Введение в распараллеливание алгоритмов и программ

Карпов Владимир Ефимович,
кандидат физико-математических наук, доцент

Система оценивания

Составные части

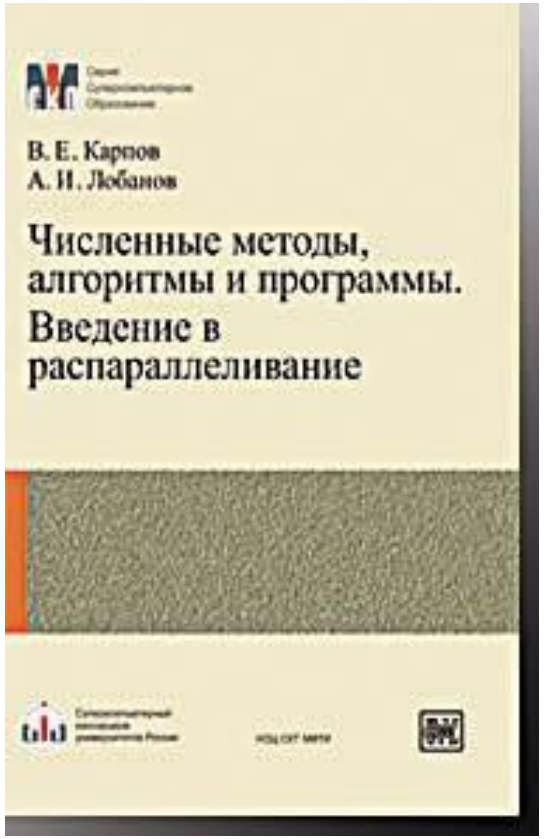
Оценка за практикум – $О_{практ}$

Оценка за теорию – $О_{теор}$

Итоговая оценка

$$О_{итог} = \frac{О_{практ} * О_{теор}}{О_{практ} + О_{теор}}$$

Литература

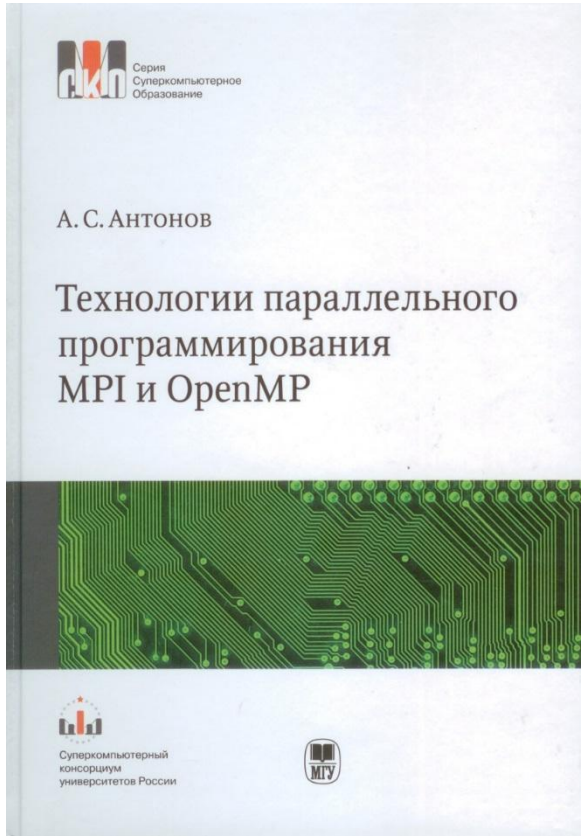


В.Е. Карпов, А.И. Лобанов

Численные методы, алгоритмы и программы. Введение в распараллеливание

М.: Физматкнига, 2014

Литература



А.С. Антонов

Технологии параллельного программирования
MPI и OpenMP

М.: Издательство МГУ, 2012

MPI
https://parallel.ru/tech/tech_dev/MPI

OpenMP
https://parallel.ru/info/parallel/openmp

Тема 1

Последовательная и параллельная парадигмы программирования

‘Кризис software’

“To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.”

E.Dijkstra, 1972 Turning Award Lecture

Первый кризис software

Временные рамки: конец 50х – 70е

- Проблема: низкоуровневое программирование.
  Компьютеры стали способны обрабатывать более сложные задачи.

- Потребность: Переход к более высокому уровню абстракции и обеспечение переносимости без потери производительности.

- Решение: языки программирования высокого уровня для фон-Неймановской архитектуры

Второй кризис software

Временные рамки: 80е – 90е

- Проблема: Создание и поддержка сложных и надежных программ, содержащих миллионы строк кода и написанных сотнями программистов

  Компьютеры стали способны обрабатывать более сложные задачи.

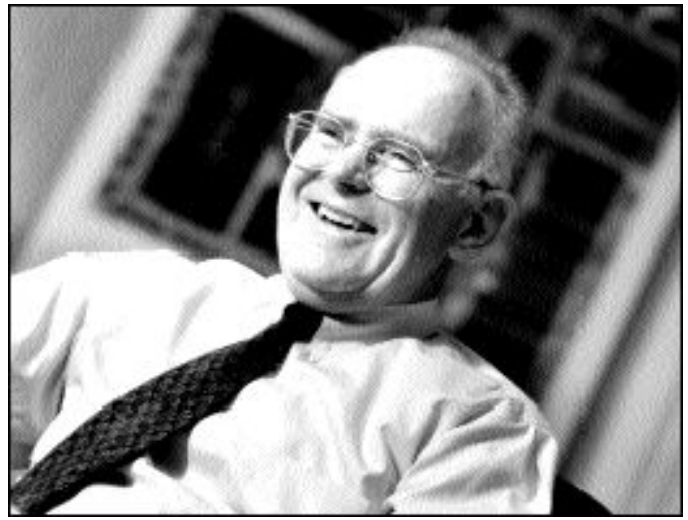
- Потребность:новые инструменты

- Решение: объектно-ориентированные языки и средства поддержки разработки больших проектов.

