СУПЕРКОМПЬЮТЕРНЫЙ КОНСОРЦИУМ УНИВЕРСИТЕТОВ РОССИИ

Проект

Создание системы подготовки высококвалифицированных кадров в области суперкомпьютерных технологий и специализированного программного обеспечения

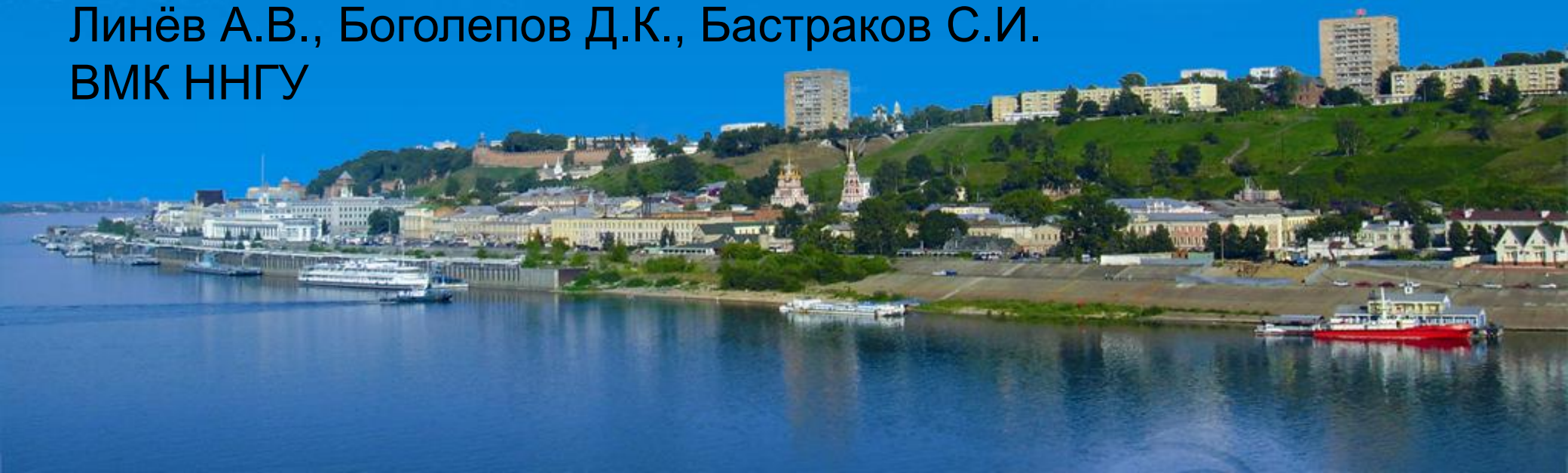
Московский государственный университет им. М.В. Ломоносова

Нижегородский государственный университет
им. Н.И. Лобачевского
- Национальный исследовательский университет -

Технологии параллельного программирования для процессоров новых архитектур

Лекция 5. Программирование для архитектуры Cell BE, часть 2

Линёв А.В., Боголепов Д.К., Бастраков С.И.
ВМК ННГУ



Содержание

- Последовательность разработки приложений для Cell BE
- Как достичь большой производительности
- Программа "Hello, world!" для Cell BE

Последовательность разработки приложений для Cell BE

Последовательность разработки приложения…

- Изучение алгоритма и его сложности
- Размещение данных, локальность данных, потоки данных
- Разработка параллельного алгоритма и экспериментальная реализация для Cell BE
- Реализация PPE-кода (управляющая часть программы), скалярной версии SPE-кода
- Реализация PPE-кода, параллельной скалярной версии SPE-кода

Последовательность разработки приложения

- Реализация PPE-кода, параллельной скалярной версии SPE-кода
  - В т.ч. взаимодействие SPE–блоков, double buffering
  - Перевод SPE-кода в SPE SIMD код
  - Балансировка вычислений и передачи данных
  - Другая оптимизация
  - Перевод PPE-кода в PPE SIMD код, устранение бутылочных горлышек, балансировка загрузки и т.д.

Как достичь большой производительности на Cell BE

Оптимизация кода…

- Разбиение приложения на параллельно выполняющиеся блоки
  - Попробуйте различные альтернативы
  - Чем больше кода выполняется на SPE – тем лучше
  - “SPE-centric” – лучше чем “PPE-centric”
  - Выгружайте на SPE функции, которые автономны и не требуют синхронизации
- DMA
  - Используйте DMA передачи, инициированные SPE, а не PPE
  - Используйте multiple-buffering на SPE
  - Используйте выравнивание передаваемых блоков данных по границе линейки кеша (128B)

Оптимизация кода…

- PPE код
  - Используйте программную предвыборку данных (‘dcbt’)
  - Избегайте доступа с PPE к массивам данных, предназначенным для обработки на SPE (чтобы DMA передачи, инициированные SPE, выполняли передачу из основной памяти, а не из кеша L2 PPE)
  - Используйте PPE dual-threading
  - Используйте большие размеры страниц для уменьшения потерь за счет исчерпания TLB
    - Cell BE поддерживает одновременно 3 размера страниц: 4Кб и 2 из 64 Кб, 1 Мб и 16 Мб.

