Тема 15
ПУТИ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ ЭВМ

15.1. Распараллеливание и конвейеризация процессов

Производительность ЭВМ является одной из важнейших ее характеристик, и совершенно понятно стремление разработчиков ЭВМ к увеличению этого параметра. Существуют различные пути увеличения производительности компьютера, часть из которых уже рассматривалась или упоминалась выше.

1. Совершенствование технологии изготовления компонентов компьютера, позволяющее повысить тактовую частоту процессора и магистралей.

2. Увеличение разрядности процессора. Очевидно, что при прочих равных условиях более многоразрядный процессор будет эффективнее производить вычислительные операции.

3. Увеличение количества программно-доступных регистров (сверхоперативной памяти) процессора, позволяющее при выполнении программы уменьшить количество обращений к оперативному запоминающему устройству ЭВМ.

4. Увеличение объема оперативной памяти ЭВМ, позволяющее в многозадачном режиме сократить частоту обмена страниц между оперативной и внешней физической памятью компьютера.

5. Применение сверхоперативной кэш-памяти, буферизация, согласование производительности устройств памяти с разным быстродействием.

Рассматривая с самых общих позиций проблему повышения производительности решения человеком каких-либо задач, решение которых состоит в выполнении набора более элементарных операций, можно отметить два фундаментальных подхода. Это — *распараллеливание* операций и *конвейеризация* операций.

Распараллеливание — это одновременное параллельное выполнение каждой из составляющих задачу элементарных операций соответствующим числом предназначенных для каждой операции операционных устройств (рис. 15.1). Простым примером служит привлечение для выполнения какой-либо работы большего количества исполнителей.