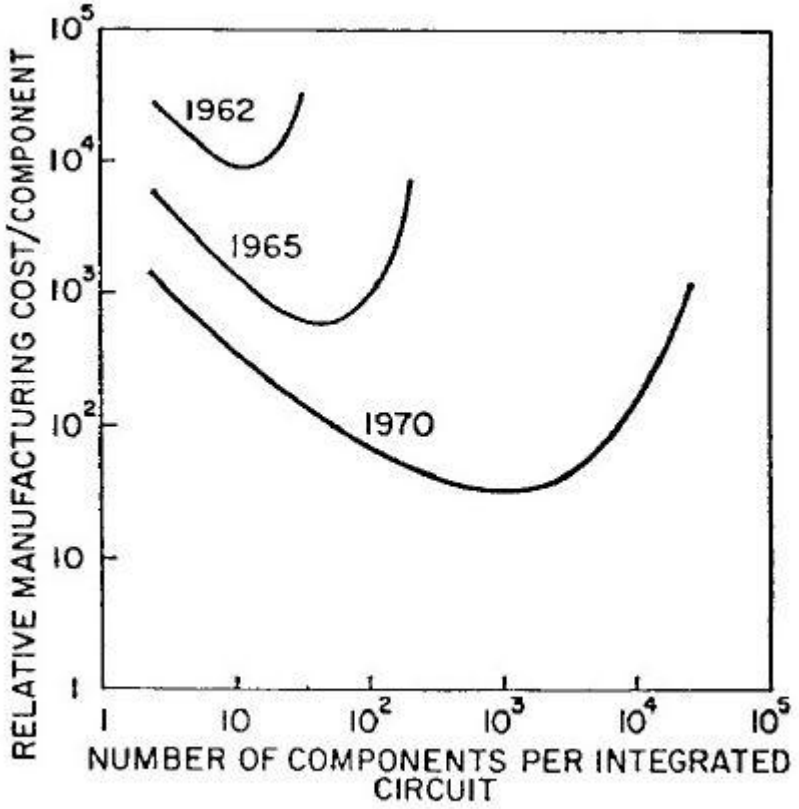
Закон Мура

Количество транзисторов
на кристалле и производительность
процессоров удваиваются
каждые полтора - два года.

Гордон Мур (Gordon Moore), 1965 –
один из будущих основателей Intel.



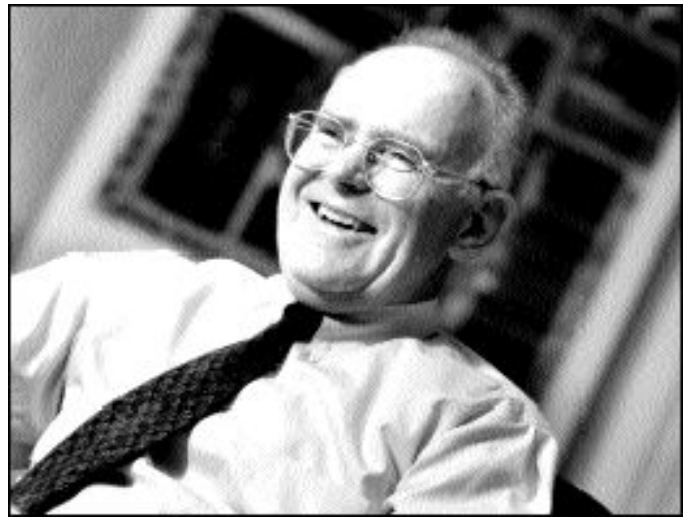
Закон Мура



Зависимость себестоимости производства одного компонента в интегральной схеме от количества компонентов

Закон Мура

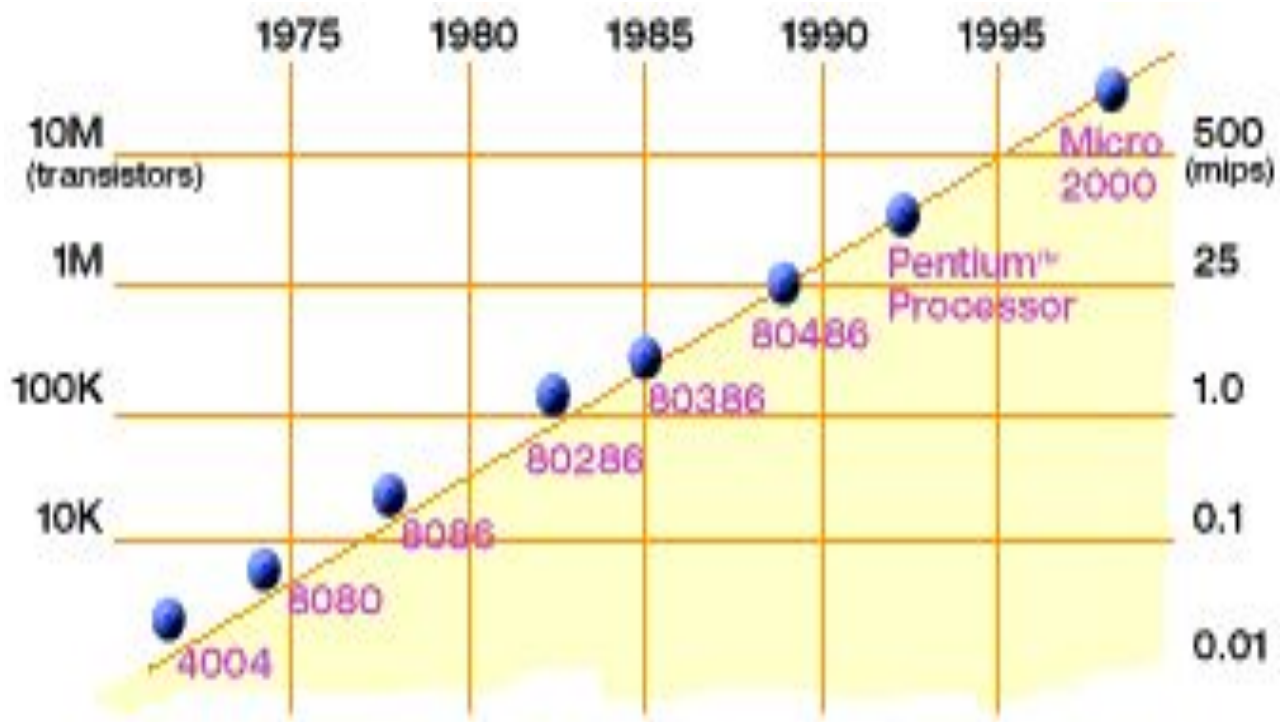
Количество транзисторов
на кристалле и производительность
процессоров удваиваются
каждые полтора - два года.



Гордон Мур (Gordon Moore), 1965 –
один из будущих основателей Intel.

Оптимальное по стоимости на один элемент количество электронных элементов для размещения на едином кристалле будет удваиваться каждые полтора - два года.