Оптимизация кода…

- SIMD
  - Пробуйте различные варианты векторизации
  - Используйте возможность реорганизации данных в регистрах (это производится параллельно с вычислениями)
  - Используйте по возможности ‘vector select’ вместо конструкций ‘if-then-else’
- Структуры данных
  - Проектируйте структуры дауных для эффективного использования на SPE
  - Учитывайте выравнивания и предполагаемую последовательность обращений
  - MFC поддерживает передачу 1,2,4,8,n*16 (до 16K) байт
    - Передачи менее 16 Б также должны быть выровнены по границе 16 Б
- SPE – циклы (особенно вложенные)
  - Разворачивайте циклы
  - В частных случаях увеличение числа инструкций, выполняемых в цикле, может уменьшить время выполнения итерации
  - Используйте SPE static timing analysis

Оптимизация кода

- SPE – предсказание переходов
  - Используйте ‘branch hints’ (обычно с этим справляется компилятор)
- SPE – целочисленная арифметика
  - Избегайте умножения целых числе (на SPE не реализовано умножение 32-битных целых)
  - SPE не поддерживает арифметику с переполнением
- SPE – SPE intrinsics
  - Используйте SPE intrinsics (расширения языка программирования) для явного указания желаемых операций
- SPE – дополнительное замечание
  - SPE содержит 2 конвейера, каждый из которых выполняет свое подмножество инструкций
  - Если 2 последовательных инструкции не содержат зависимостей и должны выполняться на разных конвейерах – они выполняются одновременно

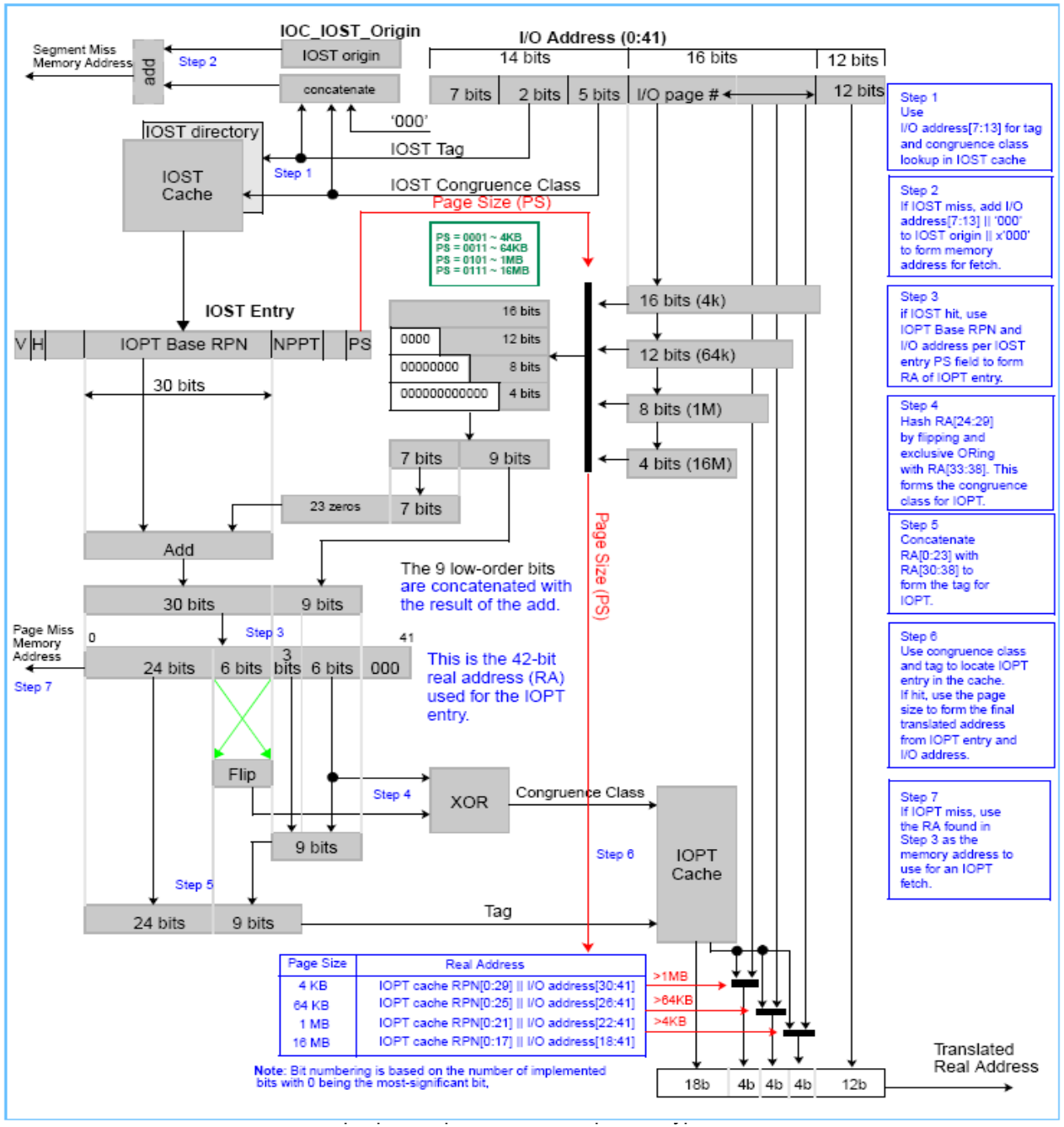
Хорошо портируемый код (до 100x)…

- Можно использовать технику double-buffering (алгоритм имеет вид: gather->compute->scatter)
- Хорошо структурированный (предсказуемая последовательность обращений к памяти)
  - Можно построить списки доступа к данным
  - Можно организовать предвыборку
  - Можно избежать условных переходов
- Много операций над одними и теми же данными
- Простая параллелизация
  - Минимум коллективных операций и синхронизации
  - Нет использования общей памяти

Хорошо портируемый код (до 100x)