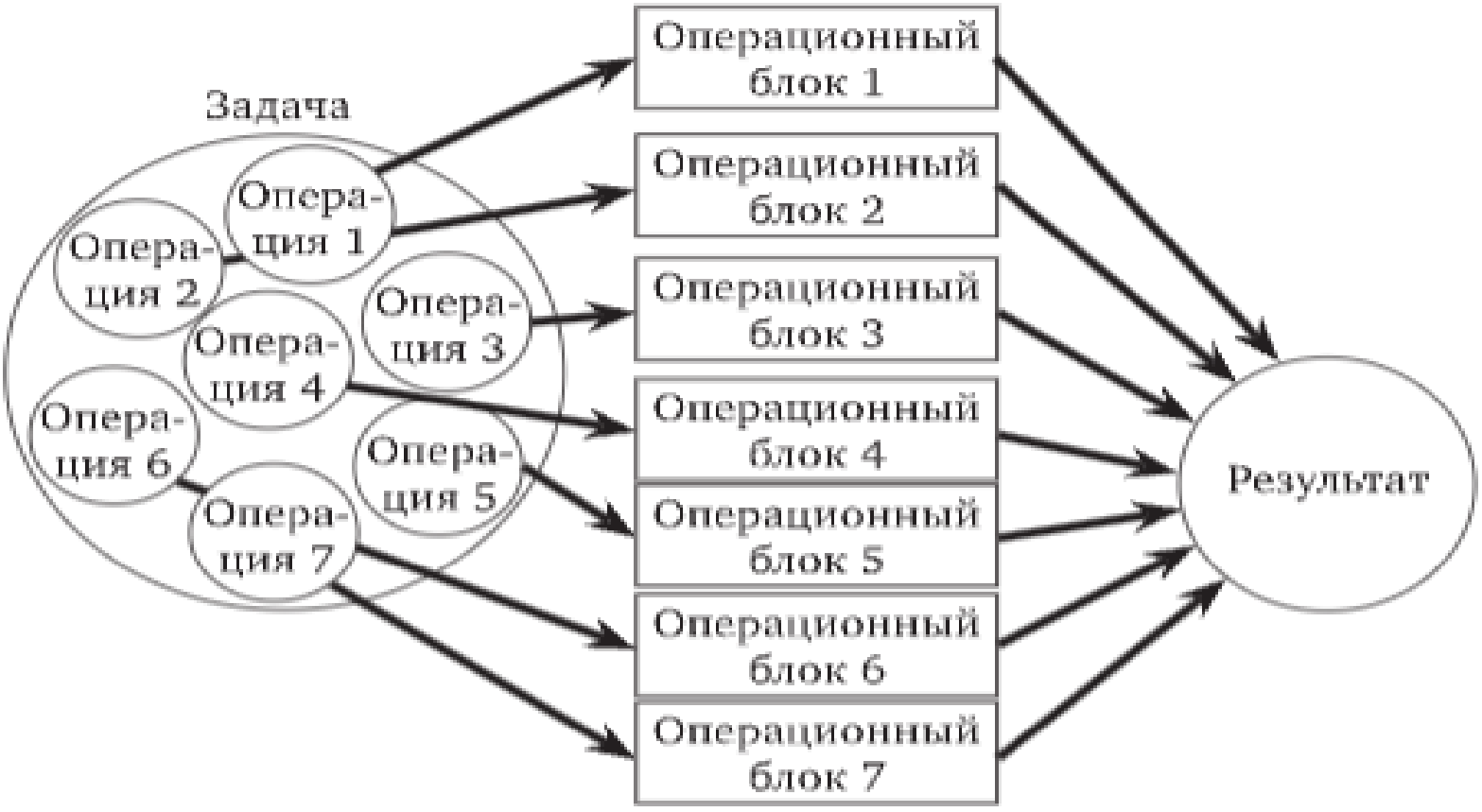

Рис. 15.1. **Параллельное выполнение операций**

То есть вместо того, чтобы выполнять последовательно каждую из *N* таких элементарных операций с помощью одного универсального операционного устройства (процессора), для каждой элементарной операции используется свое специализированное операционное устройство. И если все эти *N* операционных устройств одновременно выполняют свои частные операции, результат решения задачи может быть получен уже не за *N* шагов, а всего за один шаг.

Вполне очевидно, что эффект от использования распараллеливания операций может быть достигнут только в случае, когда составляющие задачу частные операции-подзадачи в принципе могут выполняться параллельно.

Приведенный вариант распараллеливания операций, однако, не применим в тех случаях, когда решение какой-либо задачи по обработке информационных объектов, хотя и может быть также представлено в виде выполнения над ними набора

частных операций, но операции при этом должны выполняться строго *последовательно* друг за другом вследствие того, что каждая из этих операций должна выполняться над результатом какой-либо другой операции.

Однако в случаях, когда такая последовательная обработка должна выполняться над потоком информационных объектов, на помощь может прийти *конвейеризация* этих операций.

В этом случае специализированные операционные устройства, выполняющие частные операции, выстраиваются в последовательный конвейер, в котором результат работы одного операционного устройства подается на вход операционного устройства, выполняющего следующую частную операцию (рис. 15.2). Пример — использование конвейера для повышения производительности сборки автомобилей.