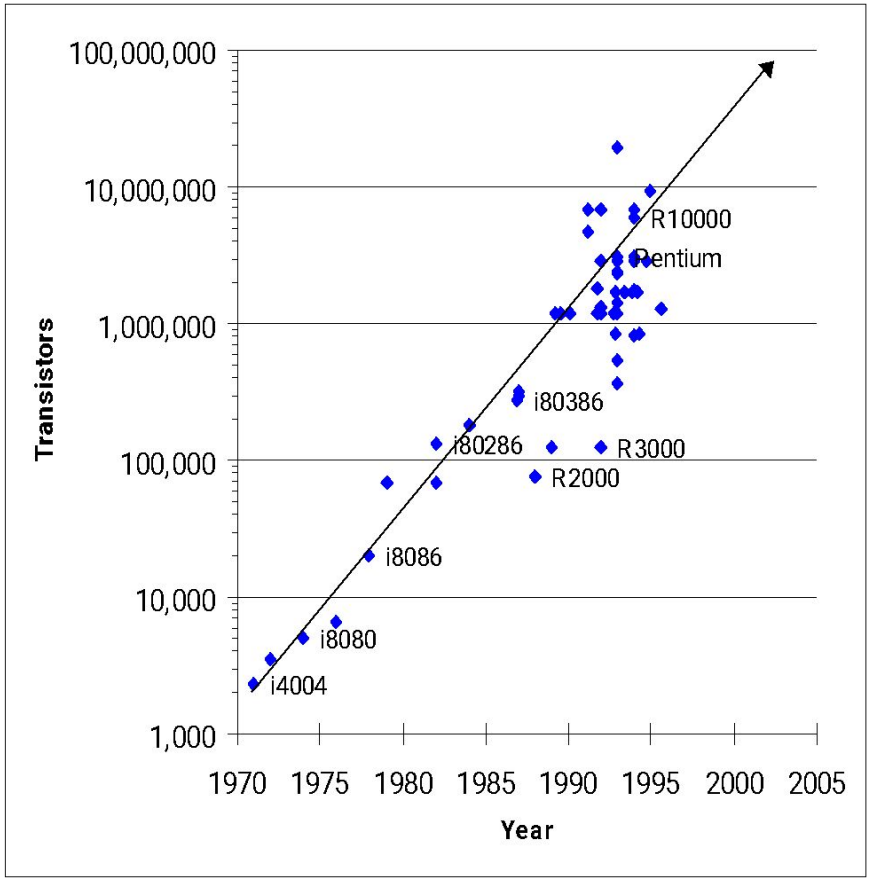
Закон Мура



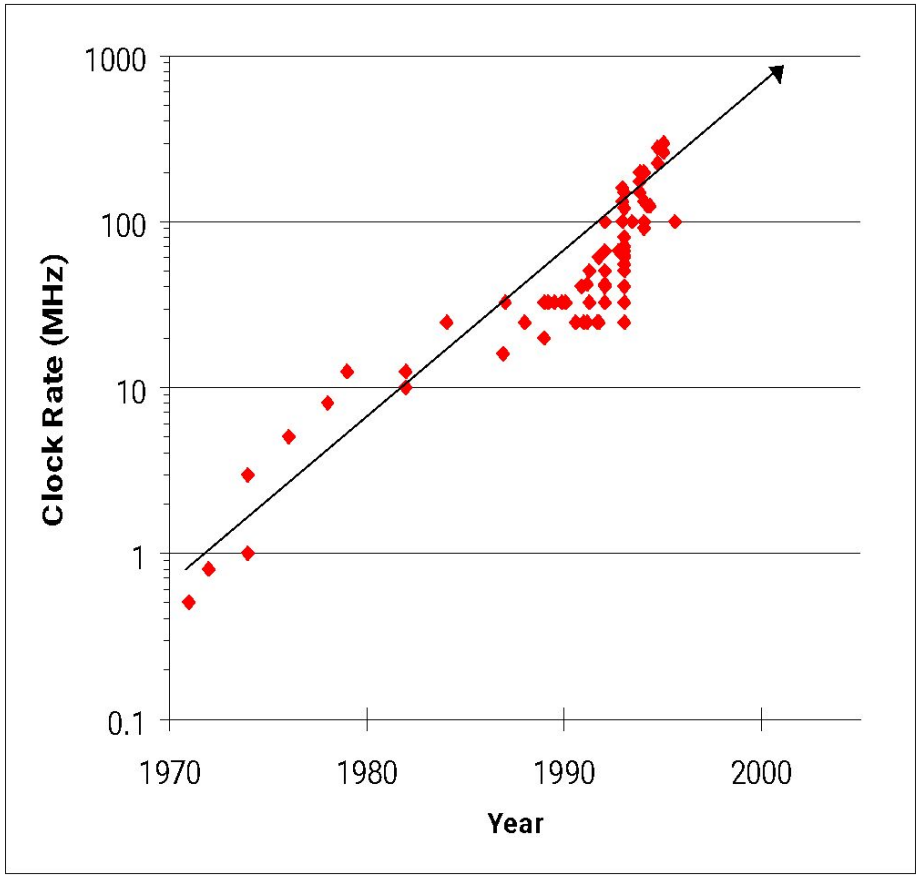
Из лекций Katherine Yelick, yelick@cs.berkeley.edu