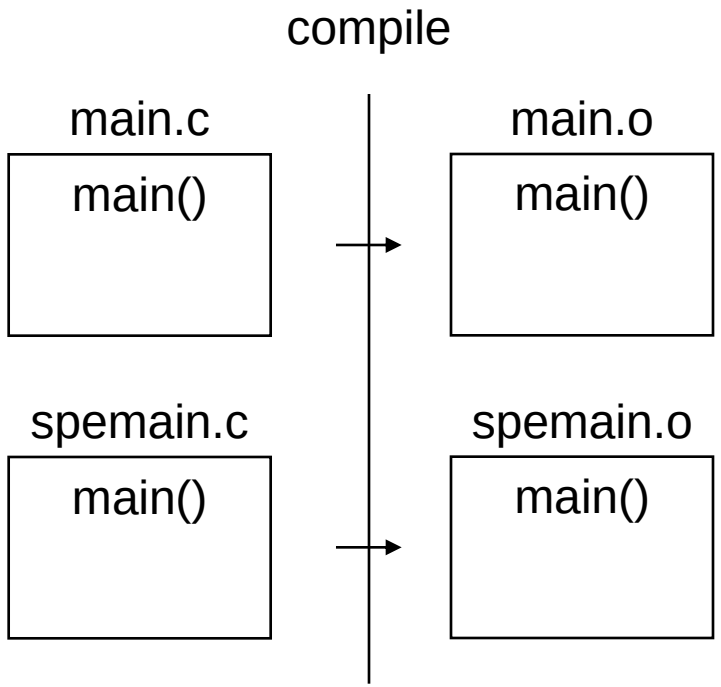
- Много вычислений
- Возможно использование потоковой моели обработки
- Вычисления в SP (32-bit float) или <=32-бит целые
- Примеры:
  - FFTw (best result about 100GFlops)
  - Terrain Rendering Engine
  - Volume rendering
  - Crypto codes (RSA, SHA, DES, etc. etc. etc.)
  - Media codes (MPEG 2, MPEG 4, H.264, JPEG)