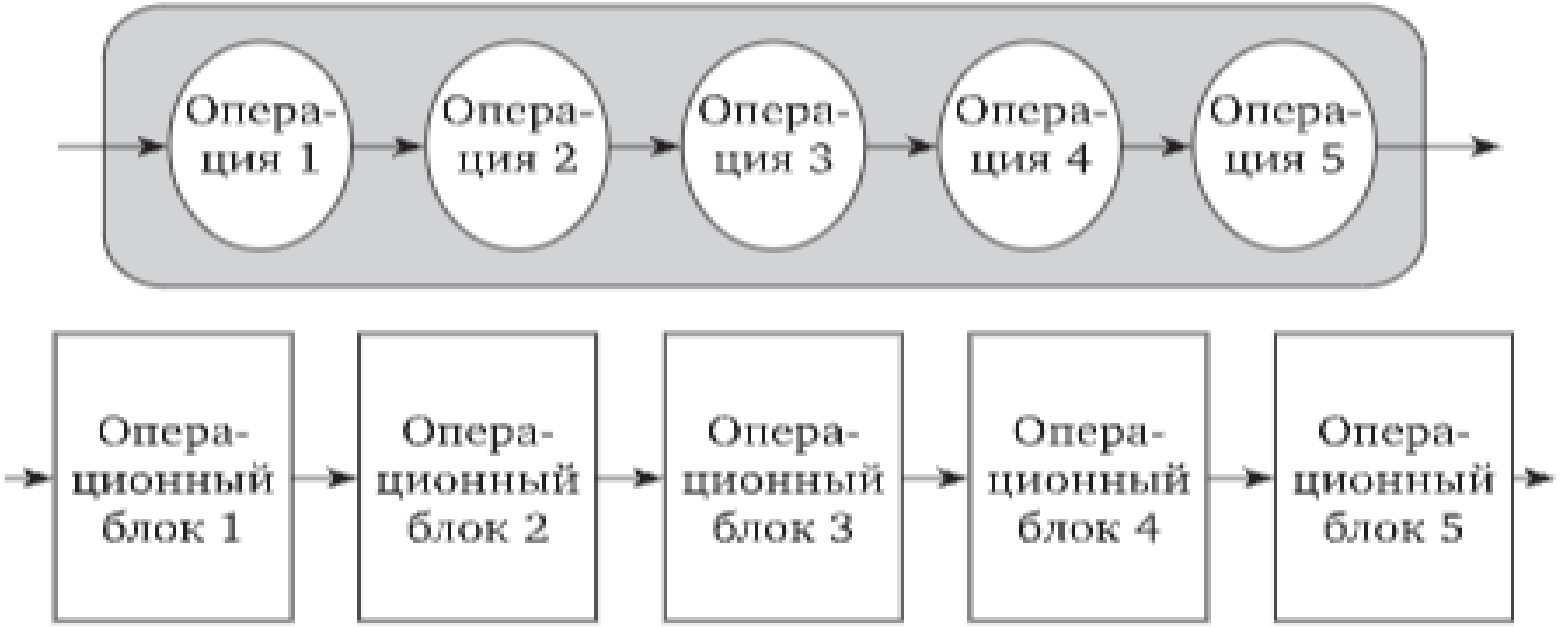

Рис. 15.2. **Конвейер обработки данных**

Эффективность конвейера основана на том, что обработка в нем следующего информационного объекта из потока таких же объектов начинается не после завершения выполнения всех частных операций над предыдущим объектом, а сразу же после того, как предыдущий объект, пройдя через первый операционный блок, переходит на обработку в следующий операционный блок, и т. д. Таким образом, общая производительность такого конвейера из N операционных блоков в идеале будет в N раз выше, чем при обычном решении, когда к обработке следующего информационного объекта компьютер приступает после полного окончания работы с предыдущим объектом.

15.2. Опережающая выборка и конвейеризация выполнения команд

В качестве примера рассмотрим возможность повышения производительности компьютера за счет использования в его архитектуре конвейерной обработки команд выполняемой программы.

Цикл выполнения команды процессором можно представить в виде последовательного выполнения следующих частных операций.

1. *Извлечение команды* (ИК) — чтение из ячейки оперативной памяти, адрес которой определяется содержимым регистра — счетчика команд, кода следующей выполняемой команды и передача его в регистр команд процессора.

2. *Декодирование команды* (ДК) — расшифровка кода операции и спецификаторов операндов в устройстве управления процессором.

3. *Вычисление адресов операндов* (АО) — вычисление исполнительных адресов всех операндов источников. При вычислении адресов учитываются заданные режимы адресации операндов.

4. *Извлечение операндов* (ИО) — извлечение всех операндов-источников из оперативной памяти.

5. *Выполнение команды* (ВК) — выполнение над выбранными операндами операции, заданной кодом операции в команде.

6. *Запись результата в память* (ЗР) — запись результата выполнения команды в ячейке оперативной памяти.

Обычный процессор приступает к циклу выполнения следующей команды только после полного завершения выполнения предыдущей команды. Но использование принципа конвейеризации выполнения команд позволяет получить существенный выигрыш в эффективности выполнения программы за счет совмещения во времени разных этапов выполнения разных команд.

Для реализации конвейера команд в процессоре выделяются соответственные аппаратные исполнительные элементы для выполнения каждой из перечисленных выше частных операций цикла выполнения команд (рис. 15.3), которые осуществляют последовательную обработку команды.

Этот конвейер работает следующим образом (рис. 15.4).