Закон Мура

Число транзисторов на кристалле



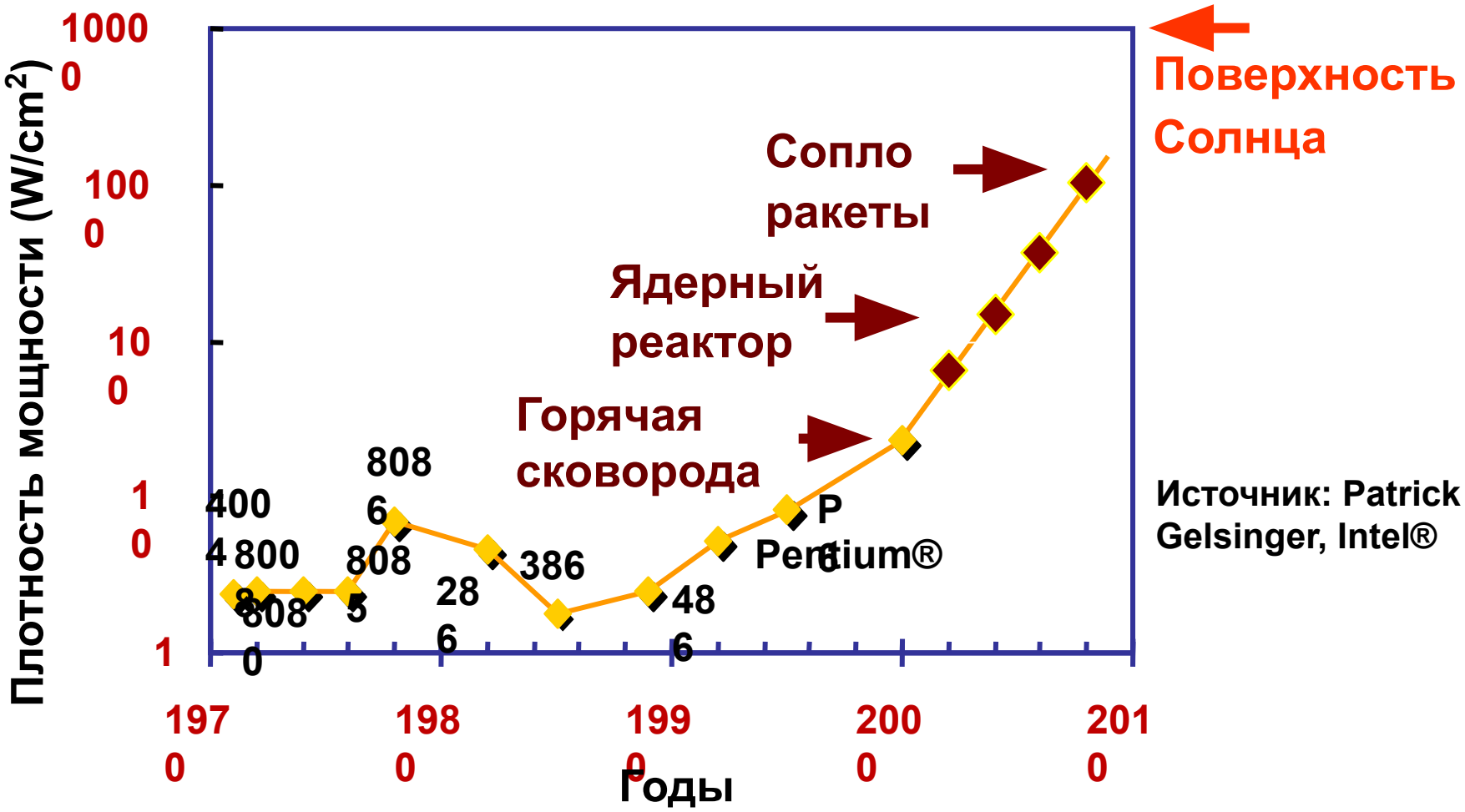
Тактовая частота



Из лекций Katherine Yelick, yelick@cs.berkeley.edu

Проблемы hardware

Теплоотвод



Источник: Patrick Gelsinger, Intel®

Проблемы hardware

EDSAC - 1949 год Кембридж - время такта 2 микросекунды ($2*10^{-6}$ секунды), выполнял в среднем 100 арифметических операций в секунду

Процессоры 2004 года – тактовая частота 3GHz – время одного такта ~ $3*10^{-10}$ секунды – но производительность по сравнению с EDSAC существенно больше, чем в 10000 раз

История параллельности

- Разрядно-параллельная память и разрядно-параллельная арифметика (1953-55)
- Spooling. Независимые устройства ввода-вывода (1958)
- Ускорение доступа к памяти за счет разделения ее на банки памяти (1961)
- Конвейеризация выполнения команд (1962)
- Независимые функциональные устройства (1964)
- Конвейеризация функциональных устройств (1969)

История параллельности

Практически исчерпаны резервы параллелизма, спрятанные от программистов внутри hardware.

История параллельности

- Матричные процессоры (1967-71).
- Векторно-конвейерные ЭВМ (1976).
- Многопроцессорные вычислительные комплексы.
- Кластеры.
- Высокопроизводительные системы на графических процессорах.

Новый закон Мура

Количество ядер на одном процессоре удваивается каждые два-три года.

Multicore processors – (2 – 10x ядер)

Manycore processors – (100 – 100x ядер)

Myriacore processors – (1000 - ? ядер)

Третий кризис software

Временные рамки: 2005 – ???

- Проблема: противоречие между последовательной парадигмой программирования и наличием нескольких исполнителей (ядра, процессоры) – в устройствах.

  Компьютеры стали способны обрабатывать более сложные задачи.

- Потребность: смена парадигмы программирования

- Решение: до конца неизвестно

Что такое парадигма?

В философии науки:
Термин введен Томасом Куном (Thomas S. Kuhn) в 1962 г. (“Структура научных революций”).

Парадигма – совокупность научных идей и взглядов, в рамках которой определенной группой ученых ведутся научные исследования.

Вся история науки – это чередование этапов развития в рамках парадигм и научных революций.

Что такое парадигма?

В computer science:

Термин впервые употреблен Робертом Флойдом (Robert W Floyd) в 1978 г.

(“Парадигмы программирования” – Тьюринговская лекция).

До сих пор не существует общепринятого определения термина «парадигма программирования».

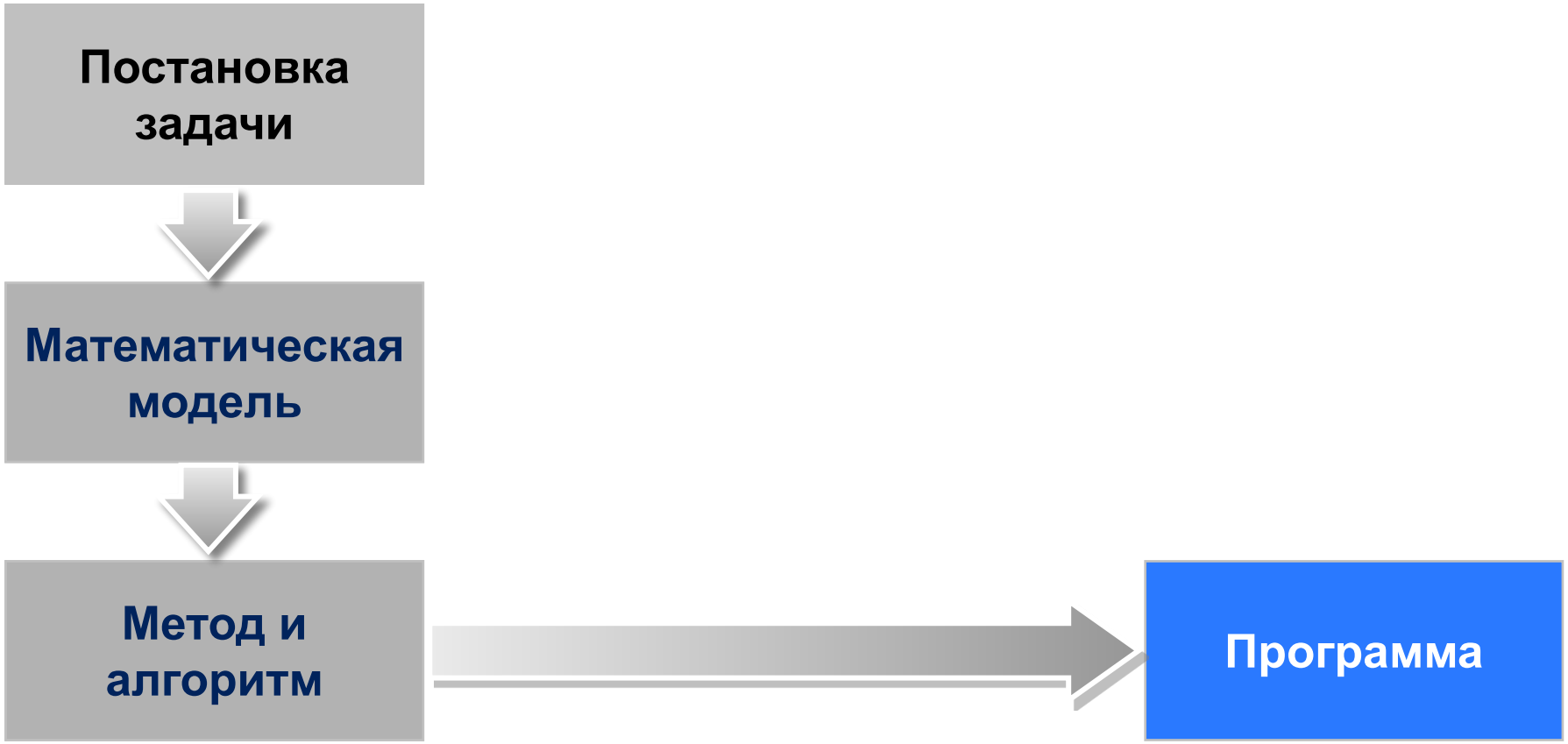
Это понятие часто смешивают с понятиями «стиль программирования», «модель программирования» и т.д. .

