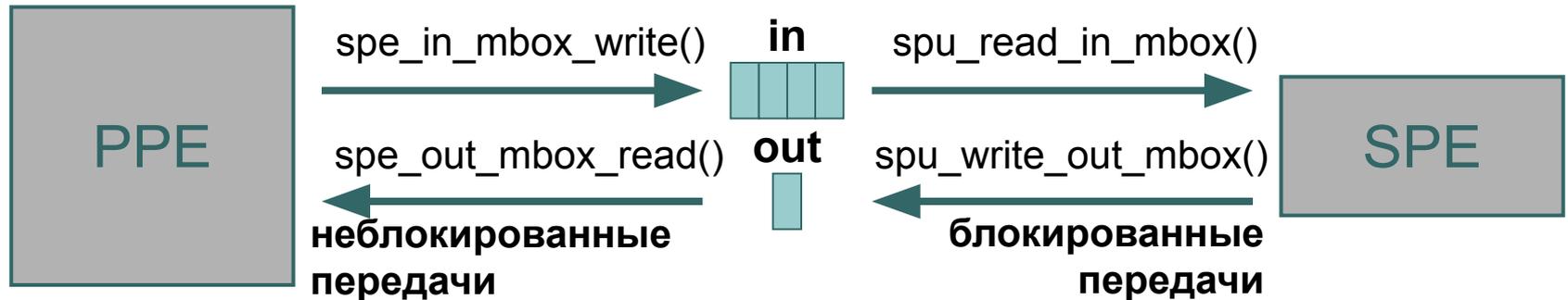


# Механизмы передачи данных и сообщений

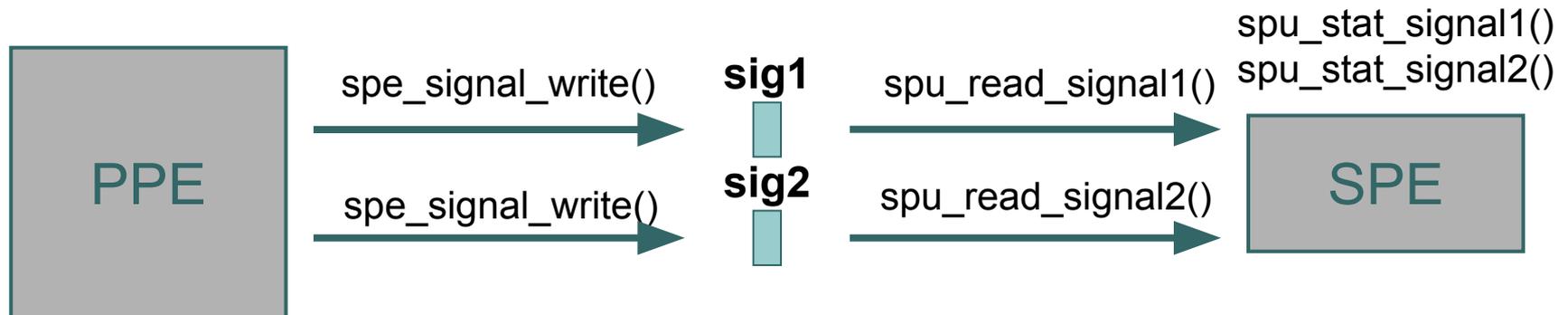
- Mailbox: передача 4-байтовых сообщений

spe\_out\_mbox\_status()  
spe\_in\_mbox\_status()

spu\_stat\_in\_mbox()  
spu\_stat\_out\_mbox()



- Signal: передача 4-байтовых сообщений



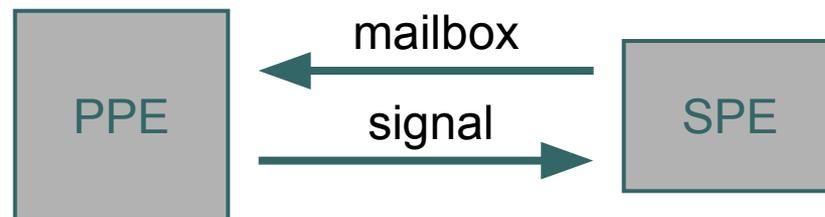
# Библиотека libspe 2.0: Программа «Ping-pong»

