Параллельное программирование

(Численные методы, алгоритмы и программы. Введение в распараллеливание)

Лисицын Сергей
ФРКТ МФТИ 2019 г.

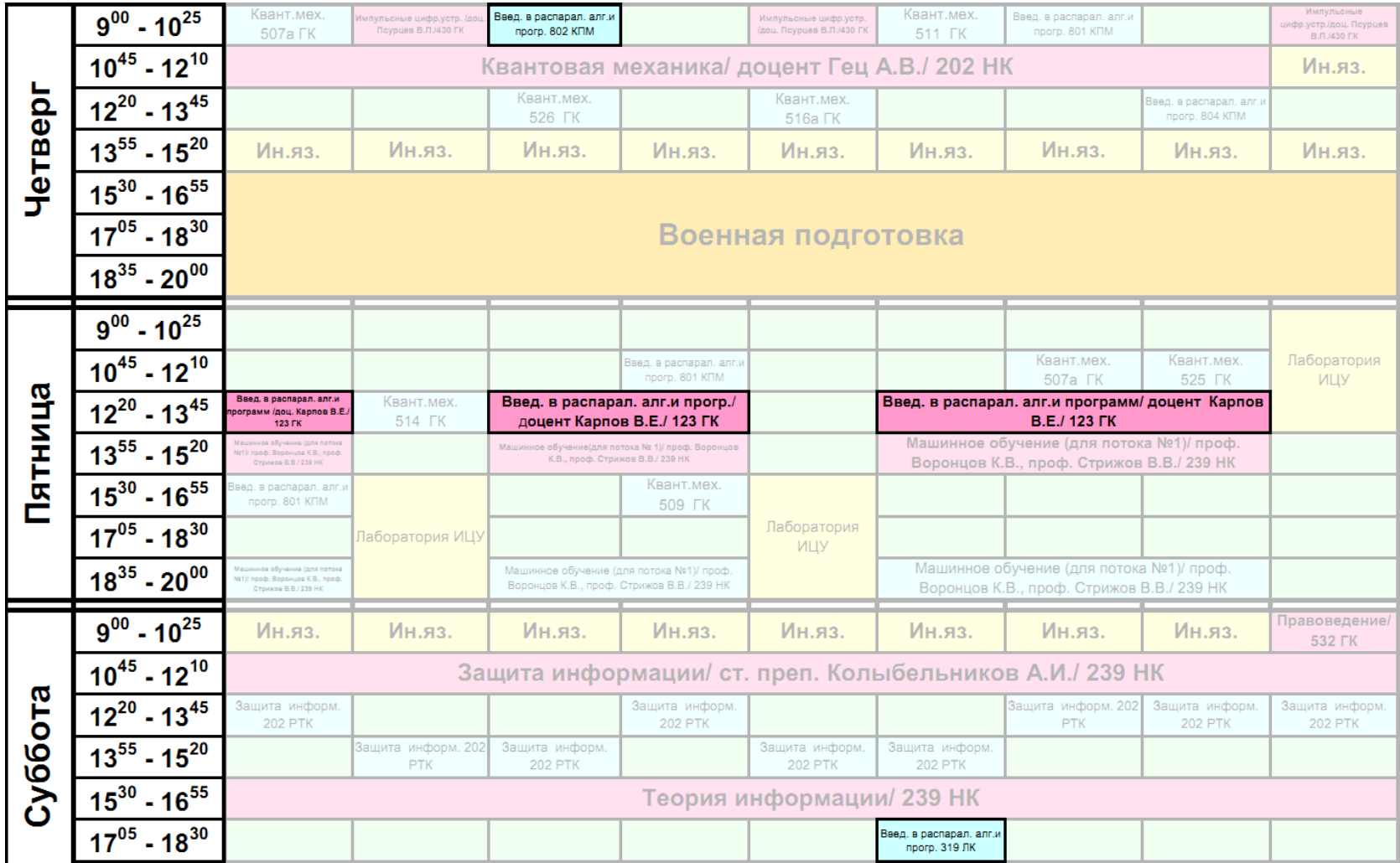
Программа 2 семестра

15 занятий:
Сентябрь (4) – OpenMP

Октябрь (4) – Распараллеливание Циклов

Ноябрь (5) – Вычматы

Декабрь (2) – Долги



Четверг	9^{00} - 10^{25}	Квант.мех. 507а ГК	Импульсные цифр.устр. /доц. Псурцев В.П./430 ГК	Введ. в распарал. алг.и прогр. 802 КПМ		Импульсные цифр.устр. /доц. Псурцев В.П./430 ГК	Квант.мех. 511 ГК	Введ. в распарал. алг.и прогр. 801 КПМ		Импульсные цифр.устр./доц. Псурцев В.П./430 ГК
	10^{45} - 12^{10}	Квантовая механика/ доцент Гец А.В./ 202 НК								Ин.яз.
	12^{20} - 13^{45}			Квант.мех. 526 ГК		Квант.мех. 516а ГК			Введ. в распарал. алг.и прогр. 804 КПМ	
	13^{55} - 15^{20}	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.
	15^{30} - 16^{55}	Военная подготовка								
	17^{05} - 18^{30}									
	18^{35} - 20^{00}									
Пятница	9^{00} - 10^{25}									Лаборатория ИЦУ
	10^{45} - 12^{10}				Введ. в распарал. алг.и прогр. 801 КПМ			Квант.мех. 507а ГК	Квант.мех. 525 ГК	
	12^{20} - 13^{45}	Введ. в распарал. алг.и программ /доц. Карпов В.Е./ 123 ГК	Квант.мех. 514 ГК	Введ. в распарал. алг.и прогр./ доцент Карпов В.Е./ 123 ГК			Введ. в распарал. алг.и программ/ доцент Карпов В.Е./ 123 ГК			
