## ТЕМА 1.

### Предмет курса. Основные определения. Классификация.

Прошло всего полвека с момента появления первой ЭВМ. Но за этот короткий срок сменилось несколько поколений вычислительных машин, значительно отличающихся технологической и элементной базой, архитектурой,, языками и ПО. Исторически первыми и до сих пор широко распространёнными, занимающими господствующее положение на рынке, являются одномашинные *СИСТЕМЫ ОБРАБОТКИ ДАННЫХ (СОД),* построенные на базе единственной ЭВМ с традиционной однопроцессорной схемой.

Но с ростом объёмов и сложности задач часто такие системы иногда становятся неэффективными из-за малой мощности или низкой надёжности. Вообще, в дальнейшем, при рассмотрении любых СОД, будем анализировать их важнейшие параметры:

- ***производительность;***
- ***S надёжность;***
- ***S стоимость.***

Не следует забывать, что одним из основополагающих факторов, влияющих на эти параметры, является сложность ОС.

**СОД** - совокупность технических средств и ПО, предназначенная для информационного обслуживания пользователей и технических объектов.

Основа СОД - технические средства, т.к. их производительностью и надёжностью в наибольшей степени определяется эффективность СОД.

Существуют 2 естественных пути повышения производительности и надёжности:

1. Совершенствование элементной базы;
2. Изменение структурной организации СОД.

Первый путь - самый простой, "неинтеллектуальный", но он имеет ряд ограничений: существующие **технологии** и **стоимость.** Рост тактовой частоты и скорости передачи информации сдерживаются физическими параметрами

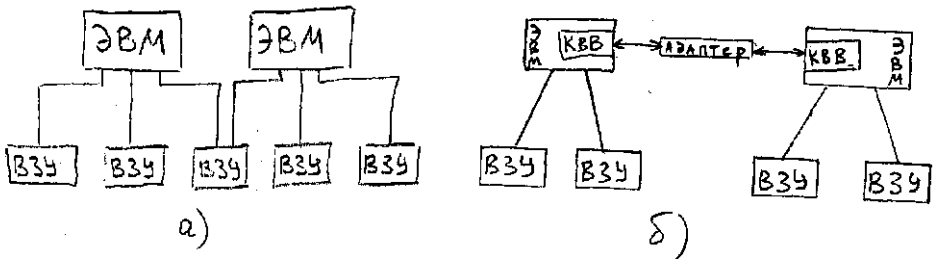
материалов. Не зря сейчас ведутся активные исследования в области работы со светом.

Второй - наиболее эффективный и широкий - состоит из нескольких "шагов":

- совмещение во времени выполняемых операций;
- распространение принципа параллельной обработки на сами устройства обработки информации;
- создание ММВС и МПВС;
- параллельная реализация нескольких программ и нескольких задач.

Первые попытки создания ММВК и МПВК и систем относятся к 60-м годам.

Для повышения надёжности и производительности СОД несколько ЭВМ связывались между собой, образуя простейший **многом,,,»,.,,,,...* вычислительный комплекс**

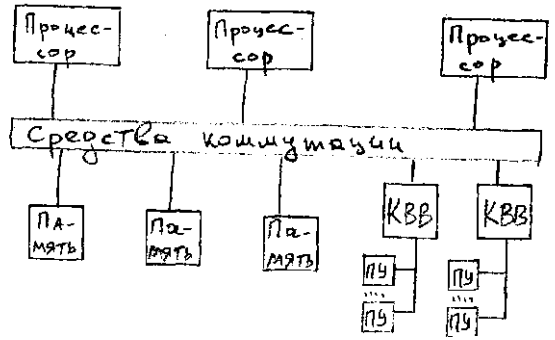



По типу межмашинной связи они делятся на несколько видов:

а) с косвенной связью;

б) с прямой связью. *Г?*

**Вычислительный комплекс,** содержащий несколько процессоров с общей памятью и периферийными устройствами, называется **многопропессорным.**




В настоящее время термин "комплекс" практически не используется, его полностью вытеснил термин "система". Они очень близки по смыслу, но различие есть:

**Вычислительный комплекс** - совокупность технических средств, включающих несколько ЭВМ или процессоров, и *общесистемного* ПО.

**Вычислительная система** - СОД, настроенная на решение задач конкретной области применения. Включает в себя технические средства и *специальное* ПО.

Можно сказать, что **ВС** - проблемно-ориентированный **ВК.** Существуют два способа ориентации ВС:

1. **ВС=стандартный ВК+спецПО,** ориентированное на решение определённой совокупности задач;

2. Ориентация на заданный класс задач достигается за счёт использования специализированных ЭВМ и ВК. В этом случае возможно уменьшение оборудования и увеличение быстродействия, но расходуются дополнительные средства на разработку нового оборудования и ОС.

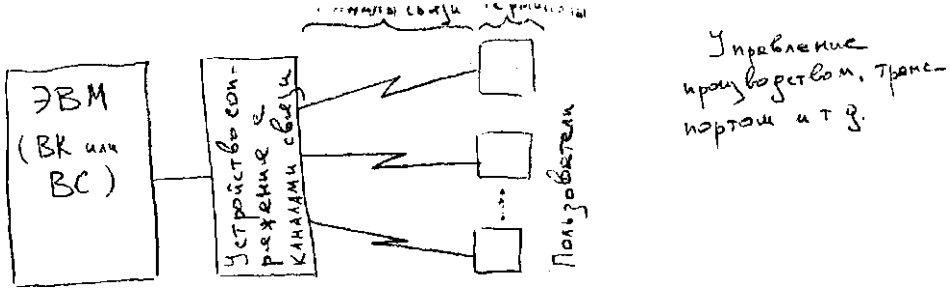
Специализированные ВС второго типа широко используются при решении задач векторной и матричной алгебры, решении дифуров, обработке изображений, распознавании образов и т.п.

Отдельный тип систем образуют **СИСТЕМЫ ТЕЛЕОБРАБОТКИ ДАННЫХ.**

При использовании СОД для управления производством, технологическим процессом, транспортом и т.п. эффективность систем может быть значительно повышена, если ввод информации обеспечивать непосредственно с места её появления и выдачу результатов непосредственно к месту их использования. Для этого рабочие места пользователей и СОД связываются *каналами связи* (КС).

Системы, предназначенные для обработки данных, передаваемых по КС, называются **системами перУ^ачи данных.**

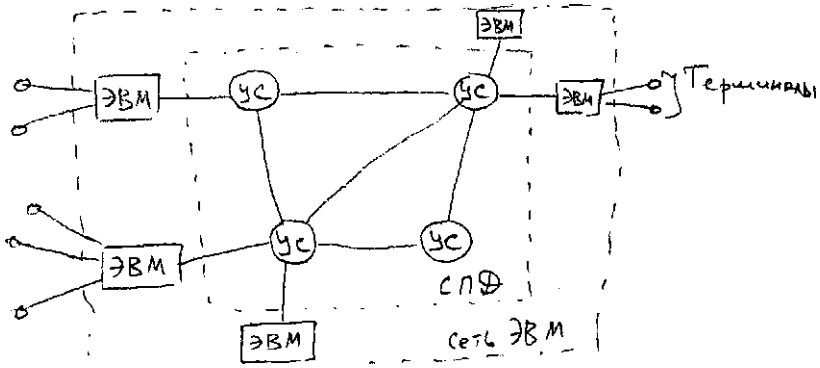
Отличительная особенность - сообщения кроме данных несут ещё и служебную информацию для управления и коррекции ошибок. В ПО добавлены специальные средства для передачи данных и организации доступа к ним.




Однако практика (рост использования ВТ, рост объёмов обрабатываемой информации) требует объединения СОД и перехода от использования отдельных ЭВМ к обработке информации в распределённых по большой территории многомашинных ассоциациях. В такой обработке может участвовать большое число взаимосвязанных машин, находящихся на большом расстоянии друг от друга соединённых между собой множеством физических каналов.

Выходом является построение **вычислительной Гили компьютерной) сети** *(ВО* - совокупности ЭВМ, объединённых *сетью передачи данных* (СПД).

Ядром любой ВС является СПД, состоящая из каналов и узлов связи (УС).




УС обеспечивают качественный приём и передачу данных по назначению. К УС могут быть подключены несколько ЭВМ. Совокупность всех ЭВМ, подключенных ко всем Э***Щ** называется *сетью ЭВМ*. К каждой ЭВМ м.б. подключены несколько терминалов, образующих в комплексе *терминальную сеть*.

Т.о. вычислительная сеть состоит из:

- сети передачи данных;
- сети ЭВМ;
- терминальной сети.

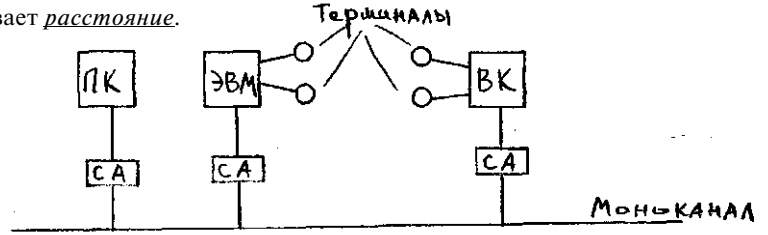
Естественно, для функционирования сети создаётся специальное ПО, установленное на каждой машине.

В настоящее время вычислительная сеть - наиболее эффективный способ построения крупномасштабной СОД.

Частным случаем вычислительной сети является **локальная вычислительная сеть ГЛВО.**

ЛВС - совокупность *близкорасположенных* персональных ЭВМ, связанных *последовательными интерфейсами* и оснащёнными ПО, обеспечивающим обмен информацией между процессами в разных ЭВМ по *моноканалу*.

Очень удобная конфигурация и~ достаточно простое оборудование и ПО, обусловленное именно моноканалом. Единственное ограничение на развитие ЛВС накладывает *расстояние*.



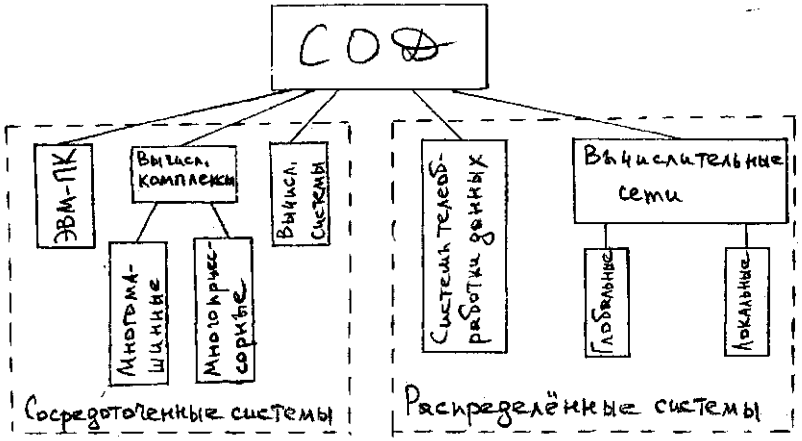

Пропускная способность моноканала (витая пара, коаксиальный кабель, оптоволокно) не превышает $10^7$-$10^9$ бит/сек.

**Классификация СОД.**

Удобнее всего рассмотренные СОД классифицировать по способу построения. Они делятся на:

- сосредоточенные (централизованные) и
- распределённые.

При этом системы телеобработки лишь условно можно отнести к распределённым, т.к. собственно обработка данных производится централизованно, в одном месте, а каналы связи и терминалы обеспечивают лишь ввод-вывод данных.

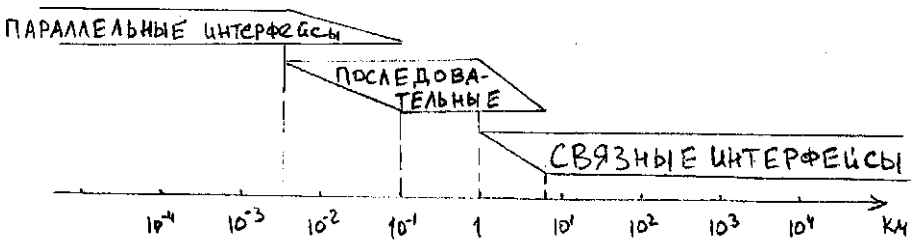



**Глобальные типы интерфейсов, используемые в СОД.**

Интерфейс определяет количество линий для передачи сигналов и данных и способ (алгоритм) передачи информации по линиям связи. Все интерфейсы, используемые в ВТ относят к трём классам:

- *параллельные* - 8-128 разрядов со скоростью до 10 бит/сек;
- *последовательные* - одна линия со скоростью до $10^9$ бит/сек;
- *связные* - одна линия со скоростью до $10^5$ бит/сек.

Расстояния, на которых целесообразно использовать тот или иной интерфейс, обозначены на рисунке.



# ТЕМА 2.

## Режимы обработки данных. Способы параллельной обработки. Параллелизм.

### Режимы обработки данных.

**Режим обработки данных** - способ выполнения заданий (задач), характеризующийся порядком распределения ресурсов системы между заданиями. Обеспечивается управляющими программами ОС, которые выделяют память и другие ресурсы.