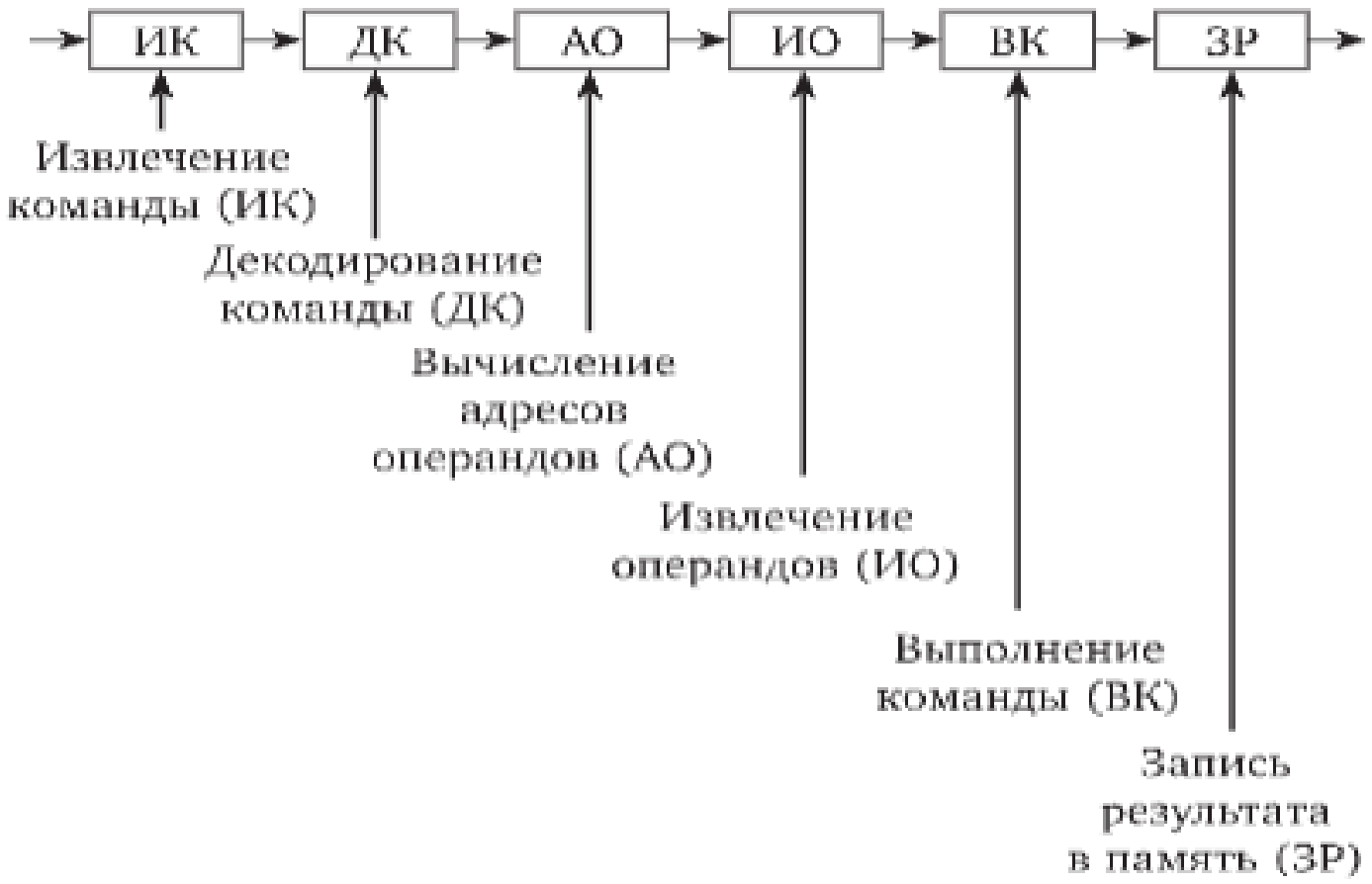

Рис. 15.3. **Операционные элементы конвейера команд**

Рис. 15.4. **Работа конвейера команд**

После того как первая команда, пройдя первый шаг обработки, извлечена из оперативной памяти в регистр команд и передана на следующий этап обработки для ее декодирования в устройстве управления, процессор одновременно приступает к извлечению из памяти кода следующей по порядку

команды; благо ее адрес уже находится в регистре — счетчике команд процессора. Этот прием называется *«опережающая выборка команд»*.

Когда первая команда, пройдя второй шаг обработки, переходит к третьему этапу, к определению исполнительных адресов операндов источников, уже извлеченная вторая команда передается на второй этап обработки, то есть к ее декодированию, и одновременно с этим начинается этап извлечения из памяти кода третьей команды. И такой процесс происходит аналогичным образом с последовательными запуском и обработкой следующих по порядку команд выполняемой программы. Получается, что после завершения выполнения первой команды процессор приступает к извлечению из памяти не второй команды, как в обычном процессоре, а уже седьмой команды.

Несомненно, описанная только что картина конвейерной опережающей выборки и выполнения команд и получаемый при этом выигрыш в производительности выполнения программы являются идеализированными. Во-первых, в действительности будет не одинаковым время выполнения различных частных операций конвейера, что, конечно же, снизит фактический выигрыш от совмещения во времени разных этапов обработки разных команд. И во-вторых, описанная ситуация относится к выполнению программы с линейной структурой, когда все команды программы следуют строго друг за другом. В случае же, когда в программе встречается команда условного перехода, эффект от опережающей выборки команд, конечно, пропадает, так как для выполнения команды условного перехода обязательно нужно знать состояние результата выполнения предыдущей команды, и уже от выполнения этой команды зависит, откуда должна быть выбрана следующая команда.

И тем не менее использование в архитектуре процессора средств конвейеризации для опережающей выборки команд дает существенный выигрыш в эффективности выполнения программ и широко используется в современных процессорах.

15.3. Классификация организации вычислительных систем

М. Флинном (M. I. Flynn) была предложена классификация вычислительных систем в зависимости от числа используемых

в них операционных блоков (процессоров), характеристик потоков обрабатываемых данных и организации их обработки. Эта классификация приведена на схеме, представленной на рис. 15.5.