## □ Фрагмент программы для PPE

```
...  
while ( spe_out_mbox_status(spe) == 0 ); // ожидание данных в очереди  
spe_out_mbox_read(spe ,&data ,1); // чтение одного элемента из очереди  
data++; // изменение данных  
spe_signal_write(spe, SPE_SIG_NOTIFY_REG_1, data); // запись в регистр сигнала 1  
...
```

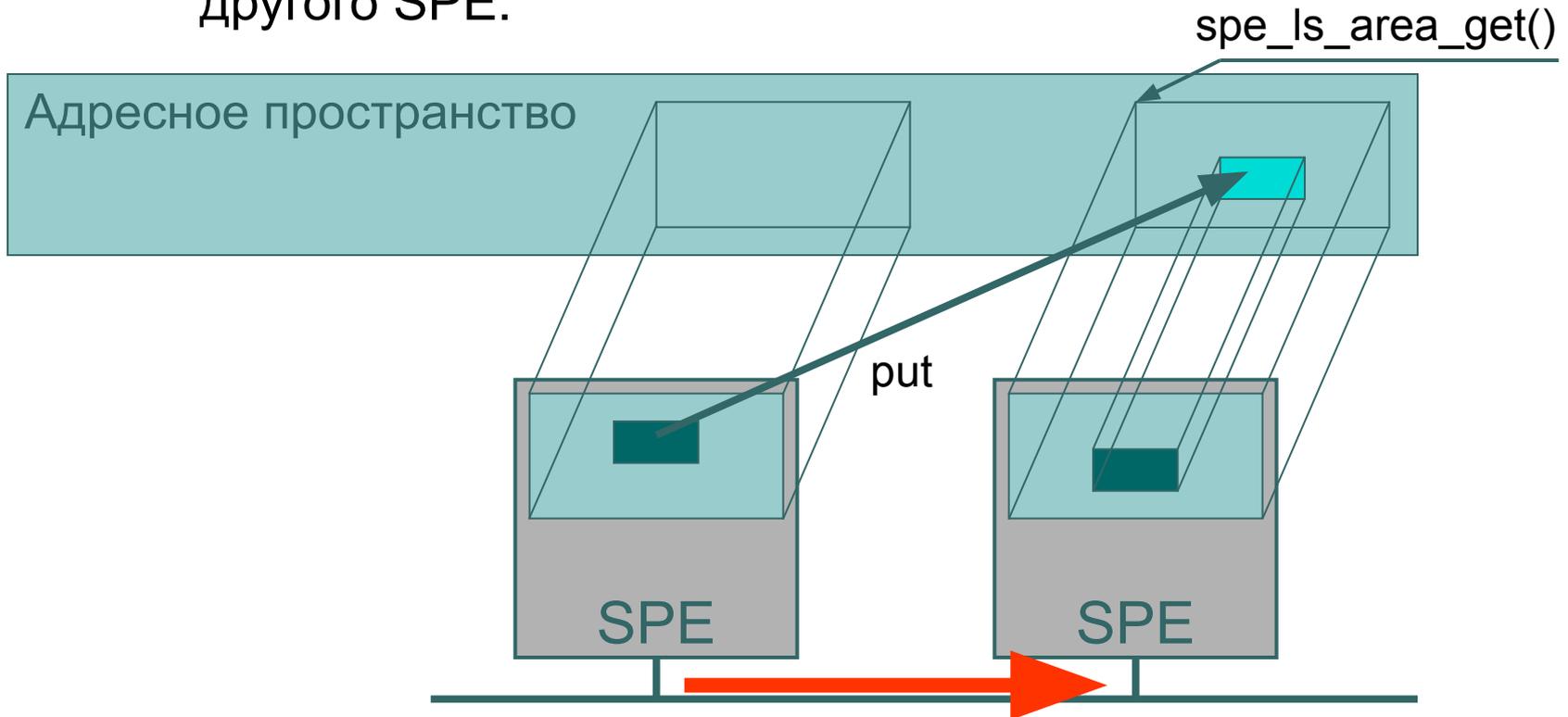
## □ Фрагмент программы для SPE

```
...  
spu_write_out_mbox(data); // запись элемента данных в очередь  
data=spu_read_signal1(); // чтение данных из регистра сигнала 1  
...
```