К вопросу
о сложности
программирования
для Cell BE

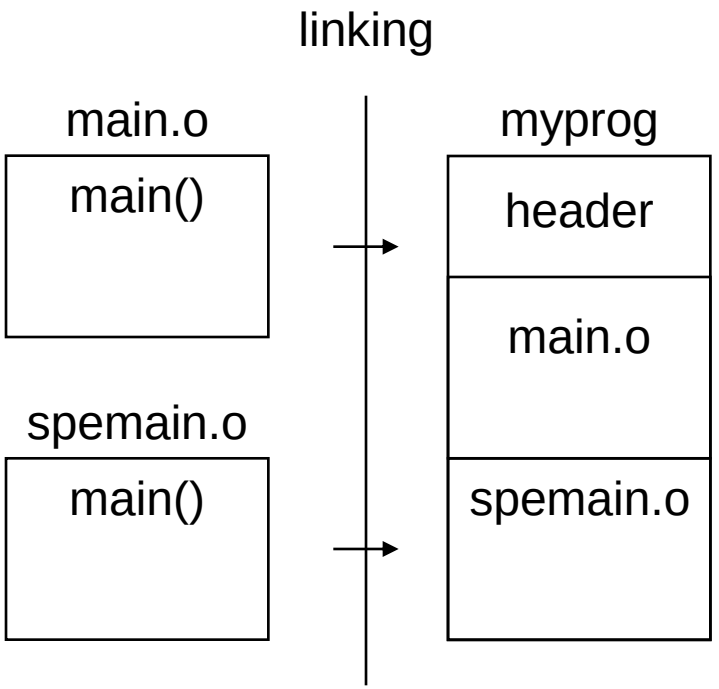


Программа "Hello, world!" для Cell BE

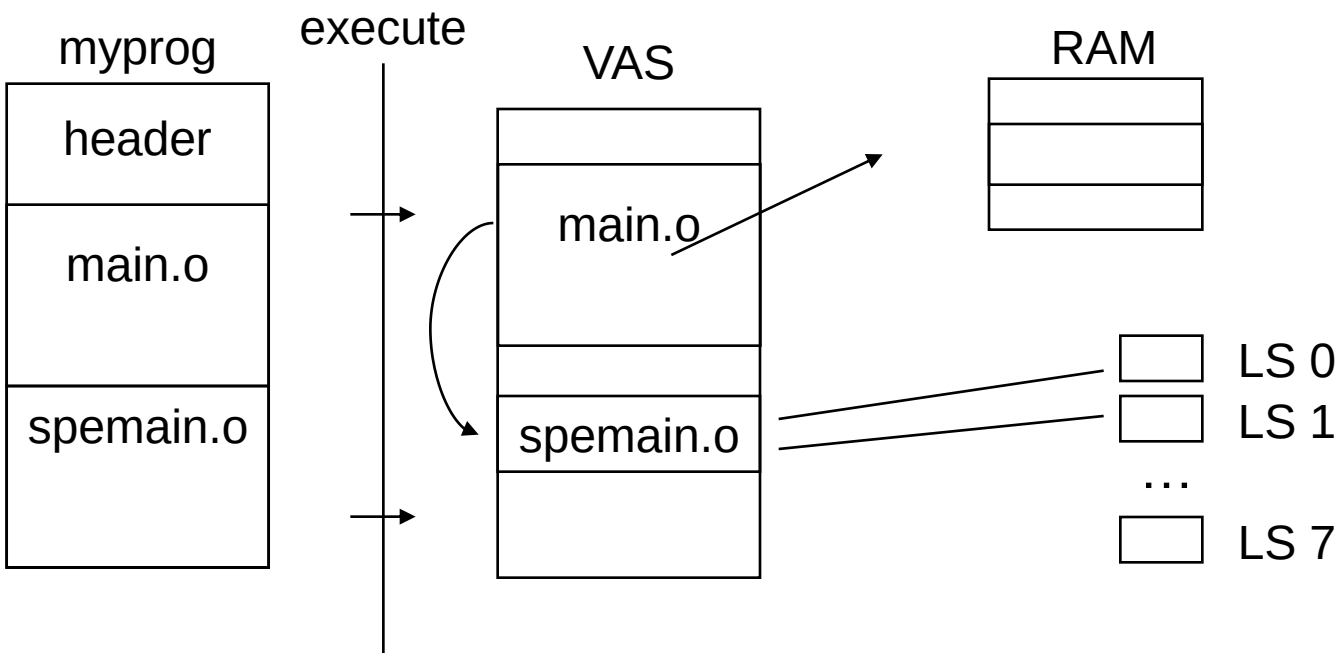
Компиляция приложения для Cell BE



Линковка приложения для Cell BE



Запуск приложения для Cell BE



Hello, world! – SPE-часть

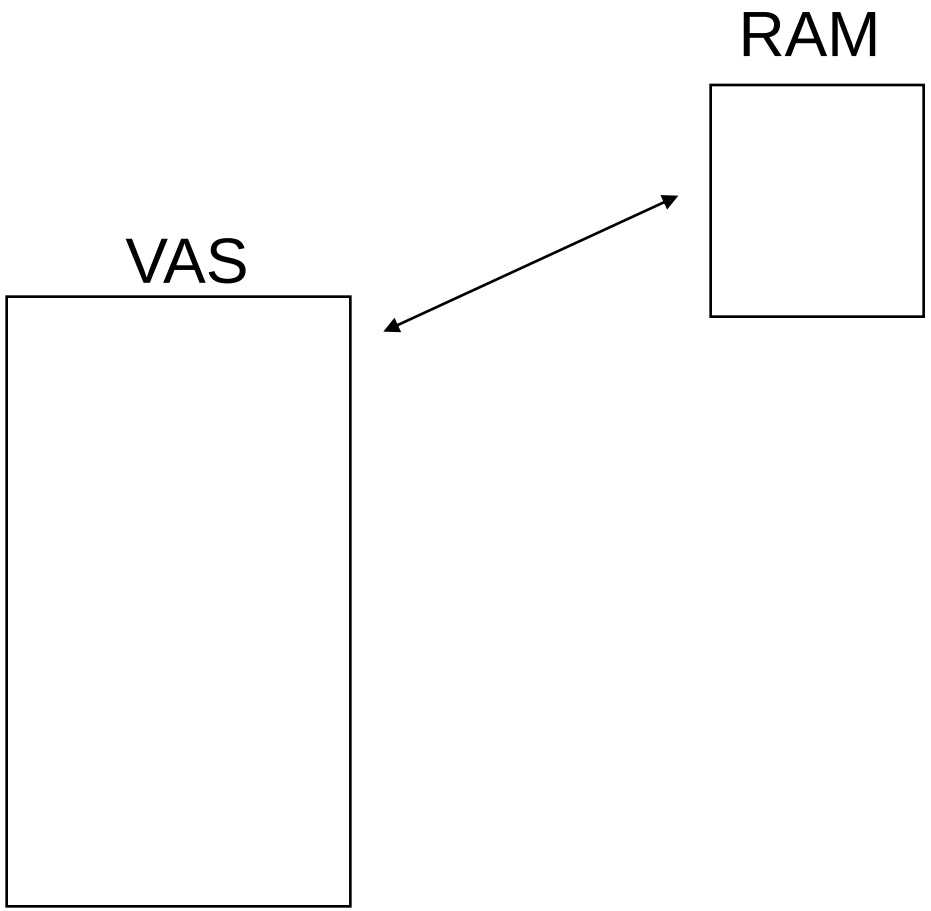
```
#include <stdio.h>

int main()
{
    printf("Hello, world!\n");
    return 0;
}
```

Hello, world! – PPE-часть (1 SPE-поток)