Рассмотрим основные режимы обработки данных и их влияние на характеристики СОД.

**Оперативная и пакетная обработка данных.**

Эти два режима используются для обслуживания пользователей, а не технических объектов.

**Оперативная** обработка характеризуется:

- малым объёмом вводимых-выводимых данных на один запрос и
- высокой интенсивностью взаимодействия - малым временем ответа.

В рамках оперативной различают ещё два подтипа - режим "запрос-ответ" и диалоговый. "Запрос-ответ" - справочная служба на основе ЭВМ, банковские операции. Диалоговый предполагает активное участие пользователя в управлении процессом. Он самый интенсивный.

**Пакетная** обработка данных характеризуется:

- большим объёмом вводимых-выводимых данных на один запрос и
- более низкой интенсивностью взаимодействия - возможно большое время ответа.

Используется чаще всего для обработки больших массивов статистических или технических данных.

**Обработка в реальном масштабе времени (РМВ).**

Используется в основном в системах управления реальными (в первую очередь техническими) объектами.

Важный фактор - наличие предельно допустимого времени на получение ответа по какой-то задаче.

РМВ - режим, при котором организация обработки данных подчиняется темпу процессов вне СОД.

**Режим телеобработки данных.**

Самая важная отличительная черта - наличие *линий связи* и необходимости *подключения и отключения* от них. Пользователю нужно доступиться к системе, а системе - к данным. После этого идёт работа в режимах "запрос-ответ", диалоговом или пакетном.

**Мультипрограммная обработка.**

Это способ организации вычислительного процесса, при котором на одном процессоре *попеременно* выполняются сразу несколько программ. Эти программы совместно используют не только процессор, но и *другие* ресурсы: ОП, ВЗУ, УВВ, ПУ. Задачи реализуются *на разных устройствах,* способных функционировать параллельно.

Наиболее реальная ситуация - совмещается по времени работа процессора по одной программе с работой УВВ по другим.

Мультипрограммирование призвано повысить эффективность использования системных ресурсов, однако эффективность можно понимать по-разному. Наиболее характерными критериями эффективности ВС являются:

- пропускная способность - кол-во задач, выполняемых ВС в единицу времени;
- удобство работы пользователей - они имеют возможность работать с несколькими приложениями на одной машине;
- реактивность системы - способность выдерживать заранее заданные интервалы времени (иногда - очень малые!) между запуском программы и получением результата.

При использовании мультипрограммирования 1-й и 2-й критерии имеют положительную динамику, 3-й - скорее отрицательную.

Самым эффективным способом повышения надёжности и производительности системы является параллельная обработка данных. Первой попыткой использовать этот способ была мультипрограммная обработка, но и она не справилась с растущими задачами. Качественно новой идеей была

**Мультипроцессорная обработка** (или мультипроцессорный режим обработки) данных.

Это такой способ организации вычислительного процесса в системах с несколькими процессорами, при котором несколько задач (процессов, потоков) могут одновременно выполняться на разных процессорах системы.

Эта концепция известна с начала 70-х, но до середины 80-х доступных многопроцессорных систем не существовало. Однако к настоящему времени стало обычным включение нескольких процессоров в архитектуру даже ПК (пример - сопроцессор). Более того, многопроцессорность теперь является одним из необходимых требований, предъявляемых к компьютерам, используемым в качестве центрального сервера более-менее крупной сети.

*Не путать мультипрограммную и мультипроцессорную обработки!*

В мультипрограммных системах параллельная работа разных устройств позволяет одновременно вести обработку разных программ, но при этом в процессоре в каждый момент времени выполняется только *одна* программа, т.е. несколько задач выполняются попеременно но *одном* процессоре. Здесь лишь *видимость* параллелизма.

В мультипроцессорных системах *несколько* задач выполняются одновременно, т.к. имеется *несколько* обрабатывающих устройств - процессоров.

Мультипроцессорная организация системы приводит к усложнению всех алгоритмов управления ресурсами. Планировать задачи для нескольких процессоров значительно сложнее, чем для одного. Кроме того, кто именно из процессоров будет это делать? Есть и другие сложности - возникают конфликтные ситуации при обращении к УВВ, ОП, памяти данных и т.д.

ПО

Всё это решается СО путём синхронизации процессов, введения очередей, при помощи системы приоритетов (процессоров или процессов) и планирования ресурсов, но всё предусмотреть непросто.

В наши дни общепринято вводить в ОС функции поддержки мультипроцессорной обработки данных. Такие функции имеются во всех популярных ОС:

Microsoft Windows NT;

Novell NetWare 4.x;

IBM OS/2;

Sun Solaris 2.x;

Santa Cruz Operations Open Server 3.x.

**Понятия процесс и поток.**

Чтобы поддерживать мультипрограммную и мультипроцессорную обработки, ОС должна определить и оформить для себя те внутренние единицы работы, между которыми будут распределяться процессоры и другие ресурсы.

В настоящее время в большинстве ОС определены два типа единиц работы. Более крупная - *процесс или задача,* требует для своего исполнения несколько более мелких работ, для обозначения которых используют понятие *поток*.

**Способы организации параллельной обработки** и **Виды параллелизма.**

Существуют три основных способа:

- **совмещение во времени различных этапов разных задач;**
- **одновременное решение различных задач или частей одной задачи;**
- **конвейерная обработка информации.**

Рассмотрим каждый из них отдельно.

^ *Первый путь* - не что иное, как известная нам мультипрограммная обработка данных. Она возможна даже в однопроцессорных ЭВМ. Широко используется.

*Второй путь* - одновременное решение различных задач или частей одной задачи - возможен только при наличии нескольких обрабатывающих устройств. Этот путь нам тоже известен, мы кратко рассматривали мультипроцессорную обработку данных.

Но мультипроцессорный режим - всего лишь режим. Он должен быть обеспечен средствами ОС, а они, в свою очередь, строятся на идеях параллелизма, точнее, на элементах параллелизма, присущих задачам.

Можно выделить **несколько видов параллелизма:**

1. ***Естественный параллелизм независимых задач.*** Он заключается в том, что в систему поступает непрерывный поток не связанных между собой задач, т.е. решение любой задачи не зависит от результатов решения других задач. В этом случае использование нескольких процессоров однозначно повышает производительность.

2. ***Параллелизм независимых ветвей.*** Имеет очень широкое реальное распространение. Заключается в том, что при решении большой задачи могут быть выделены отдельные независимые части - ветви программы, которые при наличии нескольких процессоров могут выполняться параллельно и независимо друг от друга.

Двумя независимыми ветвями будем считать части, для которых выполняются следующие условия:

- ни одна из входных величин для ветви не является выходной для другой ветви (отсутствие функциональных связей);
- обе ветви не производят запись в одни и те же ячейки памяти (отсутствие связи по использованию ОП);
- условия выполнения одной ветви не зависят от результатов или признаков, полученных при выполнении другой ветви (независимость по управлению);
- обе ветви должны выполняться по разным блокам программы (программная независимость).

На самом деле задача обнаружения и реализации параллелизма независимых ветвей - очень сложная и объёмная. Фактически - это математическая задача сетевого планирования. С ней вы столкнётесь в курсе "ОПУП" на классическом примере "Утро на даче". Особенная сложность - неизвестность длительности

исполнения каждой из множества ветвей. На практике невозможно избежать простоев в работе процессоров, мы лишь минимизируем их.

3. ***Параллелизм объектов или данных.*** Имеет место тогда, когда по одной и той же программе обрабатывается некая совокупность данных, поступающая в систему одновременно. Примеры: данные от РЛС - все сигналы обрабатываются по одной и той же программе, задачи векторной алгебры, в которых выполняются одинаковые операции над парами чисел двух аналогичных объектов.

Забегая вперёд можно сказать, что этот вид параллелизма служит базой для построения матричных систем.

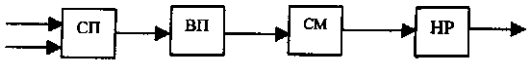
*Третий путь* параллельной обработки - конвейерная обработка информации.

Его реализация возможна даже в системе с одним процессором, разделённым на несколько последовательно включенных операционных блоков, каждый из которых ориентирован на некоторые строго определённые части операции.

Формулу работы конвейерного процессора можно описать так: *"Когда i-и операционный блок выполняет i-ю часть j-ой операции, (i-l)-и операционный блок выполняет (i-lj-ю часть (j+l)-ои операции, а (i+l)-и операционный блок выполняет (i+lj-ю часть (j-l)-ои операции"*

Работу конвейера мо^но легко продемонстрировать на примере операции сложения двух чисел с плавающей запятой, она выполняется за 4-ре шага:

- сравнение порядков (СП);
- выравнивание порядков (ВП);
- сложение мантисс (СМ);
- нормализация результата (НР);




| Этап | 1 | 2 | 3 | 4 | 5 | « | | i | | n | n+1 | n+2 | n+3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| СП | a1b1 | a2b2 | a3b3 | a4b4 | a5b5 | абьб | | aibi | | anbn | | | |
| ВП | | a1b1 | a2b2 | a3b3 | a4b4 | a5b5 | | | | •M)>Vi) | anbn | | |
| СМ | | | a1b1 | a2b2 | a3b3 | a4b4 | | | | | | anbn | |
| НР | | | | cl | c2 | c3 | | | | | <V2> | | cn |

Т.о., если такты одинаковые по длине, первый результат будет получен через-четыре-такта, но затем в каждом такте мы будем получать результат! Наш выигрыш тем больше, *чем длиннее цепочка данных и чем большее количество операционных блоков*.

Только что мы рассмотрели конвейер арифметических операций. Но ведь по такому же принципу мы можем организовать и конвейер команд, которые уже давно используется практически во всех ЭВМ.

Цикл команды разбивается на ряд этапов, которые могут выполняться независимо друг от друга:

- формирование адреса команды (ФА);
- выборка команды из памяти (ВК);
- расшифровка кода операции (РКО);
- формирование адреса операнда (ФАО);
- выборка операнда из памяти (ВО);
- арифметическая или логическая операция (АЛО);

| Этап | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| ФА | K, | **к2** | K3 | K, | **к5** | **к«** | **к7** |
| ВК | | Ki | **к2** | K3 | **К4** | **к5** | Кб |
| РКО | | | **к,** | **к2** | K3 | **к,** | **к5** |
| ФАО | | | | Ki | **к2** | K3 | **К4** |
| ВО | | | | | **к.** | **к2** | K3 |
| АЛО | | | | | | **к,** | **к2** |

Т.о. в конвейере арифметических операций мы параллельно обрабатываем *m* пар операндов, а в конвейере команд совмещаем во времени / операций.

К сожалению, выиграть в / раз практически невозможно из-за операций условных переходов, нарушающих работу конвейера. В ОС существуют специальные средства, позволяющие определить переходы как можно раньше и уменьшить их влияние.

В вычислительных системах используют и конвейер арифметических операций, и конвейер команд, причём по несколько штук, работающих параллельно.

## ТЕМА 3.

### Системы параллельной обработки данных. Классификация.

**Параллельные ВС** - системы, которые обладают 2-мя или более процессорами, АЛУ и т.п., или возможностью распараллеливания на уровне устройств управления.

Цели создания параллельных ВС :

- повышение производительности;
- повышение надёжности;
- повышение живучести;
- упрощение каналов передачи данных.

**Надежность** - свойство объекта сохранять во времени способность выполнять требуемые функции в заданных условиях эксплуатации и технического обслуживания.

**Живучесть** - возможность выполнения основных функций (но не всех в полном объёме) при выходе из строя блоков или устройств системы. **Деградация.**

**Производительность** - характеристика мощности системы, определяющая количество вычислительной работы, выполняемой в единицу времени. Мерой производительности является быстродействие - число операций, выполняемых ЭВМ и устройствами в секунду.

Параллелизм иногда рассматривают как редкую и экзотическую область ВТ, интересную, но не имеющую практической области применения и не используемую обычными программистами. Однако изучение тенденций развития современных программных приложений, архитектуры ЭМ и вычислительных сетей показывает, что Параллелизм является в настоящее время вездесущим, а параллельное программирование становится основным для разработчиков ПО для современных вычислительных средств.