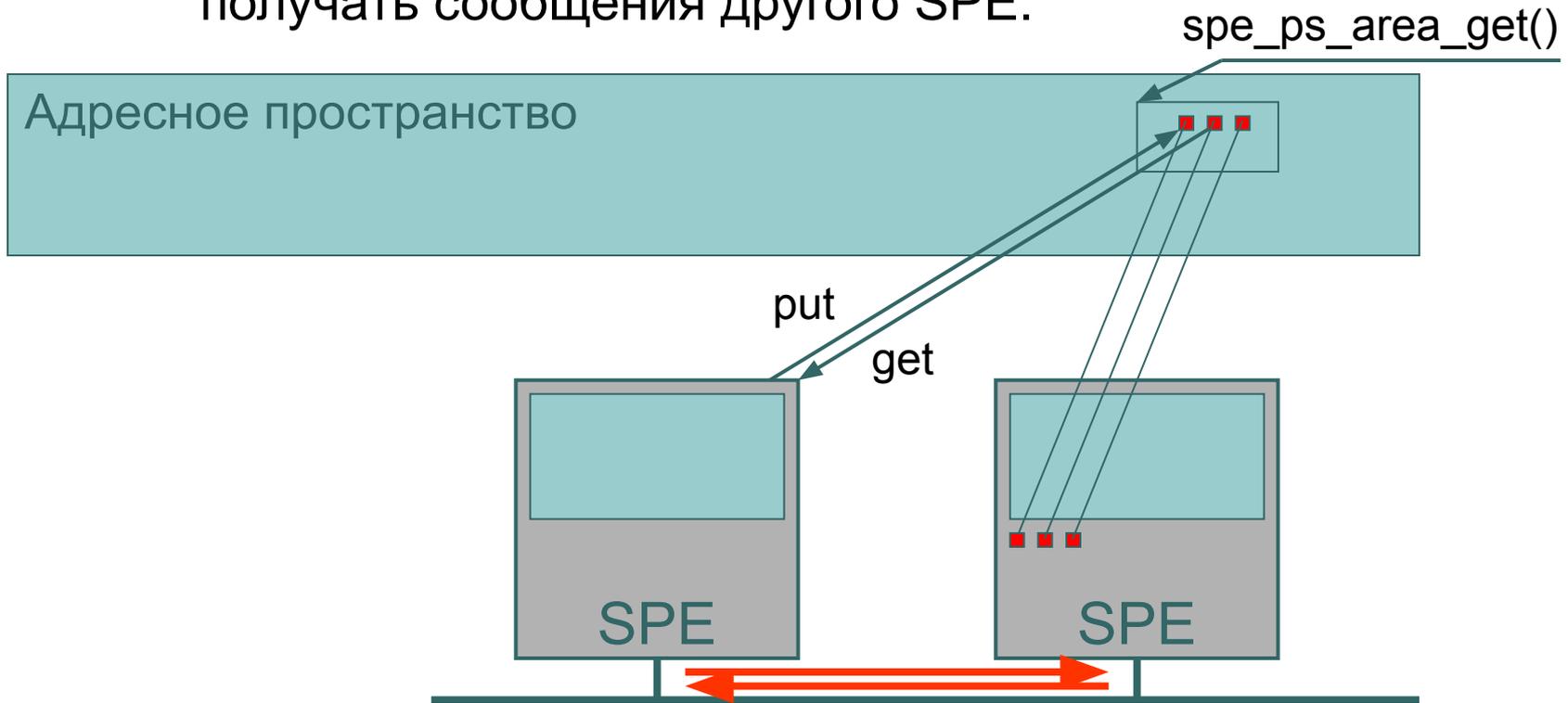
# Передача данных между SPE

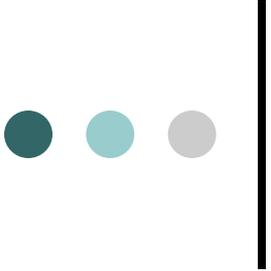
- Локальная память SPE отображается в общее адресное пространство.
- Обращение SPE к заданному участку адресного пространства позволяет обращаться к памяти другого SPE.