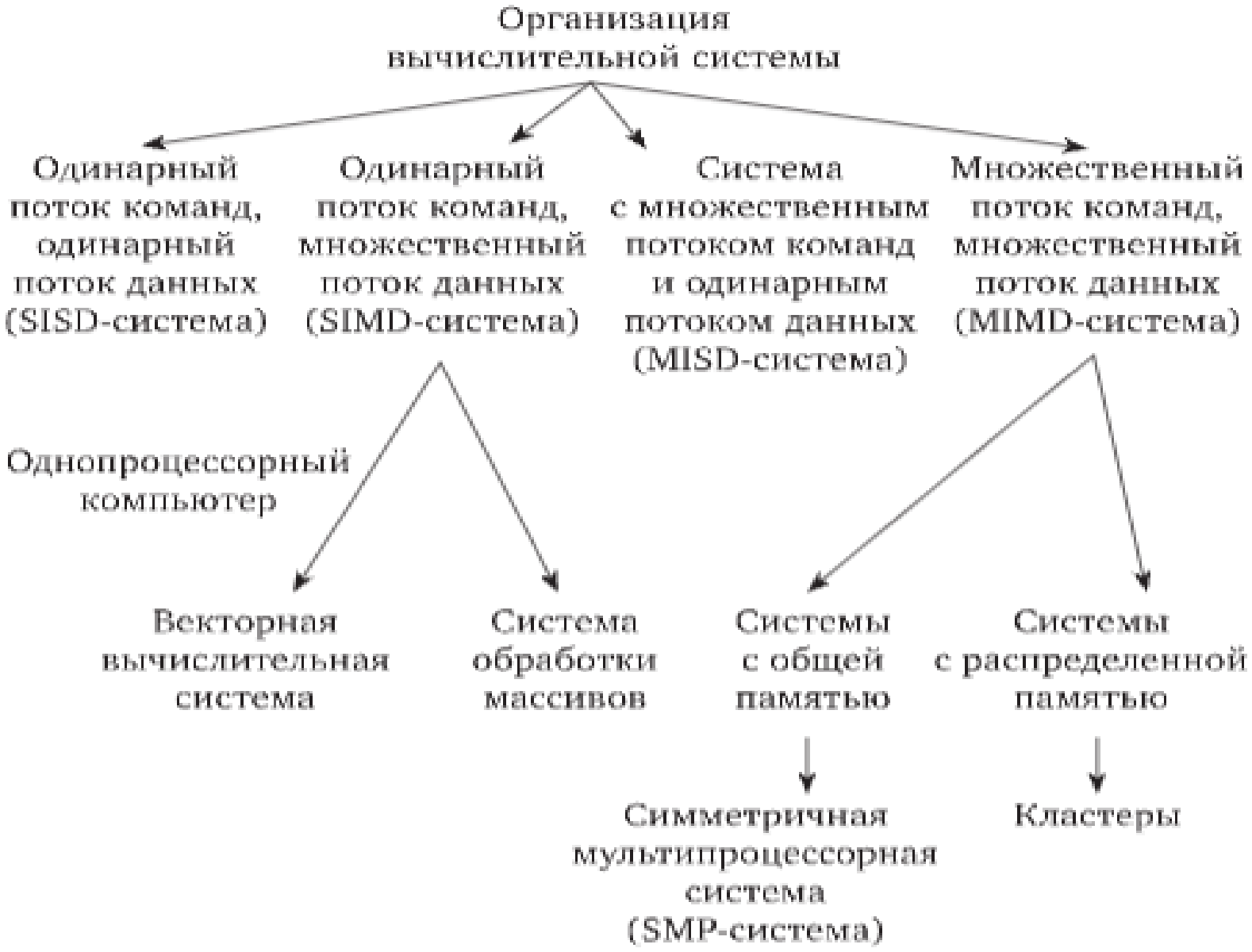

Рис. 15.5. **Классификация организации вычислительных систем**

Рассмотрим представленные на схеме разновидности организации вычислительных систем.

Система с одинарным потоком команд и одинарным потоком данных — SISD-системы (single instruction, single data stream) (рис. 15.6).

Рис. 15.6. **Система с одинарными потоками команд и данных**

В такой системе единственный процессор выполняет единственный поток команд (в каждый момент времени выполняет-

ся одна команда) и обрабатывает единственный поток используемых этими командами данных, хранящихся в единственном блоке памяти.

К этой категории относятся все классические однопроцессорные системы.

Система с одинарным потоком команд и множественным потоком данных — SIMD-системы (single instruction, multiple data stream) (рис. 15.7)