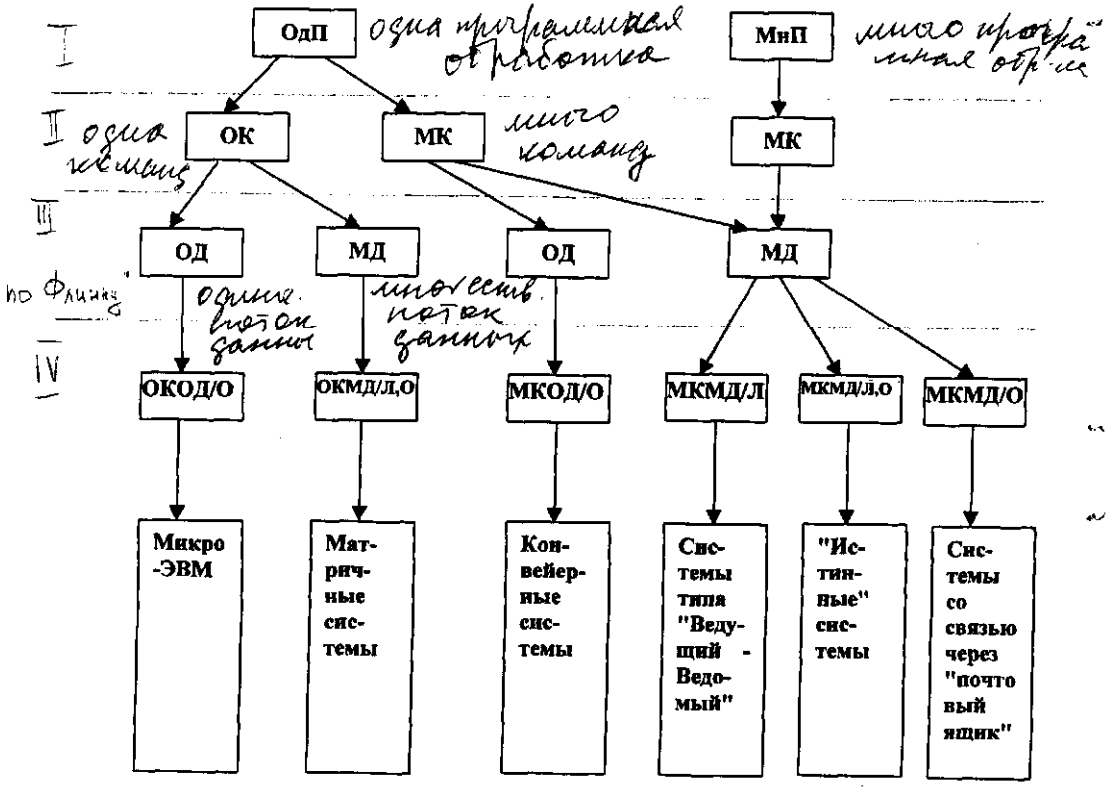
### Классификация систем параллельной обработки (по М.Дж. Флинну).

Процесс решения задачи можно представить как воздействие определённой последовательности команд программы (потока команд) на соответствующую

последовательность данных (поток данных), вызываемый этой последовательностью команд. Тогда разные способы организации параллельной обработки информации можно представить как способы организации воздействия одного или нескольких потоков команд на один или несколько потоков данных.

Под *множественным потоком команд или данных* будем понимать наличие в системе нескольких последовательностей команд, находящихся в стадии реализации, или нескольких последовательностей данных, обрабатываемых командами.

Если несколько модифицировать классификацию Флинна, выделив в ней 4 уровня, то получим следующую стройную систему:




**1-й уровень** - по числу одновременно выполняемых программ. Чем меньше программ, тем ниже живучесть.

**2-й уровень** - деление по типу потока команд - одиночный или множественный.

**3-й уровень** - по типу потока данных между процессором и ОП. В дальнейшем мы наиболее полно будем рассматривать системы, различаемые именно на этом уровне (собственно классификация Флинна). Русским аббревиатурам ОКОД, ОКМД, МКОД, МКМД соответствуют английские SISD, SIMD, MISD, MIMD (single/multiple instruction/data).

**4-й уровень** - деление по способу организации связи процессора с ОП. Есть три основных типа:

- с общей (глобальной) равнодоступной основной памятью **(/О);**
- с локальной (индивидуальной) основной памятью (/Л);
- с локальной основной и общей вспомогательной памятью (/Л,0)

Каждому из четырёх рассмотренных классов присущи свои способы организации параллельной обработки информации.

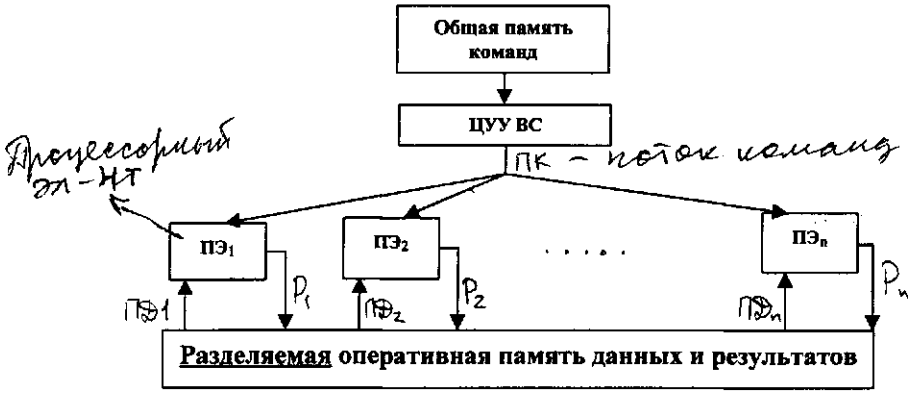
В общем виде архитектуры систем 3-го уровня классификации выглядят следующим образом:

**ОКОД (SISD)**


Такая структура наиболее характерна для однопроцессорных машин.

## ОКМД (SIMD)

Часто называют матричными или векторными системами. Для них характерно выполнение одной и той же команды.




Обрабатывается *одна* программа, система может содержать до тысячи идентичных процессоров, управляемых от *одного* устройства управления.

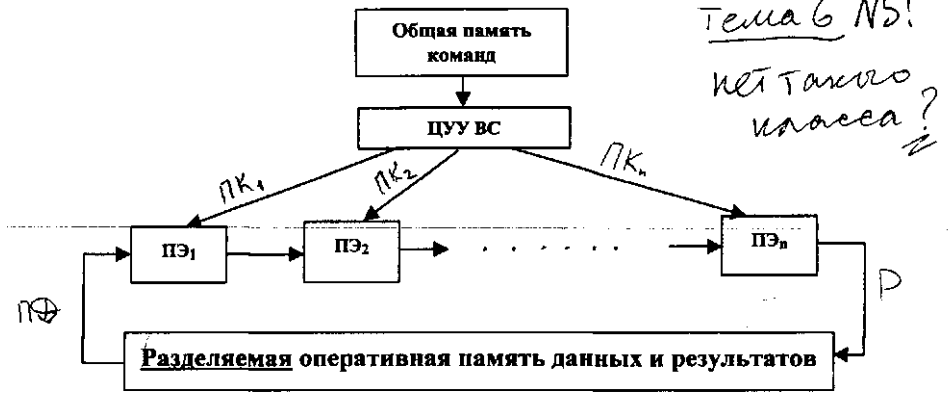
Анализируя систему по живучести и надёжности можно увидеть, что отказ ЦУУ приводит к отказу всей системы. В зависимости от структуры и способов реконфигурации система может быть устойчивой к отказу одного или нескольких процессоров (надёжность), а при выходе из строя большего количества снижать свои параметры (живучесть).

## МКОД (MISD)

Часто называют конвейерными системами или системами с магистральной обработкой информации.

Такая система позволяет с помощью множества потоков команд обрабатывать один поток данных одной программы. Каждый процессор содержит оперативную память программ и выполняет различные команды одной программы. Отказ любого из процессоров может привести к отказу системы (если нет резервирования или не заложена реконфигурация) или её деградации.

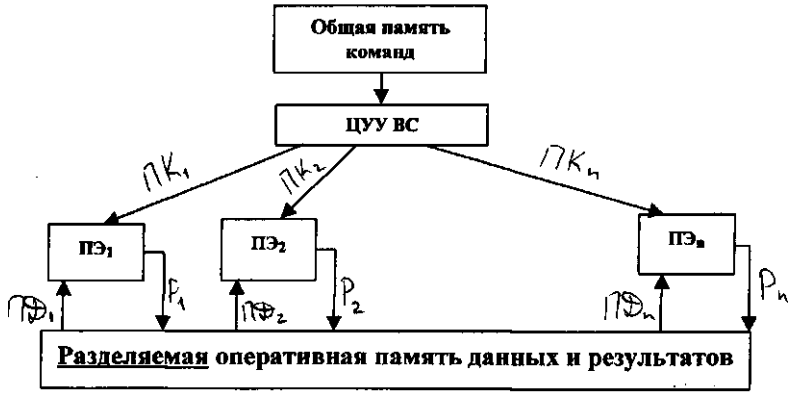





## МКМД (MIMD) - системы.

Существует два основных способа построения таких систем:

- как совокупность элементарных ОКОД *(многомашинная система):*
- и по такой схеме *(многопроцессорная система):*




Такая схема даёт возможность с помощью множества потоков команд одновременно обрабатывать множество потоков данных *нескольких* программ. Такая идея получила наибольшее распространение как в параллельных системах, так и в сетях. В таких структурах легче всего (аппаратно и программно) обеспечить высокую надёжность и живучесть.

**ТЕМА 4.**

**ОКОД (SISD) - системы,**

Это обычные однопроцессорные системы (или ЭВМ).

В современных системах этого класса наиболее широко используется первый путь параллельной обработки - ***совмещение во времени разных этапов решения задач,*** при котором в системе одновременно работают различные устройства: УВВ, обработки информации. ***(Иначе - принцип макросовмещения).***

Введение большого числа *параллельно* работающих периферийных устройств позволяет существенно сократить время на ВВ информации в ОЗУ, уменьшая тем самым время решения задачи и сгладить разрыв между скоростями работы ЦУП, ОП и периферией.

Но при параллельной работе устройств ВВ возникают *конфликты* интересов. Борьба с ними - введение системы приоритетов, устанавливающей очередь запросов. Но очереди вызывают простои. От них в наибольшей степени страдает процессор, как наиболее быстродействующее устройство, способное ожидать сколь угодно долго, в отличие от внешних электромеханических устройств.

Значительный эффект в производительности ОКОД даёт разбиение ОЗУ на *несколько функционально самостоятельных модулей, работающих независимо друг от друга*. Повышение производительности достигается за счёт уменьшения простоев устройств из-за конфликтов при обращении к ОЗУ, т.к. образуются несколько очередей.

Кроме ***совмещения во времени разных этапов решения задач (макросовмещение)*** существенное повышение производительности достигается за счёт *введения конвейерной обработки,* точнее, ***конвейера команд. (Или принцип микросовмещения).***

***Конвейер команд*** - совмещение во времени работы нескольких различных блоков, выполняющих отдельные части общей операции.

- команды и данные размещаются в разных модулях, следовательно, можно параллельно выбирать *и команды, и операнды:*
- и команды, и операнды обычно располагаются и выбираются из памяти из последовательности ячеек *с возрастающими адресами*.

Тогда ОЗУ строится так: все чётные адреса в одном модуле, все нечётные - в другом. В пределе при выполнении программы среднее время обращения к ОЗУ уменьшится в 2 раза.

При разбиении на N среднее время обращения составит 1/ N цикла ОЗУ. Такая память - память с чередованием адресов или память с расслоением обращений.

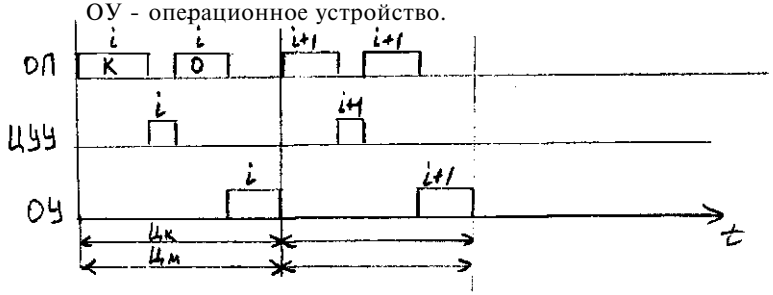
Возможно также построение ***конвейера арифметических и логических операций.***

Рассмотрим на временных диаграммах более подробно *цикл выполнения команды* в ОКОД.

ОП - оперативная память;

ЦУУ - центральное управляющее устройство;

ОУ - операционное устройство.




r~tl- выборка i-ой команды;

r-^- выборка i-ого операнда;

Цк - цикл команды;

Цм - цикл машины.

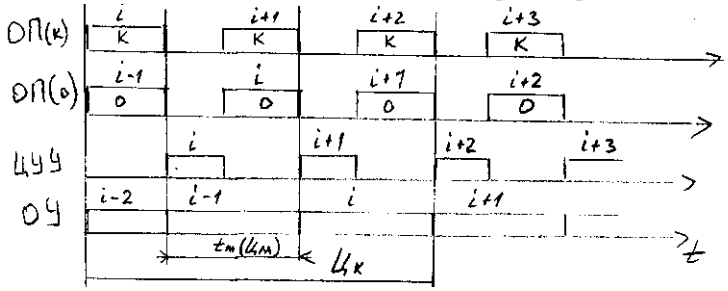
Порядок действий: выбрать команду, дешифрировать её, выбрать операнд и выполнить операцию.

Очевидна неэффективность использования ОУ и ОП и, следовательно, нужно что-то предпринять. Естественным развитием таких машин явилась такая организация, которая позволила *совместить во времени выполнение выборки из памяти и работы ОУ для смежных команд*.


время, **которое Т1спользуется,** остальное время ОУ простаивает.

Во время выборки i'-ой команды имеем возможность выполнить (М)-ю операцию. Время машинного цикла уменьшилось.