# Передача сообщений между SPE

- Регистры mailbox-ов и сигналов SPE отображаются в общее адресное пространство.
- Запись/чтение заданного элемента адресного пространства позволяет одному SPE отправлять и получать сообщения другого SPE.



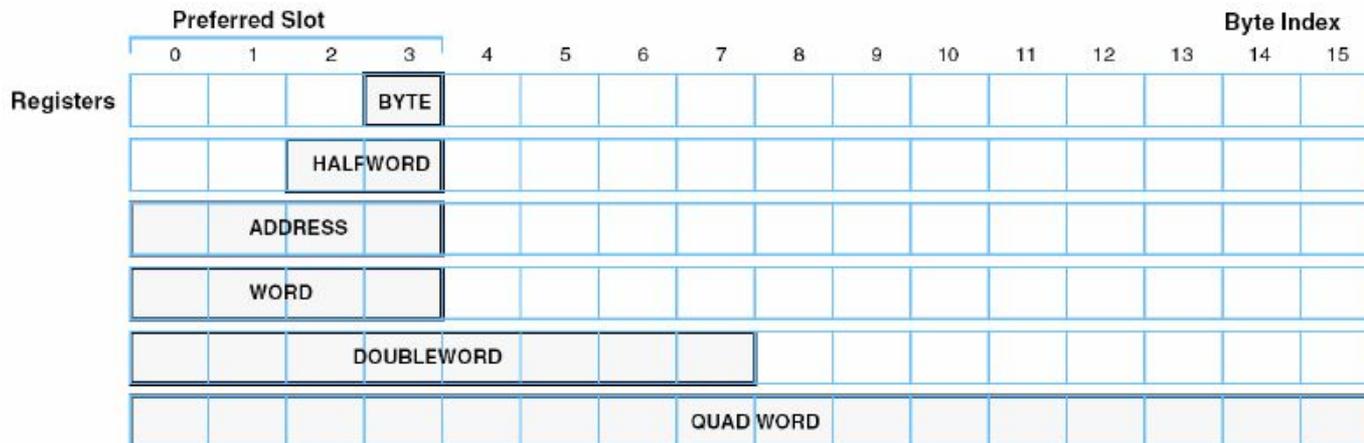


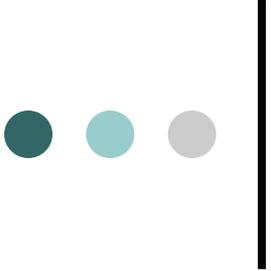
# План

- Архитектура процессора Cell
- Принципы программирования
- Базовые средства программирования
  - Создание и компиляция простых программ
  - Механизмы передачи данных и сообщений
- **Программирование вычислений на SPE**
- Библиотеки

# Программирование вычислений на SPE

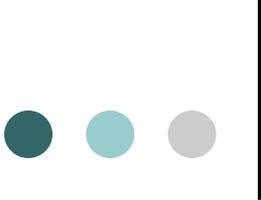
- Все регистры SPE векторные длиной 16 байт.
- Все арифметические инструкции являются параллельными по данным и работают с векторами.
- Скалярные операции выполняются путем соответствующего использования векторных команд.
- Скалярные аргументы, необходимые некоторым инструкциям, располагаются в «предпочтительных» позициях вектора:





# Локальная память SPE

- Объем локальной памяти для данных и кода: 256 KB
- Чтение и запись локальной памяти всегда выполняется выровненными блоками размером 16 байт:
  - Если адрес не выровнен, он автоматически округляется до 16 байт
- Защита памяти отсутствует
  - Можно свободно читать и писать в области данных, кода и стека



# Векторные типы данных

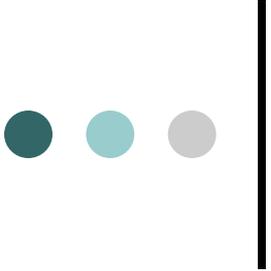
Vector Data Type	PPE	SPU
vector unsigned char	x	x
vector signed char	x	x
vector bool char	x	—
vector unsigned short	x	x
vector signed short	x	x
vector bool short	x	—
vector pixel	x	—
vector unsigned int	x	x
vector signed int	x	x
vector bool int	x	—
vector float	x	x
vector unsigned long long	—	x
vector signed long long	—	x
vector double	—	x

