1. АРХИТЕКТУРЫ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Параллельные компьютеры интересны тем, что предлагают концентрацию вычислительных процессорных ресурсов, ресурсов памяти, высокоскоростных каналов ввода – вывода для решения важных вычислительных проблем.

Именно такое неформальное определение понятия параллельного компьютера, как «совокупность процессорных элементов, которые взаимодействуют и кооперируются для быстрого решения серьезных задач» (Almasi and Gottlieb 1989), включает в себя и суперкомпьютеры с сотнями и тысячами процессоров, и рабочие станции, объединенные в сеть, и многопроцессорные рабочие станции, и встроенные системы.

Детальный анализ современных тенденции в развитии вычислительной техники [1,2] в области использования вычислительных систем, развития технологической базы, компьютерных архитектур, суперкомпьютеров, показывают естественный переход от однопроцессорных систем к многопроцессорным.

Целью данного раздела является определение структур вычислительных систем для изучения всего многообразия параллельных компьютерных архитектур и понимания взаимосвязи и взаимного влияния между ними. Одновременно будет кратко изложен обзор эволюции параллельных машин.

По - существу, параллельные компьютеры расширяли обычные концепции компьютерных архитектур, за счет добавления коммуникационной среды. Коммуникационная архитектура, как и компьютерная, имеет две важные грани. Они определяются базовыми операциями взаимодействия и синхронизации, а так же организационной структурой, которая реализует данные операции.

Высшим уровнем коммуникационной архитектуры является программная модель [2], которая реализована в параллельной системе и используется программистом в соответствии с областью применения. Модель параллельного программирования специфицирует образ частей программы, выполняемых параллельно и обменивающихся между собой информацией, и операции синхронизации, доступные для координации взаимодействия параллельных частей программы.

**Ниже рассмотрены основные классические архитектуры параллельных систем, реализованные в серийных образцах [1].**

**1.1.    АРХИТЕКТУРЫ С РАЗДЕЛЯЕМОЙ ОБЩЕЙ ПАМЯТЬЮ**

Один из наиболее важных классов параллельных машин – *shared memory multiprocessors* – многопроцессорные системы с разделяемой общей областью памяти. Ключевой характеристикой данных систем является то, что взаимодействие процессоров осуществляется как обычное выполнение инструкций доступа к памяти (т.е. loads and stores). Данный класс систем имеет большую историю развития, начало которой датируется 1960 годом (система BINAC).

Основа программной модели (Shared address) для таких архитектур, по существу, есть разделение времени доступа к общей области памяти (time – sharing). В процессах часть их адресного пространства является разделяемой с другими процессами. Каждый процесс имеет виртуальную область памяти, состоящую из адресного пространства разделяемой памяти и собственного адресного пространства. На рис. 1.1 представлена типовая модель взаимодействия процессоров через механизм разделяемой общей памяти. На рисунке показана связь виртуального адресного пространства процессов (P0 – Pn), состоящего из разделяемой и собственной областей, с физической областью памяти.

Операции записи и чтения в разделяемую область памяти требует дополнительного контроля. Т.е. операционная среда выполняет специальные функции синхронизации процессов (операций записи и чтения в разделяемую область памяти). Например, должна быть блокирована операция чтения данных одного процесса до тех пор, пока в данную ячейку не будет записан результат процесса – предшественника.

Коммуникационное оборудование систем с общей памятью позволяет расширять системную память естественным образом. По - существу, большинство компьютерных систем позволяют процессору и ряду контроллеров ввода – вывода обеспечивать доступ к набору модулей памяти через некоторую коммуникационную среду, как показано на рис 1.2.

Большинство вычислительных систем содержат один или более модулей памяти (П), доступной процессору и контроллеры ввода-вывода через аппаратную коммуникационную среду.

Наращивание мощности системы достигается простым добавлением процессоров, модулей памяти и числа контроллеров ввода-вывода (которые также являются разделяемыми), в зависимости от требований к системе. На рис. 1.2. дополнительный процессор выделен тонировкой. При этом реальный рост производительность всей системы существенно зависит от специфики системной организации конкретного компьютера, т.к. рост числа процессоров и процессов приводит, постепенно, к несбалансированности между частотой обращений к разделяемой памяти и выполнением собственно программ. Это определяется тем, что не удается на практике реализовать идеальную - *Parallel Random Access Machine* (PRAM), когда любой процессор может осуществить доступ к любой ячейки памяти в любой момент времени. Для реализации данного принципа, обычно, используют иерархическую организацию разделяемой памяти, т.е. уменьшают количество обращений за счет, например, использования cache – памяти (С).