	13^{55} - 15^{20}	Машинное обучение (для потока №1)/ проф. Воронцов К.В., проф. Стрижов В.В./ 239 НК		Машинное обучение(для потока № 1)/ проф. Воронцов К.В., проф. Стрижов В.В./ 239 НК			Машинное обучение (для потока №1)/ проф. Воронцов К.В., проф. Стрижов В.В./ 239 НК			
	15^{30} - 16^{55}	Введ. в распарал. алг.и прогр. 801 КПМ	Лаборатория ИЦУ		Квант.мех. 509 ГК	Лаборатория ИЦУ				
	17^{05} - 18^{30}									
	18^{35} - 20^{00}	Машинное обучение (для потока №1)/ проф. Воронцов К.В., проф. Стрижов В.В./ 239 НК		Машинное обучение (для потока №1)/ проф. Воронцов К.В., проф. Стрижов В.В./ 239 НК			Машинное обучение (для потока №1)/ проф. Воронцов К.В., проф. Стрижов В.В./ 239 НК			
Суббота	9^{00} - 10^{25}	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Правоведение/ 532 ГК
	10^{45} - 12^{10}	Защита информации/ ст. преп. Колыбельников А.И./ 239 НК								
	12^{20} - 13^{45}	Защита информ. 202 РТК			Защита информ. 202 РТК			Защита информ. 202 РТК	Защита информ. 202 РТК	Защита информ. 202 РТК
	13^{55} - 15^{20}		Защита информ. 202 РТК	Защита информ. 202 РТК		Защита информ. 202 РТК	Защита информ. 202 РТК			
	15^{30} - 16^{55}	Теория информации/ 239 НК								
	17^{05} - 18^{30}						Введ. в распарал. алг.и прогр. 319 ЛК			

Программа 2 семестра

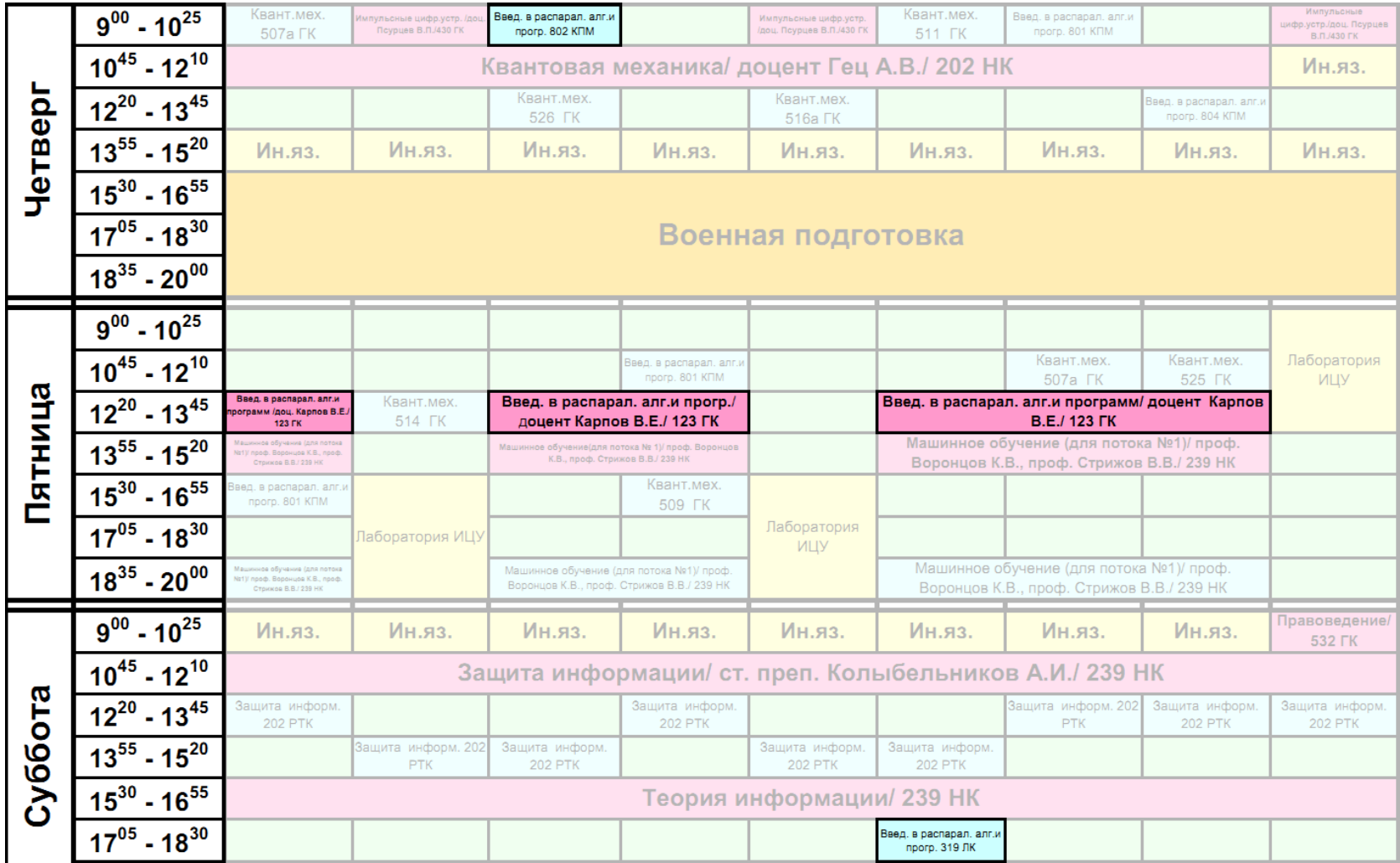
15 занятий:

Сентябрь (4) – OpenMP + Float арифметика

Октябрь (4) – Распараллеливание Циклов + Вычматы

Ноябрь (5) – Архитектура

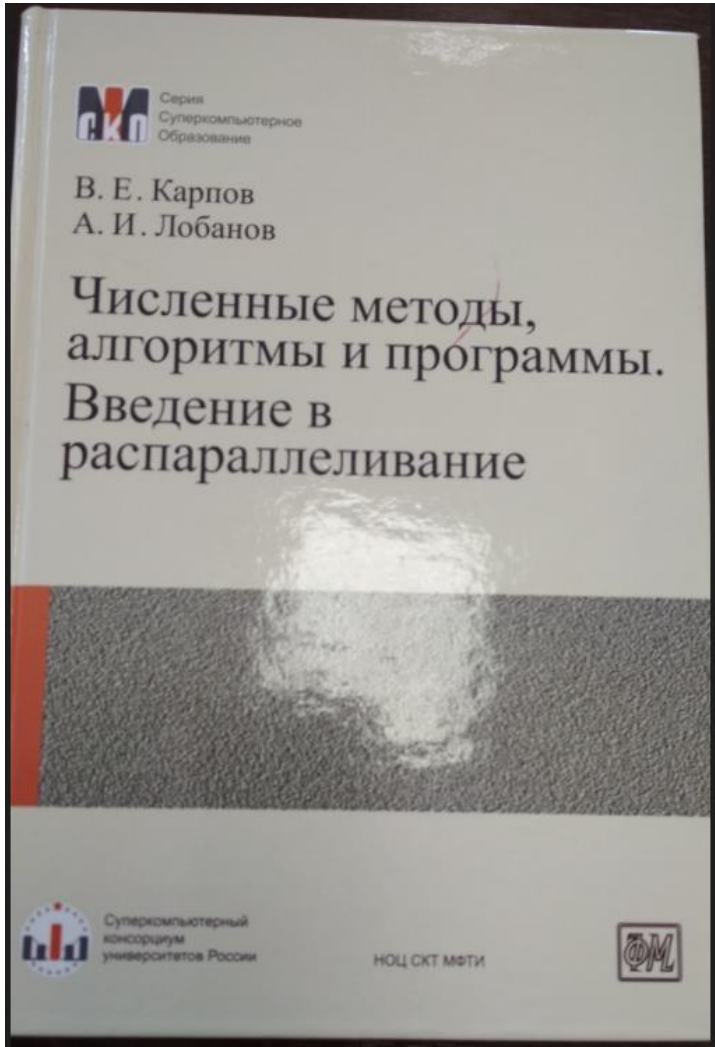
Декабрь (2) – Долги



Четверг	9^{00} - 10^{25}	Квант.мех. 507а ГК	Импульсные цифр.устр. /доц. Псурцев В.П./430 ГК	Введ. в распарал. алг.и прогр. 802 КПМ		Импульсные цифр.устр. /доц. Псурцев В.П./430 ГК	Квант.мех. 511 ГК	Введ. в распарал. алг.и прогр. 801 КПМ		Импульсные цифр.устр./доц. Псурцев В.П./430 ГК
	10^{45} - 12^{10}	Квантовая механика/ доцент Гец А.В./ 202 НК								Ин.яз.
	12^{20} - 13^{45}			Квант.мех. 526 ГК		Квант.мех. 516а ГК			Введ. в распарал. алг.и прогр. 804 КПМ	
	13^{55} - 15^{20}	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.
	15^{30} - 16^{55}	Военная подготовка								
	17^{05} - 18^{30}									
	18^{35} - 20^{00}									
Пятница	9^{00} - 10^{25}									Лаборатория ИЦУ
	10^{45} - 12^{10}				Введ. в распарал. алг.и прогр. 801 КПМ			Квант.мех. 507а ГК	Квант.мех. 525 ГК	
	12^{20} - 13^{45}	Введ. в распарал. алг.и программ /доц. Карпов В.Е./ 123 ГК	Квант.мех. 514 ГК	Введ. в распарал. алг.и прогр./ доцент Карпов В.Е./ 123 ГК			Введ. в распарал. алг.и программ/ доцент Карпов В.Е./ 123 ГК			
	13^{55} - 15^{20}	Машинное обучение (для потока №1)/ проф. Воронцов К.В., проф. Стрижов В.В./ 239 НК		Машинное обучение(для потока № 1)/ проф. Воронцов К.В., проф. Стрижов В.В./ 239 НК			Машинное обучение (для потока №1)/ проф. Воронцов К.В., проф. Стрижов В.В./ 239 НК			
	15^{30} - 16^{55}	Введ. в распарал. алг.и прогр. 801 КПМ	Лаборатория ИЦУ		Квант.мех. 509 ГК	Лаборатория ИЦУ				
	17^{05} - 18^{30}									
	18^{35} - 20^{00}	Машинное обучение (для потока №1)/ проф. Воронцов К.В., проф. Стрижов В.В./ 239 НК		Машинное обучение (для потока №1)/ проф. Воронцов К.В., проф. Стрижов В.В./ 239 НК			Машинное обучение (для потока №1)/ проф. Воронцов К.В., проф. Стрижов В.В./ 239 НК			
Суббота	9^{00} - 10^{25}	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Ин.яз.	Правоведение/ 532 ГК
	10^{45} - 12^{10}	Защита информации/ ст. преп. Колыбельников А.И./ 239 НК								
	12^{20} - 13^{45}	Защита информ. 202 РТК			Защита информ. 202 РТК			Защита информ. 202 РТК	Защита информ. 202 РТК	Защита информ. 202 РТК
	13^{55} - 15^{20}		Защита информ. 202 РТК	Защита информ. 202 РТК		Защита информ. 202 РТК	Защита информ. 202 РТК			
	15^{30} - 16^{55}	Теория информации/ 239 НК								
	17^{05} - 18^{30}						Введ. в распарал. алг.и прогр. 319 ЛК			