Дальнейшее уменьшение машинного цикла связано с выделением блока памяти команд и блока памяти данных. Теперь диаграмма выглядит так:




ОП(К) - позволяет выбирать команды;

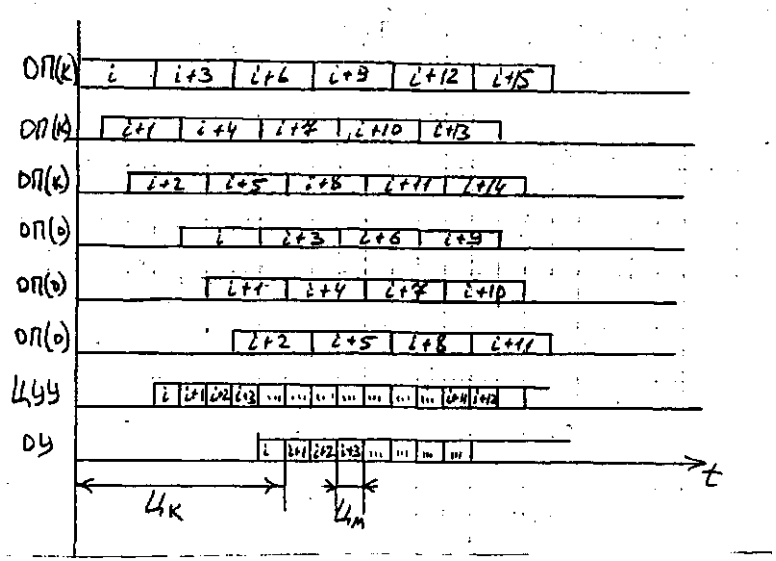
ОП(£|) - позволяет выбирать данные(0w< р « Ь \ ) V

Следующего (последнего) улучшения параметров можно добиться путём *дублирования блоков памяти* как для команд, так и для данных с целью перекрытия циклов обращения к памяти. При этом для ОКОД *максимальным перекрытием циклов* можно считать такое, когда машинный цикл становится равным времени дешифрирования команды.

Пусть $r^w$=1/3 от ***t„„,*** тогда (см. рисунок на следующей странице):

Такое совмещение считают предельным для ОКОД, т.к. дальнейшее совмещение приводит к необходимости распознавания принадлежности управляющих сигналов, вырабатываемых ЦУУ, работающего с перекрытием к той или иной команде при обработке операндов. Для этого требуется дополнительные

аппаратные затраты, которые с увеличением уровня перекрытия возрастают по показательному закону.




Рассмотренные системы с равномерным перекрытием цикла называются ***конфпюэнтнпми системами***. Такие системы достаточно эффективны при соблюдении условий конфлюэнтности, т.е. при отсутствии зависимости і-ой команды от результатов, полученных в *(i-l)-ou* команде, и при отсутствии ветвлений при выполнении команд.

## ТЕМА 5.

### ОКМД (SIMP) - системы,

Системы этого класса ориентированы на использование параллелизма объектов или данных для повышения производительности. В этой системе по одной и той же (или почти по одной и той же) программе обрабатывается несколько потоков данных, каждый из которых обрабатывается своим ПЭ, работающим под общим управлением, за счёт чего и достигается высокая производительность системы.

Организация систем этого типа на первый взгляд очень проста: общее управляющее устройство, генерирующее поток команд, и большое число устройств, работающих параллельно и обрабатывающих каждое свой поток данных. Т.о. производительность системы оказывается равной сумме производительностей всех обрабатывающих устройств.

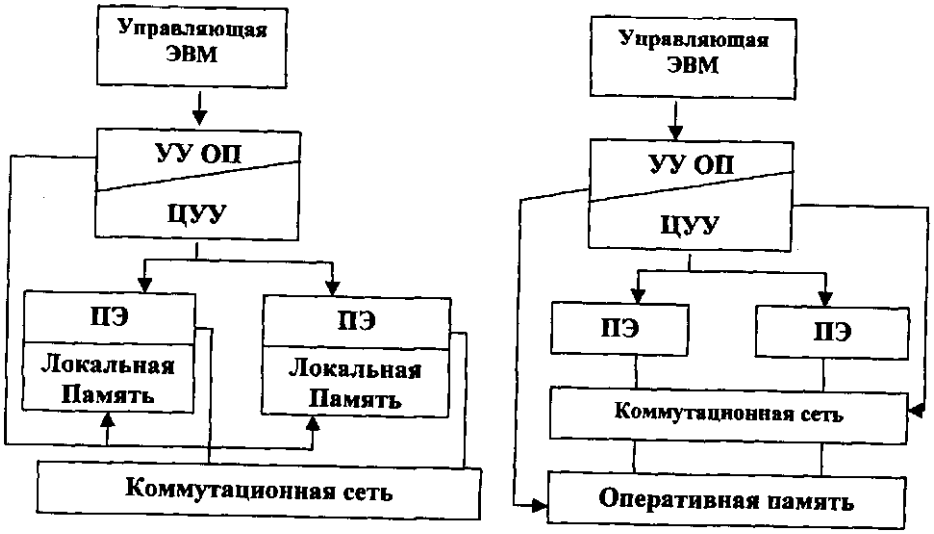
На практике решение задачи распараллеливается в рамках одной команды, т.е. используется, в основном, для матричных или векторных операций. Но в то же время в любой программе наряду с векторными операциями существует достаточно много скалярных, и чем их больше, тем ***ниже*** эффективность использования такой системы.

Системы этого класса часто называют специализированными, в лучшем случае проблемно-ориентированными. С их помощью, конечно, можно решать и другие задачи, но это будет неэффективно.

/ "^Наиболее сложными в системах класса **S1MD** являются вопросы реализации коммутации, именно от их эффективности зависит эффективность системы в целом.

Возможны 2 варианта коммутации:

- с локальной памятью и коммутацией на уровне ПЭ
- с общей памятью и коммутацией на уровне ПЭ и блоков ОП. В этом случае взаимодействие ПЭ происходит через ОП.

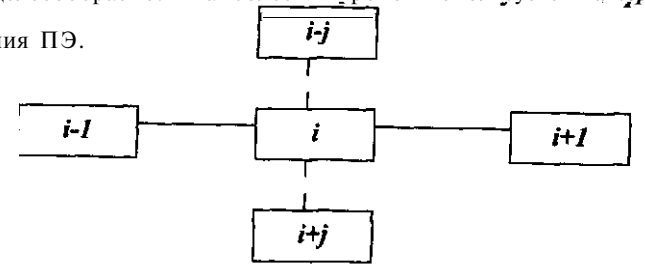



Второй вариант кажется более привлекательным, однако решать задачу перекоммутации ОП с ПЭ оказывается намного сложнее, чем задачу перекоммутации ПЭ.

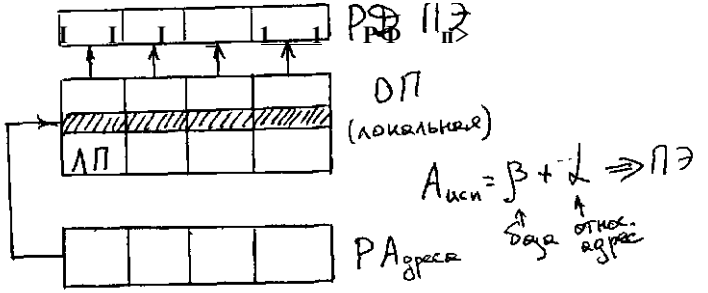
Как следствие, первый вариант нашёл наиболее широкое применение. В этом случае поток данных в каждый процессор поступает из собственного локального ОЗУ. Управление и память команд реализуются отдельной ЭВМ, управляющей ансамблем процессоров.

Следует учитывать, что количество ПЭ в системах, построенных по первой схеме, достаточно большое, и нужно минимизировать коммутационную сеть.

Чаще всего такие структуры базируются на коммутации соседних элементов, т.к. решать задачу взаимодействия «каждый с каждым» для ПЭ технически сложно, да и нецелесообразно. Наиболее широко используется ***итеративный*** способ подключения ПЭ.




'Вопросы управления выполнением операции решаются в ЦУУ (включая вопросы адресации и формирования исполнительных адресов). Это означает, что одна и та же команда, формируемая в ЦУУ, распространяется на все ПЭ. Одни и те же операции происходят во ***всех*** ПЭ над данными, которые выбираются по одним и тем же адресам из индивидуальных блоков оперативной памяти. Это далеко не всегда является приемлемым. Были разработаны механизмы (многомодальность), позволяющие избегать обязательного выполнения команды каким-либо процессором.

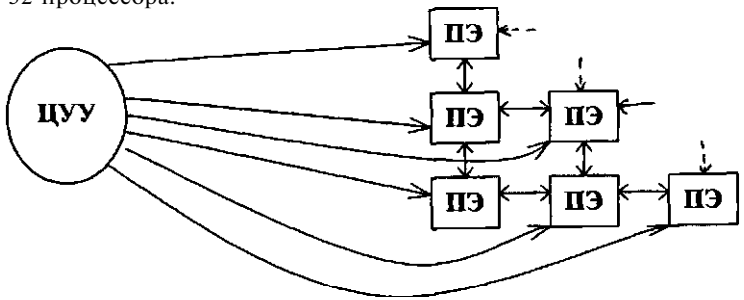



Локальная память данных может иметь не только адресную выборку, но и ассоциативную, т.е. по содержимому памяти. Поэтому все системы класса ОКМД можно глобально разделить на два класса:

- ***матричные;***
- ***ассоциативные***

**Система (машина) Унгера. 1958 год.**

32 х 32 процессора.




Узкоспециализированная и предназначена только для задач распознавания образов (для обработки изображений в реальном масштабе времени). Обладает элементарными логическими возможностями.

ЦУУ обладает большой ОП. Взаимодействие между ПЭ может осуществляться как через коммутационную сеть, так и через ОП ЦУУ. Как правило, ЦУУ - это большая ЭВМ.

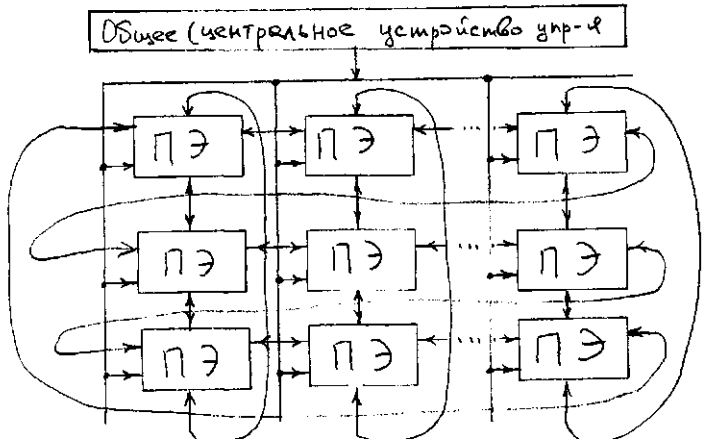
*Недостатком* этой системы явился тот факт, что и формирование и распаковка команд осуществляется только в ЦУУ и *каждая* команда является обязательной для выполнения в *каждом* ПЭ.

### Система SOLOMON.

Стала первой в полном смысле слова матричной системой. Содержит 1024 ПЭ, соединённых в виде матрицы 32x32. Каждый ПЭ в матрице соединён с четырьмя соседними и включает в себя процессор, обеспечивающий выполнение последовательных поразрядных арифметических и логических операций, а также оперативное ЗУ емкостью 16 Кбит, разбитое на модули по 4 Кбит каждый. Разрядность слов устанавливается программно. По каналам связи от УУ передаются команды и общие константы. В ПЭ используется так называемая многомодальная логика, которая позволяет каждому ПЭ выполнять (т.е. быть активным) или не выполнять (быть пассивным) общую операцию в зависимости от значений обрабатываемых данных.

В каждый момент все активные ПЭ выполняют одну и ту же операцию над данными, хранящимися в собственной памяти и имеющими один и тот же адрес. Идея многомодальное™ заключается в том, что в каждом ПЭ имеется специальный регистр на четыре состояния - *регистр моды*. Мода (или модальность) заносится в этот регистр от УУ. При выполнении последовательности команд модальность передаётся в коде операции и сравнивается с содержимым регистра моды. Если есть совпадение, то операция выполняется. В других случаях ПЭ не выполняет операцию, но может в зависимости от кода пересылать свои операнды соседнему ПЭ. Такой механизм позволяет выделить строку или столбец ПЭ, что может быть очень полезным при выполнении операций над матрицами.

Взаимодействуют ПЭ с периферийными устройствами через внешние ПЭ.