Можно выделить несколько основных типов коммуникационных сред, используемых в архитектурах разделяемой общей памяти (рис 1.3).

Для удовлетворения требований по загрузке системы она может иметь несколько каналов ввода-вывода, которые обеспечивают прямой доступ к каждому модулю памяти. Такого типа системы имеют организацию перекрестного соединения - *switch connecting* – процессоров (Пр), нескольких каналов ввода-вывода (I/O) и нескольких модулей памяти (П), показанной на рис 1.3а. Размерность рассматриваемой структуры определяется числом входов-выходов аппаратного коммутатора (swith). В ранних системах размер и стоимость рассматриваемой коммуникационной среды ограничивалась малым числом процессоров. В дальнейшем рост числа процессорных элементов определялся ростом удельного веса аппаратной части, с одновременным снижением ее стоимости. Цена размеров перекрестного соединения становится лимитирующим фактором и, в большинстве случаев, это приводит к появлению многоуровневой иерархической структуры коммуникационной среды - *multistage interconnection network* - показанной на рис 1.3б. В этом случае цена растет медленнее, чем число портов. Понятно, что экономия приводит к увеличению времени ожидания соединения *(latency)* и уменьшения полосы пропускания на порт. Способность доступа ко всей памяти прямо из каждого процессора имеет несколько преимуществ: любой процессор может запустить любые процессы или обратиться к любому событию ввода-вывода, а структуры данных могут быть разделены внутри операционной системы.

Широкое применение систем с разделяемой общей памятью связано с появлением 32-bit микропроцессоров в середине 1980 г., cache-память, плавающая запятая, и управление блоком памяти было реализовано на одной - двух платах (Bell 1985). Большинство машин среднего уровня, включающих миникомпьютеры, серверы, рабочие станции и персональные компьютеры имеют шинную организацию – *bus interconnect* - как показано на рис 1.3в. и эта шина может быть адаптирована для поддержки многопроцессорных систем. Стандартный механизм доступа к шине позволяет любому процессору достичь любого физического адреса в системе. Как и в случае перекрестного соединения (рис. 1.3а), вся память равноудалена от процессоров, поэтому все процессоры имеют одинаковое время доступа (latency) к памяти.

Такая конфигурация обычно называется symmetric multiprocessor (SMP). SMP идеально подходит как для параллельных программ, так и для малтипрограммирования [2]. Типичным примером организации многопроцессорной системы с симметричным доступом к памяти, основанной на шинной архитектуре, является компьютер Intel Pentium Pro four-processor “quad pack”, иллюстрирующий первый SMP для широкого рынка (рис. 1.4).

Материнская плата Intel quad-processor Pentium Pro использовалась во многих многопроцессорных серверах, таких как HP NetServer LX series, являясь главным элементом дизайна для систем с малым числом процессоров - *small-scale design*. Как показано на рис. 1.4, на ней можно было объединить до четырех процессорных модулей (Пр-модуль), содержащих Pentium Pro (166 MHz) процессор (CPU), cache-память, контролллер прерываний, интерфейс шины. Такой модуль был реализован на одном кристалле, имеющем разъемы для прямого включения в 64-bit шину памяти. 66 MHz шина имела пиковую пропускную способность 528 Mb/s. Двухкристальный контроллер памяти и четырехкристальный контроллер мультиплексного канала - *memory interleave unit (MIU)* - обеспечивали взаимодействие шины с модулями памяти (DRAM). Шина памяти через PCI мосты сопрягалась с двумя независимыми шинами стандарта PCI, которые обеспечивали связь с монитором, сетью и устройствами ввода-вывода (I/O). Структура Pentium Pro “quad pack” была похожа на большинство ранних машин класса SMP, но ее реализация отличалась наибольшей степенью интеграции.

На рис. 1.5 показана структура сервера Sun Enterprise Server, который так же поддерживает симметричный доступ к памяти (SMP), не смотря на то, что она физически распределена между процессорными платами.