Что такое парадигма?

В нашем курсе:

Парадигма программирования – полная совокупность этапов работы необходимых для решения задачи математического моделирования – от постановки задачи до результатов компьютерных вычислений
(Владимир Карпов, сегодня).

Парадигма последовательного программирования



Модель программирования

- Модель программмирования
  - определяет основные идеи и стиль программной реализации
  - абстрагируется от hardware (по возможности) и от конкретного языка программирования

- Названия моделей программирования до конца в литературе не устоялись.

Примеры моделей последовательного программирования

- Модель императивного программирования
  - Основывается на понятиях переменной и присваивания.
  - Вся программа строится как единая последовательность действий с условными и безусловными переходами.

Примеры моделей последовательного программирования

- Модель процедурного программирования
  - Вся программа построена как набор процедур (подпрограмм или функций).
  - Каждая процедура содержит императивные части и, возможно, вызов других процедур.

Примеры моделей последовательного программирования

- Модель функционального программирования
  - Программа представляется в виде набора взаимосвязанных математических функций.
  - Значение функции зависит только от явно заданных значений ее аргументов.
  - Не предписывает конкретного порядка вычислений

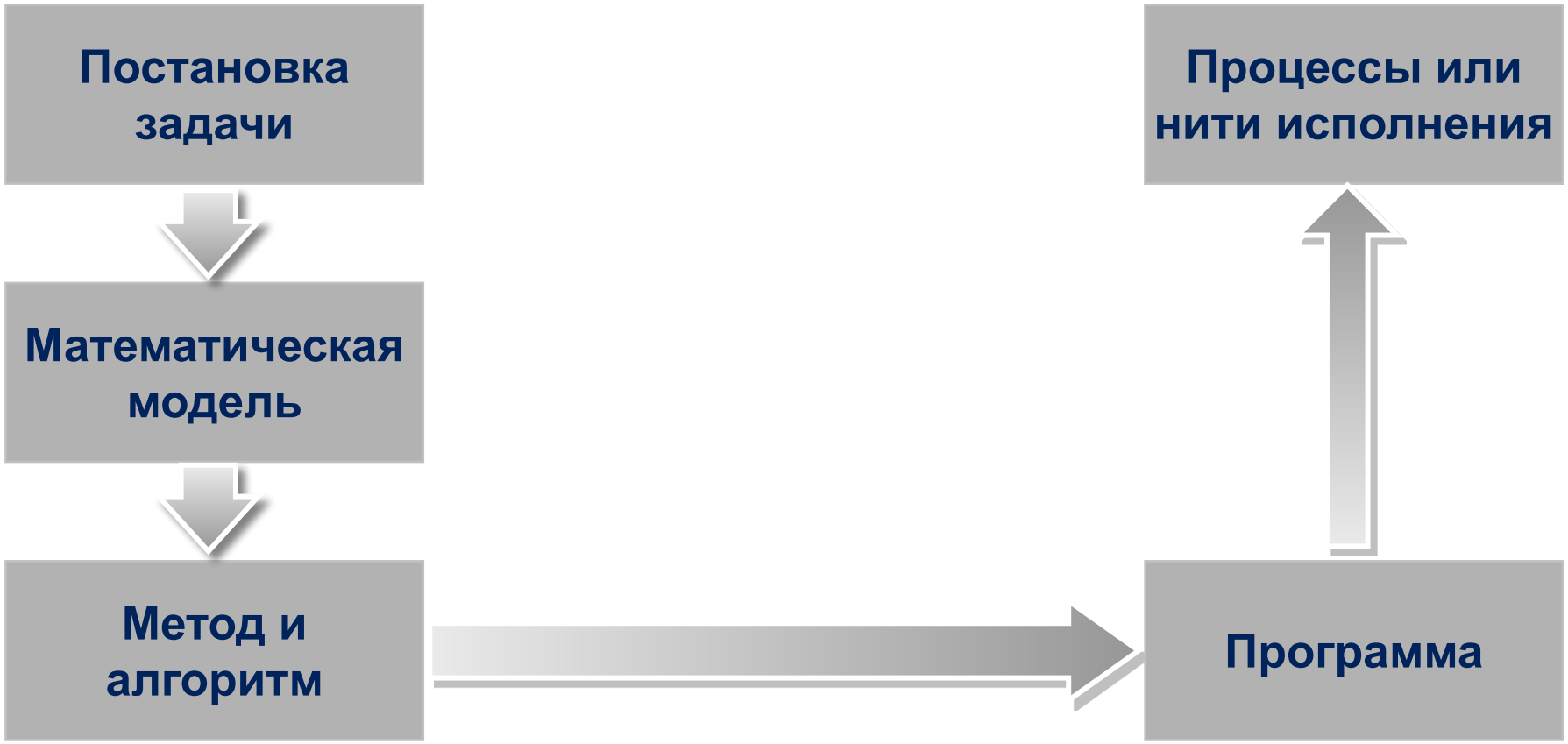
Примеры моделей последовательного программирования

- Модель логического программирования
  - Основана на математической логике первого порядка.
  - Все известные факты записываются в виде предикатов.
  - Для доказательства истинности или ложности других предикатов используются правила вывода.