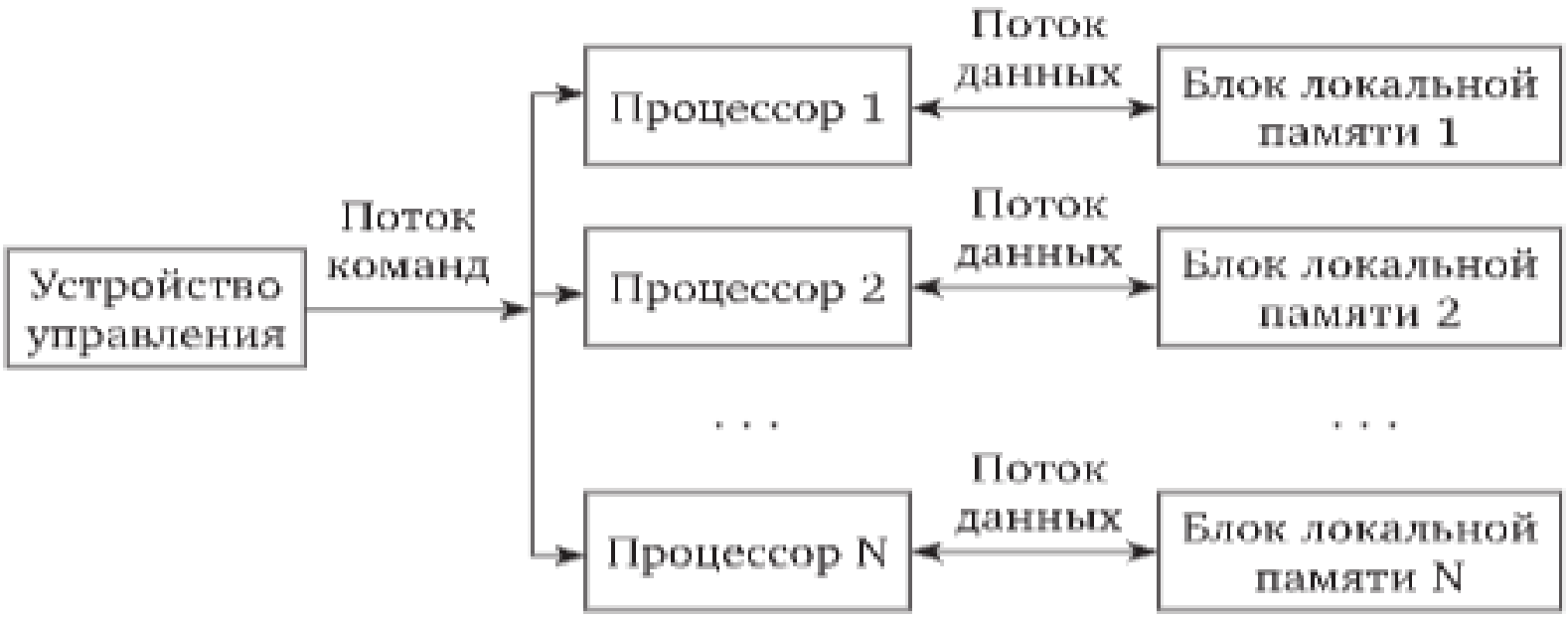

Рис. 15.7. **Система с одинарным потоком команд и множественным потоком данных**

В такой системе имеется несколько одинаковых процессоров, каждый из которых выполняет одинаковую команду из единственного потока команд, формируемого общим для них устройством управления. Каждый процессор связан со своим блоком памяти данных. То есть каждый процессор параллельно обрабатывает свои собственные данные, но по одному и тому же алгоритму.

К этой категории вычислительных систем принадлежат векторные системы.

Система с множественным потоком команд и одинарным потоком данных — MISD-система (multiple instruction, single data stream)

В такой системе единственный поток данных проходит через несколько процессоров, каждый из которых выполняет свою последовательность команд. Принципиально эта система очень напоминает систему с конвейерной обработкой, но на сей раз на каждой рабочей позиции решается своя задача, а не выполняется этап обработки машинной команды.

Такая структура до сих пор не нашла практического воплощения и применения.

Система с множественным потоком команд и множественным потоком данных — MIMD-система (multiple instruction, multiple data stream)

В такой системе имеется множество процессоров, которые одновременно выполняют разные последовательности команд, обрабатывая при этом разные наборы данных. Различают две конфигурации таких систем — MIMD-системы *с общей памятью* и MIMD-системы *с распределенной памятью*.

MIMD-система с общей памятью (рис. 15.8)