В отличии от Pentium Pro “quad pack”, многопроцессорный сервер Sun UltraSparc-based Enterprise является представителем систем большей размерности по числу процессоров - *larger-scalle design*. Широкая (256-bit), высоко поплайнизированная шина памяти имеет пропускную способность 2,5 Gb/s. Сервер имеет иерархическую структуру, где каждая плата реализует структурную единицу системы, либо вычислительный модуль (Пр - модуль) из двух процессоров и памяти, либо модуль ввода-вывода. Наличие двух типов модулей является обязательным условием работоспособности системы. Вычислительный модуль содержит два процессора UltraSparc, каждый их которых имеет двухуровневую cache-память (первый уровень (C) – 16 KB, второй уровень (С”) – 512 KB), плюс два 512-bit-wide банка памяти и внутренний коммутатор*.* Модуль ввода-вывода поддерживает три SBUS слота для расширения функции ввода-вывода, SCSI разъем, 100bT Ethernet порт и два конектора для подключения оптических каналов - FiberChannell interfaces. Стандартная конфигурация сервера включает в себя 24 процессорных и 6 модулей ввода-вывода. Хотя банки памяти являются физически разнесенными попарно между процессорными модулями, вся память равноудалена от процессоров и доступна им через общую шину, что соответствует требованиям SMP. Данные могут быть размещены в любом месте без влияния на производительность системы.

Факторы ограничивающие число процессоров в системе различны для рассмотренного случая и для архитектуры с сетью из коммутаторов. (рис.1.3;а,б). Дополнение процессоров в коммутатор – дорого, однако общая производительность системы возрастает с числом портов. Цена добавления процессоров к шине – мала, но производительность всей системы остается фиксированной. В последнем случае ограничителем является пропускная способность шины. Если цена доступа к памяти станет слишком большой, процессоры будут тратить большую часть времени на режим ожидания и преимущество большого числа процессоров будет снивелирована.

Одним из естеcтвенных подходов построения масштабируемых машин с разделяемой общей памятью, поддерживающий симметричный доступ к памяти, показан на рис. 1.2. Он обеспечивает масштабируемость коммуникационной среды между процессорами и модулями памяти. Основной недостаток заключается в том, что при каждом обращении к памяти затрачивается много времени на ожидание кругового путешествия по сети, поэтому каждый процессор должен обеспечить высокую пропускную способность.

Альтернативный подход создания масштабируемой среды взаимодействия процессоров показан на рис 1.6.

Процессор и модули памяти интегрированы между собой таким образом, что доступ к локальной памяти осуществляется существенно быстрее, чем к удаленной. Такая организация взаимодействия процессоров носит название несимметричного доступа к памяти - nonuniform memory access (NUMA) - при котором контроллер локальной памяти определяет, выполнять ли доступ к локальной памяти или осуществлять транзакцию сообщения к удаленной памяти (при этом системы ввода-вывода могут быть либо частью каждого модуля, либо консолидироваться в специальный модуль I/O). В таком случае доступ к собственным данным процессора, часто может быть выполнен локально, как и доступ к разделяемым данным, если они сохранены в локальном модуле. Доступ к локальной памяти быстрый и не возрастает во времени, по сравнению с удаленным доступом. Среднее время доступа существенно уменьшается, если большую часть занимают обращения к локальной памяти. Требования к пропускной способности сети тоже уменьшаются.

Не смотря на некую привлекательность концептуальной простоты SMP архитектуры, подход NUMA стал куда более приемлемым для больших многопроцессорных систем, благодаря его неотъемлемым преимуществам, приводящим к росту производительности таких систем.

Примером такого стиля проектирования является CRAY T3E, показанный на рис. 1.7.

CRAY T3E может содержать до тысяч процессоров, работающих с глобальным общим адресным пространством. Каждый модуль - *node* - содержит DEC Alpha процессор (Пр), локальную память (П), интегрированный с контроллером памяти сетевой интерфейс и сетевой коммутатор. Компьютер организован как трехмерный куб, в котором каждый модуль соединяется с его соседями через 650 Mb/s линки (стандарт point-to-point). Любой процессор может иметь доступ к любой памяти, однако идеология NUMA реализована в коммуникационной архитектуре, как наилучшая для характеристик производительности системы. Контроллер памяти модуля захватывает доступ к удаленной памяти и руководит транзакцией сообщения в контроллере памяти удаленного модуля от имени локального процессора. Транзакция сообщения автоматически маршрутизируется через промежуточные модули (вершины) до места назначения, с малыми задержками на каждом переходе. Данные удаленной памяти не кэшируются, поскольку нет аппаратного механизма их сохранения. Система ввода-вывода CRAY T3E распределена между совокупности вершин, располагающихся на поверхности куба, которые соединяются с внешним миром через дополнительную сеть.