Программа 2 семестра

Книга: Карпов, Лобанов



Программы:

- Обязательные задачи (9)

  2 по каждой теме для зачёта (6)
- Бонусные задачи (3)

Оценка за семестр:

+3: Лекционная контрольная

+3: Посещения/5

+3: Мгновенные обязательные задачи/3

+3: Бонусные задачи

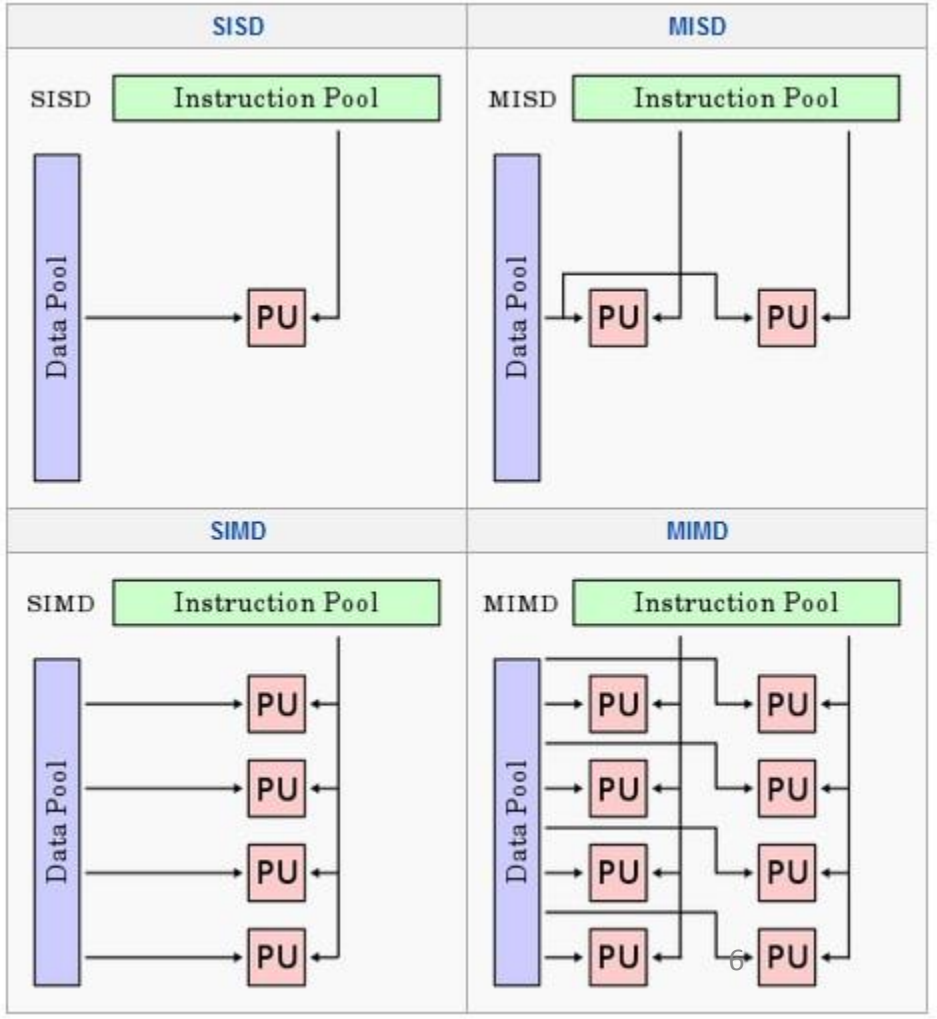
Введение

- Конвейеризация вычислений (1970-е) – микроуровневый параллелизм
- Дублирование вычислителей (1980-е) – параллелизм уровня команд (векторизация, VLIW)
- Дублирование “конвейеров” (2000-е)- параллелизм уровня потоков/заданий
- Отказ от записи промежуточных результатов в память (2010-е)

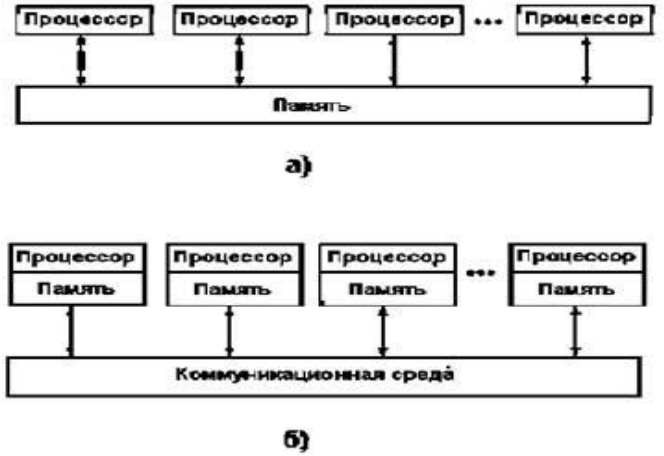
Введение

Таксономия (Классификация) Флинна (1966)