Примеры моделей последовательного программирования

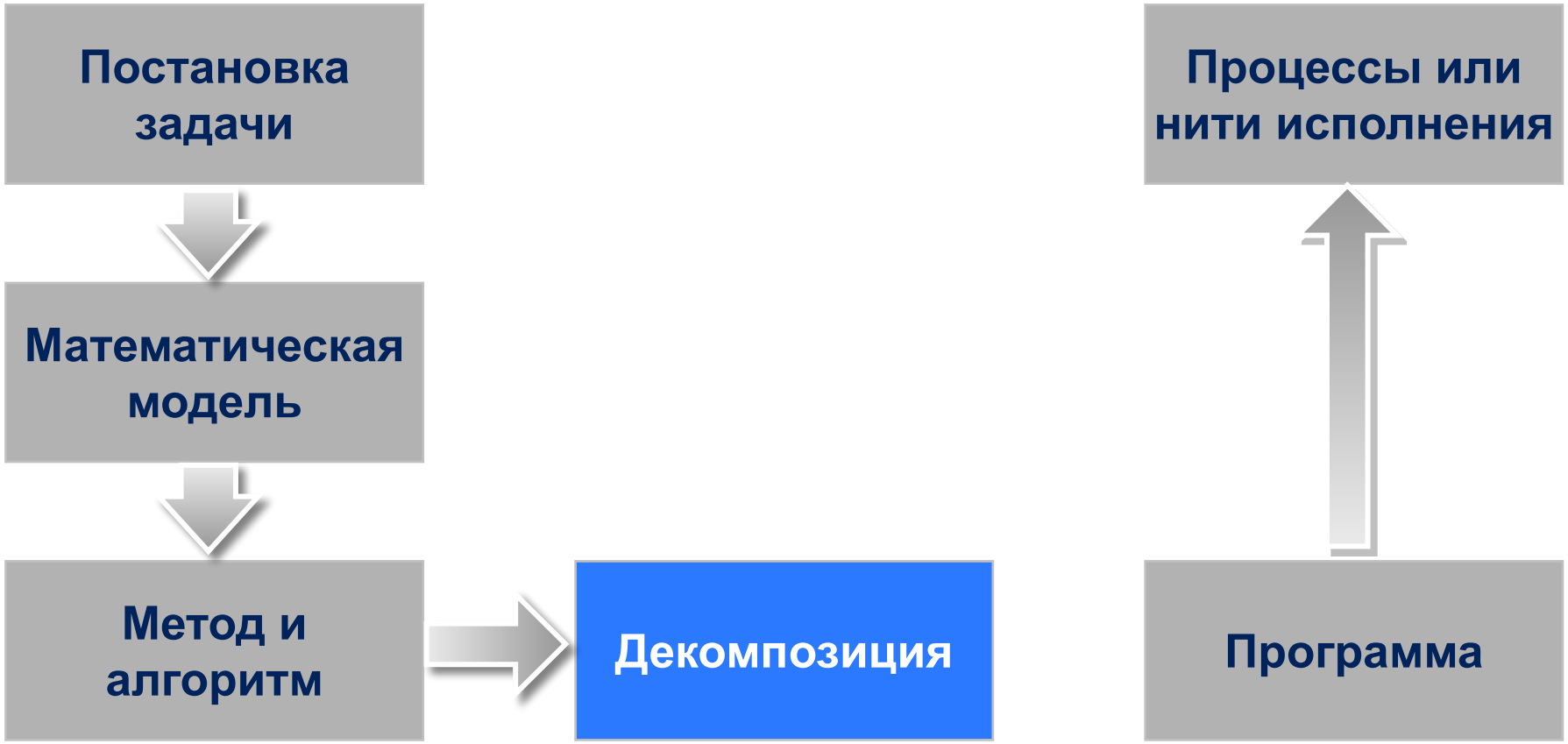
- Модель объектно-ориентированного программирования
  - Программа – совокупность взаимодействующих объектов.
  - Каждый объект – экземпляр некоторого класса в иерархической структуре классов.
  - Каждый объект способен принимать сообщения, обрабатывать данные и посылать сообщения другим объектам.

Парадигма последовательного программирования



Парадигма параллельного программирования

Парадигма параллельного программирования



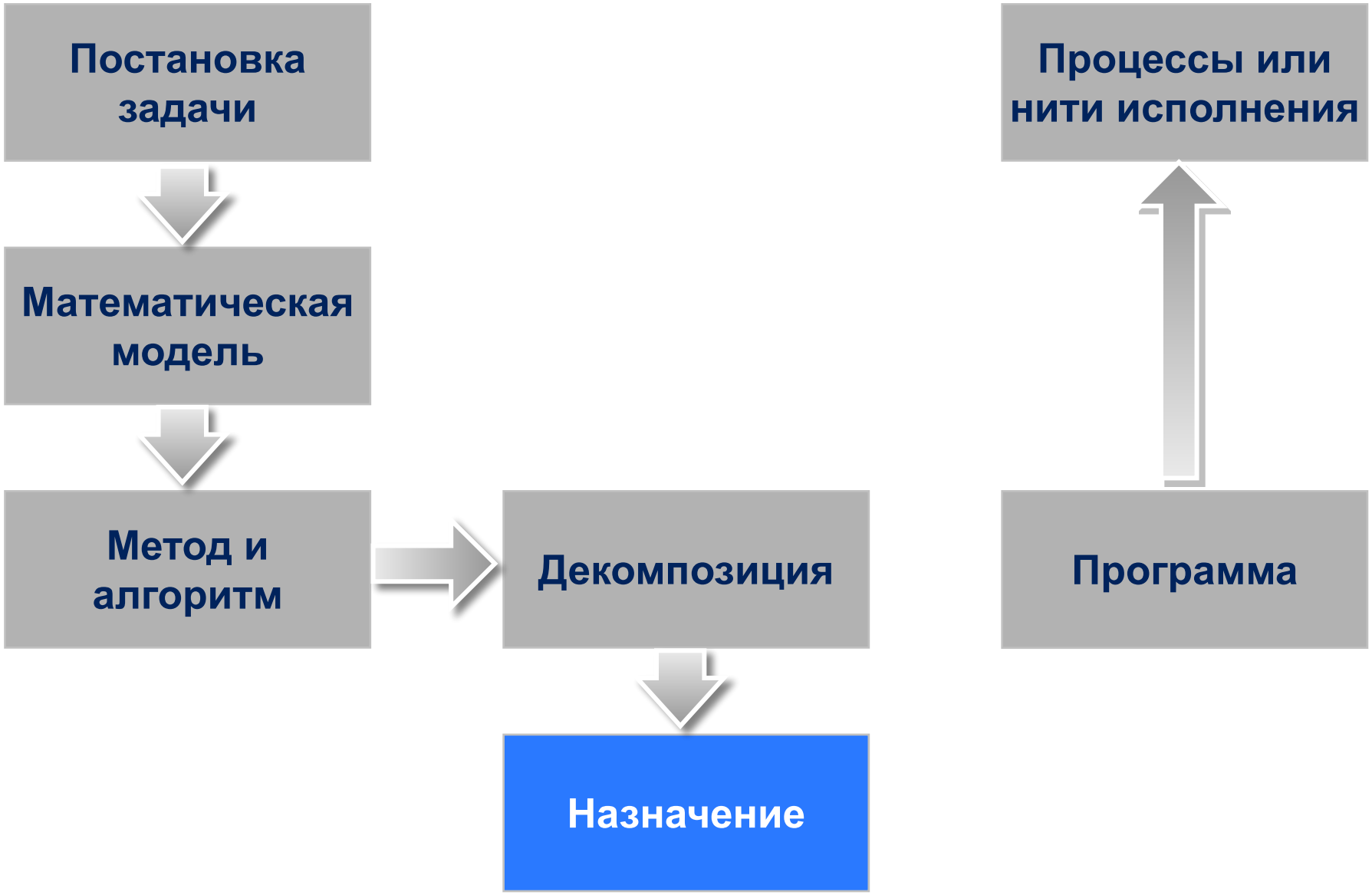
Декомпозиция

- Разделение вычислений и данных на части.

- Декомпозиция по данным
  - Разделяем данные на области ответственности.
  - Определяем, как вычисления связаны с данными.

- Декомпозиция по вычислениям
  - Разделяем вычисления на области ответственности.
  - Определяем, как данные связаны с вычислениями.