В этой машине реализована структура, при которой хоть вся память и доступна любому процессору, распределение данных между процессорами отдано программисту. Caches – память (С) используется только для хранения данных (инструкций) из локальной памяти. Т.о. задача программиста – избежать частых обращений к удаленной памяти.

В заключение, надо отметить, что операции взаимодействия и синхронизации в моделях программирования с разделяемой общей адресной областью [2], специфицируются операциями READS и WRITES разделяемых переменных. Эти операции прямо отображаются в коммуникационные абстракции, содержащие LOAD и STORE (инструкции доступа к глобальной разделяемой общей памяти), которые прямо поддержаны аппаратно через доступ к разделяемым зонам физической памяти. Программная модель и коммуникационные абстракции имеют прямую аппаратную реализацию. Для каждого процесса обращение к памяти, есть адрес в его виртуальном адресном пространстве. Адрес транслируется в процесс идентификации физической области, которая может быть локальной или удаленной, по отношению к процессору и которая может быть доступна другим процессорам. Трансляция адреса реализуется защищенно, в пределах разделяемого адресного пространства, как это делается в однопроцессорных системах.

Эффективность систем с разделяемой общей памятью зависит от времени ожидания доступа к памяти, связанного с пропускной способностью среды передачи данных. Для чтобы достичь маштабируемости таких систем, все решения, включая все механизмы связи, используемые для доступа к разделяемой памяти, должны быть правильно сбалансированы.

1.2.    АРХИТЕКТУРЫ С РАСПРЕДЕЛЕННОЙ ОБЛАСТЬЮ ПАМЯТИ

    Ко второму важному классу параллельных машин относятся многопроцессорные системы с распределенной областью памяти - *message-passing architectures* (MPA). MPA используют законченные компьютеры, включающие микропроцессор, память и подсистему ввода-вывода, как узлы для построения системы, объединенные коммуникационной средой, обеспечивающую взаимодействие процессоров посредством простых операций ввода-вывода. Структура высокого уровня для MPA практически такая же, как и для NUMA машин, т.е. машин с разделяемой памятью, показанных на рис. 1.6. Первое отличие состоит в том, что коммуникации интегрированы в уровень ввода-вывода, а не в систему доступа к памяти. Этот стиль дизайна имеет много общего с сетями из рабочих станций или кластерными системами, за исключением того, что в МРА пакетирование узлов обычно более плотное, нет монитора и клавиатуры на каждом узле, а производительность сети намного выше стандартной. Интеграция между процессором и сетью имеет склонность быть более тесной чем традиционные структуры ввода-вывода, которые поддерживают соединения с оборудованием, которое более медленное, чем процессор. Начиная с посылки сообщения MPA есть фундаментальное взаимодействие ПРОЦЕССОР – ПРОЦЕССОР.

    Системы с распределенной памятью имеют существенную дистанцию между программной моделью и действительными аппаратными примитивами. Коммуникации осуществляются через средства операционной системы или библиотеку вызовов, которые выполняют много акций более низкого уровня, включающих операции коммуникации.

    Наиболее общие операции взаимодействия на пользовательском уровне (user-level) в MPA есть варианты посылки (SEND) и получения (RECEIVE) сообщения. Совместный механизм SEND и RECEIVE, вызванный передачей данных из одного процесса в другой, показан на рис 1.8.

Передача данных из одного локального адресного пространства к другому произойдет, если посылка сообщения со стороны процесса - отправителя будет востребована процессом - получателем сообщения.

С этой целью в большинстве системах с распределенной памятью сообщение специфицируется операцией SEND, которая добавляет к сообщению специальный признак (*tag*), а операция RECEIVE в этом случае выполняет проверку сравнения данного признака.