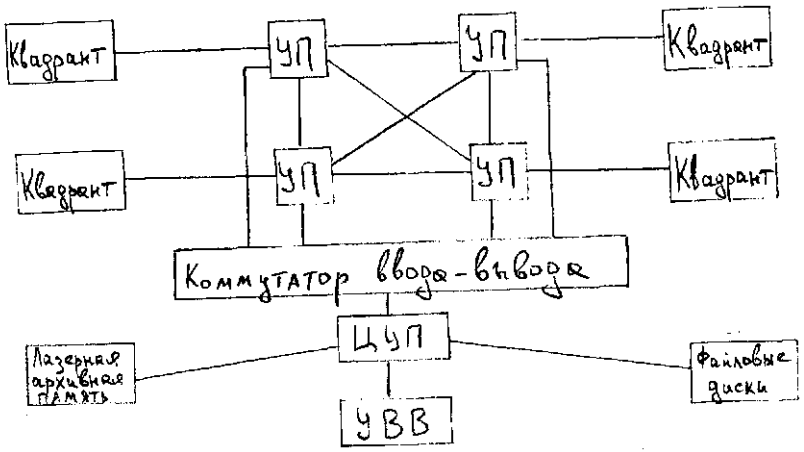

Система **SOLOMON** (как **hSOLOMON II** впоследствии) оказалась нежизнеспособной из-за громоздкости, недостаточной гибкости и эффективности. Однако идеи, заложенные в ней, получили развитие в системе **ILLIAC-IV,** разработанной Иллинойским университетом и изготовленной фирмой «Барроуз».

### ILLIAC-IV

Общая структура аналогична структуре системы **SOLOMON.** Но мощность ПЭ увеличена, и сами они стали 64-х разрядными (а не 32-х).

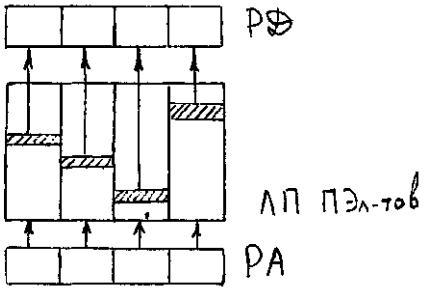
Эта система задумывалась как система класса МКМД. По первоначальному проекту она должна была включать в себя 256 ПЭ, разбитых на 4 группы - квадранты, каждый из которых должен был управляться специальным процессором (УП). Соответственно было бы 4 потока команд. Управление всей системой, содержащей кроме УП и ПЭ также внешнюю память и аппаратуру ВВ, предполагалось от центрального управляющего процессора (ЦУП). Однако реализовать этот проект не удалось из-за технологических трудностей и высокой стоимости.

В результате система была создана в составе одного квадранта (одиночный поток команд) и одного УП. Производительность, таким образом, получилась также в 5 раз меньше запланированной. Но всё равно она несколько лет была самой быстродействующей системой.




Единственным существенным отличием стало введение *независимой адресации* в различных **ПЭ.** Для этого в состав каждого **ПЭ** был добавлен индивидуальный регистр, в котором м.б. различное значение **Вго.**

Теперь $А^{исп}$=р + а + **Впэ**, где **Вго** имеет разное значение для разных **ПЭ.**




Такой способ адресации памяти особенно важен при операциях матричной алгебры, когда к двумерному массиву необходимо осуществить доступ, как по строкам, так и по столбцам.

По аналогии с регистром моды SOLOMONa **ILLIAC-IV** имеет 8-ми разрядный регистр управления состоянием ПЭ. **В** зависимости от его состояния ПЭ м.б. пассивным, активным, или выполнять ряд пересылочных операций.

Каждый ПЭ состоит из собственно процессора, оперирующего 64-х разрядными числами, и ОЗУ. Если вычисления не требуют полной разрядности, то процессор м.б. разбит на 2шт 32-х разрядных подпроцессора или даже на 8шт 8-ми разрядных. Это позволяет в случае необходимости обрабатывать 64, 2х64=128 или 8х64=512 элементов, что очень важно в векторных вычислениях.

*Система коммутации внутри квадранта, i* -ый ПЭ имеет доступ только к i-ому блоку локальной ОП и не может изменить содержимое ОП другого ПЭ. Однако информация от одного ПЭ к другому м.б. передана через сеть пересылок данных при помощи специальных команд обмена. Специальные регистры пересылок каждого **1**-го ПЭ связаны высокоскоростными линиями обмена с регистрами

Пр°Ч*с<.ар°»

пересылок ближайших р^ги^трпв (с номерами / -1, / +1, / -8 и / +8). При этом нумерация рассматривается как циклическая с переходом от 63 к 0. При такой связи передача данных между любыми двумя ПЭ осуществляется не более чем за 7 шагов **(VoT-**1) а среднее число шагов равно 4.

Из-за высокой сложности системы было принято решение о систематическом контроле ПЭ в автоматическом режиме. **В** случае отказа - останов системы и замена ПЭ.

***Отличительные особенности*** и ***преимущества*** системы **ILLIAC-IV :**

- кроме синхронного выполнения всеми ПЭ потока команд предусмотрено маскирование любых ПЭ;
- обмен информацией между любыми ПЭ не более чем за 7 шагов, наибольшая вероятность - 2-3 шага;
- каждый ПЭ обрабатывает поток 64-х разрядных данных или более одного потока данных меньших форматов в *синхронном* режиме;
- каждый ПЭ может индексировать выборку данных из своей памяти независимо от других ПЭ, что очень важно для многих задач;
- доступ к локальной ОП имеет не только АЛУ ПЭ, но и УУ системы и подсистема **ВВ.** Это фактически связывает каждый ПЭ с внешней средой.

***Недостатками*** системы являются:

- высокая производительность достигается только на определённых типах задач, таких как операции над матрицами, БПФурье, обработка сигналов, где имеет место параллелизм объектов или данных;

- разработка программ довольно дорога, были разработаны специальные языки;
- невысокая надёжность и живучесть (без использования реконфигурации и резервирования).

**ПС-2000 (СССР).**

В начале 80-х (через 10 лет после **ILLIAC-IV)** в Советском Союзе в классе систем ОКМД была создана система «Параллельная Система - 2000», которая м.б. отнесена к матричным. Она была ориентирована на решение задач, имеющих параллелизм данных, независимых ветвей и объектов. Её специализация - обработка геофизической информации, получаемой при поиске нефти и газа, решение задач плазменной кинетики, расчёт устойчивости летательных аппаратов, обработка гидролокационных сигналов и изображений, решение задач в частных производных и т.д.

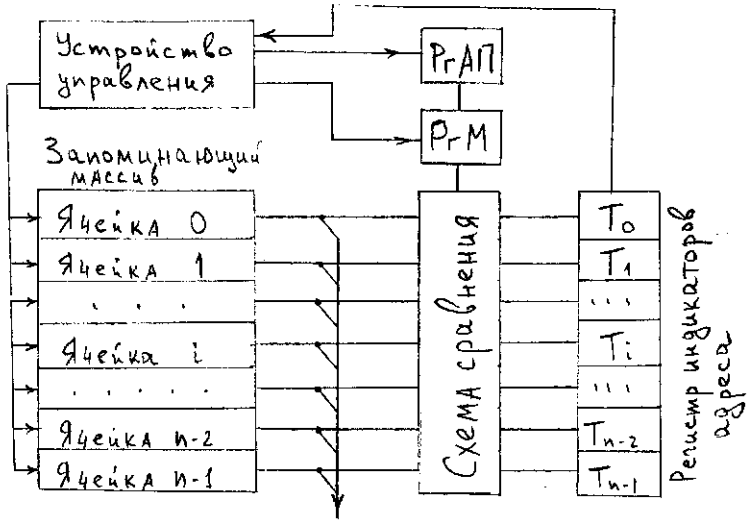
При решении ряда задач геофизики и ядерной физики была получена производительность, превосходящая параметры системы **ILLIAC-IV.**

**Ассоциативные ОКМД (SIMP) - системы.**

Эти системы, так же как и матричные, имеют большое количество операционных устройств, которые могут одновременно обрабатывать *несколько* потоков данных по командам *одного* управляющего устройства.

Основное отличие от матричных заключается в способе формирования потоков данных. В матричных системах данные поступают на обработку от общих или раздельных запоминающих устройств с адресной выборкой информации, либо непосредственно от устройств - источников данных. В ассоциативных системах информация на обработку поступает от ассоциативных ЗУ. В них выборка выполняется не по адресу, а по *содержимому* ячейки.

Рассмотрим схему ассоциативного ЗУ.




Запоминающий массив, как и в адресных ЗУ, разделён на ««-разрядные ячейки, число которых *n*. Практически для любого типа АЗУ характерно наличие следующих элементов: запоминающего массива, регистра ассоциативных признаков (РгАП); регистра маски (РгМ); регистра индикаторов адреса со схемами сравнения на входе.

Выборка информации из АЗУ происходит так: в РгАП из устройства управления передаётся код признака искомой информации. Код может иметь произвольное число разрядов - от 1 до *m*. Если код признаков используется полностью, то он без изменения поступает на схему сравнения, если же необходимо использовать только часть кода, то ненужные разряды маскируются с помощью РгМ. Перед началом поиска информации в АЗУ все разряды регистра индикаторов адреса устанавливаются в 1. После этого производится опрос первого разряда всех ячеек ЗМ и содержимое сравнивается с первым разрядом РгАП. Если содержимое первого разряда /-ой ячейки не совпадает с содержимым первого разряда РгАП, то соответствующий этой ячейке разряд регистра индикаторов адреса $T^i$ сбрасывается в 0, если совпадает - в 1. Затем эта операция повторяется со вторым, третьим и т.д. разрядами до тех пор, пока пройдёт сравнение со всеми разрядами РгАП. После поразрядного опроса и сравнения в состоянии 1 останутся те разряды регистра индикаторов адреса, которые соответствуют ячейкам, содержащим информацию,

совпадающую с записанной в РгАП. Эта информация м.б. считана в той последовательности, которая определяется устройством управления.

! Время поиска информации в АЗМ не зависит от числа ячеек ЗМ, а зависит только от числа разрядов признака и скорости опроса разрядов ! Нет нужды перебирать ***все*** ячейки запоминающего массива!

Запись новой информации в ЗМ производится *без указания номера ячейки,* только по факту свободы ячейки. Признак свободное™ устанавливается в соответствующем разряде РгАП, и в любую из свободных записываем.

Возможно также построение АЗУ т.о., что кроме ассоциативной допускается и прямая адресация данных. Это удобно при работе с ПУ.

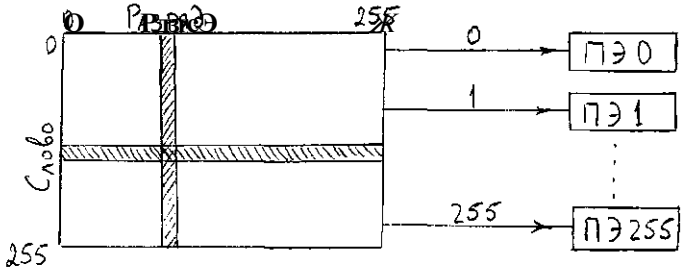
Запоминающие элементы АЗУ могут не только хранить информацию, но и выполнять определённые логические функции, что позволяет делать поиск не только по равенству заданному признаку, но и по другим условиям: >, <, <.

Наиболее эффективно использование ассоциативных систем при решении таких задач, как обработка радиолокационной информации, распознавание образов, обработка снимков и т.д. Важный фактор - упрощение и удешевление программирования.

**Система STARAN**

Это наиболее характерная ассоциативная система

Её ассоциативная память является памятью с многомерным доступом, т.е. в неё можно обратиться как поразрядно, так и пословно. Для каждого слова памяти предусмотрен свой ПЭ.



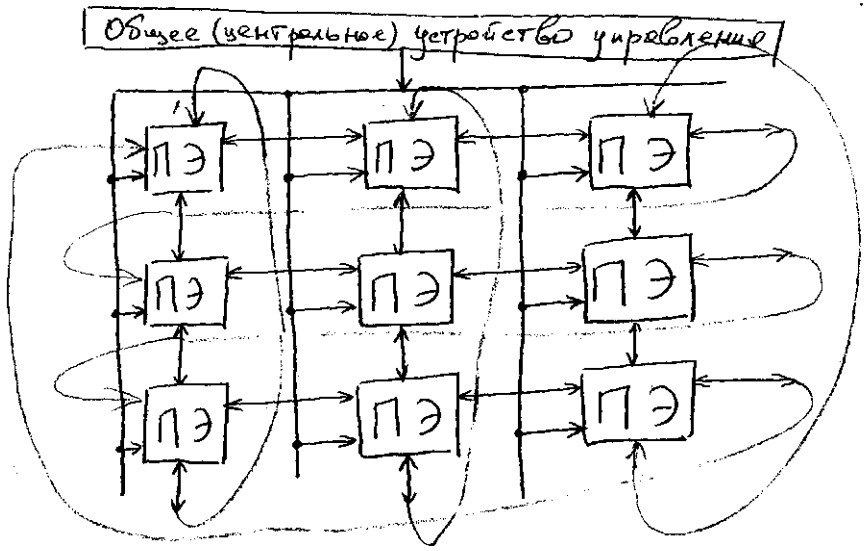

и

Основным элементом системы является многомерная ассоциативная матрица, называемая ассоциатавным модулем (АМ). Он представляет собой квадрат из 256 разрядов на 256 слов, т.е. содержит 256х256=65536 бит данных. Для обработки информации имеется 256 ПЭ, которые последовательно, разряд за разрядом, обрабатывают слова. Все ПЭ работают одновременно, по одной команде, выдаваемой УУ. Т.о. сразу по одной команде обрабатываются все выбранные по определённым признакам из памята слова.