- SISD: компьютер фон-Неймановской архитектуры
- SIMD: векторные процессоры (MMX, SSE), матричные процессоры и процессоры с архитектурой VLIW.
- MISD: не используется
- MIMD:
  - Общая память - Symmetric Multiprocessor SMP
    Разделенная память - Massively Parallel Processing MPP -> Кластерные системы



Введение



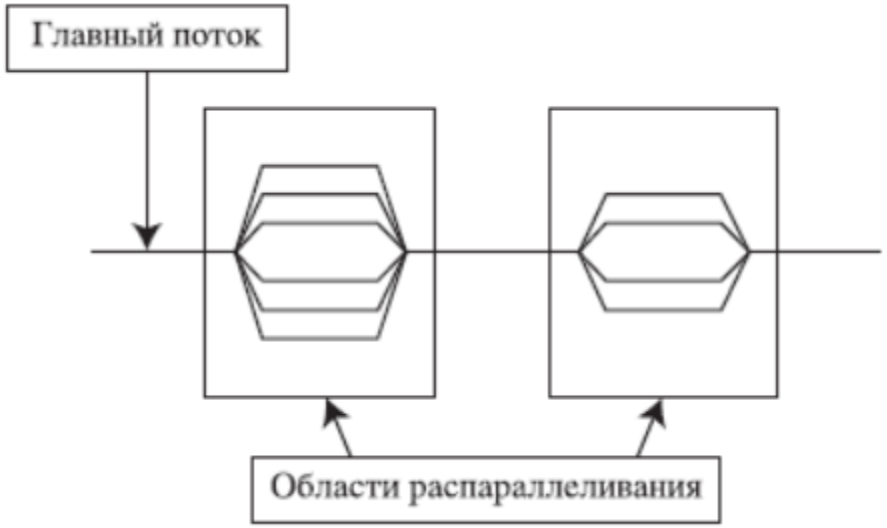
Симметричное мультипроцессирование

1. Несколько однородных процессоров и массив общей памяти
2. Когерентность кэшей, урегулирование доступа к памяти
3. Ограниченная масштабируемость
4. Работает под единой ОС
5. Модель программирования: Потоки (pthread, OpenMP)

Массивно-параллельные системы

1. Вычислительные узлы и коммуникационная среда
2. Закрытая локальная память
3. Масштабируемость ~не ограниченная
4. Полная ОС на управляющей машине, на узлах урезанная версия
5. Модель программирования: Модель передачи сообщений (“fork”, MPI, PVM, BSPlib)

OpenMP



Open Multi-Processing

Директивы компилятора, библиотечные процедуры и переменные окружения

Системы с общей памятью

Языки: Си, Си++ и Фортран

#pragma omp конструкция [предложение [предложение] …]

OpenMP

```
#include <omp.h> // Open Multi-Processing


int main(int argc, char **argv) {
	omp_set_num_threads(N); // установить число потоков в N
#pragma omp parallel // директива компилятору
{
	// параллельное исполнение
}
	return 0x00;
}
```

OpenMP

Компиляция:

```
gcc my_openMP_prog.c -o my_openMP_prog -fopenmp
```

OpenMP

```
#pragma omp parallel
{
    // parallel указывает, что данный блок кода должен быть исполнен
    // параллельно в несколько потоков
}

#pragma omp parallel // сокращенная запись
    … // блок кода, исполняющийся параллельно
```

OpenMP

```
#pragma omp parallel
{
    #pragma omp for
    for (int  i = 0; i < K; i++) // параллельное суммирование чисел 0..K
        res += i; // с теоретическим ускорением N, где N - число потоков
}

#pragma omp parallel for // сокращенная запись
    … // цикл for, исполняющийся параллельно
```

OpenMP

```
int i;

#pragma omp parallel for private(i)
        for (i = 0; i < K; i++) // параллельная печать чисел 0..K
                printf("i: %i\n", i); // с теоретическим ускорением N, где N - число потоков

int i;

#pragma omp parallel for shared(K) private(i)
        for (i = 0; i < K; i++) // параллельная печать чисел 0..K
                printf("i: %i\n", i); // с теоретическим ускорением N, где N - число потоков
```

OpenMP

```
#pragma omp parallel shared(a) private(myid, x)
{
    myid = omp_get_thread_num();
    x = work(myid);
    if(x < 1.0)
        a[myid] = x;
}
#pragma omp parallel default(private) shared(a)
{
    myid = omp_get_thread_num();
    x = work(myid);
    if(x < 1.0)
        a[myid] = x;
}
```

OpenMP

```
///Устанавливает количество потоков, которое может быть запрошено для
///параллельного блока.
void omp_set_num_threads(int num_threads)

///Возвращает количество потоков в текущей команде параллельных потоков
int omp_get_num_threads()

///Возвращает максимальное количество потоков, которое может быть установлено
int omp_get_max_threads()

///Возвращает номер потока в команде (целое число от 0 до N – 1)
int omp_get_thread_num()

///Возвращает количество физических процессоров доступных программе
int omp_get_num_procs()

///Возвращает не нулевое значение, если вызвана внутри параллельного блока
///В противном случае возвращается 0
int omp_in_parallel()
```

OpenMP

- firstprivate(var1, var2, ...) private + указанные переменные инициализируются значением до входа в параллельную секцию.
- lastprivate(var1, var2, ...) Приватные переменные сохраняют свое значение, которое они получили при достижении конца параллельного участка кода.
- reduction(оператор:var1, var2, ...) гарантирует безопасное выполнение операций редукции, например, вычисление глобальной суммы.
- if(выражение) параллельное выполнение необходимо только если выражение истинно.