Сочетание посылки и согласованного приема сообщения (на основе совпадения признаков) выполняет логическую связку – синхронизацию события, т.е. копирования из памяти в память. Имеется несколько возможных вариантов синхронизаии этих событий, в зависимости от того, завершиться ли SEND к моменту, когда RECEIV будет выполнен или нет (т.е. будет ли снова доступен буфер посылки для использования до момента получения подтверждения приема). Похожим образом RECEIV может, в принципе, подождать до момента согласованной посылки (SEND) или использовать почтовый ящик для получения сообщения. Каждый из этих вариантов имеет несколько различную интерпретацию и различные требования к реализации.

    Механизм посылки сообщений долго использовался как средство коммуникации и синхронизации совокупности арбитрирующих взаимодействующих последовательных процессов, даже на одном процессоре. В качестве примеров можно привести языки программирования типа CSP и Occam, наиболее общие функции операционных систем, типа SOCKETS.

    Первые машины с распределенной областью памяти обеспечивали аппаратную поддержку примитивов (команд), которые очень напоминали простую абстракцию взаимодействия SEND/RECEIV на пользовательском уровне, с некоторыми дополнительными ограничениями. Каждый узел системы соединялся с определенным (фиксированным) числом соседей по регулярной схеме (образцу) на основе связи точка-точка (poin-to-point), поведение которой, в свою очередь, описывалось простым FIFO. Такой тип конструкции для минимального 3D куба показан на рис. 1.9, где каждый узел имеет связи с соседями по трем направлениям через буфер FIFO.

## *Что такое архитектура и структура компьютера?*

При рассмотрении компьютерных устройств принято различать их архитектуру и структуру.

|  |
| --- |
| *Архитектурой* компьютера называется его описание на некотором общем уровне, включающее описание пользовательских возможностей программирования, системы команд, системы адресации, организации памяти и т.д. Архитектура определяет принципы действия, информационные связи и взаимное соединение основных логических узлов компьютера: процессора, оперативного ЗУ, внешних ЗУ и периферийных устройств. Общность архитектуры разных компьютеров обеспечивает их совместимость с точки зрения пользователя. |

|  |
| --- |
| ***Структура*** компьютера — это совокупность его функциональных элементов и связей между ними. Элементами могут быть самые различные устройства — от основных логических узлов компьютера до простейших схем. Структура компьютера графически представляется в виде структурных схем, с помощью которых можно дать описание компьютера на любом уровне детализации. |

Наиболее распространены следующие архитектурные решения.

***Классическая архитектура*** (архитектура фон Неймана) — ***одно арифметико-логическое устройство*** (АЛУ), через которое проходит *поток данных*, и *одно устройство управления* (УУ), через которое проходит ***поток команд — программа***.Это ***однопроцессорный компьютер***.

К этому типу архитектуры относится и архитектура персонального компьютера с ***общей шиной*.** Все функциональные блоки здесь связаны между собой общей шиной, называемой также ***системной магистралью***.

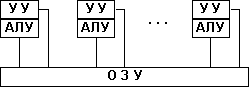
|  |
| --- |
| Физически ***магистраль*** представляет собой многопроводную линию с гнездами для подключения электронных схем. Совокупность проводов магистрали разделяется на отдельные группы: шину адреса, шину данных и шину управления. |

Периферийные устройства (принтер и др.) подключаются к аппаратуре компьютера через специальные **контроллеры — устройства управления периферийными устройствами.**

|  |
| --- |
| ***Контроллер*** — устройство, которое связывает периферийное оборудование или каналы связи с центральным процессором, освобождая процессор от непосредственного управления функционированием данного оборудования. |

### *Многопроцессорная архитектура*

. Наличие в компьютере нескольких процессоров означает, что **параллельно** может быть **организовано много потоков данных и много потоков команд**. Таким образом, **параллельно могут выполняться несколько фрагментов одной задачи**. Структура такой машины, имеющей общую оперативную память и несколько процессоров, представлена на рис.



### *Многомашинная вычислительная система*

. Здесь **несколько процессоров,** входящих в вычислительную систему, **не имеют общей оперативной памяти**, **а имеют каждый свою (локальную).** Каждый компьютер в многомашинной системе имеет классическую архитектуру, и такая система применяется достаточно широко.