Обычно 256-разрядное слово АМ разбивается программистом на поля переменной длины, и в процессе обработки именно над этими полями производятся арифметические и логические действия.

Базовая конфигурация содержит один АМ. Однако их число может варьироваться от 1 до 32. При максимальной комплектации в системе может подвергаться ассоциативной обработке 256 Кбайт информации (32х256х256=32х65536=32х8Кбайт=256Кбайт).

К системе SOLOMON



## ТЕМА 6.

### МКОД flVtlSD) - системы,

Системы этого класса также ориентированы на использование параллелизма объектов или данных для повышения производительности.

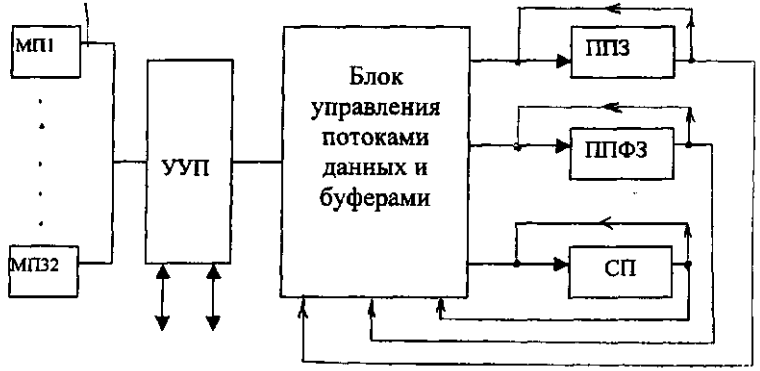
В этих системах один поток команд разделяется устройством управления на несколько потоков микроопераций, каждая из которых реализуется специализированным, настроенным на выполнение именно этой микрооперации, устройством. Одиночный поток данных проходит последовательно через все (или часть) этих специализированных АЛУ. Именно такого класса системы принято называть ***конвейерными*** системами или системами с ***магистральной обработкой*** информации.

В таких системах используется и *конвейер арифметических операций,* и *конвейер команд,* и различные способы совмещения работы многих устройств. Но основной признак - наличие конвейера арифметических и логических операций.

Максимальная производительность достигается только при решении задач, в которых существуют длинные последовательности (цепочки) однотипных операций над достаточно большой (длинной) последовательностью данных, т.е. есть *параллелизм объектов или данных.*

В современных системах, кроме того, используются несколько конвейерных процессоров, способных работать одновременно и независимо друг от друга.

Очень полно и красиво был использован принцип магистральной обработки в системе **8ТА1ЫОО**,\разработанной фирмой CDC в 1973 году.

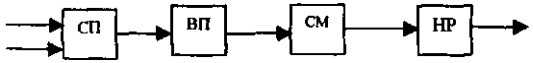


Система содержит 3 конвейерных процессора: **ППЗ** - процессор, содержащий конвейерные устройства сложения и умножения с плавающей запятой; **ППФЗ -** процессор, содержащий конвейерное устройство сложения с плавающей запятой, конвейерное многоцелевое устройство, выполняющее умножение с фиксированной запятой, деление и извлечение квадратного корня; **СП** - специальный конвейерный 16-разрядный процессор, выполняющий операции с ФЗ и ряд логических операций.

*Работу ППФЗ мы в своё время рассматривали. Немного напомню:*

*Работу конвейера можно легко продемонстрировать на примере операции сложения двух чисел с плавающей запятой. Она выполняется за 4-ре шага:*

- *сравнение порядков (СП);*
- *выравнивание порядков (ВП);*
- *сложение мантисс (СМ);*
- *нормализация результата (НР);*




Рассмотрим чуть подробнее. Пусть время выполнения каждого шага равно 60, ^ 100, 140 и 100 не. Тогда операция сложения будет выполняться последовательностью операционных блоков за 400 не. Пусть нужно сложить два вектора по и=100 элементов каждый. Тогда время работы конвейера будет

$$T_{„}=(n+м-\)\%$$

где *т* — число операционных блоков, г - максимальное время выполнения одного из этапов (в нашем случае 140 не).

Если не использовать конвейер, то время выполнения сложения векторов:

$$T^{0}=n$$

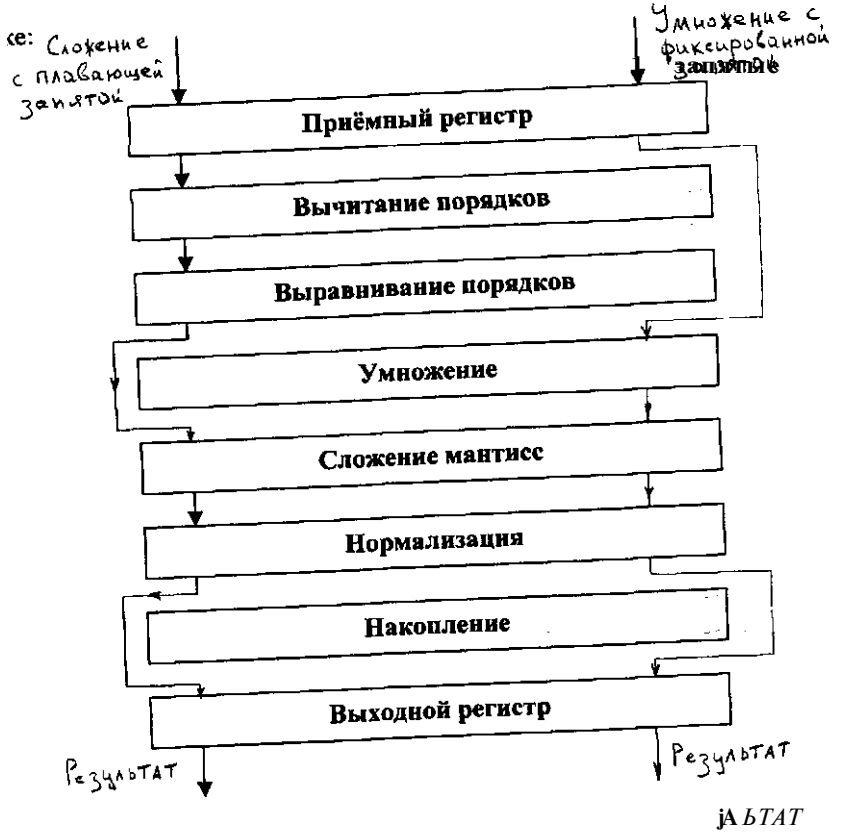
В нашем примере $Г_{к}=(100+4-1)*140=14\ 420$ не, а $T_{в}=100*400=40\ 000$ не.

Легко заметить, что наш выигрыш тем больше, *чем длиннее иепочка данных и чем большее количество операиионных блоков*.

ППЗ разбит на 8 сегментов и результат выдаётся каждые 40 не.

Однако нецелесообразно выполнять устройства конвейерной обработки с жесткой настройкой на одну определённую операцию. Часто их делают многоцелевыми, вводя в конвейер сегменты, которые необходимы для реализации полного набора операций.

процессе выполнения операций так, как показано




jA *ЬТАТ*

В максимальной степени конвейерный принцип обработки информации был использован в системе **CRAY,** долгое время бывшей самой высокопроизводительной системой в мире. Кроме конвейеров обоих типов в системе широко применялась совмещённая обработка информации несколькими устройствами.

Схема системы приведена на отдельной странице.

Система **CRAY** состоит из 4-х секций:

- функциональных устройств;
- регистров;
- управления программой;

- памяти и ввода-вывода.

В системе 12 функциональных устройств, работающих в режиме конвейера, разбитых на 4 группы: адресную, скалярную, операций с ПЗ и векторную. Число сегментов в каждом функциональном устройстве (указано в скобках), невелико, оно зависит от сложности операций и колеблется от 1 до 14 (для вычисления обратной величины). Такое сравнительно небольшое число сегментов в каждом магистральном устройстве имеет определённые преимущества - они сравнительно быстро заполняются. Длительность цикла каждого сегмента 12,5 не, т.е. каждые 12,5 не мы получаем результат от каждого функционального устройства. Система работает с 64-х разрядными словами, 8 разрядов используются для коррекции одиночных и обнаружения двойных ошибок, что обеспечивает высокую *надёжность хранения и передачи информации.*

Существенную роль в достижении высокой производительности играют быстрые регистры. Они разделены на **3** группы: адресные - А-регистры (8шт. по 24 разряда), скалярные - S-регистры (8 шт. по 24 разряда) и векторные - V-регистры (8 шт. 64-элементных, 1 элемент содержит 64 разряда). Есть также ещё 2 группы промежуточных регистров, не показанных на рисунке. Благодаря такой конструкции конвейерные устройства связываются в цепочки и *поток результатов от одного устройства,* проходя через регистр, *становится входным потоком для другого,* ***минуя ОП.***

Состав операций универсальный, только вместо деления используется операция вычисления обратной величины.

Дополнительным средством увеличения производительности явилось высокая интеграция, благодаря чему время передачи сигналов между устройствами очень мало.

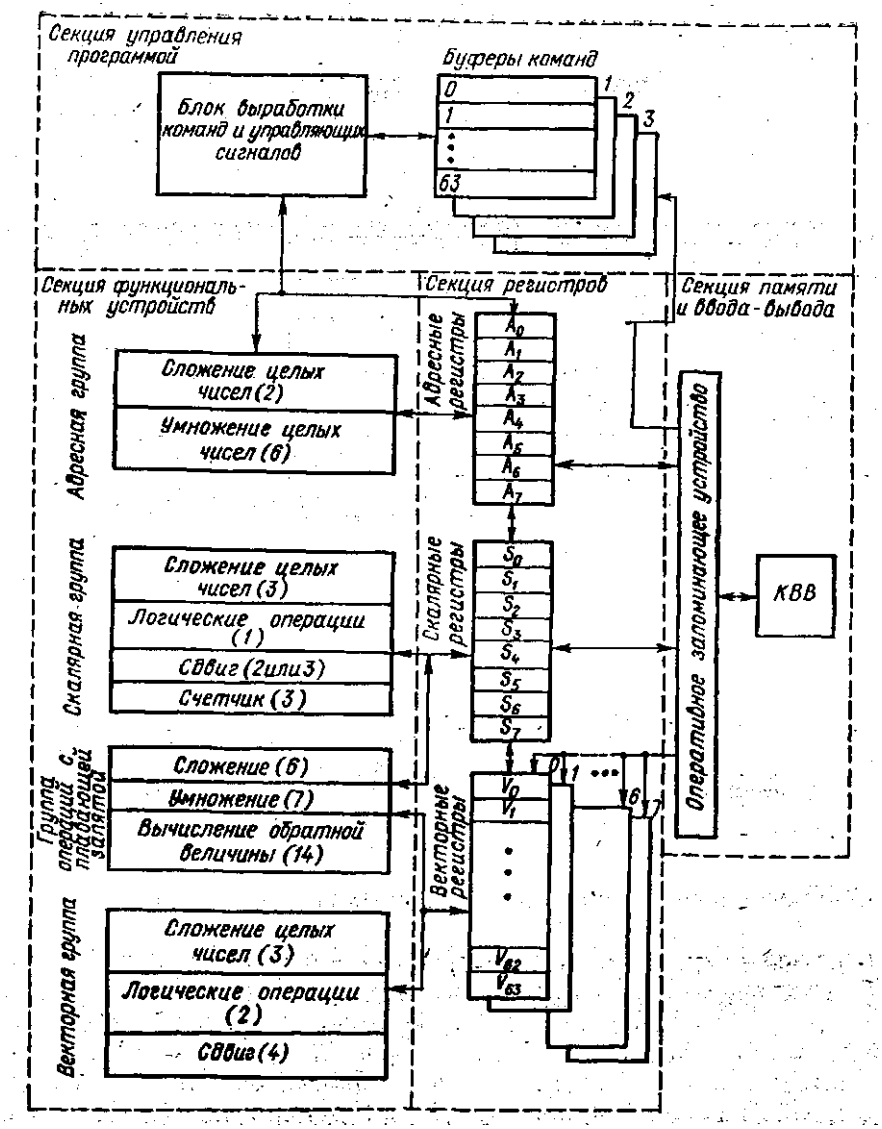



**Рис. 3.3. Система CRAY**

# ТЕМА 7.

## МКМД (МВУПУ) - системы.

В этих системах с помощью множества потоков команд одновременно обрабатывать множество потоков данных *нескольких* программ. Такая идея получила наибольшее распространение как в параллельных системах, так и в сетях. В таких структурах легче всего (аппаратно и программно) обеспечить высокую надёжность и живучесть.