OpenMP

- sections / section – разделение задач между потоками
- single – при необходимости сделать действие одним потоком в параллельном участке
- ordered – в параллельных циклах говорит о исполнении в строго фиксированной последовательности

OpenMP

```
int myid;
int a = 10;
#pragma omp parallel default(private) \
 firstprivate(a)
{
    myid = omp_get_thread_num();
    printf("Thread%d: a = %d\n", myid, a);
    a = myid;
    printf("Thread%d: a = %d\n", myid, a);
}
```

OpenMP

```
#pragma omp parallel
{
    #pragma omp for private(i) lastprivate(k)
    for(i=0; i<10; i++)
        k = i*i;
}
printf("k = %d\n", k);
```

OpenMP

```
#pragma omp parallel sections ///nowait
{
    #pragma omp section
    {
        printf("T%d: task1\n", omp_get_thread_num());
    }
    #pragma omp section
    {
        printf("T%d: task1\n", omp_get_thread_num());
    }
}
```

OpenMP

```
#pragma omp parallel
{
    #pragma for shared(x) private(i) reduction(+:sum)
    for(i=0; i<10000; i++)
        sum += x[i];
}
#pragma omp parallel
{
    #pragma for shared(x) private(i) reduction(min:gsum)
    for(i=0; i<10000; i++)
        gmin = min(gmin, x[i]);
}
```

+, - , *, &, ^, |, &&, ||, min, max

OpenMP

```
#pragma omp parallel
{
    #pragma omp for if(n>2000)
    {
        for(i=0; i<n; i++)
            a[i] = work(i);
    }
}
```

OpenMP

schedule(тип [, размер блока])

static – итерации равномерно распределяются по потокам, нудным размером блока.

dynamic – работа распределяется пакетами заданного размера между потоками. При завершении текущего блока берёт следующий.

guided – dynamic + Размер блока постепенно уменьшается вплоть до указанного значения.