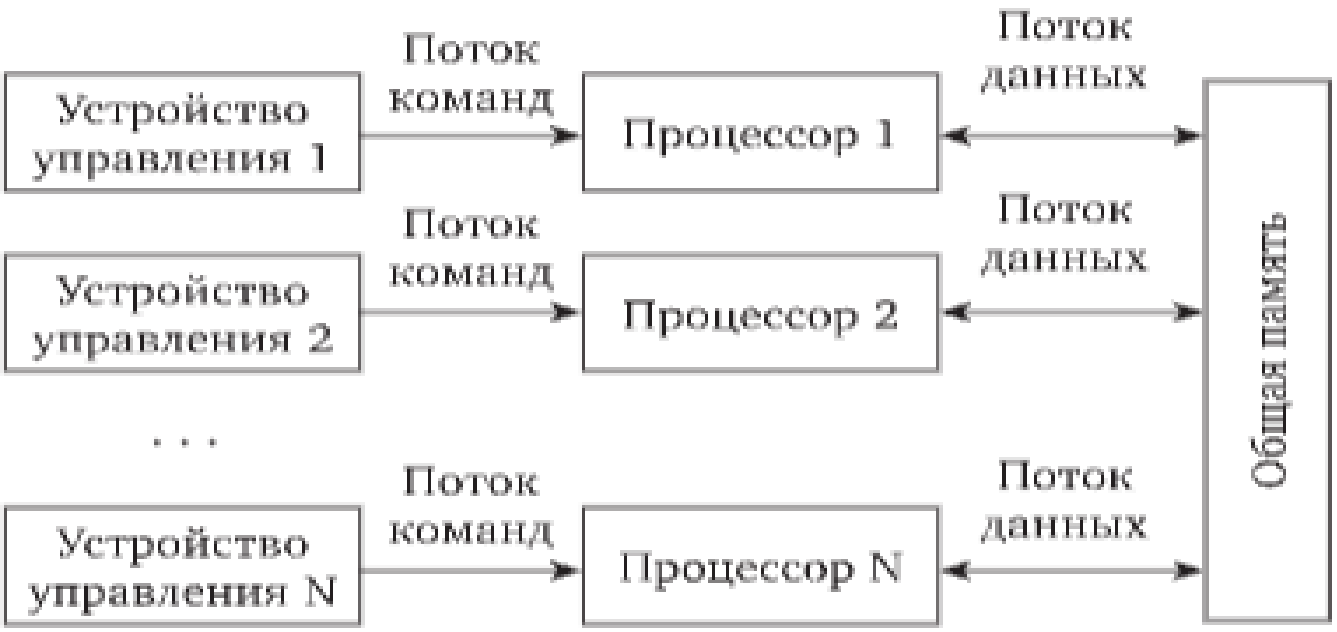

Рис. 15.8. **MIMD-система с общей памятью**

В такой системе все процессоры одинаковые и работают с единым полем общей памяти. То есть каждый из процессоров может получить доступ ко всем хранящимся в памяти командам и данным.

Такие системы получили название *симметричных многопроцессорных систем*, или *SMP-систем* (symmetric multiprocessor).

MIMD-система с распределенной памятью (рис. 15.9)

В такой системе все процессоры одинаковые, но, в отличие от SMP-систем, работают каждый со своим собственным запоминающим устройством. Взаимный обмен информацией между процессорами осуществляется через локальную сеть.