Парадигма параллельного программирования



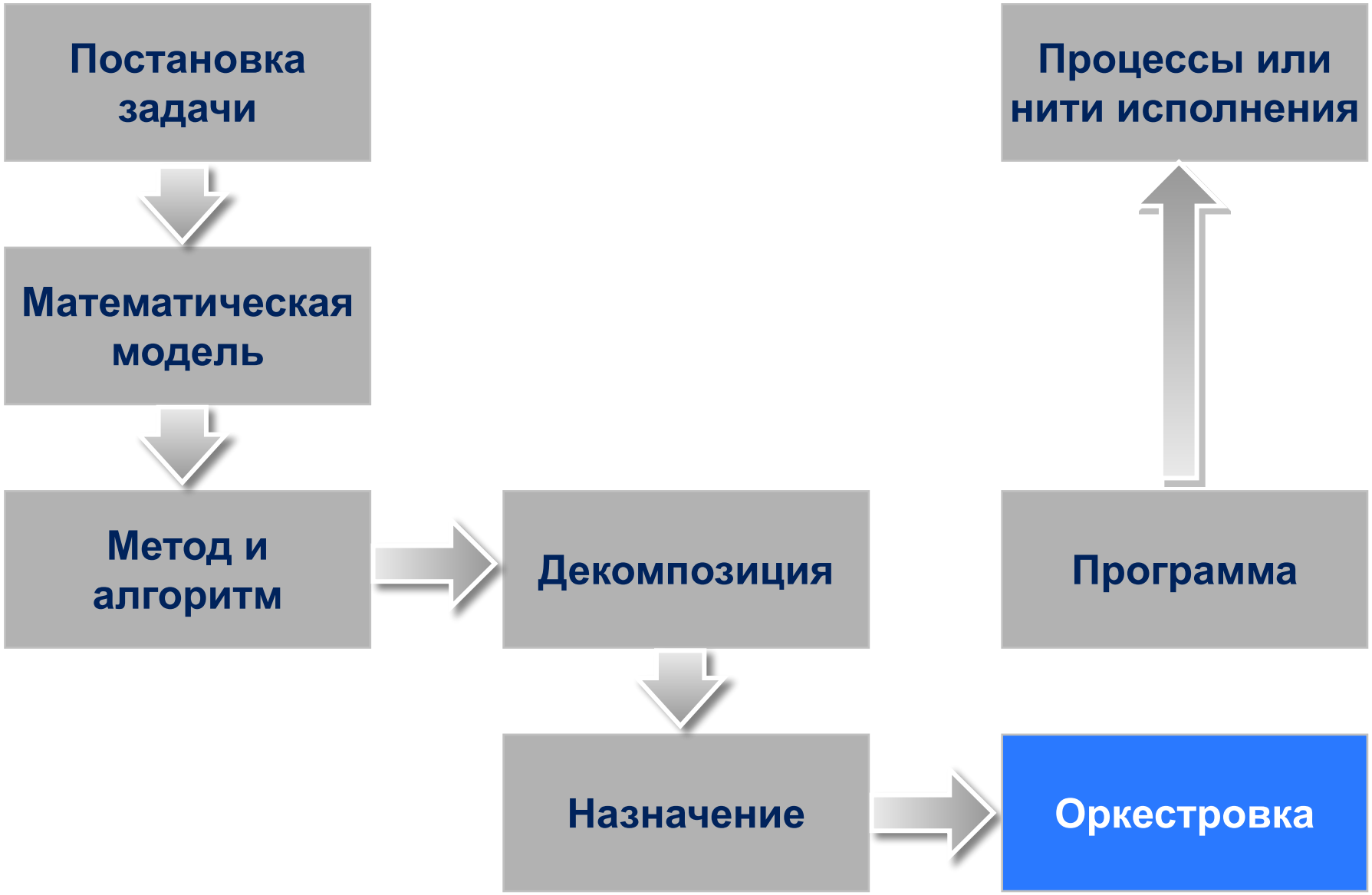
Назначение

- Определение способов распределения задач между виртуальными исполнителями.

- Цели:
  - балансировка загрузки исполнителей
  - уменьшение обменов данными
  - уменьшение накладных расходов

- Время назначения
  - статическое – до реального начала исполнения
  - динамическое – во время исполнения

Парадигма параллельного программирования



Оркестровка

- Выбор модели параллельного программирования.

- Определяет:
  - доступ к данным
  - обмен данными
  - синхронизацию обмена

- Основные вопросы:
  - Как организовать структуры данных?
  - Как улучшить локальность?
  - Какими порциями обмениваться данными?

Модели параллельного программирования

- Модель передачи сообщений:

Приложение состоит из набора процессов с различными адресными пространствами.

Процессы обмениваются сообщениями с помощью явных send/receive операций.

  - Преимущество: полный контроль над исполнением
  - Недостаток: сложность программирования

Модели параллельного программирования

- Модель разделяемой памяти:

Приложение состоит из набора thread'ов с общей памятью. Thread'ы используют примитивы для синхронизации.

- Подмодели:
  - Явное использование thread'ов
    - Преимущество: полный контроль над исполнением
    - Недостаток: сложность программирования
  - Программирование на языках высокого уровня с помощью прагм
    - Преимущество: простота программирования
    - Недостаток: сложность контроля над исполнением

Модели параллельного программирования

- Модель разделенных данных:

Приложение состоит из набора процессов или thread’ов.

Каждый процесс или thread работает со своими собственными данными. Обмена информацией при работе нет.

  - Преимущество: простота программирования
  - Недостаток: слишком узкий класс задач