OpenMP

critical – критическая секция

atomic – атомарность операции

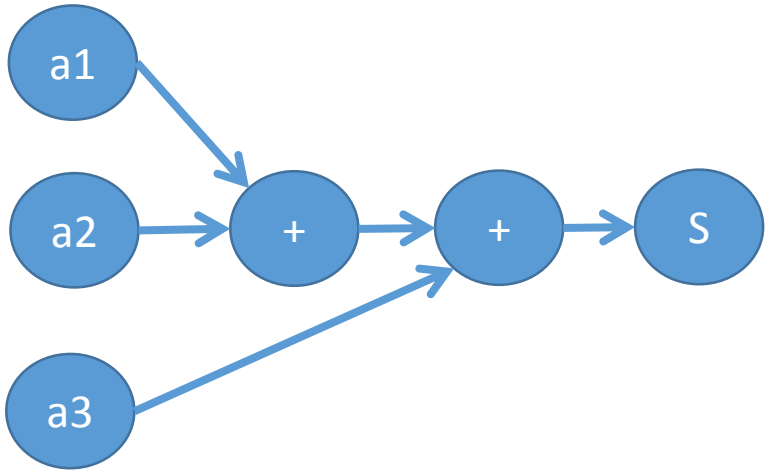
barrier – точка синхронизации

master – блок, который будет выполнен только основным потоком

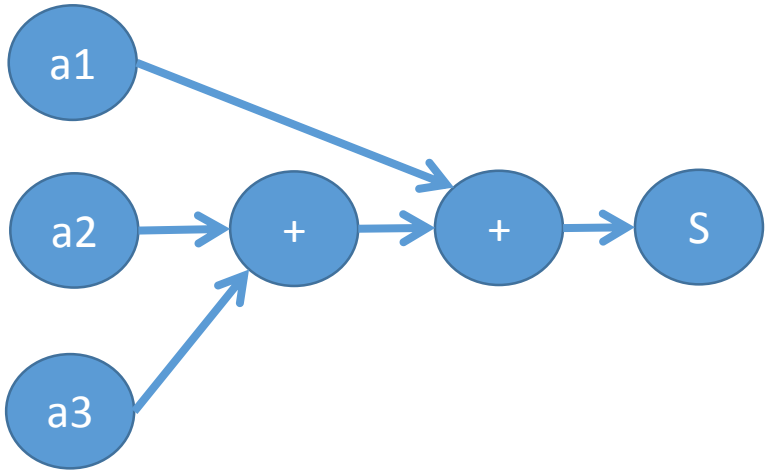
Циклы

Ярусно-параллельная форма

S = (a1 + a2) + a3

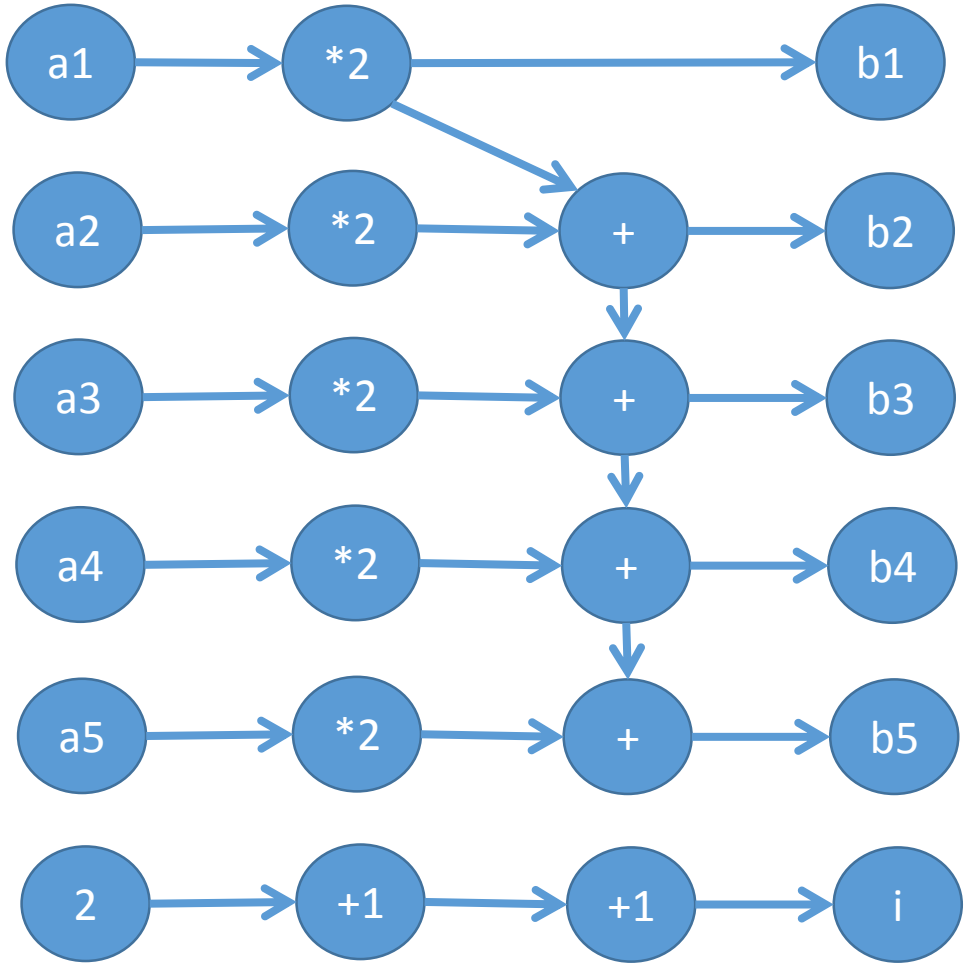


S = a1 + (a2 + a3)