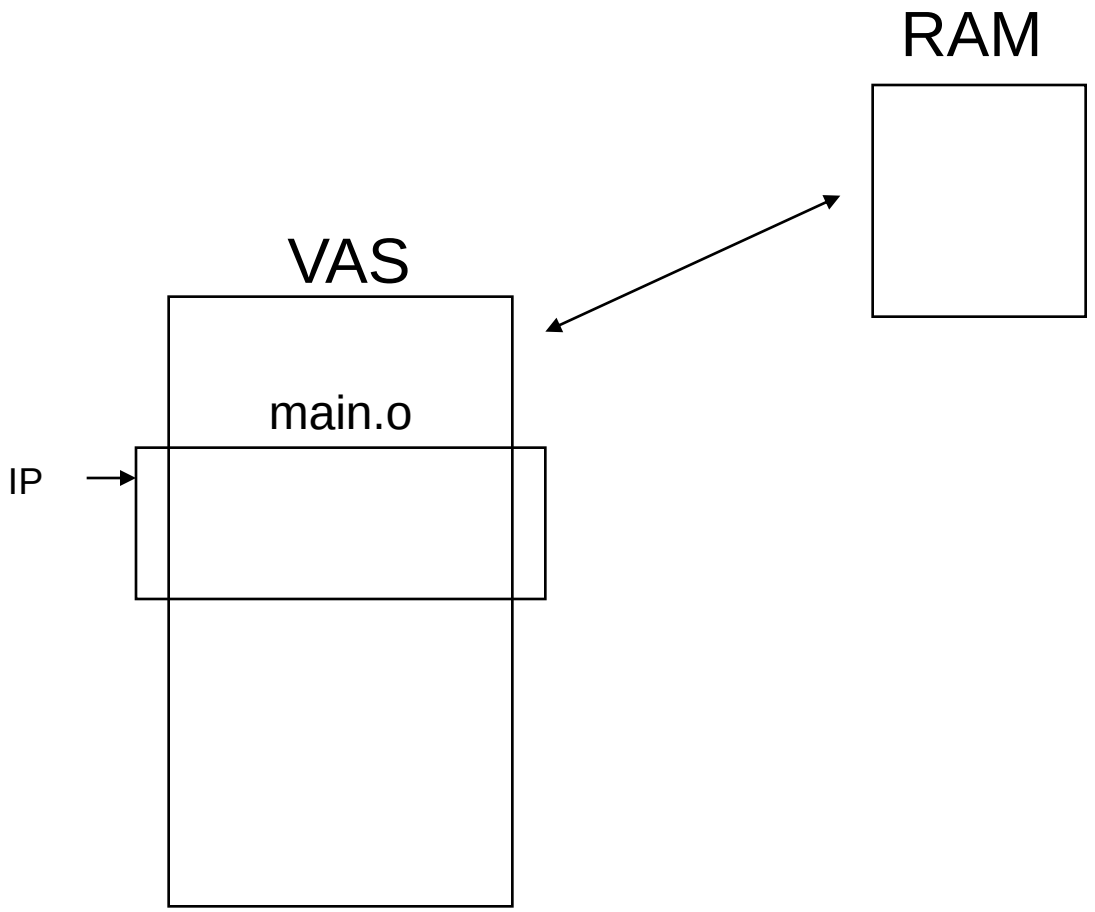
```
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <libspe2.h>
extern spe_program_handle_t hello_spu;
int main(void)
{
 // Structure for an SPE context
 spe_context_ptr_t speid;
 unsigned int flags = 0;
 unsigned int entry =
   SPE_DEFAULT_ENTRY;
 void * argp = NULL;
 void * envp = NULL;
 spe_stop_info_t stop_info;
 int rc;
 // Create an SPE context
 speid = spe_contex_creat(flags, NULL);
 if (speid == NULL) {
   perror("spe_context_create");
 return -2;
}
```

```
// Load an SPE executable object into
// the SPE context local store
if(spe_program_load(speid, hello_spu) )
{
  perror("spe_program_load");
  return -3;
}

// Run the SPE context
rc = spe_context_run(speid, &entry, 0,
                argp, envp, &stop_info);
if (rc < 0)
{
  perror("spe-context-run");
 // Destroy the SPE context
 spe_contex_deslroy(speid);
 return 0;
}
```