Как я уже говорил ранее, существует два основных способа построения таких систем:

- как совокупность элементарных ОКОД *(многомашинная вычислительная система или ММВС);*
- и как совокупность процессоров, управляемых одним ЦУУ *(многопроцессорная вычислительная система или **МП** ВС):*

# ТЕМА 7-1.

## Многомашинные вычислительные системы (ММВС).

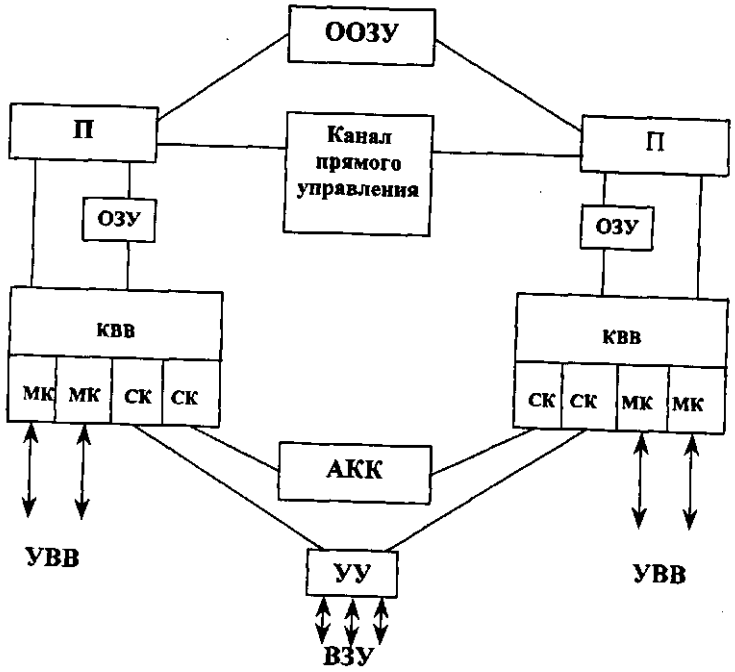
ММВС - система, включающая в себя две или более ЭВМ (каждая из которых имеет процессор, ОЗУ, набор ПУ и работает под управлением собственной ОС), связи между которыми обеспечивают выполнение функций, возложенных на систему. В состав также входит и спецПО, позволяющее ориентировать систему на конкретные типы задач.

В зависимости от достигаемых целей (высокое быстродействие, надёжность, низкая стоимость и т.п.), структура связей между ЭВМ в ММВС может существенно различаться.

По характеру таких связей ММВС можно разделить на три типа:

- косвенно- или слабосвязанные;
- прямосвязанные;
- сателлитные.

В ***косвенно- или слабосвязанных*** системах ЭВМ через ВЗУ.




АКК - адаптер канал-канал. СК - селекторный канал.

Связь только на информационном уровне. Обмен информацией идёт по принципу «почтового ящика» (см. ранее), т.е. каждая ЭВМ помещает в общую внешнюю память данные, руководствуясь только своей собственной программой, и, соответственно, другая ЭВМ принимает эти данные, исходя из своих потребностей.

Такая организация связей обычно используется, когда необходимо повысить надёжность одной ЭВМ путём резервирования. В таком случае возможны три ; алгоритма работы системы:

1. Резервная ЭВМ выключена (ненагруженный или холодный резерв) и включается только при отказе основной.

*Недостатки:* потеря времени и данных, следовательно, возможно • использование такой схемы действий только для управления неответственными» процессами.

*Достоинства:* интенсивность отказов в резервной ЭВМ носит характер отказов при хранении (0,1 от рабочего); низкое энергопотребление; возможность *резервирования с дробной кратностью* (группа из *n* основных элементов резервируется *m* элементами и при отказе одного из основных элементов включается в работу один из резервных, *m<n).*

*2.* Резервная ЭВМ включена и полностью готова к работе, но сама не решает никаких задач, а работает в режиме самоконтроля (облегчённый резерв). Переход в работе от основной к резервной достаточно быстр, но работа пойдёт с некоторой точки возврата в программе, пропадут только последние данные. Такую ММВС можно использовать при управлении медленными процессами.

В случаях 1 и 2 система не была параллельной, вторая ЭВМ использовалась нерационально, всё время простаивала.

3. Все (не обязательно две) ЭВМ решают одновременно одни и те же задачи, результаты сравниваются, но управляет только основная. При расхождении результатов используется принцип голосования по большинству или другой (например: модифицированное дуальное программирование).

В случае 3 также нерационально используются ресурсы системы с т.зр. производительности.

***Прямосвязанные*** ММВС обладают большей гибкостью. В них возможно 3 вида реализации связи: общее ОЗУ (ООЗУ), прямое управление Пр-Пр и связь через адаптер канал-канал (АКК).

Связь через ООЗУ несколько похожа на связь через «почтовый ящик», она тоже носит информационный характер, однако все процессы в системе могут протекать значительно быстрее, а потеря времени при переходе с основной на резервную почти отсутствует. Недостаток - надёжность, сложность и стоимость ООЗУ.

Связь через канал прямого доступа м.б. не только информационной, но и управляющей (+), но передача больших объёмов данных резко понижает производительность (-), т.к. задачи в это время не решаются.

Связь через АКК - объём передаваемых данных м.б. как при ВЗУ, а скорость будет почти как в случае ООЗУ.

Прямо-связанные системы позволяют реализовать все три алгоритма работы системы, рассмотренные выше, однако за счёт усложнения связей эффективность всегда будет значительно выше. В частности, в прямосвязанных системах возможен *быстрый переход* от основной к резервной даже тогда, *когда резервная загружена собственными задачами*. Это позволяет обеспечить высокую надёжность при высокой производительности.

В **реальных** системах используют несколько типов связей между ЭВМ одновременно, в том числе и связь через ВЗУ в прямосвязанных системах.

Для систем с ***сателлитньши*** ЭВМ характерным является не сам способ связи (связь чаще всего реализуется через АКК ), а принцип взаимодействия между ЭВМ. В сателлитных системах 1) ЭВМ отличаются по своим параметрам и 2) имеет место соподчинённость и различие в функциях.

Основная ЭВМ - высокопроизводительная и занимается обработкой информации, вспомогательная (сателлитная) - слабее, она производит сортировку данных, организует передачу данных, связь с ПУ, ВЗУ и т.п. Часто сателлитная ЭВМ имеет явно выраженную ориентацию на какой-то класс операций.

Целесообразно включать в состав системы несколько дешёвых ориентированных сателлитных ЭВМ, «разгружающих» основную ЭВМ и, тем самым, повышающих производительность системы в целом.

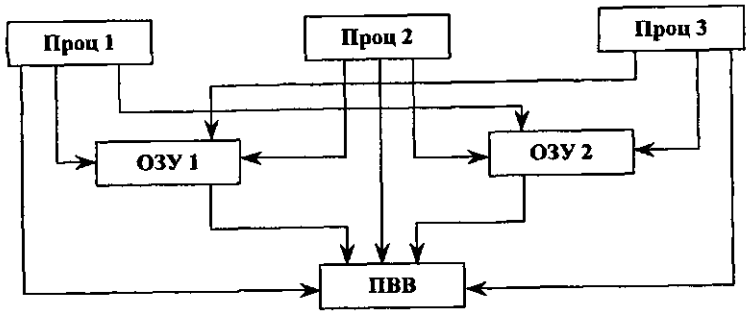
Подключение сателлитных ЭВМ наиболее удобно через АКК.

## ТЕМА 7-2.

### Многопроцессорные вычислительные системы (МПВС).

МПВС - система, которая включает в себя два или более процессора, имеющих общую ОП, общие ПУ и работающих под управлением одной ОС, включающей в себя спецПО. В то же время допускается кроме общих ОП и ПО, иметь и индивидуальные, доступные только одному процессору оперативную память и периферийные устройства.

Упрощённая схема МПВС, содержащей 3 процессора, 2 модуля ОЗУ и подсистему ввода-вывода, изображена ниже.




Даже для такой простой по количественному составу системы схема оказывается достаточно сложной, т.к. в МПВС д.б. обеспечен доступ любого процессора и любого КВВ к любой ячейке памяти; любого процессора к любому КВВ и ПУ. Если же учесть, что процессоров обычно значительно больше, что ОЗУ по соображениям надёжности и удобства наращивания ёмкости выполнено в виде нескольких модулей, а подсистема ввода-вывода включает в себя несколько каналов и большое число ПУ, становится ясным, насколько сложна ***топология*** МПВС.

Отдельная и чрезвычайно большая по сложности задача - ***создание ОС*** для МПВС. Кроме обычных функций, выполняемых ОС при мультипрограммной обработке информации, в МПВС возникают дополнительные задачи:

- распределение ресурсов.и^^йрий-между процессорами;
- синхронизация процессов при решении несколькими процессорами одной задачи;
- планирование заданий с учётом их оптимального распределения;
- решение возникающих конфликтных ситуаций и т.д.

Однако, не смотря на перечисленные трудности аппаратной и программной реализации, МПВС нашли широкое распространение, т.к. обладают рядом достоинств:

- высокая надёжность и готовность за счёт резервирования и возможности реконфигурации;
- высокая производительность за счёт параллельной обработки и^более полной загрузки оборудования;

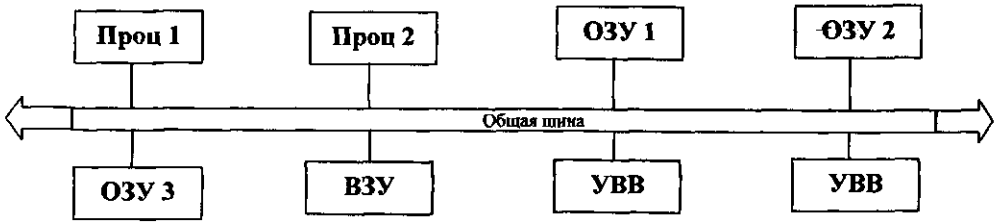
• низкая стоимость вычислений за счёт повышения коэффициента использования оборудования системы.

***Типы структурной организации МПВС.***

Существует три типа структурной организации МПВС:

1. **с общей шиной (ОШ);**
2. **с перекрёстной коммутацией;**
3. **с многовходовыми ОЗУ.**

В системах с ***общей шиной*** (ОШ) проблема связи всех устройств между собой решается очень просто: все они соединяются общей шиной. Интерфейс - односвязный, т.е. обмен информацией в любой момент может идти только между двумя устройствами. Возможны конфликтные ситуации, которые разрешаются при помощи очередей и системы приоритетов. Обычно функции управления и распределения во времени ОШ выполняются одним из процессоров, или специальным устройством - арбитром шины. Наиболее простой вариант системы с ОШ имеет такой вид:




Достоинства:

простота построения;

простота изменения; Ito K^tt.'UI'*-^^"- cu-e-Te-a*-^.

доступность модулей ОЗУ для всех устройств.

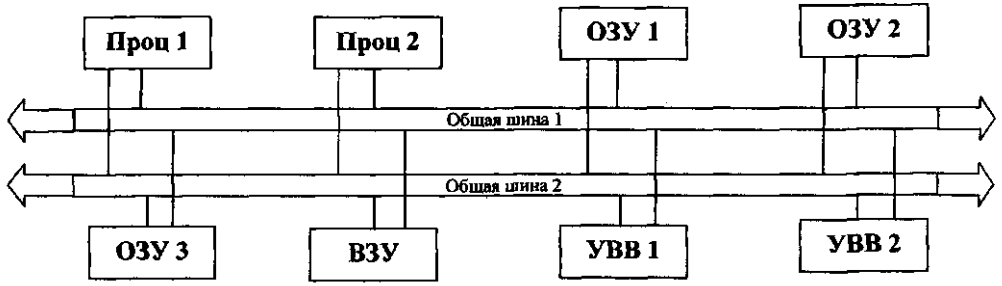
Недостатки:

• невысокое быстродействие из-за малой скорости обмена информацией (как следствие - ограничение числа процессоров до 2-4 штук);

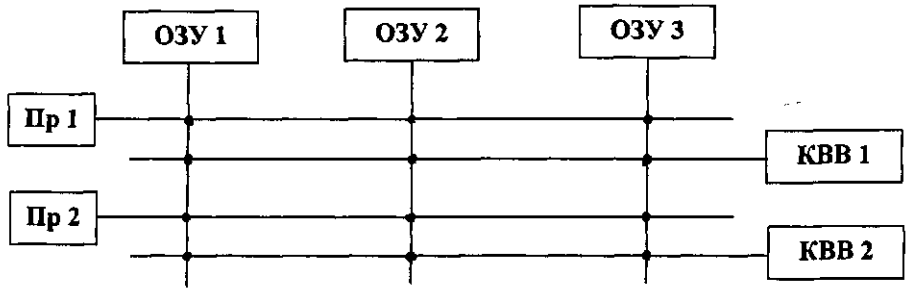
• низкая надёжность (она, фактически, определяется надёжностью шины, точнее, её компонент).

Один из простейших вариантов устранения обоих недостатков - ещё одна активная шина, т.е. шина, находящаяся в нагруженном резерве.

Схема такой системы приведена ниже.




Полностью лишены недостатков, присущих системам с ОШ ***МПВС с перекрёстной коммутацией.***




Основная структурная идея: все связи между устройствами осуществляются только с помощью специального устройства - коммутационной матрицы (КМ). Она позволяет связывать друг с другом любую пару устройств, таких пар м.б. сколько угодно: связи не зависят друг от друга. Конфликты возможны только из-за ресурсов, но не из-за связи!

Производительность повышается по сравнению с МПВС с ОШ за счёт того, что *несколько пар* устройств могут обмениваться информацией *сколь угодно долго.* Поэтому обмен данными идёт с очень высокой скоростью; в системах с ОШ передача данных целыми массивами запрещена, нужно разбивать их.