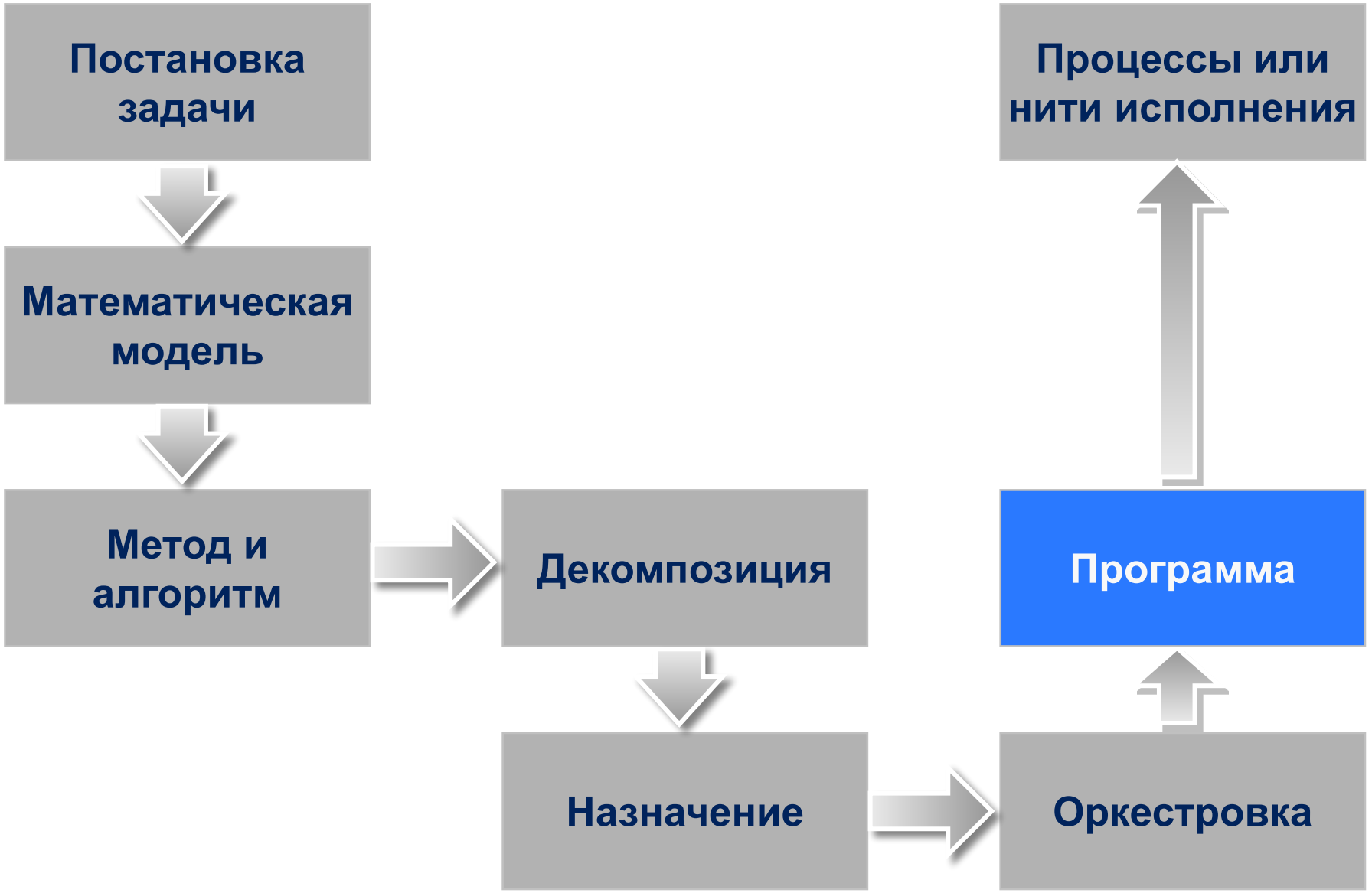
Модели параллельного программирования

- Последовательная модель:

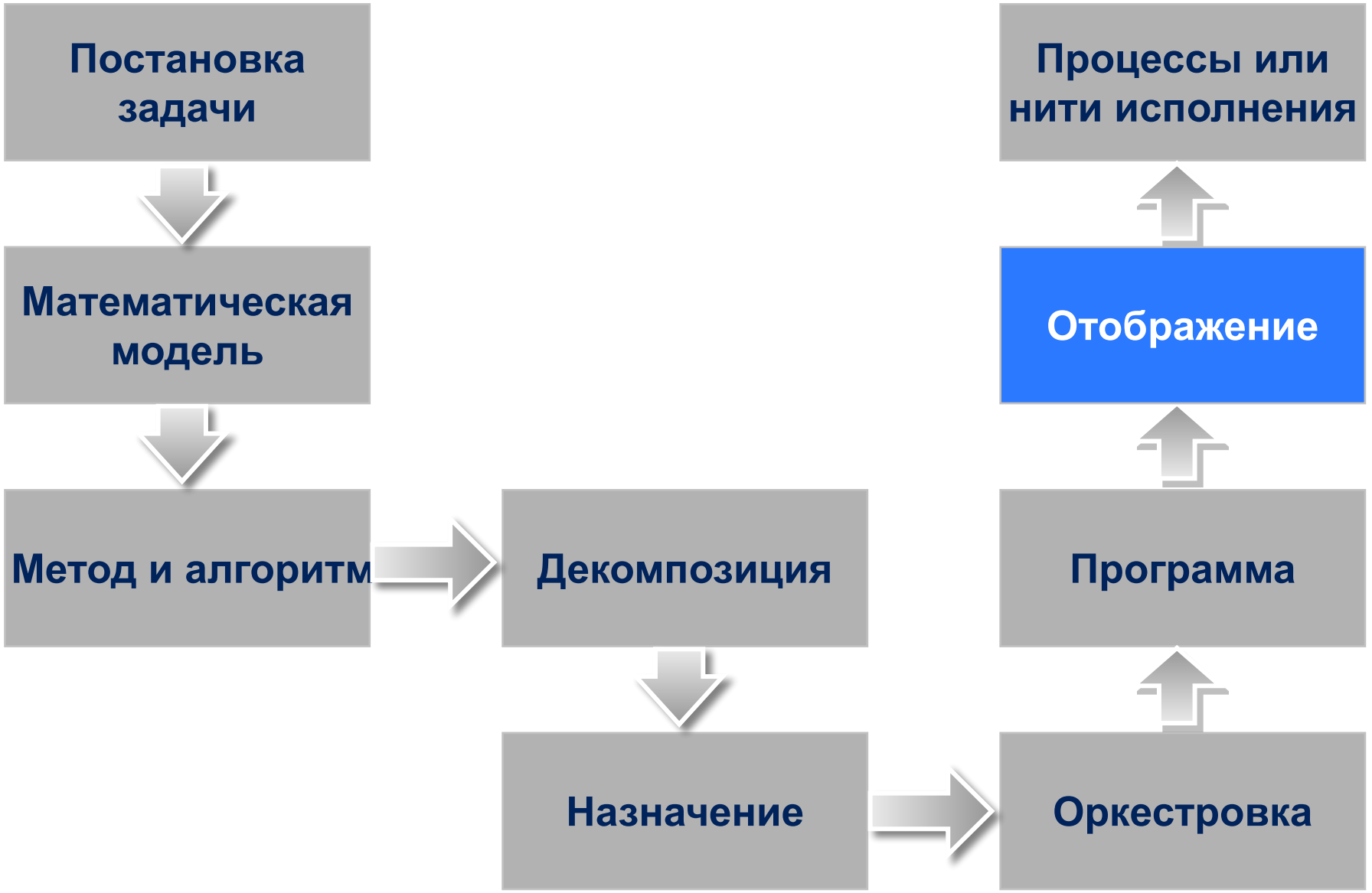
Мы создаем последовательную программу для автоматического распараллеливания.

  - Преимущество: знакомая парадигма программирования
  - Недостаток: ограниченность автоматического распараллеливания

Парадигма параллельного программирования



Парадигма параллельного программирования



Отображение

- Отображение множества виртуальных исполнителей на множество реальных исполнителей.

Производится пользователем и/или операционной системой (ОС).

Системы с общей памятью – ОС.

Системы с распределенной памятью – пользователь или ОС.

Парадигма параллельного программирования