Выполнение программы…

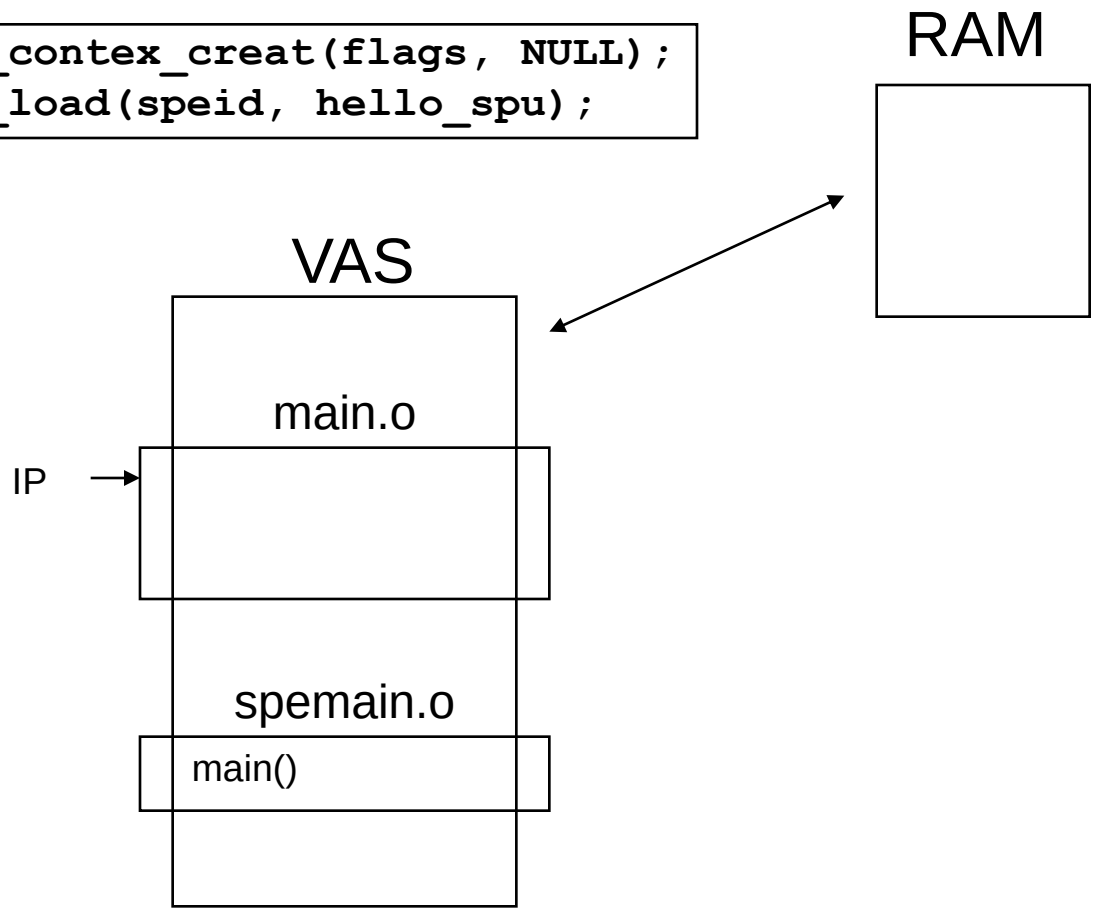


Выполнение программы…



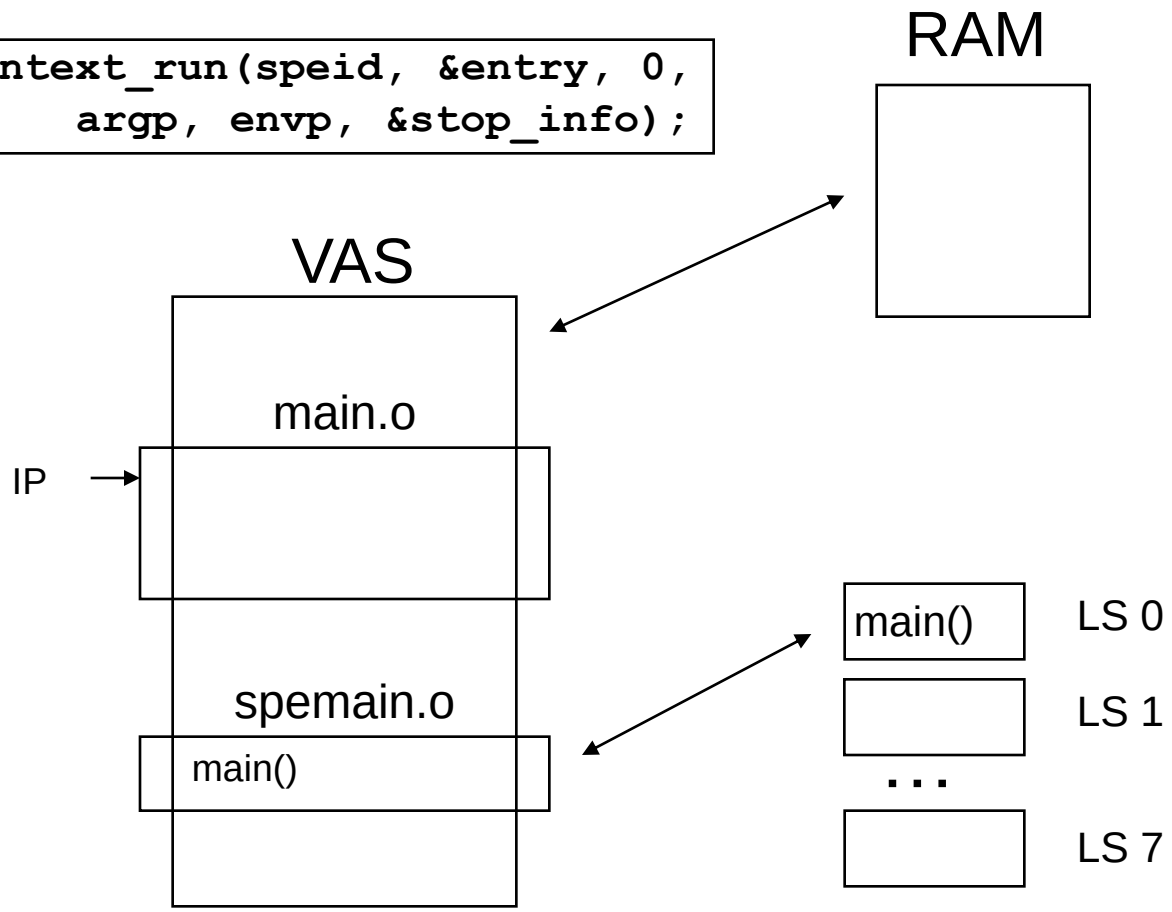
Выполнение программы…

```
speid = spe_contex_creat(flags, NULL);
spe_program_load(speid, hello_spu);
```