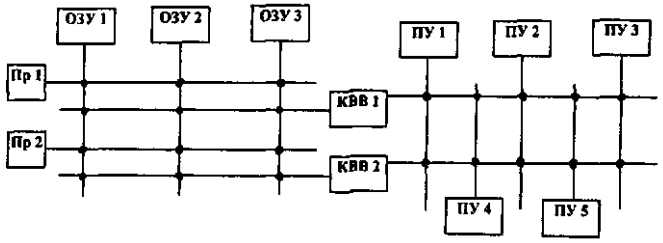
Дополнительные преимущества:

• простота и унифицированность интерфейсов всех устройств;

- выход из строя какого-то элемента КМ не разрушает систему, а лишь отключает какой-то её элемент (устройство);

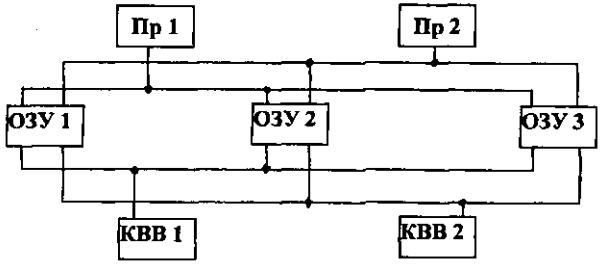
Недостатки МПВС с перекрёстной коммутацией обусловлены тем же, что и её преимущества - коммутационной матрицей. Если в системе заранее не предусмотрена возможность наращивания (т.е. в КМ нет свободных входов), то введение новых элементов в систему потребует *новую КМ*. К тому же, *сложность,* а особенно *стоимость* КМ с ростом количества входов растёт нелинейно из-за большого количества точек коммутации и высокой стоимости схем элементов КМ, которые должны иметь быстродействие выше, чем быстродействие схем обычных элементов.

Для преодоления описанных недостатков КМ стараются «разгрузить», разделяя её на две: КМ центральных устройств (КМЦУ) и КМ периферийных устройств (КМПУ) так, как показано на рисунке:

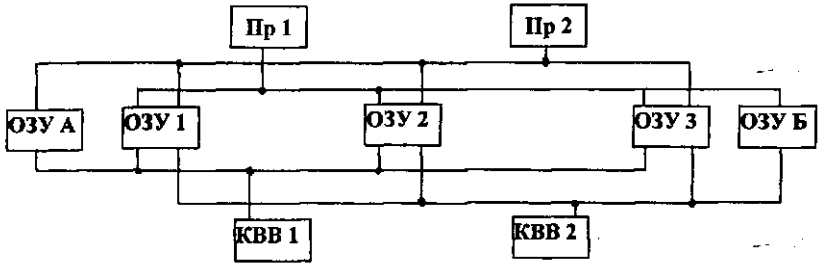



Схемы КМПУ могут иметь значительно меньшее быстродействие (и стоить дешевле), к тому же, две КМ в сумме будут значительно проще.

В МПВС ***с многоходовыми ОЗУ*** всё, что связано с коммутацией устройств, осуществляется в ОЗУ. В таких системах модули ОЗУ имеют число входов, равное числу подключаемых к ним устройств, т.е. для каждого устройства есть свой вход.




В таких системах средства коммутации не сосредоточены, как КМ, а распределены между несколькими устройствами. Т.е. система коммутации при сохранении всех преимуществ значительно упрощается. Для наращивания системы должны быть предусмотрены дополнительные входы в ОЗУ. Такая структура позволяет подключать отдельные выделенные блоки ОЗУ для каждого процессора, что позволяет избежать части конфликтов, возникающих при общей ОП.




*Не путать с системами со связью через «почтовый ящик»!*

Недостатком таких систем является то, что при отказе одного из процессоров данные его индивидуального модуля ОЗУ можно получить только через КВВ, что занимает много времени. Кроме того, многовходовые ОЗУ достаточно дороги и не очень надёжны, а при их отказе теряются недублированные данные.

Рассмотренные три типа структурной организации МПВС часто называются классическими или истинными МПВС.

## ТЕМА 7-3.

### Сравнение ММВС и МПВС.

#### А. По сложности ПО операционной системы.

Хотя и ММВС и МПВС относятся к классу МКМД, но организация процессов обработки данных, в силу разной структуры, в них существенно различна.

ММВС наиболее приспособлены для решения несвязанных между собой задач. Строго говоря, ММВС может достаточно эффективно решать и слабосвязанные задачи, но только такие, при решении которых объём передаваемых данных невелик, да и то только в случае организации связи между машинами через ОЗУ

(прямо-связанные ММВС). Решение же одной задачи параллельно на нескольких машинах проблематично из-за необходимости синхронизации этапов решения, тогда необходима ещё и связь процессор - процессор и наличие соответствующих функций в ОС.

По этим причинам ММВС используются, в основном, для повышения надёжности при решении одной задачи и/или для решения потока независимых или слабосвязанных задач.

На основе сказанного можно оценить сложность ОС ММВС: она сравнительно мало отличается от ОС одиночных ЭВМ, работающих в мультипрограммном режиме. Добавляются только следующие модули, обеспечивающие выполнение функций, связанных с комплексированием:

- модуль обеспечения обмена информацией между машинами;
- модуль взаимного контроля состояния ЭВМ (в случае обеспечения повышенной надёжности);
- модуль проведения регламентных работ (периодический переход основная - резервная и наоборот, с выполнением тестового контроля);
- модуль взаимодействия системы с оператором.

**МПВС** способны решать все типы задач для ММВС + сильносвязанные задачи + обеспечивают эффективное распараллеливание решения одной задачи.

МПВС обладают колоссальными потенциальными возможностями, как с точки зрения роста производительности, так и с точки зрения повышения надёжности и живучести (как об этом я говорил ранее). Но использование всех богатых возможностей МПВС - не простая задача, т.к. ОС, на которые падает основная тяжесть организации вычислительных процессов, оказываются очень сложными. Очень часто разработчики идут на отказ от тех или иных возможностей системы только для того, чтобы ОС для создаваемой системы была более простой.

По сравнению с ОС одиночных ЭВМ, работающими в мультипрограммном режиме, в ОС МПВС добавляются следующие модули:

- модуль, отвечающий за автоматическую реконфигурацию после отказа системы;

и

- модуль организации очереди и распределения ресурсов;
- модуль определения задач (частей одной задачи), которые могут решаться параллельно (программирование на спец языках);
- модуль синхронизации параллельных процессов (обычно при помощи семафоров);
- модуль планирования равномерной загрузки процессоров и ПУ;
- диагностический модуль (и для оперативного, и для тестового контроля).

Существуют три типа организации вычислительного процесса, и, соответственно, три типа функционирования ОС:

1. **«ведущий-ведомый»;**
2. **раздельное выполнение заданий в каждом процессоре;**
3. **симметричная, или однородная,** обработка информации всеми процессорами.

Организация работы системы по принципу ***«ведущий-ведомый»*** является наиболее простой. Один из процессоров (супервизор) управляет работой всех остальных, распределяет задачи и ресурсы, организует передачу информации, отключение неисправных устройств и реконфигурацию системы. Остальные процессоры - исполнители задач.

ОС такой МПВС тоже проста: благодаря *централизации* конфликты из-за ресурсов и др. отсутствуют, другие острые моменты также сняты или нивелированы.

Ведущий процессор м.б. таким же, как и остальные, тогда его замена при отказе проста и система почти не останавливается в работе; если же его делают узкоспециализированным (для высокой производительности), то при отказе выходит из строя вся система.

Главный недостаток - ведущий процессор может «захлебнуться» в потоке малых задач или «перегрузиться» при синхронизации решения одной задачи на нескольких процессорах. По такой схеме строят, в основном, узкоспециализированные МПВС.

Более универсальны МПВС с ***раздельным выполнением заданий в каждом процессоре.*** Все процессоры равноправны, каждый имеет и супервизорные, и исполнительские функции, но процессоры не свободны в выборе задач: каждый имеет *заранее распределённый* набор задач и набор соответствующих ресурсов для их выполнения. Проблемы возникают при отказе какого-то из процессоров: его задачи м.б. перераспределены только после выполнения своих задач другими процессорами, к тому же, необходимо участие оператора. Т.к. *несколько процессоров* должны иметь возможность работы ***с*** *одной и той же программой ОС одновременно,* то наиболее часто в таких системах каждый процессор имеет свою копию достаточно *простой* ОС.

Основной недостаток - трудно обеспечить оптимальную загрузку процессоров, она сильно зависит от конкретного набора задач. Возможны также простои ПУ.

Этот тип организации обработки информации делает МПВС похожей на ММВС, но в нашем случае ресурсы распределяются эффективнее, т.к. количество аппаратуры гораздо меньше. К тому же повышение надёжности путём резервирования значительно дешевле.

В наибольшей степени все преимущества МПВС проявляются при ***симметричной, или однородной,*** обработке информации. Все процессоры максимально самостоятельны, имеют и общесистемные, и супервизорные, и исполнительные функции. В системе нет предварительного распределения заданий и ресурсов между процессорами - каждый из них при освобождении от предыдущей задачи выбирает себе новую из общей очереди или списка. Параллельная обработка одной задачи возможна, если её части включаются в общий список заданий. Реконфигурация д.б. автоматической.

Недостатки - большое количество конфликтов и *самая сложная* ОС. Именно в этом случае разработчики часто идут на компромиссные ограничения возможностей ВС, о которых я говорил выше.

**Б. По надёжности.**



Из изложенного ранее легко видеть, что по этому параметру МПВС значительно лучше, чем ММВС. Это объясняется уровнем резервирования и возможностью реконфигурации.

**В.** По производительности.

Удобнее всего сравнить производительность ММВС и МПВС на двух примерах нагрузки системы. Первый: в ВС поступает большой поток различных независимых задач с небольшим объёмом вычислений. Второй: ВС нагружается малым числом крупных задач с большой вычислительной работой.

При первом варианте ММВС будет мало отличаться от нескольких автономных машин. Общий поток задач тем или иным способом разделяется между всеми ЭВМ и каждая из них работает независимо. Объём передаваемой между машинами информации незначителен (фактически только для сохранения данных на случай отказа), потери времени на обмен малы. Производительность такой ММВС можно оценить как *сумму производительностей всех ЭВМ, входящих в систему.*

$$n^{сист} = uii$$

Если же главная задача ММВС - обеспечение высокой надёжности процесса управления и резервные ЭВМ не решают собственных задач, то производительность такой системы рана производительности основной машины.

В МПВС также существует непрерывно пополняемый общий список задач, из которого каждый процессор выбирает новые задачи. Но, в отличие от ММВС, для всех процессоров общедоступны все ресурсы: ОП, ПУ, ВЗУ, программные средства; более того, все процессоры работаю под управлением единой ОС. Общедоступность всех ресурсов неизбежно приводит к конфликтам из-за них, а это значит, что процессоры неизбежно какую-то часть времени будут простаивать. Если ещё учесть тот факт, что временные затраты на работу ОС в МПВС выше, чем в ММВС (сама система сложнее), то становится очевидным, что суммарные потери производительности из-за простоев и медленной работы сложной ОС будут нелинейно нарастать с увеличением числа процессоров в системе. Экспериментально показано, что при подключении второго процессора

производительность увеличивается на 60-80%, но не вдвое. Добавление третьего даёт рост в 2.1 раза, но не втрое.

Таким образом, в случае нагрузки по первому варианту, преимущество по производительности на стороне **ММВС.**

При нагрузке по второму варианту картина существенно меняется. Крупные задачи обычно могут быть разделены на отдельные части, которые могут решаться параллельно и независимо друг от друга (параллелизм независимых ветвей). Но после раздельного решения эти части должны решаться совместно, т.е. подзадачи нельзя рассматривать как совершенно независимые задачи, наоборот, они — поток *сильносвязанных* задач. Для ММВС это становится не под силу,' т.к. в её ОС отсутствуют элементы для такого согласования, точнее, синхронизации параллельных подзадач. Кроме того, при синхронизации потребовалась бы передача больших объёмов данных (исходные параметры, результаты работы подзадач, данные протоколов обмена и т.п.), но в ММВС нет технических средств для быстрой передачи больших объёмов информации, не вызывающих простоев всех ЭВМ системы. Поэтому при решении больших задач производительность ММВС мало увеличивается с ростом числа ЭВМ.

Для МПВС решение крупных задач не приводит к заметным дополнительным затратам времени, есть и средства синхронизации в составе ОС, и «быстрые» каналы связи. Временные затраты мало отличаются от тех, которые имеют место при решении потока независимых задач.

Таким образом, при решении больших задач производительность МПВС, как правило, существенно превосходит производительность ММВС при равном числе процессоров.

Рассмотренные примеры показывают, что однозначно отдать предпочтение той или иной системе по производительности нельзя. Обычно решение принимается с учётом характера решаемых задач (и требований надёжности).

**Г. По экономической эффективности.**

Экономическая эффективность - интегральная оценка характеристик, рассмотренных выше. Определяется в каждом случае индивидуально.