# Векторные типы данных

- Элементы векторных типов данных выравниваются автоматически.
- Элементы других типов можно выравнивать явно:
  - `unsigned char buffer[1024] \_\_attribute\_\_\(\(aligned\(16\)\)\);`
- Пример инициализации векторной переменной:
  - `vector float vf1 = { 1.0, 2.0, 3.0, 4.0 };`
  - `vector float vf2[2] = {{1.0, 2.0, 3.0, 4.0}, {5.0, 6.0, 7.0, 8.0}};`
  - `vector float vf3[2] = {1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0};`

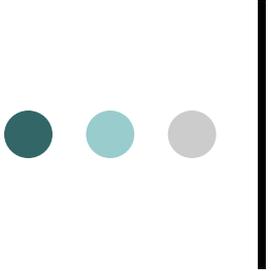
# Команды уровня компилятора

- Для манипуляции векторными данными используются `intrinsics` – встроенные в компилятор команды:
  - **Специальные (`specific`)** – отображаются в одну инструкцию процессора,
    - Например: `d = si_to_int(a);`
  - **Обобщенные (`generic`)** – отображаются в одну или несколько инструкций процессора в зависимости от входных параметров,
    - Например: `c = spu_add (a, b);`
  - **Составные (`built-ins`)** – последовательности обобщенных и специальных `intrinsics` (объединенные для удобства).
    - Например, команды DMA-передачи.



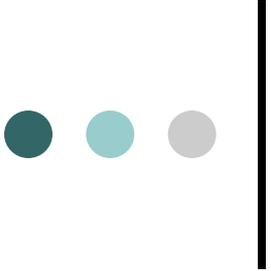
# Операции над векторами

- Арифметические операции над элементами векторов:
  - `d = spu_add(a,b);`
  - `d = spu_sub(a, b);`
  - `d = spu_madd(a, b, c);`
  - `d = spu_mul(a, b);`
  - ...
- Логические операции над битами векторов:
  - `d = spu_and(a, b);`
  - `d = spu_or(a, b);`
  - `d = spu_eqv(a, b);`
  - ...
- Более сложные операции: библиотека `simdmath`.



# Операции над векторами

- Операции преобразования скалярных и векторных данных:
  - `d = spu_insert(s, v, n);`
  - `d = spu_splats(s);`
  - `d = spu_promote(s, n);`
  - `s = spu_extract(v, n);`
- Операции преобразования типов:
  - `d = spu_convtf(a, scale);`
  - `d = spu_convts(a, scale);`
  - `d = spu_extend(a);`
  - ...



# Операции над векторами

- **Операции для манипулирования элементами векторов:**
  - **Сдвиг, вращение элементов**
    - `d = spu_rl(a, count);`
    - `d = spu_sl(a, count);`
    - ...
  - **Перестановка элементов векторов**
    - `d = spu_sel(a, b, pattern);`
    - `d = spu_shuffle(a, b, pattern);`
    - ...