Выполнение программы

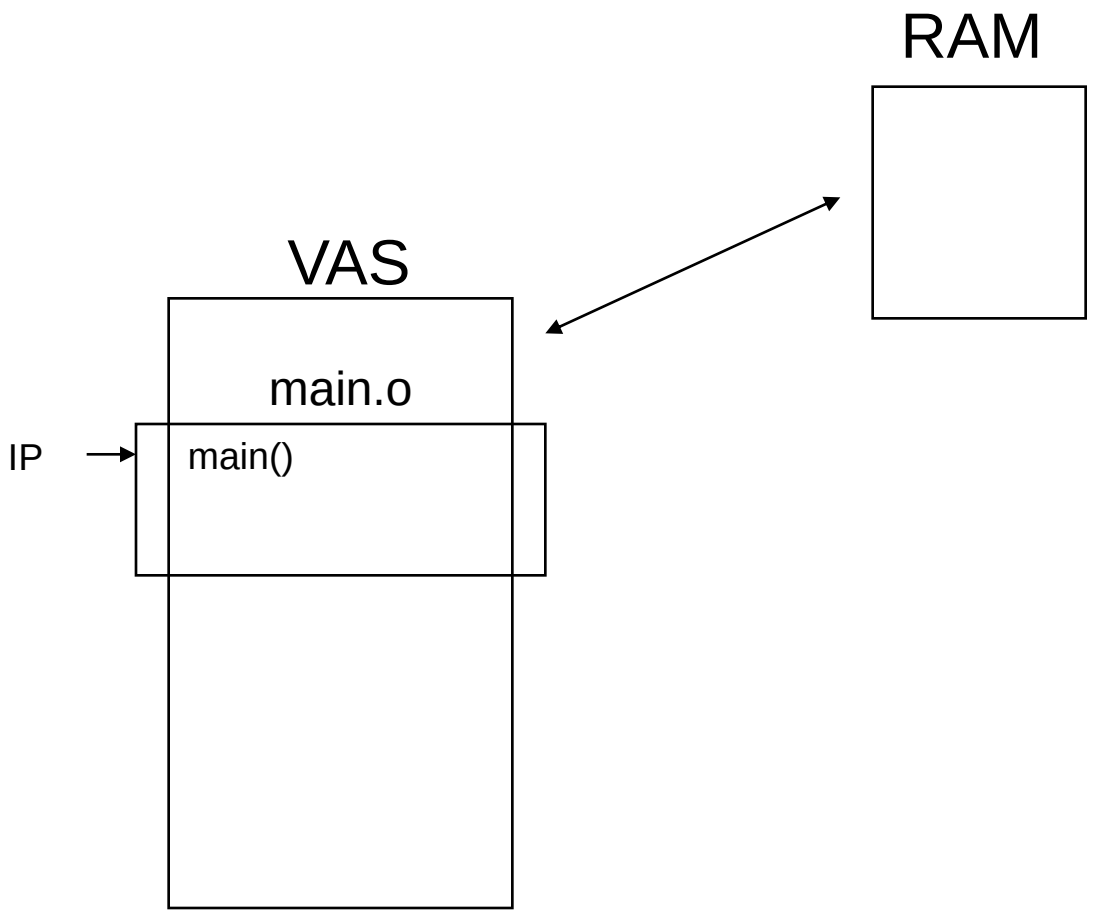
```
rc = spe_context_run(speid, &entry, 0,
                     argp, envp, &stop_info);
```



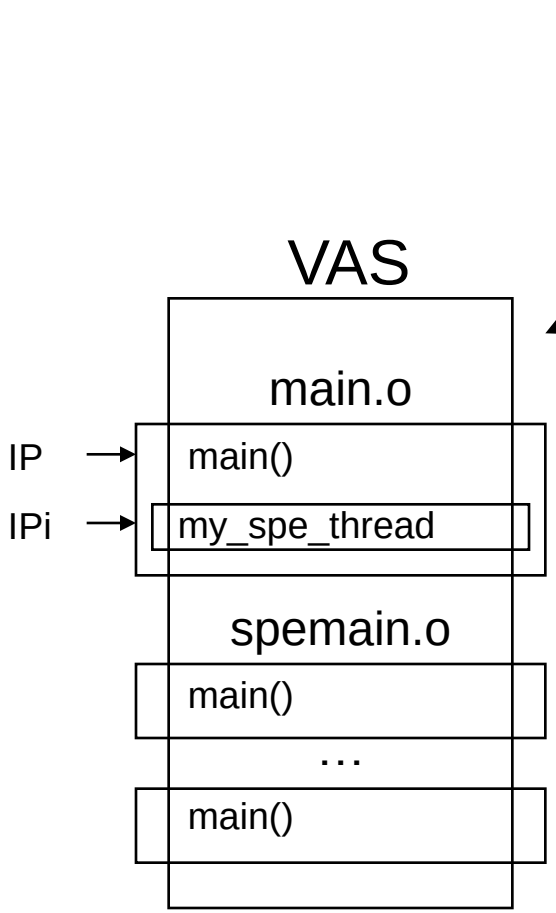
Hello, world! – PPE-часть (N SPE-потоков)

```
#include <stdlib.h>
#include <pthread.h>
#include <libspe2.h>
#define N 4

struct thread_args {
 struct spe_context spe;
 void * argp;
 void * envp;
};

void my_spe_thread(struct thread_args * arg)
    {
 unsigned int runflags = 0;
 unsigned int entry = SPE_DEFAULT_ENTRY;
 // run SPE context
 spe_context_run(arg->spe, &entry,
    runflags,arg->argp, arg->envp, NULL);
 // done - now exit thread
 pthread_exit (NULL) ;
}

int main() (
 pthread_t pts[N];
 spe_context_ptr_t spe[N];
 struct thread_args t_args[N];
 int value[N];
 int i;
 // open SPE program
 spe_program_handle_t * program;
 program = spe_image_open("hello");
 for ( i=0; i<N; i++ ) {
  // create SPE context
  spe[i] = spe_context_create(0, NULL);
  // load SPE program
  spe_program_load( spe[i] , program) ;
  // create pthread
  t_args[i].spe = spe[i];
  t_args[i].argp = &value[i];
  t_args[i].envp = NULL;
  pthread_create( pts[i], NULL,
    &my_spe_thread, t_args[i]);
 }

 // wait for all threads to finish
 for ( i=0; i<N; i++ ) {
  pthread_join (pts[i], NULL);
 }
 // close SPE program
 spe_image_close(program);
 // destroy SPE contexts
 for ( i=O; i<N; i++ ) {
  spe_context_destroy(spe[i]);
 }
return 0;
}
```