Однако эффект от применения такой вычислительной системы может быть получен только при решении задач, **имеющих очень специальную структуру**: **она должна разбиваться на столько слабо связанных подзадач, сколько компьютеров в системе.**

Преимущество в быстродействии многопроцессорных и многомашинных вычислительных систем перед однопроцессорными очевидно.

·

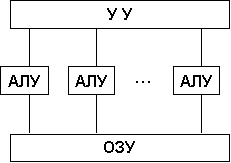
### *Архитектура с параллельными процессорами*

. Здесь ***несколько АЛУ работают под управлением одного УУ*.** Это означает, что множество данных может обрабатываться по одной программе — то есть по одному потоку команд.

*Высокое быстродействие такой архитектуры можно получить только на задачах, в которых одинаковые вычислительные операции выполняются одновременно на различных однотипных наборах данных.* Структура таких компьютеров представлена на рис.



Рис. **Архитектура с параллельным процессором**



В современных машинах часто присутствуют элементы различных типов архитектурных решений. Существуют и такие архитектурные решения, которые радикально отличаются от рассмотренных выше.

Серия Т представляет собой разработку фирмы FPS в области архитектур с массовым параллелизмом. Конфигурация системы представляет собой n-мерный гиперкуб - структуру, в которой каждый узел связан с п ближайшими соседями. Архитектура серии Т создана благодаря использованию трех важных разработок:  
1. СБИС Inmos Transputer.  
2. Язык программирования Оккам.  
3. Обновленная схема многопортовой памяти.  
Даже в быстроменяющемся мире параллельных архитектур эти идеи выделяются среди подходов, используемых в разработках других параллельных систем.  
В этой главе обсуждается архитектура серии Т и два примера программ, которые выполнялись на небольшой системе серии Т. Первая программа демонстрирует межузловые связи, вторая - использование аппаратуры и утилит векторной обработки.  
6.1. Аппаратура  
Конфигурация системы серии Т может изменяться от 8 (23) узлов (минимальная система) до 16 384 (214) узлов (максимальная система). Пиковая производительность системы в этих экстремальных случаях возрастает с 96 Mflops до 196 Gflops.   
Языки программирования: Компилятор:  
Машина / модель: Местонахождение: Процессоры: Операционные системы:  
Т/20  
Floating Point Systems, Бивертон, Орегон 16 узлов, каждый с ОЗУ 1 Мбайт VMS на внешней машине MicroVAX TOPSYS В01 на системной плате серии Т VB Main Process В01 на векторной плате серии Т  
Параллельный процессор FPS серии Т   89  
Предельной топологией системы в ее максимальной конфигурации является 14-мерный гиперкуб. Другие топологии, определяемые пользователем, такие как сетка, тор, кольцо, цилиндр, относительно легко создаются из гиперкубической схемы связи. Базовая конфигурация серии Т содержит модуль, состоящий из:  
• платы системного управления;  
• 80-Мбайтного системного диска;  
• 8 векторных плат.  
Удвоение числа векторных плат добавляет новое измерение в структуре гиперкуба. Прикладные программы можно разрабатывать независимо от размера системы, предусматривая простые средства для работы на системах серии Т различной конфигурации.  
Плата системного управления  
Плата системного управления (ПСУ) связывает модуль серии Т с восемью присоединенными векторными платами и с дисковой подсистемой, а также при необходимости с внешней машиной. Программно ПСУ не доступна пользователю, однако она содержит операционную систему, выполняющую запросы пользовательских программ на организацию взаимодействия и обмен данными с внешней машиной, векторными платами, дисковой подсистемой и, в многомодульных системах, с другими ПСУ.  
Векторные платы  
Каждую векторную плату (ВП) можно рассматривать как одноплатный матричный процессор с управляющей машиной (host computer). Уже упоминалось, что существует восемь векторных плат, подсоединенных к плате системного управления. Каждая векторная плата содержит:  
• транспьютер;  
• адаптер каналов/MUX;  
• многопортовое ОЗУ емкостью в 1 Мбайт;  
• секцию АЛУ, состоящую из устройств сложения и умножения чисел с плавающей запятой (32- или 64-разрядный форматы IEEE);  
• статическое ОЗУ емкостью в 4К 64-разрядных слов, используемое для микропрограммной реализации векторных операций.  
• АЛУ и память микропрограмм образуют векторное процессорное устройство (ВПУ).