# Пример:

## векторное умножение матриц

```
void mulv (float *a, float *b, float *c, int n)
{
    int i, j, k;
    vector float *bv = (vector float *) b;
    vector float *cv = (vector float *) c;
    vector float s, t;

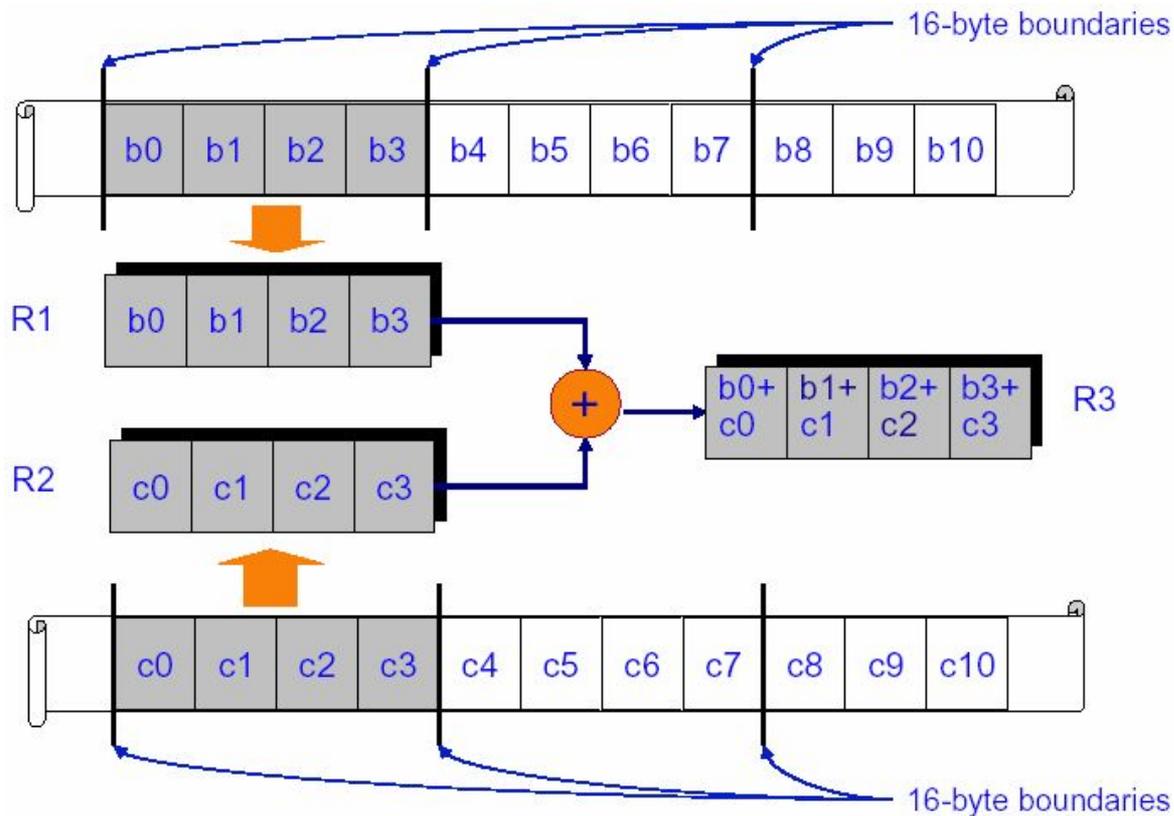
    s = spu_splats(0.0);
    for (i=0; i<n*n/4; i++) cv[i] = s;

    for (i=0; i<n; i++)
        for (k=0; k<n; k++)
            {
                s = spu_splats ( a[i*n+k] );
                for (j=0; j<n/4; j++)
                    { t = spu_mul (s, bv[k*n/4+j] );
                      cv[i*n/4+j] = spu_add ( cv[i*n/4+j], t );
                    }
            }
}
```

# Особенности векторизации вычислений

- Простой пример:

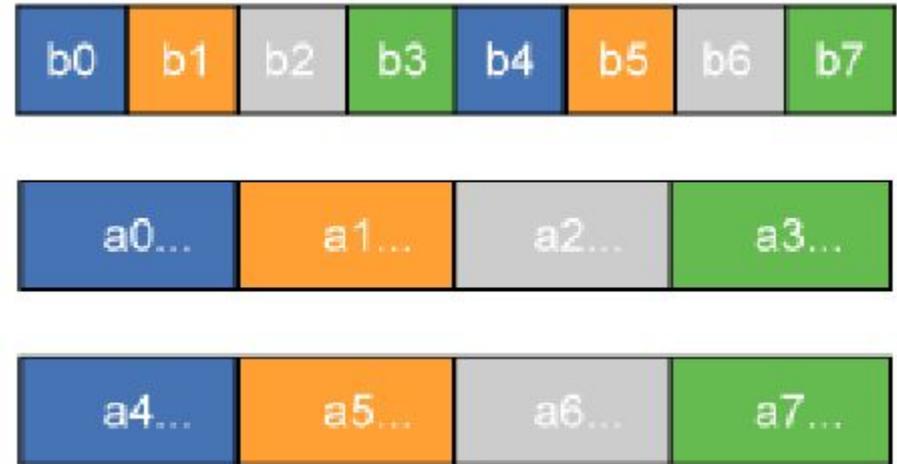
```
for (i=0; i<n; i++) a[i]=b[i]+c[i];
```