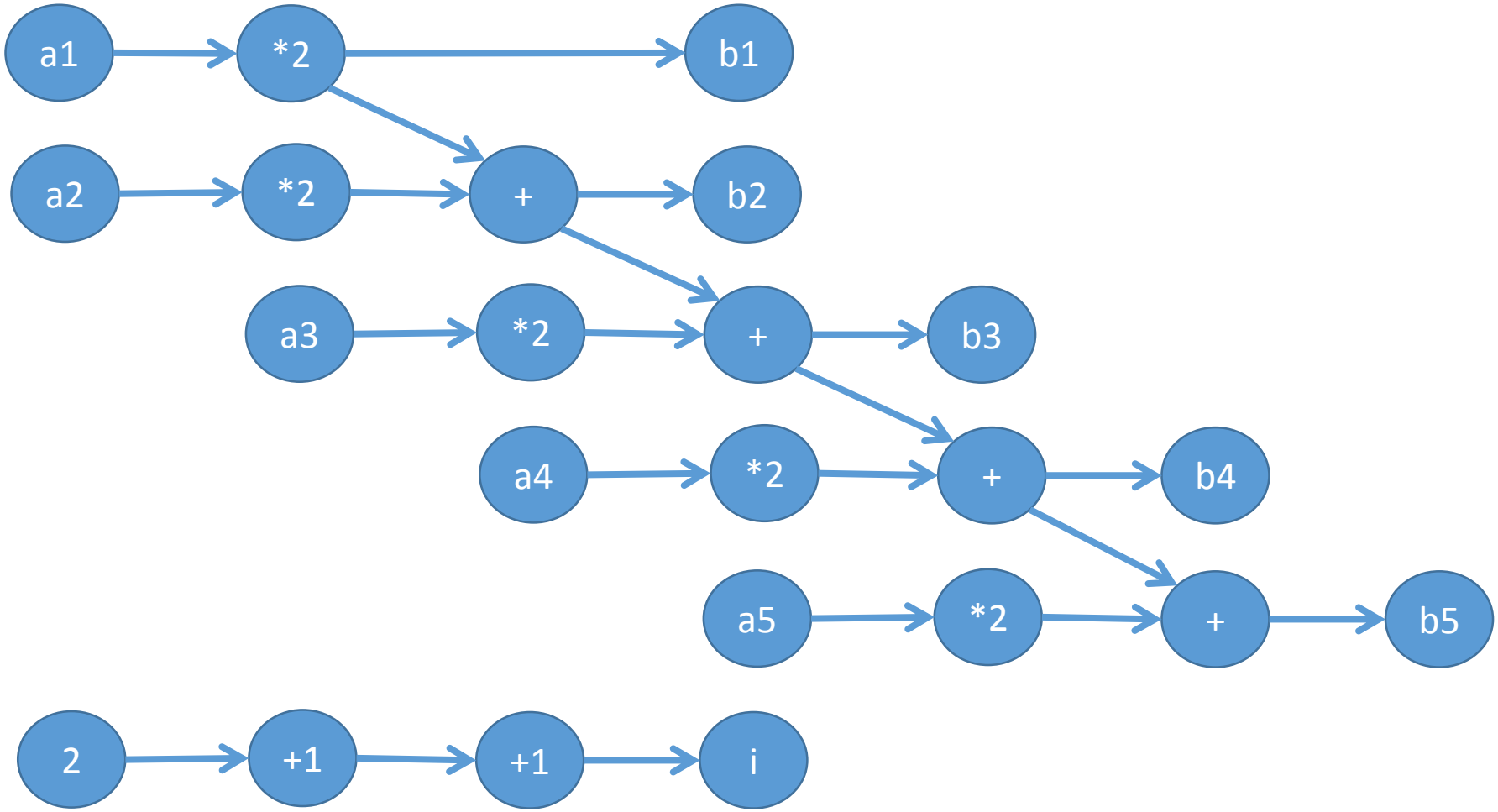


Циклы

```
b(1) = a(1)*2
do i = 2 , 5
   b(i) = b(i-1) + a(i)*2
enddo
```



Циклы



Циклы

P:

x = 2*u

y = x – 1

Q:

x = w + x

y = u*x

Входные элементы:

R(P) = {u, x}

R(Q) = {w, x, u}

Выходные элементы:

W(P) = {x, y}

W(Q) = {x, y}

Условие Бернстайна:

Пересечения:

1) W(P) и W(Q)

2) W(P) и R(Q)

3) R(P) и W(Q)

Пусты => Выполнение P и Q детерминировано.

Циклы

P:

x = 2*u

y = t – 1

Q:

z = t + u

P:

x = 2*u

y = t – 1

Q:

x = t + u

P:

x = 2*u

y = t – 1

Q:

z = t + x

P:

x = 2*u

y = t – 1

Q:

u = t + z

Циклы

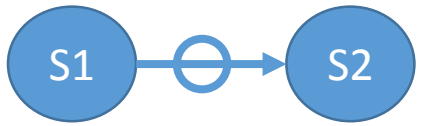
1)W(S1) и W(S2)

output dependence

S1: x = 2*y + z

S2: x = a – b

$S_1 \delta^0 S_2$



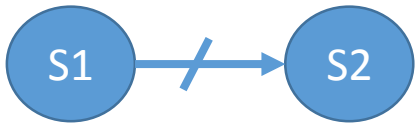
2) R(S1) и W(S2)

anti-dependence

S1: x = 2*y + z

S2: y = a – b

$S_1 \delta^{-1} S_2$



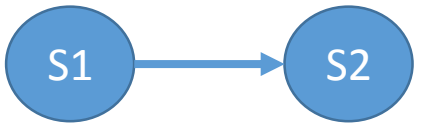
3) W(S1) и R(S2)

(true) dependence

S1: x = 2*y + z

S2: b = a – x

$S_1 \delta S_2$



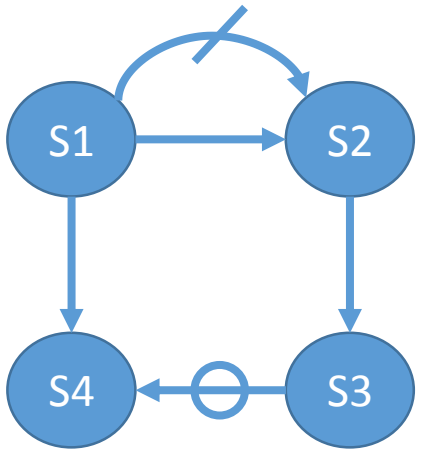
Циклы

S1: a = 2*b + 15

S2: b = a + 10*x

S3: d = b – c

S4: d = a/c



Циклы

SSA форма

x = b – c	x1 = b1 – c1
x = x + a	x2 = x1 + a1

Циклы

Зависимость по управлению

S1: x = 2*y + z

S2: y = (x > 0)? a – b : a + b

$S_1 \delta^c S_2$

Зависимость по ресурсам

S1: x = 2*y/z

S2: x = a/b

$S_1 \delta^{\mathrm{R}} S_2$

Циклы

Расстояние зависимости D

do i=1, N
S1: A[f(i)] = …. // Source (исток) = i
S2: … = … A[g(i)] … // Sink (сток) = I’

f(i) = g(i’)

D = Sink – Source

$S_1^{source} \delta S_2^{sink}$

Циклы

D < 0 – анти-зависимость

D = 0 – отсутствует зависимость

D > 0 – истинная зависимость

Можно распараллелить на D исполнителях

Циклы

Вектор расстояний **D**

```
do y=1, N
do x=1, N
S1:     A[f1(y),f2(x)] = ….          // Source (исток) = (y, x)
S2:     … = … A[g1(y),g2(x)] …       // Sink (сток) = (y’, x’)
```

(f1(y), f2(x)) = (g1(y’), g2(x’))

D = **Sink** – **Source**

Циклы

Вектор направлений

d = “=“, если D = 0 (нет зависимости)

d = “>“, если D < 0 (анти-зависимость)

d = “<“, если D > 0 (истинная зависимость)