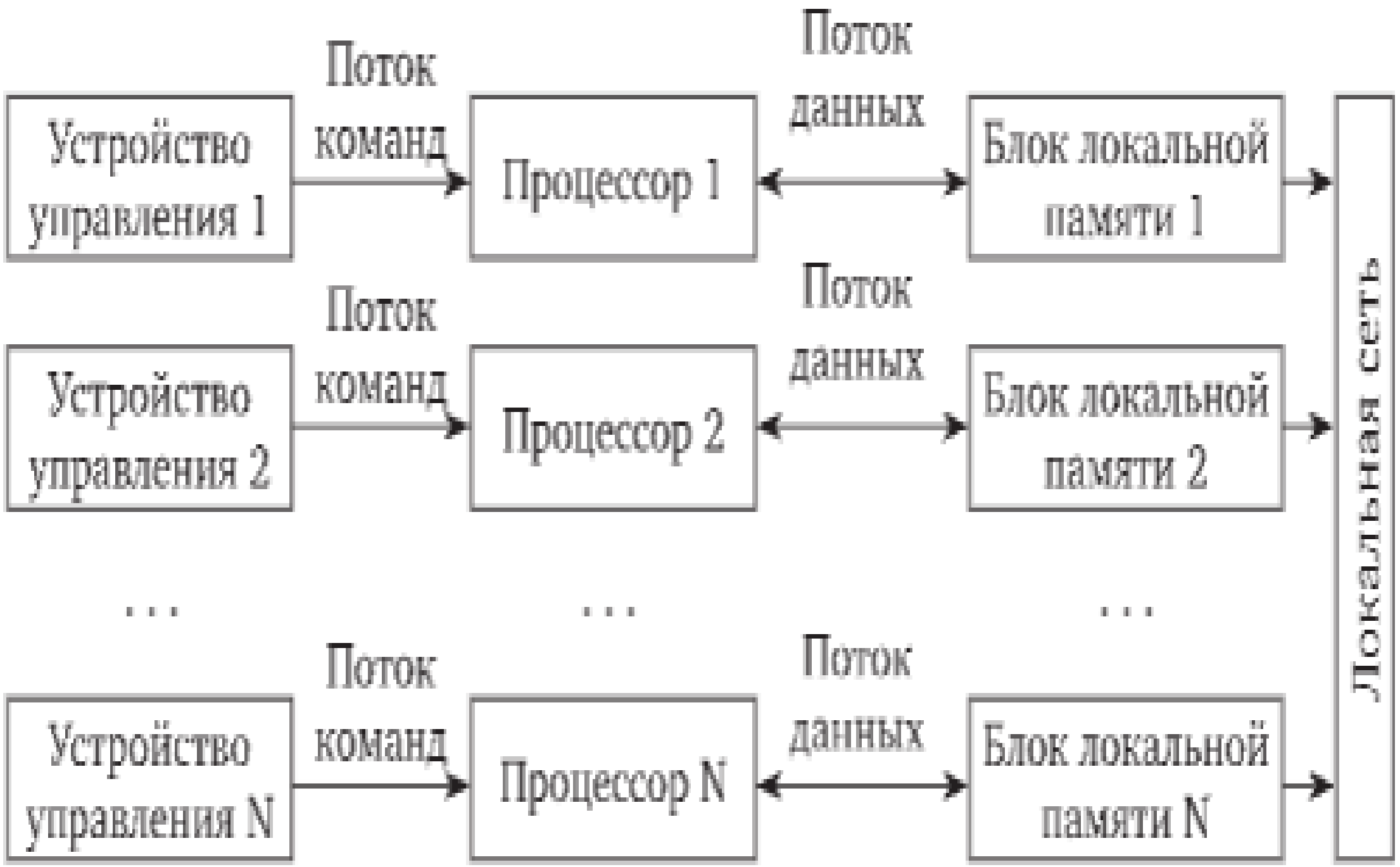

Рис. 15.9. **MIMD-система с распределенной памятью**

Глядя на схему, представленную на рис. 15.9, можно обратить внимание на то, что структура каждого процессорного узла в точности совпадает с представленной на рис. 15.6 схемой системы с одинарными потоками команд и данных, проще говоря — со схемой классической однопроцессорной ЭВМ.

Обе эти схемы (с общей и распределенной памятью) многопроцессорных систем в настоящее время нашли широкое применение. Рассмотрим их характеристики более подробно.

Симметричные мультипроцессорные (SMP) системы

Такие системы состоят из нескольких одинаковых процессоров, с помощью магистрали связанных между собой и с общей памятью в единую вычислительную машину. К этому классу относятся, например, современные многоядерные процессоры, компьютеры с несколькими процессорами, в том числе и многоядерными, расположенными на одной материнской плате, наконец суперкомпьютерные системы, содержащие тысячи процессоров.

Для SMP компьютерных систем характерно:

- наличие двух или более одинаковых или близких по характеристикам процессоров;
- все эти процессоры имеют равноправный доступ к общей памяти, к которой они подсоединены или через общую магистраль, или через другой механизм, обеспечивающий одинаковое время доступа процессоров к памяти;
- процессоры имеют доступ к общим средствам ввода-вывода либо через один общий канал, либо через раздельные каналы;
- все процессоры способны выполнять одинаковый набор функций (отсюда и определение «симметричная система»);
- весь такой многопроцессорный комплекс централизованно управляется общей операционной системой, которая обеспечивает взаимодействие между процессорами и программами на уровне заданий, файлов и элементов данных.

Отметим преимущества SMP-систем перед однопроцессорными системами.

- Повышение производительности. Отдельные задачи и приложения могут выполняться параллельно.
- Надежность. Поскольку все процессоры однотипны и могут выполнять одни и те же задачи, в случае отказа одного из процессоров выполнение задач может перераспределяться между другими процессорами.

- Возможность функционального наращивания. Если это предусмотрено конструктивом системы, пользователь может повышать ее производительность, включая в ее состав дополнительные процессоры.
- Производство однотипных систем разной производительности. Изготовитель компьютеров может предложить клиентам линейку систем с одинаковой архитектурой, но разной стоимостью и производительностью, отличающихся количеством установленных процессоров и обладающих возможностью наращивания производительности путем увеличения числа процессоров.

Кластеры как альтернатива SMP-системам

Организация *кластерных* многопроцессорных систем предполагает объединение в единую систему полноценных компьютеров, каждый из которых в состоянии работать самостоятельно, но выполняющих совместно некоторую работу таким образом, что со стороны они воспринимаются как единый вычислительный ресурс (см. рис. 15.9). Отдельные компьютеры в составе кластера называются узлами (nodes).

Кластеры обладают следующими преимуществами.

- Абсолютная масштабируемость. В состав кластера может быть включено любое число компьютеров. В отличие от SMP-систем, у кластеров нет ограничений на количество процессоров, обусловленных конструктивом системы (например, количеством резервных слотов для процессоров на материнской плате SMP-компьютера).
- Возможность наращивания в процессе эксплуатации. Вычислительную мощность кластера легко можно наращивать в процессе его эксплуатации простым добавлением в него новых узлов (компьютеров).
- Высокая надежность. Поскольку каждый узел является самодостаточным компьютером, отказ одного узла не приводит к потере работоспособности всего комплекса.
- Значительное снижение соотношения цена/производительность. При построении кластера могут использоваться обычные компьютеры массового производства, то есть гораздо более дешевые.

Перечисленные достоинства кластерных вычислительных систем привели к тому, что в настоящее время большинство высокопроизводительных компьютерных вычислительных си-

стем строятся именно по этому принципу, а благодаря присущему им существенно более низкому соотношению цена/производительность средства для высокопроизводительных вычислений стали гораздо доступнее для очень многих их потенциальных потребителей.

Контрольные вопросы

1. В чем состоят идеи повышения производительности ЭВМ путем распараллеливания и конвейеризации процессов?
2. Что такое опережающая выборка команд?
3. За счет чего происходит увеличение производительности ЭВМ при использовании конвейера команд?
4. Классификация вычислительных систем.
5. Сравнение высокопроизводительных вычислительных систем на базе SMP-систем и кластеров.