# Особенности векторизации вычислений

- Операнды должны быть приведены к одному типу
- Пример:

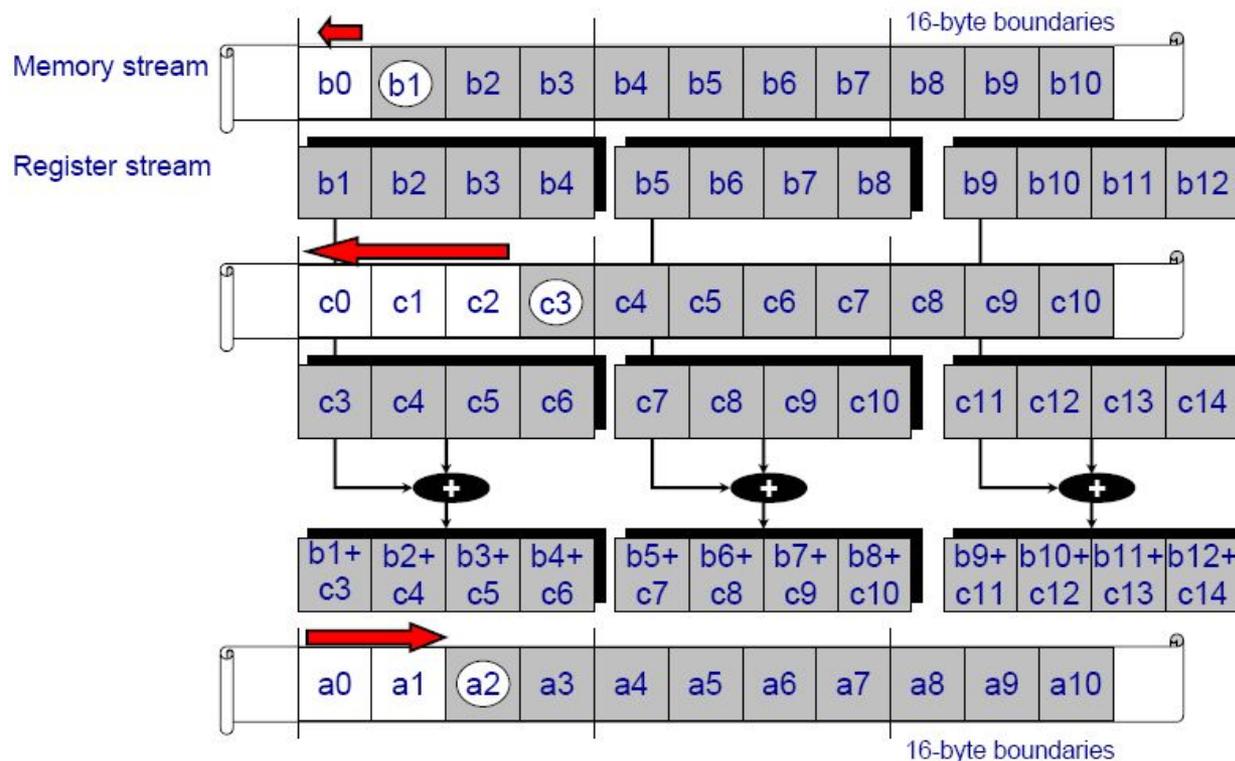
```
int a[n];  
short int b[n];  
for (i=0; i<n; i++)  
    a[i] = a[i] + b[i];
```

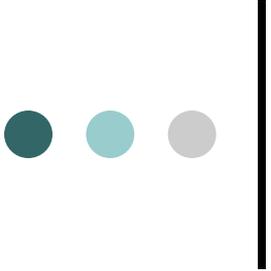


# Особенности векторизации вычислений

- Операнды должны быть выровнены друг относительно друга

□ SIMD execution of “for(i=0;i<65;i+) a[i+2] = b[i+1] + c[i+3]”

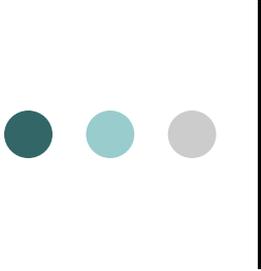




# Особенности реализации скалярных вычислений

- При выполнении операции над скалярными переменными в памяти происходит следующее:
  - Чтение векторов, содержащих операнды, в регистры,
  - Относительное выравнивание операндов в векторах,
  - Выполнение операции над векторами,
  - Чтение вектора, содержащего результат,
  - Помещение результата в нужную позицию результирующего вектора,
  - Запись результирующего вектора в память.