Выполнение программы…



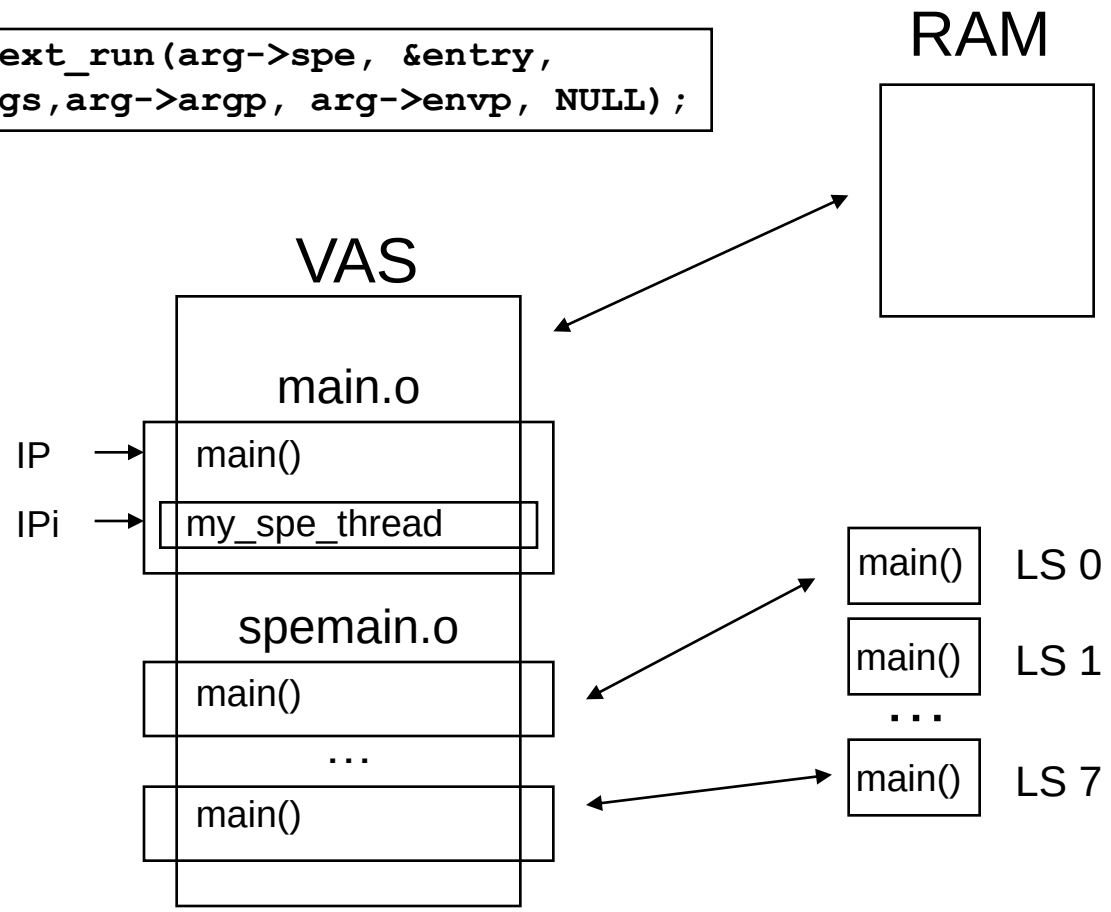
Выполнение программы…




```
for ( i=0; i<N; i++ ) {
  spe[i] = spe_context_create(0, NULL);
  spe_program_load( spe[i] , program) ;
  t_args[i].spe = spe[i];
  t_args[i].argp = &value[i];
  t_args[i].envp = NULL;
  pthread_create( pts[i], NULL,
                  &my_spe_thread, t_args[i]);
 }
```

Выполнение программы

```
spe_context_run(arg->spe, &entry,
  runflags,arg->argp, arg->envp, NULL);
```



Заключение

- Разработка приложений для архитектуры Cell BE требует знания аппаратных особенностей архитектуры и использования специальных техник оптимизации
  - «Наивный» подход к портированию приложений для архитектуры Cell BE не обеспечит достаточной производительности

- В следующей лекции – пример адаптации приложения для архитектуры Cell BE

Литература

- Cell BE Programming Handbook Including PowerXCell 8i, 2008.
- Software Development Kit for Multicore Acceleration, Programming Tutorial. Version 3.1, 2008.
- Abraham Arevalo, Ricardo M. Matinata, Maharaja Pandian, Eitan Peri, Kurtis Ruby, Francois Thomas, Chris Almond. Programming the Cell Broadband Engine: Examples and Best Practices. IBM International Technical Support Organization, 2008.

Контакты

- Нижегородский университет,
- Факультет вычислительной математики и кибернетики

- Линёв Алексей Владимирович (alin@unn.ru)
- Боголепов Денис Константинович (denisbogol@gmail.com)
- Бастраков Сергей Иванович (sergey.bastrakov@gmail.com)

СПАСИБО ЗА ВНИМАНИЕ

Вопросы ?

О проекте

Целью проекта является создание национальной системы подготовки высококвалифицированных кадров в области суперкомпьютерных технологий и специализированного программного обеспечения.

Задачами по проекту являются:

Задача 1. Создание сети научно-образовательных центров суперкомпьютерных технологий (НОЦ СКТ).

Задача 2. Разработка учебно-методического обеспечения системы подготовки, переподготовки и повышения квалификации кадров в области суперкомпьютерных технологий.

Задача 3. Реализация образовательных программ подготовки, переподготовки и повышения квалификации кадров в области суперкомпьютерных технологий.

Задача 4. Развитие интеграции фундаментальных и прикладных исследований и образования в области суперкомпьютерных технологий. Обеспечение взаимодействия с РАН, промышленностью, бизнесом.

Задача 5. Расширение международного сотрудничества в создании системы суперкомпьютерного образования.

Задача 6. Разработка и реализации системы информационного обеспечения общества о достижениях в области суперкомпьютерных технологий.

См. http://www/hpc-education.ru