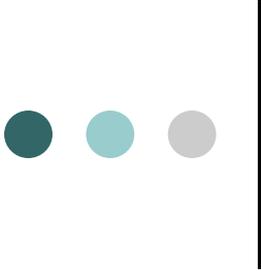
Это очень долго!



# Оптимизация целочисленной арифметики

- На SPE отсутствует 32-битное умножение.
- Используйте `short int` вместо `int`, где ЭТО ВОЗМОЖНО.

```
short int i, k;  
for (i=0; i<N; i++)  
    for (k=0; k<N; k++)  
        x[i*N+k]=i + k;
```



# Оптимизация условных переходов

## □ Устранение условных переходов

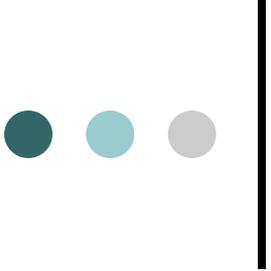
```
...  
if (a > b)  
    c += 1;  
else  
    d = a + b;  
...
```

```
...  
select = spu_cmpgt(a, b);  
c_plus_1 = spu_add(c, 1);  
a_plus_b = spu_add(a, b);  
c = spu_sel (c, c_plus_1, select);  
d = spu_sel (a_plus_b, d, select);  
...
```

## □ Подсказка компилятору

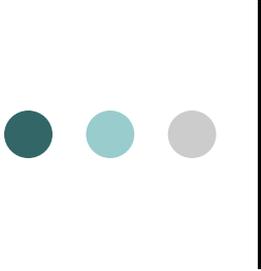
```
...  
if (a > b)  
    c += 1;  
else  
    d = a + b;  
...
```

```
...  
if ( __builtin_expect(a>b, 1) )  
    c += 1;  
else  
    d = a + b;  
...
```



# План

- Архитектура процессора Cell
- Принципы программирования
- Базовые средства программирования
  - Создание и компиляция простых программ
  - Механизмы передачи данных и сообщений
- Программирование вычислений на SPE
- **Библиотеки**



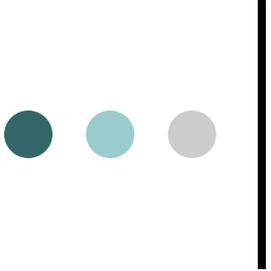
# Средства программирования процессоров Cell

## ▣ Средства от IBM (IBM Cell SDK)

- Библиотека `libspe 2.0`
- Библиотеки векторизованных операций: `SIMD Math Library`, `MASS Library`, `FFT`, `Game math`, `Image Processing`, `Matrix`, `Vector`, `Multi-precision math`, `BLAS`, `LAPACK`, `Monte-Carlo`
- `Sync Library`: атомарные операции, мьютексы, условные переменные, ...
- `Software managed cache`
- Распараллеливающие векторизующие компиляторы (OpenMP): `xlc`, `xlf`
- `DaCS` – `Data Communication and Synchronization library`
- `ALF` – `Accelerated Library Framework`

## ▣ Другие средства

- `BSC Cell Superscalar`
- `Mercury Computer Systems: MultiCore Framework`
- `RapidMind`
- `Gedae`



# ССЫЛКИ

- IBM Cell Broadband Engine resource center, <http://www.ibm.com/developerworks/power/cell/documents.html>
- Cell Developer's Corner, <http://www.power.org/resources/devcorner/cellcorner>
- STI Center of Competence for the Cell Broadband Engine Processor, <http://sti.cc.gatech.edu>
- Barcelona Supercomputing Center: Cell Superscalar, [www.bsc.es/cellsuperscalar](http://www.bsc.es/cellsuperscalar)
- RapidMind Development Platform, [www.rapidmind.net](http://www.rapidmind.net)
- Mercury Computer Systems: MultiCore Framework, [http://www.mc.com/software/multicore\\_framework.aspx](http://www.mc.com/software/multicore_framework.aspx)

